Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи

Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи

Магістра

на тему: «Аналіз алгоритмів кластеризації для обробки великих даних»

за освітньою програмою Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав: студент групи: «ПЗ2321»

_____ /Владислав ЄРМАКОВ/

Керівник: _____ /доц. Олександр ІВАНОВ/

Нормоконтролер: _____ /Світлана ВОЛКОВА/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань
Студент _____

Дніпро – 2025 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems

Department Computer information technology

Explanatory Note

to Master's Thesis

on the topic: «Analysis of clustering algorithms for big data processing»

according to educational curriculum **12 software engineering**

in the Speciality: **121 software engineering**

Done by the student of the group PZ2322: /Vladyslav Yermakov/

Scientific Supervisor: /Oleksandr IVANOV/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2025

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем

Кафедра: Комп’ютерні інформаційні технології

Рівень вищої освіти: магістр

Освітня програма: Інженерія програмного забезпечення

Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ____КІТ

_____ Вадим ГОРЯЧКІН

_____ грудня 2024 р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Магістр____

студенту Єрмакову Владиславу Віталійовичу____.

1. Тема дипломної роботи: «Аналіз алгоритмів кластеризації для обробки великих даних».

Керівник роботи: Іванов Олександр Петрович

затверджені наказом 1196 ст від 05.12.2022 року

2. Строк подання студентом роботи ___.01.2025 року

3. Вихідні дані до дипломної роботи:_____.

4. Зміст пояснювальної записки (перелік питань до розробки):

4.1. Аналітична частина: Аналіз сучасного стану дослідження за науковими літературними джерелами;

4.2. Основна частина: Дослідження та порівняння алгоритмів кластеризації;

5. Перелік демонстраційного матеріалу:

5.1. презентація;

5.2. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	01.09.23 – 01.10.23	10%
2	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	01.10.23 – 10.10.23	
3	Постановка задачі, технічне завдання	01.09.24 – 15.09.24	30%
4	Розробка інструментальних засобів дослідження	15.09.24 – 01.11.24	
5	Виконання досліджень	01.10.24 – 01.11.24	60%
6	Оформлення пояснювальної записки	01.11.24 – 01.01.25	
7	Розробка демонстраційних матеріалів	10.01.25 – 15.01.25	100%
8	Подання кваліфікаційної роботи до кафедри	18.01.25	
9	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	25.01.25	

Студент: _____ Владислав Єрмаков

Керівник роботи: _____ доц. Олександр ІВАНОВ

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра складається з 171 с., 58 рис., 3 табл., 3 додатків, 19 джерел.

Об'єктом дослідження є аналіз алгоритмів кластеризації для обробки великих даних.

Мета роботи: Дослідити ефективність алгоритмів кластеризації для обробки даних. Порівняти алгоритми K-means, ієрархічної кластеризації та DBSCAN за основними характеристиками. Оцінити їхню продуктивність при роботі з великими наборами даних. Запропонувати рекомендації щодо вибору алгоритму залежно від специфіки даних.

Методика дослідження: Теоретичний аналіз наукових і літературних джерел для визначення основних характеристик алгоритмів. Порівняння результатів за метриками, такими як Calinski-Harabasz, Silhouette Score, стійкість до шумів, та інші.

Перелік ключових слів: кластеризація, K-means, ієрархічна кластеризація, DBSCAN, ефективність алгоритмів, аналіз даних, штучний інтелект.

ЗМІСТ

ВСТУП

В еру інформаційних технологій та великих даних особливої актуальності набуває проблема ефективної обробки, аналізу та кластеризації інформації. Одним із сучасних напрямів вирішення цієї проблеми є використання алгоритмів кластеризації, які дозволяють структурувати великі масиви даних для подальшого аналізу.

Алгоритми кластеризації мають такі унікальні властивості:

– групування даних без попереднього знання про кількість чи типи кластерів;

– можливість роботи з великими наборами даних різної структури;

– адаптація до специфічних характеристик даних, таких як шум чи густина.

У даній роботі будуть розглянуті три основні підходи до кластеризації: K-means, ієрархічна кластеризація та DBSCAN.

Метою дослідження є аналіз алгоритмів кластеризації для обробки великих даних та встановлення їхніх переваг і недоліків у різних сценаріях використання.

Дослідження включає порівняльний аналіз зазначених алгоритмів, проведення тестування на синтетичних та реальних наборах даних, оцінку їх ефективності за різними метриками та визначення оптимальних умов використання кожного алгоритму.

Таким чином, метою цього дослідження є надання комплексного огляду алгоритмів K-means, ієрархічної кластеризації та DBSCAN, а також визначення їхніх переваг і обмежень, що забезпечить підстави для обґрунтованого вибору алгоритму залежно від специфіки завдання та даних.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕННЯ ЗА НАУКОВИМИ ЛІТЕРАТУРНИМИ ДЖЕРЕЛАМИ

1.1 Загальні відомості про кластери

Кластеризація є однією з ключових технік аналізу даних і належить до ненагляданих методів машинного навчання. Її метою є поділ об’єктів на групи, які називаються кластерами, таким чином, щоб об’єкти в межах одного кластера мали більшу схожість між собою, ніж із об’єктами інших кластерів. Кластери дозволяють виявляти приховані закономірності та структури в даних, що робить цей метод особливо важливим у багатьох наукових і практичних галузях.

Відмінною рисою кластеризації є її універсальність. Вона застосовується до найрізноманітніших типів даних: числових, категоріальних, текстових, зображень тощо. Наприклад, у біоінформатиці кластеризація використовується для групування генів за їхньою експресією, у маркетингу – для сегментації клієнтів, у комп’ютерному зорі – для обробки зображень, а в соціології – для аналізу поведінки користувачів[1].

Одним із важливих аспектів кластеризації є вибір метрики, яка використовується для оцінки схожості між об’єктами. Найпоширенішими метриками є:

Евклідова відстань – вимірює найкоротшу лінійну відстань між двома точками. Вона широко використовується завдяки своїй простоті, проте може бути чутливою до масштабу змінних, що вимагає попередньої нормалізації даних[2].

Манхеттенська відстань – підраховує суму модулів різниць координат точок. Ця метрика часто застосовується для задач, де важливі переміщення по координатних осях, наприклад, у транспортних моделях[2].

Косинусова схожість – використовується для вимірювання кута між двома векторами в багатовимірному просторі. Це особливо актуально для текстових даних або задач, де важливі співвідношення змінних, а не їх абсолютні значення[2].

Кластеризація ділиться на кілька типів залежно від підходу до обробки даних:

Розділова кластеризація (Partitioning): Дані розділяються на заздалегідь задану кількість кластерів. Найпоширенішим прикладом є алгоритм K-means, який

базується на повторюваному перерозподілі точок до найближчого центра кластера.

Ієрархічна кластеризація (Hierarchical): Будується дерево кластерів (дендрограма), яке показує, як об’єкти об’єднуються або розділяються на різних рівнях. Цей метод дозволяє не задавати кількість кластерів заздалегідь[2].

Кластеризація на основі густини (Density-Based): Групує точки за їхньою щільністю в просторі, що дозволяє виділяти кластери довільної форми й ігнорувати шуми. Представником цього підходу є алгоритм DBSCAN[2].

Особливу роль у кластеризації відіграє її оцінка, яка виконується за допомогою спеціальних метрик. Наприклад, індекс силуета визначає, наскільки об’єкти в межах одного кластера близькі до центру кластера й наскільки вони віддалені від інших кластерів. Інші показники, як-от індекс Девіса-Болдіна або оцінка Калінського-Харабаша, вимірюють “компактність” кластерів і відстань між ними.

Кластеризація відіграє важливу роль у дослідженні великих даних, особливо в еру розвитку штучного інтелекту. Завдяки цьому методу дослідники та практики можуть отримати цінні інсайти з великих, складних і різнорідних наборів даних. Ця техніка має великий потенціал для подальшого розвитку, особливо у поєднанні з іншими методами машинного навчання.

1.2 K-means кластеризація

K-means є одним із найвідоміших алгоритмів кластеризації, який використовується для поділу даних на k-кластерів, де k – це заздалегідь визначена кількість. Його популярність зумовлена простотою реалізації, швидкістю виконання та широким спектром застосувань у різних галузях, таких як маркетинг, аналіз поведінки користувачів, біоінформатика, комп’ютерний зір тощо[3].

Алгоритм K-means належить до методів розділової кластеризації, які передбачають ітеративне оновлення центрів кластерів і перерозподіл об’єктів між кластерами на основі схожості. Він функціонує за таким принципом:

– ініціалізація центрів кластерів: На початковому етапі обираються випадкові точки (або інші стратегії, такі як K-means++), які слугують початковими центрами кластерів;

– розподіл об'єктів: Кожна точка даних призначається до найближчого центру кластера, визначеного за обраною метрикою (наприклад, Евклідовою або Манхеттенською);

– оновлення центрів кластерів: Центри кластерів обчислюються заново як середнє значення точок, що належать до кластера;

– повторення кроків: Етапи 2 і 3 повторюються доти, доки центри кластерів не перестануть змінюватися або не буде досягнуто максимальну кількість ітерацій.

Метрики відстані в K-means

Алгоритм K-means може використовувати різні метрики для оцінки відстані між об’єктами:

Евклідова відстань: Є стандартною метрикою для K-means, оскільки дозволяє оцінювати прямолінійну відстань між точками в багатовимірному просторі. Вона ефективна для даних, де важливі абсолютні значення координат. Проте цей підхід є чутливим до масштабування, тому часто використовується нормалізація змінних;

Манхеттенська відстань: Використовується в задачах, де важливий шлях по координатах, а не прямолінійна відстань. Вона менш чутлива до викидів, ніж Евклідова, що робить її корисною для деяких реальних застосувань.

Переваги та недоліки K-means

Переваги:

– простота реалізації та розуміння;

– висока швидкість роботи, що робить алгоритм ефективним для великих наборів даних;

– Легко масштабується для великих обсягів інформації.

Недоліки:

– алгоритм є чутливим до початкового розташування центрів кластерів, що може призвести до локальних мінімумів;

– потребує попереднього вибору кількості кластерів k, що може бути складним завданням у багатьох випадках;

– погано працює з кластерами, які мають нерівномірні розміри або довільні форми, оскільки передбачає, що кластери є сферичними;

– чутливість до шуму й викидів, які можуть значно зміщувати центри кластерів.

Методи вибору оптимальної кількості кластерів

Одним із ключових викликів у застосуванні K-means є визначення оптимальної кількості кластерів k. Для цього використовуються такі методи:

– метод “лікоть” (Elbow Method): Будується графік залежності середньої відстані точок до центрів кластерів від кількості кластерів. Оптимальна кількість кластерів визначається в точці, де графік починає згинатися, тобто приріст точності стає незначним;

– силуетний аналіз (Silhouette Analysis): Оцінюється, наскільки об’єкти всередині кластерів є схожими між собою, і наскільки вони віддалені від об’єктів інших кластерів. Силуетний коефіцієнт варіюється від -1 до 1, де вищі значення вказують на кращу якість кластеризації.

Розширення алгоритму K-means

Для подолання обмежень класичного K-means було запропоновано кілька модифікацій:

– K-means++: Покращує вибір початкових центрів кластерів, щоб мінімізувати ймовірність потрапляння в локальні мінімуми. Це забезпечує кращі результати кластеризації з першої ітерації;

– Mini-Batch K-means: Замість обробки всього набору даних, використовує невеликі підмножини (батчі), що значно знижує обчислювальну складність для великих наборів даних;

– Weighted K-means: Враховує ваги точок даних, що може бути корисним у задачах із неоднорідними даними.

Сфери застосування K-means

Завдяки своїй універсальності, K-means використовується в багатьох галузях:

– сегментація клієнтів: У маркетингу алгоритм використовується для групування клієнтів за поведінкою, демографічними характеристиками або уподобаннями;

– аналіз зображень: Застосовується для сегментації зображень або зменшення кольорових палітр;

– біоінформатика: Використовується для кластеризації генів або білків на основі їхньої експресії;

– обробка тексту: Визначення тем у текстових даних або групування документів за їх змістом.

Таким чином, K-means є потужним і водночас простим у використанні інструментом для задач кластеризації, який залишається одним із найпоширеніших методів у сучасному аналізі даних. Його застосування ефективне у випадках, коли кластери мають приблизно однаковий розмір і форму, а також для задач, де важлива швидкість і масштабованість алгоритму.

1.3 Ієрархічна кластеризація

Ієрархічна кластеризація є методом кластерного аналізу, який будує ієрархічну структуру взаємозв’язків між даними. Результат такого аналізу зазвичай представлений у вигляді дендрограми – деревоподібної структури, що демонструє, як об'єкти об'єднуються у кластери на різних рівнях. Основною перевагою цього методу є його здатність працювати без попереднього визначення кількості кластерів, що робить його дуже гнучким і придатним для широкого спектру задач[3].

Підходи до ієрархічної кластеризації

Ієрархічна кластеризація реалізується двома основними підходами:

Агломеративна кластеризація:

– починається з кожного об'єкта як окремого кластера;

– на кожному кроці два найближчі кластери об’єднуються в один;

– процес триває доти, доки всі об'єкти не об'єднаються в один великий кластер;

– цей підхід є найпоширенішим і часто використовується в практиці.

Дивізійна кластеризація:

– починається з усіх об'єктів як одного великого кластера;

– на кожному кроці кластер ділиться на менші частини;

– процес триває доти, доки кожен об'єкт не стане окремим кластером;

– цей підхід менш популярний через вищу обчислювальну складність.

Метрики відстаней

Ключовим елементом ієрархічної кластеризації є вибір метрики для визначення відстані між об'єктами або кластерами. Найпоширенішими метриками є:

– евклідова відстань: Вимірює прямолінійну відстань між двома точками в багатовимірному просторі. Вона є найпоширенішою для ієрархічної кластеризації;

– манхеттенська відстань: Обчислює суму модулів різниць координат між точками. Підходить для задач, де важливий шлях між точками, а не пряма відстань;

– косинусова схожість: Використовується для текстових даних або задач, де важлива спрямованість векторів, а не їх абсолютні значення.

Методи зв'язності (Linkage Methods):

Окрім метрик відстані, важливим є метод обчислення зв'язності між кластерами. Основні підходи включають:

Одинарне з’єднання (Single Linkage):

– визначає відстань між кластерами як мінімальну відстань між будь-якими двома точками в різних кластерах;

– схильне до “ланцюгового ефекту”, коли кластери стають витягнутими та розрідженими.

Повне з’єднання (Complete Linkage):

– визначає відстань як максимальну відстань між будь-якими двома точками в різних кластерах;

– створює компактні кластери, але може ігнорувати структури даних.

– середнє з’єднання (Average Linkage):

– обчислює середню відстань між усіма парами точок із різних кластерів;

– забезпечує компроміс між “ланцюговим ефектом” і надмірною компактністю.

– центроїдне з’єднання (Centroid Linkage):

– визначає відстань між центрами мас двох кластерів;

– чутливе до зміни розмірів і форми кластерів.

Переваги та недоліки ієрархічної кластеризації

Переваги:

– не потребує попереднього визначення кількості кластерів;

– будує дендрограму, яка дає змогу аналізувати структуру даних на різних рівнях;

– може працювати з даними, що мають складні або невідомі взаємозв'язки.

Недоліки:

– висока обчислювальна складність: час виконання зростає квадратично із збільшенням кількості точок;

– чутливість до шуму: викиди можуть впливати на об’єднання кластерів;

– відсутність можливості “перерозподілу” точок: раз віднесені точки залишаються у своїх кластерах, навіть якщо це не оптимально.

Сфери застосування

Ієрархічна кластеризація має широкий спектр застосувань, особливо там, де важливо розуміти багаторівневі взаємозв’язки між об’єктами:

– біоінформатика: Аналіз генів та білків для виявлення подібностей у їх поведінці;

– аналіз тексту: Групування документів або слів за змістовними ознаками;

– соціальні мережі: Виявлення груп користувачів із подібними інтересами;

– обробка зображень: Сегментація об’єктів на основі їх візуальних характеристик.

Оцінка якості кластеризації

Для оцінки якості ієрархічної кластеризації можуть використовуватися такі метрики, як індекс силуета або індекс Девіса-Болдіна. Ці метрики дозволяють

порівнювати якість кластеризації при різних параметрах і вибирати оптимальний підхід.

Ієрархічна кластеризація є потужним інструментом для аналізу даних із багаторівневою структурою. Вона особливо корисна, коли важливо не лише знайти кластери, але й зрозуміти, як вони пов'язані між собою. Однак висока обчислювальна складність і чутливість до шуму вимагають обережності при роботі з великими наборами даних.

1.4 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) – це алгоритм кластеризації на основі густини, який виділяє кластери в даних шляхом оцінки їхньої локальної густини. Основна ідея DBSCAN полягає в тому, що кластери формуються як області із щільно розташованими точками, тоді як точки, які знаходяться далеко від інших, класифікуються як шум. Цей підхід дозволяє ефективно працювати з кластерами довільної форми, а також ігнорувати шуми й викиди[3].

Основні концепції DBSCAN:

Алгоритм базується на таких ключових поняттях:

- ядрова точка (Core Point): Точка вважається ядровою, якщо навколо неї в радіусі eps знаходиться щонайменше minPts сусідів;
- прикордонна точка (Border Point): Точка, яка знаходиться в околі ядрової точки, але сама не є ядровою;
- шумова точка (Noise Point): Точка, яка не є ні ядровою, ні прикордонною.

DBSCAN групує ядрові точки в кластери та додає до них прикордонні точки, формуючи пов’язані області. Шуми залишаються окремими та не належать до жодного кластера.

Етапи роботи алгоритму DBSCAN:

- вибір початкової точки з набору даних;
- перевірка, чи є точка ядровою (на основі параметрів eps та minPts);

– якщо точка є ядровою, формується новий кластер, і до нього додаються всі її сусіди;

– для кожного нового сусіда перевіряється, чи є він ядровою точкою. Якщо так, до кластера додаються його сусіди;

– процес повторюється, доки всі точки в кластері не будуть оброблені;

– шуми залишаються без кластеризації.

Параметри DBSCAN:

– eps: Радіус околу (ε), який визначає, наскільки близько мають бути точки, щоб їх можна було вважати частиною одного кластера. Вибір eps сильно впливає на якість кластеризації;

– minPts: Мінімальна кількість точок в околі, необхідна для того, щоб точка вважалася ядровою.

Ці параметри необхідно вибирати на основі властивостей даних. Для цього можна використовувати графік “відстань-сусіди”, який допомагає знайти оптимальне значення eps.

Оцінка якості кластеризації

Для оцінки роботи DBSCAN використовуються такі метрики:

– індекс Девіса-Болдіна (Davies-Bouldin Index):

  – порівнює “компактність” кластерів із відстанню між ними;

  – менші значення індексу вказують на кращу кластеризацію.

– оцінка Калінського-Харабаша (Calinski-Harabasz Score):

  – вимірює співвідношення дисперсії між кластерами до дисперсії всередині кластерів;

  – вищі значення вказують на якіснішу кластеризацію.

– силуетний аналіз (Silhouette Score):

  – визначає наскільки кожна точка схожа на точки свого кластера та віддалена від інших кластерів;

  – значення коливається від -1 до 1; вищі значення вказують на кращу кластеризацію.

Переваги та недоліки DBSCAN

Переваги:

- здатність виділяти кластери довільної форми;
- стійкість до шуму та викидів;
- не потребує заздалегідь визначати кількість кластерів.

Недоліки:

- чутливість до вибору параметрів eps і minPts;
- втрата ефективності при високій розмірності даних, оскільки оцінка відстаней стає обчислювально складною;
- може не впоратися з даними, які мають варіативну густину кластерів.

Сфери застосування DBSCAN

DBSCAN використовується в багатьох галузях завдяки своїй здатності працювати з шумами та кластерами довільної форми:

- географічний аналіз: Виявлення густонаселених областей на карті;
- виявлення аномалій: Ідентифікація аномальних точок у даних, наприклад, транзакцій або поведінки користувачів;
- аналіз зображень: Групування об’єктів на основі їхніх характеристик;
- аналіз мереж: Виявлення підмереж або кластерів у соціальних чи телекомунікаційних мережах.

Порівняння з іншими алгоритмами

На відміну від K-means та ієрархічної кластеризації, DBSCAN не обмежений сферичною формою кластерів і не вимагає визначення їхньої кількості. Це робить його особливо корисним у задачах, де кластери мають складну геометрію або де присутній значний шум. Однак для даних із неоднорідною щільністю DBSCAN може виділяти лише кластери, які відповідають обраним параметрам eps і minPts.

DBSCAN є потужним інструментом для кластеризації великих наборів даних, особливо там, де важлива робота з шумами та кластерами довільної форми. Незважаючи на обмеження, алгоритм демонструє високу ефективність у багатьох реальних сценаріях, особливо в задачах із густими або нерегулярними даними. Його

гнучкість і здатність адаптуватися до різних умов роблять DBSCAN одним із найважливіших методів сучасної кластеризації.

1.5 Обмеження та порівняння K-means, ієрархічної кластеризації та DBSCAN

У задачах кластеризації вибір алгоритму має вирішальне значення, оскільки різні підходи мають свої переваги, недоліки та обмеження. У цьому розділі буде проведено порівняння трьох основних алгоритмів кластеризації: K-means, ієрархічної кластеризації та DBSCAN, з урахуванням їхніх властивостей, сфер застосування та обмежень[3].

Обмеження K-means

– чутливість до початкових умов: K-means залежить від вибору початкових центрів кластерів, що може призвести до локальних мінімумів;

– форма кластерів: Алгоритм припускає, що кластери мають сферичну форму, що є неефективним для даних із нерегулярною структурою;

– шум і викиди: K-means чутливий до шуму та викидів, оскільки вони можуть значно зміщувати центри кластерів;

– кількість кластерів: Необхідно заздалегідь визначити кількість кластерів k, що не завжди очевидно.

Обмеження ієрархічної кластеризації

– обчислювальна складність: Час виконання алгоритму значно зростає з кількістю точок, що робить його непридатним для великих наборів даних;

– чутливість до шуму: Шум і викиди можуть впливати на структуру дендрограми, спотворюючи результати кластеризації;

– незмінність об’єднань: Після того як об’єкти об’єднані в кластери, алгоритм не дозволяє їх перерозподілити;

– відсутність автоматичного визначення кількості кластерів: Кількість кластерів визначається після аналізу дендрограми, що є суб’єктивним процесом.

Обмеження DBSCAN

– чутливість до параметрів: Параметри eps та minPts мають вирішальний вплив на результати кластеризації. Неправильний вибір може призвести до занадто великої кількості кластерів або їх відсутності;

– дані з нерівномірною густиною: DBSCAN має складнощі з кластеризацією наборів даних, де кластери мають різну густину;

– висока розмірність даних: Алгоритм може втратити ефективність при високій кількості вимірів через обчислювальну складність пошуку сусідів.

Таблиця 1.1 – Порівняння алгоритмів

Характеристика	K-means	Ієрархічна кластеризація	DBSCAN
Тип кластерів	Сферичні	Довільна структура	Довільна структура
Підхід	Розділова	Ієрархічна	На основі густини
Обчислювальна складність	O(n×k×t)	O(n^2)	O(n×log(n))
Шум і викиди	Чутливий	Чутливий	Стійкий до шуму
Попереднє визначення *k*	Обов'язкове	Необов'язкове	Необов'язкове
Придатність до великих даних	Висока	Низька	Середня
Можливість обробки густини	Ні	Обмежена	Так
Гнучкість до форми кластерів	Низька	Середня	Висока

Сфери застосування[4]

– K-means:

– підходить для великих наборів даних, де кластери мають приблизно рівномірний розмір і сферичну форму;

– використовується для сегментації клієнтів, аналізу зображень та задач, що вимагають швидких обчислень.

– Ієрархічна кластеризація:

– ефективна для задач із невеликою кількістю точок, де потрібно аналізувати багаторівневі взаємозв’язки між об’єктами;

– використовується в біоінформатиці, аналізі тексту та соціальних мережах.

– DBSCAN:

– найкраще підходить для задач, де кластери мають довільну форму або дані містять шуми;

– використовується в географічному аналізі, виявленні аномалій та кластеризації текстів.

Рекомендації щодо вибору алгоритму:

– якщо дані великі, кластери мають сферичну форму, і важлива швидкість виконання – слід використовувати K-means;

– якщо дані невеликі й важливо зрозуміти багаторівневу структуру кластерів – найкращим вибором буде ієрархічна кластеризація;

– якщо дані містять шуми, а кластери мають складну форму або різну густину – доцільно обрати DBSCAN.

Таким чином, вибір алгоритму кластеризації залежить від характеристик даних, задачі та вимог до точності й продуктивності. Кожен із розглянутих алгоритмів має свої переваги та обмеження, і правильний вибір може значно покращити результати аналізу.

Висновки до розділу 1

У першому розділі було проаналізовано три основні алгоритми кластеризації: K-means, ієрархічну кластеризацію та DBSCAN. Розглянуто їхні переваги, недоліки, обмеження, а також специфічні сфери застосування залежно від характеристик даних. Алгоритм K-means є швидким і ефективним для сферичних кластерів, але чутливий до шуму та початкових умов. Ієрархічна кластеризація дозволяє аналізувати

багаторівневі зв’язки між об’єктами, проте має високу обчислювальну складність. DBSCAN демонструє переваги в роботі з шумами та кластерами довільної форми, але вимагає ретельного підбору параметрів.

РОЗДІЛ 2 ОБГРУНТУВАННЯ МЕТОДІВ ДОСЛІДЖЕННЯ

Для дослідження алгоритмів кластеризації буде створено ноутбук у середовищі JupyterLab.

2.1 Основні функціональні вимоги

- зчитування та запис даних: інструмент повинен забезпечувати завантаження даних із файлів CSV, їх попередню обробку та збереження результатів у зручному форматі (наприклад, CSV або Excel);
- проведення кластеризації: дослідження повинно включати роботу з трьома алгоритмами кластеризації (K-means, ієрархічна кластеризація, DBSCAN) з можливістю налаштування параметрів для кожного алгоритму;
- оцінка кластеризації: система повинна генерувати метрики якості кластеризації (наприклад, Silhouette Score, Calinski-Harabasz Index);
- візуалізація результатів: результати кластеризації мають бути представлені у вигляді графіків (розподіл кластерів, дендрограми) для забезпечення зручного аналізу.

Ці функціональні вимоги забезпечать проведення комплексного дослідження алгоритмів кластеризації, включаючи їхню практичну реалізацію та аналіз результатів.

2.3 Вхідні дані

Вхідними даними для Jupyter Notebook є два файли у форматі CSV, які представляють різні набори даних.

2.3 Вихідні дані

Вихідними даними є текстові результати та графіки, що генеруються безпосередньо у середовищі JupyterLab.

Висновки до розділу 2

Другий розділ містить основні функціональні вимоги до інструменту, а також опис вхідних і вихідних даних для дослідження алгоритмів кластеризації.

Основні вимоги до інструменту включають зчитування та обробку даних із CSV-файлів, проведення кластеризації трьома алгоритмами, обчислення метрик якості та візуалізацію результатів у вигляді графіків і текстових звітів.

Вхідними даними є два набори даних: перший — числовий для K-means і DBSCAN, другий — із категоріальними елементами для ієрархічної кластеризації.

Вихідними даними є текстові звіти про якість кластеризації та графіки, що демонструють структуру кластерів.

Розроблений інструмент у JupyterLab забезпечить ефективне проведення дослідження, аналіз і візуалізацію результатів кластеризації.

РОЗДІЛ 3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ АНАЛІЗУ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ

Метою даного дослідження є розробка інструментальних засобів для аналізу алгоритмів кластеризації з використанням сучасних технологій програмування. Проектування інструментальних засобів виконується за допомогою Python та бібліотек Pandas, NumPy, Scikit-learn, Matplotlib, що забезпечують гнучкість і ефективність аналізу великих обсягів даних.

3.1 Вибір мови програмування та середовища розробки

Для розробки інструментального засобу було обрано мову програмування Python через її універсальність, популярність у сфері аналізу даних і багатий вибір бібліотек. Python є динамічно типізованою мовою з простим синтаксисом, що дозволяє швидко реалізувати навіть складні алгоритми кластеризації.

Середовище JupyterLab обрано для створення інтерактивного інструменту завдяки його зручності візуалізації результатів і можливості швидкого редагування та виконання коду. Jupyter Notebook є стандартом у сфері Data Science, що дозволяє інтегрувати текстову документацію, графіки та програмний код в одному робочому просторі.

Таким чином, Python у поєднанні з JupyterLab є оптимальним вибором для аналізу алгоритмів кластеризації.

3.2 Формалізація задачі

Для досягнення поставленої мети створюється інструментальний засіб, який забезпечує виконання наступних задач:

– завантаження даних із файлів CSV у зручному форматі для подальшого аналізу;

– попередня обробка даних, включаючи видалення пропущених значень, нормалізацію та фільтрацію;

– реалізація трьох основних алгоритмів кластеризації: K-means, ієрархічна кластеризація, DBSCAN;

– обчислення метрик оцінки кластеризації (Silhouette Score, Calinski-Harabasz Index);

– візуалізація результатів кластеризації у вигляді графіків і дендрограм;

– гнучке налаштування параметрів алгоритмів (наприклад, кількість кластерів, радіус для DBSCAN);

– генерація текстових звітів для аналізу результатів і порівняння алгоритмів.

Зазначені задачі забезпечать гнучкість і функціональність засобу для проведення аналізу алгоритмів кластеризації.

3.3 Особливості Python для використання у розробці

Python має низку унікальних переваг, які роблять його ідеальним вибором для реалізації дослідження:

– Pandas: Забезпечує ефективну роботу з таблицями та великими наборами даних. Використовується для імпорту CSV-файлів, очищення даних, їхньої обробки та групування.

– NumPy: Надає інструменти для швидкої обробки багатовимірних масивів і виконання складних математичних операцій. Його використання суттєво прискорює обчислення.

– Scikit-learn: Є стандартом у сфері машинного навчання, надає реалізацію алгоритмів K-means, ієрархічної кластеризації та DBSCAN. Бібліотека також забезпечує обчислення метрик кластеризації.

– Matplotlib: Використовується для візуалізації результатів у вигляді графіків, дендрограм і розподілів кластерів.

– PyPI (Python Package Index): Забезпечує доступ до тисяч бібліотек, що спрощує додавання нових функціональних можливостей у проєкт.

Основні переваги Python:

– простий синтаксис, який дозволяє швидко реалізувати навіть складні алгоритми.

– широкий вибір бібліотек для аналізу та візуалізації.

– кросплатформенність: Python однаково добре працює на Windows, macOS і Linux.

Інструментальний засіб, розроблений на Python, дозволяє забезпечити швидкий аналіз, візуалізацію даних та інтеграцію із сучасними технологіями для аналізу великих обсягів даних.

Висновки до розділу 3

У третьому розділі обґрунтовано вибір мови програмування Python та середовища розробки JupyterLab для створення інструментального засобу. Було визначено основні задачі, які вирішує програмний інструмент, включаючи завантаження та обробку даних, реалізацію алгоритмів кластеризації та оцінку їхньої якості. Python у поєднанні з бібліотеками Pandas, NumPy, Scikit-learn і Matplotlib забезпечує гнучкість, швидкість та ефективність у реалізації дослідження. Обраний підхід дозволяє створити універсальний інструмент для аналізу алгоритмів кластеризації, який відповідає сучасним вимогам до аналізу великих даних.

РОЗДІЛ 4 ПОРІВНЯННЯ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ

4.1 Налаштування даних (завантаження та попередня обробка)

Набір даних з відгуків про подорожі: Оскільки для кластеризації даних будуть використовуватися алгоритми на основі відстані, всі стовпці ознак були нормалізовані (в діапазоні 0-1) за допомогою sklearn MinMaxScaler[5], щоб уникнути домінування атрибутів з великими значеннями при обчисленні відстані. Хоча всі значення ознак коливалися в межах 0-4, було виявилено, що нормалізація даних допомогла покращити результати кластеризації.

Набір даних ICMLA: Кластеризація вимагає вимірювання відстані між вибірками. Тому нам довелося попередньо обробити текстові дані в кожному стовпчику ознак. Спочатку було використано бібліотеку Natural Language Tool Kit (nltk), щоб виконати токенізацію, розбивши текст на речення, а потім на слова. Потім були очищені дані, видаливши всі токени, що містять нелітерні символи, за допомогою пакету re (регулярні вирази), і використали метод “снігової кулі” для отримання кореневих форм. Потім було переведено їх у числове представлення, побудувавши впорядкований вектор частоти терміна – зворотної частоти документа (TF-IDF) для кожного унікального слова в корпусі.

Отримані числові дані містили 105 стовпчиків (ознак), і така велика розмірність даних могла спричинити проблеми зі зберіганням, часом обробки та візуалізацією. Отже, було використано sklearn PCA[6] (рис. 4.1) для аналізу того, скільки ознак є важливими, після чого кількість ознак було відповідно зменшено (наприклад, для стовпчика “combine” 31 ознака пояснювала щонайменше 90% різниці в цьому стовпчику, тож кількість ознак було зменшено до 31).

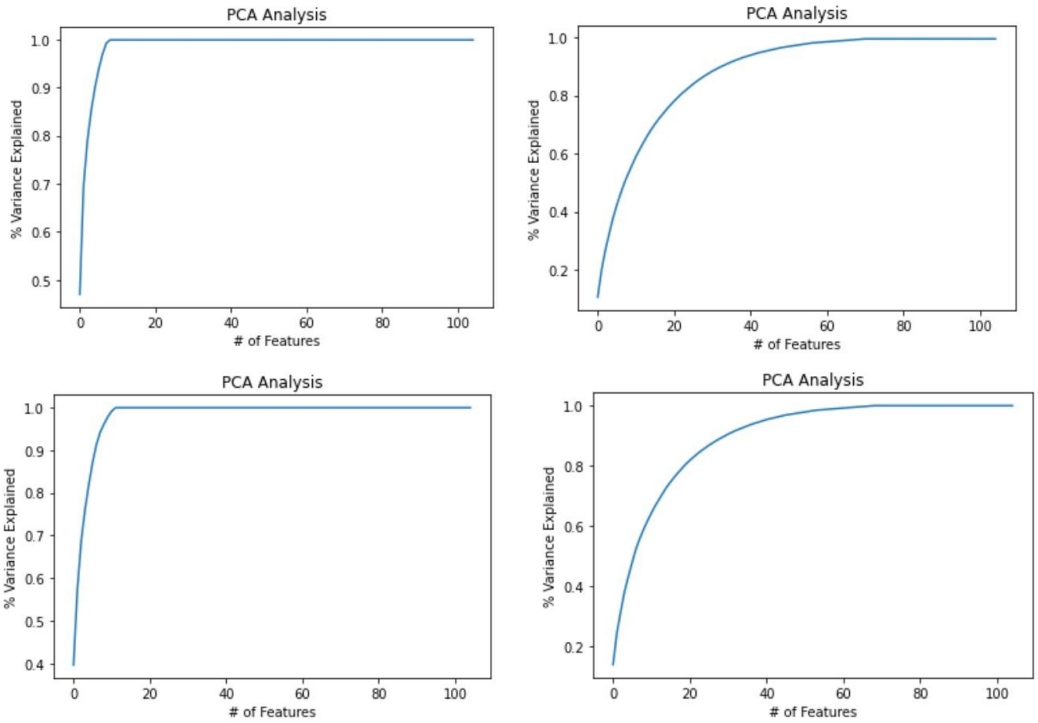


Рисунок 4.1 – PCA-аналіз заголовка (верхній лівий), анотації (верхній правий), ключових слів (нижній лівий) та об'єднаного (нижній правий) стовпчиків

На додаток до колонок назви, ключових слів та анотацій, було створено колонку “combine”, яка включає тексти з цих трьох колонок. Це дозволило спробувати різні стовпці ознак, щоб переконатися, що для нашого алгоритму кластеризації був обраний найкращий. Нарешті, золотий кластер “session” було перетворено в числа (0-23) за допомогою sklearn LabelEncoder[7] для оцінки моделі.

4.2 Кластеризація K-середніх

4.2.1 Набір даних: Відгуки про подорожі

Використовуючи sklearn KMeans[8] (Евклідова) та ручний алгоритм k-середніх (Манхеттен), п'ятдесят вісім моделей було навчено на наборі даних 1 з використанням k=1-29 та двох мір відстані. Евклідова відстань[9], яка вимірює відстань по прямій між двома точками, є загальною метрикою, що використовується для k-середніх. Манхеттенська відстань[9], яка вимірює абсолютну відстань або відстань у кварталах між будь-якими двома точками в місті, була включена для порівняння.

Оскільки для цього набору даних не було надано золотих кластерів, отримані моделі оцінювалися лише за внутрішніми показниками. Для визначення відповідної

кількості кластерів було використано метод ліктя[10], який спостерігає за граничним зменшенням суми внутрішньокластерних дисперсій. З іншого боку, для оцінки якості отриманих кластерів, тобто того, наскільки компактними і добре розділеними вони були, використовувалися показники силуету, Девіса-Булдіна[11] і Калінкі-Харабаша[12].

З графіка інерції (рис. 4.2) видно, що інерція дуже швидко падала при збільшенні k до 7 (Евклідова система) та 8 (Манхеттен), але після цих точок почала вирівнюватися (зі зменшенням інерції менше ніж на 5 на кожен додатковий кластер).

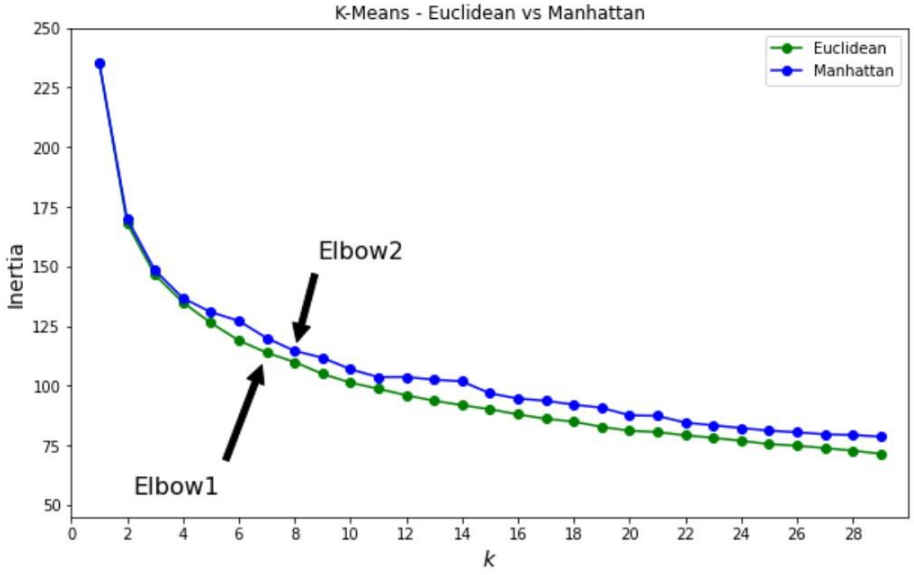


Рисунок 4.2. – Графік інерції: K-середнє евклідове (зелене) проти манхеттенського (синє)

	inertia	difference
0	235.629278	NaN
1	168.075115	67.554163
2	146.428224	21.646890
3	134.887950	11.540275
4	126.321167	8.566783
5	118.973156	7.348011
6	113.794057	5.179099
7	109.906226	3.887831
8	104.918161	4.988065
9	101.262739	3.655422
10	98.670106	2.592633
11	95.967277	2.702829
12	93.653296	2.313981
13	91.815698	1.837598

	inertia	difference
0	235.629278	NaN
1	170.059027	65.570251
2	148.336010	21.723017
3	136.686933	11.649077
4	130.828513	5.858420
5	127.150791	3.677722
6	119.957103	7.193688
7	114.576860	5.380243
8	111.696253	2.880607
9	106.892404	4.803849
10	103.548975	3.343429
11	103.614208	-0.065233
12	102.493515	1.120693
13	101.785898	0.707618

Рисунок 4.3 – Різниця в інерції: K-середнє евклідове (ліворуч) проти манхеттенського (праворуч)

Це свідчить про те, що k=7 для Евклідової системи та k=8 для Мангеттена є вдалим вибором. Загалом, евклідові моделі дають краще розділені кластери, ніж манхеттенські (на що вказують нижчі оцінки Девіса-Болдіна на рисунку (4.4).

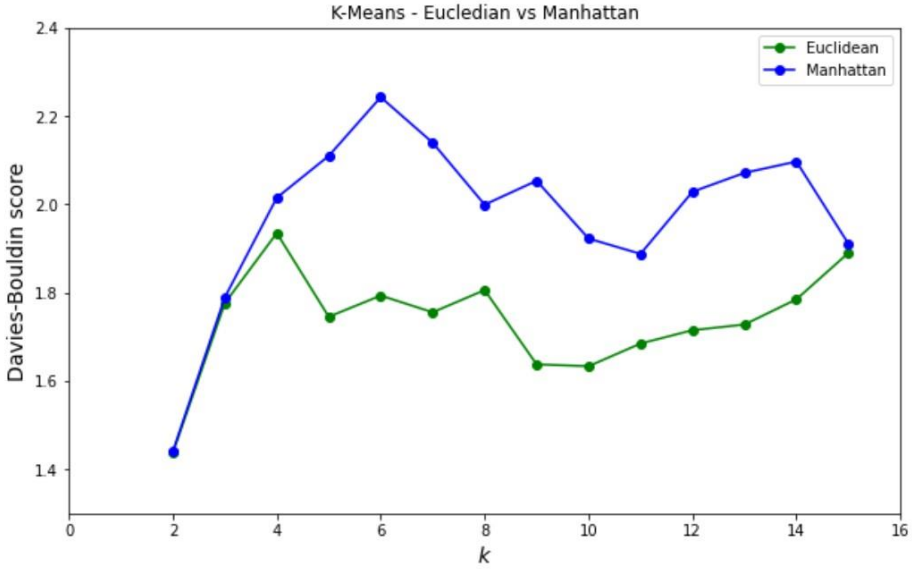


Рисунок 4.4 – Оцінка Девіса-Болдіна: K-середнє евклідове (зелене) проти манхеттенського (синє)

Ігноруючи k=2-3, евклідові кластери дали найнижчий бал при k=9 (1,64), а манхеттенські - при k=11 (1,89). Аналогічно, оцінки Калінкі-Харабаша показали, що евклідові кластери утворюють більш щільні та краще відокремлені кластери, ніж манхеттенські, хоча оцінки зменшуються зі збільшенням k (рис. 4.5).

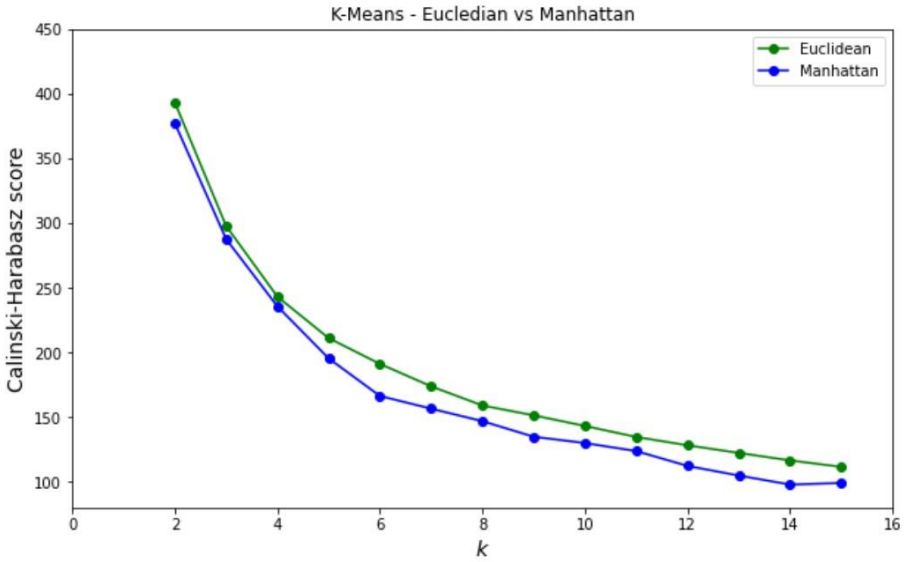


Рисунок 4.5 – Оцінка Калінського-Харабаша: K-середнє евклідове (зелене) проти манхеттенського (синє)

Алгоритм k-середніх був розроблений для кластерів сферичної форми з подібним розміром і щільністю. На жаль, цей набір даних не був ні тим, ні іншим. Тому жодна з моделей не створила ідеально хороших кластерів, про що свідчать низькі оцінки силуету (менше 0,2 для k >= 3). Тим не менш, моделі k-середніх досить добре відображаються на силуетних графіках (рис. 4.6) та проекціях t-SNE.

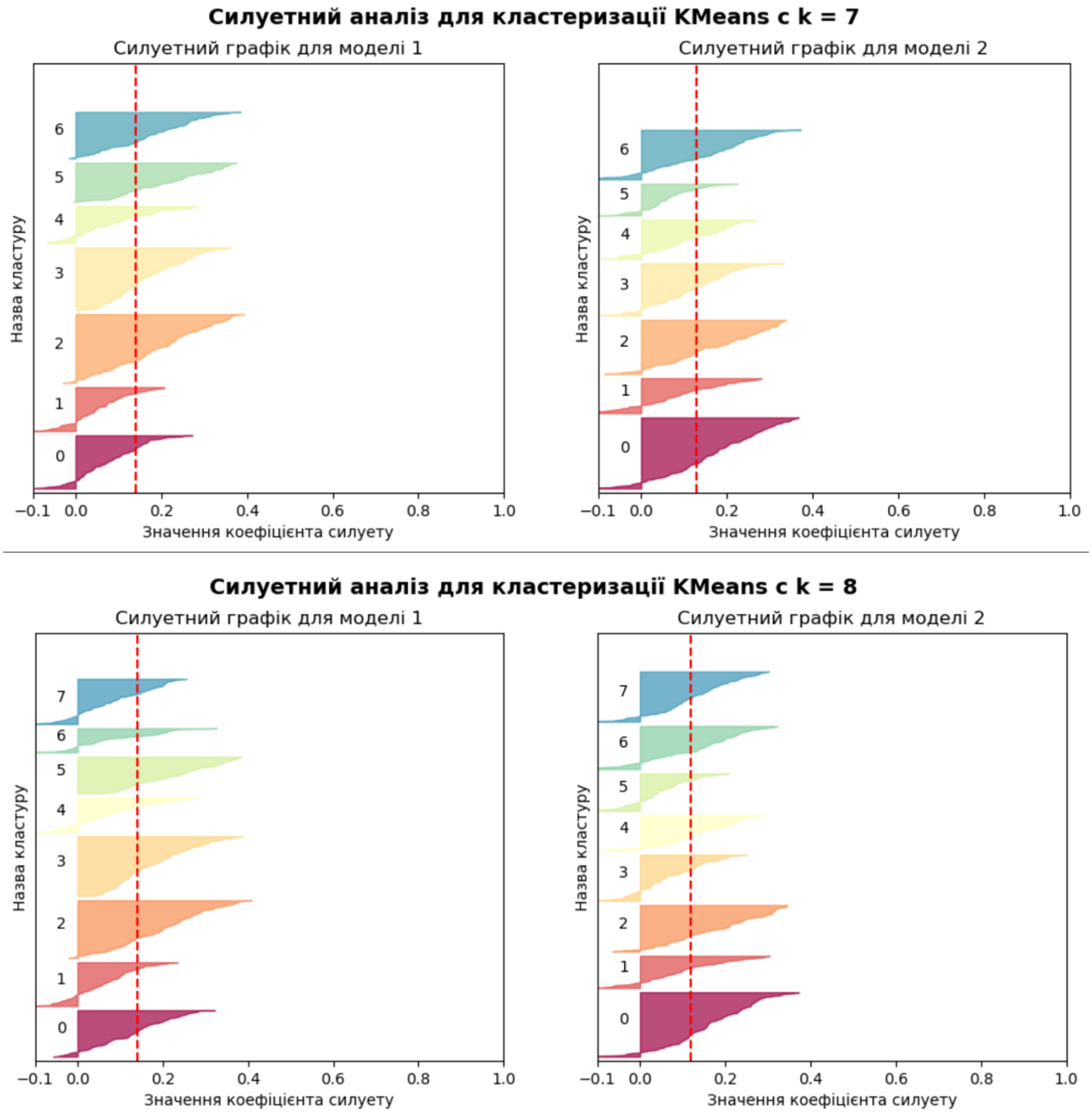


Рисунок 4.6 – Силуетний аналіз: K-середнє евклідове (ліворуч) проти манхеттенського (праворуч)

Хоча сформовані кластери не є ідеальними, вони досить щільні і досить добре розділені, особливо для евклідових (рис. 4.7).

