МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ КЛАСТЕРНОГО АНАЛІЗУ ЗАСОБАМИ C#»**

Виконала: студентка 2 курсу, групи 8.1213-з
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
І.А. Скрипник
(ініціали та прізвище)

Керівник — професор кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент — доцент кафедри комп'ютерних наук, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Скрипник Ірині Анатоліївні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка програмного застосунку кластерного аналізу засобами C#

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 384-с

2. Строк подання студентом роботи 25.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка застосунку засобами C#. Дослідження проблеми кластерного аналізу даних, висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 24.04.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.05.2024	
2.	Збір вихідних даних.	30.05.2024	
3.	Обробка методичних та теоретичних	10.07.2024	
	джерел.		
4.	Розробка першого та другого розділу.	02.09.2024	
5.	Розробка третього розділу.	21.10.2024	
6.	Оформлення та нормоконтроль	20.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	14.12.2024	

Студент _____ І.А. Скрипник
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного застосунку кластерного аналізу засобами C#»: 78 с., 40 рис., 25 джерел, 3 додатки.

АЛГОРИТМИ КЛАСТЕРИЗАЦІЇ, АНАЛІЗ ДАНИХ, ДАТАСЕТ, КЛАСТЕРНИЙ АНАЛІЗ, МАШИННЕ НАВЧАННЯ, МЕТРИКА, МОВА C#, ACCORD.NET, ML.NET, OXYPLOT.

Об’єкт дослідження – процес застосування методів кластеризації до об’єктів даних.

Предмет дослідження – засоби кластерного аналізу заданої множини даних та особливості використання цих засобів.

Мета роботи: вивчення існуючих методів та алгоритмів кластеризації та розробка застосунку кластерного аналізу на C#.

Методи дослідження – аналіз методів кластеризації, виявлення їх особливостей для наборів даних; аналіз популярних бібліотек машинного навчання та візуалізації для C#; експеримент з різними алгоритмами кластерного аналізу та даними; аналіз результатів.

В процесі дослідження проаналізовано проблему кластеризації об’єктів даних, їх попередньої обробки, розглянуто існуючі бібліотеки машинного навчання для C#, засоби візуалізації вхідних даних та результатів кластерного аналізу. Як результат, розроблено програмний застосунок мовою C# з використанням бібліотек ML.NET, Accord.NET для кластеризації, OxyPlot для візуалізації результатів. Програмне рішення реалізує кластерний аналіз наборів даних за допомогою різних алгоритмів кластеризації, візуалізує отримані результати, що дозволяє зробити висновки щодо застосування методів кластеризації даних.

SUMMARY

Master's qualifying paper "Development of a Software Application for Cluster Analysis Using C#": 78 pages, 40 figures, 25 references, 3 supplements.

ACCORD.NET, CLUSTERING ALGORITHMS, CLUSTER ANALYSIS, C# LANGUAGE, DATA ANALYSIS, DATASET, MACHINE LEARNING, METRICS, ML.NET, OXYPLOT.

The object of the research is the process of applying clustering methods to data objects.

The subject of the research is the tools for cluster analysis of a specified dataset and the peculiarities of using these tools.

The purpose of the work is to study existing clustering methods and algorithms and to develop a cluster analysis application in C#.

The research employed methods such as the analysis of clustering methods and the identification of their features for datasets; examination of popular machine learning and visualization libraries for C#; experimentation with various clustering algorithms and datasets; and analysis of the results.

The research examined the problem of clustering data objects, their preprocessing, and explored existing machine learning libraries for C#, as well as tools for visualizing input data and clustering results. As a result, a software application was developed in C# using the ML.NET and Accord.NET libraries for clustering and OxyPlot for result visualization. The solution implements cluster analysis of data sets using various clustering algorithms, visualizes the obtained results, and provides insights into the application of clustering methods for data analysis.

ЗМІСТ

Завдання на кваліфікаційну роботу...…….....2
Реферат 4
Summary 5
Вступ..... 8
1 Дослідження проблеми кластерного аналізу як інтелектуального аналізу даних 11
1.1 Огляд проблеми..... 11
1.2 Існуючі методи кластерного аналізу..... 13
1.2.1 Кластеризація на основі центроїдів 14
1.2.2 Ієрархічна кластеризація 15
1.2.3 Кластеризація на основі щільності (методи на основі моделі) ... 16
1.2.4 Кластеризація на основі розподілу 17
1.2.5 Нечітка кластеризація..... 18
1.2.6 На основі обмежень 19
1.3 Дослідження алгоритмів кластерного аналізу 20
1.3.1 Критерії якості кластеризації..... 21
1.3.2 Алгоритм кластеризації метода k-середніх (K-Means) 24
1.3.3 Алгоритм кластеризації DBSCAN 26
1.3.4 Алгоритм ієрархічної кластеризації..... 29
1.3.5 Модель суміші Гауса (GMM) 33
1.4 Висновки до розділу 1 36
2 Існуючі бібліотеки та програмні рішення на C# для кластерного аналізу... 37
2.1 Фреймворк ML.NET 37
2.1.1 Термінологія і концепція розробки моделі машинного навчання 38
2.1.2 Створення моделі машинного навчання за допомогою ML.NET 40
2.2 Фреймворк Accord.NET..... 44

2.3 Висновок з розділу 2 46
3 Розробка застосунку кластерного аналізу 47
3.1 WPF-архітектура 47
3.2 Бібліотека .NET OxyPlot 48
3.3 Розробка застосунку кластеризації з ML.NET 51
3.3.1 Створення моделі машинного навчання в Model Builder 51
3.3.2 Розробка WPF-застосунку з ML.NET 54
3.4 Розробка WPF-застосунку з Accord.NET 57
3.5 Розробка застосунку для ієрархічної кластеризації та DBSCAN 59
3.6 Висновки з розділу 3 62
Висновки 63
Перелік посилань 64
Додаток А Лістинг програмного застосунку з ML.NET 67
Додаток Б Лістинг застосунку ієрархічної кластеризації та DBSCAN 72
Додаток В Лістинг програми кластеризації з Accord.NET 76

ВСТУП

Оточуючі нас об’єкти часто можна класифікувати за якимись ознаками. Іноді достатньо значення міри розпливчастості та конкретності, аби визначити властивості об’єктів для відокремлення їх від інших. Можна розглядати групи з великої кількості образів або з декількох екземплярів. Проблема полягає в тому, щоб для попередньо визначених груп даних зробити вибір, що ґрунтується на подібності між складовими даних, прихованих ознаках, характерних рисах даних. В інтелектуальному аналізі даних процес групування об’єктів даних разом називається кластеризацією або кластерним аналізом. Отримані кластери можна використовувати для їх подальшої обробки [14].

Мета кластерного аналізу полягає в розділенні набору даних на групи (або кластери), щоб об’єкти даних у кожній групі були схожі один на одного більше, ніж об’єкти в інших групах. Цей процес часто використовується для дослідницького аналізу даних і може допомогти виявити в даних шаблони або зв’язки, які можуть бути неочевидними.

Кластерний аналіз даних використовують для виявлення закономірностей у великих наборах даних. Наприклад, за допомогою алгоритмів кластеризації можна згрупувати користувачів Інтернету за їх приналежністю до соціальних мереж або зображення на основі схожих візуальних характеристик; на основі поведінки мережевого трафіка розділити його на сегменти; а при обробці природної мови згрупувати схожі слова або документи. Крім того, що кластерний аналіз дає уявлення про глибинну структуру даних, його можна використовувати для подальшого аналізу [8].

Для кластеризації можна використовувати різні підходи та алгоритми залежно від наявних даних і доступних засобів. Перш ніж обрати метод розв’язання власних задач, треба дослідити якомога більше існуючих алгоритмів кластеризації, обрати відповідні методи візуалізації результатів

роботи алгоритмів, порівняти їх за допомогою оцінок та аналізу якості результатів [11].

Таким чином, актуальною є розробка програмного забезпечення для кластерного аналізу об’єктів даних з можливістю візуалізації та порівняння результатів.

Мета кваліфікаційної роботи полягає у вивченні існуючих методів та алгоритмів кластеризації та розробці застосунку кластерного аналізу мовою C#.

Досягнення поставленої мети передбачає реалізацію наступних **завдань**:

- вивчення існуючих підходів до кластерного аналізу множини даних;
- вивчення методів та алгоритмів кластеризації;
- дослідження сучасних бібліотек машинного навчання ML.NET та Accord.NET;
- дослідження проблеми візуалізації результатів кластерного аналізу;
- розробку програмного застосунка засобами C# для кластеризації наборів даних різними методами;
- аналіз результатів та формулювання висновків.

Об’єкт дослідження – процес застосування методів кластеризації до об’єктів даних.

Предмет дослідження – можливість використання бібліотек машинного навчання ML.NET та Accord.NET для реалізації кластерного аналізу заданої множини об’єктів даних та особливості використання цих засобів.

Методи дослідження – аналіз методів кластеризації, виявлення їх особливостей для наборів даних; аналіз популярних бібліотек машинного навчання та візуалізації для C#; експеримент з різними алгоритмами кластерного аналізу та даними; аналіз результатів.

Для досягнення результату було обрано бібліотеки ML.NET та Accord.NET, які призначені для розв’язання багатьох типів задач машинного навчання, зокрема, мають засоби для кластеризації об’єктів даних мовою C#.

Для програмної реалізації поставлених задач розроблено WPF-застосунки на C#, для візуалізації результатів використано графічну бібліотеку OxyPlot. Роботу застосунків протестовано на різних наборах даних, зокрема, на розмічених даних та прогнозованих. Виявлено переваги та недоліки алгоритмів кластеризації та сформульовано рекомендації щодо їх застосування. Дослідження показало, що бібліотеки ML.NET та Accord.NET містять велику кількість засобів для розв'язання багатьох типів задач машинного навчання, але для задач кластеризації даних вони надають обмежені можливості щодо вибору методів та алгоритмів. Тому для повноти дослідження методів кластеризації розроблено окремий застосунок без використання бібліотек машинного навчання з власними реалізаціями алгоритмів кластеризації. Виявлено особливості роботи алгоритмів.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань та 3-х додатків. Загальний обсяг роботи становить 78 сторінок. Додатки викладено на 12 сторінках. У переліку посилань 25 використаних джерел.

Перший розділ присвячений дослідженню проблеми кластерного аналізу об'єктів даних, викладено методи вирішення проблеми, проаналізовано існуючі рішення, критерії оцінки результатів кластеризації.

У другому розділі досліджено можливості бібліотек машинного навчання ML.NET, Accord.NET, описано алгоритми кластеризації, визначено їх переваги та недоліки.

В третьому розділі описано розроблені WPF-застосунки, що містять реалізацію алгоритмів кластеризації за допомогою засобів ML.NET, Accord.NET, OxyPlot, проілюстровано результати кластерного аналізу.

В додатках наведено програмну реалізацію алгоритмів K-середніх, DBSCAN, моделі суміші Гауса, агломеративної ієрархічної кластеризації та дендрограми, функцій для графічного представлення результатів, файли .xaml.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ КЛАСТЕРНОГО АНАЛІЗУ ЯК ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

1.1 Огляд проблеми

Кластерний аналіз – це метод інтелектуального аналізу даних, який використовується для групування об’єктів на основі їх схожості. В задачах кластеризації модель машинного навчання повинна сама рознести образи за кластерами, тому такі задачі відносяться до типу задач «без вчителя». Цей метод не використовує мітки класів (або індекси), які зазвичай використовуються в задачах машинного навчання «з вчителем». Натомість кластерний аналіз використовує міри схожості, щоб визначити, які об’єкти повинні бути згруповані разом. Так, для кількісних даних часто використовуються функції відстані, такі як відстань Евкліда, Уорда або косінусна відстань. Для виявлення якісних ознак об’єктів даних часто використовуються функції подібності, як подібність Хеммінга [11, 20, 22].

Кластерний аналіз можна застосувати практично в будь-якій сфері, де аналізуються дані. В маркетингу можна визначити групи клієнтів за їх купівельною спроможністю або сформувати кластери з демографічних даних. У біології гени та білки можна згрупувати відповідно до їх функцій. У соціології визначають підгрупи респондентів за результатами опитувань щодо ставлення до певних речей та переконань. У поштових сервісах за допомогою методів кластеризації можна забезпечити виявлення та керування спамом (нетипові дані) або згрупувати користувачів за наявними акаунтами.

В кластерному аналізі є три складові машинного навчання: дані, їх ознаки (мітки) та алгоритм, здатний навчатись на вхідних даних та видавати потрібний результат.

Набір даних, де окрім вхідних образів є значення цільових функцій (цільові виходи), називається розміченим та являє собою навчальну вибірку.

Розмітити вхідні об’єкти даних або зібрати розмічені дані – дійсно складна робота. Іноді вдається автоматизувати процес розмітки даних.

На практиці ж часто зустрічаються задачі, в яких початковий набір об’єктів не розмічений, і треба побудувати модель, яка б розділяла образи на групи або класи, використовуючи деяку задану метрику – функцію відстані або подібності між об’єктами. То ж, в таких задачах треба перетворити нерозмічені дані в розмічені дані, і це можна зробити за допомогою кластерів. Результати кластеризації можуть бути основою для подальшого навчання.

Розглянемо задачі, які розв’язує кластерний аналіз даних.

Виявлення структури в даних. Кластерний аналіз може допомогти зрозуміти, як дані організовані, навіть якщо вони явно не структуровані. Це може допомогти виявити тенденції, закономірності або аномалії у наборах даних.

Групування подібних об’єктів. Складні дані розбиваються за схожими групами, спрощуючи їх представлення, а, значить, і обробку в межах кожної групи. Це може бути корисним для таких задач, як маркетингові дослідження та медична діагностика.

Зменшення обсягу вибірки за рахунок збереження невеликої кількості характерних представників кожного класу. Це може забезпечити спрощення аналізу даних та покращення його ефективності.

Прогнозування поведінки об’єктів. Кластерний аналіз може бути використаний для прогнозування поведінки нових об’єктів на основі їх схожості з об’єктами в існуючих кластерах. Це може дозволити виділити нетипові об’єкти (викиди), які не можна віднести до жодного кластеру, для побудови рекомендаційних систем та виявлення зловмисників.

Візуалізація даних. Кластеризація може бути використана для візуалізації даних шляхом відображення різних кластерів різними кольорами або маркерами. Це може допомогти краще зрозуміти структуру та розподіл даних.

Властивості алгоритмів кластеризації наводяться далі.

Масштабованість. Для роботи з величезними датасетами алгоритми кластеризації повинні бути масштабованими. Відсутність зазначеної властивості призведе до неправильних результатів кластеризації.

Висока розмірність. Алгоритми кластеризації повинні мати можливість обробляти як простір даних великої розмірності, так і маленькі набори даних.

Зручність використання алгоритмів для різнотипових даних. Алгоритми кластеризації мають бути здатними обробляти об’єкти даних різних типів: дискретні, інтервальні, двійкові дані, а також категоріальні дані, які потребують попередньої обробки для використання в моделі машинного навчання.

Робота з неструктурованими даними. Серед об’єктів даних можуть бути шумові або помилкові дані, неструктуровані дані. Якщо алгоритми чутливі до таких даних, це може призвести до низької якості кластеризації. Таким чином, алгоритми кластеризації повинні вміти обробляти неструктуровані дані, організовуючи подібні об’єкти у кластери та надаючи даним деяку структуру.

Можливість інтерпретації. Результати кластеризації мають бути зрозумілими, придатними для інтерпретації та використання.

1.2 Існуючі методи кластерного аналізу

В машинному навчанні існують різні методи кластеризації [11, 14, 20 – 23]:

- на основі центроїдів (методи поділу);
- на основі щільності (методи на основі моделі);
- на основі з’єднань (ієрархічна кластеризація);
- на основі розподілу;
- нечітка кластеризація.

1.2.1 Кластеризація на основі центроїдів

В інтелектуальному аналізі даних метод кластеризації на основі центроїдів (k-середніх) є найпростішим. Він заснований на властивості близькості об’єктів даних до вибраного центрального значення, іншими словами, на мінімізації розкиду образів датасету навколо центру кластера. Основна ідея такого роду алгоритмів кластеризації полягає в розгляді середнього геометричного місця точок у просторі об’єктів даних як центру відповідного кластера [8].

Цей метод кластеризації передбачає ітеративне вимірювання відстані між об’єктами кластерів та характерними центроїдами, використовуючи різні метрики, наприклад, евклідову відстань або відстань Мінковського.

Основною проблемою є визначення кількості кластерів для початку ітераційного процесу переміщення центроїда кластера. Вибір може бути інтуїтивним або визначеним аналізом даних.

Метод поділу – це широко використовуваний підхід кластеризації для виявлення та оптимізації великих наборів даних, як то сегментація зображень або клієнтів. Приклад кластеризації методом k-середніх наведено на рисунку 1.

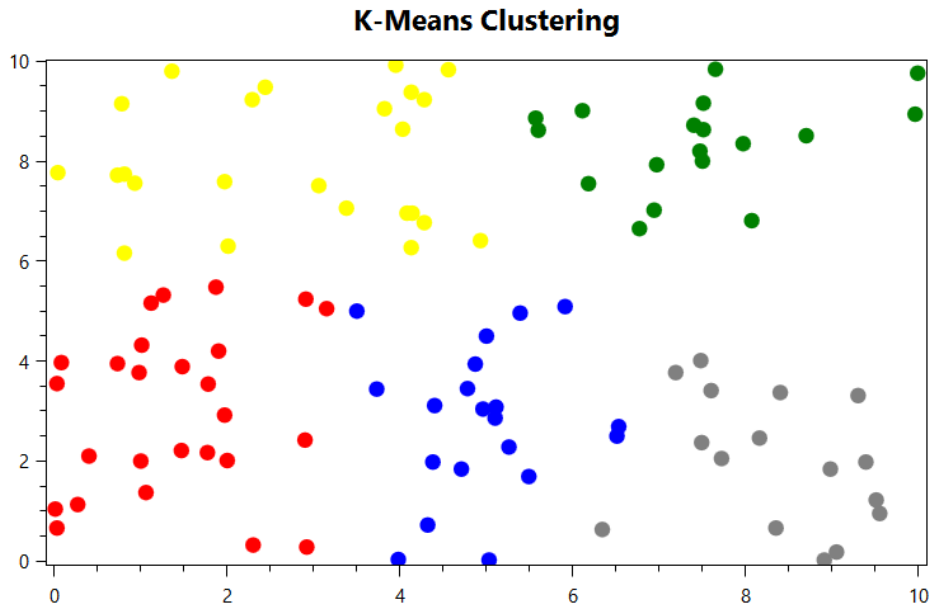


Рисунок 1 – Кластеризація методом k-середніх

1.2.2 Ієрархічна кластеризація

Ієрархічна кластеризація базується на принципі, згідно з яким сусідні об’єкти з’єднуються в ієрархічні структури залежно від їхньої близькості. Існує два типи підходів для створення ієрархічної декомпозиції: 1) агломеративний підхід (або «знизу вгору»); 2) дивізимний підхід (або «згори вниз») [1].

Агломеративний підхід [7]. Кластери являють собою великі ієрархічні структури, розділені максимальною відстанню, необхідною для з’єднання частин одного кластера.

Головна ідея методу полягає в наступному. Спочатку кожен об’єкт вважається окремим кластером. Два найбільш близьких об’єкти об’єднуються, і утворюється новий кластер. Процедура триває, поки всі об’єкти не будуть об’єднані в один кластер.

Результати кластеризації агломеративним методом зображують у вигляді дендрограми, де вісь X представляє об’єкти даних, а вісь Y – відстань, на якій зливаються кластери. Подібні об’єкти даних розташовані на мінімальній відстані в одному кластері, а об’єкти з різними властивостями розміщуються на більшій відстані в ієрархії. Приклад дендрограми наведено на рисунку 2.

