Grupowanie (analiza skupień)

MLR wykłady 3 i 4, Tomasz Dzido

Na czym polega grupowanie?

- W przeciwieństwie do klasyfikacji i regresji jest ono techniką uczenia **nienadzorowanego.**
- Celem jest automatyczny podział obserwacji na grupy obiektów o podobnych cechach, czyli na klastry lub skupienia. Obserwacje są dzielone bez wcześniejszej wiedzy na temat tego jak docelowe grupy mają wyglądać.
- Nie ma zmiennej wyjściowej. Obserwacje w ramach tych samych klastrów powinny być maksymalnie do siebie podobne i maksymalnie niepodobne do pozostałych obserwacji.
- Często stosowane w modelach deskryptywnych (znajdowanie nieoczywistych grup w danych wejściowych). Stosowane oczywiście również w modelach predykcyjnych (ukryta zmienna czyli identyfikator klastra, do którego trafiła dana obserwacja, jest dodawana do danych i następnie używana do oceny nowych przypadków).

Zastosowania grupowania (przykłady)

- Problem badacza, który szuka ukrytych informacji w danych
- Problem marketingu, który chce poznać charakterystyki klientów
- Problem badacza, który chciałby przygotować dane do dalszej analizy - jako danych treningowych używać później tych, które należą do tych samych klastrów
- Problem lekarza, który chciałby pogrupować choroby lub ich objawy
- Problem lingwisty, który chciałby znaleźć podobne do siebie dokumenty
- Problem wykrywania anomalii – sprawdzenie czy dana obserwacja nie pasuje do żadnego ze znalezionych wcześniej klastrów, czyli jest obserwacją nietypową (anomalią)

Ogólny schemat grupowania

- 1. Wybór reprezentacji obiektów – grupowane nie są obiekty, ale ich abstrakcyjne reprezentacje (najczęściej zbiory cech).
- 2. Wybór miary podobieństwa obiektów – zdefiniowanie macierzy podobieństwa (niepodobieństwa) dla przyjętej reprezentacji obiektów – najczęściej podobieństwo mierzone jako odległość.
- 3. Grupowanie obiektów – najczęściej są używane algorytmy hierarchiczne i iteracyjne.
- 4. Wybór reprezentacji klastrów – zdefiniowanie klastrów jako zbiorów należących do nich obiektów, zbiorów punktów lub wzorców.

Miary podobieństwa obiektów

- Jeśli obiekty są opisane zbiorem zmiennych numerycznych, ich niepodobieństwo można zdefiniować jako *n*-wymiarową odległość Minkowskiego: $d_{Mi}(i,j) = (\sum_{n=1}^{k}(|x_i - x_j|)^q)^{1/q}$, gdzie *k* oznacza wartość *k*-tego atrybutu. Często stosuje się odległość euklidesową (*q=2*) lub Manhattan (*q=1*).
- W każdym przypadku zmienne, których liczba wartości jest znacznie większa od liczby wartości innych zmiennych, zdominują je, a więc przed grupowaniem zmienne numeryczne należy znormalizować.

Miara (nie)podobieństwa zmiennych binarnych

Tabela 7.1. Macierz niepodobieństwa zmiennych binarnych

Obiekt X_i \ Obiekt X_j	1	0	Suma
1	q	r	$q + r$
0	s	t	$s + t$
Suma	$q + s$	$r + t$	$p = q + r + s + t$

Oznaczenia: q – liczba zmiennych przyjmujących wartość 1 dla obu obiektów, r – liczba zmiennych przyjmujących wartość 1 dla obiektu x_i i 0 dla x_j, s – liczba zmiennych przyjmujących wartość 0 dla obiektu x_i i 1 dla x_j, t – liczba zmiennych przyjmujących wartość 0 dla obu obiektów, p – podobieństwo obiektów $q + r + s + t$.

Źródło: M. Szeliga: Data Science i Uczenie Maszynowe, PWN 2017

Zmienne binarne – ciąg dalszy

- Zmienne binarne mogą być symetryczne lub asymetryczne.
- Symetryczna jest wtedy gdy obie wartości tej zmiennej mają tę samą wagę (np. płeć). Niepodobieństwo między obiektami i oraz j:

$$d(i,j) = \frac{r+s}{q+r+s+t}.$$

- Asymetryczna wtedy gdy obie wartości mają różną wagę (np. CzyOszustwo), dla niej niepodobieństwo liczymy ze wzoru Jacarda:

$$d(i,j) = \frac{r+s}{q+r+s}.$$

Zmienne binarne - przykład

Imię	Płeć	Gorączka	Kaszel	Test 1	Test 2	Test 3	Test 4
Jacek	M	T	N	P	N	N	N
Maria	K	T	N	P	N	P	N
Jan	M	T	T	N	N	N	N

- Płeć jest atrybutem symetrycznym,
- Pozostałe atrybuty są binarne ale niesymetryczne,
- Dla wygody wartości T,P ustalamy na 1, wartość N zaś na 0.

Korzystamy ze wzoru Jacarda, pomijając atrybut symetryczny:

$d(Jacek, Maria) = \frac{0+1}{2+0+1} = 0.33$, $d(Jacek, Jan) = \frac{1+1}{1+1+1} = 0.67$, $d(Jan, Maria) = \frac{1+2}{1+1+2} = 0.75$.

Wartości dla Jana i Marii są najbardziej oddalone – największa różność.

Co ze zmiennymi innych typów?

- Zmienne kategoryczne traktuje się albo jako numeryczne, albo jako uogólnienie zmiennych binarnych.
- Jeśli zmienna kategoryczna jest typu porządkowego ale nieznana jest odległość między stanami, to przekształca się ją na zmienną numeryczną o zakresie [0,1] – ponumerowane stany dzieli się przez liczbę wszystkich stanów minus 1.
- Niepodobieństwo zmiennych nominalnych (kategorycznych, wiele stanów ale bez porządku) jest definiowane jako uogólnienie macierzy niepodobieństwa zmiennych binarnych (podobne do siebie są wyłącznie identyczne stany zmiennej) i liczymy je ze wzoru:

  $d(i,j) = \frac{p-m}{p}$, gdzie *p* oznacza łączną liczbę zmiennych, a *m* -liczbę zmiennych, których wartość jest identyczna dla obu obiektów.
- Zatem takie zmienne niosą ze sobą mniej przydatnych dla grupowania informacji niż zmienne numeryczne.

Zmienne różnych rodzajów

- Najczęstsza sytuacja – zmienne różnych rodzajów.
- Ich niepodobieństwo oblicza się najczęściej jako średnią ważoną poszczególnych miar niepodobieństw.
- Co ciekawe metoda ta faworyzuje zmienne kategoryczne i binarne względem zmiennych numerycznych o dużym zakresie wartości. Należy więc znormalizować (z reguły stosuje się standaryzację) zmienne liczbowe.
- Dlaczego standaryzuje się dane liczbowe?

Przykład:

x = (0.1; 20), y = (0.9; 720) to odległość euklidesowa: $\sqrt{(0.9-0.1)^2+(720-20)^2} = 700$ jest zdominowana przez wartość drugiego atrybutu; po zestandaryzowaniu do przedziału (0; 1) wartości drugiego atrybutu zostaną sprowadzone do, odpowiednio 0.02 i 0.72 oraz odległość wyniesie 1.063 (są różne metody standaryzacji, wykorzystujemy minimum i maximum, średnie odchylenie i inne).

