КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Звіт з лабораторної роботи
з дисципліни «Кластеризація та класифікація інформації»
Тема «Порівняльний аналіз алгоритмів кластеризації»

Виконав студент 2 курсу магістратури
Факультету радіофізики, електроніки
та комп'ютерних систем
спеціальності «Комп'ютерні системи та мережі»
Долгополов Антон

Київ 2018

ЗМІСТ

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1.Загальний огляд методів кластеризації

Метою кластерного аналізу є розділення спостережень на групи ("кластери"), так що попарні розбіжності між тими, що відносяться до одного кластеру, мають тенденцію бути меншими, ніж у різних кластерах. Алгоритми кластеризації поділяються на три різних типи: комбінаторні алгоритми, моделювання сумішей та непараметричні моделі . Комбінаторні алгоритми працюють безпосередньо на спостережуваних даних без прямої посилання на базову модель імовірності. Моделювання сумішей передбачає, що дані є ідентично незалежним розподіленим зразком з деякої популяції, що описується функцією щільності ймовірності. Ця функція густини характеризується параметризованою моделлю, яка вважається сумішшю функцій щільності компонентів; кожен компонент щільності описує один з кластерів. Дана модель дає дані за максимальною ймовірністю або відповідними баєсівськими підходами. Непараметричний підхід намагається безпосередньо оцінити різні режими функції щільності ймовірностей. Спостереження, найближчі до кожного відповідного режиму, визначають окремі кластери.

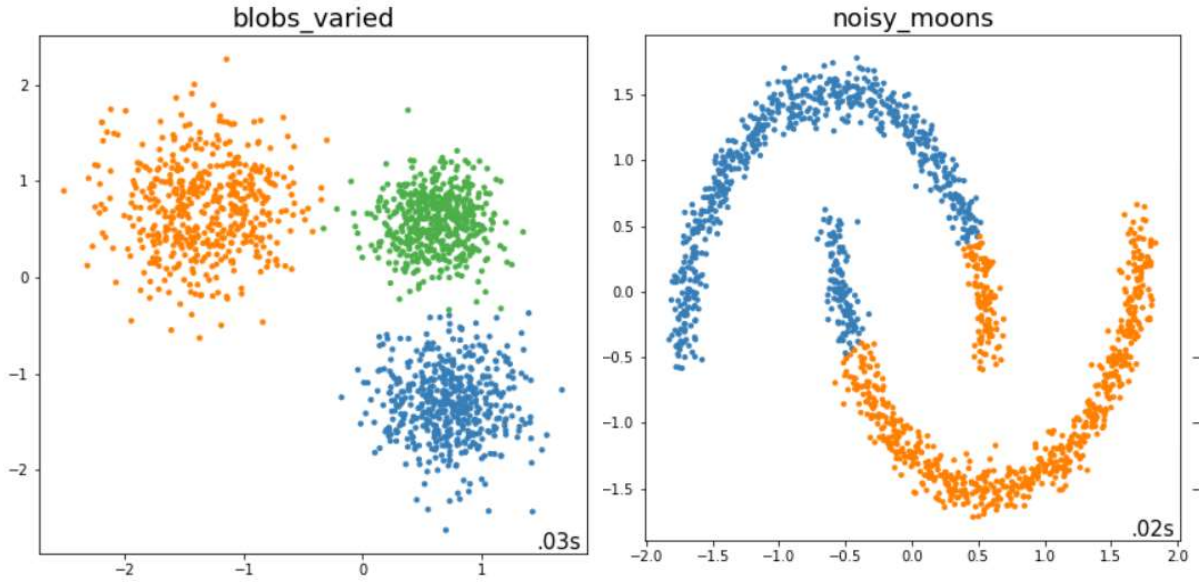
1.2.MiniBatch K-Means

Алгоритм K-means кластеризує дані, намагаючись відокремити зразки в n групах рівної дисперсії, мінімізуючи критерій, відомий як інерція або сукупність квадратів відстаней всередині кластера (рис 1.2 а). Для цього алгоритму потрібно вказати кількість кластерів. Це чудово масштабується до великої кількості зразків і використовується в широкому діапазоні областей застосування в багатьох різних областях. Алгоритм розподіляє набір зразків на непересічні кластери, кожен з яких описується середнім зразком в кластері, що зазвичай називають "центроїдом" кластера. Метою є вибір центроїдів, які мінімізують інерцію, або критерій квадрата відстаней в межах кластеру:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} \left(\left| \left| x_i - \mu_j \right| \right|^2 \right)$$

Інерцію можна визнати як міру того, як складаються внутрішньо-когерентні кластери. Це має перелік різних недоліків:

- інерція робить припущення, що кластери є опуклими та ізотропними, що не завжди так. Тому це погано працює на витягнуті скупчення або різноманіття з неправильними формами (рис 1.2 б).
- інерція не є нормованою метрикою. Це є прикладом пріорного обмеження моделі, де користувач вважає, що нижчі значення краще, а нуль - оптимальний. Але в багато просторових вимірах евклідові відстані, як правило, роздуваються (це приклад так званого "прокляття розмірності").



а *б*

Рис. 1.2 Кластери даних (з різною дисперсією – *а*, з неопуклою формою – *б*)

За умови достатнього часу, K-means завжди збігається, однак це може привести до локального мінімуму. Результат сильно залежить від ініціалізації центроїдів. Тому обчислення часто виконується кілька разів, з різними ініціалізаціями центроїдів.

