Metody rozpoznawania obrazów i podstawy uczenia maszynowego

Metody klasteryzacji

Paulina Żak

4. Kmeans

Implementacja algorytmu

algorytm przyjmuje zestaw danych do klasteryzacji, ilość klastrów i metodę, którą zostaną wyznaczone dystanse.

Wizualizacja

Zaproponowaną wizualizacją jest wykres punktowy w 2 lub 3 wymiarach.

Po wydzieleniu klastrów została przeprowadzona redukcja wymiarów metodą PCA.

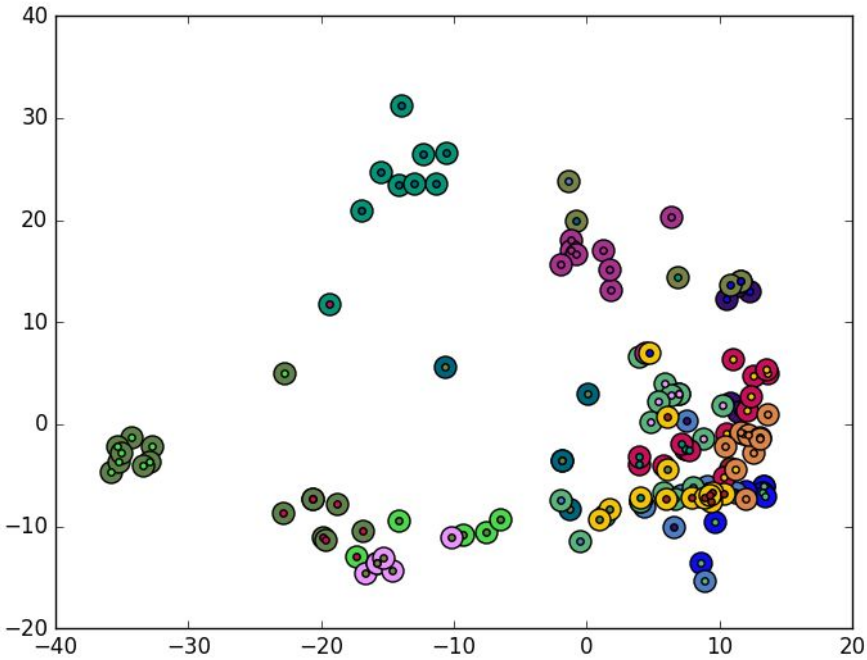
Metoda ta została wybrana, gdyż pozwala na efektywne wykreślenie podziału danych Utrata informacji związana z redukcją przestrzeni cech jest dość znacząca (tylko 33.8% i 21.1% zmienności zostało opisane przez pierwszą i drugą składową) i raczej nie nadawałaby się do analizy, niemniej jednak uważam ją za wystarczającą do analizy wizualnej danych.

Opis wykresu:
Każde z doświadczeń na zbiorze zostało wykreślone jako koło o pewnym kolorze:

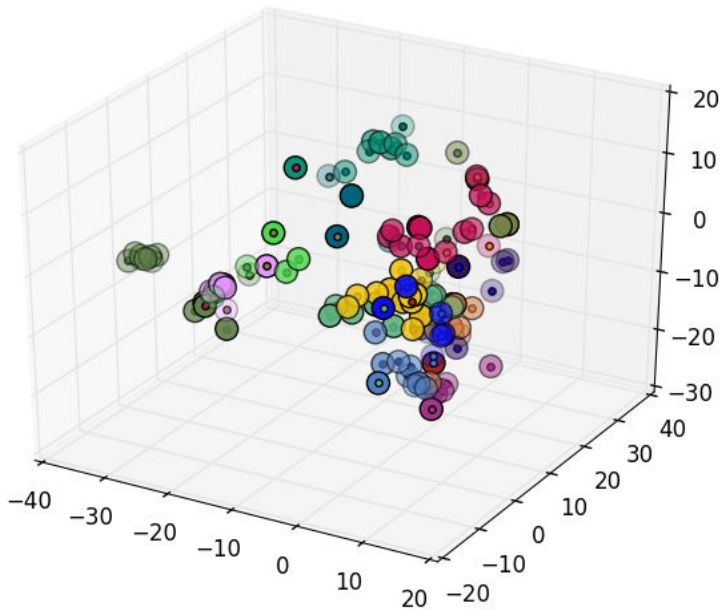
- o większym promieniu, gdzie
  kolor oznacza przypisanie do jednego z klastrów
- o mniejszym promieniu, gdzie
  kolor oznacza osobę, którą dany obrazek przedstawia