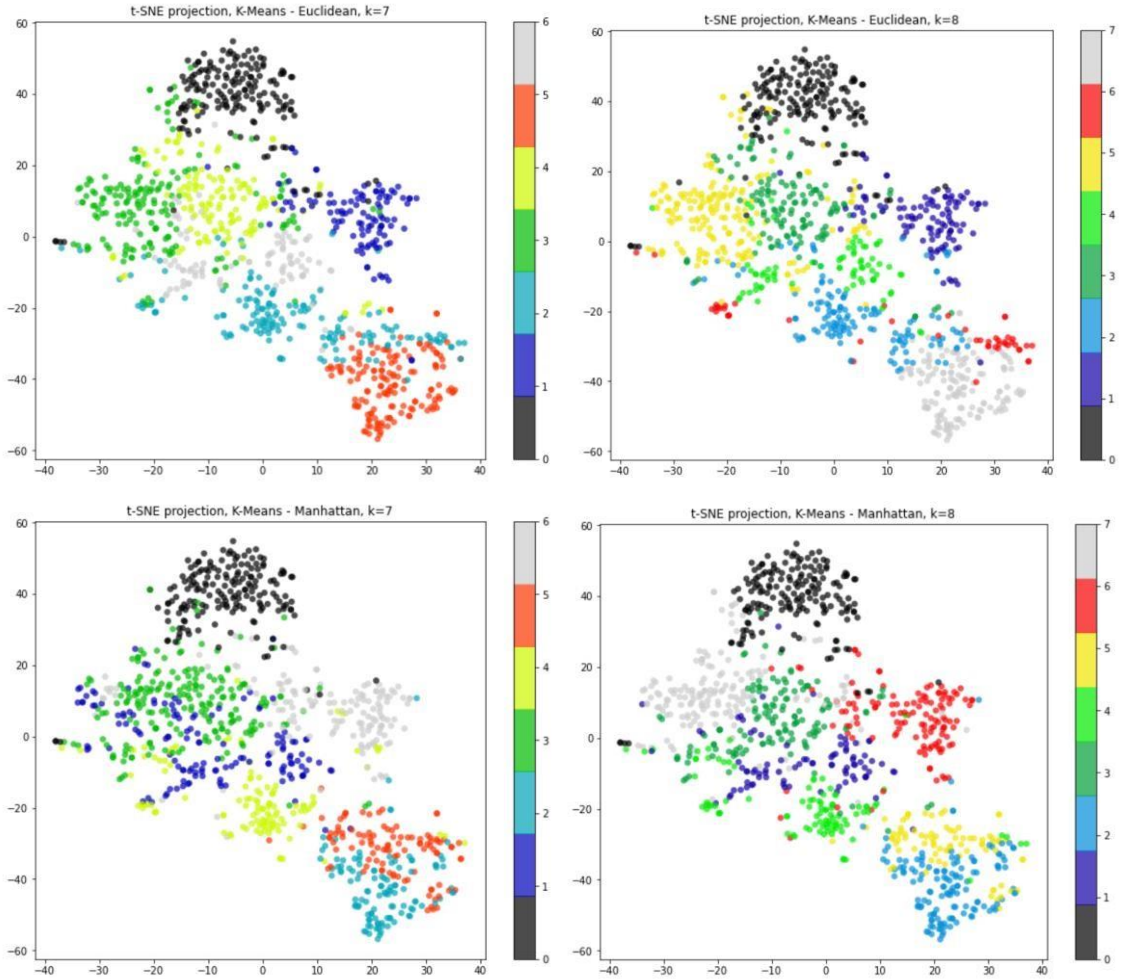


Рисунок 4.7 – Проекція t-SNE: K-середнє евклідове (вгорі) проти Мангеттенського (внизу)

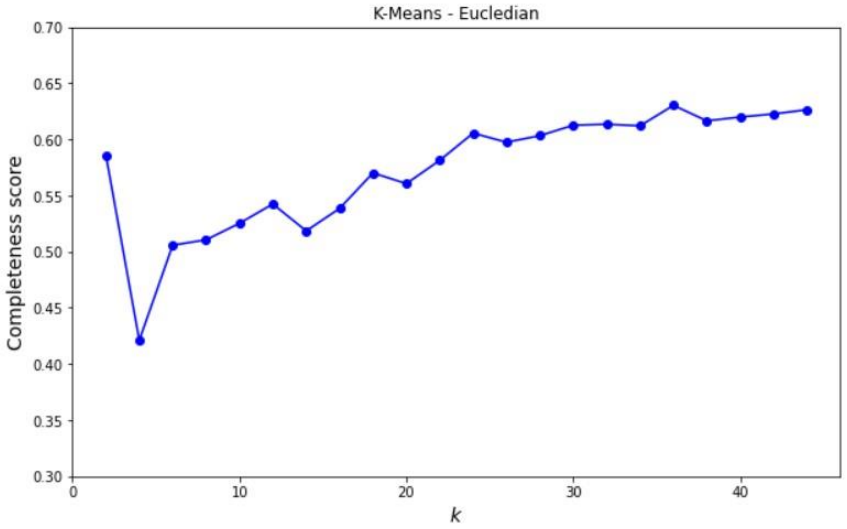
При k=7 Евклідів кластер також мав нижчий показник Девіса (1,76 проти 2,14), вищі показники Калінксі (174 проти 156) та силуету (0,16 проти 0,13), ніж Манхеттенський кластер. Тому було обрано евклідову модель як оптимальну з k=7 як оптимальною кількістю кластерів для цього набору даних.

4.2.2 Набір даних: Прийняті статті ICMLA 2014

Використовуючи два алгоритми k-середніх, на цьому наборі даних було навчено дев'яносто вісім моделей k-середніх з k=1-49 та двома мірами відстані. Як базову метрику знову було використано евклідову відстань. Для порівняння було використано метрику косинуса, оскільки вона має тенденцію добре працювати з текстом або розрідженими даними (навіть якщо дві точки знаходяться далеко один від одного за евклідовою відстанню, вони все одно можуть вважатися близькими за цією мірою відстані, якщо кут між двома точками вузький). Для цього набору даних

були надані золоті кластери (що показують 24 кластери), тому отримані моделі були оцінені за допомогою зовнішніх (і однієї внутрішньої) мір. Оцінка повноти[13] використовувалася для вимірювання ступеня, до якого всі точки даного класу належать до одного кластера, оцінка однорідності[14] використовувалася для вимірювання ступеня, до якого всі кластери містять тільки точки одного класу, і, нарешті, силует використовувався для перевірки якості кластерів при різних значеннях k.

Стовпчик “combine”, який містив тексти з “keywords”, “title”, “abstract”, було використано як вхідні дані X, оскільки він показав найкращий загальний результат у нашому тестуванні. Якщо не враховувати k=2, обидві моделі показали зростання оцінок повноти та однорідності зі збільшенням k (рис. 4.8 і 4.9).



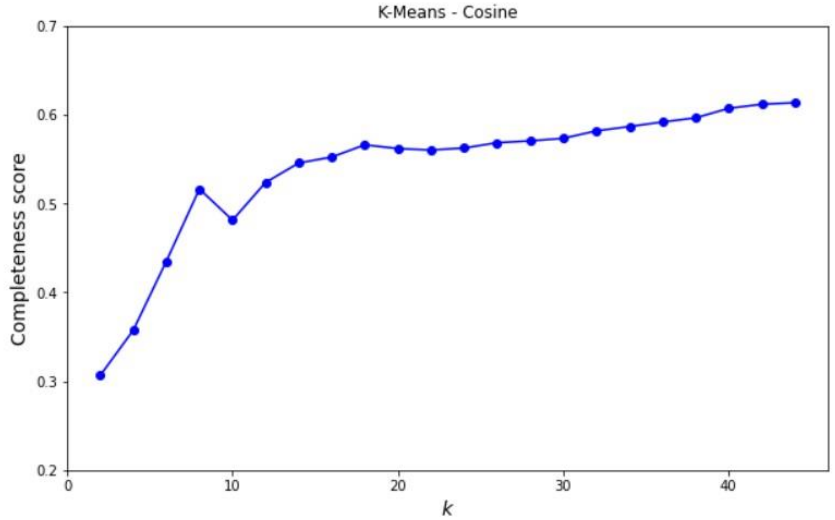


Рисунок 4.8 – Оцінки повноти: k-середнє евклідове (вгорі) проти k-середнього косинусного (внизу)

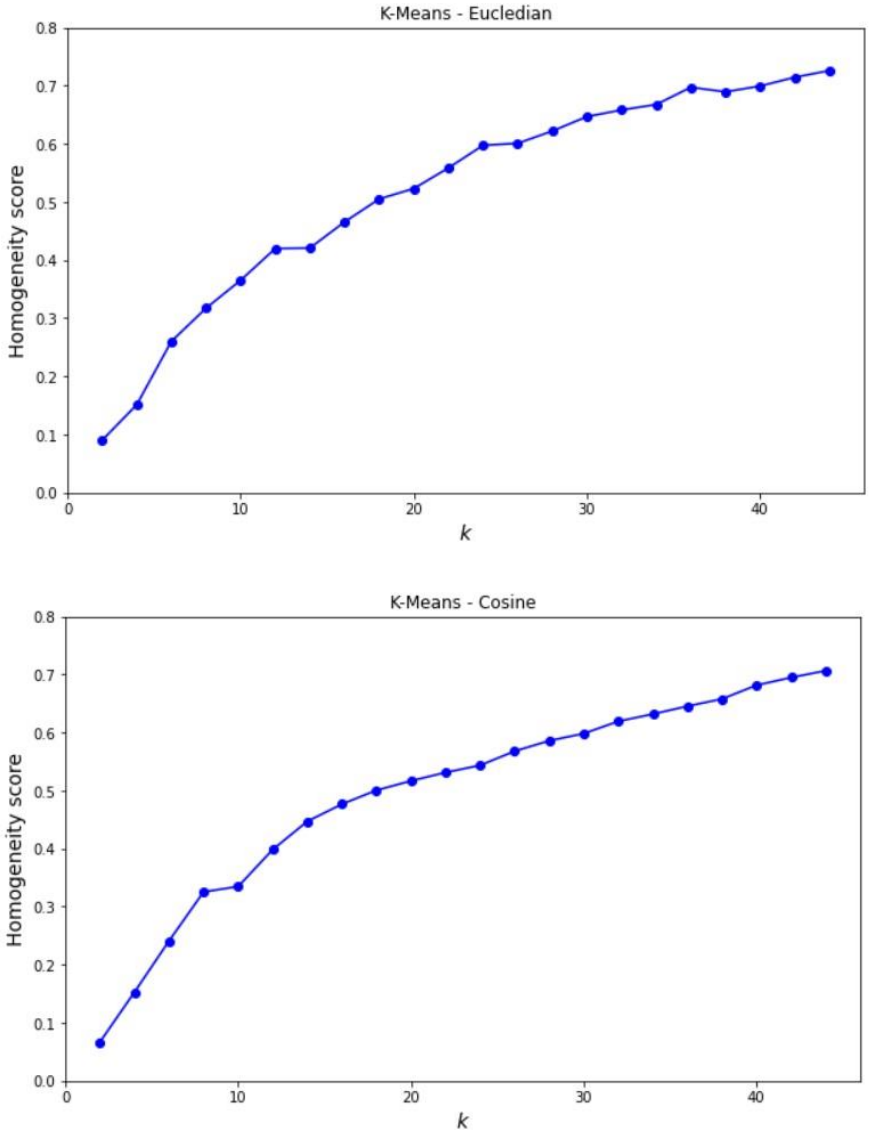


Рисунок 4.9 – Оцінки однорідності: k-середнє евклідове (вгорі) проти косинусного (внизу)

Загалом, оцінки показали, що кластери, створені евклідовими моделями, є більш повними та однорідними (рис. 4.10). Це узгоджується з позначеннями в рис. 4.11 і 4.12), де більше точок даних з одного сеансу, таких як нейронні мережі I і II, були згруповані разом за допомогою евклідової моделі, і більше її кластерів містять лише точки даних з одного класу. Евклідові нейронні мережі також мають кращі показники силуету, ніж косинусоїди, і краще відображаються на графіках силуету (рис. 4.12 і 4.13).

	k	completeness_euclidean	completeness_cosines	homogeneity_euclideans	homogeneity_cosines
0	2	0.585970	0.306342	0.090153	0.066738
1	4	0.421106	0.357319	0.151228	0.152740
2	6	0.505761	0.434965	0.260263	0.241011
3	8	0.510553	0.516255	0.317248	0.325441
4	10	0.525276	0.481175	0.365347	0.334768
5	12	0.542419	0.523547	0.419864	0.398371
6	14	0.518510	0.545460	0.420758	0.446957
7	16	0.538494	0.552170	0.465632	0.476606
8	18	0.570144	0.565922	0.505076	0.500566
9	20	0.560604	0.561677	0.523101	0.516710
10	22	0.581348	0.560059	0.558515	0.531264
11	24	0.605560	0.562309	0.597427	0.543523
12	26	0.597457	0.568320	0.600817	0.567968
13	28	0.603318	0.570369	0.622020	0.585900
14	30	0.612626	0.573218	0.646794	0.598361
15	32	0.613495	0.581640	0.658287	0.619375
16	34	0.612101	0.586500	0.667599	0.631890
17	36	0.630284	0.591631	0.697218	0.645428
18	38	0.616527	0.596263	0.689426	0.657943
19	40	0.619899	0.606986	0.699344	0.681607
20	42	0.622651	0.611578	0.714457	0.694883
21	44	0.626393	0.613308	0.725949	0.706584
22	46	0.626823	0.619410	0.734293	0.725054
23	48	0.630276	0.621747	0.745234	0.739143

Рисунок 4.10 – Оцінки повноти та однорідності для k-середніх евклідових та косинусних

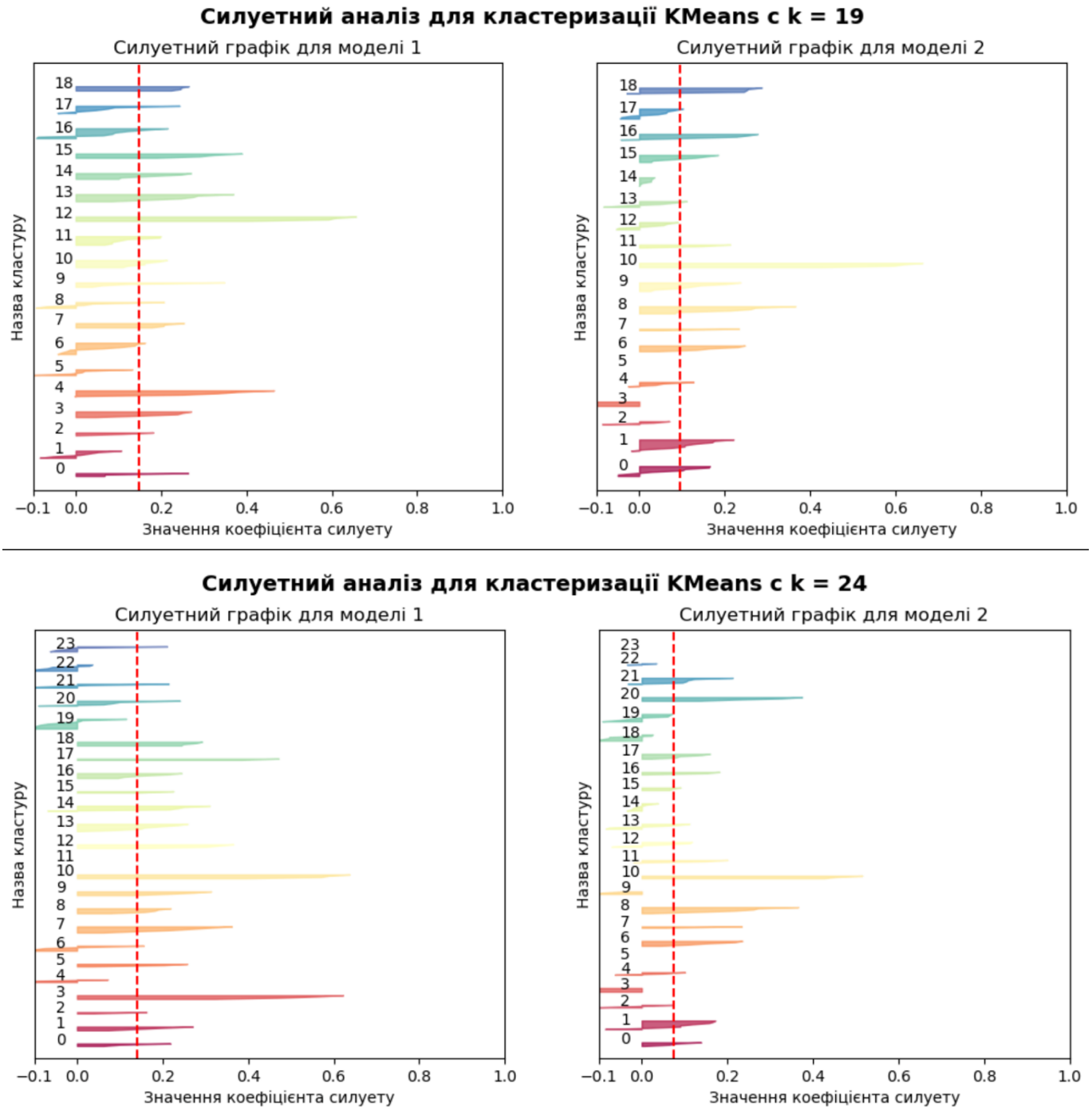


Рисунок 4.11 – Силуетний аналіз: K-середні евклідові евклідові (ліворуч) проти косинусів (праворуч)

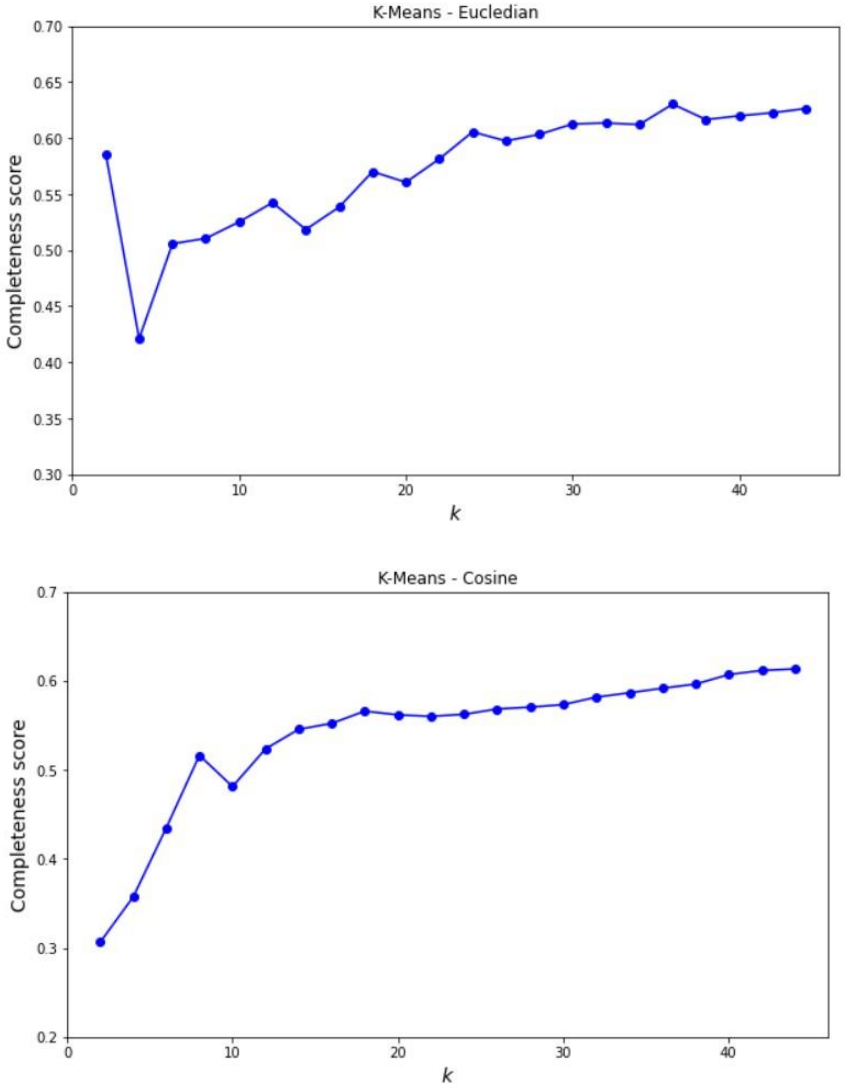


Рисунок 4.12 – Оцінки повноти: k-середнє евклідове (вгорі) проти k-середнього косинусного (внизу)

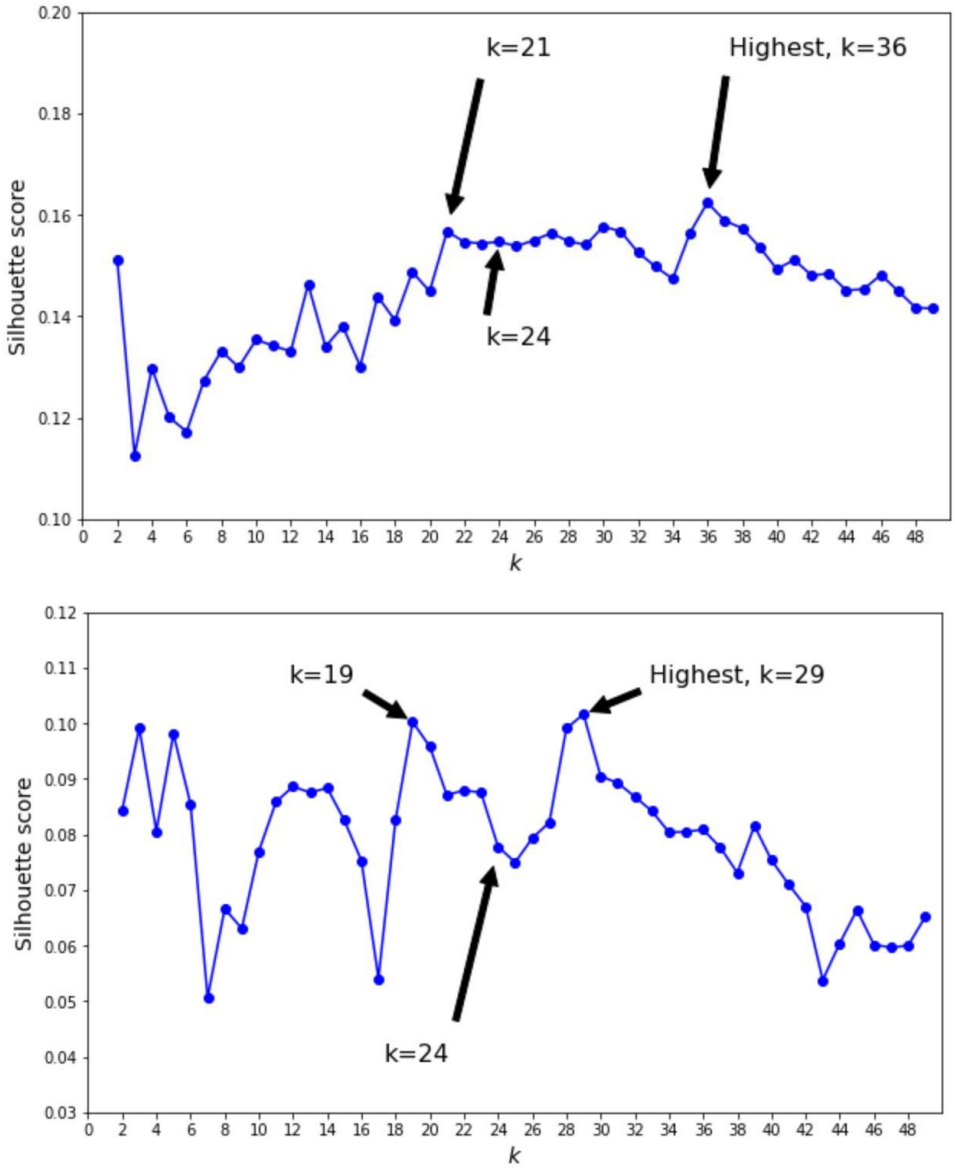


Рисунок 4.13 – Оцінки силуету: k-середнє евклідове (вгорі) проти k-середнього косинусного (внизу)

На проекціях t-SNE обидві моделі показали добре відокремлені кластери за нижчих k, хоча зі збільшенням k додаткові кластери було важче ідентифікувати (рис. 4.14 та 4.15).

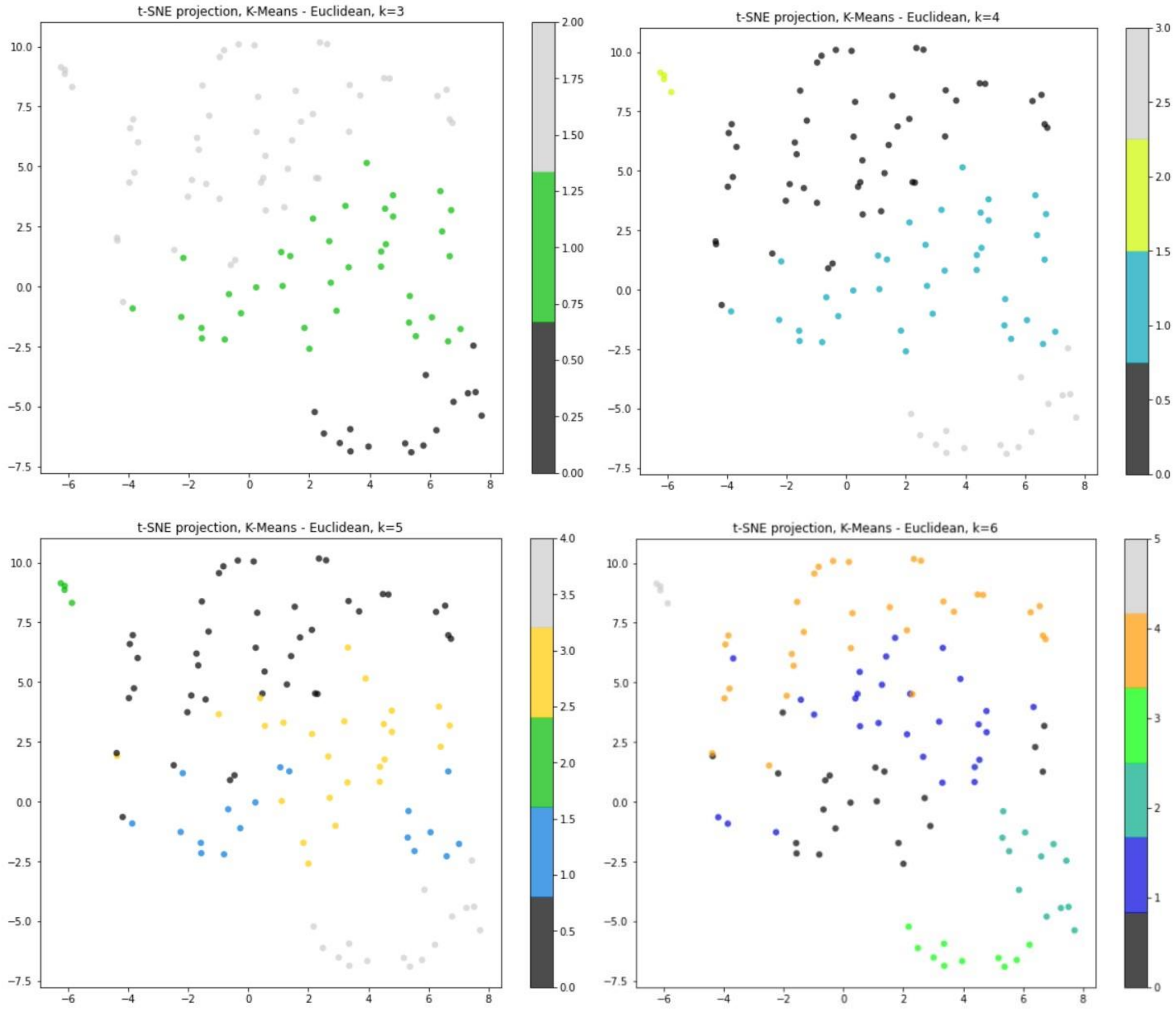


Рисунок 4.14 – Прогнози t-SNE: k-середнє евклідове при збільшенні k від 3 до 6

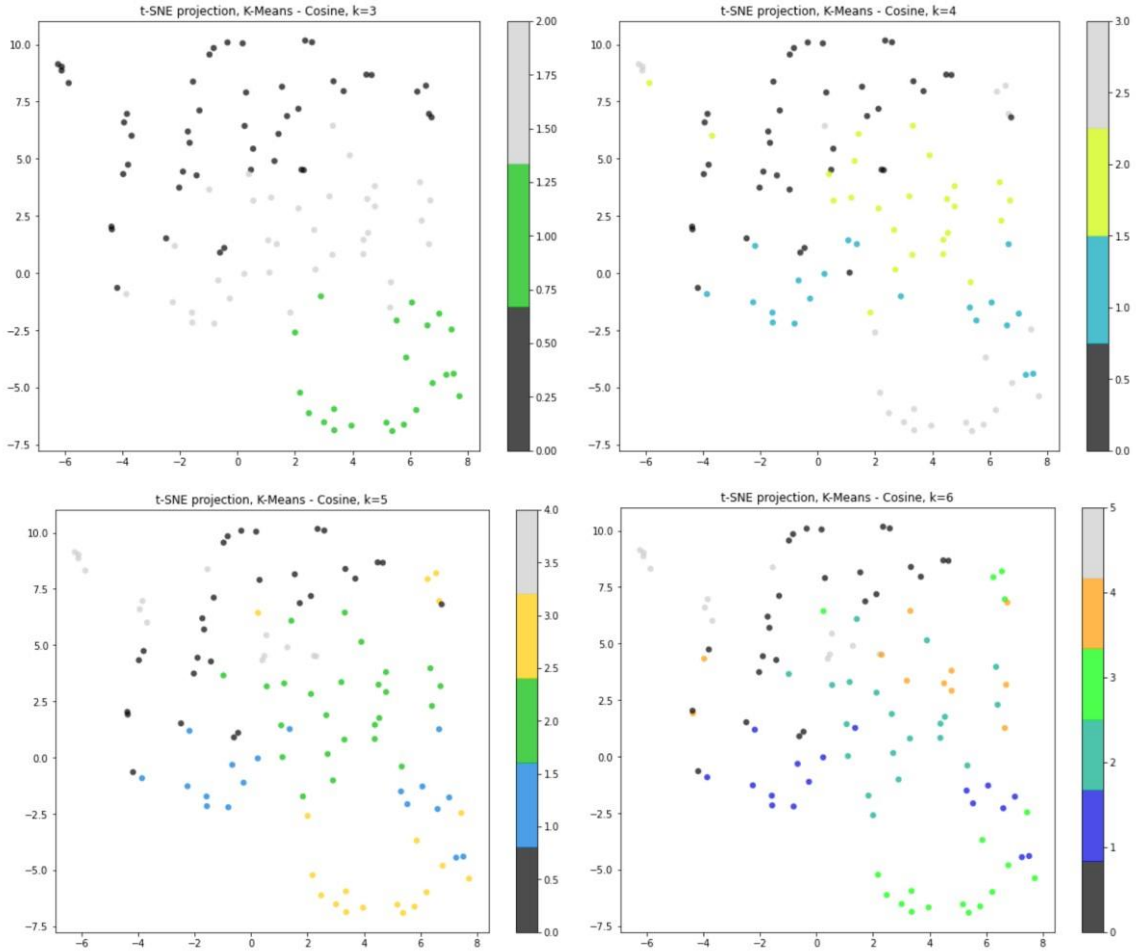


Рисунок 4.15 – Прогнози t-SNE: k-середній косинус при збільшенні k від 3 до 6

При k=24 евклідова модель дала краще розділені кластери, ніж косинусна (рис. 4.16).

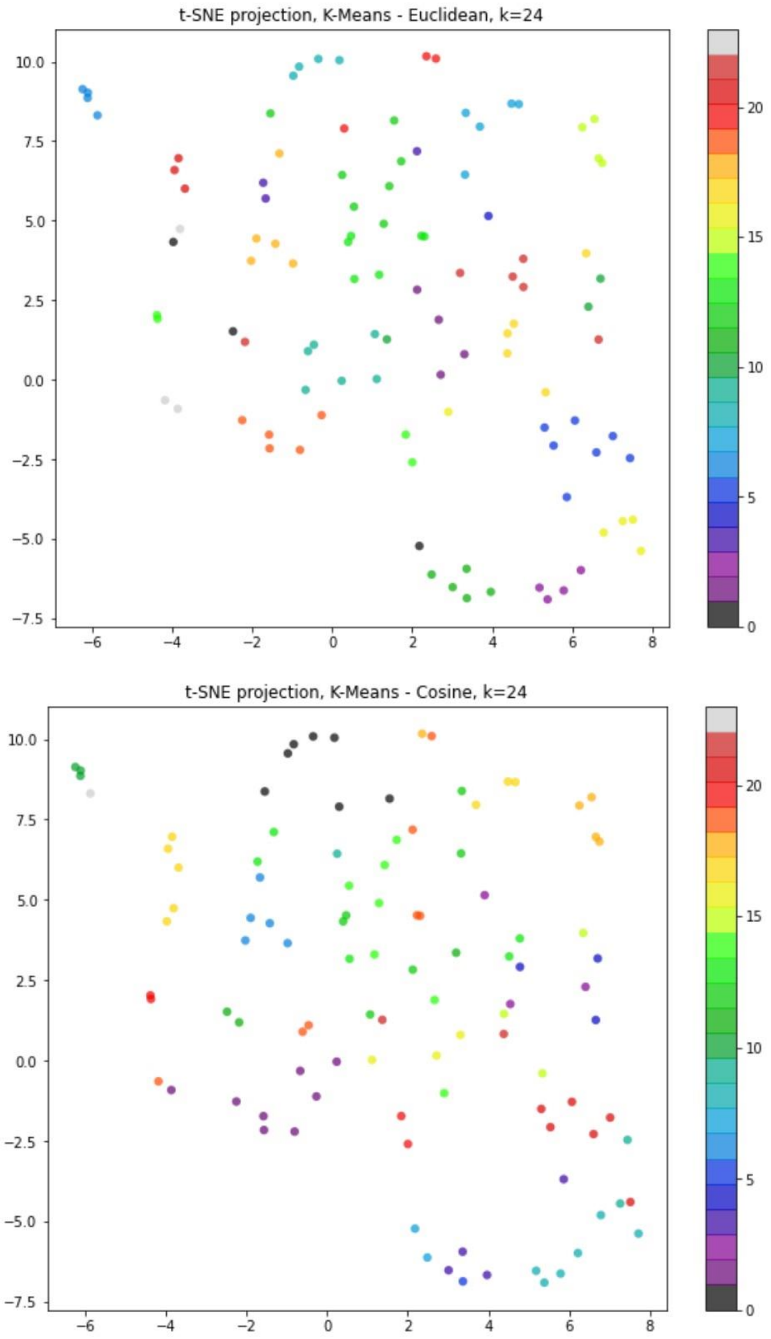


Рисунок 4.16 – Проекції t-SNE (k=24): k-середнє евклідове (вгорі) проти косинусного (внизу)

Коли текстові дані були перетворені в цифри, вийшло 105 стовпчиків ознак. Це призвело до того, що дані виявилися дуже розрідженими, і їх було важко кластеризувати. Навіть попри те, що для зменшення кількості ознак до 31 було використано метод PCA, отримані кластери все одно було важко ідентифікувати. Тим не менш, хоча моделі k-середніх не є ідеальними, вони дають досить добре відокремлені кластери. З двох моделей було обрано евклідову з k=24 як оптимальну

модель та оптимальну кількість кластерів для k-середніх. Вона має кращі показники повноти та однорідності порівняно з косинусоїдою або моделями з меншим k, а також краще відокремлює кластери як на силуеті, так і на графіках t-SNE.

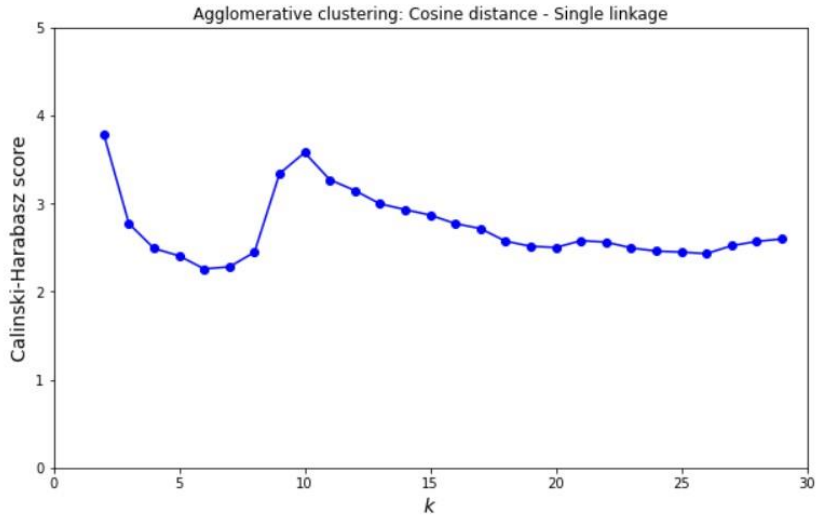
4.3 Ієрархічна кластеризація

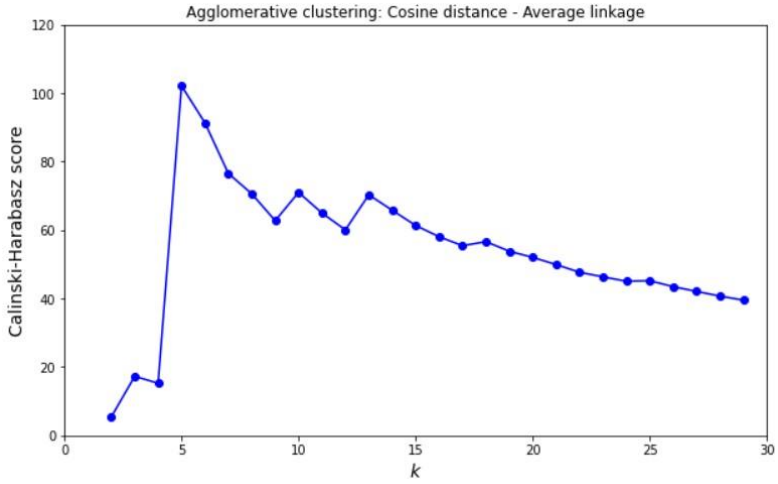
4.3.1 Набір даних: Відгуки про подорожі

Використовуючи sklearn Agglomerative Clustering[15], на цьому наборі даних було навчено понад 260 моделей з k=1-29, трьома мірами подібності (евклідова, манхеттенська та косинусна) та трьома метриками відстані (однозв'язкова, середньозв'язкова та повнозв'язкова). Косинусна подібність враховує кут між двома векторами, тоді як евклідова (пряма лінія) та манхеттенська (відстань у блоках) – відстань між двома точками. Single-link використовує найменшу відстань, average-link використовує середню відстань, а complete-link використовує найбільшу відстань при вимірюванні відстані між точкою в кластері та точкою в інших кластерах.

Знову ж таки, без золотих кластерів, отримані моделі були оцінені за трьома внутрішніми показниками: показник силуету, показник Девіса-Булдіна та показник Калінкі-Харабаша.

Моделі з повними зв'язками дають більш щільні та краще розділені кластери, ніж середні або з одним зв'язком, про що свідчать вищі показники Калінські для різних k (рис. 4.17-4.19).





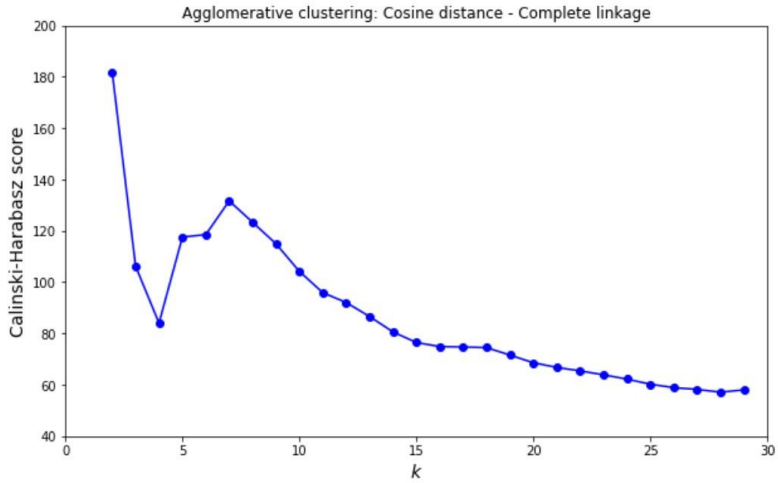
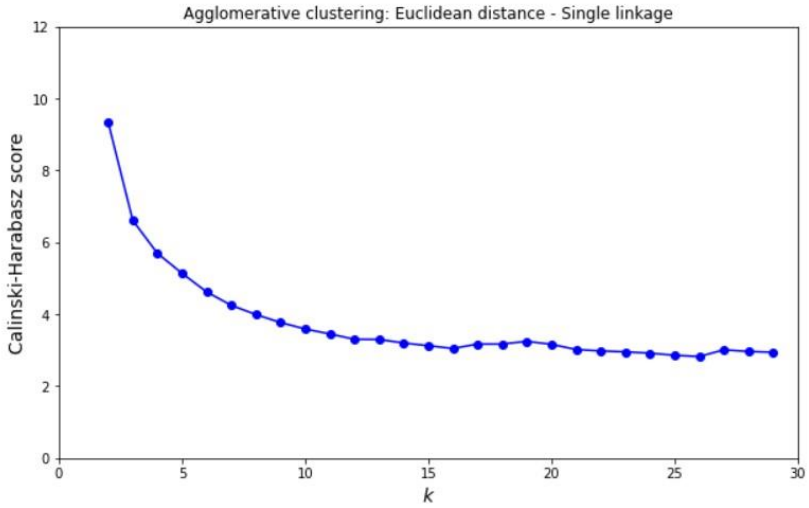
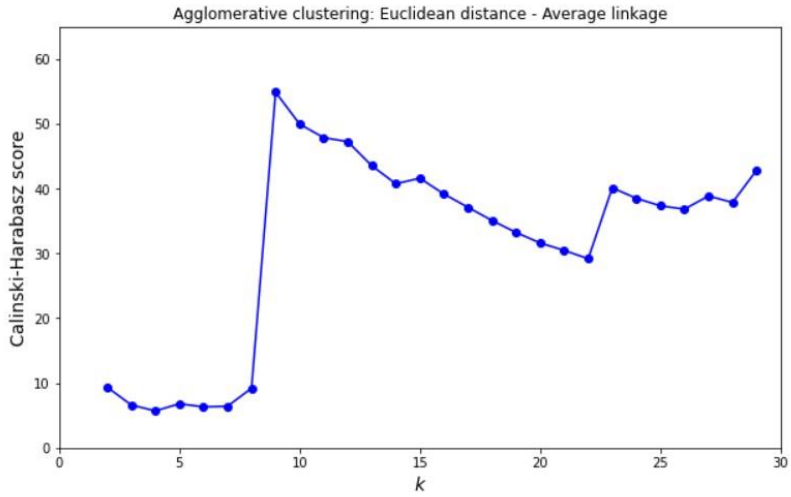


Рисунок 4.17 – Оцінки Калінського Харабаша: Косинусні моделі





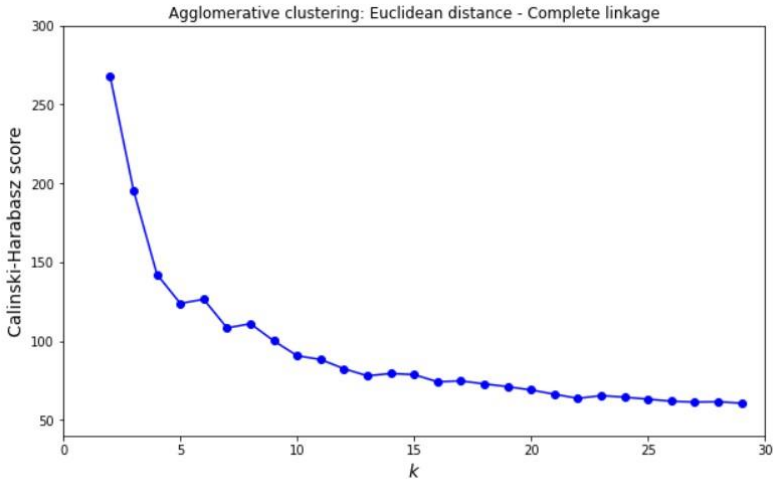


Рисунок 4.18 – Оцінки Калінського Харабаша: Евклідові моделі

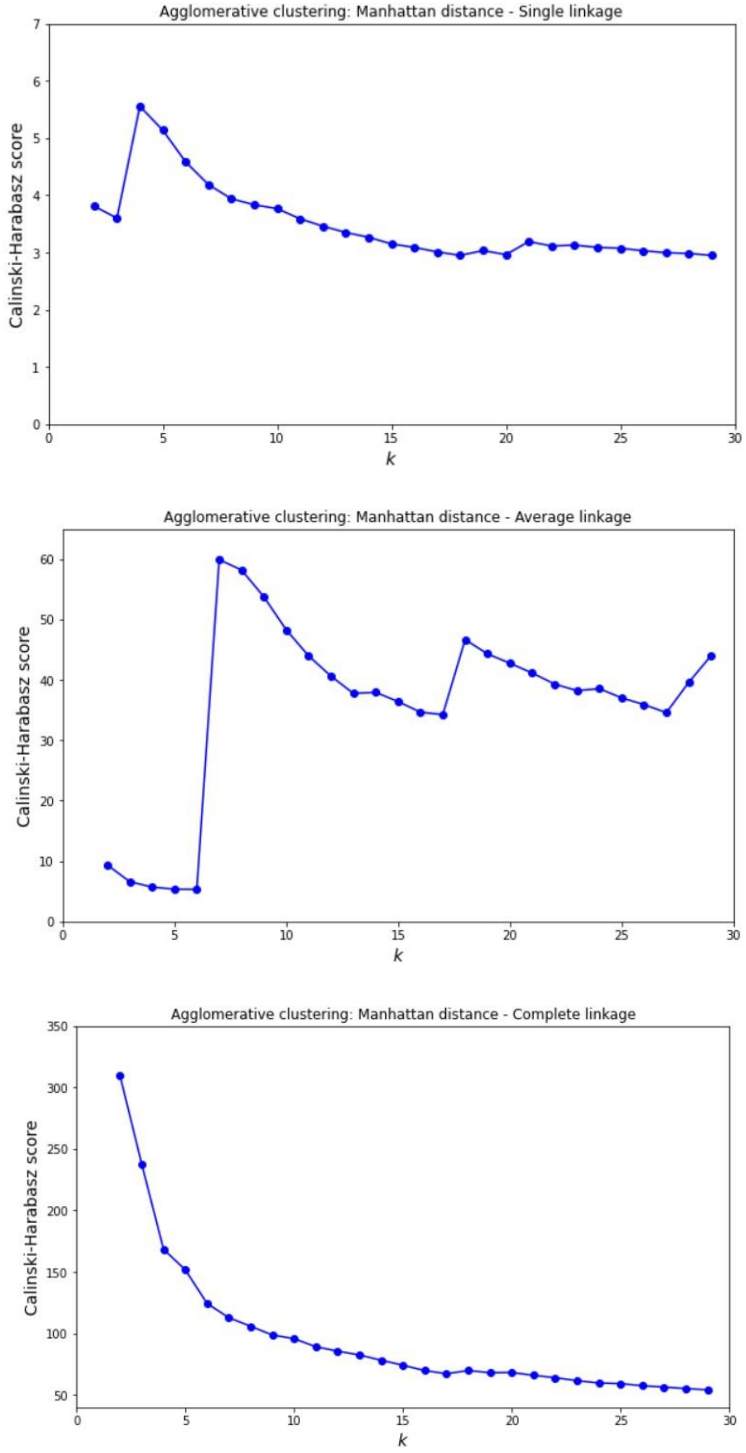


Рисунок 4.19 – Оцінки Калінського Харабаша: Мангеттенські моделі

У парі з повним зв'язком всі три міри подібності, як видається, працюють добре. Загалом, оцінки Калінські зменшувалися зі збільшенням k, але манхеттенські моделі виявилися кращими за нижчих k, тоді як косинуси були кращими за вищих k (з евклідовими моделями десь посередині між ними). Оцінки Девіса-Болдіна були досить хорошими для всіх моделей (здебільшого менше двох), причому моделі з

однією та середньою ланкою мали дещо кращі оцінки, ніж моделі з повною ланкою (рис. 4.20-4.22).

