UNIWERSYTET ŚLĄSKI W KATOWICACH
WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH
INSTYTUT FIZYKI IM. AUGUSTA CHEŁKOWSKIEGO

Kamil Piekorz
NR ALBUMU: 327270

Wykorzystanie metod uczenia maszynowego do identyfikacji cząstek w różnych eksperymentach fizyki jądrowej i cząstek

PRACA DYPLOMOWA INŻYNIERSKA

Promotor: dr Katarzyna Schmidt, prof. UŚ

Katowice 2024

Spis treści

1 Wstęp

Celem pracy jest przetestowanie dostępnych algorytmów do analizy skupień na danych pochodzących z eksperymentów fizyki jądrowej i cząstek. Bardzo często identyfikację produktów reakcji jądrowych przeprowadza się mając do dyspozycji sygnały z dwóch detektorów, ułożonych w stos, działających w koincydencji. Efektem tego jest dwuwymiarowy plot, na którym poszczególne izotopy tworzą różnego rodzaje skupienia.

Do odróżnienia skupień użyto nienadzorowanych metod uczenia maszynowego, które zostaną opisane w kolejnych rozdziałach. Nienadzorowane uczenie maszynowe to taki rodzaj uczenia maszynowego, którego zadaniem jest odnajdywanie w zbiorze danych wzorców bez znajomości ich etykiet oraz przy minimalnej ingerencji człowieka. [1]

2 Technologie i metody klasteryzacji

Klasteryzacja, znana również jako analiza skupień, to technika analizy danych, której celem jest podzielenie zbioru danych na grupy, zwane klastrami, takie że obiekty wewnątrz jednego klastra są do siebie podobne, a klastry są od siebie różne. Jest to istotne narzędzie w dziedzinie eksploracyjnej analizy danych, uczenia maszynowego oraz innych obszarów, gdzie istnieje potrzeba zrozumienia struktury i wzorców ukrytych w danych.

Istnieje wiele technologii i metod klasteryzacji, z których każda ma swoje własne zastosowania, zalety oraz ograniczenia. Duża część z nich jest zaimplementowana w bibliotece SciKitLearn [2]. W dalszej części zgłębione zostaną metody klasteryzacji, przedstawiając ich zasady działania, zastosowania oraz wady i zalety a także sposób ich używania w w/w bibliotece. Każda z tych technologii ma swoje miejsce w różnych scenariuszach analizy danych, dlatego wybór odpowiedniej metody zależy od specyfiki zbioru danych oraz celów analizy.

2.1 Algorytm MeanShift

Meanshift to algorytm używany w analizie skupień, zwłaszcza w zadaniach segmentacji obrazu oraz śledzenia obiektów. Algorytm ten został zaproponowany przez Dorina Comanaescu i Peter Meer w 2002 roku. [3]

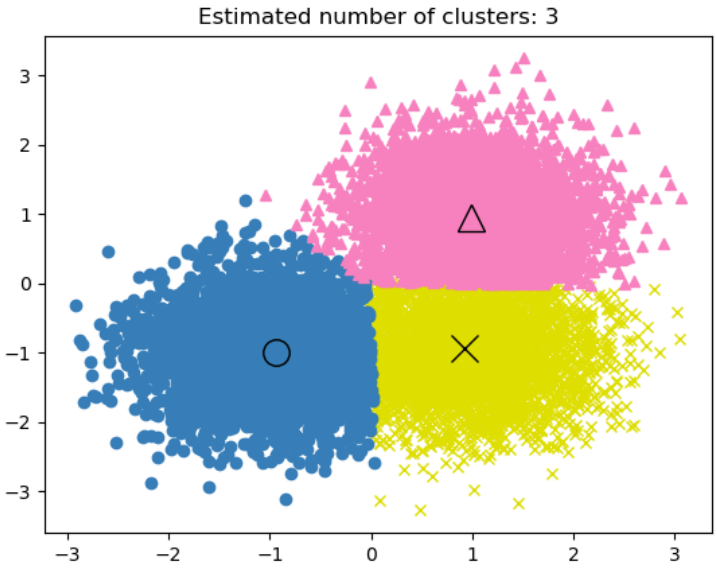
Algorytm ma na celu znalezienie plam w gładkiej gęstości próbek. Jest to algorytm oparty na centroidach, który działa poprzez aktualizowanie kandydatów na centroidy, tak aby były one średnią punktów w danym regionie. Kandydaci ci są następnie filtrowani w celu wyeliminowania bliskich sobie duplikatów, aby utworzyć ostateczny zestaw centroidów. Rysunek nr 1 przedstawia przykład klasteryzacji przy uzyciu metody MeanShift.

Głównym celem Meanshift jest znajdowanie maksimum funkcji gęstości prawdopodobieństwa danych. W kontekście analizy skupień oznacza to, że algorytm stara się znaleźć obszary, gdzie punkty danych mają największą koncentrację. Działa to w taki sposób, że dla każdego punktu w zbiorze danych przesuwa się w kierunku wzrostu funkcji gęstości, aż osiągnie obszar o maksymalnej gęstości.

Algorytm MeanShift przyjmuje następujące parametry wejściowe:

- Szerokość pasma (**bandwidth**), która określa rozmiar okna, w którym algorytm przeszukuje dane w poszukiwaniu podobnych punktów.

- Maksymalna liczba iteracji (**max iter**) określa maksymalną liczbę iteracji, jaką algorytm będzie wykonywał przed zakończeniem działania. Jeśli algorytm nie zbiegnie do stabilnego punktu, pomimo maksymalnej liczby iteracji, zostanie zakończony.
- Minimalna liczba punktów (**min bin freq**) w danym koszyku. Punkty, które są rzadziej reprezentowane niż ta wartość, zostaną uznane za szum i nie będą brane pod uwagę w dalszej analizie.



Rysunek 1: MeanShift, źródło [2]

2.2 DBSCAN

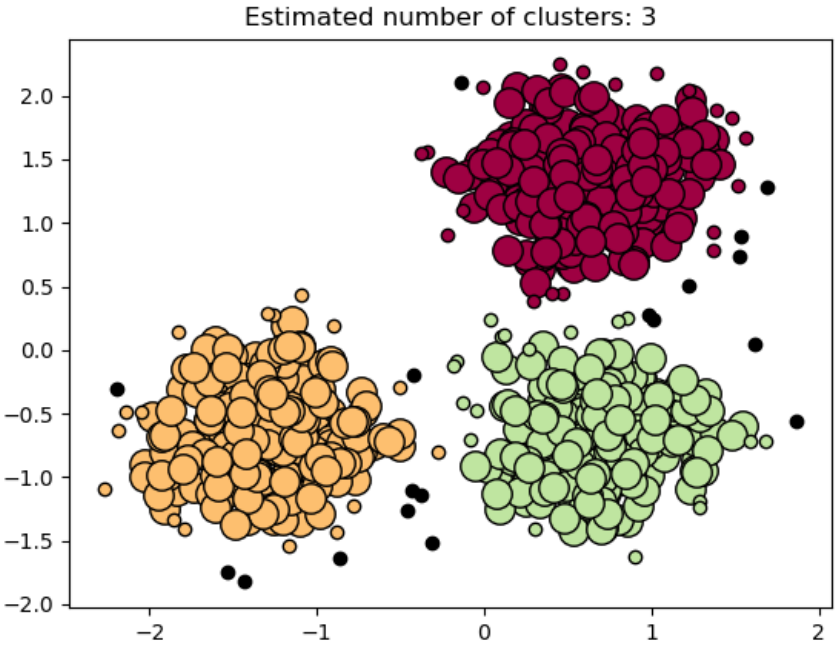
DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to algorytm służący do grupowania danych w przestrzeni wielowymiarowej na podstawie gęstości punktów. Został zaproponowany przez Martina Ester, Hans-Petera Kriegela, Jorga Sandera i Xiaowei Xu w 1996 roku. [4]

DBSCAN jest efektywny w identyfikowaniu klastrów o różnych kształtach i rozmiarach, a także radzi sobie z wykrywaniem punktów odstających (szumowych). Rysunek nr 2 przedstawia przykład klasteryzacji przy uzyciu metody DBSCAN.

Jednakże elastyczność i brak konieczności wcześniejszego określenia liczby klastrów sprawiają, że algorytm ten jest popularny w analizie danych przestrzennych i wykrywaniu grup w danych, gdzie klastry nie są jednorodne pod względem gęstości.

Algorytm DBSCAN przyjmuje dwa parametry wejściowe:

- promień sąsiedztwa (**eps**), który określa odległość, w jakiej dwa obiekty są uznawane za sąsiadów
- minimalną liczbę obiektów (**minPts**), która określa liczbę sąsiadów, jakie musi mieć obiekt, aby zostać uznanym za punkt rdzeniowy (core point).



Rysunek 2: Przykład DBSCAN, źródło [2]

2.2.1 Sposób działania DBSCAN

Algorytm DBSCAN działa w następujący sposób:

1. Algorytm rozpoczyna od losowego punktu danych. Dla każdego punktu w zbiorze danych sprawdza, czy ma on co najmniej minPts sąsiadów w promieniu ϵ. Jeśli tak, to punkt jest uznawany za punkt centralny, a jego sąsiedzi są uwzględniani w procesie.

2. Dla każdego punktu centralnego algorytm przechodzi przez jego sąsiadów i przypisuje ich do tego samego klastra.

3. Proces ten kontynuowany jest dla każdego nowego punktu centralnego, a klastry są budowane w miarę, jak punkty centralne i brzegowe są ze sobą powiązane.

4. Punkty, które nie są ani punktami centralnymi, ani brzegowymi, są traktowane jako szum i nie są przypisywane do żadnego klastra.

Niestety, wybór odpowiednich wartości dla parametrów, ϵ i **minPts**, może być trudny oraz mieć znaczączy wpływ na jakość wyników. Zbyt mała wartość ϵ może prowadzić do utworzenia zbyt wielu małych klastrów, podczas gdy zbyt duża wartość ϵ może powodować, że wszystkie punkty są przypisane do jednego klastra.

2.3 Algorytm K-Means

K-means [5], [6] to popularny algorytm w dziedzinie analizy skupień (clustering), używany do grupowania danych na podstawie ich podobieństwa. Algorytm ten został zaproponowany przez Stuarta Lloyda w 1957 roku, a później został rozwinięty i udoskonalony przez inne osoby.

Głównym celem algorytmu k-means jest podzielenie zbioru danych na k skupień (grup) w taki sposób, aby obiekty wewnątrz jednego klastra były do siebie jak najbardziej podobne, a jednocześnie klastry były jak najbardziej od siebie odmienne.