Дивізимний підхід [1, 12]. Ієрархічний метод будує ієрархію кластерів, починаючи з одного великого кластера, який містить всі об’єкти, а потім поступово намагається розділити дані на менші підкластери на основі логіки завершення або об’єкта, після якого не буде подальшого поділу даних. Логіка завершення ґрунтується на метриках, що обчислюються для точок даних усередині кластера.

Ітераційно ми розбиваємо дані одного великого кластера на n менших кластерів, до яких тепер належать об’єкти. Ітерація триває до тих пір, поки не буде виконана умова завершення або доки кожен кластер не міститиме один об’єкт (див. рис. 3).

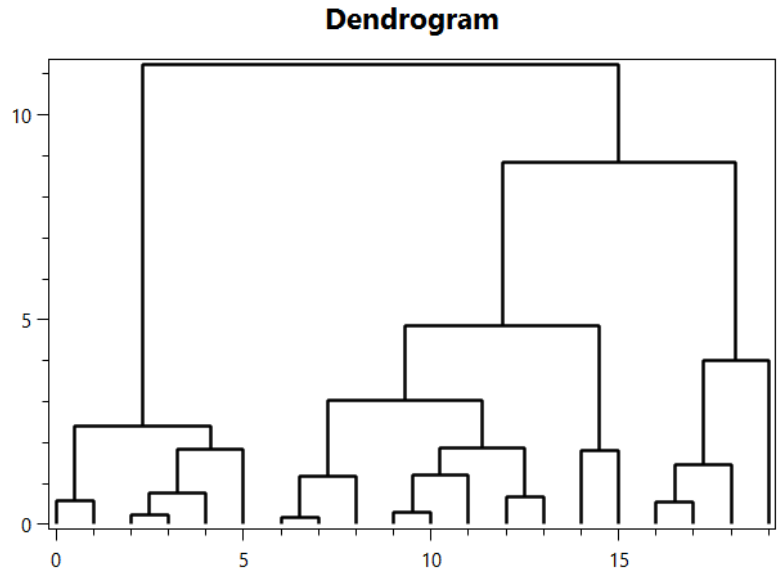


Рисунок 2 – Дендрограма ієрархічної кластеризації

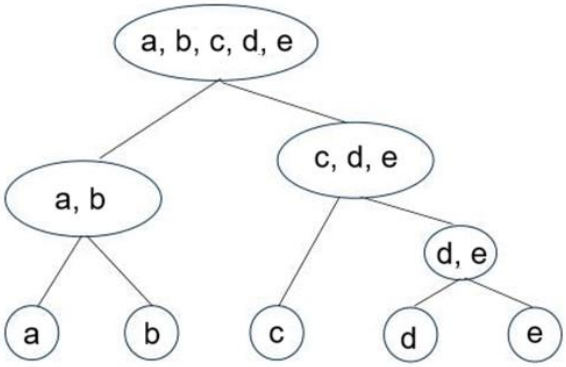


Рисунок 3 – Ієрархічна кластеризація «згори вниз»

1.2.3 Кластеризація на основі щільності (методи на основі моделі)

Розглянуті вище методи кластеризації залежать від метрики відстані (міри близькості або подібності), на цій метриці базується визначення кластера.

Кластеризація на основі щільності заснована на ідеї об’єднання в один кластер об’єктів в області їх найбільшого згущення в просторі даних. У цьому методі кожен кластер буде продовжувати зростати до тих пір, поки щільність в околицях кожної точки даних перевищує деякий поріг у межах даного кластера.

Околиця даного кластера повинна містити принаймні мінімальну кількість точок.

Отримані кластери завдяки однаковій щільності об’єктів мають максимальний ступінь однорідності, однак різняться за формами та розмірами. Цей підхід до кластеризації ефективно враховує шум і викиди в наборах даних, крім того, не потребує завдання кількості кластерів.

Приклад кластеризації на основі щільності наведено на рисунку 4.

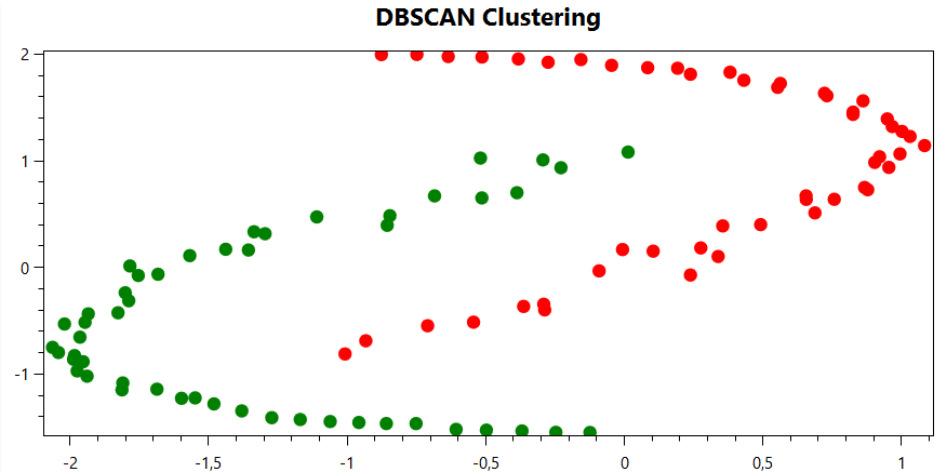


Рисунок 4 – Кластеризація на основі щільності

1.2.4 Кластеризація на основі розподілу

На відміну від розглянутих раніше методів кластеризації, в яких використовувались метрики відстані або подібності, чи міра згущення, метод кластеризації на основі розподілу враховує іншу метрику – ймовірність. Це ймовірнісний метод, який для кластеризації об’єктів даних використовує статистичні розподіли.

Основна ідея методу полягає в тому, що дані, які підпорядковуються одному розподілу, належать до одного кластеру. Кластеризація на основі розподілу створює та групує об’єкти даних на основі їх вірогідної приналежності до одного і того самого розподілу ймовірностей у даних, наприклад, розподілу Гауса або біноміального.

Хоча методи кластеризації на основі розподілу має перевагу перед методами на основі близькості та центроїда з точки зору гнучкості, правильності та форми побудованих кластерів, вони мають і недолік, пов’язаний з властивостями реальних даних. Застосовувані на практиці датасети не завжди належать до попередньо визначеного розподілу, як синтетичні чи змодельовані дані. В такому випадку результати обробки будуть надмірними.

Приклад кластеризації на основі розподілу наведено на рисунку 5.

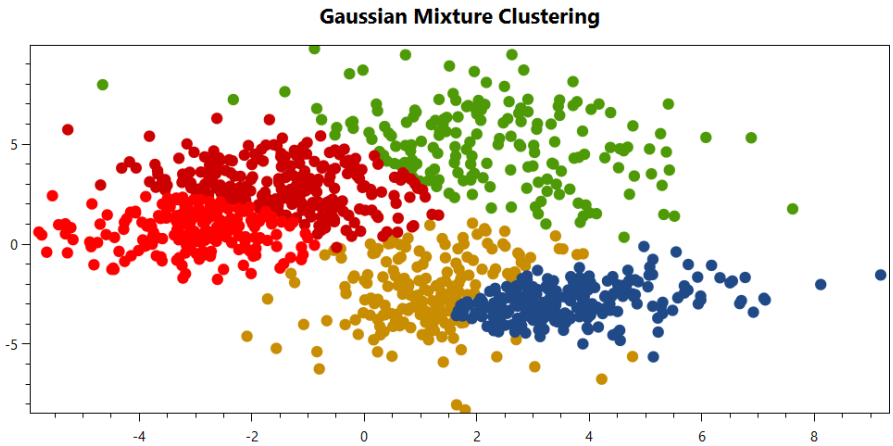


Рисунок 5 – Кластеризація на основі розподілу

1.2.5 Нечітка кластеризація

Основна ідея цього типу методів кластеризації полягає в тому, що для більш розумного визначення відношення належності між об’єктами дискретне значення мітки належності, {0, 1}, змінюється на безперервний інтервал [0, 1]. Замість того, щоб жорстко призначати кожен об’єкт даних до одного конкретного кластера, цей метод визначає ступінь належності об’єкта до кожного з кластерів. В результаті кожна точка даних має набір значень належності до кожного кластера. Це дозволяє враховувати ситуації, коли точка даних може з певною ймовірністю належати до кількох кластерів одночасно.

Для кожної точки даних обчислюється відстань до кожного центра кластера. На основі відстані обчислюється значення належності точки до

кожного кластера. Чим ближче точка до центра кластера, тим вище її значення належності до цього кластера.

Нечітка кластеризація узагальнює *метод кластеризації на основі поділу***,** дозволяючи об’єкту даних бути частиною більш ніж одного кластера.

Переваги нечіткої кластеризації:

– більш гнучкий підхід, ніж жорстка кластеризація;
– дозволяє враховувати невизначеність в даних;
– може виявити більш складні структури в даних.

1.2.6 На основі обмежень

Метод кластеризації на основі обмежень – це підхід, який дозволяє враховувати додаткову інформацію про дані у вигляді обмежень. Ці обмеження можуть виражати різні типи знань, наприклад, обмеження на розмір кластера, яке обмежує мінімальну або максимальну кількість об’єктів у кластері, кількість кластерів. Обмеження визначаються як бажані властивості результатів кластеризації або очікування користувача щодо сформованих таким чином кластерів.

Включення обмежень у процес кластеризації дозволяє отримати більш точні та інтерпретовані результати, які відповідають апріорним знанням про об’єкти даних.

Алгоритми кластеризації на основі обмежень модифікують стандартні методи кластеризації, такі як k-середніх, щоб врахувати задані обмеження. Наприклад, внесення змін в ітеративний процес кластеризації, щоб гарантувати виконання обмежень.

Переваги кластеризації на основі обмежень [8, 9]:

– підвищення точності кластеризації;
– отримання більш визначених результатів;
– можливість врахування експертних знань.

1.3 Дослідження алгоритмів кластерного аналізу

Дослідження об’єктів даних за допомогою алгоритмів кластеризації ставить за мету виявлення шаблонів у даних, виявлення нетипових (аномальних) даних, пошуку викидів. Треба враховувати, що реальні дані, як правило, неструктуровані, неоднорідні та містять шуми. Тому в залежності від виду наборів даних дослідник має вибрати алгоритм. Якщо алгоритм кластеризації відповідає досліджуваним даним, то його застосування допоможе знайти цінну інформацію для галузі, до якої відносяться дані та задача їх аналізу.

Постановка задачі. Задана навчальна вибірка, початковий набір X довжини n нерозмічених об’єктів даних: $X = \{x_1, x_2, \ldots, x_n\}$. Треба побудувати модель, яка буде розподіляти об’єкти за класами, використовуючи задану метрику $\rho(x_i, x_j)$, i, j = 1, 2, …, n, відстань між об’єктами ($\rho(x_i, x_j) = d(x_i, x_j)$), тобто знайти

- множину кластерів Y, |Y| = K;
- алгоритм кластеризації, включення кожного об’єкта xi $\in$X до того чи іншого кластера: A: X $\rightarrow$ Y.

Як було зазначено раніше, задача кластеризації відноситься до задач машинного навчання без вчителя.

На рисунку 6 точками зображено набір образів об’єктів даних, помічено образи p, q, S та відстані d_1 (p, S) між p та S, d_2 (q, S) між точками q та S.

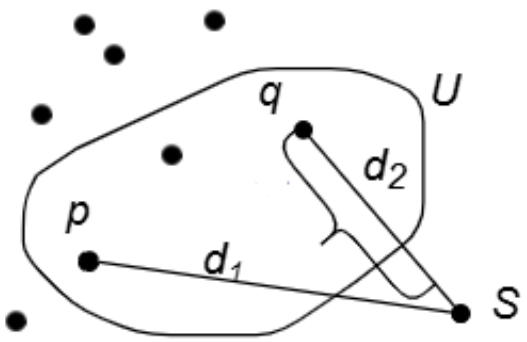


Рисунок 6 – Відстані d_1 між образами даних q та S, d_2 між q та S

1.3.1 Критерії якості кластеризації

Загальні критерії. Часто задача кластеризації не має чіткого єдиного розв'язку: для одних й тих самих даних можуть бути отримані різні результати кластеризації. Крім того, самі кластери можуть утворювати різні типи структур.

Критерії для оцінки якості розбиття образів за класами:

– середня внутрішньокластерна відстань

$$F_0 = \frac{\sum_{i<j} \rho(x_i,x_j)}{\sum_{i<j}(x_i,x_j)}, F_0 \to \min,$$

$x_i, x_j \in$ одному кластеру y_k, n_k – потужність k-го кластера, $1 \le k \le K$;

– середня міжкластерна відстань

$$F_1 = \frac{\sum_{i<j}[A_i \neq A_j]\rho(x_i,x_j)}{\sum_{i<j}(x_i,x_j)}, F_1 \to \max,$$

$x_i, x_j \in$ різним кластерам;

– відношення відстаней $(F_0 / F_1) \to \min$.

Використовуючи ознаки об'єктів x_i, $x_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]^T$, i=1, 2, …, n, p – кількість ознак об'єкта, можна обчислити:

– центри кластерів m_k, k = 1, 2, …, K:

$$m_k = (1/ n_k) \sum_{i=1}^{n_k} x_i \text{ , } k = 1, 2, \ldots, K;$$

– середню внутрішньокластерну відстань:

$$\Phi_0 = \sum_{k=1}^{K}(1/n_k) \sum_{j=1}^{n_k} \rho(x_j, m_k), \Phi_0 \to \min;$$

– середню міжкластерну відстань:

$$\Phi_1 = \sum_{i=1}^{K} \sum_{j=1}^{K} \rho(m_i, m_j),\ \Phi_1 \to \max;$$

– відношення відстаней: $(\Phi_0 / \Phi_1) \to \min$.

На практиці існують задачі, в яких можна задавати відстань (метрику), але не задавати простір ознак.

Рекомендується використовувати метрику $(F_0 / F_1) \to \min$, якщо відомі тільки відстані між об'єктами, або метрику $(\Phi_0 / \Phi_1) \to \min$, якщо заданий простір ознак об'єктів даних. Для точок на площині є дві ознаки – координати x, y, тоді розглядається площина як двовимірний простір точок даних.

Коефіцієнт силуету – це метрика, яка використовується для оцінки якості кластеризації на основі середньої відстані між точкою даних та іншими точками даних у тому самому кластері та середньої відстані між різними кластерами. Вона визнає, наскільки сильно об'єкти в одному кластері схожі один на одного, та чи вони відрізняються від об'єктів в інших кластерах. Ця метрика дозволяє розробити оптимальну кількість кластерів і оцінити, чи є кластери якісно розділеними [8, 20].

Формула коефіцієнта силуету. Для кожного об'єкта x_i у множині даних обчислюється коефіцієнт силуету s_i за формулою:

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}},\ \text{якщо } |C_I| > 1;\ \ s_i = 0,\ \text{якщо } |C_I| = 1,$$

де a_i – середня відстань між об'єктом x_i та всіма іншими об'єктами в тому ж кластері (щільність в середині кластера);

b_i – мінімальна середня відстань між об'єктом x_i та об'єктами інших кластерів (найближчий сусідній кластер);

$|C_I|$ – потужність кластера з індексом I.

Коефіцієнт силуету s_i має значення в інтервалі від -1 до 1:

– $s_i \approx 1$: об'єкт дійсно належить своєму кластеру;

– si ≈ 0: об'єкт на межі двох кластерів;

– si ≈ -1: об’єкт, ймовірно, неправильно віднесений до кластера; s_i = -1: погана кластеризація.

Індекс силуету (Silhouette Index) – це загальна метрика для оцінки якості кластеризації всієї множини даних. *Середнє значення* коефіцієнтів силуету всіх об’єктів використовується як індекс силуету.

Індекс силуету варіюється від -1 до +1:

– значення +1 відповідає ідеальній кластеризації: об’єкти чітко відокремлені між кластерами та добре згруповані всередині кластерів;
– значення 0 означає, що об’єкти знаходяться на межі між кластерами;
– значення -1 відповідає поганій кластеризації: об’єкти, ймовірно, належать до іншого кластеру.

Значення індексу силуету, близькі до 1, вказують на гарну кластеризацію. Значення в межах 0.5-1 зазвичай вказують на високоякісну кластеризацію. Значення, нижчі за 0.5, сигналізують про можливі проблеми з кластеризацією (наприклад, неправильна кількість кластерів).

Індекс Девіса-Боулдіна (Davies-Bouldin Index, *DB*) використовується для оцінки якості кластеризації. Він базується на оцінці відстаней між кластерами та внутрішньокластерної розбіжності [8, 14].

Для *K* кластерів індекс Девіса-Боулдіна обчислюється як:

$$DB = \frac{1}{K}\sum_{i=1}^{K} \max_{i \neq j}\left(\frac{S_i + S_j}{M_{ij}}\right),$$

де S_i – внутрішньокластерна розбіжність для кластеру i, обчислюється як середня відстань між точками кластеру та його центром:

$$S_i = \frac{1}{|C_i|}\sum_{x \in C_i}\|x - m_i\|;$$

C_i – множина точок в і-му кластері, m_i – центр кластеру *i*, | C_i | – потужність *i*-го кластеру;

M_{ij} – міжкластерна відстань між центрами кластерів *i*, *j*, зазвичай евклідова відстань: $M_{ij} = \| c_i - c_j \|$;

$\frac{S_i+S_j}{M_{ij}}$ – міра схожості між двома кластерами *i*, *j*;

$\max_{i\neq j}$: обирається найбільше значення схожості між кластером *i* та всіма кластерами *j*.

Індекс Девіса-Боулдіна є невід'ємним і теоретично варіюється від 0 до нескінченності. Значення 0 відповідає ідеальній кластеризації: всі об'єкти добре згруповані, а кластери добре розділені. Значення DB ≈ 0-1 вказують на гарну кластеризацію, а DB > 1.5 часто свідчать про слабку кластеризацію.

Висновки. Індекс силуету та індекс Девіса-Боулдіна варто використовувати разом, щоб отримати повніше уявлення про якість кластеризації. Високий індекс силуету та низький індекс Девіса-Боулдіна свідчать про якісну кластеризацію. Якщо ж значення обох індексів погані, то варто перевірити кількість кластерів та (або) метод кластеризації (можливо, потрібен інший підхід, наприклад, DBSCAN або агломеративна кластеризація).

1.3.2 Алгоритм кластеризації метода k-середніх (K-Means)

Задано набір з n об'єктів даних X = {x_1 , x_2 , …, x_n }. K-Means використовує евклідові відстані між точками як критерій для формування кластерів. В алгоритмі заздалегідь визначена кількість кластерів, K [16, 17]:

а) визначаємо початкові центри K кластерів: $m_1^{(0)}$, $m_2^{(0)}$, …, $m_K^{(0)}$ як зважене середнє за кожною ознакою;

б) знаходимо відстань від кожного об'єкта x_i до центрів кластерів $m_1^{(0)}$, $m_2^{(0)}$, …, $m_K^{(0)}$; об'єкт x_i відносимо до кластеру, відстань до якого мінімальна – повторюємо цей крок, доки всі точки даних не будуть призначені будь-якому кластеру: i=1, 2, …, n;

в) розраховуємо нові положення центрів кластерів $m_i^{(1)}$, i=1, 2, …, K – ітеративно може відбутися зміна значень K центрів і їхнього розташування;

г) кроки б), в) повторюємо, поки розбиття на кластери змінюються: $m_i^{(j+1)} \neq m_i^{(j)}$, i=1, 2, …, K, j > 0;

д) якщо центри більше не рухаються, кластеризація закінчена.