MiniBatchKMeans - це різновид K-Means, який використовує обрані міні-частини даних (mini batches) для зменшення часу обчислень, але намагається оптимізувати ту саму цільову функцію. Mini batches - це підмножини вхідних даних, які випадково відбираються під час кожної ітерації навчання. Це різко зменшують обсяги обчислень, необхідні для знаходження цільового рішенням. На відміну від інших алгоритмів, що зменшують час збіжності K-Means, MiniBatchKMeans виробляє результати, які дає лише трохи гірші.

Алгоритм ітератується між двома основними кроками, подібними до звичайного K-Means. На першому етапі зразки складаються випадковим чином з набору даних, щоб сформувати міні-пакет. Потім вони присвоюються до найближчого центроїда. На другому кроці центроїди оновлюються. На відміну від K-Means, це робиться на основі вибірки. Для кожного зразка в міні- частині призначений центроїд оновлюється шляхом прийняття середнього потоку зразка та всіх попередніх зразків, присвоєних цьому центроїду. Це впливає на зменшення швидкості зміни центроїди з часом. Ці кроки виконуються до досягнення конвергенції або заздалегідь заданої кількості ітерацій. MiniBatchKMeans збігається швидше, ніж KMeans, але якість результатів зменшується, але на практиці ця різниця в якості може бути досить мала.

1.3.Spectral Clustering

Метод спектральної кластеризація пов'язаний з методами локального багатовимірного масштабування, і є узагальненням стандартних методів. Він розроблений для ситуацій, коли інші методи, такі як K- Means, погано працюють з невипуклими кластерами, такими як концентричні кола (рис 1.3).

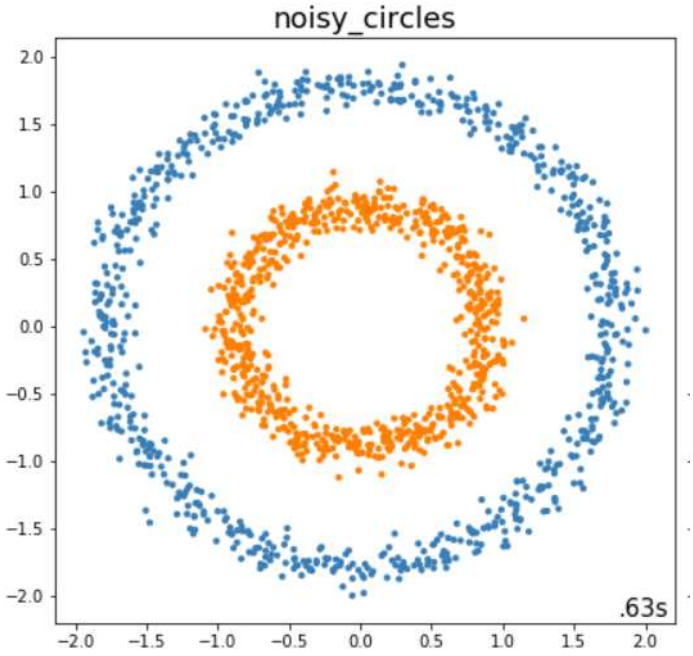


Рис. 1.3 Концентричні кола

Відправною точкою є N × N матриця попарних подібностей $s_{ii'} \geq 0$ між усіма парами спостереження. Ми представляємо спостереження на неорієнтованому графіку подібності $G = < V, E >$. Вершини представляють спостереження, а пари вершин зв'язані ребрами, якщо їх подібність є позитивною (або перевищує якусь порогову величину). Ребра є зваженими завдяки $s_{ii'}$. Кластеризація тепер перефразовується як проблема розділу графів, де ми визначаємо пов'язані компоненти з кластерами. Ми хочемо розбити графік так, щоб ребра між різними групами мали малу вагу, а всередині однієї групи - велику. Ідея спектральної кластеризації полягає в тому, щоб побудувати графіки подібності, які відображають локальні взаємозв'язки між спостереженнями.

Існує багато способів визначити матрицю подібності та пов'язаний з нею граф подібності, що відображає локальну поведінку. Найбільш популярним є взаємний граф k-nearest neighbor. Визначимо N_k як симетричний набір близьких точок-пар; а саме пара (i, i') знаходиться в N_k, якщо точка i є серед k найближчих сусідів i' або навпаки. Тоді ми з'єднуємо всі симетричні найближчі сусіди і даємо їм ребро ваги $w_{ii'} = s_{ii'}$; інакше вага ребра дорівнює нулю. Відповідно ми встановили нульові всі попарні подібності не в N_k і встановимо граф для цієї модифікованої матриці подібності.

Матриця вагів ребер $W = \{w_{ii'}\}$ з графа подібності називається матрицею сусідства. Степінь вершини i є сумою ваг пов'язаних з ним ребр $g_i = \sum_{i'} w_{ii'}$. Нехай G - діагональна матриця з діагональними елементами g_i. Нарешті, граф Лапласіана визначається$L = G - W$. Це називається ненормированним графом Лапласіана. Також, запропоновано ряд нормалізованих варіантів - це стандартизує лапласіан по відношенню до ступенів g_i вузла, наприклад, $L = I - G^{-1}W$. Спектральне кластування знаходить m власних векторах $Z_{N \times m}$, що відповідають m найменшим власним значенням L (ігноруючи тривіальну константу власного вектора). Використовуючи стандартний метод, такий як K-means, ми потім кластеризуємо рядки Z, щоб дати кластери вихідних точок даних.

