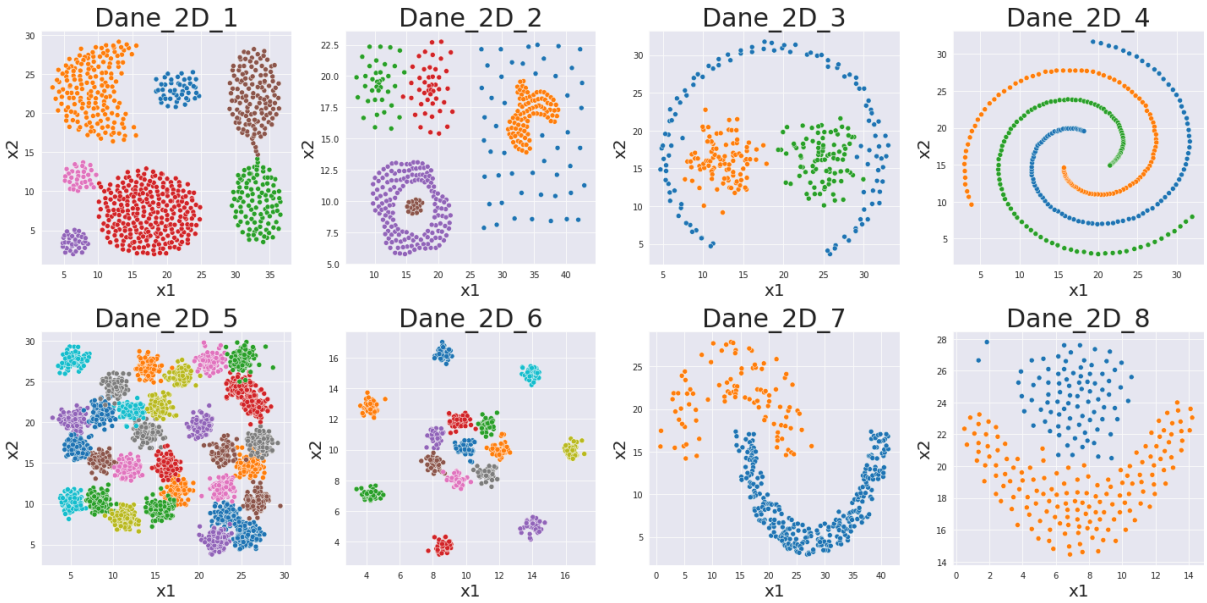
MPUM - Projekt 4

Marcin Wykpis

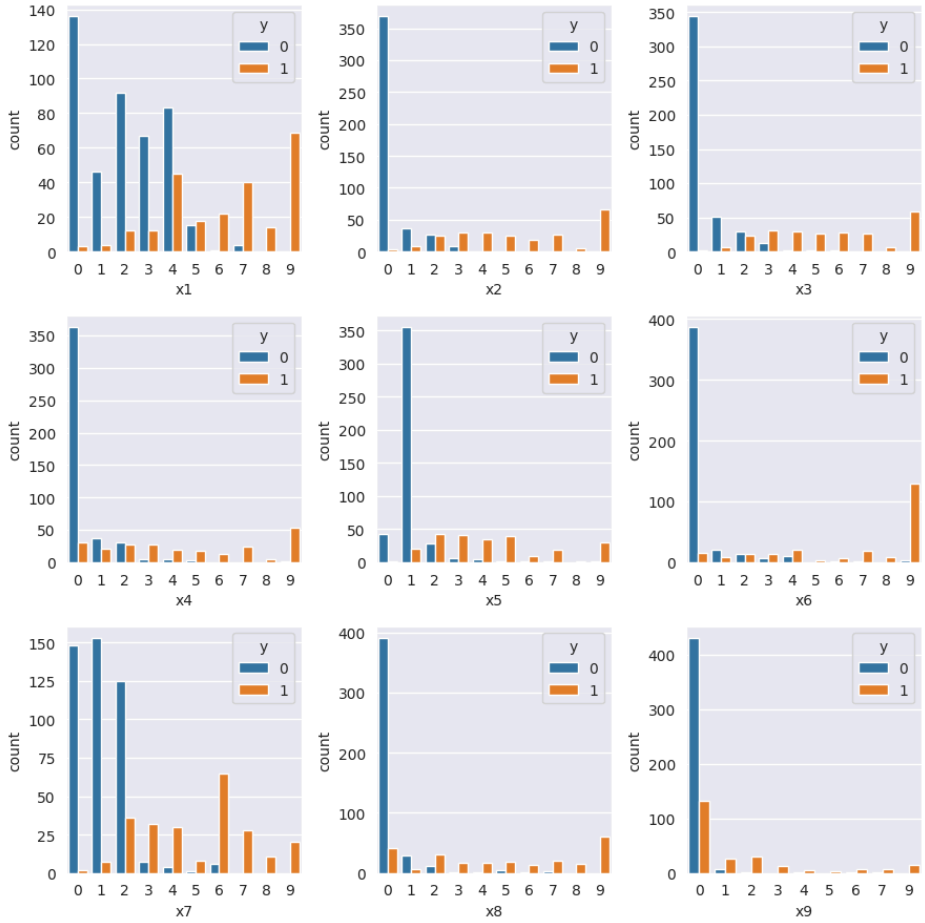
1. Analiza danych

Pierwsze 8 zbiorów danych zawiera zbiory punktów na płaszczyźnie. Dane te można łatwo zwizualizować na wykresie. Różnymi kolorami została zaznaczona przynależność do kolejnych klastrów.



W pliku `data_18D` znajdują się 18 wymiarowe punkty. Dataset nie posiada klasyfikacji na kolejne klastry. Dla tego zbioru danych przyjrzymy się „*Metody Łokcia*" dla algorytmu `K-Średnich`, w celu rozstrzygnięcia, na ile klastrów należałoby podzielić ten zbiór punktów.

W pliku `rp.data` znajduje się dataset z `miniprojektu 2`. Dane te są wielowymiarowe, a kolumna wynikowa przyjmuje jedynie wartości $\{0, 1\}$. Będziemy zatem jedynie próbować uzyskać jak najlepszy model według pewnej funkcji straty. Poniżej prezentuje się rozkład kolejnych cech ze względu na klaster, w którym powinny się znaleźć. Wartość kolumny y ma pewną interpretację, dlatego też będziemy rozróżniali utworzone klastry.



2. Przygotowanie danych

