Politechnika Wrocławska

Wydział Matematyki
Kierunek studiów: Matematyka Stosowana
Specjalność: –

Praca dyplomowa – inżynierska

WYBRANE METODY GRUPOWANIA ZESPOŁOWEGO

Marcin Kostrzewa

słowa kluczowe:
analiza skupień, grupowanie zespołowe, ensemble learning, funkcja konsensusu, metoda bootstrap

krótkie streszczenie:
Celem pracy jest omówienie oraz empiryczna analiza porównawcza wybranych metod grupowania zespołowego. Przedstawiono sposób działania oraz podstawowe własności omawianych algorytmów. Następnie porównano je szczegółowo ze sobą, wykorzystując wybrane dane rzeczywiste i syntetyczne. Ponadto zbadano wpływ doboru hiperparametrów na efektywność omawianych metod.

Opiekun pracy dyplomowej	dr inż. Adam Zagdański	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław, rok 2023

Wrocław University
of Science and Technology

Faculty of Pure and Applied Mathematics
Field of study: Applied Mathematics
Specialty: –

Engineering Thesis

SELECTED CLUSTERING ENSEMBLE METHODS

Marcin Kostrzewa

keywords:
data clustering, ensemble clustering, ensemble learning, consensus function, bootstrap

short summary:
The goal of the thesis is to discuss and conduct an empirical comparative analysis of selected ensemble clustering methods. It is shown how the considered algorithms work and what are their basic properties. Then, we compare them in detail using selected real and synthetic data. Additionally, the impact of the selection of hyperparameters on the effectiveness of the methods is investigated.

Supervisor	dr inż. Adam Zagdański	.....	.....
	Title/degree/name and surname	grade	signature

*For the purposes of archival thesis qualified to:**

a) category A (perpetual files)

b) category BE 50 (subject to expertise after 50 years)

** delete as appropriate*

stamp of the faculty

Wrocław, 2023

Spis treści

Wstęp

Analiza skupień, zwana również klasteryzacją czy grupowaniem, to jedna z najważniejszych dziedzin uczenia maszynowego. Jest istotnym narzędziem w analizie i przetwarzaniu wielowymiarowych zbiorów danych. Pozwala ona na wykrycie struktur w danych, odkrywanie występujących zależności czy też tworzenie nowych hipotez.

Klasteryzacja jest szeroko wykorzystywana w przeróżnych dziedzinach. Są to między innymi marketing, biologia, medycyna, przetwarzanie obrazów czy wyszukiwanie informacji w zbiorach tekstowych.

Do tej pory opracowano dziesiątki różnych metod klasteryzacji, np. algorytm K-średnich, grupowanie hierarchiczne, klasteryzacja spektralna (p. [32]). Zwracają one uwagę na różne charakterystyki danych, wykorzystują różne kryteria, różnią się stopniem złożoności. Wiele z tych metod zostało zaimplementowanych w popularnych pakietach obliczeniowych i jest łatwo dostępne.

Jednak pomimo takiego bogactwa metod pojawiają się pewne problemy. Nie przedstawiono do tej pory algorytmu, który poprawnie rozpoznałby wewnętrzną strukturę każdego zbioru danych. Inną trudnością jest fakt, że czasami zastosowanie dwóch różnych algorytmów, czy nawet tego samego, ale z inną inicjalizacją, może prowadzić do zupełnie odmiennych wyników.

Rozwiązaniem tych problemów może okazać się zastosowanie algorytmów grupowania zespołowego. Ideą, na której opierają się te metody, jest połączenie wielu decyzji o pogrupowaniu obserwacji podjętych przez różne algorytmy w taki sposób, by uzyskać rezultaty lepsze od pojedynczo zastosowanych metod. Do tej pory przedstawiono wiele różnych rozwiązań (p. [1]), a ciągle proponowane są nowe metody (p. [15], [19]).

Celem pracy będzie omówienie wybranych metod grupowania zespołowego oraz szczegółowe porównanie ich ze sobą przy wykorzystaniu wybranych danych rzeczywistych oraz symulowanych. Analizę porównawczą przeprowadzimy, wykorzystując aż 5 różnych kryteriów oceny jakości klasteryzacji, podczas gdy najczęściej w literaturze stosuje się jedynie wskaźnik zgodności (p. [31], [24], [21], [5]), co stanowi istotną wartość dodaną niniejszej pracy.

Praca składa się z czterech rozdziałów. W rozdziale 1 przedstawione zostanie pojęcie klasteryzacji. Omówione zostaną wybrane algorytmy oraz kryteria oceny jakości grupowania. Rozdział 2 poświęcony zostanie dokładnemu przedstawieniu klasteryzacji zespołowej, jej różnych metod i ich najważniejszych własności. Opisane zostanie siedem wybranych algorytmów. Rozdział 3 będzie stanowić studium przypadku, w którym przyjrzymy się wynikom grupowania uzyskanym przez zarówno standardowe, jak i zespołowe algorytmy klasteryzacji. Treścią rozdziału 4 będzie szczegółowa analiza porównawcza omówionych w pracy algorytmów przy wykorzystaniu odpowiednich kryteriów. Dodatkowo zbadany zostanie wpływ hiperparametrów na skuteczność rozważanych metod.

Wszystkie obliczenia i symulacje zostały przeprowadzone przy wykorzystaniu języka Python. Wizualizacje wyników zostały przygotowane przy pomocy pakietu R. Dokładny

opis wykorzystanych bibliotek i funkcji zostanie przedstawiony w Dodatku. Ponadto implementacje poszczególnych metod napisane przez autora pracy są dostępne w publicznym repozytorium w serwisie GitHub pod następującym adresem:
`https://github.com/Manik2000/bachelor_thesis`.

Rozdział 1

Idea klasteryzacji i jej podstawowe algorytmy

W niniejszym rozdziale przedstawimy krótkie wprowadzenie do analizy skupień. Omówiona będzie ogólna idea i cel stosowania klasteryzacji, a także zaprezentowane zostaną podstawowe algorytmy oraz wybrane kryteria wykorzystywane do oceny jakości grupowania.

W przypadku niepodania źródła opisy w tym rozdziale opierają się na [11], [32].

1.1 Ogólna idea i cel analizy skupień

Analiza skupień, nazywana także grupowaniem lub klasteryzacją (ang. *clustering*, *cluster analysis*), jest przykładem nienadzorowanego uczenia maszynowego — nie wykorzystujemy informacji o etykietkach klas. Jej głównym celem jest wykrycie wewnętrznej struktury danych. Zadanie to polega na wyodrębnieniu rozłącznych klastrów (czyli skupisk lub grup), w taki sposób, aby każdy klaster zawierał obiekty wzajemnie do siebie podobne (do określenia podobieństwa czy niepodobieństwa wykorzystywane będą odpowiednie miary). Jednocześnie odrębne skupiska powinny jak najbardziej różnić się od siebie.

Zakładamy, że $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ to zbiór danych, gdzie $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ to pojedynczy obiekt (obserwacja) określona jako wektor p cech (zmiennych). $\mathcal{C} = \{C_1, C_2, , \ldots, C_K\}$ będzie oznaczało partycję (czyli podział N obserwacji na K skupień), uzyskaną przy pomocy algorytmu grupującego ϕ, $|C_k|$ to liczba obserwacji w k-tym skupieniu, $C(i) \in \{1, \ldots, K\}$ to przyporządkowana i-temu obiektowi etykietka.

Odmienność

Kluczowym pojęciem w klasteryzacji jest zmierzenie niepodobieństwa (odmienności) obserwacji. Posłuży do tego miara odmienności (ang. *dissimilarity measure*).

Definicja 1.1 (Miara odmienności)**.** Miarą odmienności nazywamy taką funkcję $d(x, y)$, która dla dowolnych x, $y \in X$ spełnia:

- $d(x, y) \geqslant 0$,
- $d(x, y) = d(y, x)$,
- $d(x, y) = 0 \Leftrightarrow x = y$.

Najczęściej wykorzystywaną miarą niepodobieństwa jest odległość euklidesowa:

$$d_e(x,y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}. \tag{1.1}$$

W tej pracy wykorzystana również zostanie odległość Manhattan:

$$d_m(x,y) = \sum_{i=1}^{p} |x_i - y_i| . \tag{1.2}$$

Oznaczamy $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ dla wybranej miary odległości i konstruujemy macierz odmienności (niepododobieństwa) $D = \{d_{ij}\}_{i,j=1}^{N}$. Tak określona macierz D będzie wykorzystywana przez różne algorytmy klasteryzacji.

Powyższe miary pozwalają na zbadanie niepodobieństwa w przypadku cech ciągłych. Możliwe jest także uogólnienie pojęcia odmienności na przypadek atrybutów dowolnego typu (np. miara Gowera), które jednak w tej pracy nie będzie wykorzystywane.

Brak aksjomatycznej definicji

Warto podkreślić, że nie istnieje aksjomatyczna definicja klasteryzacji, co więcej pokazano, że bardzo trudno taką stworzyć. W [20] Kleinberg zdefiniował trzy elementarne własności, które powinien spełniać każdy algorytm analizy skupień. Są to:

- niezmienniczość na skalowanie — metoda analiza skupień powinna być niezmiennicza ze względu na zmiany jednostek pomiarowych,
- bogata przestrzeń rozwiązań — powinna zawsze istnieć taka funkcja odległości, która zapewni metodzie ϕ uzyskanie dowolnej partycji,
- zgodność — w wyniku zwiększania odległości pomiędzy klastrami i pomniejszania rozrzutu wewnątrz nich powinniśmy uzyskać ten sam rezultat.

Kleinberg następnie udowadnia, że nie istnieje taki algorytm grupowania, który posiadałby równocześnie wszystkie trzy własności.

Brak ugruntowanych podstaw teoretycznych jest dużym problemem analizy skupień. Nie istnieje bowiem sposób na przedstawienie ogromu metod grupowania w szerszym kontekście, co również uniemożliwia wyciągnięcie ogólnych wniosków.

1.2 Przegląd podstawowych algorytmów

W literaturze (p. [11], [32]) zaproponowano podział algorytmów analizy skupień na następujące kategorie:

- metody grupujące,
- metody hierarchiczne,
- inne: metody oparte na rozkładach probabilistycznych, metody wykorzystujące teorię grafów i wiele innych.

Metody grupujące

Metody te dążą do iteracyjnego rozwiązania zagadnienia optymalizacji wartości rozrzutu wewnątrz skupień:

$$W(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k} d_{ij}^2. \tag{1.3}$$

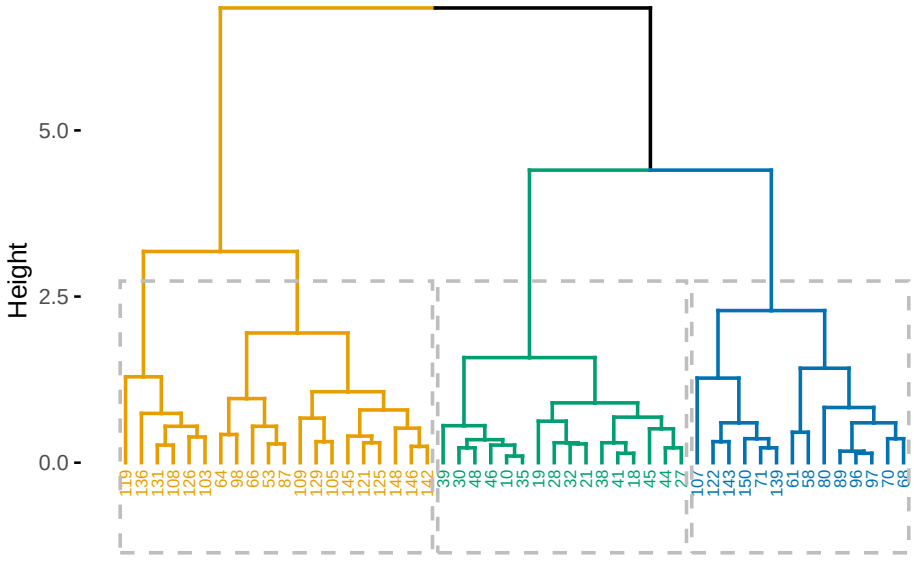
Znalezienie w zbiorze wszystkich partycji tej, która minimalizuje wartość $W(\mathcal{C})$, w rozsądnym czasie jest raczej niemożliwe. Można pokazać (p. [18]), że liczba różnych podziałów N obserwacji na K klastrów wyraża się wzorem:

$$S(N,K) = \frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}k^N. \tag{1.4}$$

Przykładowo, liczba podziałów 50 obserwacji na 2 skupienia jest rzędu 10^{14}, a w przypadku podziału na 3 skupienia rzędu 10^{23}. Konieczne zatem staje się efektywne przeszukiwanie znacznie mniejszej liczby możliwych partycji.

Metody hierarchiczne

Wynikiem działania tych metod jest hierarchia (drzewo) zagnieżdżonych skupień, tzw. dendrogram, którego przykład został przedstawiony na rysunku 1.1. Został on wyznaczony dla podzbioru danych **iris**, które będą omówione w podrozdziale 4.1.



Rysunek 1.1: Dendrogram dla zbioru **iris**.
Źródło: opracowanie własne

Metody hierarchiczne nie wymagają ustalenia z góry docelowej liczby klastrów — wyboru można dokonać już po zastosowaniu algorytmu i zapoznaniu się z wynikami.

Algorytmy hierarchiczne można podzielić na:

- metody aglomeracyjne (ang. *agglomerative*) — na początku każdy obiekt stanowi osobne skupienie, a w kolejnych krokach najbliższe sobie skupienia są łączone, aż do momentu utworzenia jednego dużego skupienia;

- metody dzielące (ang. *divisive*) — w pierwszym kroku wszystkie obiekty tworzą jedno duże skupienie, które jest następnie dzielone tak, aby otrzymać jednorodne grupy.

W następnej części tego podrozdziału zostaną przedstawione cztery standardowe algorytmy klasteryzacji, które dobrze oddają różnorodność metod analizy skupień, a będą także wykorzystywane (jako metody bazowe) przez różne algorytmy grupowania zespołowego. Będą to: metoda K-średnich (ang. *K-means*), metoda PAM, aglomeracyjne grupowanie hierarchiczne oraz klasteryzacja spektralna. Dwa pierwsze algorytmy sklasyfikować można jako metody grupujące, natomiast klasteryzacja spektralna jest przykładem metody opartej na teorii grafów.

1.2.1 Metoda K-średnich

Metoda K-średnich (ang. *K-means*) jest najpopularniejszym i najczęściej stosowanym algorytmem analizy skupień. Wykorzystywaną w tym przypadku miarą niepodobieństwa jest odległość euklidesowa, więc algorytm ten może zostać wykorzystany jedynie do danych ciągłych.

Metoda ta opiera się na iteracyjnym przypisywaniu obserwacji do najbliższych im centrów (średnich), które na początku są wybierane losowo. W każdym kroku, po przyporządkowaniu obiektów do odpowiednich grup, wyznaczane jest nowe centrum danego skupienia. Powtarzamy te kroki, aż do momentu, gdy nie otrzymujemy żadnych zmian w grupowaniu. Algorytm 1 przedstawia schemat metody K-średnich.

Algorytm 1 Metoda K-średnich

Dane wejściowe: X — zbiór danych, K — liczba klastrów

1: Zainicjalizuj losowo K wartości $m_1, \ldots, m_K$, które posłużą jako centra klastrów;

2: Przyporządkuj każdy z obiektów $\mathbf{x}_i$ do najbliższego mu skupienia C_k, gdzie

$$k = \arg\min_{1 \leqslant j \leqslant K} d_e(m_j, \mathbf{x}_i); \tag{1.5}$$

3: Wyznacz nowe centra dla każdego skupiska jako:

$$m_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i; \tag{1.6}$$

4: Powtarzaj kroki 2 i 3 aż do momentu, gdy żadne przyporządkowanie się nie zmieni.

Można pokazać, że (dla ustalonej liczby skupień K) ta metoda dąży do iteracyjnego rozwiązania zadania minimalizacji następującego kryterium:

$$\sum_{i=1}^{N} d_e(\mathbf{x}_i, m_{C(i)}). \tag{1.7}$$

Dużą zaletą algorytmu K-średnich jest jego niewielka złożoność obliczeniowa — $\mathcal{O}(KNt)$, gdzie t to liczba iteracji.

Z racji wykorzystania odległości euklidesowej metoda K-średnich jest mało odporna na obserwacje odstające i ma trudności z poprawnym grupowaniem w przypadku skupień, które nie są sferyczne.

1.2.2 Algorytm PAM

Algorytm PAM (ang. *partitioning around medoids*) polega na sekwencyjnym poszukiwaniu K reprezentatywnych obiektów (medoidów) spośród wszystkich obserwacji znajdujących się w zbiorze danych (p. [28]). Procedura ta wykorzystuje dowolną miarę niepodobieństwa d, dla której wyznaczona zostaje macierz odmienności D. Schemat metody przedstawia algorytm 2.

Algorytm 2 PAM

Dane wejściowe: D — macierz odmienności, K — liczba klastrów

1: krok **BUILD**: Spośród N obserwacji wybieramy sekwencyjnie K medoidów;
2: krok **SWAP**: Minimalizujemy sumę odległości obiektów od medoidów poprzez zastąpienie aktualnych medoidów innymi (do tej pory niewybranymi) obserwacjami.

Metoda PAM może być zastosowana dla dowolnej macierzy odmienności, więc i dla cech dowolnego typu. Jest również bardziej odporna na wpływ obserwacji odstających.

Jej niewątpliwą wadą jest natomiast złożoność obliczeniowa — $\mathcal{O}(K(N-K)^2)$. W związku z tym czas działania algorytmu dla większych zbiorów znacznie się wydłuża, co także zostanie pokazane w dalszej części pracy.

1.2.3 Aglomeracyjne grupowanie hierarchiczne

Hierarchiczne grupowanie aglomeracyjne na początku umieszcza każdy element w osobnym skupieniu. Następnie najbliższe sobie klastry są łączone w jeden (większy) klaster. Krok ten powtarzamy aż do momentu uzyskania jednego skupienia. Algorytm wykorzystuje macierz odmienności z wybraną miarą niepodobieństwa. Do łącznia klastrów stosuje się funkcję łączącą (ang. *linkage function*). Najbardziej powszechne funkcje, które mogą posłużyć do połączenia skupień G i H, to:

- odległość najbliższego sąsiada (ang. *single linkage*):

$$d_{SL}(G,H) = \min_{i\in G, j\in H} d_{ij}; \tag{1.8}$$

- odległość najdalszego sąsiada (ang. *complete linkage*):

$$d_{CL}(G,H) = \max_{i\in G, j\in H} d_{ij}; \tag{1.9}$$

- odległość średnia (ang. *average linkage*):

$$d_{AL}(G,H) = \frac{1}{|G|\,|H|} \sum_{i\in G, j\in H} d_{ij}; \tag{1.10}$$

- metoda Warda (ang. *Ward's method*):

$$d_W(G,H) = \frac{|G|\,|H|}{|G|+|H|}\left\|\overline{G}-\overline{H}\right\|_2, \tag{1.11}$$

gdzie $\overline{G}$ i $\overline{H}$ oznaczają centroidy, czyli wektorowe średnie wyznaczone na bazie wszystkich obserwacji należących odpowiednio do skupienia G lub H.

Metoda najbliższego sąsiada bierze pod uwagę jedynie odległość najbliższych dwóch obserwacji z osobnych klastrów, co często może prowadzić do nadmiernego złączenia kilku klastrów w jeden. Otrzymujemy wówczas wydłużone skupienia o dużej średnicy. Odległość najdalszego sąsiada z kolei będzie dążyć do utworzenia wyraźnie odseparowanych, zwartych klastrów. Kompromis pomiędzy obiema metodami stanowi odległość średnia, która zwraca zwarte, dobrze odseparowane skupienia. W przypadku metody Warda, w przeciwieństwie do innych metod łączenia skupień, zamiast bezpośredniego mierzenia odległości pod uwagę brany jest rozrzut (wariancja) skupień. Ponadto metoda ta jest uważana za najskuteczniejszą w przypadku obecności szumu w analizowanych danych.

Algorytm 3 przedstawia kolejne kroki metody aglomeracyjnego grupowania hierarchicznego.

Algorytm 3 Aglomeracyjne grupowanie hierarchiczne

Dane wejściowe: D — macierz odmienności

1: Stwórz z każdej obserwacji osobne skupienie;
2: Wykorzystując wybraną funkcję łączącą, połącz ze sobą najbliższe sobie skupienia;
3: Powtarzaj powyższy krok aż do momentu uzyskania jednego skupienia zawierającego wszystkie obserwacje.

Zaletą metody hierarchicznej jest brak ograniczeń dotyczących kształtu skupień. W przypadku metod grupujących, takich jak algorytm K-średnich, jedynie sferyczne, wypukłe skupienia mogłyby być poprawnie wykryte. Natomiast niewątpliwą wadą jest złożoność obliczeniowa, która uwidacznia się już przy zbiorach danych składających się z kilku tysięcy obserwacji.

1.2.4 Klasteryzacja spektralna

Klasteryzacja spektralna (ang. *spectral clustering*) ma swoje korzenie w teorii grafów. Metoda ta została opracowana z myślą o wykrywaniu w zbiorze danych skupień o kształtach wklęsłych. Wykorzystuje ona macierz podobieństwa S pomiędzy wszystkimi obserwacjami X. Taką macierz można otrzymać z macierzy odmienności, stosując np. następujące przekształcenie:

$$s_{ij} = \exp\left(\frac{-d_{ij}^2}{c}\right), \tag{1.12}$$

gdzie $c > 0$ to parametr skali.

Na podstawie macierzy S możemy następnie wyznaczyć macierz S', gdzie:

$$s'_{ij} = \begin{cases} s_{ij}, & \text{jeśli } i \text{ jest wśród } Q \text{ najbliższych sąsiadów } j, \\ 0, & \text{w przeciwnym wypadku,} \end{cases} \tag{1.13}$$

gdzie Q jest ustaloną (wybraną) liczbą najbliższych sąsiadów.

W zależności od preferencji wybieramy $W = S$ lub $W = S'$. Traktujemy W jako ważoną macierz sąsiedztwa pewnego grafu G, gdzie wierzchołki stanowią obserwacje z X, a krawędzie między nimi ważone są poprzez elementy w_{ij}. Wyznaczamy laplasjan L grafu G:

$$L = W - F, \tag{1.14}$$

gdzie elementy diagonalnej macierzy F to $f_{ii} = \sum_j w_{ij}$.

