



**Politechnika  
Śląska**

## **PRACA MAGISTERSKA**

Granularne grupowanie danych w jedną grupę

**Krystian STASICA**

Nr albumu: 290779

**Kierunek:** Informatyka

**Specjalność:** Oprogramowanie Systemowe

**PROWADZĄCY PRACĘ**

**dr hab. inż. Krzysztof Simiński**

**KATEDRA ALGORYTMIKI I OPROGRAMOWANIA**

**Wydział Automatyki, Elektroniki i Informatyki**

**Gliwice 2024**



## **Tytuł pracy**

Granularne grupowanie danych w jedną grupę

## **Streszczenie**

Niniejsza praca magisterska, skupia się na tematyce na granularnego grupowania danych w jedną grupę. Praca analizuje i porównuje algorytmy klasteryzacji, w tym gęstościowe oraz algorytmy aglomeracyjne. Przeprowadzono eksperymenty, aby określić optymalne parametry tych algorytmów, oceniąc jakość grupowania na różnych zbiorach danych. Badania koncentrują się na wpływie parametrów klasteryzacji, wrażliwości na szum oraz efektywności obliczeniowej algorytmów, co pozwala na lepsze zrozumienie i zastosowanie tych technik w analizie danych.

## **Słowa kluczowe**

analiza skupień, DBSCAN, algorytmy aglomeracyjne, obliczenia ziarniste

## **Thesis title**

Granular one-group data clustering

## **Abstract**

This master thesis focuses on the topic of granular data grouping into a single group. The work analyzes and compares clustering algorithms, including density-based and agglomerative algorithms. Experiments were conducted to determine the optimal parameters of these algorithms, assessing the quality of clustering on various datasets. The research focuses on the impact of clustering parameters, sensitivity to noise, and computational efficiency of the algorithms, allowing for a better understanding and application of these techniques in data analysis.

## **Key words**

cluster analysis, DBSCAN, agglomerative algorithms, granular computing



# Spis treści

<b>1 Wstęp</b>	<b>1</b>
<b>2 Grupowanie danych i granule informacyjne</b>	<b>3</b>
2.1 Grupowanie danych . . . . .	3
2.1.1 DBSCAN . . . . .	4
2.1.2 Klastrowanie hierarchiczne . . . . .	7
2.2 Algorytm $k$ najbliższych sąsiadów . . . . .	12
2.3 Granulacja danych . . . . .	12
<b>3 Granularne grupowanie danych w jedną grupę</b>	<b>15</b>
3.1 Idea . . . . .	15
3.2 Kryteria wyboru parametrów klasteryzacji . . . . .	16
3.2.1 Kryteria dla DBSCAN . . . . .	16
3.2.2 Kryteria dla klastrowania hierarchicznego . . . . .	16
3.2.3 Opis kryteriów . . . . .	16
3.3 Zastosowanie KNN . . . . .	18
<b>4 Badania</b>	<b>21</b>
4.1 Zbiory danych . . . . .	21
4.1.1 Generowanie punktów . . . . .	22
4.2 Metodyka badań . . . . .	23
4.2.1 Miary jakości grupowania . . . . .	24
4.3 Wyniki badań . . . . .	24
4.3.1 Wpływ kryterium . . . . .	25
4.3.2 Wrażliwość na szum . . . . .	39
4.3.3 Czas wykonywania obliczeń . . . . .	48
<b>5 Podsumowanie</b>	<b>53</b>
<b>Bibliografia</b>	<b>58</b>
<b>Spis skrótów i symboli</b>	<b>61</b>

<b>Kody źródłowe</b>	<b>63</b>
<b>Spis rysunków</b>	<b>66</b>
<b>Spis tabel</b>	<b>67</b>

# Rozdział 1

## Wstęp

Grupowanie danych to proces organizowania zbioru danych w mniejsze podzbiory, zwane grupami, na podstawie określonych kryteriów lub cech wspólnych. Celem grupowania jest ułatwienie analizy i interpretacji danych, a także poprawa efektywności przetwarzania informacji. Metody grupowania danych można podzielić na kilka kategorii, w tym oparte na gęstości, gdzie grupy są formowane na podstawie obszarów o wysokiej gęstości punktów. Innym typem jest grupowanie hierarchiczne, które tworzy struktury drzewiaste przez sekwencyjne łączenie danych w coraz większe grupy lub dzielenie większej grupy na mniejsze. Z kolei grupowanie  $k$ -średnich wykorzystuje wyznaczenie  $k$  punktów jako centra grup (centroidów) i przydziela każdy punkt do najbliższego centroidu, tworząc grupy o zbliżonych charakterystykach w obrębie każdej z nich.

Problem grupowania danych w jedną grupę polega na braku innych grup, od których mogłaby się ona odróżniać, co znaczco utrudnia wyznaczenie jej granic. Bez wyraźnych punktów odniesienia trudno jest określić, które dane powinny być częścią grupy, a które stanowią szum. W takich przypadkach istnieje ryzyko, że granice będą zbyt szerokie lub zbyt wąskie, co może prowadzić do błędnych wniosków i nieefektywnej analizy. Wyzwaniem jest wyodrębnienie jednego dużego klastra, który będzie spójny wewnętrznie i dobrze odseparowany od szumu. Aby określić granice tego klastra, opracowano kryteria oparte na zasadzie uzasadnionej granulacji. Uwzględniają one, że im większa liczba punktów w grupie, tym wyższa wartość kryterium, a im większa średnia odległość między punktami w klastrze, tym mniejsza wartość kryterium. Dzięki temu granice klastra są jasno określone: musi on być duży i jednocześnie wewnętrznie jednorodny, co minimalizuje wpływ szumu i zapewnia, że klaster reprezentuje istotne dane.

Dodatkowo, aby zbadać wpływ różnych parametrów na proces grupowania, stworzono kilka kryteriów. Te różne kryteria obejmowały podnoszenie średniej odległości między punktami w klastrze do kwadratu oraz pierwiastkowanie liczby punktów w klastrze. Takie podejście pozwoliło na dokładniejsze zrozumienie, jak parametry wpływają na formowanie się klastrów i ich granice. W rezultacie możliwe było bardziej precyzyjne określenie optymalnych warunków do tworzenia dużych i jednorodnych klastrów, minimalizując jed-

nocześnie wpływ szumów na wyniki analizy.

Istnieje wiele sposobów na przeprowadzenie klasteryzacji, w tym metody takie jak DBSCAN (ang. *Density-Based Spatial Clustering of Applications with Noise*) oraz różne techniki klastrowania hierarchicznego, takie jak Single Linkage i Complete Linkage. Każda z tych metod ma swoje zalety i wady, które wpływają na ich skuteczność w różnych scenariuszach analizy danych. Na przykład, algorytm Single Linkage dobrze radzi sobie z tworzeniem wydłużonych klastrów, ale może być wrażliwy na błędy pomiarowe odległości między obserwacjami. Ponadto w pracy zmierzono efektywność różnych algorytmów klasteryzacji. W ramach badań przeprowadzono eksperymenty mające na celu określenie optymalnych parametrów algorytmów, oceniając jakość grupowania na różnych zbiorach danych, a także zbadano wrażliwości na szum oraz czas obliczeń.

Niniejsza praca składa się z kilku rozdziałów, w których szczegółowo omówiono użyte algorytmy i kryteria wyboru parametrów klasteryzacji, metodę badań, a także wyniki i ich analizę:

- Rozdział 1 wyjaśnia zakres problemu, cel pracy oraz motywacje.
- Rozdział 2 zawiera wprowadzenie do tematyki analizy skupień i jej znaczenia we współczesnej analizie danych.
- Rozdział 3 skupia się na opisie metod klasteryzacji i granulacji danych, które zostały użyte w badaniach.
- Rozdział 4 zawiera szczegółowy opis badań, wraz z opisem zbiorów danych, metodyki badań oraz z analizą wyników.
- Rozdział 5 skupia się na podsumowaniu pracy, wyciągnięciu wniosków oraz podsumowaniu wykonanej pracy.

## Rozdział 2

# Grupowanie danych i granule informacyjne

Istotne w niniejszej pracy magisterskiej było zbadanie zagadnienia granularnego grupowania danych w jedną grupę. Proces ten czasami zwany także klasteryzacją lub analizą skupień, polega na manipulacji na zbiorach danych w taki sposób, aby dane były organizowane w grupę oraz szum. Ważne jest, aby grupa (klaster) był spójny. Celem klasteryzacji jest identyfikacja wzorców lub wspólnych cech, co umożliwia lepszą ich prezentację, analizę oraz zrozumienie. W poniższym rozdziale zostaną opisane użyte algorytmy podczas badań.

### 2.1 Grupowanie danych

Grupowanie danych, znane również jako klasteryzacja (ang. *clustering*), analiza skupień (ang. *cluster analysis*), to technika organizowania zbiorów danych w taki sposób, aby elementy w jednym klastrze były bardziej podobne do siebie niż do elementów w innych klastrach. Podobieństwo to może być mierzone na różne sposoby, w zależności od specyfiki danych i celu analizy. W praktyce oznacza to, że dane są dzielone na mniejsze struktury (podzbiory) na podstawie określonych kryteriów, takich jak wspólne cechy, wartości lub wzorce. Proces jest techniką nienadzorowaną (ang. *unsupervised learning*), co oznacza, że nie wymaga wcześniejszego oznaczania danych etykietami lub przypisywania ich do konkretnych kategorii przez człowieka.

Wśród najczęściej wykorzystywanych metod grupowania znajdują się:

- grupowanie oparte na gęstości (ang. *density-based clustering*), identyfikuje się grupy jako obszary o wysokim zagęszczeniu punktów danych, oddzielone od siebie obszarami o niskim zagęszczeniu,
- grupowanie hierarchiczne (ang. *hierarchical clustering*), polegające na tworzeniu struktury drzewiastej (dendrogramu), w której mniejsze grupy są stopniowo łączone

w większe,

- grupowanie k-średnich (ang. *k-means clustering*), gdzie dane dzielone są na określoną liczbę grup  $k$  poprzez minimalizację sumy kwadratów odległości między punktami danych a centroidami grup.

Algorytmy grupowania danych znajdą zastosowanie różnych dziedzinach, od segmentacji klientów (ang. *Customer segmentation*) [14], po analizę dokumentów (ang. *Document clustering*) [4] czy analizę obrazów w medycynie, gdzie umożliwia wykrywanie guzów [1]. W niniejszej rozdziale przedstawione zostaną główne zalety oraz wady, które umożliwiając czytelnikowi lepsze zrozumienie ich działania i odpowiedni dobór w zależności od kontekstu.

### 2.1.1 DBSCAN

DBSCAN [7] jest jednym z popularnych algorytmów klastrowania, zaprojektowanym do identyfikacji skupisk punktów w zbiorze danych oraz wykrywania anomalii (punktów odstających). Jego główną zaletą jest zdolność do wykrywania klastrów o dowolnym kształcie oraz skuteczność w obecności szumu (ang. *noise*).

Algorytm działa w oparciu na dwóch podstawowych hiperparametrach:  $\varepsilon$ , który określa promień sąsiedztwa punktu, oraz minimalną liczbę punktów  $MinPts$  wymaganą do uznania danego regionu za gęsty. Ważnym aspektem działania algorytmu jest to, że metryka odległości używana do określania sąsiedztwa punktów jest dowolna. Oznacza to, że algorytm może korzystać z różnych miar odległości, takich jak:

- euklidesowa odległość jest „powszechnie stosowaną miarą w przypadku danych numerycznych”[25], ze względu na jej intuicyjność i szerokie zastosowanie w różnych dziedzinach analizy danych;
- Manhattan: miara odległości mająca problemy z wartościami nietypowymi [25];
- kosinusowa [6] (ang. *cosine similarity*): miara używana w przypadku danych wektorowych, takich jak dokumenty tekstowe [25] lub dane o wysokiej wymiarowości [6];

### Opis działania

Proces działanie algorytmu rozpoczyna się od wyboru niezbadanych punktów, a następnie dla każdego punktu  $P$  określa się jego sąsiedztwo na podstawie wartości  $\varepsilon$ . Jeśli sąsiedztwo punktu  $P$  zawiera co najmniej  $MinPts$ , punkt ten staje się punktem centralnym (ang. *core point*), a algorytm tworzy nowy klaster i dodaje do niego wszystkie punkty z sąsiedztwa. Algorytm przegląda sąsiedztwo tych punktów i, jeśli któryś z nich jest również punktem centralnym, rozszerza sąsiedztwo o punkty z jego sąsiedztwa.

---

```

1 function DBSCAN( points , eps , min_pts )
2   clusters  $\leftarrow \emptyset$ ; // Inicjalizujemy pusty zbiór klastrów ;
3   visited  $\leftarrow \emptyset$ ; // Inicjalizujemy pusty zbiór odwiedzonych punktów ;
4   noise  $\leftarrow \emptyset$ ; // Inicjalizujemy pusty zbiór punktów szumu ;
5
6   foreach point in points do
7     if point is not in visited then
8       add point to visited ;
9       neighbors  $\leftarrow$  find_neighbors( points , point , eps );
10      if the number of neighbors is less than min_pts then
11        add point to noise
12      else
13        new_cluster  $\leftarrow$  expand_cluster( points , point , neighbors ,
14          eps , min_pts , visited );
15        add new_cluster to clusters
16      end if ;
17    end if ;
18  end foreach ;
19
20  return clusters , noise
end function ;

```

---

Rysunek 2.1: Pseudokod algorytmu DBSCAN.

Proces ten trwa, aż wszystkie punkty w sąsiedztwie zostaną przypisane do klastra. Następnie algorytm przechodzi do następnego nieprzypisanego punktu i powtarza te kroki, aż wszystkie punkty zostaną przetworzone. Algorytm został przedstawiony w pseudokodzie na rys. 2.1.

### Zalety oraz wady algorytmu

Głównymi zaletami algorytmu są:

- **Klastry o dowolnym kształcie:** DBSCAN może identyfikować klastry o nieregularnych kształtach.
- **Automatyczne wykrywanie szumu:** Algorytm naturalnie identyfikuje punkty odosobnione, czyli szum (ang. *noise*).
- **Brak potrzeby określania liczby klastrów z góry:** W przeciwieństwie niektórych algorytmów takich jak np. k-Means, DBSCAN nie wymaga wstępnego określenia liczby klastrów.

Natomiast wadami określa się:

- **Czułość na parametry:** Wyniki algorytmu są silnie zależne od wyboru  $\varepsilon$  oraz  $MinPts$ . Niewłaściwe wartości mogą prowadzić do niepoprawnych wyników.

```
1 function find_neighbors(points , point , eps)
2     neighbors ← ∅; // Inicjalizujemy pusty zbiór sąsiadów ;
3
4     foreach p in points do
5         dx ← p.x – point.x;
6         dy ← p.y – point.y;
7         d_squared ← dx * dx + dy * dy // Obliczamy kwadrat odległości ;
8         if d_squared ≤ (eps * eps) then
9             add p to neighbors;
10            end if;
11        end foreach;
12        return neighbors;
13    end function;
14
15 function expand_cluster(points , point , neighbors , eps , min_pts ,
16     visited)
16     cluster ← {point}; // Inicjalizujemy klaster z punktem startowym;
17
18     foreach neighbor in neighbors do
19         if neighbor not in visited then
20             add neighbor to visited ;
21             new_neighbors ← find_neighbors(points , neighbor , eps);
22             if the number of new_neighbors ≥ min_pts then
23                 foreach new_neighbor in new_neighbors do
24                     if new_neighbor not in neighbors then
25                         add new_neighbor to neighbors
26                     end if;
27                 end foreach;
28             end if;
29         end if;
30         if neighbor not in any cluster then
31             add neighbor to cluster
32         end if;
33     end foreach;
34     return cluster
35 end function;
```

---

Rysunek 2.2: Pseudokod funkcji rozszerzania klastra oraz wyszukiwania sąsiadujących punktów w obszarze  $\varepsilon$ .

- **Różne wyniki:** Wynik działania algorytmu może się różnić w zależności od kolejności przetwarzania danych
- **Wysoka złożoność obliczeniowa dla dużych zbiorów danych:** Mimo optymalizacji, dla bardzo dużych zbiorów danych algorytm może być czasochłonny.

### 2.1.2 Klastrowanie hierarchiczne

Algorytmy Single Linkage 2.1.2 oraz Complete Linkage 2.1.2 to dwa popularne podejścia używane w grupowaniu hierarchicznym (ang. *hierarchical clustering*), a dokładniej aglomeracyjnym. Każde z tych podejść definiuje sposób, w jaki odległość między klastrami jest mierzona podczas ich łączenia. Oba te algorytmy działają na podstawie macierzy odległości pomiędzy punktami danych, ale różnią się sposobem, w jaki mierzą odległość pomiędzy klastrami. Służą one do łączenia klastrów w jeden, domyślnie nie uwzględniają szumu.

#### Single Linkage

*Single Linkage* [9], zwany także pojedynczym połączeniem, mierzy odległość między dwoma klastrami jako najmniejszą odległość między dowolnymi dwoma punktami należącymi do tych klastrów. W praktyce algorytm działa w następujący sposób:

1. Początkowo każdy punkt danych jest traktowany jako osobny klaster.
2. Znajduje się dwa najbliższe klastry, mierząc odległości pomiędzy wszystkimi parami klastrów i wybierając te, które mają najmniejszą odległość pomiędzy swoimi najbliższymi punktami.
3. Łączy się wybrane klastry w jeden klaster.
4. Powtarza się krok 2 oraz krok 3, aż wszystkie punkty zostaną połączone w jeden duży klaster lub do momentu osiągnięcia pożąданej liczby klastrów.

Algorytm został przedstawiony w pseudokodzie na rys. 2.3.

#### Complete Linkage

*Complete Linkage* [9], zwany także kompletnym połączeniem, mierzy odległość między dwoma klastrami jako największą odległość między dowolnymi dwoma punktami należącymi do tych klastrów. Algorytm działa w podobny sposób jak Single Linkage, ale z inną miarą odległości:

1. Początkowo każdy punkt danych jest traktowany jako osobny klaster.

```
1 function SingleLinkage( points , n ) :
2     points : set of points
3     n: number of points
4
5     // wyznaczamy odleglosci kazdy z kazdym ;
6     for i  $\leftarrow$  1 to n do
7         for j  $\leftarrow$  1 to n do
8             d[i][j]  $\leftarrow$  d[j][i]  $\leftarrow$  distance(X[i] , X[j]) ;
9         end for;
10    end for;
11
12    // w macierzy d wyszukujemy minimum ;
13    min_distance  $\leftarrow$   $\infty$ 
14    cl1  $\leftarrow$  null
15    cl2  $\leftarrow$  null
16    for i = 0 to n-1:
17
18        // tylko jedna stronę przekątnej ;
19        for j = i+1 to n-1:
20            if d[i][j] < min_distance:
21                min_distance  $\leftarrow$  d[i][j];
22                cl_i  $\leftarrow$  i;
23                cl_j  $\leftarrow$  j;
24            end if;
25        end for;
26    end for;
27
28    // scalanie najblizszych klastrów ;
29    merg  $\leftarrow$  cl_i;
30    for i  $\leftarrow$  k to n do:
31        d[merg][k]  $\leftarrow$  d[k][merg]  $\leftarrow$  min(d[cl_i][k] , d[cl_j][k]);
32    end for;
33
34    // usuwanie klastrów ;
35    for i  $\leftarrow$  0 to n-1 do:
36        d[i][cl_j]  $\leftarrow$   $\infty$ 
37        d[cl_j][i]  $\leftarrow$   $\infty$ 
38    end for
39
40    return d
41 Zwracamy macierz odległości ;
42
43 end function;
```

Rysunek 2.3: Algorytm Single Linkage łączy punkty w klastry, zaczynając od najbliższych sobie par, a kończąc na pojedynczym skupisku obejmującym wszystkie punkty.

2. Znajduje się dwa najbliższe klastry, mierząc odległości pomiędzy wszystkimi parami klastrów i wybierając te, które mają najmniejszą maksymalną odległość pomiędzy swoimi najdalszymi punktami.
3. Łączy się wybrane klastry w jeden klaster.
4. Powtarza się kroki 2 i 3, aż wszystkie punkty zostaną połączone w jeden duży klaster lub do momentu osiągnięcia pożądanej liczby klastrów.

Algorytm został przedstawiony w pseudokodzie na rys. 2.4.

### Zalety oraz wady algorytmów Single Linkage i Complete Linkage

Algorytmy Single Linkage i Complete Linkage posiadają różne cechy, które wpływają na ich skuteczność w różnych scenariuszach analizy danych [17].

#### Zalety Single Linkage

- Skuteczność na różnych typach danych: Algorytm dobrze działa na danych niekuliastych, co czyni go użytecznym w różnorodnych scenariuszach analizy [11].
- Radzenie sobie z wydłużonymi klastrami: Algorytm może tworzyć grupy o dowolnym kształcie.

#### Wady Single Linkage

- Efekt łańcuchowy: Punkty mogą być dodawane do klastra przez długi ciąg połączeń najbliższych sąsiadów, co może prowadzić do tworzenia klastrów, które niekoniecznie są podobne do siebie w ogólnym sensie, ale są połączone poprzez serię bliskości do swoich najbliższych sąsiadów [17].
- Wrażliwość na błędy odległości: Metoda Single Linkage może być wrażliwa na błędy w pomiarach odległości między obserwacjami [8], co może prowadzić do nieprecyzyjnych lub niewłaściwych wyników grupowania, szczególnie w przypadku niedokładnych danych.

