Algoritmos de Clusterização: DBSCAN

Prof. Mateus Mendelson
mendelson.mateus@gmail.com

mmendelson.com

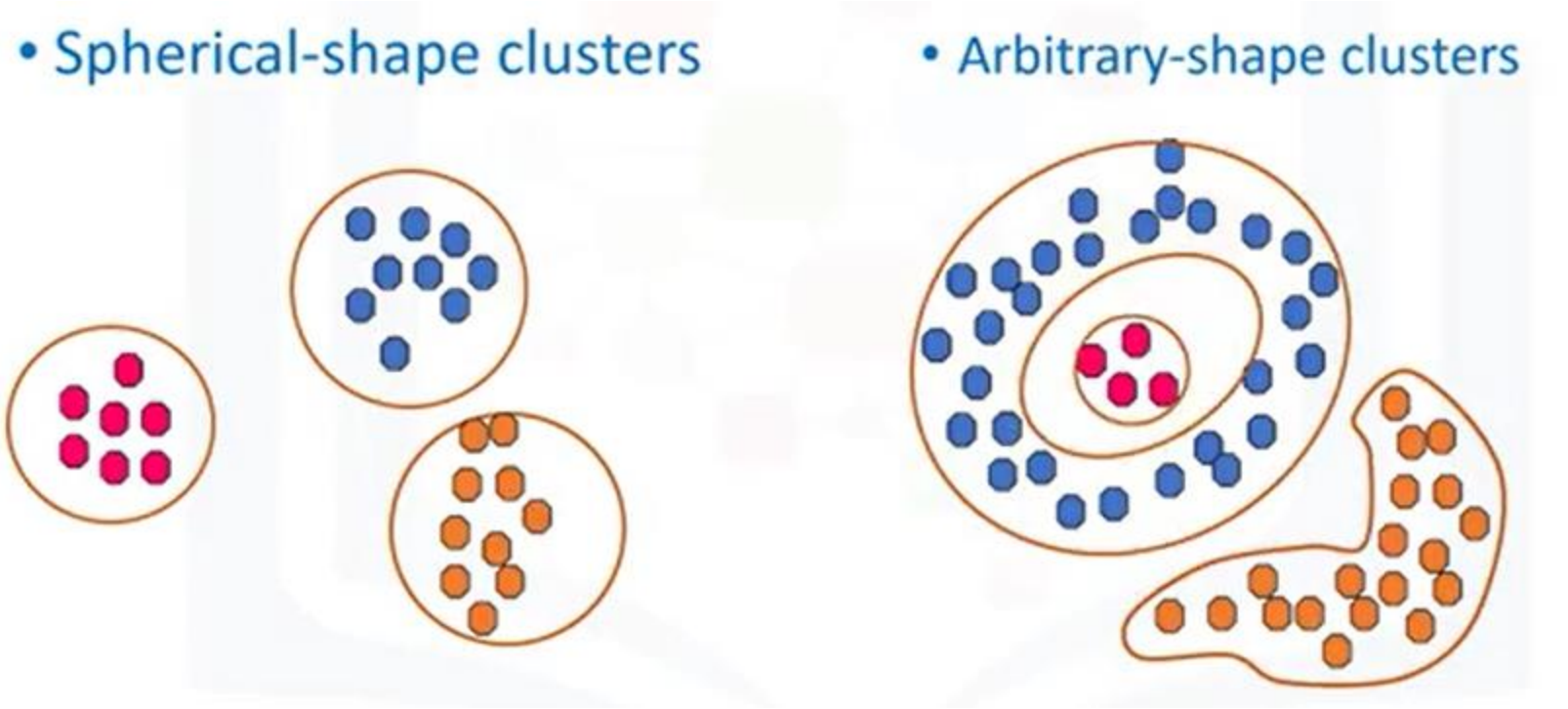
1. Introdução

- DBSCAN: Density-Based Spatial Clustering of Applications with Noise.
- 3 principais razões para utilizar esse algoritmo:
  - ✓ Requer mínimo conhecimento sobre os dados;
  - ✓ Descobre clusters de formatos arbitrários; e
  - ✓ Eficiente para grande bases de dados.
- Este é um dos algoritmos de clusterização mais utilizados.
- Como o próprio nome sugere, seu funcionamento se baseia em encontrar áreas com grandes densidades de objetos.

1. Introdução

- Possui duas características que o destacam dos outros algoritmos que vimos:
  - ✓ Sua capacidade de detectar outliers e
  - ✓ Sua capacidade de identificar clusters de formatos arbitrários (não apenas circulares).

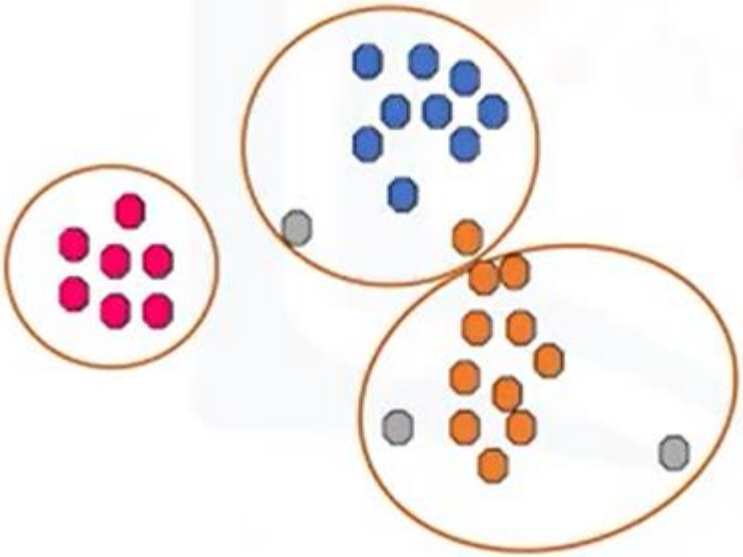
1. Introdução



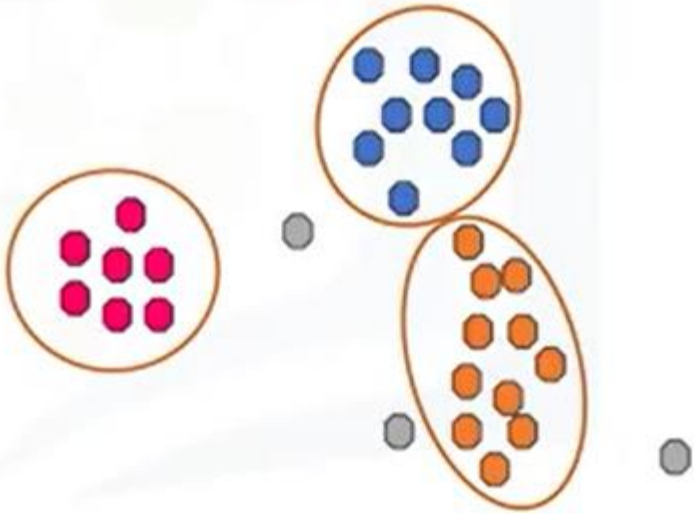
Fonte: Coursera Machine Learning with Python course

1. Introdução

- k-Means assigns all points to a cluster even if they do not belong in any



- Density-based Clustering locates regions of high density, and separates outliers



Fonte: Coursera Machine Learning with Python course

2. O algoritmo DBSCAN

1. Escolher aleatoriamente um ponto que ainda não tenha sido visitado.

2. O algoritmo DBSCAN

1. Escolher aleatoriamente um ponto que ainda não tenha sido visitado.
2. Encontrar quais e quantos pontos estão dentro da vizinhança de raio ε, tendo o ponto em análise como centro.

