Algorytm ROCK

Metody eksploracji danych w odkrywaniu wiedzy

Paweł Młyniec

28 lutego 2021

Spis treści

1 Cel zadania

Celem projektu jest implementacja algorytmu ROCK na podstawie pracy naukowej [1].

Praca powinna zawierać implementację algorytmu, testy na danych sztucznie wygenerowanych oraz na danych rzeczywistych. Powinny także być porównane czasy wykonania i wyniki w zależności od ustawionych parametrów algorytmu.

2 Charakterystyka algorytmu

Algorytm ROCK jest algorytmem grupującym.

Grupowanie polega na przypisaniu zbioru punktów do jednej kategorii na podstawie charakterystyki podobieństwa, lub odległości pomiędzy punktami. W jednym klastrze znajdują się elelementy do siebie podobne, w różnych klastrach natomiast elementy do siebie niepodobne.

Można wyróżnić dwa typy grupowania. Pierwszy z nich dzieli przestrzeń obserwacji na k klastrów tak by optymalizować pewną funkcję kryterialną. Dla danych numerycznych jest funkcja która będzie minimalizować odległości punktów klastra od jego centroidy.

Problemem w tym podejściu jest trudność w wyznaczeniu odległości pomiędzy obserwacjami a centroidą klastra w przypadku, gdy nie mamy do czynienia z danymi numerycznymi.

Drugim podejściem są algorytmy hirarchiczne. Takie algorytmy traktują początkowo każdą obserwację jako osobny klaster. Następnie klastry najbardziej do siebie podobne są łączne w jeden.

Algorytm ROCK należy do grupy algorytmów hierarchicznych. Opiera się na idei połączeń pomiędzy punktami, zamiast na metryce L_p^3 lub indeksie Jaccarda.

By wprowadzić definicję połączeń, zacznę do zdefiniowania sąsiedztwa punktów. Dane punkty są sąsiadami jeśli ich podobieństwo przekracza pewny próg. Podobieństwo może być zdefinowane odległością L_p lub indeksem Jaccarda lub inną funkcją podobieństwa.

Liczba połączeń pomiędzy punktami jest liczbą wspólnych sąsiadów pomiędzy danymi punktami. Punkty należą do jednego klastra jeśli mają duża liczbą wspólnych sąsiadów.

3 Opis implementacji

Implementacja opiera się na orginalnym artykule opisującym algortym ROCK [1]

Składa się z następujących klas:

- cluster - definiującej typ klastra oraz umożliwiającej rzutowanie punktów na klastry
- goodnes measure - określającej funkcję "dobroci"pomiędzy klastrami oraz wyznaczającej liczbę połączeń pomiędzy klastrami na podstawie macierzy połączeń
- link - wyliczającej macierz sąsiedztwa oraz na jej podstawie macierz połączeń
- heap wrapper - udostępniająca metody operowania na stertach
- rock - przeprowadzająca algorytm grupowania algorytmem ROCK

Zostały także zaimplementowane klasy pomocnicze:

- plotter - klasa umożliwiająca wizualizację wyników klastrowania
- reader - klasa odpowiadająca za odczyt danych z dysku

Dodatkowo w projekcie znajdują się testy jednostkowe w folderze *unittests*.

4 Instrukcja użytkowania

Program został napisany w języku Python 3.9. Aby uruchomić algorytm należy załączyć pakiet rock i następnie dla stworzonej klasy Rock z parametrami progu (t*hreshold*) do określenia miary podobieństwa punktów. Wartość progu powinna być z przedziału (0, 1). Wartość 1 mówi, że punkty są sąsiadami jeśli są takie same. Natomiast wartość 0 określa, że wszystkie punkty są sąsiadami.