Wejściowe dane zostały przeskalowane metodą `min-max` do zbioru $[0, 1]$.
Zaprezentowane metody w tym miniprojekcie nie wymagają podziału na zbiory treningowe, walidacyjne, testowe.

3. Ocena modelu

W celu porównania wytrenowanego modelu z prawdziwym przyporządkowaniem klastrów nie wystarczy porównać kolejno przydzielonych punktom numerów klastrów. Zauważmy, że ocena modelu powinna być w pełni symetryczna ze względu na permutację zbiorów klastrów.
Zamiast tego przypatrzmy się, ile klastrów zostało słusznie skorelowanych (lub rozdzielonych) ze sobą.
Dla każdego modelu będziemy obliczać dokładność zdefiniowaną jako iloraz:

$$\mathrm{Acc} = \frac{|\{\{u, v\} \mid u \in C \Leftrightarrow v \in C|)}{\frac{1}{2}N(N-1)}$$

4. Metoda K-Średnich

Metoda K-Średnich polega na wyznaczaniu tak zwanych centroidów dla każdego z klastrów. Centroidy te będą intuicyjnie odpowiadać „średniemu" punktowi w danym klastrze.
Sam algorytm polega na wielokrotnym powtarzaniu następujących operacji, pozwalających na odpowiednie zbieganie położenia centroidów.
Dla początkowo losowo wybrancyh centroidów $\mu_1, ..., \mu_k$:

