A Neighborhood-Based Clustering by Means of the Triangle Inequality

Maksim Makaranka

17 stycznia 2024

1 Definicja problemu

Problem grupowania (ang. clustering) polega na podziale zbioru danych na podzbiory (grupy, klastry), tak aby elementy należące do tego samego podzbioru były do siebie podobne, a elementy należące do różnych podzbiorów były od siebie różne. Podobieństwo i różnica między elementami są mierzone za pomocą odpowiednich funkcji (metryk) odległości lub podobieństwa. Problem grupowania jest ważny, ponieważ pozwala na odkrywanie ukrytych struktur, wzorców i zależności w danych, które mogą być użyteczne dla różnych celów, w tym klasyfikacja, rekomendacja, segmentacja, kompresja i inne.

Problem grupowania jest ściśle związany z innymi metodami eksploracji danych, takimi jak wykrywanie reguł asocjacyjnych i wzorców sekwencyjnych. Reguły asocjacyjne służą do odkrywania zależności i powtarzalności w zbiorach danych, które mogą być użyteczne dla różnych celów, takich jak rekomendacja, wizualizacja, kompresja czy redukcja wymiarowości. Wzorce sekwencyjne służą do odkrywania uporządkowanych ciągów działań lub zdarzeń, które występują w przewidywalnej kolejności. Grupowanie może być użyte do podziału zbioru danych na podzbiory, które mają podobne reguły asocjacyjne lub wzorce sekwencyjne, co pozwala na lepsze zrozumienie i opisanie danych. Inną metodą eksploracji danych jest klasyfikacja, która polega na przypisywaniu predefiniowanych etykiet do obiektów na podstawie ich cech. Różnica między klasyfikacją a grupowaniem polega na tym, że klasyfikacja jest nadzorowaną techniką uczenia się, która wymaga wcześniejszego zdefiniowania klas i przykładów treningowych, podczas gdy grupowanie jest nienadzorowaną techniką uczenia się, która grupuje obiekty na podstawie ich podobieństwa bez wcześniejszej wiedzy o klasach.

Jedną z najistotniejszych rodzin algorytmów grupowania są algorytmy gęstościowe, które tworzą grupy na podstawie gęstości obiektów w przestrzeni, czyli liczby obiektów w określonym promieniu (należących do otoczenia epsilonowego). Przykładem takich algorytmów może być **DBSCAN**[2], polegający na poszukiwaniu punktów rdzeniowych, czyli takich, których otoczenie epsilonowe zawiera co najmniej *MinPts* punktów (parametr określający min. ilość punktów niezbędnych do utworzenia grupy), a następnie badaniu otoczeń epsilonowych tych punktów. Innym przykładem algorytmów gęstościowych jest **NBC**[3]. Nieco skurczony opis tego algorytmu czytelnik znajdzie w sekcji 2.1.

Jednak wyznaczanie odległości pomiędzy każdą parą punktów w przestrzeni, wymagane w algorytmach **DBSCAN** i **NBC**, jest operacją nieefektywną wydajnościowo. W tym celu zostały zaprojektowane wersje tych algorytmów, eliminujące konieczność wyznaczania tych odległości w zbiorze poprzez zastosowanie nierówności trójkąta (ang. the triangle inequality property, TI). Odpowiednie wersje algorytmów noszą nazwy **TI-DBSCAN**[4] i **TI-NBC**[5], ten drugi został scharakteryzowany w sekcji 2.

2 Charakterystyka algorytmu

W dalszej części dokumentacji odległość między dwoma punktami p i q będzie oznaczana przez *distance(p,q)*. W zależności od zadania i posiadanych danych możemy stosować różne metryki odległości. Na przykład, jeśli posiadamy zbiór punktów z przestrzeni metrycznej, to można zastosować dobrze znane metryki, takie jak euklidesowa (otoczenie punktu ma kształt kulisty), Manhattan (otoczenie punktu ma kształt prostokątny)[5] lub Minkowski (którą można uznać za uogólnienie zarówno metryki euklidesowej, jak i Manhattan). W przypadku bardziej skomplikowanych zbiorów danych, np. kiedy mamy do czynienia zarówno z danymi ilościowymi (np. wiek, dochód, wzrost), jak i jakościowymi (np. płeć, kolor, kraj), powinniśmy zastosować bardziej złożone metryki.