Następnie wyznaczamy m wektorów własnych macierzy L, które odpowiadają m najmniejszym wartościom własnym i tworzymy z nich macierz $A \in \mathbf{R}^{N \times m}$. W kolejnym kroku wykorzystujemy standardowy algorytm K-średnich, a uzyskane etykietki klas przypisujemy obserwacjom ze zbioru X. Algorytm 4 przedstawia kolejne kroki klasteryzacji spektralnej.

Algorytm 4 Klasteryzacja spektralna

Dane wejściowe: K — liczba klastrów, S — macierz podobieństwa

1: Wstaw $W = S$ lub $W = S'$, gdzie s'_{ij} są określone jak w (1.13);
2: Wyznacz $L = W - F$, gdzie F to macierz diagonalna i $f_{ii} = \sum_j w_{ij}$;
3: Wyznacz m wektorów własnych L odpowiadającym m najmniejszym wartościom własnym i stwórz z nich macierz A;
4: Wykorzystaj algorytm K-średnich do pogrupowania wierszy macierzy A w K klastrów i zwróć etykietki skupień.

W przypadku tego algorytmu konieczny jest wybór sposobu tworzenia macierzy W, liczby wektorów własnych oraz liczby klastrów. W tej pracy zdecydowano, że $W = S$, a $m = K$.

Metoda klasteryzacji spektralnej z uwagi na konieczność wyznaczania wektorów własnych charakteryzuje się dużą złożoność obliczeniową ($\mathcal{O}(N^3)$).

1.3 Kryteria stosowane w ocenie wyników klasteryzacji

Rezultat grupowania w istotny sposób może zależeć od wyboru algorytmu lub wyboru parametrów algorytmu (np. liczby skupień K). Konieczna jest także możliwość oceny istotności otrzymanych wyników klasteryzacji. Warto zauważyć, że większość metod analizy skupień wyznaczy skupienia w danych, w których taka struktura nawet nie istnieje.

Potrzebne są więc narzędzia służące ocenie przeprowadzonej analizy skupień. Do tego wykorzystywane są kryteria walidacji klasteryzacji. Dzielimy je na:

- wskaźniki zewnętrzne, które odwołują się do zewnętrznej informacji o etykietkach klas, która nie jest wykorzystywana w samym procesie grupowania,
- wskaźniki wewnętrzne, które wykorzystują jedynie informacje zawarte w danych, dla których przeprowadzono analizę skupień.

Do tej pory zostało zaproponowanych bardzo wiele różnych kryteriów oceny analizy skupień (p. [9], [10]). W tej pracy wykorzystanych zostanie pięć odpowiednio zróżnicowanych wskaźników, które pozwolą na szczegółową ocenę najważniejszych aspektów otrzymanych wyników grupowania. Wskaźniki te będą opisane w dalszej części tego podrozdziału.

Wprowadzimy teraz pomocnicze oznaczenia, które będą potrzebne do zdefiniowania niektórych wskaźników. Niech $P_1 = \left\{P_1^{(1)}, \ldots, P_s^{(1)}\right\}$ i $P_2 = \left\{P_1^{(2)}, \ldots, P_t^{(2)}\right\}$ będą dwiema partycjami zbioru X. Wtedy:

- n_{11} to liczba par obiektów, które znalazły się w tym samym skupieniu dla obu partycji,
- n_{00} to liczba par, które zostały przydzielone do różnych klastrów w obu partycjach,
- n_{10} to liczba par obiektów, które znalazły się w tym samym skupieniu dla pierwszej partycji, ale w różnych skupieniach dla drugiej,
- n_{01} to liczba par w sytuacji odwrotnej.

1.3.1 Wskaźniki wewnętrzne

Wskaźniki wewnętrzne oceniają różne charakterystyki grupowania (p. [10]). Są to:

- zwartość (ang. *compactness*) — mały rozrzut wewnątrz klastrów,
- separacja przestrzenna — odpowiednio duży rozrzut pomiędzy różnymi klastrami,
- spójność (ang. *connectedness*) — przyporządkowanie sąsiadujących obserwacji do tych samych skupień,
- stabilność (ang. *stability*) — powtarzalność wyników, odporność na zaburzenia danych.

Wskaźnik silhouette

Wskaźnik silhouette (p. [27]) definiujemy jako średnią wartości silhouette wszystkich obserwacji danego zbioru danych:

$$\mathcal{S} = \frac{1}{N}\sum_{i=1}^{N} S(i), \tag{1.15}$$

gdzie dla każdej obserwacji i:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i),\, b(i)\}}, \tag{1.16}$$

$$a(i) = \frac{1}{\left|C_{C(i)}\right| - 1} \sum_{\{j:\, C(j)=C(i),\, j\neq i\}} d_{ij}, \tag{1.17}$$

$$b(i) = \min_{k\neq C(i)} \frac{1}{|C_k|} \sum_{C(j)=k} d_{ij}. \tag{1.18}$$

Zauważmy, że $a(i)$ mierzy średnią odmienność obserwacji i od pozostałych obiektów należących do tego samego skupienia, natomiast $b(i)$ średnią odmienność tej obserwacji od

najbliższego jej skupienia. Wskaźnik silhouette bierze zatem pod uwagę zwartość klastrów i ich separację przestrzenną.

Wartość wskaźnika silhouette zawiera się w przedziale $[-1, 1]$. $S(i) \approx -1$ oznacza złe przyporządkowanie obserwacji, $S(i)$ bliskie 1 poprawne przyporządkowanie. $S(i) \approx 0$, które implikuje, że $a(i) \approx b(i)$, oznacza to, że nie ma pewności, do którego skupienia powinien zostać przydzielony obiekt.

Wskaźnik spójności

Opis wskaźnika spójności (ang. *connectedness*) został oparty na [2].

Niech $nn_{i(j)}$ oznacza j-tego najbliższego sąsiada obserwacji i. Zdefiniujmy dalej

$$x_{i,nn_{i(j)}} = \begin{cases} 0, \text{ jeśli } C\left(nn_{i(j)}\right) = C(i), \\ \frac{1}{j}, \text{ w przeciwnym wypadku.} \end{cases} \tag{1.19}$$

Wskaźnik spójności definiujemy wtedy następująco:

$$\text{Conn}(\mathcal{C}) = \sum_{i=1}^{N} \sum_{j=1}^{L} x_{i,nn_{i(j)}}, \tag{1.20}$$

gdzie L jest parametrem, decydującym o tym ile sąsiadów obserwacji i jest branych pod uwagę. W tej pracy zostało przyjęte, że $L = 10$.

$\text{Conn}(\mathcal{C}) \in [0, \infty]$, a jego wartość powinna być jak najmniejsza.

Wskaźnik stabilności

Istnieje wiele sposobów na badanie stabilności (powtarzalności) analizy skupień. W [2] implementuje się metody badające wrażliwość metod na usuwanie poszczególnych cech. W tej pracy wykorzystanie zostanie podejście oparte na procedurze bootstrap, podobne do tego opisanego w [12].

Na początku skorzystamy z algorytmu grupowania ϕ na pełnych danych, by uzyskać partycję referencyjną. Następnie zastosujemy ϕ dla każdej replikacji bootstrapowej, czyli próby powstałej poprzez wylosowanie N elementów ze zwracaniem z wyjściowego zbioru X. Dla uzyskanej partycji oraz partycji referencyjnej wyznaczymy wartość współczynnika Jaccarda. Dla partycji P_1 i P_2 określa się go jako:

$$\text{Jaccard}(P_1, P_2) = \frac{n_{11}}{n_{10} + n_{11} + n_{01}}, \tag{1.21}$$

Wskaźnik stabilności wyznaczamy jako średnią arytmetyczną wartości współczynnika Jaccarda wyznaczonych dla wszystkich replikacji bootstrapowych.

Kolejne kroki procedury opisuje algorytm 5.

Algorytm 5 Procedura oceny stabilności grupowania

Dane wejściowe: X — zbiór danych, B — liczba replikacji bootstrapowych, ϕ — algorytm analizy skupień

1: $\mathbf{y}_r = \phi(X)$ — etykietki klas dla całego zbioru danych
2: **for** $i = 1$ to B **do**
3: $\quad X_B^i \leftarrow$ replikacja bootstrapowa X
4: $\quad \mathbf{y}_i \leftarrow \phi(X_B^i)$
5: $\quad \mathbf{y}_r^{'}, \mathbf{y}_i^{'} \leftarrow$ etykietki klas dla obserwacji występujących w $X_B^i \cap X$
6: $\quad \gamma_i \leftarrow \text{Jaccard}\left(\mathbf{y}_r^{'}, \mathbf{y}_i^{'}\right)$
7: **end for**
8: **return** $\frac{1}{B}\sum_{i=1}^{B}\gamma_i$

Z racji dużej złożoności obliczeniowej niektórych algorytmów klasteryzacji najczęściej, podobnie jak w tej pracy, przyjmuje się liczbę replikacji $B = 20$. Wspomniane w [13] reguły praktyczne przyjmują, że wartości $\gamma < 0.6$ świadczą o niestabilnej metodzie, $\gamma > 0.75$ wskazują na dobrą, odporną na zaburzenia danych metodę, wartości pomiędzy na metodę wykrywającą zależności w danych mimo zaburzeń, ale do której należy mieć ograniczone zaufanie.

1.3.2 Wskaźniki zewnętrzne

Skorygowany wskaźnik Randa

Skorygowany wskaźnik Randa (ang. *adjusted Rand index* — ARI) służy do mierzenia podobieństwa dwóch partycji zbioru obserwacji X (p. [17]). W tej pracy wykorzystany zostanie do porównania otrzymanego grupowania z rzeczywistym, znanym podziałem na klasy. Dla dwóch partycji wskaźnik Randa definiuje się jako:

$$\text{Rand} = \frac{n_{11} + n_{00}}{n_{11} + n_{10} + n_{01} + n_{00}}. \tag{1.22}$$

Określa on więc, jak często obie partycje były ze sobą zgodne.

Skorygowany wskaźnik Randa wprowadza poprawkę uwzględniającą prawdopodobieństwo, że obserwacje trafią do tego samego klastra lub dwóch różnych w przypadku losowych partycji.

Najpierw wyznacza się macierz kontyngencji dla partycji P_1 i P_2:

Tabela 1.1: Macierz kontyngencji wyznaczona dla dwóch partycji.

	$P_1^{(1)}$	$P_2^{(1)}$	$\dots$	$P_s^{(1)}$	Suma
$P_1^{(2)}$	p_{11}	p_{12}	$\cdots$	p_{1s}	a_1
$P_2^{(2)}$	p_{21}	p_{22}	$\cdots$	p_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$P_t^{(2)}$	p_{t1}	p_{t2}	$\cdots$	p_{ts}	a_t
Suma	b_1	b_2	$\dots$	b_s	N

Źródło: opracowanie własne.

Skorygowany wskaźnik Randa wyraża się wzorem:

$$ARI = \frac{\sum_{ij}\binom{p_{ij}}{2} - \frac{S_a S_b}{\binom{N}{2}}}{\frac{1}{2}(S_a + S_b) - \frac{S_a S_b}{\binom{N}{2}}}, \tag{1.23}$$

gdzie $S_a = \sum_{i=1}^{t}\binom{a_i}{2}$, $S_b = \sum_{j=1}^{s}\binom{b_j}{2}$.

ARI przyjmuje wartości z zakresu $[-1, 1]$. Wartość $ARI \approx 0$ wskazuje na losowy podział na klastry, wartość $ARI \approx 1$ na idealne grupowanie, a wartości ujemne na grupowanie gorsze od losowego.

Zgodność partycji

Zgodność partycji z rzeczywistym podziałem na skupienia jest bardzo często wykorzystywana w ocenie algorytmów analizy skupień (por. [15], [31]). W tym przypadku zakładamy, że rzeczywista liczba klas (grup) K jest znana. Porównujemy zatem partycję $\mathcal{C}$ z rzeczywistym podziałem na klasy $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$. Dodatkowo niech $\mathbf{r} = (r_1, r_2, \ldots, r_N)$ to etykietki klas odpowiadające poszczególnym obserwacjom. Poprzez $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ oznaczamy natomiast etykietki grup (skupień) przydzielone obiektom w wyniku klasteryzacji.

Najpierw należy wyznaczyć macierz kontyngencji:

Tabela 1.2: Macierz kontyngencji wyznaczona dla klasteryzacji i rzeczywistego podziału.

	R_1	R_2	$\ldots$	R_K
C_1	c_{11}	c_{12}	$\cdots$	c_{1K}
C_2	c_{21}	c_{22}	$\cdots$	c_{2K}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
C_K	c_{K1}	c_{K2}	$\cdots$	c_{KK}

Źródło: opracowanie własne.

Wyznaczamy permutację $\pi_o \in S_K$ taką, że:

$$\pi_o = \arg\max_{\pi \in S_K} \sum_{i=1}^{K} c_{ii}, \tag{1.24}$$

która zapewnia optymalne przypisanie klas. Po wyznaczeniu takiej permutacji wskaźnik zgodności partycji wyznaczamy jako:

$$\text{Agreement} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\{r_i = \pi_o(y_i)\}}. \tag{1.25}$$

Podobnie jak dokładność klasyfikacji, wskaźnik (1.25) jest często wyrażany w procentach.

Wyznaczenie optymalnej permutacji może okazać się kosztowne obliczeniowo. Z tego powodu w praktyce wykorzystuje się odpowiednie algorytmy kombinatoryczne (p. [3]), które pozwalają uniknąć przeszukiwania wszystkich $K!$ permutacji.

Rozdział 2

Klasteryzacja zespołowa

Zastosowanie standardowej metody analizy skupień (w szczególności jednego z algorytmów wymienionych w rozdziale 1) w wielu przypadkach nie pozwala na uzyskanie zadowalających rezultatów. W tym rozdziale omówimy podejście do klasteryzacji, które oparte jest na idei uczenia zespołowego (ang. *ensemble learning*) i często prowadzi do poprawy jakości grupowania. Oprócz przedstawienia niezbędnych podstaw teoretycznych klasteryzacji zespołowej, szczegółowo omówione będą również wybrane algorytmy.

2.1 Wprowadzenie

Grupowanie zespołowe (ang. *ensemble clustering*, *consensus clustering*) jest procesem tworzenia wielu partycji dla danego zbioru danych i wykorzystania zależności zaobserwowanych pomiędzy nimi do wyznaczenia ostatecznego rozwiązania, które będzie bardziej stabilne oraz pod pewnymi względami lepsze niż indywidualne grupowania.

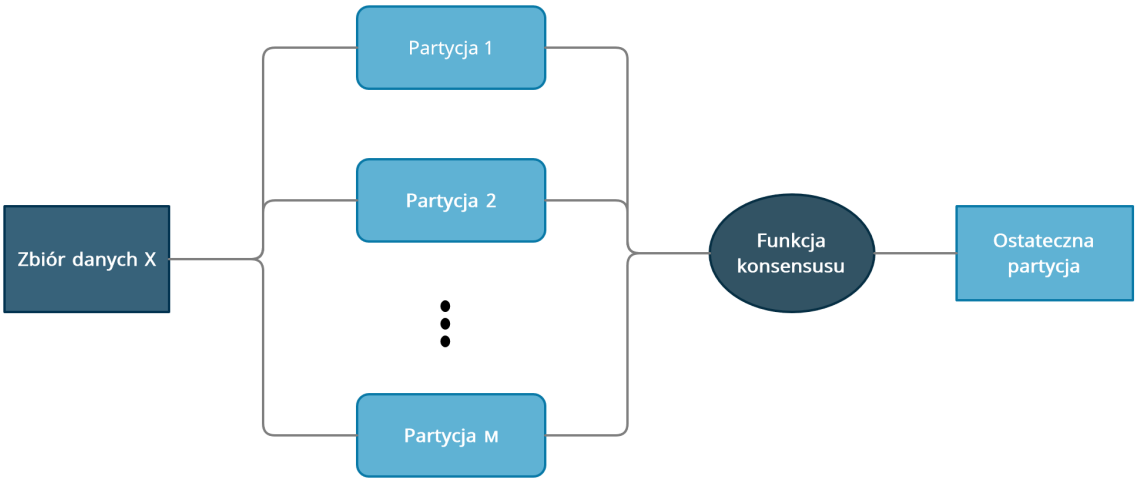
W literaturze (p. [30]) wymienia się następujące własności, którymi powinny cechować się algorytmy klasteryzacji zespołowej:

- solidność (ang. *robustness*) — kombinacja poszczególnych wyników powinna być lepsza niż każdy z nich,
- spójność (ang. *consistency*) — końcowy rezultat powinien być „zbliżony" do poszczególnych składowych,
- stabilność (ang. *stability*) — uzyskiwane wyniki powinny być mniej wrażliwe na szum i wartości odstające,
- odkrywczość (ang. *novelty*) — grupowanie zespołowe musi umożliwiać uzyskanie wyników nieosiągalnych przez standardowe algorytmy klasteryzacji.

Klasteryzacja zespołowa polega na wykonaniu dwóch kroków:

1. wygenerowaniu M partycji $\mathcal{P} = \{P_1, P_2 \ldots, P_M\}$ za pomocą wybranych algorytmów grupowana (różnych bądź takich samych),
2. połączeniu M uzyskanych partycji w jedną, końcową partycję P^* za pomocą wybranej funkcji konsensusu (ang. *consensus function*).

Rysunek 2.1 przedstawia ogólny schemat działania metod grupowania zespołowego.



Rysunek 2.1: Ogólny schemat działania algorytmów grupowania zespołowego.
Źródło: Opracowanie własne

Warto podkreślić, że każdy z dwóch wyżej wymienionych etapów w procedurze grupowania zespołowego pozwala na dużą dowolność i eksperymentowanie w celu poszukiwania odpowiedniego wariantu metody. Przyjrzymy się teraz różnym sposobom generowania partycji i różnym funkcjom konsensusu.

2.1.1 Generowanie partycji

W literaturze (p. [1]) wymienia się następujące metody generowania partycji:

- jednolity zespół algorytmów klasteryzacji — wykorzystuje się tylko jeden algorytm, którego inicjalizacja jest losowa, np. metoda K-średnich, co umożliwia uzyskanie różnych partycji składowych;
- wybieranie różnej liczby skupień dla poszczególnych partycji — można wykorzystać metody grupujące, metody hierarchiczne, klasteryzację spektralną i wiele innych;
- przekształcanie zbioru danych — kolejne partycje wyznaczamy np. dla replikacji bootstrapowych X, losowego podzbioru danych, zbioru powstałego z losowego rzutowania danych X na ustaloną podprzestrzeń, zbioru powstałego z X poprzez wybranie losowej liczby cech;
- zróżnicowany zespół algorytmów analizy skupień — wykorzystujemy różne algorytmy do uzyskania składowych partycji;
- mieszane heurystyki — czyli równoczesne zastosowanie co najmniej dwóch metod podanych powyżej.

Stosowanie różnych metod wpływa na różnorodność partycji (ang. *diversity*). W literaturze pokazano, że agregowanie partycji charakteryzujących się dużą różnorodnością w wielu sytuacjach wpływa korzystnie na otrzymywane rezultaty analizy skupień (p. [31], [5], [8]).

W tej pracy przede wszystkim stosowane będzie pierwsze podejście — poszczególne partycje będą wyznaczane za pomocą algorytmu K-średnich, który za każdym razem losowo wybiera początkowe centra skupień. Podejście to jest bardzo często stosowane w literaturze (p. [31], [24], [6]), a jego wielką zaletą jest szybki czas uzyskania nawet bardzo dużej liczby składowych partycji.

W rozdziale 4.4 zbadamy dodatkowo, jaki wpływ na uzyskane wyniki ma zastosowanie różnych bazowych algorytmów klasteryzacji, z różnie dobraną liczbą skupień.

2.1.2 Funkcja konsensusu

Formalnie funkcję konsensusu można rozumieć jako przekształcenie zbioru zawierającego M składowych partycji $\mathcal{P}$ w końcową partycję P^*. W ogólności wykorzystuje ona albo informacje o tym, jakie etykietki klas zostały nadane obserwacjom w poszczególnych partycjach, albo jak często różne obiekty były przyporządkowane do tego samego klastra, co określamy jako współwystępowanie (ang. *co-occurrence*).

Do tej pory w literaturze zaproponowano wiele różnych funkcji konsensusu. Ze względu na to, jakiego typu funkcję konsensusu wykorzystują algorytmy grupowania zespołowego, można je podzielić na cztery główne kategorie (por. [1]):

- metody bezpośrednie (ang. *direct*) — korzystając z wyznaczonych etykietek skupień, metody te stosują różne schematy głosowania większościowego (ang. *majority voting*), przydzielając tym samym obserwacje do odpowiedniego klastra;
- metody oparte na cechach (ang. *feature-based*) — algorytmy opierają swoje działanie na potraktowaniu etykietek skupień jako cech kategorycznych (nominalnych);
- metody wykorzystujące podobieństwo pomiędzy parami obserwacjami — metody te najczęściej opierają się na pojęciu macierzy współwystępowania (ang. *co-association matrix*);
- metody, które problem klasteryzacji formułują jako odpowiednie zagadnienie z teorii grafów.

Konieczne do zrozumienia działania niektórych algorytmów grupowania zespołowego przedstawionych w tej pracy jest pojęcie macierzy współwystępowania (ang. *co-association matrix*).

Definicja 2.1 (Macierz współwystępowania)**.** Dla każdej partycji P_k, $k = 1, \ldots, M$, wyznaczamy macierz $W^{(k)}$, której elementami są:

$$w_{ij}^{(k)} = \begin{cases} 1, & \text{jeśli obserwacje } i \text{ oraz } j \text{ znalazły się w tym samym klastrze,} \\ 0, & \text{w przeciwnym wypadku.} \end{cases} \tag{2.1}$$

Macierzą współwystępowania nazywamy macierz W o elementach:

$$w_{ij} = \frac{1}{M} \sum_{k=1}^{M} w_{ij}^{(k)}. \tag{2.2}$$

Macierz W wskazuje na to, jak często średnio dane obserwacje były grupowane w tych samych skupieniach.

2.2 Wybrane algorytmy

Wyboru różnych metod dokonano w taki sposób, by stanowiły one dobry przekrój różnych podejść stosowanych w grupowaniu zespołowym. Każda z wymienionych w podrozdziale 2.1.2 kategorii jest reprezentowana przez co najmniej jeden algorytm.

W przypadku metod bezpośrednich będzie to algorytm BaggedEnsemble, a dla metod *feature-based* algorytm KModesEnsemble. Omówione zostaną aż trzy metody oparte na idei współwystępowania — będą to algorytmy SpectralEnsemble, HierarchicalEnsemble oraz CoAssocEnsemble. Algorytm GraphConsensus reprezentować będzie metody oparte na teorii grafów.