Odległość euklidesowa dla K =15

Wykres punktowy 2d

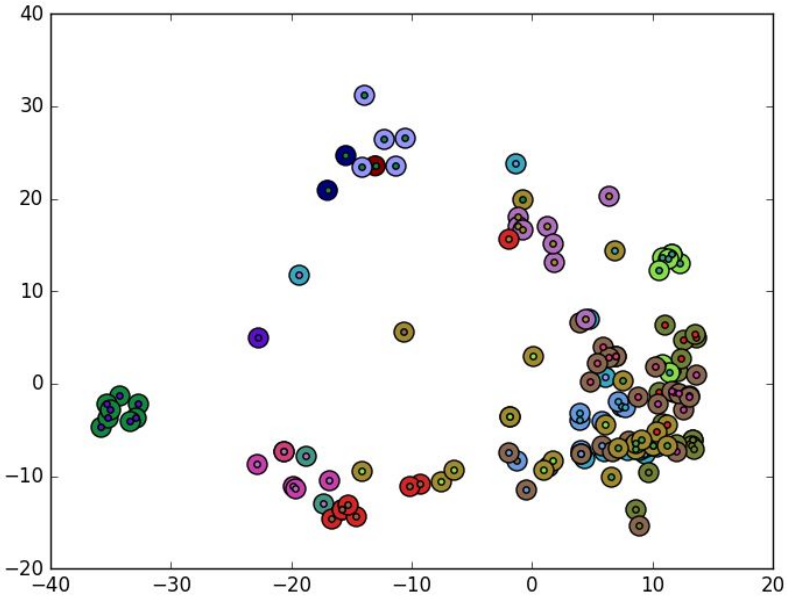


Wykres punktowy 3d

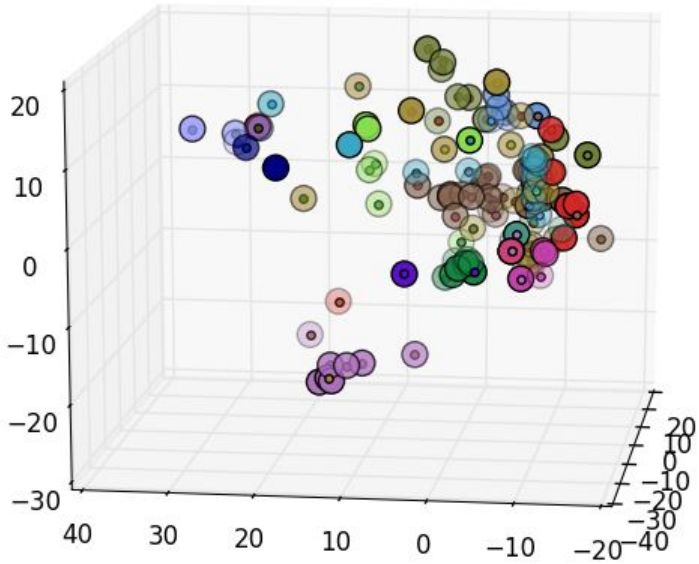


Odległość Mahalanobisa dla K=15

Wykres punktowy 2d



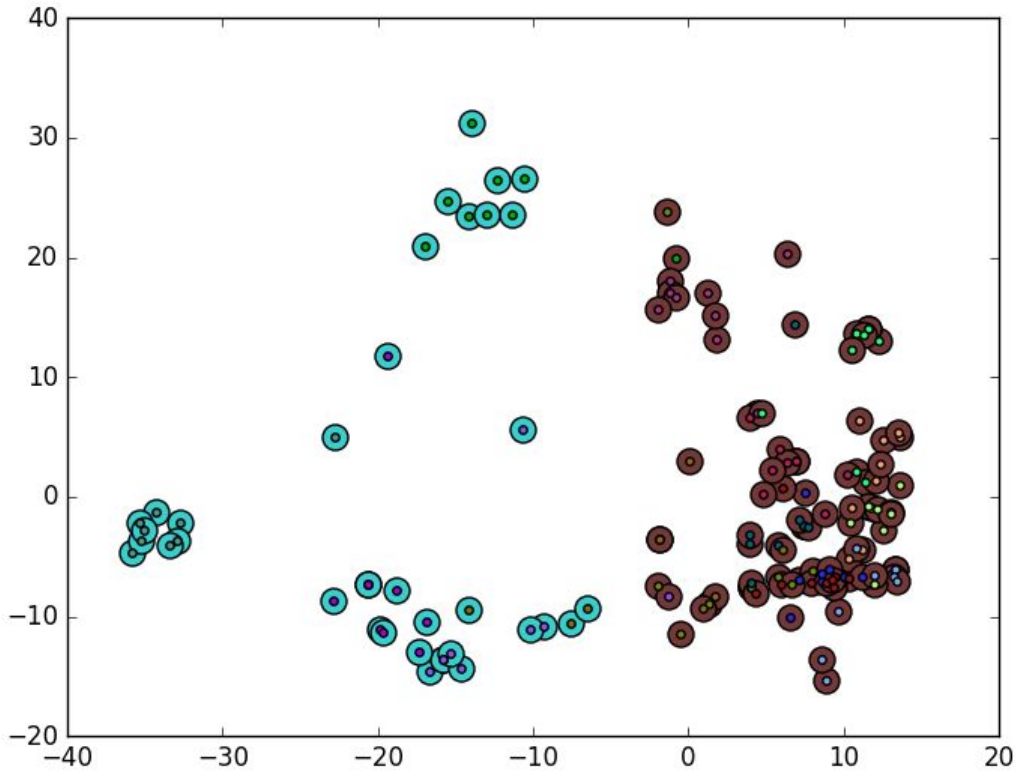
Wykres punktowy 3d



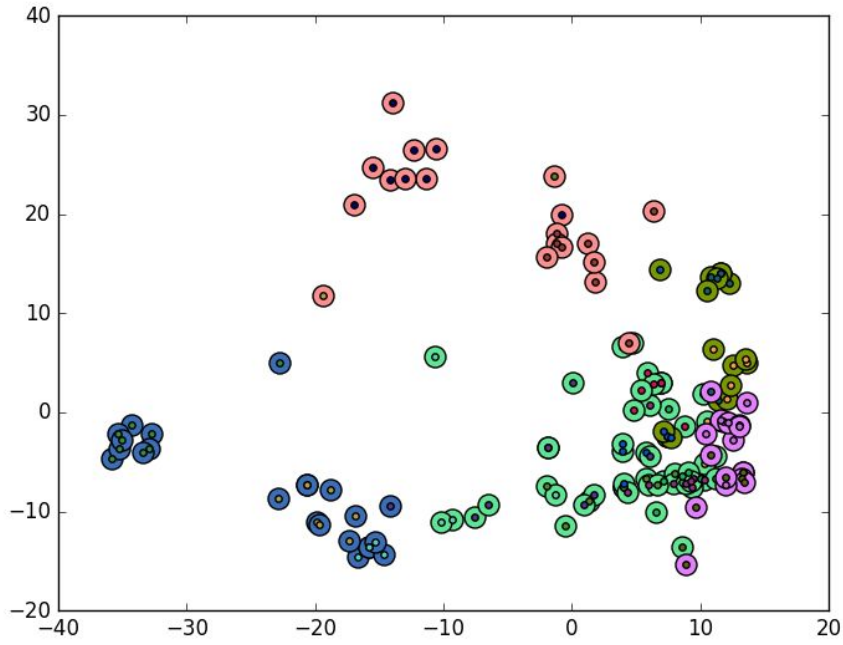
Różna ilość klastrów

Testy Kmeans z miarą euklidesową, gdy

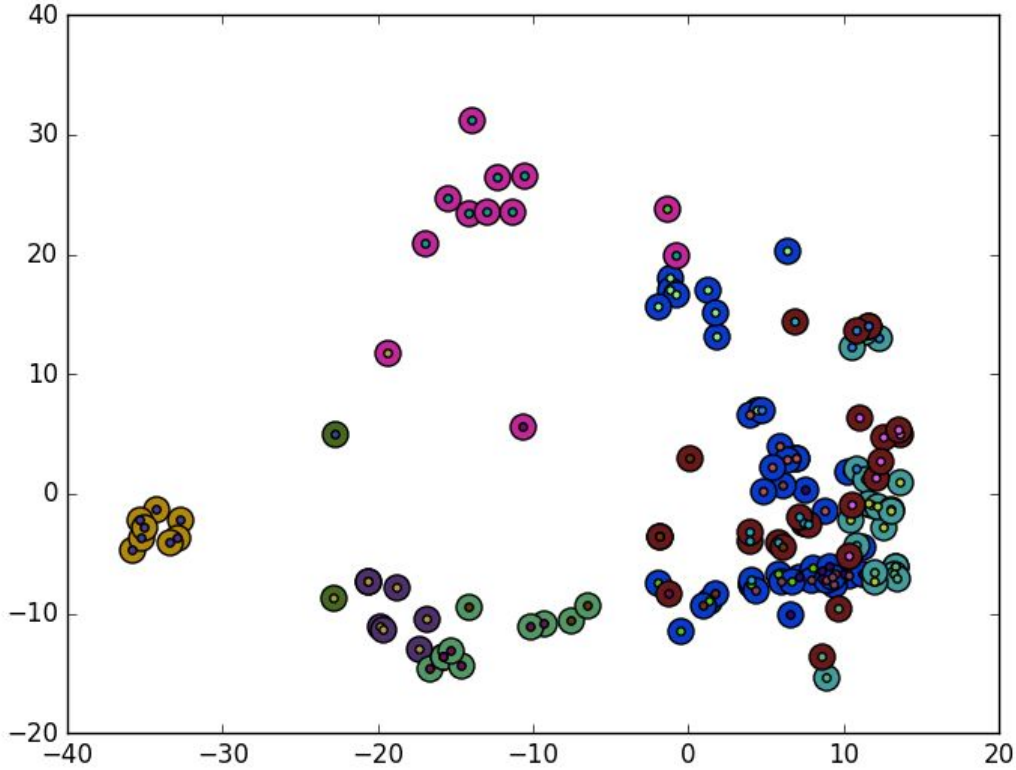
K=2


40
30
20
10
0
−10
−20
−40
−30
−20
−10
0
10
20

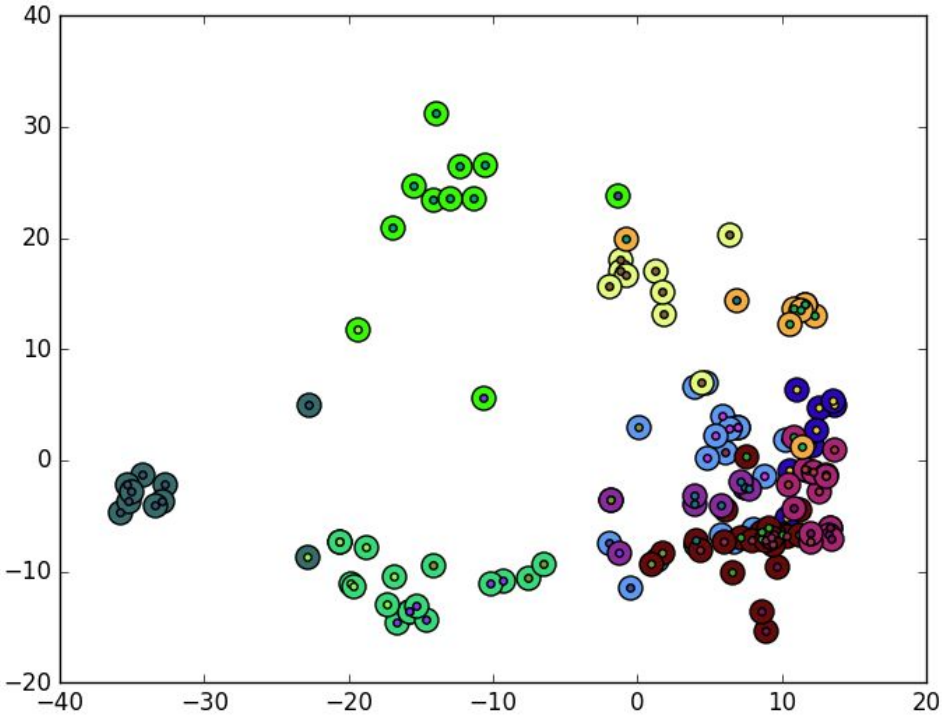
K=5


40
30
20
10
0
−10
−20
−40
−30
−20
−10
0
10
20

