КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Випускна кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки
на тему:

Тестування, дослідження та порівняльний аналіз основних алгоритмів кластеризації наборів числових даних

Виконав студент 4 курсу
Чорний Василь Вікторович

_____
(підпис)

Науковий керівник:
кандидат фізико-математичних наук, доцент
Трохимчук Ростислав Миколайович

_____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики
« » травня 2021 р., протокол №
Завідувач кафедри
проф. Крак Ю. В. _____
(підпис)

Київ – 2021

ЗМІСТ

ВСТУП

Всі речі в нашому світі можна віднести до якихось груп, які характеризуються за певними параметрами. А кластерний аналіз якраз виконує такі завдання як:

- Дослідження угрупування об’єктів;
- Вироблення гіпотез на базі дослідження даних;
- Підтвердження гіпотез на базі досліджень даних;
- Визначення присутності груп всередині даних;

У промисловості наприклад дуже важливо розрізняти різні види продукції. Наприклад у виробництві вина важливо відрізняти біле вино від червоного, чи у виробництві меблів треба відрізняти дуб від берези і так далі. Також сьогодні кластеризація використовується в біології(зокрема в біоінформатиці), соціології та в інших галузях. Також ієрархічна кластеризація часто використовується в біології, наприклад у філогенетичному дереву.

Але на сьогодні є не мала низка проблем для кластеризації. Проблеми кластерного аналізу:

- Обґрунтування якості результатів.

Проблема полягає в тому, що один і той же об'єкт може бути класифікований в різні групи незалежно від його внутрішніх властивостей, а в зв'язку з різними експертними даними або різним побудовою системи. Для уникнення цього необхідно розробляти і вводити актуальні критерії якості.

- Проблема багатовимірності, або “прокляття розмірності” для деяких методів кластеризації.

Аналіз великого числа різнотипних даних породжує методологічну проблему вибору метрик. Так само збільшення числа об'єктів навіть однотипних даних може спричинити за собою непомітність відстаней.

Розбиття вибірки на групи схожих об'єктів для спрощення розуміння кластерної структури, що спрощує обробку даних і прийняття рішення, застосовуючи до кожного кластеру свій метод аналізу.

Сьогодні це дуже перспективна тема і має дуже великий потенціал для розвитку в багатьох сферах. Але через низку проблем яку я описав зверху бувають недоліки. Також ці недоліки притаманні кожному методу і універсального точного методу для рішення всіх задач кластеризації не буває, тому треба ще правильно підбирати методи та параметри цих методів для більш оптимального розв’язку. Також інколи підводить людський фактор, коли людина не правильно зібрала данні чи не правильно обрала метод і задала йому параметри.

Об’єктом роботи у нас виступають датасети (штучні + природні) та алгоритми кластеризації.

У якості інструменту створення програмного сервісу було обрано PyCharm - інтегроване середовище розробки мовою Python. Також були використані наступні бібліотеки мови програмування Python:

-Numpy

-Pandas

-Matplotlib.pyplot

-Seaborn

Можливі сфери застосування.

Розроблена модель є більше експериментальною моделью для порівняння методів кластерного аналізу, але може застосовуватись у промисловості.

1.Постановка задачі

Задача полягає в створенні програми яка зможе займатися кластеризацією різних датасетів такими методами як **KMeans DBSCAN**

MeanShiftGaussianMixtureSpectralClusteringAgglomerativeClustering.І яка буде видавати нам оцінки якості кластеризації відносно істинного розподілу на кластори, і візуально показувати кластирезаціюданних. Завдяки цім оцінкам ми зможемо аналізувати ефективність роботи кожного методу для різних датасетів.Також для більш подрібного аналізу необхідно розглянути такі оцінки відповідності до початкових класів якГомогенність completenessscoreAdjustedrandindexAdjustedmutualinformation.

2. Кластеризація даних

Кластеризація (або кластерний аналіз) - це задача розбиття множини об'єктів на групи, які називаються кластерами. Усередині кожної групи повинні виявитися «схожі» об'єкти, а об'єкти різних групи повинні бути якомога більш відмінні. Головна відмінність кластеризації від класифікації полягає в тому, що перелік груп чітко не заданий і визначається в процесі роботи алгоритму.

Застосування кластерного аналізу в загальному вигляді зводиться до наступних етапів:

- Відбір вибірки об'єктів для кластеризації.
- Визначення безлічі змінних, за якими будуть оцінюватися об'єкти у вибірці. При необхідності - нормалізація значень змінних.
- Обчислення значень міри схожості між об'єктами.
- Застосування методу кластерного аналізу для створення груп схожих об'єктів (кластерів).
- Представлення результатів аналізу.

Після отримання та аналізу результатів можливе корегування обраної метрики і методу кластеризації до отримання оптимального результату.

2.1 Сучасні методи кластеризації

2.1.1. Ієрархічні методи кластеризації.

Основою ієрархічної кластеризації є послідовне злиття кластерів для утворення великих структур або поділ великих кластерів на менші. Така кластеризація часто має дуже хорошу наочність, але її використовують лише при невеликих обсягах наборів даних.

Ієрархічні методи кластеризації різні за правилами формування кластерів. Правила використовуються для визначення схожі чи об'єкти між собою і чи можуть вони бути віднесені в один кластер. Методи, що з'єднують примірники у групи називаються агломеративними.

У графічної аналогією ієрархічні методи базуються на побудові дендрограмм (від грецького dendron - "дерево"), які описують відстані між окремими точками і кластерами один по відношенню до одного.

Далі кожен вид буде розглянуто окремо через своїх популярних представників.

Агломеративний AGNES (AgglomerativeNesting).

Ця група методів, як було сказано раніше, характеризується послідовним об'єднанням вихідних елементів і відповідним зменшенням числа кластерів.

На початку алгоритму всі екземпляри об'єктів представлені як окремі кластери. Перший крок відбирає найбільш схожі об'єкти, формує з них кластер. З кожним наступним кроком відбувається формування нових кластерів, а також об'єднання кластерів в групи. Так триває поки не буде сформований кінцевий кластер, який містить в собі інші кластери.

CURE

Цей алгоритм призначений для кластеризації дуже великих наборів числових даних, тільки ефективно працювати здатний з даними низької розмірності.

Алгоритм заснований на наборі визначаючих точок. Спочатку алгоритм формує дерево кластерів, де кожен об'єкт є кластером одиничного розміру. Потім кластери упорядковано відповідно до віддалі одна від одної, використовуючи «Манхеттенську» або «евклидову» відстань. Після формування цих даних в «купі» оперативної пам'яті, відбувається злиття найближчих кластерів і перерахунок відстаней. Так відбувається до отримання потрібної кількості кластерів.

Під час роботи алгоритму кластери поділяються на дві групи: перша група - кластери, у яких обчислюється мінімальна відстань з новоствореним кластером, другі - всі інші кластери. При зміні кластерів і новому перерахунку відбувається чергування кластерів з якими відбувається порівняння.

Таким чином, алгоритм є високорівневим алгоритмом кластеризації, виділяє кластери різних форм і має лінійну залежність до розміру даних, що зберігаються і тимчасової складності.

ROCK

Алгоритм кластеризації ROCK. RobustClusteringAlgorithm - агломеративно-ієрархічний алгоритм, для кластеризації великої кількості даних з номінальними (булевими) атрибутами.

Алгоритм по роботі схожий на K-means, але він враховує наявність зв'язку у загальних сусідів. Робота алгоритму побудована на складанні матриць схожості. Це матриці сусідства кластерів, кількості спільних сусідів, міри близькості об'єктів. Сусідство кластерів визначається відношенням кон'юнкції і

суми по модулю 2 двох кластерів. Формула визначення близькості об'єктів залежить кількості об'єктів в кожному кластері і загальних посилань між двома кластерами. Після формування матриць починається ітеративна кластеризація, що знаходить два найбільш близьких кластера і об'єднуючи їх. При цьому формуються нові значення в матриці загальних посилань і матриці близькості, а дані про двох оригінальних кластерах видаляються. Критеріями для завершення ітерацій є або досягнення заданого їх числа, або формування вказаної кількості кластерів,

DIANA (DivisiveAnalysis)

Методи цієї групи логічно протилежні агломеративним, на початку їх роботи об'єкти знаходяться в одному кластері. В результаті роботи алгоритмів на кожному їхньому кроці такий кластер буде ділитися на менші, створюючи послідовність розщеплених груп.

BIRCH

Алгоритм запропонований ТьянЗанг і його колегами. Переваги алгоритму лежать в його високій швидкості на тлі великої масштабованості. Це досягається за рахунок двоетапного процесу кластеризації.

На першому етапі алгоритм формує попередній набір кластерів. Другий етап застосовує до виділених кластерів інші методи кластеризації, що робить цей дуже невибагливим до ресурсів.