Dodatkowo zaprezentowana zostanie metoda BaggedEnsemble, którą można uznać za podejście hybrydowe i trudno przydzielić do jakiejkolwiek kategorii.

Nazwy wszystkich metod zostały zaproponowane przez autora pracy.

2.2.1 BaggedMajority

Metoda BaggedMajority została przedstawiona w [5]. Polega ona na wykorzystaniu wybranego algorytmu bazowego do klasteryzacji B bootstrapowych replikacji zbioru X — tę rolę w artykule źródłowym pełni PAM, choć w implementacjach wykorzystywany jest algorytm K-średnich (por. [14]). Dla każdej replikacji uzyskane etykietki skupień uzgadniamy z referencyjnym grupowaniem przeprowadzonym na całych danych. Proces ten przebiega tak samo, jak przy wyliczaniu wskaźnika zgodności partycji opisanym w podrozdziale 1.3.2 — wyznaczamy macierz kontyngencji i szukamy odpowiedniej permutacji, która zapewnia optymalne przypisanie.

Na koniec zliczamy głosy — obserwacja zostaje przydzielona do tego skupienia, na które wskazywało najwięcej partycji. Algorytm 6 przedstawia kolejne kroki tej metody.

Algorytm 6 BaggedMajority

Dane wejściowe: K — liczba klastrów, B — liczba replikacji bootstrapowych, ϕ — algorytm klasteryzacji

1: Wyznacz referencyjne etykietki skupień $y_r \leftarrow \phi(X)$;
2: **for** $i = 1$ to B **do**
3: Wyznacz X_B^i — replikacja bootstrapowa X;
4: Wyznacz etykietki klastrów dla X_B^i — y_B;
5: Dla obserwacji z $X \cap X_B^i$ uzgodnij etykietki y_B z y_r;
6: **end for**
7: Przypisz obiektom te skupienia, które były przydzielane im najczęściej.

W [5] przeprowadzono empiryczną weryfikację skuteczności metody BaggedMajority w oparciu zarówno o dane symulowane, jak i rzeczywiste dane mikromacierzowe związane z zagadnieniem diagnostyki nowotworowej. Do oceny klasteryzacji wykorzystano wskaźnik zgodności partycji. Otrzymane rezultaty potwierdziły, że za pomocą tego algorytmu jesteśmy w stanie uzyskać wyniki co najmniej tak dobre, a często i lepsze niż w przypadku wykorzystywania jedynie metody bazowej. Co więcej, w [5] zaobserwowano, że BaggedMajority była bardziej odporna w przypadku występowania w zbiorze danych dużej liczby zmiennych zakłócających (ang. *noise variables*).

2.2.2 BaggedEnsemble

Algorytm BaggedEnsemble został zaproponowany przez Leischa w [21]. Polega on na B-krotnym zastosowaniu bazowego algorytmu klasteryzacji dla replikacji bootstrapowych zbioru X. Zakłada się, że algorytm bazowy powinien móc zwrócić centra klastrów — może to być metoda K-średnich czy PAM. Następnie z otrzymanych centrów (dla wszystkich replikacji) tworzymy nowy zbiór, wyznaczamy dla niego macierz niepodobieństwa i wykorzystujemy grupowanie hierarchiczne z ustalonymi hiperparametrami, aby uzyskać podział na ustaloną z góry liczbę klastrów. Wykorzystać można różne funkcje łączące i różne miary odmienności. Na koniec obserwacje z wyjściowego zbioru X przydzielamy do tego klastra, w którym znajduje się najbliższe im centrum. Algorytm 7 przedstawia schemat działania BaggedEnsemble.

Algorytm 7 BaggedEnsemble

Dane wejściowe: K — liczba klastrów, H — liczba centrów, B — liczba powtórzeń

1: Wygeneruj B bootstrapowych replikacji zbioru X;
2: Zastosuj algorytm bazowy na każdej replikacji otrzymując $B \cdot H$ centrów;
3: Stwórz z centrów nowy zbiór C_B i wykorzystując aglomeracyjne grupowanie hierarchiczne pogrupuj centra na K klastrów;
4: Przydziel obserwację z X do tego klastra, w którym znajduje się najbliższe jej centrum.

Grupowanie hierarchiczne, tak jak wspomniano w podrozdziale 1.2.3, charakteryzuje się dużą złożonością obliczeniową. Dla zbiorów składających się z kilku tysięcy obserwacji staje się procedurą bardzo kosztowną czasową. BaggedEnsemble, poprzez wstępne wyznaczenie centrów — reprezentatywnych dla wyjściowych danych punktów, redukuje rozmiar danych i tym samym sprawia, że wykorzystanie grupowania hierarchicznego nie powoduje aż tak znacznych trudności obliczeniowych. Taka dwustopniowa procedura pozwala również na wyeliminowanie potencjalnego szumu z analizowanych danych.

W [21] zbadano efektywność metody BaggedEnsemble, wykorzystując syntetyczne i rzeczywiste zbiory danych oraz wskaźnik zgodności partycji. W porównaniu do metod bazowych (algorytmów K-średnich oraz HCL (ang. *hard competitive learning*)) rozważany algorytm uzyskał lepsze wyniki, które dodatkowo charakteryzowały się mniejszym rozrzutem.

Wybór hiperparametrów metody BaggedEnsemble autor uzależnił od struktury danych i liczby obserwacji. Dla bardziej złożonych zbiorów dobierano większą liczbę bazowych centrów H oraz odpowiednio wybierano funkcję łączącą (w artykule były to odległość najbliższego sąsiada i metoda Warda). Można się zatem spodziewać dużej wrażliwości metody BaggedEnsemble na dobór jej hiperparametrów.

2.2.3 KModesEnsemble

Metoda KModesEnsemble została przedstawiona w [24]. Metoda ta wykorzystuje algorytm K-modes, zaproponowany przez Huanga jako odpowiednik K-means dla zmiennych kategorycznych (p. [16]).

Dla każdej partycji P_i, gdzie $i = 1, \ldots, M$, otrzymujemy odpowiadający jej wektor etykietek $\mathbf{y}_i$. Formujemy następnie z tych wektorów macierz Y o wymiarze $N \times M$:

$$Y = \begin{pmatrix} \mathbf{y}_1^T & \mathbf{y}_2^T & \ldots & \mathbf{y}_M^T \end{pmatrix}. \tag{2.3}$$

Uzyskujemy tym samym nowy zbiór cech, tym razem wyłącznie kategorycznych. W kolejnym kroku metoda K-modes wykorzystuje macierz Y do wyznaczenia końcowej partycji. Algorytm korzysta z następującej funkcji odmienności:

$$d(x, y) = \sum_{i=1}^{M} \left(1 - \delta(x_i, y_i)\right), \tag{2.4}$$

gdzie:

$$\delta(x_i, y_i) = \begin{cases} 1, \; x_i = y_i, \\ 0, \; x_i \neq y_i. \end{cases} \tag{2.5}$$

W przypadku algorytmu K-modes rozpoczynamy od losowego wybrania K unikalnych obserwacji ze zbioru Y, które będą wykorzystane jako początkowe centra skupień (dominanty lub mody). Wówczas każdy z obiektów jest iteracyjnie przypisywany do najbliższej mu mody. Następnie dla każdego skupienia k wyznaczone zostaje nowe „centrum" m_k, które spełnia:

$$m_k = \arg\min_{z} \sum_{C(i)=k} d(z, x_i). \tag{2.6}$$

Kolejne kroki metody KModesEnsemble przedstawia algorytm 8.

Algorytm 8 KModesEnsemble

Dane wejściowe: K — liczba klastrów, X — zbiór danych

1: Wygeneruj M partycji za pomocą wybranego algorytmu klasteryzacji;
2: Utwórz macierz Y zgodnie z równaniem (2.3);
3: Zastosuj do macierzy Y algorytm K-modes dla ustalonej (docelowej) liczby skupień K;
4: Zwróć uzyskaną partycję.

Zaletą metody KModesEnsemble wymienioną w [24] jest relatywnie niska złożoność obliczeniowa w porównaniu do innych metod. Wynika to z małej złożoności obliczeniowej metody K-modes — $\mathcal{O}\left((t+1)\,KN\right)$, gdzie t to liczba iteracji.

W [24] porównano algorytm KModesEnsemble z kilkoma innymi metodami klasteryzacji zespołowej (np. CSPA, HGPA, MCLA, które opierają się na teorii grafów, p. [29]), wykorzystując wskaźnik zgodności oraz zbiór danych **iris** (p. podrozdział 4.1). Zbadano również, jak zmienia się skuteczność metod wraz ze wzrastającą liczbą składowych partycji. Metoda KModesEnsemble, choć nie była jednoznacznie najlepsza, okazała się konkurencyjna wobec innych algorytmów. Autorzy zwrócili uwagę na jej relatywnie krótszy czas działania oraz szybką zbieżność do końcowego rozwiązania. Efektywność algorytmu poprawiała się wraz ze zwiększającą się liczbą partycji.

2.2.4 SpectralEnsemble

Autorzy [23] zaproponowali, aby wykorzystać algorytm klasteryzacji spektralnej (p. podrozdział 1.2.4) jako funkcję konsensusu. Jako dane wejściowe możemy przyjąć macierz

podobieństwa wyznaczoną dla zbioru X. W przypadku metody SpectralEnsemble tę funkcję pełni macierz współwystępowania (p. Definicja 2.1). Otrzymaną za pomocą klasteryzacji spektralnej partycję traktujemy jako końcową. Metoda wymaga podania docelowej liczby klastrów.

Kolejne kroki SpectralEnsemble przedstawia algorytm 9.

Algorytm 9 SpectralEnsemble

Dane wejściowe: W — macierz współwystępowania, K — liczba klastrów

1: Zainicjalizuj algorytm klasteryzacji spektralnej z K klastrami, traktując macierz W jako macierz podobieństwa;
2: Zwróć uzyskaną partycję.

W [23] zostało pokazane, że algorytm charakteryzuje się dużą odpornością na słabą jakość poszczególnych partycji składowych — zachowuje on wysoką skuteczność, nawet gdy część partycji jest niezgodna z rzeczywistymi klasami (oczywiście zakładając, że pozostałe partycje stanowią większość).

Z racji złożoności obliczeniowej algorytmu klasteryzacji spektralnej, w przypadku metody SpectralEnsemble odczuwalny będzie wzrost liczby obserwacji w zbiorze danych.

Autorzy [23] porównali metodę SpectralEnsemble z algorytmem K-średnich oraz trzema innymi algorytmami klasteryzacji zespołowej, wykorzystując 19 zbiorów danych rzeczywistych oraz wyznaczając wskaźnik zgodności partycji. Rozważana metoda w przypadku 14 wybranych zbiorów danych okazała się najlepsza. Była także drugą najlepszą metodą w 4 przypadkach.

2.2.5 HierarchicalEnsemble

W [7] zamiast algorytmu klasteryzacji spektralnej zastosowano aglomeracyjne grupowanie hierarchiczne z różnymi funkcjami łączącymi. Metoda aglomeracyjna, jak zostało to wspomniane w podrozdziale 1.2.3, wykorzystuje macierz niepodobieństwa. W tej roli wykorzystana jest macierz $W^{'}$ wyznaczona jako:

$$w_{ij}^{'} = 1 - w_{ij},\ 1 \leqslant i,\ j \leqslant N, \tag{2.7}$$

gdzie w_{ij} są określone jak w Definicji 2.1.

Algorytm 10 przedstawia kolejne kroki metody HierarchicalEnsemble.

Algorytm 10 HierarchicalEnsemble

Dane wejściowe: W — macierz współwystępowania, K — liczba klastrów

1: Wyznacz macierz $W^{'}$ zgodnie z równaniem (2.7);
2: Zainicjalizuj algorytm grupowania aglomeracyjnego z K klastrami, odległością średnią i macierzą $W^{'}$;
3: Zwróć uzyskaną partycję.

W tej pracy do łączenia klastrów wykorzystywana będzie tylko odległość średnia (ang. *average linkage*). Metoda HierarchicalEnsemble wymaga podania docelowej liczby klastrów.

Podobnie jak w przypadku SpectralEnsemble, metoda ta charakteryzuje się dużą złożonością obliczeniową — w tym przypadku jest to konsekwencją wykorzystania algorytmu grupowania hierarchicznego.

W [7] dokonano porównania algorytmu HierarchicalEnsemble z trzema metodami klasteryzacji zespołowej (oparte o teorię grafów algorytmy CSPA, MCLA, HGPA) oraz z metodami K-średnich, grupowania aglomeracyjnego z funkcjami łączącymi *single linkage* oraz *average linkage* i algorytmem klasteryzacji spektralnej. Zastosowanym kryterium oceny jakości analizy skupień był wskaźnik zgodności partycji, a samą analizę porównawczą przeprowadzoną dla 9 wybranych zbiorów danych rzeczywistych i syntetycznych. Metoda HierarchicalEnsemble najczęściej poradziła sobie lepiej niż pozostałe metody grupowania zespołowego (w wielu przypadkach znacząco). W przypadku podstawowych metod analizy skupień, poza algorytmem klasteryzacji spektralnej, wyniki były gorsze niż dla metody HierarchicalEnsemble prawie za każdym razem.

2.2.6 GraphConsensus

Metoda GraphConsensus została przedstawiona w [26]. Opiera się ona na wyznaczeniu macierzy G, gdzie:

$$g_{ij} = \begin{cases} 1, \, w_{ij} \geqslant t, \\ 0, \, w_{ij} < t, \end{cases} \tag{2.8}$$

gdzie w_{ij} są określone jak w Definicji 2.1, a $t > 0$ to ustalona wartość progowa (ang. *threshold*). Tak utworzoną macierz G traktujemy jako macierz sąsiedztwa pewnego grafu. Od tego momentu problem klasteryzacji zmienia się w zagadnienie detekcji społeczności w grafie (ang. *community detection*).

Zadaniem będzie wyszukanie społeczności k-klik (ang. *k-cliques community*), czyli połączenia wszystkich k-klik (podgrafów pełnych o rozmiarze k), które można odwiedzić, przechodząc z sąsiedniej na sąsiednią. Przypomnijmy, że k-kliki są sąsiednie, jeżeli współdzielą $k-1$ elementów. Powstałe społeczności wyznaczają podział na skupienia. Sposób wyszukiwania społeczności k-klik został dokładniej omówiony w [25]. Algorytm 11 przedstawia schemat działania metody GraphConsensus.

Algorytm 11 GraphConsensus

Dane wejściowe: W — macierz współwystępowania, $t \in (0, 1)$ — wartość progowa

1: Wyznacz macierz binarną G, korzystając ze wzoru (2.8);
2: Przyjmij macierz G jako macierz sąsiedztwa pewnego grafu;
3: Wyznacz społeczności k-klik i potraktuj je jako podział na skupienia.

W odróżnieniu od wcześniej omówionych metod, w przypadku algorytmu GraphConsensus nie można wybrać docelowej liczby klastrów, które powinien on wykryć. Można się domyślać, że dobór wartości progowej będzie mieć istotny wpływ na to, jak liczne będą wykryte skupienia — im będzie większa, tym większego rozdrobnienia skupień można oczekiwać.

Skuteczność metody GraphConsensus nie została sprawdzona przez autorów [26]. Nie podali oni również żadnych wskazówek dotyczących doboru wartości progowej. Algorytmowi przyjrzymy się dokładniej w podrozdziale 3.2.2.

2.2.7 CoAssocEnsemble

W [6] zaprezentowany został jeszcze inny sposób wykorzystania macierzy współwystępowania W (p. Definicja 2.1). Dla ustalonej wartości progowej t i dla każdej pary

obserwacji (i, j) łączymy te obserwacje w jeden klaster, jeśli $w_{ij} \geqslant t$. W przypadku, gdy obserwacje zostały już wcześniej umieszczone w odrębnych klastrach, łączymy te klastry ze sobą. Ponadto, gdy któraś z obserwacji nie zostanie dołączona do żadnej z utworzonych grup, tworzy własne skupienie. W przypadku tego algorytmu tak, jak dla GraphConsensus, nie jest możliwe ustalenie z góry docelowej liczby klastrów. Algorytm 12 przedstawia schemat działania metody CoAssocEnsemble.

Algorytm 12 CoAssocEnsemble

Dane wejściowe: W — macierz współwystępowania, $t \in (0, 1)$ — wartość progowa

$I \leftarrow \{(i, j) : 1 \leqslant i, j \leqslant N, i \neq j\}$
for all (i, j) in I **do**
    if $w_{ij} \geqslant t$ **then**
        Połącz obserwacje i oraz j w jeden klaster;
        Jeśli obserwacje znajdują się w różnych skupieniach, połącz te skupienia ze sobą;
    end if
end for
Z obserwacji nieprzyporządkowanych do żadnego klastra stwórz osobne grupy;
Zwróć uzyskaną partycję.

Metoda CoAssocEnsemble będzie niewątpliwie wrażliwa na wybór wartości progowej t. W [6] wartość *threshold* ustalona jest zawsze na poziomie 0,5. Wpływ wyboru wartości t zostanie pokazany w podrozdziale 3.2.2.

W [6] pokazano, wizualizując uzyskane grupowanie danych syntetycznych, że za pomocą metody CoAssocEnsemble można uzyskać rezultaty lepsze niż za pomocą metody K-średnich.

Rozdział 3

Studium przypadku

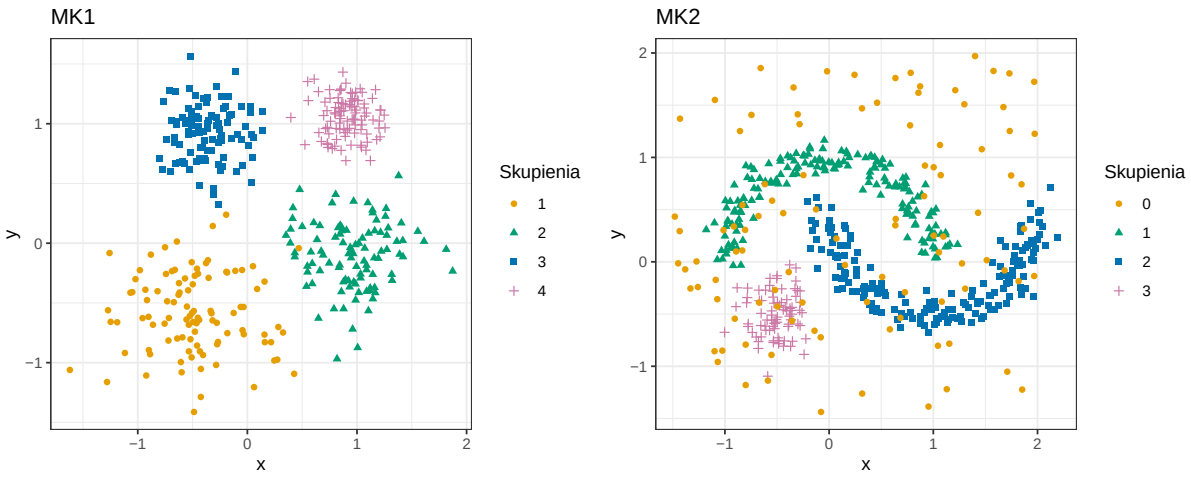
W tym rozdziale przedstawimy studium przypadku (ang. *case study*) w oparciu o wygenerowane dane syntetyczne, co pozwoli nam zilustrować najważniejsze własności omawianych algorytmów klasteryzacji zespołowej. Wybrane na potrzeby tych rozważań dane oczywiście nie pozwalają na szczegółową analizę porównawczą wybranych metod, ale dobrze ilustrują potencjalne korzyści wynikające ze stosowania algorytmów grupowania zespołowego. Dokładniejsza analiza efektywności wybranych metod zostanie przeprowadzona w rozdziale 4.

W studium przypadku uwzględnione zostaną wszystkie metody klasteryzacji zespołowej przedstawione w rozdziale 2, jak i standardowe metody grupowania omówione w rozdziale 1.

Dla każdej metody klasteryzacji zespołowej do uzyskania poszczególnych partycji użyty został algorytm K-średnich.

3.1 Wykorzystane dane

Do porównania metod zostaną wykorzystane dwa dwuwymiarowe, syntetyczne zbiory danych. Od tego miejsca określane będą jako **MK1** oraz **MK2**. Zostały one przedstawione na Rysunku 3.1.



Rysunek 3.1: Zbiory danych wykorzystane w studium przypadku.
Źródło: opracowanie własne.

Zbiór **MK1** to cztery łatwo separowalne skupiska punktów wygenerowanych z odpowiednich dwuwymiarowych rozkładów gaussowskich. Każde skupienie składa się ze 100 obserwacji. Poprawna klasteryzacja tego zbioru może być traktowana jako minimalny wymóg dla dowolnego algorytmu analizy skupień.

Zbiór **MK2** składa się z trzech skupisk — jednej grupy punktów wygenerowanych z dwuwymiarowego rozkładu gaussowskiego (liczącej 80 obserwacji) i dwóch, które składają się na charakterystyczne półksiężyce (każde z tych skupisk liczy po 175 obiektów). Dodatkowo dodane zostało jeszcze 100 punktów z rozkładu jednostajnego, które będą stanowić szum losowy. Z racji wklęsłości kształtów niektórych skupień zbiór stanowi większe wyzwanie i możemy spodziewać się, że niektóre algorytmy nie poradzą sobie z w pełni poprawnym grupowaniem.

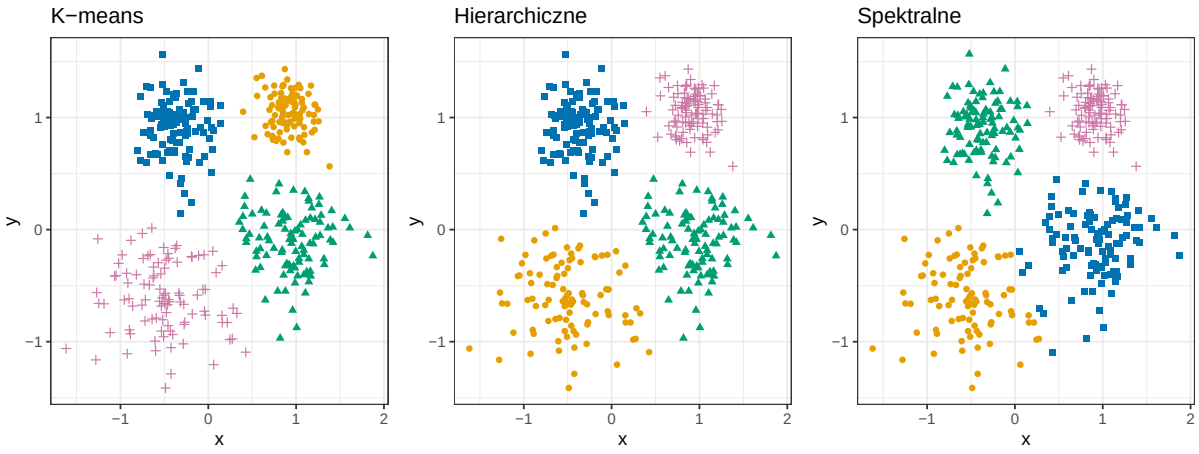
3.2 Wyniki analizy skupień

Przyjrzyjmy się teraz wynikom analizy skupień — najpierw w przypadku podstawowych algorytmów klasteryzacji, a następnie zespołowych metod grupowania.