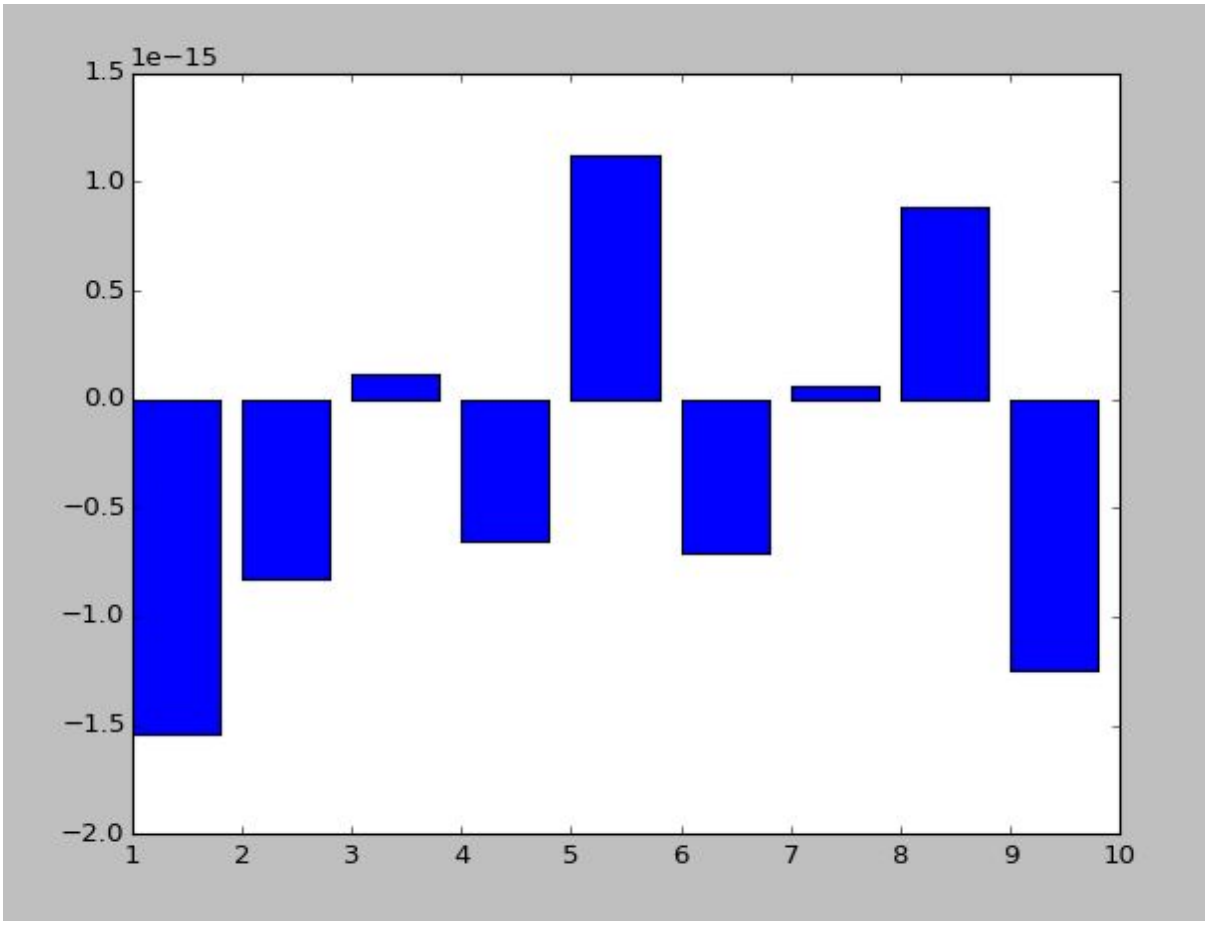
K=8


40
30
20
10
0
−10
−20
−40
−30
−20
−10
0
10
20

K=10


40
30
20
10
0
−10
−20
−40
−30
−20
−10
0
10
20

Do sprawdzenia jaka liczba klastrów będzie najbardziej odpowiednia w tym przypadku użyłam metody ‘gap statistic’. Poniżej znajduje się wykres optymalności liczby klastrów.



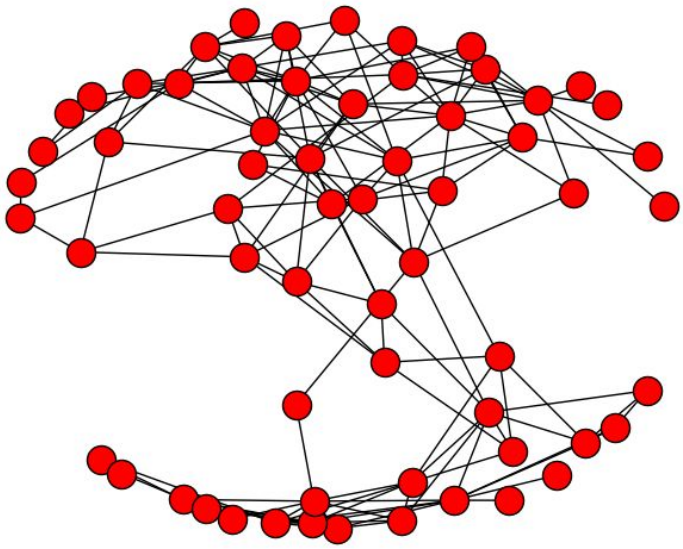
Optymalnym rozwiązaniem jest jak najmniejsza liczba ja spełniająca poniższą nierówność:

$$Gap(k) - Gap(k+1) + s_{k+1} \geq 0$$

Jak widać z powyższego wykresu i równania naszą optymalną ilością klastrów jest 5.