ТьянЗанг наводить таку аналогію щодо алгоритму BIRCH: «Якщо кожен елемент даних уявити собі як намистину, що лежить на поверхні столу, то кластери намистин можна" замінити "тенісними кульками і перейти до більш детального вивчення кластерів тенісних кульок. Число намистин може виявитися досить велике, проте діаметр тенісних кульок можна підібрати таким чином, щоб на другому етапі можна було, застосувавши традиційні алгоритми кластеризації, визначити дійсну складну форму кластерів. »

MST

Цей алгоритм заснований на побудові мінімального кістякового дерева (MST, minimumspanningtree). Граф будується на підставі подання об'єктів як вершин, а відстаней між цими об'єктами як дуг. На отриманий граф застосовується метод побудови мінімального кістякового дерева, причому може застосовуватися будь-який відомий метод, але з урахуванням великої кількості дуг в графі (при N документах в колекції дуг). Після цього видаляються ребра з найбільшими довжинами, утворюючи ліс невеликих дерев, вузли яких породжують кластери.

Це досить гнучкий алгоритм, кластеризуються довільні набори даних, виділяючи кластери різних форм, і вибираючи найбільш оптимальне рішення. Швидкість роботи буде багато в чому залежати від обраного методу отримання мінімального остовного дерева.

2.1.2. Неієрархічні (ітеративні) методи кластеризації

Із зростанням обсягів даних ієрархічні методи втрачають всі свої привабливі властивості, тоді ефективними можуть виявитися неієрархічні методи, які являють собою ітеративні алгоритми поділу вихідної сукупності. Тобто весь набір даних ділиться на певну кількість кластерів або поки не спрацює правило зупинки. У цій групі методів, використовуються два підходу, обидва відмінних від роботи дивізійних ієрархічних принципів. Методи першого підходу визначають межі кластерів по найбільш щільним ділянкам багатовимірного простору вихідних даних. Тобто відбувається процес пошуку «згущення точок». Методи другого підходу мінімізують заходи відмінності об'єктів.

К-середніх (k-means)

Це найбільш поширений неієрархічні метод, і можливо найпоширеніший метод кластеризації в цілому в зв'язку з простотою його реалізації

Перед початком роботи методу необхідно мати припущення про ймовірне кількості кластерів. K-means добре підходить для підтвердження гіпотез про кількість кластерів і може використовуватися дані від попередніх обчислень, або просто від інтуїтивно обраного числа. Алгоритм спочатку побудує задану кількість кластерів, а після цього буде розподіляти об'єкти так, щоб середнє серед всіх змінних в кластері будуть максимально відмінними.

Алгоритм CLARA (ClusteringLARgeApplications)

В результаті CLARA пропонує найкращу кластеризацию. Цей алгоритм досить ефективний для великих обсягів даних, але сильно залежить від заздалегідь обраного початкового набору даних, що призводить до гарним показникам якості кластеризації на тестовій вибірці, але може виявитися помилковим в застосуванні до всього набору даних.

LargeItem

Алгоритм LargeItem був запропонований Вангом в 1990 році. Це оптимізаційний алгоритм для кластеризаціїтрансакційних даних, заснованих на критеріальною функції, оптимізації глобального критерію. Цей глобальний критерій використовує параметр підтримки (в термінології тут багато спільного з алгоритмами для виявлення асоціативних правил). В обчислення глобального критерію робить алгоритм кластеризації у багато разів швидше, ніж при використанні локального критерію при парному порівнянні об'єктів, тому "глобалізація" оціночної функції - один із шляхів отримання масштабованих алгоритмів.

Алгоритм включає в себе дві фази: фазу розподілу і фазу поліпшення. Кращим рішенням є те, що краще мінімізує цільову функцію. Так як знайти абсолютно точне рішення не представляється можливим з достатньою часткою ймовірності, то мета цього алгоритму полягає в знаходженні наближеного рішення, що є достатнім для практичного застосування. На відміну від K-means, LargeItem дозволяє значенням k варіюватися, тобто алгоритм не зобов'язаний

знати її заздалегідь. Щоб уникнути сканування всіх транзакцій при кластеризації, деякі касетні функції, такі як | Largei |, | Uki = 1 = Smalli | і | Uki = 1 = Largei |, зберігаються після кожного розподілу або переміщення транзакції.

Алгоритм використовує деякі стандартні методи індексації, такі як хеш-таблиці і B-дерева в обслуговуванні і відновленні, доступні для кожного кластера.

CLOPE

У 2002 році група китайських вчених представила алгоритм CLOPE (ClusteringwithsLOPE). Особливістю цього алгоритму є висока продуктивність роботи в порівнянні з іншими ієрархічними методами. Робота цього алгоритму заснована на максимізації глобальної функції вартості, що підвищує близькість транзакцій в кластерах за допомогою збільшення параметра кластерної гістограми.

За допомогою параметра, названого авторами CLOPE коефіцієнтом відштовхування (repulsion), регулюється рівень подібності транзакцій всередині кластера, і, як наслідок, фінальне кількість кластерів. Цей коефіцієнт підбирається користувачем. Чим більше r, тим нижче рівень подібності і тим більше кластерів буде згенеровано.

Як видно, алгоритм CLOPE є масштабованим, оскільки здатний працювати в обмеженому обсязі оперативної пам'яті комп'ютера. Під час роботи в RAM зберігається тільки поточна транзакція і невелика кількість інформації по кожному кластеру.

2.1.3 Порівняльний аналіз ієрархічних і не ієрархічних методів кластеризації

Ключове питання постає про те, якої групи методів віддати перевагу при обробці вихідних даних. Аналітику доводиться враховувати різні чинники і

особливості між ієрархічними і ітераційними методами. Розглядаючи ці групи слід виділити основні чинники.

Застосування ітераційних методів дає високу стійкість до шумів і викидів, але в свою чергу це обумовлено заздалегідь певними даними, такими як кількість кластерів, число ітерацій або правила зупинки. У такому випадку процес аналізу даних буде багато в чому залежати від попередньої роботи аналітика та побудови системи в цілому.

Найчастіше складається ситуація, в якій визначити початкові дані неможливо, або це буде окремою трудомісткою завданням, що перевершує операції кластеризації. Наприклад, коли обсяг початкової вибірки занадто великий, то можна проводити експерименти з кількістю кластерів або їх розмірами.

За рахунок такого "варіювання" результатів досягається досить велика гнучкість кластеризації.

У свою чергу, ієрархічні методи спираються не на число кластерів, а будують повну структуру, яка містить вкладення. Цей факт накладає обмеження за обсягом набору даних. Так само часто виявляється важким вибрати міру близькості. У такому світлі методи виявляються негнучкими, оскільки для оцінки навіть наближених результатів, доводиться обробляти весь обсяг первинної вибірки раз по раз.

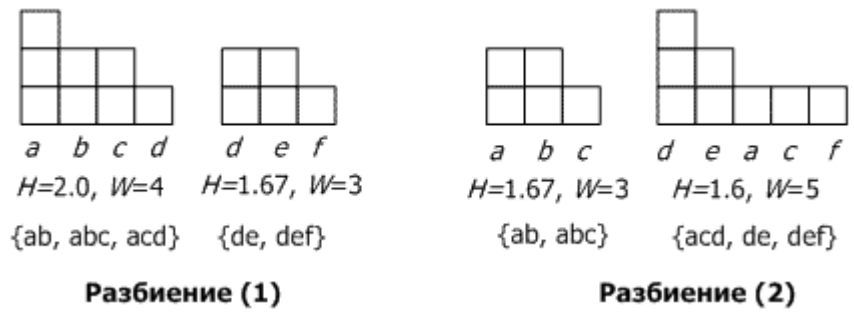
Перевага ж цієї групи методів в порівнянні з не-ієрархічній методами - їх наочність і можливість отримати детальне уявлення про структуру даних.

3. Пропоновані методи і математичне забезпечення

Алгоритм CLOPE є алгоритмом кластеризаціїтранзакційних даних. Транзакція в даному контексті - це довільний набір об'єктів. Тобто в конкретному випадку транзакція схожа на кортеж в якому можуть знаходити нетипізовані дані, що не мають ніякий зв'язку між собою, і, не вибудовуються в просторі. Завдання кластеризації подібних даних полягає в отриманні такого розбиття, щоб схожі транзакції знаходилися в одному кластері, а відмінні - в одному з інших. Для цього в алгоритмі застосовується принцип максимізації глобальної функції вартості, що зближує транзакції в кластерах, збільшуючи параметр кластерної гістограми.

Наприклад, для набору транзакцій {(a, b), (a, b, c), (a, c, d), (d, e), (d, e, f)} порівняємо два розбиття на кластери:

1. {{Ab, abc, acd}, {de, def}};
2. {{Ab, abc}, {acd, de, def}}.



3.1 Розбиття кластеризацією CLOPE

У першому випадку кластер представляє чотири незалежних унікальних об'єкта (a, b, c, d), які входять в кластер кількістю (3, 2, 2, 1) відповідно. Таким чином, кластер має ширину W = 4, тобто включає в себе 4 типи об'єктів. Площею S будемо вважати входження в кластер всіх об'єктів, тобто S = 3 + 2 + 2 + 1 = 8. Знаючи два цих параметра можна розрахувати висоту кластера H = S / W = 8/4 = 2. Якщо те ж саме зробити з іншими з кластерами, то ми побачимо, що при однаковій кількості елементів в кластері параметри H, W, S можуть довільно

змінюватися. Якщо подивитися обидва розбиття, то можна зрозуміти, що розбиття 1 вигідніше, так як забезпечує більшу кількість менша кількість унікальних об'єктів і більше число накладень. Саме в цьому полягає принцип максимізації вартості.