В результаті всі об’єкти приєднані до K отриманих кластерів.

Для тестування алгоритму обрано невеликий класичний набір даних про квіти іриси Р. Фішера (1936 р.), зображений на рисунку 7, який використовується для оцінки методів класифікації. На рисунку 8 зображено результат роботи алгоритму k-середніх (K-Means), застосованого для набору класичних даних Фішера. Кластеризація датасету відповідає визначеній для нього класифікації: кожен з побудованих кластерів містить образи з одного класу ірисів [18].

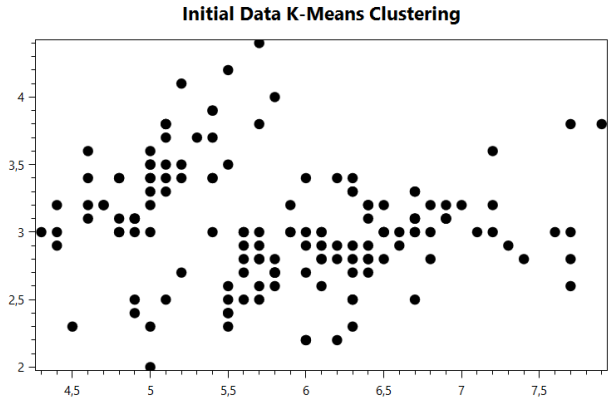


Рисунок 7 – Зображення набору класичних даних про іриси Р. Фішера

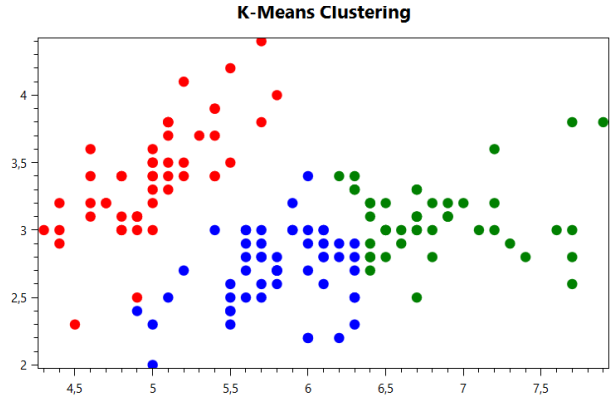


Рисунок 8 – Результат роботи алгоритму K-Means для датасету Р.Фішера

Особливістю алгоритму k-середніх є те, що він зручний для обробки великих статистичних сукупностей.

Перевагами алгоритму K-Means є простота, масштабованість.

Недоліки k-середніх:

– не визначає число кластерів за вибіркою, однак правильне значення К можна вибрати за допомогою обчислення міжкластерних та внутрішньо-кластерних відстаней та оцінок (F0 / F1) та (Ф0 / Ф1);
– повільна збіжність центрів до рівноважного стану для вибірок великих об’ємів;
– чутливість до шуму та викидів: коректне розбиття вдається досягти для яскраво виражених гаусівських наборів.

1.3.3 Алгоритм кластеризації DBSCAN

Алгоритм DBSCAN (абревіатура від «Density based spatial clastering of Application with Noise») був розроблений в 1996 р, але часто використовується сьогодні [8-10, 13, 14].

В основі DBSCAN – алгоритмічний підхід, не пов’язаний з теорією ймовірності та щільністю розподілення даних. Він базується на досить розумних евристиках, запропонованих авторами цього алгоритму.

Введемо поняття *ε-околиці об’єкта*. В метричному просторі ознак для кожного вектору x ε-околиця визначається як множина точок, віддалених від x не більше ніж на ε: $U_\varepsilon(x) = \{p \in U : \rho(x, p) \leq \varepsilon\}$, $\rho(x, p)$ – вибрана метрика простору ознак, наприклад, евклідова відстань $d(x, p)$, $\varepsilon > 0$.

Параметр ε – це *перший параметр* алгоритму DBSCAN.

Далі визначається три типи об’єктів: ядро, межа та шум.

Ядром називається той об’єкт, який містить у своїй ε-околиці не менше m інших об’єктів.

Межа – це не ядро, але міститься в околиці ядра.

Шум – це викид: не ядро, і не граничний елемент.

Алгоритм засновано на евристиках. Введені типи объектів – одна з евристик (див. рис. 9).

Другим параметром алгоритму DBSCAN є число m – кількість точок в ε-околиці.

Спочатку маємо набір об’єктів даних $X = \{x_1 , x_2 , \ldots, x_n \}$.

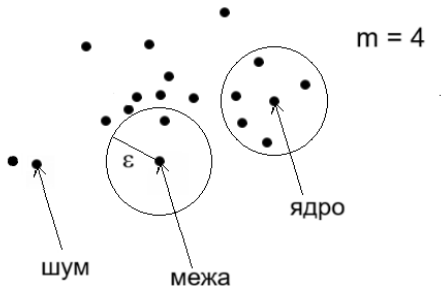


Рисунок 9 – Типи об’єктів в алгоритмі DBSCAN

Випадковим чином вибираємо об’єкт з набору та перевіряємо його ε-околицю. Якщо в ній міститься менш ніж m інших об’єктів, то вибраний об’єктів помічається як можливий шум.

Потім знову випадковим чином вибираємо об’єкт, який раніше не був відвіданий, та перевіряємо повноту його ε-околиці. Якщо в ній знаходиться не менш ніж m інших образів, то цей елемент даних помічається як ядро, і для всіх його образів – точок, що входять в ε-околицю, процедура рекурентно повторюється. Якщо об’єкт не містить достатнє число сусідів в своїй ε-околиці, то він помічається як межа.

Далі для всіх об’єктів-ядер, виключаючи відвідані раніше, процедура рекурентно повторюється. В результаті буде відокремлено перший кластер.

Потім процес повторюється спочатку, виключаючи раніше відвідані об’єкти. Аналогічно відокремлюється ще один кластер та шумові образи, тобто образи, які не увійшли в жоден з кластерів.

Алгоритм завершується, коли не залишається невідвіданих об’єктів даних. Важливо, що кількість кластерів було визначено автоматично, виходячи з параметрів ε та m.

На рисунку 11 зображено результат роботи алгоритму DBSCAN для датасету ірисів Р. Фішера (див. рис. 10). З рисунка видно, що побудовані кластери містять не всі об’єкти даних заданої вибірки: віднесені до шумів образи не потрапляють в кластери і не відображуються в області візуалізації. Відображені тільки групи щільно розташованих образів, які відповідають критерію близькості алгоритму. При цьому кластери мають майже чітку геометричну форму. Також простежується ефективність застосування алгоритму для даних, для яких кластери мають довільну форму (див. рис. 12).

Переваги алгоритму DBSCAN:

– складність алгоритму – O(n ln n), n – обсяг датасету, тому для великих n DBSCAN працює достатньо швидко (для ефективних реалізацій);
– відокремлює кластери довільної форми;
– розділяє об’єкти на ядра, межі та шуми.

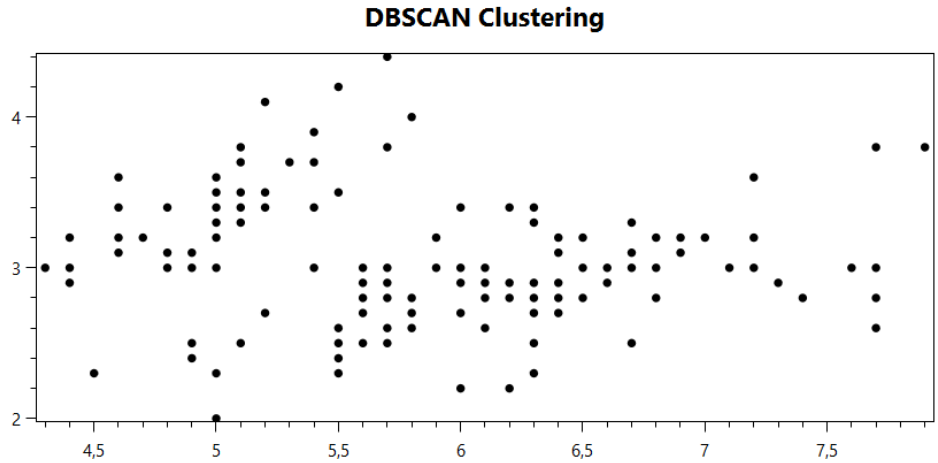


Рисунок 10 – Тестовий датасет ірисів для роботи DBSCAN

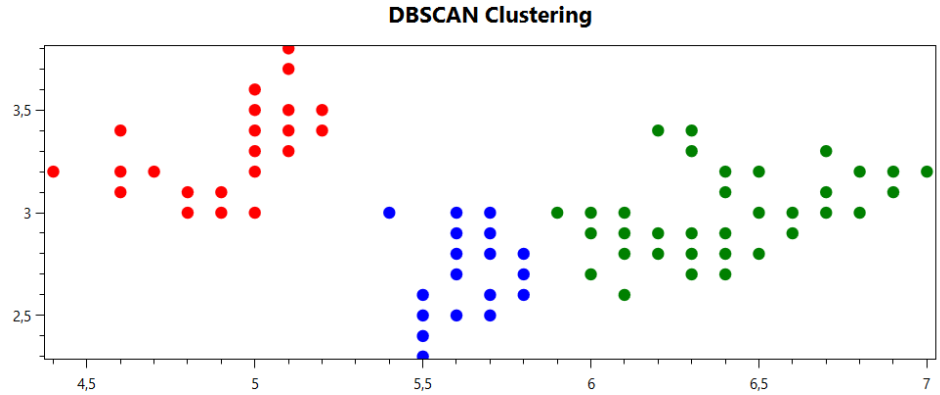


Рисунок 11 – Результат роботи алгоритму DBSCAN для вибірки ірисів

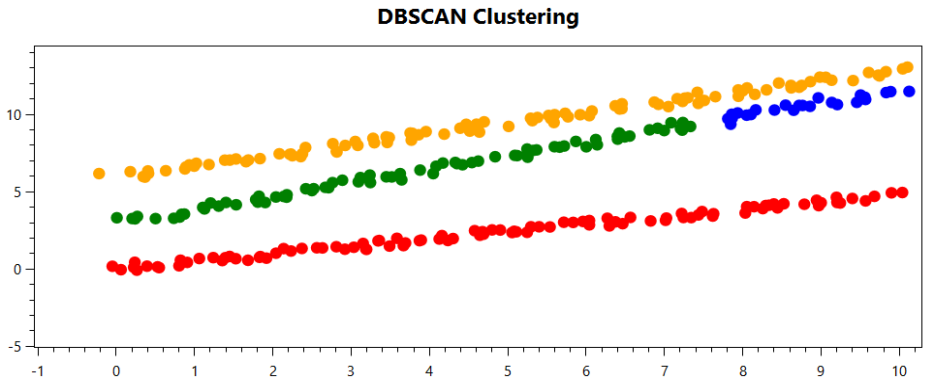


Рисунок 12 – Кластери-стрічки, побудовані алгоритмом DBSCAN

Таке розділення об’єктів на типи важливе, коли за набором даних треба визначити не тільки кластери, але й об’єкти з нетиповою поведінкою, тобто шумові або викиди. Наприклад, вони можуть відповідати небажаним (чи підозрілим) електронним листам або дивній поведінці користувачів в онлайн сервісах (виявлення зловмисників), визначення неприродності текстів, згенерованих комп’ютером і т.д. В усіх цих задачах алгоритм DBSCAN може допомогти як з групуванням об’єктів даних за деякими ознаками, так і з виокремленням дивних, нетипових об’єктів.

Алгоритм кластеризації DBSCAN допомагає суттєво спростити аналіз даних, який виконати вручну доволі складно.

1.3.4 Алгоритм ієрархічної кластеризації

Як було зазначено раніше (в пункті 1.2.2), серед ієрархічних методів кластеризації розрізняють агломеративний і дивізимний методи [1-5, 6, 22, 26].

Агломеративний підхід передбачає об’єднання найбільш близьких об’єктів або кластерів в більші за кількістю образів кластери [1, 9, 19].

Розглянемо спочатку методи групування об’єктів даних для формування кластерів. Маємо набір об’єктів даних $X = \{x_1 , x_2 , \ldots, x_n \}$ в метричному просторі ознак, тобто визначена метрика $\rho (x_i , x_j)$, $i, j = 1, 2,\ldots, n$, $i \neq j$, $\rho = d$ – евклідова відстань. Ступінь подібності об’єктів оцінюється відстанями між ними в кластері.

Метод «найближчого сусіда». Розглянемо кластер U, який включає два одиночні об’єкти p, q, та одиночний кластер S. Обчислюємо d_1 (p, S), d_2 (q, S) – евклідові відстані між p та S, q та S відповідно. Якщо $d_1 > d_2$, то об’єкти q та S є «найближчими сусідами», тоді S увійде в кластер U як «найближчий сусід» з відстанню між кластерами d_2 (див. рис. 13). Ступінь подібності об’єктів оцінюється відстанню між найближчими об’єктами кластерів.

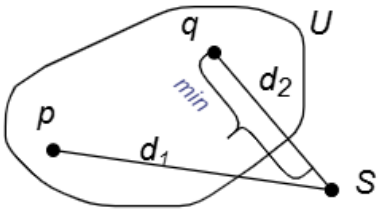


Рисунок 13 – Графічне зображення «найближчих сусідів» q, S

Метод «дальнього сусіда». Розглянемо кластер U, який включає два одиночні об’єкти p, q, та одиночний кластер S. Обчислюємо d_1 (p, S), d_2 (q, S) – евклідові відстані між p та S, q та S відповідно. Якщо $d_1 > d_2$, то об’єкти p та S є «дальніми сусідами», тоді S увійде в кластер U як «дальній сусід» з відстанню між кластерами d_1 (див. рис. 14). Ступінь подібності об’єктів оцінюється відстанню між дальніми об’єктами кластерів.

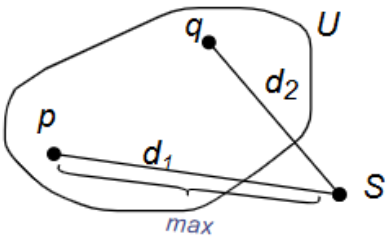


Рисунок 14 – Графічне зображення «дальніх сусідів» p, S

Метод середнього зв’язку. Обчислюємо d_1 (p, S), d_2 (q, S) – евклідові відстані між p та S, q та S відповідно. Ступінь подібності оцінюється як середня величина ступенів подібності між об’єктами кластерів, тобто як середня відстань, тоді S увійде в кластер U по відстані $d = 0.5(d_1 + d_1)$ (див. рис. 15).

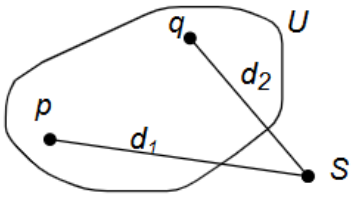


Рисунок 15 – Графічне зображення методу середнього зв’язку

Метод медіанного зв’язку. Відстань між об’єктом S та кластером U(p, q), який об’єднав p, q, визначається як відстань від центру будь-якого кластера S до середини відрізка, що з’єднує центри p і q (див. рис. 16) [17].

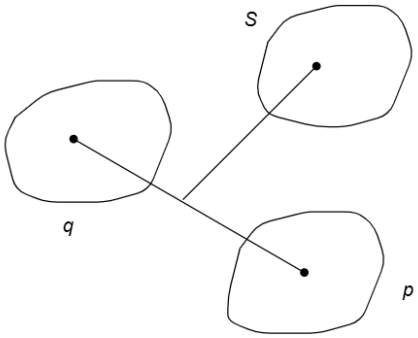


Рисунок 16 – Графічне зображення методу медіанного зв’язку

Відстань Уорда. Задана метрика – це сума квадратів відхилень:

$$\boldsymbol{V_k = \sum_{i=1}^{n_k} \sum_{j=1}^{p} (x_{ij} - x_{jk}^{\text{сер}})^2}, \qquad (1.1)$$

яку треба мінімізувати.

Алгоритм агломеративної кластеризації. Спочатку всі об’єкти даних вважаються окремими кластерами: $C_1 = \{x_1\}$, …, $C_n = \{x_n\}$, задана метрика відстані між кластерами:

а) розрахувати відстані між усіма парами кластерів (матрицю відстаней);

б) знайти два найближчих кластера за обраною метрикою відстані (наприклад, відстанню Евкліда);

в) об’єднати знайдені кластери; новому (більшому) кластеру надається менший з номерів об’єднуваних кластерів;

г) кроки а), б) повторити, поки всі об’єкти даних не будуть об’єднані в один кластер чи задану кількість кластерів, або поки не досягнуто заданий «поріг» подібності.

Модифікацією описаного алгоритму є алгоритм Уорда, який відрізняється тим, що після першого об’єднання об’єктів-кластерів для утвореного кластера обчислюється середнє квадратичне відхилення V_k за формулою (1.1). На кожному наступному кроці роботи алгоритму окремі об’єкти або кластери об’єднуються, якщо вони дають найменший приріст величини V_k .

На рисунку 17 наведено результат роботи алгоритму агломеративної ієрархічної кластеризації для датасету ірисів Р. Фішера. На рисунку 18 зображено відповідну кластеризації ірисів Р. Фішера дендрограму.

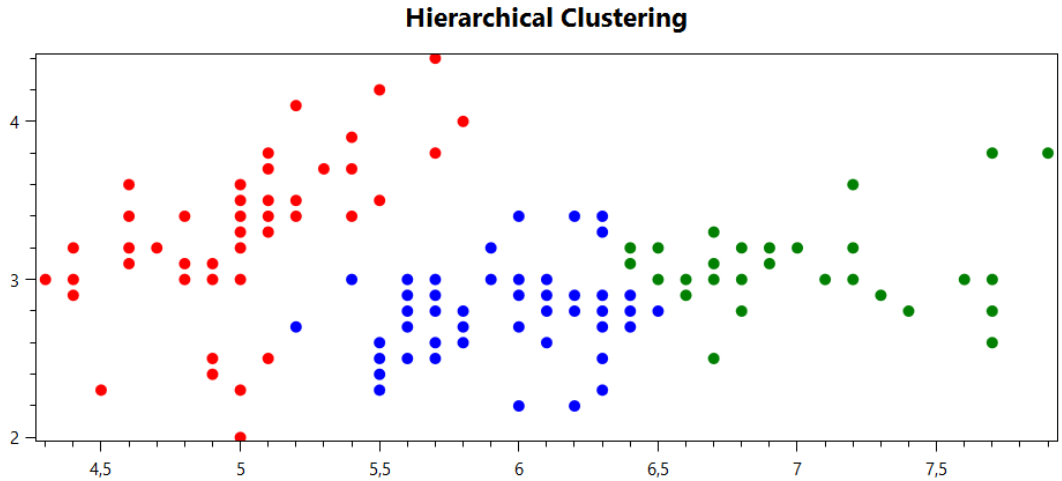


Рисунок 17 – Кластеризація агломеративним методом ірисів Р. Фішера

Недоліком агломеративної ієрархічної кластеризації може бути чутливість до шуму та викидів.

Алгоритм дивізимної ієрархічної кластеризації [1, 19]. Спочатку всі об’єкти даних належать одному кластеру: $C_1 = \{x_1 , \ldots, x_n \}$, задана метрика відстані між об’єктами та кластерами:

а) розрахувати відстані між усіма парами об’єктів (матрицю відстаней);

б) знайти два найвіддаленіших об’єкта n_i, n_j за обраною метрикою відстані (наприклад, відстанню Евкліда);

в) оцінити відстань об’єктів n_k , відмінних від n_i , n_j ($k \neq i$, $k \neq j$), до віддалених об’єктів, визначити, до якого з об’єктів n_k ближчі;

г) об’єднати близькі об’єкти в кластер; в результаті єдиний кластер розбивається на два;

д) кроки б), в), г) повторити, поки всі об’єкти даних не будуть розділені на кластери.

