Konrad Dudek 26.01.2021

Język Python - Projekt zaliczeniowy

Grupowanie punktów na dwuwymiarowej płaszczyźnie za pomocą algorytmu Kruskala oraz minimalnego drzewa rozpinającego

Uruchamianie programu:

python3 clustering.py [nazwa pliku csv z danymi] [liczba klastrów na jaką chcemy podzielić dane]

Działanie programu:
Program jako argument przyjmuje plik .csv z danymi oraz liczbę klastrów na jaką ma go podzielić. Do wizualizacji wyników użyłem biblioteki matplotlib, która “koloruje” wierzchołki zgodnie z przynależnością do danego klastra. Istnieje ograniczenie co do ilości klastrów na jakie chcemy podzielić nasze dane, które wynika z ilości kolorów. Plik csv powinien zawierać współrzędne x oraz y punktów.

Projekt korzysta z 4 klas:

1. class Vertex – klasa wierzchołka - przechowuje informacje na temat wierzchołków, ich współrzędne x, y oraz unikany dla każdego identyfikator.
2. class Edge – klasa krawędzi – przechowuje informacje na temat jakie wierzchołki łączy ze sobą krawędź oraz wagę krawędzi, czyli odległość euklidesową pomiędzy dwoma punkami.
3. class Graph – klasa grafu – zawiera informację o wszystkich wierzchołkach oraz krawędziach. Przechowuje ona również liczbę klastrów na jaką program będzie dzielił dane. Znajduje się w niej funkcja read_graph, która odpowiada za wczytanie danych z pliku csv. Podczas tego wczytywania zapisywane są informacje o współrzędnych każdego punktu oraz wyliczane wagi wszystkich krawędzi.
4. Class UnionFind – realizuje operacje dla zbiorów rozłącznych

Pozostałe funkcje programu:

- Kruskal – najważniejsza funkcja która realizuje działanie algorytmu Kruskala zmodyfikowanego w odpowiedni sposób tak aby spełniał warunki projektu.
- draw_result – funkcja, która wizualizuje wyniki działania algorytmu, wykorzystane są w niej funkcje z biblioteki matplotlib.

Wejście:

- X: zbiór punktów w przestrzeni dwumiarowej
- K: ilość grup na jakie należy podzielić zbiór danych