2. O algoritmo DBSCAN

1. Escolher aleatoriamente um ponto que ainda não tenha sido visitado.
2. Encontrar quais e quantos pontos estão dentro da vizinhança de raio ε, tendo o ponto em análise como centro.
3. Caso haja pelo menos *MinPts* dentro desse raio ε (incluindo o próprio ponto central), esse ponto central é considerado como **core** e os pontos em sua vizinhança são marcados como pontos **border**.

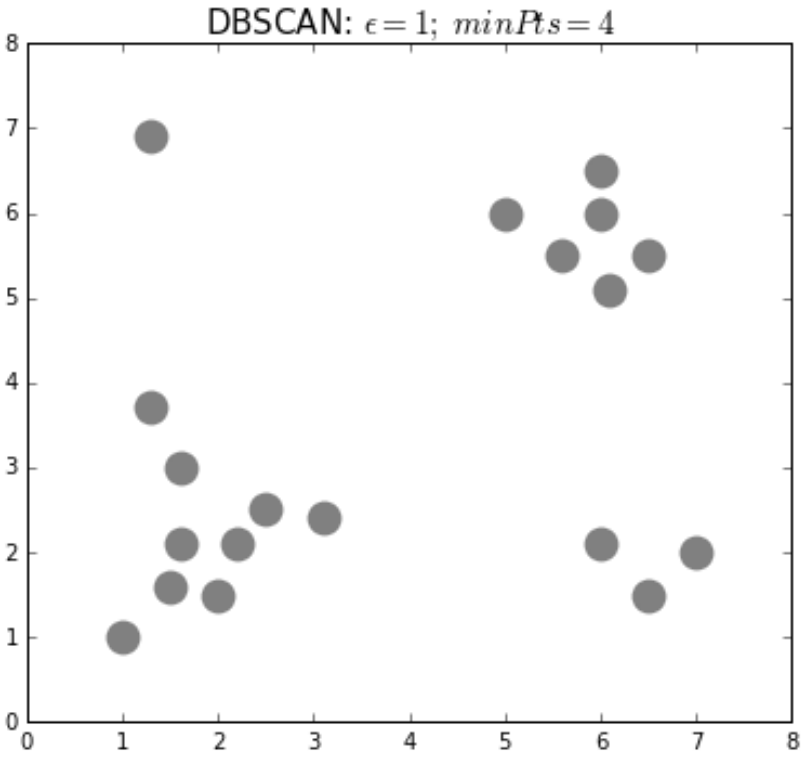
2. O algoritmo DBSCAN

1. Escolher aleatoriamente um ponto que ainda não tenha sido visitado.
2. Encontrar quais e quantos pontos estão dentro da vizinhança de raio ε, tendo o ponto em análise como centro.
3. Caso haja pelo menos *MinPts* dentro desse raio ε (incluindo o próprio ponto central), esse ponto central é considerado como **core** e os pontos em sua vizinhança são marcados como pontos **border**.
4. Caso não haja pelo menos *MinPts* na vizinhança (incluindo o próprio ponto central) e o ponto ainda não tenha sido marcado como **border**, esse ponto deve ser marcado como **outlier**. Retornar ao passo 1.

2. O algoritmo DBSCAN

1. Escolher aleatoriamente um ponto que ainda não tenha sido visitado.
2. Encontrar quais e quantos pontos estão dentro da vizinhança de raio ε, tendo o ponto em análise como centro.
3. Caso haja pelo menos *MinPts* dentro desse raio ε (incluindo o próprio ponto central), esse ponto central é considerado como **core** e os pontos em sua vizinhança são marcados como pontos **border**.
4. Caso não haja pelo menos *MinPts* na vizinhança (incluindo o próprio ponto central) e o ponto ainda não tenha sido marcado como **border**, esse ponto deve ser marcado como **outlier**. Retornar ao passo 1.
5. Por fim, todos os pontos **core** que alcançam uns aos outros por meio de suas vizinhanças formam um cluster.