#### Zalety Complete Linkage

- Jednorodne klastry: Algorytm generuje bardziej jednorodne klastry, co oznacza, że klastry mają zbliżony rozmiar i kształt, a elementy wewnętrz każdego klastra są do siebie bardziej podobne.
- Unikanie wydłużonych struktur: Jest mniej podatny na efekt łańcuchowy, co prowadzi do bardziej zwartych klastrów.

```
1 function CompleteLinkage( points , n):
2     points : set of points
3     n: number of points
4
5     // wyznaczamy odleglosci kazdy z kazdym ;
6     for i ← 1 to n do
7         for j ← 1 to n do
8             d[i][j] ← d[j][i] ← distance(X[ i ] , X[ j ]) ;
9         end for;
10    end for ;
11
12    // w macierzy d wyszukujemy minimum ;
13    min_distance ← ∞
14    cl1 ← null
15    cl2 ← null
16    for i = 0 to n-1:
17
18        // tylko jedną stronę przekątnej ;
19        for j = i+1 to n-1:
20            if d[i][j] < min_distance:
21                min_distance ← d[i][j];
22                cl_i ← i;
23                cl_j ← j;
24            end if;
25        end for;
26    end for ;
27
28    // scalanie najblizszych klastrów ;
29    merg ← cl_i;
30    for i ← k to n do:
31        d[merg][k] ← d[k][merg] ← max(d[cl_i][k] , d[cl_j][k]);
32    end for ;
33
34    // usuwanie klastrów ;
35    for i ← 0 to n-1 do:
36        d[ i ][ cl_j ] ← ∞
37        d[ cl_j ][ i ] ← ∞
38    end for
39
40    return d
41 Zwracamy macierz odległości ;
42
43 end function;
```

---

Rysunek 2.4: Algorytm Complete Linkage łączy punkty w klastry na podstawie największej odległości między punktami w klastrach, zaczynając od najbliższych sobie par, a kończąc na pojedynczym skupisku obejmującym wszystkie punkty.

- Gwarancja odległości: Jeśli odległość między klastrami wynosi  $d$ , to dla dowolnych punktów budujących te klastry odległość nie będzie większa niż  $d$ , czego Single Linkage nie gwarantuje.

### Wady Complete Linkage

- Zniekształcenie kształtu klastrów: Istnieje ryzyko, że algorytm klasteryzacji spowoduje powstanie klastrów o nieregularnych formach, które nie odzwierciedlają prawdziwych struktur danych, zwłaszcza że algorytm ten często tworzy kształty kuliste [8].
- Wrażliwość na szum: Algorytm jest wrażliwy na szum i nietypowe wartości, podobnie jak Single Linkage.

### Złożoność obliczeniowa

Złożoność obliczeniową algorytmu Single Linkage wynosi w wersji naiwnej  $O(n^3)$  [19], ponieważ każda operacja polega na znajdowaniu najmniejszej odległości między każdą możliwą parą skupień, a następnie łączeniu skupień o najmniejszej odległości. W każdej iteracji, gdzie mamy  $n - 1$  takich operacji, każda z nich może wymagać przeglądu  $O(n^2)$  par, co prowadzi do złożoności  $O(n^3)$  dla całego procesu grupowania. W przypadku SLINK [26], który jest bardziej zaawansowaną wersją Single Linkage, złożoność obliczeniowa jest redukowana do  $O(n^2)$ . Uzyskano to dzięki użyciu specjalnej struktury danych, która efektywnie zarządza najbliższymi sąsiadami i odległościami, pozwalając na znacznie szybsze i bardziej efektywne przetwarzanie grupowania. SLINK aktualizuje te odległości tylko raz, podczas inicjalizacji, i utrzymuje odpowiednie wskaźniki do najbliższych elementów przez cały proces łączenia, co wydatnie obniża nakład obliczeniowy w porównaniu do tradycyjnych metod. Istnieje także algorytm CLINK [5] opracowany na wzór SLINK, który także osiąga złożoność  $O(n^2)$ .

Complete Linkage także ma złożoność  $O(n^3)$  w swojej najprostszej formie. Wymaga to obliczenia maksymalnych odległości między wszystkimi parami skupień w każdej iteracji, co podobnie do Single Linkage, skutkuje potrzebą ciągłego aktualizowania i przeszukiwania odległości, aby znaleźć największą odległość między najdalszymi elementami każdego skupienia.

Jeżeli założymy, że obliczenie metryki między dwoma punktami wymaga czasu  $O(d)$ , gdzie  $d$  to wymiarowość przestrzeni danych. Wówczas całkowita złożoność czasowa algorytmu Single Linkage oraz Complete Linkage w wersji naiwnej wynosiłaby  $O(n^3d)$ , gdzie  $n$  to liczba punktów danych.

Podobnie jak w przypadku innych algorytmów grupowania, w *Single Linkage* i *Complete Linkage* można stosować dowolne metryki odległości. Oznacza to, że można używać miar takich jak: euklidesowa, kosinusowa czy metryka Manhattan. Dzięki możliwości

wyboru metryki odległości, algorytmy Single Linkage i Complete Linkage są elastyczne i mogą być dostosowane do różnych typów danych oraz specyficznych wymagań analitycznych.

## 2.2 Algorytm $k$ najbliższych sąsiadów

Algorytm  $k$  najbliższych sąsiadów (KNN) jest prostym, nieparametrycznym algorytmem uczenia maszynowego, który może być stosowany zarówno w zadaniach klasyfikacji, jak i regresji. Został opracowany przez Evelyn Fix i Josepha Hodgesa w 1951 roku [10], a później został dopracowany przez Thomasa Covera [3]. Jest to popularna metoda, która polega na znalezieniu z góry zdefiniowanej liczby próbek treningowych najbliższych nowemu punktowi oraz przewidywaniu etykiety na ich podstawie. Liczba próbek może być stała (uczenie  $k$  najbliższych sąsiadów) lub może się zmieniać w zależności od lokalnej gęstości punktów (uczenie oparte na promieniu). W ogólności odległość może być mierzona dowolną metryką, choć najczęściej stosowaną jest odległość euklidesowa.

Warto zauważyć, że pomimo swej prostoty, algorytm najbliższych sąsiadów odnosi sukcesy w wielu problemach klasyfikacji [27] i regresji [27], w tym w wykrywaniu pisma ręcznego [29] w tym cyfr [16]. Stosuje się go w badaniach nad rozpoznawaniem scen z obrazów satelitarnych [18]. Jest także często stosowany w sektorze bankowym do obliczenia zdolności kredytowej (ang. *creditworthiness assessment*) poszczególnych osób, porównując je z osobami o podobnych cechach [13]. Istnieją pewne kwestie, takie jak wybór odpowiedniej wartości parametru  $k$  oraz metryki, które mogą mieć wpływ na jego skuteczność i wydajność. W zastosowaniach praktycznych konieczne może być również odpowiednie przetwarzanie danych wejściowych, aby uniknąć błędów wynikających z różnic w ich skali, gdyż algorytm „KNN używa funkcji odległości, więc wymogiem jest, aby każda cecha była skalowana w podobny sposób” [20], ponadto algorytm lepiej radzi sobie gdy dane są małowymiarowe, gdyż ma problem z wielowymiarowymi danymi (ang. *High-dimensional data* [28]).

## 2.3 Granulacja danych

Granulacja danych stanowi kluczowy etap w analizie i interpretacji zbiorów danych, gdzie celem jest podział danych na bardziej spójne i łatwiejsze do analizy jednostki. W kontekście granulacji istotne jest zrozumienie struktury danych oraz identyfikacja naturalnych grup lub wzorców występujących w zbiorze. Ten proces umożliwia redukcję złożoności danych, ułatwiając jednocześnie analizę i ekstrakcję istotnych informacji. Granulacja danych może być realizowana za pomocą różnych metod, takich jak dyskretyzacja [12], grupowanie [24] czy segmentacja [2], zależnie od charakteru danych oraz celu analizy. W konsekwencji skuteczna granulacja danych stanowi kluczowy element w procesie

wyodrębniania znaczących zależności i struktur w danych.

Zasada uzasadnionej granulacji (ang. *principle of justifiable granularity*) [21] to reguła projektowania granul informacji w systemach informacyjnych, oparta na dowodach eksperymentalnych. Granule są formowane tak, by były jednocześnie dobrze uzasadnione i precyzyjnie określone semantycznie. W badaniu [23] termin „uzasadniony” odnosi się do konstrukcji granuli informacyjnej, która jest tworzona tak, aby była w pełni uzasadniona w świetle dowodów eksperymentalnych oraz by posiada dobrze sformułowaną semantykę, co oznacza, że każde ziarno informacyjne jest wyraźnie zdefiniowane, co umożliwia jego jednoznaczne zrozumienie i odróżnienie od innych, podobnych elementów. Dodatkowo te ziarna mogą być organizowane w hierarchiczne struktury, pozwalając na skuteczniejsze zarządzanie i analizę danych na różnych poziomach szczegółowości.

### **Ziarno informacyjne**

Ziarno informacyjne lub granula informacyjna (ang. *information granule*) to jednostka informacji [22] (obiekt) lub zbiór abstrakcyjnych jednostek informacji (obiektów) zebraanych razem na podstawie kryteriów takich jak nieodróżnialność, podobieństwo, bliskość lub funkcjonalność. Pozwala to na uproszczenie, reprezentację i przetwarzanie informacji czy zjawisk na wyższym poziomie abstrakcji [15]. Na przykład, ziarna ludzkiej głowy obejmują czoło, nos, policzki, uszy, oczy itp. Granulacja informacji jest hierarchiczna, co oznacza, że większe ziarna mogą być podzielone na mniejsze, bardziej szczegółowe jednostki. W kontekście obliczeń granularnych ziarna informacyjne odgrywają kluczową rolę w analizie danych poprzez grupowanie informacji w łatwiejsze do zarządzania jednostki, co poprawia efektywność przetwarzania i interpretacji danych. Dzięki temu możliwe jest efektywne zarządzanie złożonymi zbiorami danych.



# Rozdział 3

## Granularne grupowanie danych w jedną grupę

Kluczowym aspektem w procesie grupowania danych jest wybór odpowiednich parametrów algorytmu, które mają istotny wpływ na skuteczność oraz interpretowalność wyników. W niniejszym rozdziale zostaną przedstawione kryteria wyboru parametrów dla algorytmów klasteryzacji, takich jak DBSCAN (roz. 2.1.1) oraz klastrowania hierarchicznego (roz. 2.1.2) z wykorzystaniem Single Linkage i Complete Linkage. Przedstawiono również zastosowanie algorytmu  $k$  najbliższych sąsiadów (KNN) (roz. 2.2) w kontekście analizy średniej odległości między punktami, co stanowi kluczowy element procesu doboru parametrów dla niektórych algorytmów grupowania. Opisane zostaną także kryteria generowania punktów.

### 3.1 Idea

Idea badania polegała na znalezieniu zgrupowania danych w jedną grupę, a niezgrupowane elementy traktowane były jako szum. W tym celu konieczne było znalezieniu odpowiednich parametrów dla algorytmów DBSCAN oraz hierarchicznych Single Linkage i Complete Linkage. W przypadku algorytmu DBSCAN, który wspiera szum należało znaleźć parametry w taki sposób, aby została wykryta jedna grupa i szum. W przypadku algorytmów hierarchicznych, które łączą klastry, nie wykrywając przy tym szumu, należało opracować metodę jego wykrywania. W tym celu badano moment przerwania tych algorytmów, oraz sposób łącznie klastrów w taki sposób, że największy klaster jest traktowany jako ten właściwy (grupa), a wszystkie pozostałe są traktowane jako szum.

Na początku przygotowywano dane, zbiory składały się z szumu oraz z kształtu, który miał zostać wykryty, zbiory zostały opisane w roz. 4.1. Następnie określano zestaw parametrów oraz uruchamiano algorytmy grupowania w pętli dla różnych kombinacji tych parametrów. Parametry oraz kryteria oceniające grupowanie zostały opisane w rozdzia-

łach 3.2.1 (DBSCAN), oraz roz. 3.2.2 (grupowanie hierarchiczne).

Każde uruchomienie algorytmu skutkowało wygenerowaniem klastrów, które były następnie oceniane za pomocą własnych kryteriów, a wyniki zapisano do pliku `*.csv`. Następnym krokiem były dla każdego użytego kryterium, odczytanie wartości z pliku `*.csv` i uruchomienie algorytmu, aby wygenerować w plik `*.png`. W ten sposób uzyskiwano jeden plik dla każdej miary. Dodatkowo mierzono czas wykonywania się algorytmów, co zaprezentowano w tabelach 4.1, 4.2 oraz 4.3.

## 3.2 Kryteria wyboru parametrów klasteryzacji

W celu efektywnego doboru parametrów dla algorytmu DBSCAN oraz klastrowania hierarchicznego z wykorzystaniem Single Linkage i Complete Linkage zastosowano szereg miar oceny jakości klasteryzacji. W przypadku kryteria DBSCAN używane są miary oparte na ocenie struktury klastrów oraz na liczbie wykrytych klastrów. Natomiast dla klastrowania hierarchicznego miary koncentrują się na ocenie momentu przerwanie algorytmu, a więc gdzie punkty poza głównym klastrem traktowane są jako szum.

### 3.2.1 Kryteria dla DBSCAN

Metoda DBSCAN opiera się na dwóch głównych parametrach:  $\varepsilon$  i  $MinPts$ . W celu wyboru optymalnych wartości tych parametrów zaimplementowano metodę `dbscanLoop`. W tej metodzie przeszukiwane są różne kombinacje wartości  $\varepsilon$  (od 2 do 20 z krokiem 0.2) i  $MinPts$ , (od 20 do 150 z krokiem 1) a następnie oceniana jest jakość klasteryzacji za pomocą następujących miar opisanych w podrozdziale 3.2.3.

### 3.2.2 Kryteria dla klastrowania hierarchicznego

W przypadku klastrowania hierarchicznego z wykorzystaniem Single Linkage i Complete Linkage, głównym celem jest wybór odpowiedniego momentu dla przerwania grupowania, aby uzyskać pożądaną liczbę klastrów. W tym wypadku chciano uzyskać dwa klastry. Największy kластer pod względem liczby punktów zostanie określony jako ten właściwy, a pozostałe klastry jako szum. Do tego celu wykorzystuje się miary oceny jakości klasteryzacji opisanych w podrozdziale 3.2.3.

### 3.2.3 Opis kryteriów

Poniżej opisano piętnaście miar od  $P_1$  do  $P_{3,k9}$ , które obejmują trzy kategorie:  $P_1$ ,  $P_2$  oraz  $P_3$ , a także ich wariacje różniące się sposobem definiowania średniej odległości między punktami  $\bar{d}$ , zdefiniowanej w równaniu (1). Te miary realizują zasadę uzasadnionej

granulacji, ponieważ są obliczane w taki sposób, że wartość miary rośnie wraz ze wzrostem liczby punktów w klastrze oraz maleje, gdy zmniejsza się średnia odległość między punktami (lub między  $k$  najbliższymi punktami) w klastrze.

1.  $P_1$ : Liczba punktów  $N$  pomnożona przez odwrotność średniej odległości między punktami  $\bar{d}$ , oznaczona jako  $P_1$ , jest zdefiniowana jako:

$$P_1 = N \times \frac{1}{\bar{d}}. \quad (3.1)$$

2.  $P_2$ : Liczba punktów  $N$  pomnożona przez kwadrat odwrotnej średniej odległości między punktami  $\bar{d}$ , oznaczona jako  $P_2$ , jest zdefiniowana jako:

$$P_2 = N \times \left(\frac{1}{\bar{d}}\right)^2. \quad (3.2)$$

3.  $P_3$ : Pierwiastek liczby punktów  $N$  pomnożony przez kwadrat odwrotnej średniej odległości między punktami  $\bar{d}$ , oznaczona jako  $P_3$ , jest zdefiniowana jako:

$$P_3 = \sqrt{N} \times \left(\frac{1}{\bar{d}}\right)^2. \quad (3.3)$$

gdzie  $N$  to liczba punktów w klastrze, a  $\bar{d}$  to średnia odległość między punktami liczona jako:

1. średnia odległość między wszystkimi punktami w wykrytym skupieniu (grupie),

$$\bar{d} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^N d(i,j). \quad (3.4)$$

2. średnia odległość między  $k \in \{3, 5, 7, 9\}$  najbliższymi punktami w wykrytym skupieniu (grupie),

**Wzory miar dla średniej KNN** Dla jasności nazewnictwa kryteriów, warto rozróżnić wzory gdzie,  $k \in \{3, 5, 7, 9\}$ :

1.  $P_{1,k}$ : Liczba punktów  $N$  pomnożona przez odwrotność średniej odległości między punktami  $\bar{d}$ , oznaczona jako  $P_{1,k}$ , jest zdefiniowana jako:

$$P_{1,k} = N \times \frac{1}{\bar{d}_k}. \quad (3.5)$$

2.  $P_{2,k}$ : Liczba punktów  $N$  pomnożona przez kwadrat odwrotności średniej odległości między punktami  $\bar{d}$ , oznaczona jako  $P_{2,k}$ , jest zdefiniowana jako:

$$P_{2,k} = N \times \left( \frac{1}{\bar{d}_k} \right)^2. \quad (3.6)$$

3.  $P_{3,k}$ : Pierwiastek liczby punktów  $N$  pomnożony przez kwadrat odwrotności średniej odległości między punktami  $\bar{d}$ , oznaczona jako  $P_{3,k}$ , jest zdefiniowana jako:

$$P_{3,k} = \sqrt{N} \times \left( \frac{1}{\bar{d}_k} \right)^2. \quad (3.7)$$

Otrzymano w ten sposób łącznie 15 kryteriów od  $P_1$  do  $P_{3,9}$ , ponieważ mamy 5 średnich ( $\bar{d}, \bar{d}_k$ , gdzie  $k \in \{3, 5, 7, 9\}$ ) i 3 typy kryteriów ( $P_1, P_2, P_3$ ). Trzy kryteria dla średniej  $\bar{d}$  oraz dwanaście dla średniej  $\bar{d}_k$ . Średnia odległość między  $k$  najbliższymi punktami w wykrytym skupieniu została obliczona przy użyciu algorytmu KNN opisanego w rozdziale 2.2.

Dla przykładu, oznaczenia dla  $k = 3$  wyglądają następująco:

$$P_{1,3} = N \times \frac{1}{\bar{d}_3}. \quad (3.8)$$

$$P_{2,3} = N \times \left( \frac{1}{\bar{d}_3} \right)^2. \quad (3.9)$$

$$P_{3,3} = \sqrt{N} \times \left( \frac{1}{\bar{d}_3} \right)^2. \quad (3.10)$$

Są to miary, gdzie użyto średniej odległości w zbiorze między trzema najbliższymi punktami  $\bar{d}_3$ .

### 3.3 Zastosowanie KNN

W niniejszym badaniu wykorzystano algorytm  $k$  najbliższych sąsiadów (KNN) do zbadania średniej odległości między  $k$  najbliższymi punktami w danym zbiorze punktów. Ta miara została następnie użyta w algorytmie DBSCAN jako jeden z parametrów decydujących o właściwych parametralach tego algorytmu, a także w algorytmach Single Linkage i Complete Linkage do określenia warunków przerwania tych algorytmów.

Wykorzystanie algorytmu można w następujących krokach:

1. **wybór liczby sąsiadów ( $k$ ):** Na początku ustalana jest liczba najbliższych sąsiadów  $k$ , którą należy uwzględnić w analizie. Podczas badania testowanie różne wartości zaczynając na  $k = 3$ , a kończąc na  $k = 15$ . Ostatecznie badano dla  $k \in \{3, 5, 7, 9\}$  ponieważ większe  $k$  nie dawały zauważalnych różnic;
2. **obliczenie odległości:** Dla każdego punktu  $x$  w zbiorze danych obliczana jest odległość od każdego innego punktu w zbiorze. Popularnymi stosowanymi metrykami odległości są np. odległość euklidesowa, Manhattan czy Minkowskiego. W pracy użyto metryki euklidesowej;
3. **wybór najbliższych sąsiadów:** Dla każdego punktu wybierane są jego  $k$  najbliższych sąsiadów na podstawie obliczonych odległości;
4. **obliczenie średniej odległości:** Dla każdego punktu obliczana jest średnia odległość do jego  $k$  najbliższych sąsiadów;
5. **obliczenie ogólnej średniej odległości:** Obliczana jest średnia wartość ze wszystkich średnich odległości punktów w zbiorze danych;

Wynikiem działania algorytmu jest średnia odległość między  $k$  punktami, która może być wykorzystana jako parametr dla DBSCAN / algorytmów hierarchicznych. W przypadku DBSCAN oblicza się  $k$  najbliższych sąsiadów tylko dla punktów w klastrze, co prowadzi do dokładniejszego doboru parametrów. Pseudokod działania algorytmu został przedstawiony na rys. 3.1.

```
1 function KNN(data , k_values) :
2     results  $\leftarrow \emptyset$  ; // Inicjalizujemy pusty zbiór ;
3
4     foreach k in k_values:
5         overall_average_distance  $\leftarrow 0$ 
6         sum_of_average_distances  $\leftarrow 0$ 
7         foreach point i in data:
8             //Liczymy odległość od tego punkty do każdego innego punktu ;
9             for each other_point j in data from i+1 to end do
10                if point i is not equal to point j then
11                    distance  $\leftarrow$  calc_dist(point i , other_point j )
12                    append(distance) to distances
13                end if
14            end for
15            // Sortujemy dane ;
16            sort(distances)
17
18            // Wybieramy k najbliższych sąsiadów ;
19            sum  $\leftarrow 0$ ;
20            for i from i to k do
21                sum  $\leftarrow$  sum + distances [ i ]
22            end for;
23            // Obliczamy średnią odległość do k najbliższych sąsiadów ;
24            average_distance  $\leftarrow$  sum / k;
25
26            // Obliczamy średnią odległość dla każdego punktu ;
27            sum_of_average_distances  $\leftarrow$ sum_of_average_distances
28            + average_distance
29        end foreach
30
31        // Obliczamy średnią średnich odległości dla zbioru ;
32        results [k]  $\leftarrow$  sum_of_average_distances / size (data)
33    end foreach
34
35    return results
end function
```

---

Rysunek 3.1: Algorytm oblicza średnią odległość do  $k$  najbliższych sąsiadów w zbiorze punktów klastra dla różnych wartości  $k$ .