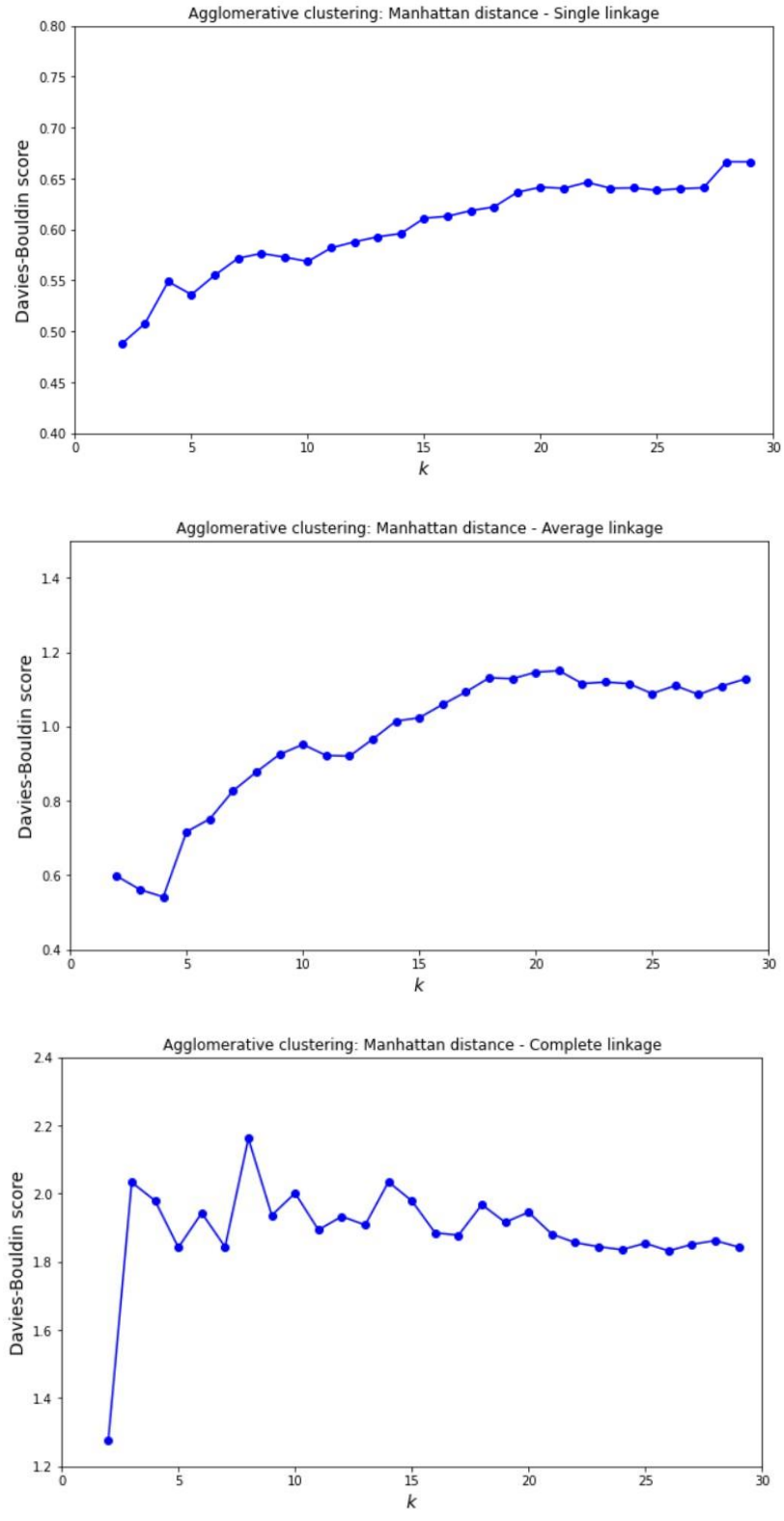
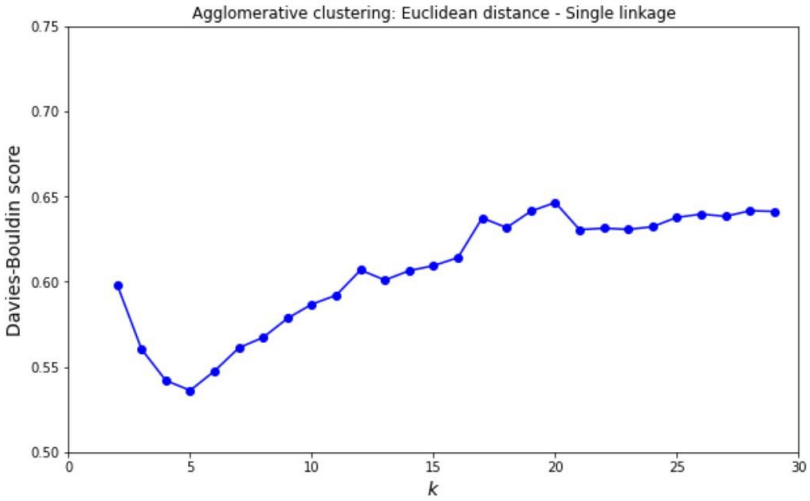
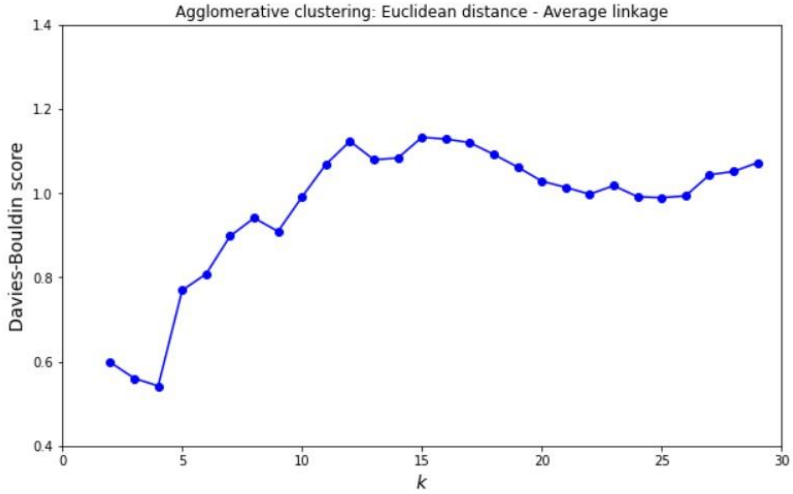


Рисунок 4.20 – Оцінки Девіса-Болдіна: Манхеттенські моделі





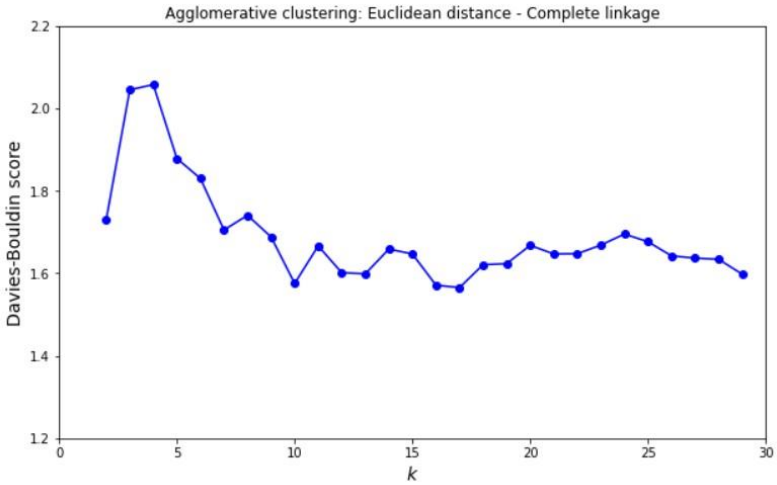


Рисунок 4.21 – Оцінки Девіса-Болдіна: Евклідові моделі

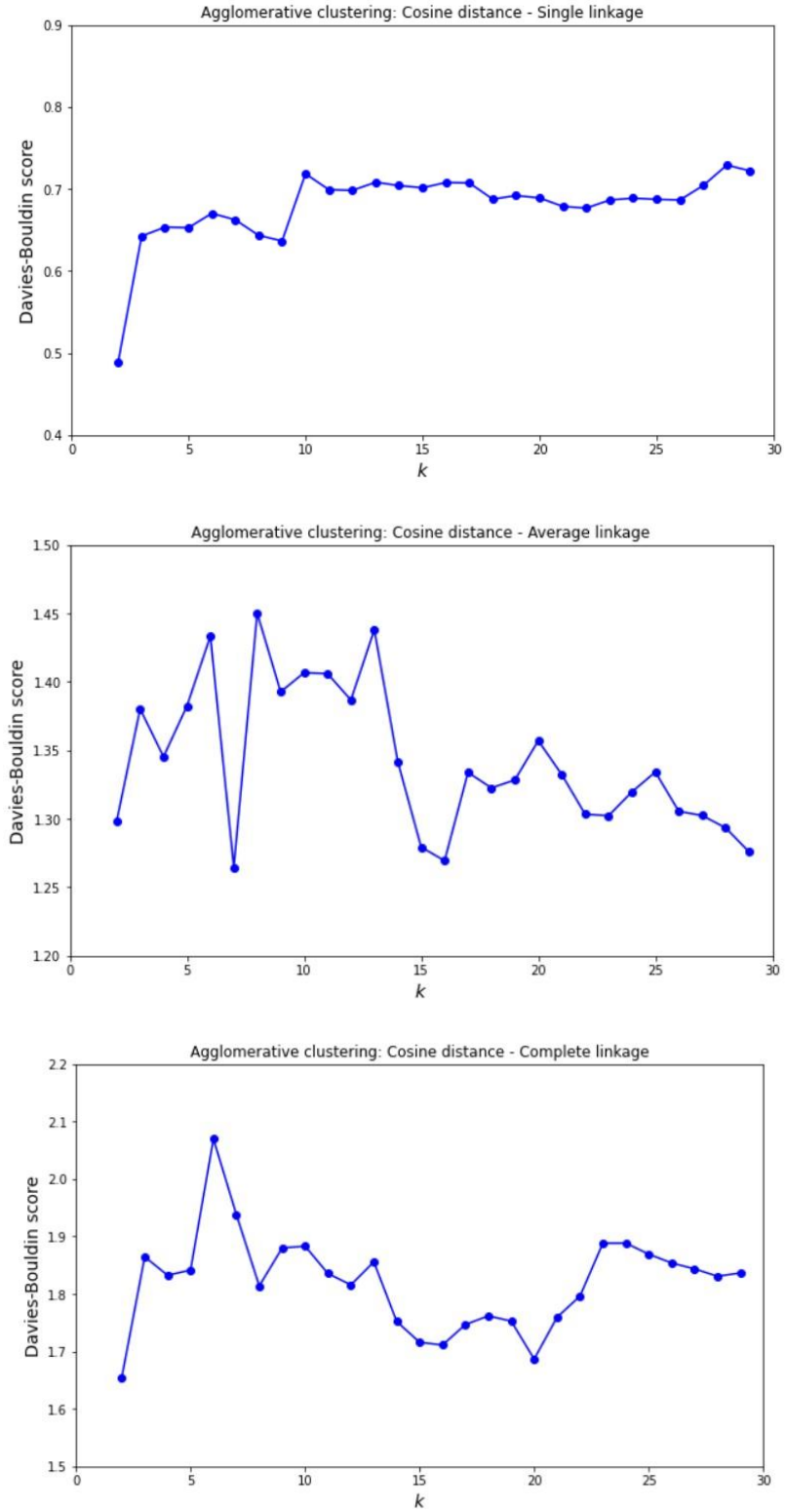


Рисунок 4.22– Оцінки Девіса-Болдіна: Косинусні моделі

У парі з одноланковою моделлю як Манхеттенська, так і Евклідова моделі дали кращі результати (0,54 при k=5), ніж косинусоїда (0,65 при k=5). Однак, як і для k-середніх, всі моделі дали низькі оцінки силуету (менше 0,3 для k>=3). На проекціях t-SNE як Манхеттенська, так і Евклідова моделі з повними зв'язками показали добре

відокремлені кластери при k=3 (рис. 4.23), але не давали хороших кластерів при збільшенні k (додаткові кластери були невеликими і їх було важко ідентифікувати).

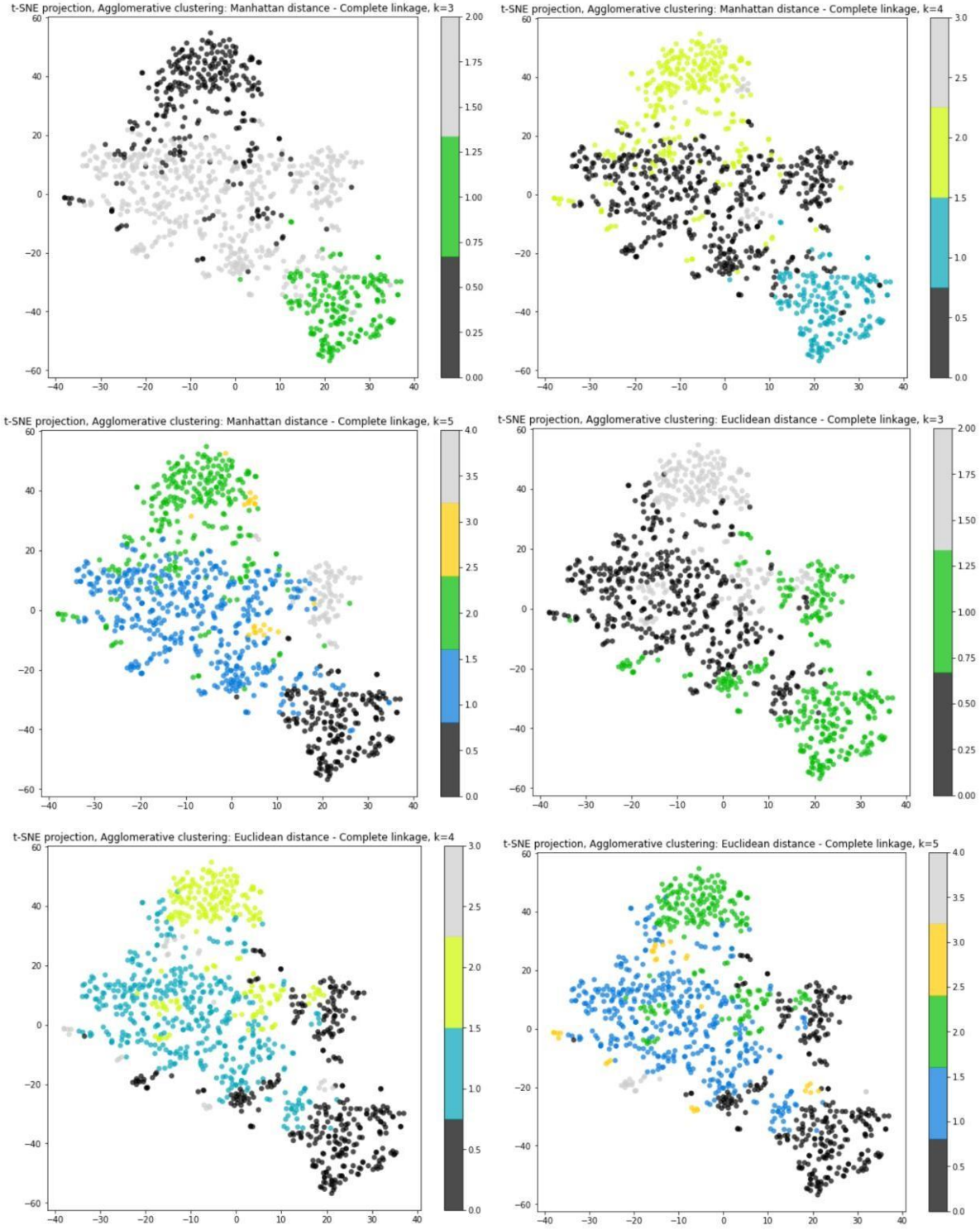


Рисунок 4.23 – Проекції t-SNE: Манхеттенська повна, k=3-5 та Евклідова повна, k=3-5

При k=7 косинусоїдальна модель з повним зв'язком показала краще розділені кластери, ніж евклідова або манхеттенська. Однак, модель з одним і повним зв'язком не дала ні добре збалансованих, ні добре відокремлених кластерів (рис. 4.24).

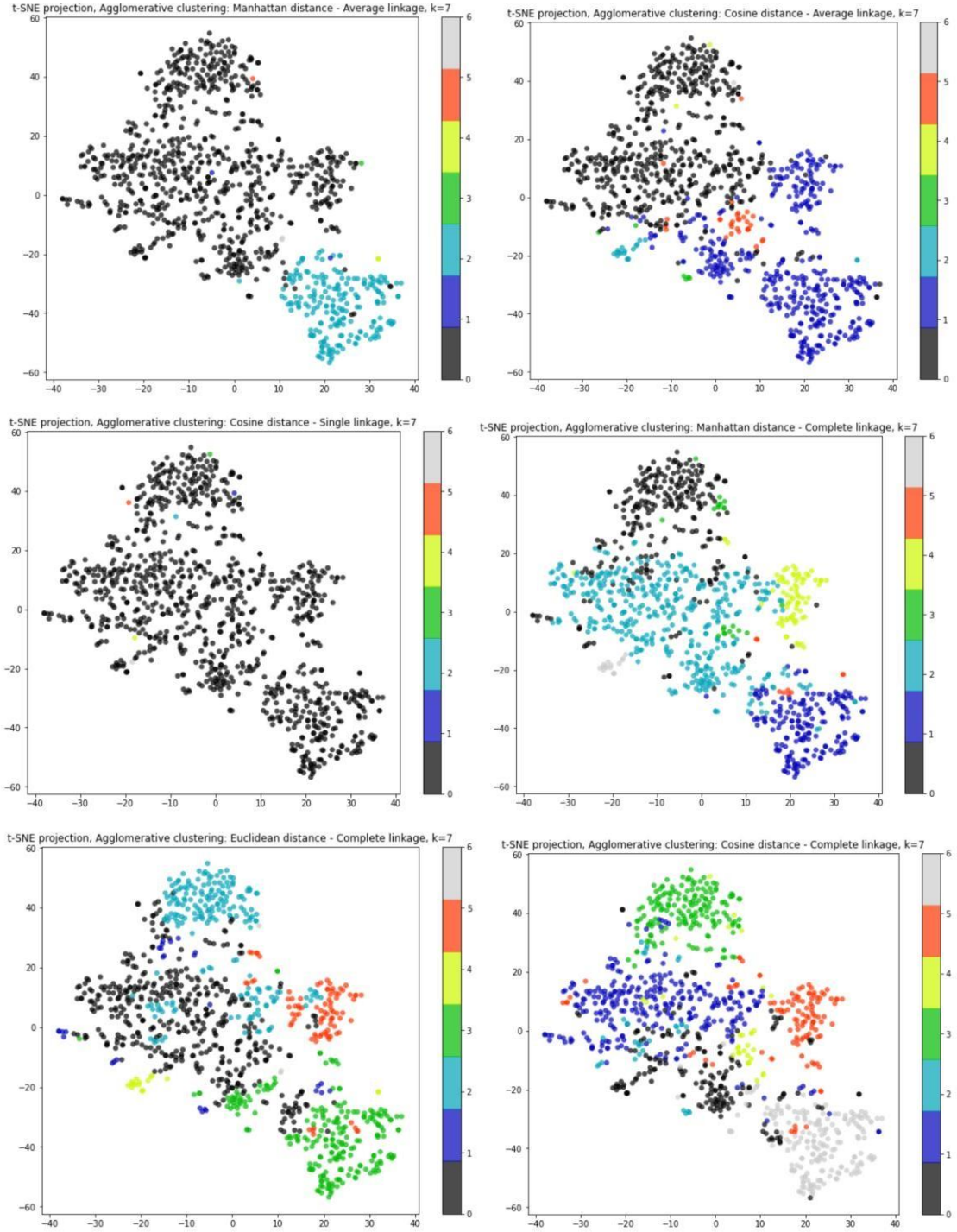


Рисунок 4.24 – Проекції t-SNE (k=7): Манхеттенська та косинусоїдальна (вгорі) проти косинусоїдальної та манхеттенської (посередині), евклідової та косинусоїдальної (внизу)

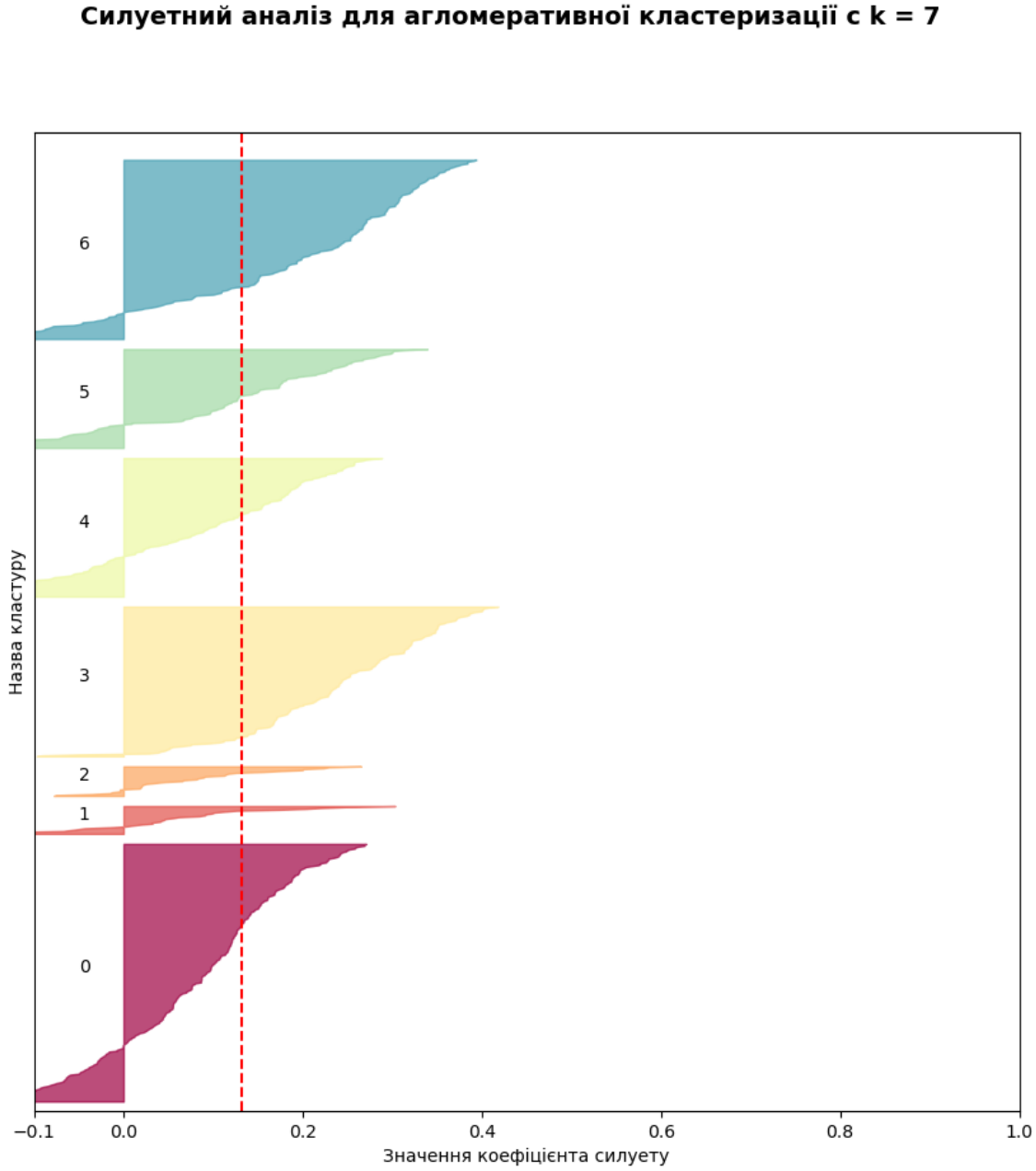
Повний зв'язок зазвичай надає перевагу кулястим формам і є менш чутливим до шуму та викидів[17]. Тому в поєднанні з косинусоїдальною мірою схожості він, як видається, дає досить хороші кластери для цього набору даних. Хоча при k=3 як евклідова, так і манхеттенська моделі повного зв'язку дали краще розділення, вищу оцінку Калінські (рис. 4.25) та нижчу оцінку Девіса (рис. 4.26), вони виявили лише три кластери, і така мала кількість кластерів не є надто значущою для справи. Тому для цього алгоритму був обран косинусоїдальний повний зв'язок з k=7 як оптимальну модель. Вона має найвищу оцінку Калінскі при k=7 (132), відносно низьку оцінку Девіса (1,94) і добре відокремлені кластери на графіку силуетного коефіцієнта (рис. 4.27) та проекції t-SNE.

	k	cosine-single	cosine-average	cosine-complete	euclidean-single	euclidean-average	euclidean-complete	manhattan-single	manhattan-average	manhattan-complete
0	2	3.787220	5.416734	181.780777	9.342267	9.342267	267.709413	3.809731	9.342267	309.543969
1	3	2.771441	17.285063	106.188634	6.609191	6.609191	195.405860	3.600932	6.591341	237.505569
2	4	2.489744	15.251828	84.043466	5.693640	5.693640	142.258031	5.556358	5.693640	168.381686
3	5	2.405053	102.109272	117.559678	5.138337	6.795051	123.811982	5.138337	5.346979	151.712328
4	6	2.257358	91.304075	118.517489	4.621251	6.312727	126.434424	4.584204	5.309232	124.403883
5	7	2.280077	76.529386	131.648893	4.247459	6.410286	108.257734	4.186017	59.903870	112.785392
6	8	2.445759	70.575933	123.321970	3.992196	9.193597	110.947673	3.939241	58.199306	105.846029
7	9	3.339324	62.736421	114.941378	3.769467	54.878018	100.053279	3.833316	53.746329	98.712930
8	10	3.577378	70.994958	104.182362	3.589807	49.943263	90.626563	3.766919	48.284952	95.745509
9	11	3.269711	64.961400	95.910786	3.452469	47.858980	88.255684	3.589841	43.976831	89.153463
10	12	3.147112	60.045221	92.061090	3.303838	47.237609	82.399388	3.459862	40.617888	85.635203
11	13	2.995780	70.288671	86.599496	3.299422	43.524787	77.911347	3.350556	37.770654	82.506105
12	14	2.930421	65.777036	80.613782	3.199715	40.738998	79.376761	3.265861	37.932504	78.212664
13	15	2.868038	61.328728	76.471149	3.125539	41.596325	78.680588	3.149814	36.413904	74.148561

Рисунок 4.26 – Оцінки Калінського-Харабаша для різних комбінацій агломеративних моделей

	k	cosine-single	cosine-average	cosine-complete	euclidean-single	euclidean-average	euclidean-complete	manhattan-single	manhattan-average	manhattan-complete
0	2	0.489301	1.298459	1.654703	0.598190	0.598190	1.729692	0.487803	0.598190	1.278057
1	3	0.642505	1.380152	1.864555	0.560328	0.560328	2.044680	0.507675	0.561395	2.033082
2	4	0.653362	1.345352	1.832398	0.541941	0.541941	2.057209	0.548818	0.541941	1.979288
3	5	0.652696	1.382374	1.841618	0.536151	0.769110	1.878246	0.536151	0.717163	1.842000
4	6	0.670213	1.433394	2.069629	0.547533	0.807978	1.829682	0.555094	0.751653	1.943182
5	7	0.662316	1.264549	1.938279	0.561141	0.897734	1.704912	0.571738	0.827202	1.843471
6	8	0.643367	1.450093	1.813936	0.567250	0.940719	1.740446	0.576565	0.877864	2.161831
7	9	0.636534	1.392667	1.879945	0.578468	0.909047	1.687928	0.572871	0.925141	1.936362
8	10	0.718422	1.406619	1.883133	0.586728	0.990788	1.575225	0.568647	0.951838	2.001326
9	11	0.699059	1.406017	1.835464	0.592014	1.068394	1.666171	0.581936	0.922115	1.893624
10	12	0.698200	1.386730	1.815625	0.606792	1.122929	1.601295	0.587823	0.920500	1.933102
11	13	0.708108	1.438003	1.855772	0.600908	1.079189	1.598570	0.592855	0.966014	1.908000
12	14	0.704017	1.341894	1.751963	0.606410	1.083124	1.658490	0.596015	1.014607	2.034362
13	15	0.701388	1.279295	1.715870	0.609351	1.132318	1.646710	0.611054	1.023663	1.978714

Рисунок 4.27 – Оцінки Девіса-Болдіна для різних комбінацій агломеративних моделей


Силуетний аналіз для агломеративної кластеризації с k = 7
Назва кластуру
6
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
Значення коефіцієнта силуету

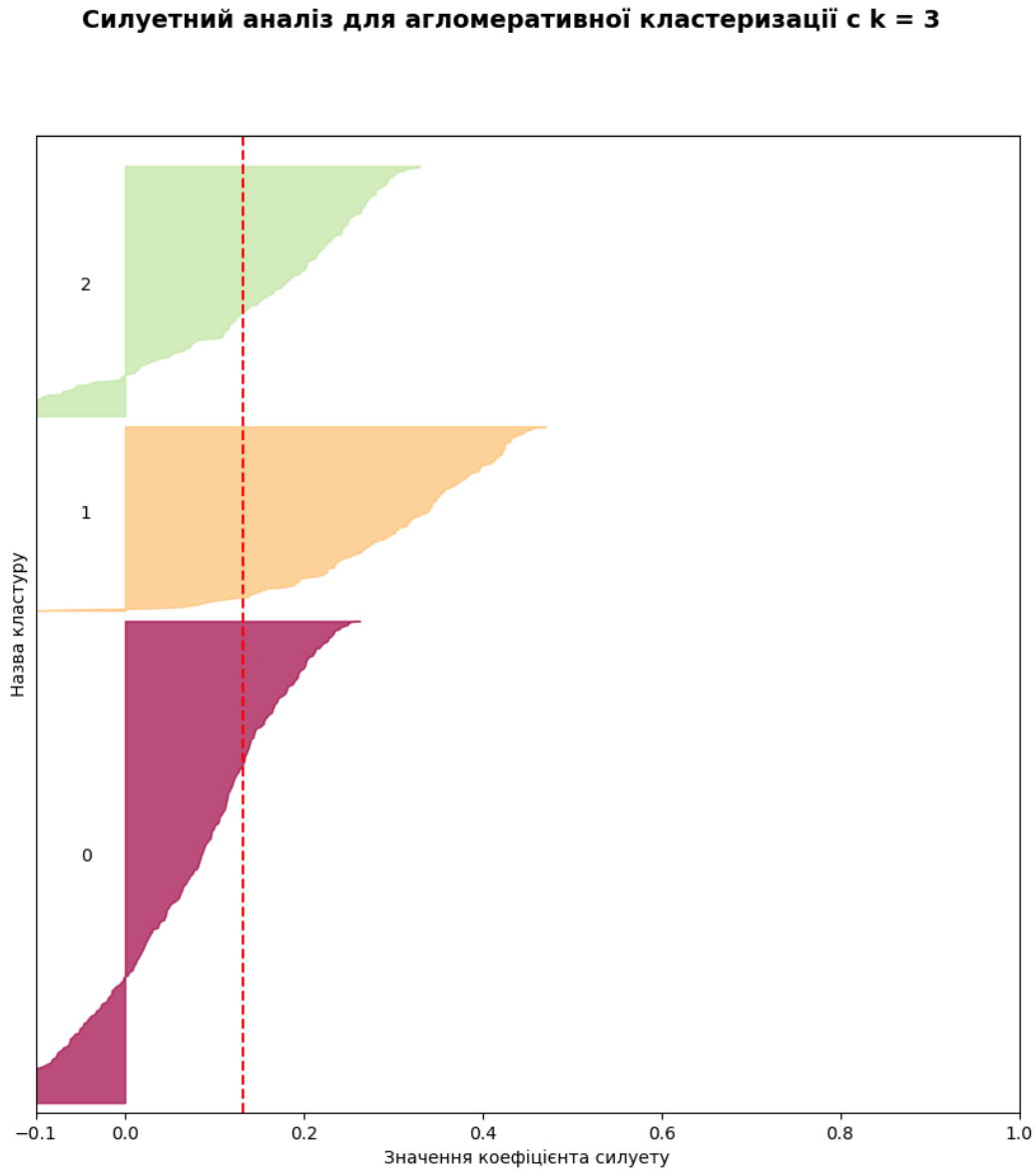


Рисунок 4.25 – Косинусоїдальна повна ланка, k=7 (ліворуч) та Манхеттенська повна ланка, k=3 (праворуч)

4.3.2 Набір даних: Прийняті статті ICMLA 2014

Використовуючи sciру Hierarchy Clustering[16], понад 260 моделей було навчено на наборі даних 2 з використанням k=1-49, трьох мір подібності (евклідова, мангеттен і косинус) і трьох метрик відстані (однозв'язкова, середньозв'язкова і повна зв'язність). Після отримання золотих кластерів отримані моделі були оцінені за допомогою зовнішніх мір: оцінки повноти та однорідності.

Загалом, одноланкові моделі отримали вищі оцінки повноти, ніж середні та повноланкові моделі при нижчих значеннях k, але оцінки зростали і почали

зближуватися зі збільшенням k, і всі моделі отримали однакові оцінки при вищих значеннях k (рис. 4.28).

	k	cosine-single	cosine-average	cosine-complete	euclidean-single	euclidean-average	euclidean-complete	cityblock-single	cityblock-average	cityblock-complete
0	2	0.601829	0.540787	0.314942	0.669615	0.669615	0.602672	0.669615	0.520184	0.327161
1	3	0.549282	0.437335	0.379742	0.652103	0.622332	0.622332	0.652103	0.462619	0.382375
2	4	0.550750	0.461752	0.381278	0.632470	0.609145	0.390259	0.653476	0.480907	0.449613
3	5	0.560877	0.468268	0.424226	0.581485	0.562680	0.414303	0.592560	0.507973	0.446936
4	6	0.532538	0.499711	0.436176	0.578208	0.550648	0.455200	0.588248	0.529484	0.464956
5	7	0.480619	0.522066	0.462697	0.584603	0.468129	0.468003	0.615533	0.557622	0.479665
6	8	0.487390	0.532924	0.479177	0.585074	0.489725	0.500475	0.613601	0.567556	0.487499
7	9	0.490479	0.547647	0.505898	0.593008	0.493800	0.510189	0.608493	0.549186	0.486495
8	10	0.495397	0.557242	0.515105	0.572367	0.517146	0.516874	0.607621	0.558417	0.500508
9	11	0.504567	0.558648	0.520747	0.579144	0.520818	0.528245	0.619001	0.566025	0.510529
10	12	0.529402	0.571616	0.530916	0.586414	0.526758	0.529906	0.596030	0.569741	0.522670
11	13	0.531820	0.582755	0.532892	0.595336	0.537483	0.549762	0.595477	0.564990	0.541628
12	14	0.549430	0.580053	0.545892	0.597762	0.549007	0.562095	0.598019	0.569367	0.541857
13	15	0.566135	0.580774	0.553666	0.599168	0.553182	0.562855	0.604810	0.576678	0.553116
14	16	0.567098	0.585873	0.554475	0.609430	0.556276	0.567527	0.612959	0.588747	0.566817
15	17	0.572724	0.591511	0.562950	0.608120	0.558460	0.570997	0.610919	0.588885	0.580978
16	18	0.578763	0.591174	0.566410	0.606459	0.561326	0.564918	0.609510	0.594467	0.583056
17	19	0.580541	0.593971	0.569761	0.599757	0.569633	0.573737	0.612755	0.603937	0.590701
18	20	0.580169	0.594522	0.571300	0.598660	0.576145	0.583698	0.611310	0.604365	0.598072
19	21	0.583573	0.598922	0.575109	0.589767	0.570128	0.584634	0.614787	0.612445	0.602417
20	22	0.585019	0.603473	0.581146	0.588849	0.578224	0.589040	0.618895	0.610315	0.606668
21	23	0.584474	0.605874	0.583063	0.591565	0.584174	0.590836	0.608067	0.607829	0.607206
22	24	0.583925	0.611040	0.587383	0.592178	0.584799	0.586379	0.605473	0.611861	0.611294
23	25	0.585144	0.609728	0.585982	0.586505	0.585996	0.592348	0.605806	0.614278	0.612717
24	26	0.586266	0.608167	0.590761	0.589243	0.589377	0.592857	0.603320	0.618783	0.614099
25	27	0.594142	0.611475	0.596591	0.591833	0.593510	0.596198	0.601406	0.619775	0.618431
26	28	0.593307	0.614728	0.595799	0.595543	0.592741	0.595415	0.600295	0.622929	0.622167
27	29	0.597047	0.620210	0.596468	0.603125	0.595742	0.600329	0.602725	0.623183	0.621189
28	30	0.602357	0.622357	0.599697	0.601525	0.600237	0.604192	0.602848	0.625940	0.621362
29	31	0.611190	0.621541	0.602876	0.610698	0.601421	0.608486	0.608581	0.628657	0.621896

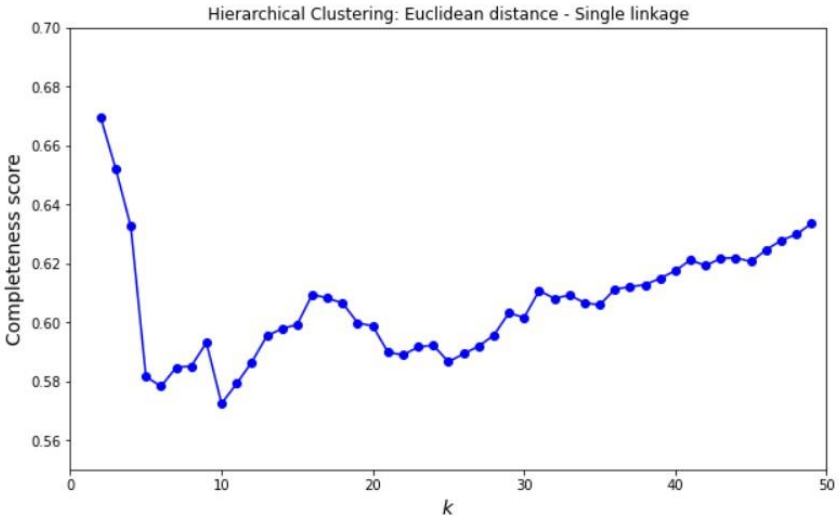
Рисунок 4.28 – Оцінки повноти для різних комбінацій моделей ієрархічної кластеризації

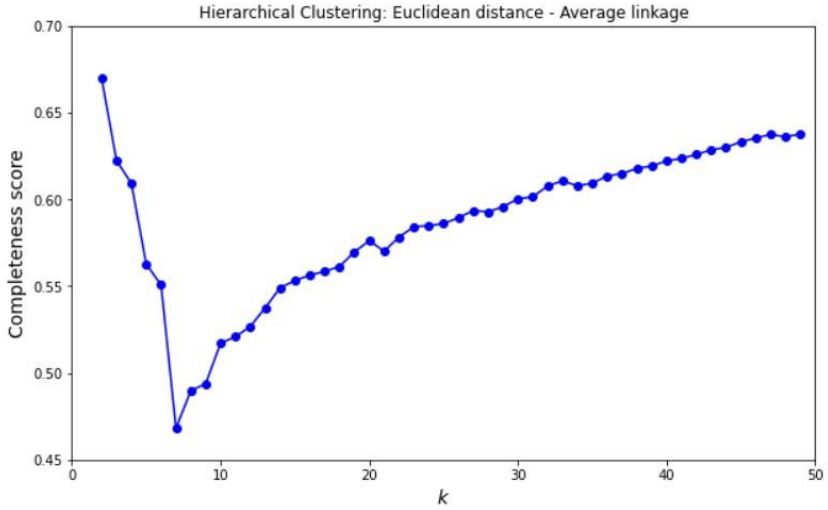
При k=24 моделі косинусоїдального середнього, середнього по міських кварталах та повного міського кварталу отримали найвищий бал – 0,61, за ними слідує модель окремого міського кварталу (0,60). Навпаки, оцінки однорідності для моделей з однією ланкою були переважно нижчими, ніж для моделей з середньою та повною ланкою. Однак оцінки зростали зі збільшенням k (рис. 4.29), і при k=24 найвищу оцінку мала модель Cityblock-complete (0,60), за нею йшли моделі Cosine-average (0,59) та Euclidean-complete (0,58).

	k	cosine-single	cosine-average	cosine-complete	euclidean-single	euclidean-average	euclidean-complete	cityblock-single	cityblock-average	cityblock-complete
0	2	0.010231	0.091853	0.055698	0.034243	0.034243	0.059888	0.034243	0.015501	0.066461
1	3	0.025674	0.145212	0.114170	0.044356	0.092902	0.092902	0.044356	0.091631	0.130819
2	4	0.035057	0.191606	0.157488	0.053680	0.108555	0.130876	0.055463	0.164212	0.176023
3	5	0.045171	0.209099	0.199839	0.093608	0.162940	0.148708	0.095391	0.189984	0.213099
4	6	0.083535	0.256978	0.236069	0.102679	0.208492	0.202222	0.104462	0.211777	0.255902
5	7	0.114834	0.300979	0.266376	0.113499	0.241010	0.239988	0.117064	0.237218	0.282055
6	8	0.124425	0.332491	0.299005	0.123264	0.270806	0.294221	0.126862	0.266824	0.305943
7	9	0.133221	0.352798	0.341293	0.190979	0.285930	0.330886	0.135867	0.317825	0.323706
8	10	0.153797	0.386735	0.365724	0.211322	0.330704	0.347103	0.153168	0.340473	0.350734
9	11	0.159543	0.402795	0.385971	0.223009	0.361698	0.374395	0.160697	0.369539	0.365080
10	12	0.176230	0.433654	0.413264	0.229777	0.392494	0.386305	0.178139	0.386226	0.394129
11	13	0.185638	0.453907	0.427872	0.238417	0.409772	0.423549	0.187695	0.419782	0.425317
12	14	0.226890	0.462594	0.450858	0.252020	0.434332	0.445247	0.198238	0.432850	0.435579
13	15	0.242790	0.477707	0.465243	0.262090	0.448838	0.466793	0.210318	0.451663	0.455831
14	16	0.258799	0.487833	0.477152	0.273582	0.459725	0.481226	0.278965	0.474649	0.481897
15	17	0.270400	0.499326	0.493839	0.282586	0.474367	0.495733	0.294625	0.491679	0.510628
16	18	0.277169	0.514451	0.508346	0.298335	0.485254	0.501896	0.303450	0.503172	0.520842
17	19	0.287152	0.526549	0.520861	0.331497	0.501941	0.520278	0.307622	0.529771	0.537529
18	20	0.296068	0.533870	0.531802	0.340083	0.515479	0.541976	0.316401	0.537092	0.554216
19	21	0.300239	0.549951	0.540145	0.361113	0.521642	0.552157	0.327730	0.555621	0.564342
20	22	0.310133	0.569235	0.553683	0.369440	0.547083	0.562283	0.333476	0.557196	0.574469
21	23	0.318958	0.574982	0.563897	0.373612	0.560621	0.572497	0.363897	0.572390	0.584649
22	24	0.327737	0.587584	0.574023	0.382906	0.567942	0.576116	0.371517	0.582173	0.601336
23	25	0.337492	0.593331	0.576620	0.404404	0.578882	0.590502	0.380867	0.597285	0.611550
24	26	0.347198	0.595928	0.588113	0.414984	0.587018	0.597823	0.388378	0.608778	0.621763
25	27	0.358691	0.604271	0.602498	0.425502	0.597144	0.606166	0.402627	0.618144	0.633256
26	28	0.367327	0.612615	0.606670	0.452844	0.601316	0.610338	0.417197	0.626487	0.643383
27	29	0.373073	0.627000	0.616850	0.499069	0.608845	0.622940	0.427777	0.636668	0.647554
28	30	0.381417	0.632746	0.625194	0.505965	0.620338	0.633067	0.436720	0.644197	0.654875
29	31	0.395802	0.641673	0.633537	0.532328	0.629704	0.644560	0.447329	0.651726	0.665816

Рисунок 4.29 – Оцінки однорідності для різних комбінацій моделей ієрархічної кластеризації

Одноланкова метрика[17] добре обробляє нееліптичні форми, але чутлива до шуму та викидів. З іншого боку, повна метрика[17] менш чутлива до шуму і викидів, але може розбивати великі кластери і надає перевагу кулястим формам (а середня метрика[17] є проміжним підходом між цими двома методами) (рис. 4.30 – 4.35).