Dla ułatwienia przeprowadzenia analizy będziemy zakładać, że docelowa liczba klastrów jest znana.

3.2.1 Podstawowe algorytmy klasteryzacji

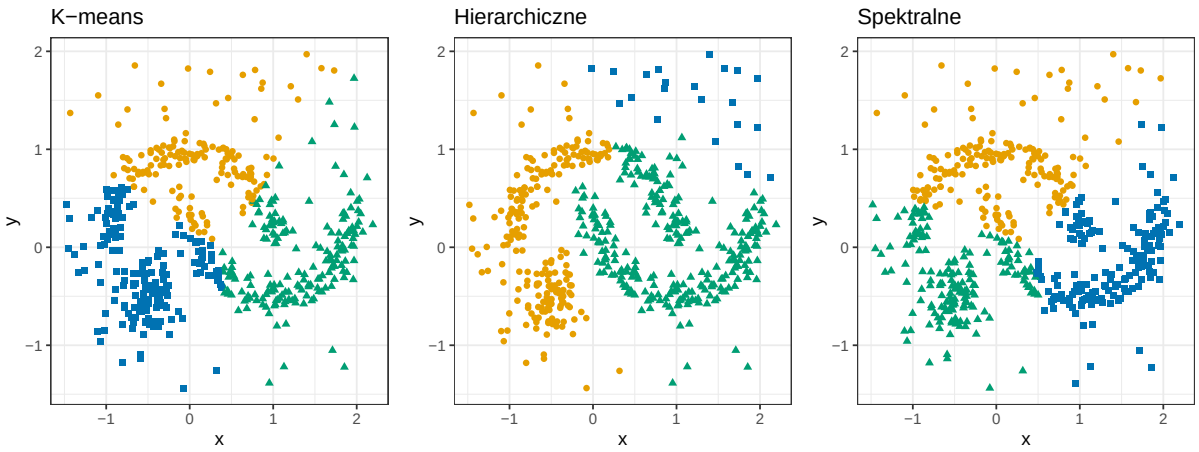
W tej części przedstawione zostaną wyniki dla algorytmów: K-średnich, grupowania hierarchicznego z funkcją łączącą *average linkage* oraz klasteryzacji spektralnej. Tak jak pokazuje Rys.3.2, każdy algorytm poprawnie zidentyfikował wszystkie skupienia w zbiorze **MK1**, a wyniki są do siebie bardzo zbliżone.



Rysunek 3.2: Wyniki klasteryzacji dla zbioru **MK1** .
Źródło: opracowanie własne.

Inaczej wygląda to w przypadku klasteryzacji **MK2**, której rezultaty zostały graficznie przedstawione na Rysunku 3.3. Żaden algorytm nie poradził sobie z poprawnym rozróżnieniem skupień o kształcie półksiężycy. Algorytm K-średnich, tak jak zostało to podkreślone w podrozdziale 1.2.1, nie poradzi sobie z rozpoznaniem skupisk, które nie są wypukłe. Jednakże algorytmom grupowania hierarchicznego oraz klasteryzacji spektralnej, pomimo braku ograniczeń dotyczących struktury danych, również nie udało się poprawnie

rozpoznać skupisk. Rozważane algorytmy nie mają także wbudowanego mechanizmu pozwalającego na wyodrębnienie obserwacji uznawanych za szum — każdy punkt musi być przyporządkowany do jednego z klastrów.



Rysunek 3.3: Wyniki klasteryzacji dla zbioru MK2.
Źródło: opracowanie własne

Powyższe wyniki, które można uznać za nieudane, rodzą pytanie, czy w przypadku zbioru **MK2** w ogóle możliwa będzie prawidłowa identyfikacja struktury danych. W następnym kroku przyjrzymy się, jak z tym zadaniem poradzą sobie metody grupowania zespołowego.

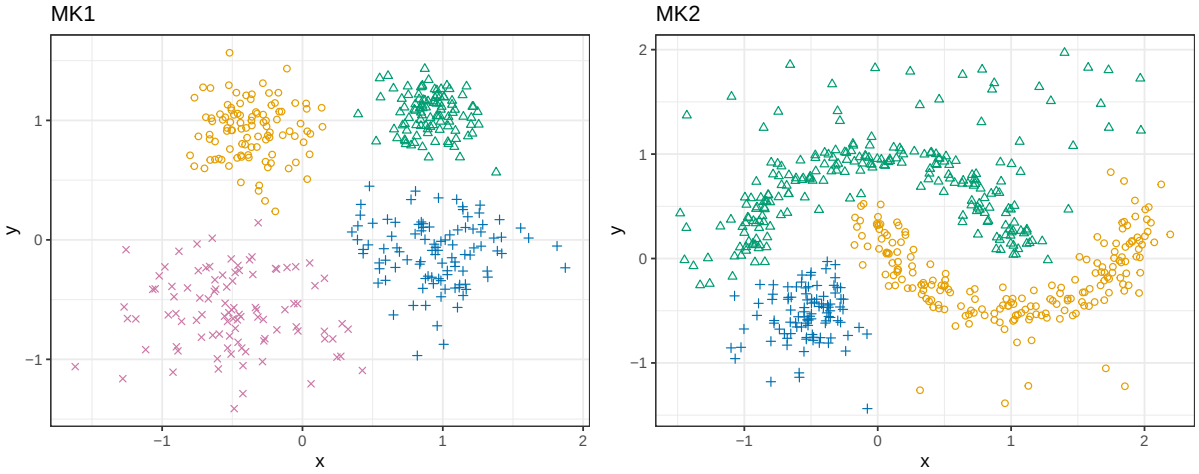
3.2.2 Algorytmy klasteryzacji zespołowej

Rezultaty przedstawione w tej sekcji zostały wygenerowane na bazie 30 replikacji bootstrapowych w przypadku algorytmów BaggedMajority i BaggedEnsemble (p. podrozdział 2.2.1) oraz 100 składowych partycji w przypadku pozostałych algorytmów (p. podrozdział 2.2.4).

W przypadku każdej metody liczba skupień dla bazowego algorytmu K-średnich została wybrana na podstawie wizualnego porównania rezultatów otrzymywanych dla różnych wartości K. Dokładniejszy wpływ doboru liczby klastrów zostanie zbadany w podrozdziale 4.3.

SpectralEnsemble

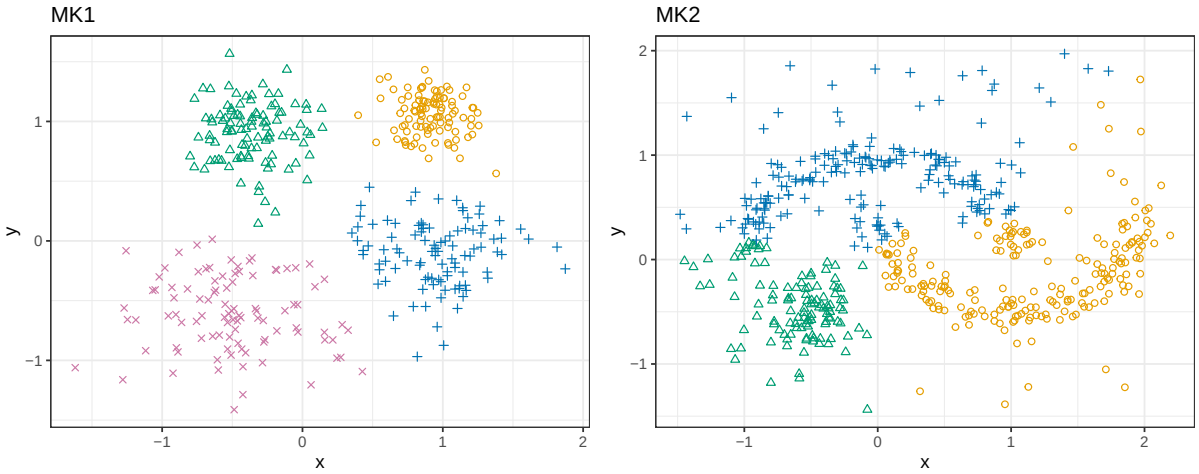
Wyniki dla tego algorytmu zostały przedstawione na Rys.3.4. Otrzymane rezultaty są bardzo zadowalające — dla obu zbiorów danych uzyskano poprawny wynik grupowania, w przypadku **MK2** oba „półksiężyce” oraz skupisko o kształcie sferycznym są dobrze odseparowane. Wyniki uzyskano, przyjmując dla algorytmu bazowego K-średnich liczbę skupień równą 15.



Rysunek 3.4: Wyniki grupowania dla algorytmu SpectralEnsemble.
Źródło: opracowanie własne.

KModesEnsemble

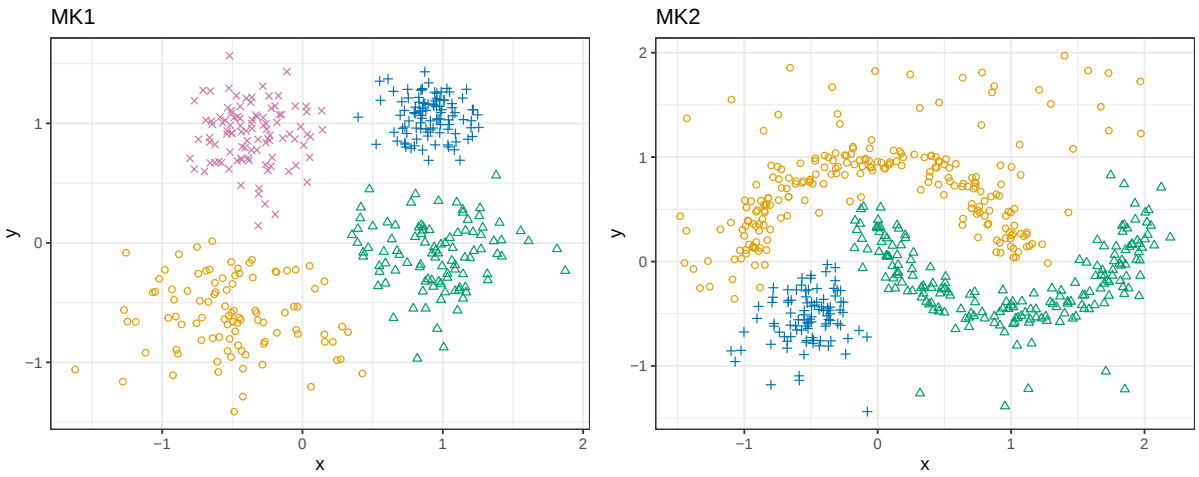
Wyniki klasteryzacji przedstawione są na Rys.3.5. Dla **MK1** są one w pełni zadowalające. W tym przypadku algorytm K-średnich został zainicjalizowany z liczbą klastrów równą 10. Wyniki dla **MK2** nie są już idealne — błędy są zauważalne w przypadku niektórych punktów brzegowych należących do skupisk o kształtach półksiężycy. Jednak nadal są one lepsze od tych uzyskanych przez podstawowe metody analizy skupień. Algorytm K-means w tym przypadku szukał pięciu skupień.



Rysunek 3.5: Wyniki grupowania dla algorytmu KModesEnsemble.
Źródło: opracowanie własne.

HierarchicalEnsemble

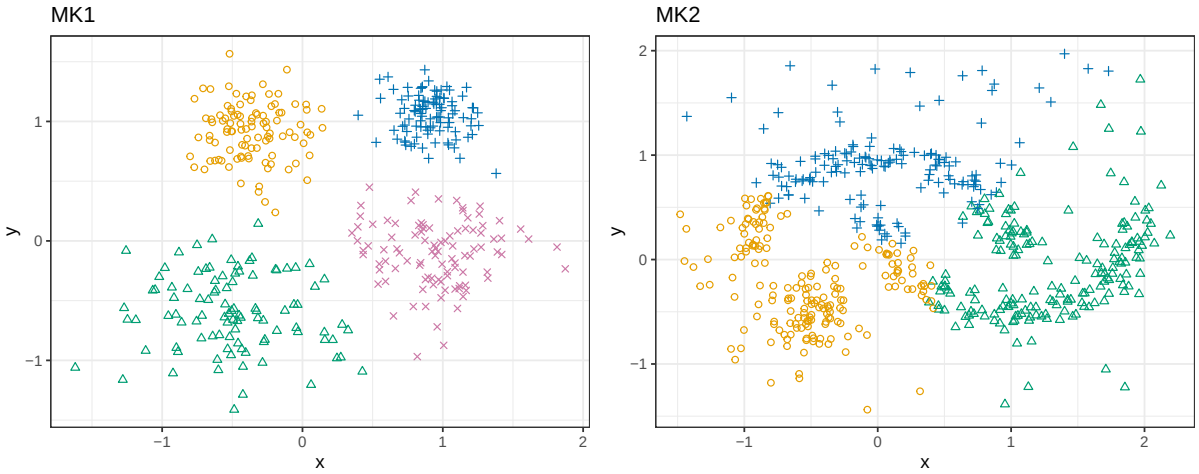
W przypadku metody HierarchicalEnsemble algorytm K-średnich został uruchomiony z kolejno 10 i 15 klastrami. Uzyskane wyniki, przedstawione na Rys.3.6, są w pełni zadowalające w obu przypadkach.



Rysunek 3.6: Wyniki grupowania dla algorytmu HierarchicalEnsemble.
Źródło: opracowanie własne.

BaggedMajority

Rezultaty dla algorytmu BaggedMajority zostały przedstawione na Rys.3.7. Podobnie jak we wcześniejszych przypadkach grupowanie zbioru **MK1** zakończyło się sukcesem. Wynik klasteryzacji danych **MK2** nie jest jednak poprawny. Można zauważyć, że otrzymany podział na skupienia jest zbliżony do wyniku uzyskanego przez pojedyncze zastosowanie algorytmu K-średnich, co może wskazywać na silne uzależnienie BaggedMajority od algorytmu bazowego.

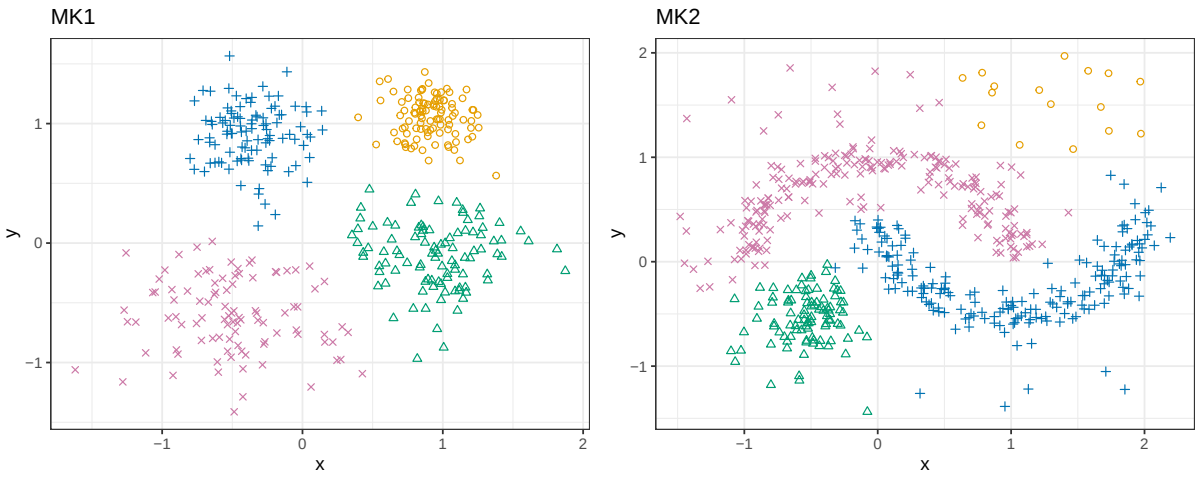


Rysunek 3.7: Wyniki grupowania dla algorytmu BaggedMajority.
Źródło: opracowanie własne.

GraphClosure

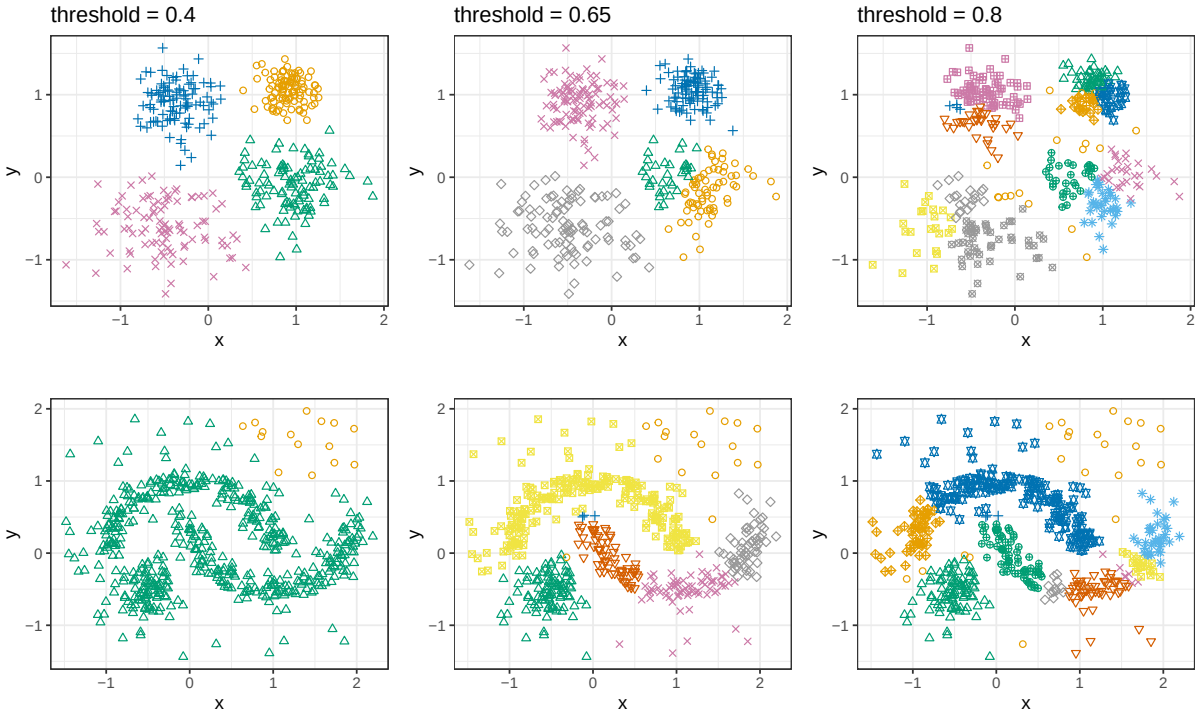
Teraz przejdziemy do rezultatów uzyskanych przez algorytm GraphClosure. W jego przypadku nie mamy możliwości ustalenia docelowej liczby skupień — może ona w istotny sposób zależeć od wyboru wartości progowej. Wyniki, przedstawione na Rysunku 3.8,

zarówno dla danych **MK1**, jak i **MK2**, są satysfakcjonujące. Uzyskano je, przyjmując wartość progową równą 0,5.



Rysunek 3.8: Wyniki grupowania dla algorytmu GraphClosure.
Źródło: opracowanie własne.

Metoda GraphClosure jest jednak wrażliwa na zmiany tego hiperparametru, co pokazuje Rys.3.9. Mniejsza wartość *threshold* prowadzi do „sklejenia" ze sobą wszystkich klastrów zbioru **MK2**. Wartości większe od 0,5 skutkują natomiast coraz większym rozdrobnieniem skupisk. Można skłonić się do wniosku, że GraphClosure jest algorytmem, który pozwala na uzyskanie bardzo dobrych wyników, wymagającym jednak trudu włożonego w odpowiedni dobór wartości hiperparametrów.

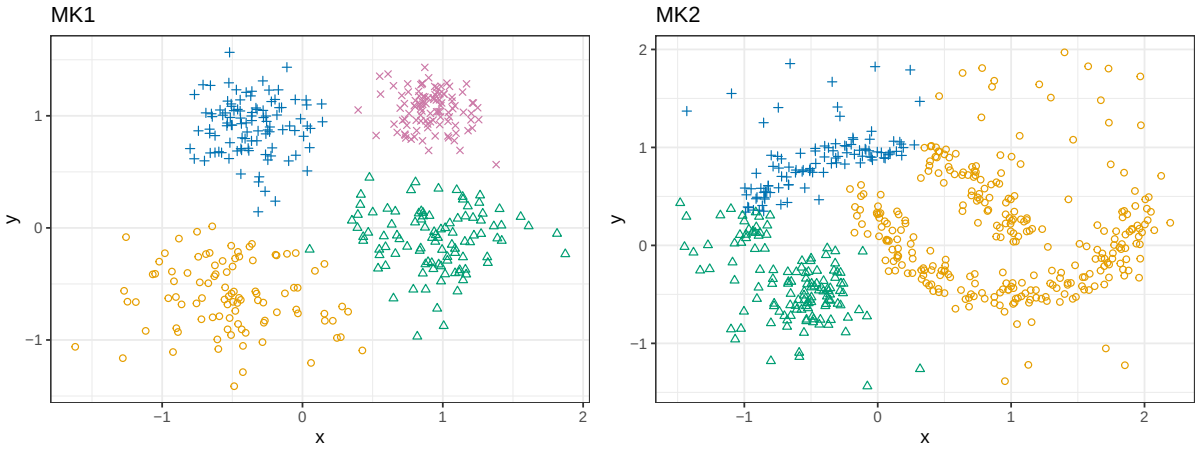


Rysunek 3.9: Zależność wyników grupowania na bazie metody GraphClosure od wyboru wartości progowej (*threshold*).
Źródło: opracowanie własne.

BaggedEnsemble

Wyniki dla metody BaggedEnsemble zostały przedstawione na Rysunku 3.10. Przyjęta liczba bazowych centrów to, odpowiednio: 15 - w przypadku zbioru **MK1** i 20 w przypadku danych **MK2**. Wyniki dla zbioru **MK1** są prawidłowe, natomiast w przypadku **MK2** otrzymany podział na skupienia nie odpowiada prawdziwej strukturze tych danych.