2.1 NBC - grupowanie ze względu na k$^+$-sąsiadów

Aby móc scharakteryzować badany algorytm **TI-NBC**, należy najpierw przyjrzeć się wersji podstawowej **NBC** oraz wprowadzić kilka związanych pojęć:

- otoczeniem epsilonowym (ang. Eps-neighborhood) $N_{Eps}(p)$ punktu p jest zbiór wszystkich punktów q w danym zbiorze D, które są odległe od punktu p o nie więcej niż *Eps*

$$N_{Eps} = \{q \in D | distance(p,q) \leqslant Eps\} \tag{1}$$

- k^+*-sąsiedztwo* punktu $p(k^+NN(p))$ to zbiór wszystkich punktów ze zbioru D, które różnią się od p i są od niego nie dalej niż jego najdalszy k-ty sąsiad

- odwrotne k^+-*sąsiedztwo* punktu $p(Rk^+NN(p))$ to zbiór punktów ze zbioru D, dla których p należy do ich k^+-*sąsiedztwa*

$$Rk^+NN(p) = \{q \in D | p \in k^+NN(p)\} \tag{2}$$

- gęstość podprzestrzeni jest określana przez współczynnik gęstości NDF, który jest ilorazem liczby punktów w odwrotnym k^+-*sąsiedztwie* i liczby punktów w k^+-*sąsiedztwie*

$$NDF(p) = \frac{|Rk^+NN(p)|}{|k^+NN(p)|} \tag{3}$$

- punkt p jest punktem rdzeniowym, jeśli $NDF(p) \geqslant 1$
- punkt rdzeniowy jest traktowany jako ziarno, które wraz z punktami z jego k^+-*sąsiedztwa* tworzy gęstą przestrzeń, którą można uznać za grupę lub część grupy
- kiedy punkt rdzeniowy jest dołączany do grupy, wszystkie punkty z jego k^+-*sąsiedztwa* również są włączane do tej grupy, chyba że wcześniej zostały przydzielone do innej grupy

Jednak w takim klasycznym podejściu musimy obliczać odległości od punktu p do wszystkich punktów w zbiorze D, co jest operacją bardzo kosztowną.

2.2 Wykorzystanie TI do efektywnego wyznaczania sąsiedztw

Nierówność trójkąta dla dowolnych trzech punktów p, q, r może mieć postać:

$$distance(p,r) \leqslant distance(p,q) + distance(q,r) \Leftrightarrow distance(p,q) \geqslant distance(p,r) - distance(q,r) \tag{4}$$

Powyższa tożsamość razem z definicją epsilon-otoczenia (1) prowadzą do lematu, dowód którego można znaleźć w oryginalnej pracy autorów[5]:

$$distance(p,q) \geqslant distance(p,r) - distance(q,r) > Eps \implies q \notin N_{Eps}(p) \wedge p \notin N_{Eps}(q) \tag{5}$$

Z tego lematu wynika, że jeśli znamy różnicę odległości dwóch punktów p i q do jakiegoś punktu r większą niż *Eps*, możemy stwierdzić, że $q \notin N_{Eps(p)}$ bez obliczania rzeczywistej odległości między p i q. Zgodnie z tym autorzy pracy sformułowali i udowodnili następujące twierdzenie[5]:

Twierdzenie 1 *Niech r będzie dowolnym punktem, a D zbiorem punktów uporządkowanym w sposób niemalejący względem ich odległości do r. Niech p będzie dowolnym punktem w D, q_f punktem następującym po p w D takim, że:*

$$distance(q_f,r) - distance(p,r) > Eps, \tag{6}$$

a q_b punktem poprzedzającym p w D takim, że:

$$distance(p,r) - distance(q_b,r) > Eps. \tag{7}$$

Wtedy zachodzi:

a) q_f i wszystkie punkty następujące po q_f w D nie należą do $N_{Eps}(p)$.

b) q_b i wszystkie punkty poprzedzające q_b w D nie należą do $N_{Eps}(p)$.