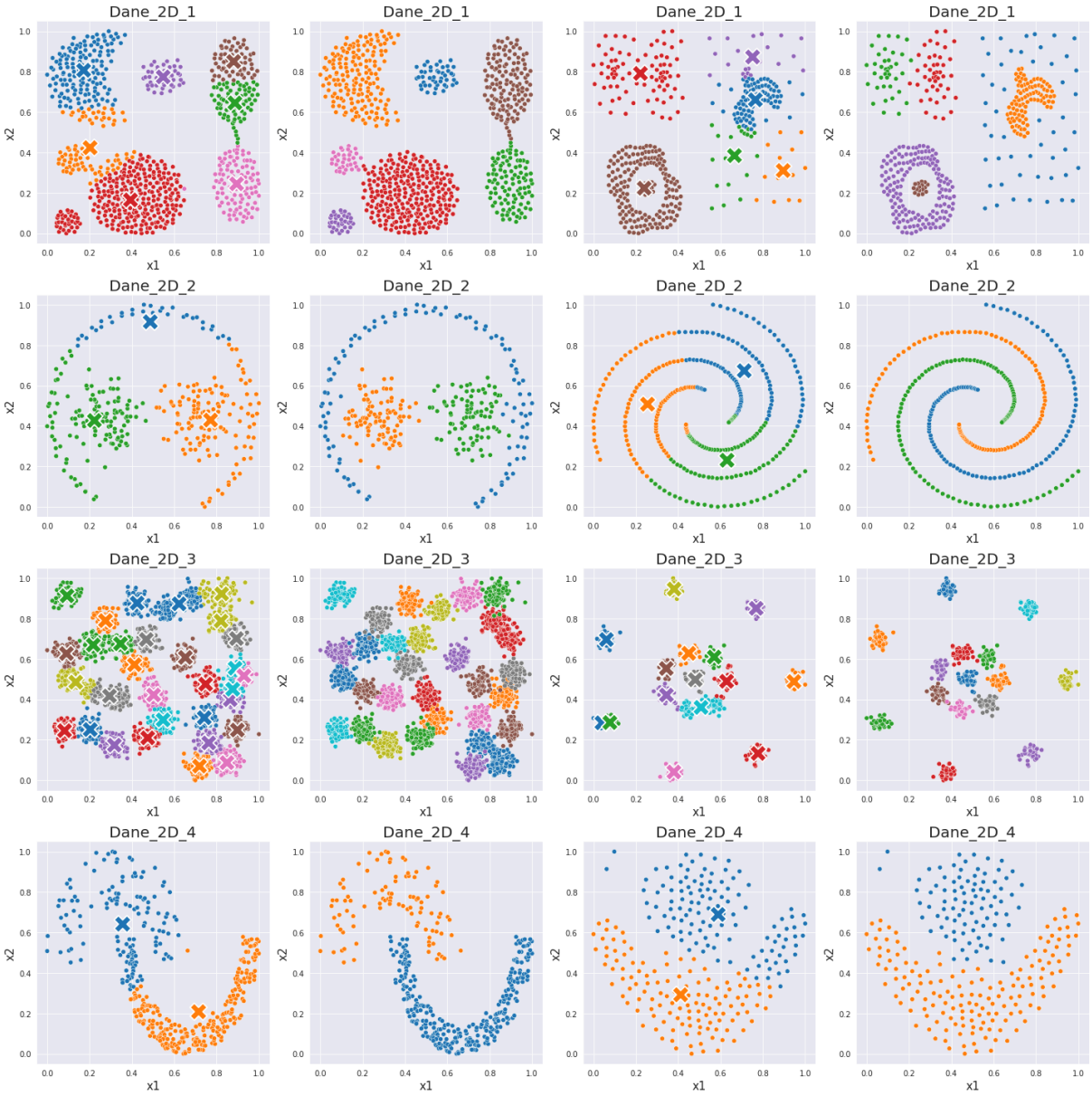
$$C_j := \left\{ x \in D : j = \arg\min_l d(x, \mu_l) \right\}$$

$$\mu_j := \arg\min_{\mu \in \mathcal{X}} \sum_{x \in C_j} d(x, \mu)^2$$

Do wyznaczania odległości między dwoma punktami użyłem metryki euklidesowej.

Poniżej prezentuje się tabelka wyników dla kolejnych zbiorów danych 2D. Dokładność podana została w oparciu na 10 przebiegów dla różnego wyboru startowych centroidów. Spośród nich wybrany został najlepszy wynik. Czas trenowania modelu dla `1000` iteracji był pomijalny.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.9204	0.9112	0.7478	0.5543	0.9912	0.9871	0.7913	0.7558


Dane_2D_1
Dane_2D_1
Dane_2D_1
Dane_2D_1
Dane_2D_2
Dane_2D_2
Dane_2D_2
Dane_2D_2
Dane_2D_3
Dane_2D_3
Dane_2D_3
Dane_2D_3
Dane_2D_4
Dane_2D_4
Dane_2D_4
Dane_2D_4
x1
x2