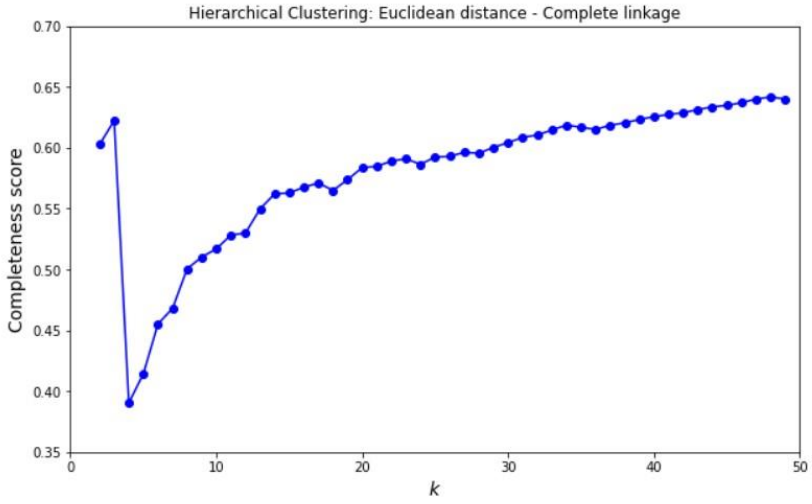
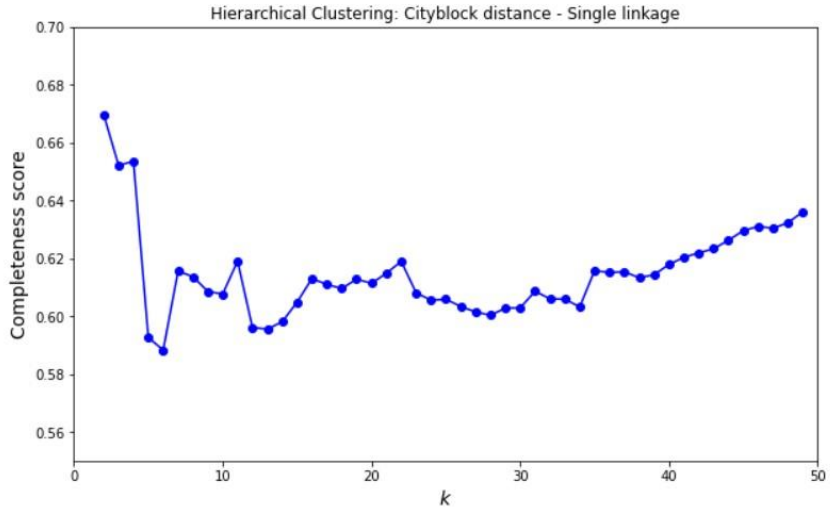
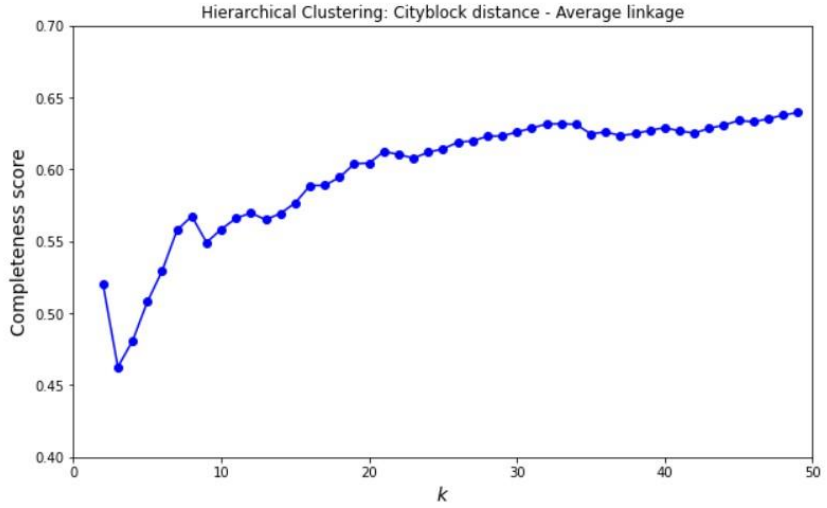


Рисунок 4.30 – Оцінки повноти: Ієрархічні евклідові моделі





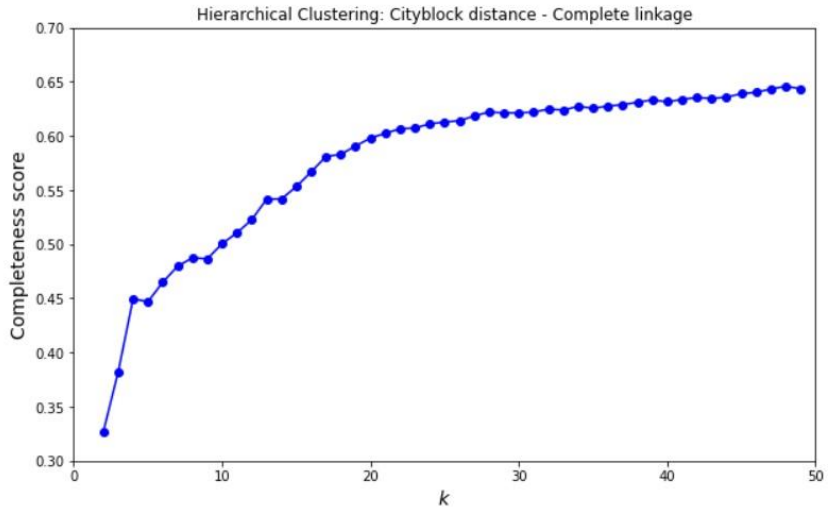


Рисунок 4.31 – Оцінки повноти: Ієрархічні моделі міських кварталів

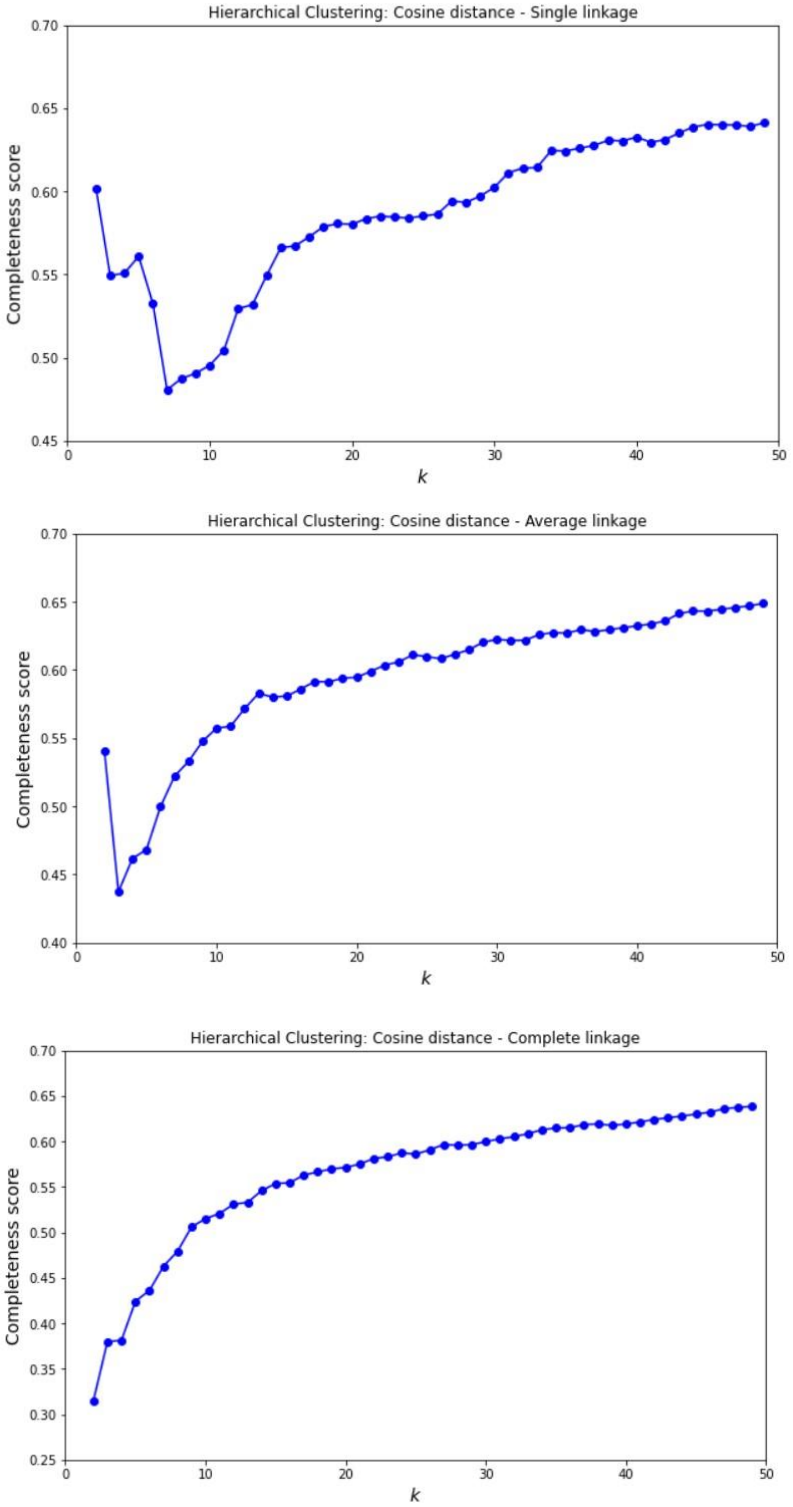
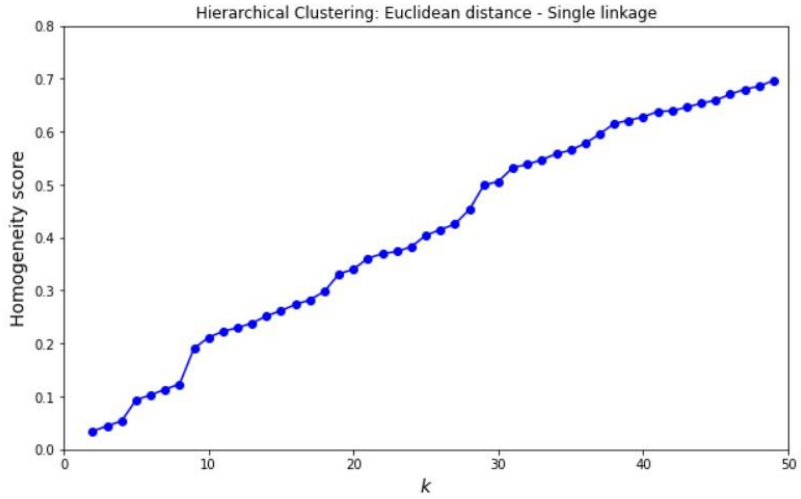
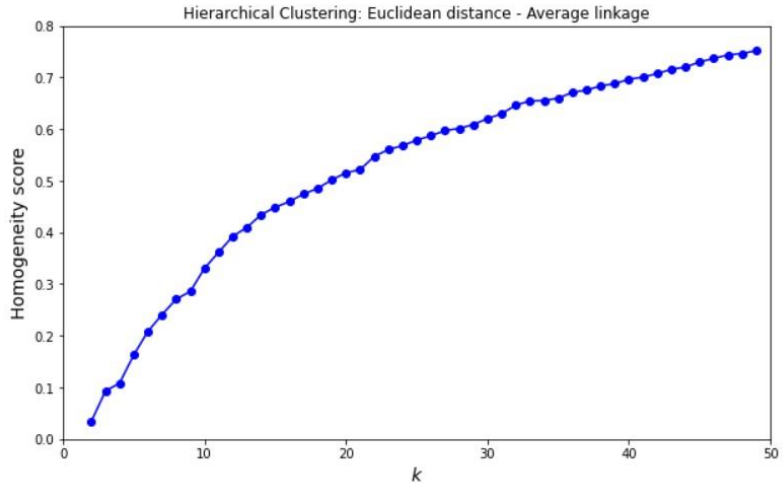


Рисунок 4.32 – Оцінки повноти: Ієрархічні косинусні моделі





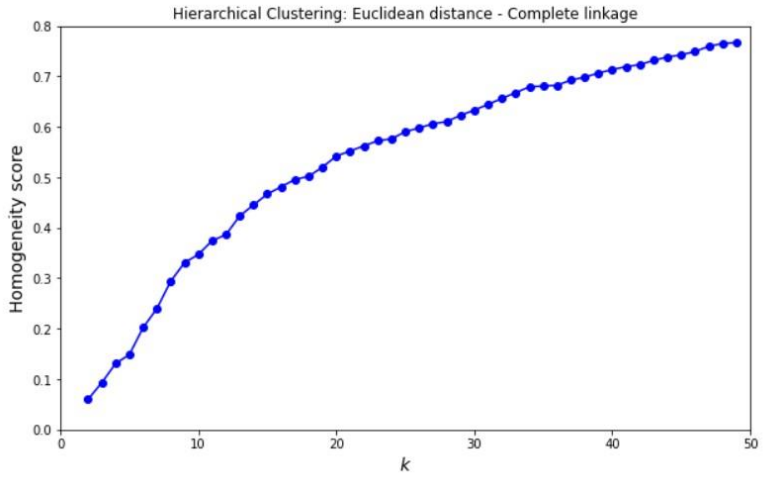


Рисунок 4.33 – Показники однорідності: Ієрархічні евклідові моделі

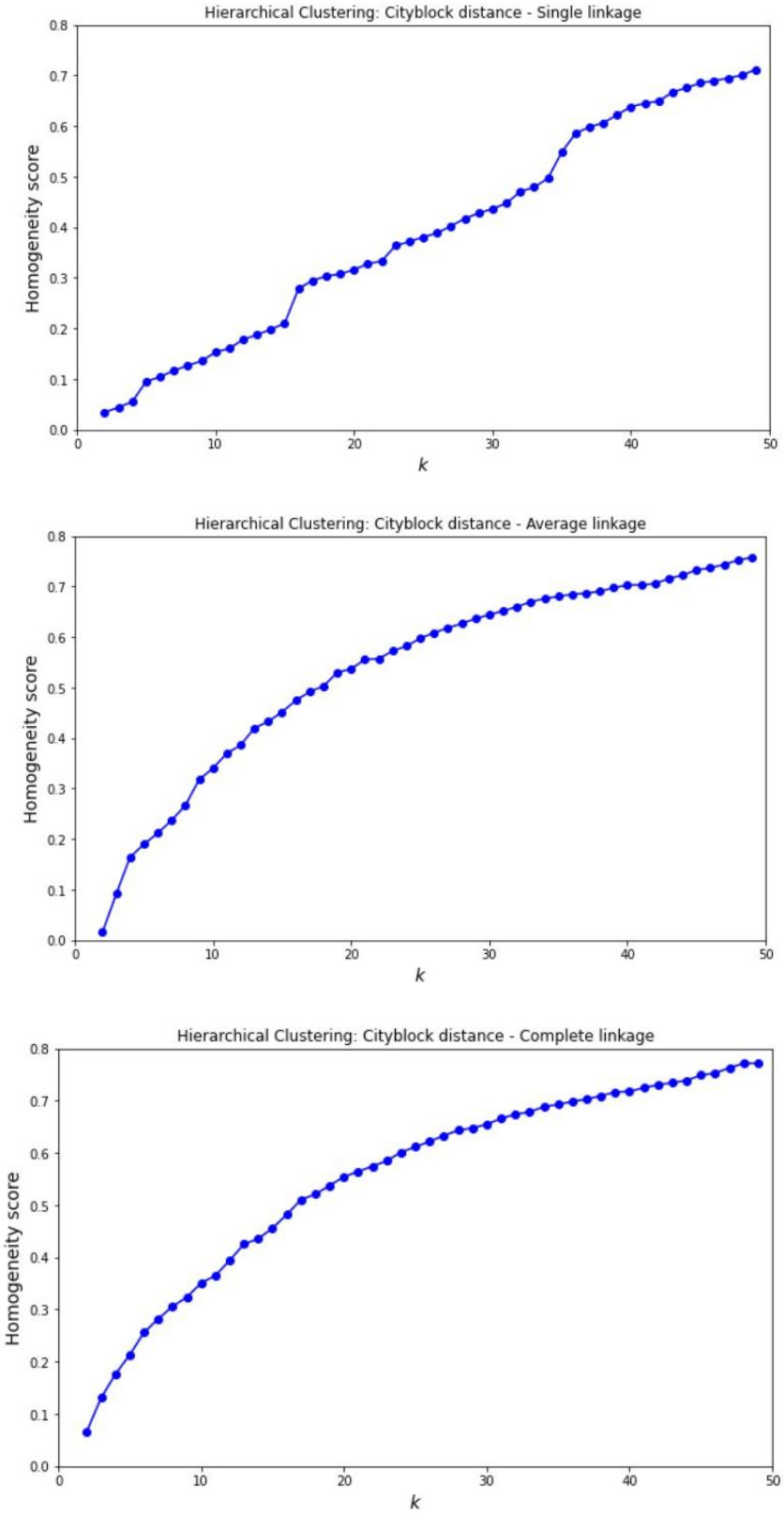


Рисунок 4.34 – Оцінки однорідності для ієрархічних моделей міських кварталів

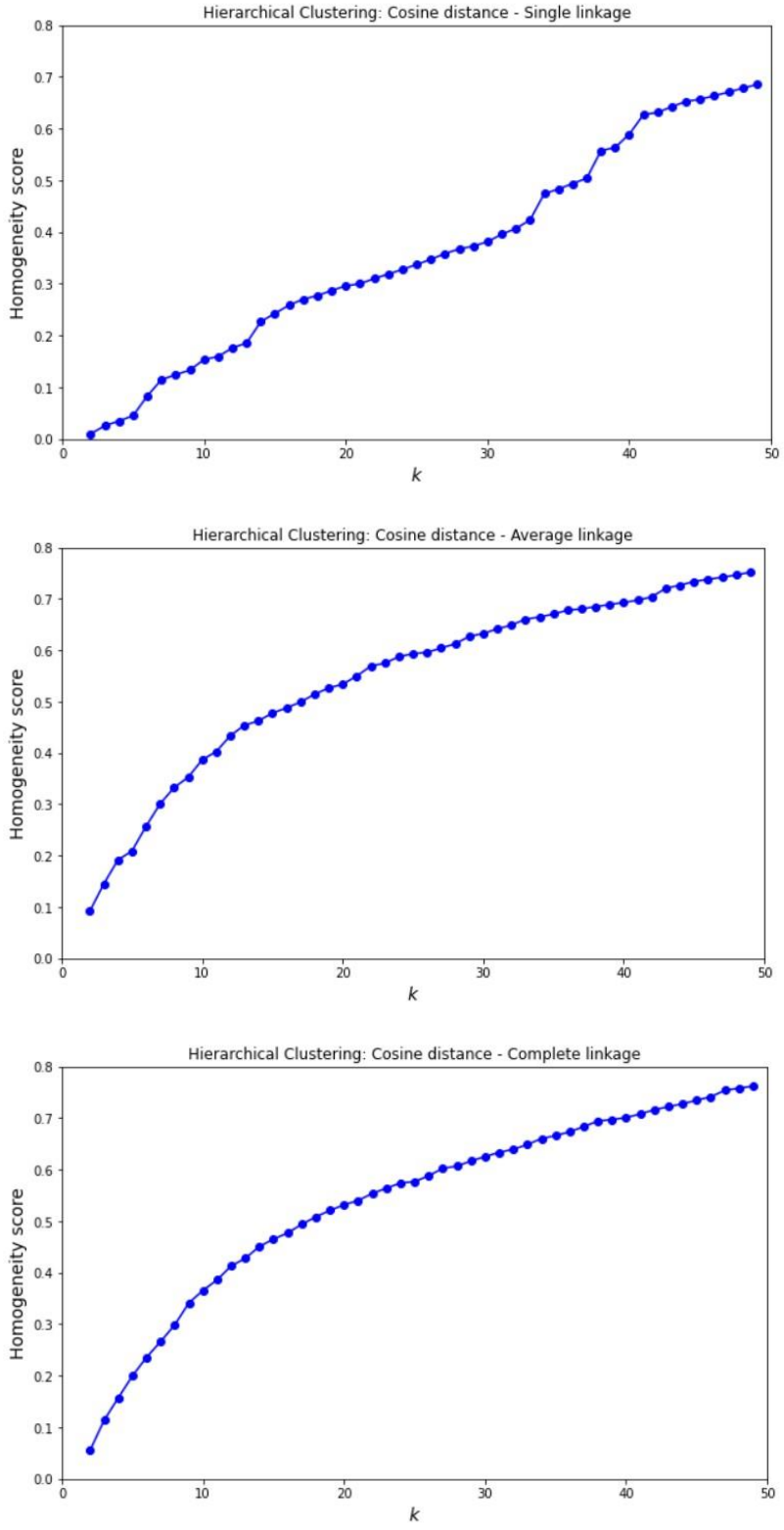


Рисунок 4.35 – Показники однорідності: Ієрархічні косинусні моделі

Цей набір даних містить розріджені дані з поєднанням кулястих (globular) (в середині) і некулястих форм (вгорі), а також деякі виключення (outliers) (рис. 4.36).

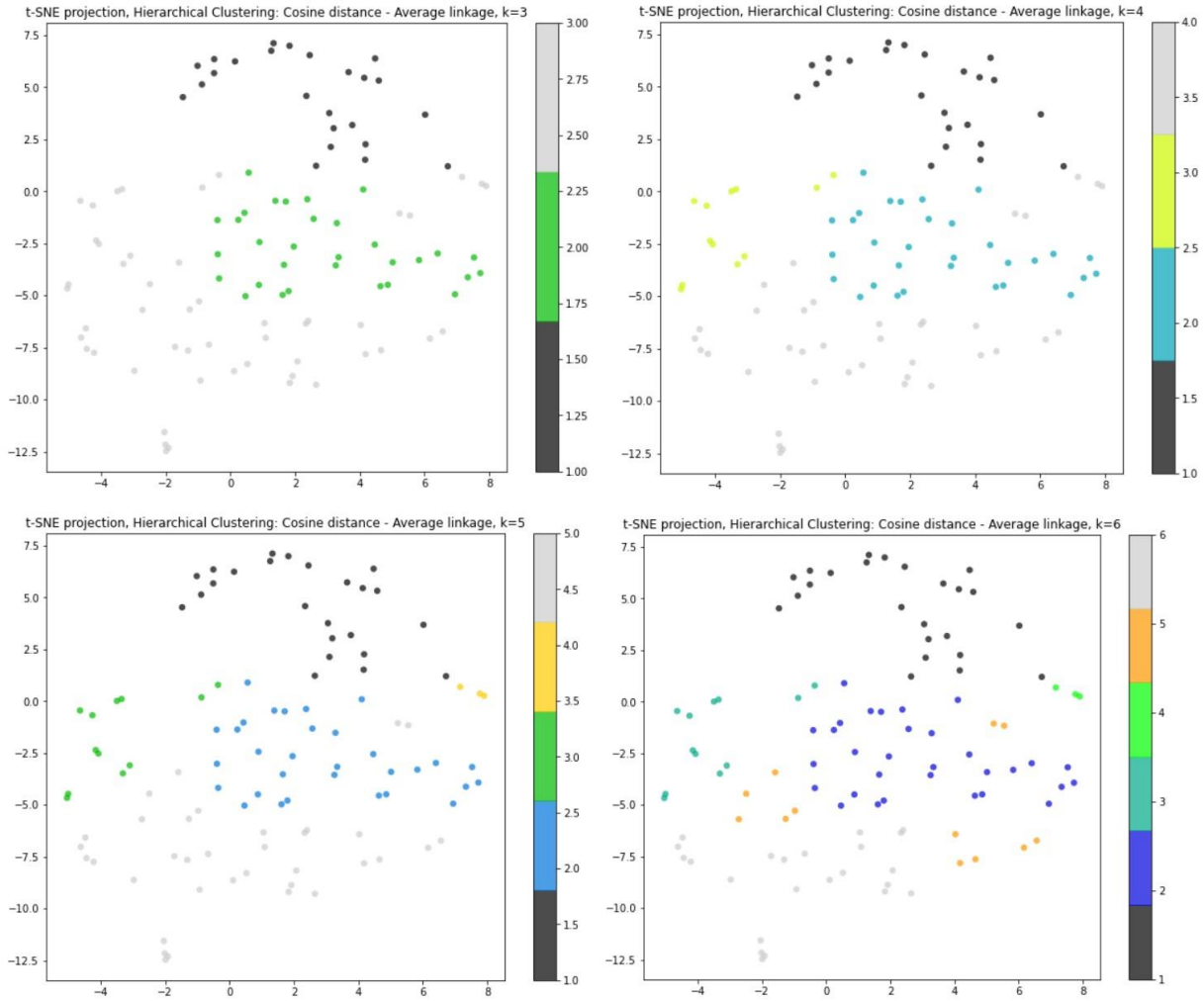


Рисунок 4.36 – Прогнози t-SNE: Косинусоїдальне середнє при збільшенні k від 3 до 6

Таким чином, середній зв'язок виявився кращим для цього набору даних, оскільки він досить добре справлявся з різними формами кластерів і викидами. При менших k-середній та повний зв'язок показали більш збалансовані кластери, ніж одиночний зв'язок (рис. 4.37).

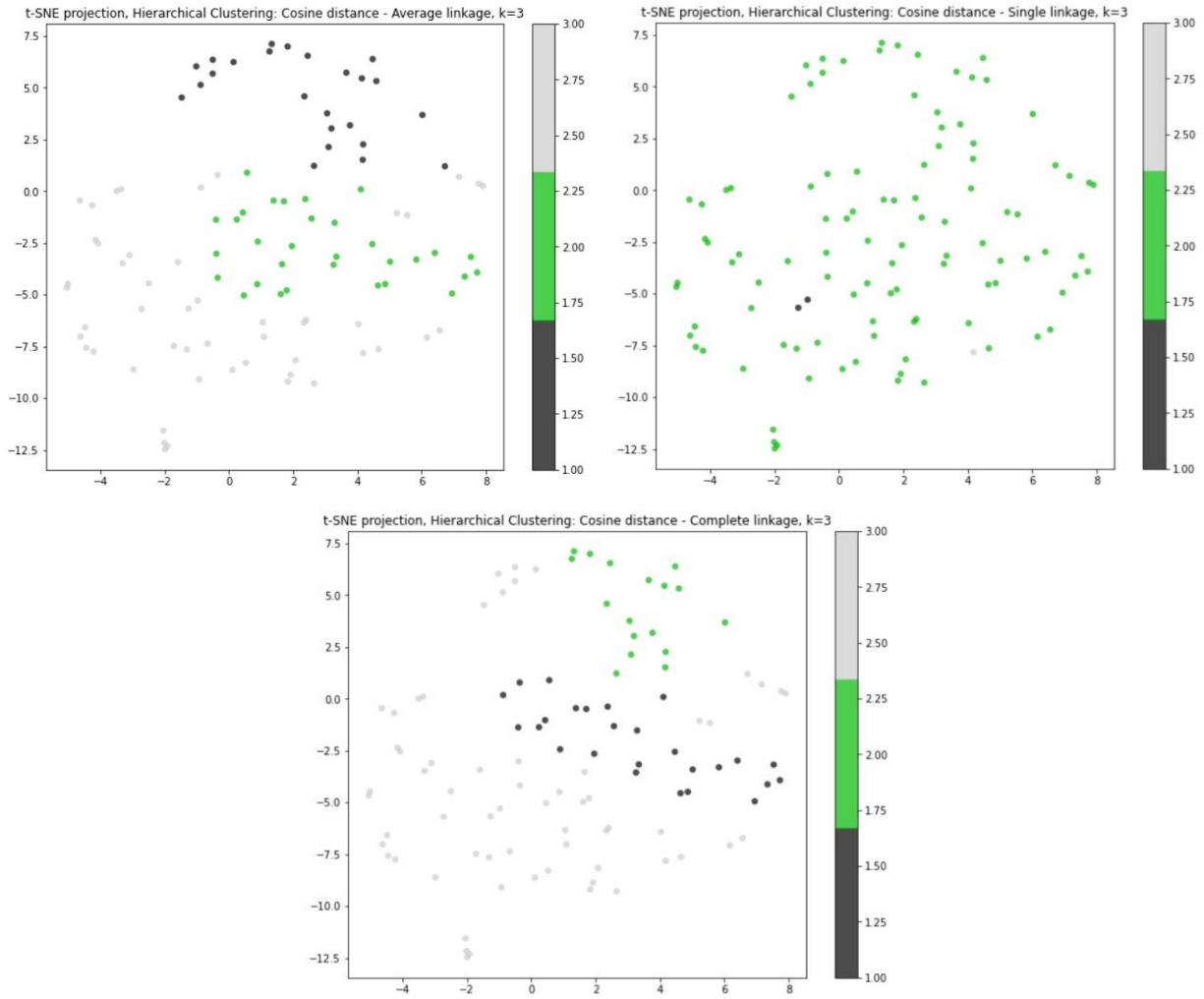


Рисунок 4.37 – Прогнози t-SNE (k=3): Косинус одиничний проти косинуса середнього проти косинуса повного

При k=24 всі моделі дали досить добре розділені, але різні кластери (рис. 4.38).

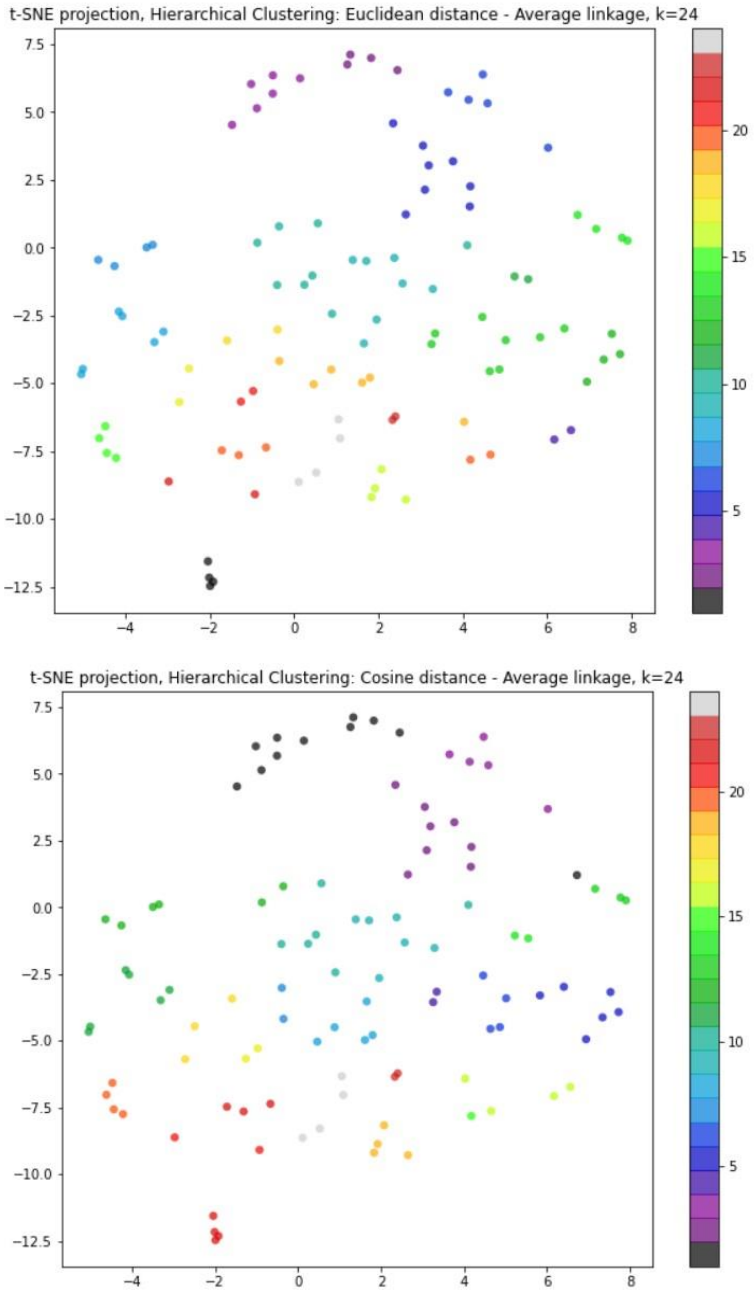


Рисунок 4.38 – Прогнози t-SNE (k=24): Евклідове середнє (вгорі) проти косинусного середнього (внизу)

На дендограмі[18] середній та повний зв'язок також показали кращі групування, ніж однозв'язковий, оскільки більше точок з одного сеансу, таких як Виділення ознак (Feature Extracture) та Відбір (Selection), були згруповані разом (рис. 4.39-4.41).

Рисунок 4.39 – Дендограма: Ієрархічна косинусоїдальна кластеризація зв'язків

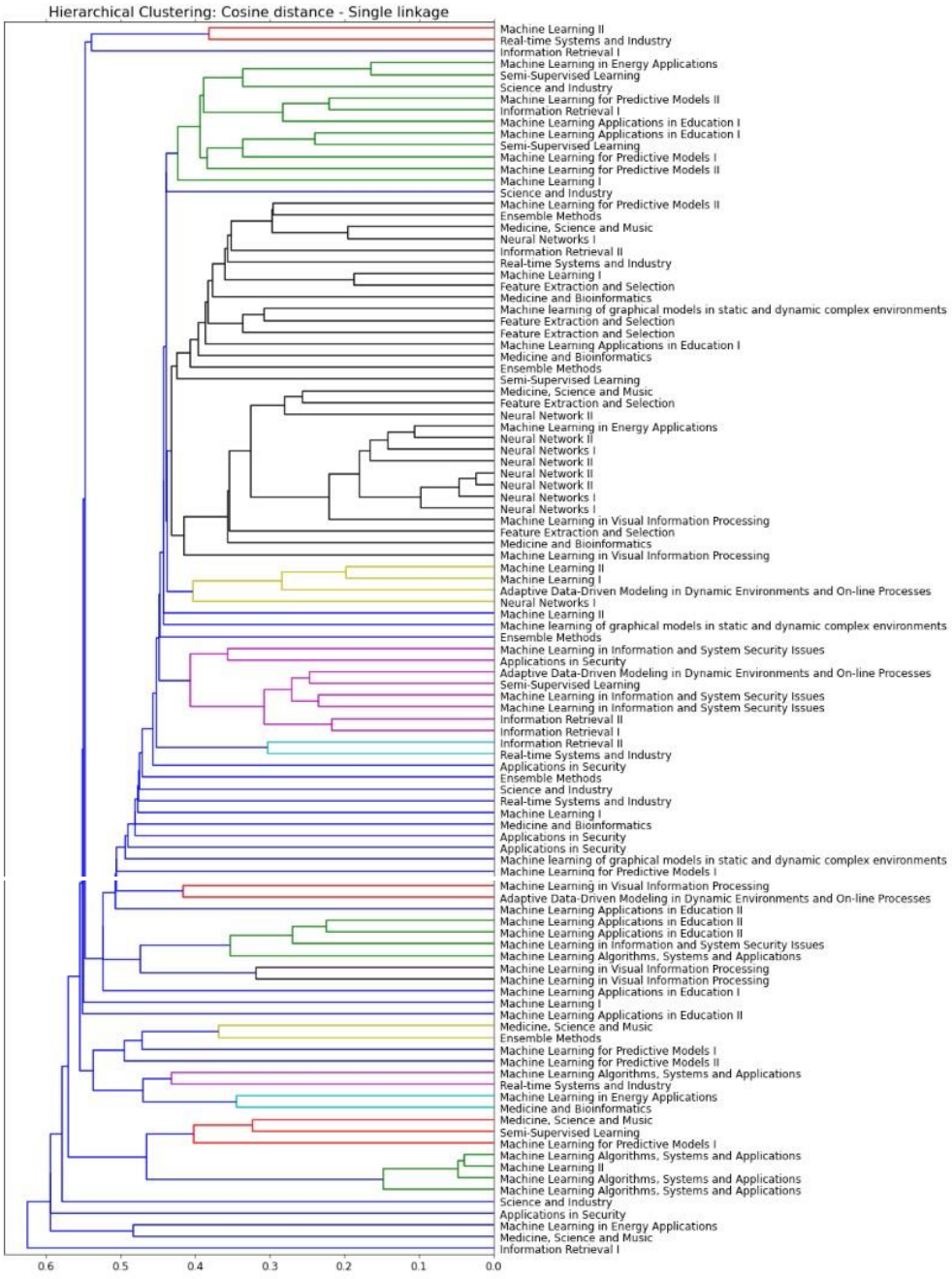


Рисунок 4.40 – Дендограма: Ієрархічна косинусоїдальна одноланкова кластеризація

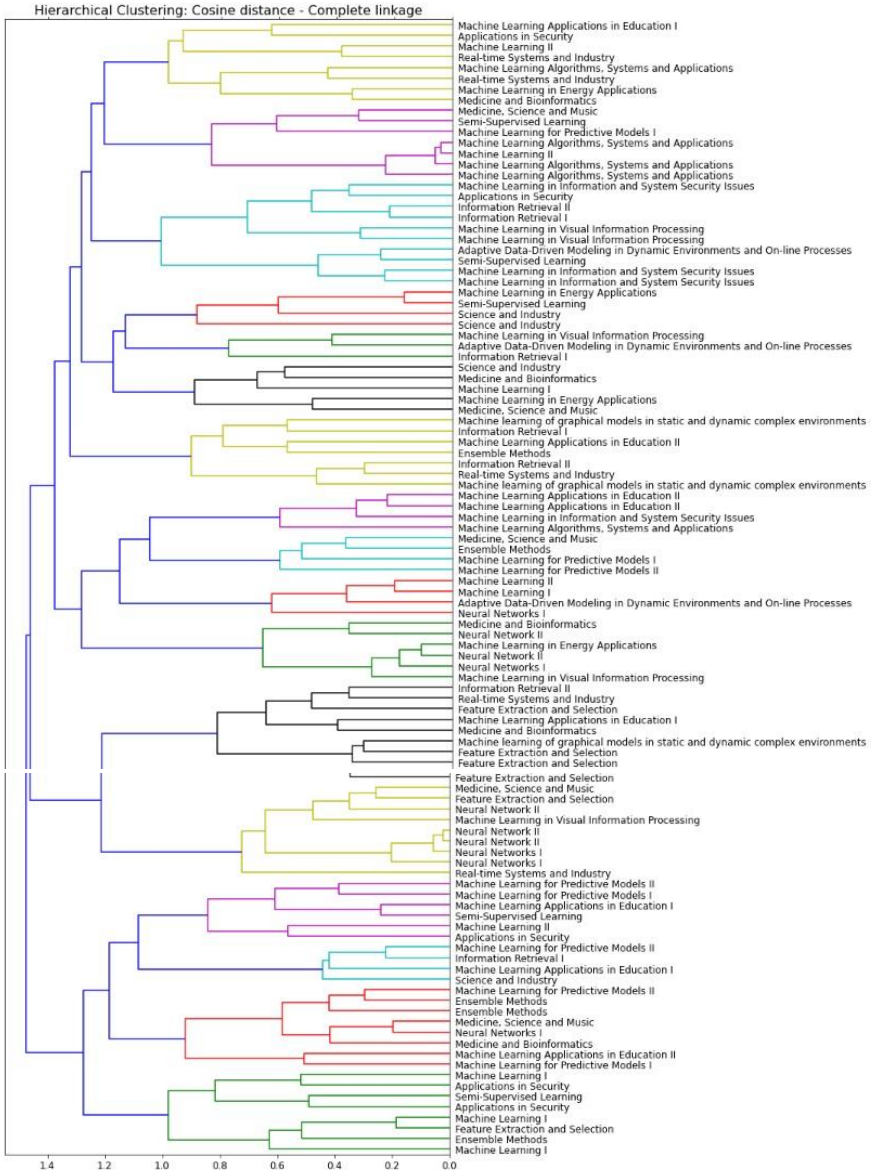


Рисунок 4.41 – Дендограма: Ієрархічна косинусоїдальна кластеризація з повними зв'язками

З трьох мір схожості косинус показав кращі результати, ніж евклідова або мангеттен (рис. 4.42-4.43).

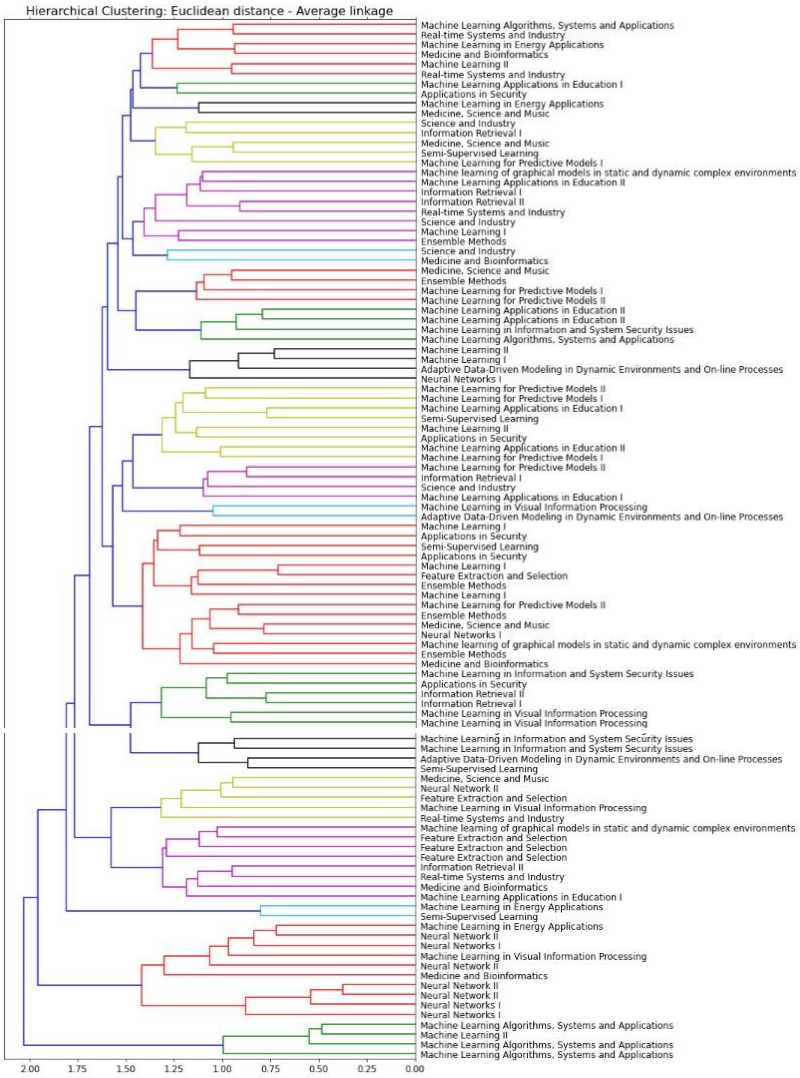


Рисунок 4.42 – Дендограма: Ієрархічна евклідова кластеризація з середньою ланкою

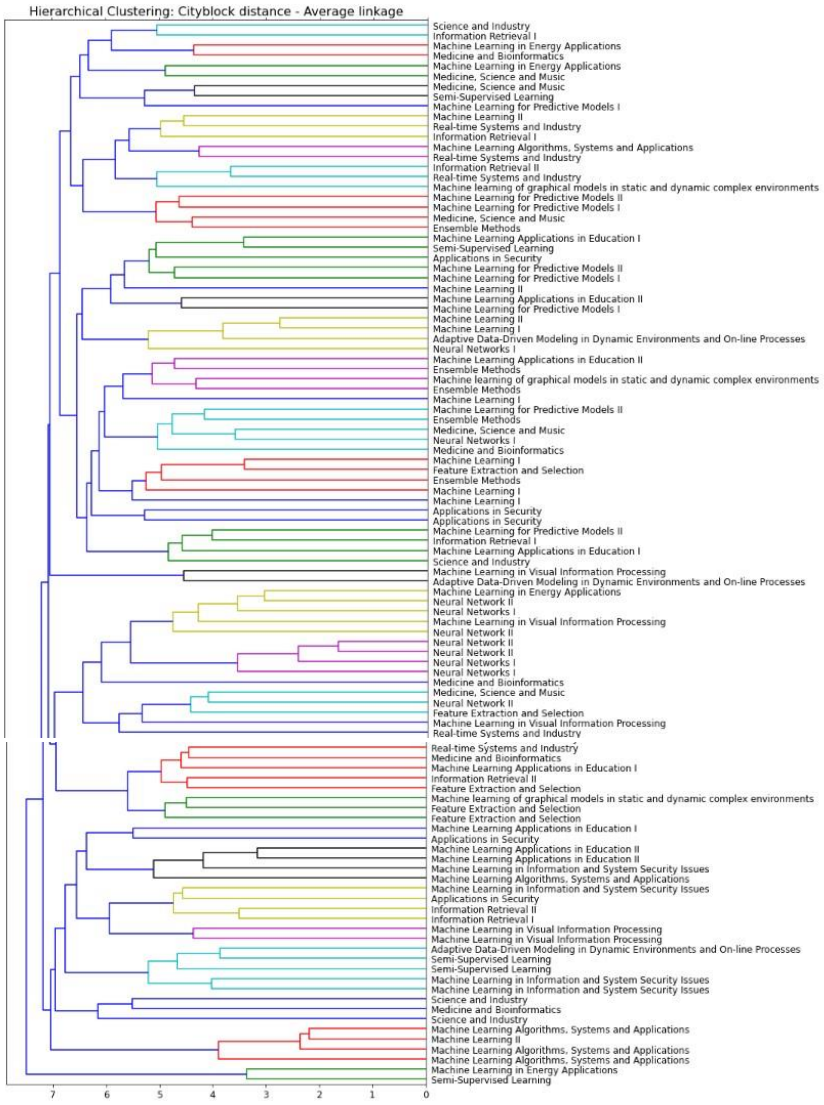


Рисунок 4.43 – Дендограма: Ієрархічна кластеризація “Мангеттен – середня ланка

Замість того, щоб вимірювати фактичну відстань, косинус вимірює кут між точками, і для наших нечисленних даних він працює досить добре. Відповідно, для цього алгоритму було обрано косинусоїдальне середнє з k=24 як оптимальну модель. Вона має кращі показники повноти (0,61) та однорідності (0,59) порівняно з більшістю інших моделей, а при k=24 дає досить добре відокремлені кластери на графіку t-SNE та дендограмі.

4.4 Кластеризація DBSCAN

4.4.1 Набір даних: Відгуки про подорожі

За допомогою алгоритму sklearn DBSCAN[19] на цьому наборі даних було навчено понад 100 моделей з використанням трьох мір відстані (евклідова, мангеттен

та косинус) та різних значень епсилонів (0,01-5) і мінімальних вибірок (2-10). Епсилон задає максимальний радіус околиці, тоді як мінімальні вибірки задають мінімальну кількість точок в околиці для того, щоб точка вважалася основною точкою[3]. Без золотих кластерів отримані моделі оцінювалися лише за внутрішніми показниками: оцінка Девіса-Болдіна та оцінка Калінкі-Харабаша.