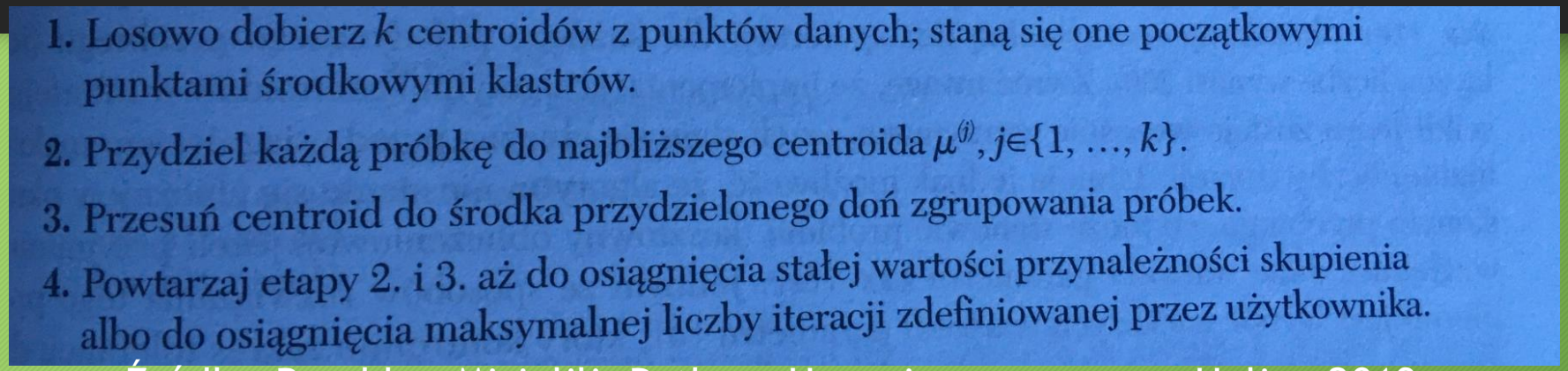
Podstawowe metody grupowania

- 1. Bazujące na prototypach (ang. prototype-based clustering) – każda grupa jest reprezentowana przez prototyp stanowiący albo **centroid** (wartość średnią) podobnych punktów o cechach ciągłych albo **medoid** (najbardziej reprezentatywny lub najczęściej występujący punkt) w przypadku cech kategoryzujących.
- 2. Metody hierarchiczne: tworzona jest struktura poprzez rekurencyjne dzielenie lub łączenie istniejących grup.
- 3. Metody oparte o gęstość.

Algorytm centroidów (k-średnich, ang. k-means)

- Łatwy w implementacji, skuteczny obliczeniowo, bardzo popularny. Należy do kategorii grupowania bazującego na prototypach.
- Świetnie radzi sobie z wykrywaniem grup o kulistym kształcie, wadą jego jest konieczność odgórnego zdefiniowania liczby klastrów *k*. Niewłaściwy jej dobór to niska skuteczność algorytmu (są metody oceny jakości analizy skupień, które ułatwiają określenie optymalnej liczby *k*). Nie mamy żadnych pewnych informacji kategoryzujących, próbki musimy pogrupować na podstawie podobieństw definiowanych przez cechy.

Algorytm centroidów - etapy



1. Losowo dobierz k centroidów z punktów danych; staną się one początkowymi punktami środkowymi klastrów.
2. Przydziel każdą próbkę do najbliższego centroida $\mu^{(j)}$, $j \in \{1, \ldots, k\}$.
3. Przesuń centroid do środka przydzielonego doń zgrupowania próbek.
4. Powtarzaj etapy 2. i 3. aż do osiągnięcia stałej wartości przynależności skupienia albo do osiągnięcia maksymalnej liczby iteracji zdefiniowanej przez użytkownika.

Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

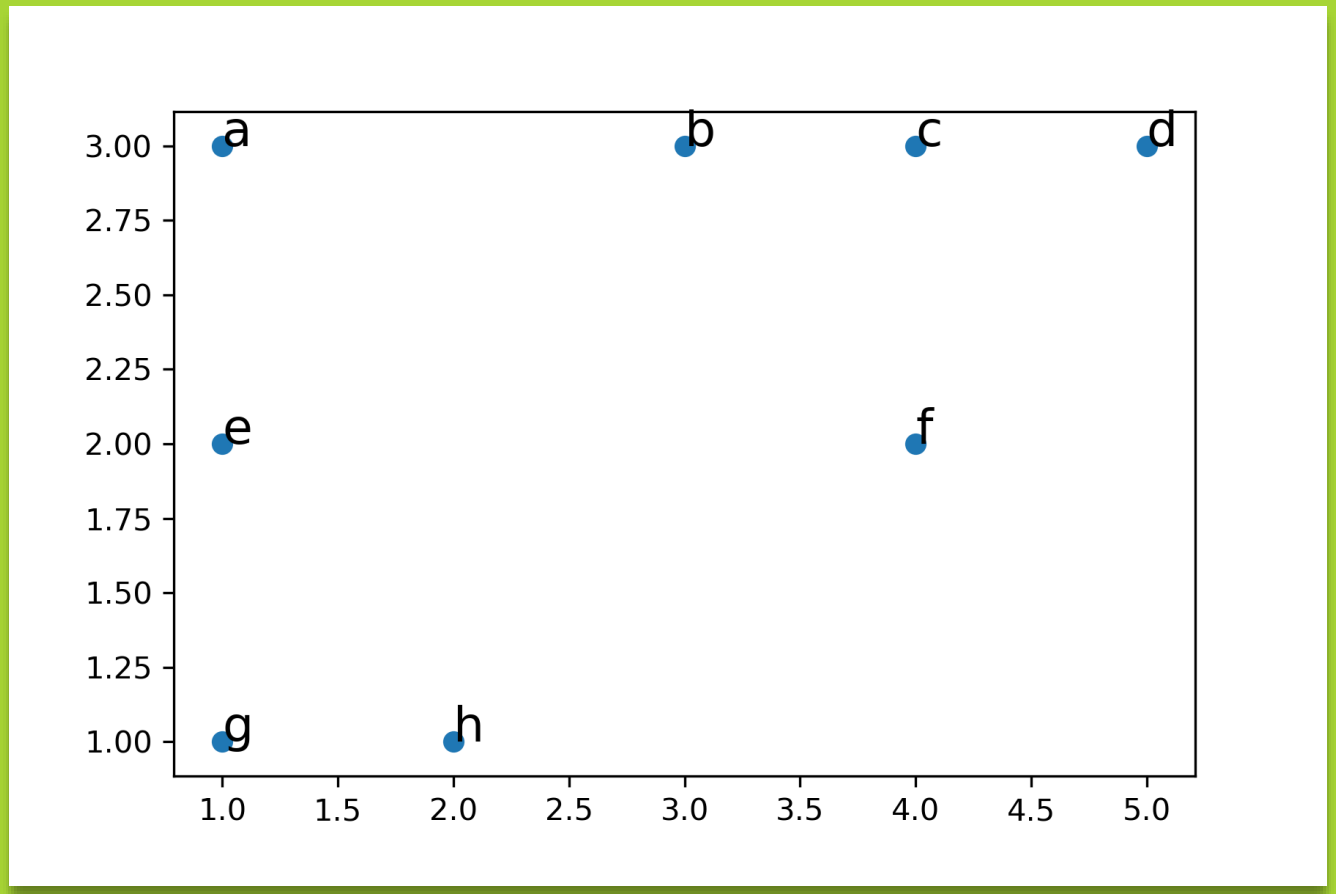
W jaki sposób mierzymy podobieństwo pomiędzy obiektami? Często stosowaną metryką odległości wobec zgrupowanych próbek o cechach ciągłych jest kwadrat odległości euklidesowej pomiędzy dwoma punktami *x* i *y* w *m*-wymiarowej przestrzeni. Na podstawie tej metryki możemy opisać algorytm centroidów jako proste zagadnienie optymalizacji, przyrostową metodę minimalizacji wewnątrzgrupowej sumy kwadratów błędów (ang. within-cluster sum of squared errors), zwaną także bezwładnością klastra (ang. cluster interia):

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} ||x^{(i)} - \mu^{(j)}||^2,$$

gdzie $\mu^{(j)}$ to centroid klastra *j*, a $w^{(i,j)}$=1 jeśli próbka $x^{(i)}$ znajduje się w skupieniu *j*, w przeciwnym przypadku $w^{(i,j)}$=0.