Algorytm k-means ma kilka wad, takich jak wrażliwość na początkowe wartości środków klastrów oraz trudności z obsługą klastrów o różnych kształtach i rozmiarach. Wrażliwość na wybór początkowych centroidów oznacza, że wynik grupowania może się różnić w zależności od tego, jakie centroidy zostaną wybrane losowo na początku. W związku z tym istnieją różne modyfikacje i rozszerzenia algorytmu, które zostały zaproponowane w celu poprawy jego wydajności w różnych sytuacjach. Aby zminimalizować ten problem, można wybrać centroidy w kilku różnych punktach i wybrać te, które dają najlepszy wynik.

Mimo tych ograniczeń, k-means jest nadal szeroko stosowany w praktyce do analizy skupień, zwłaszcza w dziedzinach takich jak analiza danych, eksploracyjna analiza danych, rozpoznawanie obrazów czy przetwarzanie języka naturalnego. Rysunek nr 3 przedstawia przykład klasteryzacji przy uzyciu metody K-Means.

Algorytm K-Means przyjmuje następujące parametry wejściowe:

- Liczba klastrów (**n clusters**), określa oczekiwaną liczbę klastrów, na jakie ma zostać podzielony zbiór danych. W praktyce, dobór odpowiedniej liczby klastrów może wymagać przeprowadzenia analizy i oceny danych.

- Inicjalizacja początkowych środków klastrów (**n init**), określa liczbę razy, jakie algorytm będzie próbował losowo inicjalizować początkowe środki klastrów. Domyślnie parametr ten jest ustawiony na wartość 10.

2.3.1 Sposób działania K-Means

Algorytm k-means jest algorytmem grupowania nienadzorowanego, który dzieli zbiór danych na k grup, tak aby dane w każdej grupie były jak najbardziej podobne do siebie, a dane z różnych grup były jak najbardziej różne. Algorytm działa w następujących krokach:

1. Algorytm losowo wybiera k punktów, które będą początkowymi centroidami grup.

   Centroidy są punktami, które reprezentują każdą grupę. Ważne jest, aby centroidy były dobrze rozmieszczone w przestrzeni danych, aby reprezentowały różnorodne grupy.

2. Algorytm przypisuje każdy punkt danych do najbliższego centroidu.

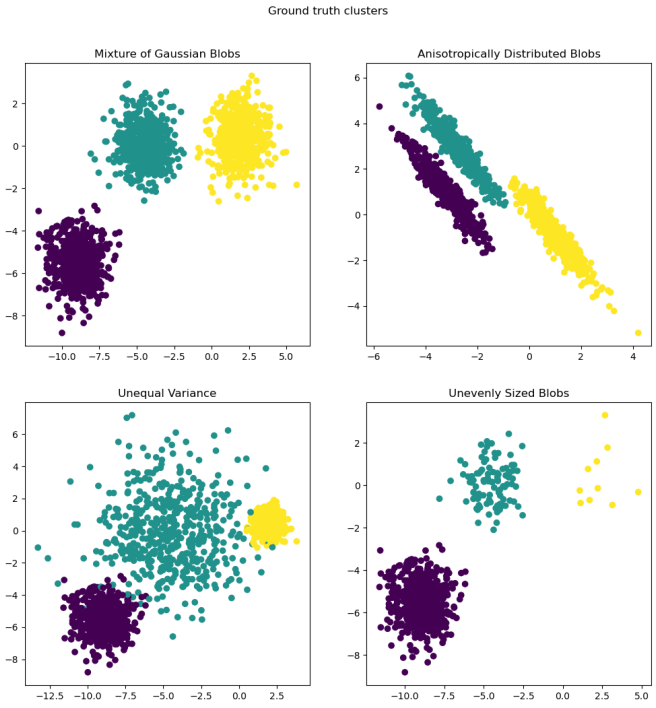
   Do tego celu można użyć dowolnej miary odległości, np. odległości euklidesowej lub odległości Manhattana. Odległość Manhattan, znana również jako metryka miejska lub city block distance, jest miarą odległości między dwoma punktami w przestrzeni euklidesowej. Jest ona obliczana jako suma wartości bezwzględnych różnic współrzędnych tych punktów. [7]

3. Algorytm oblicza nowe centroidy grup, jako średnią ważoną punktów danych przypisanych do każdej grupy.

   Nowe centroidy będą reprezentować nowe grupy, które powstały po przypisaniu punktów danych do centroidów.

4. Algorytm powtarza kroki 2 i 3, dopóki centroidy grup nie przestaną się zmieniać.

   Oznacza to, że żaden punkt danych nie zostanie ponownie przypisany do innej grupy.



Rysunek 3: K-Means, źródło [2]

2.4 Algorytm Optics

Algorytm OPTICS (Ordering Points To Identify the Clustering Structure)[8] to algorytm grupowania danych przestrzennych oparty na gęstości. Został opracowany przez Mihaela Ankersta, Markusa M. Breuniga, Hansa-Petera Kriegla i Jörga Sandera.

Algorytm OPTICS jest dość wydajnym, elastycznym i odpornym na szum algorytmem grupowania danych. Jego elastyczność pozwala na dostowanie wielu parametrów takich jak **max eps**, **cluster method** czy **min samples**. Niestety algorytm OPTICS może być podatny na ślepe zaułki, jeśli dane są gęste lub nieregularne.

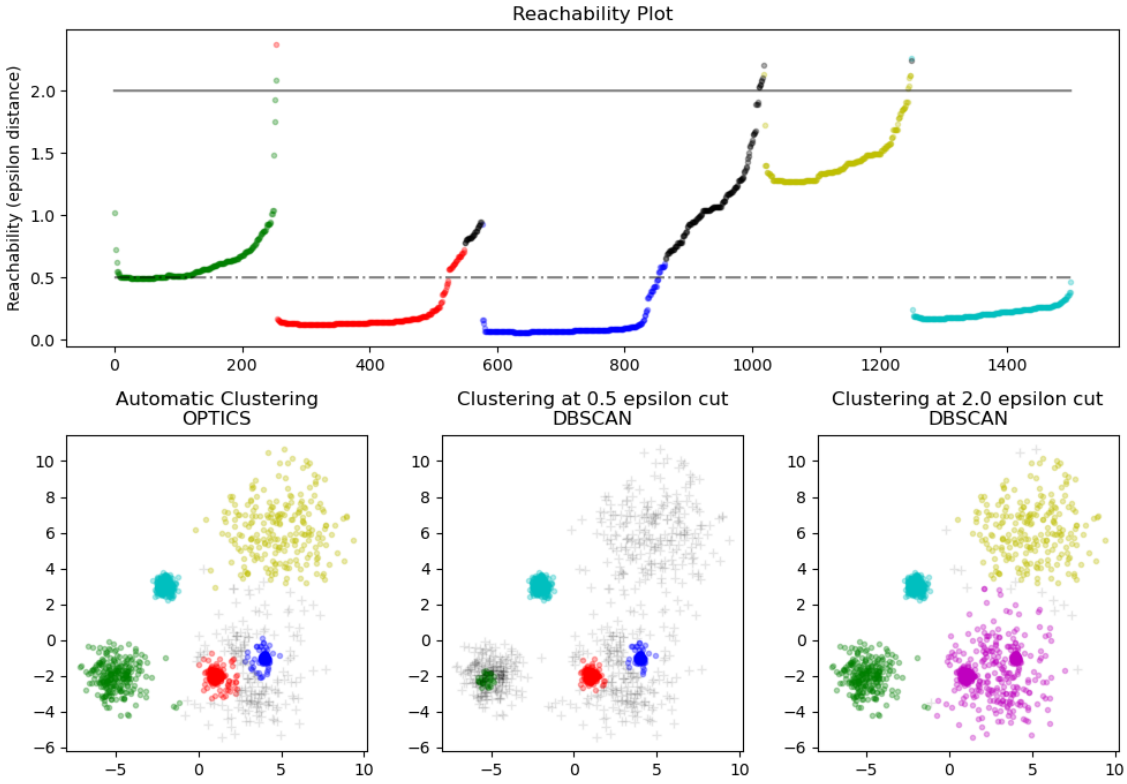
Ślepe zaułki w algorytmie OPTICS to sytuacje, w których algorytm nie jest w stanie znaleźć wszystkich grup danych. Dzieje się tak, gdy dane są gęste lub nieregularne, a parametr **max eps** jest zbyt mały. Przykładem

ślepego zaułka może być sytuacja, w której dane zawierają dwie grupy, które są rozdzielone wąską przerwą. Jeśli parametr **max eps** jest zbyt mały, to algorytm może uznać, że te dwie grupy to jedna grupa. Aby zmniejszyć ryzyko wystąpienia ślepych zaułków, należy ustawić parametr **max eps** na wartość większą. Jednak zbyt duża wartość parametru **max eps** może prowadzić do powstania zbyt wielu grup danych. OPTICS może stać się nieefektywny w przypadku bardzo dużych zbiorów danych, zwłaszcza jeśli wymagane jest przechowywanie informacji o wszystkich punktach w pamięci.

Algorytm ma wiele podobieństw do algorytmu DBSCAN i można go uznać za jego uogólnienie. Dodatkowo metoda ekstrakcji używana do wyodrębniania klastrów w OPTICS przy użyciu obliczonej osiągalności i uporządkowania umożliwia korzystanie z metody DBSCAN.

Główną różnicą między DBSCAN a OPTICS jest to, że algorytm OPTICS tworzy graf dostępności, który przypisuje każdej próbce zarówno odległość dostępności, jak i atrybut kolejności w klastrze. Te dwa atrybuty są przypisywane podczas dopasowania modelu i służą do określenia przynależności do klastra. Jeśli algorytm OPTICS jest uruchomiony z domyślną wartością **inf** ustawioną dla **max eps**, to można wielokrotnie wykonywać ekstrakcję klastrów w stylu DBSCAN w czasie liniowym dla dowolnej wartości eps przy użyciu metody **cluster optics dbscan**. Ustawienie **max eps** na niższą wartość skróci czas wykonania i można je traktować jako maksymalny promień sąsiedztwa od każdego punktu w celu znalezienia innych potencjalnie osiągalnych punktów. Rysunek nr 4 przedstawia przykład klasteryzacji (dolny plot) oraz graf dostępności (górny plot) przy uzyciu metody OPTICS.

Graf dostępności jest to reprezentacja wizualna porządku punktów danych według ich dostępności, która reprezentuje relacje sąsiedztwa między punktami w przestrzeni danych. Dostępność jest miarą, która odzwierciedla, jak blisko sąsiedztwa punktu znajduje się inny punkt danych. Im mniejsza odległość, tym większa dostępność.