1.4.DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) – алгоритм кластеризації даних на основі щільності. Якщо заданий набір точок в деякому просторі, він об'єднує точки, які тісно розташовані разом (точками з багатьма близькими сусідами), та позначуючи точки що лежать в інших регіонах низької щільності як викиди. Формальний опис метода: нехай задана деяка симетрична функція відстані $\rho(x, y)$ і константи ϵ і m. Тоді

1. Назвемо область $E(x)$, для якої $\forall y: \rho(x, y) \leq \epsilon$, ϵ – окіл об'єкта.
2. Кореневим об'єктом або ядерним об'єктом ступеня m називається об'єкт, ϵ – окіл якого містить не менше m об'єктів: $|E(x)| \geq m$.
3. Об'єкт p безпосередньо щільно-досяжний з об'єкта q, якщо $p \in E(q)$ і q – кореневий об'єкт.
4. Об'єкт p щільно-досяжний з об'єкта q, якщо $\exists p_1, p_2 \ldots p_n, p_1 = q, p_n = p$, такі що $\forall i \in 1 \ldots n-1: p_{i+1}$ безпосередньо щільно-досяжний з p_i

Виберемо який-небудь кореневої об'єкт p з даних, позначимо його і помістимо всіх його безпосередньо щільно-досяжних сусідів в список обходу. Тепер для кожного q зі списку: позначимо цю точку, і, якщо вона теж коренева, додамо всіх її сусідів в список обходу. Тривіально доводиться, що кластери помічених точок, сформовані в ході цього алгоритму максимальні (тобто їх не можна розширити ще однією точкою, щоб задовольнялися умови) і зв'язні в сенсі щільно-досяжності. Звідси випливає, що якщо ми обійшли не всі точки, можна перезапустити обхід з якого-небудь іншого кореневого об'єкта, і новий кластер не поглине попередній. Переваги алгоритму:

1. DBSCAN не вимагає визначити кількість кластерів в даних apriori, на відміну від k-means.
2. DBSCAN може знаходити кластери довільної форми (рис 1.4). Він може навіть знайти кластер повністю оточений (але не пов'язаний з) іншим кластером. Через параметр m, зменшується ефект, коли кластери з'єднуються тонкою лінією точок.
3. DBSCAN має поняття шуму, і є стійким до викидів, та інше.

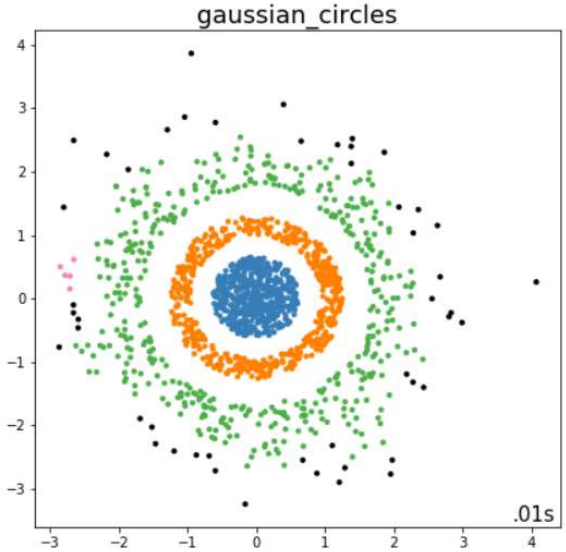


Рис. 1.4 Гаусівські кільця

Але також є і ряд недоліків. Якість DBSCAN залежить від використовуваної відстані. Найбільш поширеною метрикою відстані є евклідова відстань. Спеціально для великогабаритних даних цей показник може бути практично марним завдяки так званому "Прокляттю розмірності", що ускладнює пошук відповідного значення для ε. Цей ефект, однак, також присутній в будь-якому іншому алгоритмі на основі евклідової відстані.

1.5.Gaussian Mixture та EM алгоритм

Припустимо, нам надається навчальний набір даних $\{x^{(1)}\dots x^{(m)}\}$ та ми хочемо змоделювати дані, використовуючи спільний розподіл $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})*p(z^{(i)})$, де $z^{(i)} \sim$ Multinomial(φ) і $x^{(i)}|z^{(i)} = j \sim N(\mu_j, \Sigma_j)$. Нехай k означає число значень, що приймає $z^{(i)}$. Отже, наша модель постулує, що кожен $x^{(i)}$ що був згенерований так: випадковим вибором $z^{(i)}$ від {1,..., k}, а потім $x^{(i)}$ був отриманий від одного з k Гаусіанів в залежності від $z^{(i)}$. Така модель має назву Gaussians mixture (суміш Гаусіанів). Таким чином, параметри нашої моделі φ, μ та Σ. Щоб оцінити їх, ми можемо записати правдоподібність наших даних:

$$\begin{aligned}\ell(\phi,\mu,\Sigma) &= \sum_{i=1}^{m}\log p(x^{(i)};\phi,\mu,\Sigma)\\ &= \sum_{i=1}^{m}\log\sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)};\mu,\Sigma)p(z^{(i)};\phi)\end{aligned}$$

Якщо знати, якими мають бути $z^{(i)}$, проблема максимальної правдоподібності стає легшою з точки зору максимізації по відношенню до параметрів φ, μ і Σ.

Алгоритм EM (expectations-maximization – максимізація очікувань) є ітераційним алгоритмом, який має два основні етапи. На E-кроці, алгоритм присвоює " функції" для кожної точки даних, заснованої на її відносній щільності під кожним компонентом суміші, а M-етап перерозподіляє параметри щільності компонентів на основі поточних функцій. Застосовуючись до нашої проблеми, на E-кроці він намагається "вгадати" значення $z^{(i)}$. На M-кроці він оновлює параметри нашої моделі на основі нашого припущення. Оскільки в M-кроці ми вважаємо, що припущення в першій частині були правильними, максимізація ставала легшою.

