

# Praca domowa 4, Wstęp do uczenia maszynowego

## Jan Pogłód

### 1. Wstęp

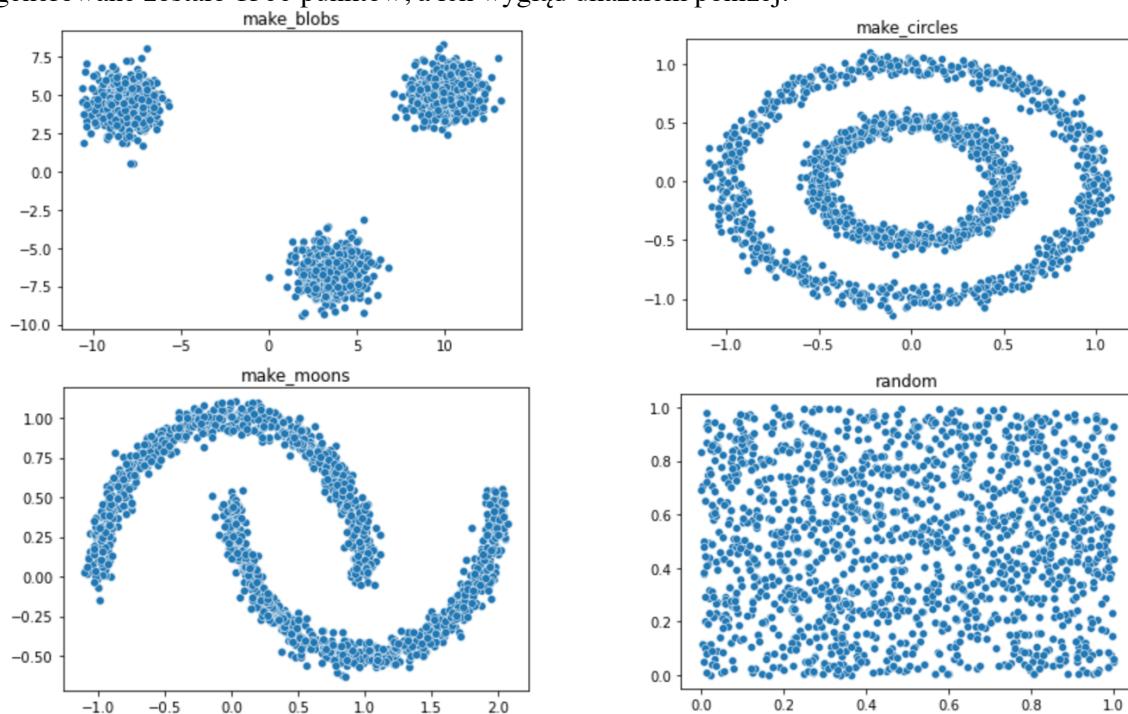
Celem pracy domowej było zbadanie jakości klastrowania danych na trzech różnych modelach uczenia maszynowego nienadzorowanego. Podczas eksperymentu wykorzystane zostały modele:

- KMeans
- DBSCAN
- Metoda hierarchiczna

Dane jakie dysponowaliśmy zostały pobrane z biblioteki sklearn.datasets. Pierwszy etap badania polegał na doborze optymalnych parametrów dla wymienionych wyżej algorytmów, sprawdzeniu jak każdy model zmienia się w zależności od tych parametrów oraz który z otrzymanych modeli można uznać za najlepszy pod względem trafności doboru klastrów. Drugi etap polegał na zbadaniu wpływu szumu na dane zaś trzeci na zbadaniu przedstawieniu wykresu, który pokazuje zmianę sumy całkowitej odległości punktów w klastrach od liczby klastrów.