Alg. centroidów – przykład rachunkowy

- Załóżmy, że mamy 8 punktów. Ustalmy na początek:
- k=2
- $\mu^{(1)} = (1,1)$ czyli punkt g
- $\mu^{(2)} = (2,1)$ czyli punkt h
- Należy obliczyć odpowiednie odległości, następnie SSE i zmodyfikować centroidy.
- Kolejne obliczenia są na kolejnych slajdach.



Przykład - rachunki

Punkt	Odległość od $\mu^{(1)}$	Odległość od $\mu^{(2)}$	Klaster
a	2.00	2.24	C_1
b	2.83	2.24	C_2
c	3.61	2.83	C_2
d	4.47	3.61	C_2
e	1.00	1.41	C_1
f	3.16	2.24	C_2
g	0.00	1.00	C_1
h	1.00	0.00	C_2

SSE= $2^2 + 2.24^2 + 2.83^2 + 3.61^2 + 1^2 + 2.24^2 + 0^2 + 0^2 = 36.07$,
Nowe centroidy: $\mu^{(1)} = (1, 2)$ i $\mu^{(2)} = (3.6, 2.4)$.

Przykład – rachunki c.d.

Punkt	Odległość od $\mu^{(1)}$	Odległość od $\mu^{(2)}$	Klaster
a	1.00	2.67	C_1
b	2.24	0.85	C_2
c	3.16	0.72	C_2
d	4.12	1.52	C_2
e	0.00	2.63	C_1
f	3.00	0.57	C_2
g	1.00	2.95	C_1
h	1.41	2.13	C_1

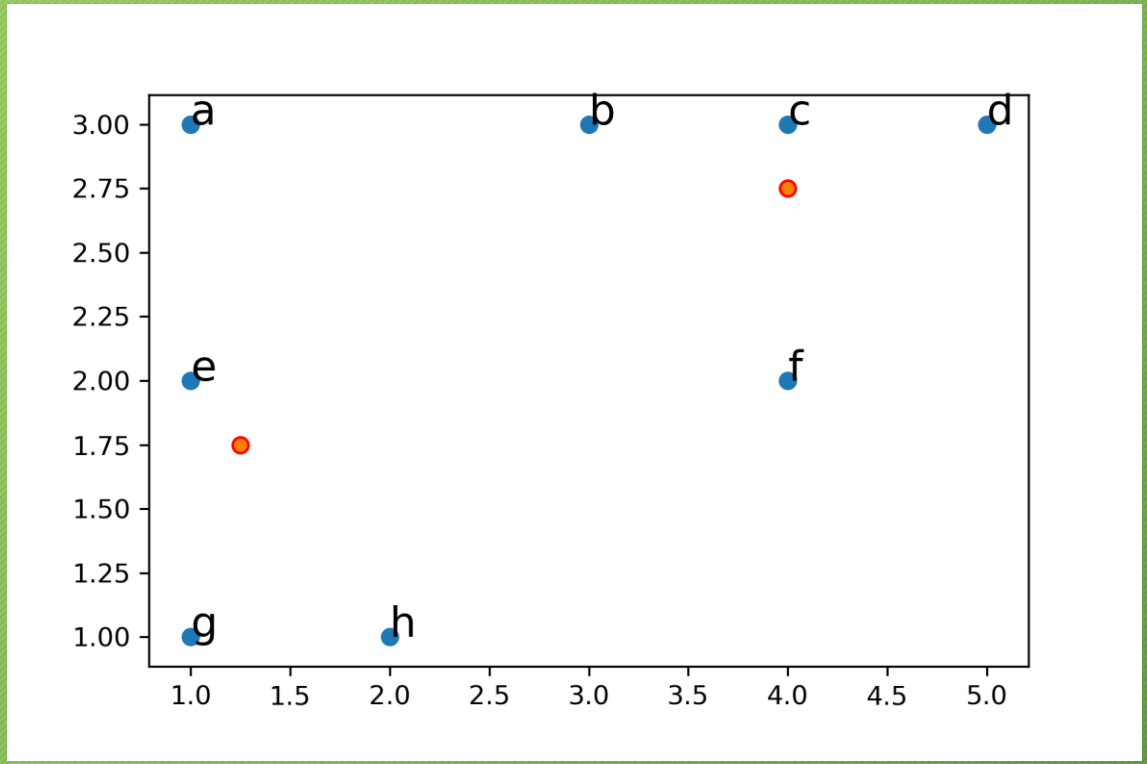
SSE= 7.876,
Nowe centroidy: $\mu^{(1)} = (1.25, 1.75)$ i $\mu^{(2)} = (4.0, 2.75)$.

Przykład – rachunki c.d.

Punkt	Odległość od $\mu^{(1)}$	Odległość od $\mu^{(2)}$	Klaster
a	1.27	3.01	C_1
b	2.15	1.03	C_2
c	3.02	0.25	C_2
d	3.95	1.03	C_2
e	0.35	3.09	C_1
f	2.75	0.75	C_2
g	0.79	3.47	C_1
h	1.06	2.66	C_1

SSE= 6.23,
Nowe centroidy: $\mu^{(1)} = (1.25, 1.75)$ i $\mu^{(2)} = (4.0, 2.75)$.
Koniec algorytmu.

Przykład – wynik (centroidy)



Algorytm centroidów w scikit-learn

- Wykorzystamy zbiór danych *make_blobs* z *sklearn.datasets*, a następnie zaimplementujemy klasę *KMeans*, która stanowi część modułu *cluster*.
- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html
- Ciekawe parametry (klasy *KMeans*):
- n_init = 10 – uruchamiamy 10 niezależnych algorytmów skupień zawierających różne losowe centroidy (wybrany zostanie model z najmniejszym SSE)
- max_iter = 300 – łączna maksymalna liczba iteracji, jeśli implementacja staje się zbieżna przed osiągnięciem tej liczby, to zostaje przerwana
- tol = 1e-04 (0.0001) – zakres tolerancji dla zmian wewnątrzgrupowej SSE, gdy weźmiemy większy to zabezpieczamy się przed kosztownym obliczeniowo problemem gdy za duże max_iter i algorytm nie staje się zbieżny.

