Usprawnienie algorytmu k-means przez wyznaczenie centroidów za pomocą średniej ważonej i średniej arytmetycznej.

1. Opis algorytmu k-means.

Metoda k-średnich jest metodą należącą do grupy algorytmów analizy skupień tj. analizy polegającej na szukaniu i wyodrębnianiu grup obiektów podobnych (skupień) . Reprezentuje ona grupę algorytmów niehierarchicznych. Główną różnicą pomiędzy niehierarchicznymi i hierarchicznymi algorytmami jest konieczność wcześniejszego podania ilości skupień. Przy pomocy metody k-średnich zostanie utworzonych k różnych możliwie odmiennych skupień. Algorytm ten polega na przenoszeniu obiektów ze skupienia do skupienia tak długo, aż zostaną zoptymalizowane zmienności wewnątrz skupień oraz pomiędzy skupieniami.

2. Wady i zalety algorytmu k-means.

Wady:

- **Wymagają ustalenia liczby grup** – zanim uruchomimy algorytm, musimy podać liczbę grup, które mają zostać wyznaczone. Bez uprzedniego wizualizowania zbioru lub wykonania dodatkowych analiz jest to dosyć trudne.
- **Wrażliwe na dobór punktów startowych** – w pierwszej iteracji swojego działania algorytm **losowo** dobiera punkty startowe. To jak dobre wyniki uzyska, zależy zatem w pewnym stopniu od czynnika losowego.
- **Wrażliwy na wpływ obserwacji odstających i szum** – podobnie jak wartość średnia, tak i algorytm k-średnich jest wrażliwy na wpływ obserwacji odstających. Przyczyna jest błaha: do wyznaczenia przeciętnej obserwacji używana jest wartość średnia współrzędnych wszystkich obserwacji danej grupy.
- **Każdy klaster ma w przybliżeniu tę samą liczbę obserwacji**.
- **Używa jedynie zmiennych numerycznych** – aby liczyć średnie ze współrzędnych obserwacji konieczne jest używanie zmiennych numerycznych

Zalety:

- **Szybsze na dużych zbiorach niż algorytmy hierarchiczne** – wynika to bezpośrednio ze sposobu jego działania. Niższa złożoność obliczeniowa sprawia, że w porównaniu z grupowaniem aglomeracyjnym, algorytm k-średnich działa błyskawicznie. Wielkość zbioru przestaje więc być tak dużym problemem.

- **Bardzo szybki** (relatywnie niska złożoność obliczeniowa: iteracje * liczba_grup * instancje * wymiary) – złożoność obliczeniowa każdej iteracji to: O(k * N), gdzie N-liczba obserwacji w grupie. Problemem jest liczba iteracji potrzebna do osiągnięcia zbieżności.

Duża ilość wad algorytmu sprawia, że jest on podatny na różne modyfikacje i ulepszenia. Oto niektóre z nich:

- Różne sposoby znajdowania odległości (stosowanie różnych metryk)
- Zmiana liczby grup w trakcie pracy algorytmu (zapobieganie nadmiernej unifikacji i przesadnemu rozdrobnieniu)
- Wykorzystanie ważonej miary odległości uwzględniającej znaczenie atrybutów

3. Działanie algorytmu k-means.

1. **Ustalamy liczbę skupień.**
   Jedną z metod ustalenia ilości skupień jest umowny jej wybór i ewentualna późniejsza zmiana tej liczby w celu uzyskania lepszych wyników. Wybór liczby skupień może być oparty również na wynikach innych analiz.

2. **Ustalamy wstępne środki skupień.**
   Środki skupień tak zwane centroidy możemy dobrać na kilka sposobów: losowy wybór k obserwacji, wybór k pierwszych obserwacji, dobór w taki sposób, aby zmaksymalizować odległości skupień. Jedną z najczęściej stosowanych metod jest kilkakrotne uruchomienie algorytmu i wybór najlepszego modelu, gdy wstępnie środki skupień były wybierane losowo.