Metoda BaggedEnsemble tak, jak zostało to omówione w podrozdziale 2.2.2, opiera się na algorytmach K-średnich oraz grupowania hierarchicznego, które nie poradziły sobie z poprawną identyfikacją skupień. Może to być więc przyczyną uzyskania tak niesatysfakcjonujących rezultatów. Co więcej, zastosowanie innych wartości hiperparametrów nie wpłynęło pozytywnie na otrzymywane rezultaty. W podrozdziale 4.3.2 przyjrzymy się dokładniej, jak na otrzymywane wyniki wpływa zmiana liczby centrów czy replikacji bootstrapowych.

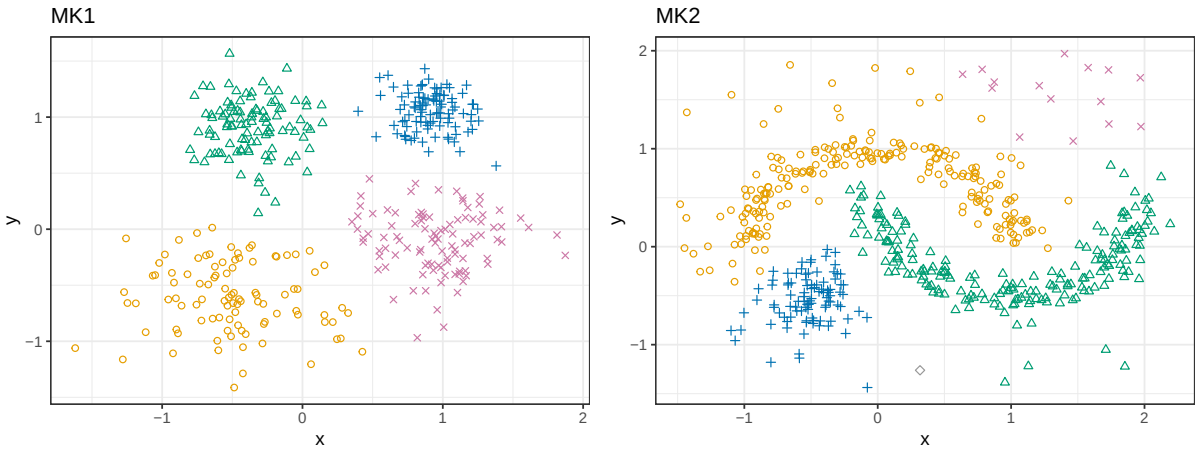


Rysunek 3.10: Wyniki dla algorytmu BaggedEnsemble.
Źródło: opracowanie własne.

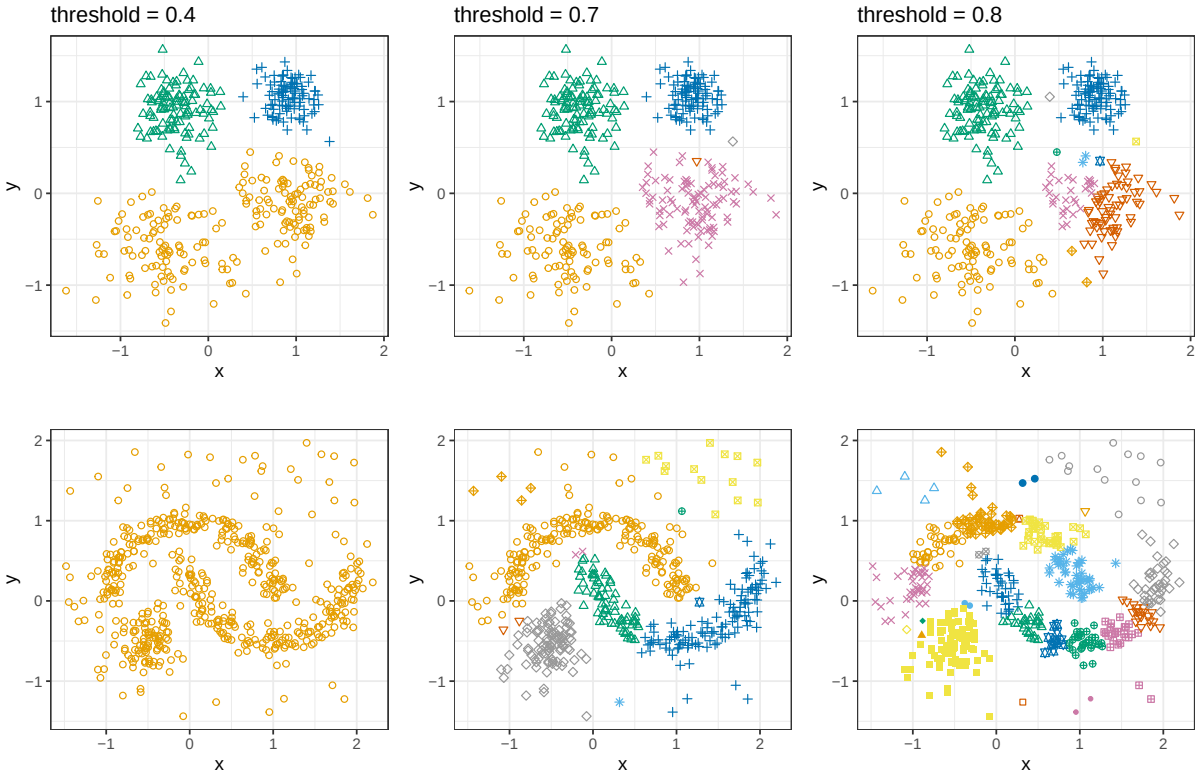
CoAssocEnsemble

Jako ostatnia przedstawiona zostanie metoda CoAssocEnsemble. Podobnie jak w przypadku metody GraphClosure nie mamy możliwości bezpośredniego ustalenia docelowej liczby skupień. Można to uzyskać, wybierając odpowiednią wartość progową (ang. *threshold*). Otrzymane dla algorytmu CoAssocEnsemble wyniki przedstawia Rysunek 3.11. Wartość progową została ustalona na poziomie 0,6. Uzyskane skupienia dobrze pokrywają się z rzeczywistymi. Zauważalna jest wrażliwość algorytmu na szum — spośród punktów do niego należących wyróżnił dodatkowe dwa skupiska, w tym jedno składające się z jednej obserwacji.

Podobnie jak dla algorytmu GraphClosure działanie metody CoAssocEnsemble jest mocno uzależnione od wyboru wartości progowej, co dobrze pokazuje Rys.3.12. Dla wartości *threshold* na poziomie 0,4 otrzymaliśmy mniej skupień niż rzeczywiście jest — odpowiednio 3 i 1. Dla wartości progowych większych od 0,6 obserwujemy coraz większe rozdrobnienie klastrów. Dla zbioru **MK2** i dla *threshold* na poziomie 0,8 uzyskana liczba klastrów to aż 29. Możemy więc wysnuć wniosek, że dla tej metody również koniecznością staje się odpowiedni dobór wartości progowej dla ustalonych danych.



Rysunek 3.11: Wyniki grupowania dla algorytmu CoAssocEnsemble.
Źródło: opracowanie własne.



Rysunek 3.12: Zależność wyników grupowania na bazie metody CoAssocEnsemble od wyboru wartości progowej (*threshold*).
Źródło: opracowanie własne.

3.3 Podsumowanie

W tym rozdziale przedstawiliśmy uzyskane rezultaty analizy skupień przeprowadzonej dla dwóch syntetycznych zbiorów danych. Zastosowano zarówno standardowe algorytmy klasteryzacji, jak i metody grupowania zespołowego. W przypadku zbioru **MK1** wyniki

uzyskane przez wszystkie rozważane algorytmy były satysfakcjonujące. Różnice uwidoczniły się dopiero dla danych **MK2**, na które składały się słabo odseparowane od siebie skupienia, z których dwa charakteryzowały się wklęsłym kształtem. Żadna z podstawowych metod analizy skupień nie poradziła sobie z poprawnym wykryciem struktury danych, natomiast aż cztery algorytmy grupowania zespołowego poprawnie wyróżniło każde z trzech skupień. Rezultaty te pokazały potencjał i korzyści wynikające z zastosowania omawianych w pracy metod. Najbardziej obiecujące wydają się algorytmy SpectralEnsemble, HierarchicalEnsemble, GraphConsensus oraz CoAssosEnsemble. Wyróżnić również można metodę KModesEnsemble, dla której wynik, choć nie w pełni poprawny, był wyraźnie lepszy niż w przypadku standardowych algorytmów klasteryzacji.

Ponadto na podstawie przeprowadzonych rozważań można dojść do wniosku, że potrzebne jest zbadanie wpływu hiperparametrów na skuteczność algorytmów. Szczególnie mocno uwidoczniło się to w przypadku metod GraphConsenus oraz CoAssocEnsemble. Duży wpływ na uzyskiwane rezultaty miał w ich przypadku dobór wartości progowej (ang. *threshold*). Jednak i w przypadku innych metod można się zastanawiać, jak na otrzymywane wyniki może wpłynąć zmiana liczby skupień w bazowej metodzie K-średnich czy zmienianie liczby partycji. Takie rozważania zostaną podjęte w podrozdziale 4.3.

Rozdział 4

Porównanie algorytmów

Przedstawimy teraz szczegółowe porównanie efektywności rozważanych algorytmów grupowania zespołowego, wykorzystując w tym celu odpowiednio zróżnicowany zestaw danych testowych oraz wybrane kryteria walidacyjne, w tym zarówno wskaźniki wewnętrzne, jak i zewnętrzne. W analizie porównawczej nie będą uwzględnione algorytmy GraphClosure i MajorityVote, jako że nie pozwalają one na wybranie z góry liczby otrzymanych skupień. Tak jak zauważyliśmy w rozdziale 3, pozostałe metody mają duży potencjał, są różnorodne i stanowią dobry przekrój metod klasteryzacji zespołowej. Jako punkt odniesienia w analizach traktować będziemy rezultaty uzyskane przy pomocy algorytmu K-średnich.

4.1 Wykorzystane dane

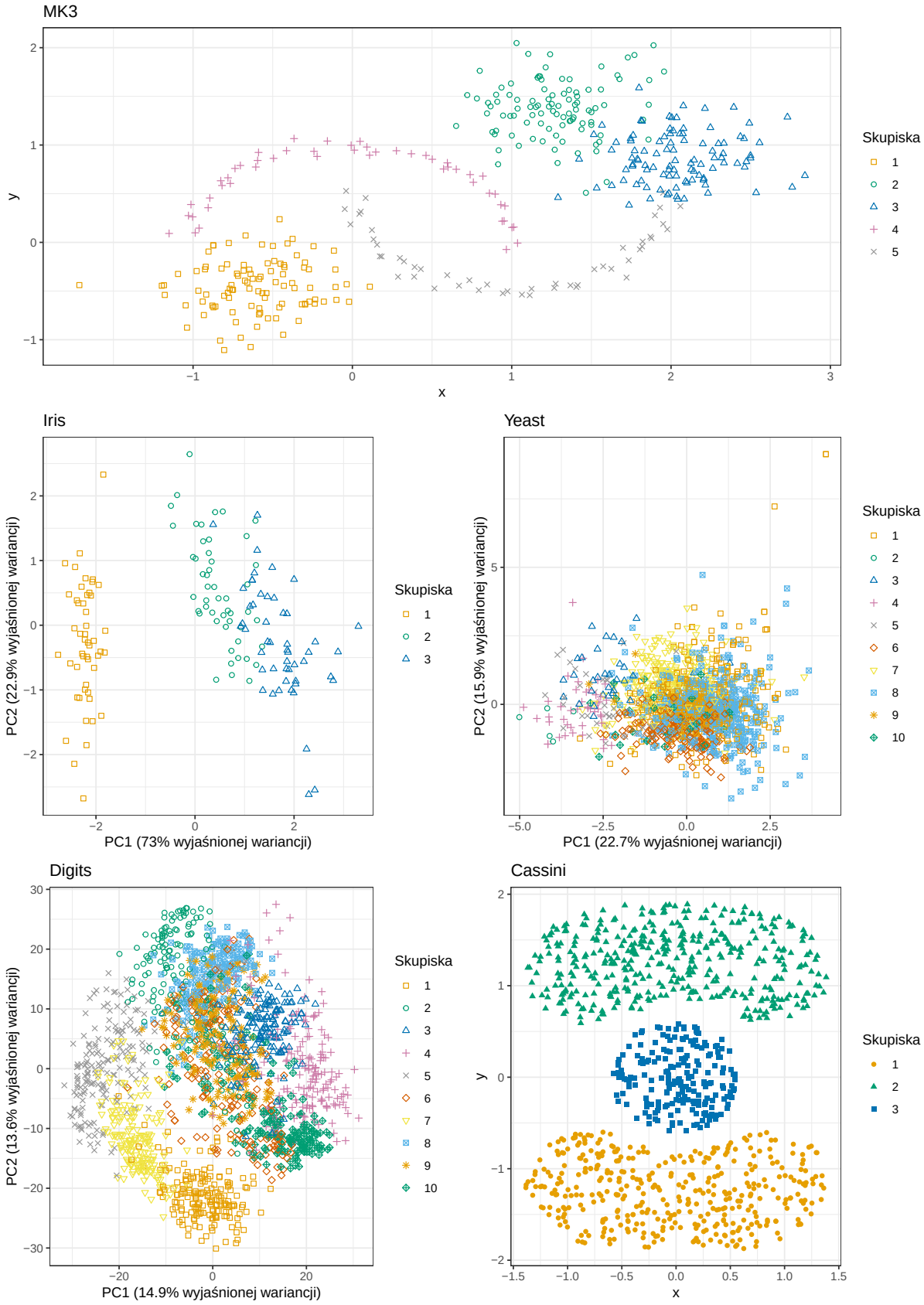
W celu porównania ze sobą algorytmów zostało wybranych pięć różnych zbiorów danych. Są to:

- **iris** — dane opisujące różne gatunki irysów,
- **digits** — dane składające się z 8x8 pikselowych obrazków z ręcznie zapisanymi cyframi,
- **yeast** — dane opisujące zależność między lokalizacjami białek w komórkach drożdży a pewnymi ich parametrami,
- **Cassini** — syntetyczne dane dostępne w pakiecie *mlbench* języka R (p. [22]),
- **MK3** — syntetyczne dane przygotowane przez autora pracy.

Pierwsze trzy zbiory danych są publicznie dostępne w repozytorium UCI (p. [4]).

Dokładniejsze informacje o wykorzystanych danych przedstawione są w Tabeli 4.1, a Rysunek 4.1 przedstawia odpowiadające im dwuwymiarowe wykresy rozrzutu. W przypadku danych więcej niż dwuwymiarowych zastosowana została metoda składowych głównych (ang. *principal components analysis*, PCA). Przekształca ona początkowe zmienne w ich ortogonalne, liniowe kombinacje, które wyjaśniają największą możliwą frakcję całkowitej zmienności danych. W przypadku zastosowania metody PCA na osiach wykresu podano również frakcję wyjaśnionej wariancji.

Taki dobór danych pozwoli nam zbadać, jak algorytmy poradzą sobie z różną liczbą obserwacji, zmiennych oraz skupień. Jak łatwo można zauważyć w Tabeli 4.1, te trzy charakterystyki danych różnią się na przestrzeni wybranych zbiorów.



Rysunek 4.1: Dwuwymiarowe wykresy rozrzutu analizowanych danych. W przypadku zbiorów **iris**, **yeast** i **digits** zastosowana została metoda PCA.
Źródło: opracowanie własne.

Tabela 4.1: Informacje o użytych zbiorach danych.

Zbiór danych	Liczba obserwacji	Liczba cech	Liczba klastrów
Iris	150	4	3
Cassini	1000	2	3
Yeast	1484	8	10
Digits	1797	64	10
MK3	400	2	5

Źródło: opracowanie własne.

4.2 Analiza porównawcza

Dla każdego zbioru danych i każdego algorytmu proces klasteryzacji zostanie powtórzony 20 razy. Posłuży to uśrednieniu uzyskiwanych wyników, co ułatwi wyciąganie wniosków. W każdym powtórzeniu poszczególne partycje uzyskiwać będziemy, wykorzystując losowo zainicjalizowany algorytm K-średnich z ustaloną liczbą klastrów. Dokładne informacje o dobrze hiperparametrów znaleźć można w kodach źródłowych, które są dostępne w repozytorium wspomnianym w Dodatku.

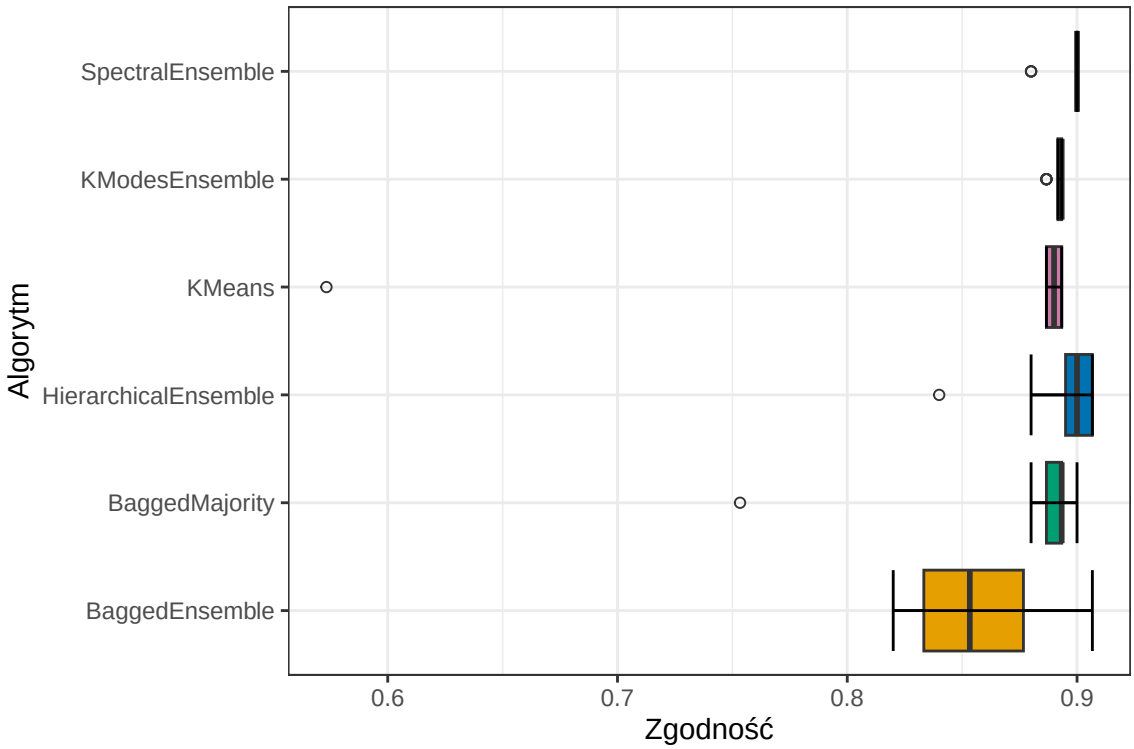
Podczas każdej iteracji zostaną wyznaczone wartości wybranych wskaźników walidacyjnych (patrz podrozdział 1.3): zgodności partycji końcowej z rzeczywistymi klasami, skorygowany wskaźnik Randa, wskaźnik silhouette i spójności. Dodatkowo zmierzony zostanie czas działania algorytmu. W analizie uwzględnimy także ocenę powtarzalności otrzymanych wyników grupowania, wyznaczając w tym celu wskaźnik stabilności zgodnie ze schematem przedstawionym w podrozdziale 1.3.1.

W każdym badanym przypadku wyniki dla kryterium zgodności zostaną przedstawione na wykresach pudełkowych, a także wraz z innymi wskaźnikami w tabelach. Będą one zawierać informacje o średnich i odchyleniach standardowych otrzymanych wyników (podane w nawiasach).

Często, jeszcze przed zastosowaniem algorytmu klasteryzacji, przeprowadza się standaryzację danych. Zaleca się ją zastosować, gdy mamy pomiary w różnych jednostkach, bądź jeśli wariancje wyjściowych zmiennych są bardzo zróżnicowane. W przypadku wybranych zbiorów danych zmienne charakteryzują się zbliżoną zmiennością, a jak się okazało wyniki otrzymane po standaryzacji danych były gorsze od tych, gdy tej procedury nie zastosowano. Stąd w analizie porównawczej przedstawione wyniki wyznaczone zostały dla danych niepoddanych standaryzacji.

4.2.1 Dane iris

Otrzymane wartości zgodności partycji zostały przedstawione na Rysunku 4.2. Można na nim zauważyć, że większość metod poradziła sobie w zbliżony sposób. Pudełka w przypadku algorytmów KModesEnsemble oraz SpectralEnsemble są bardzo wąskie, więc wartości zgodności dla tych algorytmów mają mały rozrzut. Zupełnie odmienny rezultat widzimy dla BaggedEnsemble, który poradził sobie najgorzej. Dla tej metody otrzymujemy największy rozrzut, co może świadczyć o jej ograniczonej stabilności.



Rysunek 4.2: Zgodności partycji w przypadku zbioru **iris**.
Źródło: opracowanie własne.

Tabela 4.2 przedstawia wartości rozważanych kryteriów walidacyjnych. Wszystkie metody poza BaggedEnsemble uzyskały średnio lepszy wynik zgodności oraz wskaźnika Randa od metody K-średnich. Wartości wskaźnika silhouette są w większości przypadków większe lub na podobnym poziomie co algorytmu referencyjnego. Otrzymane partycje końcowe poza wynikami uzyskanymi dla metod opartych na bootstrapowych replikacjach (tzn. algorytmy BaggedEnsemble i BaggedMajority), cechują się również większą spójnością.

Możemy więc stwierdzić, że zbiór **iris** nie stanowi dużego wyzwania dla algorytmów analizy skupień. Otrzymane wyniki pokazują, że badane algorytmy zespołowe bez większego problemu poradziły sobie z pogrupowaniem tych danych.