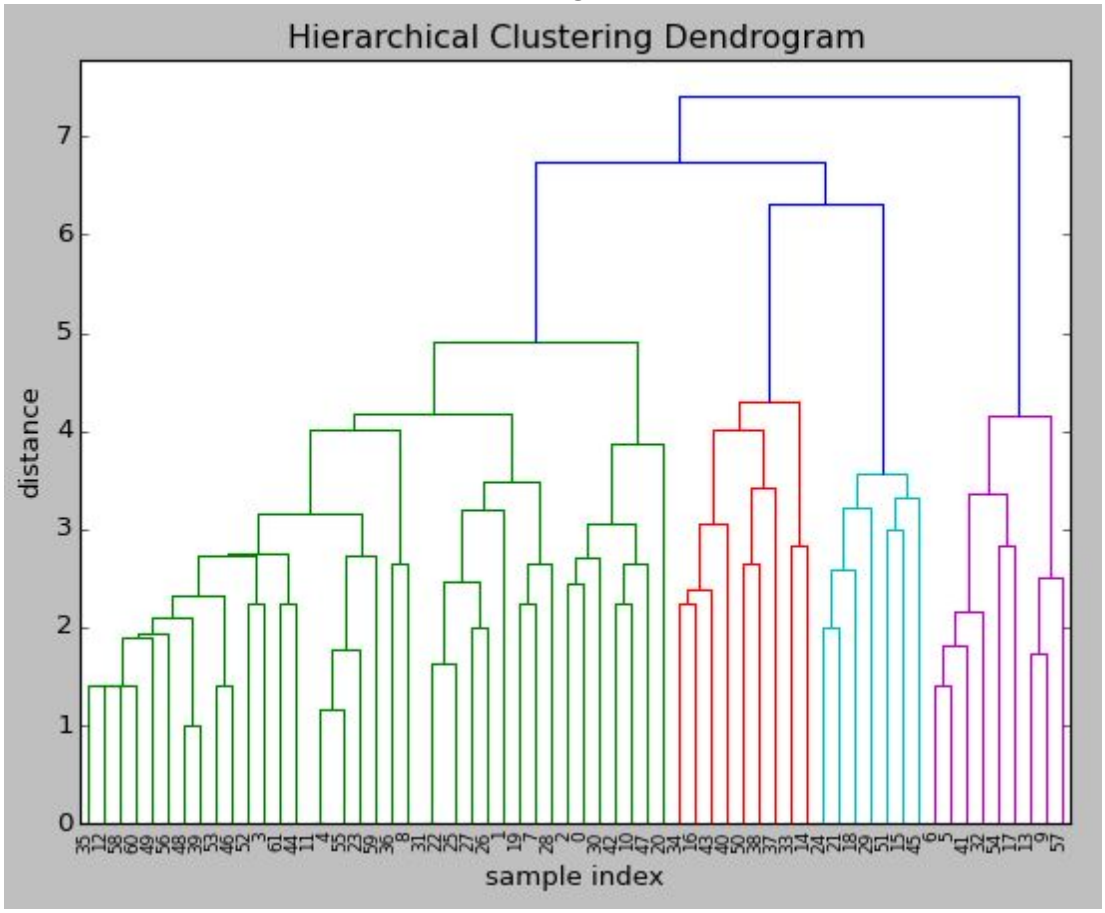
5. Klasteryzacja hierarchiczna

Wizualizacja grafu z zestawu dolphins.gml

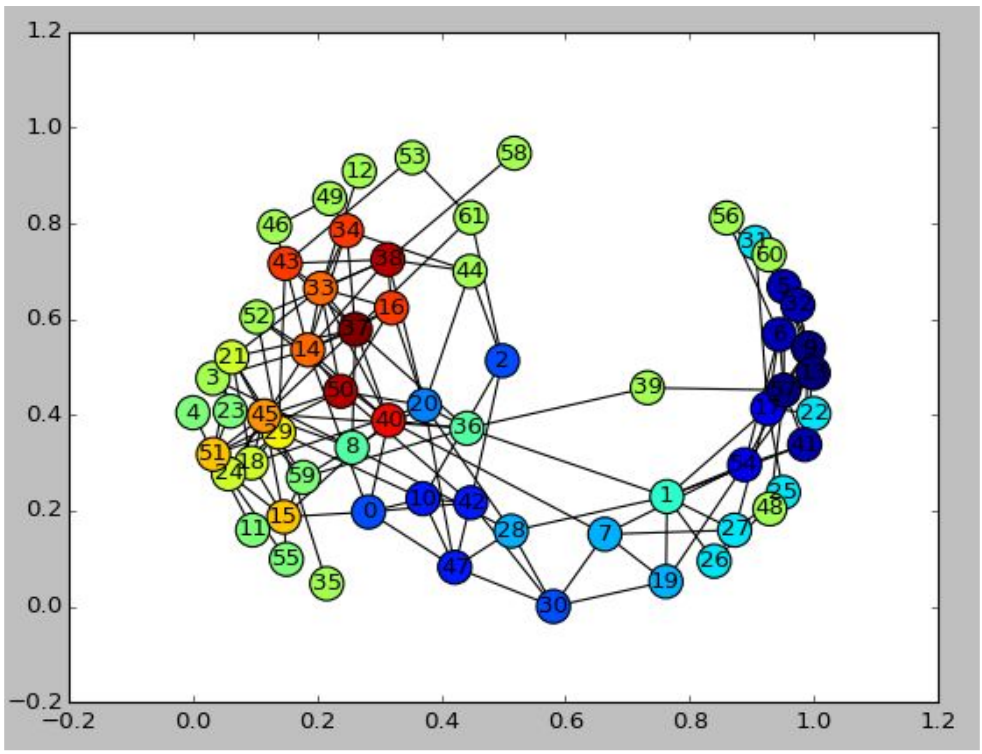


Odległość Euklidesowa zbudowana na podstawie macierzy adiacencji

Dendrogram

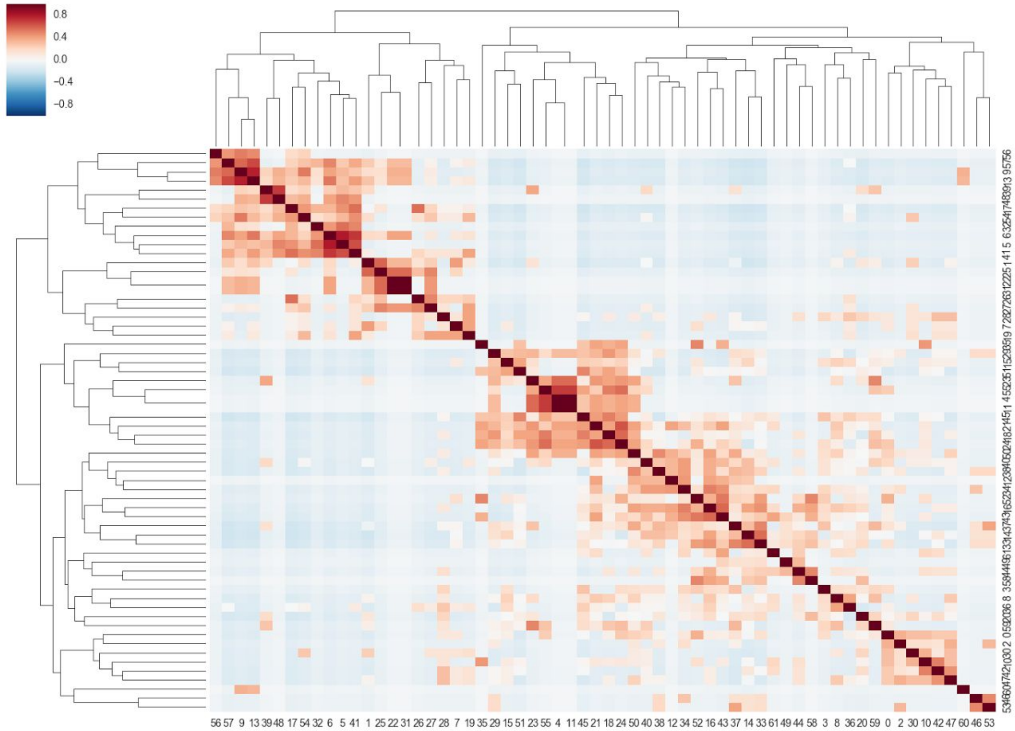


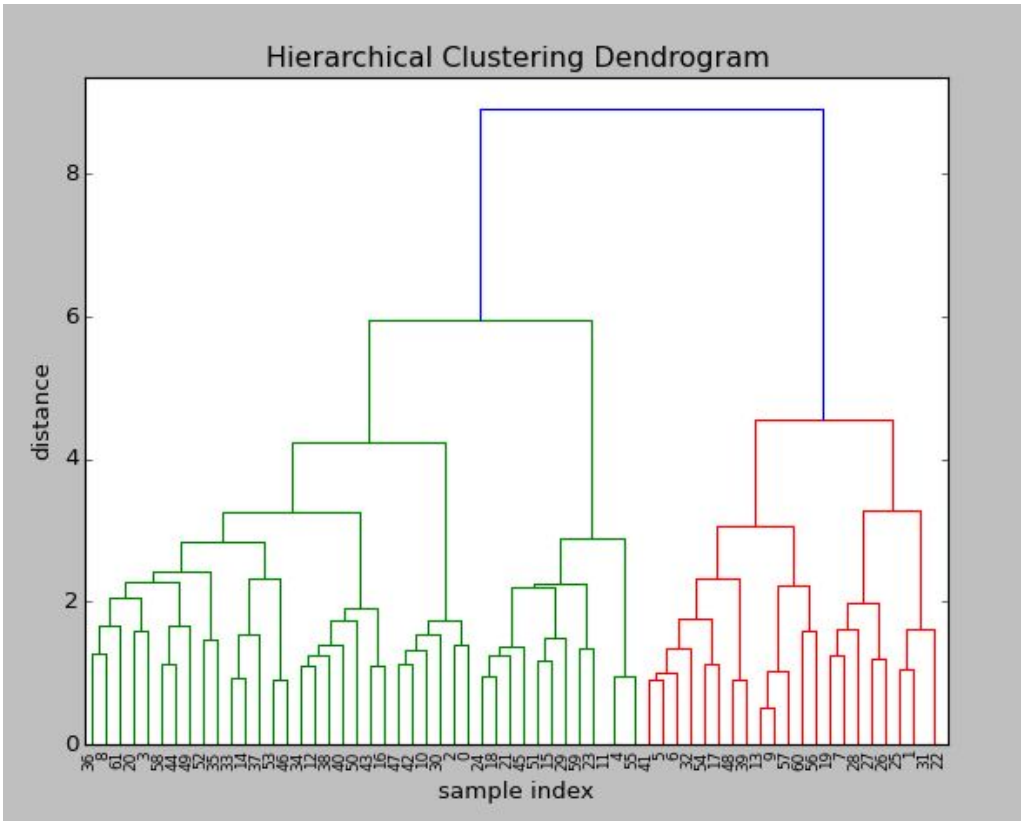
Graf



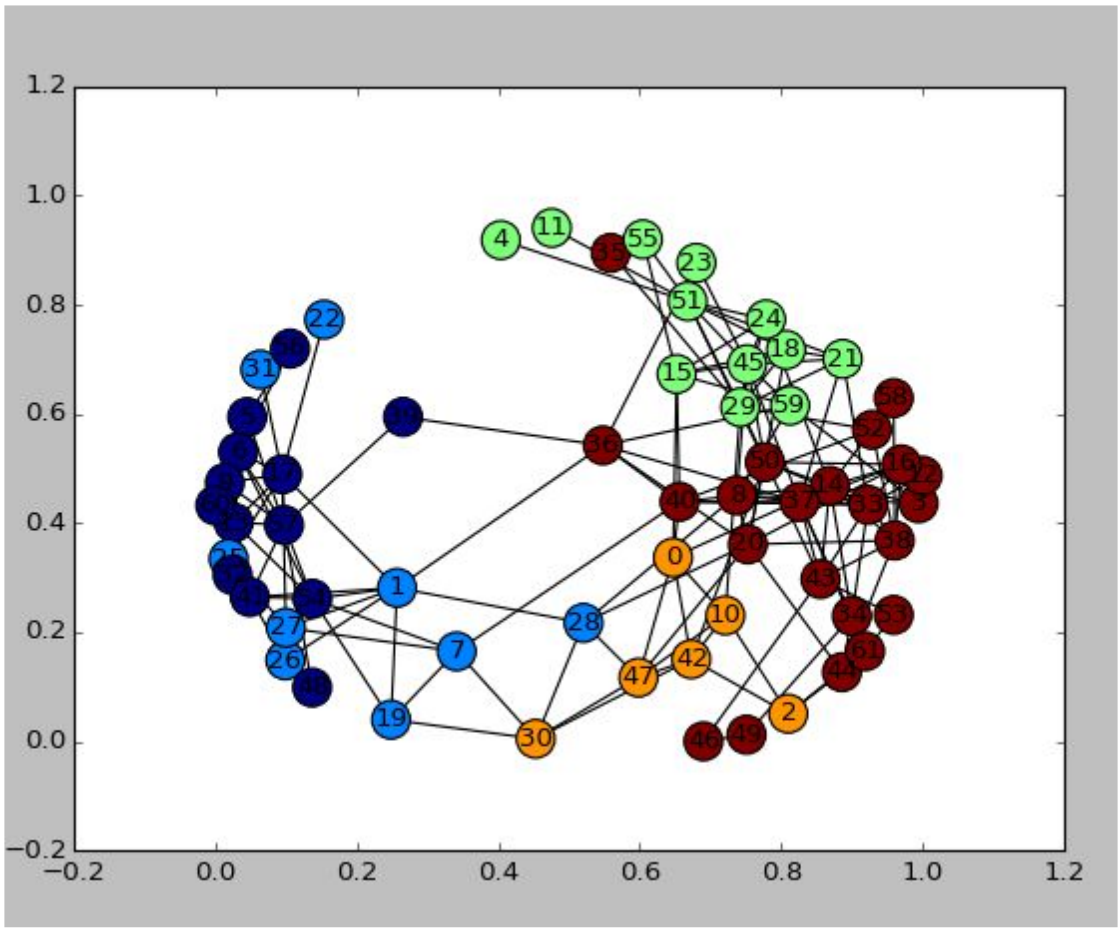