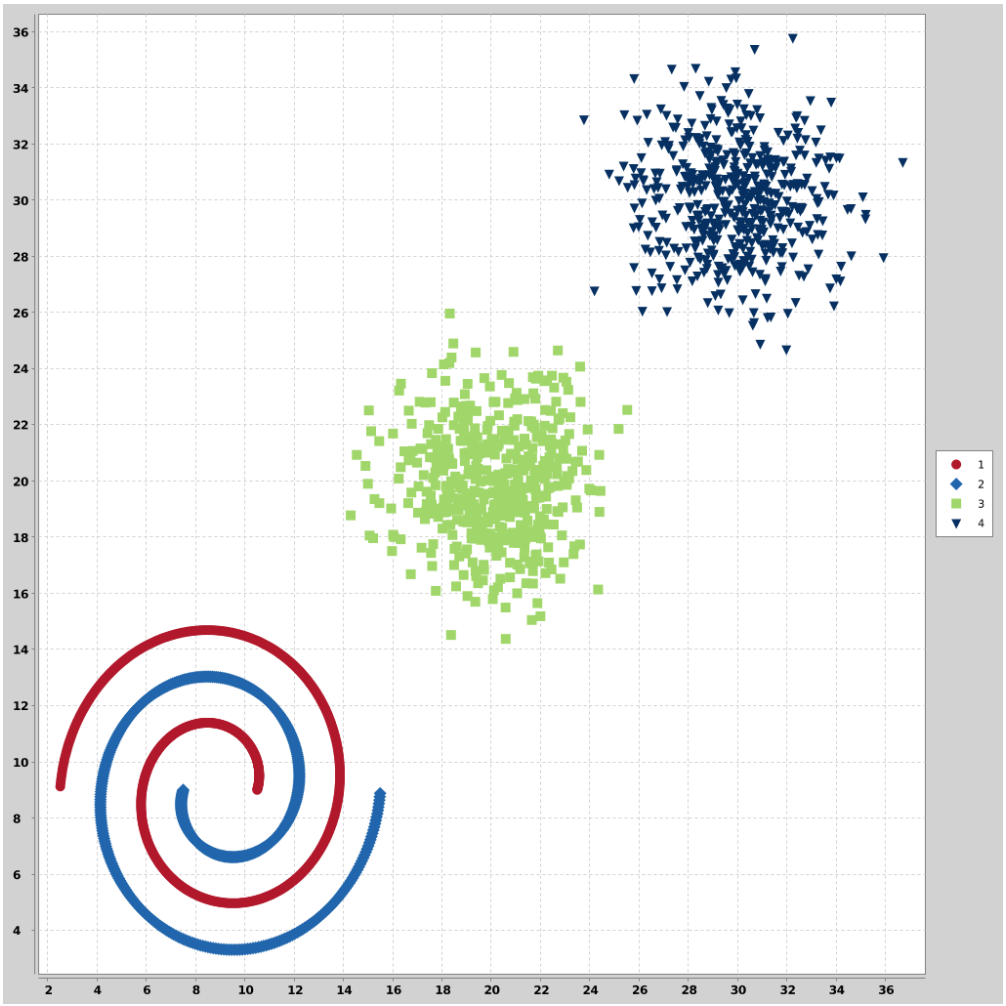
Do tak stworzonej klasy należy wywołać metodę *cluster*, która to dla danych i liczby docelowych klastrów zwróci klastry z indeksami punktów.

Testy jednostkowe zostały napisane z użyciem *pytest* i można je uruchomić w standardowy sposób.

5 Charakterystyka wykorzystywanych zbiorów danych

Pierwszym zbiorem dla którego były przeprowadzone testy jest zbiór 2sp2glob z [2].