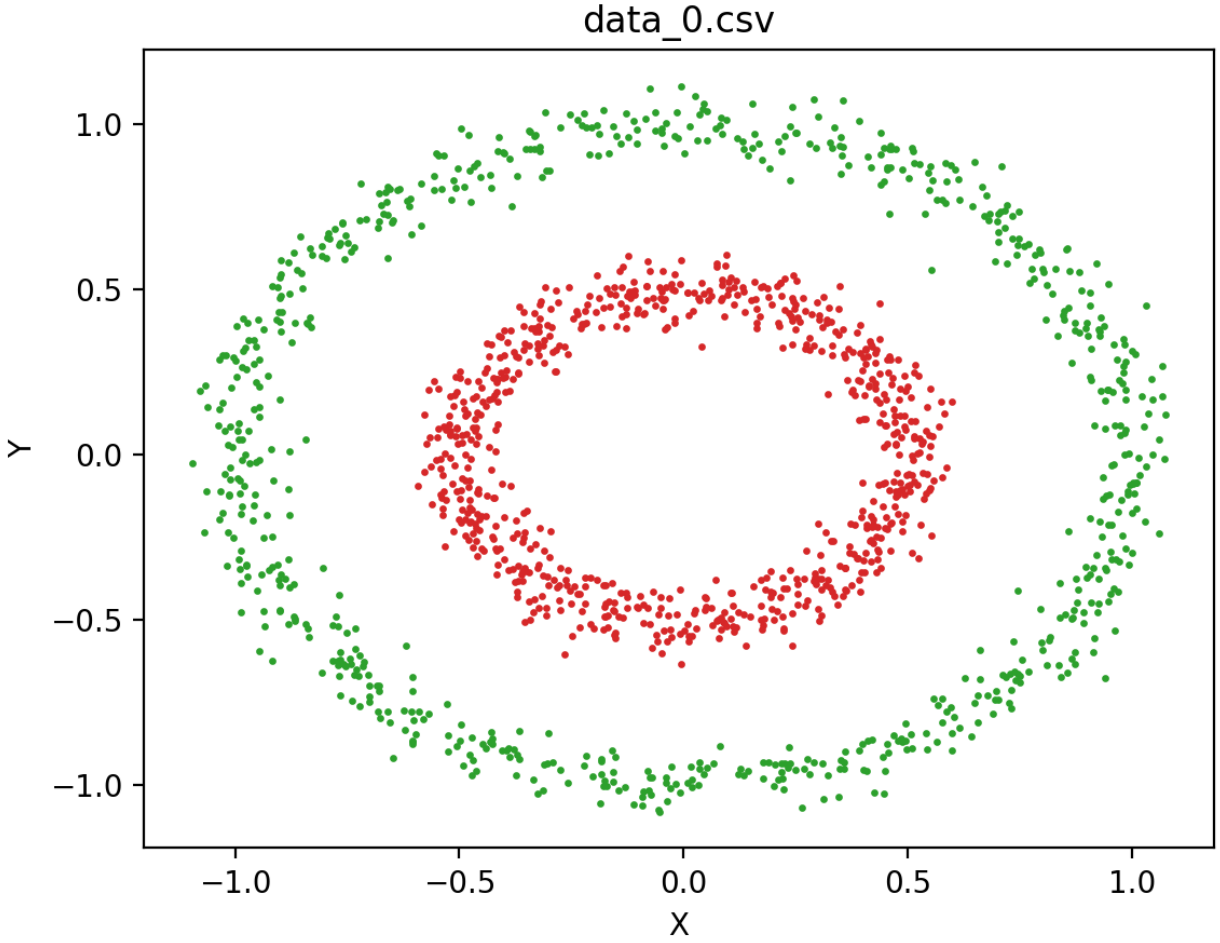
Wyjście:

Wykres ilustrujący pogrupowane dane.