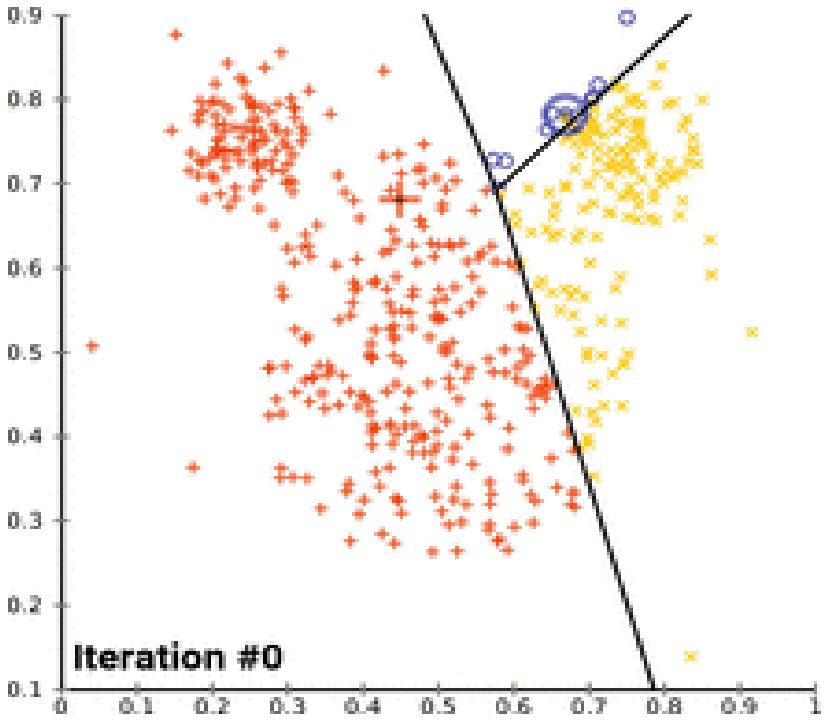
3. **Obliczamy odległości obiektów od środków skupień.**
   Wybór metryki jest bardzo istotnym etapem w algorytmie. Wpływa ona na to, które z obserwacji będą uważane za podobne, a które za zbyt różniące się od siebie. Najczęściej stosowaną odległością jest odległość euklidesowa. Stosuje się również kwadrat tej odległości czy też odległość Czebyszewa.

4. **Przypisujemy obiekty do skupień**
   Dla danej obserwacji porównujemy odległości od wszystkich skupień i przypisujemy ją do skupienia, do którego środka ma najbliżej.

5. **Ustalamy nowe środki skupień**
   Najczęściej nowym środkiem skupienia jest punkt, którego współrzędne są średnią arytmetyczną współrzędnych punktów należących do danego skupienia.

6. **Wykonujemy kroki 3,4,5 do czasu, aż warunek zatrzymania zostanie spełniony.**
   Najczęściej stosowanym warunkiem stopu jest ilość iteracji zadana na początku lub brak przesunięć obiektów pomiędzy skupieniami.

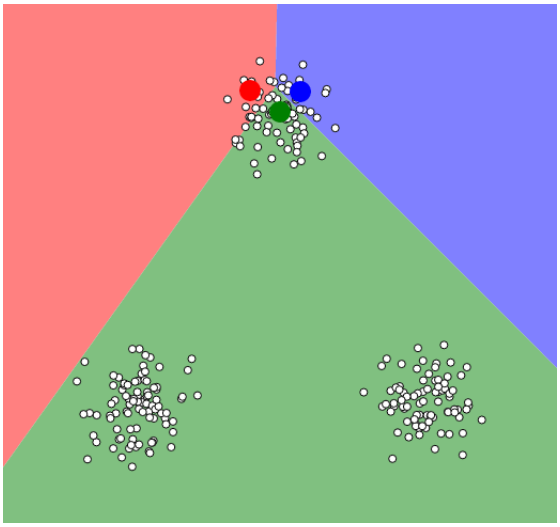
Przykładowa animacja działania algorytmu:



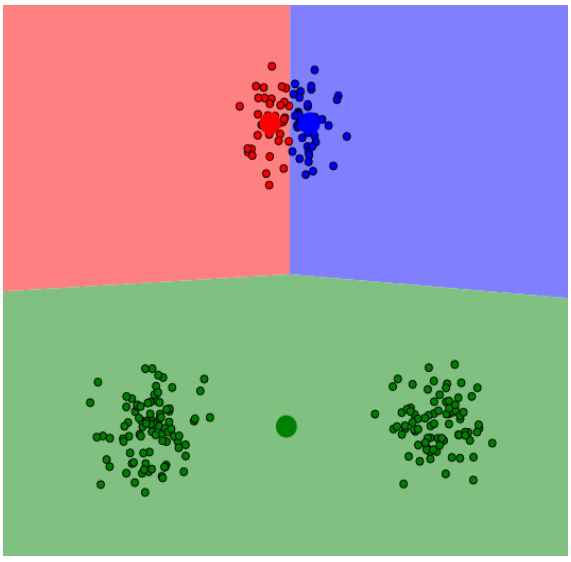
4. Problem inicjalizacji centroidów.

Powszechnie stosowaną metodą inicjalizacji centroidów jest losowe przydzielenie punktu do każdego klastra jako jego środek. Wskutek dalszego działania algorytmu, każde centrum jest aktualizowane i równe środkowi ciężkości zbioru. Takie rozwiązanie ma jednak dużo wad. Wyznaczenie środka każdego klastra ma wpływ na ostateczne pogrupowanie danych oraz na ilość iteracji jaką musi wykonać algorytm, aby otrzymać wynik końcowy. Zbyt duża ilość iteracji powoduje bardzo długie działanie algorytmu co jest niewskazane przy próbie pogrupowania dużej ilości danych.

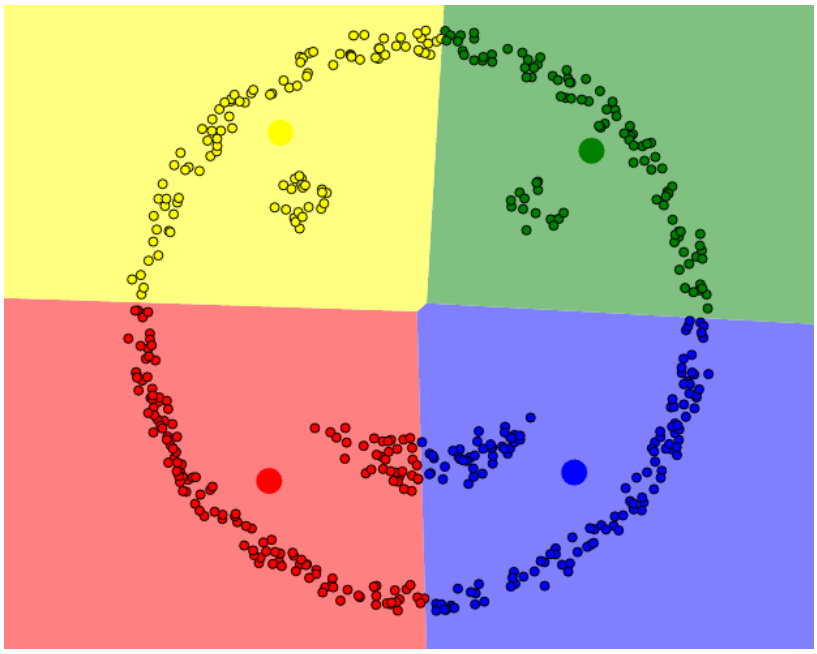
Przykład pechowego rozmieszczenia centroidów:



Rezultat działania algorytmu k-means:



Jedną ze strategii obrony jest wybór punktów początkowych, które będą od siebie „maksymalnie” oddalone. Niestety nie zawsze daje to dobre rezultaty. Poniższy przykład pokazuje, że pogrupowane dane nie zawsze odzwierciedlają naszą intuicję.