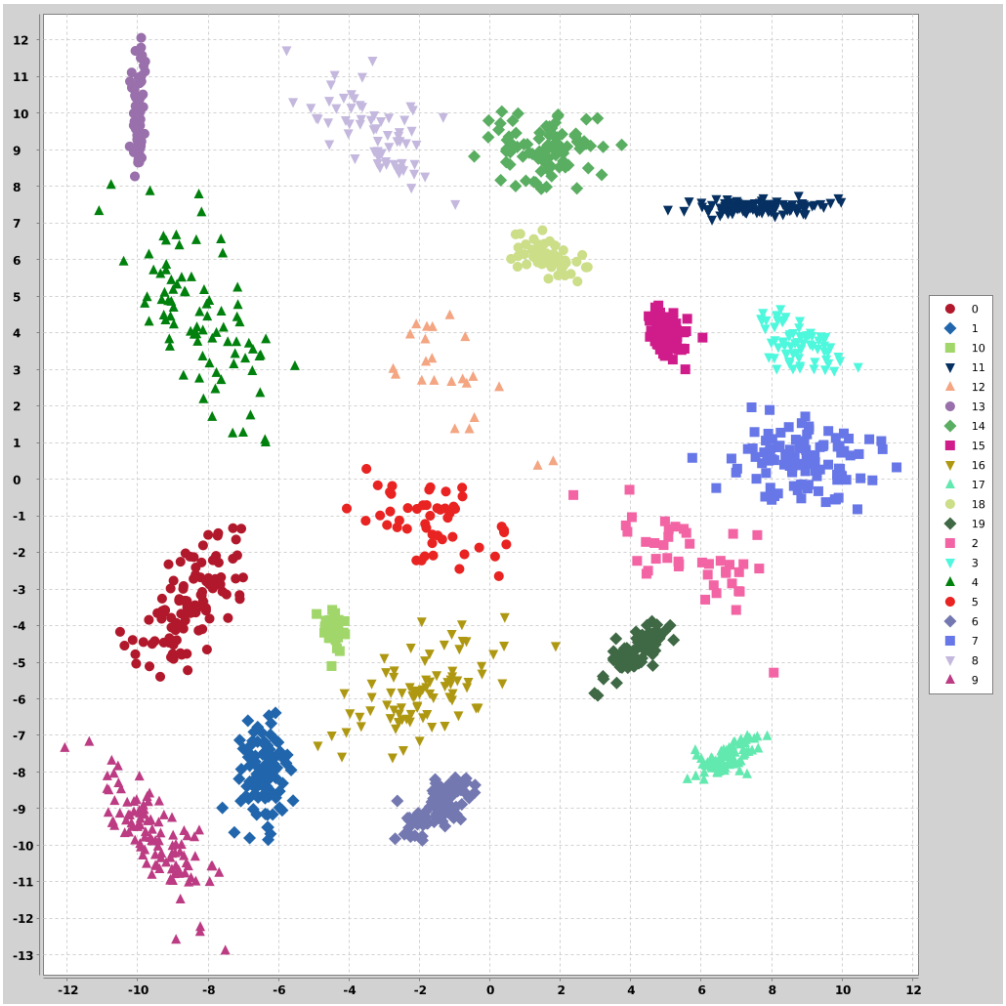
Dany zbiór posiada 2000 punktów o dwóch wymiarach (x i y) i przedstawiony został na poniższym rysunku (kolory odpowiadają docelowym klasom punktów)



Rysunek 1: Wykres przedstawiający pierwszy zbiór danych, 2sp2glob

Jak widać dany zbiór posiada cztery klastry które różnią się od siebie kształtem. Warto zwrócić uwagę na zbiory punktów w prawym górnym rogu, które to posiadają obserwacje odstające, mogące być problematyczne do poprawnego pogrupowania

Drugim testowanym zbiorem był zbiór 2d20C z [3]. Posiada on 20 docelowych klastrów, a liczba obserwacji wynosi 2000. Zbiór dla poklasyfikowanych danych wgląda następująco:



Rysunek 2: Wykres przedstawiający pierwszy zbiór danych, 2d-20c-no

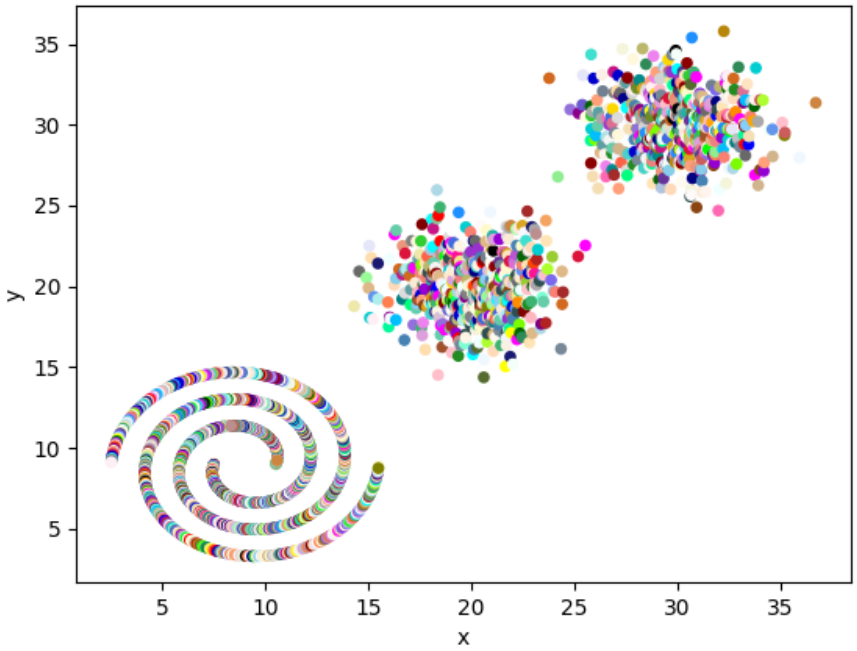
Kolejnym zbiorem będzie zbiór cech grzybów z strony [4]. Zbiór ten zawiera 8124 obserwacji, każda posiadająca 22 cechy. Zadaniem dla tego zbioru będzie pogrupowanie grzybów na jadalne i niejadalne. Stosunek tych klas wynosi 4208 do 3916. Dokładny opis danych znajduje się w źródle.