Нехай D - безліч транзакцій {t1, ..., tn}, де t - набір об'єктів {i1, ..., in}. Знайти таку силу-силенну кластерів {C1, ..., Ck} для безлічі {t1, ..., tn}, таке що

Гістограма кластера - це графічне зображення його розрахункових характеристик.

Алгоритм повинен враховувати висоту H при розбитті, тому що транзакція, максимально збільшує площу по відношенню до інших існуючих кластерів, не збільшуючи ширини, найбільш підходить до вже містяться в кластері транзакціях. Але оцінка висоти не є ефективним критерієм в разі розбиття з однаковою висотою, наприклад H = 1. У такому випадку замість оцінки висоти необхідно обчислювати градієнт G (C) = H (C) / W (C) = S (C) / W (C).

Узагальнюючи, виведемо формулу глобального критерію

$$\text{Profit}(C) = \frac{\sum_{i=1}^{k} G(C_i) \times |C_i|}{\sum_{i=1}^{k} C_i} = \frac{\sum_{i=1}^{k} \frac{S(C_i)}{W(C_i)^r} \times |C_i|}{\sum_{i=1}^{k} C_i}$$

Де | Ci | - кількість об'єктів в i-м кластері

k - кількість кластерів.

r - коефіцієнт відштовхування (repulsion), позитивно дійсне число перевершує 1.

За допомогою коефіцієнта відштовхування визначається ступінь подібності транзакцій в кластері, причому залежність обернено пропорційна,

тобто з підвищенням значення коефіцієнта, знижується показник схожості транзакцій при порівнянні, і тим самим збільшується кількість кластерів.

Таким чином, постановка задачі кластеризації для алгоритму CLOPE виглядає так:

Робота алгоритму проходить методом ітеративного перебору записів бази даних, а глобальність критерію оптимізації, заснованому на розрахунку параметрів кластерів, дозволяє обробляти дані значно швидше, ніж в порівнянні транзакцій між собою.

Алгоритм працює в два етапи. На першому етапі відбувається перший прохід по базі з метою ініціалізації даних і побудови первинного розбиття. Другий етап здійснює ітеративні обходи бази, оптимізуючи функцію вартості до припинення змін до кластерної структурі.

Цілком очевидно, що CLOPE - масштабований алгоритм, здатний працювати на обмежених ресурсах. Під час його роботи в пам'яті знаходиться тільки поточна транзакція і кластерні характеристики (CF - clusterfeatures). При входженні 10 000 об'єктів в 1 000 кластер потрібно всього близько 40 Мб пам'яті для зберігання даної інформації.

Обчислювальна складність алгоритму зростає лінійно зі збільшенням таблиці і зростанням числа кластерів, вона дорівнює

O (N * K * A), де

N - загальне число транзакцій;

K - максимальне число кластерів;

A - середня довжина транзакції.

Таким чином алгоритм відноситься до ефективних на великим обсягах даних.

Даний алгоритм можна успішно застосовувати не тільки для транзакційних даних, але і для будь-яких категорійних. Основною вимогою є нормалізація даних. Це може бути бінарна матриця або відтворення унікальних об'єктів {u1, u2, ..., uq} і безліччю цілих чисел {1, 2, ..., q}. Таким чином до виду транзакції можна звести будь-який набір даних, тому CLOPE знаходить своє місце в обробці категоріальних даних.

3.2 K-means

Тепер розглянемо докладно найпопулярніший і простий в реалізації алгоритм k-means. Цей алгоритм був відкрити в різних дисциплінах Ллойдом (1957), Фордж (1965), Фрідманом і Рубіном (1967), а так же МакКуїн (1967).

K-means застосуємо до об'єктів в d-вимірному векторному просторі, які представлені як набір D = {xi | i = 1, ..., N}, де xi∈Rd - i-й об'єкт. Суть алгоритму полягає в об'єднанні D так, щоб кожна xi потрапила тільки в один k розділ. В результаті утворюється кластерний складовою вектор m довжиною N, де mi - номер кластера xi. Параметр k - дійсне значення роботи алгоритму і є кінцевим числом кластерів. Цей параметр є експертним введенням, заснованим на спостереженнях, аналізі попередніх результатів або просто інтуїтивному перевагу аналітика. Неважливо яким буде значення k для розуміння поділу набору даних, але оптимальне значення досягається шляхом ряду ітеративних експериментів.

В роботі алгоритму кожен кластер представлений точкою в Rd і представляється як безліч C = {cj | j = 1, ..., k}. Ці стани називають центроїдами кластера. Для кластеризації застосовуються заходи близькості, зокрема евклідова відстань, вже розглянутий раніше. Робота алгоритму полягає в мінімізації функції вартості, яка представлена як квадрат відстані між кожною точкою xi і найближчим представником кластера cj.

$$\mathbf{Cost} = \sum_{i=1}^{N} (\arg\min_j \| x_i - c_j \|_2^2)$$

Наведене рівняння часто називають цільовою функцією k-means.

Алгоритм будується в 2 кроки, які итеративно чергуються між собою:

1. Переписування номера кластера для всіх об'єктів в D
2. Ефективно використовувати час кластера за що містяться в ньому об'єктів

Спочатку не започатковано представники кластеру, але підставі вибірки k з Rd випадковим чином, встановлюючи їх як рішення підмножини, або порушуючи середнє значення даних k-раз. Потім виконується ітерації до збіжності за 2 кроки:

1. Кожному об'єкту присвоюється найближчий представник, довільно порушуючи зв'язку даних, що призводить до їх поділу;
2. Центроид кластера переміщається на місце середнього арифметичного всіх об'єктів в кластері.

Алгоритм сходиться при неможливості здійснювати переміщення. Таким чином цільова функція зменшується з кожним кроком, а зближення гарантовано за кінцеве число ітерацій.

На жаль, значення цільової функції не інформативно з точки зору підбору кількості кластерів, тому що функція вартості приймає мінімальне значення в тому випадку, коли число кластерів дорівнює числу об'єктів.

Реалізація алгоритму:

1. Виберемо випадковим чином k точок з D як представників класу C відповідно до формули
2. Обчислимо центр для кожного кластера

3. Перерозподілили об'єкти по кластерам

Кожна ітерація в такому випадку буде мати складність O (N * k), а число ітерацій, необхідних невідомо, але росте з числом об'єктів N. Рішення проблеми швидкості в даному випадку може бути запропоновано шляхом розподілу обчислень на нащадки, в цьому випадку необхідно розбити безліч даних на P частин рівному за значенням кількості потоків, а кожен потік буде працювати зі своїм набором даних.

Актуальною проблемою розбиття є проблема «порожніх кластерів». Цей недолік посилюється зі збільшенням числа k, коли в момент роботи утворюється центроїд кластера cj, такий, що всі крапки xi в D виявляються ближче до центроїду іншого кластера. В такому випадку точки перерозподіляються, а вихідного кластеру будуть призначені нульові значення, утворюючи з нього порожня множина. В такому випадку необхідно передбачити переініціалізація центроїда порожнього кластера.

Незважаючи на недоліки, алгоритм k-means є найбільш поширеним інструментом кластеризації на практиці. Це простий і масштабований метод, який володіє достатньою ефективністю і здатний працювати як доповнення до більш складних методів, що я розгляну далі.

3.3 BIRCH

Алгоритм BIRCH являє з себе двоетапний процес кластеризації, який добре кластеризує великі набори числових даних. Раніше було розглянуто алгоритм k-means, який передбачається використовувати як другий етап в даному методі. Гарний зв'язок цих методів полягає в тому, що вони обидва працюють з числовими даними і будують кластери сферичних форм. Для роботи алгоритму BIRCH необхідно тільки вказівку порогових значень, а кількість кластерів, необхідне для k-means буде визначено під час першого етапу.

Розглянемо роботу алгоритму:

Побудова початкового CF-дерева (CF Tree, кластерне дерево). Кластерний дерево - це виважено збалансоване двопараметричного дерево, де B- коефіцієнт розгалуження, а T - порогова величина. Кожен вузол, який не є листом дерева, має не більше B входжень вузлів, представлених у формі [CFi, Childi], де i = 1, ..., B; а Childi - покажчик на дочірній i-й вузол. Лист дерева має покажчик на два сусідніх вузла, а радіус кластера, що складається з елементів цього вузла не повинен перевершувати порогове значення.

Кластер представляється як трійка (N, LSS, SS), де N - число елементів вхідних даних кластера, LS - сума елементів вхідних даних, SS - сума квадратів елементів вхідних даних.

Стиснення даних (необов'язковий етап) здійснюється перестроюванням кластерного дерева зі збільшенням граничної величини T.