5. Usprawnienie inicjalizacji centroidów za pomocą średniej ważonej i średniej arytmetycznej.

Do próby usprawnienia inicjalizacji centroidów posłuży nam wartość następującego wyrażenia - wartość bezwzględna z różnicy średniej ważonej i średniej arytmetycznej z odległości między punktami.

$$|śr_w - śr_a|$$

Do wyznaczenia takiej wartości będzie nam potrzebny zbiór składający się z próbek, które zawierają punkty o tej samej ilości co liczba klastrów. Wielkość zbioru jest dowolna, ale nieprzekraczająca wszystkich możliwych kombinacji zestawienia ze sobą punktów, które nie mogą się powtarzać w próbce. Oczywiście będziemy liczyć wartość wyrażenia dla każdej poszczególnej próbki, tworząc zbiór współczynników.

Zbadajmy zatem funkcję $|śr_w - śr_a|$, przyjmując wagę odcinka między dwoma punktami jako długość odcinka podzielona przez sumę wszystkich odcinków. Z tego wynika, że suma wag zawsze będzie równa 1. Dla prostoty dalszych obliczeń przyjmiemy 3 podane odcinki a, b oraz c.

$$|śr_w - śr_a|$$

$$\left|\left(a * \frac{a}{a+b+c}\right) + \left(b * \frac{b}{a+b+c}\right) + \left(c * \frac{c}{a+b+c}\right) - \left(\frac{a+b+c}{3}\right)\right|$$

$$\left|\frac{1}{a+b+c} * (a^2 + b^2 + c^2) - (\frac{a+b+c}{3})\right|$$

$$\left|\frac{3a^2+3b^2+3c^2-(a+b+c)^2}{3(a+b+c)}\right|$$

Po rozpisaniu nawiasu i uproszczeniu wyrażenia otrzymujemy:

$$\left|\frac{2a^2+2b^2+2c^2-2ab-2bc-2ac}{3(a+b+c)}\right|$$

Następnie rozbijamy $2a^2$ na sumę a^2+a^2. Tak samo z $2b^2$ oraz $2c^2$, grupując w następujący sposób.

$$\left|\frac{(a^2-2ab+b^2)+(b^2-2bc+c^2)+(a^2-2ac+c^2)}{3(a+b+c)}\right|$$

Teraz korzystamy ze wzoru skróconego mnożenia, otrzymując ostateczną formę:

$$\left|\frac{(a-b)^2 + (b-c)^2 + (a-c)^2}{3(a+b+c)}\right|$$

Rozważmy teraz przykładowe długości odcinków: $a = b = c$. Podstawiając podane wartości do wzoru, który wyprowadziliśmy, otrzymamy 0.

Rozważmy teraz długości odcinków: $a \neq b \neq c$. Podstawiając wartości do wzoru otrzymamy liczbę dodatnią, większą bądź mniejszą, w zależności od konkretnych długości.

Wnioski jakie możemy wyciągnąć z tych wyników to:

Jeżeli rozkład klastrów jest zbliżony do wielokąta foremnego to punkty, które miały najmniejszy współczynnik, będą naszymi centroidami. Natomiast, jeśli rozkład klastrów jest daleki od kształtu wielokąta foremnego to punkty, które miały największy współczynnik będą centroidami.

Decydując się na dużą ilość próbek musimy mieć na uwadze wolniejsze działanie algorytmu, ale za to większą efektywność, a wybierając mniejszą ilość musimy się liczyć z mniejszą wydajnością algorytmu, ale za to szybszym działaniem.

6. Działanie ulepszonego algorytmu k-means.

Kroki algorytmu będą takie same jak w rozdziale nr.3. Jedyny punkt, który uległ zmianie to nr.2. Rozpiszmy go więc jeszcze raz:

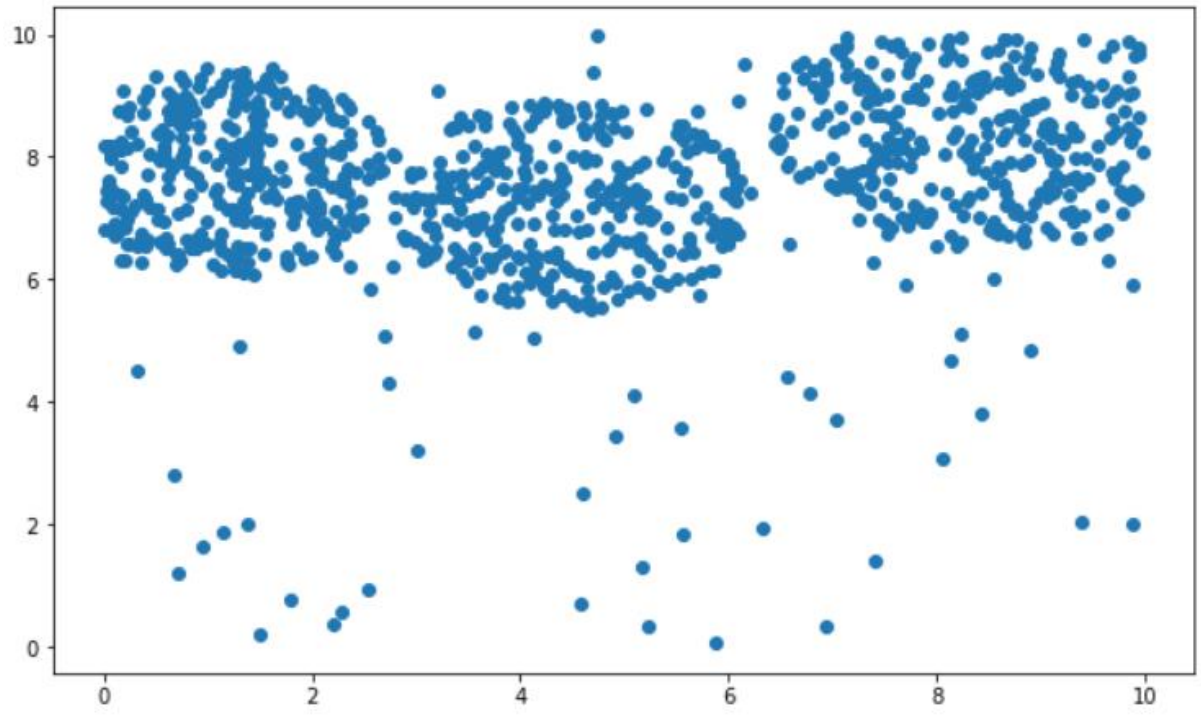
2. **Ustalamy wstępne środki skupień.**
   Z danych tworzymy zbiór zawierający próbki. Każda próbka zawiera punkty o tej samej ilości co liczba klastrów. Maksymalna wielkość zbioru zawierającego próbki jest równa wszystkim możliwym kombinacjom zestawienia ze sobą punktów, bez powtórzenia punktu w próbce. Następnie, dla każdej próbki, liczymy wartość wyrażenia: $|śr_w - śr_a|$, przyjmując wagę odcinka między dwoma punktami jako długość odcinka podzielona przez sumę wszystkich odcinków. Tworzymy w ten sposób nowy zbiór zawierający współczynniki. W zależności od sytuacji, centroidami będą punkty, które miały najmniejszy, bądź największy współczynnik z całego zbioru. Najmniejszy współczynnik będzie nas interesował jeśli rozkład klastrów przypomina figurę foremną, a największy wtedy, kiedy rozkład klastrów jest daleki od figury foremnej.