6 Wyniki eksperymentów

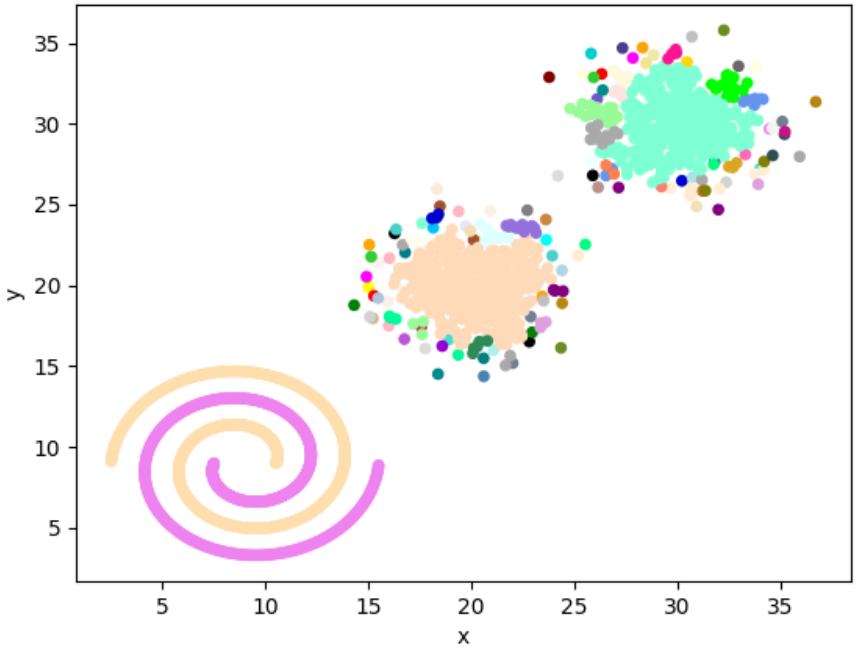
Pierwszym testem był test na zbiorze danych 2*sp*2*glob*.

Algorytm ROCK został przetestowany dla wartości progu równych [1, 0.99, 0.98, 0.95, 0.9] oraz zadanej docelowej liczbie klastrów wynoszącej 4