Następnie autorzy przedstawiają podstawy teoretyczne przydatne do określania *k-sąsiedztwa* dowolnego punktu $p(k^+NN(p))$, formułując i dowodząc kolejne twierdzenie[5]:

Twierdzenie 2 *Niech r będzie dowolnym punktem, a D zbiorem punktów uporządkowanym w sposób niemalejący względem ich odległości do r. Niech p będzie dowolnym punktem w D, a Eps wartością taką, że $|N_{Eps}(p)| > k$, q_f punktem następującym po p w D takim, że:*

$$distance(q_f,r) - distance(p,r) > Eps, \tag{8}$$

a q_b punktem poprzedzającym p w D takim, że:

$$distance(p,r) - distance(q_b,r) > Eps. \tag{9}$$

Wtedy zachodzi:

a) q_f i wszystkie punkty następujące po q_f w D nie należą do $kNB(p)$.

b) q_b i wszystkie punkty poprzedzające q_b w D nie należą do $kNB(p)$.

2.3 Budowanie indeksu k-sąsiedztwa przy użyciu TI

W tym rozdziale rozpatrzmy algorytm, który wykorzystuje **Twierdzenie 2** do wyznaczania *k-sąsiedztw* dla wszystkich punktów w danym zbiorze danych D i zapisuje je jako *indeks k-sąsiedztwa*. Algorytm składa się z następujących kroków:

1. Obliczamy odległość każdego punktu w D do punktu odniesienia r, np. do punktu o wszystkich współrzędnych równych 0.
2. Sortujemy punkty względem ich odległości do r.
3. Dla każdego punktu p wD wywołujemy funkcję, która zwraca $kNB(p)$. Funkcja ta działa następująco:
   (a) Identyfikuje najpierw te k punktów q następujących i poprzedzających punkt p w D, dla których różnica między *distance(p,r)* a *distance(q,r)* jest najmniejsza. Te punkty są traktowane jako kandydaci na k najbliższych sąsiadów p.
   (b) Sprawdza pozostałe punkty poprzedzające i następujące po p w D (zaczynając od punktów bliższych p w uporządkowanym zbiorze D) jako potencjalni k najbliżsi sąsiedzi p, aż zostaną spełnione warunki określone w **Twierdzeniu 2**. Jeśli tak, żadne inne punkty w D nie są sprawdzane, ponieważ jest gwarantowane, że nie należą do $kNB(p)$.
   (c) Modyfikuje wartość *Eps* za każdym razem, gdy znajdowany jest nowy kandydat na k najbliższego sąsiada.

3 Struktura rozwiązania

Struktura rozwiązania została zbudowana w następujący sposób:

- Folder `/data` zawiera zbiory danych pobierane podczas uruchamiania notebooków, a także wstępnie obliczone wyniki grupowania.
- Folder `/src` zawiera wszystkie pliki źródłowe projektu. Składa się on z:
  - Folderu `/models`, który zawiera implementacje modeli.
  - Pliku `AnalysisHelper.py`, który jest plikiem pomocniczym do wizualizacji wyników i obliczania miar.
- Plik `config.py` zawiera stałe i ścieżki do danych.
- `Jupyter Notebooki` zawierają wyniki algorytmów dla różnych 2D i 3D zbiorów danych i wartości parametru k.
- Plik `README.md` zawiera krótki opis rozwiązania, instrukcję użytkowania i wymagania technologiczne.