Таке налаштування EM - "м'яка" версія кластеризації K-means, що робить ймовірнісні (а не детерміністичні) призначення точок для кластерних центрів. У міру зміни дисперсії $\sigma 2 \to 0$ ці ймовірності стають 0 і 1, а два алгоритми збігаються. Подібно до K- means, він також чутливий до локальних оптимумів, тому повторна кластеризація з ініціалізацією різними початковими параметрами є гарною практикою.

Алгоритм EM у вигляді псевдо коду:

Repeat until convergence: {

(E-step) For each i, j, set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

$$\begin{aligned}
\phi_j &:= \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}, \\
\mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}
\end{aligned}$$

}

2. ПРАКТИЧНА ЧАСТИНА

В процесі виконання роботи було досліджено стабільність роботи описаних раніше алгоритмів кластеризації на основі різних типів кластерів. Зокрема, розглядаються випадки опуклих та неопуклих скупчень точок різної форми та характеристик, вкладені та «екзотичні» форми кластерів. Базові представлення даних та результати роботи алгоритмів можна знайти у додатку 1.

Використовуючи різні методи трансформації даних, було здійснено порівняльний аналіз якості роботи алгоритмів. Для цього використовувались метрики двох видів: 1) ті, що потребують даних вигляду $\{x^{(i)}, y^{(i)}\}$ – де $x^{(i)}$ – вибірка характеристик та $y^{(i)}$ – відповідна приналежність до кластера; 2) ті, що потребують лише характеристик вхідних даних, тобто $\{x^{(i)}\}$.

Для генерації, обробки, візуалізації та аналізу даних було створено окремий модуль (utils) та використовувались такі інструменти з відкритим кодом як scikit-learn, numpy, matplotlib, pandas, що базуються на мові програмування Python. Приклад програмного коду що використовувався доступний у додатку 3.

2.1.Методи трансформації даних

Для трансформації даних використовувались ряд методів, що загалом базується на генераторі випадкових чисел, з встановленням стану (задля отримання однакових результатів при багатьох запусках). До переліку входить:

- стиснення та розтягування простору шляхом множення на різні матриці
- змінна положення $n_1, n_2 \ldots n_m$ точок всередині набору вхідних даних на δ величину, де n – кількість точок, отримані від ітерування послідовності Фібоначчі, а значення δ отримане з нормального розподілу з дискретною дисперсією 0.05, 0.1 та 0.15

Метод трансформації простору та його обґрунтування описане у [9]. Прийом для δ-зміни положення точок використовується для отримання більшої кількості даних та перевірки стабільності роботи алгоритмів кластеризації. У додатку 2 знаходяться приклади описаних методів.

2.2.Метрики оцінювання якості

У випадку наявності приналежності вхідних даних до відповідних кластерів, можна визначити інтуїтивно зрозумілу метрику за допомогою умовного ентропійного аналізу. Зокрема, такими двома цільовими функціями для будь-якого розподілу кластерів є:

- однорідність – у кожному кластері містяться лише члени одного класу
- повнота – всі члени даного класу призначаються одному кластеру

Їх гармонійним середнім є V-величина.

Якщо ж приналежність до класу невідома (що досить поширено у повсякденно житті), оцінка повинна виконуватися за допомогою самої моделі. Silhouette коефіцієнт є прикладом такої оцінки, де більший показник коефіцієнта

відноситься до моделі з більш точно визначеними кластерами. Він визначається для кожного зразка і складається з двох величин:

- a – середня відстань між зразком та всіма іншими точками того ж класу
- b – середня відстань між зразком та всіма іншими точками в найближчому кластері.

Коефіцієнт s для одиничної вибірки потім задається як:

$$s = \frac{b - a}{max(a, b)}$$

Детальна інформація про використані метрики описана у [11] та [12].

2.3.Налаштування алгоритмів та результати роботи

В більшій мірі, для аналізу використовувались налаштування за замовчуванням, що доступні в пакеті програм scikit-learn. Модифіковані параметри моделей, що змінювались в залежності від вхідних даних:

	dataset_name	eps	n_clusters
0	squares	0.30	3
1	blobs	0.70	3
2	circles	0.30	3
3	blobs_varied	0.50	3
4	noisy_circles	0.20	2
5	gaussian_circles	0.25	3
6	noisy_moons	0.30	2

. Значення цих параметрів наступне: *eps* – значення параметра найближчого сусіда в алгоритмі DBSCAN; *n_clusters* – наперед задана кількість шуканих кластерів; *dataset_name* – назва набору даних для збереження.

Для порівняння роботи алгоритмів, було розраховано середнє значення оцінюваних значень метрик для кожного типу набору даних, а саме V-величина та Silhouette коефіцієнт. Остаточні дані що включають результати відображені у таблицях 1 і 2, додатку 4. У додатку 5 також наведено їх графічне відображення.

Спектральна кластеризація дає найкращі результати на майже всіх даних (окрім blobs та blobs_varied, де є 2м), проте це досягається ціною дуже великого часу виконання (в середньому 35-40с на 1500 точок, в той час як k-means справляється за 0.01с). Окрім того, 3 з 4 розглянутих методів потребують від користувача наперед визначену кількість шуканих кластерів, що не завжди є відомим. Розглянутий DBSCAN вирішує проблему, проте потребує зусиль на налаштування. Така статична модель одразу стає не придатною до зміни природи вхідних даних (зміна розмірності чи відстаней у просторі), а отже потребує нового втручання. Слід відзначити, що отримані результати не є вичерпними та потребують подальшою роботи. Тестування інших форм кластерів та знаходження оптимальних параметрів алгоритмів в залежності від трансформації простору даних може бути важливим внеском в покращення їх роботи.