Korelacja między rzędami macierzy adiacencji

Dendrogram

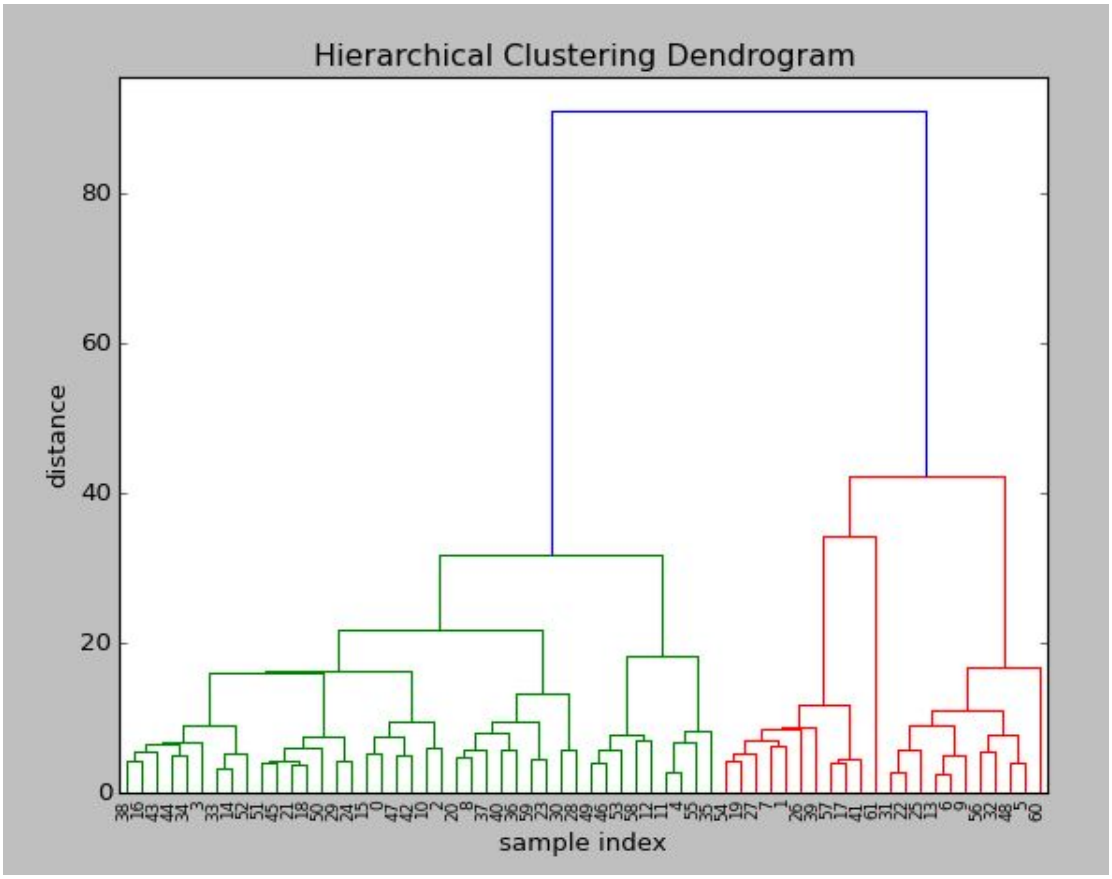




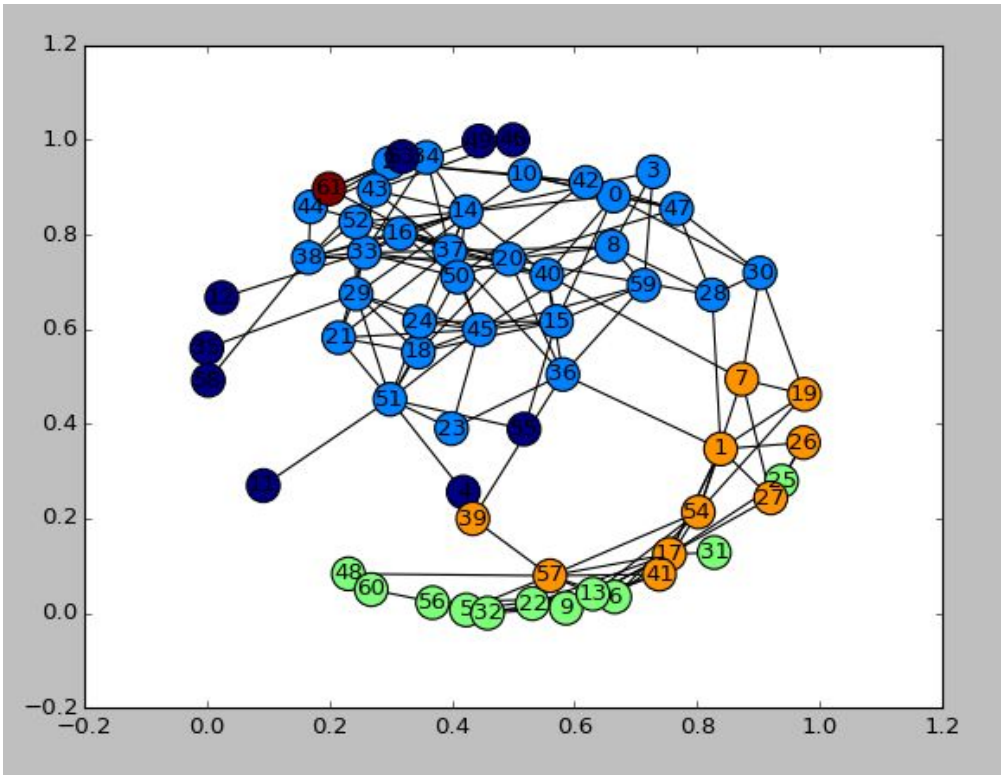
Graf


1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
4
11
55
23
51
24
18
21
15
45
29
59
58
52
50
16
14
40
8
37
53
5
20
38
0
43
1
10
27
28
54
53
61
26
7
42
47
44
2
19
30
46
49
22
31