4 Badania

Celem badań jest sprawdzenie działania i efektywności zaimplementowanego algorytmu grupowania **TI-NBC** na różnych zbiorach danych. Badania te obejmują porównanie algorytmu **TI-NBC** do podstawowej wersji **NBC**, wersji **NBC** zaimplementowanej z użyciem biblioteki języka Python - *scikit-learn*, oraz wyników algorytmu **k-means**, które zostały wstępnie obliczone przy użyciu już wspomnianej biblioteki *sklearn*. Do tego celu został wykorzystany *clustering-benchmark*, zaimplementowany przez pracowników Politechniki Warszawskiej i dostępny publicznie.

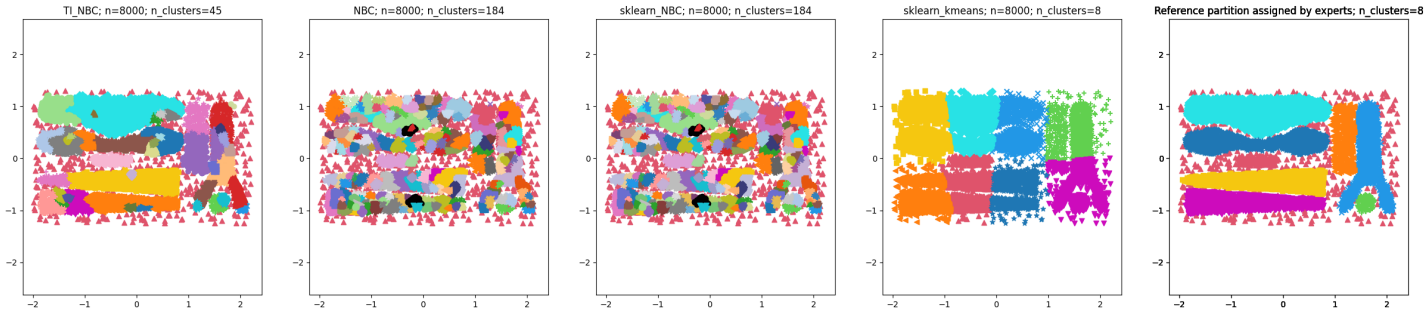
W celu oceny efektywności czasowej algorytmów, przeprowadzono porównania czasów ich wykonania dla różnych wielkości zbiorów danych oraz dla różnych wartości parametru k. Wyniki porównań znajdują się w Tabelach 1 i 2. Wizualizacje grupowań dla różnych k przedstawiono na Rysunkach 1, 2, 3.

	TI-NBC	**NBC**	**NBC sklearn**
$n = 800$	$3.61\,s$	$3.33\,s$	$15.9\,ms$
$n = 4096$	$12.1\,s$	$37.6\,s$	$66.8\,ms$
$n = 8000$	$26.7\,s$	$2min\,26s$	$172\,ms$

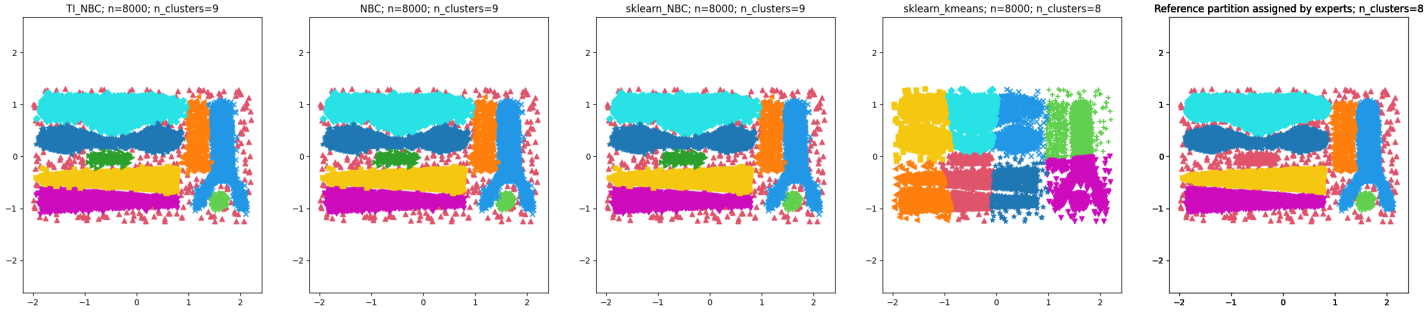
Tabela 1: Wpływ wielkości zbioru danych na czas wykonania, $k = 40$