Перевагою алгоритму є можливість візуалізувати ієрархію кластерів (побудувати дерево).

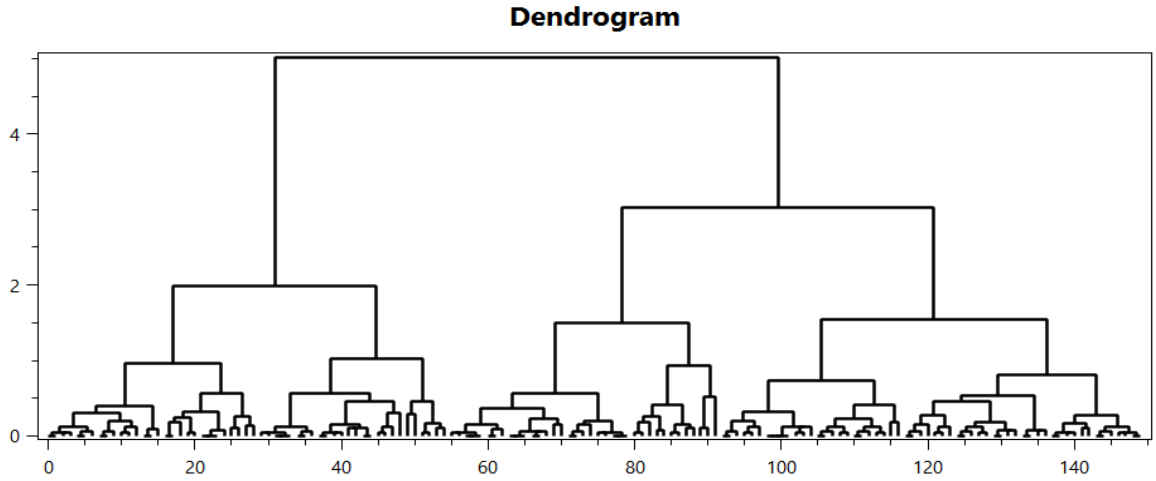


Рисунок 18 – Дендрограма кластеризації ірисів Р. Фішера

Недоліком є той факт, що алгоритм дивізимної кластеризації чутливий до вибору критерію поділу: якщо розділення відбулось, то його неможливо відмінити, навіть якщо воно було неточним.

1.3.5 Модель суміші Гауса (GMM)

Основна ідея алгоритму GMM (Gaussian Mixture Model) полягає в тому, що набір об’єктів складається з кількох розподілів Гауса, з яких генеруються початкові вибірки даних. Об’єкти даних, що відповідають одному й тому самому незалежному розподілу Гауса, вважаються приналежними до одного кластера [8].

Алгоритм використовує метод очікування-максимізації для оцінки параметрів цих розподілів і присвоєння точок даних відповідним кластерам [10].

Метод очікування-максимізації (EM, Expectation-Maximization) – це ітеративний алгоритм, який використовується для оцінки параметрів моделі, коли дані частково приховані або невідомі. У контексті моделі суміші Гауса, EM-алгоритм допомагає знайти параметри кількох розподілів Гауса, що складають суміш, навіть коли для кожної точки невідомо, до якого кластера вона належить.

Сутність EM-алгоритму в GMM. GMM моделює дані як суміш K гаусівських розподілів. Алгоритм EM оцінює [14, 20]:

– параметри кожного розподілу: середнє μk , коваріаційну матрицю Σk, вагу кожного кластера pk;
– ймовірності γik того, що точка xi належить до кластера k, використовуючи поточні значення параметрів (μ_k, Σ_k, p_k).

EM працює за двома основними етапами: E-step і M-step, які повторюються до збіжності. Основні етапи EM-алгоритму:

а) ініціалізація: задаються початкові значення параметрів (μ_k, Σ_k, p_k) для K кластерів або випадково, або через попередній алгоритм (наприклад, K-Means);
б) етап очікування (E-step): на цьому етапі для кожної точки обчислюється ймовірність γ_{ik};
в) етап максимізації (M-step): на цьому етапі оновлюються параметри моделі, щоб максимізувати правдоподібність на основі обчислених у E-step ймовірностей γ_{ik}:
    1) оновлення ваг кластерів p_k,
    2) оновлення середнього μ_k,
    3) оновлення коваріаційної матриці Σ_k;
г) повторення: E-step і M-step повторюються до збіжності, тобто поки параметри (μ_k, Σ_k, p_k) змінюються або поки не досягнуто максимального числа ітерацій.

Переваги алгоритму GMM:

– може моделювати складні розподіли даних;
– більш реалістичне визначення ймовірності приналежності;
– добре масштабується для середніх і невеликих наборів даних шляхом зміни розподілу, кількості кластерів, підтримується розвиненою статистичною наукою;
– має явну перевагу перед методами кластеризації на основі близькості та центроїда з точки зору гнучкості, правильності та форми сформованих кластерів;
– забезпечує теоретично оптимальну оцінку параметрів для моделі GMM.

Недоліки алгоритму GMM:

– алгоритм може бути важкообчислювальним (витратним) для великих і багатовимірних даних: часова складність – O(n k t d2);
– точність алгоритму залежить від початкової ініціалізації – підтвердженням цього є якісний результат роботи алгоритму GMM з початковою ініціалізацією за допомогою алгоритму K-Means (див. рис. 19).

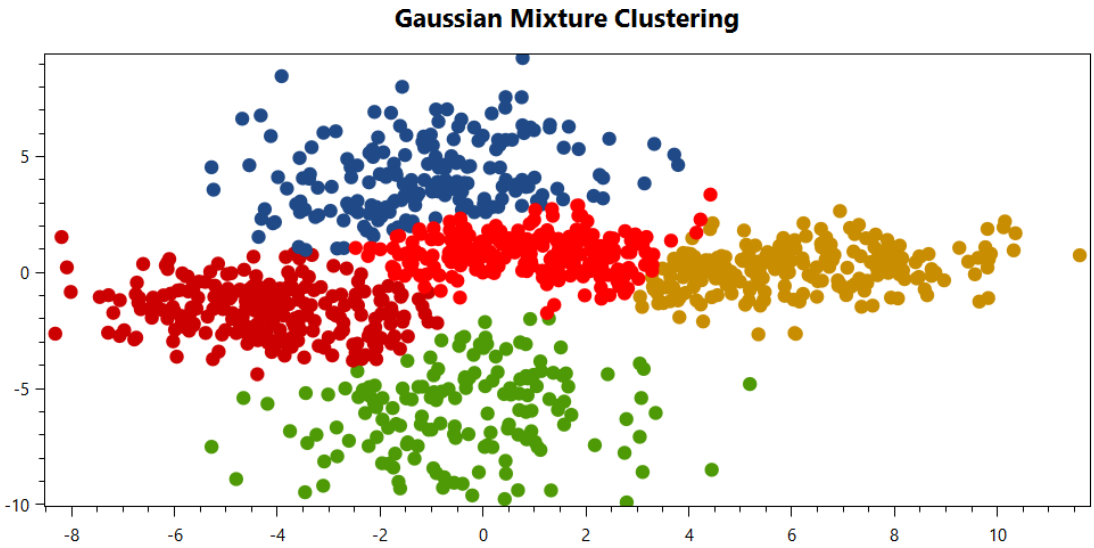


Рисунок 19 – Кластеризація алгоритмом GMM

1.4 Висновки до розділу 1

Кластерний аналіз є одним із найважливіших інструментів обробки великих обсягів даних, спрямованим на виявлення прихованих закономірностей та структури даних. Його використання дозволяє покращити якість прийняття рішень у багатьох прикладних областях.

Для практичного застосування важливою є розробка програмної системи, яка буде проводити кластеризацію заданих наборів об’єктів даних за допомогою різних алгоритмів, забезпечувати візуалізацію результатів, надавати можливість проводити їх аналіз та приймати рішення у відповідності до обраних предметних областей.

2 ІСНУЮЧІ БІБЛІОТЕКИ ТА ПРОГРАМНІ РІШЕННЯ НА C# ДЛЯ КЛАСТЕРНОГО АНАЛІЗУ

Машинне навчання змінило підхід до проблем і вилучення інформації з даних. Оскільки обсяг даних продовжує зростати, складність проблем теж зростає. Машинне навчання з його здатністю виявляти закономірності та взаємозв'язки у величезних наборах даних є засобом вирішення багатьох проблем. У сучасному технологічному світі ефективне використання машинного навчання, як одного з практичних способів додати інтелект до програмного забезпечення, надає конкурентну перевагу розроблюваним системам. Машинне навчання є незамінним інструментом, який відкриває нові можливості аналізувати дані, створювати прогнози та автоматизувати процеси прийняття рішень [7].

2.1 Фреймворк ML.NET

ML.NET – потужна та зручна система машинного навчання, розроблена Microsoft. Вона особливо вирізняється глибокою інтеграцією з іншими засобами .NET, що дозволяє розробникам використовувати наявний досвід у C# та інших мовах .NET для безпроблемного створення та розгортання моделей машинного навчання [7].

ML.NET був випущений у 2018 році як крос-платформний фреймворк з відкритим кодом для реалізації машинного навчання на .NET. Відтоді ML.NET продовжував розвиватися, пропонуючи користувачам багатофункціональне та надійне рішення для задач машинного навчання.

ML.NET відомий своєю універсальністю, надаючи підтримку для широкого кола задач машинного навчання, включаючи класифікацію, регресію, кластеризацію та системи рекомендацій.

ML.NET дає можливість включати машинне навчання до застосунків .NET у різних сценаріях. Завдяки цій можливості можна робити автоматичні прогнози, використовуючи дані, доступні програмі. Замість явного програмування програми машинного навчання використовують шаблони, наявні в даних, щоб робити прогнози [15].

2.1.1 Термінологія і концепція розробки моделі машинного навчання

Розуміння машинного навчання потребує знання фундаментальних термінів і понять.

Дані. Дані – це необроблена інформація, яка використовується алгоритмами машинного навчання для прогнозування або виявлення закономірностей. Вона може приймати різні форми: текст, зображення, числові значення тощо.

Функції. Функції – це конкретні атрибути або характеристики даних, які модель машинного навчання використовує для прогнозування. Це змінні або параметри даних, які впливають на результат.

Мітки. Мітки представляють цільову або вихідну змінну в завданні машинного навчання. У контрольованому навчанні, з учителем, модель навчена передбачати ці мітки на основі наданих функцій. У задачах кластеризації мітки представляють належність об’єктів до певних кластерів, які визначаються алгоритмом на основі схожості або відстані між об’єктами. На відміну від задач із учителем, у кластеризації спочатку немає заздалегідь визначених міток (тобто класів чи груп). Алгоритм самостійно формує групи на основі подібності даних. Після виконання кластеризації кожен об’єкт отримує мітку, яка відповідає номеру (або ідентифікатору) кластера, до якого його віднесено. Таким чином, визначаються категорії об’єктів даних, тобто мітки не мають кількісного сенсу.

Моделі. Моделі машинного навчання – це алгоритми, які вивчають закономірності та зв’язки в даних. Вони є основним компонентом, який відповідає за створення прогнозів, й ефективність моделей покращується завдяки навчанню на помічених даних.

Життєвий цикл систем машинного навчання складається з 4-х етапів.

а) аналіз проблеми;

б) збір та обробка даних: необроблені дані, необхідні для вирішення проблеми, потребують перетворення у формати, що підходять для обробки алгоритмами машинного навчання;

в) моделювання ділиться на три етапи:
   - ідентифікація характеристик або визначення міток – це значення, що впливають на прогнозування,
   - навчання моделі – вибір та застосування алгоритмів, встановлення гіперпараметрів моделі,
   - оцінка моделі;

г) розгортання моделі: впровадження моделі в програмний застосунок для прогнозування реальних даних.

Гіперпараметри встановлюються перед навчанням. Вони впливають на продуктивність, швидкість тренування та здатність моделі узагальнювати закономірності в даних. Неправильний вибір гіперпараметрів може призвести до перенавчання або недонавчання моделі. Оптимальні гіперпараметри забезпечують баланс між точністю та узагальненням моделі.

Після розгортання моделі для підтримки її ефективності необхідно донавчати модель, бо з часом передбачення можуть змінюватись під впливом закономірних змін даних та взаємозв’язків між ними.

Набір даних, призначений для обробки, необхідно розділити на два: дані для навчання моделі та дані для тестування. Навчальний набір використовується для навчання моделі, а тестовий набір зарезервовано для оцінки продуктивності моделі. Цей розподіл гарантує, що модель перевіряється на даних, які вона не бачила під час навчання, що дає більш реалістичну оцінку її можливостей.

2.1.2 Створення моделі машинного навчання за допомогою ML.NET

Розробка моделі машинного навчання може здатися складною, але з ML.NET цей процес стає доступним та інтуїтивно зрозумілим. Цей робочий процес охоплює наступні етапи [7, 15].

Завантаження даних. ML.NET підтримує завантаження даних з наступних форматів: текст (csv, tsv), Parquet (стовпчастий формат збереження даних для обробки великих обсягів структурованих даних), двійковий файл, IEnumerable<T>, набори файлів.

Перетворення. ML.NET підтримує наступні перетворення: обробка тексту, зміна схеми (структури), обробка пропущених значень, кодування категорійних значень, нормалізація.

Алгоритми. ML.NET підтримує наступні алгоритми: лінійні, посилені дерева (*Boosted Trees*), метод k-середніх, метод опорних векторів та алгоритм середнього персептрона.

Навчання моделі. Воно полягає у виклику методу ML.NET Train для створення моделі PredictionModel, яку можна використовувати для прогнозування.

Оцінка моделі. ML.NET підтримує декілька методів для оцінки точності моделі за різними метриками.

Розгортання моделі. ML.NET дозволяє експортувати модель у вигляді двійкового файлу для розгортання в застосунку .NET.

Концепція навчання. Оскільки система типів .NET не була розроблена для машинного навчання і аналізу даних, для цих задач доцільно застосовувати спеціалізовані типи.

Модель ML.NET – це об’єкт, що містить перетворення, які потрібно виконати над вхідними даними, щоб отримати прогнозований результат.

Для роботи з моделями ML.NET використовують наступні типи .NET.

Інтерфейс IDataView. Він являє собою набір даних з наступними властивостями:

– незмінні;
– придатні для перебирання курсором; перетворення таких даних виконується, тільки коли перебираються курсором;
– різнородні (можуть бути різних типів);
– схематизовані (мають чітко визначену структуру).

Абстрактний клас DataViewType. Похідними від нього є стовпці IDataView. Для векторного типу потрібна інформація про розмірності.

Інтерфейс ITransformer. Він являє собою компонент, який отримує вхідні дані, а повертає перетворені дані; в одиницю часу перетворює дані одного стовпця.

Інтерфейс IDataReader<T>. Він представляє компонент для створення даних з екземпляру T: повертає створені з T дані.

Інтерфейс IEstimator <T>. Він представляє об'єкт, який навчається на даних. Результатом навчання є перетворювач. Інтерфейс має властивість «жадібності»: кожен виклик методу Fit призводить до навчання.

Клас PredictionEngine<TSrc, TDst>. Він представляє функцію, яка застосовує перетворювач до одного рядка під час прогнозування. Якщо треба зробити прогнози для декількох рядків, то застосовують механізм прогнозування для моделі даних за допомогою методу Transform.

Завантаження даних. ML.NET надає різні API завантаження даних для імпорту даних із різних джерел, зокрема, текстові файли з роздільниками, файли CSV та бази даних. Найчастіше використовуються API *TextLoader* і *DatabaseLoader*.

TextLoader – це один із API для завантаження структурованих даних із текстових файлів. Він може обробляти різні типи даних, зокрема, цілі числа, числа з плаваючою точкою та рядки, а також може автоматично виявляти та обробляти відсутні значення. TextLoader може не підходити для завантаження інших типів даних, таких як зображення чи аудіо.

DatabaseLoader – це API, наданий ML.NET, який дозволяє завантажувати дані безпосередньо з реляційної бази даних у модель машинного навчання. Він

може підключатися до різних баз даних, включаючи SQL Server, MySQL і PostgreSQL, і отримувати дані за допомогою запитів SQL.

Щоб використовувати DatabaseLoader, спочатку потрібно визначити схему вхідних даних. Для цього створити клас, який відображається на таблицю бази даних: кожна властивість представляє стовпець у таблиці. Потім можна використовувати DatabaseLoader, щоб завантажити дані з бази даних у IEnumerable визначеного класу.

Попередня обробка даних є важливим кроком у підготовці даних для моделей машинного навчання. Він включає в себе різні методи, такі як очищення даних, нормалізація, масштабування та кодування, щоб гарантувати, що дані мають формат, який легко зрозуміти алгоритмам. Правильна попередня обробка даних може допомогти підвищити точність і ефективність моделей машинного навчання, що призведе до більш точних прогнозів.

Нормалізація та масштабування є важливими методами попередньої обробки даних, які використовуються для стандартизації діапазону та масштабу функцій у наборі даних. Стандартизація та мінімально-максимальне масштабування є двома поширеними методами, які використовуються для нормалізації та масштабування функцій у наборі даних. Стандартизація передбачає перетворення значень кожної ознаки за формулою $z = (x-\mu)/\sigma$, де x – значення ознаки, μ – середнє значення ознаки, а σ – стандартне відхилення ознаки, щоб мати середнє значення 0 і стандартне відхилення 1. Ця техніка може бути корисною, коли розподіл даних є ненормальним і містить викиди. Передбачено також масштабування значень кожної функції до діапазону від 0 до 1 за формулою: $x' = (x - x_{min})/(x_{max} - x_{min})$. Ця техніка корисна, коли діапазон значень ознаки сильно змінюється. Використання цих методів призведе до того, що жодна ознака не домінує над іншими за величиною.

Більшість алгоритмів машинного навчання не можуть використовувати категоріальні дані в необробленому вигляді, оскільки вони потребують введення числових даних. Тому категоріальні дані перетворюються в

числовий формат за допомогою різних методів кодування. Однією з найбільш часто використовуваних технік кодування є представлення категоріальної ознаки як двійкове значення в новому стовпці. Також важливо переконатися, що кодування є зворотним, щоб у разі необхідності можна було відновити вихідні категоріальні значення.

Діаграма на рисунку 20 представляє загальний ітераційний процес розробки моделі машинного навчання в ML.NET.