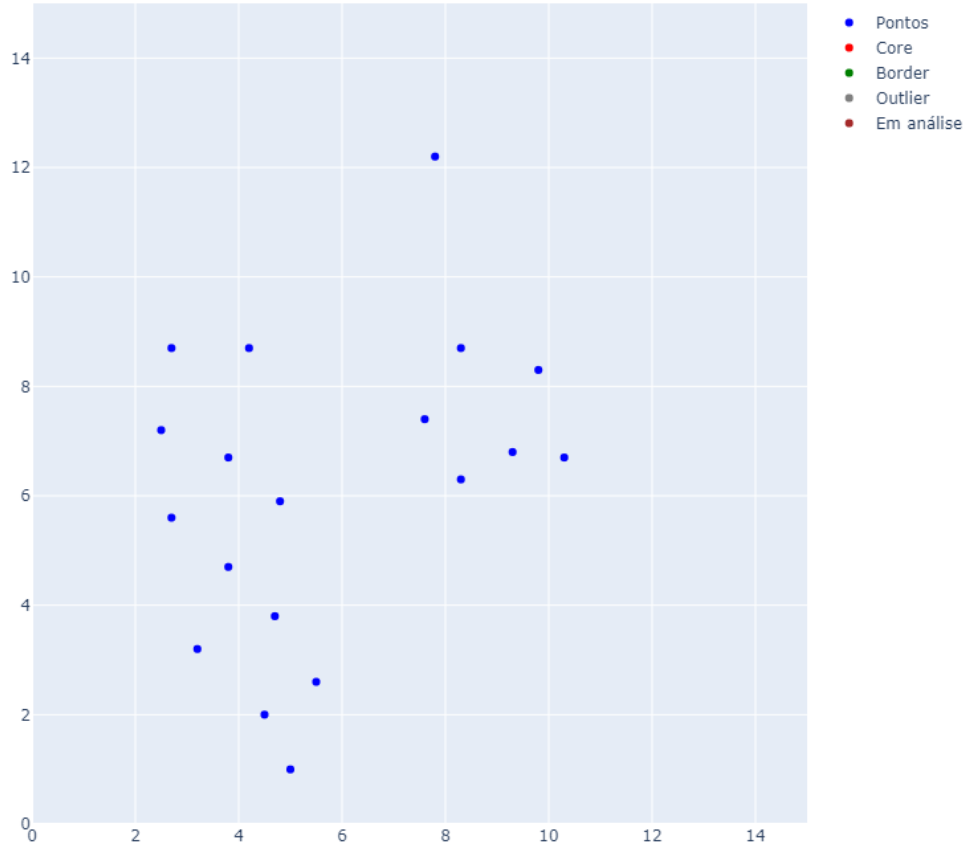
2. O algoritmo DBSCAN



Fonte: https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

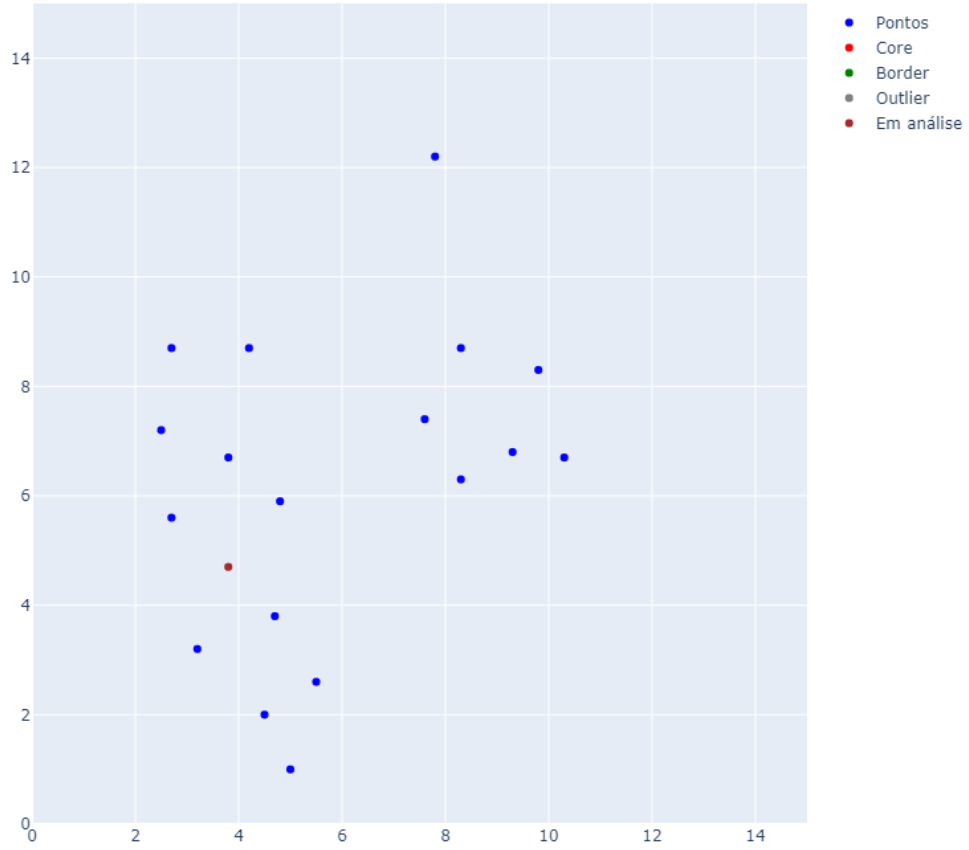
2. O algoritmo DBSCAN

- Considere o conjunto de pontos abaixo.



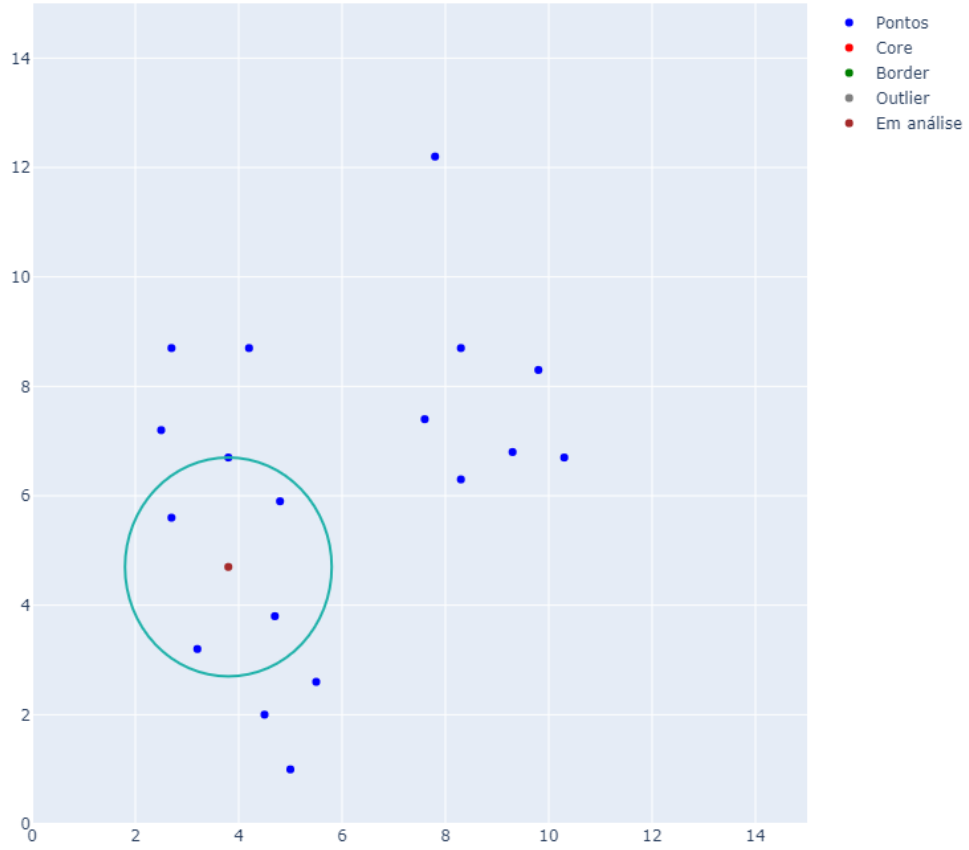
2. O algoritmo DBSCAN

- Sorteamos um ponto para analisar.