	TI-NBC	**NBC**	**NBC sklearn**
$k = 10$	$11.1\,s$	$2min\,24s$	$120\,ms$
$k = 40$	$26.7\,s$	$2min\,26s$	$172\,ms$
$k = 120$	$1min\,6s$	$2min\,27s$	$294\,ms$

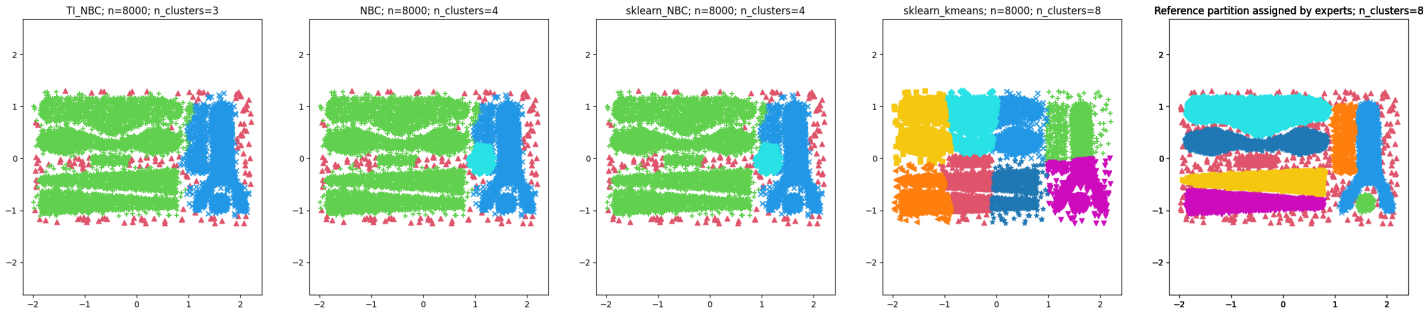
Tabela 2: Wpływ wartości parametru k na czas wykonania, $n = 8000$



Rysunek 1: Wyniki grupowania na zbiorze danych `chameleon_t8_8k`, $n = 8000$, $k = 10$



Rysunek 2: Wyniki grupowania na zbiorze danych `chameleon_t8_8k`, $n = 8000$, $k = 40$

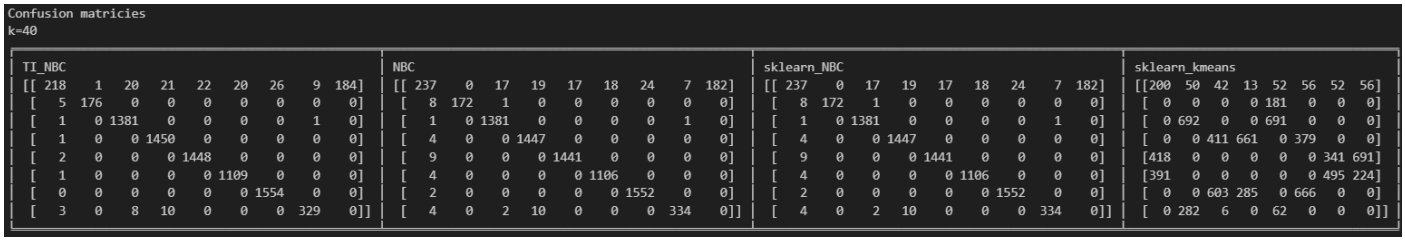


Rysunek 3: Wyniki grupowania na zbiorze danych `chameleon_t8_8k`, $n = 8000$, $k = 120$