# Rozdział 4

## Badania

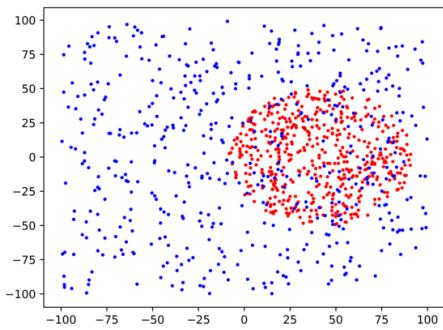
W rozdziale poświęconym badaniom skupiono się na analizie i doborze optymalnych hiperparametrów dla algorytmu DBSCAN (roz. 2.1.1), ( $\varepsilon$ , oraz  $MinPts$ ) odpowiednich parametrów przerwania  $max_d$  dla algorytmów aglomeracyjnych (roz. 2.1.2), takich jak metoda Single Linkage czy Complete Linkage. Celem tego procesu była maksymalizacja efektywności grupowania danych, poprzez eksperymentalne sprawdzanie różnorodnych kombinacji parametrów w wyznaczonych przedziałach.

Znaczącą część analizy stanowiła walidacja wyników grupowania. Użyto do tego celu różnych kryteriów (roz. 3.2), takich jak liczba punktów w klastrze oraz średnia odległość między punktami, co pozwoliło na ocenę skupisk. Dodatkowo zastosowano bardziej złożone metody oceny jakości klastrów, w tym obliczenia oparte na zmianie takich parametrów jak pierwiastek liczby punktów, kwadrat średniej odległości, oraz wykorzystanie metody  $k$  najbliższych sąsiadów (KNN) (roz. 2.2) do oceny średniej odległości między najbliższymi punktami w klastrze.

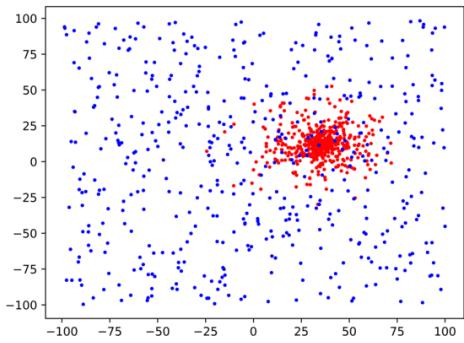
### 4.1 Zbiory danych

Zbiory danych były tworzone specjalnie dla potrzeb badania. Punkty w zbiorach miały współrzędne miesiące się w przedziale od  $-100$  do  $100$ . Początkowo generowano jednorodny szum, a następnie tworzono zageszczenia, czyli skupiska punktów o określonym kształcie. Możliwe kształty to między innymi koło, które mogło być formowane zgodnie z jednorodnym rozkładem, rozkładem normalnym oraz rozkładzie normalnym kątowym (rozkład von Misesa) lub w formie pierścienia. Wprowadzono także własne, niestandardowe kształty przypominające kotwicę czy literę 'X'.

Dodatkowo stworzono osobną funkcję do zapisywania ruchów myszy na ekranie, co umożliwiło tworzenie własnych kształtów. W sumie generowano 1000 punktów, przy czym parametry szumu wały się od 30% do 70%, a wielkość koła (tj. jego promień) była zmienna. Pozycja koła lub innego wybranego kształtu była losowana tak, aby zawsze znajdować się w granicach planszy  $(-100, 100)$  i nie wystawać poza nią nawet częściowo.



Rysunek 4.1: Koło jednorodne



Rysunek 4.2: Rozkład von Misesa

#### 4.1.1 Generowanie punktów

Algorytm generowania punktów został opracowany w celu przygotowania danych do eksperymentów z algorytmami klastrowania. W badaniu wykorzystano kilka różnych metod generowania punktów, z których każda symuluje inną strukturę danych.

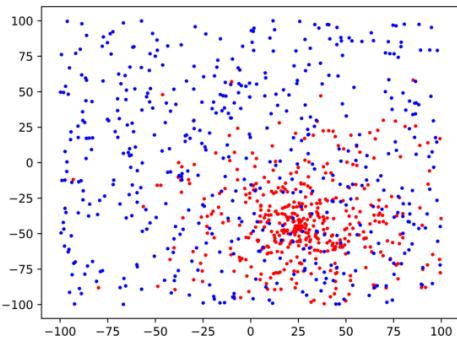
##### Generowanie punktów na okręgu

Pierwszą metodą generowania punktów jest generowanie punktów na okręgu o zadanym promieniu. Wykorzystując funkcję `generate_points_circle`, można wygenerować zestaw punktów rozmieszczonych na okręgu o określonym promieniu. Metoda ta dodatkowo uwzględnia obecność szumu, generując losowe punkty w obrębie wyznaczonego obszaru. Kolejną metodą jest generowanie punktów na okręgu z wykorzystaniem rozkładu von Misesa. Funkcja `generate_points_mises` generuje punkty na okręgu, których rozkład podlega rozkładowi von Misesa. Podobnie jak w poprzedniej metodzie, możliwe jest dodanie szumu przez generowanie losowych punktów w określonym obszarze. Opisane kształty przedstawiają wykres 4.1, wygenerowanym za pomocą funkcji `plot_points`.

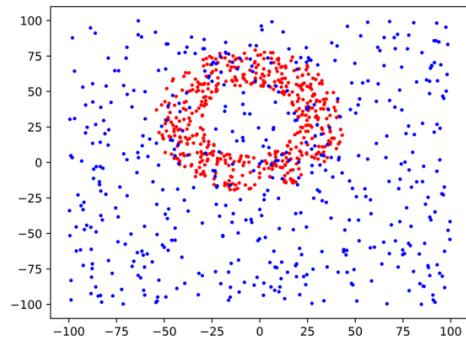
Trzecią metodą jest generowanie punktów na okręgu z wykorzystaniem rozkładu normalnego. Funkcja `generate_points_normal` generuje punkty na okręgu, których odległość od środka podlega rozkładowi normalnemu rys. 4.3. Podobnie jak w poprzednich przypadkach, istnieje szum. Czwartą metodą jest generowanie punktów w kształcie pierścienia (ang. *ring*). Funkcja `generate_points_ring` generuje punkty w postaci pierścienia, który jest tworzony podobnie jak koło jednorodne. Po wygenerowaniu punktów można je zobaczyć na wykresie 4.4, wygenerowanym za pomocą funkcji `plot_points`

##### Generowanie punktów w kształcie

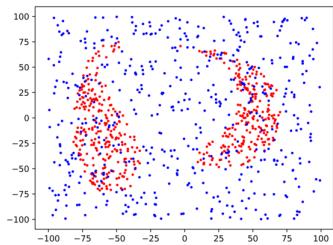
Ostatnią metodą jest generowanie punktów w kształcie zdefiniowanym przez użytkownika. Funkcja `generate_points_in_shape` generuje punkty wewnątrz podanego kształtu, który jest wczytywany z pliku. Może to być dowolny kształt opisany za pomocą ścieżki.



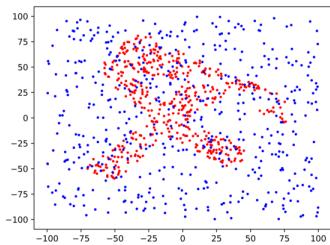
Rysunek 4.3: Rozkład normalny



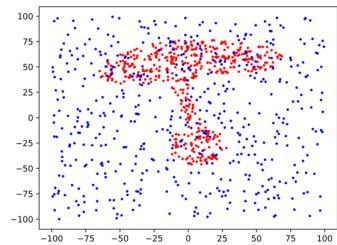
Rysunek 4.4: Kształt pierścienia



Rysunek 4.5: Kształt półksiężyca, (shape\_A)



Rysunek 4.6: Kształt „X”, (shape\_B)



Rysunek 4.7: Kształt grzyba, (shape\_C)

Podczas badań wykorzystano trzy kształty, są to kolejno kształt dwóch półksiężyćów (`shape_A`), kształt litery „X” (`shape_B`), oraz kształt grzyba lub kotwicy (`shape_C`). Po wygenerowaniu punktów można je zobaczyć na wykresie 4.3 które zostały wygenerowane za pomocą funkcji `plot_points`.

### Zapis punktów do pliku

Na zakończenie procesu generowania punktów, możliwe jest zapisanie ich do pliku tekstowego za pomocą funkcji `save_points_to_file`. Ta funkcja zapisuje współrzędne punktów `*.csv`, co pozwala na dalszą analizę lub wykorzystanie w innych eksperymentach.

## 4.2 Metodyka badań

Badania skoncentrowane były na optymalizacji parametrów dla algorytmów grupowania DBSCAN roz. 2.1.1 oraz algorytmów aglomeracyjnych Single Linkage i Complete Linkage. Szczegółowe kryteria doboru tych parametrów zostały opisane w rozdziale 3.2.3. Do eksperymentów użyto zagnieżdżonych pętli, w których zmieniano kolejno pojedyncze parametry, co pozwalało na przetestowanie wszystkich możliwych kombinacji w ustalonym zakresie.

Podczas badania z algorymem DBSCAN kluczowymi parametrami były  $\varepsilon$  i  $MinPts$ .

Wartość  $\varepsilon$  była modyfikowana w zakresie od 2 do 20.0, z krokiem 0.2, natomiast  $MinPts$  zmieniano w zakresie od 20 do 150, z krokiem 1. Algorytm ten obsługiwał szum. W metodach hierarchicznych początkowo tworzono jeden klaster, ponieważ te metody nie obsługują szumu. Kluczowe były odpowiednie przerwanie algorytmu i połączenie pozostałych klastrów tak, aby uzyskać ich dwa (klaster główny oraz szum).

W tym celu analizowano dendrogram (wynik Single Linkage lub Complete Linkage), aby uzyskać najmniejsze  $d$  oraz największe  $d$  z dendrogramu. Te odległości były wykorzystywane do obliczenia wartości kroku dla pętli, co pozwalało na uzyskanie określonej liczby równomiernie rozłożonych pomiarów. W pracy badano 100 równo rozłożonych  $d_{max}$  jako parametru przerwania pętli. Parametr  $d_{max}$  określał moment przerwania algorytmu.

Jeśli po zakończeniu grupowania liczba klastrów wynosiła więcej niż dwa, wszystkie klastry poza największym były traktowane jako szum. To podejście umożliwiało efektywne rozróżnienie między rzeczywistymi grupami a szumem, co było kluczowe dla celów badawczych. Dane były zapisywane do pliku `*.csv`, co umożliwiało ich efektywne odczytywanie i analizę.

#### 4.2.1 Miary jakości grupowania

Do zmierzenia jakości grupowania użyto dokładności (ang. *accuracy*) zdefiniowanej jako:

$$A(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}(\hat{y}_i = y_i), \quad (4.1)$$

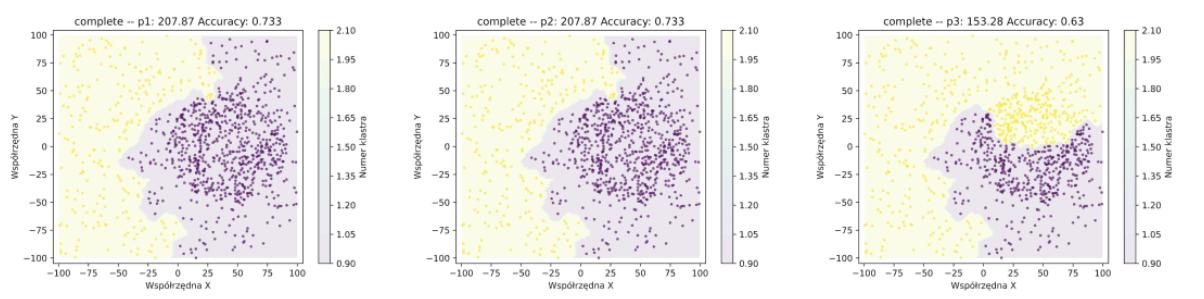
gdzie:

- $y$  to rzeczywiste etykiety,
- $\hat{y}$  to przewidywane etykiety,
- $n$  to liczba wszystkich próbek,
- $\mathbf{1}(\hat{y}_i = y_i)$  to funkcja wskaźnikowa, która jest równa 1, jeśli przewidywana etykieta  $\hat{y}_i$  jest równa rzeczywistej etykiecie  $y_i$ , i 0, jeśli nie są równe.

Już na etapie generowania punktów przypisywano im etykiety, określając, czy pochodzą one z szumu, czy z zagęszczenia. Po znalezieniu właściwych parametrów dla algorytmów wykonuje się go z tymi parametrami, a następnie analizowano, jaki procent punktów został poprawnie zaklasyfikowany zgodnie z ich pierwotnymi etykietami (szum lub grupa).

### 4.3 Wyniki badań

Poniżej przedstawiono wyniki grupowania w jedną grupę dla różnych zbiorów danych (kształt zagęszczenia) przy pomocy kolejno Complete Linkage, Single Linkage oraz DBSCAN



Rysunek 4.8: Algorytm Complete Linkage dla zbioru koła jednorodnego

wraz z porównaniem czasu obliczeń co jest widoczne w tabeli (tab. 4.1). Badano zarówno wpływ stworzonych kryteriów oceny, jak i też wpływ szumu na grupowanie danych. Wyniki prezentowane są na wykresach nazwą metody, wartością przerwania algorytmu  $d_{max}$  (odległość dendrogramu), a także dokładnością grupowania.

### 4.3.1 Wpływ kryterium

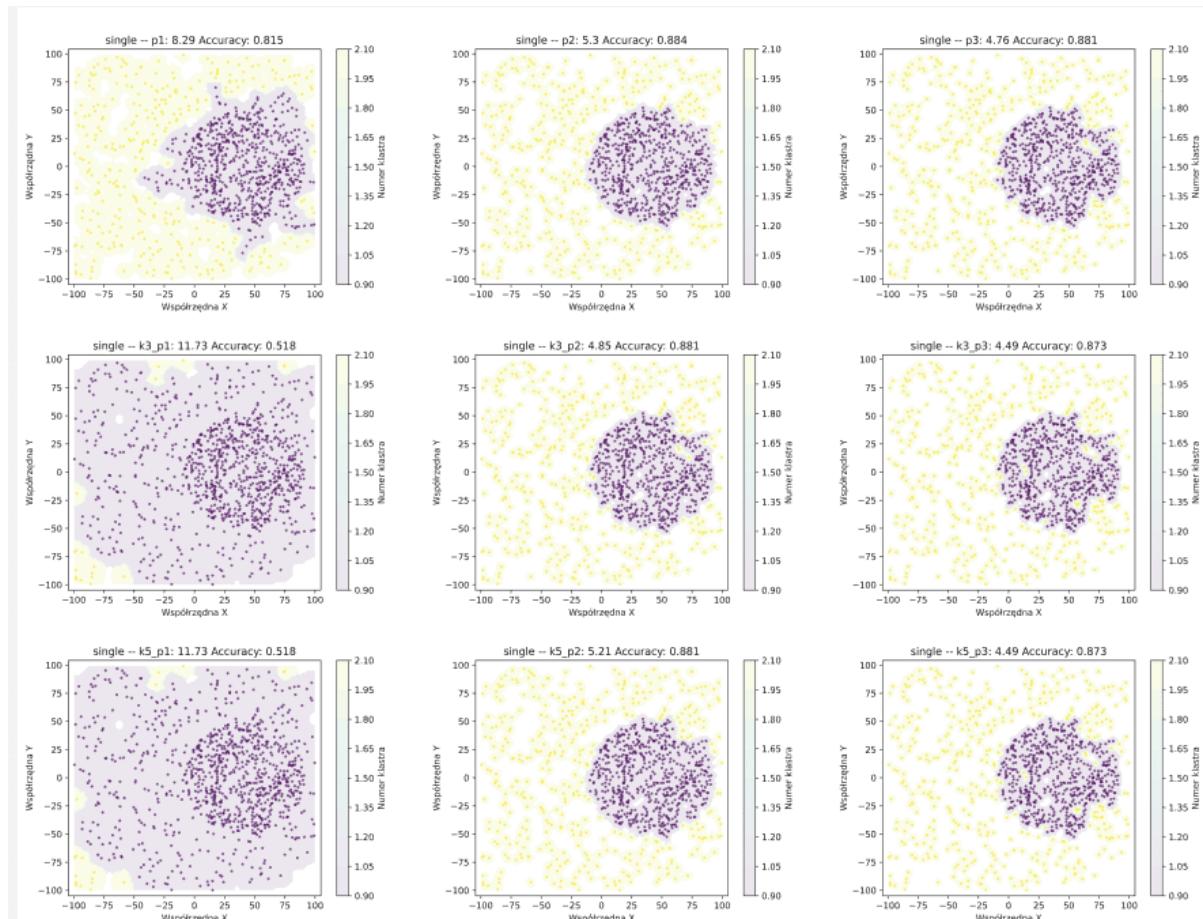
Prezentowane wyniki są dla wartości 50% szumu, co oznacza, że 500 punktów zostało wygenerowanych losowo, oraz że 500 punktów zostało wygenerowanych dodatkowo w obrębie zadanego kształtu. Dodatkowo za średnice koła przyjęto za 50, tak aby móc porównać wpływ parametrów (kryteriów) na grupowanie danych. Miarą jakości grupowania była dokładność 4.1, która została opisana w rozdziale 4.2.1.

#### Koło jednorodne

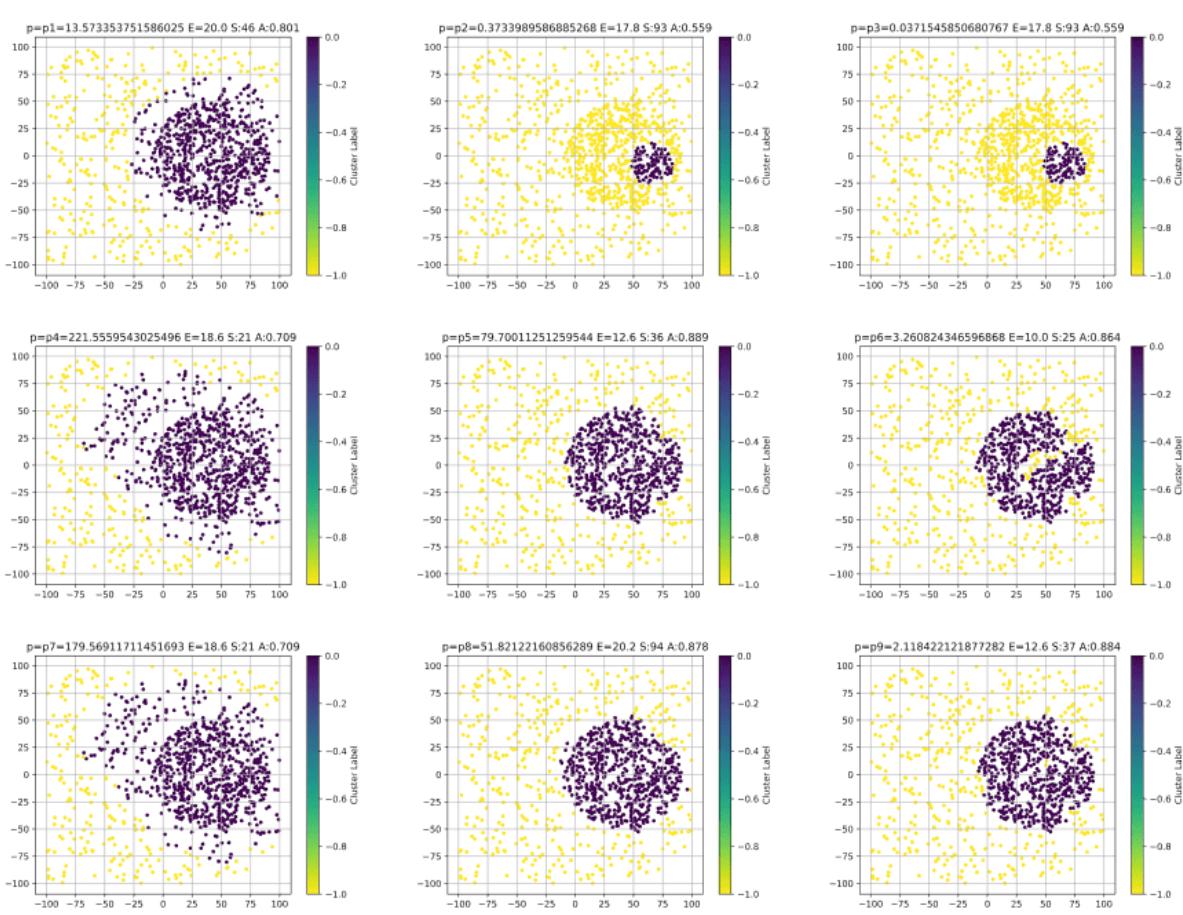
Na ilustracji 4.8 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważyć, że wykresy dla  $P_1$  oraz  $P_2$  przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 207.87. Dokładność została określona na 73,3%. Dla parametru  $P_3$  uzyskano  $d$  równe 153.28. Dokładność została określona na 63%. Dla pozostałych parametrów od  $P_{1,k3}$  (`k3_p1`) do  $P_{3,k9}$  (`k9_p3`), uzyskano ten sam wynik jak w przypadku  $P_1, P_2, P_3$ , więc pominięto ich prezentacje. Pełne wynik będą załączone w załączniku.