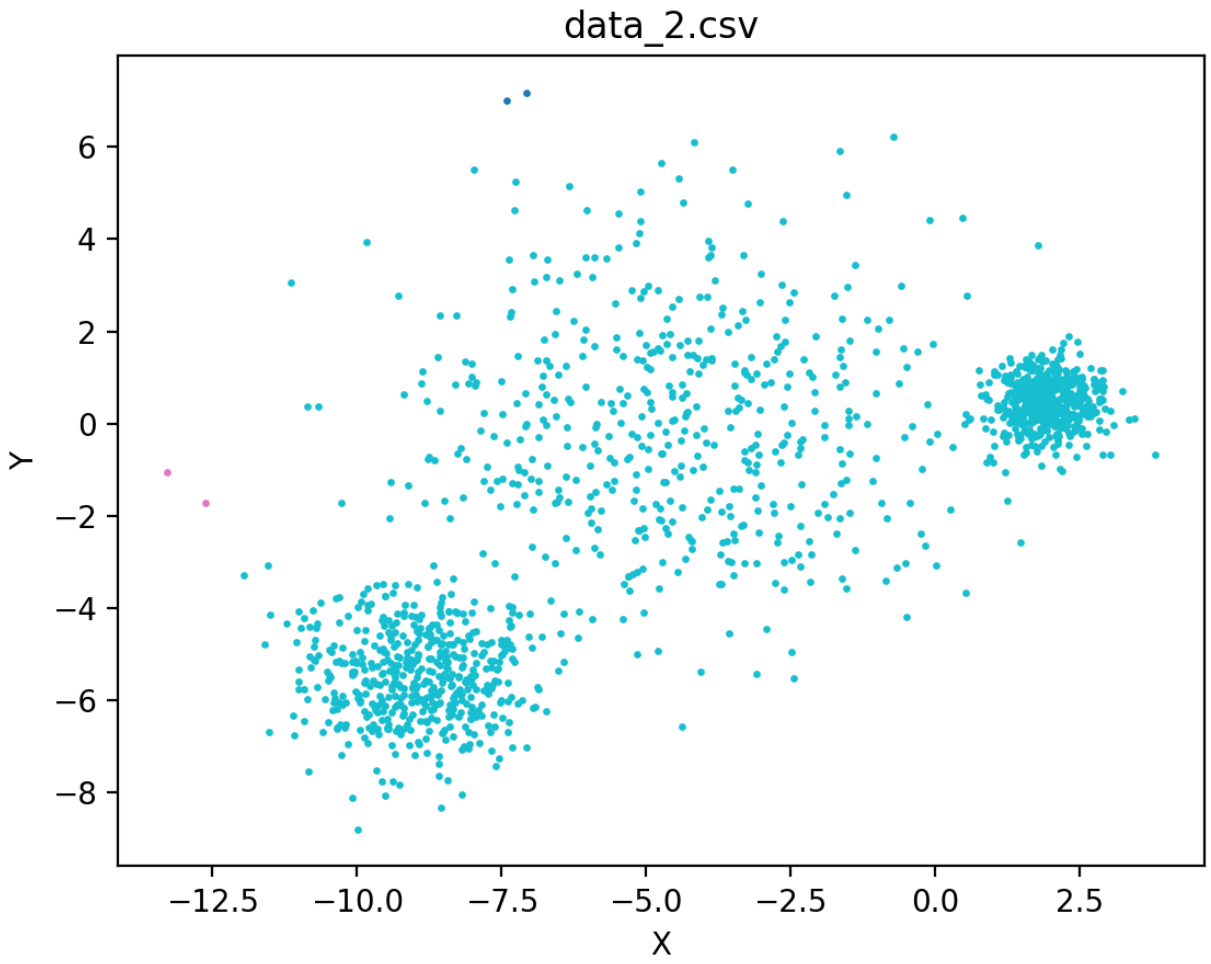
Działanie algorytmu:
Algorytm bazuje na budowie minimalnego drzewa rozpinającego za pomocą algorytmu Kruskala. Przekazane do programu dane będą reprezentowane jako graf. Punkty określają położenie wierzchołków grafu na przestrzeni dwuwymiarowej. Wagami w grafie będą odległości euklidesowe między punktami. Na początku wyliczane są wszystkie możliwe połączenia pomiędzy wierzchołkami a następnie usuwane tak aby utworzyć minimalne drzewo rozpinające. Podczas budowy minimalnego drzewa rozpinającego algorytm zatrzymuje swoje działanie w momencie kiedy stworzona już będzie odpowiednia liczba klastrów. Kiedy podział na odpowiednie klastry jest już skończony program zgodnie z tym podziałem tworzy wykres z punktami.

Wizualizacja działania algorytmu dla poszczególnych przykładowych danych:

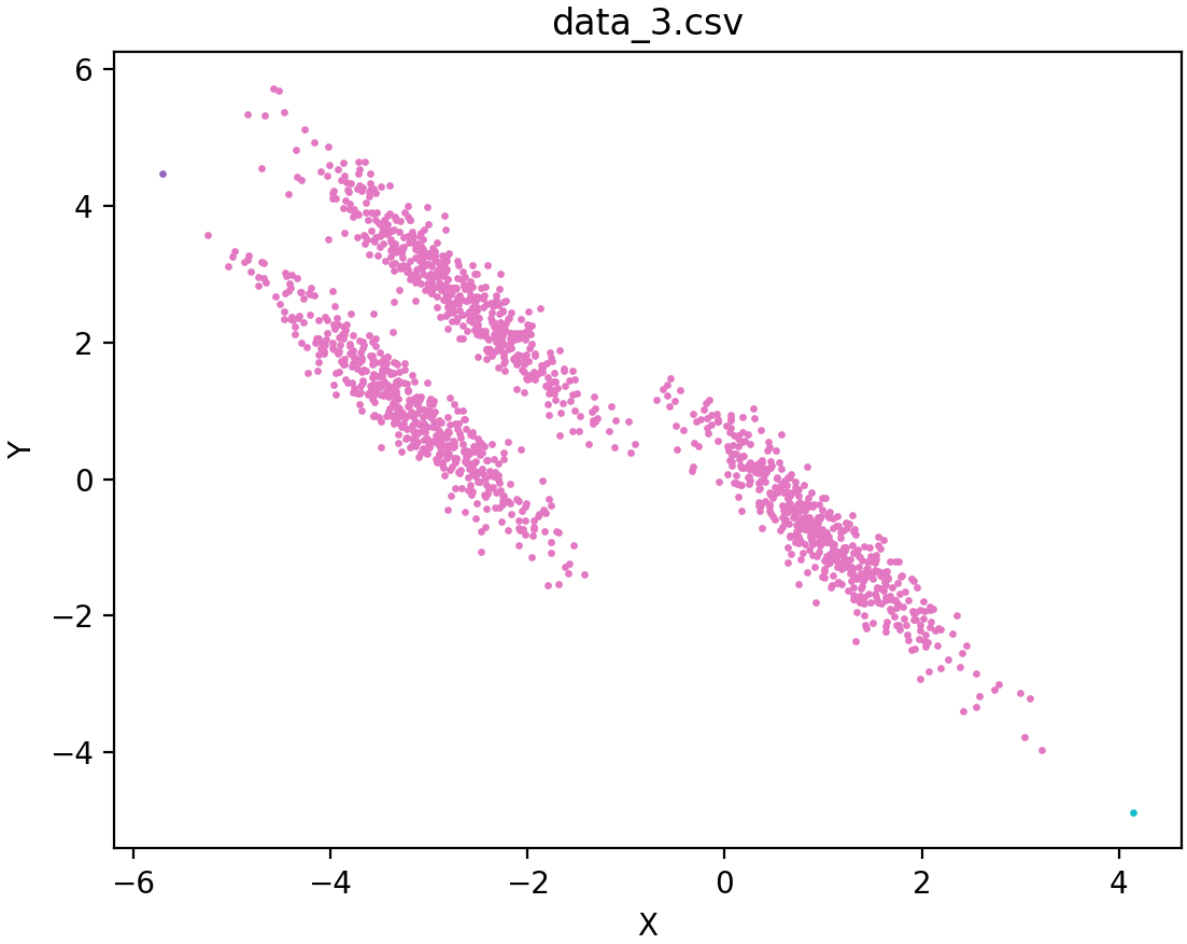
Nazwa pliku: data_0.csv
Liczba punktów: 1500
Liczba klastrów: 2