ВИСНОВКИ

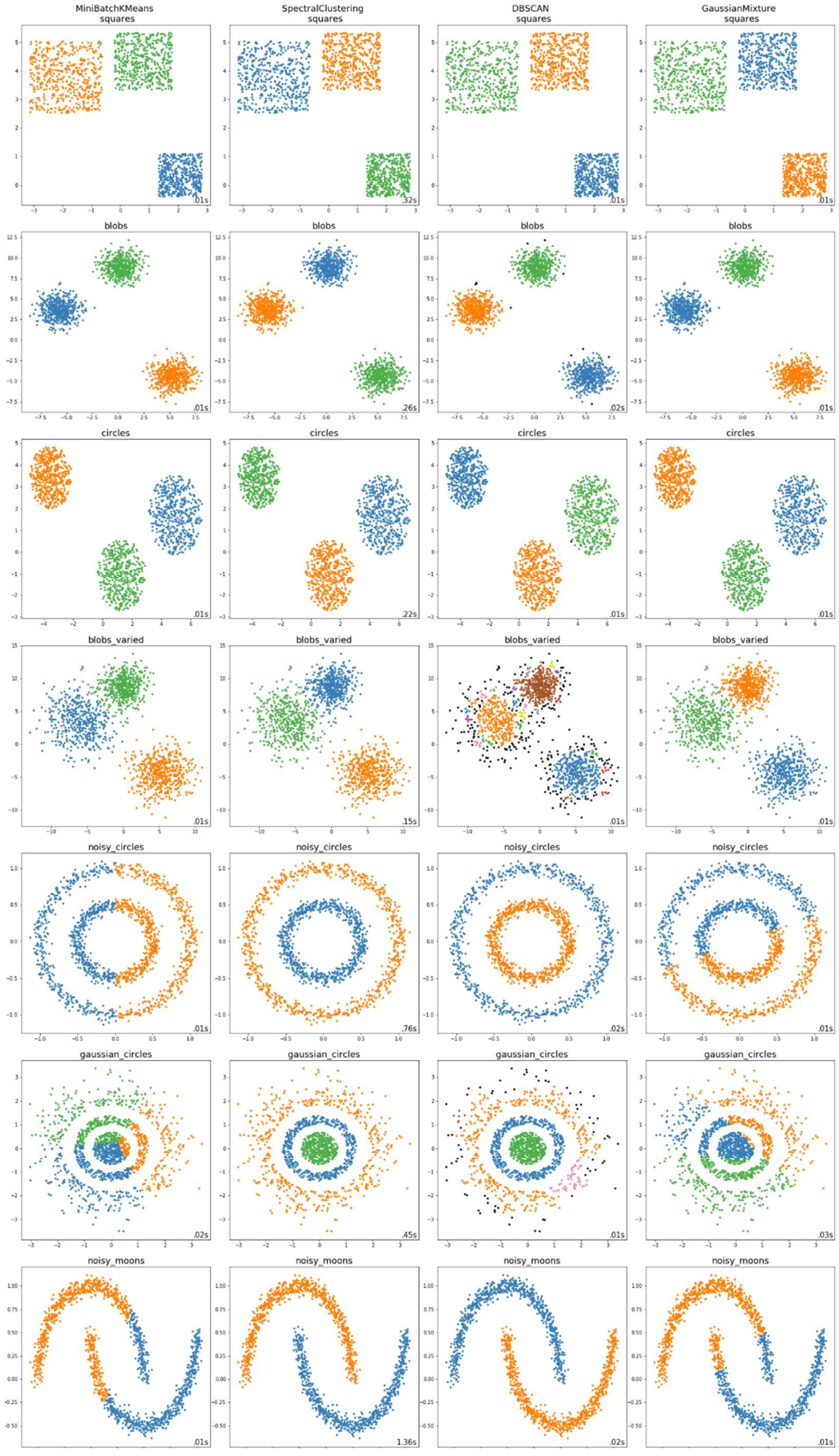
В ході виконання роботи було досліжено роботу алгоритмів кластеризації, що базуються на різних принципах. Аналіз показує, що не має універсального методу, який може ефективно обробити усі форми даних. Це є гарним відображенням теореми “No free launch”. Стабільність алгоритмів у великій мірі залежить від форми та розташування кластерів. MiniBatch K-means та Gaussian Mixture показують гарні результаті у випадку опуклих кластерів. Останній навіть справляється із витягнутими формами скупчень точок з їх близьким розташуванням, в той час як перший все одно намагається мінімізувати відстані всередині груп. Проте вони погано працюють у випадку неопуклих та вкладених кластерів. Два інші методи, а саме DBSCAN та Spectral Clustering, справляються з цим та дають кращі результати в загальному. Вони також успішно здійснюють пошук неочевидних груп, але мають ряд недоліків. Великий час роботи та потреба в тонкому налаштуванні звужує сферу їх використання.