### 2. Implementacja metod

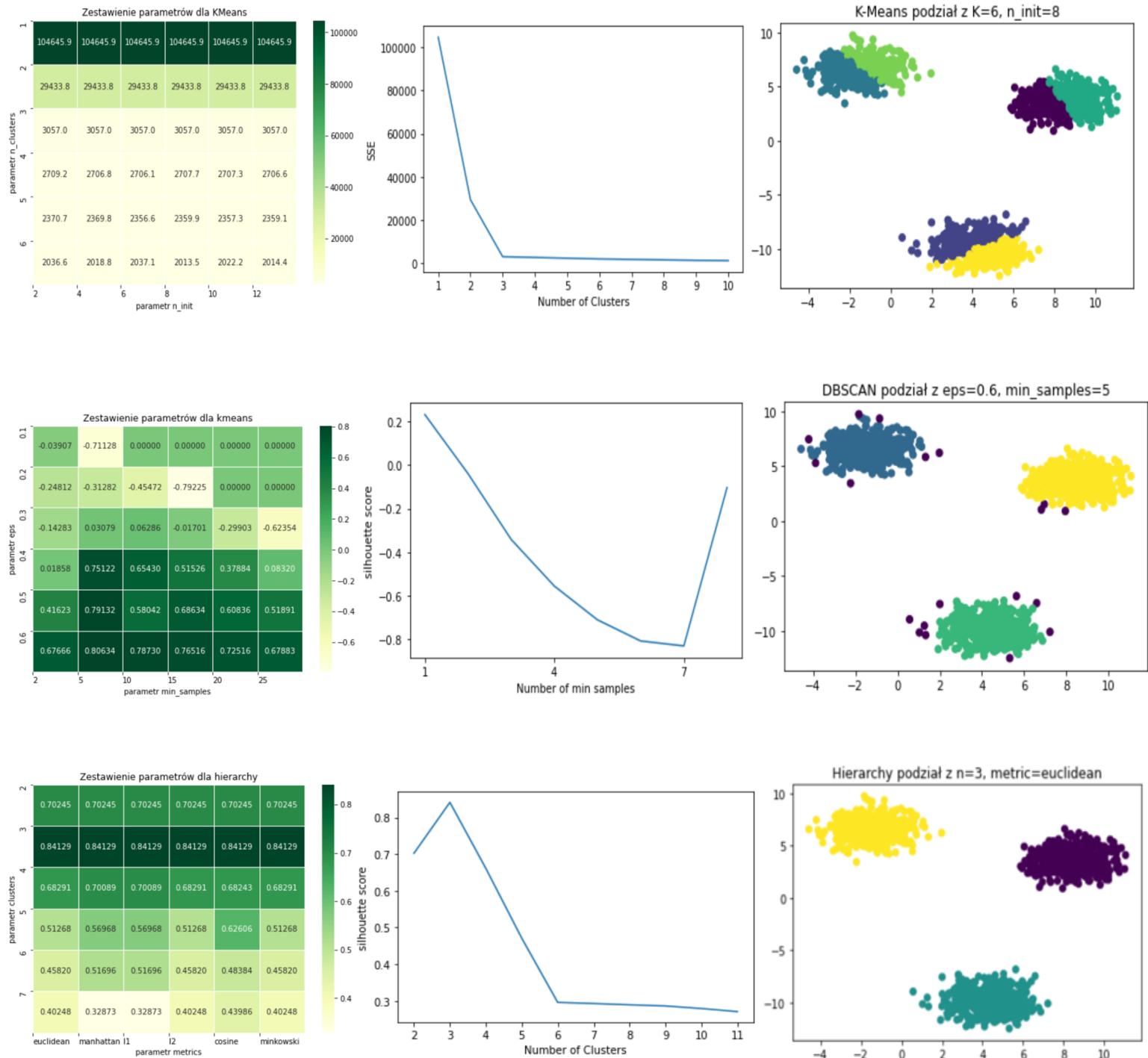
Aby dobrać najlepsze parametry dla każdego modelu stworzyłem funkcję, która rozważa jednocześnie dwa parametry i zmienia ich wartości w każdym powtórzeniu pętli. Następnie funkcja ta oblicza obecny wynik i wypisuje go na heatmapie. Druga funkcja rysuje wykres zależności wyniku od liczby klastrów (lub w przypadku dbscan od najmniejszej liczby próbek w obszarze punktu początkowego). W przypadku KMeans jako funkcja obliczająca wynik aktualnie rozpatrywanego modelu posłużyła mi metoda internal\_, która zwraca błąd średniokwadratowy, a dla dbscan i metody hierarchicznej użyłem miary silhouette\_score, która jest szczególną miarą dla problemów klastrowania. Ostatecznie stworzyłem funkcję, która rysuje zbiór podzielony na klastry wedle otrzymanych najlepszych parametrów. Zbiory, które należało przetestować to make\_circles, make\_moons, make\_blobs oraz zbiór składający się z losowo wygenerowanej próbki. Dla każdego z nich wygenerowane zostało 1500 punktów, a ich wygląd ukazałem poniżej.