Długość najkrótszej ścieżki pomiędzy wierzchołkami
Dendrogram

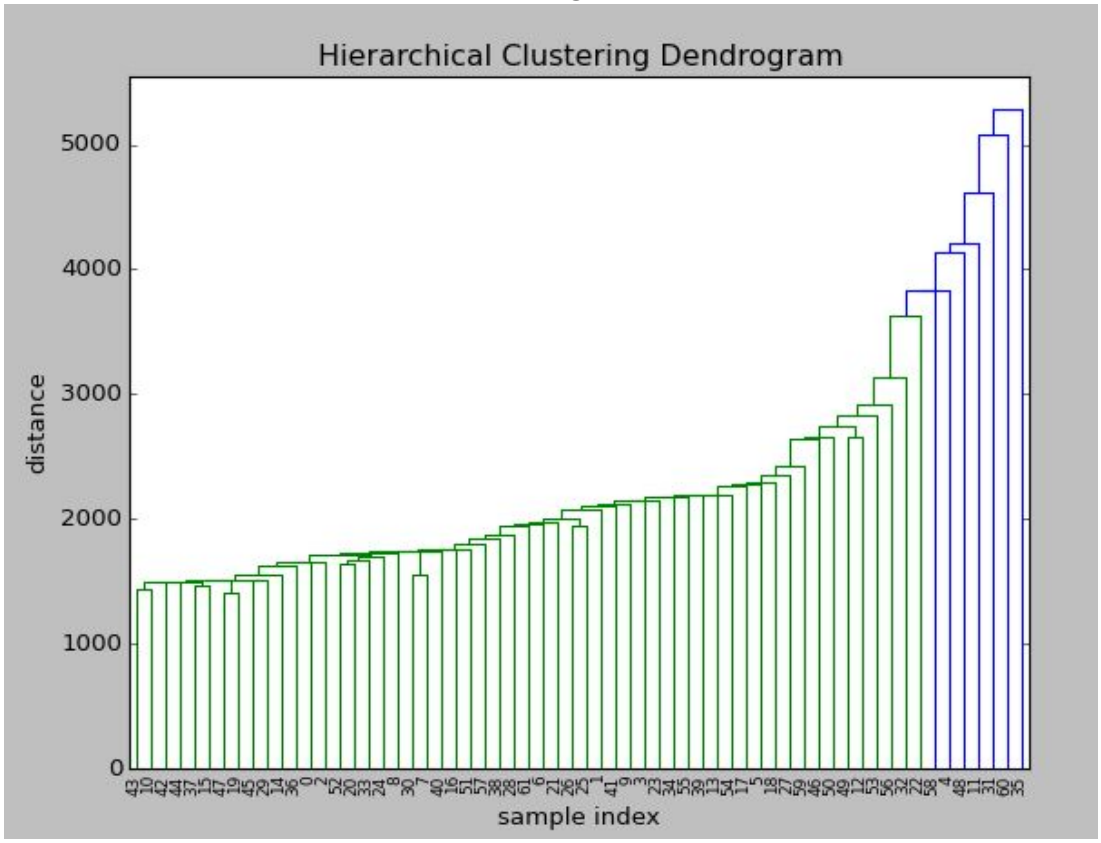


Graf

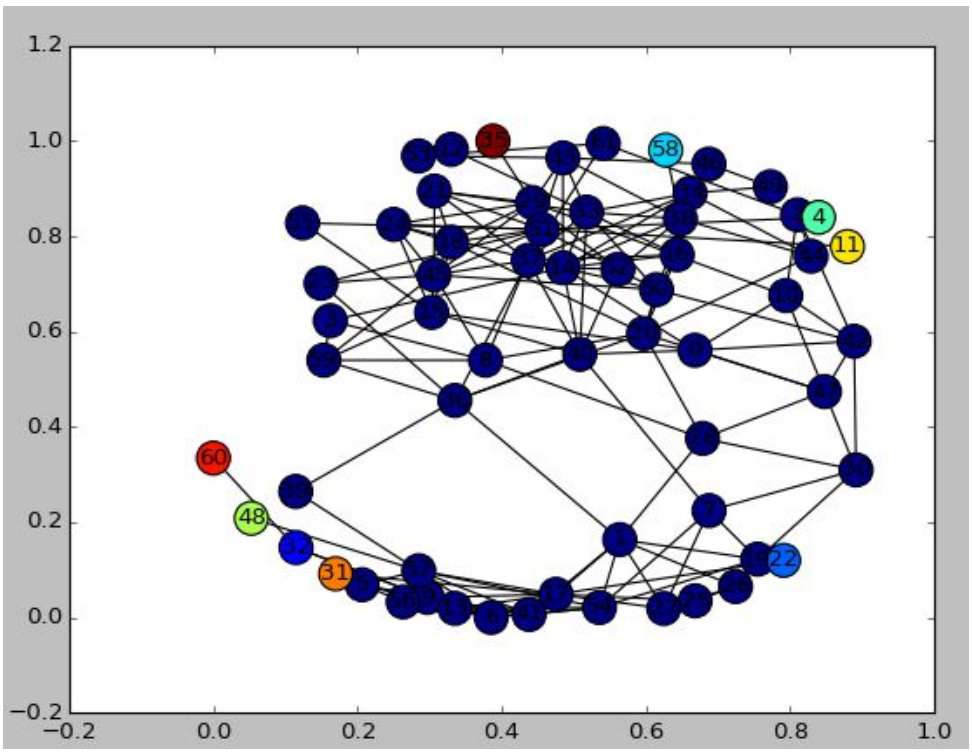


Zaimplementowane zostało ‘random walk’ z metryką commutte time

Dendrogram

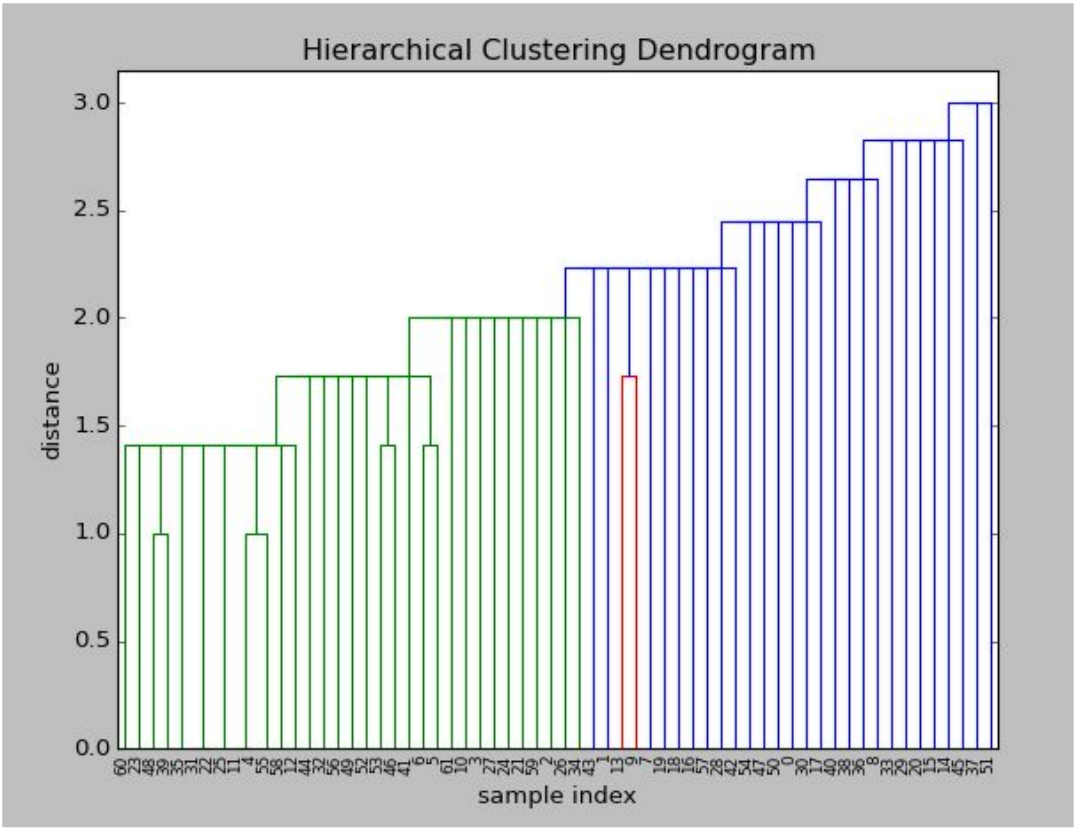


Graf

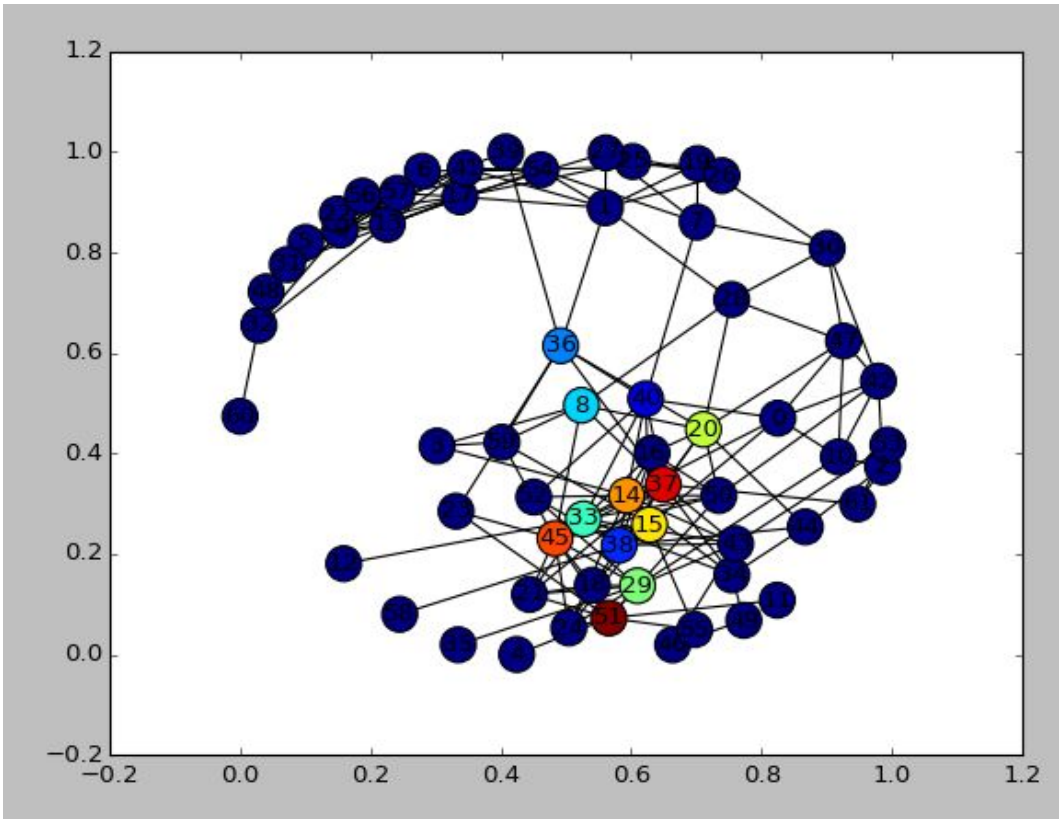


Single-link clustering