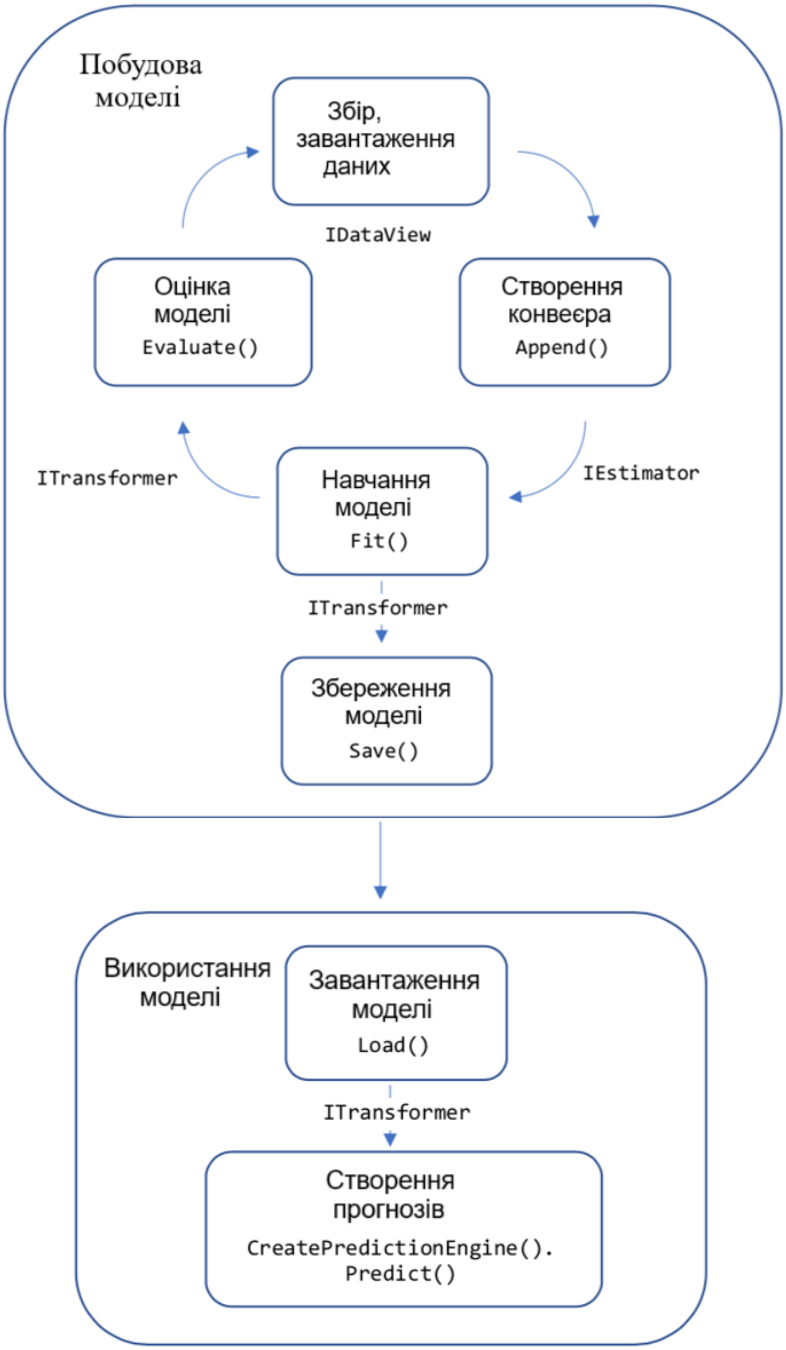


Рисунок 20 – Процес створення моделі машинного навчання в ML.NET

Спочатку необхідно зібрати та завантажити дані для навчання. У ML.NET дані представляються у вигляді об’єкта IDataView, який є масштабованим і зручним форматом для обробки великих наборів даних.

Наступним кроком є визначення конвеєра обробки даних. Конвеєр включає послідовність операцій, таких як попередня обробка даних (нормалізація, заповнення пропущених значень, кодування категорійних даних) і вибір алгоритму машинного навчання.

Модель тренується, викликавши метод Fit() на визначеному конвеєрі. Цей метод застосовує алгоритм машинного навчання до даних, щоб створити модель.

Після навчання модель оцінюється для визначення її точності. На основі отриманих результатів модель може бути покращена шляхом зміни гіперпараметрів, додавання нових характеристик або коригування алгоритму.

Завершену модель можна зберегти у двійковому форматі за допомогою методу Save(). Це дозволяє використовувати її в інших програмах або середовищах.

Для використання раніше створеної моделі її можна завантажити назад у вигляді об’єкта ITransformer за допомогою методу Load().

Щоб використати завантажену модель для виконання прогнозів, створюється об’єкт PredictionEngine, який викликає метод Predict() для виконання передбачень на основі нових даних.

2.2 Фреймворк Accord.NET

Машинне навчання відіграє ключову роль у сучасній науці про дані, забезпечуючи ефективну аналітику прогнозування та створення інтелектуальних систем. Accord.NET, відома платформа для машинного навчання в середовищі .NET, пропонує великий набір алгоритмів і засобів для побудови моделей машинного навчання [21, 24].

Accord.NET – це фреймворк для машинного навчання в середовищі .NET, який надає комплексний набір засобів для роботи з машинним навчанням, комп'ютерним зором та обробкою сигналів. Він підтримує широкий спектр алгоритмів для класифікації, регресії, кластеризації та інших задач, що робить його гнучким і потужним інструментом для створення сучасних програмних рішень.

Accord.NET відомий своєю простотою використання та надійною функціональністю, а велика документація допомагає розібратися в деталях. Фреймворк поділено на бібліотеки, доступні через автономні архіви та менеджер пакетів NuGet.

Кластеризація є методом неконтрольованого машинного навчання, який застосовується для об'єднання схожих точок даних у групи. У Accord.NET доступні алгоритми кластеризації, зокрема K-Means та Gaussian Mixture Models.

Аналіз продуктивності моделей машинного навчання є важливим кроком для оцінки їхньої ефективності. Accord.NET пропонує засоби для обчислення різних метрик, зокрема, для кластеризації використовуються внутрішньокластерна відстань, логарифм правдоподібності (Log-Likelihood), коефіцієнт силуету. У випадку Gaussian Mixture Models силуетний коефіцієнт використовується для оцінки якості кластеризації.

Коефіцієнт розраховується як різниця між середньою відстанню до точок у своєму кластері та середньою відстанню до точок найближчого сусіднього кластера, нормалізована на максимальне значення. Діапазон: від -1 до 1: чим ближче до 1, тим якісніша кластеризація.

Висновок. Accord.NET надає широкий спектр інструментів та алгоритмів для впровадження моделей машинного навчання в .NET-застосунках. Ця платформа забезпечує ефективні засоби для роботи з класифікацією, регресією, кластеризацією та оцінюванням моделей. Завдяки цьому розробники можуть створювати потужні рішення для аналізу даних і прогнозування, що відповідають сучасним викликам.

2.3 Висновок з розділу 2

Проведене дослідження програмних рішень мовою C# для кластерного аналізу показало, що хоча ML.NET і Accord.NET мають широкий функціонал для машинного навчання в цілому, але засоби для кластеризації об’єктів даних в цих бібліотеках представлені лише алгоритмами методів на основі розділу та на основі розподілу, але відсутні алгоритми ієрархічної кластеризації та кластеризації на основі щільності. Це питання потребує розгляду в наступному розділі в контексті розробки окремого застосунку.

3 РОЗРОБКА ЗАСТОСУНКУ КЛАСТЕРНОГО АНАЛІЗУ

В кваліфікаційній роботі розроблено три WPF-застосунка:

– з використанням ML.NET;
– з використанням Accord.NET;
– реалізація алгоритмів ієрархічної кластеризації та DBSCAN.

3.1 WPF-архітектура

WPF є частиною .NET Framework. Архітектура WPF (Windows Presentation Foundation) побудована на взаємодії кількох компонентів, кожен з яких виконує свою роль у забезпеченні ефективного відображення та управління графікою, інтерфейсом користувача та системними ресурсами (див. рис. 21) [15].

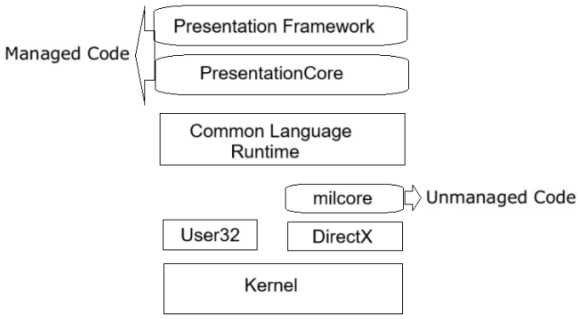


Рисунок 21 – Архітектура WPF-застосунку

CLR (Common Language Runtime) – це основа для розробки в .NET, що забезпечує управління пам’яттю, обробку помилок та інші функції, які роблять процес розробки зручнішим і продуктивнішим. CLR інтегрується з іншими частинами WPF, що дозволяє керувати пам’яттю, багатозадачністю та помилками в програмі.

Компонент PresentationFramework відповідає за створення елементів інтерфейсу користувача, таких як елементи керування, стилі, макети та вікна.

PresentationFramework залежить від PresentationCore, надаючи інтерфейс та забезпечуючи керування відображенням елементів інтерфейсу.

PresentationCore – містить базові типи, такі як UIElement та Visual, які використовуються для створення елементів управління та вікон в додатку. Цей компонент відповідає за відображення елементів UI та взаємодію з ними. Він будує абстракцію над графічними елементами, що забезпечує взаємодію з низькорівневими компонентами, такими як **Milcore**.

Milcore – є частиною некерованого коду, який забезпечує інтеграцію з DirectX для роботи з графікою. Milcore тісно пов’язаний з **DirectX**, надаючи необхідну інфраструктуру для ефективного відображення графічних елементів.

DirectX – це низькорівневий API, який дозволяє працювати з графічними елементами у WPF. Він відповідає за відтворення графіки на екрані, взаємодіє з драйверами.

User32 – є основним API для взаємодії з користувачем на рівні операційної системи та визначає, як елементи інтерфейсу користувача будуть відображатися на екрані, керує взаємодією з вікнами та елементами управління.

Ядро Windows – це низькорівневий компонент операційної системи, що обробляє вхідні дані та забезпечує їх правильну передачу до відповідних процесів і коректну роботу з елементами управління через **User32**.

Компоненти WPF призначаються для створення динамічних систем представлення, керованих даними. Перевагою WPF є можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованій на xml.

3.2 Бібліотека .NET OxyPlot

OxyPlot – це потужна бібліотека для створення графіків і діаграм у .NET-застосунках, зокрема у WPF. Вона є відкритою, кросплатформною та

легко інтегрується у проєкти для візуалізації даних [25]. Ключові характеристики OxyPlot наводяться далі.

Простота використання:

– легке підключення до WPF-застосунків через XAML або C#;
– інтуїтивний API для налаштування графіків.

Підтримка різних типів графіків:

– лінійні графіки (Line Plot);
– стовпчасті діаграми (Bar Chart);
– розподільчі діаграми (Scatter Plot);
– контурні діаграми (Contour Plot);
– гістограми, кругові діаграми, фінансові графіки тощо.

Висока продуктивність: підходить для створення графіків із великим обсягом даних.

Кросплатформність: працює з WPF, WinForms та іншими платформами .NET.

Налаштовуваність:

– широкі можливості для стилізації: кольори, лінії, легенди, шрифти;
– підтримка інтерактивності: масштабування, налаштування подій (наприклад, кліки по точках даних).

Підвищена продуктивність: навіть на пристроях із обмеженими ресурсами (наприклад, старі ПК або мобільні пристрої), графіки працюватимуть плавно.

Для інтеграції OxyPlot у WPF-застосунок її треба додати до проєкту через *NuGet* (див. рис. 22).

Після підключення бібліотеки графік додається безпосередньо у файл XAML (див. рис. 23).

У файлі .cs бібліотека OxyPlot підключається в директиві using та використовується об’єкт класу PlotModel для визначення графіка (див. рис. 24).

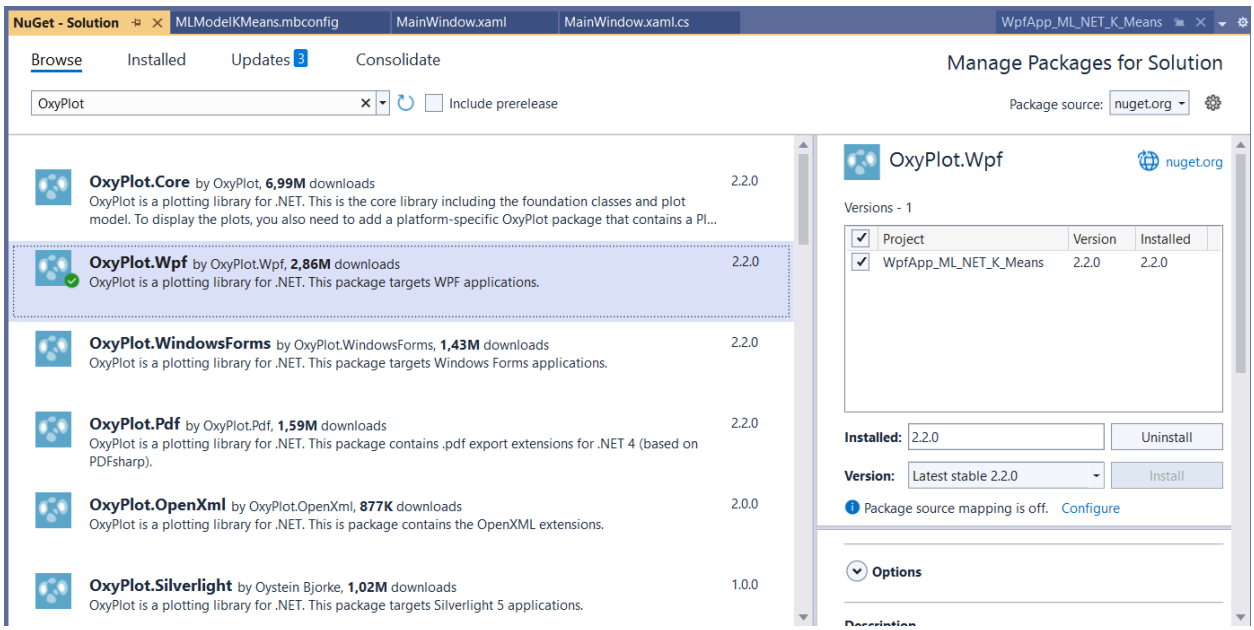


Рисунок 22 – Підключення бібліотеки OxyPlot до застосунку

Фрагмент коду файлу .xaml:

```
<Window x:Class="WpfApp_ML_NET_K_Means.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:oxy=http://oxyplot.org/wpf
<Grid>
     <oxy:PlotView x:Name="PlotView"
              Model="{Binding PlotModel}" Width="600" Height="400"
              VerticalAlignment="Bottom" HorizontalAlignment="Right"
              Margin="0,0,0,20"/>
  </Grid>
</Window>
```

Рисунок 23 – Включення бібліотеки OxyPlot у файл XAML

Фрагмент коду файлу .cs:

```
using OxyPlot;
using OxyPlot.Series;
using OxyPlot.Wpf;
……….
public PlotModel plotModel;
var plotModel = new PlotModel { Title = "K-Means Clustering" };
```

Рисунок 24 – Включення бібліотеки OxyPlot у файл .cs

3.3 Розробка застосунку кластеризації з ML.NET

Розробка починається з встановлення ML.NET у створеному WPF Application. Для інсталяції пакету «Microsoft.ML» в середовищі Visual Studio 2022 використовується менеджер пакетів «NuGet», вкладка меню Tools -> NuGet Package Manager, вкладка Browse (див. рис. 25).

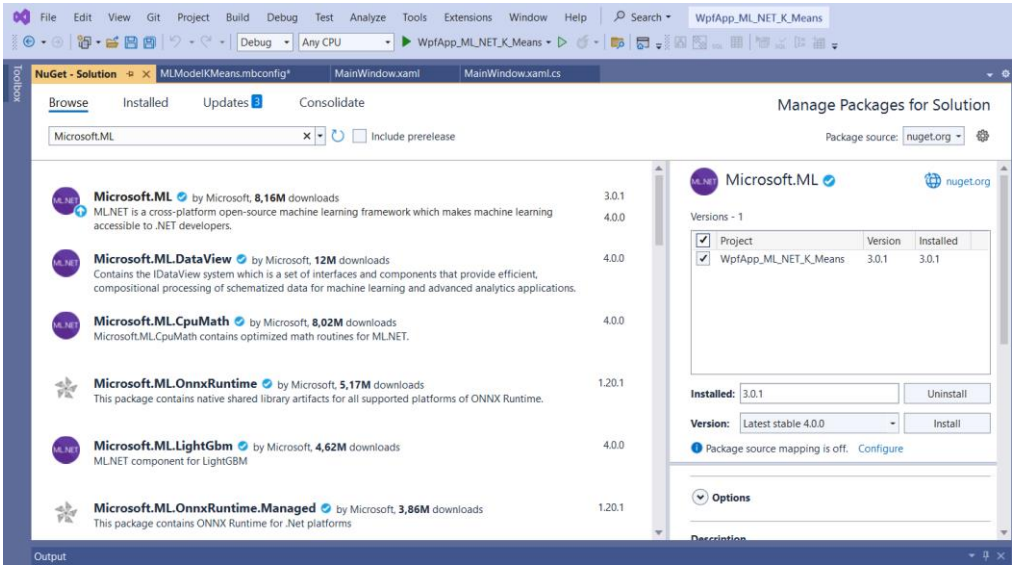


Рисунок 25 – Інсталяція пакету «Microsoft.ML»

3.3.1 Створення моделі машинного навчання в Model Builder

Для підготовки даних для алгоритмів кластеризації ML.NET до WPF-застосунку у вкладці Solution Explorer додається модель машинного навчання Machine Learning Model (див. рис. 26) і файл .mbconfig [15]. Серед доступних сценарієв немає моделі кластеризації, тому була вибрана модель класифікації (див. рис. 27). Ця модель використовувалась для створення датасету для подальшої роботи моделей кластеризації, створених без використання Model Builder. Для вибраної моделі встановлюється середовище навчання, для моделі класифікації підтримується тільки локальне навчання (див. рис. 28). Наступним кроком є завантаження в модель даних (див. рис. 29). Серед доступних форматів файлів –.csv, .tsv, .txt, файли SQL Server.