ДЖЕРЕЛА

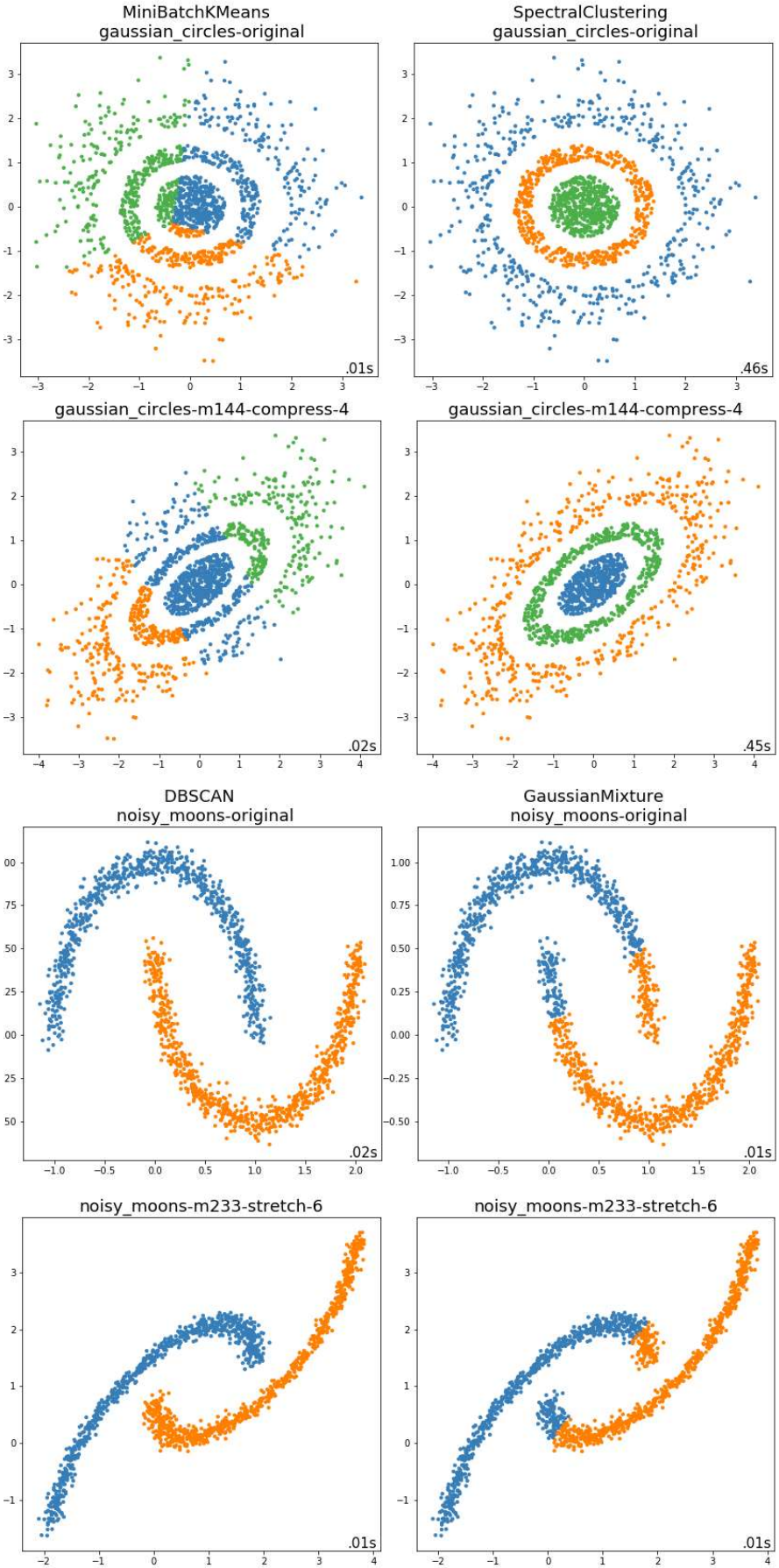
1. Wikipedia contributors. "Cluster analysis." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 11 Dec. 2018. Web. 12 Dec. 2018.
2. Ng, A.Y., Jordan, M.I. and Weiss, Y., 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849-856).
3. Ng A., *Lecture notes on Machine learning course*
4. Sculley, D., 2010, April. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web* (pp. 1177-1178). ACM.
5. Ester, M., Kriegel, H.P., Sander, J. and Xu, X., 1996, August. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, No. 34, pp. 226-231).
6. Arthur, D. and Vassilvitskii, S., 2007, January. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.
7. Rosenberg, A. and Hirschberg, J., 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.
8. Davies, D.L. and Bouldin, D.W., 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2), pp.224-227.
9. Pei, Y. and Zaïane, O., 2006. A synthetic data generator for clustering and outlier analysis. *Department of Computing science, University of Alberta, edmonton, AB, Canada*.
10. Moon, T.K., 1996. The expectation-maximization algorithm. *IEEE Signal processing magazine*, *13*(6), pp.47-60.
11. Peter J. Rousseeuw (1987). “Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis”. Computational and Applied Mathematics 20: 53–65
12. V-Measure: A conditional entropy-based external cluster evaluation measure Andrew Rosenberg and Julia Hirschberg, 2007
13. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. No. 10. New York, NY, USA: Springer series in statistics, 2001.

ДОДАТКИ

Додаток 1



Додаток 2


MiniBatchKMeans
gaussian_circles-original
.01s
SpectralClustering
gaussian_circles-original
.46s
gaussian_circles-m144-compress-4
.02s
gaussian_circles-m144-compress-4
.45s
DBSCAN
noisy_moons-original
.02s
GaussianMixture
noisy_moons-original
.01s
noisy_moons-m233-stretch-6
.01s
noisy_moons-m233-stretch-6
.01s