Dendrogram

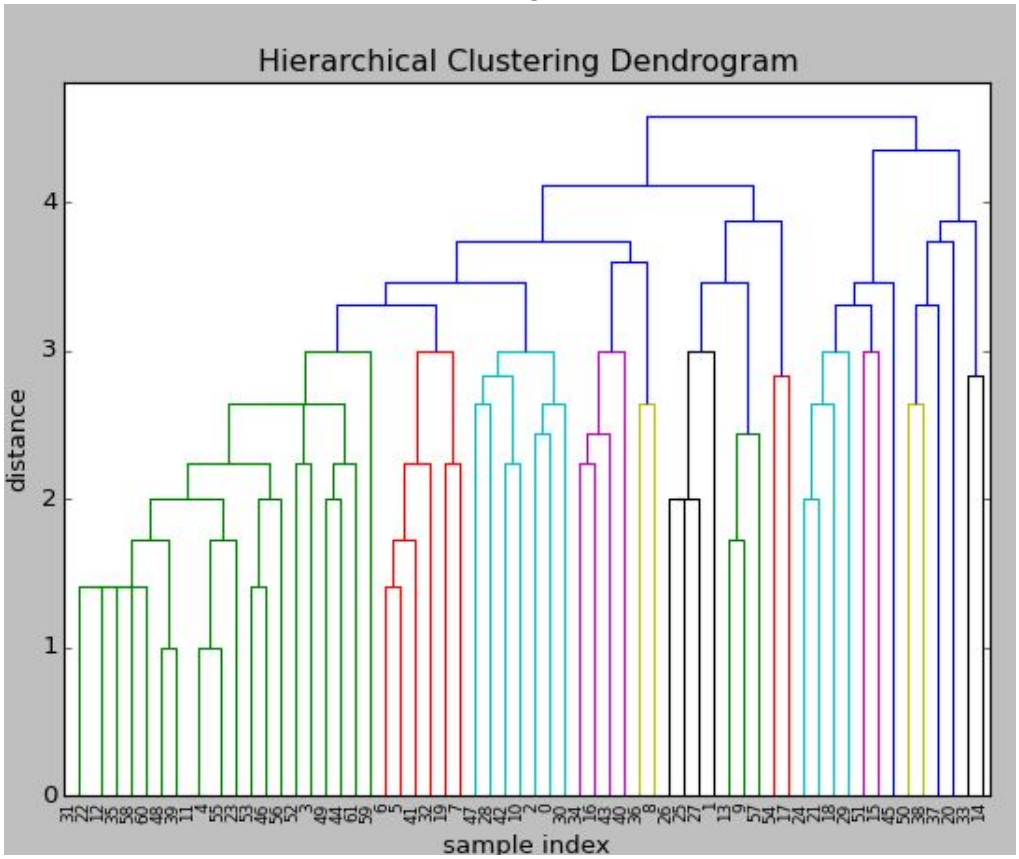


Graf

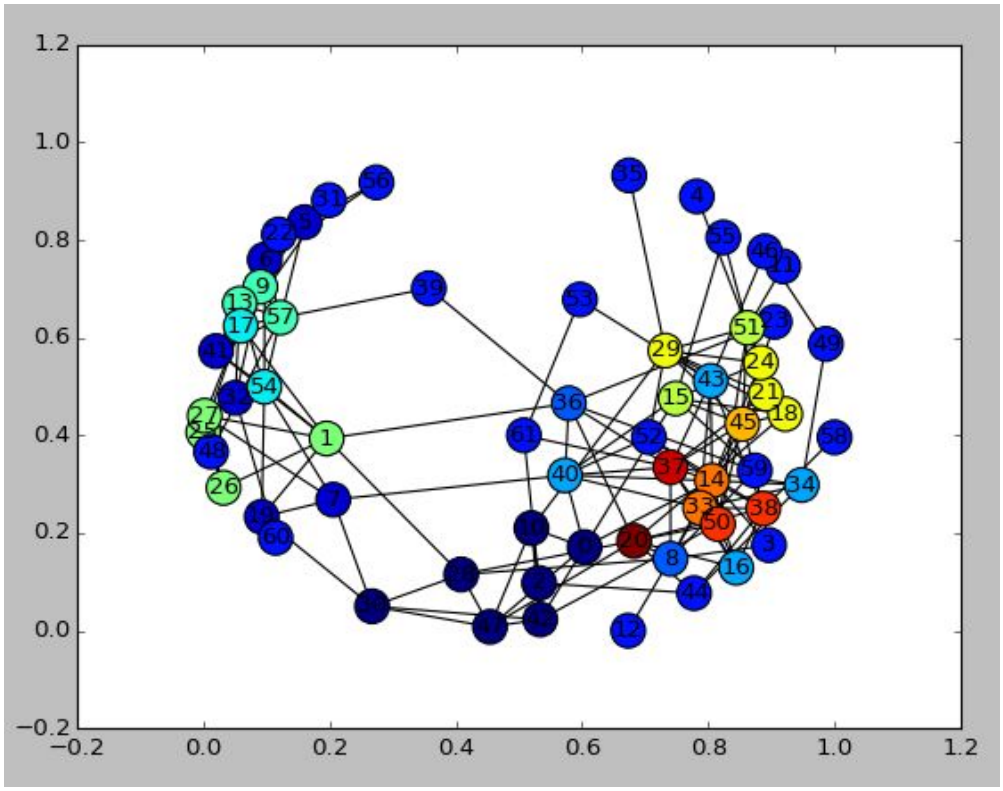


Complete-link clustering

Dendrogram

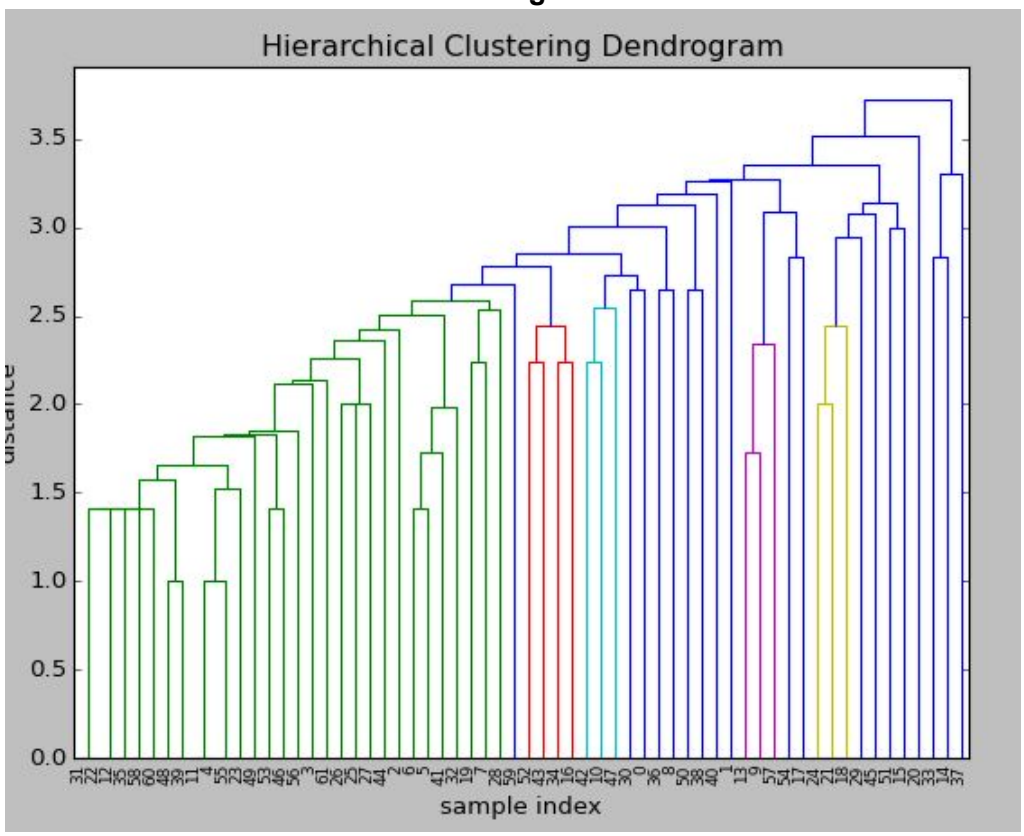


Graf

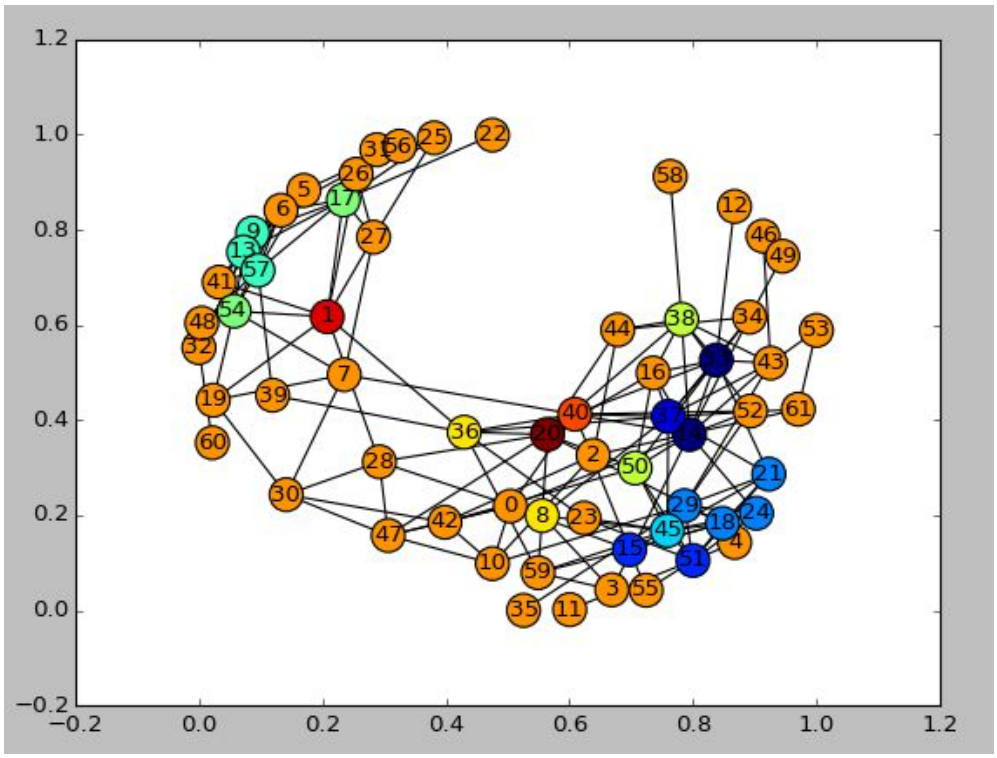


Average-link clustering

Dendrogram

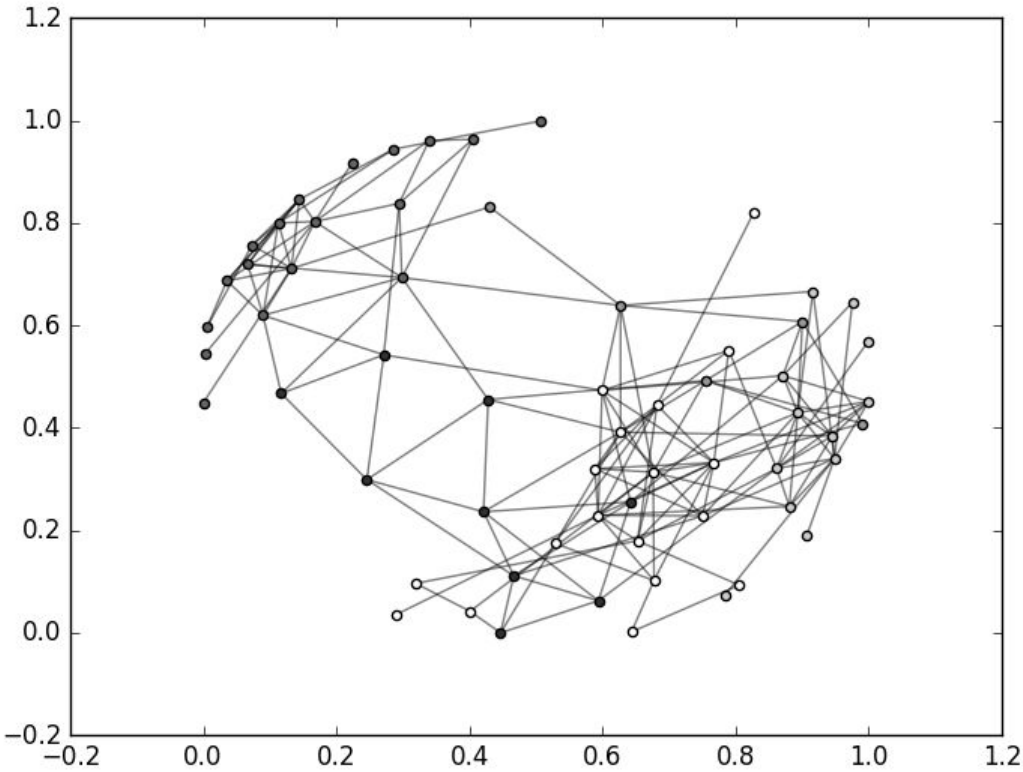


Graf



Metody community detection

Wykorzystana została metoda Louvaina



Bibliografia i wykorzystane narzędzia:

https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/
https://datasciencelab.wordpress.com/2013/12/27/finding-the-k-in-k-means-clustering/
https://bitbucket.org/taynaud/python-louvain
https://networkx.github.io/
http://seaborn.pydata.org/