Глобальна кластеризація відбувається застосуванням обраного алгоритму кластеризації на Лістьєва компонентах кластера. В обраному випадку тут вступає в роботу алгоритм K-means, з отриманим на попередніх етапах числом кластерів.

Поліпшення кластерів (необов'язковий етап) використовує центри тяжкості, отримані в результаті глобальної кластеризації, перерозподіляючи дані між «близькими» кластерами. Даний етап гарантує що однакові дані потрапляють в один кластер.

3.4 Метод обчислення цінових моделей

У разі автоматизації вибору ваг пропонується використовувати метод підбору значень на підставі використання методу перехресного контролю. Ідея даного методу полягає в тому, щоб розділяти дані на навчальний і тестовий набори. Навчальний набір повинен бути оснащений правильними результатами

в ту подію, для якого будується прогнозування. Після цього дані передаються в алгоритм, який намагається їх спрогнозувати. Найчастіше процедура проводиться кілька разів з різних розбиттям. Тестовий набір становить 5% від вибірки, а решта 95% - навчальний набір. В якості оцінки використовується сума квадратів різниць. Підсумовування квадратів різниць - хороший метод, так як зі збільшенням різниці вплив на суму збільшується нелінійно. Таким чином, породжуючи великі відхилення, метод працює ефективніше інших, утворюють помірно близькі результати. Підбір ваг трудомістка і тривала операція, але вона проводиться один раз для кожної нової навчальної вибірки.

Теоретично, можна підбирати безліч різних поєднань вагових коефіцієнтів вручну, але подібний підхід неймовірно трудомісткий. Для оптимізації процесу необхідно задати область визначення змінних, діапазон і цільову функцію. Областю визначення - діапазон всіх ваг по кожному виміру. Перевага оптимізації масштабів по осях, це наочність важливості тих чи інших змінних, рівень їх впливу. Іноді одне лише знання таких даних дозволяє переглянути всі раніше отримані результати і гіпотези.

4. Підготовка до реалізації проекту

4.1 Аналіз інструментальних засобів

Розглянемо популярні мови і програмні середовища з точки зору пристосованості під різні класи задач.

BASIC

BASIC (англ. Beginner's All-purposeSymbolicInstructionCode - універсальний код символічних інструкцій для початківців; англ. BASIC - основний, базовий). Мова був розроблений в 1963 році викладачами Дартмутського коледжу Джоном Кемені і Томасом Куртц.

BASIC був спроектований для навчання студентів без математичного програмування і призначений для розробників заінтересованих в простоті реалізації, закриваючи очі на структурність, швидкості роботи і багато інших Спекта. Згодом з'явилося безліч діалектів і реалізацій даного мови.

Вісім вимоги, представлятися при розробці мови:

- простота у використанні для початківців;
- спільність призначення (відсутність спеціалізації);
- можливість розширення функціональності засобами, доступними програмістам;
- інтерактивність;
- чіткі і зрозумілі повідомлення про помилки;
- висока швидкість роботи на невеликих програмах;
- відсутність необхідності розуміння роботи апаратного забезпечення для написання програм;
- ефективне посередництво між користувачем і операційною системою.

У 70-ті роки Microsoft популяризували цю мову, як Бизов для програмування на своїх системах, а в 1991 з'явився VisualBasic - сучасна реалізація, що стала найбільш популярним на платформі Windows. Сьогодні BASIC - це ціле сімейство мов з розвиненим деревом реалізацій, але в своїй основі це все той же мова, призначення для вирішення невеликих прикладних завдань.

Pascal

Pascal - один з найбільш популярних мов, так само як і BASIC спрямованого на навчання. Мова виділяється строгою типізацією і засобами процедурного програмування, ставши першопрохідником в цьому напрямку.

Паскаль (англ. Pascal) - мова програмування загального призначення. Один з найбільш відомих мов програмування, використовується для навчання програмуванню в старших класах і на перших курсах ВНЗ, є базою для ряду інших мов.

Детальний опис всіх недоліків привів початку 1980-х БрайанКерниган в статті «Чому Паскаль не є моїм улюбленим мовою програмування».

Найбільш відомою реалізацією Паскаля, що забезпечила широке поширення і розвиток мови, є TurboPascal фірми Borland, що виросла потім в об'єктний Паскаль для DOS (починаючи з версії 5.5) і Windows і далі в Delphi, в якій були впроваджені значні розширення мови.

C і C ++

В основі мови С - вимоги системного програміста: повний і ефективний доступ до всіх ресурсів комп'ютера, засоби програмування високого рівня, переносимість програм між різними платформами і операційними системами. С ++, зберігаючи сумісність з С, вносить можливості об'єктно-орієнтованого програмування, висловлюючи ідею класу (об'єкта) як визначається користувачем типу. Завдяки перерахованим якостям, С / С ++ зайняв позицію універсальної мови для будь-яких завдань. Але його застосування може стати неефективним там, де потрібно отримати готовий до вживання результат в найкоротші терміни, або там, де невигідним стає сам процедурний підхід.

Python

В основі мови Python лежить принцип з'єднання декількох парадигм програмування, таких як структурний, функціональний, імперативне, об'єктно і аспектно-орієнтування. У мові застосовується динамічна типізація, інтроспекція, багатопотокові обчислення, автоуправління пам'яттю.

Python - один з найбільш динамічних мов совремнном часу. Нові версії виходять часто, тому відсутні стандарт ANSI, ISO або інші офіційні стандарти.

Perl

Perl (PracticalExtractionandReportLanguage, англ. - Практичний Мова для Витяги Даних і Складання Звітів) розроблений лінгвістом ЛарріУоллом. Мова багатий можливостями для роботи з текстом і регулярними виразами, вбудованими в синтаксис.

Perlунаследовн від Сі і є процедурним, реалізуючи змінні, присвоювання, керуючі структури і функції.

Загальна структура Perl в загальних рисах веде свій початок від мови Сі. Perl - процедурний за своєю природою, має змінні, вирази присвоювання, блоки коду, відокремлювані фігурними дужками, керуючі структури і функції. Регулярні вирази добре працюють для «парсинга тексту»

Perl запозичує масиви з Lisp, регулярні вирази з AWK і sed, з AWK також запозичені хеші («асоціативні масиви»). Регулярні вирази полегшують виконання багатьох завдань по парсингу, обробці тексту та маніпуляцій з даними. У мові реалізоана потужна функція автоматичної типізації даних, але неможливі операції призводять до фатальних помилок.

Java

Об'єктно-орієнтована мова програмування, від компанією SunMicrosystems 23 травня 1995. Додатки транслюються в байт-код і працюють усередині віртуальної машини. Це знімає залежність коду від середовища виконання, але накладає обмеження на наявність віртуальної машини, а будь-які неповноважним операції завершують виконання віртуальної машини.

За даними сайту shootout.alioth.debian.org, Java в окремих випадках в декілька разів повільніше C / C ++, що в середньому в півтора-два рази більше, а споживання пам'яті Java-машиною було в 10-30 разів більше.

1. ВИСНОВКИ

У висновку зауважимо, що з професійної точки зору не так важливо якою мовою і в якому середовищі працює програміст, скільки як він виконує свою роботу. Змінюється апаратура та операційні системи. Виникають нові завдання з найрізноманітніших предметних областей. Відходять у минуле і з'являються нові мови. Але залишаються люди - ті, хто пише і ті, для кого пишуть нові програми і чиї вимоги до якості залишаються тими ж незалежно від цих змін.

5. Реалізація

Для початку імпортуємо бібліотеки, які можуть на знадобитися. Де numpy і pandas використовуються для роботи с даними, і для графіків matplotlib.pyplot, seaborn.

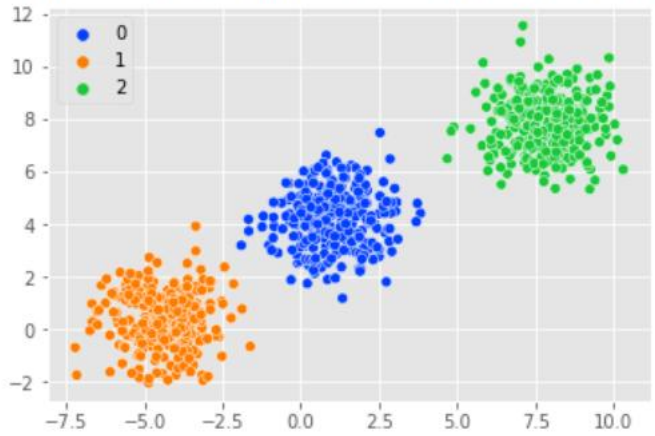
```
# для роботи з даними (data-wrangle, "крутити-вертіти даними")
import numpy as np
import pandas as pd

# для графіків
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')
```

Потім ми генеруємо наші купки з випадковими точками задаючи в параметрах кількість кучок і точок

```
# зробити купки
X, y = make_blobs(900, random_state=3)
```

Ось подібний приклад

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f115bcd2850>
```



Потім ми імпортуємо метрики, якими оцінимо якість кластеризації де X-таблиця координат а y-класи. А параметри в імпорті означають completenessscore - чи немає "розрізів"(чи не розрізає кластер наші класи), а homogeneityscore - перевірити "чистоту" передбачених кластерів, чи немає різних класів у рамках одного кластеру і adjustedrandindex - чи співпадають передбачені кластери та справжні класи, з точністю до перенумерації.

А також adjustedmutualinformation - наскільки подібні (в інформаційному сенсі) кластеризація та оригінальні класи.

Малюємо сітку графіків

Задаємо табличку з розрахованими значеннями метрик.