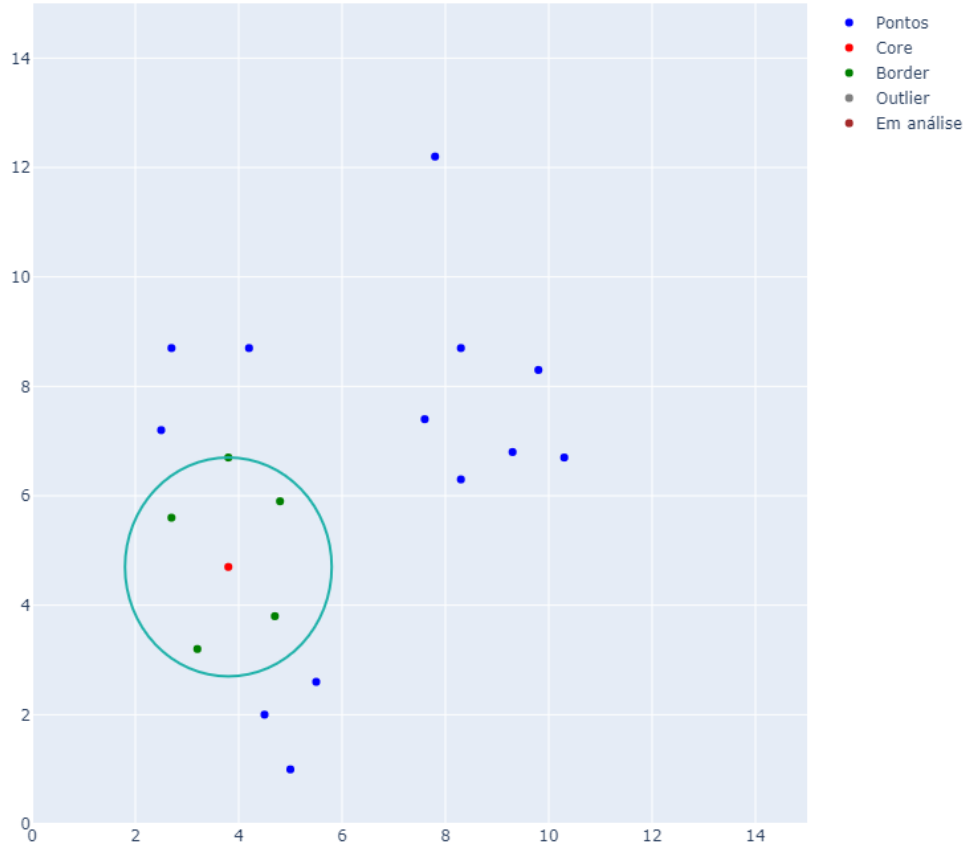
2. O algoritmo DBSCAN

- Utilizando ε = 2, vamos ver sua vizinhança.



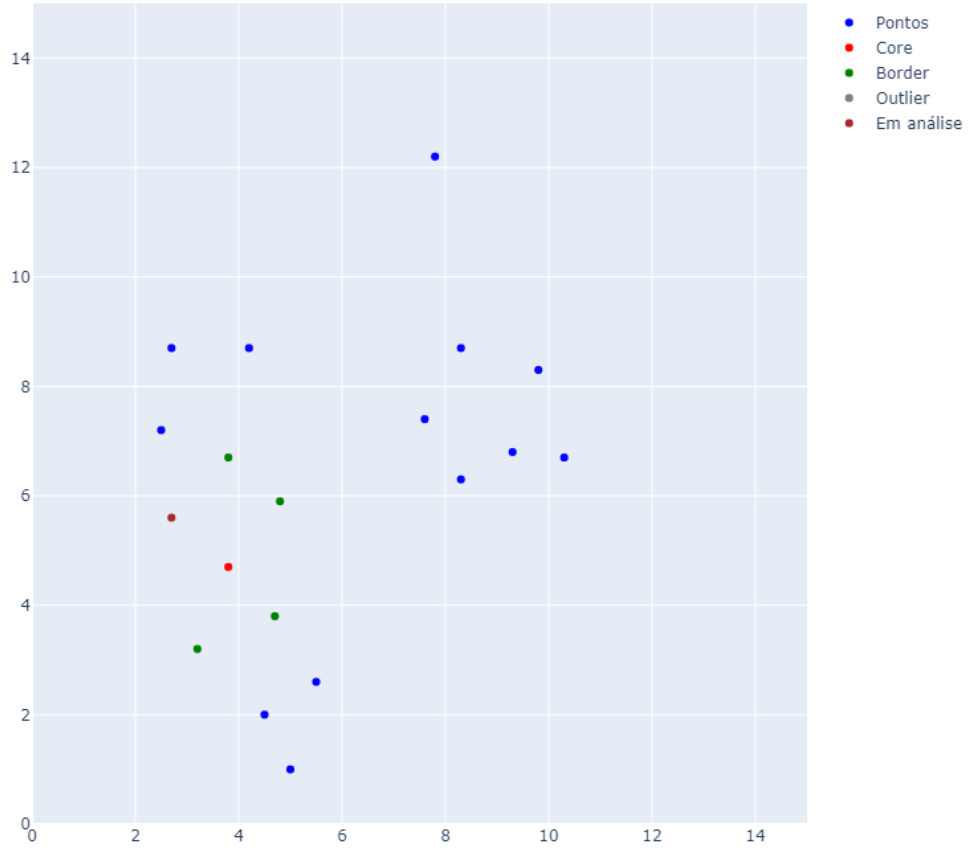
2. O algoritmo DBSCAN

- Usando *MinPts* = 6, marcamos o **core** e os **border**s.



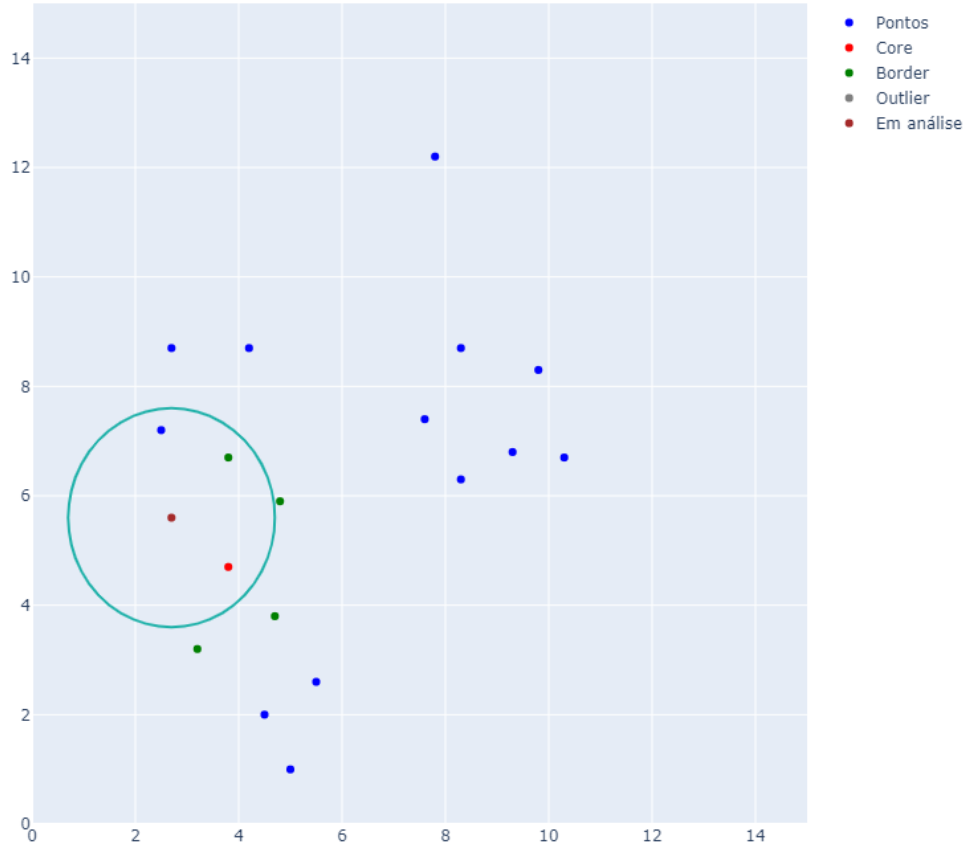
2. O algoritmo DBSCAN

- Vamos sortear outro ponto que ainda não tenha sido centro.