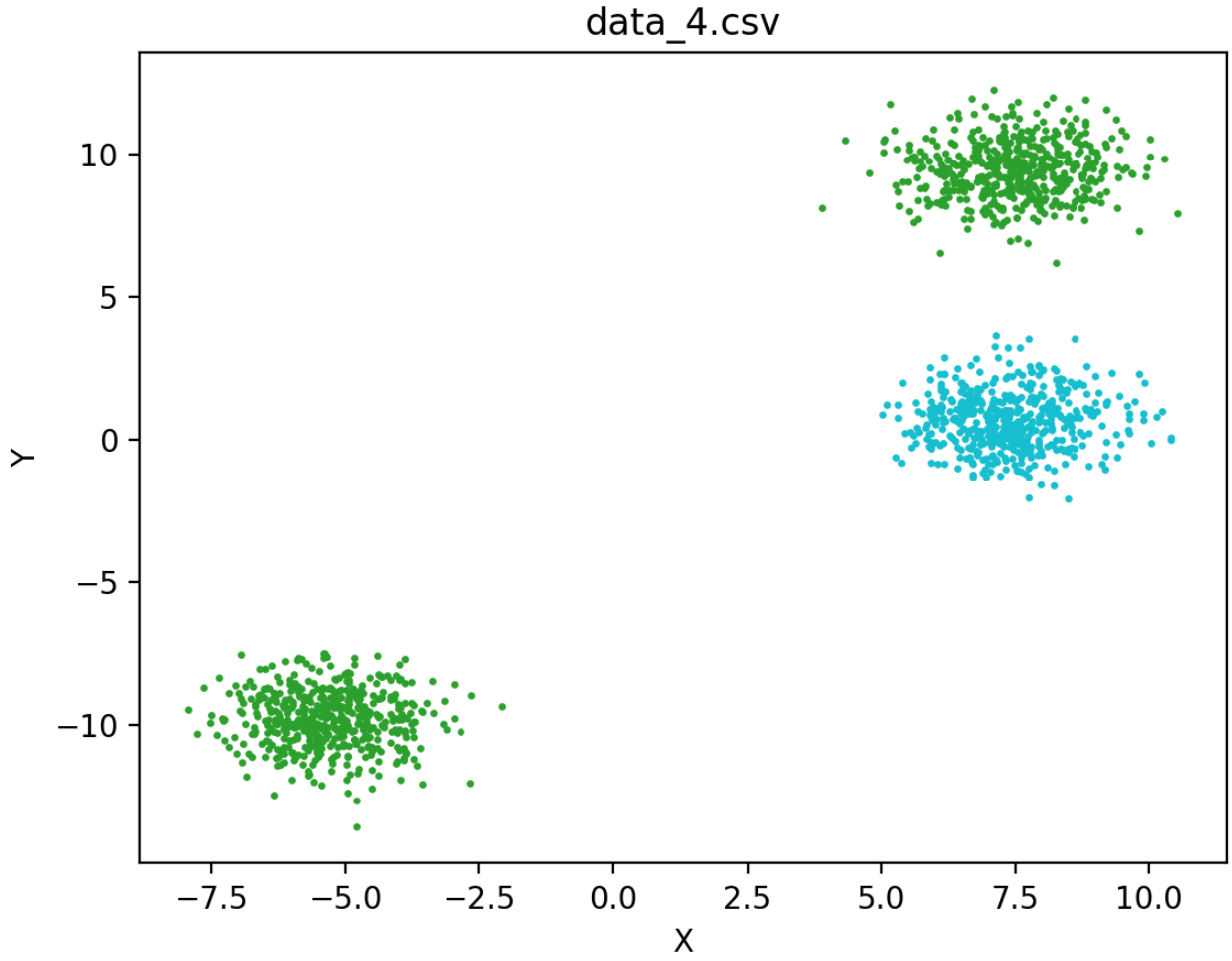
Nazwa pliku: data_2.csv
Liczba punktów: 1500
Liczba klastrów: 3