Найвищий бал Калінкі-Харабаша (62) отримала евклідова модель (з eps=0,22 та min вибірок=7). Вона дозволила виділити п'ять кластерів. Найнижчий показник Девіса-Болдіна (0,40), отриманий за допомогою евклідової моделі (eps=0,63 та min вибірок=4), виявив лише два кластери. Інші комбінації (Cityblock, eps=0.40, min samples=8 та Cosine, eps=0.01, min samples=8) виявили вісім; Cosine, eps=0.40, min samples=8 – шість; Euclidean, eps=0.20, min samples=5 – 13; Euclidean, eps=0.20, min вибірок=8 – десять, а косинусний, eps=0.01, min вибірок=6 – 11 кластерів) змогли виявити більше кластерів, але з нижчими показниками Калінського або вищими показниками Девіса (рис. 4.44-4.49).

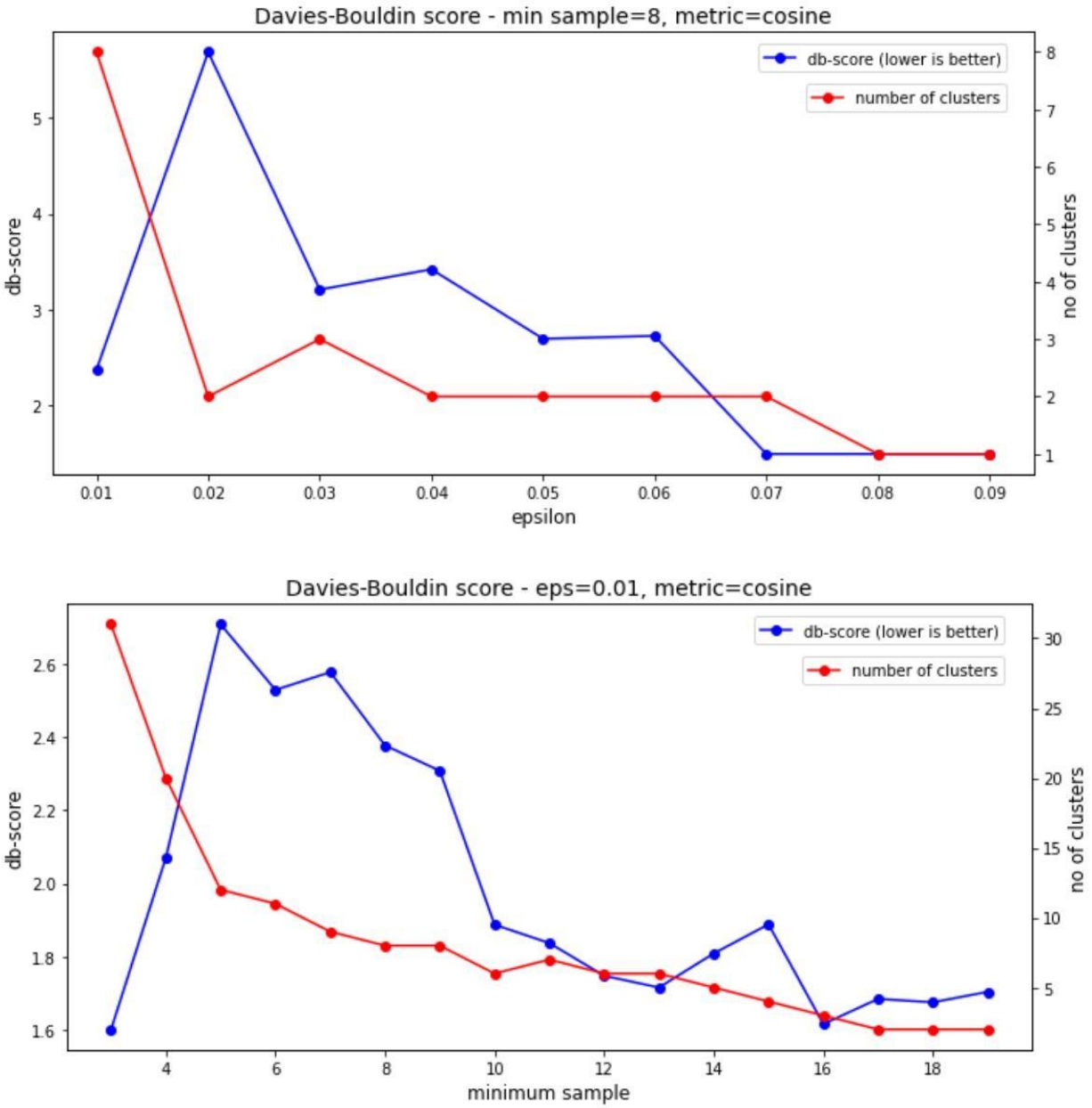


Рисунок 4.44 – Оцінки Девіса-Болдіна: Косинусна відстань

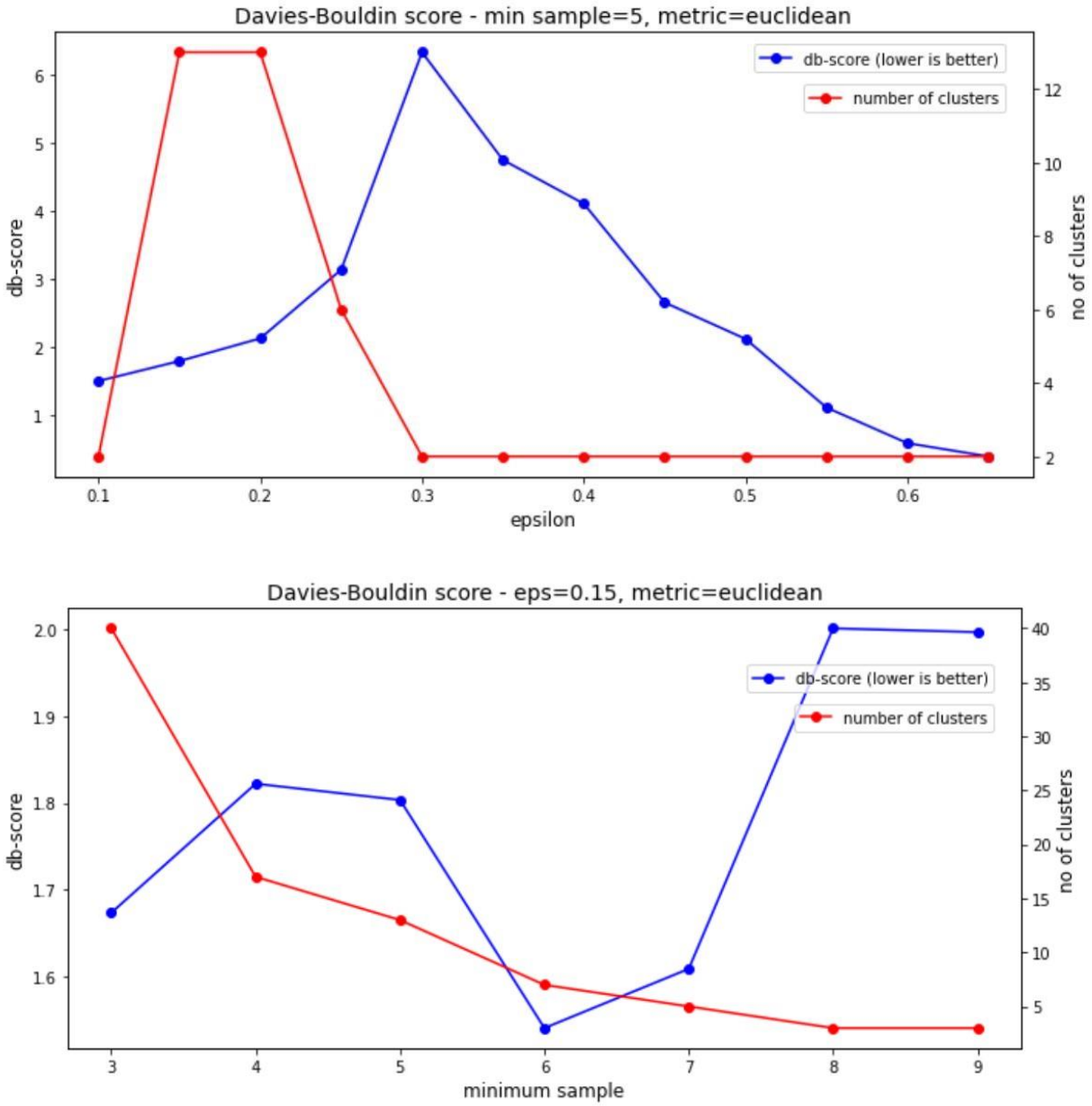


Рисунок 4.45 – Оцінки Девіса-Болдіна: Евклідова відстань

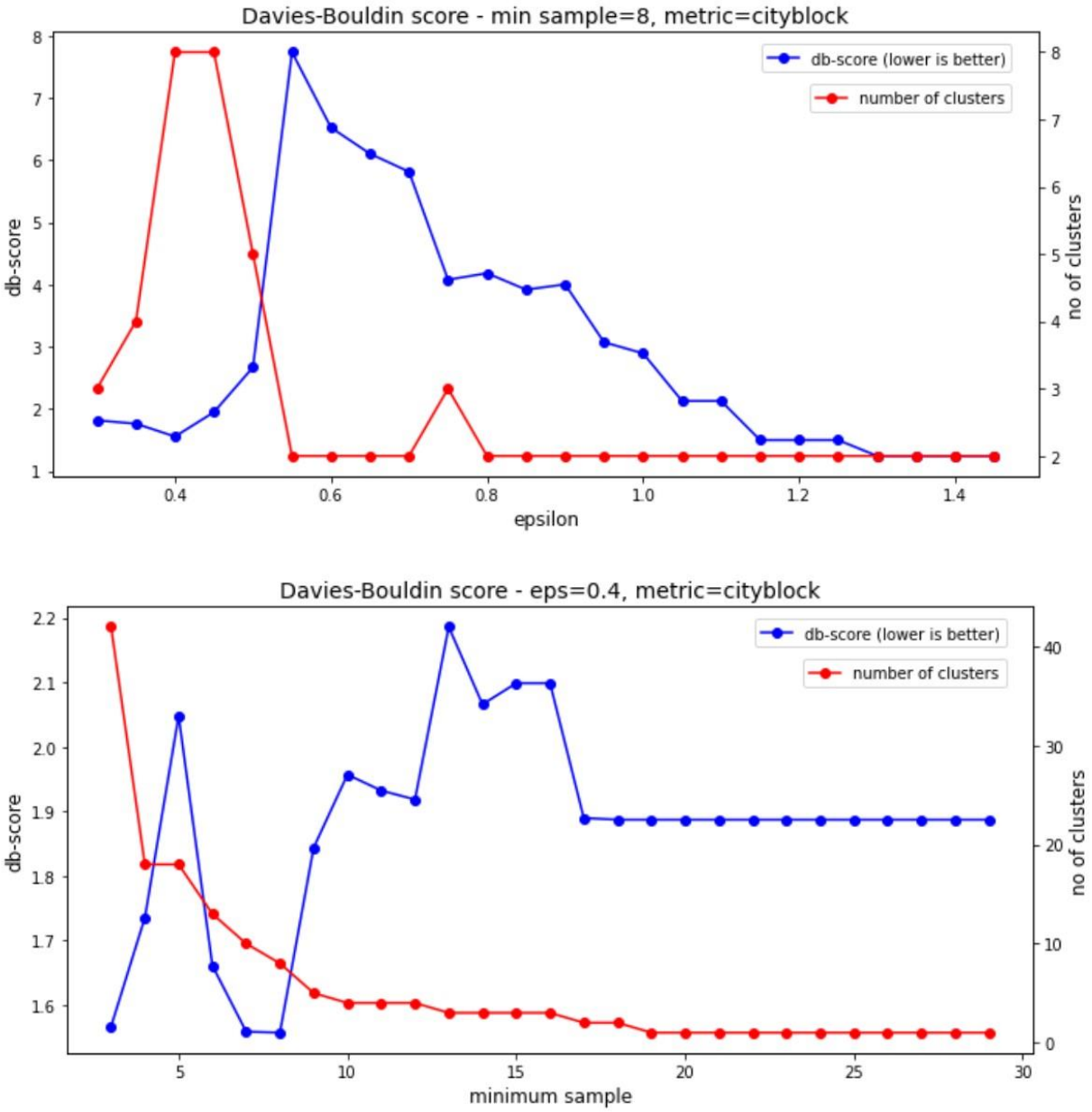


Рисунок 4.46 – Оцінка за шкалою Девіса-Болдіна: Відстань між кварталами міста

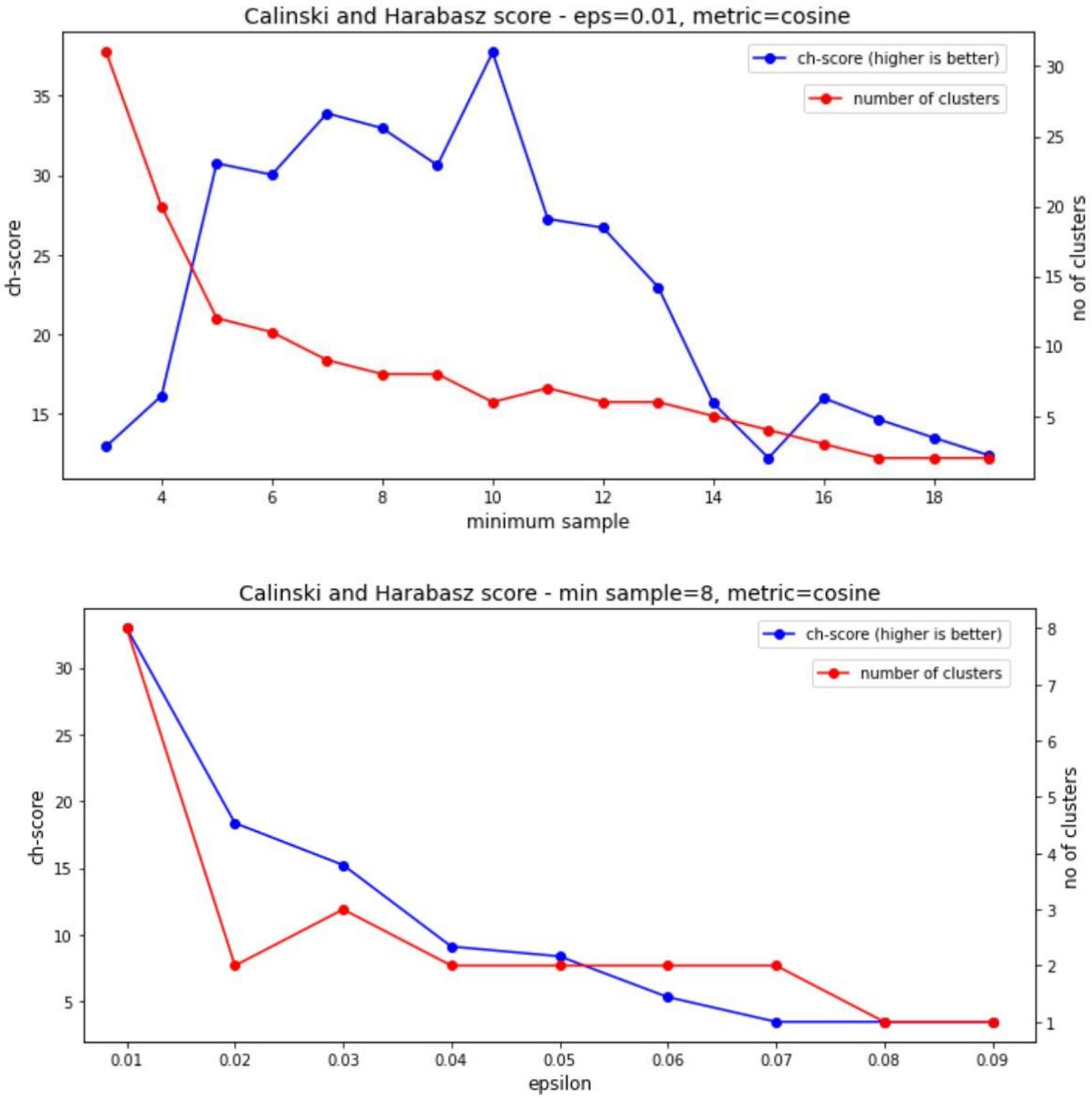


Рисунок 4.47– Оцінки Калінського-Харабаша: Косинусна відстань

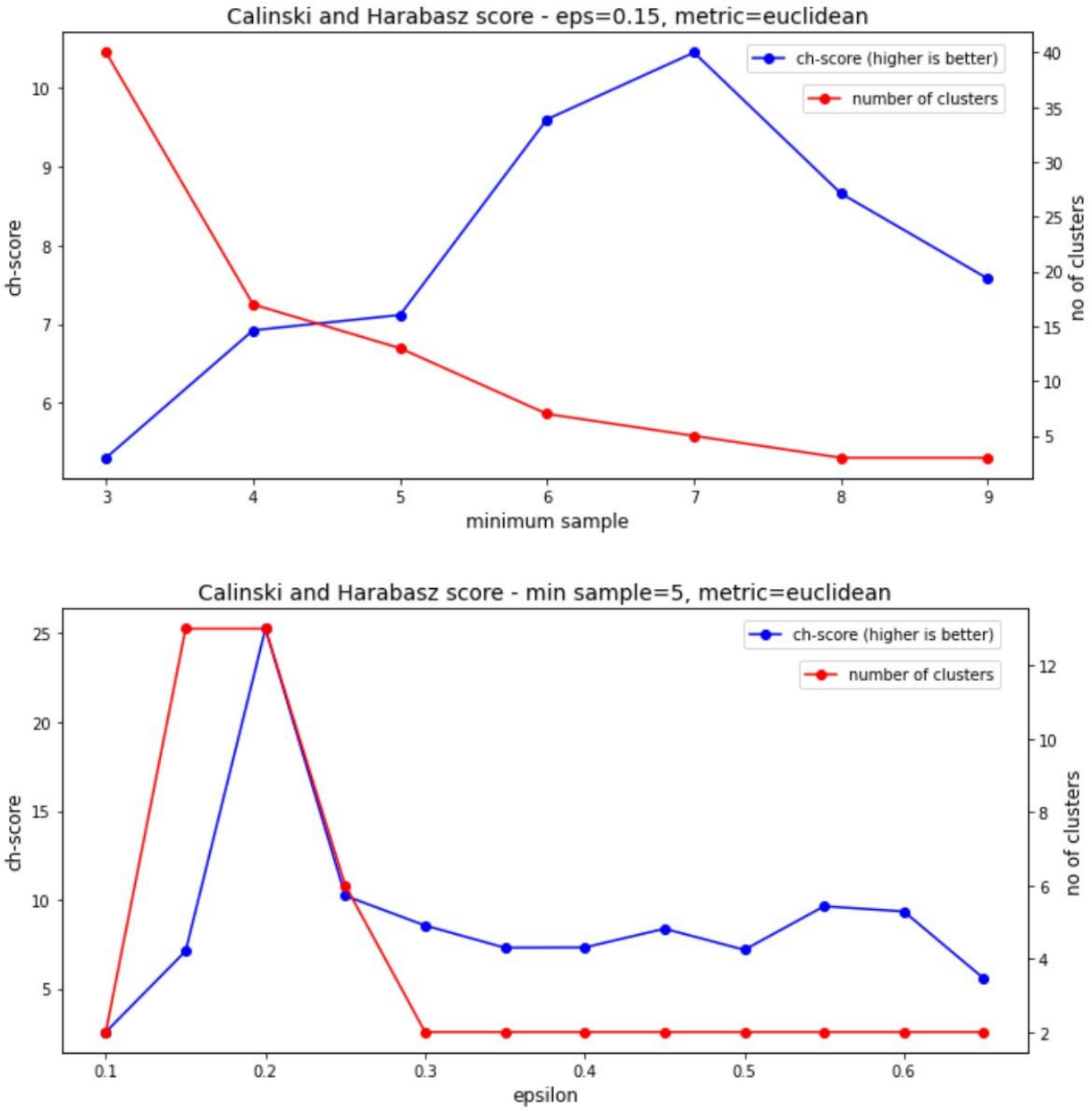


Рисунок 4.48 – Оцінки Калінського-Харабаша: Евклідова відстань

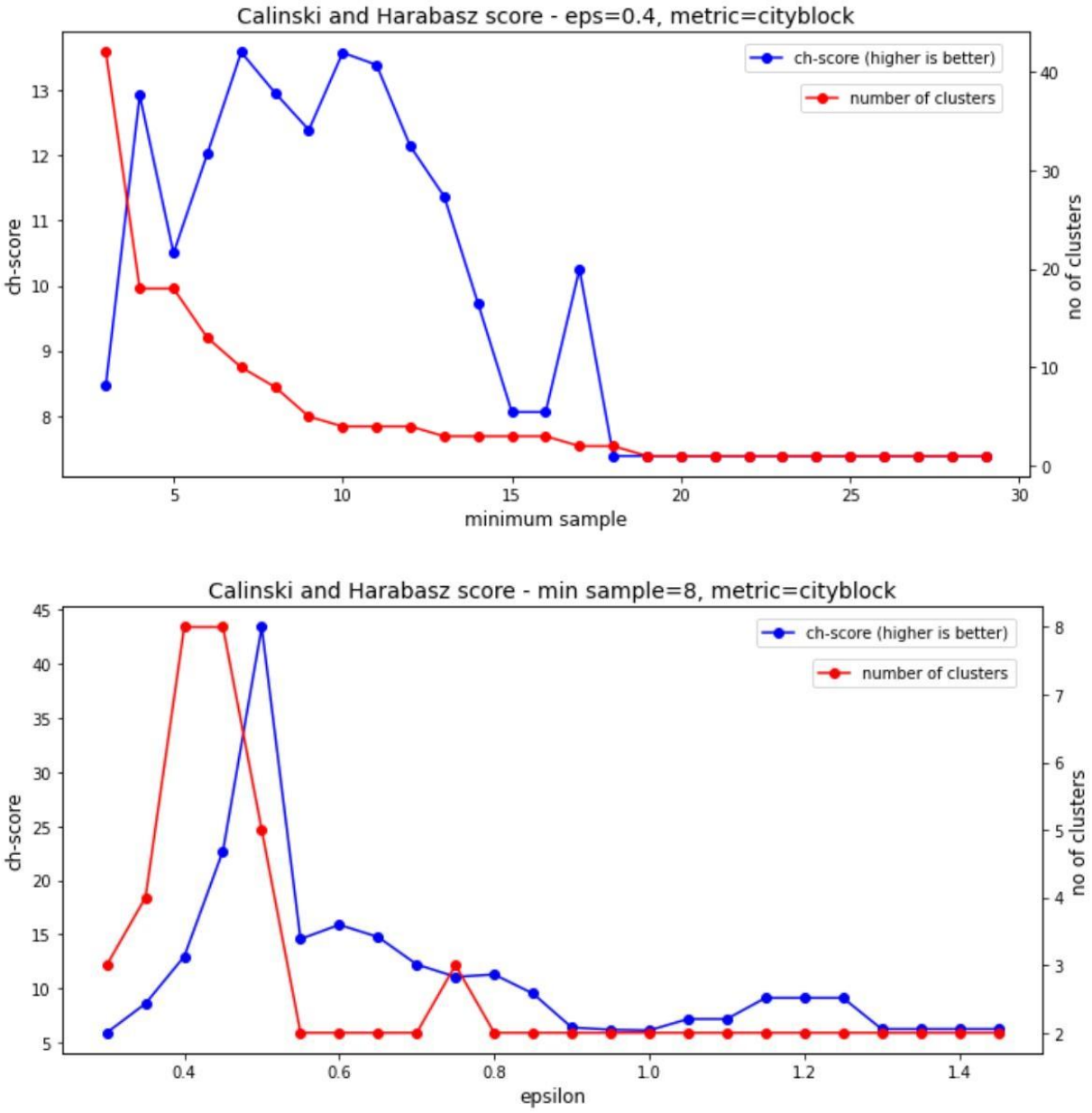


Рисунок 4.49 – Оцінка Калинського-Харабаша: Відстань між кварталами міста

На проекціях t-SNE жодна з моделей не дала хорошої кластеризації, один кластер був розкиданий навколо інших кластерів (рис. 4.50).

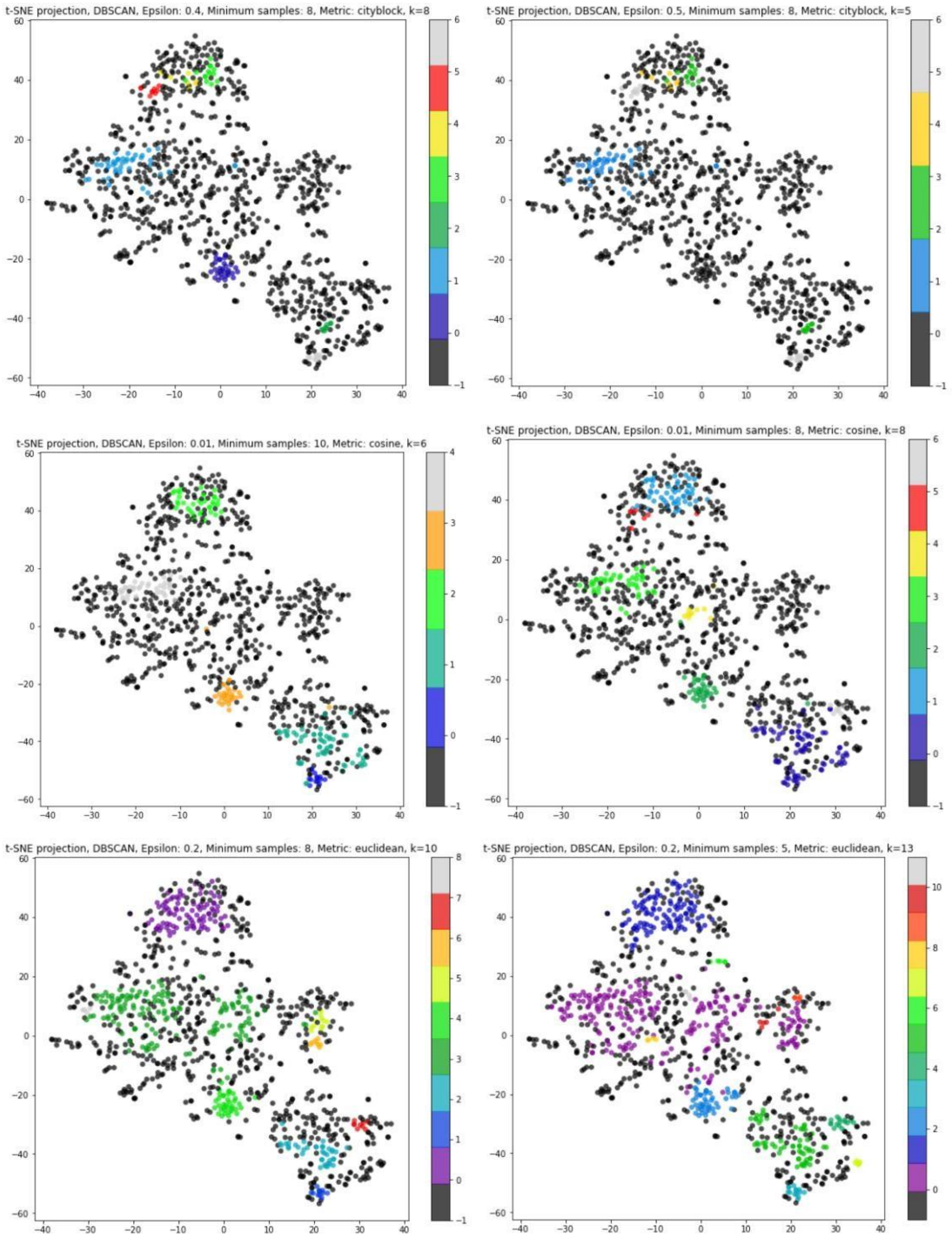


Рисунок 4.50 – Проекції t-SNE: Сітіблок (вгорі), косинусна (посередині) та евклідова (внизу) з різними комбінаціями епсилонів та мінімальних вибірок

Навіть оптимальна модель мала цю проблему (рис. 4.51).

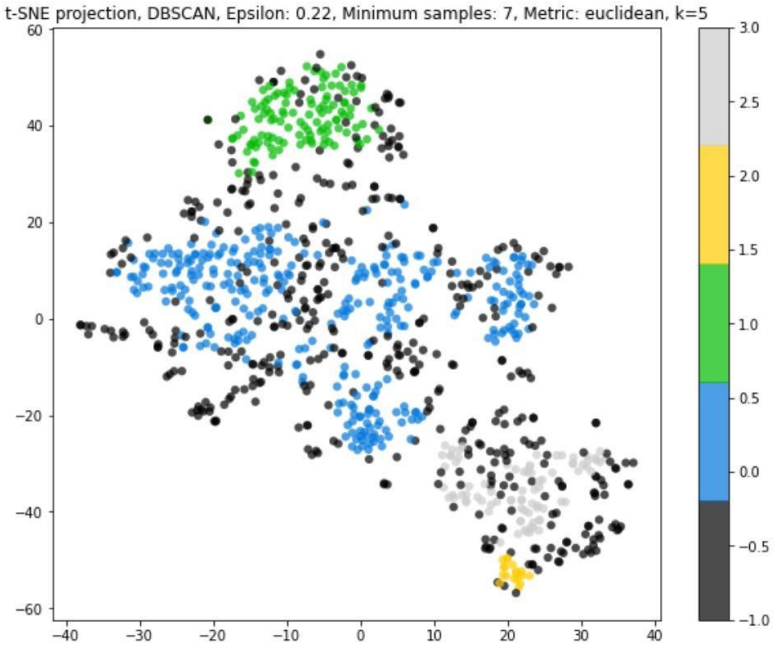


Рисунок 4.51 – Проекція t-SNE: Евклідова з eps=0.22 та min samples=7

Алгоритм DBSCAN добре працює з кластерами різної форми та розміру і є стійким до шуму. На жаль, він погано працює з даними різної щільності, тому на цьому наборі даних він не працював добре. З різних комбінацій було обрано евклідову (eps=0.22 та min samples=7) як оптимальну модель. Вона має низький показник Девіса (3,63), найвищий показник Калінського (62) і показує п'ять досить добре, хоча і не ідеально розділених кластерів на графіку.

4.4.2 Набір даних: Прийняті статті ICMLA 2014

Понад 100 моделей було навчено на цьому наборі даних з використанням тих самих трьох мір відстані та різних значень епсилон (0,01-5) і мінімальних вибірок (2-15). Маючи золоті кластери, отримані моделі були оцінені за допомогою двох зовнішніх мір: оцінки повноти та однорідності.

Найвищу оцінку однорідності (0,48) отримала модель Cityblock (з eps=4,63 та min вибірок=2), яка змогла виявити 20 кластерів. Найвищу оцінку повноти (0,75) отримала модель Cosine (eps=0,33 та min вибірок=5), але вона виявила лише два кластери. Інші комбінації (мангеттен, eps=4,60, min вибірок=2, виявлено 20 кластерів; евклідова, eps=1, min вибірок=2, 20 кластерів; і косинусна, eps=0,42, min вибірок=2, 15 кластерів) виявили більше кластерів, але з нижчими показниками повноти або однорідності (рис. 4.52-4.57).

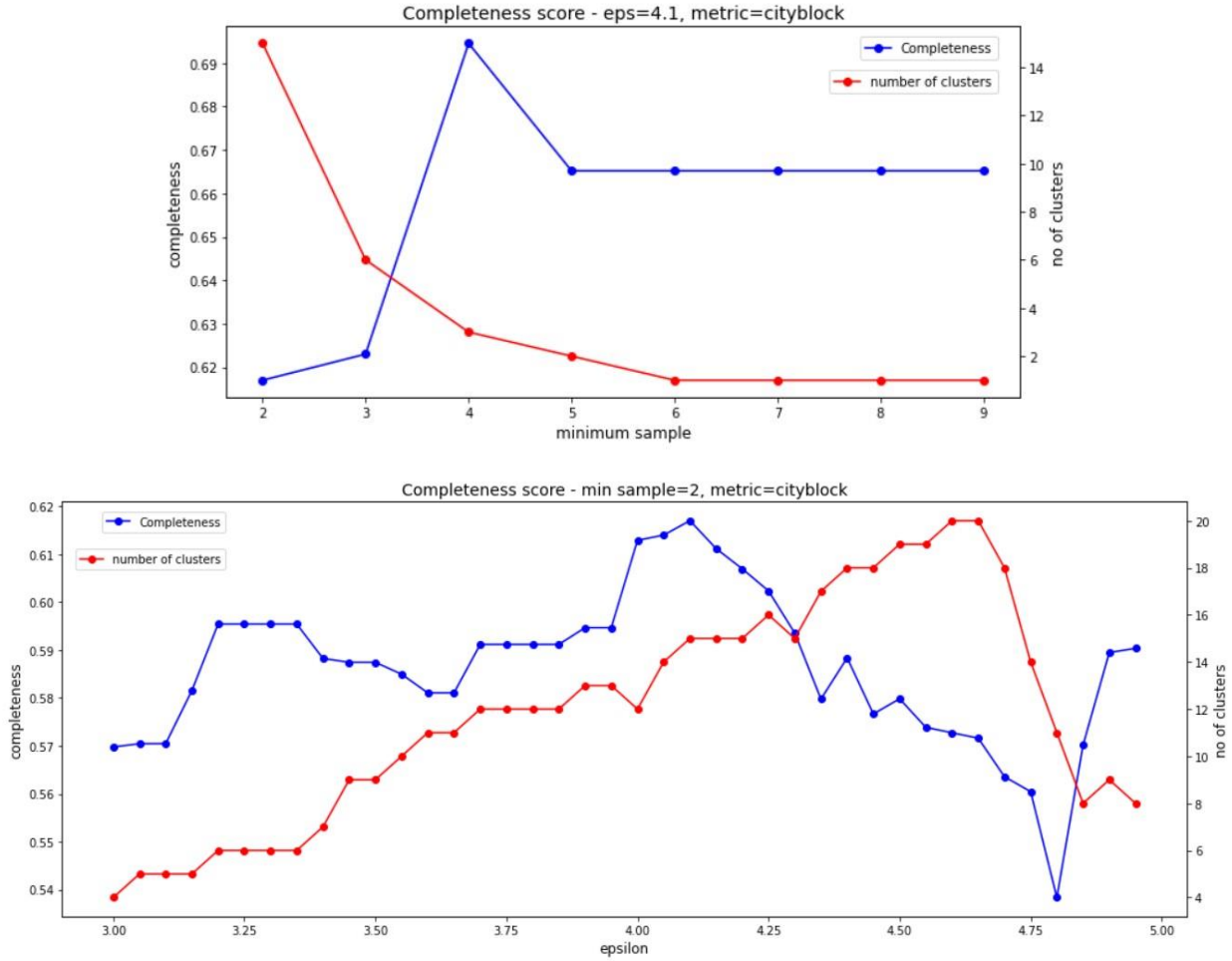


Рисунок 4.52 – Оцінки повноти: Моделі міських кварталів DBSCAN

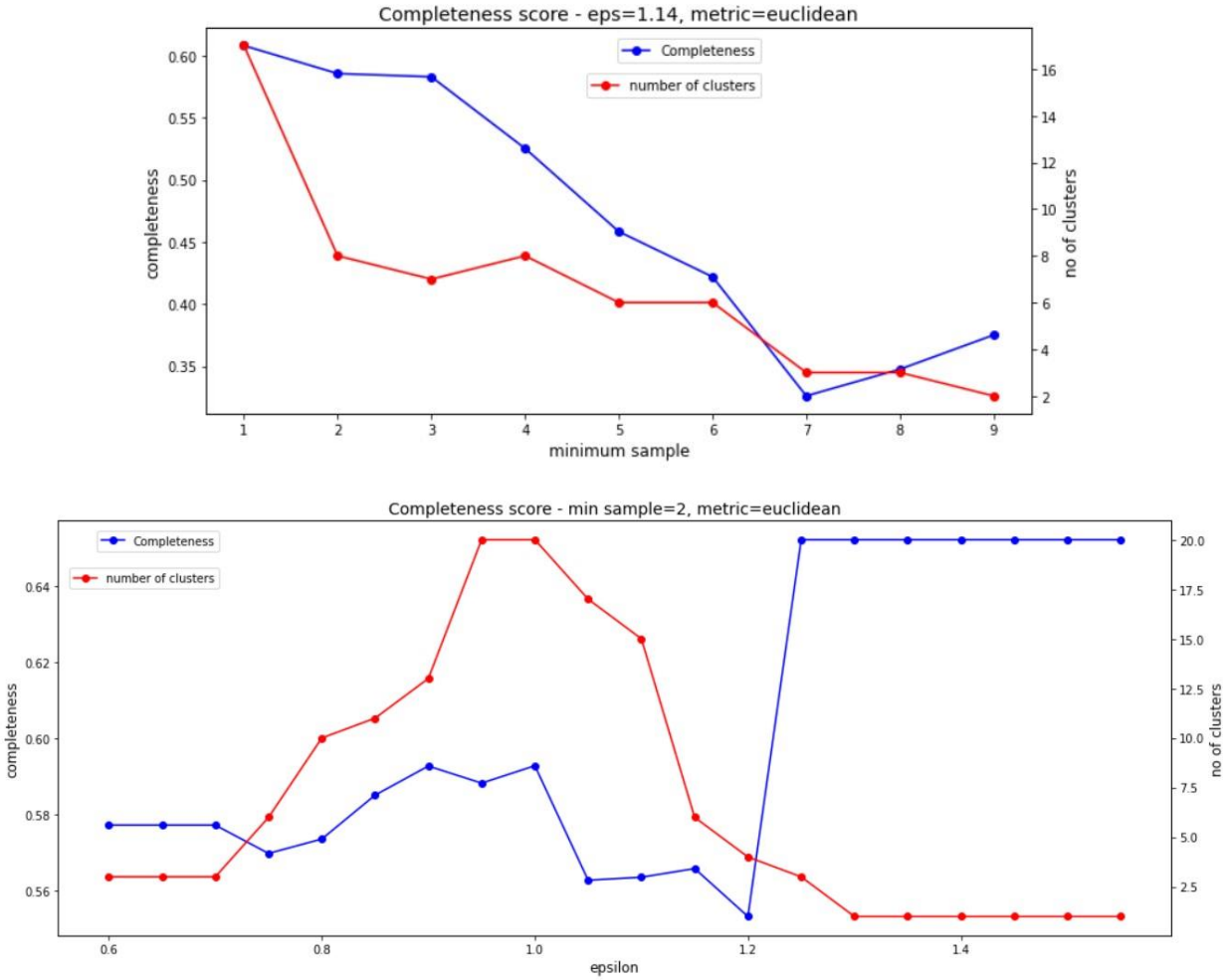


Рисунок 4.53 – Оцінки повноти: Евклідові моделі DBSCAN

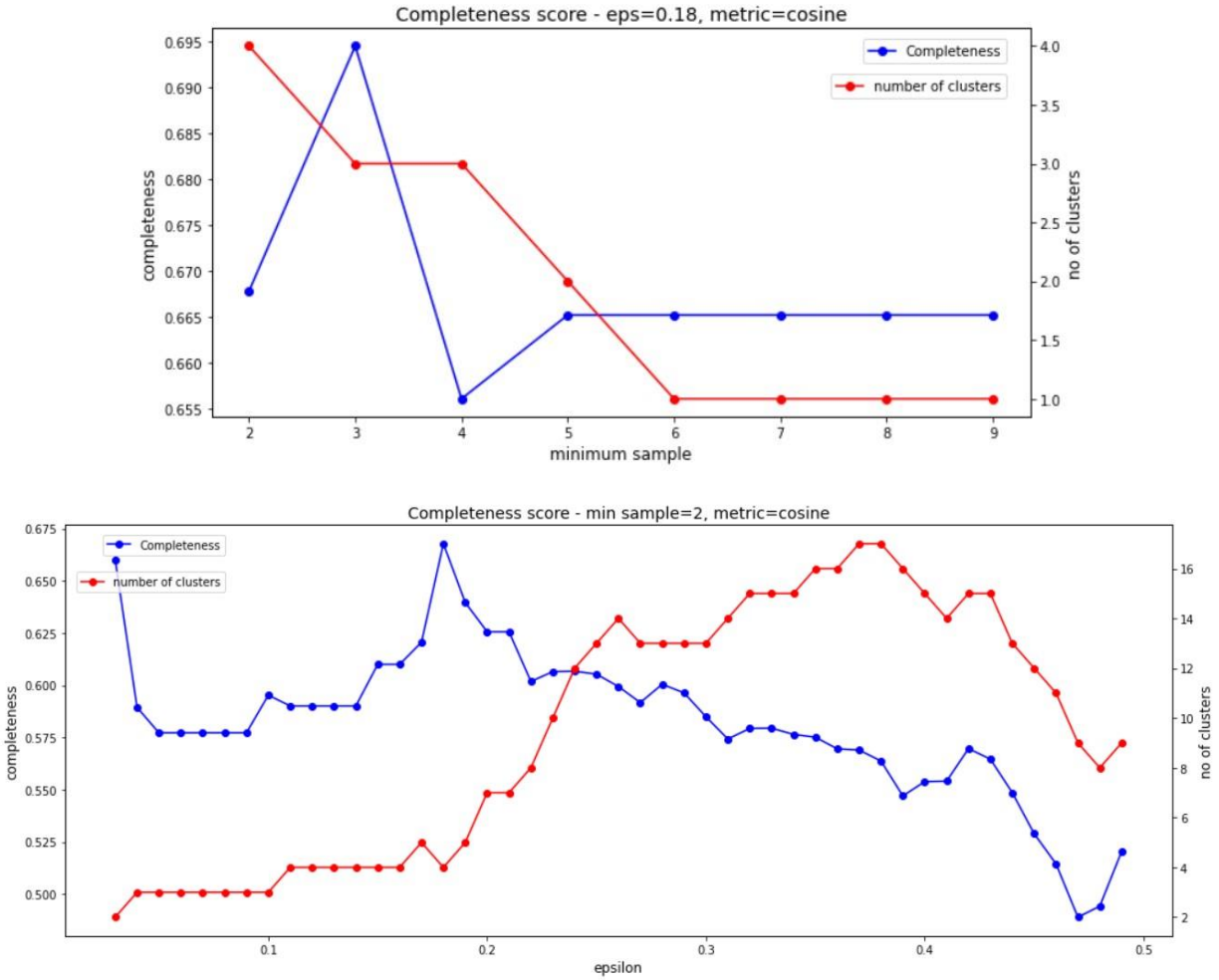


Рисунок 4.54– Оцінки повноти: Косинусні моделі DBSCAN

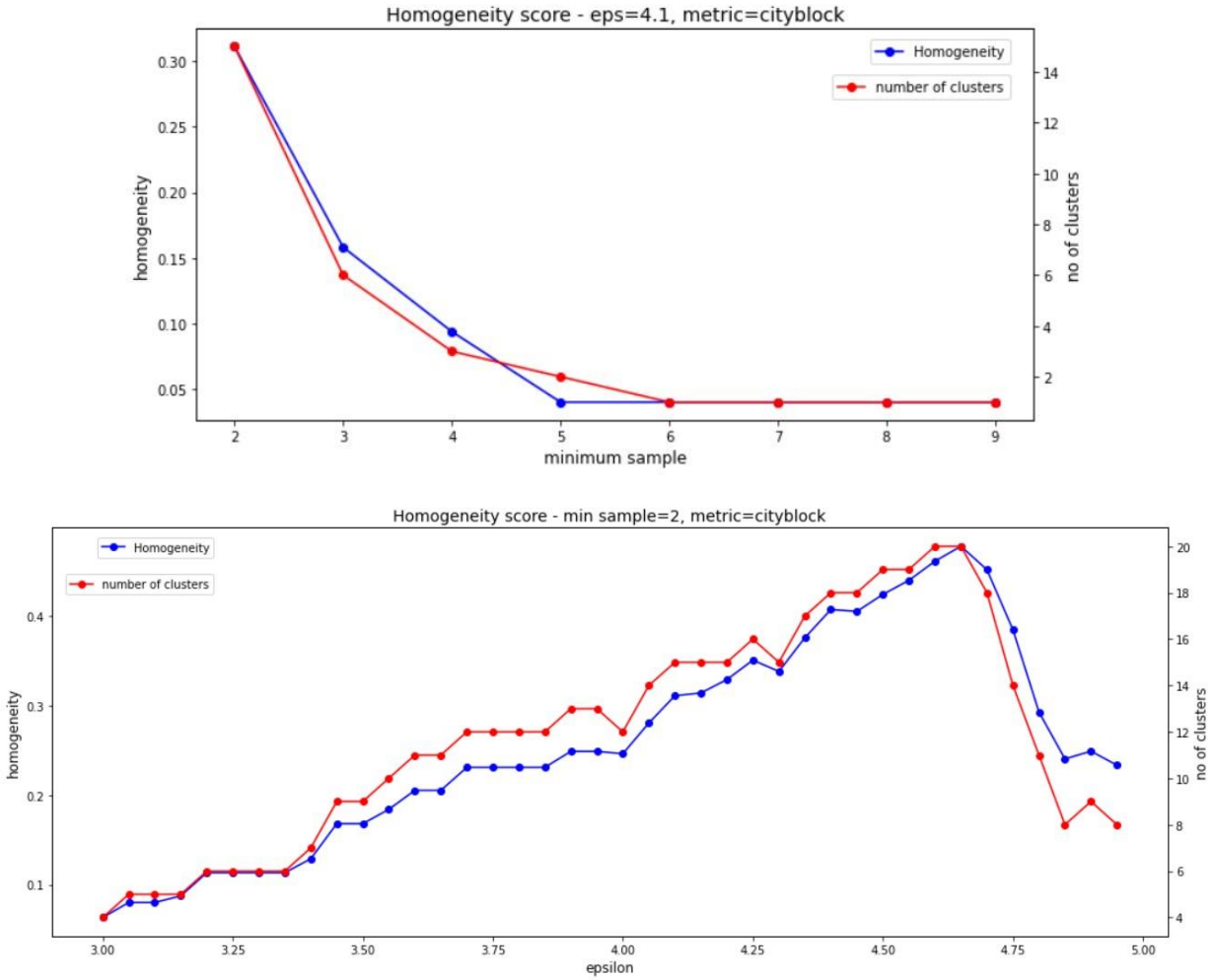


Рисунок 4.55 – Показники однорідності: Моделі міських кварталів DBSCAN

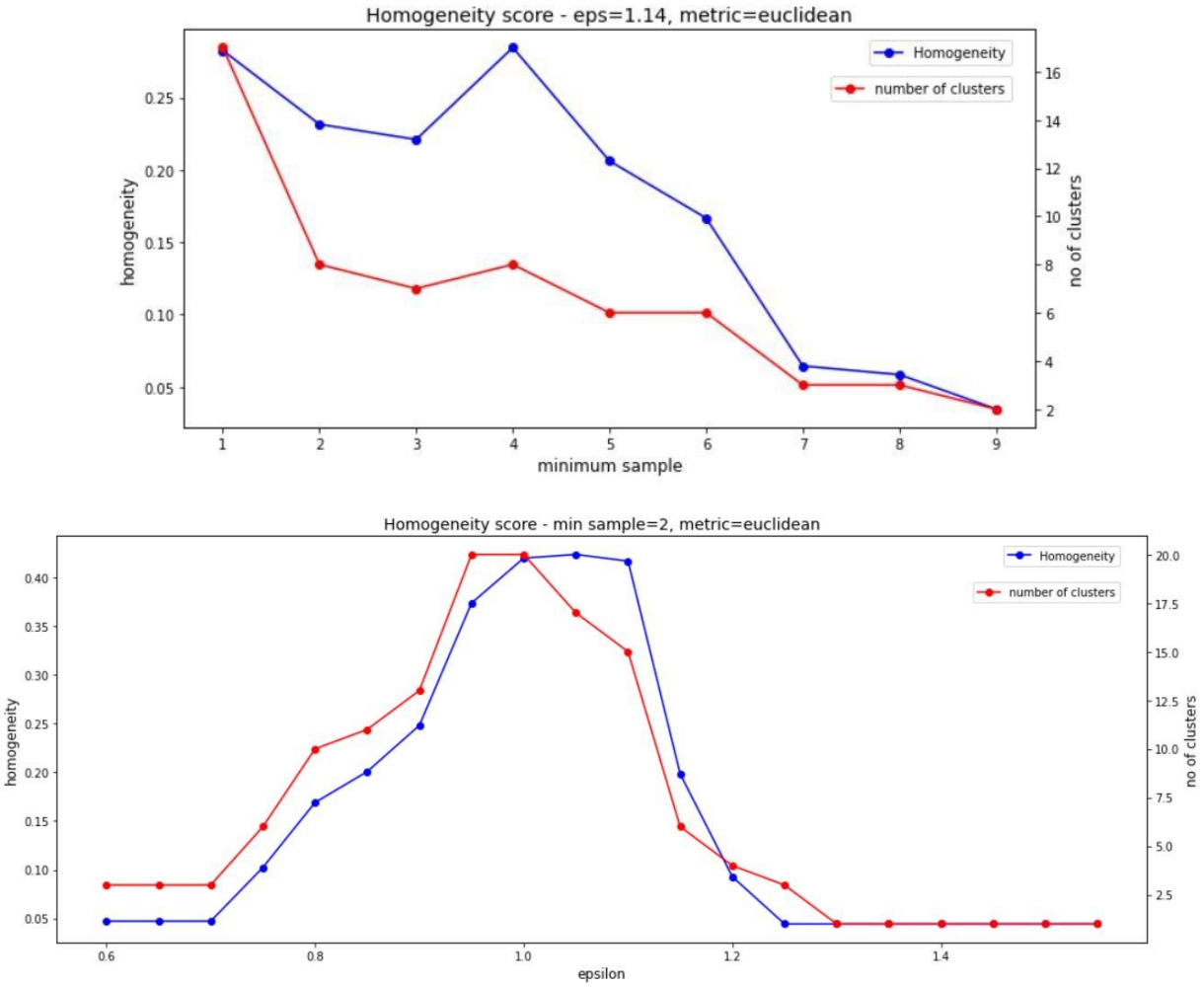


Рисунок 4.56 – Оцінки однорідності: Евклідові моделі DBSCAN

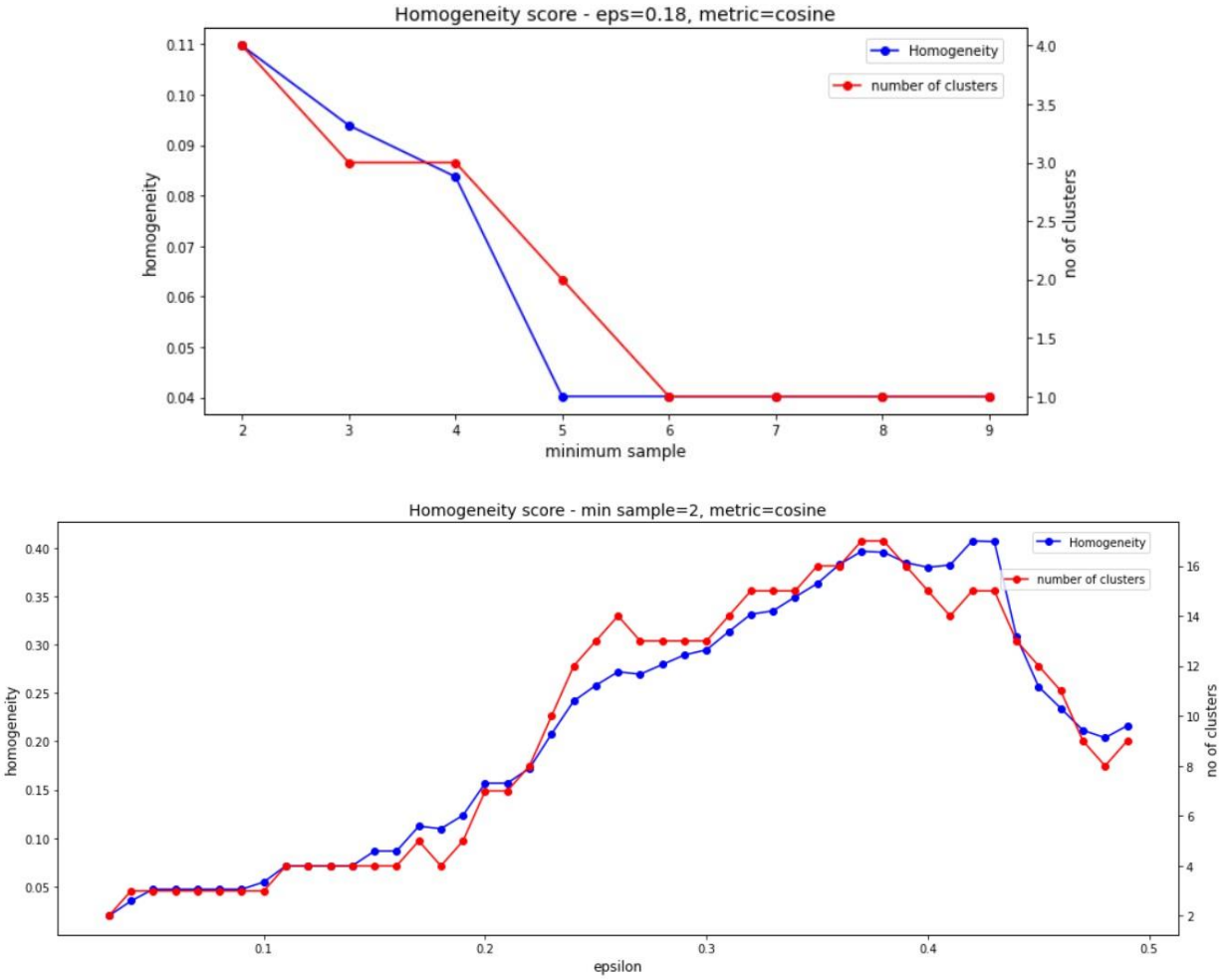


Рисунок 4.57 – Показники однорідності: Косинусні моделі DBSCAN

Загалом, обидва показники знижувалися, і зі збільшенням мінімальної вибірки було виявлено менше кластерів (оскільки для формування кластера потрібно більше точок). Однак, коли змінювався епсилон, коливання обох оцінок були більшими. Знову ж таки, через рівномірно розріджені дані, DBSCAN не дуже добре працював на цьому наборі даних. Сформовані кластери не були чітко розділені (рис. 4.58), а маркування, здавалося, об'єднувало багато різних класів в один (рис. 4.52).