```
from sklearn.metrics import completeness_score, homogeneity_score, adjusted_rand_score, adjusted_mutual_info_score


fig, axes = plt.subplots(2, len(algos), figsize=(len(algos)*3, 8))

scores = pd.DataFrame(index=['Homogenity', 'Completeness', 'Adjusted Rand', 'Adjusted MutInfo'])
```

Потім ми для кожного алгоритму підганяємо модель. Потім ми отримуємо результат кластеризації і зберігаємо його у змінну. Після цього ми зберігаємо обчисленні метрики.

```
y_pred = model.fit_predict(X)


scores[type(algo).__name__] = [
    homogeneity_score(y, y_pred),
    completeness_score(y, y_pred),
    adjusted_rand_score(y, y_pred),
    adjusted_mutual_info_score(y, y_pred)
]

if X_disp is None:
    X_disp = X
```

Виводимо результат кластеризації транспонуємо та розпаковуємо матрицю на два вектори. Де hue -- колір, відтінок. кольори відповідають результатам кластеризації.

І виводимо правильні відповіді

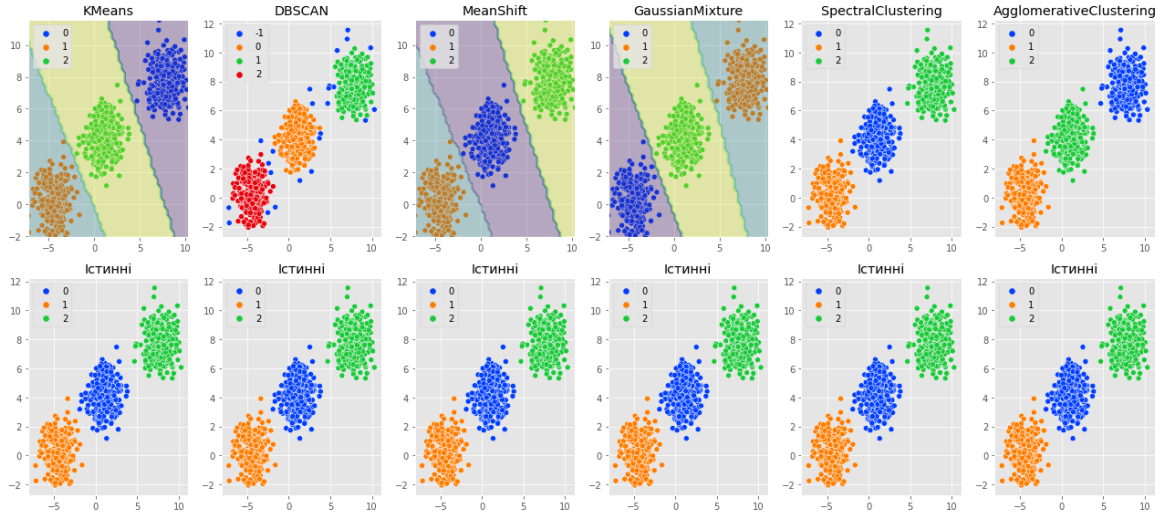
```
sns.scatterplot(*X_disp.T, hue=y_pred, palette='bright', ax=axes[0][i], **kwargs)
axes[0][i].set_title(f'{type(algo).__name__}')

if X.shape[1] == 2:

    if hasattr(model, 'predict'):
        n = 50
        a, b = np.meshgrid(np.linspace(X_disp[:, 0].min(), X_disp[:, 0].max(), n),
                           np.linspace(X_disp[:, 1].min(), X_disp[:, 1].max(), n))
        c = model.predict(np.c_[a.flatten(), b.flatten()]).reshape(n, n)

        axes[0][i].contourf(a, b, c, alpha=0.3)


sns.scatterplot(*X_disp.T, hue=y, palette='bright', ax=axes[1][i], **kwargs)
```

Імпортуємо бібліотеки з алгоритмами. І задаємо параметри алгоритмам завдяки яким можна регулювати точність кластеризації. Де наприклад у K-means це кількість класторів, а у DBSCAN це eps тобто дистанція від точки через яку точки можуть включатися у кластор. І після кластеризації оцінюємо якість кластеризації.

```
from sklearn.cluster import KMeans, DBSCAN, MeanShift, SpectralClustering, AgglomerativeClustering
from sklearn.mixture import GaussianMixture

algos = [KMeans(3), DBSCAN(eps=0.7), MeanShift(3), GaussianMixture(3), SpectralClustering(3), AgglomerativeClustering(3)]


run_algos(algos, X, y)
```

І ось така картина у нас виходить якщо я задам такі параметри як на картинці.

	KMeans	DBSCAN	MeanShift	GaussianMixture	SpectralClustering	AgglomerativeClustering
Homogenity	1.0	0.973699	1.0	1.0	1.0	1.0
Completeness	1.0	0.897218	1.0	1.0	1.0	1.0
Adjusted Rand	1.0	0.959446	1.0	1.0	1.0	1.0
Adjusted MutInfo	1.0	0.933700	1.0	1.0	1.0	1.0



Аналіз:Тут ми бачимо, що всі алгоритми відпрацювали добре. Кожен із алгоритмів чітко виявив згенеровані кластери, що видно і на графіках, і при аналізі метрик. Єдиний алгоритм, якій відпрацював дещо гірше за інші — це DBSCAN. Він не зміг правильно зрозуміти поодинокі точки, які лежать подалі від центрів кластерів

Гомогенність:Всі методи окрім DBSCANвідпрацювали ідеально. Він не зміг правильно зрозуміти поодинокі точки, які лежать подалі від центрів кластерів.

completenessscore:Всі методи окрім DBSCAN відпрацювали ідеально.Він не зміг правильно зрозуміти поодинокі точки, які лежать подалі від центрів кластерів

Adjustedrandindex:Всі методи окрім DBSCAN відпрацювали ідеально.Він не зміг правильно зрозуміти поодинокі точки, які лежать подалі від центрів кластерів

Adjustedmutualinformation:Всі методи окрім DBSCAN відпрацювали ідеально.Він не зміг правильно зрозуміти поодинокі точки, які лежать подалі від центрів кластерів

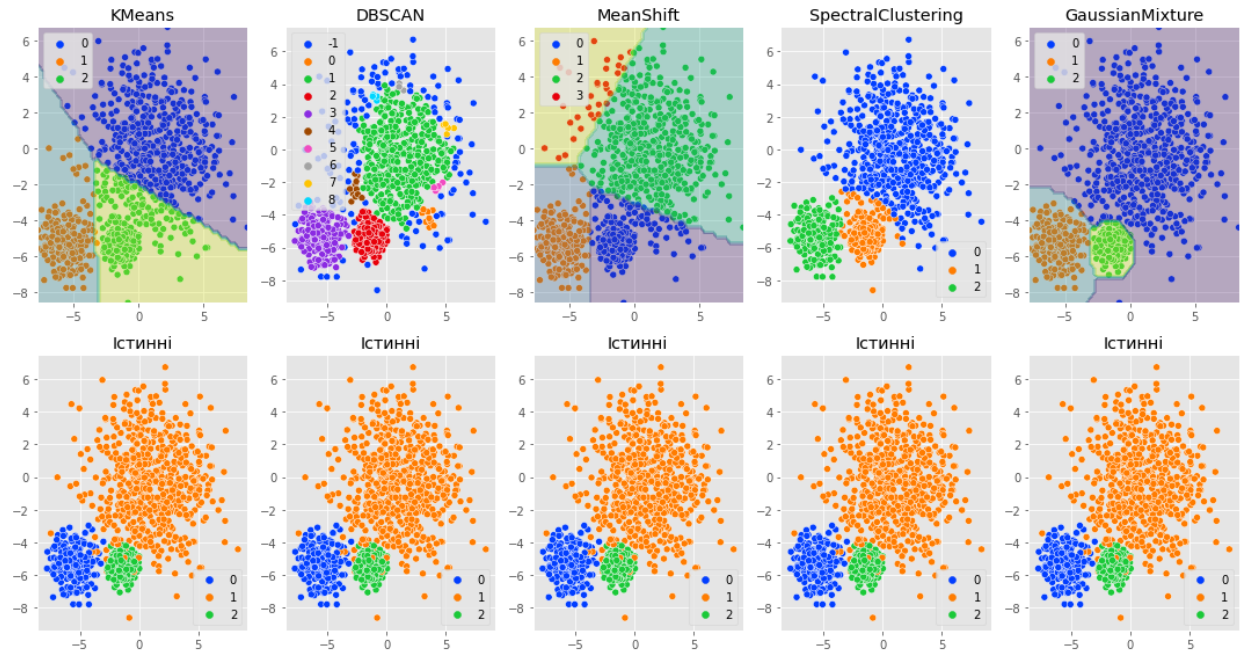
Потім я беру складніший синтетичний двовимірний датасет.

І таким же чином задаю кучки з випадковими точками. І провожу кластерний аналіз визиваючи методи і задаючи їм параметри.