Na ilustracji 4.9 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważyć, że algorytm osiągnął lepszą dokładność niż Complete Linkage. Najwyższa dokładność została osiągnięta dla kryterium  $P_2$ ,  $p = 5.3$  osiągnęła 88,4% dokładności. Należy także zauważyć, że dla  $P_{1,k3}$  (`k3_p1`), oraz  $P_{1,k5}$  (`k5_p1`) uzyskano wynik bliski 51%, co było kiepskim wynikiem, porównywalnym z Complete Linkage. W pozostałych przypadkach dokładność była powyżej 80%.

Na ilustracji 4.10 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$  (`k5_p3`). Najwyższa dokładność została osiągnięta dla  $P_{2,k3}$  (`k3_p2`) i osiągnęła



Rysunek 4.9: Algorytm Single Linkage dla zbioru koła jednorodnego



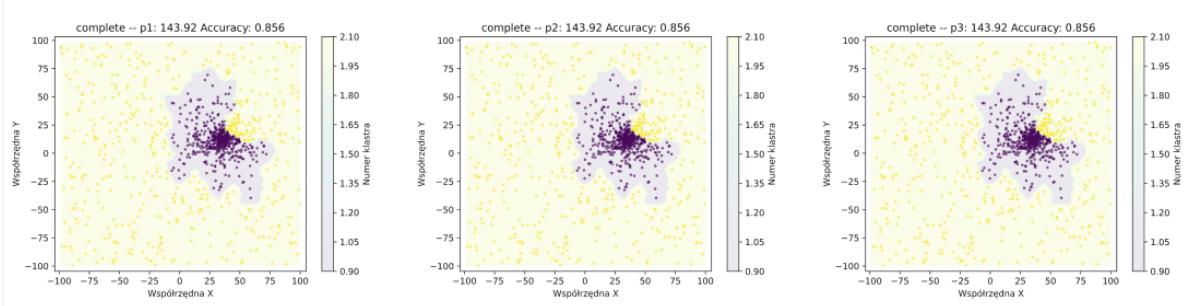
Rysunek 4.10: Algorytm DBSCAN dla zbioru koła jednorodnego

88,9%. Co ciekawe, dla  $P_{3,k5}$  (k5\_p3) oraz  $P_{3,k3}$  (k3\_p3) algorytm wyciął mały kawałek ze środka koła, co zmniejszyło dokładność. Algorytm uzyskał lepsze wyniki niż Single Linkage i Complete Linkage.

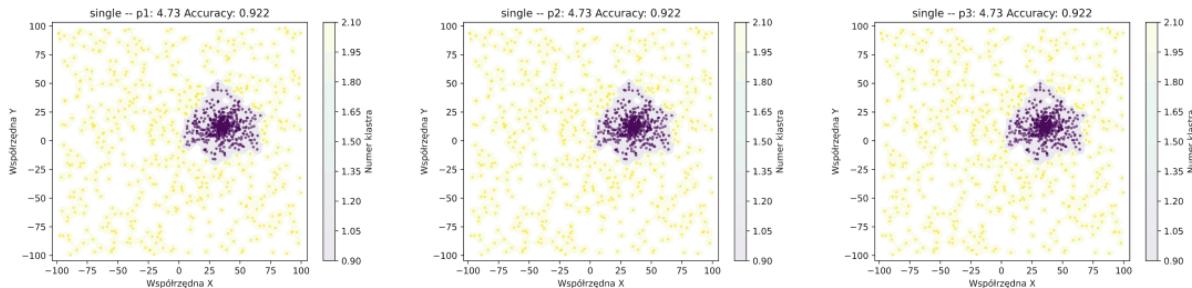
### Koło w rozkładzie von Misesa

Na ilustracji 4.11 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów od  $P_1$  do  $P_{3,k5}$  (k5\_p3). Można zauważyc, że wszystkie wykresy przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 143.92. Dokładność została określona na 85,6%. Dla pozostałych parametrów od  $P_{1,k3}$  (k3\_p1) do  $P_{3,k9}$  (k9\_p3), uzyskano ten sam wynik, więc pominięto ich prezentacje. Pełne wyniki będą załącznikiem.

Na ilustracji 4.12 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważyc, że algorytm osiągnął lepszą dokładność niż Complete Linkage. Wykresy przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 4.73. Dokładność została określona na 92,2%. Dla po-



Rysunek 4.11: Algorytm Complete Linkage dla zbioru rozkładu von Misesa



Rysunek 4.12: Algorytm Single Linkage dla zbioru rozkładu von Misesa

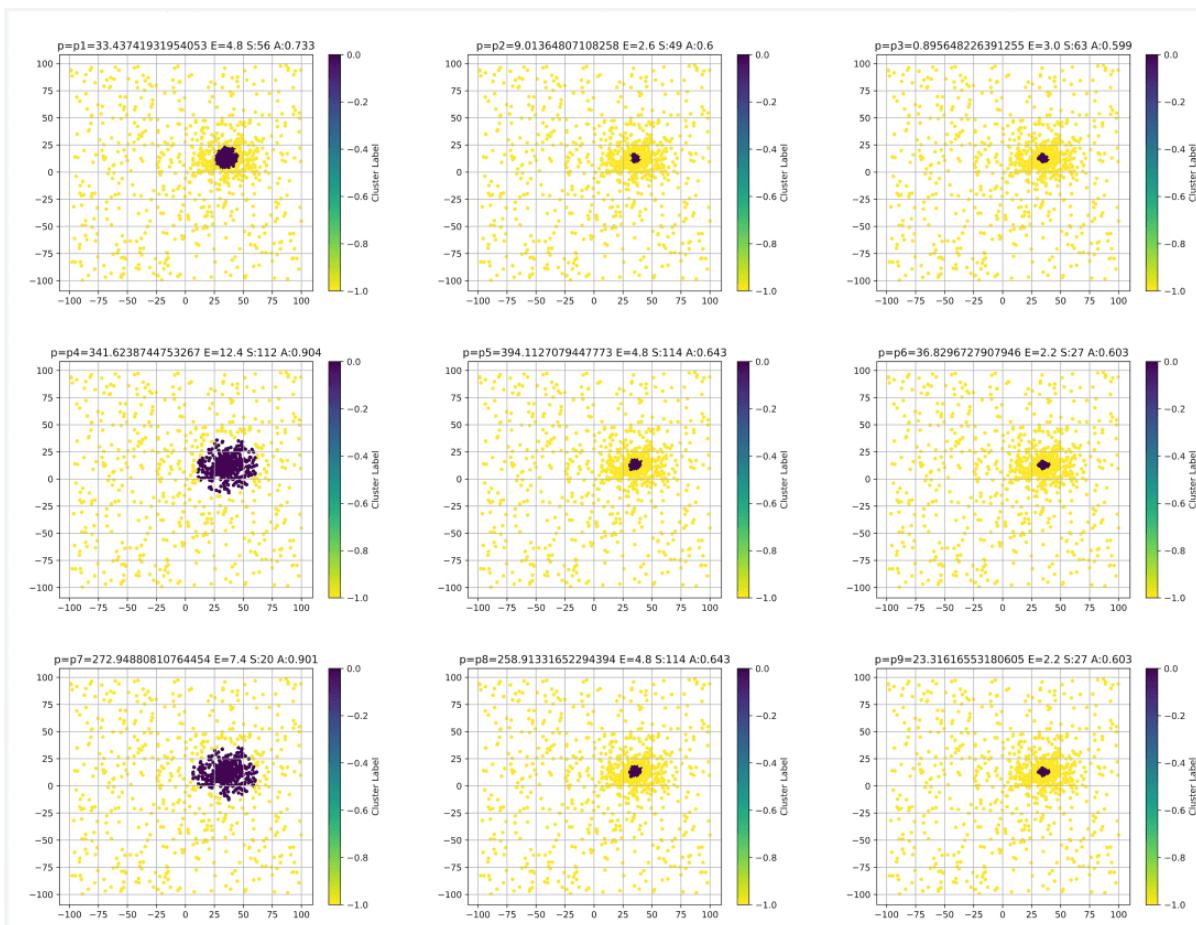
zostałych parametrów od  $P_{1,k3}$  (**k3\_p1**) do  $P_{3,k9}$  (**k9\_p3**), uzyskano ten sam wynik, więc pominięto ich prezentacje. Pełne wyniki będą załączone w załączniku.

Na ilustracji 4.13 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$  (**k5\_p3**). Najwyższa dokładność została osiągnięta dla  $P_{1,k3}$  (**k3\_p1**) i osiągnęła 90,4%. Co ciekawe, dla  $P_{1,k5}$  (**k5\_p1**) algorytm uzyskał podobny wynik, jednak dla każdego innego kryterium  $p$  uzyskał on słabą dokładność w granicach od 50 do 70 procent. Wynika z tego, że uzyskał gorszy najlepszy wynik niż Single Linkage, oraz Complete Linkage ze swoją dokładnością 85,6% okazał się lepszy od większości kryteriów dla DBSCAN.

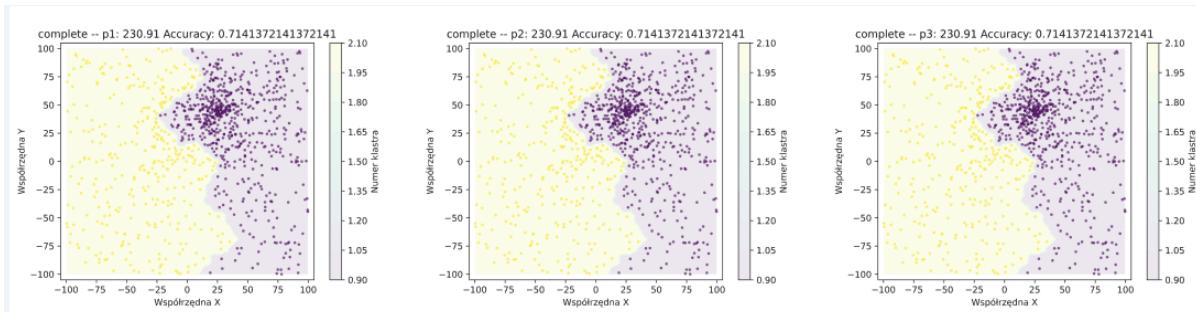
### Koło w rozkładzie normalnym

Na ilustracji 4.14 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważyć, że wszystkie wykresy przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 230.91. Dokładność została określona na 71,4%. Dla pozostałych parametrów od  $P_{1,k3}$  (**k3\_p1**) do  $P_{3,k9}$  (**k9\_p3**), uzyskano ten sam wynik, więc pominięto ich prezentacje. Pełne wyniki będą załączone w załączniku.

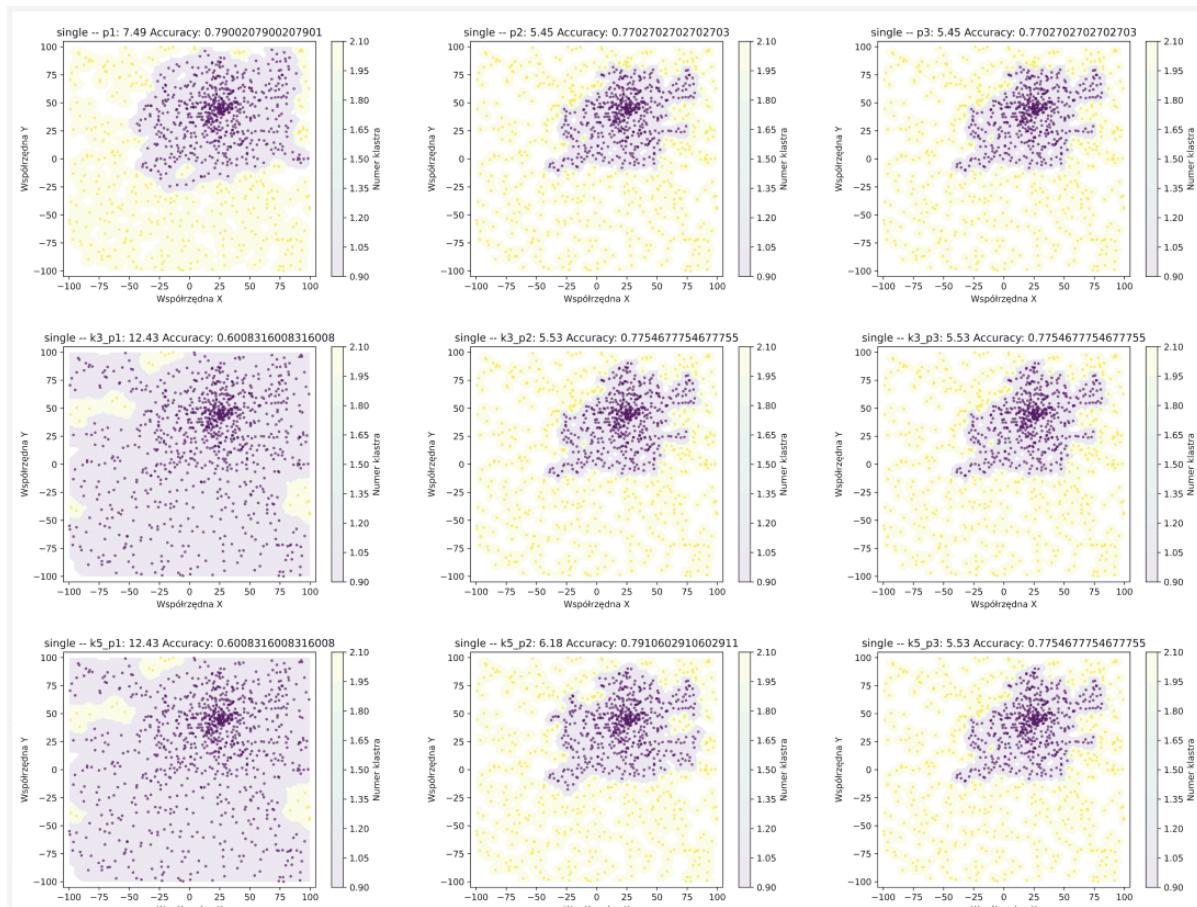
Na ilustracji 4.15 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $P_1$  do  $P_{3,k5}$  (**k5\_p3**). Należy zauważać większą różnorodność wyników, od dokładności 60% do 79,1% w przypadku parametru  $d$  równemu 6.18. Większość wyników uzyskała dokładność powyżej 70%. W



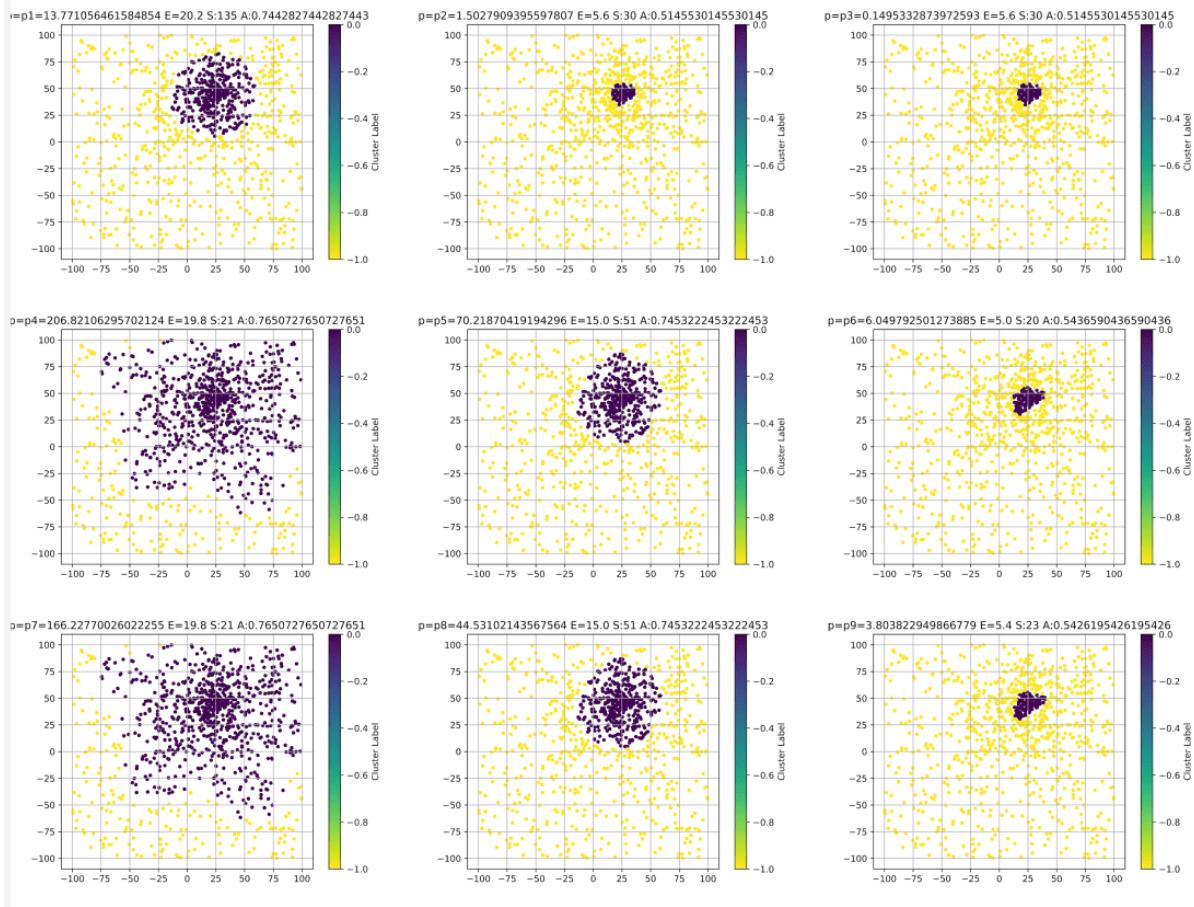
Rysunek 4.13: Algorytm DBSCAN dla zbioru rozkładu von Misesa



Rysunek 4.14: Algorytm Complete Linkage dla zbioru rozkładu normalnego



Rysunek 4.15: Algorytm Single Linkage dla zbioru rozkładu normalnego

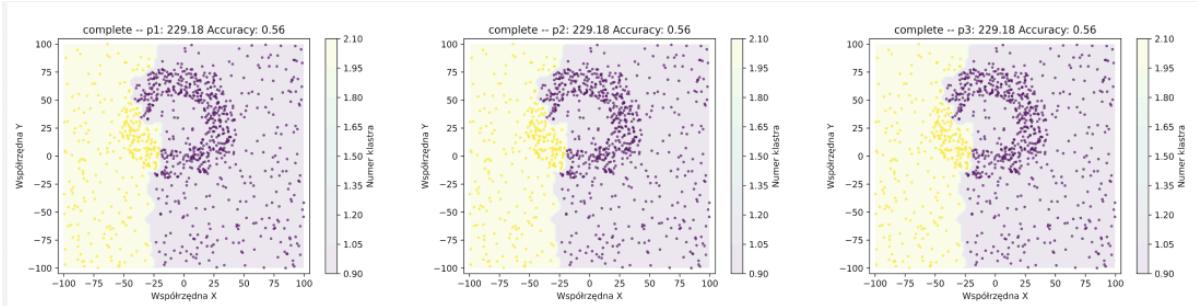


Rysunek 4.16: Algorytm DBSCAN dla zbioru rozkładu normalnego

Na ilustracji 4.16 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$  (**k5\_p3**). Należy zauważać, że najlepszy wyniki zostały osiągnięte dla  $P_1$  oraz  $P_{1,k3}$  z tym że dla tego pierwszego klaster jest bardziej zwarty, w przypadku  $P_{1,k3}$  natomiast ma znacznie większą powierzchnię. Dla obu kryteriów algorytm osiągnął wyniki w granicach 75% dokładności, co jest gorszym wynikiem od najlepszego wyniku Single Linkage, a także jest porównywalne z Complete Linkage który osiągnął 71%. Słabe wyniki w granicach 50 procent dokładności osiągnięto dla kryteriów z kategorii  $P_3$  czyli  $P_3$ ,  $P_{3,k3}$  (**k3\_p3**) oraz  $P_{3,k5}$  (**k5\_p3**).

## Pierścień

Na ilustracji 4.17 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważać, że wszystkie wykresy przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 229.91. Dokładność została określona na 56%. Dla pozostałych parametrów od  $P_{1,k3}$  (**k3\_p1**) do  $P_{3,k9}$  (**k9\_p3**), uzyskano ten sam wynik, więc pominięto ich prezentacje. Pełne wyniki będą załącznikiu w załączniku.