Rysunek 4: Graf dostępności (górny plot) oraz efekt użycia algorytmu OPTICS (dolny plot), źródło [2]

2.4.1 Sposób działania Optics

Algorytm OPTICS działa w następujący sposób:

1. Obliczanie odległości do najbliższych sąsiadów

   Dla każdego punktu danych x obliczany jest jego promień gęstości r(x), który jest równy odległości do jego najdalszego sąsiada.

$$r(x) = \max_{y \in N(x)} \|x - y\| \tag{1}$$

   [8]

   - x - punkt danych
   - N(x) - zbiór najbliższych sąsiadów punktu x

- $\| - \|$ - odległość między dwoma punktami

2. Sortowanie punktów danych

   Punkty danych są sortowane według ich odległości do najbliższych sąsiadów. Punkty z większą odległością do najbliższych sąsiadów są sortowane na początku.

3. Tworzenie ścieżek gęstości

   Dla każdego punktu danych x tworzona jest ścieżka gęstości, która zawiera wszystkie punkty danych, które są gęściej położone niż dany punkt. Ścieżka gęstości jest tworzona w następujący sposób:

   - Punkt x jest dodawany do ścieżki gęstości.
   - Dla każdego sąsiada punktu x, który nie jest jeszcze w ścieżce gęstości, obliczany jest parametr $\epsilon(x, y)$, który jest równy różnicy między odległością do najbliższych sąsiadów punktu x a odległością do najbliższych sąsiadów punktu y.
   - Jeśli parametr $\epsilon(x, y)$ jest mniejszy niż parametr ϵ, to punkt y jest dodawany do ścieżki gęstości.

4. Grupowanie punktów danych

   Punkty danych są grupowane na podstawie ich ścieżek gęstości. Punkty danych, które znajdują się na tej samej ścieżce gęstości, są uważane za należące do tej samej grupy.

2.5 Algorytm afCEC

Aktywna funkcja klastrowania krzyżowo-entropijnego dzieli n-wymiarowe dane na klastry, znajdując parametry mieszanego uogólnionego wielowymiarowego rozkładu normalnego, który optymalnie przybliża rozproszenie danych w przestrzeni n-wymiarowej. Powyższe uogólnienie przeprowadza się poprzez wprowadzenie tzw. „gęstości Gaussa przystosowanych do f" (czyli zwykłych gęstości Gaussa adaptowanych przez „funkcję aktywną"). Dodatkowo aktywna funkcja grupowania między entropiami dokonuje automatycznej redukcji niepotrzebnych klastrów [9]. Więcej informacji można znaleźć w [10].

3 Metody oceny skuteczności klastrowania

Ocena wydajności algorytmu grupowania jest znacznie bardziej złożona niż zwykły pomiar liczby lub dokładności typowej dla algorytmów klasyfikacyjnych. Szczególnie ważne jest, aby miary oceny nie opierały się wyłącznie na wartościach bezwzględnych etykiet, ale na kwestii wiedzy, czy proces grupowania skutecznie oddziela dane tworząc zbiór podobieństw klas podstawowych lub spełniających założenia. Kluczowym aspektem jest wzięcie pod uwagę, że członkowie tego samego klastra powinni być bardziej do siebie podobni niż do członków różnych klastrów, co jest ustalone miarą podobieństwa.

Ważnym jest, aby metryki oceny algorytmów klastrowania brały pod uwagę struktury wewnętrzne klastrów i ich zdolność do wydobywania sensownych grup danych. Odpowiednie metryki powinny koncentrować się na mierzeniu spójności wewnętrznej klastrów oraz separacji między nimi, a nie na samej dokładności przypisania etykiet.

W wyniku czego ocena wydajności algorytmu klastrowania powinna uwzględniać kontekst specyficzny dla danego zadania i założeń dotyczących struktury danych. Kluczownym jest dążenie do wyważenia między poprawnością przypisanych klastrów a ich interpretowalnością dla rzetelnej oceny skuteczności algorytmu klastrowania.

3.1 Współczynnik Silhouette

Współczynnik Silhouette jest miarą jakości grupowania w analizie skupień. Służy do oceny, jak dobrze obserwacje w danym skupieniu są podobne do siebie, a jednocześnie jak różnią się od obserwacji w innych skupieniach. [11]

Współczynnik Silhouette jest obliczany dla każdej obserwacji w następujący sposób:

$$WspolczynnikSilhouette = \frac{b - a}{\max(a, b)} \tag{2}$$

[11]

gdzie:

- a to średnia odległość obserwacji do innych obserwacji w jej grupie
- b to średnia odległość obserwacji do najbliższej grupy, do której nie należy

Współczynnik Silhouette jest często używany do porównywania różnych metod grupowania. Wyższy współczynnik Silhouette oznacza, że metoda grupowania jest skuteczniejsza w identyfikowaniu naturalnych skupień w danych.

Współczynnik Silhouette ma kilka zalet. Jest łatwy do obliczenia i interpretacji. Jest również odporny na zmiany skali danych.

Jedną z wad współczynnika Silhouette jest to, że może być niestabilny dla skupień o nieregularnych kształtach. Ponadto współczynnik Silhouette może być zniekształcony, jeśli grupy mają różne rozmiary.

Współczynnik Silhouette to przydatne narzędzie do oceny jakości grupowania. Należy jednak pamiętać o jego ograniczeniach i stosować go z ostrożnością.

3.2 Współczynnik Calinski-Harabasz

Indeks Calinski-Harabasza (CHI) to miara oceny algorytmów grupowania. Jest on również znany jako Kryterium Współczynnika Wariancji (VRC). Indeks CHI mierzy stosunek rozproszenia międzygrupowego do rozproszenia wewnątrzgrupowego. Wyższa wartość CHI wskazuje na lepszą wydajność grupowania, ponieważ sugeruje, że grupy są dobrze odseparowane, a obserwacje w każdej grupie są do siebie podobne. [12]

Indeks CHI jest obliczany w następujący sposób:

$$CHI = \frac{B^2}{W^2} \cdot \frac{N}{K} \tag{3}$$

[12]

gdzie:

- $\mathbf{B^2}$ to suma kwadratów rozproszenia międzygrupowego. Suma kwadratów rozproszenia międzygrupowego jest miarą zmienności między grupami. Jest obliczana przez sumowanie kwadratów odległości między środkami grup.
- $\mathbf{W^2}$ to suma kwadratów rozproszenia wewnątrzgrupowego. Suma kwadratów rozproszenia wewnątrzgrupowego jest miarą zmienności wewnątrz grup. Jest obliczana przez sumowanie kwadratów odległości między obserwacjami w każdej grupie a środkiem ich grupy.

- N to całkowita liczba obserwacji
- K to liczba grup

Indeks CHI jest powszechnie uważany za bardziej odporną miarę wydajności grupowania niż inne miary, takie jak współczynnik Silhouette, z uwagi, że jest on mniej wrażliwy na wartości odstające i zmiany skali. Należy jednak pamiętać o jego ograniczeniach i stosować go w połączeniu z innymi miarami przy podejmowaniu decyzji dotyczących algorytmów grupowania.

3.3 Indeks Davies-Bouldin

Indeks Daviesa-Bouldina (DBI) to miara jakości grupowania, która jest obliczana dla każdej grupy w zbiorze danych. Indeks DBI jest miarą podobieństwa każdej grupy do jej najbliższych sąsiadów. Mniejsze wartości indeksu DBI wskazują na lepsze grupowanie, ponieważ sugerują, że grupy są bardziej odseparowane od siebie. [13]

Indeks Daviesa-Bouldina jest obliczany w następujący sposób:

$$DBI = \frac{1}{K} \sum_{k=1}^{K} \frac{\overline{d_k}}{\min_{l \neq k} \overline{d_{kl}}} \tag{4}$$

[13]

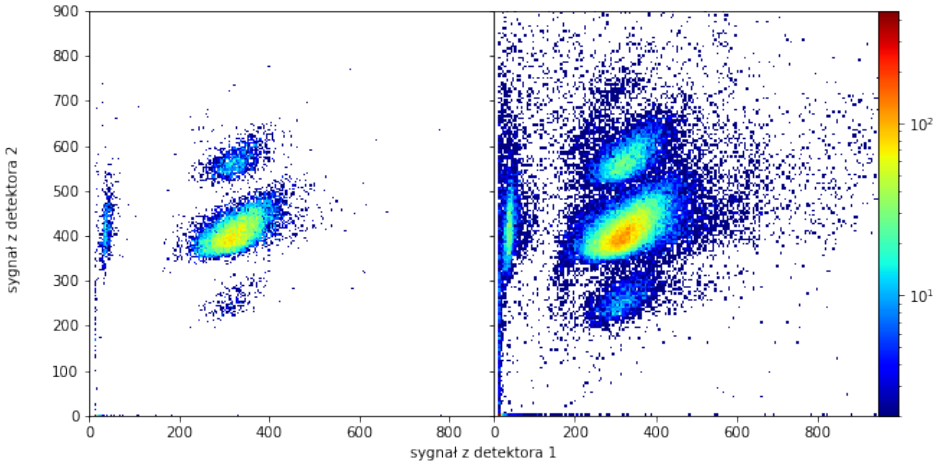
gdzie:

- K to liczba grup
- $\overline{d_k}$ to średnia odległość obserwacji w grupie k do jej środka
- $\overline{d_{kl}}$ to średnia odległość między grupami k i l

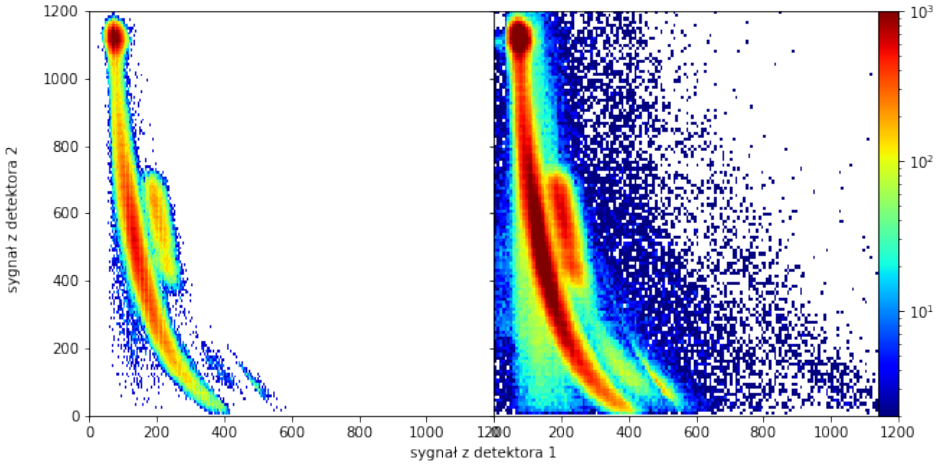
Indeks Daviesa-Bouldina ma kilka zalet. Jest to prosta miara, która jest łatwa do obliczenia i interpretacji. Jest również odporna na zmiany skali danych. Indeks Daviesa-Bouldina jest często wykorzystywany do porównywania różnych metod grupowania. Wyższy indeks DBI dla danej metody grupowania wskazuje, że metoda ta jest bardziej skuteczna w identyfikowaniu naturalnych skupień w danych.