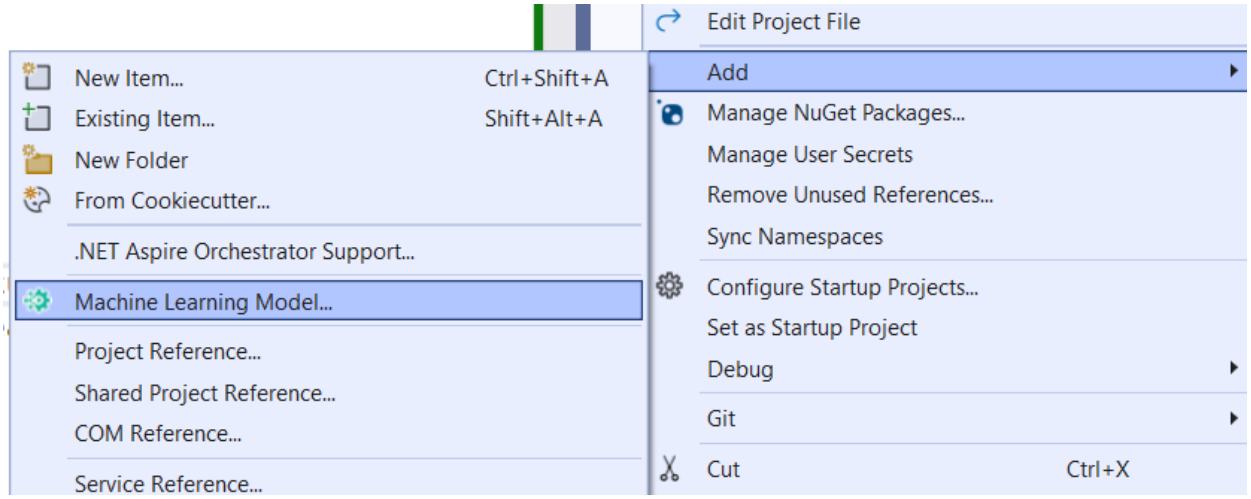


Рисунок 26 – Включення в застосунок моделі машинного навчання

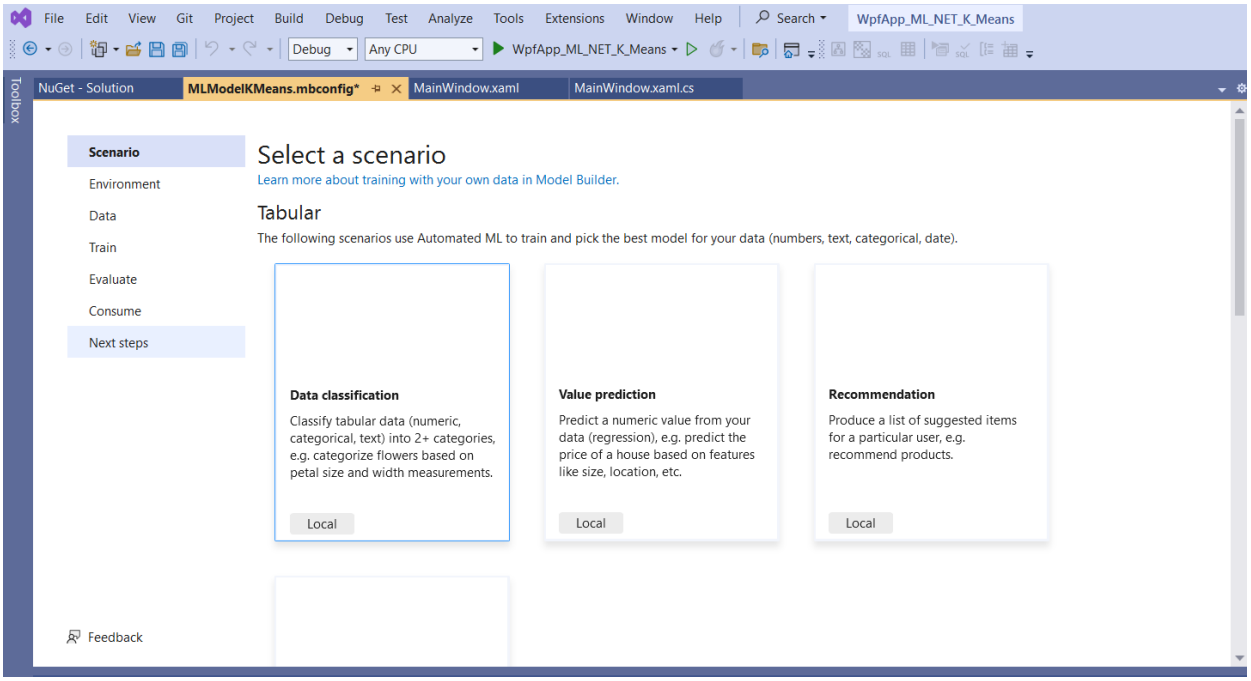


Рисунок 27 – Вибір моделі машинного навчання

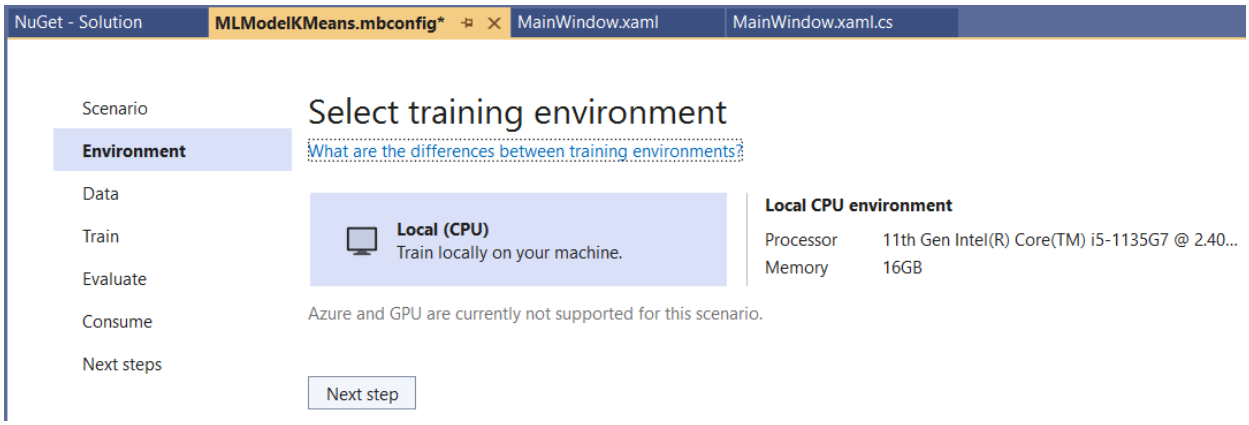


Рисунок 28 – Вибір середовища навчання моделі

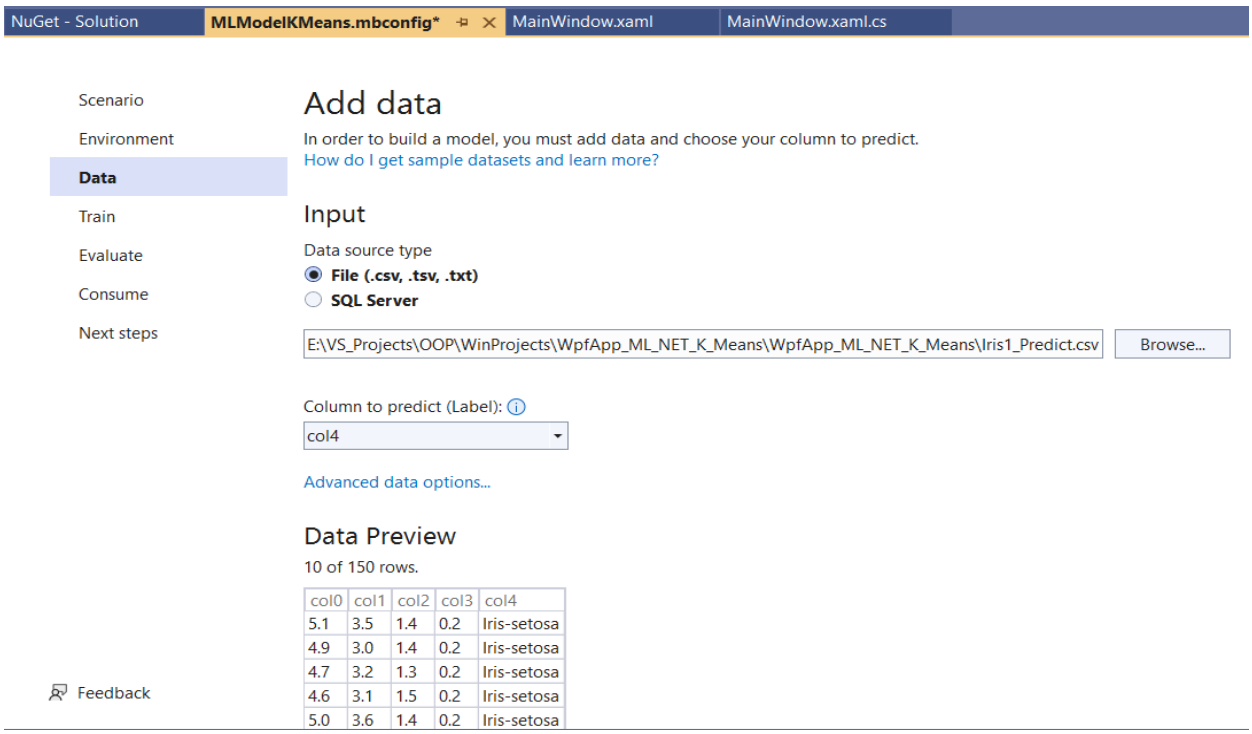


Рисунок 29 – Завантаження даних для навчання моделі

В моделі кластеризації ML.NET WPF-застосунку був використаний вже розмічений датасет – класичний набір даних ірисів Р. Фішера [18]. На сторінці Data Model Builder для розмічених даних відображуються заголовки, серед яких обираються мітки для навчання моделі.

Після завершення навчання 4 файли автоматично додаються в якості коду програмної частини WpfApp_ML_NET_K_Means.mbconfig.

Файл .consumption.cs містить вхідні та вихідні класи моделі, а також метод Predict, який можна використовувати для роботи моделі.

Файл .evaluate.cs: містить метод CalculatePFI, який викликає метод вагомості ознаки PFI для обчислення того, які ознаки мають найбільший вплив на прогнози моделі.

Файл .mlnet являє собою навчену модель ML.NET, яка є серіалізованим ZIP-файлом.

Файл .training.cs містить код для розуміння важливості вхідних стовпців для прогнозів моделі.

На кроці Consume в Model Builder надається фрагмент коду, який створює вибірку вхідних даних для моделі та використовує модель для

прогнозування цих даних. Model Builder надає також шаблони проєктів, які можна додати до рішення застосунку. Існує два шаблони проєкту: консольний застосунок та web-API), які використовують навчену модель. В даній роботі було створено окремий консольний застосунок, в якому навчена модель виконувала прогнозування для наборів даних для подальшого використання в моделях машинного навчання WPF-застосунку.

3.3.2 Розробка WPF-застосунку з ML.NET

В ML.NET методи кластеризації представлені тільки алгоритм K-Means. Для його використання розроблено class KMeansClustering, який включає основні об’єкти класів MLContext, ITransformer, IDataView для підготовки даних, навчання та роботи моделі машинного навчання. Код застосунку наводиться в Додатку А. На рисунку 30 наведено код функції **TrainModel**(), яка налаштовує і навчає модель кластеризації KMeans ML.NET. Результатом виконання функції є модель, яку можна використовувати для аналізу кластерів.

```
public void TrainModel (int clustersCount)
{
var pipeline = mlContext.Transforms.Concatenate("Features", nameof(DataPoint.Feature1), 
nameof(DataPoint.Feature2))
        .Append(mlContext.Clustering.Trainers.KMeans("Features", numberOfClusters: 
clustersCount));
        model = pipeline.Fit(dataView);
}
```

Рисунок 30 – Код основної функції **TrainModel**()

Метод mlContext.Transforms.Concatenate об’єднує значення полів Feature1 і Feature2 об’єкта DataPoint в єдиний вектор ознак Features, необхідний для роботи алгоритмів ML.NET.

Метод .Append (mlContext.Clustering.Trainers.KMeans)) додає алгоритм кластеризації KMeans до обробки даних. KMeans розбиває дані на задану кількість кластерів (clustersCount передається параметром) на основі їхніх ознак.

Метод pipeline.Fit навчає модель на основі переданих даних в dataView у форматі IDataView. Конвеєр виконує всі етапи: об'єднання ознак, навчання, кластеризацію.

Функція **Predict()** бере навчену модель і застосовує її до даних, які потрібно кластеризувати. Метод model.Transform(dataView) застосовує навчену модель (model) до набору даних (dataView).

Метод mlContext.Data.CreateEnumerable<ClusterPrediction>() перетворює табличне представлення даних IDataView у список об'єктів типу ClusterPrediction (цей клас визначає структуру даних для передбачень).

Функція **Predict()** (див. рис. 31) повертає список передбачень clusteredData – список об'єктів типу ClusterPrediction для кожного рядка вхідного набору даних. Результат кластеризації повертається у вигляді списку, де кожен об'єкт представляє передбачення – номер кластера – для одного об'єкта даних.

```
public List<ClusterPrediction> Predict()
{
   var predictions = model.Transform(dataView);
   var clusteredData = mlContext.Data.CreateEnumerable<ClusterPrediction>(predictions,
reuseRowObject: false).ToList();
   return clusteredData;
}
```

Рисунок 31 – Код функції **Predict()**

На рисунку 32 показано результат кластеризації алгоритму k-Means та значення індексу Девіса Боулдіна (DB) для оцінки роботи алгоритму: 0≤ DB ≤1. Значення DB=0.76 свідчить про достатньо якісну кластеризацію.

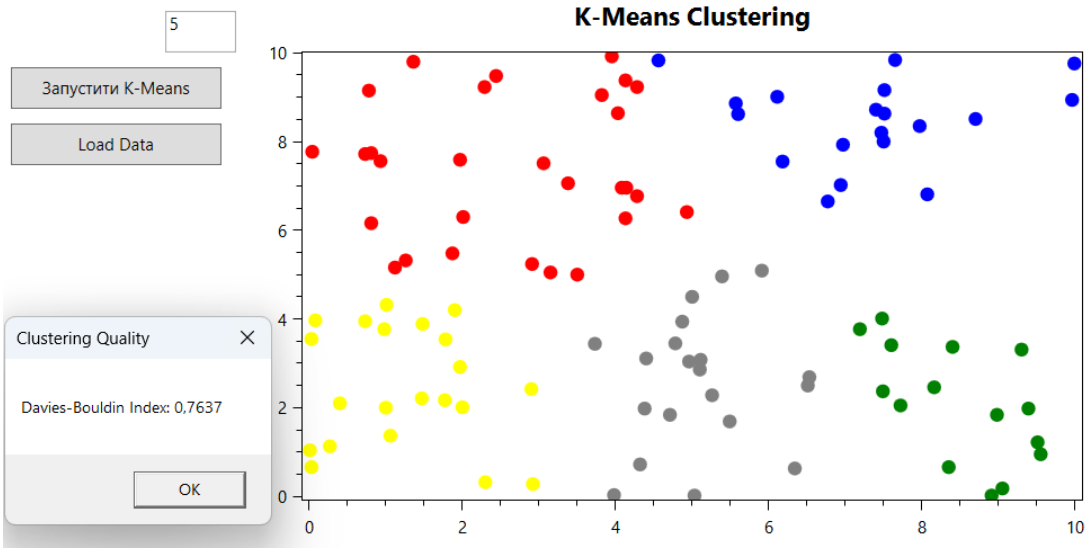


Рисунок 32 – Кластеризація k-Means з обчисленням оцінки, DB=0.76

Окрім індексу Девіса Боулдіна для оцінки роботи алгоритму використовується коефіцієнт силуету (sI): -1 ≤ sI ≤ 1. Він обчислюється для кожного об’єкта, а також середній коефіцієнт силуету. Чим ближче значення середнього коефіцієнта силуету до 1, тим більш якісна кластеризація. На рисунку 33 показано sI за результатами кластеризації алгоритму k-Means.

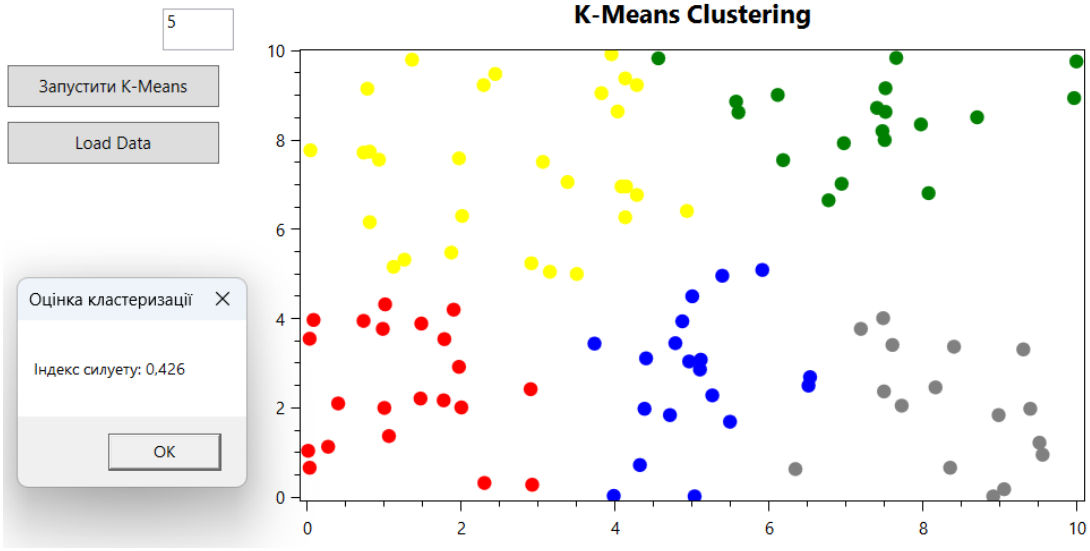


Рисунок 33 – Оцінка кластеризації алгоритмом k-Means, sI=0.426

Візуалізація результатів кластеризації в застосунку виконується за допомогою бібліотеки OxyPlot, яка підтримує масштабування та прокручування, що зручно для аналізу результатів. OxyPlot завдяки підтримці

інтерактивності дозволяє налаштовувати події, зокрема, клікаючи по точках даних, можна обирати довільний об’єкт та аналізувати його місцеположення, перетягувати, збільшуючи чи зменшуючи масштаб. Це підвищує якість візуалізації результатів.

3.4 Розробка WPF-застосунку з Accord.NET

Розробка починається з встановлення Accord.NET у створеному WPF Application. Для інсталяції пакету «Accord.NET» в середовищі Visual Studio 2022 використовується менеджер пакетів NuGet, вкладка меню Tools -> NuGet Package Manager, вкладка Browse.

В Accord.NET методи кластеризації представлені алгоритмами K-Means та Gaussian Mixture Clastering. Для їх використання треба підключити пакет машинного навчання Accord.MachineLearning, крім того, для перетворення та представлення даних в алгоритмах кластеризації додатково підключаються бібліотеки Accord.Statistics.Distributions.Multivariate, Accord.Math.

За допомогою NuGet до застосунку треба підключити бібліотеку OxyPlot, яка використовується для візуалізації результатів кластеризації.

В програмі створюється об’єкт моделі кластеризації KMeans [21, 24]. Для моделі генеруються декілька наборів, за кількістю кластерів, випадкових даних з різними гаусівськими розподілами. Всі згенеровані точки з кожного розподілу об’єднуються в один загальний масив для подальшого кластерного аналізу.

Друга модель кластеризації, що використовується в застосунку, – Gaussian Mixture Model (GMM). Для її використання створюється об’єкт моделі кластеризації GaussianMixtureModel. GMM є імовірнісною моделлю, яка припускає, що дані складаються з кількох нормальних розподілів [24].

На рисунку 34 наведено код функції, яка реалізує кластеризацію за допомогою двох вказаних моделей машинного навчання Accord.NET. Якщо

кластеризація K-Means доступна, тобто була запущена модель K-Means, її результати використовуються як початкова ініціалізація для GMM. Це прискорює процес і підвищує точність, оскільки початкові значення центрів кластерів вже відомі.

```
private void btnCompute_Click(object sender, RoutedEventArgs e)
{
    if (observations == null) return;
    var gmm = new GaussianMixtureModel(k);
    if (kmeans != null)
        gmm.Initialize(kmeans);
    GaussianClusterCollection clustering = gmm.Learn(observations);
    int[] classifications = clustering.Decide(observations);
    UpdateGraph(classifications);
    double silhouetteScore = ComputeSilhouetteScore(observations, classifications, k);
    MessageBox.Show($"Silhouette Score: {silhouetteScore:F2}", "Performance Metrics");
}
```

Рисунок 34 – Реалізація кластеризації за допомогою моделі GMM

Результати кластеризації за допомогою моделі GMM з ініціалізацією моделлю K-Means наведено на рисунку 35.