Rysunek 4.17: Algorytm Complete Linkage dla zbioru z pierścieniem

Na ilustracji 4.18 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $P_1$  do  $P_{3,k5}$  ( $k5\_p3$ ). Należy zauważać mniejszą różnorodność wyników, od dokładności 81% do 89,2% w przypadku parametru  $d$  równemu 5.93. Większość wyników uzyskała dokładność powyżej 85%. W

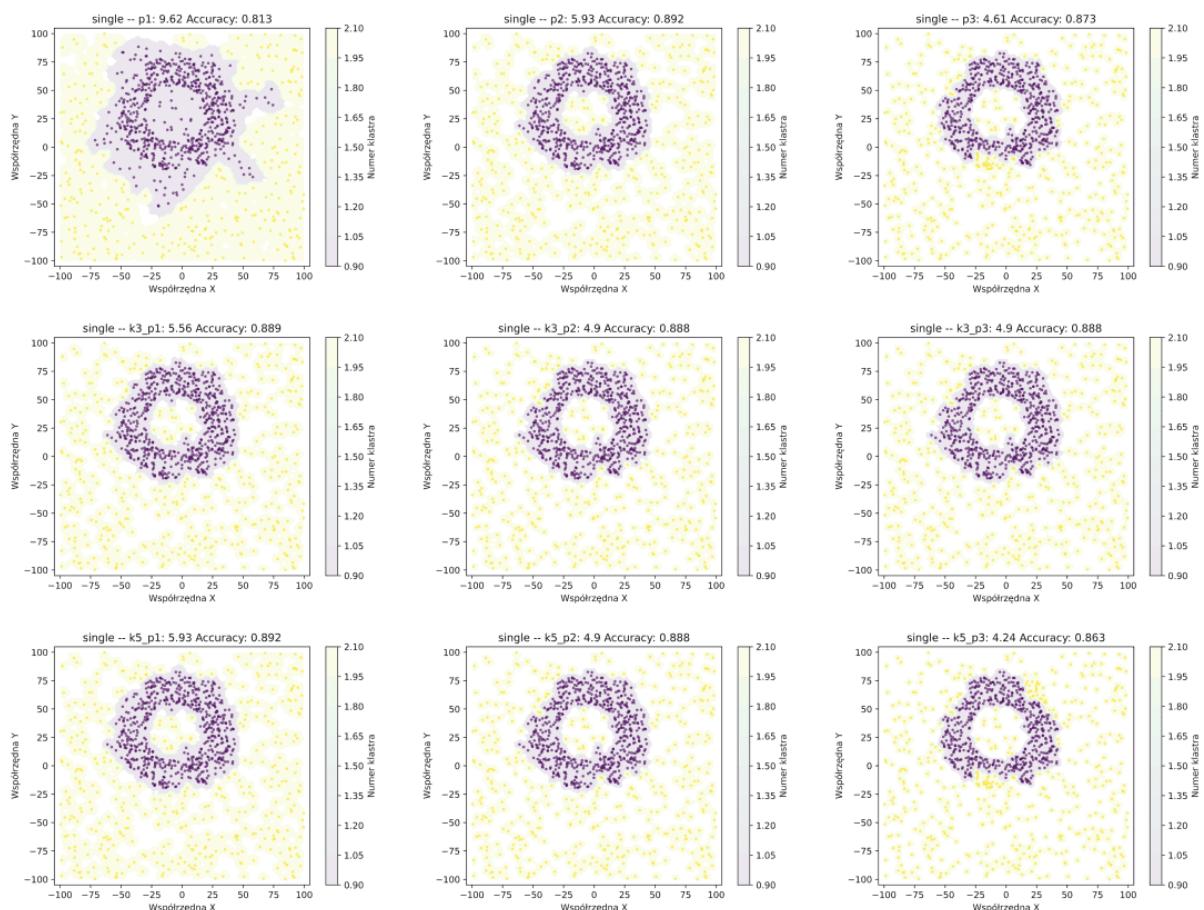
Na ilustracji 4.19 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$  ( $k5\_p3$ ). Należy zauważać, że algorytm poradził sobie z tym kształtem (z wyjątkiem miar  $P_2$  oraz  $P_3$ ) i osiągnął dokładność powyżej 88% w większości przypadków, co jest wynikiem zbliżonym do Single Linkage. Najgorzej poradził sobie Complete Linkage ze swoją dokładnością na poziomie 56%.

## Kształt A

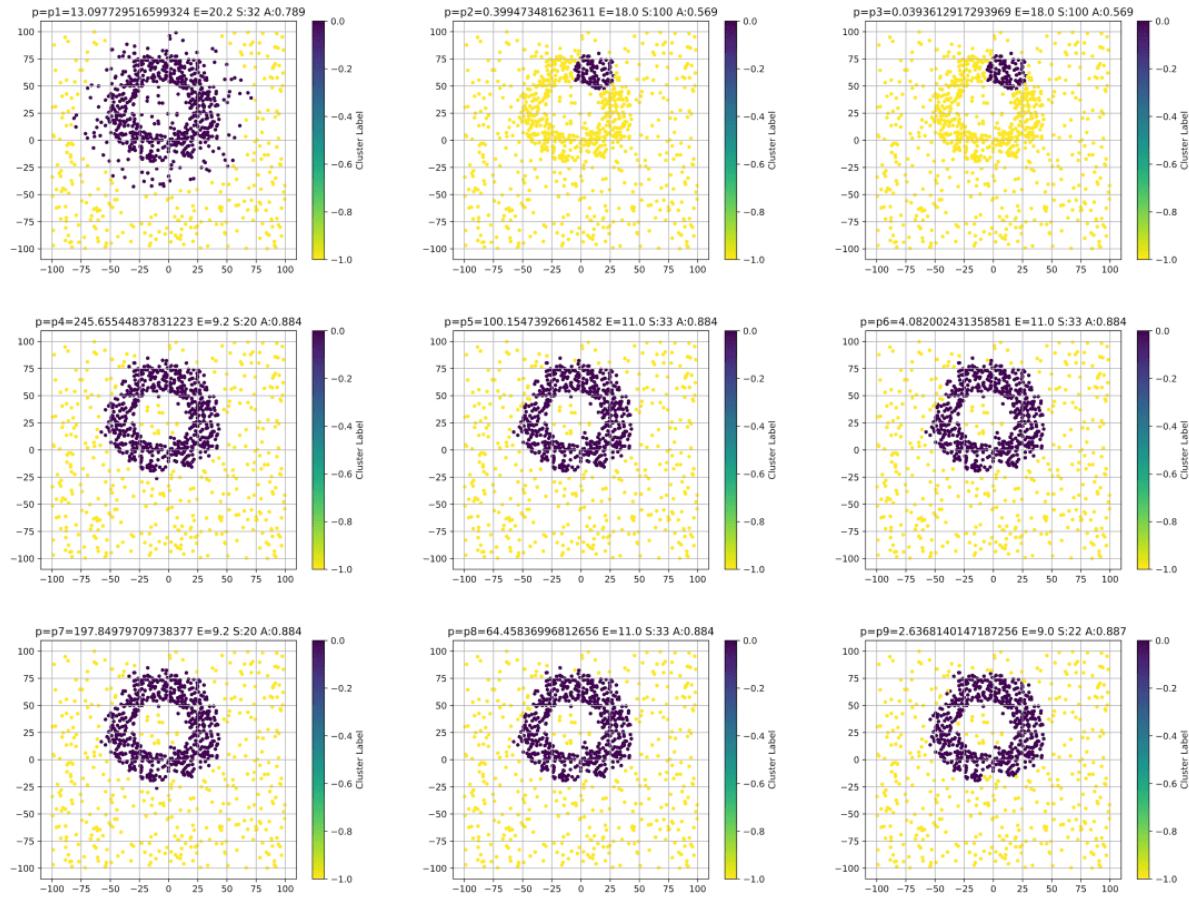
Na ilustracji 4.20 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważać, że wszystkie wykresy przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 224.67. Dokładność została określona na 48,3%. Dla pozostałych parametrów od  $P_{1,k3}$  ( $k3\_p1$ ) do  $P_{3,k9}$  ( $k9\_p3$ ), uzyskano ten sam wynik, więc pominięto ich prezentacje. Pełne wyniki będą załączone w załączniku.

Na ilustracji 4.21 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $P_1$  do  $P_{3,k5}$ . Należy zauważać mniejszą dokładność 50% do 60,8% w przypadku parametru  $d$  równemu 9.77. Cztery z dziewięciu prezentowanych wykresów posiadają tę samą dokładność 60,8%, która i tak jest dość niska i porównywalna dla algorytmu Complete Linkage. Co ciekawe on także słabo poradził sobie z tym zbiorem, co sugeruje, że był on trudny dla obu algorytmów.

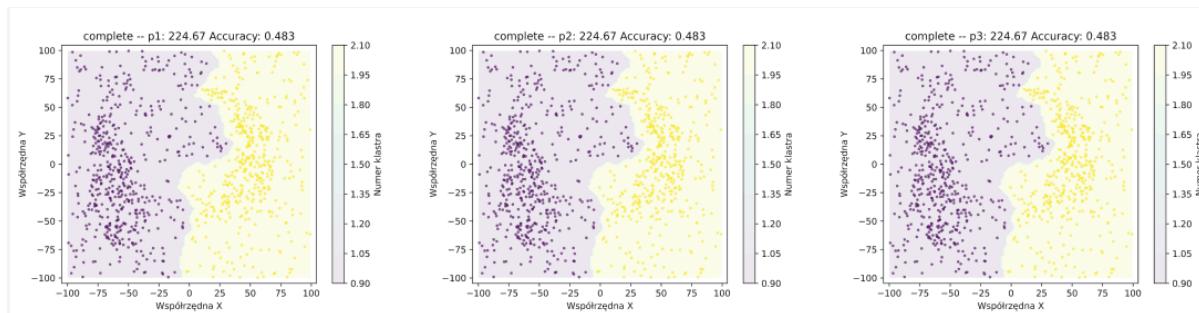
Na ilustracji 4.22 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$ . Należy zauważać, iż jest to kształt półksiężyca z dwoma zageszczeniami, co czyni ten zbiór trudnym przypadkiem. Algorytm osiągnął najlepszą dokładność dla miary  $P_{2,k5}$  z dokładnością wynoszącą 65,6%, co lepszym wynikiem od najlepszego rezultatu Single Linkage oraz znacznie lepszym od Complete Linkage. Warto zauważać, że dla niektórych  $p$  algorytm osiągnął słabą dokładność, i grupował małe (pod względem ilości punktów)



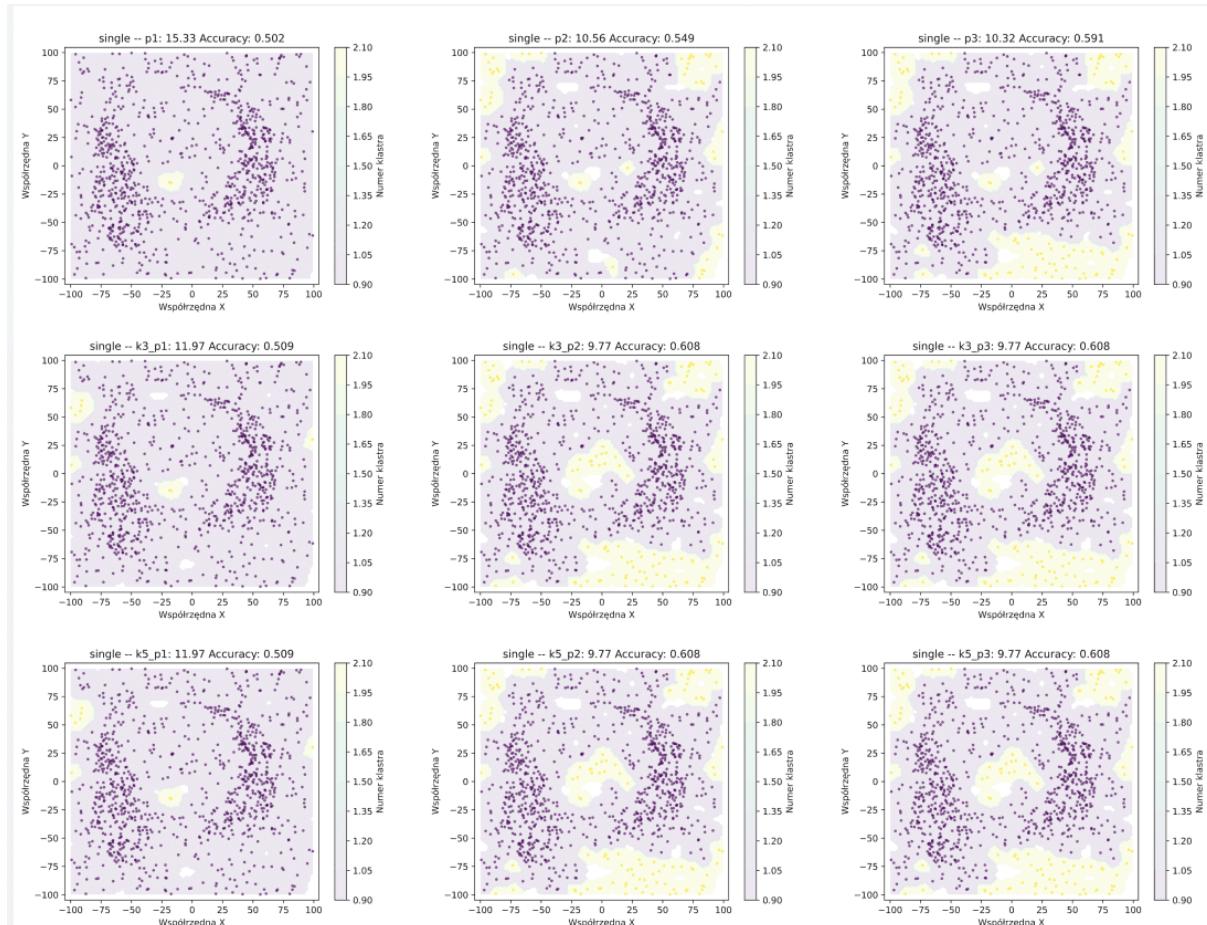
Rysunek 4.18: Algorytm Single Linkage dla zbioru z pierścieniem



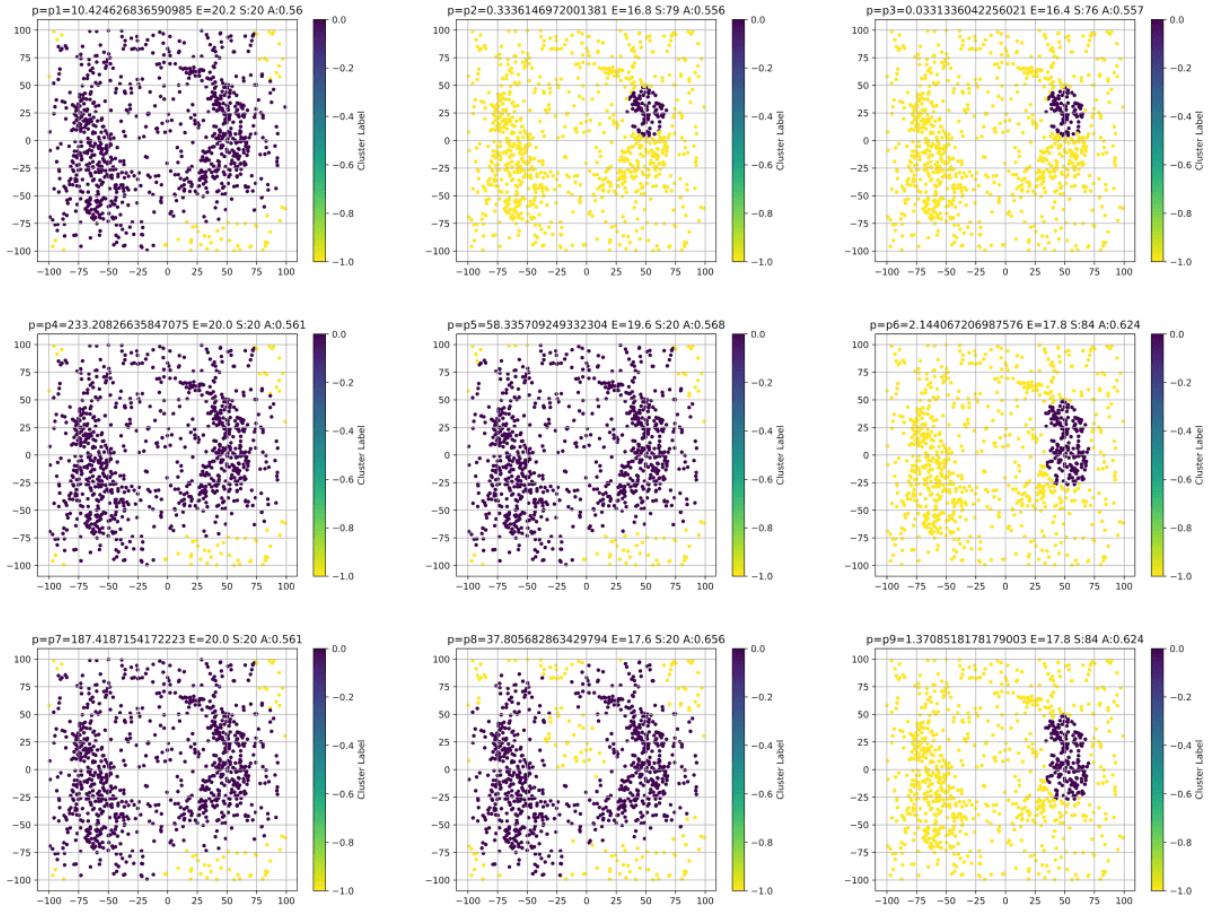
Rysunek 4.19: Algorytm DBSCAN dla zbioru z pierścieniem



Rysunek 4.20: Algorytm Complete Linkage dla zbioru z kształtem półksiężyca



Rysunek 4.21: Algorytm Single Linkage dla zbioru z kształtem półksiężyca



Rysunek 4.22: Algorytm DBSCAN dla zbioru z kształtem półksiężyca

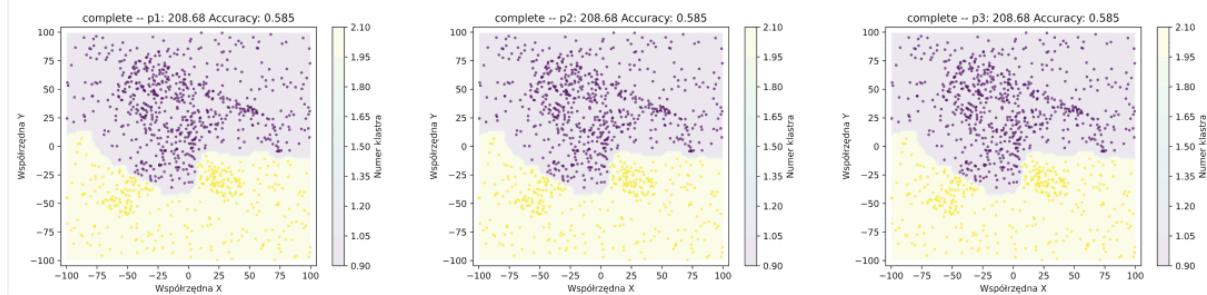
i zwarte klastry co jest widoczne dla  $P_3$ ,  $P_{3,k3}$  (k3\_p3) oraz  $P_{3,k5}$  (k5\_p3).

## Kształt B

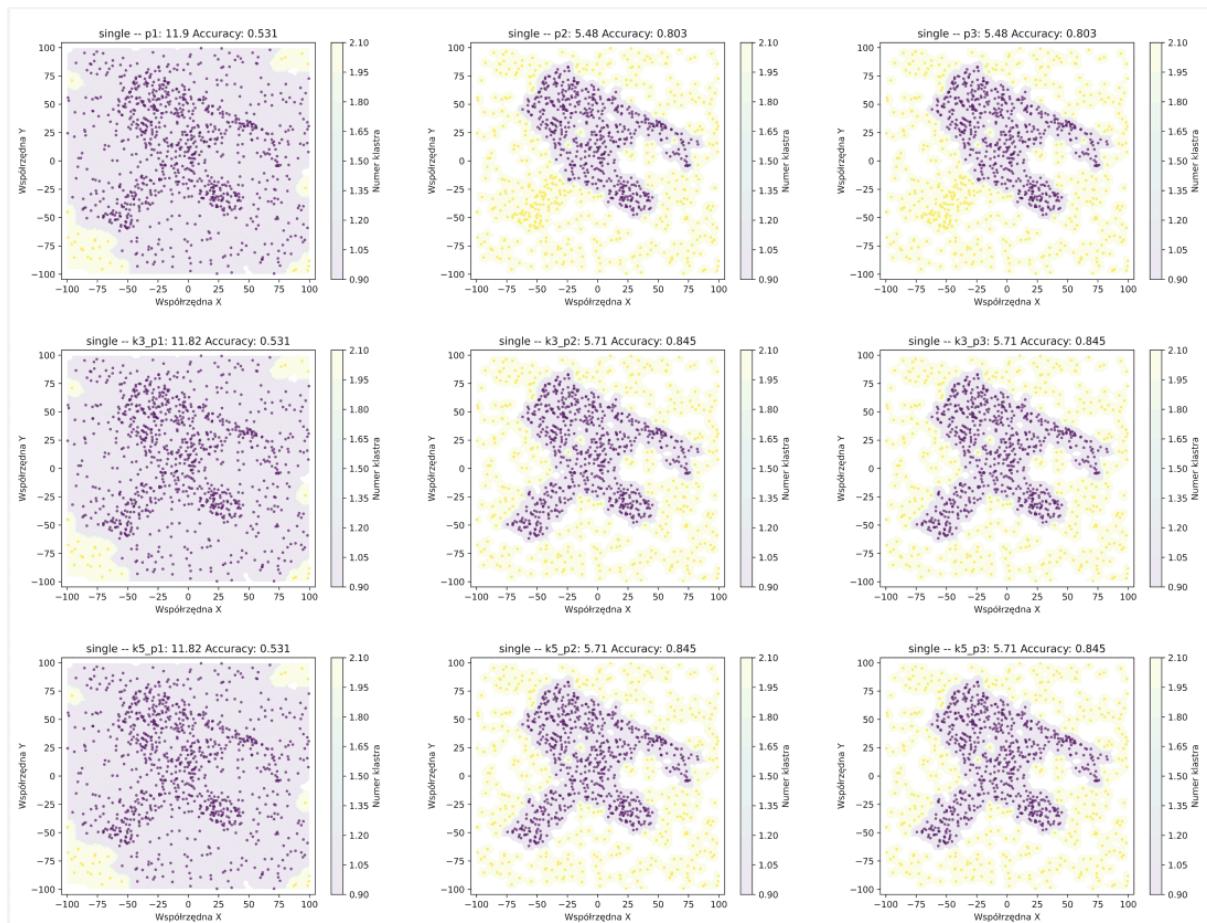
Na ilustracji 4.23 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ . Można zauważać, że wszystkie wykresy przedstawiają ten sam obraz, co wynika z faktu, że mimo zastosowania różnych kryteriów, wybrano tę samą konfigurację parametrów przerwania algorytmu. W tym wypadku było to  $d$  równe 208.68. Dokładność została określona na 58.5%. Dla pozostałych parametrów od  $P_{1,k3}$  (k3\_p1) do  $P_{3,k9}$  (k9\_p3), uzyskano ten sam wynik, więc pominięto ich prezentacje. Pełne wyniki będą załączone w załączniku.