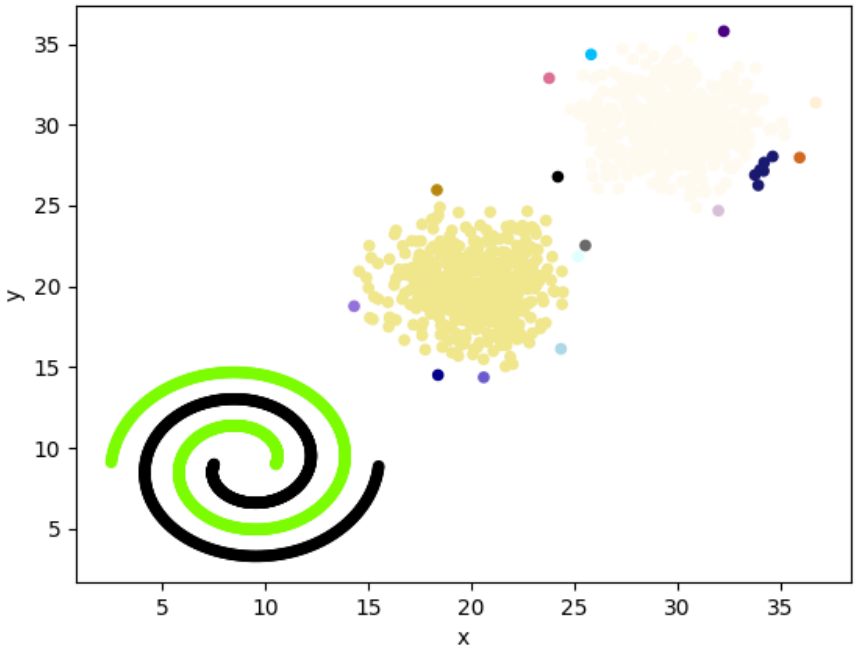
Otrzymano następujące rezultaty



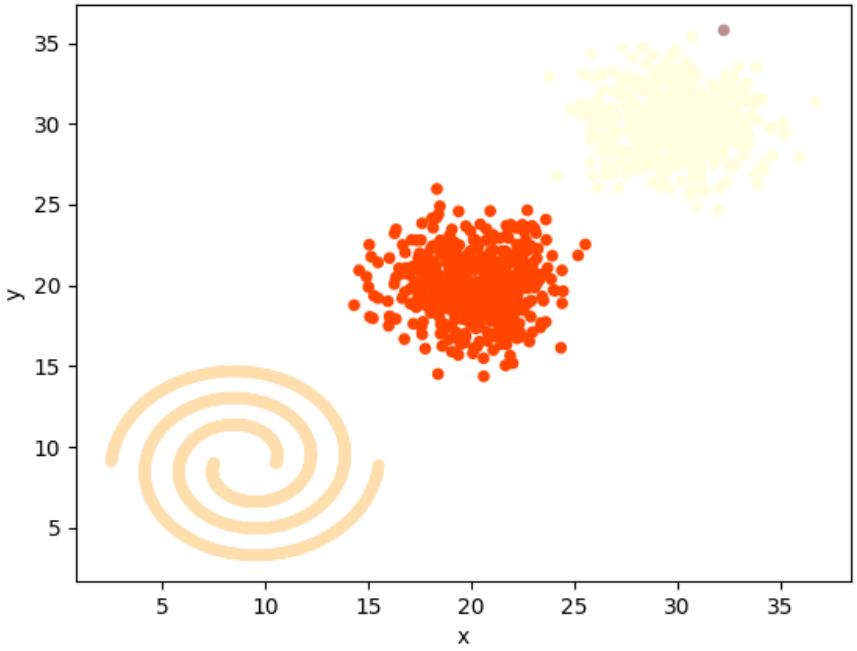
Rysunek 3: Wykres grupowania dla wartości progu 1



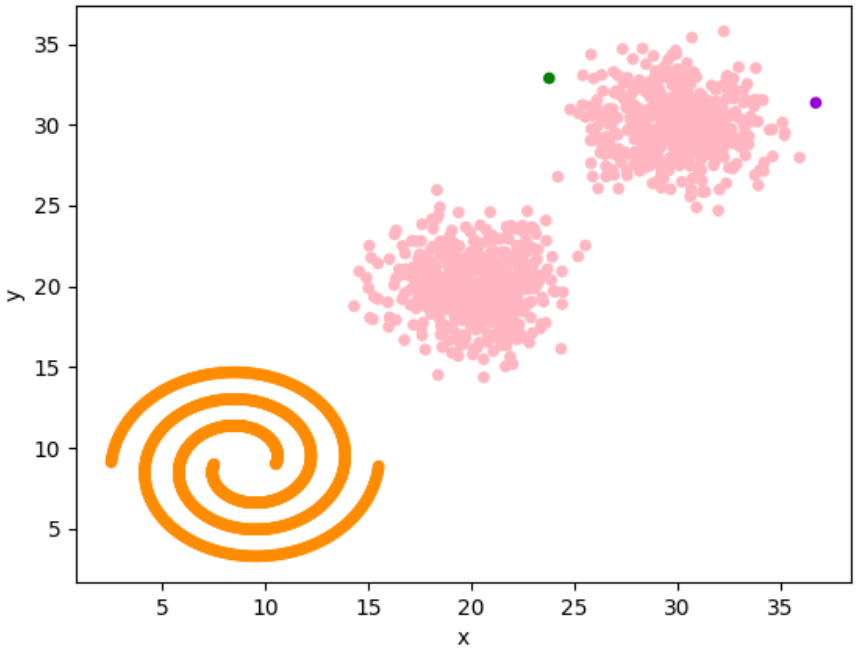
Rysunek 4: Wykres grupowania dla wartości progu 0.99



Rysunek 5: Wykres grupowania dla wartości progu 0.98



Rysunek 6: Wykres grupowania dla wartości progu 0.95



Rysunek 7: Wykres grupowania dla wartości progu 0.9

wartość progu	Liczba klastrów	Średnia czas wykonania
1	2000	14s
0.99	127	26s
0.98	20	78s
0.95	4	372s
0.9	4	433s