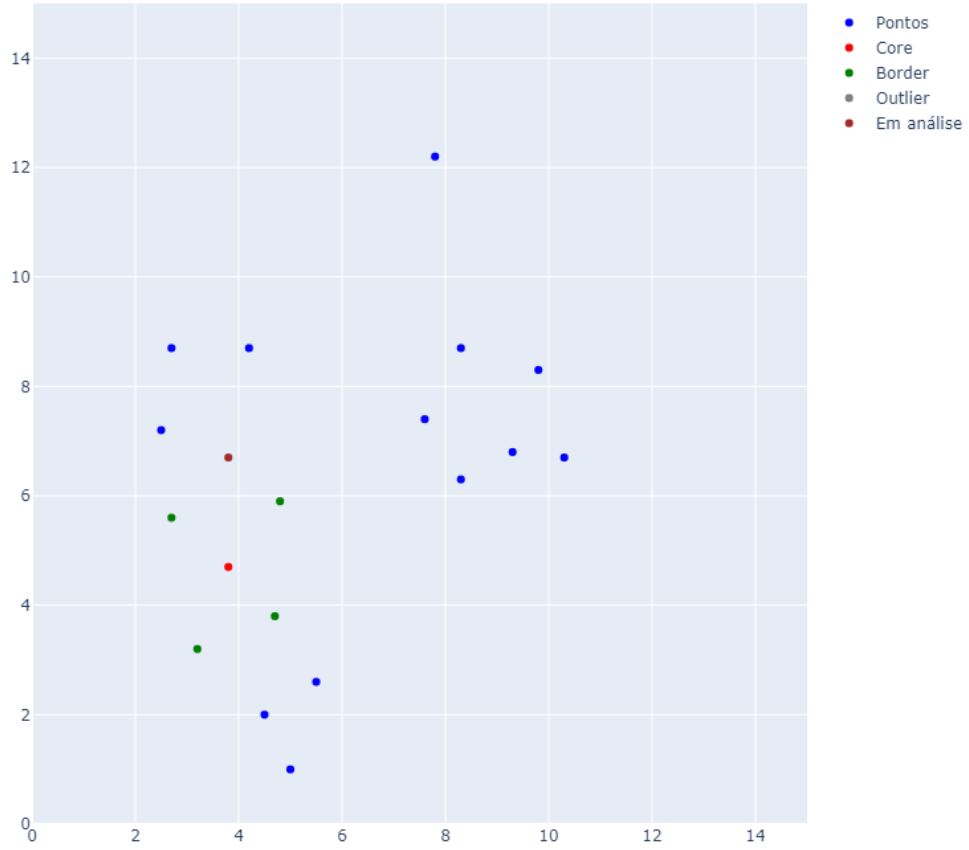
2. O algoritmo DBSCAN

- Há apenas 4 pontos em sua vizinhança -> continua **border**.



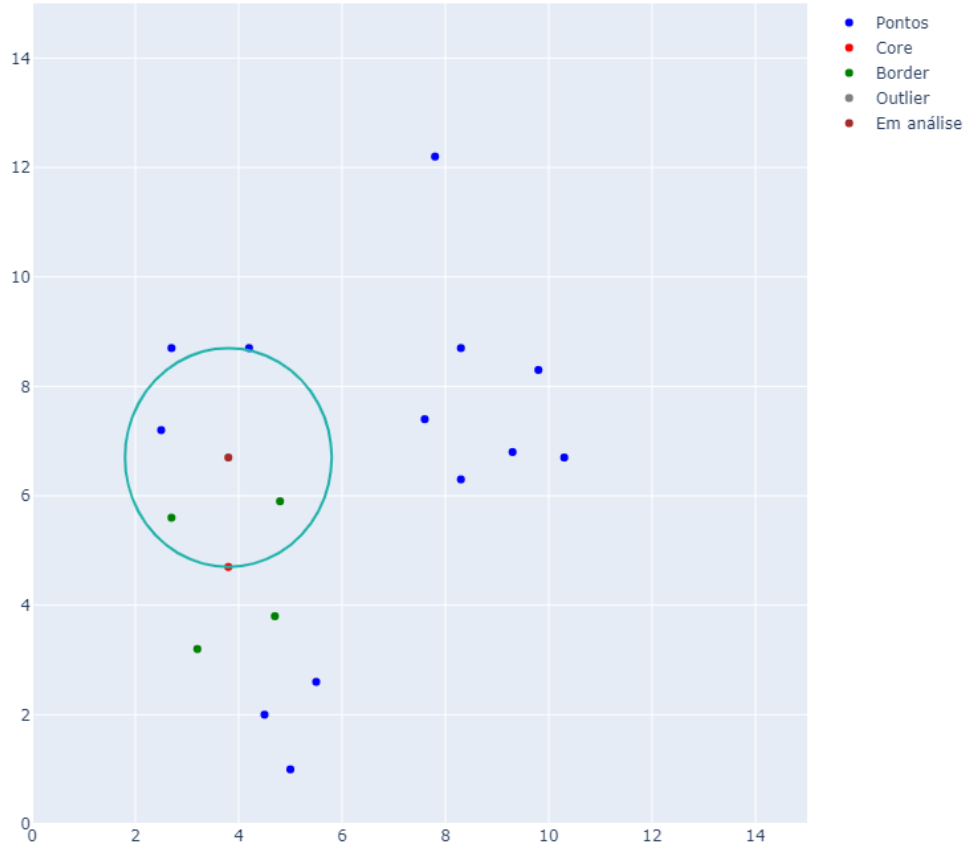
2. O algoritmo DBSCAN

- Vamos sortear outro ponto para analisar.