4 Opis danych

Do testowania algorytmów skupień otrzymałem cztery zestawy danych w formacie pliku csv (coma sepatated values) [14] . Dane pochodziły z dwóch eksperymentów fizyki jądrowej i cząstek. Każda para danych składała się z danych bardziej lub mniej zaszumianych. Na Rysunku 5 (5a - lewy panel) widac dane z mniejszą ilością szumu, w przeciwieństwie do panelu prawego czyli 5b, na którym widać dużą ilość szumu. Podobnie na lewym i prawym panelu Rysunku 6



Rysunek 5: Dane z pierwszego eksperymentu. Lewy panel 5a: dane z małą ilością szumu, prawy panel 5b: dane z dużą ilością szumu.



Rysunek 6: Dane z drugiego eksperymentu. Lewy panel 6a: dane z małą ilością szumu, prawy panel 6b: dane z dużą ilością szumu.

Na rysunkach można zauważyć charakterystyczne skupiska danych o różnej gęstości. Celem algorytmów będzie ich skuteczyne wyselekcjonowanie jako obszary o zwiększonej gęstości. W ramach konsultacji z zespołem fizyków ustalone zostało, że wykorzystane algorytmy powinny odizolować dane dla zbiorów przedstawionych na Rysunku 6 na 2 obszary.

5 Wyniki Badań

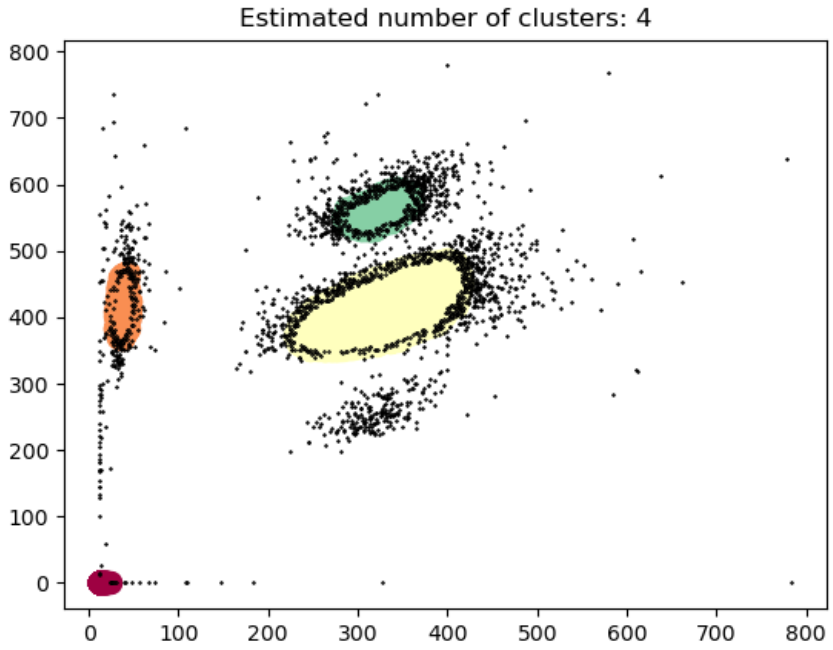
Badania zostały przeprowadzone z użyciem języka Python oraz R dla pakietu afCEC, na lokalnej platformie jupyter server, Rstudio oraz zdalnej platformie jupyter(serwer uczelniany). Większość użytych modułów do przeprowadzenia badań pochodzi z biblioteki SciKitLearn, z wyjątkiem pakietu afCEC, który zapożyczono z bibilioteki Cran-R [15], natomiast gotowy program w języku R został napisany i użyczony przez grupę naukowców z Uniwersytetu Jagiellońskiego.

Na większości algorytmów przeprowadzano dostrajanie hiperparametrów, aby osiągnąć oczekiwane efekty. Dodatkowo, użyto metryk: Davies-Bouldin Index, Calinski-Harabasz Index oraz Silhouette Coefficient, w celu sprawdzenia jakości analizy skupień przeprowadzonych przy pomocy różnych algorytmów oraz ich porównania. Metryki te świetnie się sprawdzają w sytuacji, w której nie znamy prawdziwych etykiet (oczekiwanych centr klastrów).

Po przeprowadzniu wstępnych testów Silhouette Coefficient zwracał w znacznym stopniu niezadowalające rezultaty, dlatego na potrzeby dostrajania hiperparametrów zdecydowano o naprzemiennym wykorzystaniu Davies-Bouldin Index oraz Calinski-Harabasz Index.

Złożoność algorytmów oraz ograniczenia sprzętowe pozwoliły na niewielkie dostrojenia, jednakże dostrojenia te znacznie skróciły czas poszukiwania odpowiednich parametrów. Największym powodzeniem okazało się dostrajanie parametrów dla zestawu danych 3N z użyciem DBSCAN.

5.1 Wyniki badań dla DBSCAN

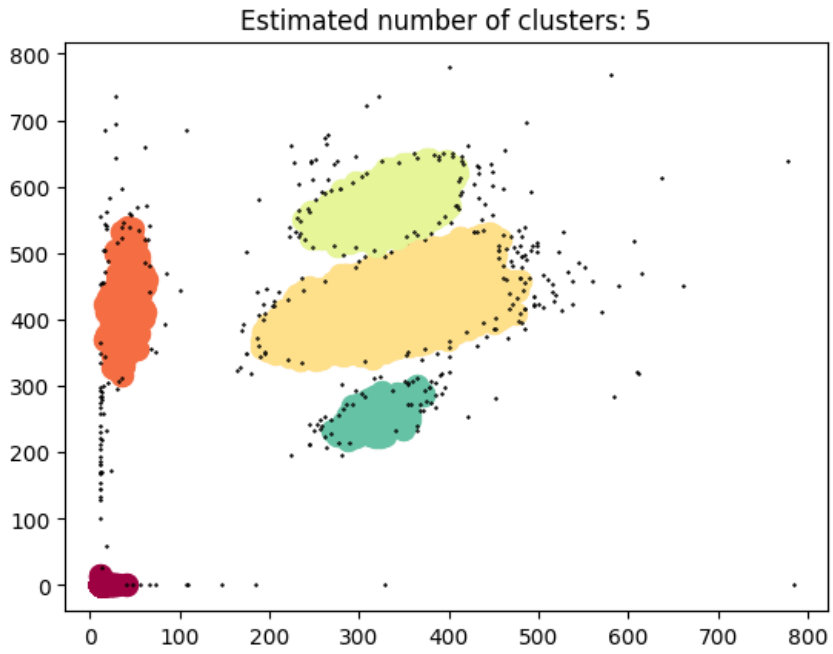


Rysunek 7

Do obliczeń na danych z rysunku 5a algorytm użył wybranych przeze mnie wartości **eps** = 10 oraz **min samples** = 100. Na rysunku 7 można zauważyć, że DBSCAN odizolował dane na 5 klastrów. Kwestią sporną może pozostawać kawałek na początku wykresu (bordowy klaster), czy powinien zostać wykryty jako osobny klaster, czy jako punkty szumu. Dodatkowo DBSCAN odseparował punkty szumu jako czarne punkty w ilości 2880.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.54
- Davies-Bouldin Index: 2.30
- Calinski-Harabasz Index: 13177.02



Rysunek 8

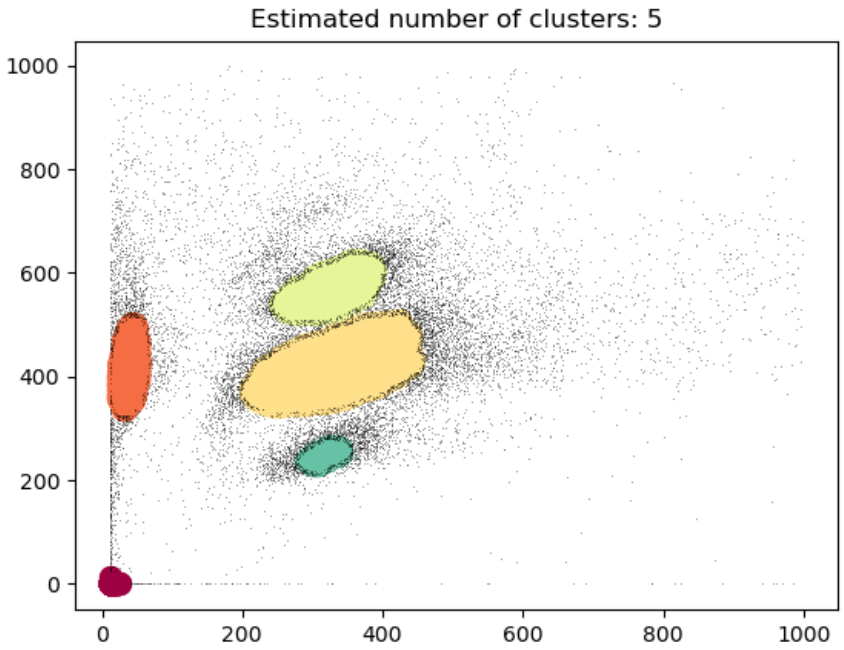
Następnie dla tych samych danych z rysunku 5a użyto tunera hiperparametrów. Do dostosowania hiperparametrów zastosowano obserwację wzrokową jak zachowuje się algorytm z użyciem tunera GridSearchCV. Dostrajanie hiperparametrów metodą GridSearchCV polega na automatycznym przeszukaniu zadanej przestrzeni danych z użyciem walidacji krzyżowej [16] w celu znalezienia najlepszego zestawu hiperparametrów na podstawie zadanych metryk. Do algorytmu DBSCAN użyto następującej przestrzeni hiperparametrów:

```
param_grid = {eps: [10, 12, 14, 16, 18, 20],
              min_samples: [10, 11, 12, 13, 14, 15,
              16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
              26, 27, 28, 29, 30]}
```

Najlepszym modelem okazał się ten z hiperparametrami: eps = 16 oraz min samples = 22, który cechował się zadowalającymi wynikami z metryk oraz poprawnym odwzorowaniem wizualnym klastrów. Na rysunku 8 można zauważyć, że DBSCAN odizolował dane na 5 klastrów. Dodatkowo DBSCAN odseparował punkty szumu jako czarne punkty w ilości 412.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.57
- Davies-Bouldin Index: 3.04
- Calinski-Harabasz Index: 18644.29