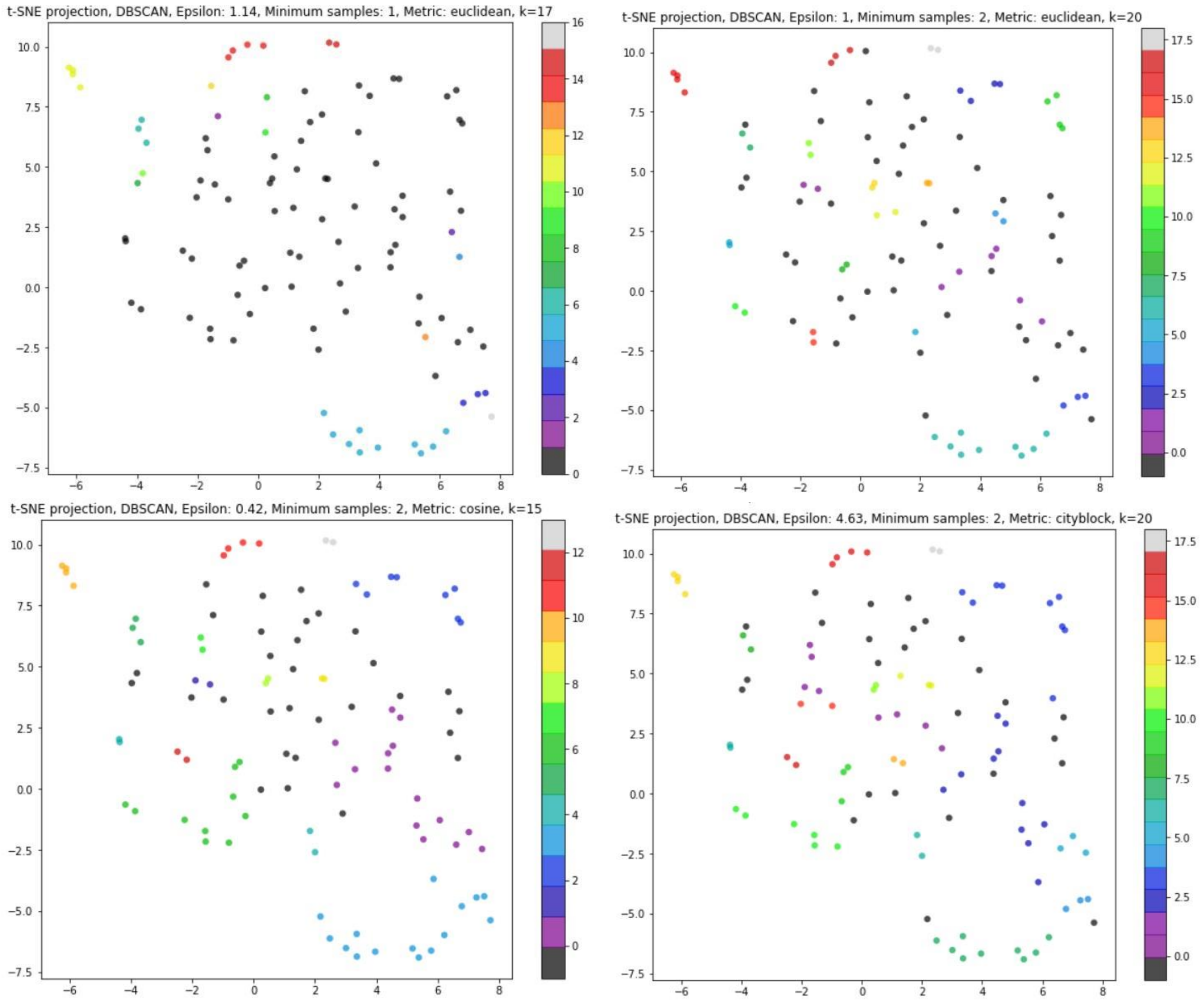


Рисунок 4.58 – Проекції t-SNE: DBSCAN Евклідова (вгорі), косинусна та сітьова (внизу)

Тим не менш, було обрано Cityblock (з eps=4.63 та min samples=2) як оптимальну модель для цього алгоритму. Вона має прийнятні показники однорідності (0,48) та повноти (0,58), досить добре відображається на проекції та створює 20 кластерів, які, як видається, краще відповідають золотим кластерам, ніж інші моделі.

4.5 Найкраща модель

4.5.1 Набір даних: Відгуки про подорожі

Таблиця 4.1 – Порівняння алгоритмів

	K-середнє	Ієрархічний	DBSCAN
Параметри	Евклідова	Косинус, повний зв'язок	Евклідова, eps=0.22, min вибірка=7

Кількість кластерів	7	7	5
Калинський-Гарабаш	174	132	62
Девіс-Булдін	1.75	1.94	3.63

З усіх трьох моделей, k-середнє дало найвищий бал за Калінскі і найнижчий бал за Девісом. Вона дала сім досить щільних і добре відокремлених кластерів на проекції t-SNE, а метод ліктя підтвердив, що для цієї моделі k=7 є оптимальною кількістю кластерів для даного набору даних. Відповідно, було обрано k-середнє евклідове з k=7 як найкращу модель для цього набору даних.

4.5.2 Набір даних: Прийняті статті ICMLA 2014

Таблиця 4.2 – Порівняння алгоритмів

	K-середнє	Ієрархічний	DBSCAN
Параметри	Евклідова	Косинус, середня ланка	Cityblock, eps=4.63, min sample=2
Кількість кластерів	24	24	20
Повнота	0.61	0.61	0.58
Однорідність	0.60	0.59	0.48

Загалом, усі моделі отримали схожі оцінки повноти. Однак модель DBSCAN отримала значно нижчі оцінки однорідності, ніж дві інші моделі. Це не дивно, оскільки DBSCAN, яка формує кластери на основі щільності точок даних, не змогла впоратися з рівномірно розрідженим набором даних. З іншого боку, k-середні та ієрархічні моделі працювали краще і мали вищі показники повноти та однорідності. Хоча обидві моделі мали однаково високі показники повноти та однорідності, добре відображалися на проекціях t-SNE і досить добре групувалися в таблиці маркування (рис. 5.2.1b і рис. 5.2.2c), дендограма зробила ієрархічну модель більш інтерпретованою, ніж k-середня, і вона забезпечує більшу гнучкість для бізнесу щодо вибору кількості кластерів (рис. 5.2.2j).

Тому було обрано ієрархічний косинусоїдальний зв'язок як оптимальну модель для цього набору даних.

ЗАГАЛЬНИЙ ВИСНОВОК

Під час виконання дипломної роботи було досліджено алгоритми кластеризації для обробки великих даних, зокрема K-means, ієрархічну кластеризацію та DBSCAN. Було розглянуто основні принципи роботи цих алгоритмів, їхні переваги та недоліки у різних сценаріях використання.

У процесі дослідження було розроблено програмне забезпечення, яке дозволяло виконувати кластеризацію великих наборів даних, що дало змогу на практиці протестувати роботу алгоритмів. Було враховано особливості роботи кожного алгоритму, зокрема:

– ефективність при різній кількості даних;
– стійкість до шумів та аномалій;
– точність кластеризації залежно від характеристик даних.

Особлива увага була приділена тестуванню алгоритмів та заміру продуктивності основних операцій, а саме:

– обчислення кластерів із використанням K-means;
– побудови дендрограм для ієрархічної кластеризації;
– виділення кластерів різної густини за допомогою DBSCAN.

Було проведено порівняльний аналіз роботи алгоритмів за такими метриками, як точність кластеризації, стійкість до шумів та гнучкість у роботі з великими наборами даних. На основі отриманих результатів надано рекомендації щодо вибору алгоритму залежно від конкретних умов та вимог завдання.

Крім того, у роботі розглянуто використані програмні бібліотеки та інструменти, які були застосовані для реалізації та тестування алгоритмів, зокрема: Scikit-learn, NumPy, Pandas, SciPy та Matplotlib.

Таким чином, проведена робота дозволила надати комплексний огляд алгоритмів кластеризації та їхньої ефективності, а також забезпечити практичні рекомендації для їх використання у задачах аналізу великих даних.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Bruce P. Practical Statistics for Data Science / Peter Bruce, Andrew Bruce, Peter Gedeck. – [S. l.] : OReilly, 2020. – 360 p.

2. Deisenroth M. P. Mathematics for Machine Learning / Marc Peter Deisenroth. – [S. l.] : Cambridge University Press, 2020. – 398 p.

3. Mathematical statistics with applications / ed. by M. William, S. R. L. – 7th ed. – Belmont, CA : Thomson Brooks/Cole, 2007. – 912 p.

4. Raschka S. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python / Sebastian Raschka, Yuxi Hayden Liu, Vahid Mirjalili. – [S. l.] : Packt Publishing, 2022. – 770 p.

5. MinMaxScaler [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (date of access: 21.11.2024). – Title from screen.

6. PCA [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA (date of access: 21.11.2024). – Title from screen.

7. LabelEncoder [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder (date of access: 21.11.2024). – Title from screen.

8. KMeans [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans (date of access: 21.11.2024). – Title from screen.

9. Data mining. – [S. l.] : Morgan Kaufmann, 2012. – 744 p.

10. Contributors to Wikimedia projects. Elbow method (clustering) - Wikipedia [Electronic resource] / Contributors to Wikimedia projects // Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Elbow_method_(clustering) (date of access: 21.11.2024). – Title from screen.

11. Davies-Bouldin score [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html (date of access: 21.11.2024). – Title from screen.

12. Calinski and Harabasz score [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html (date of access: 21.11.2024). – Title from screen.

13. Completeness score [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html#sklearn.metrics.completeness_score (date of access: 21.11.2024). – Title from screen.

14. Homogeneity score [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html#sklearn.metrics.homogeneity_score (date of access: 21.11.2024). – Title from screen.

15. Agglomerative clustering [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering (date of access: 21.11.2024). – Title from screen.

16. Hierarchical clustering – SciPy v1.14.1 Manual [Electronic resource] // Numpy and Scipy Documentation – Numpy and Scipy documentation. – Mode of access: https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html (date of access: 21.11.2024). – Title from screen.

17. Introduction to Data Mining, (First Edition). – [S. l.] : Addison Wesley, 2005. – 769 p.

18. Dendrogram – SciPy v1.14.1 Manual [Electronic resource] // Numpy and Scipy Documentation – Numpy and Scipy documentation. – Mode of access: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html#scipy.cluster.hierarchy.dendrogram (date of access: 21.11.2024). – Title

from screen.

19. DBSCAN [Electronic resource] // scikit-learn. – Mode of access: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (date of access: 21.11.2024). – Title from screen.

ДОДАТОК А

Технічно завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

«АНАЛІЗ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ»

Технічне завдання

44165850.1351-01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександр ІВАНОВ
Виконавець
_____Владислав ЄРМАКОВ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1351-01

«АНАЛІЗ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ»

Технічне завдання

44165850.1351-01

Листів 15

2025

44165850.1351-01

ЗМІСТ

ВСТУП

У сучасному світі аналіз та обробка великих обсягів даних стали однією з найважливіших складових інформаційних технологій. З метою виявлення прихованих закономірностей, покращення якості рішень та підвищення ефективності роботи з інформацією, все більше організацій звертаються до алгоритмів кластеризації. Одним із ключових завдань при цьому є вибір оптимального методу кластеризації, що дозволяє структурувати дані для подальшого аналізу.

У даному дослідженні зосередимо увагу на порівнянні та аналізі функціональних можливостей трьох популярних підходів до кластеризації: K-means, ієрархічної кластеризації та DBSCAN. K-means є класичним методом, що базується на поділі даних на кластери з фіксованим числом центрів, демонструючи високу швидкість роботи з великими наборами даних. Ієрархічна кластеризація дозволяє побудувати структуру у вигляді дерева (дендрограми), забезпечуючи гнучкість у виборі кількості кластерів. DBSCAN, у свою чергу, виділяється здатністю знаходити кластери різної густини та стійкістю до шумів у даних.

Це дослідження розглядає основні властивості кожного алгоритму, їхні переваги та недоліки, а також порівнює їх за такими критеріями, як точність кластеризації, швидкість виконання, стійкість до аномалій і шумів. Також дослідимо практичні сценарії, в яких кожен з алгоритмів є найбільш ефективним, і проаналізуємо результати їх застосування на реальних та синтетичних наборах даних.

Основною метою цього дослідження є допомога аналітикам і розробникам у виборі оптимального алгоритму кластеризації для своїх проєктів, враховуючи специфіку даних і поставлені задачі. Спробуємо відповісти на питання, який із цих алгоритмів є найкращим вибором для конкретних умов використання, та які переваги вони можуть принести в задачах аналізу великих даних у сучасних інформаційних технологіях.

44165850.1351-01

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ ректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №1196 ст від 05.12. 2022 року.

Тема проекту: «Аналіз алгоритмів кластеризації для обробки великих даних».

Керівник дипломного проекту: Іванов О. П.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Ця розробка спрямована на проведення аналізу та порівняння ефективності трьох популярних алгоритмів кластеризації даних: K-means, ієрархічної кластеризації та DBSCAN. Головною метою є дослідження та оцінка їхніх можливостей у задачах обробки великих обсягів даних для визначення найбільш придатного алгоритму для конкретних проєктів та завдань.

Ця робота покликана виявити переваги та недоліки кожного алгоритму в таких аспектах, як точність кластеризації, швидкість виконання, стійкість до шумів, здатність працювати з великими наборами даних та ефективність у різних сценаріях. Результати дослідження допоможуть аналітикам та розробникам обґрунтовано вибрати алгоритм кластеризації, який найкраще відповідає їхнім потребам.

Крім того, ця робота спрямована на аналіз конкретних прикладів використання алгоритмів кластеризації, що дасть змогу отримати практичні рекомендації для їхнього успішного застосування. Результати дослідження можуть бути корисними для організацій та фахівців, які працюють із великими обсягами даних і прагнуть оптимізувати процеси їхнього аналізу.

2.1 Функціональне призначення розробки

Основні аспекти функціонального призначення розробки:

– проведення аналізу алгоритмів кластеризації – основною функцією розробки є докладний аналіз трьох популярних алгоритмів кластеризації: K-means, ієрархічної кластеризації та DBSCAN. Це включає вивчення їхньої ефективності, точності, продуктивності, здатності працювати з великими даними та стійкості до шумів;

– порівняння алгоритмів – розробка дозволить порівняти функціональні особливості алгоритмів, визначити їхні переваги та недоліки, а також надати об’єктивну оцінку їхнього потенціалу для застосування в реальних умовах;

– вивчення практичних застосувань – розробка буде спрямована на дослідження сценаріїв використання алгоритмів кластеризації у різних галузях,

включаючи аналіз клієнтських даних, сегментацію ринків, виділення аномалій та інші завдання, що стосуються великих обсягів даних;

– надання рекомендацій – результати розробки допоможуть створити практичні рекомендації для вибору алгоритму кластеризації залежно від специфіки задач, характеристик даних та вимог до точності й продуктивності.

Висновки та розробка документації – розробка завершиться формулюванням висновків та підготовкою документації, яка включатиме результати аналізу, порівняльні характеристики алгоритмів та рекомендації для їхнього майбутнього використання.

2.2 Експлуатаційне призначення розробки:

Основні аспекти експлуатаційного призначення розробки:

– оцінка і вибір алгоритму кластеризації – результати дослідження допоможуть організаціям і аналітикам визначити, який з алгоритмів кластеризації (K-means, ієрархічна кластеризація чи DBSCAN) найкраще відповідає їхнім потребам і специфіці завдань;

– планування аналізу даних – отримані знання про функціональні можливості кожного алгоритму допоможуть розробникам і аналітикам планувати та реалізовувати проекти, які вимагають використання кластеризації, забезпечуючи оптимальне використання їхніх властивостей;

– оптимізація роботи з даними – знання про ефективність алгоритмів дозволить оптимізувати процеси обробки даних, покращити точність аналізу та зменшити час виконання завдань;

– підтримка та розвиток аналітичних проектів – розробка надає можливість підтримувати та вдосконалювати існуючі проекти шляхом впровадження кластеризації як частини аналізу великих даних. Це забезпечить гнучкість і адаптивність системи до змінних потреб користувачів;

– підготовка документації – експлуатаційне призначення включає підготовку документації, яка містить рекомендації, найкращі практики, опис метрик

та результати дослідження, що можуть бути використані для майбутніх проектів та аналітичних задач.

У підсумку, експлуатаційне призначення розробки полягає в практичному використанні отриманих знань для вдосконалення процесів кластеризації в проектах, пов’язаних з аналізом великих даних, а також у впровадженні рекомендацій для вибору оптимального алгоритму кластеризації залежно від специфіки проектів та задач.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Основні вимоги до функціональних характеристик:

– групування даних – алгоритми повинні мати можливість ефективного розподілу даних по кластерах, зокрема для роботи з великими наборами структурованих та неструктурованих даних;

– точність кластеризації – дослідження повинно включати аналіз здатності алгоритмів K-means, ієрархічної кластеризації та DBSCAN створювати точні кластери залежно від характеристик даних (густина, шум, кількість вимірів тощо);

– швидкість виконання – необхідно оцінити продуктивність кожного алгоритму при роботі з наборами даних різних обсягів, а також заміряти час виконання основних операцій кластеризації;

– стійкість до шумів – дослідження має включати аналіз здатності алгоритмів працювати з даними, що містять шуми чи аномалії, і їх вплив на якість кластеризації;

– гнучкість у виборі параметрів – алгоритми повинні дозволяти налаштовувати ключові параметри, такі як кількість кластерів (для K-means), радіус кластера (DBSCAN) та тип зв'язків (ієрархічна кластеризація);

– візуалізація результатів – важливо забезпечити можливість візуального представлення результатів кластеризації, наприклад, через графіки чи дендрограми для ієрархічного методу.

– масштабованість – алгоритми мають демонструвати стабільну продуктивність і точність навіть при роботі з великими наборами даних;

– підтримка високої кількості вимірів – дослідження повинно оцінити здатність алгоритмів працювати з багатовимірними даними без значної втрати точності;

– інтеграція з іншими технологіями – алгоритми повинні бути сумісними з популярними бібліотеками для аналізу даних (наприклад, Scikit-learn, NumPy) та легко інтегруватися в аналітичні та дослідницькі проєкти;

– аналіз складності обчислень – важливо оцінити алгоритмічну складність кожного методу та визначити їх ефективність при зростанні обсягу даних;

– підтримка змішаних типів даних – дослідження має охоплювати можливості роботи алгоритмів із наборами даних, що містять як числові, так і категоріальні значення;

– адаптивність до реальних сценаріїв – необхідно перевірити, наскільки алгоритми відповідають вимогам реальних задач, наприклад, у сегментації клієнтів чи виділенні аномалій.

Ці вимоги до функціональних характеристик дослідження алгоритмів кластеризації допоможуть зрозуміти їхні можливості та обрати найкращий підхід для конкретних потреб проєкту.

Вхідні дані:

– вхідними даними для Jupyter Notebook є два файли у форматі CSV, які представляють різні набори даних.

Вихідні дані:

– вихідними даними є текстові результати та графіки, що генеруються безпосередньо у середовищі JupyterLab.

3.2 Вимоги до надійності

Вимоги до надійності наступні:

– програма повинна інформувати користувача про стан виконання операцій кластеризації та результати роботи;

– текст програми та результати дослідження мають зберігатися у вигляді резервних копій на зовнішньому носії або у віддаленому репозиторії;

– резервне копіювання даних забезпечить відновлення результатів у разі втрати або пошкодження інформації.

3.3 Вимоги експлуатації

Програмою може користуватися людина, яка має базові навички роботи з комп’ютером та десктопними пристроями..

3.4 Вимоги до складу та параметрів технічних засобів

Продукти, що розробляється повинен використовуватись на пристроях, що мають наступні характеристики:

– операційна система – комп'ютер повинен працювати під управлінням операційної системи, яка підтримує виконання програмного продукту. Для прикладу, це може бути операційна система Windows, macOS або Linux;

– діагональ монітора – монітор комп'ютера повинен мати достатньою діагональ для зручного відображення інтерфейсу продукту та взаємодії з ним. Ідеально, це може бути монітор із середньою діагоналлю, наприклад, 21-24 дюйми;

– роздільна здатність монітора – монітор повинен мати достатньо високу роздільну здатність для чіткого та якісного відображення інтерфейсу продукту. Рекомендована роздільна здатність - Full HD (1920x1080) або вища;

– оперативна пам'ять (RAM) – комп'ютер повинен мати достатньо оперативної пам'яті для ефективної роботи програмного продукту. Рекомендована кількість оперативної пам'яті - не менше 4 ГБ;

– вбудована пам'ять (жорсткий диск або SSD) – комп'ютер повинен мати достатньо внутрішньої пам'яті для зберігання програмного продукту та його даних. Рекомендована ємність вбудованої пам'яті - не менше 256 ГБ;

– частота процесора – комп'ютер повинен мати процесор з достатньою частотою для швидкої обробки даних та виконання завдань програмного продукту. Рекомендована частота процесора - не менше 2.5 ГГц;

– порти та комунікації – комп'ютер повинен мати необхідні порти та комунікаційні можливості для підключення до мережі, зовнішніх пристроїв та інтернету. Доцільно мати USB-порти, Ethernet-порт для мережевого підключення, а також можливість підключення до Wi-Fi.

3.5 Вимоги до інформаційної та програмної сумісності

Програмні продукти розробляється для всіх видів операційних систем сімейства “Windows” починаючи від версії 10 та наступні версії.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

– текст програми;

– керівництво користувача для користування додатком.

Вся документація програмних додатків повинна задовольняти вимоги до програмної документації.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 5.1 – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	04.09.24 –15.09.24
Робочий проект	Програмування та відлагодження програми.	18.09.24 – 22.09.24
	Тестування програми	25.09.24 – 27.09.24
	Розробка, узгодження і затвердження програмної документації.	28.09.24 – 29.09.24

6 ПОРЯДОК ПРИЙНЯТТЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Іванов О.П.

7 ТЕХНІЧНО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Показники та їх розрахунок не були описані у документах бо розробка програмного продукту несе в собі навчальний характер, а не комерційний.

ДОДАТОК Б

Технічне завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

«АНАЛІЗ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ»

Текст програми

44165850.01223–01 12–01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександр ІВАНОВ
Виконавець
_____Владислав Єрмаков
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01223–01 12–01

АНАЛІЗ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ

Текст програми

44165850.01223–01 12–01

Листів 17

2025

44165850.01223–01 12

АНОТАЦІЯ

Документ 44165850.01223–01 12 01 «Аналіз алгоритмів кластеризації для обробки великих даних» є частиною програмної документації на програму, що виконує роль прототипу для аналізу та порівняння методів кластеризації великих обсягів даних.

У цьому документі представлено текст програми, створеної мовою програмування Python із використанням бібліотек Scikit-learn, NumPy, Pandas та Matplotlib. Розробка здійснювалася в середовищі Jupyter Notebook.

Даний документ охоплює ключові аспекти функціональності, реалізації та тестування алгоритмів кластеризації, таких як K-means, ієрархічна кластеризація та DBSCAN, для забезпечення їхньої адаптації до задач аналізу великих даних.

ЗМІСТ

44165850.01223–01 12

1 ТЕКСТ ПРОГРАМИ

44165850.01223–01 12

1. Налаштування даних

1.1 Завантаження даних

```
import pandas as pd
import numpy as np
import os
import time
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy.spatial.distance import cdist
```

Travel Review - Набір даних

Додаткова інформація:

Цей набір даних наповнюється шляхом сканування сайту TripAdvisor.com. Розглядаються відгуки про напрямки в 10 категоріях, згаданих по всій Східній Азії. Кожна оцінка мандрівників відображається як відмінна (4), дуже добра (3), середня (2), погана (1) і жахлива (0), а середня оцінка використовується проти кожної категорії для кожного користувача.

Відсутні значення?:

Ні

Опис категорій:

1) художні галереї

2) танцювальні клуби

3) сокові бари

4) ресторани

5) музеї

6) курорти

7) парки/місця для пікніків

8) пляжі

9) театри

10) релігійні установи

```
# Travel Review - Набір даних
review_df = pd.read_csv('tripadvisor_review.csv')
```

ICMLA - Набір даних

Додаткова інформація:

CSV-формат, де кожен рядок - це документ, а кожен стовпець - атрибут.

Має пропущені значення?:

Ні

Опис стовпців:

* Paper_Id: Номер; ідентифікатор документа

* Paper_Title: Вільний текст; назва статті

* Автор_Ключові слова: Вільний текст; авторські ключові слова

* Анотація: Вільний текст; анотація доповіді

* Сесія: Категоричний; сесія конференції, обрана організатором конференції, на якій було представлено статтю

```
# ICMLA - Набір даних
icmla_df = pd.read_csv('ICMLA_2014.csv', encoding='ISO-8859-1')

review_df = review_df.iloc[:, 1:]
icmla_df = icmla_df.iloc[:, 1:]
```

1.2 Попереднє опрацювання даних

Travel Review - Набір даних

У цьому наборі даних є 980 прикладів, кожен з яких представляє середні оцінки, надані мандрівниками за десятьма категоріями. З даних видно, що парки/місця для пікніків (категорія 7), пляжі (8) та релігійні установи (10) отримали високу оцінку мандрівників

(середній бал >2,5), тоді як художні галереї (1), ресторани (4) та музеї (5) були найменш популярними (середній бал <1)

review_df.head()

Опис основних статистичних даних

review_df.describe()

Перетворення даних

Оскільки для кластеризації даних використовуватимуться алгоритми на основі відстані, всі стовпчики ознак було нормалізовано (до значень від нуля до одиниці) за допомогою sklearn MinMaxScaler, щоб уникнути домінування атрибутів з великими значеннями при обчисленні відстані. Хоча всі значення ознак знаходяться в діапазоні від 0 до 4, виявили, що нормалізація даних допомогла покращити результати кластеризації.

MinMaxScaler нормалізує дані, масштабуючи кожну ознаку окремо так, щоб вони знаходилися в заданому діапазоні, наприклад, між нулем і одиницею. Перетворення задається формулою:

(1) X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

(2) X_scaled = X_std * (max - min) + min.

перетворення даних в нумпічний масив

X = np.array(review_df)

нормалізація даних за допомогою MinMaxScaler з sklearn

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_norm = scaler.fit_transform(X)

ICMLA - Набір даних

У цьому наборі даних 105 прикладів, кожен з яких представляє прийняту доповідь на конференції ICMLA 2014 року, включаючи її ідентифікатор, назву доповіді, ключові слова автора та тези, з колонкою «сесія», представленою у вигляді золотих кластерів.

icmla_df.head()

На додаток до колонок назви, ключових слів та анотацій, було створено «комбіновану» колонку, яка включає тексти з інших трьох колонок. Це дозволило нам спробувати різні стовпці характеристик, щоб переконатися, що для нашого алгоритму кластеризації обрано найкращий. Крім того, стовпець «сеанс» (золоті кластери) було перетворено в числа (0-23) за допомогою класу sklearn LabelEncoder, щоб його можна було обробити для оцінки моделі.

об'єднує три характеристики в один стовпчик, щоб їх можна було обробити векторизатором (далі в коді)

icmla_df['combine'] = icmla_df['paper_title'] + icmla_df['author_keywords'] + icmla_df['abstract']

LabelEncoder - перетворення «сесії» золотих кластерів у числа

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

Присвоєння числових значень і збереження в іншому стовпчику

icmla_df['session_code'] = labelencoder.fit_transform(icmla_df['session'])

перетворення кожного стовпеця у масив, щоб їх можна було обробити алгоритмом стовпців-характеристик

X2 = np.array(icmla_df)

X2_title = np.array(icmla_df['paper_title'])

X2_keywords = np.array(icmla_df['author_keywords'])

X2_abstract = np.array(icmla_df['abstract'])

X2_combine = np.array(icmla_df['combine'])

золоті кластери

y2_string = np.array(icmla_df['session'])

y2 = np.array(icmla_df['session_code'])

np.unique(y2) # у цьому наборі даних є 24 справжніх кластера

Оскільки кластеризація вимагає вимірювання відстані між зразками, попередньо обробили текст у кожному стовпчику ознак і побудували числове представлення за допомогою бібліотеки Natural Language Tool Kit (nltk). Спочатку було виконано токенізацію, розбивши текст на речення, а потім на словосполучення. Потім було очищено дані, видаливши всі токени, що містять нелітерні символи, за допомогою пакету re (регулярні вирази) і виконали стемінг за допомогою методу «снігової кулі» для отримання кореневих форм. Нарешті, перевели їх у числове представлення, побудувавши впорядкований вектор частоти терміна - зворотної частоти документа (TF-IDF) для кожного унікального слова в корпусі.

```
import nltk
from nltk.stem.snowball import SnowballStemmer
import re

stemmer = SnowballStemmer("english")
nltk_data_path = "/opt/homebrew/Caskroom/miniconda/base/envs/diploma/nltk_data"
os.environ["NLTK_DATA"] = nltk_data_path
nltk.download('punkt', download_dir=nltk_data_path)
nltk.download('punkt_tab', download_dir=nltk_data_path)

# функція для токенізації та виведення текстових даних
def tokenize_and_stem(text):
  # спочатку токенізуємо речення, потім слово, щоб переконатися, що розділові знаки сприймаються як власний токен
  tokens = [word for sent in nltk.sent_tokenize(text) for word in nltk.word_tokenize(sent)]
  filtered_tokens = []
  # відфільтруємо всі токени, що не містять літер (наприклад, числові токени, сирі пунктуаційні знаки)
  for token in tokens:
     if re.search('[a-zA-Z]', token):
        filtered_tokens.append(token)
  stems = [stemmer.stem(t) for t in filtered_tokens]
  return stems

def trans_vector(X):
  from sklearn.feature_extraction.text import TfidfVectorizer
  from sklearn.metrics.pairwise import cosine_similarity
  import seaborn as sns

  tfidf_vectorizer = TfidfVectorizer(max_df=0.81, max_features=200000,
                    min_df=0.08, stop_words='english',
                    use_idf=True, tokenizer=tokenize_and_stem, ngram_range=(1,3))

  tfidf_matrix = tfidf_vectorizer.fit_transform(X)
  return (1 - cosine_similarity(tfidf_matrix))

def make_pca_analysis(X):
  from sklearn.decomposition import PCA
  pca = PCA()
  X_pca = pca.fit(X)

  print("Коефіцієнт поясненого відхилення:")
  var = np.cumsum(np.round(pca.explained_variance_ratio_, decimals=3))
  print(var)
  for n, v in enumerate(var):
   if v>0.9:
     n_comp = n + 1
     break
  plt.ylabel('% Пояснення відхилень')
  plt.xlabel('# Кількість функцій')
  plt.title('PCA-аналіз')
  plt.plot(var)
  plt.show()
  return n_comp

def reduce(X,n_component):

  from sklearn.decomposition import PCA
  pca = PCA(n_components=n_component)
  X_centered = X - X.mean(axis=0)
```

```
 pca.fit(X_centered)
 X_pca = pca.transform(X_centered)

 return X_pca

print(nltk.data.path)

X_title = trans_vector(X2_title)
X_keywords = trans_vector(X2_keywords)
X_abstract = trans_vector(X2_abstract)
X_combine = trans_vector(X2_combine)
X_keywords.shape
```

Оскільки отримані числові представлення містили 105 стовпчиків ознак, а така велика розмірність даних може спричинити проблеми зі зберіганням, часом обробки та візуалізацією, для аналізу кількості ознак, необхідних для пояснення щонайменше 90% дисперсії даних, було використано метод головних компонент (sklearn Principal Component Analysis), після чого кількість ознак було відповідно зменшено (наприклад, для стовпчиків «ключові слова» та «об'єднати» 20 та 51 ознака пояснюють щонайменше 90% дисперсії в цих стовпчиках, тож кількість ознак було зменшено до 20 та 51, відповідно).

```
n_comp = make_pca_analysis(X_title)
print(n_comp)

X_title_pca = reduce(X_title,n_comp)

n_comp = make_pca_analysis(X_keywords)
print(n_comp)

X_keywords_pca = reduce(X_keywords,n_comp)

n_comp = make_pca_analysis(X_abstract)
print(n_comp)

X_abstract_pca = reduce(X_abstract,n_comp)

n_comp = make_pca_analysis(X_combine)
print(n_comp)

X_combine_pca = reduce(X_combine,n_comp)

---
# 2. Кластеризація за методом K-середніх

## 2.1 Travel Review - Набір даних

### Функції

from sklearn.metrics import pairwise_distances_argmin

def find_clusters(X, metric, n_clusters, rseed=2):

    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        labels = pairwise_distances_argmin(X, centers, metric=metric)
        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])

        if np.all(centers == new_centers):
            break
        centers = new_centers
    inertias = sum(((centers[label] - x)**2).sum()
                   for x, label in zip(X, labels))

    return centers, labels, inertias

def plot_elbow(inertias,k_range,elbow,ymin,ymax,xtext,ytext,title=""):
```

```
  plt.figure(figsize=(10, 6))
  plt.plot(range(1, k_range), inertias, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Інерція", fontsize=14)
  plt.xticks(np.arange(0, k_range, step=2))
  plt.annotate('Лікоть',
               xy=(elbow, inertias[elbow-1]),
               xytext=(xtext, ytext),
               textcoords='figure fraction',
               fontsize=16,
               arrowprops=dict(facecolor='black', shrink=0.1)
               )
  plt.axis([0, k_range, ymin, ymax])
  plt.title(title, size=12)
  plt.show()


def plot_2elbow(inertias1,inertias2,k_range,elbow1,elbow2,ymin,ymax, xtext1, ytext1, xtext2, ytext2, title=""):
  plt.figure(figsize=(10, 6))
  plt.plot(range(1, k_range), inertias1, "bo-" , label='Евклідова', color='green')
  plt.plot(range(1, k_range), inertias2, "bo-", label='Мангеттен', color='blue')
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Інерція", fontsize=14)
  plt.xticks(np.arange(0, k_range, step=2))
  plt.annotate('Лікоть1',
               xy=(elbow1, inertias1[elbow1-1]),
               xytext=(xtext1, ytext1),
               textcoords='figure fraction',
               fontsize=16,
               arrowprops=dict(facecolor='black', shrink=0.1)
               )
  plt.annotate('Лікоть2',
               xy=(elbow2, inertias2[elbow2-1]),
               xytext=(xtext2, ytext2),
               textcoords='figure fraction',
               fontsize=16,
               arrowprops=dict(facecolor='black', shrink=0.1)
               )
  plt.axis([0, k_range, ymin, ymax])
  plt.title(title, size=12)
  plt.legend()
  plt.show()


def calc_and_plot_silhouette(X, distance, labels, k_range, ymin, ymax, model=None, title=''):
  from sklearn.metrics import silhouette_score


  if distance == 'евклідовий':
    silhouette_scores = [silhouette_score(X, mod.labels_)
                         for mod in model[1:]]
  else:
    silhouette_scores = [silhouette_score(X, labels[k]) for k in range(1, k_range-1)]


  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), silhouette_scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Силуетний показник", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.show()


  return silhouette_scores


def plot_silhouette(X, no_cluster, labels1, labels2):
  from sklearn.metrics import silhouette_samples
  from sklearn.metrics import silhouette_score


  import matplotlib.cm as cm
  range_n_clusters = range(2,no_cluster+1)
```

```
 for n_clusters in range_n_clusters:
    figure, (axis1, axis2) = plt.subplots(1, 2)
    figure.set_size_inches(12, 5)

    axis1.set_xlim([-0.1, 1])
    axis1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    cluster_labels = labels1[n_clusters]

    silhouette_avg = silhouette_score(X, cluster_labels)
    print("\nСередній силуетний показники (модель 1): {:.4f}".format(silhouette_avg))

    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.Spectral(float(i) / n_clusters)
        axis1.fill_betweenx(np.arange(y_lower, y_upper),
                            0, ith_cluster_silhouette_values,
                            facecolor=color, edgecolor=color, alpha=0.7)

        axis1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        y_lower = y_upper + 10

    axis1.set_title("Силуетний графік для моделі 1")
    axis1.set_xlabel("Значення коефіцієнта силуету")
    axis1.set_ylabel("Назва кластуру")

    axis1.axvline(x=silhouette_avg, color="red", linestyle="--")

    axis1.set_yticks([])
    axis1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    axis2.set_xlim([-0.1, 1])
    axis2.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    cluster_labels = labels2[n_clusters]

    silhouette_avg = silhouette_score(X, cluster_labels)
    print("Середній силуетний показники (модель 2): {:.4f}".format(silhouette_avg))

    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.Spectral(float(i) / n_clusters)
        axis2.fill_betweenx(np.arange(y_lower, y_upper),
                            0, ith_cluster_silhouette_values,
                            facecolor=color, edgecolor=color, alpha=0.7)

        axis2.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
```

```
        y_lower = y_upper + 10

    axis2.set_title("Силуетний графік для моделі 2")
    axis2.set_xlabel("Значення коефіцієнта силуету")
    axis2.set_ylabel("Назва кластуру")

    axis2.axvline(x=silhouette_avg, color="red", linestyle="--")

    axis2.set_yticks([])
    axis2.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    plt.suptitle(("Силуетний аналіз для кластеризації KMeans "
                  "c k = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

    plt.show()

def plot_calinski(chscores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), chscores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка Калинського-Харабаша", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])

def plot_2calinski(chscores1, chscores2, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), chscores1, "bo-", label='Евклідова', color='green')
  plt.plot(range(2, k_range), chscores2, "bo-", label='Мангеттен', color='blue')
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка Калинського-Харабаша", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.legend()

def plot_davies(dbscores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), dbscores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка Девіса-Болдіна", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])

def plot_2davies(dbscores1, dbscores2, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), dbscores1, "bo-", label='Евклідова', color='green')
  plt.plot(range(2, k_range), dbscores2, "bo-", label='Мангеттен', color='blue')
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка Девіса-Болдіна", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.legend()

def plot_tsne(X,k,label,title=''):
  # %%time
  from sklearn.manifold import TSNE
  tsne = TSNE(random_state=42)
  X_tsne = tsne.fit_transform(X)
  plt.figure(figsize=(10,8))
  plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=label,
              edgecolor='none', alpha=0.7, s=40,
              cmap=plt.cm.get_cmap('nipy_spectral', k))
  plt.colorbar()
  title = 'Проекція t-SNE, ' + title + ', k=' + str(k)
  plt.title(title)

### Тренуємо Кластеризацію за K-середніми
```

П'ятдесят вісім моделей було навчено на цьому наборі даних з використанням k=1-29 та двох мір відстані. Евклідова distance, яка вимірює distance по прямій між двома точками, є загальною метрикою для k-середніх. Манхеттенська distance (або distance між кварталами), яка вимірює абсолютну distance або distance у кварталах між будь-якими двома точками в місті, була включена для порівняння.