Algorytm centroidów w scikit-learn

- Podczas korzystania z implementacji algorytmu centroidów w bibliotece *scikit-learn* jest możliwość wyznaczenia pustych skupień. W przypadku pustego skupienia algorytm będzie szukał przykładu znajdującego się najdalej od centroidu pustego klastra. Następnie przekształci ten centroid w najdalszy punkt.
- Trzeba upewnić się, że wszystkie cechy są mierzone w tej samej skali (zrobić standaryzację lub normalizację min.-max.)
- Wartości centroidów są przechowywane w atrybucie *centers_* obiektu *KMeans*.
- Analiza przykładu: Centroidy.py

Algorytm k-means++ (scikit-learn)

- Za pomocą tego algorytmu rozmieszczamy początkowe centroidy jak najdalej od siebie, co zapewnia lepsze i spójniejsze wyniki niż klasyczny algorytm k-średnich (D. Arthur, S. Vassilvitskii 2007).
- W programie należy pole *init* ustawić na *k-means++* zamiast *random*.

Inicjację algorytmu k-means++ możemy opisać następująco:

1. Stwórz pusty zbiór $\boldsymbol{M}$ przechowujący k wybranych centroidów.
2. Losowo dobierz pierwszy centroid $\boldsymbol{\mu}^{(j)}$ z dostępnych próbek i umieść go w zbiorze $\boldsymbol{M}$.
3. Dla każdej próbki $\boldsymbol{x}^{(i)}$ niebędącej częścią zbioru $\boldsymbol{M}$ znajdź minimalny kwadrat odległości $d\left(\boldsymbol{x}^{(i)}, \boldsymbol{M}\right)^2$ od każdego centroidu umieszczonego w zestawie danych $\boldsymbol{M}$.
4. W celu losowego doboru następnego centroidu $\boldsymbol{\mu}^{(p)}$ użyj ważonego rozkładu prawdopodobieństwa: $\frac{d\left(\boldsymbol{\mu}^{(p)}, \boldsymbol{M}\right)^2}{\sum_i d\left(\boldsymbol{x}^{(i)}, \boldsymbol{M}\right)^2}$.
5. Powtarzaj etapy 2. i 3. aż do znalezienia k centroidów.
6. Wykonaj dalsze obliczenia za pomocą klasycznego algorytmu centroidów.

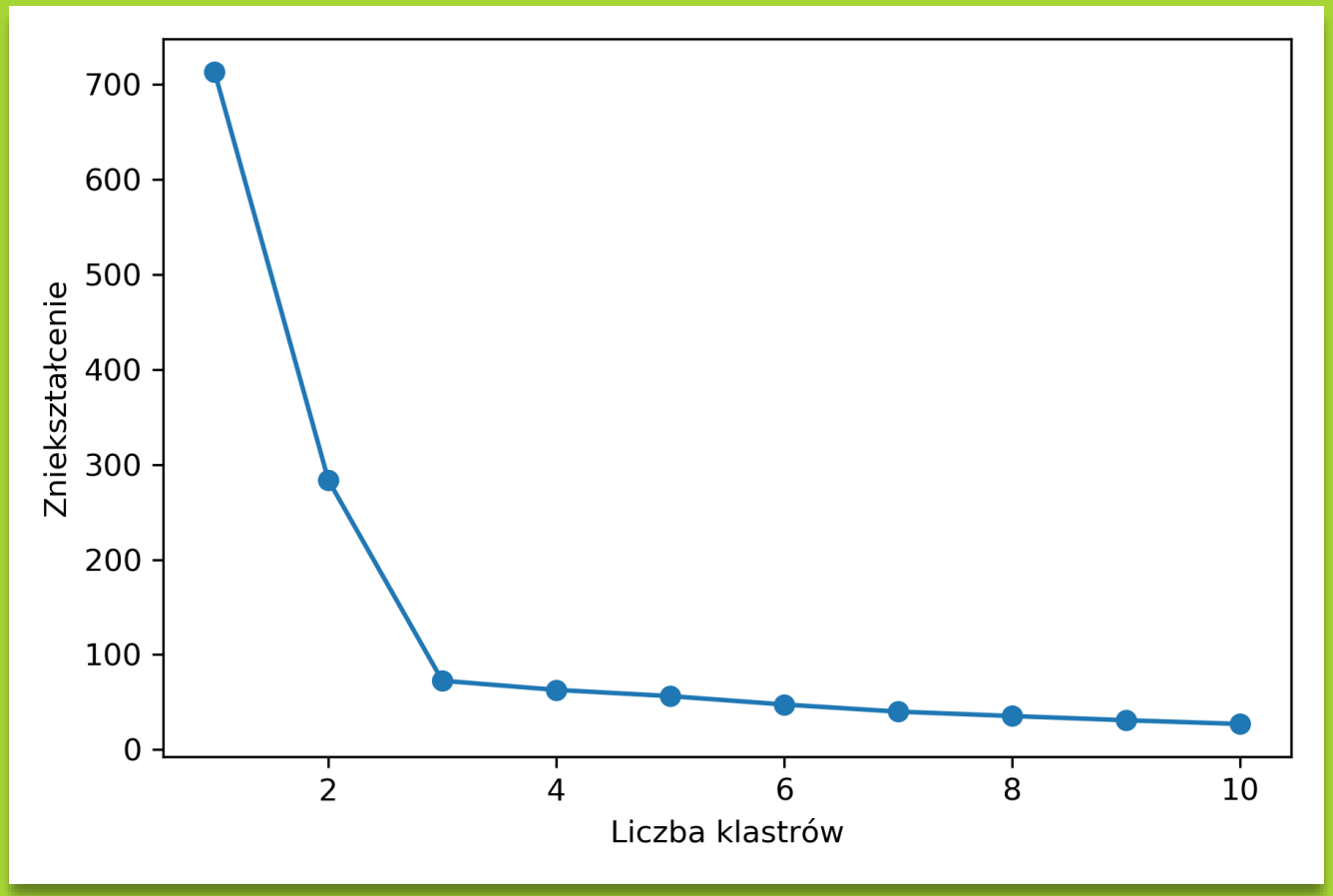
Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

Stosowanie metody łokcia do wyszukiwania optymalnej liczby skupień

- W uczeniu nienadzorowanym nie znamy ostatecznej odpowiedzi, nie możemy korzystać z metod oceny skuteczności charakterystycznych dla modeli nadzorowanych. W związku z tym korzystamy z wewnętrznych metryk jak SSE (atrybut *inertia_*) – przykład: Centroidy.py.
- **Metoda łokcia** (ang. elbow method) to narzędzie graficzne, które korzysta z oczywistego faktu, że jeśli liczba skupień *k* rośnie, to maleje zniekształcenie (próbki będą bliżej centroidów, do których zostały przydzielone). Celem metody łokcia jest określenie wartości *k*, dla której zniekształcenie zaczyna najszybciej wzrastać (rysuje się wykres zniekształcenia dla różnych *k*).

Metoda łokcia – c.d.

- Wyraźnie widać jaka będzie optymalna wartość liczby skupień.
- Inną metodą jest **analiza profilu**. Często przyjmuje ona również postać narzędzia graficznego, które generuje wykres pokazujący stopień upakowania próbek w danym skupieniu (to ważne – rozpatrzmy k = 4 w powyższym przykładzie).