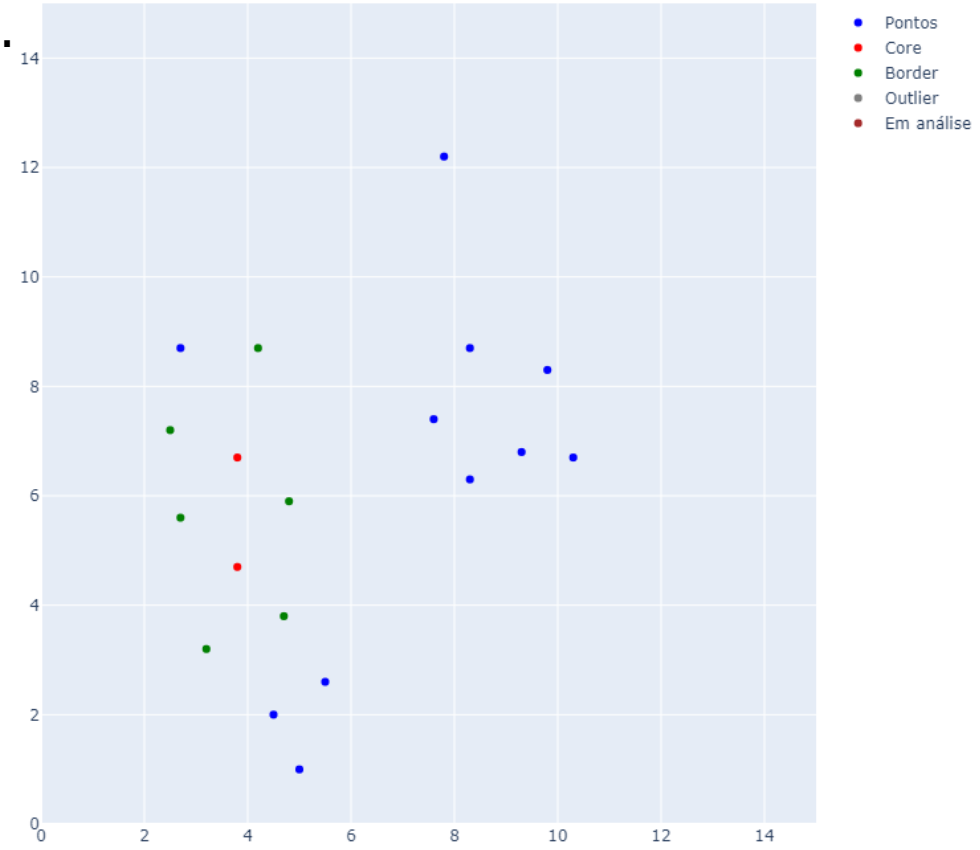
2. O algoritmo DBSCAN

- Sua vizinhança possui 6 pontos -> vira **core**.



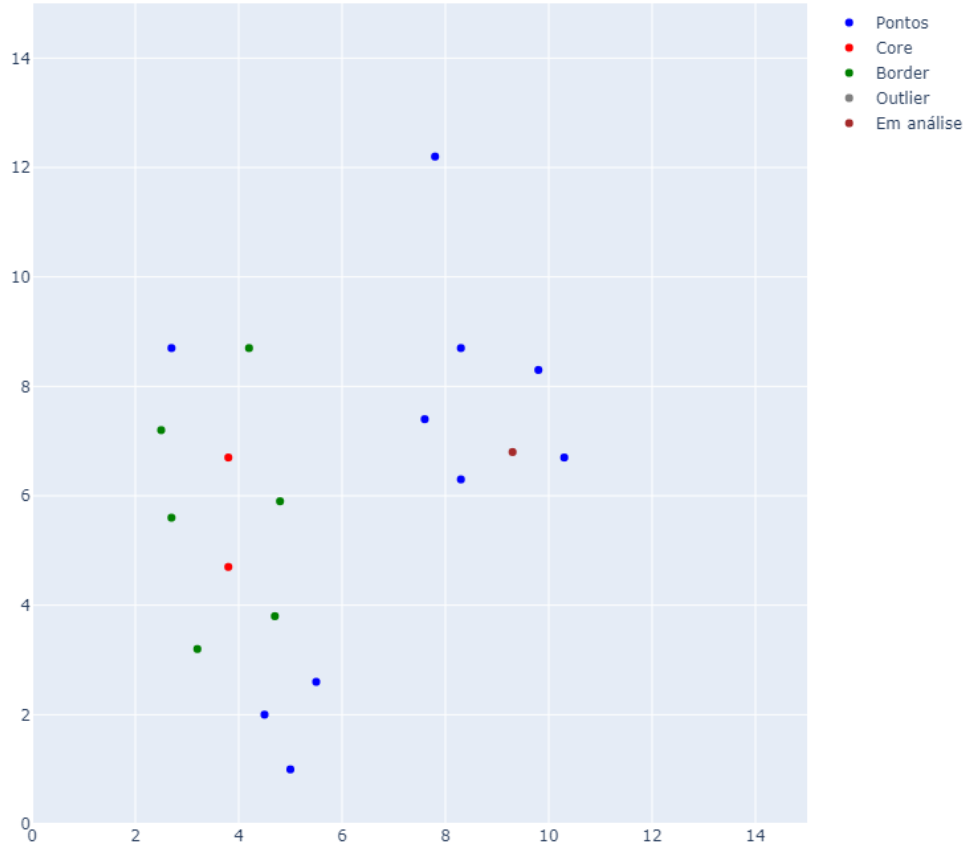
2. O algoritmo DBSCAN

- Sua vizinhança possui 6 pontos -> vira **core** e cria **border**s.



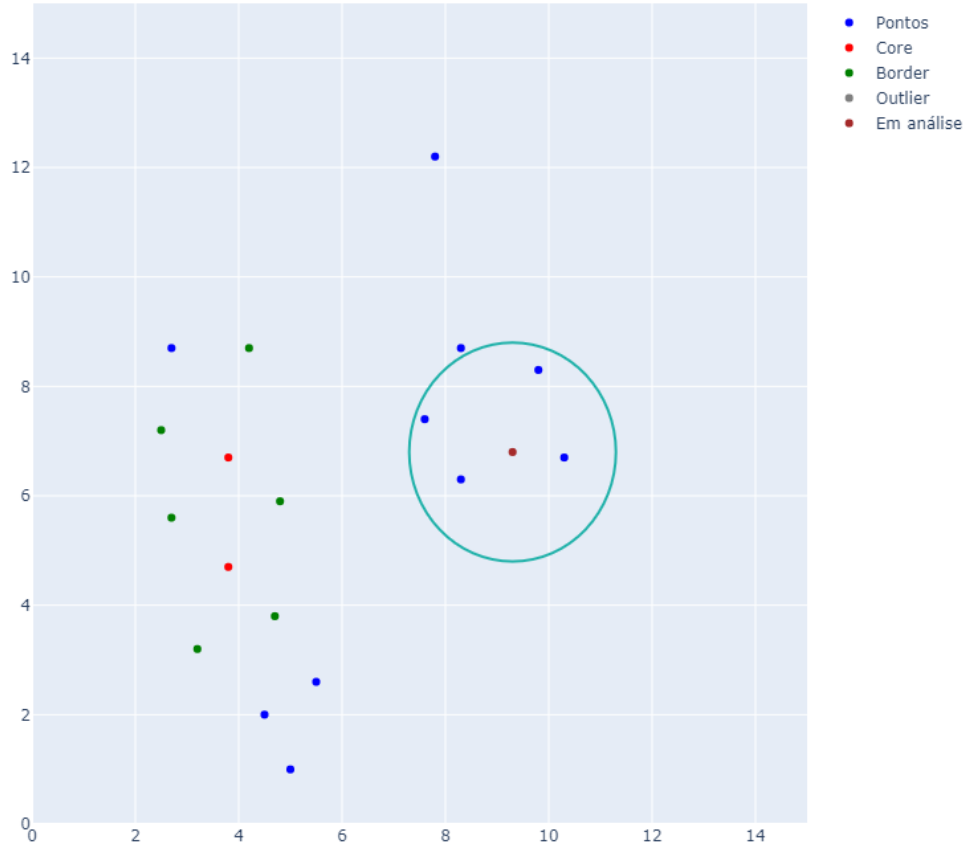
2. O algoritmo DBSCAN

- Vamos sortear o próximo ponto.



2. O algoritmo DBSCAN

- Há apenas 5 pontos em sua vizinhança e não está no alcance de um **core**. Ele vira um **outlier**.