K-середня кластеризація з використанням евклідової відстані

```
k_range = 30
X1_travel_df = X_norm

from sklearn.cluster import KMeans

kmeans_euclidean = [KMeans(n_clusters=k, random_state=42).fit(X1_travel_df)
        for k in range(1, k_range)]

labels_euclidean = [model.fit_predict(X1_travel_df) for model in kmeans_euclidean]

inertias_euclidean = [model.inertia_ for model in kmeans_euclidean]
```

K-середня кластеризація з використанням Манхеттенської відстані

```
k_range2 = 30

clusters_manhattan = [find_clusters(X1_travel_df, 'manhattan', n_clusters=k)
        for k in range(1, k_range2)]

centers_manhattan = [clusters_manhattan[k][0] for k in range(0, k_range2-1)]

labels_manhattan = [clusters_manhattan[k][1] for k in range(0, k_range2-1)]

inertias_manhattan = [clusters_manhattan[k][2] for k in range(0, k_range2-1)]
```

Перша оцінка (evaluation)

Метод "ліктя"

Метод ліктя, який спостерігає граничне зменшення суми внутрішньокластерних дисперсій (інерційність), був використаний для визначення відповідної кількості кластерів, які слід використовувати для цього набору даних.

```
df_inertia = pd.DataFrame(inertias_euclidean, columns=['inertia'])
df_inertia['difference'] = -(df_inertia.diff())
df_inertia.head(14)

plot_elbow(inertias_euclidean,k_range,elbow=7,ymin=45,ymax=250, xtext=0.35, ytext=0.65, title="K-Means - Евклідовий")

df_inertia2 = pd.DataFrame(inertias_manhattan, columns=['inertia'])
df_inertia2['difference'] = -(df_inertia2.diff())
df_inertia2.head(14)

plot_elbow(inertias_manhattan,k_range2,elbow=8,ymin=45,ymax=250, xtext=0.35, ytext=0.65, title="K-Means - Мангеттен")
```

З графіка інерції видно, що інерція дуже швидко падала при збільшенні k до 7 (Евклідова) і 8 (Манхеттен), але після цих точок почала вирівнюватися. Це свідчить про те, що k=7 для Евклідової системи та k=8 для Манхеттена є вдалим вибором.

```
plot_2elbow(inertias_euclidean,inertias_manhattan,k_range,elbow1=7,elbow2=8,ymin=45,ymax=250,
    xtext1=0.15, ytext1=0.15, xtext2=0.35, ytext2=0.55, title="K-Means - Евклідова vs Мангеттен")
```

Друга оцінка (evaluation)

Силуетний метод

Силуетний коефіцієнт, що варіюється від -1 (поганий) до 1 (хороший), вимірює компактність кластера, до якого належить точка, та відокремленість цієї точки від усіх інших кластерів (а оцінка силуету - це середній силуетний коефіцієнт для всіх точок у даних).

```
ss_au = calc_and_plot_silhouette(X1_travel_df, 'евклідовий', labels_euclidean, k_range, ymin=0.1, ymax=0.3, model=kmeans_euclidean, title="K-Means - Евклідова")

print("Силуетний показники (k=5): {:.04f}".format(ss_au[3]))
print("Силуетний показники (k=6): {:.04f}".format(ss_au[4]))
print("Силуетний показники (k=7): {:.04f}".format(ss_au[5]))
print("Силуетний показники (k=8): {:.04f}".format(ss_au[6]))
print("Силуетний показники (k=9): {:.04f}".format(ss_au[7]))
print("Силуетний показники (k=10): {:.04f}".format(ss_au[8]))

ss_man = calc_and_plot_silhouette(X1_travel_df, 'manhattan', labels_manhattan, k_range2, ymin=0.075, ymax=0.3, title="K-Means - Мангеттен")

print("Силуетний показники (k=5): {:.04f}".format(ss_man[3]))
print("Силуетний показники (k=6): {:.04f}".format(ss_man[4]))
print("Силуетний показники (k=7): {:.04f}".format(ss_man[5]))
print("Силуетний показники (k=8): {:.04f}".format(ss_man[6]))
print("Силуетний показники (k=9): {:.04f}".format(ss_man[7]))
print("Силуетний показники (k=10): {:.04f}".format(ss_man[8]))
print("Силуетний показники (k=11): {:.04f}".format(ss_man[9]))
```

На жаль, всі моделі дали низькі оцінки силуету (менше 0,2 для k>=3), хоча графіки коефіцієнтів силуету показали досить добре відокремлені кластери для k=7 і k=8.

```
plot_silhouette(X1_travel_df, no_cluster=12, labels1=labels_euclidean, labels2=labels_manhattan)
```

Третя оцінка (evaluation)

Індекс Девіса-Болдіна та індекс Калінкі-Харабаша

Оцінка Девіса-Болдіна вимірює середню «схожість» між кластерами, яка визначається відстанню між кластерами з урахуванням розміру кластерів (нижчий показник, близький до 0, вказує на кращу відокремленість між кластерами). Показник Калінкі-Харабаша вимірює співвідношення між внутрішньою та міжкластерною дисперсіями (вищий показник означає щільні та добре відокремлені кластери).

```
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

ks = []
dbs_euclideans = []
dbs_manhattans = []
chs_euclideans = []
chs_manhattans = []

for k in range(1, 15):
  dbse = davies_bouldin_score(X1_travel_df,labels_euclidean[k])
  dbsm = davies_bouldin_score(X1_travel_df,labels_manhattan[k])
  chse = calinski_harabasz_score(X1_travel_df,labels_euclidean[k])
  chsm = calinski_harabasz_score(X1_travel_df,labels_manhattan[k])

  ks.append(k+1)
  dbs_euclideans.append(dbse)
  dbs_manhattans.append(dbsm)
  chs_euclideans.append(chse)
  chs_manhattans.append(chsm)

df_scores = pd.DataFrame({"k":ks, "dbs_euclidean":dbs_euclideans,
                "dbs_manhattan":dbs_manhattans,
                "chs_euclidean":chs_euclideans,
                "chs_manhattan":chs_manhattans})
df_scores

plot_davies(dbs_euclideans, k_range=16, ymin=1.4, ymax=2, title='K-Means - Евклідовий')
```

plot_calinski(chs_euclideans, k_range=16, ymin=90, ymax=450, title='K-Means - Евклідовий')

plot_davies(dbs_manhattans, k_range=16, ymin=1.3, ymax=2.4, title='K-Means - Мангеттен')

plot_calinski(chs_manhattans, k_range=16, ymin=80, ymax=400, title='K-Means - Мангеттен')

Загалом, евклідові кластери краще розділяють кластери, ніж манхеттенські (на що вказують нижчі оцінки Девіса-Болдіна). Ігноруючи k=2-3, евклідові кластери дали найнижчий бал при k=9 (1,64), а манхеттенські - при k=11 (1,89). Аналогічно, оцінка Калінкі-Харабаша показала, що евклідові кластери утворюють більш щільні та краще відокремлені кластери, ніж манхеттенські, хоча оцінка зменшувалася зі збільшенням k.

plot_2davies(dbs_euclideans, dbs_manhattans, k_range=16, ymin=1.3, ymax=2.4, title='K-Means - Евклідовий vs Мангеттен')

plot_2calinski(chs_euclideans, chs_manhattans, k_range=16, ymin=80, ymax=450, title='K-Means - Евклідовий vs Мангеттен')

Зображення кластерів

Проекції t-SNE

Моделі k-середніх досить добре відображаються на проекціях t-SNE. Хоча вони не є ідеальними, утворені кластери є досить щільними і досить добре розділеними, особливо для евклідових.

plot_tsne(X1_travel_df,k=7,label=labels_euclidean[6], title='K-Means - Евклідова')

plot_tsne(X1_travel_df,k=8,label=labels_euclidean[7], title='K-Means - Евклідова')

plot_tsne(X1_travel_df,k=9,label=labels_euclidean[8], title='K-Means - Евклідова')

plot_tsne(X1_travel_df,k=10,label=labels_euclidean[9], title='K-Means - Евклідова')

plot_tsne(X1_travel_df,k=7,label=labels_manhattan[6], title='K-Means - Мангеттен')

plot_tsne(X1_travel_df,k=8,label=labels_manhattan[7], title='K-Means - Мангеттен')

plot_tsne(X1_travel_df,k=9,label=labels_manhattan[8], title='K-Means - Мангеттен')

plot_tsne(X1_travel_df,k=10,label=labels_manhattan[9], title='K-Means - Мангеттен')

plot_tsne(X1_travel_df,k=11,label=labels_manhattan[10], title='K-Means - Мангеттен')

2.2 ICMLA - Набір даних

Функції

```
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

def trans_vector2(X,max,min):
  from sklearn.feature_extraction.text import TfidfVectorizer
  from sklearn.metrics.pairwise import cosine_similarity
  import seaborn as sns

  tfidf_vectorizer = TfidfVectorizer(max_df=max, max_features=200000,
                     min_df=min, stop_words='english',
                     use_idf=True, tokenizer=tokenize_and_stem, ngram_range=(1,3))
```

```
  tfidf_matrix = tfidf_vectorizer.fit_transform(X)
  return (1 - cosine_similarity(tfidf_matrix))


from sklearn.metrics import pairwise_distances_argmin


from sklearn.metrics import pairwise_distances_argmin


def find_clusters(X, metric, n_clusters, rseed=2):
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]


    while True:
        labels = pairwise_distances_argmin(X, centers, metric=metric)


        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])


        if np.all(centers == new_centers):
            break
        centers = new_centers


    inertias = sum(((centers[label] - x)**2).sum()
                   for x, label in zip(X, labels))


    return centers, labels, inertias


def plot_silhouette2(X, no_cluster, labels1, labels2):
  from sklearn.metrics import silhouette_samples
  from sklearn.metrics import silhouette_score


  import matplotlib.cm as cm
  range_n_clusters = range(19,no_cluster+1)


  for n_clusters in range_n_clusters:
        figure, (axis1, axis2) = plt.subplots(1, 2)
        figure.set_size_inches(12, 5)


        axis1.set_xlim([-0.1, 1])
        axis1.set_ylim([0, len(X) + (n_clusters + 1) * 10])


        cluster_labels = labels1[n_clusters]


        silhouette_avg = silhouette_score(X, cluster_labels)
        print("\nСередній силуетний показники (модель 1): {:.4f}".format(silhouette_avg))


        sample_silhouette_values = silhouette_samples(X, cluster_labels)


        y_lower = 10
        for i in range(n_clusters):
            ith_cluster_silhouette_values = \
                sample_silhouette_values[cluster_labels == i]


            ith_cluster_silhouette_values.sort()


            size_cluster_i = ith_cluster_silhouette_values.shape[0]
            y_upper = y_lower + size_cluster_i


            color = cm.Spectral(float(i) / n_clusters)
            axis1.fill_betweenx(np.arange(y_lower, y_upper),
                                0, ith_cluster_silhouette_values,
                                facecolor=color, edgecolor=color, alpha=0.7)


            axis1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))


            y_lower = y_upper + 10


        axis1.set_title("Силуетний графік для моделі 1")
        axis1.set_xlabel("Значення коефіцієнта силуету")
        axis1.set_ylabel("Назва кластуру")
```

```
    axis1.axvline(x=silhouette_avg, color="red", linestyle="--")

    axis1.set_yticks([])
    axis1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    axis2.set_xlim([-0.1, 1])

    axis2.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    cluster_labels = labels2[n_clusters]

    silhouette_avg = silhouette_score(X, cluster_labels)
    print("Середній силуетний показники (модель 2): {:.4f}".format(silhouette_avg))

    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):

        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.Spectral(float(i) / n_clusters)
        axis2.fill_betweenx(np.arange(y_lower, y_upper),
                            0, ith_cluster_silhouette_values,
                            facecolor=color, edgecolor=color, alpha=0.7)

        axis2.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        y_lower = y_upper + 10

    axis2.set_title("Силуетний графік для моделі 2")
    axis2.set_xlabel("Значення коефіцієнта силуету")
    axis2.set_ylabel("Назва кластуру")

    axis2.axvline(x=silhouette_avg, color="red", linestyle="--")

    axis2.set_yticks([])
    axis2.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    plt.suptitle(("Силуетний аналіз для кластеризації KMeans "
                  "c k = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

    plt.show()

def plot_completeness2(scores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(np.arange(2, k_range, 2), scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка завершеності", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.show()

def plot_homogeneity2(scores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(np.arange(2, k_range, 2), scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка однорідності", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.show()
```

```
def plot_tsne(X,k,label,title=''):

 from sklearn.manifold import TSNE
 tsne = TSNE(random_state=42)

 X_tsne = tsne.fit_transform(X)

 plt.figure(figsize=(10,8))
 plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=label,
          edgecolor='none', alpha=0.7, s=40,
          cmap=plt.cm.get_cmap('nipy_spectral', k))
 plt.colorbar()
 title = 'Проекція t-SNE, ' + title + ', k=' + str(k)
 plt.title(title)
```

Тренуємо Кластеризацію за K-середніми

Звернемо увагу, що наведений нижче код виконується досить довго. Він використовується для визначення оптимальних значень параметрів max_df та min_df у TfidfVectorizer за показниками повноти та однорідності.

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, completeness_score, homogeneity_score

accu_max = 0
complete_max = 0
homo_max = 0

max_range = np.arange(0.7,1,0.01)
min_range = np.arange(0,0.3,0.01)

max_range = np.round(max_range, 4)
min_range = np.round(min_range, 4)

for max in max_range:
 for min in min_range:
  X = trans_vector2(X2_combine,max,min)

  pca = PCA()
  pca.fit(X)
  var = np.cumsum(np.round(pca.explained_variance_ratio_, decimals=3))
  for n, v in enumerate(var):
   if v>0.9:
    n_comp = n+1
    break

  pca = PCA(n_components=n_comp)
  pca.fit(X)
  X_pca = pca.transform(X)

  k = 24
  X_data = X_pca
  kmeans_euclidean224 = KMeans(n_clusters=k, random_state=42)
  kmeans_euclidean224.fit(X_data)
  y_kmeans_euclidean224 = kmeans_euclidean224.predict(X_data)

  accu = accuracy_score(y2, y_kmeans_euclidean224)
  if accu>accu_max:
   accu_max = accu
   max_accu = max
   min_accu = min

  complete = completeness_score(y2, y_kmeans_euclidean224)
  if complete>complete_max:
   complete_max = complete
   max_c = max
   min_c = min
```

```
    homogeneity = homogeneity_score(y2, y_kmeans_euclidean224)
    if homogeneity>homo_max:
      homo_max = homogeneity
      max_h = max
      min_h = min

print("Точність: {:.4f}".format(accu_max))
print("max:{:.4f}".format(max_accu))
print("min:{:.4f}".format(min_accu))

print("Повноцінність: {:.4f}".format(complete_max))
print("max:{:.4f}".format(max_c))
print("min:{:.4f}".format(min_c))

print("Однорідність: {:.4f}".format(homo_max))
print("max:{:.4f}".format(max_h))
print("min:{:.4f}".format(min_h))
```

Дев'яносто вісім моделей k-середніх були навчені на наборі даних 2 з використанням k=1-49 та двох мір відстані. Як базову метрику знову було використано евклідову distance. Для порівняння було використано метрику косинуса, оскільки вона має тенденцію добре працювати з текстом або розрідженими даними (навіть якщо дві точки знаходяться далеко за евклідовою відстанню, вони все одно можуть вважатися близькими за цією мірою відстані, якщо кут між двома точками вузький).

Стовпчик «combine», який включав тексти з «keywords», «title», «abstract», було використано як вхідні дані X, оскільки він показав найкращий загальний результат у нашому тестуванні.

```
k_range = 50
X_data = X_combine_pca

from sklearn.cluster import KMeans
kmeans_euclidean2 = [KMeans(n_clusters=k, random_state=42).fit(X_data) for k in range(1, k_range)]
labels_euclidean2 = [model.fit_predict(X_data) for model in kmeans_euclidean2]

inertias_euclidean2 = [model.inertia_ for model in kmeans_euclidean2]

k = 24
kmeans_euclidean224 = KMeans(n_clusters=k, random_state=42)
labels_euclidean224 = kmeans_euclidean224.fit_predict(X_data)

k_range2 = 50
X_icmla_df = X_combine_pca
try:
  clusters_cosine2 = [find_clusters(X_icmla_df, 'cosine', n_clusters=k)
            for k in range(1, k_range2)]

  centers_cosine2 = [clusters_cosine2[k][0] for k in range(0, k_range2-1)]

  labels_cosine2 = [clusters_cosine2[k][1] for k in range(0, k_range2-1)]

  inertias_cosine2 = [clusters_cosine2[k][2] for k in range(0, k_range2-1)]
except:
  print("Не вдалося згенерувати кластери. Спробуйте інший X_icmla_df або іншу міру відстані.")

clusters_cosine2c24 = find_clusters(X_icmla_df, 'cosine', 24)

centers_cosine2c24 = clusters_cosine2c24[0]
labels_cosine2c24 = clusters_cosine2c24[1]
inertias_cosine2c24 = clusters_cosine2c24[2]
```

Перша оцінка (evaluation)

Силуетний метод

Силуетний коефіцієнт, що варіюється від -1 (поганий) до 1 (хороший), вимірює компактність кластера, до якого належить точка, та відокремленість цієї точки від усіх інших кластерів (а оцінка силуету - це середній силуетний коефіцієнт для всіх точок у даних).

```
from sklearn.metrics import silhouette_score
silhouette_scores_euclidean2 = [silhouette_score(X_data, model.labels_)
                for model in kmeans_euclidean2[1:k_range]]

plt.figure(figsize=(10, 6))
plt.plot(range(2, k_range), silhouette_scores_euclidean2, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Силуетний показник", fontsize=14)
plt.xticks(np.arange(0, k_range, step=2))
plt.axis([0, k_range, 0.10, 0.20])
plt.annotate('Найвища кількість, k=36',
          xy=(36, silhouette_scores_euclidean2[34]),
          xytext=(0.75, 0.88),
          textcoords='figure fraction',
          fontsize=16,
          arrowprops=dict(facecolor='black', shrink=0.1)
         )
plt.annotate('k=24',
          xy=(24, silhouette_scores_euclidean2[22]),
          xytext=(0.5, 0.4),
          textcoords='figure fraction',
          fontsize=16,
          arrowprops=dict(facecolor='black', shrink=0.1)
         )
plt.annotate('k=21',
          xy=(21, silhouette_scores_euclidean2[19]),
          xytext=(0.5, 0.88),
          textcoords='figure fraction',
          fontsize=16,
          arrowprops=dict(facecolor='black', shrink=0.1)
         )
plt.show()

print("Силуетний показники (k=21): {:.04f}".format(silhouette_scores_euclidean2[19]))
print("Силуетний показники (k=24): {:.04f}".format(silhouette_scores_euclidean2[22]))
print("Силуетний показники (k=36): {:.04f}".format(silhouette_scores_euclidean2[34]))

from sklearn.metrics import silhouette_score
silhouette_scores_cosine2 = [silhouette_score(X_icmla_df, labels_cosine2[k]) for k in range(1, 49)]

plt.figure(figsize=(10, 6))
plt.plot(range(2, 50), silhouette_scores_cosine2, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Силуетний показник", fontsize=14)
plt.xticks(np.arange(0, k_range2, step=2))
plt.axis([0, 50, 0.03, 0.12])
plt.annotate('Найвища кількість, k=29',
          xy=(29, silhouette_scores_cosine2[27]),
          xytext=(0.68, 0.85),
          textcoords='figure fraction',
          fontsize=16,
          arrowprops=dict(facecolor='black', shrink=0.1)
         )
plt.annotate('k=24',
          xy=(24, silhouette_scores_cosine2[22]),
          xytext=(0.4, 0.2),
          textcoords='figure fraction',
          fontsize=16,
          arrowprops=dict(facecolor='black', shrink=0.1)
         )
plt.annotate('k=19',
          xy=(19, silhouette_scores_cosine2[17]),
          xytext=(0.3, 0.85),
          textcoords='figure fraction',
          fontsize=16,
          arrowprops=dict(facecolor='black', shrink=0.1)
         )
```

```
plt.show()
```

```
print("Силуетний показники (k=19): {:.04f}".format(silhouette_scores_cosine2[17]))
print("Силуетний показники (k=24): {:.04f}".format(silhouette_scores_cosine2[22]))
print("Силуетний показники (k=29): {:.04f}".format(silhouette_scores_cosine2[27]))
```

```
plot_silhouette2(X_data, 36, labels_euclidean2, labels_cosine2)
```

Друга оцінка (evaluation)

Оцінка повноти

Оцінка повноти вимірює ступінь належності всіх точок даного класу до одного кластеру (оцінка 1 означає ідеально повне маркування).

```
from  sklearn.metrics import completeness_score, homogeneity_score

ks = []
complete_euclideans = []
complete_cosines = []
homo_euclideans = []
homo_cosines = []

for k in range(1, 49, 2):
  comp_eu = completeness_score(y2, labels_euclidean2[k])
  comp_co = completeness_score(y2, labels_cosine2[k])
  homo_eu = homogeneity_score(y2, labels_euclidean2[k])
  homo_co = homogeneity_score(y2, labels_cosine2[k])

  ks.append(k+1)
  complete_euclideans.append(comp_eu)
  complete_cosines.append(comp_co)
  homo_euclideans.append(homo_eu)
  homo_cosines.append(homo_co)

df_scores2 = pd.DataFrame({"k":ks,
"completeness_euclidean":complete_euclideans,
"completeness_cosines":complete_cosines,
"homogeneity_euclideans":homo_euclideans,
"homogeneity_cosines":homo_cosines})

df_scores2
```

Загалом, оцінки показали, що кластери, створені евклідовими нейронними мережами, були більш повними та однорідними. Це узгоджується з маркуванням, де більше точок даних з одного сеансу, наприклад, нейронні мережі I і II, були згруповані разом за допомогою евклідової моделі, а більша кількість її кластерів містить лише точки даних з одного класу.

```
plot_completeness2(complete_euclideans, k_range=50, ymin=0.3, ymax=0.7, title='K-Means - Евклідовий')

plot_completeness2(complete_cosines, k_range=50, ymin=0.2, ymax=0.7, title='K-Means - Косинусова')
```

Третя оцінка (evaluation)

Показник однорідності

Показник однорідності вимірює ступінь, до якого всі кластери містять лише точки одного класу (оцінка 1 означає абсолютно однорідне маркування).

```
plot_homogeneity2(homo_euclideans, k_range=50, ymin=0, ymax=0.8, title='K-Means - Евклідовий')

plot_homogeneity2(homo_cosines, k_range=50, ymin=0, ymax=0.8, title='K-Means - Косинусова')
```

Зображення кластерів

Проекції t-SNE

На проекціях t-SNE обидві моделі показали добре відокремлені кластери при менших k, хоча зі збільшенням k додаткові кластери було важче ідентифікувати. При k=24 кластери, отримані за

допомогою евклідової моделі, були краще розділені, ніж за допомогою косинусоїдальної.

plot_tsne(X_icmla_df,k=2,label=labels_euclidean2[1], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=3,label=labels_euclidean2[2], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=4,label=labels_euclidean2[3], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=5,label=labels_euclidean2[4], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=6,label=labels_euclidean2[5], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=7,label=labels_euclidean2[6], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=8,label=labels_euclidean2[7], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=9,label=labels_euclidean2[8], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=24,label=labels_euclidean2[23], title='K-Means - Евклідова')

plot_tsne(X_icmla_df,k=24,label=labels_cosine2[23], title='K-Means - Косинусова')

plot_tsne(X_icmla_df,k=3,label=labels_cosine2[2], title='K-Means - Косинусова')

plot_tsne(X_icmla_df,k=4,label=labels_cosine2[3], title='K-Means - Косинусова')

plot_tsne(X_icmla_df,k=5,label=labels_cosine2[4], title='K-Means - Косинусова')

plot_tsne(X_icmla_df,k=6,label=labels_cosine2[5], title='K-Means - Косинусова')

plot_tsne(X_icmla_df,k=7,label=labels_cosine2[6], title='K-Means - Косинусова')

plot_tsne(X_icmla_df,k=8,label=labels_cosine2[7], title='K-Means - Косинусова')

Таблиці маркування

У наведених нижче таблицях фреймів даних видно, що більше точок даних з одного сеансу, таких як нейронні мережі I і II, були згруповані разом за евклідовим методом, і більша кількість кластерів містить лише точки даних з одного класу.

pd.set_option('display.max_rows', 105)

icmla_df_label24e = pd.DataFrame({"label":labels_euclidean2[23], "session":y2_string})

icmla_df_label24e.sort_values(by='label')

icmla_df_label24c = pd.DataFrame({"label":labels_cosine2[23], "session":y2_string})

icmla_df_label24c.sort_values(by='label')

---

3. Ієрархічна кластеризація

3.1 Travel Review - Набір даних

Функції

def calc_and_plot_silhouette2(X, labels, k_range, ymin, ymax, title=""):

  from sklearn.metrics import silhouette_score

  silhouette_scores = [silhouette_score(X, labels[k]) for k in range(1, k_range-1)]

```
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), silhouette_scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Силуетний показник", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.show()

  return silhouette_scores

def plot_silhouette2(X, labels, k1, k2):
  from sklearn.metrics import silhouette_samples
  from sklearn.metrics import silhouette_score

  import matplotlib.cm as cm
  range_n_clusters = [k1,k2]

  for n_clusters in range_n_clusters:
      figure, ax = plt.subplots(1, 1)
      figure.set_size_inches(10, 10)

      ax.set_xlim([-0.1, 1])
      ax.set_ylim([0, len(X) + (n_clusters + 1) * 10])

      cluster_labels = labels[n_clusters]

      silhouette_avg = silhouette_score(X, cluster_labels)

      sample_silhouette_values = silhouette_samples(X, cluster_labels)

      y_lower = 10
      for i in range(n_clusters):
          ith_cluster_silhouette_values = \
              sample_silhouette_values[cluster_labels == i]
          ith_cluster_silhouette_values.sort()

          size_cluster_i = ith_cluster_silhouette_values.shape[0]
          y_upper = y_lower + size_cluster_i

          color = cm.Spectral(float(i) / n_clusters)
          ax.fill_betweenx(np.arange(y_lower, y_upper),
                           0, ith_cluster_silhouette_values,
                           facecolor=color, edgecolor=color, alpha=0.7)

          ax.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

          y_lower = y_upper + 10

      ax.set_xlabel("Значення коефіцієнта силуету")
      ax.set_ylabel("Назва кластуру")

      ax.axvline(x=silhouette_avg, color="red", linestyle="--")

      ax.set_yticks([])
      ax.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

      plt.suptitle(("Силуетний аналіз для агломеративної кластеризації "
                    "с k = %d" % n_clusters),
                   fontsize=14, fontweight='bold')

      plt.show()

def plot_calinski(scores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка Калинського-Харабаша", fontsize=14)
```

```
  plt.axis([0, k_range, ymin, ymax])


def plot_davies(scores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(range(2, k_range), scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка Девіса-Болдіна", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])


def plot_tsne(X,k,label,title=None):

  from sklearn.manifold import TSNE
  tsne = TSNE(random_state=42)

  X_tsne = tsne.fit_transform(X)

  plt.figure(figsize=(10,8))
  plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=label,
            edgecolor='none', alpha=0.7, s=40,
            cmap=plt.cm.get_cmap('nipy_spectral', k))
  plt.colorbar()
  title = 'Проекція t-SNE, ' + title + ', k=' + str(k)
  plt.title(title)
```

Тренуємо Ієрархічну кластеризацію

Понад 260 моделей було навчено на наборі даних 1 з використанням k=1-29, трьох мір подібності та трьох метрик відстані. Косинусована схожість враховує кут між двома векторами, тоді як евклідова (пряма лінія) та манхеттенська (distance у блоках) - distance між двома точками. При вимірюванні відстані між точкою в кластері та точками в інших кластерах використовується найменша distance, при вимірюванні середньої відстані - середня, а при вимірюванні повної відстані - найбільша.

```
from sklearn.cluster import AgglomerativeClustering

k_range3 = 30
X1_review_df = X_norm

affinities = ['euclidean','manhattan','cosine']
linkages = ['single','complete','average']

names = []
agglo_models = dict()

for affin in affinities:
  for linka in linkages:
    agglo_cluster = [AgglomerativeClustering(n_clusters=k, metric=affin, linkage=linka) for k in range(1, k_range3)]

    labels_agglo = [model.fit_predict(X1_review_df) for model in agglo_cluster]

    name = affin + '-' + linka
    names.append(name)
    print(name)

    agglo_models[name] = [agglo_cluster, labels_agglo]

print(agglo_models['cosine-complete'][0][4])
print(agglo_models['cosine-complete'][1][4])
```

Перша оцінка (evaluation)

Силуетний метод

Силуетний коефіцієнт, що варіюється від -1 (поганий) до 1 (хороший), вимірює компактність кластера, до якого належить точка, та відокремленість цієї точки від усіх інших кластерів (а оцінка силуету - це середній силуетний коефіцієнт для всіх точок у даних).

```
ss_eu_single = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['euclidean-single'][1],
```

```
k_range=k_range3, ymin=-0.3, ymax=0.5, title="Агломеративна кластеризація: Евклідова distance - Одинарне з\'єднання")

print("Силуетний показник (k=4): {:0.4f}".format(ss_eu_single[2]))

print("Силуетний показник (k=5): {:0.4f}".format(ss_eu_single[3]))

print("Силуетний показник (k=6): {:0.4f}".format(ss_eu_single[4]))

print("Силуетний показник (k=7): {:0.4f}".format(ss_eu_single[5]))


ss_eu_average = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['euclidean-average'][1],

k_range=k_range3, ymin=0, ymax=0.5, title="Агломеративна кластеризація: Евклідова distance - Середній зв\'язок")

print("Силуетний показник (k=4): {:0.4f}".format(ss_eu_average[2]))

print("Силуетний показник (k=9): {:0.4f}".format(ss_eu_average[7]))

print("Силуетний показник (k=10): {:0.4f}".format(ss_eu_average[8]))

print("Силуетний показник (k=11): {:0.4f}".format(ss_eu_average[9]))


ss_eu_complete = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['euclidean-complete'][1],

k_range=k_range3, ymin=0, ymax=0.25, title="Агломеративна кластеризація: Евклідова distance - Повний зв\'язок")


print("Силуетний показник (k=5): {:0.4f}".format(ss_eu_complete[3]))


ss_man_single = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['manhattan-single'][1],

k_range=k_range3, ymin=-0.3, ymax=0.5, title="Агломеративна кластеризація: Мангеттен distance - Одинарне з\'єднання")

print("Силуетний показник (k=4): {:0.4f}".format(ss_man_single[2]))

print("Силуетний показник (k=5): {:0.4f}".format(ss_man_single[3]))

print("Силуетний показник (k=6): {:0.4f}".format(ss_man_single[4]))

print("Силуетний показник (k=7): {:0.4f}".format(ss_man_single[5]))


ss_man_average = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['manhattan-average'][1],

k_range=k_range3, ymin=0, ymax=0.5, title="Агломеративна кластеризація: Мангеттен distance - Середній зв\'язок")

print("Силуетний показник (k=4): {:0.4f}".format(ss_man_average[2]))

print("Силуетний показник (k=9): {:0.4f}".format(ss_man_average[7]))

print("Силуетний показник (k=10): {:0.4f}".format(ss_man_average[8]))

print("Силуетний показник (k=11): {:0.4f}".format(ss_man_average[9]))


ss_man_complete = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['manhattan-complete'][1],

k_range=k_range3, ymin=0, ymax=0.3, title="Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок")

print("Силуетний показник (k=4): {:0.4f}".format(ss_man_complete[2]))

print("Силуетний показник (k=7): {:0.4f}".format(ss_man_complete[5]))


ss_cos_single = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['cosine-single'][1],

k_range=k_range3, ymin=-0.3, ymax=0.5, title="Агломеративна кластеризація: Косинусова distance - Одинарне з\'єднання")


print("Силуетний показник (k=4): {:0.4f}".format(ss_cos_single[2]))

print("Силуетний показник (k=5): {:0.4f}".format(ss_cos_single[3]))

print("Силуетний показник (k=6): {:0.4f}".format(ss_cos_single[4]))


ss_cos_average = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['cosine-average'][1],

k_range=k_range3, ymin=0, ymax=0.3, title="Агломеративна кластеризація: Косинусова distance - Середній зв\'язок")

print("Силуетний показник (k=5): {:0.4f}".format(ss_cos_average[3]))

print("Силуетний показник (k=6): {:0.4f}".format(ss_cos_average[4]))

print("Силуетний показник (k=7): {:0.4f}".format(ss_cos_average[5]))

print("Силуетний показник (k=8): {:0.4f}".format(ss_cos_average[6]))
```

```
ss_cos_complete = calc_and_plot_silhouette2(X1_review_df, labels=agglo_models['cosine-complete'][1],
                k_range=k_range3, ymin=0, ymax=0.2, title="Агломеративна кластеризація: Косинусова distance - Повний зв\'язок")
print("Силуетний показник (k=7): {:0.4f}".format(ss_cos_complete[5]))
print("Силуетний показник (k=8): {:0.4f}".format(ss_cos_complete[6]))
```

На жаль, всі моделі отримали низькі оцінки силуету (менше 0,3 для k>=3). Графіки коефіцієнта силуету також виглядають не дуже добре.

```
plot_silhouette2(X1_review_df, labels=agglo_models['cosine-complete'][1], k1=7, k2=8)

plot_silhouette2(X1_review_df, labels=agglo_models['manhattan-complete'][1], k1=3, k2=7)

plot_silhouette2(X1_review_df, labels=agglo_models['euclidean-complete'][1], k1=3, k2=7)

plot_silhouette2(X1_review_df, labels=agglo_models['manhattan-average'][1], k1=7, k2=8)

plot_silhouette2(X1_review_df, labels=agglo_models['cosine-average'][1], k1=5, k2=6)
```

Друга оцінка (evaluation)

Індекс Девіса-Болдіна та індекс Калінкі-Харабаша

Оцінка Девіса-Болдіна вимірює середню «схожість» між кластерами, яка визначається відстанню між кластерами з урахуванням розміру кластерів (нижчий показник, близький до 0, вказує на кращу відокремленість між кластерами). Показник Калінкі-Харабаша вимірює співвідношення між внутрішньою та міжкластерною дисперсіями (вищий показник означає щільні та добре відокремлені кластери).

```
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

db_agglo = dict()
ch_agglo = dict()
dbscores = []
chscores = []
ks =[]

for name in names:
  for k in range(1, k_range3-1):
    dbscore = davies_bouldin_score(X1_review_df, agglo_models[name][1][k])
    chscore = calinski_harabasz_score(X1_review_df, agglo_models[name][1][k])
    dbscores.append(dbscore)
    chscores.append(chscore)

  db_agglo[name] = dbscores
  ch_agglo[name] = chscores
  dbscores = []
  chscores = []

df_chscores = pd.DataFrame({"k":np.arange(2,30), "cosine-single":ch_agglo['cosine-single'],
                "cosine-average":ch_agglo['cosine-average'],
                "cosine-complete":ch_agglo['cosine-complete'],
                "euclidean-single":ch_agglo['euclidean-single'],
                "euclidean-average":ch_agglo['euclidean-average'],
                "euclidean-complete":ch_agglo['euclidean-complete'],
                "manhattan-single":ch_agglo['manhattan-single'],
                "manhattan-average":ch_agglo['manhattan-average'],
                "manhattan-complete":ch_agglo['manhattan-complete'],
                })
df_chscores.head(14)
```

Моделі з повними зв'язками дають більш щільні та краще розділені кластери, ніж середні або з одним зв'язком, про що свідчать вищі показники Калінські для різних k. У поєднанні з моделлю повного зв'язку всі три міри подібності працювали добре. Загалом, показники

Калінські зменшувалися зі збільшенням k, але манхеттенські моделі виявилися кращими за нижчих k, тоді як косинуси були кращими за вищих k (з евклідовими десь посередині між ними).

```
df_dbscores = pd.DataFrame({"k":np.arange(2,30), "cosine-single":db_agglo['cosine-single'],
                "cosine-average":db_agglo['cosine-average'],
                "cosine-complete":db_agglo['cosine-complete'],
                "euclidean-single":db_agglo['euclidean-single'],
                "euclidean-average":db_agglo['euclidean-average'],
                "euclidean-complete":db_agglo['euclidean-complete'],
                "manhattan-single":db_agglo['manhattan-single'],
                "manhattan-average":db_agglo['manhattan-average'],
                "manhattan-complete":db_agglo['manhattan-complete'],
                })
df_dbscores.head(14)
```

Оцінки Девіса-Болдіна були досить хорошими для всіх моделей, причому моделі з однією та середньою ланкою давали дещо кращі оцінки, ніж моделі з повною ланкою. У парі з одинарною ланкою манхеттенська та евклідова моделі показали кращі результати, ніж косинусоїдальна.

```
plot_calinski(scores=ch_agglo['cosine-single'], k_range=k_range3, ymin=0, ymax=5, title='Агломеративна кластеризація: Косинусова distance - Одинарне з\'єднання')

plot_calinski(scores=ch_agglo['cosine-average'], k_range=k_range3, ymin=0, ymax=120, title='Агломеративна кластеризація: Косинусова distance - Середній зв\'язок')

plot_calinski(scores=ch_agglo['cosine-complete'], k_range=k_range3, ymin=40, ymax=200, title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_calinski(scores=ch_agglo['euclidean-single'], k_range=k_range3, ymin=0, ymax=12, title='Агломеративна кластеризація: Евклідова distance - Одинарне з\'єднання')

plot_calinski(scores=ch_agglo['euclidean-average'], k_range=k_range3, ymin=0, ymax=65, title='Агломеративна кластеризація: Евклідова distance - Середній зв\'язок')

plot_calinski(scores=ch_agglo['euclidean-complete'], k_range=k_range3, ymin=40, ymax=300, title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_calinski(scores=ch_agglo['manhattan-single'], k_range=k_range3, ymin=0, ymax=7, title='Агломеративна кластеризація: Мангеттен distance - Одинарне з\'єднання')

plot_calinski(scores=ch_agglo['manhattan-average'], k_range=k_range3, ymin=0, ymax=65, title='Агломеративна кластеризація: Мангеттен distance - Середній зв\'язок')

plot_calinski(scores=ch_agglo['manhattan-complete'], k_range=k_range3, ymin=40, ymax=350, title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_davies(scores=db_agglo['manhattan-single'], k_range=k_range3, ymin=0.4, ymax=0.8, title='Агломеративна кластеризація: Мангеттен distance - Одинарне з\'єднання')

plot_davies(scores=db_agglo['manhattan-average'], k_range=k_range3, ymin=0.4, ymax=1.5, title='Агломеративна кластеризація: Мангеттен distance - Середній зв\'язок')

plot_davies(scores=db_agglo['manhattan-complete'], k_range=k_range3, ymin=1.2, ymax=2.4, title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_davies(scores=db_agglo['euclidean-single'], k_range=k_range3, ymin=0.5, ymax=0.75, title='Агломеративна кластеризація: Евклідова distance - Одинарне з\'єднання')

plot_davies(scores=db_agglo['euclidean-average'], k_range=k_range3, ymin=0.4, ymax=1.4, title='Агломеративна кластеризація: Евклідова distance - Середній зв\'язок')
```

plot_davies(scores=db_agglo['euclidean-complete'], k_range=k_range3, ymin=1.2, ymax=2.2, title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_davies(scores=db_agglo['cosine-single'], k_range=k_range3, ymin=0.4, ymax=0.9, title='Агломеративна кластеризація: Косинусова distance - Одинарне з\'єднання')

plot_davies(scores=db_agglo['cosine-average'], k_range=k_range3, ymin=1.2, ymax=1.5, title='Агломеративна кластеризація: Косинусова distance - Середній зв\'язок')

plot_davies(scores=db_agglo['cosine-complete'], k_range=k_range3, ymin=1.5, ymax=2.2, title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

Зображення кластерів

Проекції t-SNE

На проекціях t-SNE як Манхеттенська, так і Евклідова моделі з повними зв'язками показали добре відокремлені кластери при k=3, але не давали хороших кластерів при збільшенні k. При k=7 косинусоїдальна модель з повними зв'язками показала значно краще розділені кластери, ніж евклідова чи манхеттенська. Однак модель з одним і повним зв'язком не дала ні добре збалансованих, ні добре відокремлених кластерів.

plot_tsne(X1_review_df,k=2,label=agglo_models['manhattan-complete'][1][1], title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=3,label=agglo_models['manhattan-complete'][1][2], title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=4,label=agglo_models['manhattan-complete'][1][3], title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=5,label=agglo_models['manhattan-complete'][1][4], title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=6,label=agglo_models['manhattan-complete'][1][5], title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=7,label=agglo_models['manhattan-complete'][1][6], title='Агломеративна кластеризація: Мангеттен distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=2,label=agglo_models['euclidean-complete'][1][1], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=3,label=agglo_models['euclidean-complete'][1][2], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=4,label=agglo_models['euclidean-complete'][1][3], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=5,label=agglo_models['euclidean-complete'][1][4], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=6,label=agglo_models['euclidean-complete'][1][5], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=7,label=agglo_models['euclidean-complete'][1][6], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=8,label=agglo_models['euclidean-complete'][1][7], title='Агломеративна кластеризація: Евклідова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=2,label=agglo_models['cosine-complete'][1][1], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

```
plot_tsne(X1_review_df,k=3,label=agglo_models['cosine-complete'][1][2], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=4,label=agglo_models['cosine-complete'][1][3], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=5,label=agglo_models['cosine-complete'][1][4], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=6,label=agglo_models['cosine-complete'][1][5], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=7,label=agglo_models['cosine-complete'][1][6], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=8,label=agglo_models['cosine-complete'][1][7], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=9,label=agglo_models['cosine-complete'][1][8], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=10,label=agglo_models['cosine-complete'][1][9], title='Агломеративна кластеризація: Косинусова distance - Повний зв\'язок')

plot_tsne(X1_review_df,k=4,label=agglo_models['manhattan-average'][1][3], title='Агломеративна кластеризація: Мангеттен distance - Середній зв\'язок')

plot_tsne(X1_review_df,k=7,label=agglo_models['manhattan-average'][1][6], title='Агломеративна кластеризація: Мангеттен distance - Середній зв\'язок')

plot_tsne(X1_review_df,k=7,label=agglo_models['cosine-average'][1][6], title='Агломеративна кластеризація: Косинусова distance - Середній зв\'язок')

plot_tsne(X1_review_df,k=7,label=agglo_models['cosine-single'][1][6], title='Агломеративна кластеризація: Косинусова distance - Одинарне з\'єднання')

plot_tsne(X1_review_df,k=5,label=agglo_models['manhattan-single'][1][4], title='Агломеративна кластеризація: Мангеттен distance - Одинарне з\'єднання')

plot_tsne(X1_review_df,k=6,label=agglo_models['euclidean-single'][1][5], title='Агломеративна кластеризація: Евклідова distance - Одинарне з\'єднання')

plot_tsne(X1_review_df,k=6,label=agglo_models['cosine-single'][1][5], title='Агломеративна кластеризація: Косинусова distance - Одинарне з\'єднання')

plot_tsne(X1_review_df,k=5,label=agglo_models['cosine-average'][1][4], title='Агломеративна кластеризація: Косинусова distance - Середній зв\'язок')

## 3.2 ICMLA - Набір даних

### Функції

def plot_completeness2(scores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
  plt.title(title)
  plt.plot(np.arange(2, k_range), scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка завершеності", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.show()

def plot_homogeneity2(scores, k_range, ymin, ymax, title=''):
  plt.figure(figsize=(10, 6))
```

```
  plt.title(title)
  plt.plot(np.arange(2, k_range), scores, "bo-")
  plt.xlabel("$k$", fontsize=14)
  plt.ylabel("Оцінка однорідності", fontsize=14)
  plt.axis([0, k_range, ymin, ymax])
  plt.show()

def plot_dendogram(X,method,metric,labels=y2_string,title=''):
  from scipy.cluster.hierarchy import dendrogram, fcluster, linkage

  linkage_matrix = linkage(X, method=method, metric=metric)

  figure, ax = plt.subplots(figsize=(15, 20))
  ax = dendrogram(linkage_matrix, orientation="left", labels=y2_string)

  plt.tick_params(\
     axis= 'x',
     which='both',
     bottom='off',
     top='off',
     labelbottom='off',
     labelsize=12
     )
  plt.tick_params(\
     axis= 'y',
     labelsize=12
     )
  plt.title(title, fontsize=16)
  plt.tight_layout()

def plot_tsne(X,k,label,title=None):

  from sklearn.manifold import TSNE
  tsne = TSNE(random_state=43)
  X_tsne = tsne.fit_transform(X)

  plt.figure(figsize=(10,8))
  plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=label,
           edgecolor='none', alpha=0.7, s=40,
           cmap=plt.cm.get_cmap('nipy_spectral', k))
  plt.colorbar()
  title = 'Проекція t-SNE, ' + title + ', k=' + str(k)
  plt.title(title)

### Тренуємо Ієрархічну кластеризацію
```

Використовуючи scipy Hierarchy Clustering15, понад 260 моделей було навчено на наборі даних 2 з використанням k=1-49, трьох мір подібності (евклідова, сіті-блок та косинусна) та трьох метрик відстані (однозв'язкова, середньозв'язкова та повно-зв'язкова). Після отримання золотих кластерів отримані моделі були оцінені за допомогою зовнішніх мір: оцінки повноти та однорідності.

```
from scipy.cluster.hierarchy import dendrogram, fcluster, linkage

k_range4 = 50
X2_icmla_df3 = X_combine_pca

metrics = ['euclidean','cityblock','cosine']
methods = ['single','complete','average']

names = []
agglo_models2 = dict()
labels = dict()

for metr in metrics:
  for meth in methods:
    linkage_matrix = linkage(X2_icmla_df3, method=meth, metric=metr)

    for k in range(2, k_range4):
      label = fcluster(linkage_matrix,t=k, criterion='maxclust')
```

```
    labels[k] = label


  name = metr + '-' + meth
  names.append(name)
  print(name)


  agglo_models2[name] = [labels]
  labels= dict()


agglo_models2['cosine-average'][0][24]
```

Перша оцінка (evaluation)

Оцінка повноти

Оцінка повноти вимірює ступінь належності всіх точок даного класу до одного кластеру (оцінка 1 означає ідеально повне маркування).

```
from  sklearn.metrics import completeness_score, homogeneity_score


homo_agglo = dict()
comp_agglo = dict()
comps = []
homos = []


for name in names:
 for k in range(2, k_range4):
  comp = completeness_score(y2, agglo_models2[name][0][k])
  homo = homogeneity_score(y2, agglo_models2[name][0][k])
  comps.append(comp)
  homos.append(homo)
 comp_agglo[name] = comps
 homo_agglo[name] = homos
 comps = []
 homos = []
```

```
df_comps = pd.DataFrame({"k":np.arange(2,50), "cosine-single":comp_agglo['cosine-single'],
                         "cosine-average":comp_agglo['cosine-average'],
                         "cosine-complete":comp_agglo['cosine-complete'],
                         "euclidean-single":comp_agglo['euclidean-single'],
                         "euclidean-average":comp_agglo['euclidean-average'],
                         "euclidean-complete":comp_agglo['euclidean-complete'],
                         "cityblock-single":comp_agglo['cityblock-single'],
                         "cityblock-average":comp_agglo['cityblock-average'],
                         "cityblock-complete":comp_agglo['cityblock-complete'],
                         })
df_comps.head(30)
```

Загалом, одноланкові моделі отримали вищі оцінки повноти, ніж середні та повноланкові моделі за нижчих k, але оцінки зростали і почали зближуватися зі збільшенням k, і всі моделі отримали однакові оцінки за вищих k. При k=24 моделі косинусоїдального середнього, середнього по міських кварталах та повної міської ланки отримали найвищий бал - 0,61, за ними слідує модель з одним міським кварталом (0,60).