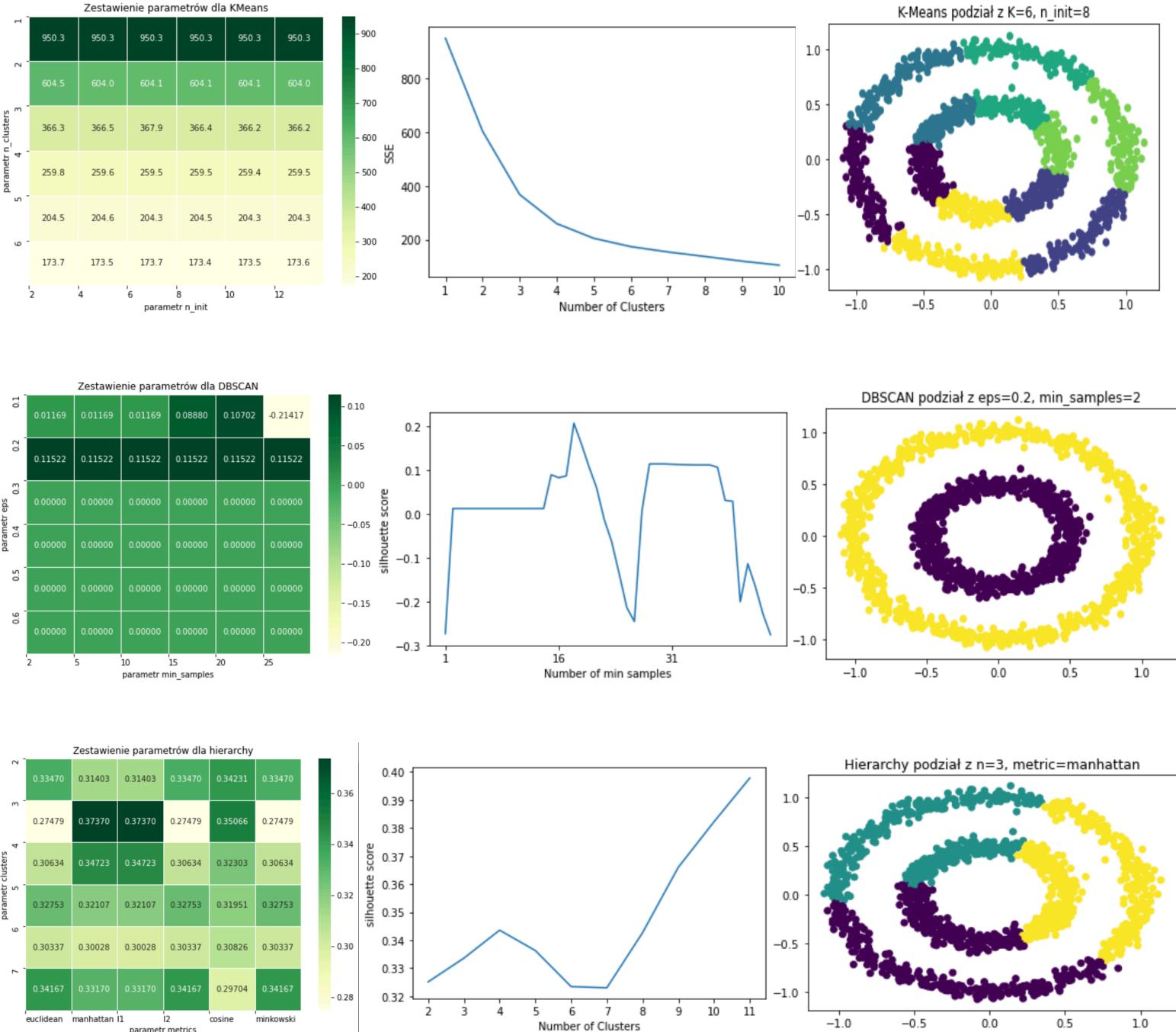
Dzięki stworzonym przezemnie funkcjom na poniższych stronach przedstawiłem w jaki sposób każdy algorytm dokonywał klastrowania w zależności od parametrów. Pierwsze wyniki dotyczą Kmeans, drugie DBSCAN zaś trzecie metody hierarchicznej.