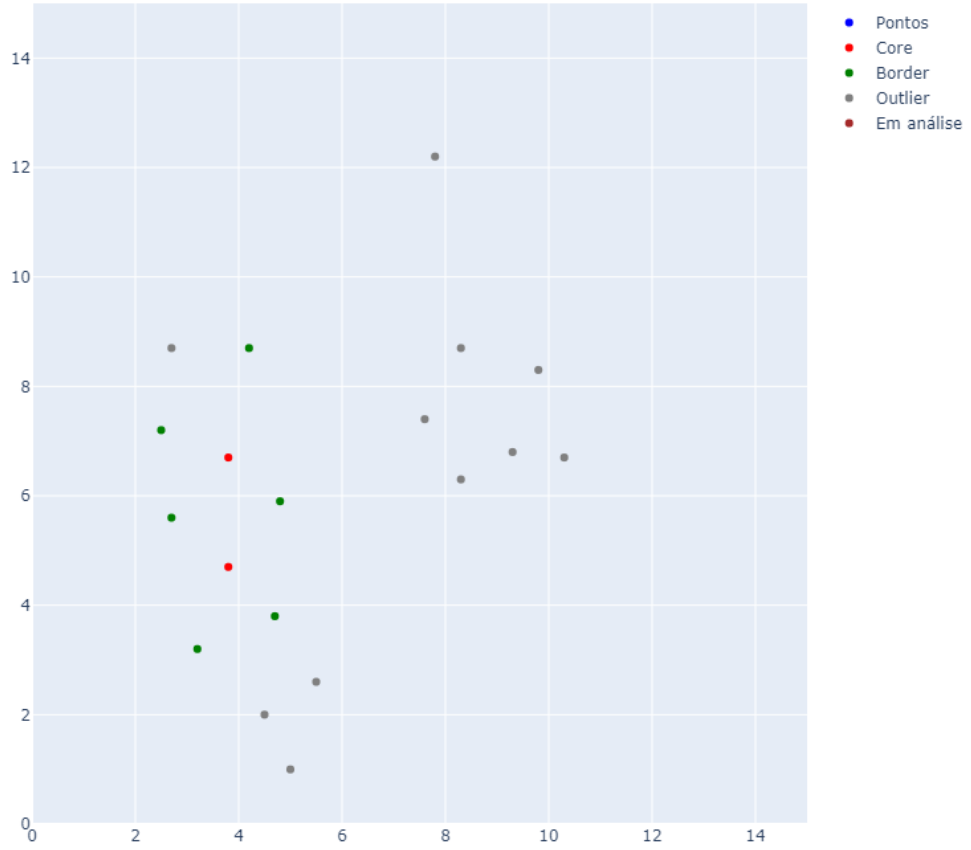


2. O algoritmo DBSCAN

- Avançando até o final...

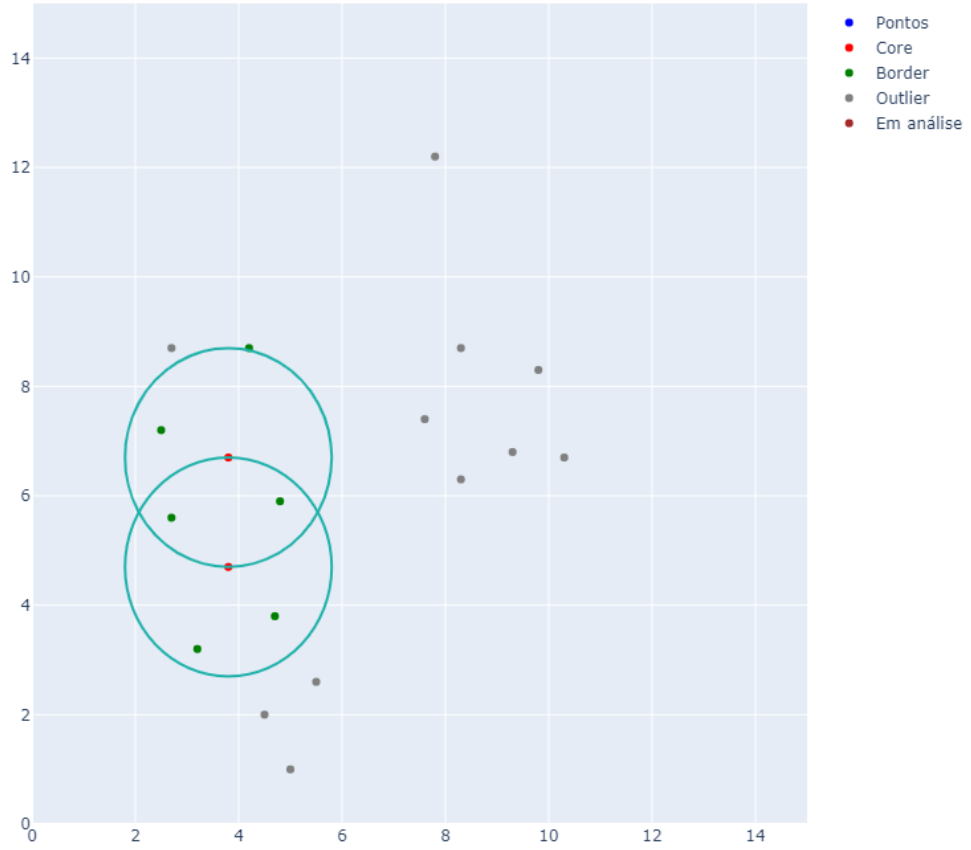
2. O algoritmo DBSCAN

- Essas são as marcações finais de todos os pontos.