Tablica 1: Liczba klastrów i czas dla kolejnych wartości progu, zbiór 2d20C

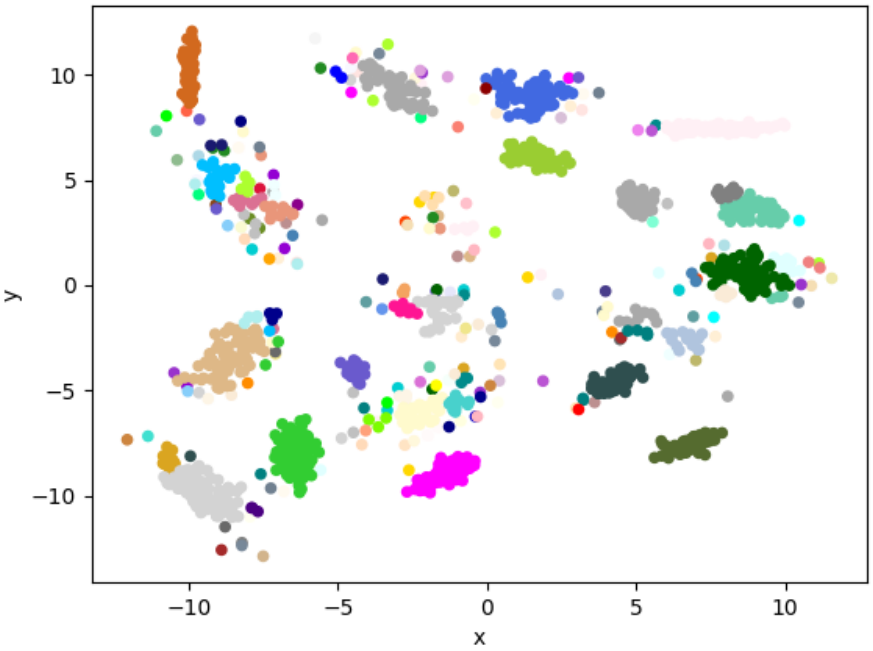
Jak widać odpowiednie ustawienie parametru decydującego o sąsiedztwie punktów decyduje o liczbie klastrów do których zostały przypisane obserwacje. Dla zbyt dużych wartości, każdy punkt jest osobnym klastrem (co zresztą wynika z definicji progu). Dla zbyt małych wartości punkty, które powinny być w osobnych klastrach są przypisane do jednego. Najlepsze wyniki wyszły dla wartości 0.98. Wprawdzie stworzona tam 20 klastrów, poprawie pogrupowano większość punktów poza obserwacjami odstającymi.

Widać też znacząco rosnący czas wykonania wraz z zmniejszającym się parametrem progu.

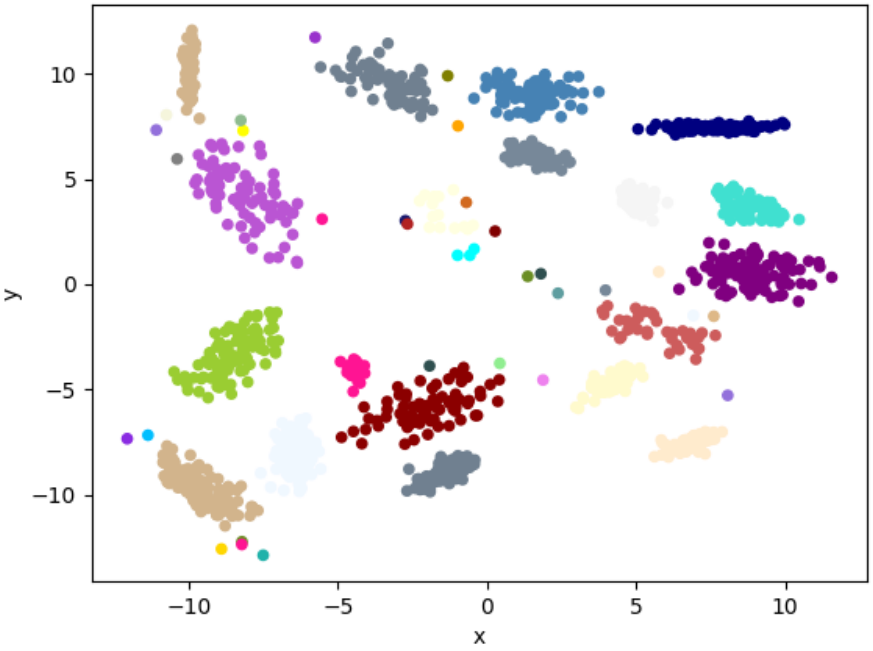
Drugi test został przeprowadzony na zbiorze 2d-20c-no [3].