Додаток 3

```
import os
import time
import warnings
from itertools import cycle, islice

warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import cluster, datasets, mixture
from sklearn.neighbors import kneighbors_graph
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

from IPython.display import display

import utils


def process(X,y, params):
    df = []
    for i_m, (X_m, y_m) in zip(utils.fibs(n_samples), utils.modify_dataset(X, y, n_samples, scale=0.05, seed=seed)):
        compressed = utils.trfm_dataset(X_m,y_m,'c', seed)
        stretched  = utils.trfm_dataset(X_m,y_m,'s', seed)
        pipe = []

        default = params.copy()
        default['dataset_name'] = '{0}-{1}'.format(params['dataset_name'],'original')
        pipe.append(((X, y), default))

        default = params.copy()
        default['dataset_name'] = '{0}-m{1}'.format(params['dataset_name'], i_m)
        pipe.append(((X_m, y_m), default))

        for i,((Xtc,ytc),(Xts,yts)) in enumerate(zip(compressed[1:], stretched[1:])):
            default = params.copy()
            default['dataset_name'] = '{0}-m{1}-{2}-{3}'.format(params['dataset_name'],i_m,'compress', i)
            pipe.append(((Xtc, ytc), default))

            default = params.copy()
            default['dataset_name'] = '{0}-m{1}-{2}-{3}'.format(params['dataset_name'],i_m,'stretch', i)
            pipe.append(((Xts, yts), default))

        df.append(utils.clustering(pipe, standardize=False, pic_name='{0}/m{1}'.format(folder+params['dataset_name'], i_m)))
    return pd.concat(df)
```

Додаток 4

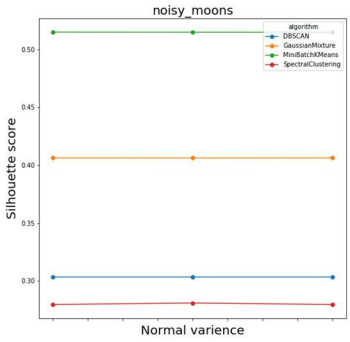
Таблиця 1. V-величина

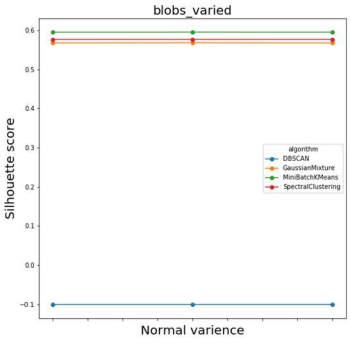
	varience	0.05	0.10	0.15
	algorithm			
blobs	DBSCAN	0.967784	0.967784	0.967784
	GaussianMixture	1.000000	0.997782	1.000000
	MiniBatchKMeans	0.976327	0.976512	0.976578
	SpectralClustering	0.972699	0.972160	0.973642
blobs_varied	DBSCAN	0.539400	0.539400	0.539400
	GaussianMixture	0.942608	0.943623	0.942630
	MiniBatchKMeans	0.818091	0.819240	0.819279
	SpectralClustering	0.860161	0.860135	0.860160
circles	DBSCAN	0.943183	0.943183	0.943183
	GaussianMixture	0.955044	0.954541	0.958969
	MiniBatchKMeans	0.954944	0.956235	0.954930
	SpectralClustering	0.999018	0.995518	0.996928
gaussian_circles	DBSCAN	0.555874	0.555874	0.555874
	GaussianMixture	0.275644	0.275305	0.272498
	MiniBatchKMeans	0.218993	0.220910	0.222098
	SpectralClustering	0.732200	0.750900	0.750067
noisy_circles	DBSCAN	0.391304	0.391304	0.391304
	GaussianMixture	0.000036	0.000036	0.000034
	MiniBatchKMeans	0.000072	0.000070	0.000077
	SpectralClustering	0.900335	0.900335	0.900335
noisy_moons	DBSCAN	0.397993	0.397993	0.397993
	GaussianMixture	0.295171	0.295292	0.295264
	MiniBatchKMeans	0.089638	0.089905	0.089063
	SpectralClustering	0.909130	0.907395	0.909128
squares	DBSCAN	0.924128	0.924128	0.924128
	GaussianMixture	0.967844	0.966280	0.967806
	MiniBatchKMeans	0.898768	0.900356	0.899371
	SpectralClustering	0.968651	0.971173	0.972107

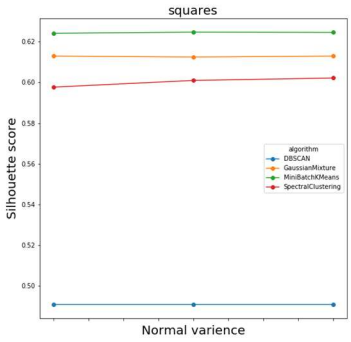
Таблиця 2. Silhouette коефіцієнт

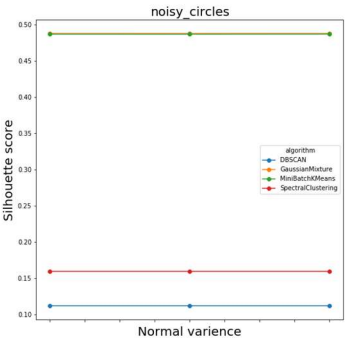
	varience	0.05	0.10	0.15
	algorithm			
blobs	DBSCAN	0.675098	0.675098	0.675098
	GaussianMixture	0.765749	0.764906	0.765749
	MiniBatchKMeans	0.772469	0.772487	0.772510
	SpectralClustering	0.754838	0.755061	0.754822
blobs_varied	DBSCAN	-0.100417	-0.100417	-0.100417
	GaussianMixture	0.567289	0.567800	0.567261
	MiniBatchKMeans	0.594707	0.594926	0.594816
	SpectralClustering	0.576530	0.576526	0.576528
circles	DBSCAN	0.479899	0.479899	0.479899
	GaussianMixture	0.690972	0.692509	0.693513
	MiniBatchKMeans	0.690943	0.691447	0.690689
	SpectralClustering	0.667792	0.664029	0.665041
gaussian_circles	DBSCAN	0.075065	0.075065	0.075065
	GaussianMixture	0.410997	0.409347	0.409469
	MiniBatchKMeans	0.411428	0.411578	0.412612
	SpectralClustering	0.166301	0.165185	0.165476
noisy_circles	DBSCAN	0.111784	0.111784	0.111784
	GaussianMixture	0.487480	0.487453	0.487469
	MiniBatchKMeans	0.486561	0.486527	0.486605
	SpectralClustering	0.159701	0.159701	0.159701
noisy_moons	DBSCAN	0.303347	0.303347	0.303347
	GaussianMixture	0.406249	0.406220	0.406236
	MiniBatchKMeans	0.514997	0.514927	0.514842
	SpectralClustering	0.279571	0.280886	0.279569
squares	DBSCAN	0.490699	0.490699	0.490699
	GaussianMixture	0.612956	0.612470	0.612952
	MiniBatchKMeans	0.624169	0.624736	0.624600
	SpectralClustering	0.597671	0.600965	0.602145