Na ilustracji 4.24 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $P_1$  do  $P_{3,k5}$ . Należy zauważać mniejszą dokładność 53% do 84,5% w przypadku parametru  $d$  równemu 5.71. Sześć z dziewięciu prezentowanych wykresów posiadają dokładność powyżej 80%. Co ciekawe uzyskano widocznie lepsze wyniki niż dla Complete Linkage, ponadto dla parametrów z kategorii  $P_2, P_3$  uzyskano widocznie lepsze wyniki aniżeli  $P_1$ .

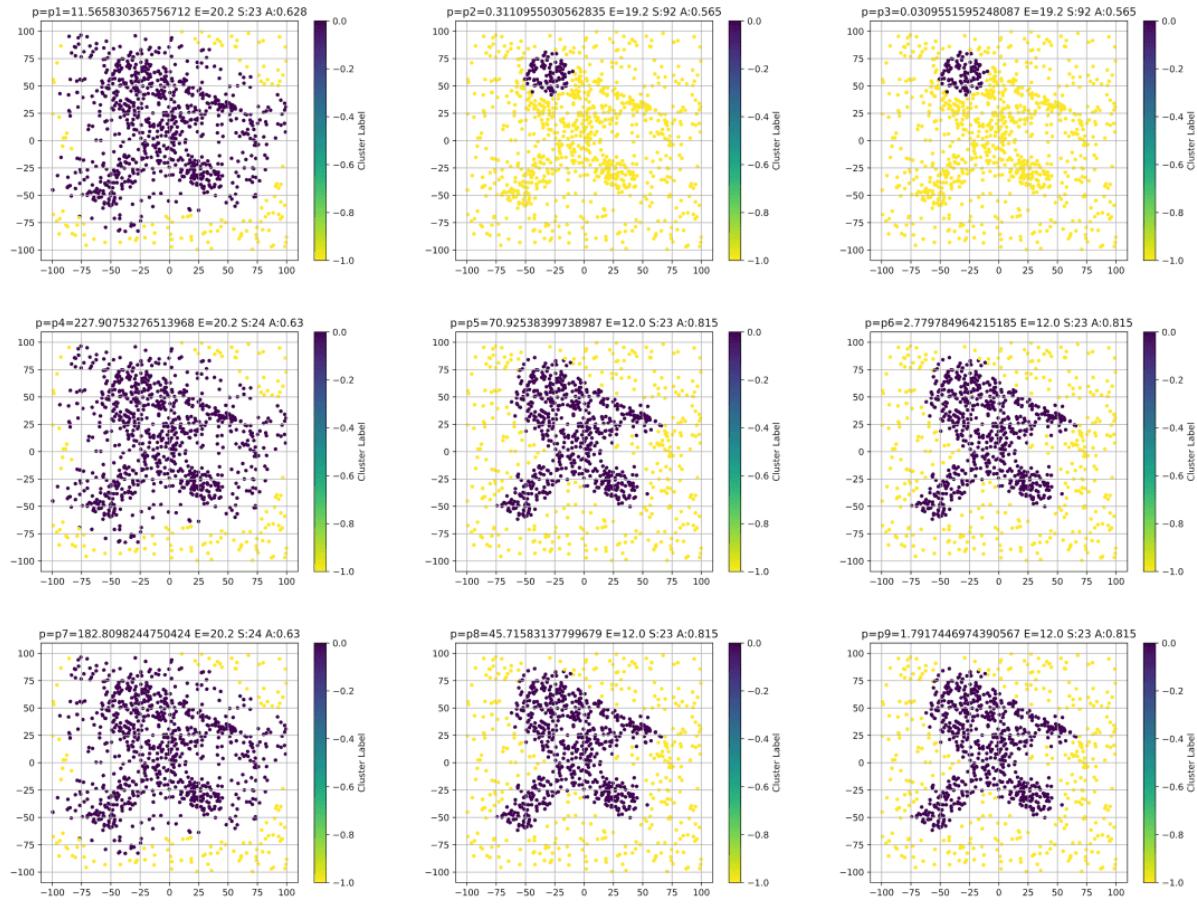
Na ilustracji 4.25 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$ . Algorytm osiągnął tę samą dokładność dla  $P_5, P_6, P_8, P_9$ , równą 81,5% co jest



Rysunek 4.23: Algorytm Complete Linkage dla zbioru z literą „X”



Rysunek 4.24: Algorytm Single Linkage dla zbioru z literą „X”

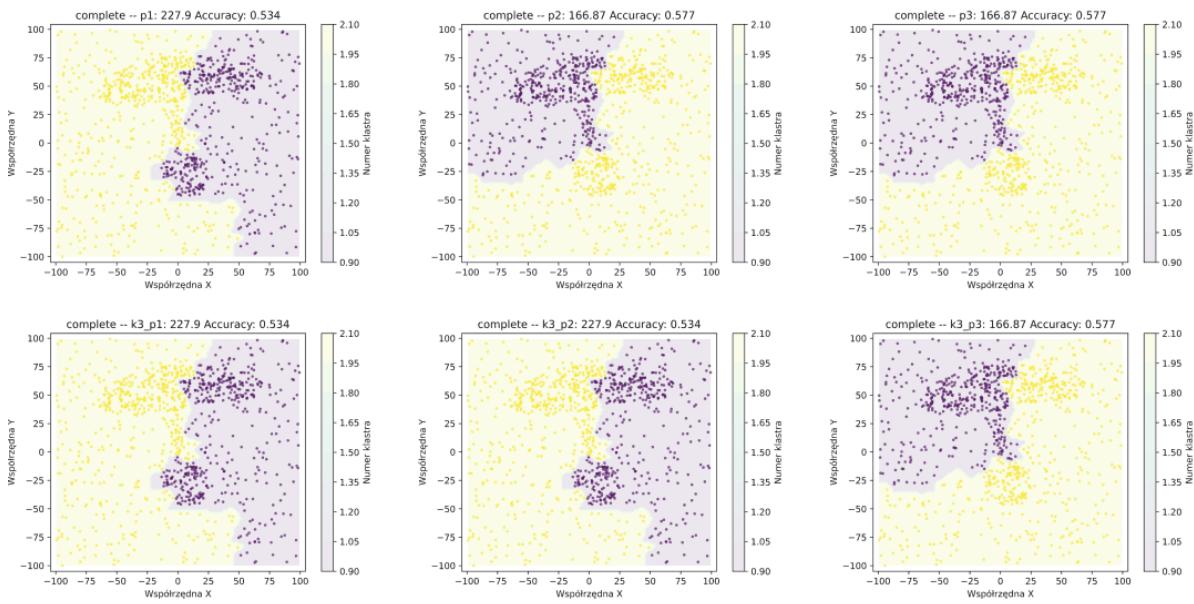


Rysunek 4.25: Algorytm DBSCAN dla zbioru z literą „X”

gorszym wynikiem od najlepszego wyniku Single Linkage. Dla kryteriów  $P_2, P_3$  widoczne jest tworzenie małych (liczbowo), i zwartych klastrów. Algorytm poradził sobie lepiej od Complete Linkage porównywalnie do Single Linkage. Warto zaznaczyć, że w obu algorytmach w kategorii  $P_1$  czyli  $P_1, P_4, P_5$  algorytmy osiągnęły wyraźnie gorsze rezultaty.

## Kształt C

Na ilustracji 4.26 widoczne są wyniki dla algorytmu Complete Linkage, dla kryteriów  $p \in P_1, P_2, P_3$ , a także  $p \in P_4, P_5, P_6$ . Można zauważyc, że istnieją dwa utworzone zgrupowania. Dla  $d$  równego 166.87, dokładność została określona na 57,7% oraz dla  $d$  równego 227.9, dokładność została określona na 53,4%. Dla pozostałych parametrów od  $P_{1,k3}$  (`k3_p1`) do  $P_{3,k9}$  (`k9_p3`), uzyskano te same wyniki (jeden z tych dwóch zgrupowań zbiorów), więc pominięto ich prezentacje. Pełne wyniki będą załączone w załączniku. Warto zaznaczyć, że to jedyny przypadek gdzie kryteria, a następnie parametry dały inne wyniki dla Complete Linkage, w pozostałych przypadkach wszystkie parametry dawały ten sama wartość  $d$  będącą warunkiem przerwania algorytmu wpływającą na sposób grupowania.



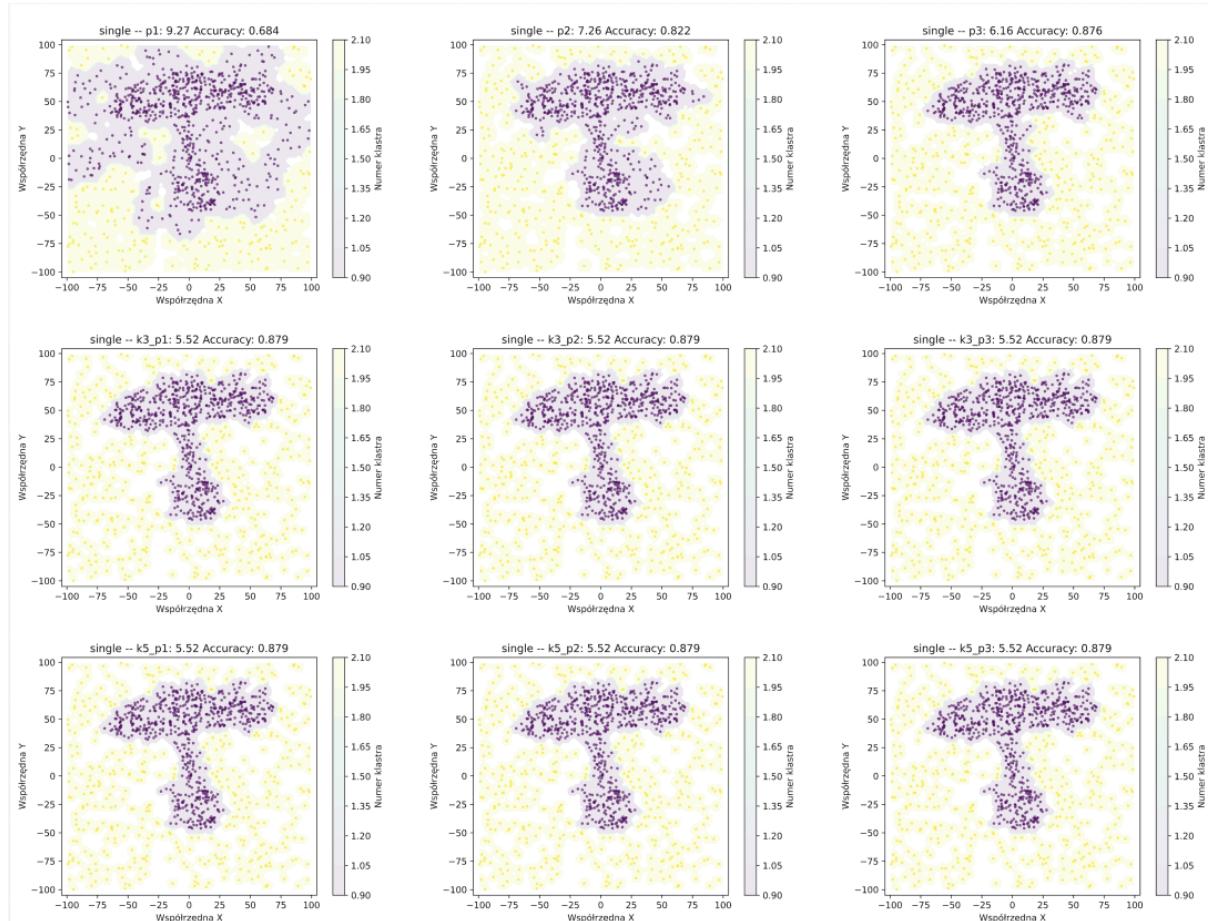
Rysunek 4.26: Algorytm Complete Linkage dla zbioru z kształtem grzyba

Na ilustracji 4.27 widoczne są wyniki dla algorytmu Single Linkage, dla kryteriów  $P_1$  do  $P_{3,k5}$  (`k5_p3`). Należy zauważać mniejszą dokładność 68% do 87,9% w przypadku parametru  $d$  równemu 5.52. Jedyne parametr  $P_1$  znaczaco odstawał od reszty wyników, ze słabą dokładnością wynoszącą 68%. Co ciekawe uzyskano widocznie lepsze wyniki niż dla Complete Linkage, ponadto dla parametrów z kategorii  $P_2, P_3$  uzyskano widocznie lepsze wyniki aniżeli  $P_1$ .

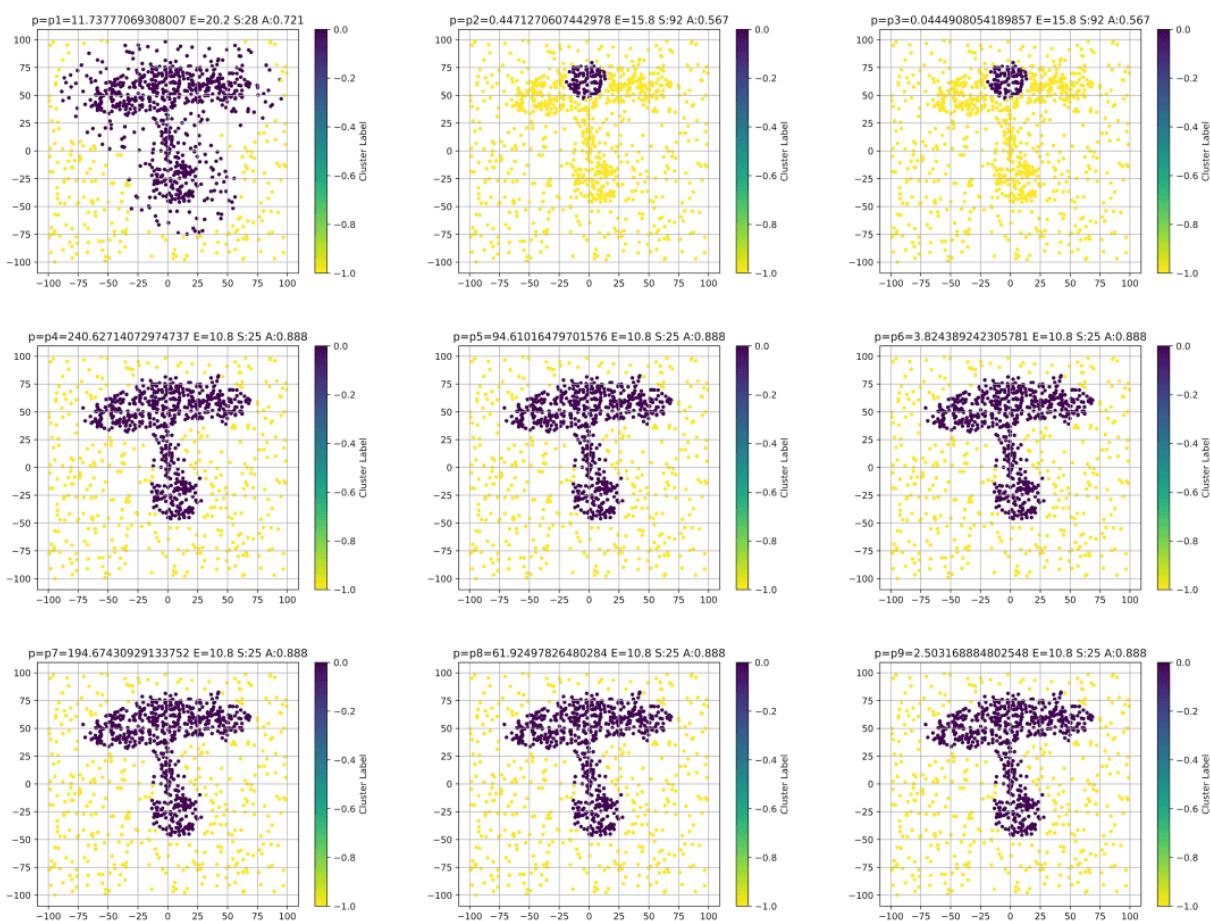
Na ilustracji 4.28 widoczne są wyniki dla algorytmu DBSCAN, dla kryteriów od  $P_1$  do  $P_{3,k5}$ . Algorytm osiągnął tę samą dokładność dla  $P_4, P_5, P_6, P_7, P_8, P_9$ , równą 88,8% co jest minimalnie lepszym wynikiem od najlepszego wyniku Single Linkage. Dla kryteriów  $P_2, P_3$  widoczne jest tworzenie małych (liczbowo), i zwartych klastrów co było już zauważone w przypadku poprzednich kształtów 4.25 (ksztalt „X”) oraz 4.22 (ksztalt półksiężyca). Algorytm poradził sobie lepiej od Complete Linkage porównywalnie do Single Linkage. Warto zaznaczyć, że w obu algorytmach w miarę  $P_1$  algorytmy osiągnęły wyraźnie gorsze rezultaty.

### 4.3.2 Wrażliwość na szum

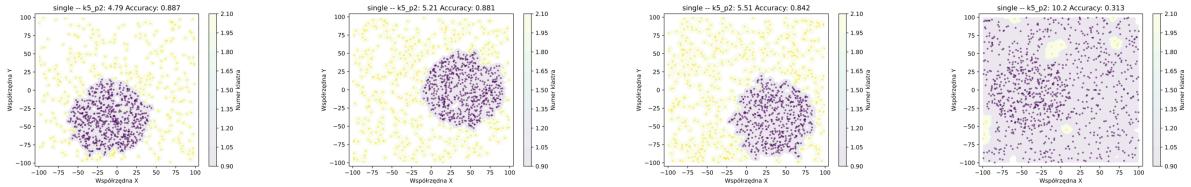
W tym podrozdziale prezentowane będą badania ma różnych zbiorach dla różnego poziomu szumu (od 40% do 70%) dla tego samego parametru. Wybrano kryterium oznaczone jako `k5_p2`, co oznacza, że uzyto kryterium z drugiej kategorii  $P_2$  gdzie zastosowano algorytm KNN roz. 2.2 do obliczenia średniej odległości między pięcioma ( $k = 5$ ) najbliższymi punktami dla każdego punktu w zbiorze (należącym do klastra). Zdecydowano się na to kryterium, ponieważ osiągnęło ono zadowalające efekty grupowania dla stałego szumu (50%), dla różnych kształtów. Celem było zbadanie wrażliwości algorytmów na szum



Rysunek 4.27: Algorytm Single Linkage dla zbioru z kształtem grzyba



Rysunek 4.28: Algorytm DBSCAN dla zbioru z kształtem grzyba



Rysunek 4.29: Single Linkage, dla koła jednorodnego. Szum od 40% do 70%

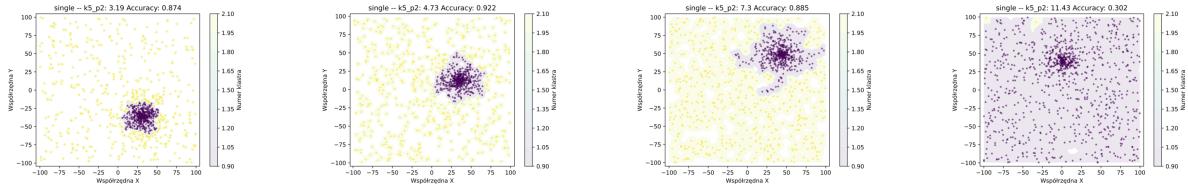
i wyciągnięcie na tej podstawie wniosków. Poniższe wykresy przedstawiają wyniki dla algorytmu Single Linkage, oraz dla algorytmu DBSCAN. Dla algorytmu Complete Linkage szum nie miał większego wpływu, przez co prezentacja danych została pominuta. Pełne dane, wykresy dla wszystkich iteracji głównej pętli programu znajdują się w dołączonym załączniku.

### Single Linkage

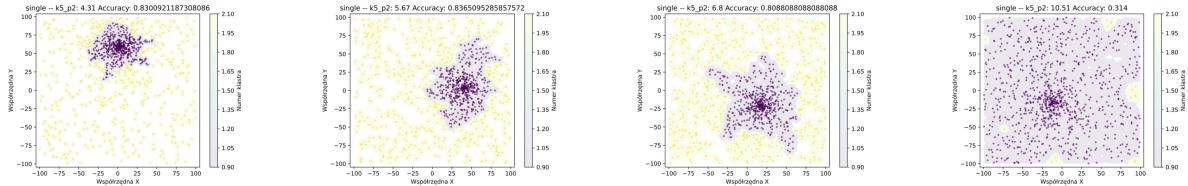
Rys. 4.29 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie k5\_p2, a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. W przypadku pierwszych trzech wykresów algorytmowi udało się wykryć skupienie grupy (koło), z dokładnością ponad 88%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność poniżej 32%. Można zauważyć, że wraz ze zwiększeniem się zawartości szumu dokładność minimalnie malała od 88,8% do 84,2%. Istotny spadek dokładności należy zauważać między wartością 60% a 70% szumu.