Liczba klastrów docelowych została ustawiona na 20 a wartości progu były testowane z zbioru [0.99, 0.98, 0.95, 0.9].

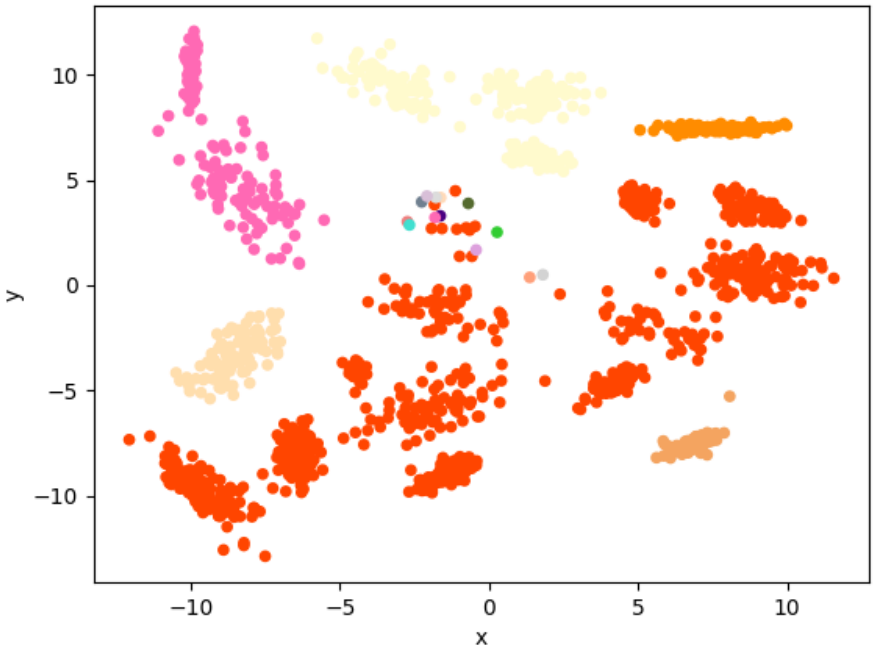
Wyniki przedstawiono na kolejnych obrazkach.



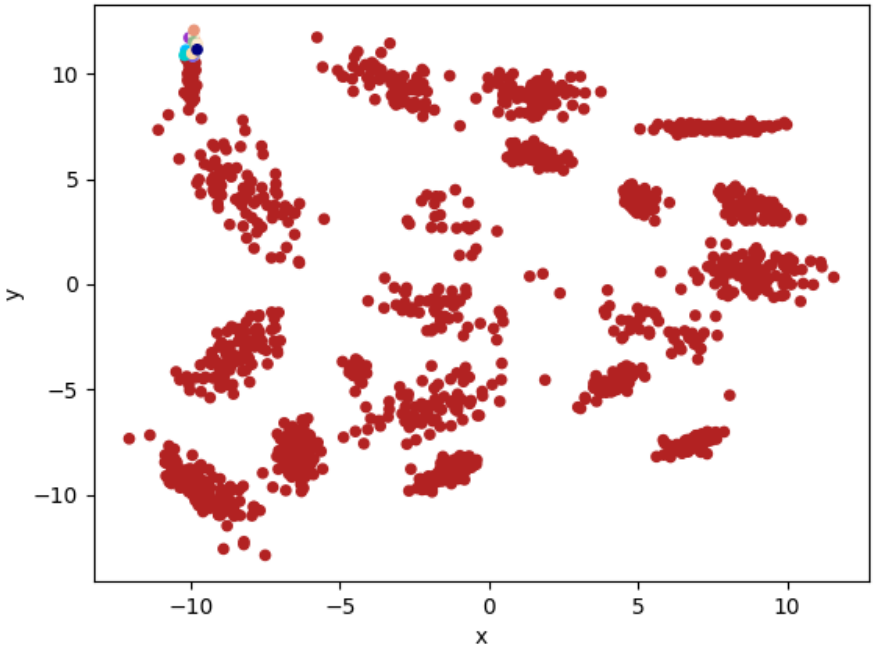
Rysunek 8: Wykres grupowania dla zbioru 2D20C z wartością progu 0.99



Rysunek 9: Wykres grupowania dla zbioru 2D20C z wartością progu 0.98



Rysunek 10: Wykres grupowania dla zbioru 2D20C z wartością progu 0.95



Rysunek 11: Wykres grupowania dla zbioru 2D20C z wartością progu 0.9

Liczba klastrów do których zostały pogrupowane dane z zbioru:

wartość progu	Liczba klastrów	Średnia czas wykonania
0.99	233	12s
0.98	51	15s
0.95	20	27s
0.9	20	266s

Tablica 2: Liczba klastrów i czas dla kolejnych wartości progu, zbiór 2d20C

Najlepsze rezultaty otrzymano dla wartości progu 0.98. Poprawnie pogrupowało większość punktów poza obserwacjami odstającymi w liczbie 31 obserwacji.