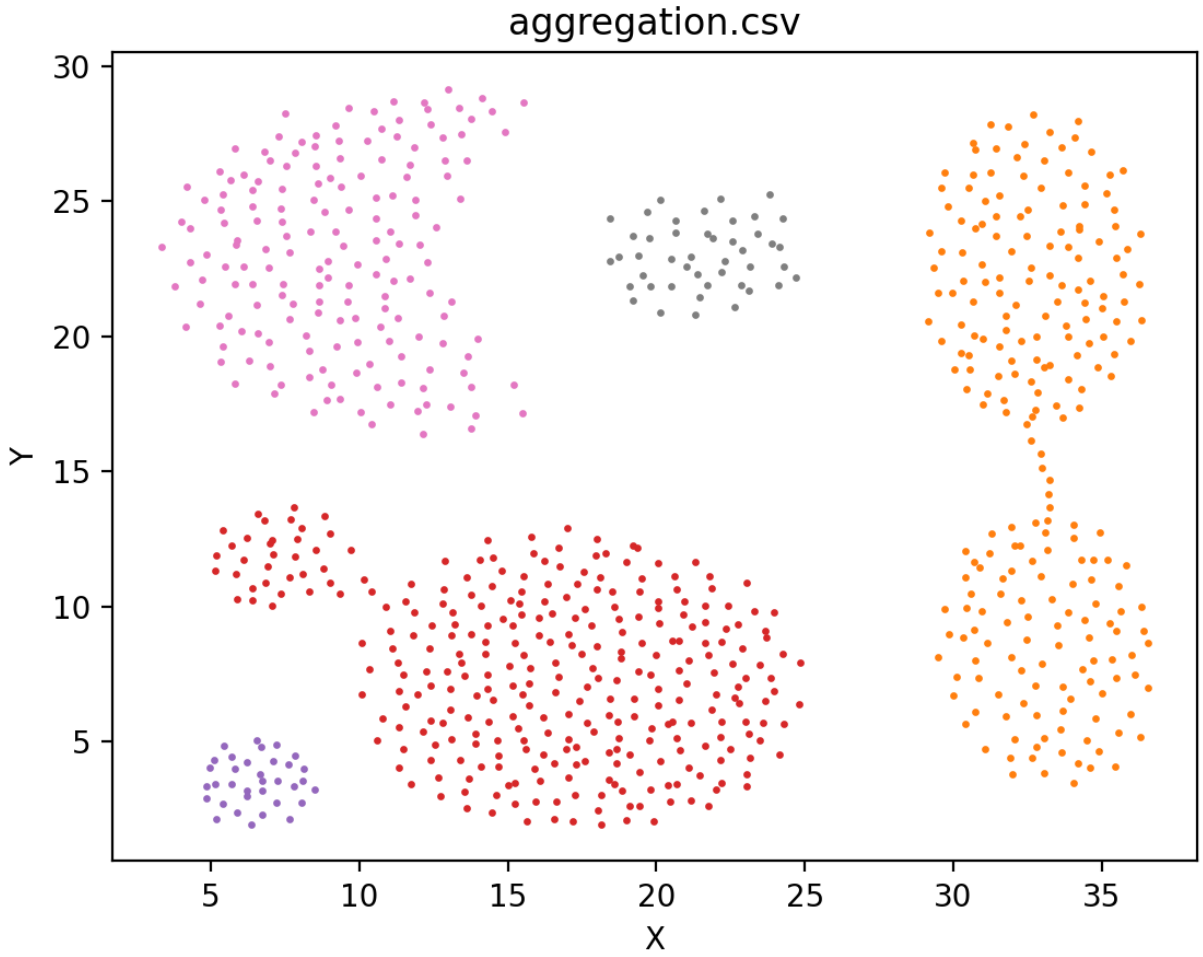
Nazwa pliku: data_3.csv
Liczba punktów: 1500
Liczba klastrów: 3