5. Grupowanie Hierarchiczne

Metoda grupowania hierarchicznego polega na kolejnym łączeniu najbliższych (w pewnej metryce, np. euklidesowej) kolejnych klastrów.
Odległość połączonego klastra od innych klastrów możemy wyznaczać na kilka sposobów. Zaimplementowane metody łączenia, na których przetestowane zostały modele to łączenia:

- pojedyncze (single)
- pełne (full)
- średnie (mean)
- centroidalne
- Warda

Metoda ta została zaimplementowana na podstawie ***Algorytmu Lance'a-Williasa***, przedstawionego na wykładzie.
Nową odległość klastra K od klastra C powstałego z połączenia klastrów A i B wyliczamy w następujący sposób:

$$D_{KC} = \alpha_A D_{KA} + \alpha_B D_{KB} + \beta D_{AB} + \gamma \, |D_{KA} - D_{KB}|$$

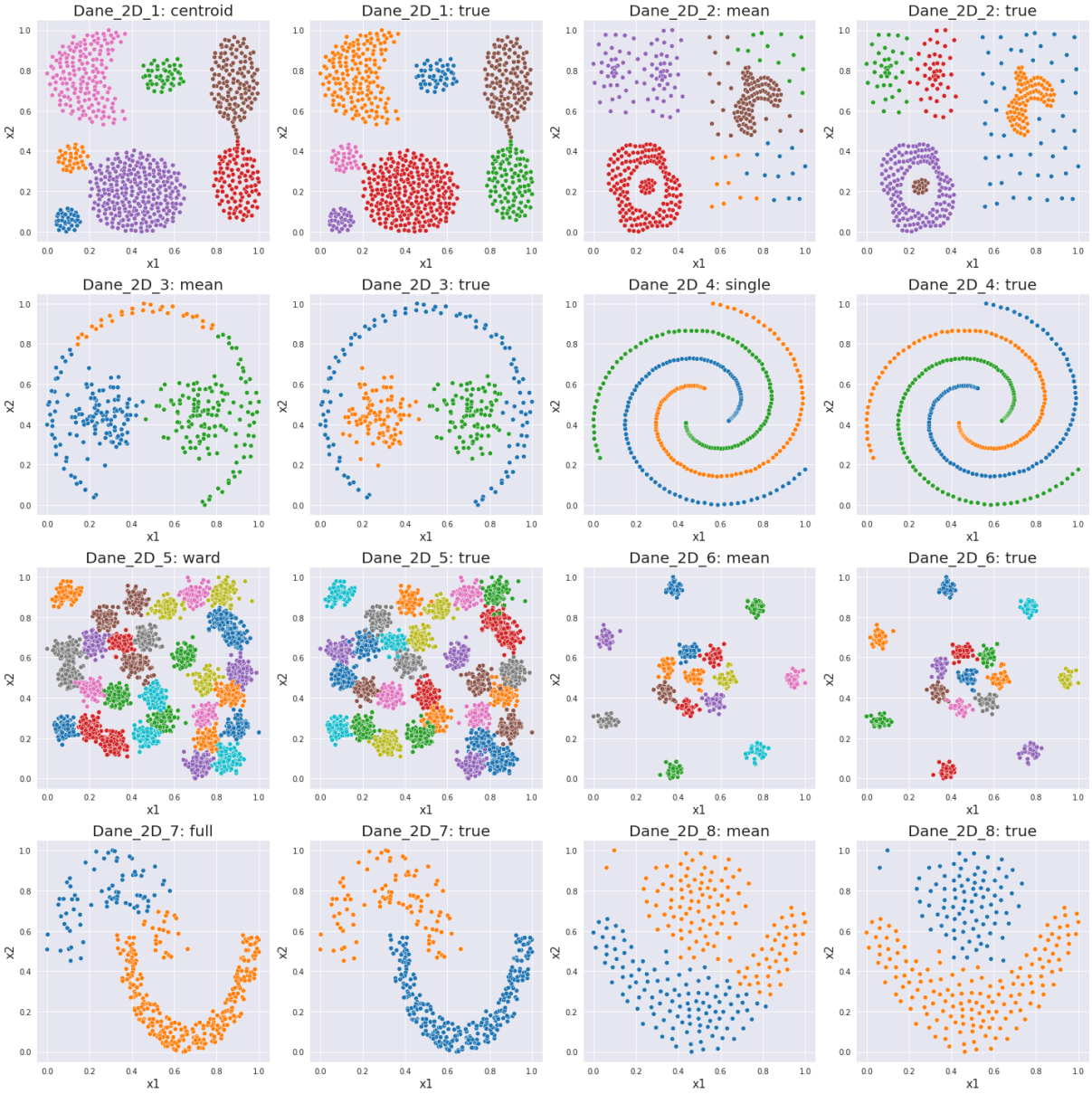
Gdzie parametry $\alpha_A, \alpha_B, \beta, \gamma$ zdefiniowane są następująco:

metoda łączenia	α_A	α_B	β	γ
pojedyncze	$1/2$	$1/2$	0	$-1/2$
pełne	$1/2$	$1/2$	0	$1/2$
średnie	$\frac{\lvert A\rvert}{\lvert A\rvert+\lvert B\rvert}$	$\frac{\lvert B\rvert}{\lvert A\rvert+\lvert B\rvert}$	0	0
centroidalne	$\frac{\lvert A\rvert}{\lvert A\rvert+\lvert B\rvert}$	$\frac{\lvert B\rvert}{\lvert A\rvert+\lvert B\rvert}$	$-\frac{\lvert A\rvert\lvert B\rvert}{(\lvert A\rvert+\lvert B\rvert)^2}$	0
Warda	$\frac{\lvert A\rvert+\lvert K\rvert}{\lvert A\rvert+\lvert B\rvert+\lvert K\rvert}$	$\frac{\lvert B\rvert+\lvert K\rvert}{\lvert A\rvert+\lvert B\rvert+\lvert K\rvert}$	$-\frac{\lvert K\rvert}{\lvert A\rvert+\lvert B\rvert+\lvert K\rvert}$	0

Wydaje mi się, że tabelka podana na wykładzie posiada drobny błąd. Zdziwiła mnie trochę obecność $|C|$ we wzorze Warda, gdyż jest to po prostu $|A| + |B|$. Znalazłem w zewnętrznych źródłach inne współczynniki, tym razem zależne od $|K|$. Postanowiłem więc użyć tej wersji.

Poniżej znajduje się tabelka z wynikami na każdej z metod i czasami trenowania. Wyniki te zostały uzyskane na 1 przebiegu algorytmu, gdyż nie dzielimy zbioru danych, ani kolejność punktów w zbiorze nie powinna mieć znaczenia.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
centroid	mean	mean	single	ward	mean	full	mean
1.0	0.9164	0.7379	1.0	0.9949	0.9987	0.8840	0.7211


Dane_2D_1: centroid
Dane_2D_1: true
Dane_2D_2: mean
Dane_2D_2: true
Dane_2D_3: mean
Dane_2D_3: true
Dane_2D_4: single
Dane_2D_4: true
Dane_2D_5: ward
Dane_2D_5: true
Dane_2D_6: mean
Dane_2D_6: true
Dane_2D_7: full
Dane_2D_7: true
Dane_2D_8: mean
Dane_2D_8: true
x1
x2

6. Klasteryzacja Spektralna

W metodzie klasteryzacji spektralnej tworzymy macierz sąsiedztwa:

$$A_{ij} = \begin{cases} w_{ij}, & \text{gdy } \{i,j\} \in E, \\ 0, & \text{gdy } \{i,j\} \notin E \end{cases}$$

Ponadto tworzymy *laplasjan* L grafu definiując

$$D_{ij} = \begin{cases} \sum_{j:\{i,j\}\in E} w_{ij}, & \text{gdy } i = j, \\ 0, & \text{gdy } i \neq j \end{cases}$$