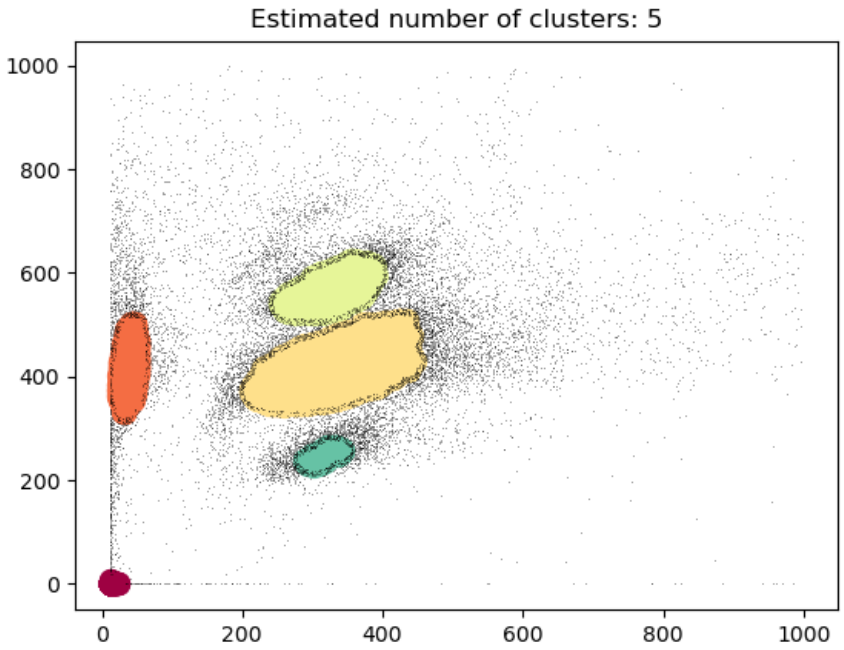


Rysunek 9

W kolejnym kroku analizie skupień poddano dane bardziej zaszumiane z rysunku 5b. Do obliczeń algorytm użył użył wybranych przeze mnie wartości **eps** = 10 oraz **min samples** = 100. Na rysunku 9 można zauważyć, że DBSCAN podzielił dane na 5 klastrów. Kwestią sporną może pozostawać kawałek na początku wykresu, czy algorytm też powinien go wykrywać jako osobny klaster. Dodatkowo DBSCAN odseparował całkiem sporą ilość punktow jako punkty szumu w ilości 13225, dzięki czemu może pełnić dodatkową rolę odszumiacza danych.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.30
- Davies-Bouldin Index: 2.42
- Calinski-Harabasz Index: 6383.24



Rysunek 10

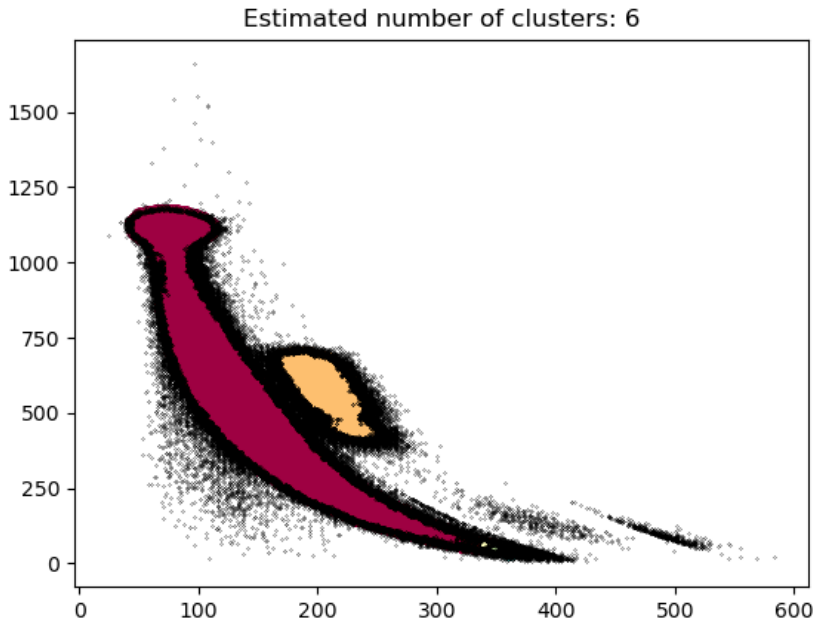
Następnie dla tych samych danych z rysunku 5b użyto tunera hiperparametrów. Do dostosowania hiperparametrów zastosowano obserwację wzrokową jak zachowuje się algorytm z użyciem tunera GridSearchCV. Do algorytmu DBSCAN użyto następującej przestrzeni hiperparametrów:

```
param_grid = {eps: [10, 11, 12, 13, 14, 15],
              min_samples: [70, 80, 90, 100,
              110, 120, 130, 140]}
```

Najlepszym modelem okazał się ten z hiperparametrami: eps = 12 oraz min samples = 140, który cechował się zadowalającymi wynikami z metryk oraz poprawnym odwzorowaniem wizualnym klastrów. Na rysunku 8 można zauważyć, że DBSCAN odizolował dane na 5 klastrów. Dodatkowo DBSCAN odseparował punkty szumu jako czarne punkty w ilości 12838.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.31
- Davies-Bouldin Index: 2.42
- Calinski-Harabasz Index: 6594.01

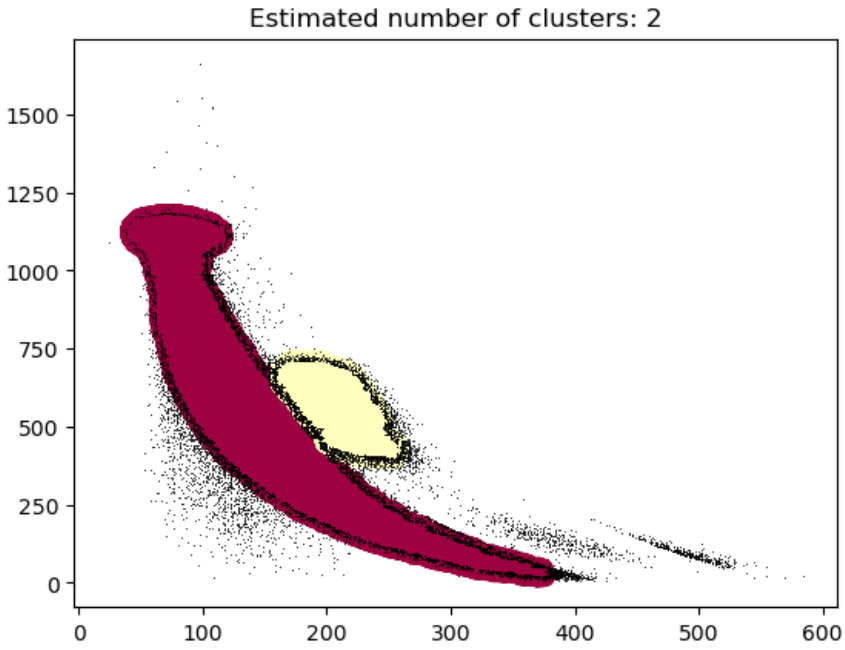


Rysunek 11

W kolejnym kroku analizie skupień poddano dane z rysunku 6a. Do obliczeń algorytm użył użył wybranych przeze mnie wartości **eps** = 3 oraz **min samples** = 100. Na rysunku 11 można zauważyć, że DBSCAN odizolował dane na 6 klastrów. Dodatkowo DBSCAN odseparował punkty szumu jako czarne punkty w ilości 61249.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: -0.40
- Davies-Bouldin Index: 3.10
- Calinski-Harabasz Index: 5171.80



Rysunek 12

Następnie dla tych samych danych z rysunku 5b użyto tunera hiperparametrów. Do dostosowania hiperparametrów zastosowano obserwację wzrokową jak zachowuje się algorytm z użyciem tunera GridSearchCV. Do algorytmu DBSCAN użyto następującej przestrzeni hiperparametrów:

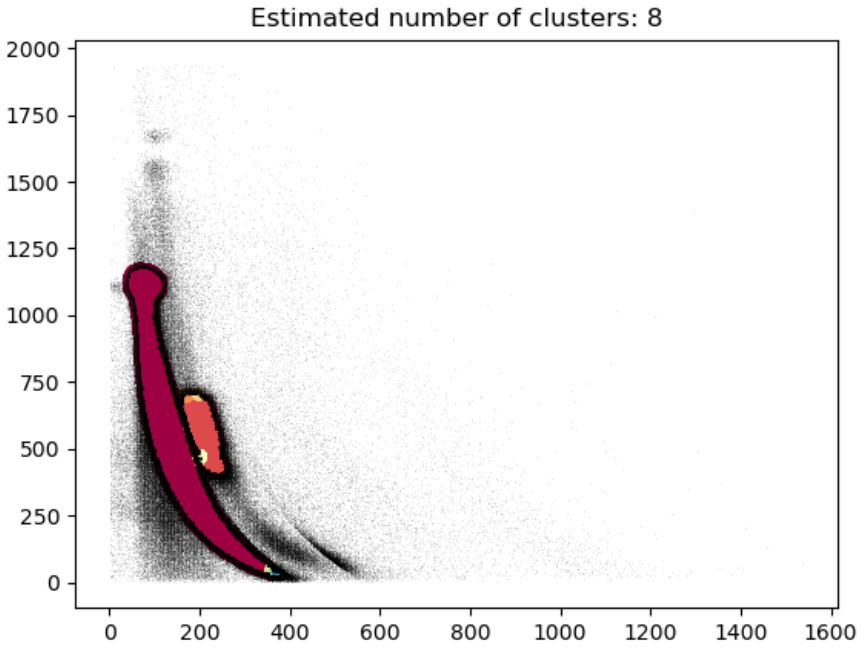
```
param_grid = {eps: [4.1, 4.2, 4.3, 4.4,
              4.5, 4.6, 4.7, 4.8, 4.9, 5],
              min_samples: [70, 80, 90, 100,
              110, 120, 130, 140]}
```

Najlepszym modelem okazał się ten z hiperparametrami: eps = 4.3 oraz min samples = 187, który cechował się zadowalającymi wynikami z metryk oraz poprawnym odwzorowaniem wizualnym klastrów. Na rysunku 12 można

zauważyć, że DBSCAN odizolował dane na 2 klastry. Dodatkowo DBSCAN odseparował punkty szumu jako czarne punkty w ilości 32390.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: -0.23
- Davies-Bouldin Index: 2.19
- Calinski-Harabasz Index: 7405.09