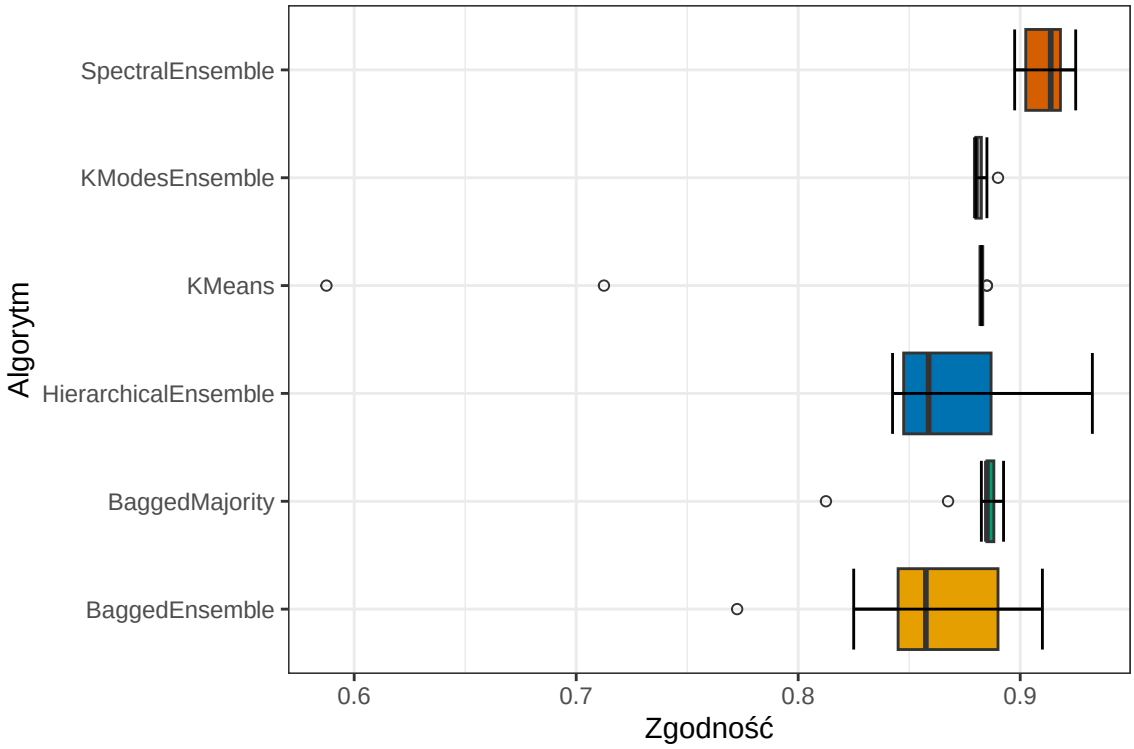
Tabela 4.2: Wyniki wskaźników oceny klasteryzacji dla zbioru **iris**.

	Zgodność	Wskaźnik Randa	Silhouette	Spójność
KMeans	0.874 (±0.071)	0.709 (±0.066)	0.550 (±0.008)	10.704 (±0.545)
KModesEnsemble	0.892 (±0.003)	0.727 (±0.006)	0.552 (±0.001)	10.519 (±0.135)
SpectralEnsemble	0.897 (±0.007)	0.740 (±0.014)	0.555 (±0.001)	6.331 (±2.164)
BaggedEnsemble	0.859 (±0.029)	0.676 (±0.048)	0.523 (±0.021)	11.251 (±3.679)
BaggedMajority	0.885 (±0.031)	0.719 (±0.039)	0.544 (±0.029)	14.373 (±11.695)
HierarchicalEnsemble	0.896 (±0.017)	0.738 (±0.030)	0.552 (±0.009)	7.998 (±2.302)

Źródło: opracowanie własne.

4.2.2 Dane MK3

Wyniki zgodności dla zbioru **MK3** przedstawione zostały na Rys.4.3. Natychmiastowo uwagę zwraca algorytm SpectralEnsemble, któremu udało się uzyskać najlepsze wyniki ze wszystkich metod. Zauważamy również, że metody HierarchicalEnsemble oraz BaggedEnsemble cechują się dużym rozrzutem.



Rysunek 4.3: Zgodności partycji w przypadku zbioru **MK3**.
Źródło: opracowanie własne.

Tabela 4.3 przedstawia dokładniejsze wyniki kryteriów oceny klasteryzacji. Możemy zauważyć, że otrzymane skorygowane wskaźniki Randa oraz wskaźniki zgodności dla algorytmów zespołowych są lepsze niż dla referencyjnej metody K-średnich. Z drugiej strony, wewnętrzne kryteria walidacyjne nie pozwalają na wyciągnięcie jednoznacznych wniosków. Na pewno warto jednak odnotować dużą wartość zarówno średniej, jak i odchylenia standardowego dla wskaźnika spójności w przypadku algorytmu BaggedMajority. Sytuacja ta powtarza się dla każdego zbioru danych.

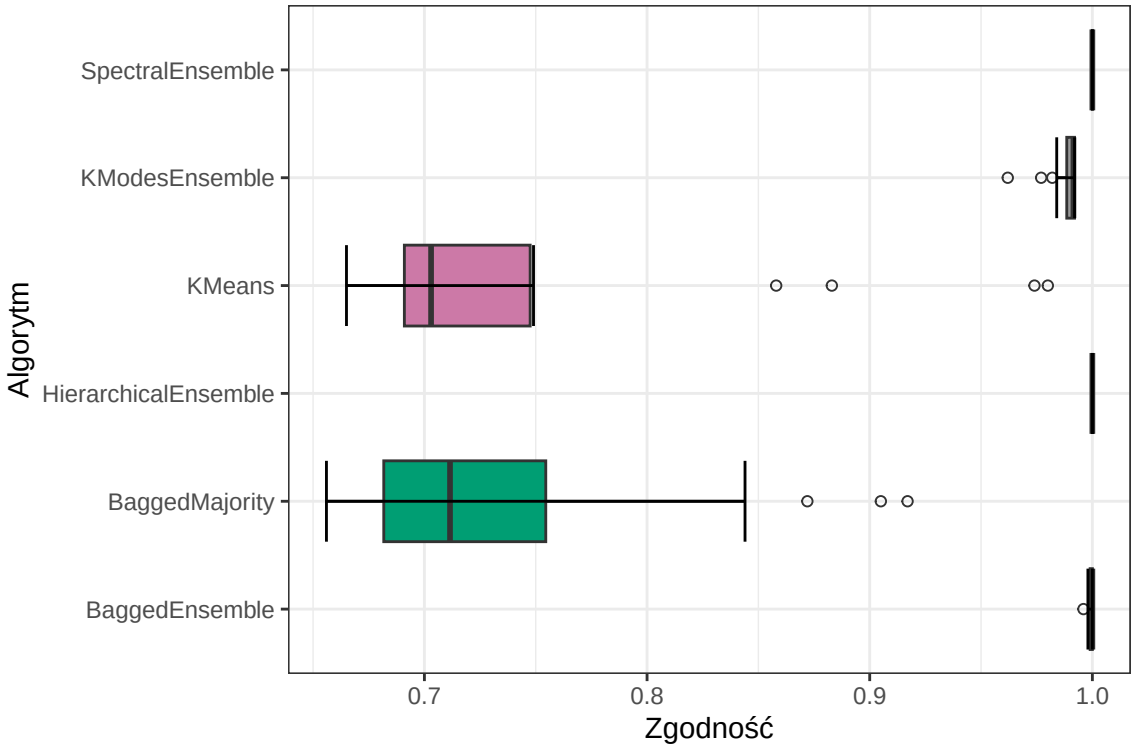
4.2.3 Dane Cassini

Wykresy pudełkowe zgodności dla zbioru **Cassini** przedstawione są na Rysunku 4.4. Od razu możemy zauważyć bardzo dobre wyniki dla czterech algorytmów. Jak potwierdza Tabela 4.4 metody HierarchicalEnsemble oraz SpectralEnsemble za każdym razem w pełni poprawnie rozpoznają występujące skupienia, a KModesEnsemble i BaggedEnsemble prawie w każdym przypadku. Algorytm BaggedMajority charakteryzuje się dużym rozrzutem, podobnie jak bazowa metoda K-średnich.

Tabela 4.3: Wyniki wskaźników oceny klasteryzacji dla zbioru **MK3**.

	Zgodność	Wskaźnik Randa	Silhouette	Spójność
KMeans	0.859 (±0.074)	0.781 (±0.068)	0.480 (±0.031)	47.225 (±5.073)
KModesEnsemble	0.882 (±0.003)	0.800 (±0.004)	0.491 (±0.000)	47.513 (±1.553)
SpectralEnsemble	0.912 (±0.009)	0.825 (±0.018)	0.478 (±0.004)	37.783 (±2.470)
BaggedEnsemble	0.863 (±0.033)	0.755 (±0.054)	0.443 (±0.023)	48.678 (±9.744)
BaggedMajority	0.882 (±0.017)	0.797 (±0.037)	0.484 (±0.027)	51.797 (±29.919)
HierarchicalEnsemble	0.873 (±0.032)	0.769 (±0.037)	0.438 (±0.007)	38.946 (±2.618)

Źródło: opracowanie własne.



Rysunek 4.4: Zgodności partycji w przypadku zbioru **Cassini**.
Źródło: opracowanie własne.

Otrzymane wartości indeksu silhouette w przypadku metod zespołowych są gorsze niż dla algorytmu K-średnich. Z drugiej strony, wyniki dla wskaźnika spójności są, z wyjątkiem przypadku BaggedMajority, lepsze niż dla metody referencyjnej. Przy tej okazji warto zauważyć, że wyniki uzyskane w przypadku wskaźnika silhouette pokazują, jak trudna i niejednoznaczna może być interpretacja rezultatów analizy skupień — otrzymane partycje, które pokrywają się nawet w pełni z rzeczywistymi etykietkami klas, są przez ten wskaźnik określane jako te o gorszych własnościach geometrycznych, takich jak zwartość i separacja przestrzenna skupień.

4.2.4 Dane yeast

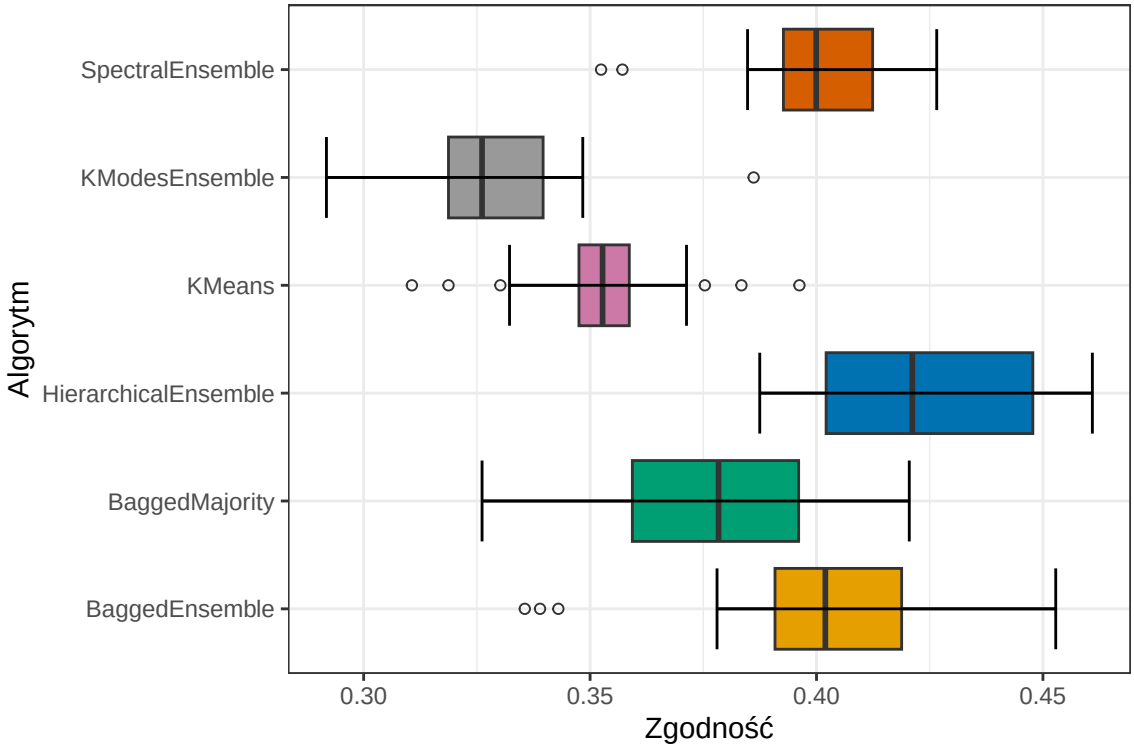
Wykresy pudełkowe dla zbioru **yeast** zostały przedstawione na Rysunku 4.5. Jest on najpoważniejszym wyzwaniem ze wszystkich — ponieważ klasy nie są dobrze odseparo-

Tabela 4.4: Wyniki wskaźników oceny klasteryzacji dla zbioru **Cassini**.

	Zgodność	Wskaźnik Randa	Silhouette	Spójność
KMeans	0.746 (±0.098)	0.594 (±0.132)	0.421 (±0.023)	54.743 (±7.455)
KModesEnsemble	0.988 (±0.007)	0.969 (±0.018)	0.363 (±0.003)	19.868 (±5.241)
SpectralEnsemble	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)
BaggedEnsemble	0.999 (±0.001)	0.998 (±0.003)	0.358 (±0.001)	3.138 (±4.234)
BaggedMajority	0.741 (±0.080)	0.580 (±0.093)	0.384 (±0.057)	165.146 (±92.346)
HierarchicalEnsemble	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)

Źródło: opracowanie własne.

wane, co pokazywał też ich wykres rozrzutu (Rys.4.1). Pudełka dla wszystkich metod są stosunkowo szerokie, co może świadczyć o ich względnie dużej niestabilności. Zwraca uwagę słaby wynik KModesEnsemble — algorytm poradził sobie najgorzej ze wszystkich. W przypadku tego zbioru danych mamy $K = 10$ skupień, więc zgodnie z oczekiwaniami zgodność partycji z rzeczywistymi etykietkami klas jest dużo niższa w porównaniu do danych składających się z dwóch czy trzech skupień.



Rysunek 4.5: Zgodności partycji w przypadku zbioru **yeast**.
Źródło: opracowanie własne.

Wyniki dla wskaźnika silhouette w Tabeli 4.5 ponownie pokazują, że rezultaty uzyskane z wykorzystaniem algorytmów grupowania zespołowego są według tego kryterium gorsze niż dla algorytmu K-średnich. Ponadto, w przypadku metod SpectralEnsemble, BaggedEnsemble i HierarchicalEnsebmle otrzymane partycje są bardziej spójne niż dla pozostałych metod, a otrzymane zgodności o przynajmniej 5% większe.

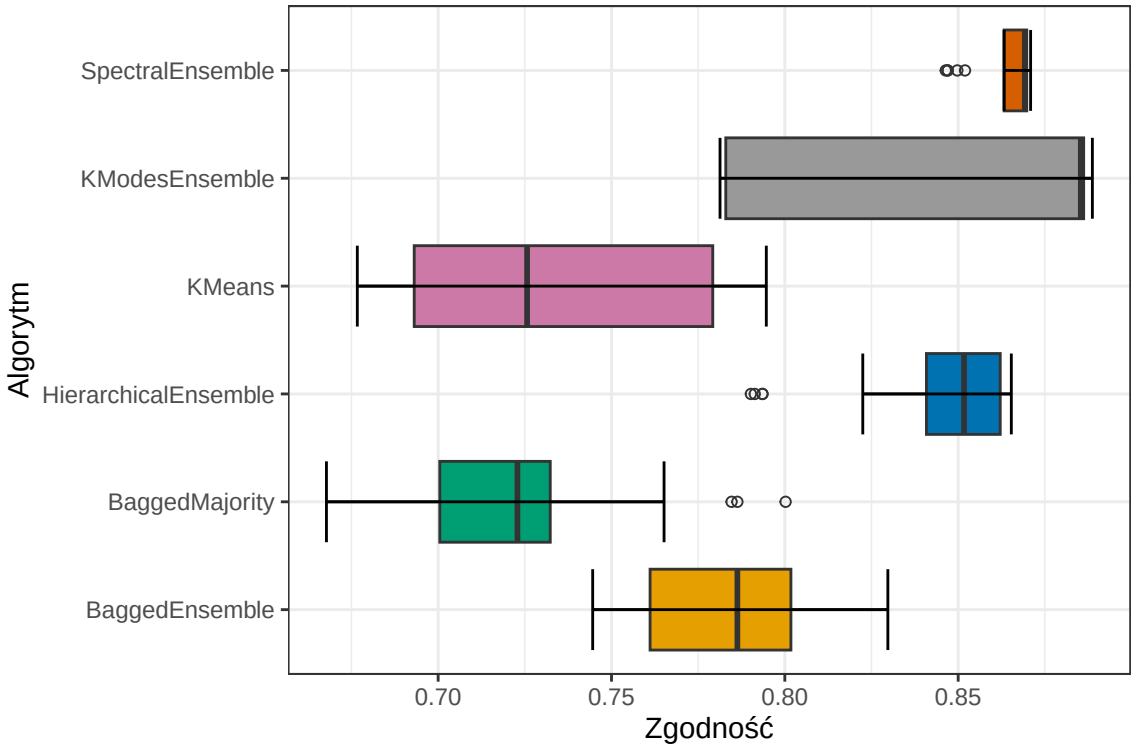
Tabela 4.5: Wyniki wskaźników oceny klasteryzacji dla zbioru **yeast**.

	Zgodność	Wskaźnik Randa	Silhouette	Spójność
KMeans	0.353 (±0.020)	0.136 (±0.009)	0.166 (±0.009)	790.065 (±40.327)
KModesEnsemble	0.329 (±0.020)	0.104 (±0.023)	0.092 (±0.052)	894.079 (±33.107)
SpectralEnsemble	0.399 (±0.020)	0.143 (±0.012)	0.157 (±0.006)	648.177 (±21.193)
BaggedEnsemble	0.399 (±0.031)	0.146 (±0.022)	0.143 (±0.011)	788.055 (±68.873)
BaggedMajority	0.377 (±0.026)	0.144 (±0.015)	0.146 (±0.009)	911.370 (±71.017)
HierarchicalEnsemble	0.425 (±0.025)	0.176 (±0.013)	0.145 (±0.014)	525.586 (±27.730)

Źródło: opracowanie własne.

4.2.5 Dane digits

Wykresy pudełkowe (Rys.4.6) dla zbioru **digits** pokazują, że mimo dużej liczby obserwacji i dużej liczby klastrów wszystkie analizowane metody poradziły sobie co najmniej dobrze z postawionym zadaniem. Szczególną uwagę powinniśmy zwrócić na wyniki algorytmów HierachicalEnsemble, SpectralEnsemble oraz KModesEnsemble. Dwie pierwsze metody charakteryzują się małym rozrzutem otrzymanych zgodności, a średnie wyniki tego kryterium i skorygowanego wskaźnika Randa są najlepsze wśród analizowanych metod. Pokazuje to Tabela 4.6. Gorzej, nawet od algorytmu K-średnich, poradziła sobie metoda BaggedMajority.



Rysunek 4.6: Zgodności partycji w przypadku zbioru **digits**.
Źródło: opracowanie własne.

Wnioski dla wskaźników spójności i silhouette są podobne jak dla wcześniej analizowanych zbiorów danych.

Tabela 4.6: Wyniki wskaźników oceny klasteryzacji dla zbioru **digits**.

	Zgodność	Wskaźnik Randa	Silhouette	Spójność
KMeans	0.733 (±0.046)	0.618 (±0.041)	0.177 (±0.010)	413.801 (±42.404)
KModesEnsemble	0.850 (±0.050)	0.754 (±0.036)	0.171 (±0.002)	321.330 (±16.559)
SpectralEnsemble	0.864 (±0.009)	0.772 (±0.011)	0.176 (±0.001)	188.832 (±14.720)
BaggedEnsemble	0.785 (±0.026)	0.714 (±0.029)	0.174 (±0.003)	224.314 (±33.281)
BaggedMajority	0.725 (±0.035)	0.605 (±0.031)	0.174 (±0.006)	434.591 (±61.744)
HierarchicalEnsemble	0.842 (±0.027)	0.743 (±0.018)	0.174 (±0.005)	197.697 (±21.145)

Źródło: opracowanie własne.

4.2.6 Stabilność i złożoność obliczeniowa

Tabela 4.7 przedstawia wartości średnie i odchylenia standardowe (podane w nawiasach) wskaźników stabilności uzyskane przez badane algorytmy dla poszczególnych zbiorów danych. Algorytmy BaggedMajority oraz KModesEnsemble są za każdym razem bardziej stabilne niż metoda K-średnich, co może wskazywać, że bardzo dobrze radzą sobie z możliwymi zaburzeniami zbiorów danych. W przypadku pozostałych metod wartości stabilności są większe od poziomu referencyjnego dla zbiorów **Cassini** i **digits**. Jednak nadal, z wyłączeniem zbioru **yeast**, wartości te są na poziomie co najmniej 0,7, co świadczy o tym, że rozważane metody grupowania zespołowego w większości przypadków charakteryzują się zadowalającą stabilnością i prowadzą do powtarzalnych rezultatów.

Tabela 4.7: Wartości wskaźnika stabilności dla wszystkich analizowanych danych.

	Iris	MK3	Cassini	Yeast	Digits
KMeans	0.826 (±0.216)	0.895 (±0.110)	0.568 (±0.180)	0.519 (±0.088)	0.728 (±0.078)
KModesEnsemble	0.858 (±0.169)	0.946 (±0.052)	0.875 (±0.159)	0.549 (±0.083)	0.896 (±0.055)
SpectralEnsemble	0.764 (±0.217)	0.834 (±0.069)	1.000 (±0.000)	0.498 (±0.057)	0.875 (±0.063)
BaggedEnsemble	0.752 (±0.164)	0.800 (±0.073)	0.892 (±0.159)	0.354 (±0.080)	0.830 (±0.034)
BaggedMajority	0.879 (±0.121)	0.929 (±0.034)	0.632 (±0.080)	0.611 (±0.076)	0.864 (±0.057)
HierarchicalEnsemble	0.816 (±0.183)	0.830 (±0.082)	1.000 (±0.000)	0.514 (±0.060)	0.889 (±0.056)

Źródło: opracowanie własne.

Przeanalizujemy teraz złożoność obliczeniową rozważanych algorytmów. Tabela 4.8 przedstawia średnie i odchylenia standardowe dla pomiarów czasu działania wszystkich metod. Krótkim czasem wykonania wyróżnia się BaggedMajority. Jest to niewątpliwie duża zaleta tego algorytmu, który, choć nie zwraca najlepszych wyników, to może poszczycić się krótkim czasem działania. Również w przypadku metody BaggedEnsemble zmiana liczby obserwacji nie wpływa drastycznie na wydłużenie przeprowadzanych przez algorytm operacji. Czasy działania zawierają się w przybliżeniu w przedziale od 1 do 4 sekund.