Kolejny test został przeprowadzony na realnym zbierze danych posiadającym kategoryczne cechy dotyczące grzybów, agaricus-lepiota [4].

Zadaniem algorytmu było pogrupowanie grzybów w klastry z grzybami jadalnymi i trującymi.

Dla danego problemu funkcją decydującą o dystansie pomiędzy punktami jest ilość wspólnych cech dzielona przez ilość wszystkich cech.

Liczba docelowych klastrów została ustawiona na 20, a wartość progowa na 0.8, zgodnie z doświadczeniami przeprowadzonymi w [1].

Czas wykonania algorytmu to 9055s a wyniki grupowania są zaprezentowane w poniższej tabeli:

suma obserwacji w klastrze	Liczba grzybów jadalnych w klastrze	liczba grzybów trujących w klastrze
1	0	1
1	0	1
1	0	1
1	0	1
1728	0	1728
1	0	1
1	0	1
1	0	1
1	0	1
1296	0	1296
16	16	0
768	768	0
36	0	36
48	48	0
288	288	0
32	0	32
192	192	0
3656	2848	808
48	48	0
8	0	8

Tablica 3: Pogrupowanie grzybów jadalnych dla zbioru agaricus-lepiota, parametr progu 0.8

Algorytm dość dobrze poradził sobie z przypasowaniem obserwacji grzybów do kategorii jadalne i trujące poza jednym klastrem gdzie liczba grzybów trujących wynosi 808 a jadalnych 2848.

Został także przeprowadzony drugi test z parametrem progu 0.9 oraz z dwoma docelowymi klastrami. Wyniki są przestawione na poniższej tabeli:

suma obserwacji w klastrze	Liczba grzybów jadalnych w klastrze	liczba grzybów trujących w klastrze
1728	1728	0
768	768	0
1296	0	1296
192	192	0
7048	704	0
1728	0	1728
192	192	0
256	0	256
288	288	0
36	0	36
192	0	192
48	48	0
16	16	0
105	32	72
96	96	0
288	0	288
32	0	32
96	96	0
56	48	8
8	0	8
8	0	8

Tablica 4: Pogrupowanie grzybów jadalnych dla zbioru agaricus-lepiota, parametr progu 0.9

W tym przypadku algorytm dobrze pogrupował prawie wszystkie obserwacje. Istnieje tylko jedna klasa do której przyporządkował grzyby jadalne i trujące.

7 Wnioski

Algorytm ROCK dość dobrze radzi sobie z pogrupowaniem danych zarówno numerycznych jak i kategorycznych. Problematyczne dla niego są wartości odstające, lecz prawdopodobnie poprawne ich zaklasyfikowanie wymagałoby dokładnego wyznaczenia wartości progu.

Widać też znacząco dłuższy czas wykonania wraz z mniejszym parametrem progu, co wiąże się z większą macierzą sąsiedztwa i większą ilością punktów które są dla siebie sąsiadami.

Algorytm nie został przyrównany do istniejących implementacji, ponieważ znaleziona implementacja algorytmu ROCK [5] okazała się nie być implementacja tego algorytmu, ponieważ bazowała tylko i wyłącznie na odległości pomiędzy punktami, a nie liczbą połączeń a co za tym idzie wspólnych sąsiadów obserwacji.

8 Bibliografia

[1] Guha S., Rastogi R., Shim K., *ROCK: A robust clustering algorithm for categorical attributes*, Proceedings of the International Conference on Data Engineering, Sydney 1999, pp. 512–521

[2] 2sp2glob *https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/2sp2glob.ar*

[3] Zbiór danych 2d20c *https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/2*
20c-no0.arff

[4] Mashroom dataset *https://archive.ics.uci.edu/ml/datasets/mushroom*

[5] PyClustering ROCK *https://pyclustering.github.io/docs/0.8.2/html/d8/dde/classpyclustering*$_{11}$*cluster*$_{11}$*rock*$_{11}$*rock.html*