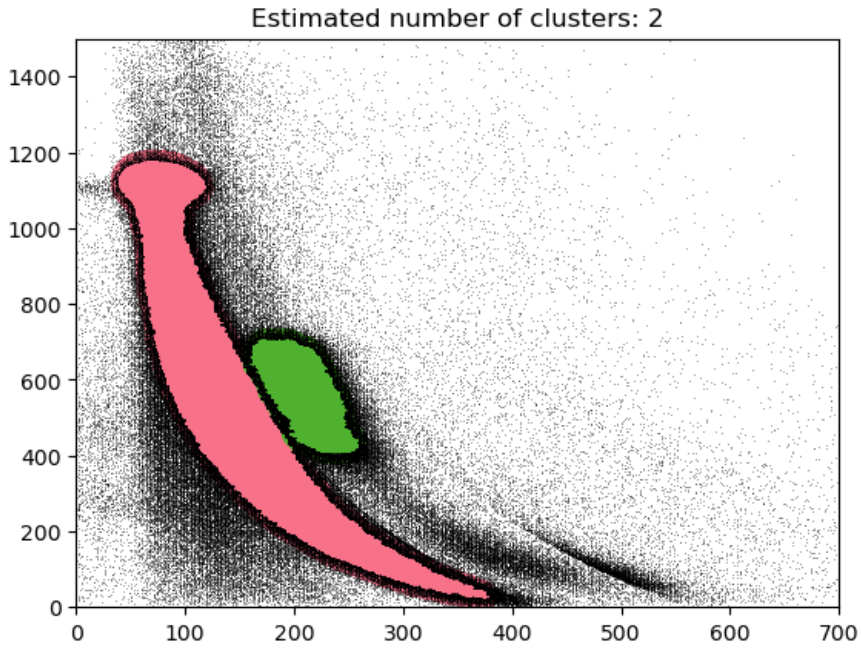


Rysunek 13

W kolejnym kroku analizie skupień poddano dane bardziej zaszumiane z rysunku 5b. Do obliczeń algorytm użył wybranych przeze mnie wartości **eps** = 3 oraz **min samples** = 100. Na rysunku 13 można zauważyć, że DBSCAN odizolował dane na 8 klastrów, pomimo zwiększosnej ilości punktów (potencjalny szum) względem rysunku 11 . Dodatkowo DBSCAN odseparował całkiem sporą ilość punktow jako punkty szumu w ilości 158238.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: -0.53
- Davies-Bouldin Index: 5.89
- Calinski-Harabasz Index: 5576.93



Rysunek 14

Następnie dla tych samych danych z rysunku 5a użyto tunera hiperparametrów. Do dostosowania hiperparametrów zastosowano obserwację wzrokową jak zachowuje się algorytm z użyciem tunera GridSearchCV. Do algorytmu DBSCAN użyto następującej przestrzeni hiperparametrów:

```
param_grid = {eps: [4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5],
              min_samples: [70, 80, 90, 100,
              110, 120, 130, 140]}
```

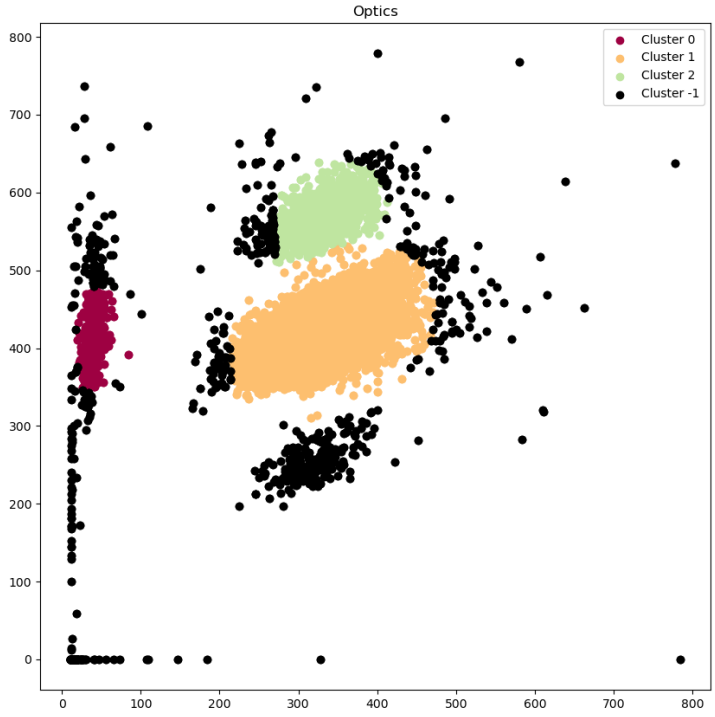
Najlepszym modelem okazał się ten z hiperparametrami: eps = 4.3 oraz min samples = 187, który cechował się zadowalającymi wynikami z metryk oraz poprawnym odwzorowaniem wizualnym klastrów. Na rysunku 14 można

zauważyć, że DBSCAN odizolował dane na 2 klastry. Dodatkowo DBSCAN odseparował punkty szumu jako czarne punkty w ilości 136929.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: -0.18
- Davies-Bouldin Index: 3.76
- Calinski-Harabasz Index: 20832.51

5.2 Wyniki badań dla Optics

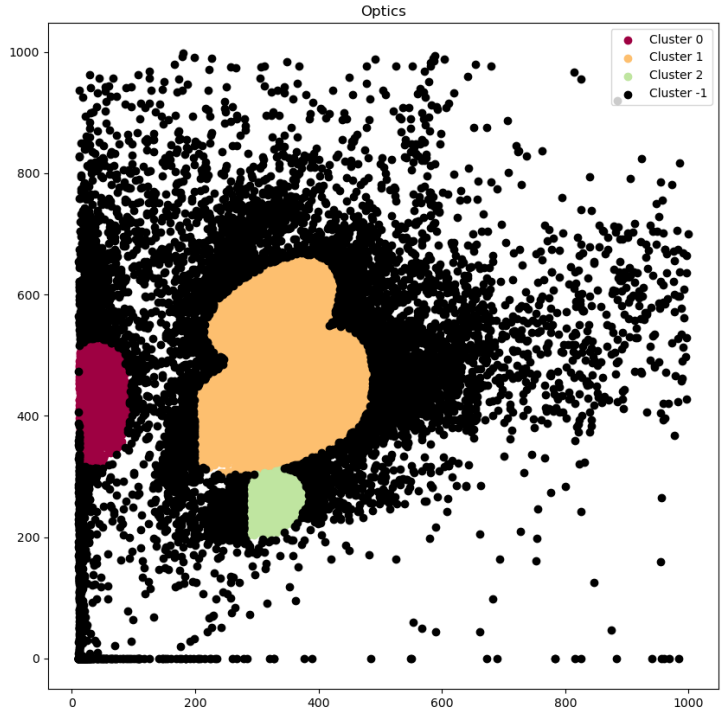


Rysunek 15

Do obliczeń na danych z rysunku 5a algorytm użył wybranych przeze mnie wartości **max eps** = 37 oraz **min samples** = 650. Czas wykonywania algorytmu nie pozwolił na zastosowanie tunera GridSearchCV. Na rysunku 15 można zauważyć, że OPTICS odizolował dane na 4 klastry. Dodatkowo OPTICS odseparował punkty szumu jako czarne punkty w ilości 1817.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.58
- Davies-Bouldin Index: 1.41
- Calinski-Harabasz Index: 10388.22



Rysunek 16

Do obliczeń na danych z rysunku 5b algorytm użył wybranych przeze mnie wartości **max eps** = 30 oraz **min samples** = 600. Czas wykonywania algorytmu nie pozwolił na zastosowanie tunera GridSearchCV. Na rysunku 15 można zauważyć, że OPTICS odizolował dane na 4 klastry. Dodatkowo OPTICS odseparował punkty szumu jako czarne punkty w ilości 8401.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.33
- Davies-Bouldin Index: 9.90
- Calinski-Harabasz Index: 5195.89

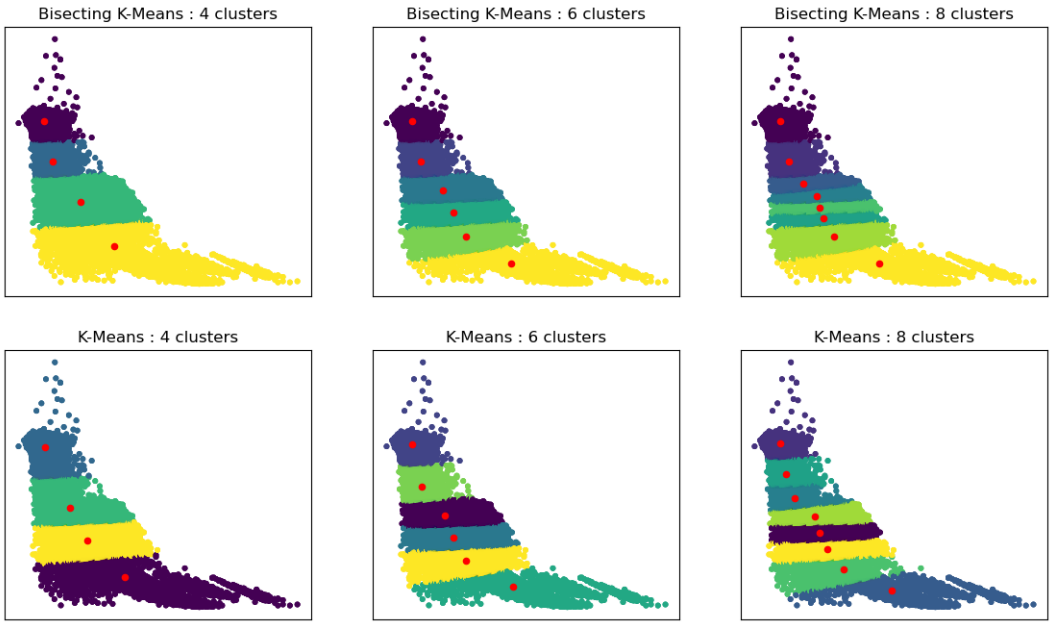
5.3 Wyniki badań dla KMeans

Rysunek 17: Wyniki KMeans z zestawu 5a dla różnej ilości klastrów

Dla zestawu danych z rysunku 5a zastosowano domyślną wartość **n init** oraz przedstawiono zachowanie algorytmu dla różnych wartości **n clusters**. Próba dostrojenia parametrów algorytmu nie przysniosła zadowalających efektów, co można zauważyć na rysunku 17. Algorytm w nieprawidłowy sposób zdefiniował klastry. Fakt, że mamy do czyniena z klastrami niepłaskimi sprawia, że algorym nie rozponaje klastrów w przewidziany przez zespół fizyków sposób. W Tabeli 1 zostały zaprezentowane wyniki metryk dla poszczególnych algorytmów oraz ich hiperparametrów

Algorithm	Number of Clusters	Silhouette	Davies-Bouldin	Calinski-Harabasz
Bisecting K-Means	4	0.62	0.40	34252.51
Bisecting K-Means	6	0.37	0.69	46330.21
Bisecting K-Means	8	0.38	0.66	47754.75
K-Means	4	0.58	0.52	36359.31
K-Means	6	0.41	0.64	51160.11
K-Means	8	0.39	0.66	51049.98