Inaczej wygląda to w przypadku pozostałych metod. Algorytmy KModesEnsemble, HierarchicalEnsemble, SpectralEnsemble wyraźnie reagują na zwiększającą się liczbę obserwacji w zbiorze danych. Dla tego pierwszego czas działania dla danych **digits** jest przeszło 5,5 raza dłuższy od tego dla zbioru **iris**. W przypadku metod HierarchicalEnsemble oraz SpectralEnsemble proporcje te wynoszą kolejno przeszło 6,5 i prawie 7. Działanie drugiego algorytmu jest najbardziej czasochłonne ze wszystkich.

Tabela 4.8: Wyniki pomiaru czasu działania algorytmów.

	Iris	MK3	Cassini	Yeast	Digits
KMeans	0.007 (±0.001)	0.004 (±0.001)	0.016 (±0.004)	0.051 (±0.014)	0.015 (±0.004)
KModesEnsemble	1.376 (±0.027)	1.080 (±0.057)	5.099 (±0.161)	7.840 (±0.553)	8.167 (±0.714)
SpectralEnsemble	1.865 (±0.038)	1.260 (±0.136)	11.841 (±0.591)	10.577 (±0.526)	12.927 (±0.509)
BaggedEnsemble	0.982 (±0.028)	0.573 (±0.081)	1.897 (±0.203)	2.584 (±0.096)	3.333 (±0.076)
BaggedMajority	0.076 (±0.005)	0.144 (±0.020)	0.563 (±0.028)	0.428 (±0.047)	1.268 (±0.099)
HierarchicalEnsemble	1.803 (±0.029)	1.297 (±0.020)	11.519 (±0.463)	8.715 (±0.200)	11.820 (±0.631)

Źródło: opracowanie własne.

4.3 Dobór hiperparametrów

W rozdziale 3 oraz podrozdziale 4.2 przedstawiliśmy rezultaty analizy skupień przeprowadzonej za pomocą metod grupowania zespołowego. Wyboru hiperparametrów dokonywano najczęściej, stosując metodę prób i błędów. Starania te pokazały, że różne ich konfiguracje mogą czasami skutkować znaczącymi różnicami w otrzymywanych wynikach. W tym podrozdziale przedstawione zostaną bardziej formalne próby zbadania wpływu doboru hiperparametrów na skuteczności metod.

Najpierw zbadamy, jak na skuteczność działania algorytmu BaggedMajority będzie wpływała zmiana liczby replikacji bazowych oraz zastosowanie jako metody bazowej algorytmu PAM (rekomendowanego w artykule źródłowym [5]). Następnie przyjrzymy się temu, jaki wpływ na efektywność metody BaggedEnsemble ma różny dobór liczby bazowych centrów i liczby partycji.

W przypadku metod KModesEnsemble, HierarchicalEnsemble oraz SpectralEnsemble zostanie zbadany wpływ zwiększania liczby partycji, a także liczby skupień w bazowym algorytmie K-średnich.

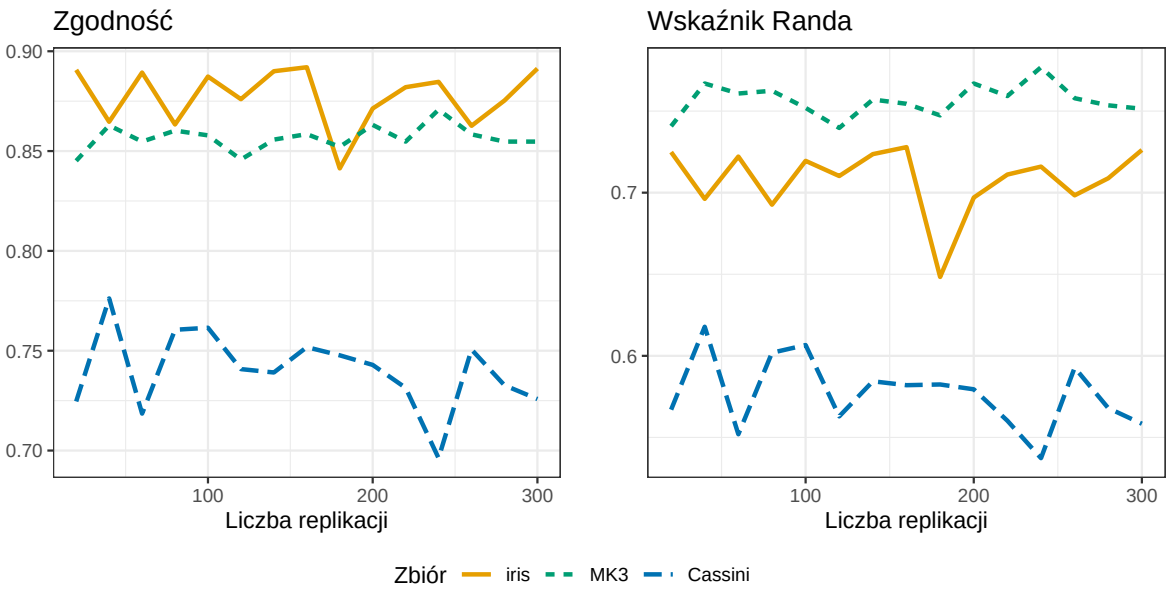
Wszystkie analizy, z wyjątkiem tej dotyczącej zastosowania algorytmu PAM, zostały przeprowadzone dla danych **iris**, **Cassini** oraz **MK3**. Przeprowadzenie badań dla tych trzech zbiorów pozwoli nam odpowiedzieć na postawione pytania badawcze w bardziej przejrzysty i jednoznaczny sposób. Pozostałe zbiory danych są bardziej złożone, mają więcej skupień i cech. Dodatkowo składają się z dużej liczby obserwacji, co spowodowałoby, że przeprowadzone dla nich symulacje byłby bardzo czasochłonne. Co więcej, by ograniczyć czas potrzebny na przeprowadzenie symulacji, zdecydowano się na 10-krotne powtarzanie procedury klasteryzacji dla każdego zbioru danych i ustalonych hiperparametrów.

4.3.1 Algorytm BaggedMajority

W przypadku metody BaggedMajority zbadamy, jaki wpływ na jej działanie ma zmiana liczby replikacji bootstrapowych oraz zmiana bazowego algorytmu klasteryzacji.

Wpływ liczby replikacji bootstrapowych

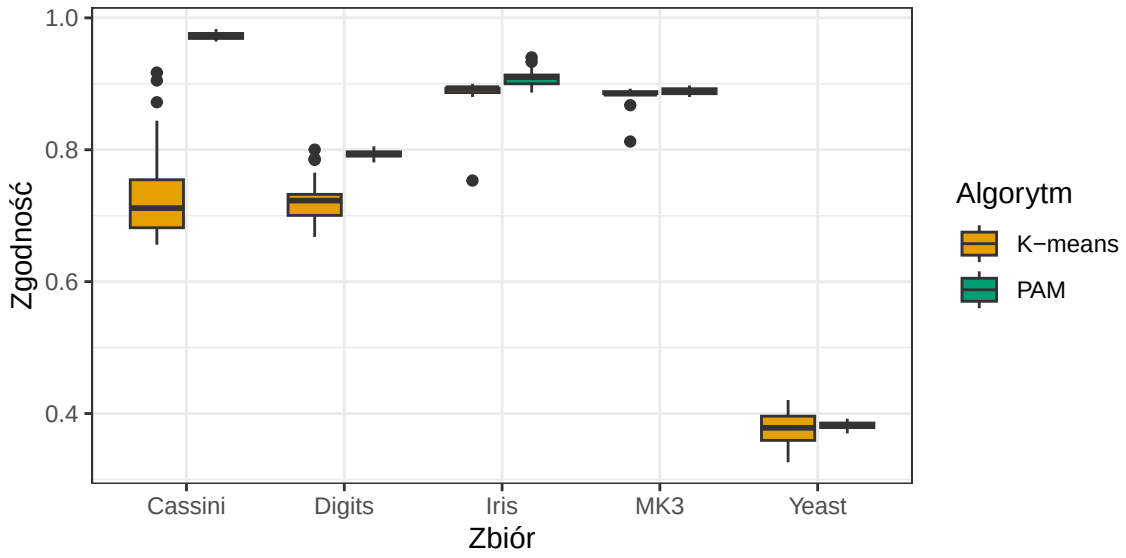
Rysunek 4.7 przedstawia średnie wartości wskaźnika zgodności i skorygowanego wskaźnika Randa w zależności od liczby replikacji bootstrapowych. Interpretując otrzymane wyniki, nie dostrzegamy żadnego trendu, zależności — wartości obu wskaźników oscylują wokół pewnego poziomu, który zależy oczywiście od konkretnego zbioru danych. Widać więc, że nie ma potrzeby wybierania bardzo dużej liczby replikacji, ponieważ nie prowadzi to do poprawienia uzyskanych wyników, a jedynie zwiększa czas działania algorytmu.



Rysunek 4.7: Wartości zgodności i skorygowanego wskaźnika Randa w zależności od liczby replikacji.
Źródło: opracowanie własne.

Wykorzystanie algorytmu PAM

We wszystkich wcześniejszych analizach bazowym algorytmem wykorzystywanym w metodzie BaggedMajority był algorytm K-średnich. Sprawdzimy teraz, jak na otrzymywane rezultaty wpłynie zastosowanie w tej roli algorytmu PAM. Wyniki porównujące otrzymane w obu przypadkach zgodności końcowej partycji z rzeczywistymi klasami przedstawione zostały na Rysunku 4.8.



Rysunek 4.8: Porównanie zgodności klasteryzacji dla metod bazowych K-średnich oraz PAM.
Źródło: Opracowanie własne.

Możemy zauważyć, że w każdym przypadku wyniki, czasami znacząco, poprawiły się. We wszystkich przypadkach zgodność charakteryzuje się mniejszym rozrzutem. Algorytm BaggedMajority z PAM może być więc postrzegany jako konkurencyjne podejście w stosunku do pozostałych metod.

Dużą wadą zastosowania algorytmu PAM jest jednak wzrost złożoności obliczeniowej. Spójrzmy na wyniki w Tabeli 4.9, gdzie zestawiony został średni czas działania metody BaggedMajority z dwoma różnymi algorytmami bazowymi. Możemy zauważyć, że dla zbiorów z mniejszą liczbą obserwacji różnice nie są aż tak znaczące, ale dla zbiorów **digits** czy **yeast** są już one ewidentne. Tak długo nie działała żadna z wcześniej omawianych metod. Możemy wysnuć więc wniosek, że zastosowanie PAM jako algorytmu bazowego, z racji dobrych rezultatów, może być dobrym rozwiązaniem w przypadku niewielkich zbiorów danych ($N < 1000$).

Tabela 4.9: Porównanie wyników pomiaru czasu działania dla metod bazowych K-średnich oraz PAM.

	Iris	MK3	Cassini	Yeast	Digits
K-means	0.076 (±0.005)	1.268 (±0.099)	0.428 (±0.047)	0.144 (±0.020)	0.563 (±0.028)
PAM	0.044 (±0.003)	0.721 (±0.022)	1.954 (±0.158)	18.993 (±1.075)	22.836 (±1.492)

Źródło: opracowanie własne.

4.3.2 Algorytm BaggedEnsemble

Przyjrzyjmy się teraz jaki wpływ na otrzymywane rezultaty mają hiperparametry metody BaggedEnsemble — liczba centrów i liczba replikacji bootstrapowych. Wyniki przedstawione zostały w postaci map ciepła (Rys.4.9).

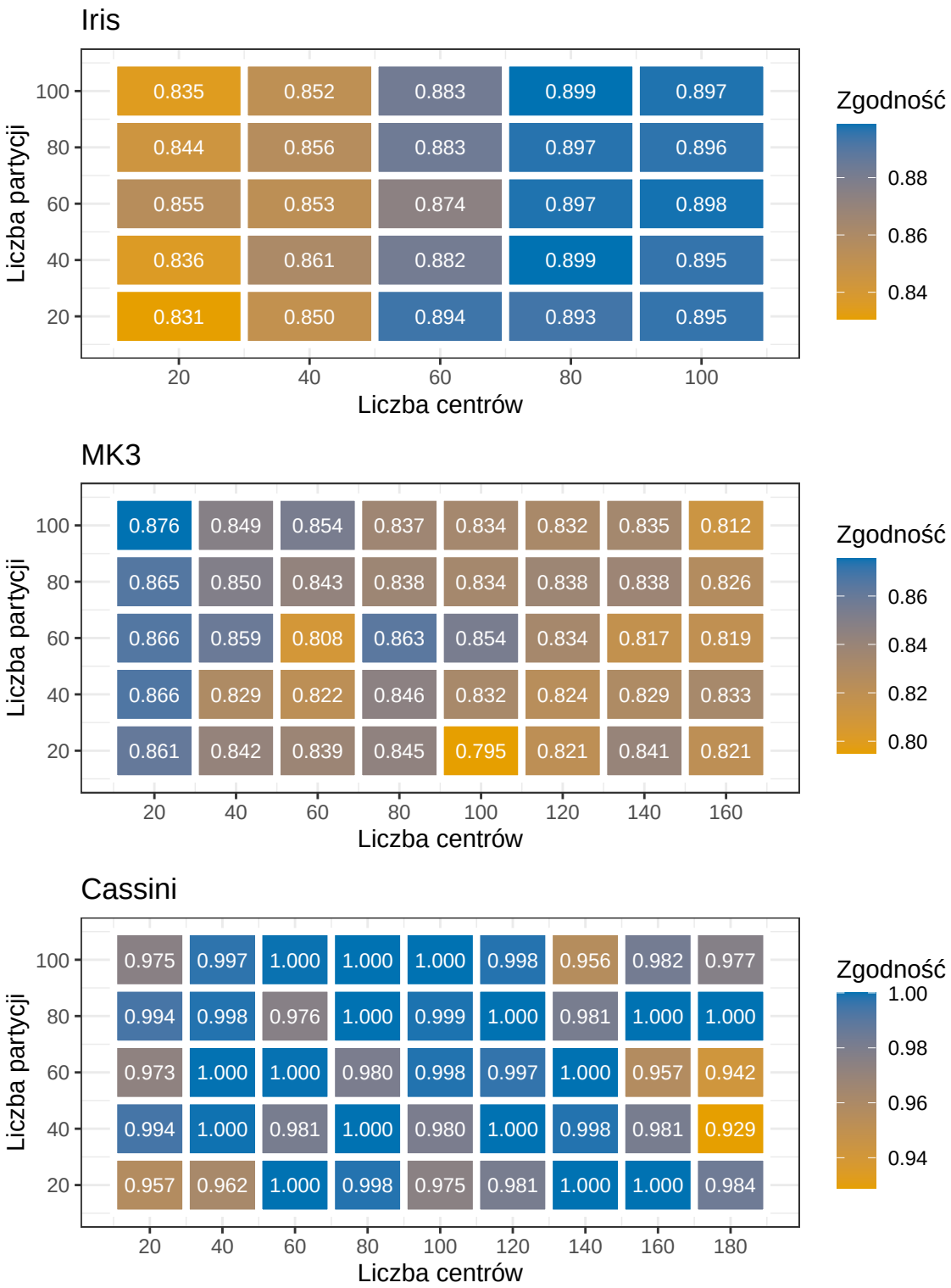
W przypadku zbioru **iris** możemy zauważyć, że wzrost liczby centrów poprawia wyniki, niezależnie od przyjętej liczby partycji. Zmiany wartości drugiego hiperparametru nie powodują jednak zauważalnych różnic.

Dla zbiorów **digits** i **Cassini** ciężko sformułować jednoznaczne wnioski. Zarówno zmiany jednego hiperparametru przy ustalonej wartości drugiego, jak i jednoczesny wzrost wartości ich obu nie powodują monotonicznej poprawy czy pogorszenia jakości klasteryzacji.

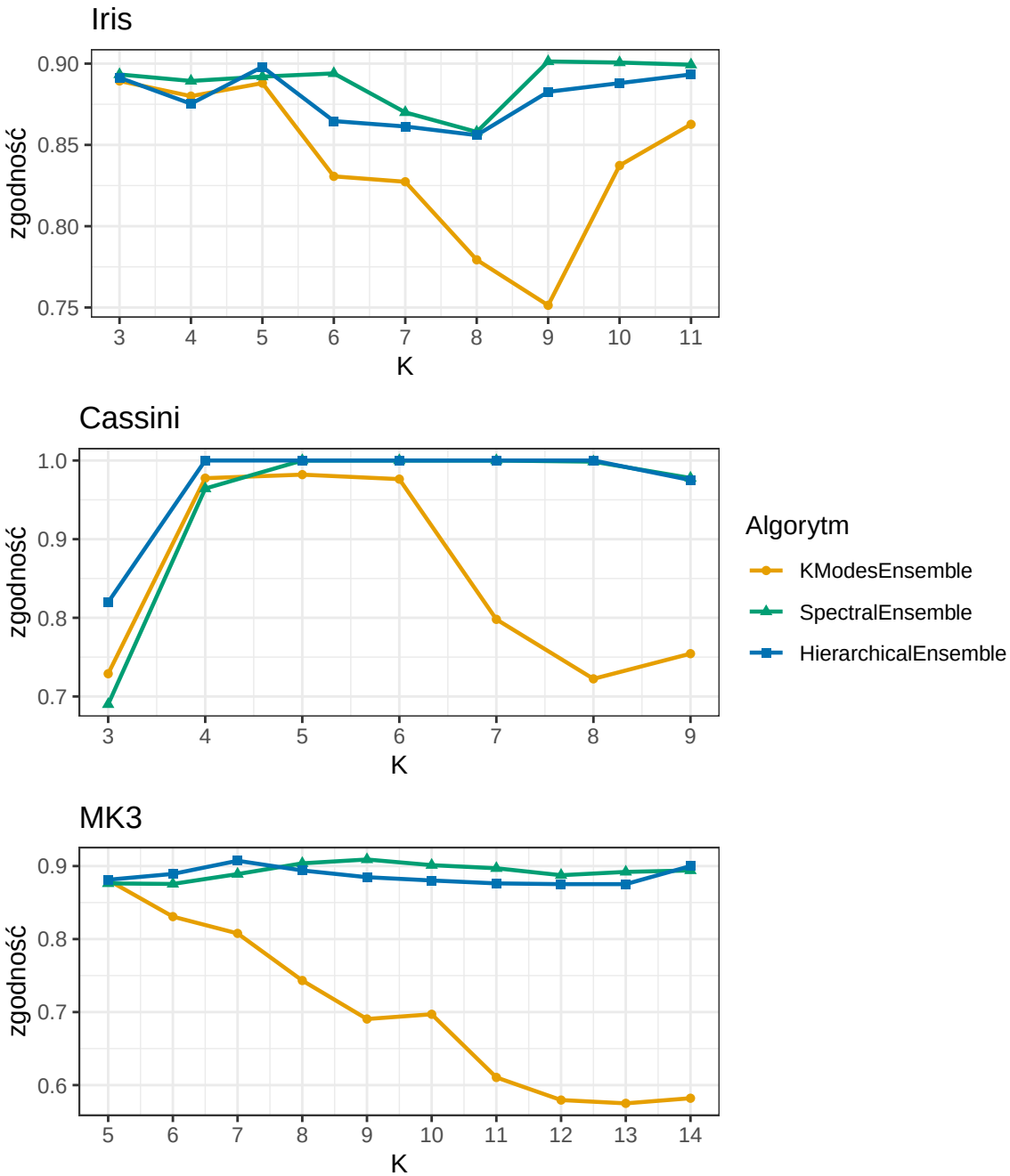
Na podstawie powyższych wyników możemy dojść do konkluzji, że dla algorytmu BaggedEnsemble nie istnieje ogólna reguła doboru hiperparametrów. Należy więc w przypadku każdego nowego zagadnienia, badać jak zmieniają się wyniki w zależności od doboru liczby centrów czy liczby partycji.

4.3.3 Pozostałe metody

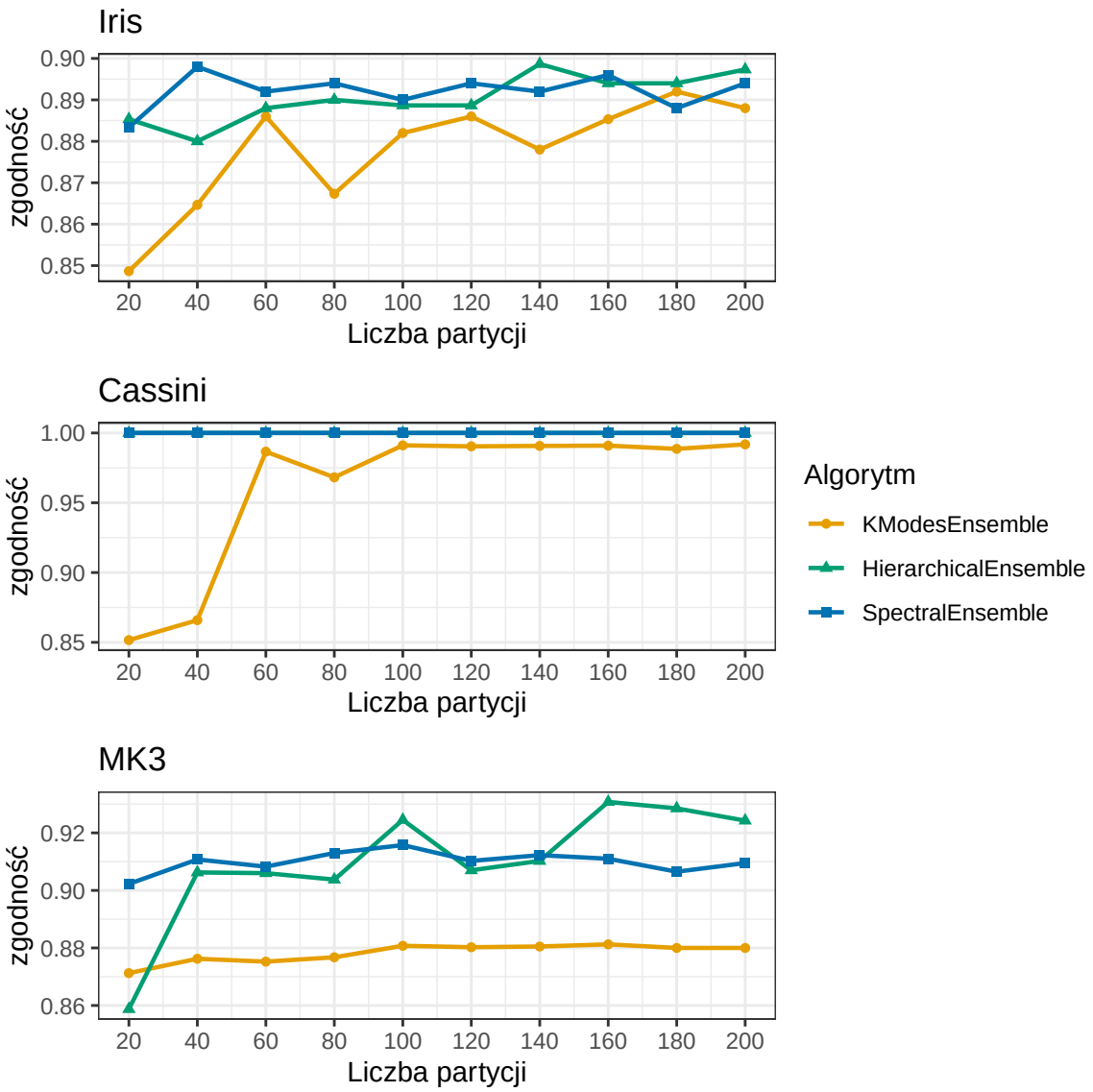
Dla metod KModesEnsemble, HierarchicalEnsemble oraz SpectralEnsemble zbadamy teraz, czy i jaki wpływ na ich skuteczność ma zmiana liczby skupień dla bazowego algorytmu K-średnich oraz przyjęta liczba partycji. Zależność wskaźnika zgodności od liczby skupień została przedstawiona na Rysunku 4.10. Możemy zauważyć, że w przypadku metod HierarchicalEnsemble oraz SpectralEnsemble jakość klasteryzacji na ogół poprawia się wraz ze wzrostem wybranej liczby skupień. W przypadku algorytmu KModesEnsemble obserwujemy odwrotną zależność — działanie algorytmu pogarsza się wraz z rosnącym K.