Wówczas $L = D - A$.
Następnie normalizujemy L:

$$L_{\text{norm}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

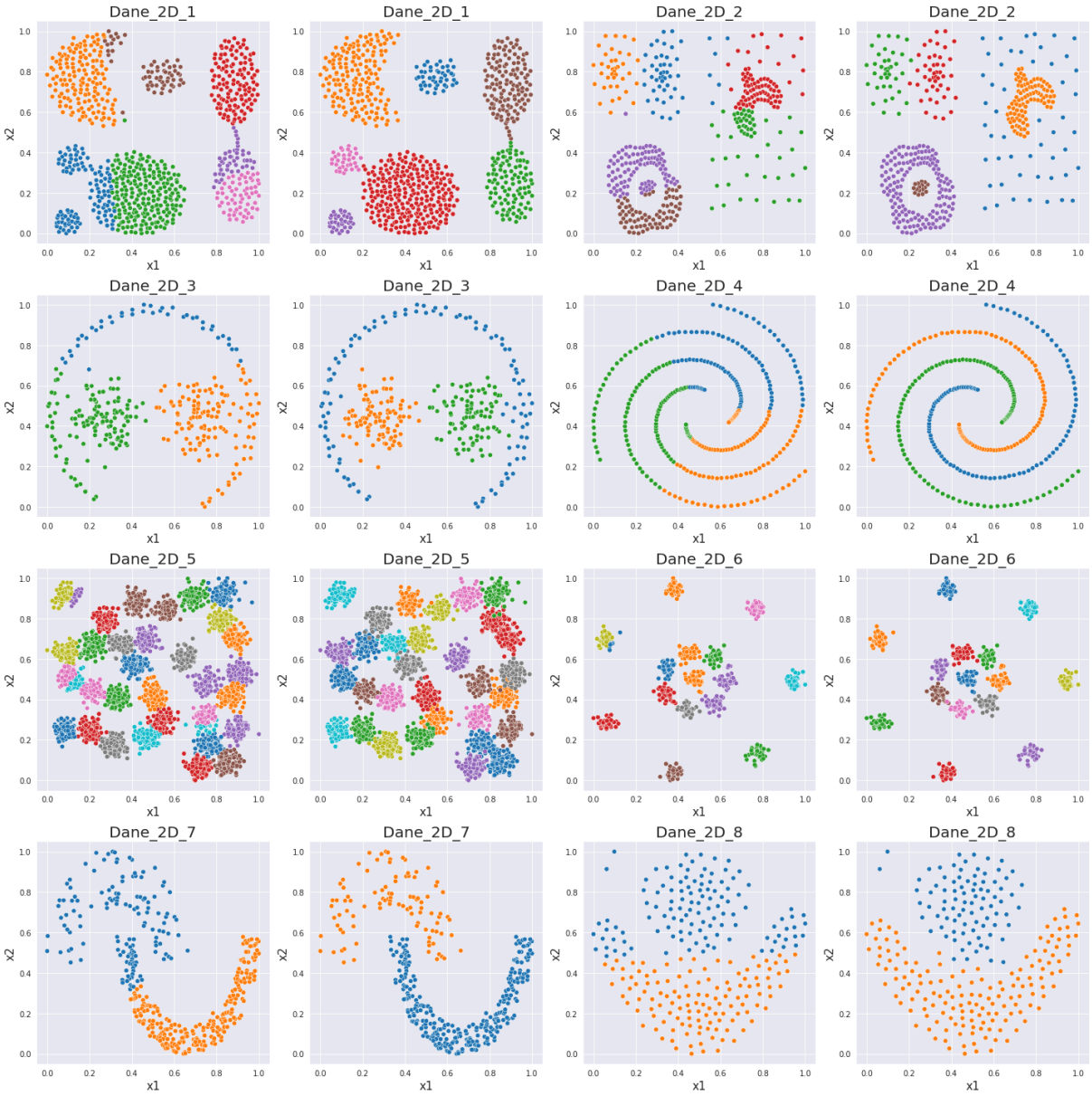
.

Dla macierzy L_{norm} wyznaczamy pierwsze (według porządku na wartościach własnych) k wektorów własnych $v_1, ..., v_k$ i ustawiamy je w macierz $\mathbb{R}^{m \times k}$. Jako k przyjmujemy hiperparametr - docelową liczbę klastrów.
Utworzoną macierz traktujemy jako macierz k-wymiarowych punktów i przepuszczamy ją na pewnym modelu klasteryzacji (np. jednym z wcześniej opisanych modeli).