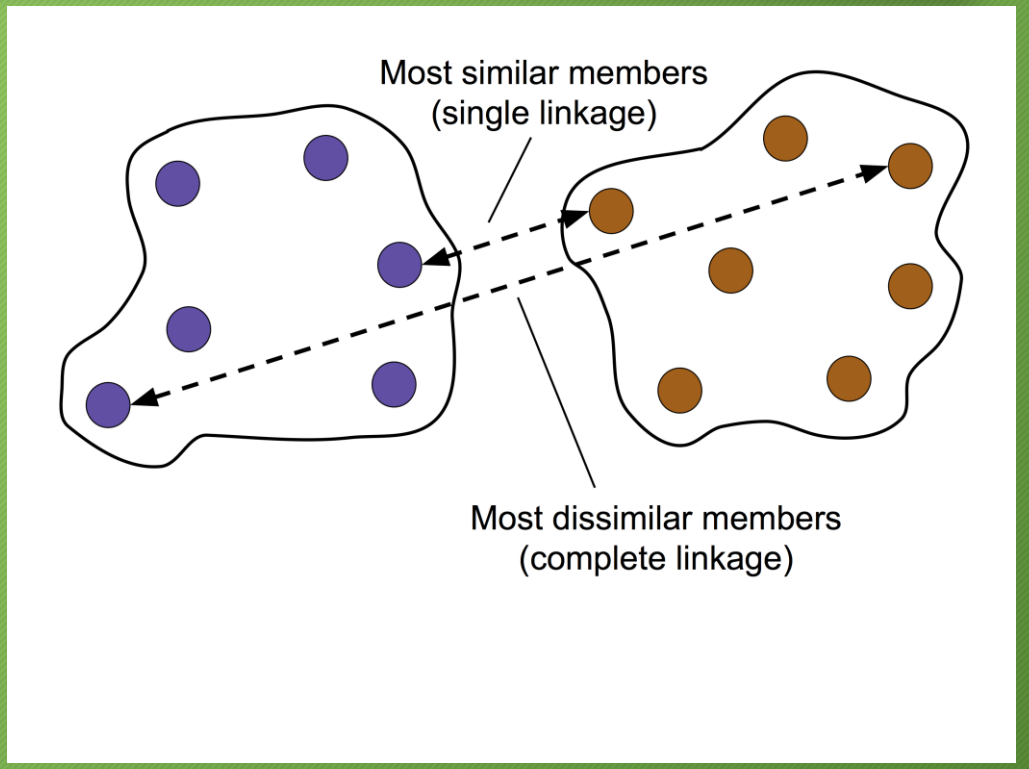
Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

Klasteryzacja hierarchiczna

- Zalety: możliwość generowania **dendrogramów** (graficznego przedstawienia binarnej, hierarchicznej analizy skupień) oraz brak konieczności definiowania liczby skupień.
- 2 główne techniki: metody **aglomeracyjne** (tą przedstawimy) i **deglomeracyjne**. W tej drugiej zaczynamy od jednego klastra zawierającego w sobie wszystkie pozostałe skupienia i stopniowo dzielimy go na mniejsze grupy, aż do osiągnięcia stanu, w którym każdy klaster będzie zawierał tylko jedną próbkę. Metoda aglomeracyjna jest przeciwieństwem (każda próbka stanowi osobne skupienie i dążymy do łączenia par najbliższych elementów, aż do uzyskania jednego wielkiego klastra.
- W aglomeracyjnej analizie skupień stosowane są 2 standardowe algorytmy: **pojedynczego wiązania** (najbliższego sąsiedztwa) oraz **pełnego wiązania** (najdalszego wiązania).

Grupowanie skupień

Za pomocą algorytmu pojedynczego wiązania wyszukujemy 2 punkty we wszystkich parach skupień (po jednym punkcie na każdy klaster) cechujących się najmniejszą odległością pomiędzy klastrami. W algorytmie pełnego wiązania łączymy klastry, dla których została wyliczona największa odległość. Istnieją inne metody np. Warda gdzie łączone są te skupienia, dla których następuje najmniejszy wzrost wewnątrz-grupowej sumy kwadratów błędów.



Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

Klasteryzacja aglomeracyjna przy użyciu algorytmu pełnego wiązania.

- Ten hierarchiczny algorytm to następująca procedura przyrostowa:

1. Oblicz macierz odległości dla wszystkich próbek.
2. Stwórz reprezentację każdego elementu jako oddzielnego klastra.
3. Połącz dwa najbliższe klastry na podstawie odległości pomiędzy dwoma najbardziej różniącymi się (oddalonymi) elementami.
4. Zaktualizuj macierz podobieństwa.
5. Powtarzaj kroki 2. – 4., dopóki nie pozostanie tylko jedno skupienie.

Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

Przykład (w Pythonie)

- W celu wyliczenia macierzy odległości wykorzystamy funkcję *pdist* będącą składową podmodułu *spatial.distance* z biblioteki *SciPy*, której wynik działania stanowić będzie wejście funkcji *squareform*. Program: Hierarchiczny.py
- Następnie należy przeprowadzić aglomerację klastrów metodą pełnego wiązania za pomocą funkcji *linkage* stanowiącej część podmodułu *cluster.hierarchy* (biblioteka *SciPy*). Zwraca ona tak zwaną **macierz wiązania** (ang. linkage matrix).

```
from scipy.cluster.hierarchy import linkage
help(linkage)
```

Przykład – c.d.

- Możemy wykorzystać zwartą macierz odległości (trójkątną górną) wygenerowaną przez pdist lub początkową tablicę. Nie można jednak wykorzystać kwadratowej macierzy odległości (błędne wyniki by były). Stosujemy metrykę „euclidean” – 3 scenariusze – patrz program.
- Otrzymujemy macierz wiązania – nie jest łatwa jej analiza.

Przykład – c.d. (macierz wiązania)

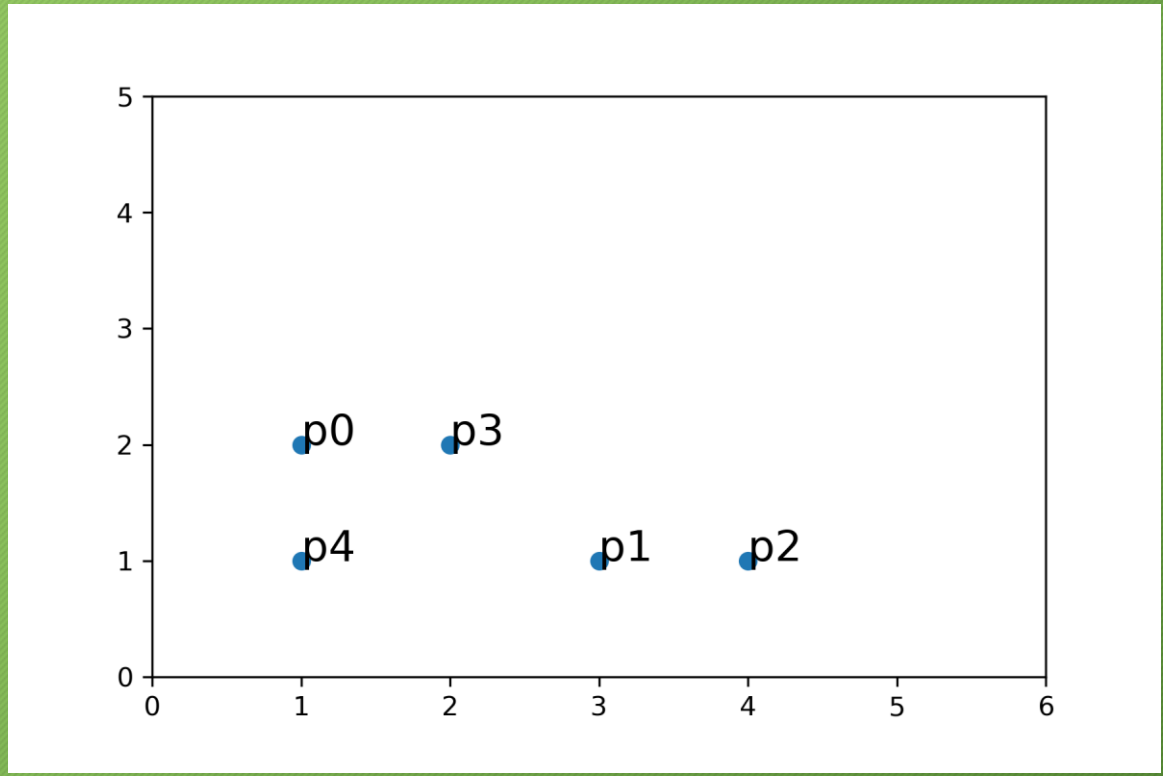
- Macierz wiązania dla przykładu z programu: (Jupiter Notebook)