### 3.1 Wyniki na pierwszym zbiorze

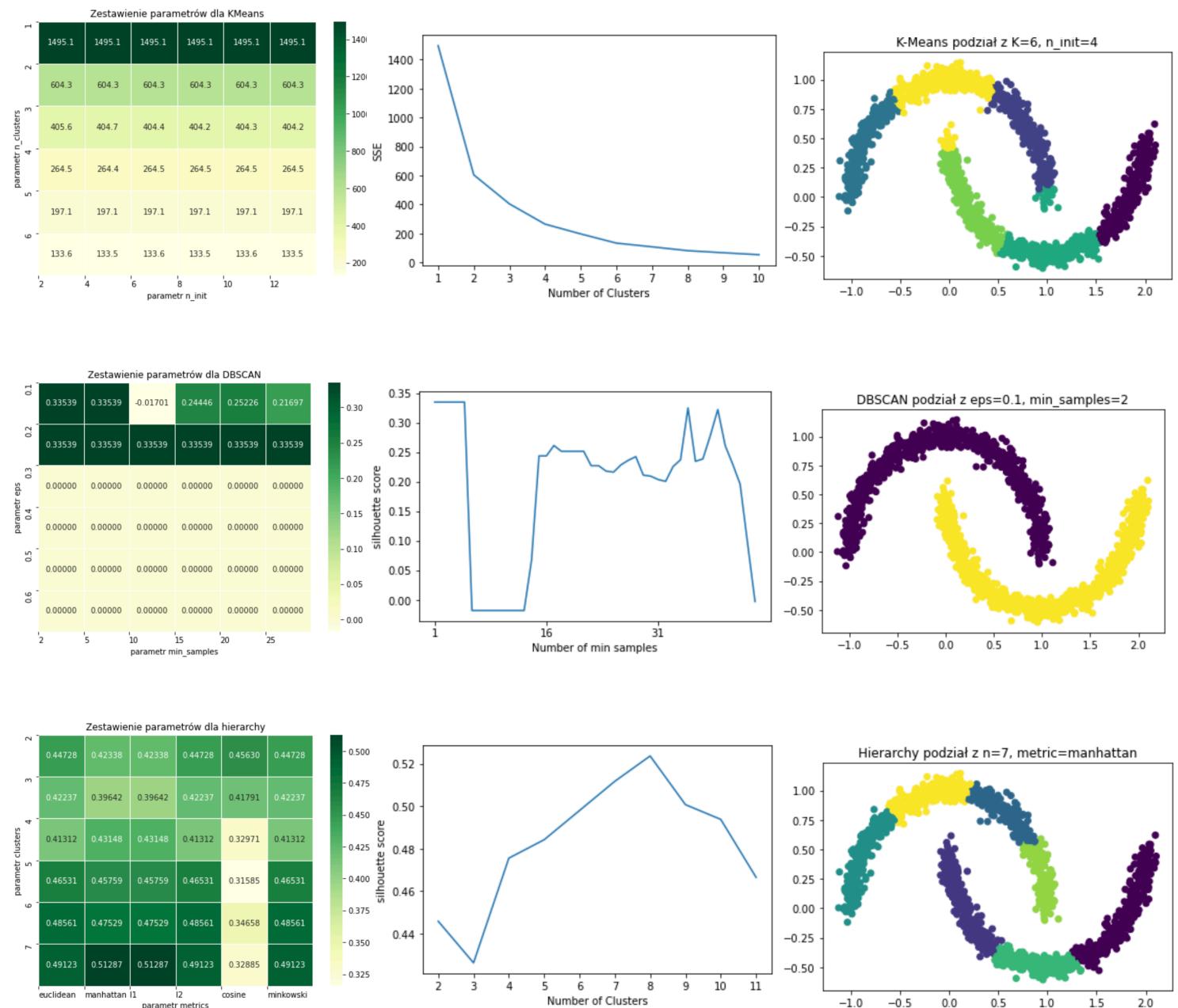
	KMeans	DBSCAN	Metoda hierarchiczna
Parametr 1	$N\_clusters = [1,2,3,4,5,6]$	$Min\_samples = [2,5,10,15,20,25]$	$N\_clusters = [2,3,4,5,6,7]$
Parametr 2	$N\_init = [2,4,6,8,10,12]$	$Eps = [0.1,0.2,0.3,0.4,0.5,0.6]$	$Metric = ['l1','l2','euclidean','manhattan','cosine','minkowski']$