Rys. 4.30 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie k5\_p2, a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. W przypadku pierwszych trzech wykresów algorytmowi udało się wykryć skupienie grupy (koło w rozkładzie von Misesa), z dokładnością ponad 87%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność poniżej 30%, co czyni go najgorszym prezentowanym dotychczas wynikiem. Można zauważyć, że największą dokładność osiągnięto nie dla szumu 40%, a dla 50% i wyniosła aż 92,2%, co jest bardzo dobrym wynikiem. Wykres dla szumu 60%, także uzyskał lepszą dokładność niż wykres dla 40%. Warto zauważać, że pierwszy wykres dla 40% wykrył grupę zawierającą mniej punktów niż pozostałe wykresy, tworzył on mały (punktowo) i zwięzły klaster.

Rys. 4.31 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie k5\_p2, a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. Wykresy



Rysunek 4.30: Single Linkage, dla von Misesa. Szum od 40% do 70%

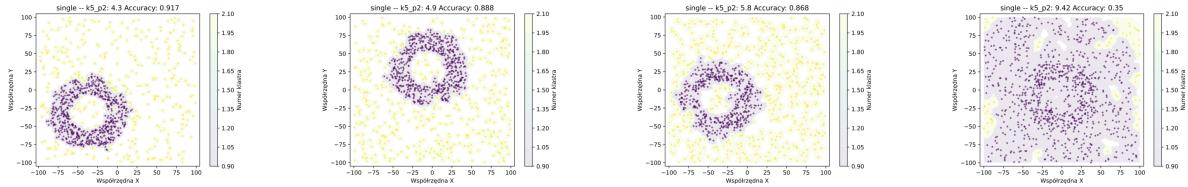


Rysunek 4.31: Single Linkage, dla rozkładu normalnego. Szum od 40% do 70%

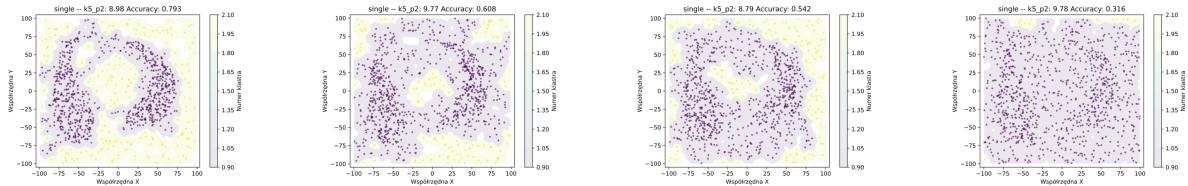
przedstawiają wyniki grupowania dla zbioru z kołem z rozkładem normalnym, co czyni go trudniejszego do wykrycia skupisk, gdyż istnieje większy rozrzut punktów od środka koła. W przypadku pierwszych trzech wykresów algorytmowi udało się wykryć skupienie grupy (koło w rozkładzie normalnym), z dokładnością ponad 80%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność około 31%. Można zauważać, że największą dokładność wynoszącą 83% osiągnięto na wykresach z szumami 40%, oraz 50%. Należy zauważać minimalny spadek dla szumu 60%, oraz widoczny jest spadek dla 70%.

Rys. 4.32 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie  $k5\_p2$ , a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. Wykresy przedstawiają wyniki grupowania dla zbioru z pierścieniem, co czyni go ciekawszym przypadkiem. Celem takiego zbioru było zbadanie czy algorytm poradzi sobie z wykryciem szumu, wewnątrz skupienia. W przypadku pierwszych trzech wykresów algorytmowi udało się wykryć skupienie grupy (pierścień), z dokładnością ponad 85%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność około 35%. Można zauważać, że największą dokładność wynoszącą 91% osiągnięto na wykresie z najmniejszym szumem wynoszącym 40%. Należy zauważać spadek dokładności dla szumu 50%, oraz 60%. Na podstawie tego i powyższych wykresów można wstępnie powiedzieć, że algorytm dobrze sobie radzi do wartości 60% szumu.

Rys. 4.33 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie  $k5\_p2$ , a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. Wykresy przedstawiają wyniki grupowania dla zbioru z kształtem własnym, a dokładniej dwoma



Rysunek 4.32: Single Linkage, dla pierścienia. Szum od 40% do 70%

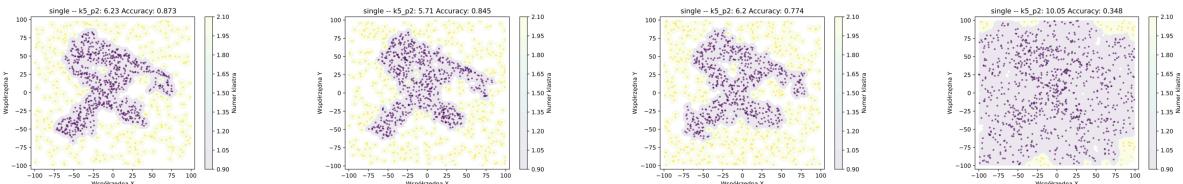


Rysunek 4.33: Single Linkage, dla półksiężyca. Szum od 40% do 70%

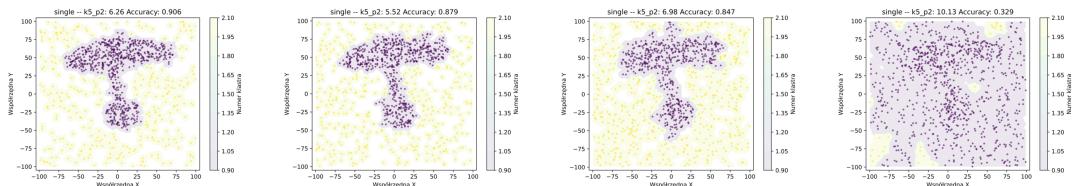
półksiężycami. Jest to najtrudniejszy jak dotąd zbiór, ponieważ posiada on dwa obszary będące skupiskami. W przypadku pierwszego wykresu algorytmowi udało się wykryć skupienie grupy (półksiężyce), z dokładnością ponad 79%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność około 15%. W pozostałych przypadkach, czyli dla szumu 50% oraz 60% widoczny jest znaczny spadek dokładności do 60% oraz 54%. Można zauważać, że algorytm dla pierwszych trzech wykresów wykrył skupienie oraz udało mu się wykryć szum między półksiężycami.

Rys. 4.34 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie `k5_p2`, a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. Wykresy przedstawiają wyniki grupowania dla zbioru z kształtem własnym, a dokładniej w kształcie litery „X”. W przypadku pierwszego wykresu algorytmowi udało się wykryć skupienie grupy („X”), z dokładnością ponad 77%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność około 34%. Co ciekawe dla szumu 50% algorytm uzyskał zblizioną dokładność niż dla 40%. W pozostałym przypadku, czyli dla szumu 60% widoczny jest znaczny spadek dokładności do 77,4%.

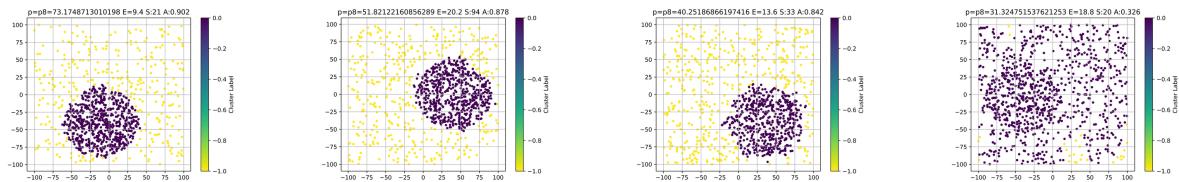
Rys. 4.35 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą Single Linkage dla danych zawierających różne poziomy szumu, od 40% do 70%. Każdy wykres



Rysunek 4.34: Single Linkage, dla kształtu „X”. Szum od 40% do 70%



Rysunek 4.35: Single Linkage, dla kształtu „grzyba”. Szum od 40% do 70%



Rysunek 4.36: DBSCAN, dla koła jednorodnego. Szum od 40% do 70%

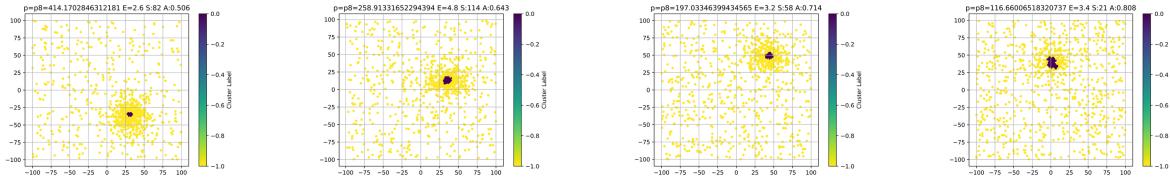
ma tytuł wskazujący metodę klasteryzacji oraz parametry użyte w eksperymencie `k5_p2`, a więc kryterium  $P_{2,k5}$ , a także dokładność uzyskaną dla danego poziomu szumu. Wykresy przedstawiają wyniki grupowania dla zbioru z kształtem własnym, a dokładniej w kształcie grzyba lub kotwica. Jest to ciekawy zbiór, ponieważ posiada on dwa obszary będące skupiskami oraz posiada on obszar między skupiskami (połączenie) niebędące szumem, co czyni go łatwiejszym od przypadku na rysunku 4.33. W przypadku pierwszych trzech wykresów algorytmowi udało się wykryć skupienie grupy (grzyb), z dokładnością ponad 90%. W przypadku szumu wynoszącego 70% algorytm nie poradził sobie, uzyskał dokładność około 34%. W pozostałych przypadkach widoczny jest spadek dokładności, lecz niewielki. Udało mu się uzyskać 87% oraz 84%, co jest dobrym wynikiem. Po raz kolejny można zauważać nagły spadek dokładności dla szumu wynoszącego 70%.

## DBSCAN

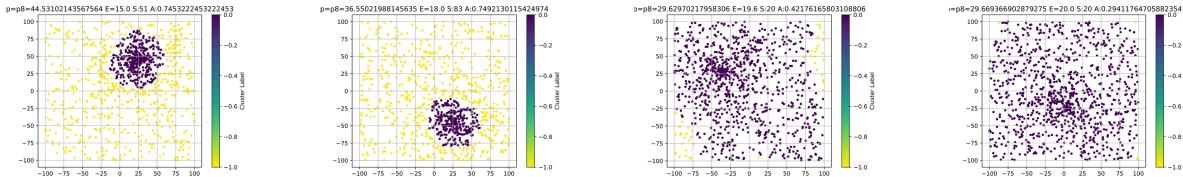
Poniżej prezentowane jest siedem rysункów dla algorytmu DBSCAN. Każdy zawiera po 4 wykresy dla szumu od 40% do 70%. Celem tego podrozdziału jest sprezentowanie wpływu szumu na działanie tego algorytmu, porównanie z algorytmem Single Linkage oraz wyciągnięcie wniosków. Zastosowano kryterium  $P_{2,k5}$ .

Rys. 4.36 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie koła jednorodnego, czyniło go jednym z łatwiejszych zbiorów, największa dokładność, równa 90% została osiągnięta dla najmniejszej wartości szumu 40%. Dla wartości 50% oraz 60% algorytm osiągnął podobne wyniki, lecz dokładność spadała do 84%. Dla wartości szumu 70% algorytm sobie nie poradził i osiągał dokładność 36%.

Rys. 4.37 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie koła z rozkładem von Misesa, co czyniło go znacznie trudniejszym zbiorem niż poprzedni. Wynika to z faktu, że istnieje bardzo duże za-



Rysunek 4.37: DBSCAN, dla von Misesa. Szum od 40% do 70%

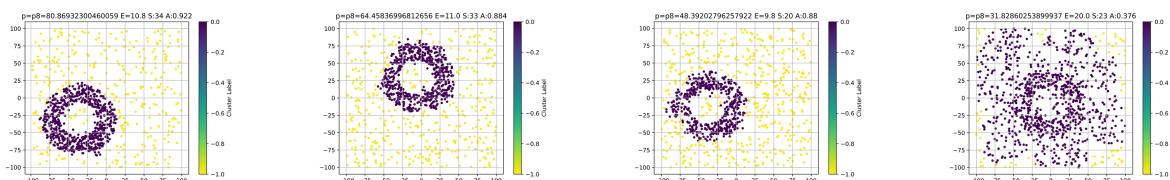


Rysunek 4.38: DBSCAN, dla rozkładu normalnego. Szum od 40% do 70%

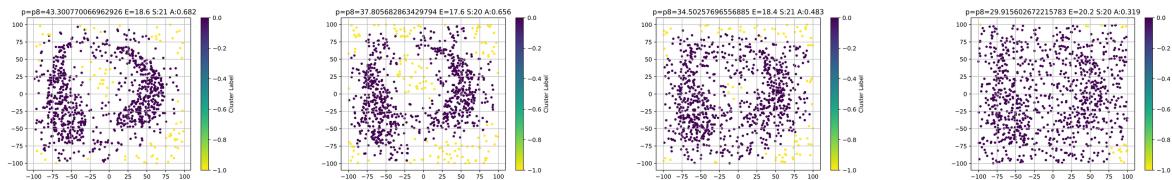
geszczenie w im bliżej środka koła. Największa dokładność, równa 80% została osiągnięta dla największej wartości szumu 70%, co jest rzadki stosunku do innych wyników. Dla wartości szumu 50% oraz 60% algorytm osiągnął podobne wyniki, dokładność w okolicy 64% – 61%. Dla wartości szumu 40% algorytm osiągnął dokładność najgorszą dokładność 50%. Warto zauważyć, że dokładność rosła wraz ze wzrostem szumu, co jest ciekawe i może wynikać z tego, że miary premiowały związkę klastry co w połączeniu z rozkładem von Misesa doprowadziło do stworzenia klastrów o małej liczebności punktów.

Rys. 4.38 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie koła z normalnym, co czyniło podobnym zbiorem niż poprzedni. Zagęszczenie punktów w pobliżu środka okręgu jest większe, niż w przypadku koła jednorodnego i mniejsze, niż przypadku rozkładu von Misesa. . Największa dokładność, równa 74% została osiągnięta dla wartości szumu 40% 50%. Dla wartości szumu 60% oraz 70% algorytm osiągnął podobne do siebie wyniki, a mianowicie nie wykrył poprawnie skupienia i osiągnął dokładność poniżej 42%.

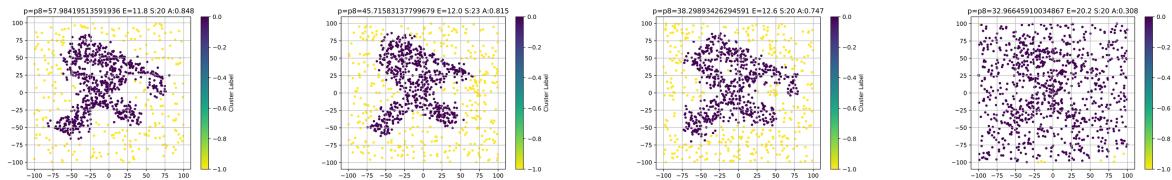
Rys. 4.39 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie pierścienia, czyniło go interesującym zbiorem, gdzie pojawiał się szum wewnętrz pierścienia. Największa dokładność, równa 92% została osiągnięta dla najmniejszej wartości szumu 40%. Dla wartości 50% oraz 60% algorytm osiągnął podobne wyniki, lecz dokładność spadała do 88%. Dla wartości szumu 70% al-



Rysunek 4.39: DBSCAN, dla pierścienia. Szum od 40% do 70%



Rysunek 4.40: DBSCAN, dla półksiężyca. Szum od 40% do 70%



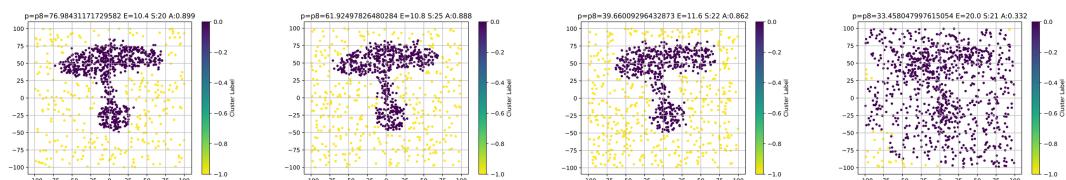
Rysunek 4.41: DBSCAN, dla kształtu „X”. Szum od 40% do 70%

algorytm sobie nie poradził i osiągał dokładność 37,6%. Algorytm osiągnął podobne wyniki jak dla koła jednorodnego, wraz ze wzrostem szumu zauważono spadek dokładności. Szczególnie jest to widoczne między wartościami 60% a 70% szumu.

Rys. 4.40 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie półksiężyca, czyniło go trudnym zbiorem, gdzie istniały dwa zageszczenia i szum między nimi. Największa dokładność, równą tylko 68% została osiągnięta dla najmniejszej wartości szumu 40%. Dla wartości 50% algorytm osiągnął podobny wynik w okolicy 65%. Dla wartości szumu 70% algorytm sobie nie poradził i osiągał dokładność 32%. Dla wartości 60% szumu algorytm osiągnął dokładność 48% zbliżoną do wartości 60% szumu.

Rys. 4.41 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie litery „X”, czyniło go ciekawym zbiorem. Największa dokładność, równą 84% została osiągnięta dla najmniejszej wartości szumu 40%. Dla wartości 50% algorytm osiągnął podobny wynik w okolicy 81,5%. Dla wartości szumu 70% algorytm sobie nie poradził i osiągał dokładność tylko 32%. Dla wartości 60% szumu algorytm osiągnął dokładność 74% zbliżoną do wartości 50% szumu. Można zauważyc wzrost dokładności przy spadku ilości szumu w zbiorze.

Rys. 4.42 przedstawia cztery wykresy ilustrujące wyniki klasteryzacji metodą DBSCAN. Zbiorem zawierał skupienie w kształcie grzyba lub kotwicy, czyniło go podobnym zbi-



Rysunek 4.42: DBSCAN, dla kształtu „grzyba”. Szum od 40% do 70%

rem do kształtu półksiężyca, ponieważ istnieją dwa skupienia, połączone wąskim pasmem. Największa dokładność, równa 90% została osiągnięta dla najmniejszej wartości szumu 40%. Dla wartości 50% algorytm osiągnął podobny wynik w okolicy 88%. Dla wartości 60% szumu algorytm osiągnął dokładność 86%, co jest zbliżone do wartości 50% oraz 40% szumu. Dla wartości szumu 70% algorytm sobie nie poradził i osiągał dokładność tylko 32%. Można zauważać nagły spadek dokładności dla ostatniego wykresów.

#### 4.3.3 Czas wykonywania obliczeń

Poniżej prezentowane są wyniki pomiaru czasu wykonywania się algorytmów DBSCAN, Single Linkage oraz Complete Linkage w trzech tabelach. Czas mierzony był w sekundach, Pierwsza tabela 4.1, prezentuje wyniki dla zbiorów danych o zagęszczeniu w kształcie koła jednorodnego rys. 4.1 oraz koła w rozkładzie von Misesa (rys. 4.2). Z kolei druga tabela 4.2, przedstawia wyniki dla zbiorów danych o zagęszczeniu w kształcie koła w rozkładzie normalnym (rys. 4.3) oraz dla pierścienia (rys. 4.4). Trzecia tabela 4.3, reprezentuje rezultaty dla zbiorów o zagęszczeniu w kształtach własnych, to są: kształt półksiężyca (rys. 4.5), litery „X” (rys. 4.6) oraz grzyba (rys. 4.7).

Można zauważać, że wraz ze zwiększeniem się średnicy koła czas wykonywanie algorytmu DBSCAN malał (dla `circle_A`  $t = 286.14$  [s] do  $t = 200.4$  [s]), dla każdej wartości szumu. Zależność się utrzymała dla każdego zbiorku danych. Dodatkowo można zauważać dla algorytmu Complete Linkage wykonywał się zawsze szybciej od Single Linkage, ta wartość wyniosła nawet 16% w przypadku `circle_A` przy szumie 70% oraz średnicy koła równej 70.

Na podstawie tabeli 4.2 pokazuje, że kształt danych (koło normalne vs pierścień) ma minimalny wpływ na czasy działania algorytmów Single Linkage i Complete Linkage, podczas gdy średnica koła również nie wpływa znaczco na te czasy. Algorytm DBSCAN działa znacznie wolniej niż algorytmy linkage w każdej analizowanej sytuacji. Jest on od 4 do 6 razy wolniejszy od algorytmów aglomeracyjnych, niezależnie od kształtu i średnicy koła.

Na podstawie tabeli 4.3 można powiedzieć, że algorytm DBSCAN podobnie jak w poprzedniej tabeli. Single Linkage oraz Complete Linkage wykazują zbliżone czasy działania, przy czym Complete Linkage jest nieznacznie szybszy w większości przypadków (lecz nie radził sobie z grupowaniem). Porównując wszystkie tabele, widzimy, że różnica w czasie działania między DBSCAN a algorytmami linkage utrzymuje się niezależnie od kształtu danych.