Tabela 1



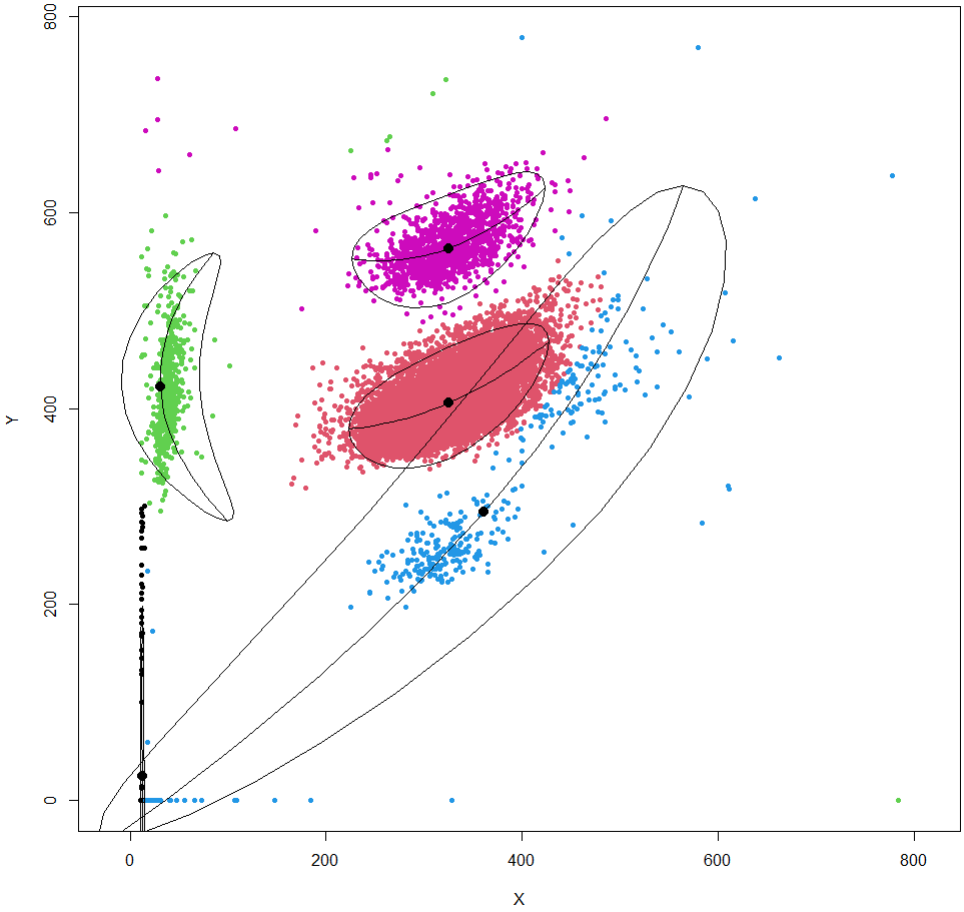
Rysunek 18: Wyniki KMeans z zestawu 6a dla różnej ilości klastrów

Dla zestawu danych z rysunku 6a zastosowano domyślną wartość **n init** oraz przedstawiono zachowanie algorytmu dla różnych wartości **n clusters**. Próba dostrojenia parametrów algorytmu nie przysniosła zadowalających efektów, co można zauważyć na rysunku 18. Algorytm KMeans w nieprawidłowy sposób zdefiniował klastry. Algorytm Kmeans skategoryzował klastry jako płaszczyzny oddzielone linią w sposób horyzontalny (reprezentacja w formie płaskiej), gdzie spodziewanym rezultatem było zgrupowanie danych w skupiska o kształcie zbliżonym do koła lub elipsy w przestrzeni danych. W Tabeli 2 zostały zaprezentowane wyniki metryk dla poszczególnych algorytmów oraz ich hiperparametrów.

Algorithm	Number of Clusters	Silhouette	Davies-Bouldin	Calinski-Harabasz
Bisecting K-Means	4	0.54	0.52	2578174.34
Bisecting K-Means	6	0.52	0.60	3929552.32
Bisecting K-Means	8	0.47	0.78	3445750.06
K-Means	4	0.55	0.57	3201071.61
K-Means	6	0.52	0.60	3938435.25
K-Means	8	0.49	0.67	4181931.24

Tabela 2

5.4 Wyniki badań dla afCEC

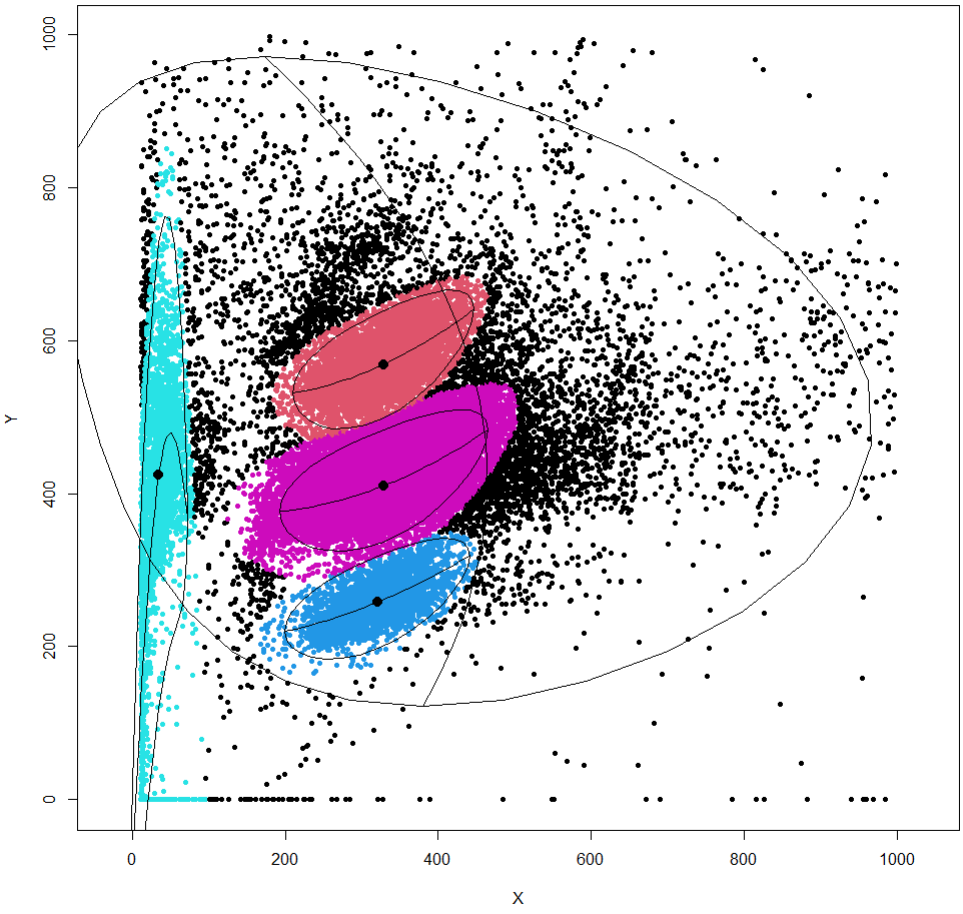


Rysunek 19

Do obliczeń na danych z rysunku 5a algorytm użył wybranych przeze mnie wartości **maxClusters** = 6, reszta wartości pozostała domyślna. Na rysunku 19 można zauważyć, że algorytm afCEC odizolował dane na 5 klastrów. Kwestią sporną może pozostawać rozszerzona przestrzeń niebieskiego klastra, która zakrawa o przestrzeń klastra czerwonego. Natomiast z uwagi na ograniczone zasoby sprzętowe oraz technologiczne pozostawiono wyniki bez dostrajania hiperparametrów.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.61
- Davies-Bouldin Index: 0.97
- Calinski-Harabasz Index: 22335.66



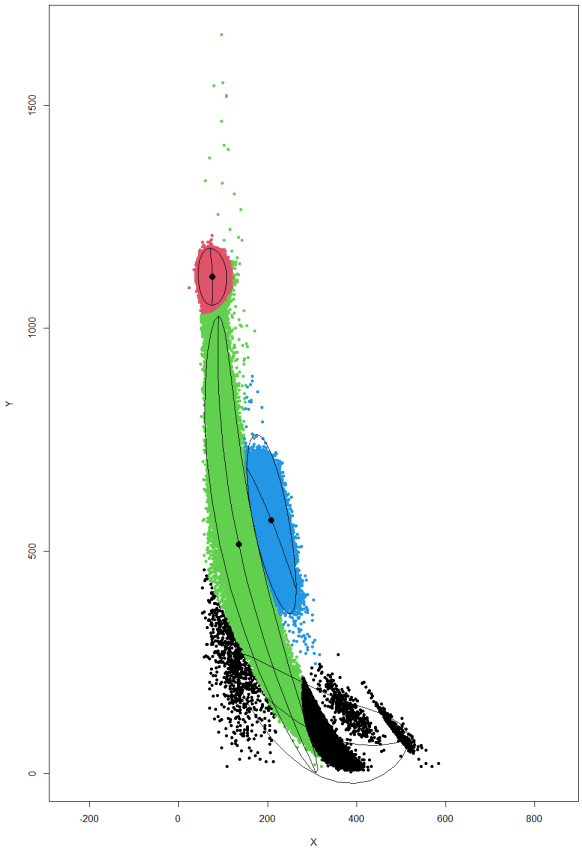
Rysunek 20

W kolejnym kroku analizie skupień poddano dane bardziej zaszumiane z rysunku 5b. Do obliczeń na danych z rysunku 5a algorytm użył wybranych przeze mnie wartości **maxClusters** = 4, reszta wartości pozostała domyślna. Na rysunku 20 można zauważyć, że algorytm afCEC w perfekcyjny sposób odizolował dane na 3 klastry (tj. różowy, niebieski oraz czerowny klaster),

dodatkowo afCEC odzizolował dane zaszumione (szum) od klastrów właściwych w oczecikawny sposób. Kwestią sporną może pozostawać turkosowy klaster, jego granica zachaczyła również o skupisko danych przy punkcie 0.0. Natomiast z uwagi na ograniczone zasoby sprzętowe oraz technologiczne pozostawiono wyniki bez dostrajania hiperparametrów.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.39
- Davies-Bouldin Index: 2.90
- Calinski-Harabasz Index: 12784.52