```
algos = [KMeans(3), DBSCAN(eps=0.5), MeanShift(2), SpectralClustering(3), GaussianMixture(3)]

run_algos(algos, X, y)
```

```
KMeans  почав роботу.
DBSCAN  почав роботу.
MeanShift  почав роботу.
SpectralClustering  почав роботу.
GaussianMixture  почав роботу.
```

	KMeans	DBSCAN	MeanShift	SpectralClustering	GaussianMixture
Homogenity	0.704671	0.869609	0.768772	0.832627	0.893474
Completeness	0.667865	0.587316	0.670596	0.811450	0.884656
Adjusted Rand	0.625086	0.648358	0.693519	0.844524	0.928130
Adjusted MutInfo	0.685213	0.698876	0.715600	0.821579	0.888840



Аналіз:Гетерогенна структура кластерів привела до серйозних труднощів для деяких алгоритмів. К-середніх описав кластери досить приблизно, не зумівши чітко окреслити межу між найбільшим та найменшим кластерами.Метод середніх зсувів відпрацював схожим чином, єдина

відмінність — він знайшов ще один маленький кластер там, де його бути не повинно.

DBSCAN правильно зрозумів "ядра" усіх кластерів, але внаслідок того, що кластери мають різну густину точок, найбільший (і, відповідно, найбільш розріджений кластер) було описано лише у його центральній частині. По краям результати гірші, бо алгоритм розпізнав невеликі стохастичні угруповування як окремі кластери.

Гаусове змішування відпрацювало найкраще. Це, вочевидь, пояснюється тим, що базові припущення цього алгоритму повністю співпадають із методом генерації датасету (тобто кожен кластер описується багатовимірним гаусовим розподілом).

Гомогенність: Краще за все відпрацював **GaussianMixture**тому що він майже повністю повторює контури і не розрізає початкові класи 0.89. Гірше за все все відпрацював KMeansвін достатньо сильно розрізав один з початкових класів 0.70.

completenessscore: Краще за все відпрацював **GaussianMixture**він майже в точності зберіг всі початкові класи і майже не має в собі інших класів 0.88. Гірше за все все відпрацювавDBSCANвін знайшов дуже багато класторів всередині одного класа 0.58.

Adjustedrandindex:Краще за все відпрацював **GaussianMixture**0.92. Гірше за все все відпрацював KMeans 0.62.

Adjustedmutualinformation:Краще за все відпрацював 0.88. Гірше за все все відпрацював KMeans 0.68.

Реальний датасет

Потім я взяв реальний датасет, в якому збережені дані про характеристики вин такі як: густина, градус, алкоголь, цукор, тип. І по цим параметрам задача програми визначити кластери білого чи чергового вина. Також для більш зручних перетворень ми тут користуємося формулою узагальнення даних

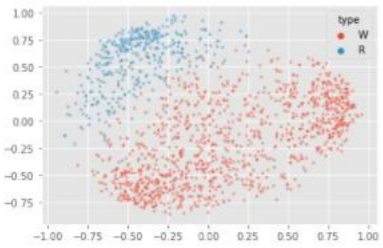
$$X' = \frac{X - \bar{X}}{se(X)}$$

де $\bar{X}$—середнє значення відповідної колонки, а $se(X)$ — її стандартне відхилення. Фактично, після цього перетворення усі дані будуть центровані у нулі, а діапазон усіх значень буде приблизно $(-3, \dots, 3)$.

Всього у нас 11 параметрів. Оскільки зрозуміло візуалізувати 12-вимірну гіперповерхню неможливо, то для полегшення цього завдання зменшуємо розмірність. Для цього використовуємо метод головних компонент — як бачимо на зображенні нижче, уже на двох вимірах візуально кластери добре розділяються.

```
from sklearn.decomposition import KernelPCA

X2 = KernelPCA(2, kernel='cosine').fit_transform(X, y)

sns.scatterplot(*X2.T, hue=y, s=10, alpha=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f114674e590>
```

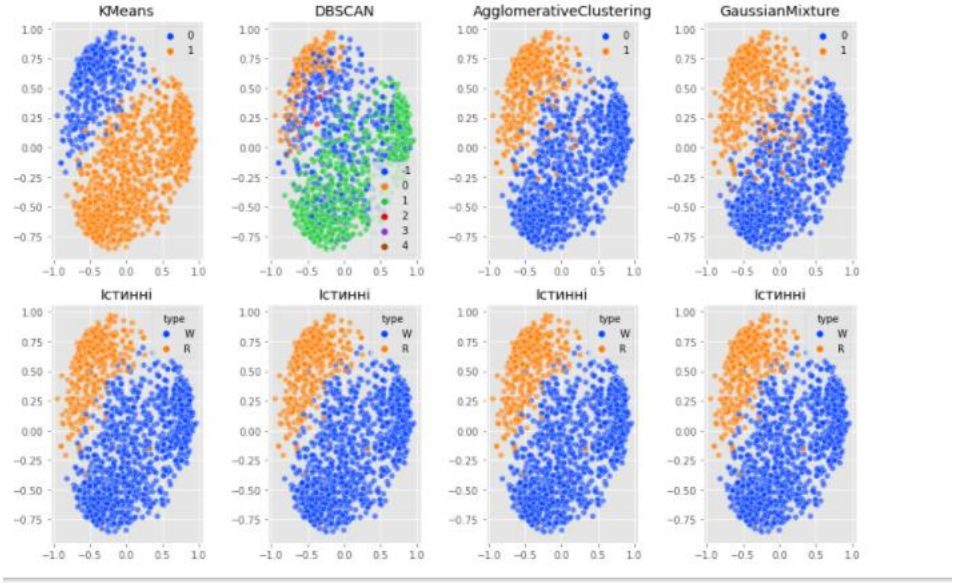


Тепер проведемо власне кластеризацію. Алгоритм середнього зсуву був виключених із порівняння, бо працював занадто довго. Для візуалізації кластеризації ми використовуємо ПЦА із косинусовимКарнелом щоб перевести данні з багатовимірного простору в двовимірний.

```
algos = [KMeans(2), DBSCAN(eps=4, metric='manhattan'), AgglomerativeClustering(2), GaussianMixture(2)]