	Etykieta rzędu 1	Etykieta rzędu 2	Odległość	Liczba elementów klastra
Klaster 1.	0.0	4.0	3.835396	2.0
Klaster 2.	1.0	2.0	4.347073	2.0
Klaster 3.	3.0	5.0	5.899885	3.0
Klaster 4.	6.0	7.0	8.316594	5.0

Zanim przeanalizujemy powyższy przykład, rozpatrzmy prostszy jak na kolejnym slajdzie:

Macierz wiązania – prostszy przykład rachunkowy

- Załóżmy, że mamy 5 punktów. Na początek potrzebujemy macierz odległości. Jej macierz trójkątną górną możemy zapisać w postaci wektora:
- [2.236 3.162 1.0 1.0 1.0 1.414 2.0 2.236 3.0 1.414]
- Łatwo widać, że najpierw punkty p0 i p3 trafią do pierwszego klastra, a następnie punkty p1 i p2 do drugiego …
- Następnie punkt p4 trafi tam gdzie p0 i p3 (odl. p3 i p4), a na koniec oba nowe klastry utworzą jeden wielki 5-elementowy.



Przykład rachunkowy – c.d.

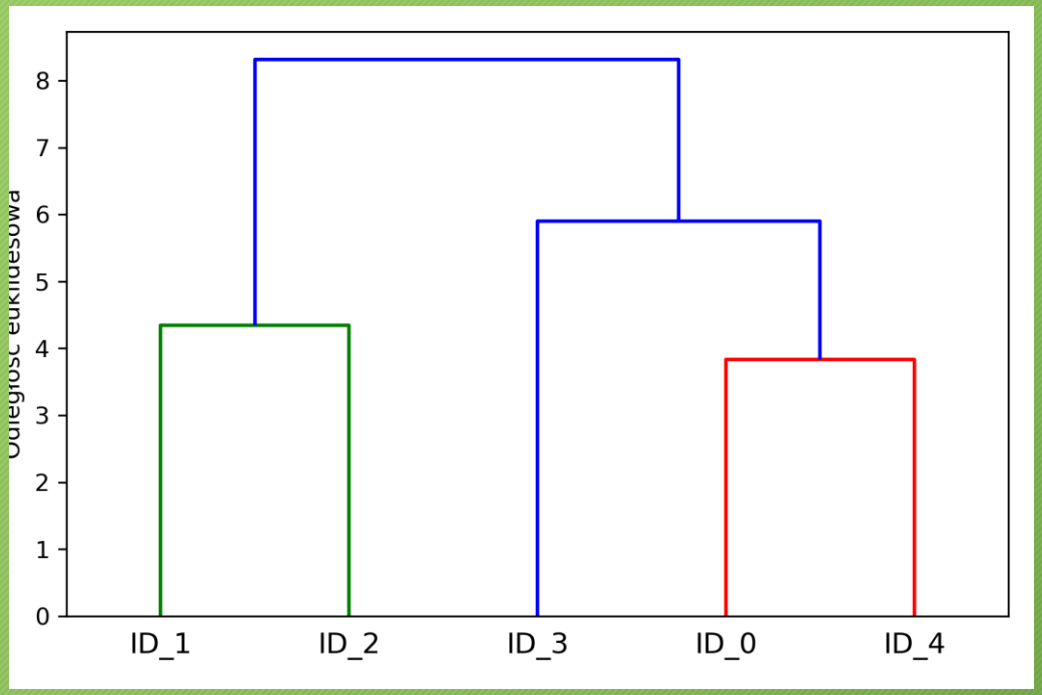
- Ostatecznie wynikowa macierz wiązania przyjmie postać:

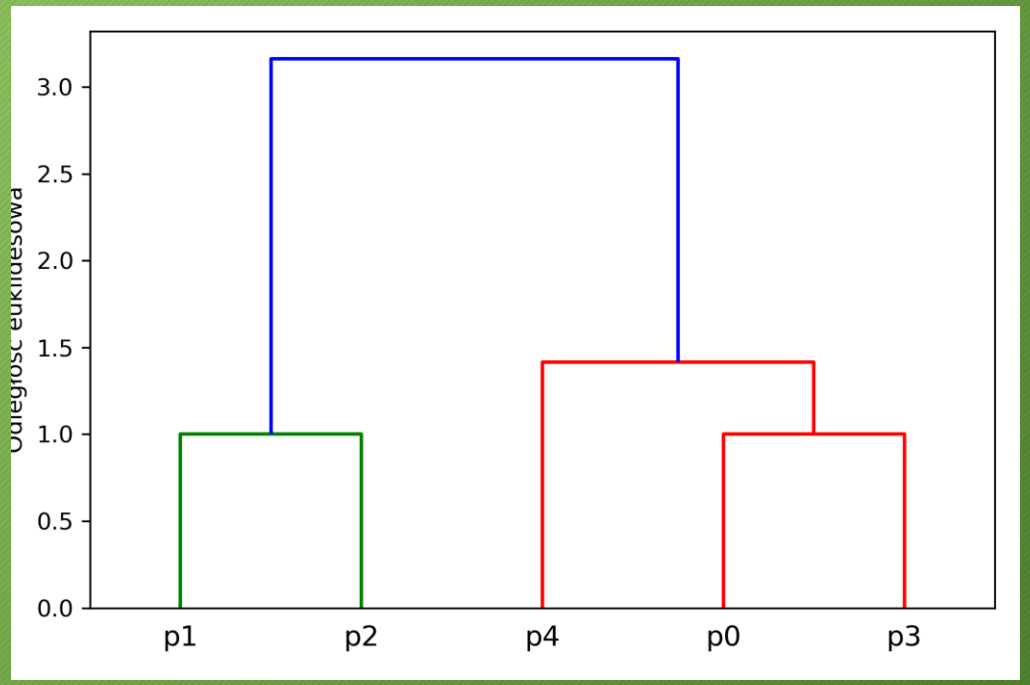
```
[[0.         3.         1.                 2.        ]
 [1.         2.         1.                 2.        ]
 [4.         5.         1.41421356         3.        ]
 [6.         7.         3.16227766         5.        ]]
```

- Zwróćmy uwagę, że zawsze braliśmy najmniejszą z największych odległości między klastrami.