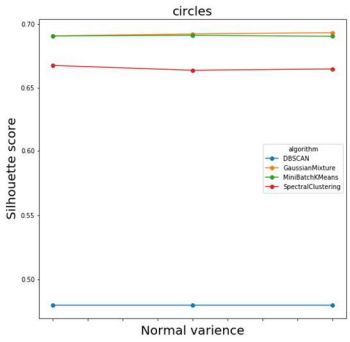
Додаток 5

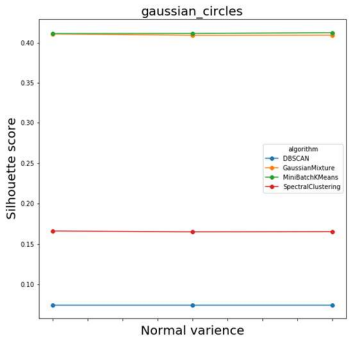

noisy_moons
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

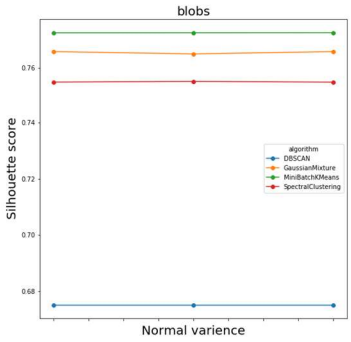

blobs_varied
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

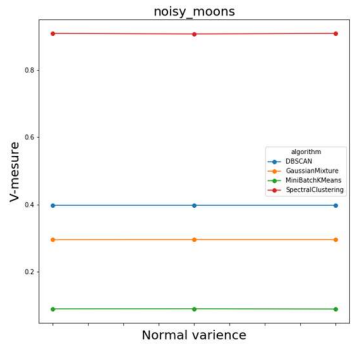

squares
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

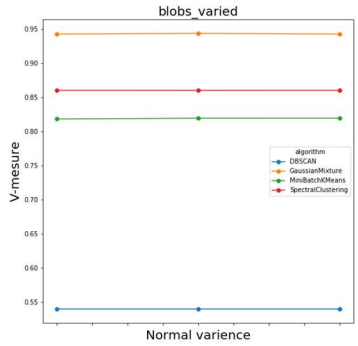

noisy_circles
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

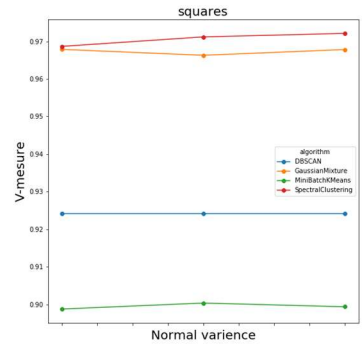

circles
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

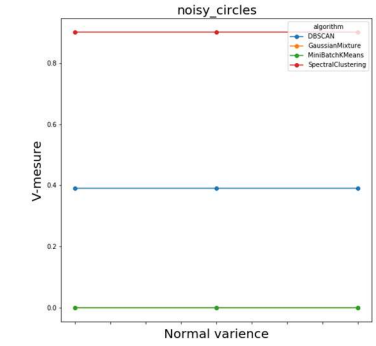

gaussian_circles
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

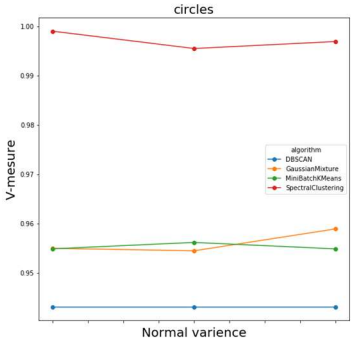

blobs
Silhouette score
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

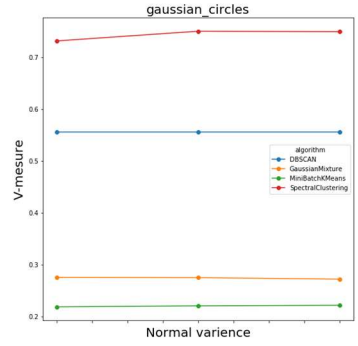

noisy_moons
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering

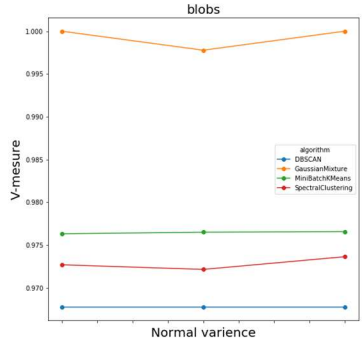

blobs_varied
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering


squares
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering


noisy_circles
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering


circles
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering


gaussian_circles
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering


blobs
V-mesure
Normal varience
algorithm
DBSCAN
GaussianMixture
MiniBatchKMeans
SpectralClustering