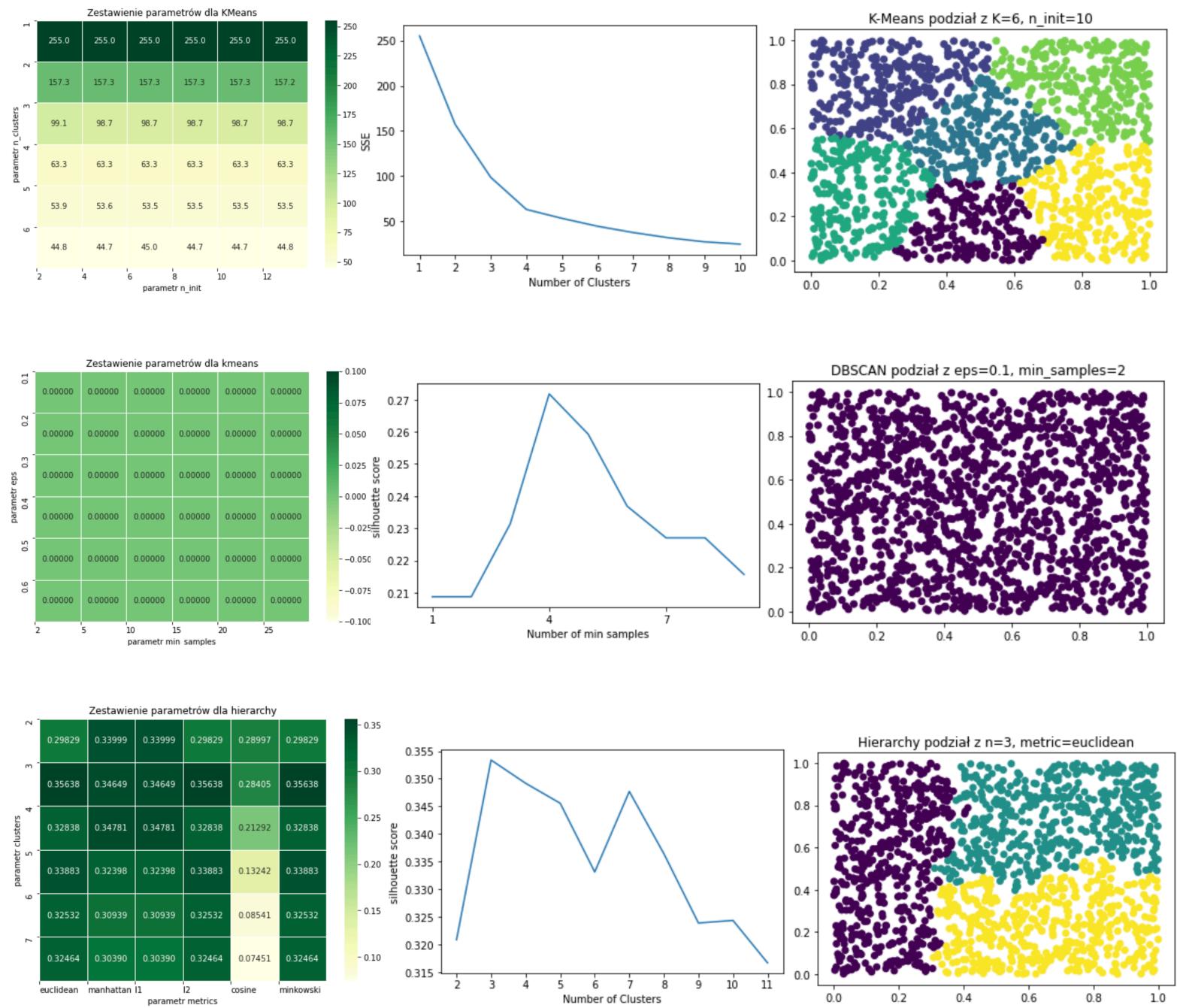
### 3.2. Wyniki na drugim zbiorze



### 3.3. Wyniki na trzecim zbiorze



### 3.4. Wyniki na czwartym zbiorze

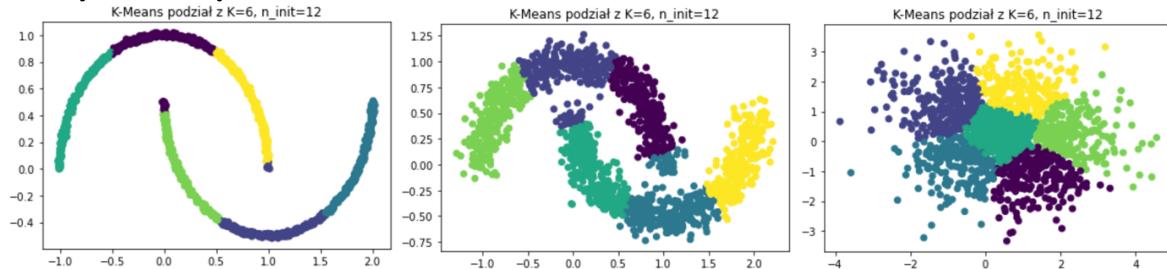


Analizując powyższe wizualizacje możemy wysnuć wniosek, że algorytm DBSCAN średnio najlepiej poradził sobie dla postawionych problemów. Wybierając optymalne parametry poprawnie zakwalifikował księżyce jak i kółka, czym potwierdził swoją skuteczność. W przypadku pierwszego zbioru lepszym od niego okazała się być metoda hierarchiczna, która poprawnie uznała, że klastrów powinno być trzy, a nie cztery jak zrobił to zwycięzca pozostałych kategorii. Możemy również zauważać, że KMeans nie radzi sobie dobrze z bardziej złożonymi i trudniej rozłożonymi klastrami, natomiast po jego wykresach możemy sprawdzić, w którym miejscu był największy spadek i w ten sposób zgadywać ile powinno być klastrów. Heatmaps w przypadku metody hierarchicznej i DBSCAN ukazują nam – im ciemniejsza pola tym wyższy *silhouette\_score* – dla których parametrów model okazał się być najskuteczniejszy co jest bardzo użytecznym i wygodnym sposobem do analizowania tych modeli. W przypadku ostatniego zbioru danych każdy z modeli inaczej zakwalifikował klastry, jednakże