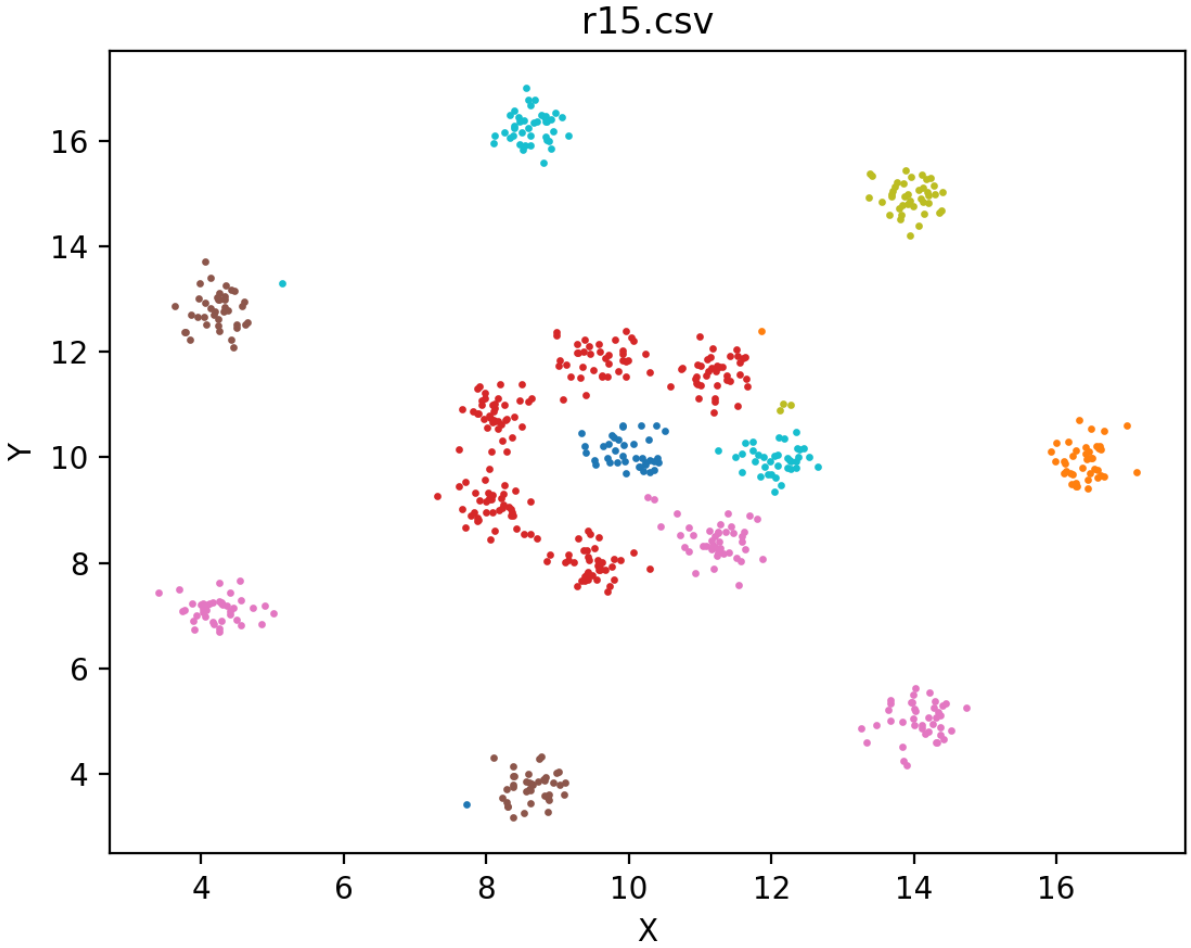
Nazwa pliku: data_4.csv
Liczba punktów: 1500
Liczba klastrów: 6



Nazwa pliku: aggregation.csv
Liczba punktów: 788
Liczba klastrów: 5



Nazwa pliku: r15.csv
Liczba punktów: 600
Liczba klastrów: 15



Uwagi końcowe:
Jak widać na powyższych zrzutach ekranu działanie programu jest prawidłowe i logiczne, bo zgodne z budową minimalnego drzewa rozpinającego. Jednak nie zawsze jest ono zadowalające, ponieważ człowiek intuicyjnie inaczej rozdzieliłby klastry. Tak naprawdę program oddziela punkty najbardziej odległe od innych grup punktów.

Źródła:

- https://www.programiz.com/dsa/kruskal-algorithm
- https://ufkapano.github.io/algorytmy/index.html
- https://matplotlib.org