```
df_homos = pd.DataFrame({"k":np.arange(2,50), "cosine-single":homo_agglo['cosine-single'],
                         "cosine-average":homo_agglo['cosine-average'],
                         "cosine-complete":homo_agglo['cosine-complete'],
                         "euclidean-single":homo_agglo['euclidean-single'],
                         "euclidean-average":homo_agglo['euclidean-average'],
                         "euclidean-complete":homo_agglo['euclidean-complete'],
                         "cityblock-single":homo_agglo['cityblock-single'],
                         "cityblock-average":homo_agglo['cityblock-average'],
                         "cityblock-complete":homo_agglo['cityblock-complete'],
                         })
df_homos.head(30)


plot_completeness2(comp_agglo['euclidean-single'], k_range=k_range4, ymin=0.55, ymax=0.7, title='Hierarchical Clustering: Евклідова distance - Одинарне з\'єднання')
```

plot_completeness2(comp_agglo['euclidean-average'], k_range=k_range4, ymin=0.45, ymax=0.7, title='Hierarchical Clustering: Евклідова distance - Середній зв\'язок')

plot_completeness2(comp_agglo['euclidean-complete'], k_range=k_range4, ymin=0.35, ymax=0.7, title='Hierarchical Clustering: Евклідова distance - Повний зв\'язок')

plot_completeness2(comp_agglo['cityblock-single'], k_range=k_range4, ymin=0.55, ymax=0.7, title='Hierarchical Clustering: Cityblock distance - Одинарне з\'єднання')

plot_completeness2(comp_agglo['cityblock-average'], k_range=k_range4, ymin=0.4, ymax=0.7, title='Hierarchical Clustering: Cityblock distance - Середній зв\'язок')

plot_completeness2(comp_agglo['cityblock-complete'], k_range=k_range4, ymin=0.3, ymax=0.7, title='Hierarchical Clustering: Cityblock distance - Повний зв\'язок')

plot_completeness2(comp_agglo['cosine-single'], k_range=k_range4, ymin=0.45, ymax=0.7, title='Hierarchical Clustering: Косинусова distance - Одинарне з\'єднання')

plot_completeness2(comp_agglo['cosine-average'], k_range=k_range4, ymin=0.4, ymax=0.7, title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_completeness2(comp_agglo['cosine-complete'], k_range=k_range4, ymin=0.25, ymax=0.7, title='Hierarchical Clustering: Косинусова distance - Повний зв\'язок')

Друга оцінка (evaluation)

Показник однорідності

Показник однорідності вимірює ступінь, до якого всі кластери містять лише точки одного класу (оцінка 1 означає абсолютно однорідне маркування).

plot_homogeneity2(homo_agglo['euclidean-single'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Евклідова distance - Одинарне з\'єднання')

plot_homogeneity2(homo_agglo['euclidean-average'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Евклідова distance - Середній зв\'язок')

plot_homogeneity2(homo_agglo['euclidean-complete'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Евклідова distance - Повний зв\'язок')

plot_homogeneity2(homo_agglo['cityblock-single'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Cityblock distance - Одинарне з\'єднання')

plot_homogeneity2(homo_agglo['cityblock-average'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Cityblock distance - Середній зв\'язок')

plot_homogeneity2(homo_agglo['cityblock-complete'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Cityblock distance - Повний зв\'язок')

plot_homogeneity2(homo_agglo['cosine-single'], k_range=k_range4, ymin=-0, ymax=0.8, title='Hierarchical Clustering: Косинусова distance - Одинарне з\'єднання')

plot_homogeneity2(homo_agglo['cosine-average'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_homogeneity2(homo_agglo['cosine-complete'], k_range=k_range4, ymin=0, ymax=0.8, title='Hierarchical Clustering: Косинусова distance - Повний зв\'язок')

Оцінки однорідності для одноланкових моделей були здебільшого нижчими, ніж для середніх та повних ланок. Однак оцінки зростали зі збільшенням k, і при k=24 найвищу оцінку мала модель Cityblock-complete (0,60), за нею йшли моделі Косинусова-середня (0,59) та Евклідова-complete (0,58).

Зображення кластерів

Прогнози t-SNE

За нижчих k середня та повна ланка показали більш збалансовані кластери, ніж одноланкова. При k=24 всі моделі дали досить добре розділені, але різні кластери.

plot_tsne(X2_icmla_df3,k=3,label=agglo_models2['cityblock-single'][0][3], title='Hierarchical Clustering: Cityblock distance - Одинарне з\'єднання')

plot_tsne(X2_icmla_df3,k=3,label=agglo_models2['euclidean-single'][0][3], title='Hierarchical Clustering: Евклідова distance - Одинарне з\'єднання')

plot_tsne(X2_icmla_df3,k=3,label=agglo_models2['cosine-single'][0][3], title='Hierarchical Clustering: Косинусова distance - Одинарне з\'єднання')

plot_tsne(X2_icmla_df3,k=3,label=agglo_models2['cosine-complete'][0][3], title='Hierarchical Clustering: Косинусова distance - Повний зв\'язок')

plot_tsne(X2_icmla_df3,k=3,label=agglo_models2['cosine-average'][0][3], title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=4,label=agglo_models2['cosine-average'][0][4], title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=5,label=agglo_models2['cosine-average'][0][5], title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=6,label=agglo_models2['cosine-average'][0][6], title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['euclidean-single'][0][24], title='Hierarchical Clustering: Евклідова distance - Одинарне з\'єднання')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['euclidean-average'][0][24], title='Hierarchical Clustering: Евклідова distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['euclidean-complete'][0][24], title='Hierarchical Clustering: Евклідова distance - Повний зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['euclidean-single'][0][24], title='Hierarchical Clustering: Евклідова distance - Одинарне з\'єднання')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['cosine-average'][0][24], title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['cosine-complete'][0][24], title='Hierarchical Clustering: Косинусова distance - Повний зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['cosine-single'][0][24], title='Hierarchical Clustering: Косинусова distance - Одинарне з\'єднання')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['cityblock-average'][0][24], title='Hierarchical Clustering: Cityblock distance - Середній зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['cityblock-complete'][0][24], title='Hierarchical Clustering: Cityblock distance - Повний зв\'язок')

plot_tsne(X2_icmla_df3,k=24,label=agglo_models2['cityblock-single'][0][24], title='Hierarchical Clustering: Cityblock distance - Одинарне з\'єднання')

Дендограма

На дендограмі середній та повний зв\'язок також показав краще групування, ніж одиночний зв\'язок, оскільки більше точок з одного сеансу, таких як Виділення ознак та Відбір, були згруповані разом (Рисунок 5.2.2j-l). З трьох мір схожості косинус, як видається, краще відображається на графіку, ніж евклідова або сіті-блок. Замість того, щоб вимірювати фактичну distance, косинус вимірює кут між точками, і для наших розріджених даних він працює досить добре.

```
plot_dendogram(X2_icmla_df3,method='average',metric='cosine',labels=y2_string,title='Hierarchical Clustering: Косинусова distance - Середній зв\'язок')

plot_dendogram(X2_icmla_df3,method='single',metric='cosine',labels=y2_string,title='Hierarchical Clustering: Косинусова distance - Одинарне з\'єднання')

plot_dendogram(X2_icmla_df3,method='complete',metric='cosine',labels=y2_string,title='Hierarchical Clustering: Косинусова distance - Повний зв\'язок')

plot_dendogram(X2_icmla_df3,method='average',metric='euclidean',labels=y2_string,title='Hierarchical Clustering: Евклідова distance - Середній зв\'язок')

plot_dendogram(X2_icmla_df3,method='complete',metric='euclidean',labels=y2_string,title='Hierarchical Clustering: Евклідова distance - Повний зв\'язок')

plot_dendogram(X2_icmla_df3,method='complete',metric='cityblock',labels=y2_string,title='Hierarchical Clustering: Cityblock distance - Повний зв\'язок')

plot_dendogram(X2_icmla_df3,method='average',metric='cityblock',labels=y2_string,title='Hierarchical Clustering: Cityblock distance - Середній зв\'язок')
```

Таблиці маркування

Нижче наведено таблицю міток для запропонованого косинусоїдального середнього з k=24.

```
pd.set_option('display.max_rows', 105)

icmla_df_label24hca = pd.DataFrame({"label":agglo_models2['cosine-average'][0][24], "session":y2_string})

icmla_df_label24hca.sort_values(by='label')


---

# 4. Кластеризація DBSCAN


## Travel Review - Набір даних


### Функції


def dbscan_grid(X, epsilons, min_samples, metrics, high_range):

 from sklearn.cluster import DBSCAN

 from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score


 dbscore_min = 999999

 chscore_max = 0


 dbscores = []

 chscores = []

 x_epsilons = []

 x_min_samples = []

 nclusters = []


 for eps in epsilons:

  for min_sample in min_samples:

   for metric in metrics:

    dbscan = DBSCAN(eps=eps, min_samples=min_sample, metric=metric)

    dbscan.fit(X)

    labels = dbscan.labels_

    num_cluster = len(np.unique(labels))


    if num_cluster>1  and num_cluster<high_range:

     dbscore = davies_bouldin_score(X, labels)
```

```
        chscore = calinski_harabasz_score(X, labels)

        if dbscore<dbscore_min:
          dbscore_min = dbscore
          eps_min = eps
          sample_min = min_sample
          metric_min = metric
          num_cluster_min = len(np.unique(labels))
          chscore_min = chscore

        if chscore>chscore_max:
          chscore_max = chscore
          eps_max = eps
          sample_max = min_sample
          metric_max = metric
          num_cluster_max = len(np.unique(labels))
          dbscore_max = dbscore

      if dbscore_min!=999999:
        dbscores.append(dbscore)

      if chscore_max!=0:
        chscores.append(chscore)
        x_epsilons.append(eps)
        x_min_samples.append(min_sample)
        nclusters.append(num_cluster)

  bestdb = [dbscore_min, eps_min, sample_min, metric_min, num_cluster_min, chscore_min, x_epsilons, x_min_samples, nclusters]
  bestch = [chscore_max, eps_max, sample_max, metric_max, num_cluster_max, dbscore_max, x_epsilons, x_min_samples, nclusters]

  return bestdb, bestch, dbscores, chscores

def plot_tsne(X,k,label,title=None):
  from sklearn.manifold import TSNE
  tsne = TSNE(random_state=42)

  X_tsne = tsne.fit_transform(X)

  plt.figure(figsize=(10,8))
  plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=label,
          edgecolor='none', alpha=0.7, s=40,
          cmap=plt.cm.get_cmap('nipy_spectral', k))
  plt.colorbar()
  title = 'Проекція t-SNE, ' + title + ', k=' + str(k)
  plt.title(title)

### Тренуємо кластеризацію DBSCAN
```

За допомогою алгоритму sklearn DBSCAN на наборі даних 1 було навчено понад 100 моделей з використанням трьох мір відстані (евклідова, мангеттен і косинус) та різних значень епсилонів (0,01-5) і мінімальних вибірок (2-10). Епсилон задає максимальний радіус околиці, тоді як мінімальні вибірки задають мінімальну кількість точок в околиці для того, щоб точка вважалася основною.

```
try:
  epsilons = np.arange(0.01,5,0.01)
  min_samples = range(2,11,1)
  metrics = ['euclidean', 'cityblock', 'cosine']

  bestdb1, bestch1, dbscores1, chscores1 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)

  print("Найкращі параметри на основі dbscore:")
  print("Епсілон: {:.2f}".format(bestdb1[1]))
  print("Мінімальна к-сть зразків:", bestdb1[2])
  print("Метрика:", bestdb1[3])
  print("dbscore: {:.4f}".format(bestdb1[0]))
  print("chscore: {:.4f}".format(bestdb1[5]))
  print("Кількість знайдених кластерів:", bestdb1[4])
```

```
  print("\nНайкращі параметри на основі chscore:")
  print("Епсілон: {:.2f}".format(bestch1[1]))
  print("Мінімальна к-сть зразків:", bestch1[2])
  print("Метрика:", bestch1[3])
  print("chscore: {:.4f}".format(bestch1[0]))
  print("dbscore: {:.4f}".format(bestch1[5]))
  print("Кількість знайдених кластерів:", bestch1[4])
except:
  print("Не вдалося знайти > 1 кластерів. Будь ласка, спробуйте різні комбінації epsilon або min_samples.")


from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan1 = DBSCAN(eps=bestch1[1], min_samples=bestch1[2], metric=bestch1[3])
dbscan1.fit(X_norm)
labels_db1 = dbscan1.labels_
num_cluster_db1 = len(np.unique(labels_db1))

dbscore1 = davies_bouldin_score(X_norm, labels_db1)
chscore1 = calinski_harabasz_score(X_norm, labels_db1)
print("Кількість знайдених кластерів:", num_cluster_db1)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore1))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore1))


from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan2 = DBSCAN(eps=0.4, min_samples=8, metric='cityblock')
dbscan2.fit(X_norm)
labels_db2 = dbscan2.labels_
num_cluster_db2 = len(np.unique(labels_db2))

dbscore2 = davies_bouldin_score(X_norm, labels_db2)
chscore2 = calinski_harabasz_score(X_norm, labels_db2)
print("Кількість знайдених кластерів:", num_cluster_db2)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore2))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore2))


from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan3 = DBSCAN(eps=0.01, min_samples=8, metric='cosine')
dbscan3.fit(X_norm)
labels_db3 = dbscan3.labels_
num_cluster_db3 = len(np.unique(labels_db3))

dbscore3 = davies_bouldin_score(X_norm, labels_db3)
chscore3 = calinski_harabasz_score(X_norm, labels_db3)
print("Кількість знайдених кластерів:", num_cluster_db3)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore3))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore3))


from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan4 = DBSCAN(eps=0.01, min_samples=6, metric='cosine')
dbscan4.fit(X_norm)
labels_db4 = dbscan4.labels_
num_cluster_db4 = len(np.unique(labels_db4))

dbscore4 = davies_bouldin_score(X_norm, labels_db4)
chscore4 = calinski_harabasz_score(X_norm, labels_db4)
print("Кількість знайдених кластерів:", num_cluster_db4)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore4))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore4))
```

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan7 = DBSCAN(eps=0.2, min_samples=8, metric='euclidean')
dbscan7.fit(X_norm)
labels_db7 = dbscan7.labels_
num_cluster_db7 = len(np.unique(labels_db7))

dbscore7 = davies_bouldin_score(X_norm, labels_db7)
chscore7 = calinski_harabasz_score(X_norm, labels_db7)
print("Кількість знайдених кластерів:", num_cluster_db7)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore7))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore7))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan5 = DBSCAN(eps=0.01, min_samples=10, metric='cosine')
dbscan5.fit(X_norm)
labels_db5 = dbscan5.labels_
num_cluster_db5 = len(np.unique(labels_db5))

dbscore5 = davies_bouldin_score(X_norm, labels_db5)
chscore5 = calinski_harabasz_score(X_norm, labels_db5)
print("Кількість знайдених кластерів:", num_cluster_db5)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore5))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore5))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan6 = DBSCAN(eps=0.2, min_samples=5, metric='euclidean')
dbscan6.fit(X_norm)
labels_db6 = dbscan6.labels_
num_cluster_db6 = len(np.unique(labels_db6))

dbscore6 = davies_bouldin_score(X_norm, labels_db6)
chscore6 = calinski_harabasz_score(X_norm, labels_db6)
print("Кількість знайдених кластерів:", num_cluster_db6)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore6))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore6))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan8 = DBSCAN(eps=0.5, min_samples=8, metric='cityblock')
dbscan8.fit(X_norm)
labels_db8 = dbscan8.labels_
num_cluster_db8 = len(np.unique(labels_db8))

dbscore8 = davies_bouldin_score(X_norm, labels_db8)
chscore8 = calinski_harabasz_score(X_norm, labels_db8)
print("Кількість знайдених кластерів:", num_cluster_db8)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore8))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore8))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan9 = DBSCAN(eps=0.15, min_samples=6, metric='euclidean')
dbscan9.fit(X_norm)
labels_db9 = dbscan9.labels_
num_cluster_db9 = len(np.unique(labels_db9))

dbscore9 = davies_bouldin_score(X_norm, labels_db9)
chscore9 = calinski_harabasz_score(X_norm, labels_db9)
```

```
print("Кількість знайдених кластерів:", num_cluster_db9)

print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore9))

print("Калінський Харабаш Партитура: {:.4f}".format(chscore9))
```

Перша оцінка (evaluation)

Показник Девіса-Болдіна*.

Показник Девіса-Болдіна вимірює середню «схожість» між кластерами, яка визначається відстанню між кластерами та розміром кластерів (нижчий показник, близький до 0, вказує на кращу відокремленість між кластерами).

```
k_range3 = 30

epsilons = [0.40]

min_samples = range(2,k_range3,1)

metrics = ['cityblock']

bestdb2, bestch2, dbscores2, chscores2 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()

figure.set_size_inches(10, 5)

axis1.set_xlabel('мінімальна вибірка', size=12)

axis1.set_ylabel('db-score', size=12)

axis1.plot(bestdb2[7], dbscores2, "bo-", label='db-score (менше - краще)')

axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))

axis2 = axis1.twinx()

axis2.set_ylabel('к-сть кластерів', size=12)

axis2.plot(bestdb2[7], bestdb2[8], "bo-", label='кількість кластерів',color='red')

axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка Девіса-Болдіна - eps=0.4, metric=cityblock',size=14)

figure.tight_layout()

plt.show()

k_range3 = 1.5

epsilons = np.arange(0.1,k_range3,0.05)

min_samples = [8]

metrics = ['cityblock']

bestdb3, bestch3, dbscores3, chscores3 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()

figure.set_size_inches(10, 5)

axis1.set_xlabel('епсілон', size=12)

axis1.set_ylabel('db-score', size=12)

axis1.plot(bestdb3[6], dbscores3, "bo-", label='db-score (менше - краще)')

axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))

axis2 = axis1.twinx()

axis2.set_ylabel('к-сть кластерів', size=12)

axis2.plot(bestdb3[6], bestdb3[8], "bo-", label='кількість кластерів',color='red')

axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка Девіса-Болдіна - min sample=8, metric=cityblock',size=14)

figure.tight_layout()

plt.show()

k_range3 = 10

epsilons = [0.15]

min_samples = range(2,k_range3,1)

metrics = ['euclidean']

bestdb4, bestch4, dbscores4, chscores4 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)
```

```
figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)

axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('db-score', size=12)
axis1.plot(bestdb4[7], dbscores4, "bo-", label='db-score (менше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.89))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb4[7], bestdb4[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.8))

plt.title('Оцінка Девіса-Болдіна - eps=0.15, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

k_range3 = 0.7
epsilons = np.arange(0.1,k_range3,0.05)
min_samples = [5]
metrics = ['euclidean']


bestdb5, bestch5, dbscores5, chscores5 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)

axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('db-score', size=12)
axis1.plot(bestdb5[6], dbscores5, "bo-", label='db-score (менше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb5[6], bestdb5[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка Девіса-Болдіна - min sample=5, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

k_range3 = 20
epsilons = [0.01]
min_samples = range(2,k_range3,1)
metrics = ['cosine']

bestdb6, bestch6, dbscores6, chscores6 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)

axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('db-score', size=12)
axis1.plot(bestdb6[7], dbscores6, "bo-", label='db-score (менше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb6[7], bestdb6[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка Девіса-Болдіна - eps=0.01, metric=cosine',size=14)
figure.tight_layout()
plt.show()
```

```
k_range3 = 0.1

epsilons = np.arange(0.01,k_range3,0.01)

min_samples = [8]

metrics = ['cosine']


bestdb7, bestch7, dbscores7, chscores7 = dbscan_grid(X_norm, epsilons, min_samples, metrics, high_range=50)


figure, axis1 = plt.subplots()

figure.set_size_inches(10, 5)


axis1.set_xlabel('епсілон', size=12)

axis1.set_ylabel('db-score', size=12)

axis1.plot(bestdb7[6], dbscores7, "bo-", label='db-score (менше - краще)')

axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))


axis2 = axis1.twinx()

axis2.set_ylabel('к-сть кластерів', size=12)

axis2.plot(bestdb7[6], bestdb7[8], "bo-", label='кількість кластерів',color='red')

axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))


plt.title('Оцінка Девіса-Болдіна - min sample=8, metric=cosine',size=14)

figure.tight_layout()

plt.show()


### Друга оцінка (evaluation)


**Показник Калінкі-Харабаша**

Показник Калінкі-Харабаша вимірює співвідношення між внутрішньою та міжкластерною дисперсіями (вищий показник означає щільні та добре відокремлені кластери).


figure, axis1 = plt.subplots()

figure.set_size_inches(10, 5)


axis1.set_xlabel('мінімальна вибірка', size=12)

axis1.set_ylabel('ch-score', size=12)

axis1.plot(bestch2[7], chscores2, "bo-", label='ch-score (більше - краще)')

axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))


axis2 = axis1.twinx()

axis2.set_ylabel('к-сть кластерів', size=12)

axis2.plot(bestdb2[7], bestdb2[8], "bo-", label='кількість кластерів',color='red')

axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))


plt.title('По оцінці Калінскі та Харабаш - eps=0.4, metric=cityblock',size=14)

figure.tight_layout()

plt.show()


figure, axis1 = plt.subplots()

figure.set_size_inches(10, 5)


axis1.set_xlabel('епсілон', size=12)

axis1.set_ylabel('ch-score', size=12)

axis1.plot(bestch3[6], chscores3, "bo-", label='ch-score (більше - краще)')

axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))


axis2 = axis1.twinx()

axis2.set_ylabel('к-сть кластерів', size=12)

axis2.plot(bestdb3[6], bestdb3[8], "bo-", label='кількість кластерів',color='red')

axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))


plt.title('По оцінці Калінскі та Харабаш - min sample=8, metric=cityblock',size=14)

figure.tight_layout()

plt.show()


figure, axis1 = plt.subplots()
```

```
figure.set_size_inches(10, 5)



axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('ch-score', size=12)
axis1.plot(bestch4[7], chscores4, "bo-", label='ch-score (більше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))



axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb4[7], bestdb4[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('По оцінці Калінскі та Харабаш - eps=0.15, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)



axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('ch-score', size=12)
axis1.plot(bestch5[6], chscores5, "bo-", label='ch-score (більше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))



axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb5[6], bestdb5[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('По оцінці Калінскі та Харабаш - min sample=5, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)



axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('ch-score', size=12)
axis1.plot(bestch6[7], chscores6, "bo-", label='ch-score (більше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))



axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb6[7], bestdb6[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('По оцінці Калінскі та Харабаш - eps=0.01, metric=cosine',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)



axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('ch-score', size=12)
axis1.plot(bestch7[6], chscores7, "bo-", label='ch-score (більше - краще)')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))
```

```
axis2 = axis1.twinx()

axis2.set_ylabel('к-сть кластерів', size=12)

axis2.plot(bestdb7[6], bestdb7[8], "bo-", label='кількість кластерів',color='red')

axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))


plt.title('По оцінці Калінскі та Харабаш - min sample=8, metric=cosine',size=14)

figure.tight_layout()

plt.show()
```

Зображення кластерів

Прогнози t-SNE

На проекціях t-SNE жодна з моделей не дала хорошої кластеризації, один кластер був розкиданий навколо інших кластерів (Рисунок 5.1.3g). Навіть найкраща модель мала цю проблему.

```
plot_tsne(X_norm,k=5,label=labels_db1,title='DBSCAN, Епсілон: 0.22, Мінімум зразків: 7, Метрика: euclidean')


plot_tsne(X_norm,k=13,label=labels_db6,title='DBSCAN, Епсілон: 0.2, Мінімум зразків: 5, Метрика: euclidean')


plot_tsne(X_norm,k=10,label=labels_db7,title='DBSCAN, Епсілон: 0.2, Мінімум зразків: 8, Метрика: euclidean')


plot_tsne(X_norm,k=7,label=labels_db9,title='DBSCAN, Епсілон: 0.15, Мінімум зразків: 6, Метрика: euclidean')


plot_tsne(X_norm,k=11,label=labels_db4,title='DBSCAN, Епсілон: 0.01, Мінімум зразків: 6, Метрика: cosine')


plot_tsne(X_norm,k=8,label=labels_db3,title='DBSCAN, Епсілон: 0.01, Мінімум зразків: 8, Метрика: cosine')


plot_tsne(X_norm,k=6,label=labels_db5,title='DBSCAN, Епсілон: 0.01, Мінімум зразків: 10, Метрика: cosine')


plot_tsne(X_norm,k=8,label=labels_db2,title='DBSCAN, Епсілон: 0.4, Мінімум зразків: 8, Метрика: cityblock')


plot_tsne(X_norm,k=5,label=labels_db2,title='DBSCAN, Епсілон: 0.5, Мінімум зразків: 8, Метрика: cityblock')
```

4.2 ICMLA - Набір даних

Функції

```
def dbscan_grid(X, epsilons, min_samples, metrics, high_range):
  from sklearn.cluster import DBSCAN
  from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score, completeness_score, homogeneity_score

  dbscore_min = 999999
  chscore_max = 0
  comp_max = 0
  homo_max = 0

  dbscores = []
  chscores = []
  comps = []
  homos = []
  x_epsilons = []
  x_min_samples = []
  nclusters = []

  for eps in epsilons:
    for min_sample in min_samples:
      for metric in metrics:
        dbscan = DBSCAN(eps=eps, min_samples=min_sample, metric=metric)
        dbscan.fit(X)
        labels = dbscan.labels_
        num_cluster = len(np.unique(labels))
```

```
if num_cluster>1  and num_cluster<high_range:
  dbscore = davies_bouldin_score(X, labels)
  chscore = calinski_harabasz_score(X, labels)
  compl = completeness_score(y2, labels)
  homog = homogeneity_score(y2, labels)

  if dbscore<dbscore_min:
    dbscore_min = dbscore
    eps_min = eps
    sample_min = min_sample
    metric_min = metric
    num_cluster_min = len(np.unique(labels))
    chscore_min = chscore

  if chscore>chscore_max:
    chscore_max = chscore
    eps_max = eps
    sample_max = min_sample
    metric_max = metric
    num_cluster_max = len(np.unique(labels))
    dbscore_max = dbscore

  if homog>homo_max:
    homo_max = homog
    eps_maxh = eps
    sample_maxh = min_sample
    metric_maxh = metric
    num_cluster_maxh = len(np.unique(labels))
    comp_maxh = compl

  if compl>comp_max:
    comp_max = compl
    eps_maxc = eps
    sample_maxc = min_sample
    metric_maxc = metric
    num_cluster_maxc = len(np.unique(labels))
    homo_maxc = homog

  if dbscore_min!=999999:
    dbscores.append(dbscore)

  if chscore_max!=0:
    chscores.append(chscore)
    x_epsilons.append(eps)
    x_min_samples.append(min_sample)
    nclusters.append(num_cluster)

  if comp_max!=0:
    comps.append(compl)

  if homo_max!=0:
    homos.append(homog)

  bestdb = [dbscore_min, eps_min, sample_min, metric_min, num_cluster_min, chscore_min, x_epsilons, x_min_samples, nclusters]
  bestch = [chscore_max, eps_max, sample_max, metric_max, num_cluster_max, dbscore_max, x_epsilons, x_min_samples, nclusters]
  bestcomp = [comp_max, eps_maxc, sample_maxc, metric_maxc, num_cluster_maxc, homo_maxc, x_epsilons, x_min_samples, nclusters]
  besthomo = [homo_max, eps_maxh, sample_maxh, metric_maxh, num_cluster_maxh, comp_maxh, x_epsilons, x_min_samples, nclusters]

  return bestdb, bestch, dbscores, chscores, comps, homos, bestcomp, besthomo

def plot_tsne(X,k,label,title=None):

  from sklearn.manifold import TSNE
  tsne = TSNE(random_state=42)

  X_tsne = tsne.fit_transform(X)
```

```
  plt.figure(figsize=(10,8))
  plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=label,
              edgecolor='none', alpha=0.7, s=40,
              cmap=plt.cm.get_cmap('nipy_spectral', k))
  plt.colorbar()
  title = 'Проекція t-SNE, ' + title + ', k=' + str(k)
  plt.title(title)
```

Тренуємо кластеризацію DBSCAN

Понад 100 моделей було навчено з використанням тих самих трьох мір відстані та різних значень епсилон (0,01-5) і min вибірок (2-15).

```
try:
  X2_icmla_df = X_combine_pca
  epsilons = np.arange(0.01,5,0.01)
  min_samples = range(2,15,1)
  metrics = ['euclidean', 'cityblock', 'cosine']

  bestdb1, bestch1, dbscores1, chscores1 , comps1, homos1, bestcomp1, besthomo1 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)

  print("Найкращі параметри на основі dbscore:")
  print("Епсілон: {:.2f}".format(bestdb1[1]))
  print("Мінімальна к-сть зразків:", bestdb1[2])
  print("Метрика:", bestdb1[3])
  print("dbscore: {:.4f}".format(bestdb1[0]))
  print("chscore: {:.4f}".format(bestdb1[5]))
  print("Кількість знайдених кластерів:", bestdb1[4])

  print("\nНайкращі параметри на основі chscore:")
  print("Епсілон: {:.2f}".format(bestch1[1]))
  print("Мінімальна к-сть зразків:", bestch1[2])
  print("Метрика:", bestch1[3])
  print("chscore: {:.4f}".format(bestch1[0]))
  print("dbscore: {:.4f}".format(bestch1[5]))
  print("Кількість знайдених кластерів:", bestch1[4])

  print("\nНайкращі параметри на основі completeness:")
  print("Епсілон: {:.2f}".format(bestcomp1[1]))
  print("Мінімальна к-сть зразків:", bestcomp1[2])
  print("Метрика:", bestcomp1[3])
  print("completeness: {:.4f}".format(bestcomp1[0]))
  print("homogeneity: {:.4f}".format(bestcomp1[5]))
  print("Кількість знайдених кластерів:", bestcomp1[4])

  print("\nНайкращі параметри на основі homogeneity:")
  print("Епсілон: {:.2f}".format(besthomo1[1]))
  print("Мінімальна к-сть зразків:", besthomo1[2])
  print("Метрика:", besthomo1[3])
  print("completeness: {:.4f}".format(besthomo1[5]))
  print("homogeneity: {:.4f}".format(besthomo1[0]))
  print("Кількість знайдених кластерів:", besthomo1[4])
except:
  print("Не вдалося знайти > 1 кластерів. Будь ласка, спробуйте різні комбінації epsilon або min_samples.")

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score, completeness_score, homogeneity_score

dbscan1 = DBSCAN(eps=besthomo1[1], min_samples=besthomo1[2], metric=besthomo1[3])
dbscan1.fit(X2_icmla_df)
labels_db1 = dbscan1.labels_
num_cluster_db1 = len(np.unique(labels_db1))

dbscore1 = davies_bouldin_score(X2_icmla_df, labels_db1)
chscore1 = calinski_harabasz_score(X2_icmla_df, labels_db1)
compl1 = completeness_score(y2, labels_db1)
```

```
homog1 = homogeneity_score(y2, labels_db1)
print("Кількість знайдених кластерів:", num_cluster_db1)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore1))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore1))
print("Повноцінність Score: {:.4f}".format(compl1))
print("Однорідність Score: {:.4f}".format(homog1))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan2 = DBSCAN(eps=4.1, min_samples=2, metric='cityblock')
dbscan2.fit(X2_icmla_df)
labels_db2 = dbscan2.labels_
num_cluster_db2 = len(np.unique(labels_db2))

dbscore2 = davies_bouldin_score(X2_icmla_df, labels_db2)
chscore2 = calinski_harabasz_score(X2_icmla_df, labels_db2)
compl2 = completeness_score(y2, labels_db2)
homog2 = homogeneity_score(y2, labels_db2)
print("Кількість знайдених кластерів:", num_cluster_db2)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore2))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore2))
print("Повноцінність Score: {:.4f}".format(compl2))
print("Однорідність Score: {:.4f}".format(homog2))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan3 = DBSCAN(eps=1.14, min_samples=1, metric='euclidean')
dbscan3.fit(X2_icmla_df)

labels_db3 = dbscan3.labels_
num_cluster_db3 = len(np.unique(labels_db3))

dbscore3 = davies_bouldin_score(X2_icmla_df, labels_db3)
chscore3 = calinski_harabasz_score(X2_icmla_df, labels_db3)
compl3 = completeness_score(y2, labels_db3)
homog3 = homogeneity_score(y2, labels_db3)
print("Кількість знайдених кластерів:", num_cluster_db3)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore3))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore3))
print("Повноцінність Score: {:.4f}".format(compl3))
print("Однорідність Score: {:.4f}".format(homog3))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

dbscan4 = DBSCAN(eps=0.42, min_samples=2, metric='cosine')
dbscan4.fit(X2_icmla_df)

labels_db4 = dbscan4.labels_

num_cluster_db4 = len(np.unique(labels_db4))

dbscore4 = davies_bouldin_score(X2_icmla_df, labels_db4)
chscore4 = calinski_harabasz_score(X2_icmla_df, labels_db4)
compl4 = completeness_score(y2, labels_db4)
homog4 = homogeneity_score(y2, labels_db4)
print("Кількість знайдених кластерів:", num_cluster_db4)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore4))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore4))
print("Повноцінність Score: {:.4f}".format(compl4))
print("Однорідність Score: {:.4f}".format(homog4))
```

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score


dbscan5 = DBSCAN(eps=4.6, min_samples=2, metric='cityblock')
dbscan5.fit(X2_icmla_df)

labels_db5 = dbscan5.labels_

num_cluster_db5 = len(np.unique(labels_db5))


dbscore5 = davies_bouldin_score(X2_icmla_df, labels_db5)
chscore5 = calinski_harabasz_score(X2_icmla_df, labels_db5)
compl5 = completeness_score(y2, labels_db5)
homog5 = homogeneity_score(y2, labels_db5)
print("Кількість знайдених кластерів:", num_cluster_db5)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore5))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore5))
print("Повноцінність Score: {:.4f}".format(compl5))
print("Однорідність Score: {:.4f}".format(homog5))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score


dbscan6 = DBSCAN(eps=1, min_samples=2, metric='euclidean')
dbscan6.fit(X2_icmla_df)

labels_db6 = dbscan6.labels_

num_cluster_db6 = len(np.unique(labels_db6))


dbscore6 = davies_bouldin_score(X2_icmla_df, labels_db6)
chscore6 = calinski_harabasz_score(X2_icmla_df, labels_db6)
compl6 = completeness_score(y2, labels_db6)
homog6 = homogeneity_score(y2, labels_db6)
print("Кількість знайдених кластерів:", num_cluster_db6)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore6))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore6))
print("Повноцінність Score: {:.4f}".format(compl6))
print("Однорідність Score: {:.4f}".format(homog6))

from sklearn.cluster import DBSCAN
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score


dbscan7 = DBSCAN(eps=4.85, min_samples=1, metric='cityblock')
dbscan7.fit(X2_icmla_df)

labels_db7 = dbscan7.labels_

num_cluster_db7 = len(np.unique(labels_db7))


dbscore7 = davies_bouldin_score(X2_icmla_df, labels_db7)
chscore7 = calinski_harabasz_score(X2_icmla_df, labels_db7)
compl7 = completeness_score(y2, labels_db7)
homog7 = homogeneity_score(y2, labels_db7)
print("Кількість знайдених кластерів:", num_cluster_db7)
print("Оцінка Девіса Боулдіна: {:.4f}".format(dbscore7))
print("Калінський Харабаш Партитура: {:.4f}".format(chscore7))
print("Повноцінність Score: {:.4f}".format(compl7))
print("Однорідність Score: {:.4f}".format(homog7))

### Перша оцінка (evaluation)

**Оцінка повноти**
```

Оцінка повноти вимірює ступінь належності всіх точок даного класу до одного кластеру (оцінка 1 означає ідеально повне маркування).

```
k_range3 = 10
epsilons = [4.1]
min_samples = range(1,k_range3,1)
metrics = ['cityblock']


bestdb2, bestch2, dbscores2, chscores2, comps2, homos2, bestcomp2, besthomo2 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)


axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('повноцінність', size=12)
axis1.plot(bestcomp2[7], comps2, "bo-", label='Повноцінність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))


axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb2[7], bestdb2[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка завершеності - eps=4.1, metric=cityblock',size=14)
figure.tight_layout()

plt.show()

k_range3 = 5
epsilons = np.arange(3,k_range3,0.05)
min_samples = [2]
metrics = ['cityblock']


bestdb3, bestch3, dbscores3, chscores3, comps3, homos3, bestcomp3, besthomo3 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()
figure.set_size_inches(15, 6)


axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('повноцінність', size=12)
axis1.plot(bestcomp3[6], comps3, "bo-", label='Повноцінність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.15, 0.99))


axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb3[6], bestdb3[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.15, 0.9))

plt.title('Оцінка завершеності - min sample=2, metric=cityblock',size=14)
figure.tight_layout()
plt.show()

k_range3 = 10
epsilons = [1.14]
min_samples = range(1,k_range3,1)
metrics = ['euclidean']
```

```
bestdb4, bestch4, dbscores4, chscores4, comps4, homos4, bestcomp4, besthomo4 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)

axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('повноцінність', size=12)
axis1.plot(bestcomp4[7], comps4, "bo-", label='Повноцінність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.68, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb4[7], bestdb4[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.68, 0.9))

plt.title('Оцінка завершеності - eps=1.14, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

k_range3 = 1.6
epsilons = np.arange(0.6,k_range3,0.05)
min_samples = [2]
metrics = ['euclidean']

bestdb5, bestch5, dbscores5, chscores5, comps5, homos5, bestcomp5, besthomo5 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)

figure, axis1 = plt.subplots()
figure.set_size_inches(15, 6)

axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('повноцінність', size=12)
axis1.plot(bestcomp5[6], comps5, "bo-", label='Повноцінність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.15, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb5[6], bestdb5[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.15, 0.9))

plt.title('Оцінка завершеності - min sample=2, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

k_range3 = 10
epsilons = [0.18]
min_samples = range(1,k_range3,1)
metrics = ['cosine']

bestdb6, bestch6, dbscores6, chscores6, comps6, homos6, bestcomp6, besthomo6 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)
```

```
figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)


axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('повноцінність', size=12)
axis1.plot(bestcomp6[7], comps6, "bo-", label='Повноцінність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))



axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb6[7], bestdb6[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))


plt.title('Оцінка завершеності - eps=0.18, metric=cosine',size=14)
figure.tight_layout()
plt.show()


k_range3 = 0.5
epsilons = np.arange(0.01,k_range3,0.01)
min_samples = [2]
metrics = ['cosine']


bestdb7, bestch7, dbscores7, chscores7, comps7, homos7, bestcomp7, besthomo7 = dbscan_grid(X2_icmla_df, epsilons, min_samples, metrics, high_range=50)
```

```
figure, axis1 = plt.subplots()
figure.set_size_inches(15,6)



axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('повноцінність', size=12)
axis1.plot(bestcomp7[6], comps7, "bo-", label='Повноцінність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.15, 0.99))



axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb7[6], bestdb7[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.15, 0.9))


plt.title('Оцінка завершеності - min sample=2, metric=cosine',size=14)
figure.tight_layout()
plt.show()
```

Друга оцінка (evaluation)

Показник однорідності

Показник однорідності вимірює ступінь, до якого всі кластери містять лише точки одного класу (оцінка 1 означає абсолютно однорідне маркування).

```
figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)


axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('однорідністьі', size=12)
axis1.plot(besthomo2[7], homos2, "bo-", label='Однорідність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))
```

```
axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb2[7], bestdb2[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка однорідності - eps=4.1, metric=cityblock',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(15, 6)

axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('однорідністьі', size=12)
axis1.plot(besthomo3[6], homos3, "bo-", label='Однорідність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.15, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb3[6], bestdb3[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.15, 0.9))

plt.title('Оцінка однорідності - min sample=2, metric=cityblock',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)

axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('однорідністьі', size=12)
axis1.plot(besthomo4[7], homos4, "bo-", label='Однорідність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb4[7], bestdb4[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка однорідності - eps=1.14, metric=euclidean',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(15, 6)

axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('однорідністьі', size=12)
axis1.plot(besthomo5[6], homos5, "bo-", label='Однорідність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))

axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb5[6], bestdb5[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка однорідності - min sample=2, metric=euclidean',size=14)
figure.tight_layout()
plt.show()
```

```
figure, axis1 = plt.subplots()
figure.set_size_inches(10, 5)


axis1.set_xlabel('мінімальна вибірка', size=12)
axis1.set_ylabel('однорідністьі', size=12)
axis1.plot(besthomo6[7], homos6, "bo-", label='Однорідність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))


axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb6[7], bestdb6[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка однорідності - eps=0.18, metric=cosine',size=14)
figure.tight_layout()
plt.show()

figure, axis1 = plt.subplots()
figure.set_size_inches(15, 6)


axis1.set_xlabel('епсілон', size=12)
axis1.set_ylabel('однорідністьі', size=12)
axis1.plot(besthomo7[6], homos7, "bo-", label='Однорідність')
axis1.legend(loc='upper right', bbox_to_anchor=(0.98, 0.99))


axis2 = axis1.twinx()
axis2.set_ylabel('к-сть кластерів', size=12)
axis2.plot(bestdb7[6], bestdb7[8], "bo-", label='кількість кластерів',color='red')
axis2.legend(loc='upper right', bbox_to_anchor=(0.98, 0.9))

plt.title('Оцінка однорідності - min sample=2, metric=cosine',size=14)
figure.tight_layout()
plt.show()
```

Зображення кластерів

Проекції t-SNE

Моделі DBSCAN не дуже добре відображаються на проекціях t-SNE. Утворені кластери не були чітко відокремлені, і їх важко ідентифікувати.

```
plot_tsne(X2_icmla_df,k=20,label=labels_db1,title='DBSCAN, Епсілон: 4.63, Мінімум зразків: 2, Метрика: cityblock')

plot_tsne(X2_icmla_df,k=15,label=labels_db2,title='DBSCAN, Епсілон: 4.10, Мінімум зразків: 2, Метрика: cityblock')

plot_tsne(X2_icmla_df,k=17,label=labels_db3,title='DBSCAN, Епсілон: 1.14, Мінімум зразків: 1, Метрика: euclidean')

plot_tsne(X2_icmla_df,k=15,label=labels_db4,title='DBSCAN, Епсілон: 0.42, Мінімум зразків: 2, Метрика: cosine')

plot_tsne(X2_icmla_df,k=20,label=labels_db5,title='DBSCAN, Епсілон: 4.60, Мінімум зразків: 2, Метрика: cityblock')

plot_tsne(X2_icmla_df,k=20,label=labels_db6,title='DBSCAN, Епсілон: 1, Мінімум зразків: 2, Метрика: euclidean')

plot_tsne(X2_icmla_df,k=21,label=labels_db7,title='DBSCAN, Епсілон: 4.85, Мінімум зразків: 1, Метрика: cityblock')
```

Таблиця маркування

```
pd.set_option('display.max_rows', 105)
icmla_df_label20 = pd.DataFrame({"label":labels_db1, "session":y2_string})
```

```
icmla_df_label20.sort_values(by='label')
```

ДОДАТОК В

Технічно завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

«АНАЛІЗ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ»

Керівництво користувача

44165850.01351 – 01 І3 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександр ІВАНОВ
Виконавець
_____Владислав Єрмаков
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01351-01 І3 01

«АНАЛІЗ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ»

Керівництво користувача

44165850.01351 – 01 І3 01

Листів 9

2025

44165850.01351-01 І3 01

АНОТАЦІЯ

Документ 44165850.01223–01 12 01 «Аналіз алгоритмів кластеризації для обробки великих даних» є частиною програмної документації на програму, що виконує роль прототипу для аналізу та порівняння методів кластеризації великих обсягів даних.

У цьому документі представлено текст програми, створеної мовою програмування Python із використанням бібліотек Scikit-learn, NumPy, Pandas та Matplotlib. Розробка здійснювалася в середовищі Jupyter Notebook.

Даний документ охоплює ключові аспекти функціональності, реалізації та тестування алгоритмів кластеризації, таких як K-means, ієрархічна кластеризація та DBSCAN, для забезпечення їхньої адаптації до задач аналізу великих даних.

ЗМІСТ

1 ВВЕДЕННЯ

Програмний засіб «Аналіз алгоритмів кластеризації для обробки великих даних» призначений для дослідження та порівняння алгоритмів кластеризації, таких як K-means, ієрархічна кластеризація та DBSCAN.

Головною метою розробки є надання інструментарію для виконання кластеризації даних, проведення дослідження характеристик алгоритмів (наприклад, точності, швидкості та стійкості до шумів), а також збору метрик для обґрунтованого вибору алгоритму залежно від конкретних завдань.

Продукт призначений для використання аналітиками, дослідниками та іншими спеціалістами для оцінки ефективності алгоритмів у різних сценаріях. Програма не вимагає високого рівня технічних навичок і може бути використана будь-якою особою, знайомою з основами роботи з комп’ютером.

Програмний засіб є інтуїтивно зрозумілим і не потребує ознайомлення з додатковою експлуатаційною документацією.

2 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Функціональним призначенням додатку «Аналіз алгоритмів кластеризації для обробки великих даних» є проведення аналізу алгоритмів кластеризації з метою вибору найбільш ефективного алгоритму для кластеризації даних у конкретних умовах.

Експлуатаційне призначення додатку «Аналіз алгоритмів кластеризації для обробки великих даних» – тестування та порівняння алгоритмів кластеризації K-means, ієрархічної кластеризації та DBSCAN з використанням реальних і синтетичних наборів даних для визначення їхньої продуктивності, точності та практичності.

Для забезпечення сталого функціонування програми користувачеві і програмісту необхідно дотримуватися таких умов - наявність інтернет з’єднання.

Додаток, що розробляється, розрахований на використання на пристроях що мають:

процесор з тактовою частотою не нижче 2.5 ГГц;

не менше 4ГБ оперативної пам’яті.

Для функціонування додатку неохідна система Windows 10 версії і вище.

3 ПІДГОТОВКА ДО РОБОТИ

Для початку роботи додатку необхідно завантажити програмний файл та вхідні дані в одну директорію для аналізу.

4 ОПИС ОПЕРАЦІЙ

Після встановлення та запуску додатку “ “ на формі додатку присутні:

поле вводу параметрів алгоритму кластеризації, наприклад, кількість кластерів для K-means або радіус для DBSCAN;

кнопка «Run», яка виконує комірку з кодом для запуску алгоритму кластеризації або обробки даних;

кнопка «+», яка додає нову комірку для введення коду чи текстових пояснень;

кнопка «Interrupt Kernel», яка зупиняє виконання довготривалого або некоректного коду;

кнопка «Restart Kernel», яка перезапускає середовище Jupyter Notebook для очищення пам’яті;

таблиця з фільтрацією даних, яка дозволяє попередньо підготувати дані для кластеризації, наприклад, відфільтрувати шум або обрати необхідні стовпці;

список результатів, який відображає таблиці, метрики (Silhouette Score, Calinski-Harabasz) або графіки;

кнопка «Plot», яка будує графіки для візуалізації результатів кластеризації, наприклад, дендрограми або розташування кластерів;

кнопка «Clear Output», яка очищає вивід для подальших запусків аналізу;

таблиця зі зміненими даними, яка дозволяє адаптувати дані перед наступною кластеризацією;

кнопка «Update Analysis», яка запускає алгоритм на нових параметрах або змінених даних;

чек бокс або текстове поле, яке дозволяє вибирати алгоритм кластеризації для запуску, наприклад, K-means, DBSCAN або ієрархічну кластеризацію;

кнопка для збереження результатів, яка експортує отримані результати у формат CSV або JSON.

5 АВАРІЙНІ СИТУАЦІЇ

Якщо програмний засіб буде запускатися з пристрою де немає інтернет з’єднання, додаток виведе на екран повідомлення про помилку та завершить роботу.

Якщо програмний засіб під час роботи буде поводити некоректно чи із помилкою, необхідно перезапустити мобільним додаток для продовження роботи.

6 РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ

Після встановлення Jupyter Notebook необхідно запустити середовище через відповідний виконуючий файл або команду в терміналі.

Після запуску середовища перед користувачем відкриється інтерфейс у браузері з усім необхідним функціоналом для роботи.

Після завершення роботи з ноутбуком його можна закрити через меню File > Close and Halt або закрити термінал.