Dendrogram

- Wykorzystamy funkcję *dendrogram* ze *scipy.cluster.hierarchy*. (zwróćmy uwagę na kolorystykę obu wykresów).





Dobór liczby zwracanych skupień w aglomeracyjnej klasteryzacji

- Wykorzystamy *AgglomerativeClustering* z *scikit-learn*. Przydaje się w sytuacji gdy chcemy przyciąć drzewo klasteryzacji.
- Ponownie metryka euklidesowa, wiązanie pełne ale dodatkowo parametr *n_cluster*=3 (lub 2). Bardzo proste stosowanie:

```
ac = AgglomerativeClustering(n_clusters=3,
metric='euclidean', linkage='complete’)
labels = ac.fit_predict(X)
print('Etykiety skupień: %s' % labels)
```

- Analiza wyników w obu przypadkach ([1 0 0 2 1] oraz [0 1 1 0 0])

Gęstościowa klasteryzacja przestrzenna z uwzględnieniem szumu DBSCAN (ang. density-based spatial clustering of applicaions with noise)

- Nie występują tu założenia dotyczące kulistych skupień (w przeciwieństwie do algorytmu centroidów), a zestawy danych nie są dzielone na hierarchie wymagające ręcznego wyznaczenia punktu odcięcia. Analiza skupień wyznacza tutaj etykiety klastrów na podstawie obszarów zagęszczenia punktów.
- Gęstość w algorytmie DBSCAN jest zdefiniowana jako liczba punktów w określonym promieniu ε.
- W tej technice każda próbka otrzymuje specjalną etykietę wyznaczaną na podstawie następujących kryteriów:

DBSCAN - działanie

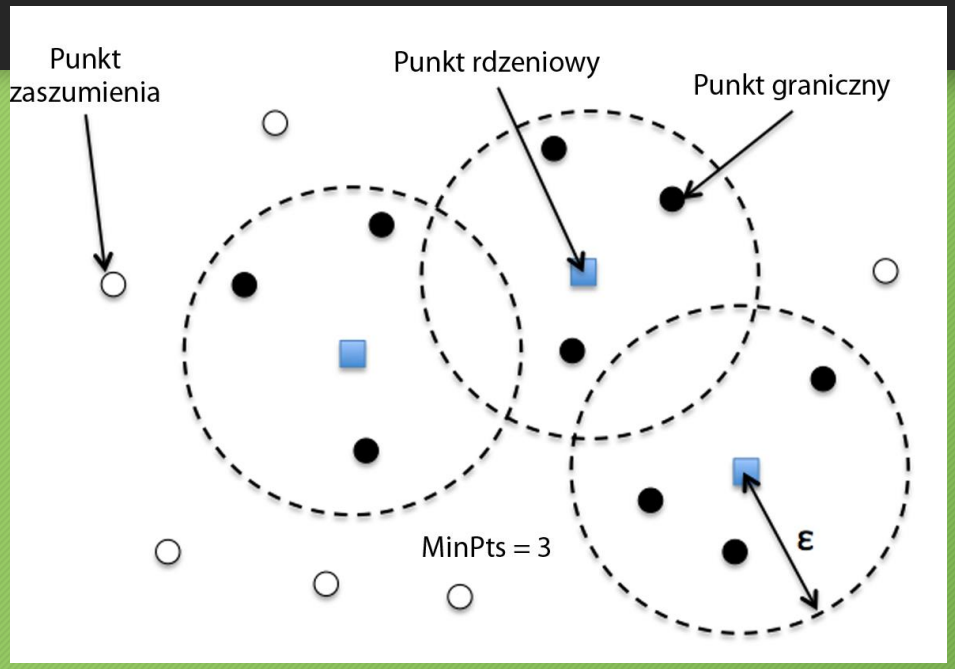
- punkt jest uznawany za **rdzeniowy** (ang. *core point*), jeżeli w określonym od niego promieniu występuje co najmniej minimalna liczba (MinPts) sąsiadujących punktów w określonym promieniu ε;
- punkt **graniczny** (ang. *border point*) ma w promieniu ε mniej sąsiadujących punktów od wartości minimalnej, ale znajduje się w promieniu ε punktu rdzeniowego;
- wszystkie pozostałe punkty niebędące rdzeniowymi ani granicznymi są uznawane za punkty **zaszumienia** (ang. *noise point*).

Po wyznaczeniu powyższych punktów DBSCAN przeprowadza następujące 2 czynności:

1. Tworzy oddzielne skupienie dla każdego punktu rdzeniowego lub połączonej grupy punktów rdzeniowych (punkty rdzeniowe są ze sobą połączone, jeżeli odległość pomiędzy nimi nie przekracza długości promienia ε).
2. Przydziela każdy punkt graniczny do klastra zawierającego odpowiedni punkt rdzeniowy.

Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

DBSCAN – ilustracja działania

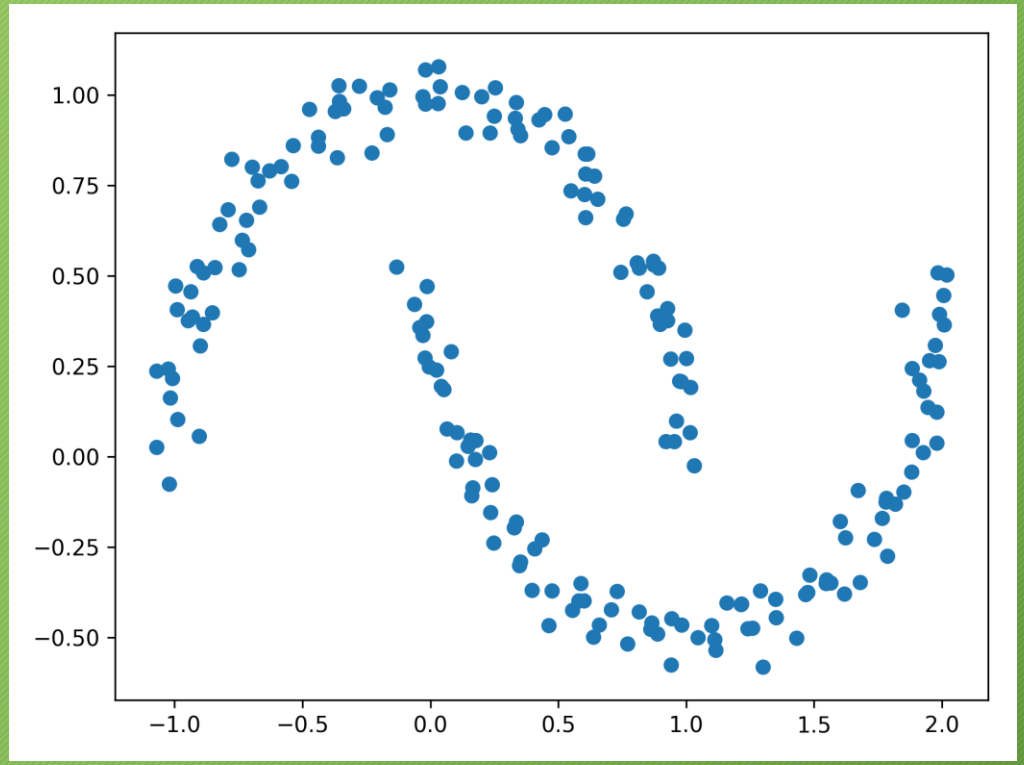


Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

Algorytm DBSCAN odróżnia się od algorytmów centroidów i hierarchicznych tym, że brak jest w nim konieczności przydzielania wszystkich punktów do skupień. Ponadto daje możliwość usuwania szumów. Można tak dobrać przykład działania by uchwycić jego zalety.