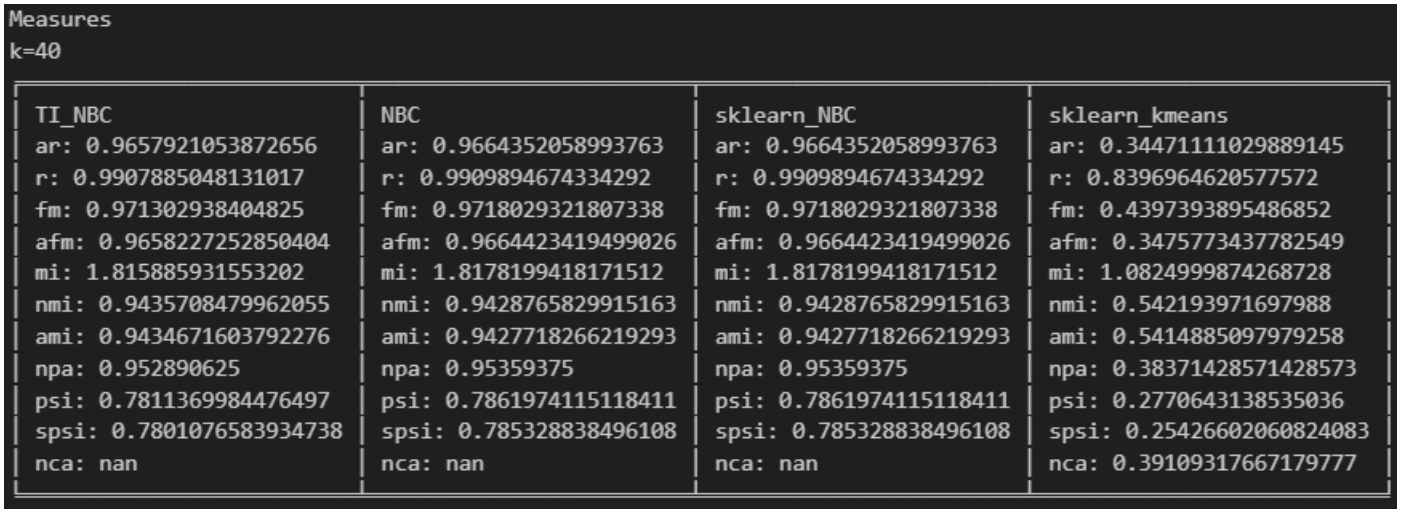
Aby ocenić skuteczność grupowania przeprowadzono porównania grupowań, uzyskanych za pomocą różnych algorytmów, do grupowania wskazanego przez twórcę datasetu. Zastosowano różnych metryk, dostępnych w bibliotece *clustering-benchmark*, w szczególności:

- **Confusion Matrix** - macierz, która pokazuje liczbę prawidłowych i nieprawidłowych prognoz podzielonych według klas. Umożliwia ocenę skuteczności modelu i identyfikację wzorców błędów.
- **Rand Score (R, AR)** - obliczają procent par obiektów, które są umieszczone w tych samych lub różnych klastrach w obu podziałach.
- **Pair Sets Index (PSI, SPSI)** - obliczane są jako liczba par obiektów, które są umieszczone w tych samych klastrach w obu podziałach, podzieloną przez liczbę wszystkich par obiektów.
- **Fowlkes–Mallows (FM, AFM)** - obliczają średnią geometryczną precyzji i czułości.
- **Mutual Information (MI, NMI, AMI)** - metryki oparte na teorii informacji, które oceniają, jak dobrze jeden podział na klastry przewiduje inny.

Wyniki zastosowania metryk dla zbioru danych `other/chameleon_t8_8k` i parametru $k = 40$ znajdują się na rysunkach 4 i 5.