najrozsądzniejszy wydaje się wynik algorytmu DBSCAN, który uznał, że jest tylko jedna klastra jako cały zbiór, co również potwierdza jego wykres, w którym widzimy szybki wzrost na poziomie 4 minimalnych próbek do najwyższego wyniku, a później szybki spadek – tak jakby liczba minimalnych próbek dookoła niego straciła już znaczenie. W przypadku tego modelu kolejną obserwacją jaką możemy zauważać jest fakt, że zadziałał on szybko – dla danych po za pierwszym zbiorem zeruje on *silhouette\_score* dla większych parametrów, co oznacza w przypadku mojej implementacji, że model znalazł tylko jedną klastre. Dzięki temu widać, że działa on znacznie inaczej od pozostałych modeli, które dla księżyców i kółek obliczyły 3,4 klastry, zaś algorytm DBSCAN w momencie gdy znalazł dwie klastry z bardzo dobrym wynikiem może jedynie zejść do jednej klastre, co nie ma sensu w przypadku tych danych. Na podstawie wszystkich tych wniosków uważam, że DBSCAN najlepiej poradził sobie z klastrowaniem z łatwością pokonując pozostałe modele i sprawiając wrażenie jakby dokonywał klasyfikacji podobnie do człowieka.

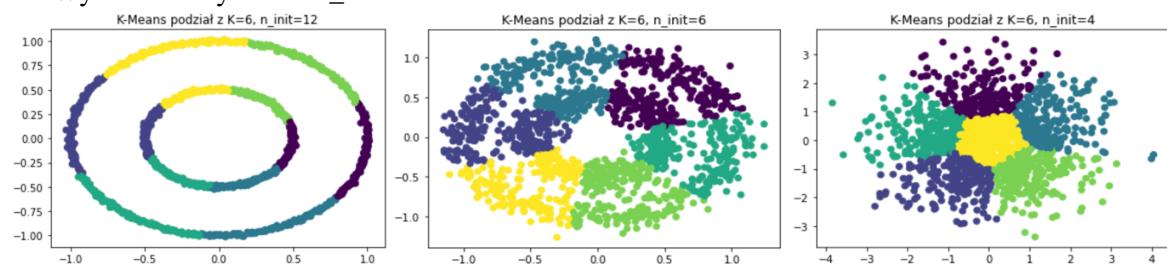
#### 4. Badanie wpływu szumu na dane

##### 4.1 KMeans

Wynik na danych make\_moons:



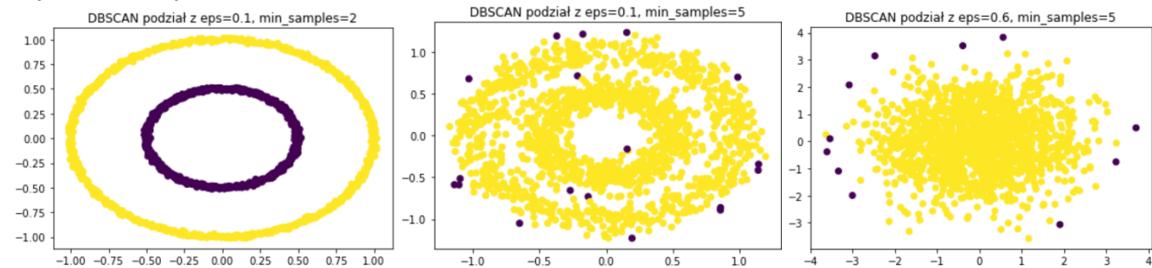
Wynik na danych make\_circles:



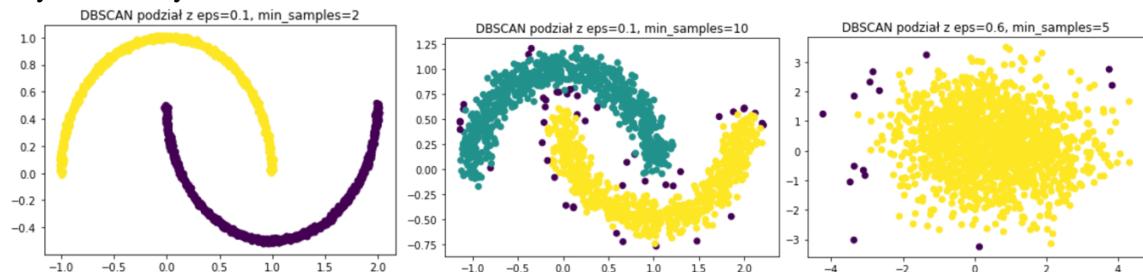
Widzimy, że w przypadku tego algorytmu szum nie wpływa na poprawę wyboru klastrów. Dodatkowo można zobaczyć, że mimo zmiany szumu optymalne parametry, jakie są wybierane nie zmieniają się (poza *n\_init* dla kółek) i w każdym przypadku otrzymaliśmy 6 klastrów.

## 4.1 DBSCAN

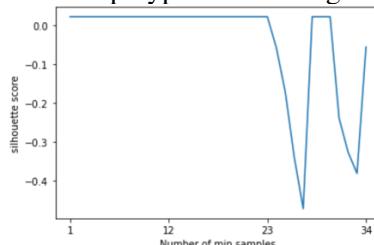
Wynik na danych make\_moons:



Wynik na danych make\_circles:



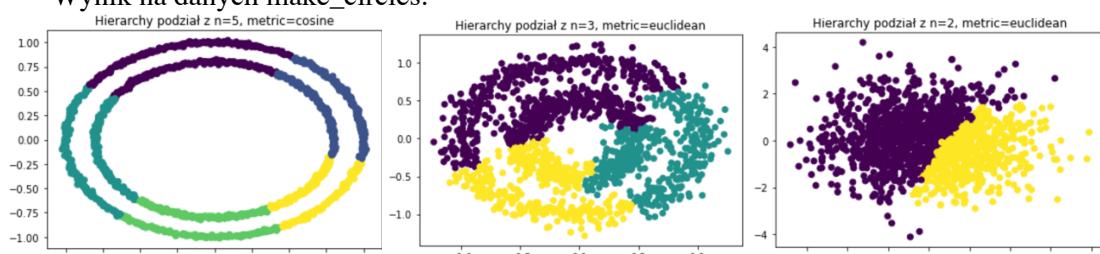
Dla DBSCAN szum znacznie wpływa na pogorszenie wyboru klastrów. Zauważmy, że dla danych make\_moons dla szumu 0.1 algorytm ten nie poradził sobie dobrze z wyborem klastrów, uznał że w środku jest tylko jeden klaster, zaś odstające wartości dopasował jako drugi. Dla największego szumu model klasyfikuje główny klaster w środku i wartości odstające co jest dobrym wynikiem. Również w przypadku średniego szumu dla księżyców klastry wydają się być podzielone prawidłowo.