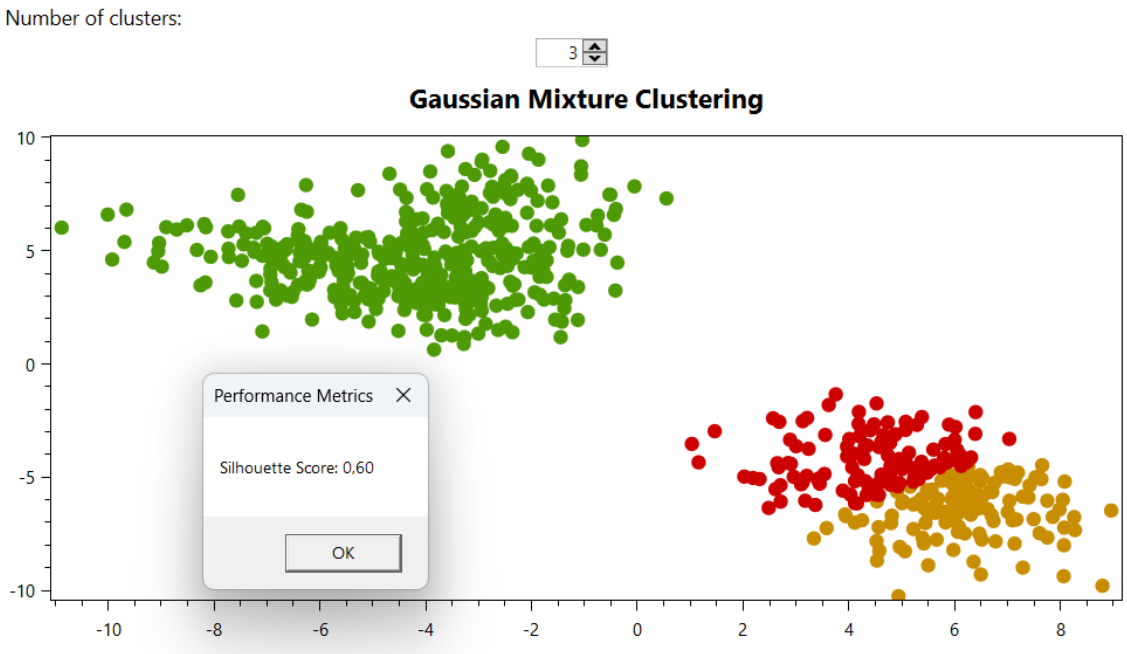


Рисунок 35 – Реалізація кластеризації за допомогою моделі GMM

Для оцінки кластеризації обчислюється показник продуктивності моделі – коефіцієнт силуету. Його значення 0.6 підтверджує достатньо хорошу роботу моделі кластеризації – достатньо правильне розбиття об’єктів даних на кластери: об’єкти в кожному
кластері розташовані досить щільно, бо підпорядковані одному закону розподілу.

3.5 Розробка застосунку для ієрархічної кластеризації та DBSCAN

Як було зазначено раніше, ML.NET та Accord.NET мають обмежену функціональність для кластерного аналізу: в них не представлені методи ієрархічної кластеризації та кластеризації на основі щільності. Для повноти дослідження розроблено третій WPF-застосунок, в якому реалізовано алгоритми агломеративної ієрархічної кластеризації та DBSCAN.

В застосунку використовується тільки бібліотека OxyPlot для візуалізації результатів. В цьому WPF Application організовано дві сторінки. Перша сторінка містить вкладку для агломеративної ієрархічної кластеризації і вкладку для зображення дендрограми. На другій сторінці представлено алгоритм DBSCAN з параметрами: Epsilon (ε-околиця) та minPoints (кількість точок в ній). Інтерфейс застосунку, 1 та 2 сторінки, представлено на рисунках 36, 37.

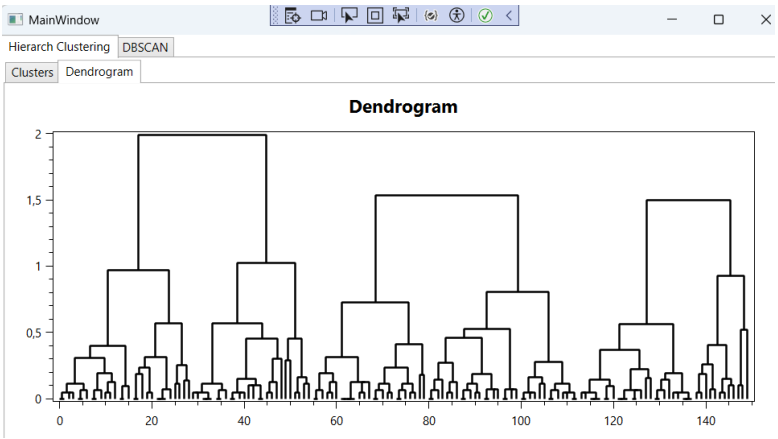


Рисунок 36 – Інтерфейс сторінки застосунка: ієрархічна кластеризація

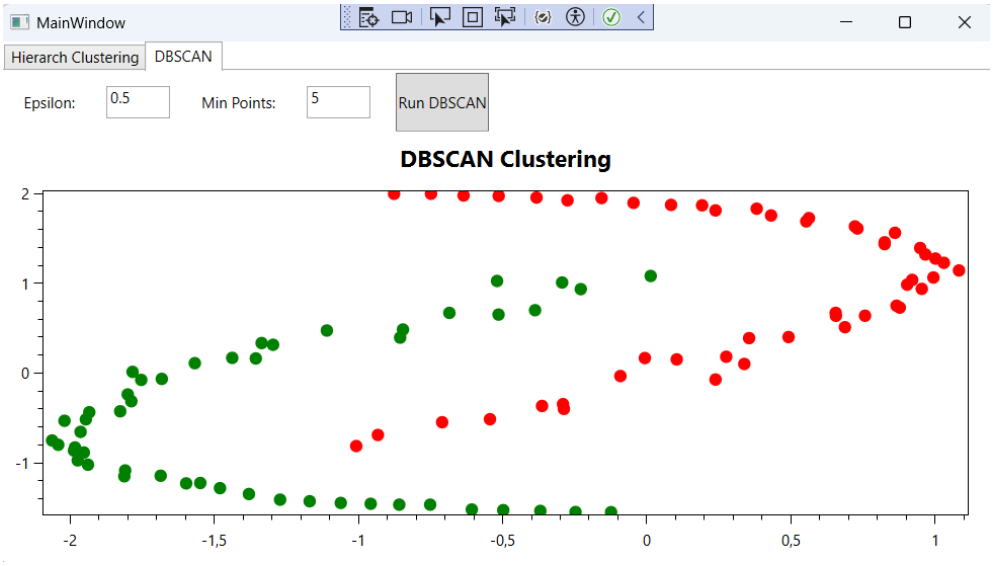


Рисунок 37 – Інтерфейс сторінки застосунка: DBSCAN

На рисунку 38 зображено результати роботи алгоритму агломеративної ієрархічної кластеризації на класичному датасеті ірисів Р. Фішера та обчислено коефіцієнт Девіса-Боулдіна. DB = 0.8, що підтверджує достатню ефективність алгоритму. Дані згруповані в три кластери, що відповідає класифікації датасету Фішера. На рисунку 36 представлено відповідну дендрограму кластеризації, на якій відображено групування даних в окремі кластери. В застосунку можна варіювати побудову дендрограми, і не з’єднувати піддерева, що відповідають різним кластерам, в одне дерево, щоб показати окремі кластери.

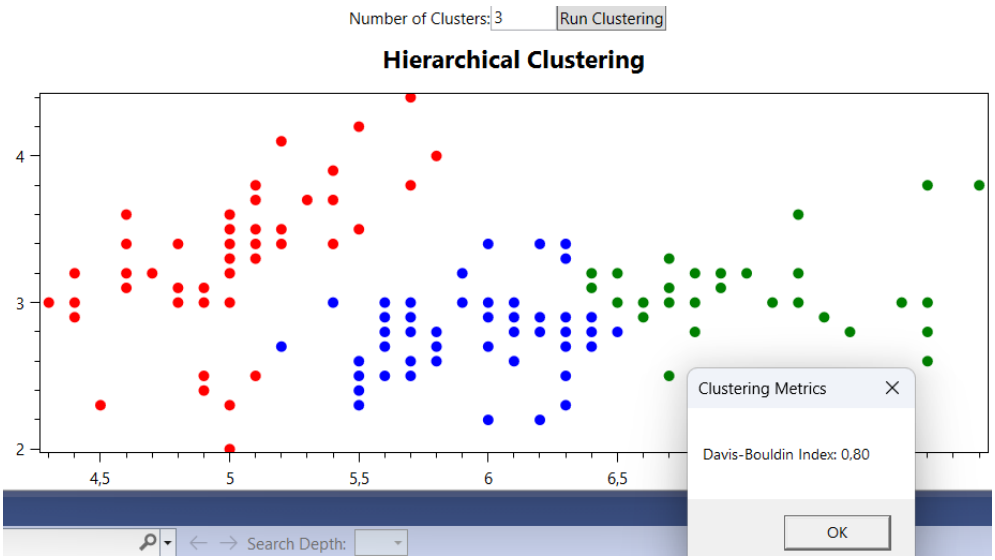


Рисунок 38 – Ієрархічна кластеризація ірисів з оцінкою

На рисунку 39 наводяться результати алгомеративної кластеризації випадкових даних, коефіцієнт Девіса-Боулдіна дорівнює 0.87. Це ще раз підтверджує хорошу роботу алгоритму кластеризації. Фактично кластери розбивають площину на області зі спільними межами.

На рисунках 37, 40 показано кластеризацію алгоритмом DBSCAN. Алгоритм відокремлює кластери з чіткими геометричними формами. Він виявляє області більшої щільності даних та поєднує такі дані в кластери.

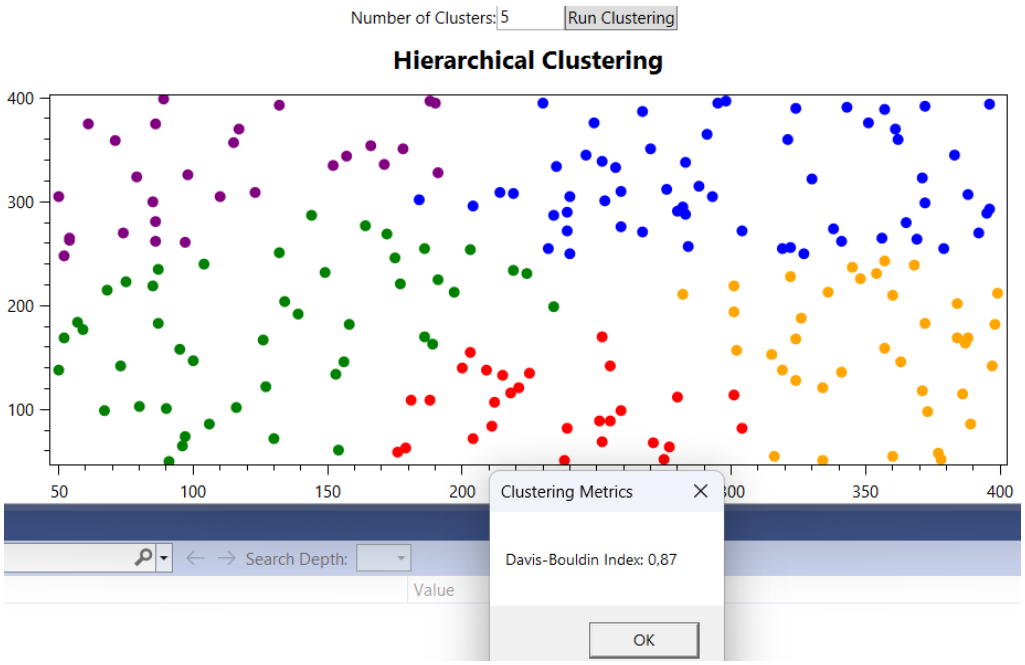


Рисунок 39 – Алгомеративна кластеризація випадкових даних з оцінкою Девіса-Боулдіна

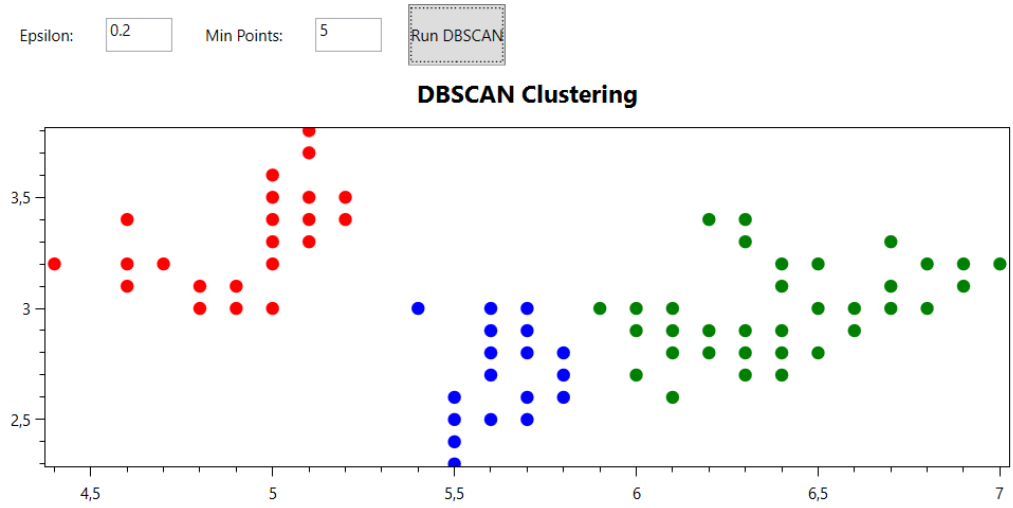


Рисунок 40 – Ієрархічна кластеризація ірисів Фішера

3.6 Висновки з розділу 3

Досліджені засоби розробки застосунків для задач кластеризації, ML.NET та Accord.NET виявилися ефективними, хоча не забезпечують повний функціонал в контексті методів кластерного аналізу, необхідний для практичного застосування. Фреймворки ML.NET та Accord.NET мають свої переваги та недоліки, кожен адаптований до конкретних потреб і вподобань. Accord.NET показав простоту застосування для задач кластеризації. ML.NET виявився більш гнучким і адаптованим підходом для кластеризації. Обидва мають інтеграцію з .NET, зокрема, бібліотекою OxyPlot, та дуже зручне середовище розробки.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи отримано наступні результати:

- досліджено проблему кластерного аналізу як задачі інтелектуального аналізу даних;
- розглянуто основні підходи до кластеризації, проведено порівняльний аналіз алгоритмів кластеризації;
- досліджено існуючі бібліотеки машинного навчання та візуалізації даних на C#, ML.NET і Accord.NET мають широкий функціонал для машинного навчання в цілому, але обмежений набір засобів для кластеризації об’єктів даних;
- розроблено WPF-застосунки на C# для кластерного аналізу з використанням бібліотек ML.NET, Accord.NET та графічної бібліотеки OxyPlot; виконано оцінювання алгоритмів кластеризації;
- розроблено WPF-застосунок на C# з використанням графічної бібліотеки OxyPlot для ієрархічної кластеризації та кластеризації на основі щільності; зроблено оцінку алгоритмів кластеризації.

Таким чином, проведене дослідження та розробка застосунку поглибило розуміння можливостей та обмежень методів кластерного аналізу, що є основою для їх практичного застосування та подальших розробок.

ПЕРЕЛІК ПОСИЛАНЬ

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

2. Інтелектуальний аналіз даних та машинне навчання. Частина 1. Базові методи та засоби аналізу даних / Я. В. Іванчук, В. І. Месюра, А. А. Яровий, О. Д. Манжілевський. Вінниця : ВНТУ, 2021. 69 с.

3. Кононова К. Ю. Машинне навчання: методи та моделі : підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. Харків : ХНУ імені В. Н. Каразіна, 2020. 301 с.

4. Ланде Д. В. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник / Д. В. Ланде, І. Ю. Субач, Ю. Є. Бояринова. К. : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. 300 с.

5. Литвин В. В. Аналіз даних і знань: навчальний посібник / В. В. Литвин, В. В. Пасічник, Ю. В. Нікольський. Львів : Магнолія 2006, 2019. 276 с.

6. Машинне навчання: Навчальний посібник призначений для студентів, що навчаються за першим (бакалаврським) рівнем вищої освіти за спеціальностями галузі знань 12 «Інформаційні технології» / за ред. д. т. н., професора, В. В. Пасічника. Львів : «Новий Світ-2000», 2024. 330 с.

7. Mahasivabha K. Machine Learning with ML.NET. Orange Education Pvt Ltd, AVA, 2024. ISBN: 978-81-97256-3.

8. Xu D., Tian Y. A Comprehensive Survey of Clustering Algorithms. *Ann Data*. 2015. 2 (2). P. 165–193. URL: https://doi.org/10.1007/s40745-015-0040-1 (дата звернення: 26.05.2024).

9. Рясна М. В., Литвиненко Ю. О. Кластерний аналіз в маркетингових дослідженнях. URL: https://core.ac.uk/reader/300238166 (дата звернення: 26.03.2024).

10. Шевченко С. М., Жданова Ю. Д., Шевцова Т. І. Застосування кластерного аналізу для просування бізнесу у соціальних мережах. *Вісник*

ХНТУ Інформаційні технології. № 4(87). 2023. URL: https://doi.org/10.35546/kntu2078-4481.2023.4.32 (дата звернення: 12.04.2024).

11. Wierzchoń S., Kłopotek M. Modern algorithms of cluster analysis. 2018. URL: https://doi.org/10.1007/978-3-319-69308-8 (дата звернення: 26.03.2024).

12. Melnykov V., Michael S., Melnykov I. Recent developments in model-based clustering with applications. Partitional clustering algorithms. Cham, 2014. P. 1–39.

13. Lv Y., Jiang H., Pan P. NI-DBSCAN: DBSCAN under Non-IID. *Journal of physics: conference series*. 2020. T. 1533. P. 022110. URL: https://doi.org/10.1088/1742-6596/1533/2/022110 (дата звернення: 06.04.24).

14. Prasad S. Different Types of Clustering Methods and Applications. Analytix Labs. 2020. URL: https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/ (дата звернення: 06.11.24).

15. DotNET machine-learning. Microsoft Learn. ML.NET Documentation. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work (дата звернення: 06.11.24).

16. Yinyang K-Means: A Drop-In Replacement of the Classic K-Means with Consistent Speedup. *Proceedings of the 32nd International Conference on Machine Learning* (Lille, France, 2015). JMLR: W&CP: 2015. Vol. 37.

17. Shindler M. Approximation Algorithms for the Metric k-Median Problem. URL: https://web.archive.org/web/20110927100642/ (дата звернення: 21.10.24).

18. Фішер Р. А. Репозторій машинного навчання. URL: https://archive.ics.uci.edu/dataset/53/iris (дата звернення: 21.10.24).

19. Чабан Г. М. Використання спеціалізованого програмного забезпечення для аналізу економіко-математичного моделювання. *Ефективна економіка*. 2020. №9. URL: http://www.economy.nayka.com.ua/?op=1&z=8197 (дата звернення: 05.09.2024).

20. Data Mining – Cluster Analysis. URL: https://www.geeksforgeeks.org/data-mining-cluster-analysis/ (дата звернення: 03.11.24).

21. Exploring Machine Learning Algorithms with Accord.NET. URL: https://clouddevs.com/dot-net/machine-learning-algorithms-with-accord/#1-Implementing-Classification-Algorithms (дата звернення: 01.11.24).

22. Computationaland Applied Mathematics. P. 53–65. URL: https://archive.org/details/sim_journal-of-computational-and-applied-mathematics_1987-11_20/page/53 (дата звернення: 06.11.24).

23. Fowlkes E. B., Mallows C. L. A method for comparing two hierarchical clusterings. *Journal of the american statistical association*. 1989. Т. 78, № 383. P. 553–569.

24. The Accord.NET Image Processing and Machine Learning Framework http://accord-framework.net/index.html (дата звернення: 06.11.24).