Podczas implementacji tej metody napotkałem problemy z wyliczaniem wektorów własnych w przypadku gdy utworzony Laplacjan nie był liniowo niezależny (np. gdy dla pewnego wyboru ε-otoczki punkt nie miał sąsiadów). W celu uniknięcia podobnych trudności stworzyłem graf pełny, w którym waga krawędzi była szybko malejącą funkcją od ich odległości. W wytrenowanym modelu przyjąłem funkcję $\exp(-x)$.

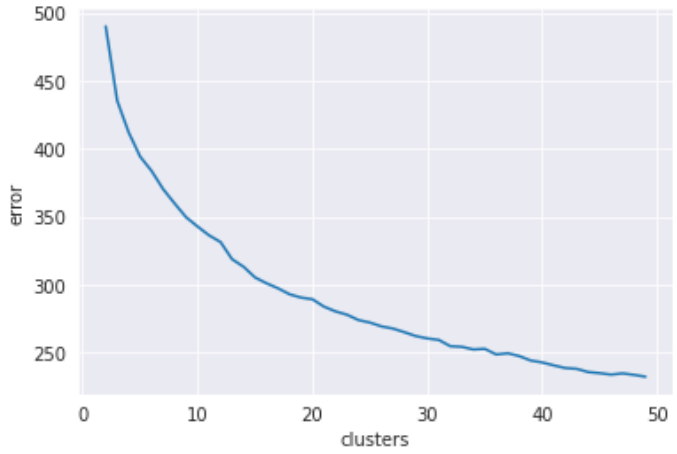
Poniżej prezentuje się tabelka wyników. W ostatnim kroku algorytmu ponownie użyliśmy metody `k-means` z 1000 iteracjami. Podany w tabelce wynik jest najlepszy spośród 10 wywołań modelu dla różnej wartości `seed`).

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.9268	0.8457	0.7625	0.5542	0.9872	0.9889	0.7955	0.7324


Dane_2D_1
Dane_2D_1
Dane_2D_2
Dane_2D_2
Dane_2D_3
Dane_2D_3
Dane_2D_4
Dane_2D_4
Dane_2D_5
Dane_2D_5
Dane_2D_6
Dane_2D_6
Dane_2D_7
Dane_2D_7
Dane_2D_8
Dane_2D_8
x1
x2

7. Wyniki dla datasetu data_18D

W tym zestawie danych nie mamy danego przyporządkowania dla klastrów. Jest to również zbiór wielowymiarowy. W celu wyznaczenia liczby klastrów dla danych wejściowych zastosujmy „*Metodę Łokcia*" dla modelu `K-Średnich`.
Poniżej prezentuje się wykres błędu (sumy odległości od centroidów) dla kolejnej liczby klastrów.



Z wykresu tego ciężko wywnioskować, jaka jest oczekiwana liczba klastrów. Być może w zależności od wymagań, w jakich będzie użyty model, akceptowalne będą różne odpowiedzi. Na przykład, chcąc jedynie określić jaki gatunek muzyki odpowiada konkretnemu klastrowi, weźmiemy `40` klastrów, a rozpoznając dzień tygodnia w zależności od zakupów klienta, wzięlibyśmy tylko kilka klastrów.

8. Wyniki dla datasetu rp.data

Ponownie wytrenowałem modele dla metod `K-means`, `Hierarchical Grouping`, `Spectral Clasterization`

Poniżej znajduje się tabelka z uzyskanymi wynikami dla poszczególnych metod. Z racji że dane są wielowymiarowe, nie jesteśmy niestety w stanie zwizualizować klastrów jak w 2 wymiarowym zbiorze punktów. W tabelce zawarłem również wyniki dla algorytmów wykorzystanych w `miniprojekcie 2`.
Z racji, że mamy jedynie 2 klasy, do której będziemy przydzielać dane, możemy obliczyć dokładność wpadania punktu do odpowiedniej klasy.
Wystarczy, że weźmiemy maksymalną z dokładności, które dostaniemy przyrównując otrzymane klastry z wartościami $\{0, 1\}$ z prawdziwej klasyfikacji.

K-Means	**H. Grouping + Ward**	**Spectral + K-Means**	**Naive Bayes**	**Log Reg.**
0.962	0.963	0.967	0.975	0.95

9. Implementacja

Implementację powyższych modeli oraz użyte w raporcie wykresy można znaleźć w repozytorium na githubie: https://github.com/Marwyk2003/mpum_miniproject4