7. Sprawdzenie poprawności wniosków.

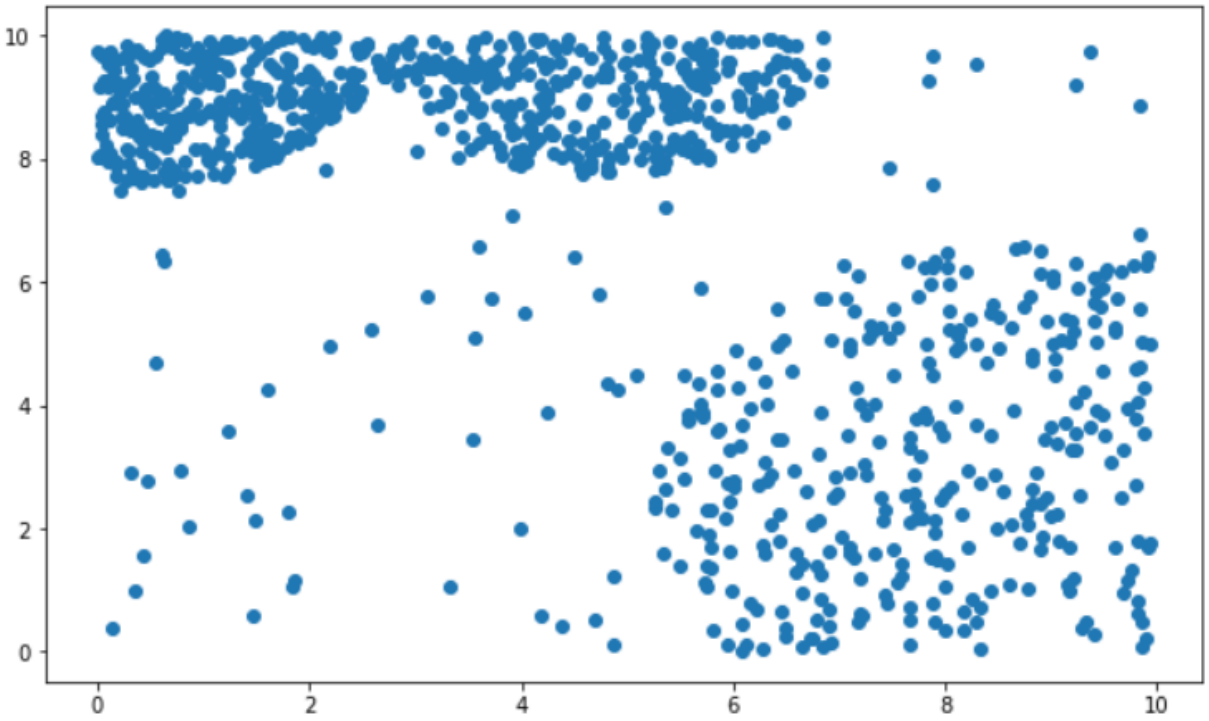
Do sprawdzenia wniosków posłużymy się implementacją jednego i drugiego algorytmu w języku Python. Kod, z którego będziemy korzystać został podany w źródłach.

Dane, na których będą testowane wydajności algorytmów, zostały stworzone przez własnoręcznie napisany algorytm. Wybrano cztery następujące klastry:

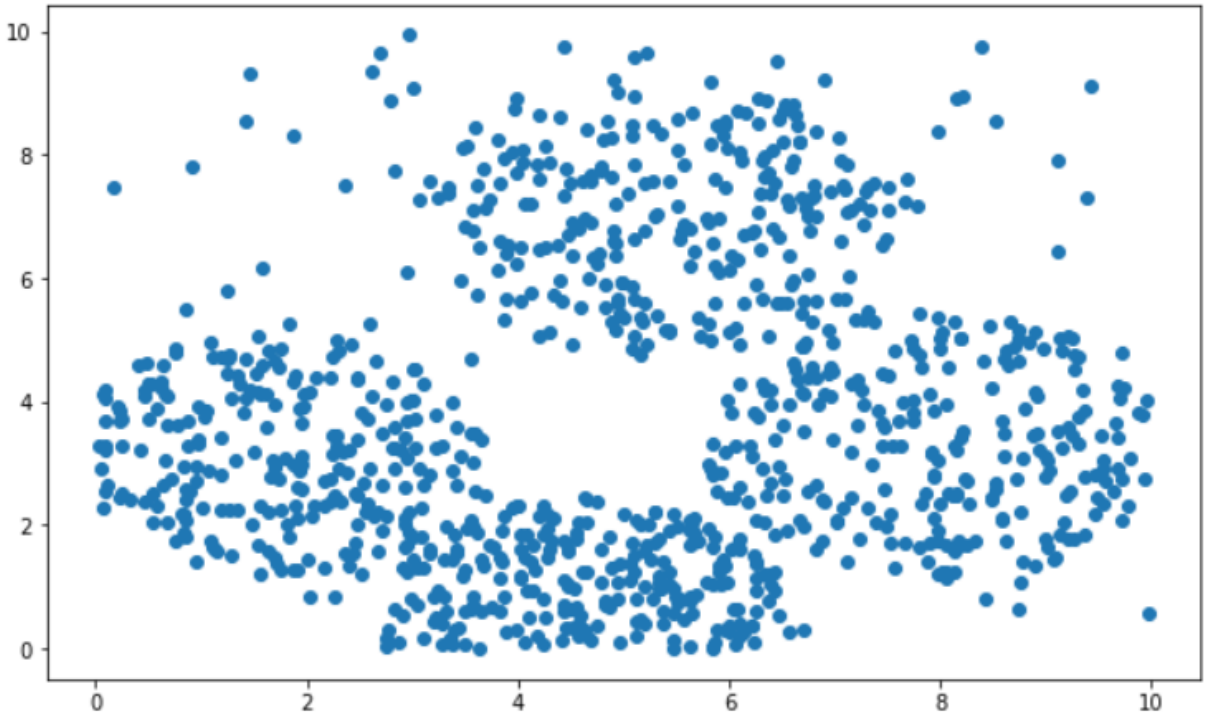
- Symetryczny rozkład trzech klastrów – szukanie minimum ze zbioru próbek o wielkości 33% i 66% wszystkich danych.