run_algos(algos, X, y, X_disp=X2, alpha=0.6)
```

```
KMeans  почав роботу.
DBSCAN  почав роботу.
AgglomerativeClustering  почав роботу.
GaussianMixture  почав роботу.
```

	KMeans	DBSCAN	AgglomerativeClustering	GaussianMixture
Homogenity	0.830820	0.620417	0.809873	0.681883
Completeness	0.801671	0.344094	0.788567	0.615636
Adjusted Rand	0.895990	0.413883	0.890391	0.728630
Adjusted MutInfo	0.815878	0.441354	0.798961	0.646871



Аналіз**:** Тут я кластеризую по усіх вимірах, але відображаю графічно лише двовимірну проекцію для простоти сприйняття людиною.

В DBSCAN ми змінили дефолтну метрику на манхеттенську (її ще називають "мінковська, першого ступеню". Другого ступеню — то звичайна евклідова)

Причиною такого рішення були численні експерименти, які на практиці показали, що так краще.

Евклідова відстань це коли ми йдемо напряму до нашої точки як по гіпотенузі, а манхетонська це коли ми йдемо по квадрату(типу як вулиці в Манхеттені).

Краще всього тут спрацював KMeans, а гірше за все DBSCAN хоча ми навіть для нього застосували Манхетанську відстань але все одно результат погіршився через багатовимірність та його “Прокляття розмірності”.

Гомогенність:Найкраще відпрацював KMeansвін майже не “розрізав” початкові класи. І майже ідеально повторює контури 0.83. Найгірше відпрацював DBSCAN він створив нові кластери які не зміг віднести ні до одного з класів і таким чином “розрізав” початкові класи 0.62.

completenessscore:Найкраще відпрацював KMeansвін майже в точності зберіг всі початкові класи і майже не має в собі інших класів 0.80. Найгірше відпрацював DBSCAN він створив нові кластери які входять всередину початкових класів 0.34.

Adjustedrandindex:Найкраще відпрацювавKMeans 0.89. Найгірше відпрацювавDBSCAN 0.41.

Adjustedmutualinformation:Найкраще відпрацював KMeans 0.81. Найгірше відпрацювавDBSCAN 0.44.

Параметри алгоритмів

Більшість параметрів алгоритмів були залишені в значенні за замовчуванням (відповідно до реалізації у рамках бібліотеки scikit-learn). Деякі

параметри, однак, було модифіковано для кожного датасету для кожного алгоритму (наприклад, кількість кластерів). Ще деякі параметри були змінені лише для конкретних алгоритмів та датасетів (наприклад, метрика для DBSCAN для останнього датасету). Такі зміни будуть обговорені окремо.

Список незмінених параметрів для усіх використаних алгоритмів.

1. К-середніх.
    - **кількість кластерів:** *залежно від датасету.*
    - **ініціалізація центрів:** «kmeans++». Це просунутий алгоритм, що працює ефективніше за випадкову ініціалізацію. Альтернативою йому є абсолютно випадкові початкові точки.
    - **кількість запусків:** 10. Кількість разів, що алгоритм запускався до конвергенції. Запуск відбувається більше одного разу для стабільності фінальної конвергенції.
    - **максимум ітерацій:** 300. Максимальна кількість ітерацій алгоритму, перш ніж він буде зупинений. Ми жодного разу не зіткнулись із проблемою несходження алгоритму.
    - **толерантність:** 1e-4. Межа у відносній зміні у нормі Фробеніуса різниці між кластерами для зупинки алгоритму; іншими словами, наскільки мало мають зміщуватись кластери при ітерації для оголошення сходження та зупинки.
    - **передрозрахунок відстаней:** «автоматично», що у нашому випадку означає «так» для усіх датасетів. Чи створювати у пам’яті матриці, що триматиме відстані між усіма точками. Це чисто технічна деталь реалізації для підвищення швидкості роботи, і ніяк не відображається на якості кластеризації.
    - **метод:** «автоматично», що у нашому випадку означає «elkan». Некласична реалізація алгоритму к-середніх, що використовує нерівність трикутника. Дає швидше сходження для даних із чіткими кластерами.
2. Ієрархічна кластеризація.
    - **кількість кластерів:** *залежно від датасету.*
    - **афінність:** евклідова. Метрика, за якою рахується відстань між точками і яка використовується для лінкажу.
    - **зв’язність:** відсутня. Дозволяє задати структуру для даних (іншими словами, дозволяє використати структуровану ієрархічну кластеризацію). Ми не використовуємо цю можливість для жодного із наших датасетів.
    - **лінкаж:** Уорда (Ward linkage). Який лінкажний критерій використовувати. Доступні варіанти:
        i. *«Уорда».* Мінімізує варіацію кластерів, що зливаються.

ii. *«Середній».* Середнє попарних відстаней між точками у кластерах.
iii. *«Повний»* (чи *«максимальний»*). Найбільша відстань від точками кластерів (супремум попарних відстаней).
iv. *«одиночний».* Найменша відстань між точками кластерів (інфінум попарних відстаней).

- **поріг відстані:** відсутній. Найбільша відстань, для якої відбувається злиття кластерів. Ми не використовуємо цей параметр.
- **рахувати відстані:** ні. Використовується для побудови дендрограм, і, відповідно, нам не потрібно.

3. Метод середнього зсуву.
   - **ширина кернелу:** *залежно від датасету.* Ширина, що використовується у радіальному кернелі алгоритму. У випадках, коли ми її явно не задавали, вона обчислювалась методом sklearn.cluster.estimate_bandwidth.
   - **початкові точки:** відсутні. Початкові точки, що використовуються при ініціалізації кернелу. Якщо не задаються явно (як у нашому випадку), то для їх оцінки використовується метод sklearn.cluster.get_bin_seeds.
   - **початкові біни:** ні. Якщо так, то початкові точки — це не точки, а дискретизовані версії точок, де точки бінуються на сітці, гранулярність якої залежить від ширини кернелу. Впливає, в першу чергу, на швидкість сходження.
   - **мінімальна частота бінів:** 1. Розглядати лише ті біни, для яких кількість точок щонайменше така. Впливає, в першу чергу, на швидкість сходження.
   - **кластеризувати всі:** так. Кластеризувати усі точки, включаючи так званих «сиріт» — точки, що не належать жодному кернелу. Такі точки приписуються найближчим кластерам. Альтернативою є приписування таким точкам -1 (некластеризовані).
   - **максимум ітерацій:** 300. Максимальна кількість ітерацій алгоритму, перш ніж він буде зупинений. Ми жодного разу не зіткнулись із проблемою несходження алгоритму.
4. DBSCAN.
   - **eps:** *залежно від датасету.* Найбільша відстань між двома семплами така, при якій семпли все ще вважаються сусідніми. *Це **не** найбільша відстань, що зв'язує точки у кластері.*
   - **мінімально семплів:** 5. Кількість семплів (чи їх сукупна вага) у оточенні точки, при якій точка вважається опорною. Рахується включно з самою точкою.

- **метрика:** *залежно від датасету.* Метрика, за якою рахується відстань між будь-якими двома точками.
- **метричні параметри:** відсутні. Використовуються при заданні власної метрики. Ми використовували лише уже доступні «евклідову» та «манхеттенську».
- **метод:** автоматично. Яким саме способом шукати найближчі точки. Впливає лише на швидкість сходження, не на її якість.
- **p:** *залежно від датасету.* Ступінь метрики Мінковського. Ми використовували лише 1 («манхеттен») та 2 («евклід»).

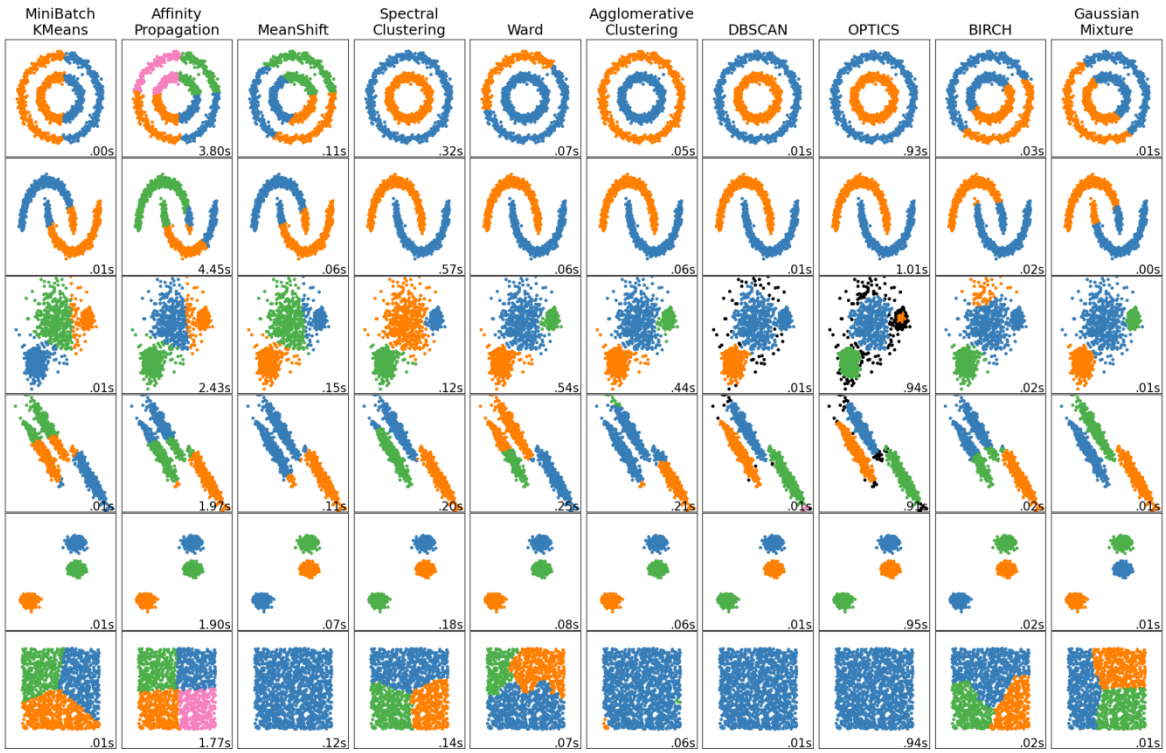
5. Спектральна кластеризація.
   - **кількість кластерів:** *залежно від датасету.*
   - **розв'язник:** «arpack». Яку бібліотеку використовувати для пошуку власних значень. Впливає, в першу чергу, на швидкість та скалювання.
   - **кількість компонент:** відповідає кількості кластерів. Скільки власних векторів використовувати у спектральному вкладенні.
   - **кількість запусків:** 10. Кількість разів, що внутрішній алгоритм «к-середніх» запускався до конвергенції. Запуск відбувається більше одного разу для стабільності фінальної конвергенції.
   - **гамма:** 1. Кернельний коефіцієнт.
   - **афінність:** радіально-кернельна. Спосіб побудови афінної матриці. Доступні способи
     i. «найближчі сусіди». Афінна матриця будується з графу найближчих сусідів.
     ii. «радільно-кернельний». Афінна матриця будується з використанням функції радіального базису (Radian Basis Function, RBF).
     iii. «передрозрахований». Не рахувати афінну матрицю. Вважати вхідну матрицю X не матрицею даних, а афінною матрицею.
     iv. інші варіанти з іншими кернелами.
   - **присвоєння міток:** к-середніх. Спосіб присвоювання міток у просторі вкладення (іншими словами, спосіб кластеризації компонент власних векторів).
6. Гаусове змішування.
   - **кількість кластерів:** *залежно від датасету.*
   - **тип коваріантної матриці:** повна. Кожен компонент має свою, повністю незалежну, матрицю коваріацій, у якій присутні всі елементи. Альтернативні варіанти:
     i. «зв'язані». Усі компоненти мають спільну матрицю коваріацій.

ii. «діагональні». Усі компоненти мають незалежні матриці коваріацій, у яких присутні лише діагональні елементи.
iii. «сферичні». Усі компоненти описуються скалярними варіаціями (іншими словами, матриця коваріацій вироджується у скаляр).

- **толерантність:** 1e-3. Поріг сходження. Критерій для зупинки алгоритму максимізації очікувань.
- **регуляризація коваріацій:** 1e-6. Коефіцієнт регуляризації, що додається до діагоналей кожної матриці. Гарантує, що матриці коваріацій будуть додатніми.
- **максимум ітерацій:** 100. Максимальна кількість ітерацій алгоритму, перш ніж він буде зупинений. Ми жодного разу не зіткнулись із проблемою несходження алгоритму.
- **кількість запусків:** 1. Кількість разів, що алгоритм запускався до конвергенції.
- **спосіб ініціалізації:** к-середніх. Спосіб, яким задаються початкові значення вагам, середнім та точностям (точності — це обернені матриці коваріацій).

ВИСНОВКИ

Дослідженя і аналіз проведено успішно. Якщо проаналізувати дані, то можна зрозуміти, що кожен алгоритм по-різному справляється зі своєю задачею особливо, коли задача стає складніше. Наприклад такі алгоритми, які опираються на відстань, не люблять розмірності, особливо, якщо це евклідові відстані як у DBSCAN через це він деколи показує дуже дивовижні результати. Але на практиці все сильно залежить від параметрів алгоритму та датасету. Проблема з відстанями називається ще “прокляття розмірності”. Ось на малюнку показано яку кластеризацію, який метод найкраще визначає.



Даний термін, який приписують Річарду Беллманом, висловлював труднощі застосування грубої сили (пошуку по решітці) для оптимізації функцій з дуже великою кількістю вхідних змінних. Уявіть що станеться, якщо ви запустите оптимізацію за допомогою Excel Solver з 100,000 вхідними змінними, інакше кажучи - 100,000 різними важелями, кожен з яких можливо треба буде тягнути (можна бути впевненими, що Excel просто вилетіти).

У сучасному світі цей термін також може означати кілька інших проблем, що виникають в ситуаціях, коли дані мають занадто високу розмірність:

У разі, якщо у нас ознак більше ніж спостережень (observations), ми ризикуємо перенавчити модель, що приведе до невідповідним результатами її роботи, так як вона зможе аналізувати тільки вибірку (sample), використану при навчанні.

При занадто великій кількості ознак, розбивати спостереження на кластери стає важко. Занадто висока розмірність призводить до того, що навчається модель думає, що всі спостереження знаходяться на рівній відстані один від одного. А оскільки при кластеризації для кількісної оцінки подібності між спостереженнями вимірюються відстані, наприклад Евклідова відстань, це ставати значною проблемою. Якщо все відстані приблизно рівні один одному, то все спостереження будуть також схожі один на одного, а відповідно ніякі осмислені кластери не можуть бути сформовані.

ДОДАТКИ