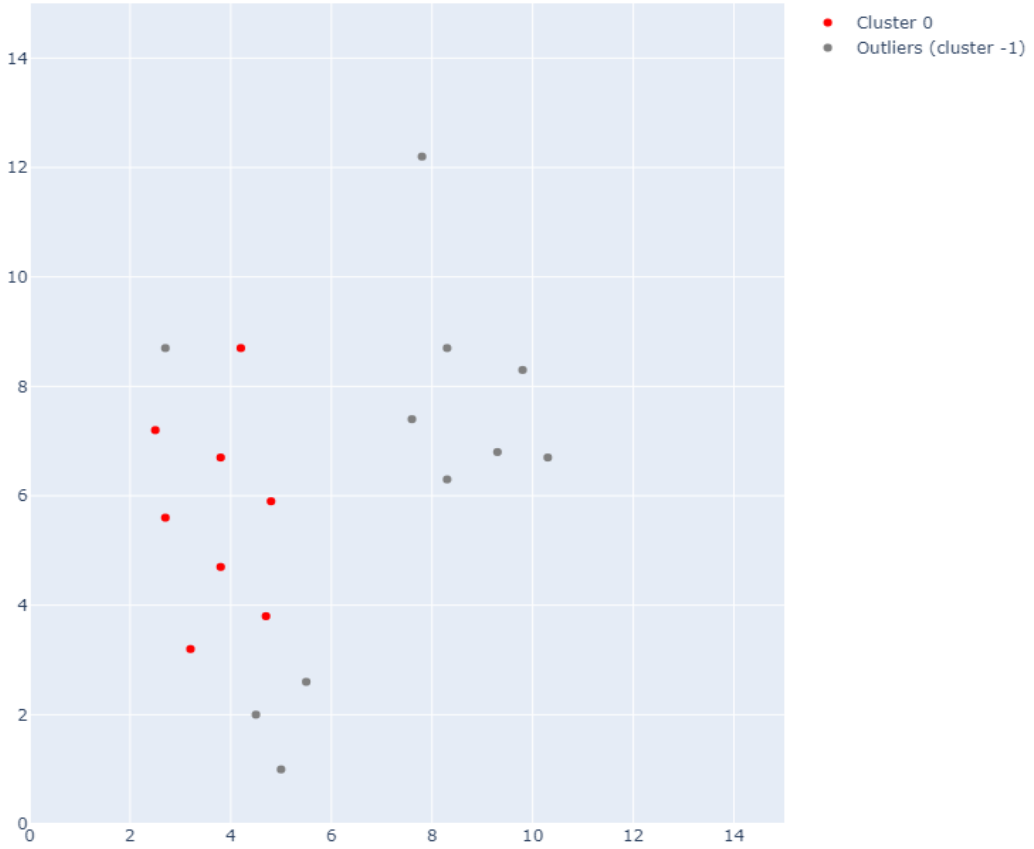
2. O algoritmo DBSCAN

- Os **core**s que se alcançam formam um cluster junto com seus **border**s.



2. O algoritmo DBSCAN

- Sendo assim, temos apenas 1 cluster e 11 **outliers** *MinPts* = *6* e ε = 2.



2. O algoritmo DBSCAN

- O que vocês acham desse resultado? Visualmente, vocês acham que foi o melhor resultado possível ou havia outros clusters por ali que não foram formados?

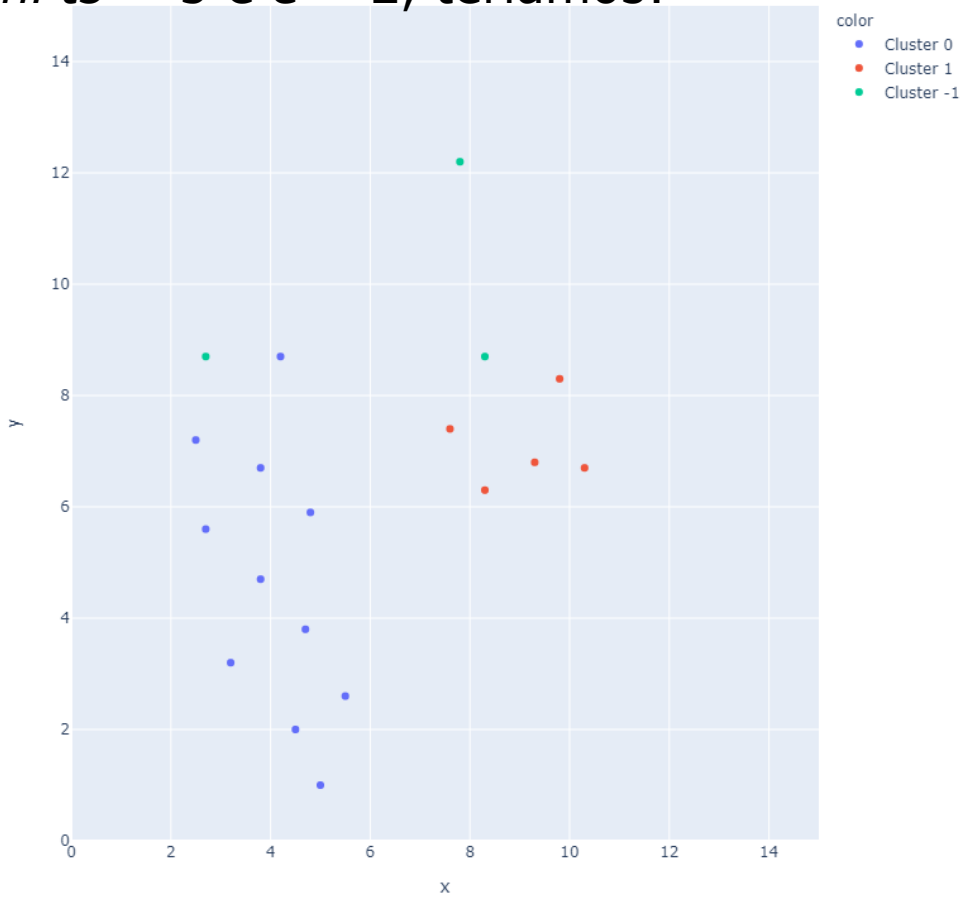
2. O algoritmo DBSCAN

- O que vocês acham desse resultado? Visualmente, vocês acham que foi o melhor resultado possível ou havia outros clusters por ali que não foram formados?

- Note que uma simples alteração em um dos parâmetros do algoritmo teria resultado na criação de dois clusters.

2. O algoritmo DBSCAN

- Para *MinPts* = 5 e ε = 2, teríamos:



2. O algoritmo DBSCAN

- Agora, podemos apontar duas grandes desvantagem deste algoritmo:

  - ✓ Se a base de dados possuir pontos que formam clusters de diferentes densidades, então o DBSCAN irá falhar em suas detecções, pois ele depende de dois parâmetros **fixos** que não podem variar entre um cluster e outro; e

  - ✓ Sem um conhecimento prévio dos dados, a escolha de seus dois parâmetros pode se tornar um processo complicado, exigindo que sejam executadas várias comparações entre os diferentes resultados obtidos com cada valor para cada parâmetro.

- Vamos abrir o notebook desta aula e codificar.

3. Mini-projeto

• Para este projeto, iremos utilizar o dataset de COVID-19 dos casos nos Estados Unidos no período de 21 de janeiro até 09 de abril de 2020 processado.

• Seu objetivo deve ser implementar um código que replique exatamente os mesmos resultados obtidos através do notebook executado em sala de aula.

• Tarefa: implementar a função "fit_dbscan(pontos, eps, min_samples)". Retorne uma lista indicando os clusters a que cada ponto pertence (utilize -1 para indicar outliers).

- ✓ pontos: conjunto de pontos 2D (casos x mortes) que serão clusterizados
- ✓ eps: valor de ε que indica o raio que deve ser considerado como vizinhança de um ponto central
- ✓ min_samples: a quantidade mínima de pontos que devem estar dentro de uma vizinhança, incluindo o próprio ponto central, para que ele seja considerado como um **core**

3. Mini-projeto

• O seu relatório será o notebook exportado para um arquivo HTML e deve conter:

- ✓ Um scatter plot mostrando os clusters, cada cluster deve estar em uma cor distinta e deve haver legenda para as cores, eixos e título.
- ✓ Discorra sobre cada cluster: o que eles indicam?
- ✓ Desafio: implementar uma visualização iterativa do processo de treinamento igual às imagens dessa aula, porém como um vídeo ou algo do tipo.