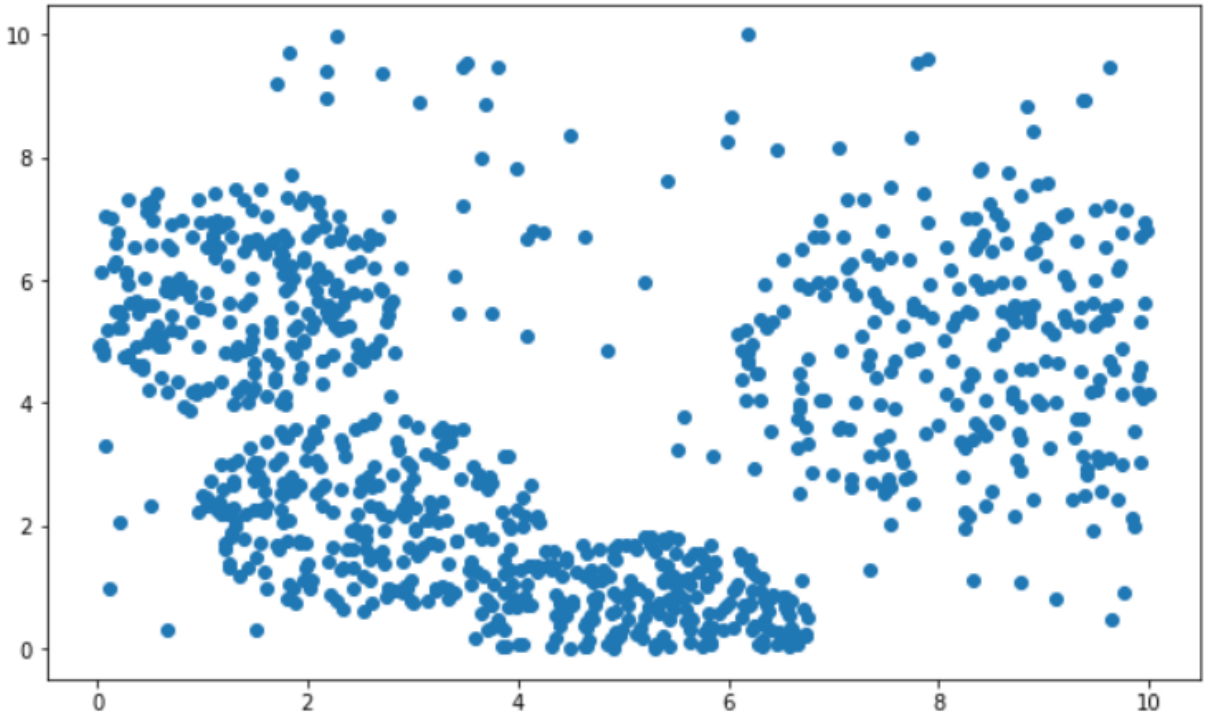
- Niesymetryczny rozkład trzech klastrów – szukanie maksimum ze zbioru próbek o wielkości 33% i 66% wszystkich danych.