```
# для роботи з даними (data-wrangle, "крутити-вертіти даними")

importnumpyasnp

importpandasaspd

# для графіків

importmatplotlib.pyplotasplt

importseabornassns

plt.style.use('ggplot')

importwarnings

warnings.filterwarnings('ignore')

fromsklearn.datasetsimportmake_blobs

# зробити купки

X, y = make_blobs(900, random_state=3)

sns.scatterplot(*X.T, hue=y, palette='bright')

defrun_algos(algos, X, y, X_disp=None, **kwargs):

# X - координати, y - класи

# імпортуємо метрики, якими оцінимо якість кластеризації

# completenessscore - чи немає "розрізів" у кластерах

# homogeneityscore - перевірити "чистоту" передбачених кластерів, чи немає різних класів у рамках одного кластеру
```

adjustedrandindex - чи співпадають передбачені кластери та справжні класи, з точністю до перенумерації

adjustedmutualinformation - наскільки подібні (в інформаційному сенсі) кластеризація та оригінальні класи

fromsklearn.metricsimportcompleteness_score, homogeneity_score, adjusted_rand_score, adjusted_mutual_info_score

сітка графіків

fig, axes = plt.subplots(2, len(algos), figsize=(len(algos)*3, 8))

табличка з порахованими значеннями метрик

scores = pd.DataFrame(index=['Homogenity', 'Completeness', 'AdjustedRand', 'AdjustedMutInfo'])

для кожного алгоритму

for i, algoinenumerate(algos):

print(type(algo).__name__, ' почав роботу.')

підганяємо модель (кластеризуємо)

model = algo.fit(X)

fit -- підігнати

отримати результат кластеризації і зберегти у змінну

y_pred = model.fit_predict(X)

зберегти у табличку обчислені метрики

scores[type(algo).__name__] = [

```
homogeneity_score(y, y_pred),

completeness_score(y, y_pred),

adjusted_rand_score(y, y_pred),

adjusted_mutual_info_score(y, y_pred)

        ]

ifX_disp isNone:

X_disp = X

# виводимо результат кластеризації

# транспонуємо та || розпаковуємо матрицю на два вектори

# hue -- колір, відтінок. кольори відповідають результатам кластеризації

sns.scatterplot(*X_disp.T, hue=y_pred, palette='bright', ax=axes[0][i], **kwargs)

axes[0][i].set_title(f'{type(algo).__name__}')


ifX.shape[1] == 2:

# якщо аглоритм дозволяє присвоїти кластер НОВІЙ точці після кластеризації

# або якщо алгоритм розділяє площину на кластери

ifhasattr(model, 'predict'):

                n = 50

                a, b = np.meshgrid(np.linspace(X_disp[:, 0].min(), X_disp[:, 0].max(), n),

np.linspace(X_disp[:, 1].min(), X_disp[:, 1].max(), n))
```

```
c = model.predict(np.c_[a.flatten(), b.flatten()]).reshape(n, n)


axes[0][i].contourf(a, b, c, alpha=0.3)


# виводимо правильні відповіді

sns.scatterplot(*X_disp.T, hue=y, palette='bright', ax=axes[1][i], **kwargs)

# задати заголовок графіку

axes[1][i].set_title('Істинні')

plt.tight_layout()

returnscores

# importalgorithmstouse

fromsklearn.clusterimportKMeans, DBSCAN, MeanShift, SpectralClustering, AgglomerativeClustering

fromsklearn.mixtureimportGaussianMixture


# make a listofalgorithms

algos = [KMeans(3), DBSCAN(eps=0.7), MeanShift(3), GaussianMixture(3), SpectralClustering(3), AgglomerativeClustering(3)]


# кластеризувати та оцінити якість кластеризації

run_algos(algos, X, y)
```

```
rs = np.random.randint(0, 1000)

X, y = make_blobs([200, 600, 300], cluster_std=[1.0, 2.5, 0.5], random_state=804)


sns.scatterplot(*X.T, hue=y, palette='bright')

plt.title(rs)

algos = [KMeans(3), DBSCAN(eps=0.5), MeanShift(2), SpectralClustering(3), GaussianMixture(3)]


run_algos(algos, X, y)

wine_red = pd.read_csv('winequality-red.csv', delimiter=';')

wine_red['type'] = 'R'


wine_white = pd.read_csv('winequality-white.csv', delimiter=';')

wine_white['type'] = 'W'


data = pd.concat([wine_red, wine_white], ignore_index=True)

data = data.sample(1500)


X, y = data.drop('type', axis=1), data.type

# нормалізація даних
```

```
X = (X - X.mean()) / X.std()

fromsklearn.decompositionimportKernelPCA


X2 = KernelPCA(2, kernel='cosine').fit_transform(X, y)


sns.scatterplot(*X2.T, hue=y, s=10, alpha=0.5)

algos = [KMeans(2), DBSCAN(eps=4, metric='manhattan'), AgglomerativeClustering(2), GaussianMixture(2)]

run_algos(algos, X, y, X_disp=X2, alpha=0.6)
```

ДЖЕРЕЛА

1. К.В.Воронцов Лекции по алгоритмам кластеризации и многомерного шкалирования

2. Kleinberg J. An Impossibility Theorem for Clustering

3. E.M. Mirkes, K-meansand K-medoidsapplet

4. *YizongCheng.* Mean Shift, ModeSeeking, andClustering

5. Permuter, H.; Francos, J.; Jermyn, I.H. (2003). *Gaussian mixture models of texture and colour forimage database retrieval*

6.*E. Arias-Castro, G. Chen, G. Lerman.* Spectral clustering basedon local linear approximations