Tabela 4.1: Porównanie czasu algorytmów dla koła jednorodnego i z rozkładem normalnym

kształt	szum [%]	średnica koła	czas [s]		
			DBSCAN	SingleLinkage	CompleteLinkage
koło jednorodne	30	30	286.14	44.75	43.49
koło jednorodne	30	50	235.71	45.18	43.25
koło jednorodne	30	70	200.40	49.54	44.43
koło jednorodne	40	30	278.28	46.59	44.01
koło jednorodne	40	50	234.30	50.40	44.29
koło jednorodne	40	70	199.50	52.53	46.30
koło jednorodne	50	30	258.34	47.30	43.54
koło jednorodne	50	50	208.69	48.40	43.90
koło jednorodne	50	70	185.38	52.17	43.91
koło jednorodne	60	30	243.14	50.47	45.30
koło jednorodne	60	50	203.78	50.07	44.37
koło jednorodne	60	70	189.92	50.49	43.95
koło jednorodne	70	30	231.84	57.64	48.29
koło jednorodne	70	50	191.78	52.37	46.97
koło jednorodne	70	70	171.96	49.27	41.75
koło von Mises	30	30	377.60	44.44	43.68
koło von Mises	30	50	344.82	45.27	41.69
koło von Mises	30	70	327.25	46.69	42.86
koło von Mises	40	30	362.68	48.28	45.03
koło von Mises	40	50	330.17	48.43	45.02
koło von Mises	40	70	301.06	46.88	41.91
koło von Mises	50	30	309.01	46.94	43.05
koło von Mises	50	50	296.96	50.05	42.91
koło von Mises	50	70	282.20	51.68	44.03
koło von Mises	60	30	296.22	47.79	43.29
koło von Mises	60	50	262.59	51.74	44.19
koło von Mises	60	70	254.29	47.55	40.74
koło von Mises	70	30	248.72	47.12	40.53
koło von Mises	70	50	236.25	47.70	40.85
koło von Mises	70	70	228.53	49.49	40.82

Tabela 4.2: Porównanie czasu algorytmów dla koła z rozkładem normalnym i z pierścieniem

kształt	szum [%]	średnica koła	czas [s]		
			DBSCAN	SingleLinkage	CompleteLinkage
koło normalne	30	30	264.98	45.13	39.95
koło normalne	30	50	223.04	43.73	40.00
koło normalne	30	70	209.71	45.84	41.51
koło normalne	40	30	256.93	45.90	40.57
koło normalne	40	50	219.27	46.63	40.55
koło normalne	40	70	199.85	45.84	40.47
koło normalne	50	30	243.38	46.58	40.58
koło normalne	50	50	205.67	47.16	40.18
koło normalne	50	70	187.60	47.23	40.30
koło normalne	60	30	237.67	48.31	41.16
koło normalne	60	50	198.39	48.70	40.46
koło normalne	60	70	189.00	47.27	40.71
koło normalne	70	30	223.55	50.81	42.14
koło normalne	70	50	191.77	48.96	41.81
koło normalne	70	70	180.53	48.87	41.83
pierścień	30	30	269.91	43.38	40.37
pierścień	30	50	231.45	44.91	40.50
pierścień	30	70	204.03	47.56	42.73
pierścień	40	30	249.72	44.44	40.55
pierścień	40	50	210.07	45.01	40.55
pierścień	40	70	185.68	46.67	40.70
pierścień	50	30	232.30	45.79	40.67
pierścień	50	50	198.88	45.46	40.71
pierścień	50	70	185.79	47.87	41.24
pierścień	60	30	224.15	52.49	43.57
pierścień	60	50	219.66	52.12	47.82
pierścień	60	70	182.95	48.61	42.45
pierścień	70	30	202.75	50.15	41.25
pierścień	70	50	180.70	49.04	41.54
pierścień	70	70	177.16	48.14	41.20

Tabela 4.3: Porównanie czasu algorytmów dla koła z własnymi kształtami, półksiężyca, „X” oraz grzyb

kształt	szum [%]	czas [s]		
		<i>DBSCAN</i>	<i>SingleLinkage</i>	<i>CompleteLinkage</i>
półksiężyca	30	154.73	45.56	39.50
półksiężyca	40	164.02	45.14	40.91
półksiężyca	50	154.61	48.38	40.66
półksiężyca	60	159.22	50.11	42.72
półksiężyca	70	150.66	50.81	42.09
„X”	30	187.75	41.81	40.39
„X”	40	174.32	43.68	41.28
„X”	50	171.98	47.88	41.64
„X”	60	164.27	47.39	41.79
„X”	70	157.82	48.06	40.93
grzyb	30	201.86	44.50	39.04
grzyb	40	198.21	46.40	40.06
grzyb	50	192.42	60.49	43.08
grzyb	60	168.76	46.37	43.88
grzyb	70	171.94	49.51	41.06



# Rozdział 5

## Podsumowanie

W niniejszej pracy magisterskiej podjęto temat z dziedziny analizy danych, koncentrując się na problemie granularnego grupowania danych w jedną grupę. Granularne grupowanie danych to technika mająca na celu podzielenie zbioru danych na mniejsze, jednorodne grupy, zwane klastrami. Wyzwanie polegało na znalezieniu jednego dużego klastra, który byłby wewnętrznie spójny i dobrze oddzielony od reszty danych, stanowiących szum. W celu określenia granicy pojedynczego klastra opracowano kryteria oparte na zasadzie uzasadnionej granulacji. Kryteria te zakładają, że im większa liczba punktów w grupie, tym wyższa jest wartość kryterium, a także że im mniejsza jest średnia odległość między punktami, tym wyższa jest wartość kryterium. Dzięki temu granice klastra są jasno określone: klaster musi być duży i jednocześnie wewnętrznie jednorodny, co minimalizuje wpływ szumu i zapewnia, że klaster rzeczywiście reprezentuje istotne dane.

Podczas badań przeanalizowano i porównano różne algorytmy klasteryzacji, takie jak DBSCAN, Single Linkage oraz Complete Linkage, w kontekście granularnego grupowania danych. Przeprowadzono eksperymenty na siedmiu różnych zbiorach danych, zdefiniowano własne kryteria oceny grupowania, aby określić optymalne parametry dla tych algorytmów. Następnie porównywano poszczególne algorytmy i kryteria pod względem do dokładności. Dodatkowo eksperymentowano z zawartością szumu w zbiorach danych, aby zbadać jego wpływ na możliwości grupowania.

Na podstawie uzyskanych wyników można wywnioskować, że algorytm DBSCAN dobrze radzi sobie z różnymi kształtami, poprawnie wykrył skupiska w sześciu z siedmiu prezentowanych zbiorów, co czyni go efektywnym w różnych warunkach. Jednak wymaga precyzyjnego doboru parametrów  $\varepsilon$  i  $MinPts$ , co może być trudne, więc wymaga jakiejś formy wstępnego uzyskania tych parametrów. Z kolei algorytmy Single Linkage i Complete Linkage wymagają wstępnego określenia momentu przerwana scalania klastrów, to także tworzy potrzebę znalezienia tych parametrów.

Podczas badań analizowano wpływ szumu na efektywność algorytmów. Zauważono, że szum znaczaco wpływał na dokładność klasteryzacji, a efekty te różniły się w zależności od zastosowanego algorytmu i kształtu skupień. Algorytm Complete Linkage nie

radził sobie z grupowaniem niezależnie od ilości szumu. W większości wypadków dla algorytmów DBSCAN oraz Single Linkage dokładność grupowania malała wraz ze wzrostem szumu. Dla 70% szumu algorytmy wykazały znaczący spadek dokładności, co było naglem spadkiem w stosunku do 60% szumu. Jeśli chodzi o kształty to koło w rozkładzie von Misesa okazało się bardzo ciężkie dla algorytmu DBSCAN, co ciekawe Single Linkage lepiej poradził sobie z tym zbiorem.

Jeśli chodzi o czas wykonywana się algorytmów to DBSCAN był znacznie wolniejszy od algorytmów aglomeracyjnych. Zarówno Single Linkage, jak i Complete Linkage miały zbliżony czas wykonywania. Warto zauważyć, że kształt nie miał istotnego wpływu na czas wykonywania się algorytmów. Dalsze prace mogą skupić się na tworzeniu lepszych kryteriów procesu doboru optymalnych parametrów dla algorytmów grupowania, co może zwiększyć ich użyteczność i efektywność. Ponadto warto rozważyć przeprowadzenie badań na większych i bardziej zróżnicowanych zbiorach danych, aby jeszcze lepiej zrozumieć ich zachowanie w różnych warunkach. Możliwe jest również rozszerzenie badań o analizę innych metod analizy skupisk oraz ich hybrydowych wersji.

# Bibliografia

- [1] Samir Kumar Bandyopadhyay i Tuhin Utsab Paul. „Segmentation of brain tumour from MRI image analysis of k-means and dbSCAN clustering”. W: *International Journal of Research in Engineering and Science* 1.1 (2013), s. 48–57.
- [2] Lauren Barghout. „Visual Taxometric Approach to Image Segmentation Using Fuzzy-Spatial Taxon Cut Yields Contextually Relevant Regions”. W: *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Red. Anne Laurent, Olivier Strauss, Bernadette Bouchon-Meunier i Ronald R. Yager. Cham: Springer International Publishing, 2014, s. 163–173. ISBN: 978-3-319-08855-6.
- [3] T. Cover i P. Hart. „Nearest neighbor pattern classification”. W: *IEEE Transactions on Information Theory* 13.1 (1967), s. 21–27. DOI: 10.1109/TIT.1967.1053964.
- [4] Radu Cretulescu, Daniel Morariu, Macarie Breazu i Daniel Volovici. „DBSCAN Algorithm for Document Clustering”. W: *International Journal of Advanced Statistics and IT&C for Economics and Life Sciences* 9 (czer. 2019), s. 58–66. DOI: 10.2478/ijasitels-2019-0007.
- [5] D. Defays. „An efficient algorithm for a complete link method”. W: *The Computer Journal* 20.4 (sty. 1977), s. 364–366. ISSN: 0010-4620. DOI: 10.1093/comjnl/20.4.364. eprint: <https://academic.oup.com/comjnl/article-pdf/20/4/364/1108735/200364.pdf>. URL: <https://doi.org/10.1093/comjnl/20.4.364>.
- [6] Vimal Dubey i Amit Saxena. „A Cosine-Similarity Mutual-Information Approach for Feature Selection on High Dimensional Datasets”. W: *Journal of Information Technology Research* 10 (sty. 2017), s. 15–28. DOI: 10.4018/JITR.2017010102.
- [7] Martin Ester, Hans-Peter Kriegel, Jörg Sander i Xiaowei Xu. „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. W: *Knowledge Discovery and Data Mining*. 1996, s. 226–231. URL: <https://api.semanticscholar.org/CorpusID:355163>.
- [8] Brian Everitt. *Cluster Analysis*. ISBN 9780470977811. Chichester, West Sussex, U.K.: Wiley, 2011, s. 92–93.
- [9] Brian S. Everitt, Sabine Landau, Morven Leese i Daniel Stahl. *Cluster Analysis*. Wiley, 2011, s. 73–84. ISBN: 9780470749913.

- [10] Evelyn Fix i J. L. Hodges. „Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. W: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), s. 238–247. ISSN: 03067734, 17515823. URL: <http://www.jstor.org/stable/1403797> (term. wiz. 10.06.2024).
- [11] J. C. Gower i G. J. S. Ross. „Minimum Spanning Trees and Single Linkage Cluster Analysis”. W: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 18.1 (1969), s. 54–64. ISSN: 00359254, 14679876. URL: <http://www.jstor.org/stable/2346439>.
- [12] Xu He, Fan Min i William Zhu. „A Comparative Study of Discretization Approaches for Granular Association Rule Mining”. W: *Canadian Conference on Electrical and Computer Engineering* 37 (grud. 2012), s. 3. DOI: 10.1109/CCECE.2013.6567823.
- [13] W. E. Henley i D. J. Hand. „A K-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk”. W: *Journal of the Royal Statistical Society Series D: The Statistician* 45.1 (grud. 2018), s. 77–95. ISSN: 2515-7884. DOI: 10.2307/2348414. URL: <https://doi.org/10.2307/2348414>.
- [14] A.S.M. Shahadat Hossain. „Customer segmentation using centroid based and density based clustering algorithms”. W: *2017 3rd International Conference on Electrical Information and Communication Technology (EICT)*. 2017, s. 1–6. DOI: 10.1109/EICT.2017.8275249.
- [15] JiYe Liang i Yuhua Qian. „Information granules and entropy theory in information systems”. W: *Science in China Series F: Information Sciences* 51 (paź. 2008), s. 1427–1444. DOI: 10.1007/s11432-008-0113-2.
- [16] Wenfei Liu, Jingcheng Wei i Qingmin Meng. „Comparisons on KNN, SVM, BP and the CNN for Handwritten Digit Recognition”. W: *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. 2020, s. 587–590. DOI: 10.1109/AEECA49918.2020.9213482.
- [17] Glenn W. Milligan. „An examination of the effect of six types of error perturbation on fifteen clustering algorithms”. W: *Psychometrika* 45 (1980), s. 325–342. DOI: 10.1007/BF02293907. URL: <https://doi.org/10.1007/BF02293907>.
- [18] Peter Mills. „Efficient statistical classification of satellite measurements”. W: *International Journal of Remote Sensing* 32 (lut. 2012), s. 6120–6132. DOI: 10.1080/01431161.2010.507795.
- [19] Fionn Murtagh i Pedro Contreras. „Algorithms for hierarchical clustering: an overview”. W: *WIREs Data Mining and Knowledge Discovery* 2.1 (2012), s. 86–97. DOI: <https://doi.org/10.1002/widm.53>. URL: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.53>.

- [20] Amit Pandey i Achin Jain. „Comparative Analysis of KNN Algorithm using Various Normalization Techniques”. W: *International Journal of Computer Network and Information Security* 9 (list. 2017), s. 36–42. DOI: 10.5815/ijcnis.2017.11.04.
- [21] Witold Pedrycz. „The Principle of Justifiable Granularity”. W: *An Introduction to Computing with Fuzzy Sets: Analysis, Design, and Applications*. Cham: Springer International Publishing, 2021, s. 147–160. ISBN: 978-3-030-52800-3. DOI: 10.1007/978-3-030-52800-3\_10. URL: [https://doi.org/10.1007/978-3-030-52800-3\\_10](https://doi.org/10.1007/978-3-030-52800-3_10).
- [22] Witold Pedrycz i Shyi-Ming Chen. „Granular computing and intelligent systems: design with information granules of higher order and higher type”. W: 2011. URL: <https://api.semanticscholar.org/CorpusID:117930725>.
- [23] Witold Pedrycz i Wladyslaw Homenda. „Building the fundamentals of granular computing: A principle of justifiable granularity”. W: *Applied Soft Computing* 13.10 (2013), s. 4209–4218. ISSN: 1568-4946. DOI: <https://doi.org/10.1016/j.asoc.2013.06.017>. URL: <https://www.sciencedirect.com/science/article/pii/S1568494613002068>.
- [24] Yuhua Qian, Honghong Cheng, Jieting Wang, Jiye Liang, Witold Pedrycz i Chuangyin Dang. „Grouping granular structures in human granulation intelligence”. W: *Information Sciences* 382 (list. 2016), s. 150–169. DOI: 10.1016/j.ins.2016.11.024.
- [25] Alireza S. Shirkhorshidi, Saeed Aghabozorgi i Wah TY. „A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data”. W: *PLoS ONE* 10.12 (2015), e0144059. DOI: 10.1371.
- [26] R. Sibson. „SLINK: An optimally efficient algorithm for the single-link cluster method”. W: *The Computer Journal* 16.1 (sty. 1973), s. 30–34. ISSN: 0010-4620. DOI: 10.1093/comjnl/16.1.30. eprint: <https://academic.oup.com/comjnl/article-pdf/16/1/30/1196082/160030.pdf>. URL: <https://doi.org/10.1093/comjnl/16.1.30>.
- [27] Yunsheng Song, Jiye Liang, Jing Lu i Xingwang Zhao. „An efficient instance selection algorithm for k nearest neighbor regression”. W: *Neurocomputing* 251 (2017), s. 26–34. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2017.04.018>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231217306884>.
- [28] Thanh N. Tran, Ron Wehrens i Lutgarde M.C. Buydens. „KNN-kernel density-based clustering for high-dimensional multivariate data”. W: *Computational Statistics and Data Analysis* 51.2 (2006), s. 513–525. ISSN: 0167-9473. DOI: <https://doi.org/10.1016/j.csda.2005.10.001>.

- [29] Cleber Zanchettin, Byron Leite Dantas Bezerra i Washington W. Azevedo. „A KNN-SVM hybrid model for cursive handwriting recognition”. W: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. 2012, s. 1–8. DOI: [10.1109/IJCNN.2012.6252719](https://doi.org/10.1109/IJCNN.2012.6252719).

## **Dodatki**



# Spis skrótów i symboli

$d$  wymiarowość danych

$\bar{d}$  średnia odległość między punktami w grupie

$\varepsilon$  parametr algorytmu DBSCAN

$MinPts$  parametr algorytmu DBSCAN

$N$  liczba punktów w klastrze

$\bar{d}_k$  średnia odległość między k najbliższymi punktami

$P_1$  liczba punktów pomnożona przez odwrotność średniej odległości między punktami

$P_2$  liczba punktów pomnożona przez kwadrat odwrotności średniej odległości między punktami

$P_3$  pierwiastek liczby punktów pomnożony przez kwadrat odwrotności średniej odległości między punktami

CSV format pliku używany do zapisywania wyników



# Kody źródłowe

[Kody źródłowe powstałego w ramach pracy systemu są dostępne w repozytorium: ].



# Spis rysunków

2.1	Pseudokod algorytmu DBSCAN. . . . .	5
2.2	Pseudokod funkcji rozszerzania klastra oraz wyszukiwania sąsiadujących punktów w obszarze $\varepsilon$ . . . . .	6
2.3	Algorytm Single Linkage łączy punkty w klastry, zaczynając od najbliższych sobie par, a kończąc na pojedynczym skupisku obejmującym wszystkie punkty. . . . .	8
2.4	Algorytm Complete Linkage łączy punkty w klastry na podstawie największej odległości między punktami w klastrach, zaczynając od najbliższych sobie par, a kończąc na pojedynczym skupisku obejmującym wszystkie punkty.	10
3.1	Algorytm oblicza średnią odległość do $k$ najbliższych sąsiadów w zbiorze punktów klastra dla różnych wartości $k$ . . . . .	20
4.1	Koło jednorodne . . . . .	22
4.2	Rozkład von Misesa . . . . .	22
4.3	Rozkład normalny . . . . .	23
4.4	Kształt pierścienia . . . . .	23
4.5	Kształt półksiężyca, ( <code>shape_A</code> ) . . . . .	23
4.6	Kształt „X”, ( <code>shape_B</code> ) . . . . .	23
4.7	Kształt grzyba, ( <code>shape_C</code> ) . . . . .	23
4.8	Algorytm Complete Linkage dla zbioru koła jednorodnego . . . . .	25
4.9	Algorytm Single Linkage dla zbioru koła jednorodnego . . . . .	26
4.10	Algorytm DBSCAN dla zbioru koła jednorodnego . . . . .	27
4.11	Algorytm Complete Linkage dla zbioru rozkładu von Misesa . . . . .	28
4.12	Algorytm Single Linkage dla zbioru rozkładu von Misesa . . . . .	28
4.13	Algorytm DBSCAN dla zbioru rozkładu von Misesa . . . . .	29
4.14	Algorytm Complete Linkage dla zbioru rozkładu normalnego . . . . .	29
4.15	Algorytm Single Linkage dla zbioru rozkładu normalnego . . . . .	30
4.16	Algorytm DBSCAN dla zbioru rozkładu normalnego . . . . .	31
4.17	Algorytm Complete Linkage dla zbioru z pierścieniem . . . . .	32
4.18	Algorytm Single Linkage dla zbioru z pierścieniem . . . . .	33

4.19 Algorytm DBSCAN dla zbioru z pierścieniem . . . . .	34
4.20 Algorytm Complete Linkage dla zbioru z kształtem półksiężyca . . . . .	34
4.21 Algorytm Single Linkage dla zbioru z kształtem półksiężyca . . . . .	35
4.22 Algorytm DBSCAN dla zbioru z kształtem półksiężyca . . . . .	36
4.23 Algorytm Complete Linkage dla zbioru z literą „X” . . . . .	37
4.24 Algorytm Single Linkage dla zbioru z literą „X” . . . . .	37
4.25 Algorytm DBSCAN dla zbioru z literą „X” . . . . .	38
4.26 Algorytm Complete Linkage dla zbioru z kształtem grzyba . . . . .	39
4.27 Algorytm Single Linkage dla zbioru z kształtem grzyba . . . . .	40
4.28 Algorytm DBSCAN dla zbioru z kształtem grzyba . . . . .	41
4.29 Single Linkage, dla koła jednorodnego. Szum od 40% do 70% . . . . .	42
4.30 Single Linkage, dla von Misesa. Szum od 40% do 70% . . . . .	43
4.31 Single Linkage, dla rozkładu normalnego. Szum od 40% do 70% . . . . .	43
4.32 Single Linkage, dla pierścienia. Szum od 40% do 70% . . . . .	44
4.33 Single Linkage, dla półksiężyca. Szum od 40% do 70% . . . . .	44
4.34 Single Linkage, dla kształtu „X”. Szum od 40% do 70% . . . . .	44
4.35 Single Linkage, dla kształtu „grzyba”. Szum od 40% do 70% . . . . .	45
4.36 DBSCAN, dla koła jednorodnego. Szum od 40% do 70% . . . . .	45
4.37 DBSCAN, dla von Misesa. Szum od 40% do 70% . . . . .	46
4.38 DBSCAN, dla rozkładu normalnego. Szum od 40% do 70% . . . . .	46
4.39 DBSCAN, dla pierścienia. Szum od 40% do 70% . . . . .	46
4.40 DBSCAN, dla półksiężyca. Szum od 40% do 70% . . . . .	47
4.41 DBSCAN, dla kształtu „X”. Szum od 40% do 70% . . . . .	47
4.42 DBSCAN, dla kształtu „grzyba”. Szum od 40% do 70% . . . . .	47

# Spis tabel

4.1	Porównanie czasu algorytmów dla koła jednorodnego i z rozkładem normalnym	49
4.2	Porównanie czasu algorytmów dla koła z rozkładem normalnym i z pierścieniem	50
4.3	Porównanie czasu algorytmów dla koła z własnymi kształtami, półksiężyca, „X” oraz grzyb	51