25. OxyPlot’s documentation. URL: https://oxyplot.readthedocs.io/en/latest/ (дата звернення: 12.11.24).

ДОДАТОК А

Лістинг програмного застосунку з ML.NET

```
// k_means ML.NET, протестовано на даних Iris.csv
// кластеризує також зображення .jpg

using System.Windows;
using System;
using System.Linq;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using Microsoft.ML;
using Microsoft.ML.Data;
using OxyPlot;
using OxyPlot.Series;
using OxyPlot.Wpf;

using System.Collections.Generic;
using static WpfApp_ML_NET_K_Means.MainWindow;

namespace WpfApp_ML_NET_K_Means
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>

    public class KMeansClustering
    {
        private MLContext mlContext;
        private ITransformer model;
        private IDataView dataView;

        public KMeansClustering()
        {
            mlContext = new MLContext();
        }

        public void LoadData(string filePath)
        {
            if (filePath.EndsWith(".csv"))
            {
                var data =
mlContext.Data.LoadFromTextFile<DataPoint>(filePath,
                    separatorChar: ',', hasHeader: true);
                dataView = data;
            }
            else if (filePath.EndsWith(".jpg"))
            {
                var dataPoints = LoadImageData(filePath);
                dataView =
mlContext.Data.LoadFromEnumerable(dataPoints);
            }
        }

      public IEnumerable<DataPoint> LoadImageData(string imagePath)
        //public void LoadImageData(string imagePath)
        {
```

```
                var data = new List<DataPoint>();
                using (var bitmap = new Bitmap(imagePath))
                {
                    for (int x = 0; x < bitmap.Width; x++)
                    {
                        for (int y = 0; y < bitmap.Height; y++)
                        {
                            var pixel = bitmap.GetPixel(x, y);
// Яскравість як характеристика
       // Вибираємо компоненти, наприклад, яскравість або кольори (R, G)
                float feature1 = pixel.R / 255f; // нормалізація до [0, 1]
                float feature2 = pixel.G / 255f;
      data.Add(new DataPoint{Feature1 = feature1, Feature2 = feature2 });
                          };
                        }
                    }
                }
        return data;
      }

              public void TrainModel(int clustersCount)
              {
                  var pipeline = mlContext.Transforms.Concatenate("Features",
nameof(DataPoint.Feature1), nameof(DataPoint.Feature2))
      .Append(mlContext.Clustering.Trainers.KMeans("Features",
numberOfClusters: clustersCount));
                  model = pipeline.Fit(dataView);
              }

              public List<ClusterPrediction> Predict()
              {
                  var predictions = model.Transform(dataView);
                  var clusteredData =
mlContext.Data.CreateEnumerable<ClusterPrediction>(predictions,
reuseRowObject: false).ToList();
                  return clusteredData;
              }

      public PlotModel CreatePlot(List<ClusterPrediction> predictions)
          {  // візуалізація кластерів
             var model = new PlotModel { Title = "K-Means Clustering" };
             var clusterColors = new Dictionary<uint, OxyColor>
          {
              { 1, OxyColors.Red },
              { 2, OxyColors.Blue },
              { 3, OxyColors.Green },
              { 4, OxyColors.Yellow },
              { 5, OxyColors.Gray },
              // можна більше кольорів для різних кластерів
          };
                  foreach (var prediction in predictions)
                  {
                      var color =
clusterColors.ContainsKey(prediction.PredictedClusterId) ?
clusterColors[prediction.PredictedClusterId] : OxyColors.Black;
                      var scatterSeries = new ScatterSeries { MarkerType =
MarkerType.Circle, MarkerFill = color };

                      scatterSeries.Points.Add(new
ScatterPoint(prediction.Feature1, prediction.Feature2));
                      model.Series.Add(scatterSeries);
                  }
                  return model;
              }
```

```
public PlotModel CreatePlotInitialData(List<ClusterPrediction> predictions)
             { // початкова ініціалізація
      var model = new PlotModel{Title="Initial Data K-Means Clustering" };

                  var clusterColors = new Dictionary<uint, OxyColor>
          {
              { 1, OxyColors.Red },
              { 2, OxyColors.Blue },
              { 3, OxyColors.Green },
              { 4, OxyColors.Yellow },
              { 5, OxyColors.Gray },
              // можна більше кольорів для різних кластерів
          };
                  foreach (var prediction in predictions)
                  {
                      var color =
clusterColors.ContainsKey(prediction.PredictedClusterId) ?
clusterColors[prediction.PredictedClusterId] : OxyColors.Black;
                      var scatterSeries = new ScatterSeries { MarkerType =
MarkerType.Circle, MarkerFill = OxyColors.Black };

                      scatterSeries.Points.Add(new
ScatterPoint(prediction.Feature1, prediction.Feature2));
                      model.Series.Add(scatterSeries);
                  }
                  return model;
              }

      public double CalculateSilhouetteIndex(List<ClusterPrediction>
predictions)
             {//  індекс силуету
var groupedByCluster = predictions.GroupBy(p => p.PredictedClusterId)
                      .ToDictionary(g => g.Key, g => g.ToList());
                  double totalSilhouetteScore = 0;
                  int totalPoints = predictions.Count;
                  foreach (var point in predictions)
                  {
                      var ownCluster =
groupedByCluster[point.PredictedClusterId];
                      // Середня відстань до точок у своєму кластері (a)
                      double a = ownCluster
                          .Where(other => !ReferenceEquals(other, point))
                         .Select(other => EuclideanDistance(point, other))
                          .DefaultIfEmpty(0)
                          .Average();

            // Мінімальна середня відстань до точок в інших кластерах (b)
double b = groupedByCluster.Where(kvp => kvp.Key != point.PredictedClusterId)
            .Select(kvp => kvp.Value
            .Select(other => EuclideanDistance(point, other)).Average())
            .DefaultIfEmpty(double.MaxValue).Min();
                      // Індекс силуету для точки
                      double s = (b - a) / Math.Max(a, b);
                      totalSilhouetteScore += s;
                  }
                  // Середній індекс силуету для всіх точок
                  return totalSilhouetteScore / totalPoints;
              }

      // обчислення евклідової відстані між двома точками
           private double EuclideanDistance(ClusterPrediction p1,
ClusterPrediction p2)
              {
```

```
      return Math.Sqrt(Math.Pow(p1.Feature1 - p2.Feature1, 2) +
Math.Pow(p1.Feature2 - p2.Feature2, 2));
              }

              public double
CalculateDaviesBouldinIndex(List<ClusterPrediction> predictions, int
numberOfClusters)
      {
      var clusterGroups = predictions.GroupBy(p => p.PredictedClusterId)
                      .ToDictionary(g => g.Key, g => g.ToList());
                  // Розрахунок центрів кластерів
      var centroids = clusterGroups.ToDictionary(g => g.Key,g => new
                      {
                          X = g.Value.Average(p => p.Feature1),
                          Y = g.Value.Average(p => p.Feature2)
                      });
               // Розрахунок міжкластерних відстаней (середній радіус)
      var intraClusterDistances = clusterGroups.ToDictionary(
      g => g.Key,g => g.Value.Average(p =>
      Math.Sqrt(Math.Pow(p.Feature1 - centroids[g.Key].X, 2) +
Math.Pow(p.Feature2 - centroids[g.Key].Y, 2))));
                  // Розрахунок DBI
                  double dbIndex = 0;
                  for (uint i = 1; i <= numberOfClusters; i++)
                  {
                      double maxRatio = 0;
                      for (uint j = 1; j <= numberOfClusters; j++)
                      {
                          if (i != j)
                          {
      double interClusterDistance = Math.Sqrt(
      Math.Pow(centroids[i].X - centroids[j].X, 2) +
      Math.Pow(centroids[i].Y - centroids[j].Y, 2));

      double ratio = (intraClusterDistances[i] + intraClusterDistances[j]) /
interClusterDistance;
      maxRatio = Math.Max(maxRatio, ratio);
                          }
                      }
                      dbIndex += maxRatio;
                  }
                  return dbIndex / numberOfClusters;
              }
          }

          public class ClusterPrediction
          {
              [ColumnName("PredictedLabel")]
              public uint PredictedClusterId;
              [ColumnName("Score")]
              public float[] Distances;
              public float Feature1;
              public float Feature2;
          }

          public class DataPoint
          {
              [LoadColumn(0)]
              public float Feature1;
              [LoadColumn(1)]
              public float Feature2;
          }

          public partial class MainWindow : Window
```

```
        {
            private KMeansClustering kMeans;//об’єкт моделі
        public MainWindow()
            {
                InitializeComponent();
                kMeans = new KMeansClustering();
            }
            // Завантаження початкових даних
      private void LoadDataButton_Click(object sender, RoutedEventArgs e)
            {
                // чи введено правильне число кластерів
      if (int.TryParse(ClustersCountTextBox.Text, out int clustersCount))
                {
                    kMeans.LoadData("Iris1.csv");// Завантажуємо дані

kMeans.TrainModel(clustersCount);// Тренуємо модель з clustersCount
      List<ClusterPrediction> predictions = kMeans.Predict();  // Прогнозуємо
кластери
                    // Оновлюємо графік
                    PlotView.Model =
kMeans.CreatePlotInitialData(predictions);
                }
                else
                {
MessageBox.Show("Введіть коректне число для кількості кластерів.");
                }
            }
            // Обробник події для кнопки
 private void RunKMeansButton_Click(object sender, RoutedEventArgs e)
            {
                // Перевіряємо, чи введено правильне число кластерів
 if (int.TryParse(ClustersCountTextBox.Text, out int clustersCount))
                {
                    kMeans.LoadData("Iris1.csv");
              //kMeans.LoadData("leaf.jpg");  // Завантажуємо дані
kMeans.TrainModel(clustersCount);  // Тренуємо модель з clustersCount
  List<ClusterPrediction> predictions = kMeans.Predict();// Прогнозуємо
кластери
                    // ClusteringMetrics:  Розрахунок індексу силуету
double silhouetteIndex = kMeans.CalculateSilhouetteIndex(predictions);
                    // Оновлюємо графік
                    PlotView.Model = kMeans.CreatePlot(predictions);
                    // Відображення індексу силуету
       MessageBox.Show($"Індекс силуету: {silhouetteIndex:F3}", "Оцінка
кластеризації");
                    // Розрахунок індексу Девіса-Боулдіна
       double dbIndex = kMeans.CalculateDaviesBouldinIndex(predictions,
clustersCount);
                    // Відображення індексу
MessageBox.Show($"Davies-Bouldin Index: {dbIndex:F4}", "Clustering Quality");
                }
                else
                {
      MessageBox.Show("Введіть коректне число для кількості кластерів.");
                }
            }
        }
      }
```

ДОДАТОК Б

Лістинг застосунку ієрархічної кластеризації та DBSCAN

Б.1 Код функції реалізації алгоритму DBSCAN

```
// Реалізація DBSCAN
private List<List<DataPoint>> DBSCAN(List<DataPoint> points, double epsilon,
int minPts)
{
    var clusters = new List<List<DataPoint>>();
    var visited = new HashSet<DataPoint>();
    var noise = new List<DataPoint>();
    foreach (var point in points)
    {
        if (visited.Contains(point))
            continue;
        visited.Add(point);
        var neighbors = GetNeighbors(point, points, epsilon);
        if (neighbors.Count < minPts)
        {
            noise.Add(point); // Маркуємо як шум
        }
        else
        {
            var cluster = new List<DataPoint>();
 clusters.Add(ExpandCluster(point, neighbors, cluster, visited, points,
                epsilon, minPts));
        }
    }
    return clusters;
}
```

Б.2 Код функції пошуку точок-сусідів та обчислення відстані для DBSCAN

```
private List<DataPoint> GetNeighbors(DataPoint point, List<DataPoint> points,
double epsilon)
{
    return points.Where(p => GetDistance(point, p) <= epsilon).ToList();
}
private double GetDistance(DataPoint p1, DataPoint p2)
{
    return Math.Sqrt(Math.Pow(p1.X - p2.X, 2) + Math.Pow(p1.Y - p2.Y, 2));
}
```

Б.3 Код функції пошуку кластера DBSCAN

```
private List<DataPoint> ExpandCluster(DataPoint point, List<DataPoint> neighbors,
List<DataPoint> cluster,
    HashSet<DataPoint> visited, List<DataPoint> points, double epsilon, int
minPts)
{
    cluster.Add(point);
    for (int i = 0; i < neighbors.Count; i++)
    {
        var neighbor = neighbors[i];

        if (!visited.Contains(neighbor))
        {
            visited.Add(neighbor);
            var neighborNeighbors = GetNeighbors(neighbor, points, epsilon);
            if (neighborNeighbors.Count >= minPts)
            {
                neighbors.AddRange(neighborNeighbors);
            }
        }
        if (!cluster.Contains(neighbor))
        {
            cluster.Add(neighbor);
        }
    }
    return cluster;
}
```

Б.4 Код класу Cluster для дендрограми агломеративної кластеризації

```
public class Cluster
{
// The set of points in the cluster (a data point in the dendrogram).
    public HashSet<int> PointsX { get; set; }

    // Starting coordinates (x, y) for drawing the cluster on the dendrogram.
    public DataPoint StartCoordinates { get; set; }

    //public double DistanceSum { get; set; } = 0;

    // Constructor to initialize the cluster with a starting point.
    public Cluster(double startX, double startY)
    {
        PointsX = new HashSet<int>();
// Initial position for the cluster
    StartCoordinates = new DataPoint(startX, startY);
    }
    // Method to add a point to the cluster.
    public void AddPoint(int pointX)
    {
        PointsX.Add(pointX);
    }
    // Method to remove a point from the cluster.
    public void RemovePoint(int pointX)
    {
```

```
            PointsX.Remove(pointX);
        }
    }
```

Б.5 Вміст файлу DBSCANPage.xaml

```
<Page x:Class="WpfOxyPlot_DBSCAN_Hierarch_Dend.DBSCANPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:WpfOxyPlot_DBSCAN_Hierarch_Dend"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      xmlns:oxy="http://oxyplot.org/wpf"
      Title="DBSCANPage">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <!-- Параметри для DBSCAN -->
        <StackPanel Orientation="Horizontal" Grid.Row="0" Margin="0">
            <Label Content="Epsilon:" Grid.Row="0" Grid.Column="0" Margin="10" />
            <TextBox x:Name="EpsilonTextBox" Grid.Row="1" Grid.Column="1"
                     Width="50" Text="0.5" Margin="10"/>
            <Label Content="Min Points:" Grid.Row="1" Grid.Column="0"
Margin="10"/>
            <TextBox x:Name="MinPtsTextBox" Grid.Row="1" Grid.Column="1"
Width="50"
                     Text="5" Margin="10"/>
            <Button Content="Run DBSCAN" Grid.Row="2" Grid.Column="1"
                    Grid.ColumnSpan="2" Click="RunDBSCAN_Click" Margin="10,0"/>
        </StackPanel>

        <!-- Відображення графіка OxyPlot -->
        <oxy:PlotView x:Name="plotView" Grid.Row="1" Model="{Binding plotModel}"
/>
    </Grid>
</Page>
```

Б.6 Вміст файлу DBSCANPage.xaml

```
<Page x:Class="WpfOxyPlot_DBSCAN_Hierarch_Dend.HierarchClusteringPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

      xmlns:oxy="http://oxyplot.org/wpf"
      Title="HierarchClusteringPage">
```

```
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <TabControl>
            <!-- Вкладка для відображення кластерів -->
            <TabItem Header="Clusters">
                <StackPanel>
                    <!-- Поле для введення кількості кластерів і кнопка для
запуску -->
      <StackPanel Orientation="Horizontal" HorizontalAlignment="Center"
Margin="0">
      <TextBlock Text="Number of Clusters:" VerticalAlignment="Center"
Margin="0"/>
      <TextBox x:Name="ClustersTextBox" Width="50" Text="2" Margin="0"/>
      <Button Content="Run Clustering" Click="RunClustering_Click" Margin="0"/>
                    </StackPanel>
                    <!-- Графік кластерів -->
                    <oxy:PlotView x:Name="plotView" Model="{Binding plotModel}"
                                  HorizontalAlignment="Center"
                                  VerticalAlignment="Stretch" Height="350"
Width="750" />
                </StackPanel>
            </TabItem>
            <!-- Вкладка для відображення дендрограми -->
            <TabItem Header="Dendrogram">
                <!-- Графік дендрограми -->
 <oxy:PlotView x:Name="dendrogramView" Model="{Binding dendrogramModel}"
  HorizontalAlignment="Stretch"  VerticalAlignment="Bottom" Height="350"
Width="750" />
            </TabItem>
        </TabControl>
    </Grid>
</Page>
```

ДОДАТОК В

Лістинг програми кластеризації з Accord.NET

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

using Accord.MachineLearning;
using Accord.Math;
using Accord.Statistics.Distributions.Multivariate;
using OxyPlot;
using OxyPlot.Series;
using AccordVector = Accord.Math.Vector;
using AccordMath = Accord.Math;

namespace GMMWpfApp
{

    public partial class MainWindow : Window
    {
        int k; // кількість кластерів
        double[][] observations; // набір даних,спостереження
        KMeans kmeans;
        // Для графічного відображення
        PlotModel plotModel;

        public MainWindow()
        {
            InitializeComponent();
            plotModel = new PlotModel { Title = "Gaussian Mixture Clustering"
};
            plotView.Model = plotModel;
        }

        // Генерація випадкових даних
        private void btnGenerateRandom_Click(object sender, RoutedEventArgs
e)
        {
            k = 3; // можна змінити на динамічний вхід
            //int value
            //k = (int)numericUpDown.Value;
            k = integerUpDown.Value ?? 0;
            // Генерація даних з випадковими розподілами Гауса
            double[][][] data = new double[k][][];
            for (int i = 0; i < k; i++)
            {
                double[] mean = AccordVector.Random(2, -6.0, +6.0);
```

```
                double[,] covariance =
Accord.Statistics.Tools.RandomCovariance(2, -5, 5);
        var gaussian = new MultivariateNormalDistribution(mean, covariance);

                int samples = AccordMath.Random.Generator.Random.Next(150,
250);
                data[i] = gaussian.Generate(samples);
            }
            // Об'єднання даних
            observations = AccordMath.Matrix.Stack(data);

            // Оновлення графіка
            CreateScatterplot(observations, k);
            kmeans = null;
        }

        // Створення графіка
        public void CreateScatterplot(double[][] data, int clusters)
        {
            plotModel.Series.Clear();
            // Створення серії точок
            var series = new ScatterSeries { MarkerType = MarkerType.Circle
};

            foreach (var point in data)
            {
                series.Points.Add(new ScatterPoint(point[0], point[1]));
            }

            plotModel.Series.Add(series);
            plotView.InvalidatePlot(true);
        }

        // Обчислення моделі GMM
        private void btnCompute_Click(object sender, RoutedEventArgs e)
        {
            if (observations == null) return;

            var gmm = new GaussianMixtureModel(k);

            // Ініціалізація за допомогою K-Means, якщо доступно
            if (kmeans != null)
                gmm.Initialize(kmeans);

            // Обчислення моделі
            GaussianClusterCollection clustering = gmm.Learn(observations);

            // Класифікація всіх точок
            int[] classifications = clustering.Decide(observations);

            // Оновлення графіка з класифікацією
            UpdateGraph(classifications);
        }
        // Ініціалізація K-Means
        private void btnInitialize_Click(object sender, RoutedEventArgs e)
        {
            if (observations == null) return;

            kmeans = new KMeans(k);
            KMeansClusterCollection clustering = kmeans.Learn(observations);
            int[] classifications = clustering.Decide(observations);
             // Оновлення графіка
            UpdateGraph(classifications);
        }
```

```
        // Оновлення графіка після класифікації
        private void UpdateGraph(int[] classifications)
        {
            plotModel.Series.Clear();

            // Створення серій для кожного кластеру
            for (int i = 0; i < k; i++)
            {
                var series = new ScatterSeries { MarkerType =
MarkerType.Circle };
      var points = observations.Where((obs, idx) => classifications[idx] ==
i);

                foreach (var point in points)
                {
                    series.Points.Add(new ScatterPoint(point[0], point[1]));
                }

                plotModel.Series.Add(series);
            }

            plotView.InvalidatePlot(true);
        }
    }
}
```