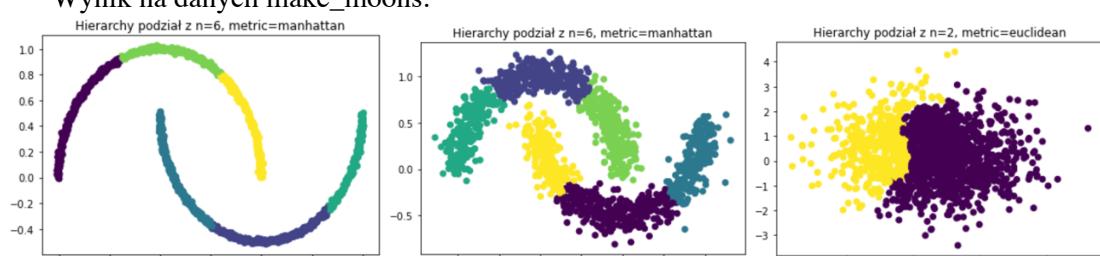
Ciekawą obserwacją jest również wynik miary *silhouette* w zależności od minimalnej ilości próbek w obszarze – jeżeli wartość jest ujemna to oznacza, że parametry wpływają negatywnie na ilość klastrów. Widzimy na wykresie jak znacząco algorytm nie myli się w wyborze optymalnych parametrów.

## 4.1 Metoda hierarchiczna

Wynik na danych make\_circles:



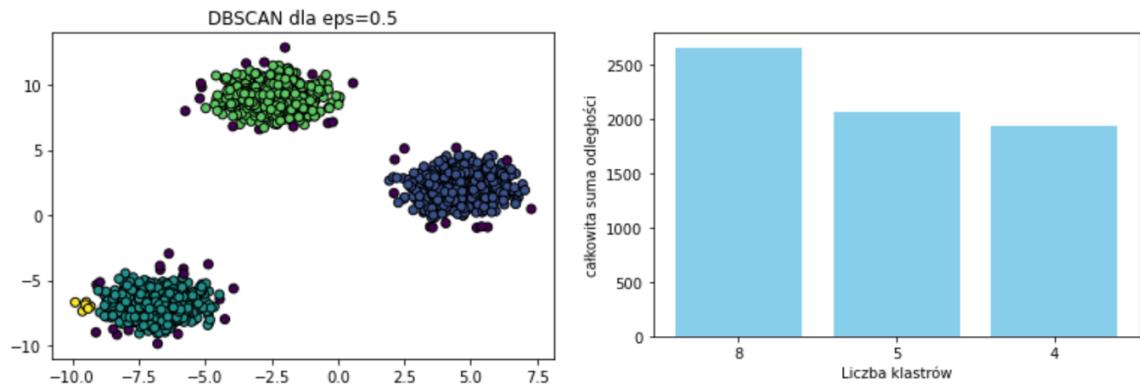
Wynik na danych make\_moons:



W przypadku tego algorytmu widzimy słabe wyniki podobnie jak dla KMeans, natomiast zmiana szumu wpływa na wybór parametrów szczególnie na wybór metryki. Niestety metoda hierarchiczna na tych danych sprawdza się gorzej niż zaprezentowany wyżej DBSCAN

## 5. Odległość między punktami w klastrach

Z powodu, że algorytm DBSCAN okazał się być najlepszym dla przedstawionych wyżej problemów, przeprowadziłem na nim badanie ukazujące jak całkowita suma odległości między punktami w klastrach zależy od liczby klastrów.



## 6. Podsumowanie

Po przeprowadzeniu powyższego badania widzimy jakie metody klastrowania są najbardziej odpowiednie do konkretnych problemów oraz jak działają. Wybór odpowiednich parametrów jest kluczowym elementem każdego zadania, co ułatwia później jak w przypadku algorytmu DBSCAN doprowadzenie do idealnego podziału danych na klastry.