Rysunek 4: Macierzy pomyłek (confusion matricies), `chameleon_t8_8k`, $n = 8000$, $k = 40$



Rysunek 5: Wartości metryk, `chameleon_t8_8k`, $n = 8000$, $k = 40$

5 Wnioski

Na podstawie przeprowadzonych badań, można sformułować następujące wnioski:

- Czas wykonania każdego algorytmu zwiększa się wraz z rozmiarem zbioru danych. Ten wzrost jest najbardziej widoczny w przypadku zwykłego **NBC**. Zauważono, że dla większych zbiorów danych, działanie **TI-NBC** jest znacznie szybsze od zwykłego **NBC**, np. dla $n = 8000$, **TI-NBC** działał prawie 6 razy szybciej. Jednakże, dla małych zbiorów danych (w naszym przypadku $n = 800$) może się okazać, że zwykłe **NBC** działa równie szybko, a nawet szybciej niż **TI-NBC**. Warto również wspomnieć, że algorytm zaimplementowany z użyciem komponentów dostarczanych przez *sklearn.neighbors* jest znacznie szybszy od zaimplementowanych **TI-NBC** i **NBC** we wszystkich testowanych przypadkach.

- Zauważono, że zwykłe **NBC** ma mniej więcej ten sam czas wykonania dla różnych wartości parametru k, co wynika z tego, że polega na brutalnym obliczeniu odległości pomiędzy każdą parą punktów i nie zależy od k. W przypadku **TI-NBC**, czas wykonania rośnie razem z k, co wynika z faktu, że współczynnik ten bezpośrednio wpływa na ilość sprawdzanych punktów i obliczanych odległości podczas poszukiwania *k-sąsiedstw*.

- Wizualizacja wyników grupowania zbioru danych `chameleon_t8_8k` wskazuje, że dobór parametru k w dla algorytmów **NBC** mocno zależy od przetwarzanego zbioru danych. Dla mniejszych k, algorytm wykrywa większą liczbę klastrów niż powinien rzeczywiście, a dla większych k - odwrotnie, mniejszą liczbę klastrów. Dla $k = 40$ udało się osiągnąć dobre pogrupowanie danych. Algorytm bardzo dobrze radzi sobie z wykrywaniem szumu w danych i znacznie lepiej niż **k-means** wykrywa grupy punktów o różnej gęstości, nie łącząc szumu i części tych gęstych grup.

- Macierzy pomyłek grupowania `chameleon_t8_8k` wskazują, że wszystkie algorytmy z grupy **NBC** skutecznie odnalazły grupy, zapowiedziane przez twórcę datasetu, jednocześnie tworząc z najgęściej ułożonego szumu dodatkowy klastr. Dla algorytmu **k-means** macierzy bardzo klarownie wykazują niepoprawność przypisań punktów dla poszczególnych grup.

- Wartości policzonych metryk grupowania ww. zbioru danych sygnalizują dobre pogrupowanie danych w przypadku wszystkich algorytmów **NBC** (delikatnie gorsze dla **TI-NBC**), i znacznie gorsze dla **k-means**.

Podsumowując, badania te dostarczyły cennych informacji na temat działania i efektywności algorytmu grupowania **TI-NBC** na różnych zbiorach danych. Wyniki te mogą być przydatne dla innych badaczy i praktyków zainteresowanych zastosowaniem tego algorytmu w swojej pracy.

Literatura

[1] Kod źródłowy https://github.com/maksanm/TI-NBC

[2] https://en.wikipedia.org/wiki/DBSCAN

[3] Zhou, S., Zhao, Y., Guan, J., Huang, J. (2005). A Neighborhood-Based Clustering Algorithm. In: Ho, T.B., Cheung, D., Liu, H. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2005. Lecture Notes in Computer Science(), vol 3518. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11430919_43

[4] Kryszkiewicz, M., Lasek, P. (2010). TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds) Rough Sets and Current Trends in Computing. RSCTC 2010. Lecture Notes in Computer Science(), vol 6086. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13529-3_8

[5] Kryszkiewicz, M., Lasek, P. (2010). A Neighborhood-Based Clustering by Means of the Triangle Inequality. In: Fyfe, C., Tino, P., Charles, D., Garcia-Osorio, C., Yin, H. (eds) Intelligent Data Engineering and Automated Learning – IDEAL 2010. IDEAL 2010. Lecture Notes in Computer Science, vol 6283. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15381-5_35

[6] https://clustering-benchmarks.gagolewski.com/index.html