- Symetryczny rozkład czterech klastrów – szukanie minimum ze zbioru próbek o wielkości 25% i 50% wszystkich danych.



- Niesymetryczny rozkład czterech klastrów – szukanie maksimum ze zbioru próbek o wielkości 25% i 50% wszystkich danych.



Każdy rozkład składał się z tysiąca punktów, z czego sto punktów było tzw. szumem (losowe punkty rozdzielone po całej płaszczyźnie). Reszta punktów została podzielona na ilość klastrów i stanowiła ich gęstość.

Do wyznaczenia wydajności algorytmów, każdy rozkład został 50 razy przepuszczony przez każdy algorytm. Dane jakie zostały sprawdzone to:

- Ilość prób, w których algorytm odpowiednio sklasyfikował dane
- Średnia ilość iteracji jaka była potrzebna przy odpowiednim sklasyfikowaniu danych
- Średnia ilość iteracji we wszystkich próbach klasyfikacji

Oto otrzymane wyniki:

- Symetryczny rozkład trzech klastrów (33%)

```
Efficiency of improved algorithm: 0.92   5.3478260869565215   5.56
Efficiency of classical algorithm: 0.92   4.891304347826087   5.14
```

- Symetryczny rozkład trzech klastrów (66%)

```
Efficiency of improved algorithm: 0.84   5.285714285714286   5.72
Efficiency of classical algorithm: 0.94   4.425531914893617   4.64
```

- Niesymetryczny rozkład trzech klastrów (33%)

```
Efficiency of improved algorithm: 0.68   5.411764705882353   5.48
Efficiency of classical algorithm: 0.72   4.944444444444445   5.28
```

- Niesymetryczny rozkład trzech klastrów (66%)

```
Efficiency of improved algorithm: 0.7   5.914285714285715   5.9
Efficiency of classical algorithm: 0.88   5.25   5.44
```

- Symetryczny rozkład czterech klastrów (25%)

```
Efficiency of improved algorithm: 1.0   6.7   6.7
Efficiency of classical algorithm: 1.0   6.74   6.74
```

- Symetryczny rozkład czterech klastrów (50%)

```
Efficiency of improved algorithm: 1.0   6.66   6.66
Efficiency of classical algorithm: 1.0   6.12   6.12
```

- Niesymetryczny rozkład czterech klastrów (25%)

```
Efficiency of improved algorithm: 0.78   5.076923076923077   5.82
Efficiency of classical algorithm: 0.88   8.068181818181818   7.98
```

- Niesymetryczny rozkład czterech klastrów (50%)

```
Efficiency of improved algorithm: 0.98   4.795918367346939   4.84
Efficiency of classical algorithm: 0.86   7.534883720930233   7.62
```

Otrzymane wyniki są bardzo różne. Wraz ze wzrostem zbioru próbek, nie idzie większa dokładność algorytmu opartego na inicjalizacji centroidów za pomocą średniej ważonej i średniej arytmetycznej. Cały proces pozostaje losowy. **Liczenie różnicy z średniej ważonej i średniej arytmetycznej nie gwarantuje nam tego, że punkty, które zostały wyznaczone, będą znajdywać się w osobnych klastrach, co spowodowałoby usprawnienie klasycznej wersji algorytmu k-means.**

Źródła:

- https://en.wikipedia.org/wiki/K-means_clustering
- http://itcraftsman.pl/algorytm-k-srednich-uczenie-nienadzorowane/
- https://mateuszgrzyb.pl/k-srednich-teoria/
- https://www.statystyka.az.pl/analiza-skupien/metoda-k-srednich.php
- https://www.kdnuggets.com/2017/03/naive-sharding-centroid-initialization-method.html
- https://github.com/Pawkooo13/K-means (kod)