Przykład działania algorytmu DBSCAN

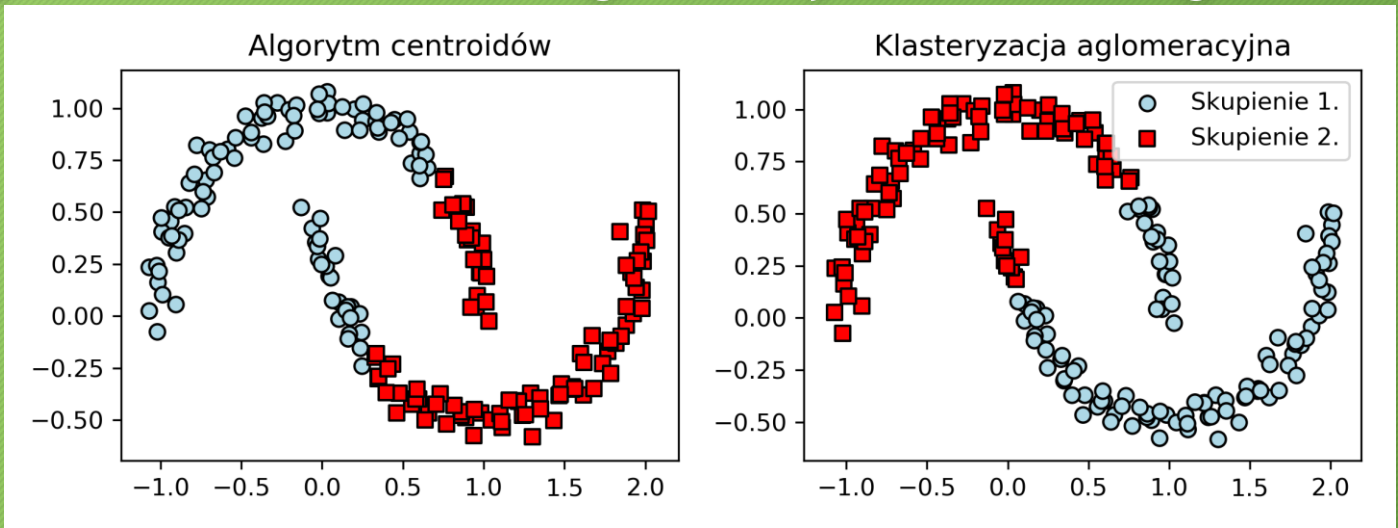
- Wykorzystamy *make_moons* z *sklearn.datasets* (program DBSCAN.py).



Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

DBSCAN – przykład c.d.

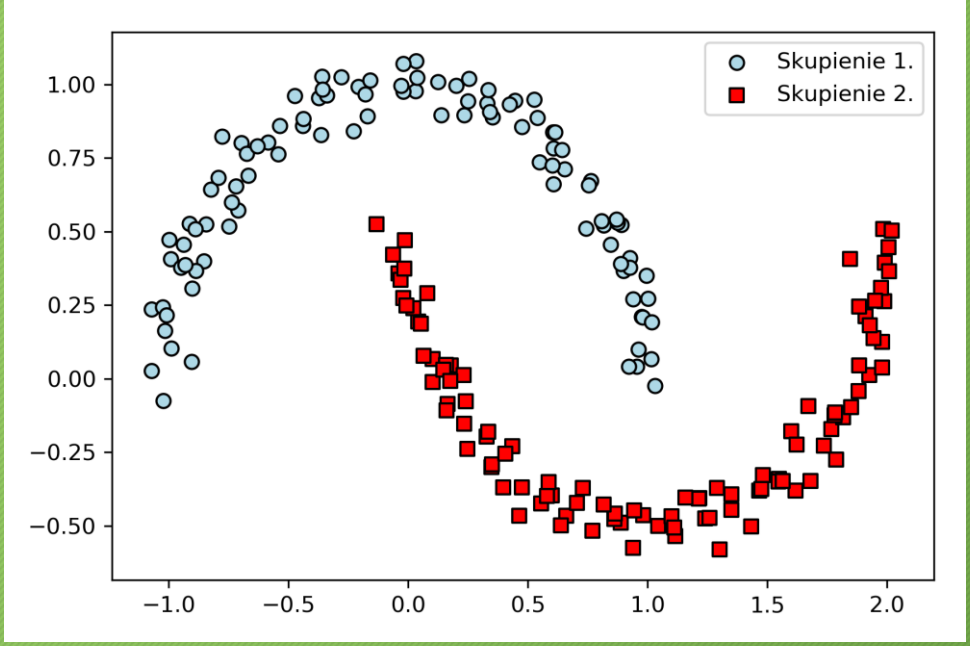
- Przebadać parametry *noise=0.05* i *n_samples=200*.
- Dla powyższych danych tworzących półksiężycowate struktury zastosujemy najpierw algorytmy *Kmeans* i *AgglomerativeClustering*. Co wyszło? Dlaczego?



Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

DBSCAN – przykład c.d.

- Teraz czas na wykorzystanie *DBSCAN* z *sklearn.cluster*. Ważne parametry: *eps=0.2* i *min_samples=5* (program DBSCAN.py).



Źródło: Raschka, Mirjalili: Python. Uczenie maszynowe, Helion 2019

Wady algorytmu DBSCAN

- Wraz ze wzrostem liczby cech w zestawie danych (przy tej samej liczbie przykładów uczących) wzmocnieniu ulega negatywny wpływ **klątwy wielowymiarowości** (ang. curse of dimensionality). Szczególnie gdy stosujemy metrykę euklidesową. Oznacza to, że przestrzeń cech staje się coraz bardziej rozległa (w wielowymiarowej przestrzeni nawet najbliżsi sąsiedzi znajdują się zbyt daleko, aby uzyskać za ich pomocą dobre oszacowanie). Zjawisko to dotyka oczywiście pozostałych metod klasteryzacji.
- Znalezienie dobrej kombinacji parametrów (MinPts i ε) może być kłopotliwe w sytuacji gdy różnice w gęstości zestawu danych są względnie duże.

Który z algorytmów klasteryzacji wybrać?

- W praktyce dobór trudny, szczególnie gdy dane są wielowymiarowe, co utrudnia lub uniemożliwia ich wizualizację.
- Skuteczna klasteryzacja nie zależy wyłącznie od algorytmu lub jego hiperparametrów. Liczy się również dobór właściwej metryki odległości, a także duża wiedza z analizowanej dziedziny pozwalająca na skuteczniejsze eksperymentowanie z konfiguracją modelu.
- Przedstawione na wykładzie metody klasteryzacji nie wyczerpują tematu np. jest grafowa analiza skupień (i należąca do niej klasteryzacja spektralna – stosuje się wektory własne macierzy podobieństwa lub odległości do wykrywania współzależności w klastrach).

https://blog.etrapez.pl/macierze/wartosci-i-wektory-wlasne-macierzy/

- W kontekście klątwy wielowymiarowości przeprowadza się redukcję wielowymiarowości. Powszechne jest także kompresowanie zestawu danych do podprzestrzeni dwuwymiarowej, co pozwala na wizualizację klastrów.
- Różne są kryteria oceny skuteczności algorytmów klasteryzacji, ale o tym w innym momencie.