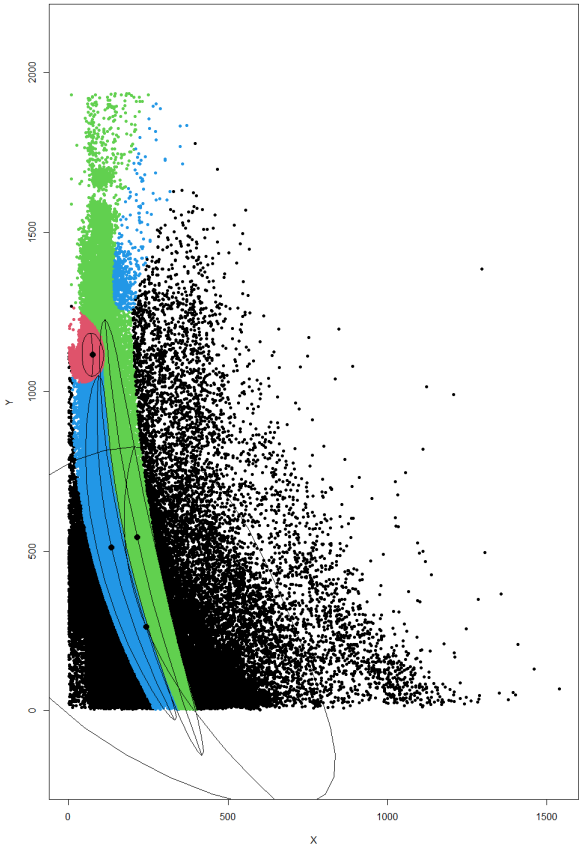


Rysunek 21

Do obliczeń na danych z rysunku 6a algorytm użył wybranych przeze mnie wartości **maxClusters** = 3, reszta wartości pozostała domyślna. Na rysunku 21 można zauważyć, że algorytm afCEC odizolował dane na 3 klastry (tj. zielony, niebieski oraz czerowny klaster), dodatkowo afCEC odzizolował dane zaszumione (szum). Kwestią sporną może pozostawać czerowny klaster klaster. Zgodnie z ustaleniami dane te powinny zostać zakwalfikowane jako zielony klaster. Natomiast z uwagi na ograniczone zasoby sprzętowe oraz technologiczne pozostawiono wyniki bez dostrajania hiperparametrów.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.07
- Davies-Bouldin Index: 1.75
- Calinski-Harabasz Index: 494379.14



Rysunek 22

W kolejnym kroku analizie skupień poddano dane bardziej zaszumiane z rysunku 6b. Do obliczeń algorytm użył wartości **maxClusters** = 4, reszta wartości pozostała domyślna. Na rysunku 22 można zauważyć, że algorytm afCEC lekko się pogubił w podczas kwalfikacji danych. Z uwagi na znaczną ilośc danych zaszumionych algorytm nieskutecznie zakwalfikował dane na poszczególne klastry. Algorytmowi udało się wyodrębnić sporą część danych jako dane zaszumione.

Poniżej przedstawione zostały wyniki metryk:

- Silhouette Score: 0.04
- Davies-Bouldin Index: 2.43
- Calinski-Harabasz Index: 314986.43

6 Podsumowanie

Głównym celem pracy było znalezienie optymalnych algorytmów klasteryzacji dla poszczególnych zestawów danych, który został osiągnięty poprzez analizę dostępnych rozwiązań. Przeprowadzone analizy pokazały skuteczność zaproponowanych rozwiązań oraz ich przydatność w praktyce.

W ramach pracy skupiono się na wytrenowaniu modeli z użyciem różnych hiperparametrów oraz analizie wykresów. Wyniki uzyskane podczas badań zostały przeanalizowane w celu dostosowania wykresów do zamierzonych rezultatów. Kolejnym krokiem było dostrojenie hiperparametrów z użyciem tunera GridSearchCV oraz ponowna analiza wykresów z użyciem najlepszych parametrów wskazanyc przez tuner.

Po przeanalizowaniu wszystkich wyników najlepszym algorytmem okazał się algorytm DBSCAN. Charakteryzował się on stosunkowo krótkim czasem wykonywania obliczeń co pozwoliło na lepsze dostrojenie hiperparametrów. Wyniki na wykresach były zbliżone wynikom oczekiwanym przez zespół fizyków.

Po przeanalizowaniu wyników z użyciem algorytmów OPTICS oraz MeanShiht, doszedłem do wniosku, że są one nieodpowiednie dla otrzymanych zestawów danych. Ich złożoność, która przekładała się czas trwania wykonywania algorytmu nie pozwalały na dostrojenie hiperparametrów. Ręczna próba znalezienia oczekiwanych wartości również zakończyła się niepowodzeniem.

Na wyróżnienie zasługuje algorytm afCEC. Charakteryzował się on wynikami mocno zbliżonymi do oczekiwanych przez zespól fizyków. Pomimo długiego czasu oczekiwania na wyniki poradził sobie wzrocowo z zaszumionymi danymi z rysunku 5b. Niestety z uwagi na ograniczone zasoby technologiczne pozostawiono wyniki bez dostrajania hiperparametrów.

Podsumowując sama analiza pod względem wyników metryk czy wyników na wykresach to za mało. Podczas analizy doszedłem do wniosku, że jedynie połączenie analizy wykresów wraz z konsultacją oczekiwanych rezultatów oraz analiza pod względem metryk dają podstawy do wybrania najkorzystniejszych algorytmów. Natomiast warto mieć na uwadzę, że nie każdy algorytm nadaję się do każdego zestawu danych, dlatego dzięki analizie można wybrać najlepsze algorytmu dla konkretnych zestawów danych. W Tabeli 3 przedstawiono zestawienie uzyskanych wyników dla poszczególnych

algorytmów, zestawów danych oraz hiperparametrów.

Nazwa	Nr. Rysunku	Hiperparametry	Silhouette	Davies-Bouldin	Calinski-Harabasz
DBSCAN	5a	eps = 10 min samples = 100	0.54	2.30	13177.02
DBSCAN	5a	eps = 16 min samples = 22	0.57	3.04	18664.29
DBSCAN	5b	eps = 10 min samples = 100	0.30	2.42	6383.24
DBSCAN	5b	eps = 12 min samples = 140	0.31	2.42	6594.01
DBSCAN	6a	eps = 3 min samples = 100	-0.40	3.10	5171.80
DBSCAN	6a	eps = 4.3 min samples = 187	-0.23	2.19	7405.09
DBSCAN	6b	eps = 3 min samples = 100	-0.53	5.89	5576.93
DBSCAN	6b	eps = 4.3 min samples = 187	-0.18	3.76	20832.51
Optics	5a	max eps = 37 min samples = 650	0.58	1.41	10388.22
Optics	5b	max eps = 30 min samples = 650	0.33	9.90	5195.89
Bisecting K-Means	5a	num of clusters = 4	0.62	0.40	34252.51
Bisecting K-Means	5a	num of clusters = 6	0.37	0.69	46330.21
Bisecting K-Means	5a	num of clusters = 8	0.38	0.66	47754.75
K-Means	5a	num of clusters = 4	0.58	0.52	36359.31
K-Means	5a	num of clusters = 6	0.41	0.64	51160.11
K-Means	5a	num of clusters = 8	0.39	0.66	51049.98
Bisecting K-Means	6a	num of clusters = 4	0.54	0.52	2578174.34
Bisecting K-Means	6a	num of clusters = 6	0.52	0.60	3929552.32
Bisecting K-Means	6a	num of clusters = 8	0.47	0.78	3445750.06
K-Means	6a	num of clusters = 4	0.55	0.57	3201071.61
K-Means	6a	num of clusters = 6	0.52	0.60	3938435.25
K-Means	6a	num of clusters = 8	0.49	0.67	4181931.24
afCEC	5a	maxClusters = 6	0.61	0.97	22335.66
afCEC	5b	maxClusters = 4	0.39	2.90	12784.52
afCEC	6a	maxClusters = 4	0.07	1.75	94379.14
afCEC	6b	maxClusters = 4	0.04	2.43	314986.43

Tabela 3: Zestawienie wszystkich wyników

Bibliografia

[1] K. Sawka, *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*, II. Gliwice: Helion, 2020, Aktualizacja do modułu TensorFlow 2, ISBN: 978-83-283-6002-0.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort i in., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, t. 12, s. 2825–2830, 2011. adr.: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

[3] D. Comaniciu i P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 24, nr. 5, s. 603–619, 2002. DOI: 10.1109/34.1000236.

[4] E. Schubert, J. Sander, M. Ester, H. P. Kriegel i X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," *ACM Trans. Database Syst.*, t. 42, nr. 3, lip. 2017, ISSN: 0362-5915. DOI: 10.1145/3068335. adr.: https://doi.org/10.1145/3068335.

[5] J. A. Hartigan i M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, t. 28, nr. 1, s. 100–108, 1979, ISSN: 00359254, 14679876. adr.: http://www.jstor.org/stable/2346830 (term. wiz. 18. 03. 2024).

[6] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, t. 28, nr. 2, s. 129–137, 1982. DOI: 10.1109/TIT.1982.1056489.

[7] E. Krause, *Taxicab Geometry: An Adventure in Non-Euclidean Geometry* (Dover Books on Mathematics Series). Dover Publications, 1986, ISBN: 9780486252025. adr.: https://books.google.pl/books?id=IW7ICV0QXWwC.

[8] M. Ankerst, M. M. Breunig, H.-P. Kriegel i J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," w *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '99, Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1999, s. 49–60, ISBN: 1581130848. DOI: 10.1145/304182.304187. adr.: https://dl.acm.org/doi/10.1145/304181.304187.

[9] P. S. Krzysztof Byrski, *afCEC: Active Function Cross-Entropy Clustering*, Accessed: March 13, 2024. adr.: https://cran.r-hub.io/web/packages/afCEC/README.html.

[10] P. Spurek, J. Tabor i K. Byrski, "Active function cross-entropy clustering," *Expert Systems with Applications*, t. 72, s. 49–66, 2017.

[11] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, t. 20, s. 53–65, 1987, ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. adr.: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[12] T. Caliński i H. JA, "A Dendrite Method for Cluster Analysis," *Communications in Statistics - Theory and Methods*, t. 3, s. 1–27, sty. 1974. DOI: 10.1080/03610927408827101.

[13] D. Davies i D. Bouldin, "A Cluster Separation Measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, t. PAMI-1, s. 224–227, maj 1979. DOI: 10.1109/TPAMI.1979.4766909.

[14] Y. Shafranovich, *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, RFC 4180, paź. 2005. DOI: 10.17487/RFC4180. adr.: https://www.rfc-editor.org/info/rfc4180.

[15] Adalbert F.X. Wilhelm i Walter F. Tichy, *Cran-R afCEC: Active Function Cross-Entropy Clustering*, version 1.0.2, 2018. adr.: https://cran.r-hub.io/web/packages/afCEC/.

[16] Developers, *Machine Learning in Python*, Accessed: March 5, 2024, 2024. adr.: https://scikit-learn.org/stable/modules/cross_validation.html.