Rysunek 4.9: Mapy ciepła dla wartości wskaźników zgodności.
Źródło: opracowanie własne.



Rysunek 4.10: Zależność między ustaloną liczbą skupień algorytmu bazowego a wartościami zgodności.
Źródło: opracowanie własne.



Rysunek 4.11: Zależność wyników zgodności od liczby partycji.
Źródło: opracowanie własne.

Sprawdzimy teraz, jak na jakość grupowania wpływa zmiana liczby składowych partycji (Rys.4.11). Jak możemy zauważyć, wzrost liczby partycji ma na ogół pozytywny wpływ — wartości zgodności albo się poprawiają, albo pozostają na podobnym poziomie. W przypadku metody KModesEnsemble wynika to prawdopodobnie z tego, że powiększa się zbiór cech, co pozwala algorytmowi K-modes na lepsze rozpoznanie struktury danych. Algorytmy SpectralEnsemble i HierarchicalEnsemble korzystają natomiast z tego, że macierz podobieństwa wyznaczona na podstawie większej liczby składowych partycji pozwala na lepsze pogrupowanie obserwacji w klastry.

4.4 Zwiększenie różnorodności partycji

Do tej pory do generowania partycji wykorzystywany był jedynie algorytm K-średnich. W przypadku metod uczenia zespołowego (ang. *ensemble learning*) korzyść wynikająca z agregacji jest zazwyczaj większa w przypadku większej różnorodności (ang. *diversity*)

modeli składowych, co w naszym przypadku oznacza większą różnorodność partycji. W praktyce, tak jak zostało to omówione w podrozdziale 2.1.1, zwiększenie różnorodności można uzyskać na różne sposoby, w tym np. zaburzając wyjściowy zbiór danych lub wykorzystując różne algorytmy.

W tym podrozdziale przyjrzymy się jak na działanie metod KModesEnsemble, SpectralEnsemble oraz HierarchicalEnsemble będzie wpływało zwiększenie różnorodności składowych partycji, uzyskane poprzez zastosowanie różnych (lub inaczej inicjalizowanych) bazowych algorytmów grupowania. W szczególności będą to następujące metody:

- klasteryzacja spektralna, korzystająca z macierzy podobieństwa wyznaczonej za pomocą różnych funkcji odległości,
- aglomeracyjne grupowanie hierarchiczne z każdą możliwą kombinacją różnych funkcji łączących (*average*, *complete*, *single*, *Ward*) i różnych metryk,
- metoda K-średnich z różną liczbą ustalonych klastrów.

Wspomniane metryki to odległość euklidesowa oraz Manhattan. Algorytmy klasteryzacji spektralnej i grupowania hierarchicznego będą również inicjalizowane z różną liczbą skupień. Szczegóły dotyczące inicjalizacji metod znaleźć można w kodach źródłowych, które dostępne są w repozytorium wspomnianym w Dodatku.

Tabela 4.10 zawiera informacje o liczbie składowych partycji wyznaczonych dla poszczególnych zbiorów danych. Wyniki z podrozdziału 4.2 były uzyskane na bazie 100 partycji.

Tabela 4.10: Liczba partycji dla poszczególnych zbiorów danych.

	Iris	MK3	Cassini	Yeast
Liczba partycji	98	28	42	70

Źródło: opracowanie własne.

W Tabelach 4.11, 4.12 i 4.13 zestawione są wyniki uzyskane przy pomocy jedynie algorytmu K-średnich z wynikami opartymi na zwiększonej różnorodności partycji (określonymi w tabelach jako *Diverse*).

W przypadku każdego z rozważanych algorytmów grupowania zespołowego uzyskane na bazie nowej metody końcowe partycje charakteryzują się lepszą spójnością, a dla metod HierarchicalEnsemble i SpectralEnsemble wartości wskaźnika silhouette są większe niż w przypadku wykorzystania jedynie bazowej metody K-średnich. Otrzymane wskaźniki zgodności czy skorygowanego indeksu Randa najczęściej utrzymują się na podobnym poziomie i nie różnią się znacząco. Zwróćmy jednak uwagę na wyniki uzyskane dla zbioru **yeast**. Dla metod KModesEnsemble i SpectralEnsemble uzyskaliśmy znaczącą poprawę zgodności partycji, jednak w przypadku algorytmu HierarchicalEnsemble wynik pogorszył się.

Analiza czasu działania pokazuje, że rozwiązanie oparte na zwiększeniu różnorodności partycji charakteryzuje się słabą skalowalnością w przypadku danych o coraz większej liczbie obserwacji. Wynika to z faktu konieczności wielokrotnego uruchomienia metody grupowania hierarchicznego, która jest wrażliwa na zwiększanie rozmiaru danych. Z tego

powodu obliczenia nie zostały wykonane dla zbioru **digits**, który składa się z największej liczby obserwacji.

Możemy więc dojść do wniosku, że korzystanie z zaprezentowanego w tej części rozwiązania, jest dobrym wyborem w przypadku zbiorów o niedużej liczbie obserwacji ($N < 1000$).

Tabela 4.11: Zestawienie wyników dla metody SpectralEnsemble

	Zgodność	Wskaźnik Randa	Silhoutte	Spójność	Stabilność	Czas działania [s]
Iris						
K-means	0.897 (±0.007)	0.740 (±0.014)	0.555 (±0.001)	6.331 (±2.164)	0.764 (±0.217)	1.865 (±0.038)
Diverse	0.900 (±0.000)	0.746 (±0.000)	0.555 (±0.000)	5.445 (±0.000)	0.750 (±0.165)	1.378 (±0.049)
MK3						
K-means	0.912 (±0.009)	0.825 (±0.018)	0.478 (±0.004)	37.783 (±2.470)	0.834 (±0.069)	1.260 (±0.136)
Diverse	0.872 (±0.002)	0.791 (±0.011)	0.484 (±0.002)	37.108 (±2.399)	0.817 (±0.063)	1.602 (±0.080)
Cassini						
K-means	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)	1.000 (±0.000)	11.841 (±0.591)
Diverse	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)	1.000 (±0.000)	10.563 (±0.843)
Yeast						
K-means	0.399 (±2e-02)	0.143 (±1e-02)	0.157 (±6e-03)	648.177 (±2e+01)	0.498 (±6e-02)	10.577 (±5e-01)
Diverse	0.468 (±0.036)	0.196 (±0.039)	0.192 (±0.019)	473.483 (±45.018)	0.396 (±0.082)	57.741 (±6.009)

Źródło: opracowanie własne.

Tabela 4.12: Zestawienie wyników dla metody HierarchicalEnsemble

	Zgodność	Wskaźnik Randa	Silhoutte	Spójność	Stabilność	Czas działania [s]
Iris						
K-means	0.896 (±0.017)	0.738 (±0.030)	0.552 (±0.009)	7.998 (±2.302)	0.816 (±0.183)	1.803 (±0.029)
Diverse	0.907 (±0.000)	0.759 (±0.000)	0.554 (±0.000)	4.794 (±0.000)	0.707 (±0.131)	1.400 (±0.084)
MK3						
K-means	0.873 (±0.032)	0.769 (±0.037)	0.438 (±0.007)	38.946 (±2.618)	0.830 (±0.082)	1.297 (±0.020)
Diverse	0.877 (±2e-03)	0.784 (±2e-03)	0.486 (±5e-05)	34.400 (±1e+00)	0.797 (±1e-01)	1.599 (±1e-01)
Cassini						
K-means	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)	1.000 (±0.000)	11.519 (±0.463)
Diverse	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)	1.000 (±0.000)	9.235 (±1.162)
Yeast						
K-means	0.425 (±0.025)	0.176 (±0.013)	0.145 (±0.014)	525.586 (±27.730)	0.514 (±0.060)	8.715 (±0.200)
Diverse	0.3644 (±0.001)	0.0886 (±0.002)	0.2611 (±0.003)	154.9739 (±5.121)	0.8183 (±0.109)	58.5194 (±1.205)

Źródło: opracowanie własne.

Tabela 4.13: Zestawienie wyników dla metody KModesEnsemble.

	Zgodność	Wskaźnik Randa	Silhoutte	Spójność	Stabilność	Czas działania [s]
Iris						
K-means	0.892 (±3e-03)	0.727 (±6e-03)	0.552 (±7e-04)	10.519 (±1e-01)	0.858 (±2e-01)	1.376 (±3e-02)
Diverse	0.907 (±0.000)	0.759 (±0.000)	0.554 (±0.000)	4.794 (±0.000)	0.782 (±0.217)	1.190 (±0.097)
MK3						
K-means	0.882 (±3e-03)	0.800 (±4e-03)	0.491 (±5e-04)	47.513 (±2e+00)	0.946 (±5e-02)	1.080 (±6e-02)
Diverse	0.872 (±0.003)	0.799 (±0.006)	0.480 (±0.001)	36.457 (±2.035)	0.819 (±0.083)	2.230 (±0.097)
Cassini						
K-means	0.988 (±0.007)	0.969 (±0.018)	0.363 (±0.003)	19.868 (±5.241)	0.875 (±0.159)	5.099 (±0.161)
Diverse	1.000 (±0.000)	1.000 (±0.000)	0.358 (±0.000)	0.211 (±0.000)	1.000 (±0.000)	2.594 (±0.178)
Yeast						
K-means	0.3290 (±0.020)	0.1044 (±0.023)	0.0924 (±0.052)	894.0786 (±33.107)	0.5495 (±0.083)	7.8399 (±0.553)
Diverse	0.350 (±2e-02)	0.136 (±9e-03)	0.108 (±2e-02)	843.488 (±5e+01)	0.371 (±4e-02)	14.551 (±4e-01)

Źródło: opracowanie własne.

4.5 Podsumowanie

W wyniku przeprowadzonej analizy porównawczej możemy przyjąć, że warto rozważać korzystanie z zespołowych algorytmów klasteryzacji, w tym zwłaszcza z KModesEnsemble, HierarchicalEnsemble i SpectralEnsemble. Na przestrzeni analiz dla różnych zbiorów danych te metody charakteryzowały się wysokimi wynikami zgodności, dobrą spójnością końcowych partycji, a także stabilnością uzyskanych rozwiązań. Ich niewątpliwą wadą jest czas działania, jednak tak jak zostało to wykazane, jest to koszt warty poniesienia.

Dzięki zbadaniu wpływu hiperparametrów na efektywność rozważanych metod grupowania zespołowego (p. podrozdział 4.3) również uzyskaliśmy wartościowe wnioski. Przede wszystkim pokazaliśmy, że wykorzystując metodę BaggedMajority z algorytmem bazowym PAM, można znacząco poprawić uzyskiwane przez nią wyniki. Z drugiej strony, wiązało się to ze znacznym wydłużeniem działania algorytmu. W przypadku metod KModesEnsemble, HierarchicalEnsemble oraz SpectralEnsemble pokazaliśmy, że dla dwóch ostatnich warto przyjmować liczbę skupień w bazowym algorytmie K-średnich większą od rzeczywistej liczby skupisk w zbiorze danych. Wszystkie trzy metody zyskiwały na powiększaniu liczby składowych partycji.

Na koniec, w podrozdziale 4.4, wykorzystaliśmy inny schemat generowania składowych partycji. Wykorzystując różne algorytmy bazowe, z różnymi liczbami skupień, otrzymaliśmy końcowe partycje, które charakteryzowały się lepszą spójnością, zwartością i separacją przestrzenną, na co wskazywały wartości kryteriów wewnętrznych. Wyniki wskaźnika zgodności okazały się być na ogół zbliżone do tych, które otrzymaliśmy w przypadku stosowania jedynie algorytmu K-średnich, choć zaobserwowaliśmy również kilka rezultatów znacząco lepszych. Warto zaznaczyć, że potrzebna do uzyskania takich rezultatów liczba różnorodnych partycji była za każdym razem mniejsza niż w przypadku partycji generowanych za pomocą jedynie algorytmu K-średnich.

Podsumowanie

Celem pracy było przedstawienie metod grupowania zespołowego oraz przeprowadzenie ich analizy porównawczej.

Najpierw wprowadziliśmy pojęcie i ideę analizy skupień. Omówione zostały również podstawowe algorytmy klasteryzacji, ich podstawowe własności oraz kryteria, które można wykorzystać w ocenie otrzymanego grupowania.

Następnie przedstawiliśmy ogólną koncepcję klasteryzacji zespołowej oraz powody, dla których warto rozpatrywać stosowanie tego podejścia. Omówiono siedem wybranych metod, które stanowią dobry przekrój algorytmów grupowania zespołowego. W przypadku każdej metody przedstawiony został schemat jej działania oraz własności, jakimi się charakteryzuje.

Przeprowadzone analizy w rozdziałach 3 i 4 ukazały duży potencjał algorytmów grupowania zespołowego. Jak pokazały wyniki uzyskane w studium przypadku otrzymanie poprawnego grupowania jednego z syntetycznych zbiorów danych możliwe było dopiero dzięki wykorzystaniu metod klasteryzacji zespołowej. Przeprowadzona analiza porównawcza wskazała na dużą efektywność metod SpectralEnsemble, KModesEnsemble oraz HierarchicalEnsemble. Pokazaliśmy także, że w przypadku niektórych metod warto dokonać odpowiedniego doboru hiperparametrów dla ustalonych danych, ponieważ może to prowadzić do uzyskania jeszcze lepszych rezultatów. Na koniec zbadano, jaki wpływ na wyniki może mieć zastosowanie bardziej zróżnicowanych metod generowania składowych partycji. Uzyskane rezultaty pokazały, że otrzymane końcowe grupowania charakteryzowały się dobrymi własnościami geometrycznymi, a wskaźniki zgodności pozostały na wysokim poziomie.

Dzięki przeprowadzonym badaniom możemy stwierdzić, że metody grupowania zespołowego umożliwiają poprawienie uzyskiwanych rezultatów. Jest to podejście bardziej uniwersalne, które umożliwia dobre grupowanie różnych zbiorów danych.

Przeprowadzone analizy, choć dostarczyły wiele wartościowych wniosków, z całą pewnością nie są wystarczające. Dalsze kierunki badań mogłyby objąć włączenie większej liczby metod klasteryzacji zespołowej (np. metod opartych na założeniach probabilistycznych) czy wykorzystanie w symulacjach większej liczby zróżnicowanych zbiorów danych, co pozwoliłoby na bardziej szczegółowe zbadanie zalet i ograniczeń poszczególnych algorytmów. Dodatkowo warto rozważyć bardziej formalne przeanalizowanie wpływu wykorzystywania różnorodnych partycji składowych na otrzymywane rezultaty, np. stosując odpowiednie miary różnorodności. Ponadto, z uwagi na małą liczbę dostępnych implementacji metod klasteryzacji zespołowej w popularnych pakietach (R, Python, Julia) autor pracy planuje w przyszłości stworzenie bibliotek zawierających wybrane algorytmy.

Dodatek

Wszystkie obliczenia zostały wykonane na komputerze o następujących parametrach technicznych: 4-rdzeniowy procesor Intel Core i7-1165G7, 512GB dysku SSD, 16GB pamięci RAM.

Do otrzymania wyników z rozdziałów 3 i 4 wykorzystano język Python oraz następujące moduły spoza biblioteki standardowej: `numpy`, `sklearn`, `scipy`, `pandas`, `kmedoids`. W szczególności warto wymienić następujące funkcje:

- `linear_sum_assignment` z podmodułu `scipy.optimize` — posłużyła do rozwiązywania problemu optymalnego przypisania klas,
- `KMeans`, `AgglomerativeClustering`, `SpectralClustering` z `sklearn.cluster` oraz `pam` z `kmedoids`, które implementują poszczególne standardowe metody grupowania,
- `make_moons`, `make_blobs` z `sklearn.datasets`, które zostały wykorzystane do wygenerowania danych syntetycznych.

Interfejsy do metod klasteryzacji zespołowej zostały zaimplementowane przez autora pracy. Klasy `BaggedMajority` oraz `BaggedEnsemble` implementujące odpowiednie algorytmy klasteryzacji zespołowej wykorzystują wyłącznie niektóre biblioteki wspomniane powyżej. Implementacje metod GraphClosure oraz CoAssocEnsemble znajdują się w bibliotece `openensembles`. W klasach `KModesEnsemble`, `SpectralEnsemble`, `HierarchicalEnsemble` etap generowania partycji opiera się na rozwiązaniach z modułu `openensmebles`, nastomiast funkcje konsensusu to odpowiednie funkcje z bibliotek `sklearn` oraz `kmedoids`.

Kody implementujące powyższe metody, jak i cały kod źródłowy potrzebny do przeprowadzenia obliczeń, dostępny jest w repozytorium w serwisie GitHub pod adresem: `https://github.com/Manik2000/bachelor_thesis`.

Do przetwarzania uzyskanych symulacyjnie wyników, stworzenia wszystkich wizualizacji oraz tabel, posłużył pakiet R, w szczególności następujące biblioteki: `ggplot2`, `ggpubr`, `dplyr`, `reshape2`, `kableExtra`, `rjson`, `tidyverse`, `magrittr`.

Bibliografia

[1] Boongoen, T., Iam-On, N. Cluster ensembles: A survey of approaches with recent extensions and applications. *Computer Science Review 28* (2018), 1–25.

[2] Brock, G., Pihur, V., Datta, S., Datta, S. clvalid: An r package for cluster validation. *Journal of Statistical Software 25*, 4 (2008), 1–22.

[3] Crouse, D. F. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems 52*, 4 (2016), 1679–1696.

[4] Dua, D., Graff, C. UCI machine learning repository, 2017.

[5] Dudoit, S., Fridlyand, J. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics 19*, 9 (06 2003), 1090–1099.

[6] Fred, A. Finding consistent clusters in data partitions. In *Multiple Classifier Systems* (Berlin, Heidelberg, 2001), J. Kittler and F. Roli, Eds., Springer Berlin Heidelberg, pp. 309–318.

[7] Fred, A. L., Jain, A. K. Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 6 (2005), 835–850.

[8] Gionis, A., Mannila, H., Tsaparas, P. Clustering aggregation. *TKDD 1* (03 2007).

[9] Halkidi, M., Batistakis, Y., Vazirgiannis, M. On clustering validation techniques. *Journal of Intelligent Information Systems 17* (10 2001).

[10] Handl, J., Knowles, J., Kell, D. Bioinformatics computational cluster validation in post-genomic data analysis. *Bioinformatics (Oxford, England) 21* (09 2005), 3201–12.

[11] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics. Springer, 2009.

[12] Hennig, C. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis 52*, 1 (2007), 258–271.

[13] Hennig, C. *fpc: Flexible Procedures for Clustering*, 2020. R package version 2.2-9.

[14] Hornik, K. *clue: Cluster Ensembles*, 2022. R package version 0.3-63.

[15] Huang, Q., Gao, R., Akhavan, H. An ensemble hierarchical clustering algorithm based on merits at cluster and partition levels. *Pattern Recognition 136* (2023), 109255.

[16] Huang, Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery 2*, 3 (1998), 283–304.

[17] Hubert, L. J., Arabie, P. Comparing partitions. *Journal of Classification 2* (1985), 193–218.

[18] Jain, A. K., Dubes, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[19] Jia, Y., Tao, S., Wang, R., Wang, Y. Ensemble clustering via co-association matrix self-enhancement, 2022.

[20] Kleinberg, J. An impossibility theorem for clustering. *Adv Neural Inform Process Syst (NIPS) 15* (01 2003).

[21] Leisch, F. Bagged clustering. WorkingPaper 51, SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, 1999.

[22] Leisch, F., Dimitriadou, E. *mlbench: Machine Learning Benchmark Problems*, 2021. R package version 2.1-3.

[23] Liu, H., Liu, T., Wu, J., Tao, D., Fu, Y. Spectral ensemble clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2015), KDD '15, Association for Computing Machinery, p. 715–724.

[24] Luo, H., Kong, F., Li, Y. Combining multiple clusterings via k-modes algorithm. In *Advanced Data Mining and Applications* (Berlin, Heidelberg, 2006), X. Li, O. R. Zaïane, and Z. Li, Eds., Springer Berlin Heidelberg, pp. 308–315.

[25] Palla, G., Derényi, I., Farkas, I., Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature 435* (07 2005), 814–818.

[26] Ronan, T., Anastasio, S., Qi, Z., Tavares, P. H. S. V., Sloutsky, R., Naegle, K. M. Openensembles: A python resource for ensemble clustering. *Journal of Machine Learning Research 19*, 26 (2018), 1–6.

[27] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20* (1987), 53–65.

[28] Schubert, E., Rousseeuw, P. J. Faster k-medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms. In *Similarity Search and Applications*. Springer International Publishing, 2019, pp. 171–187.

[29] Strehl, A., Ghosh, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res. 3*, null (mar 2003), 583–617.

[30] Vega-Pons, S., Ruiz-Shulcloper, J. A survey of clustering ensemble algorithms. *IJPRAI 25* (05 2011), 337–372.

[31] Wu, X., Ma, T., Cao, J., Tian, Y., Alabdulkarim, A. A comparative study of clustering ensemble algorithms. *Computers & Electrical Engineering 68* (2018), 603–615.

[32] Xu, D., jie Tian, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science 2* (2015), 165–193.