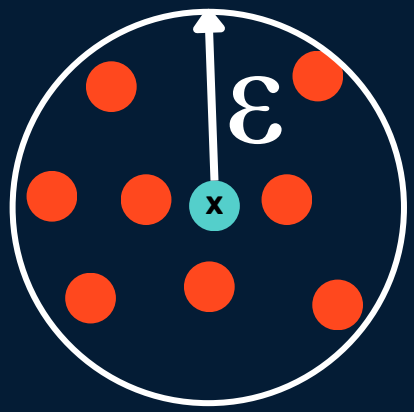
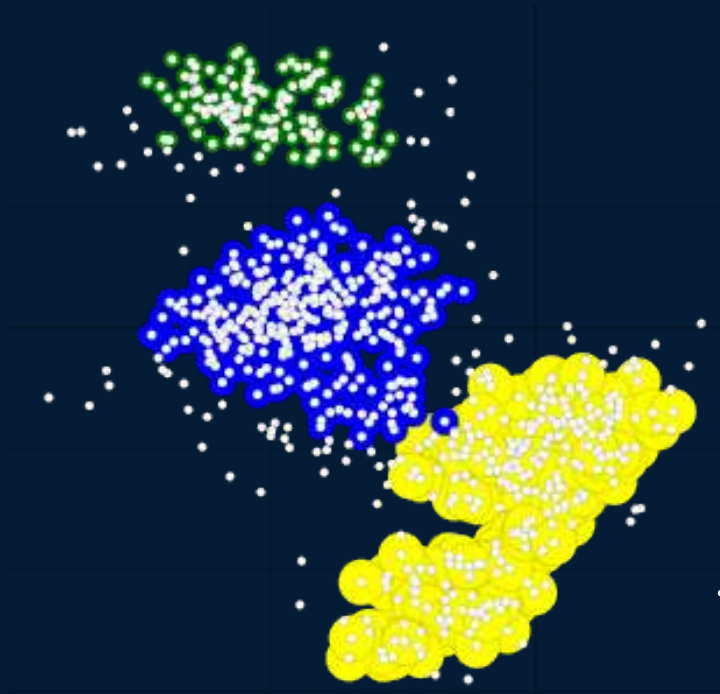
Algoritmo DBSCAN

Density-Based Spatial Clustering of Applications with Noise





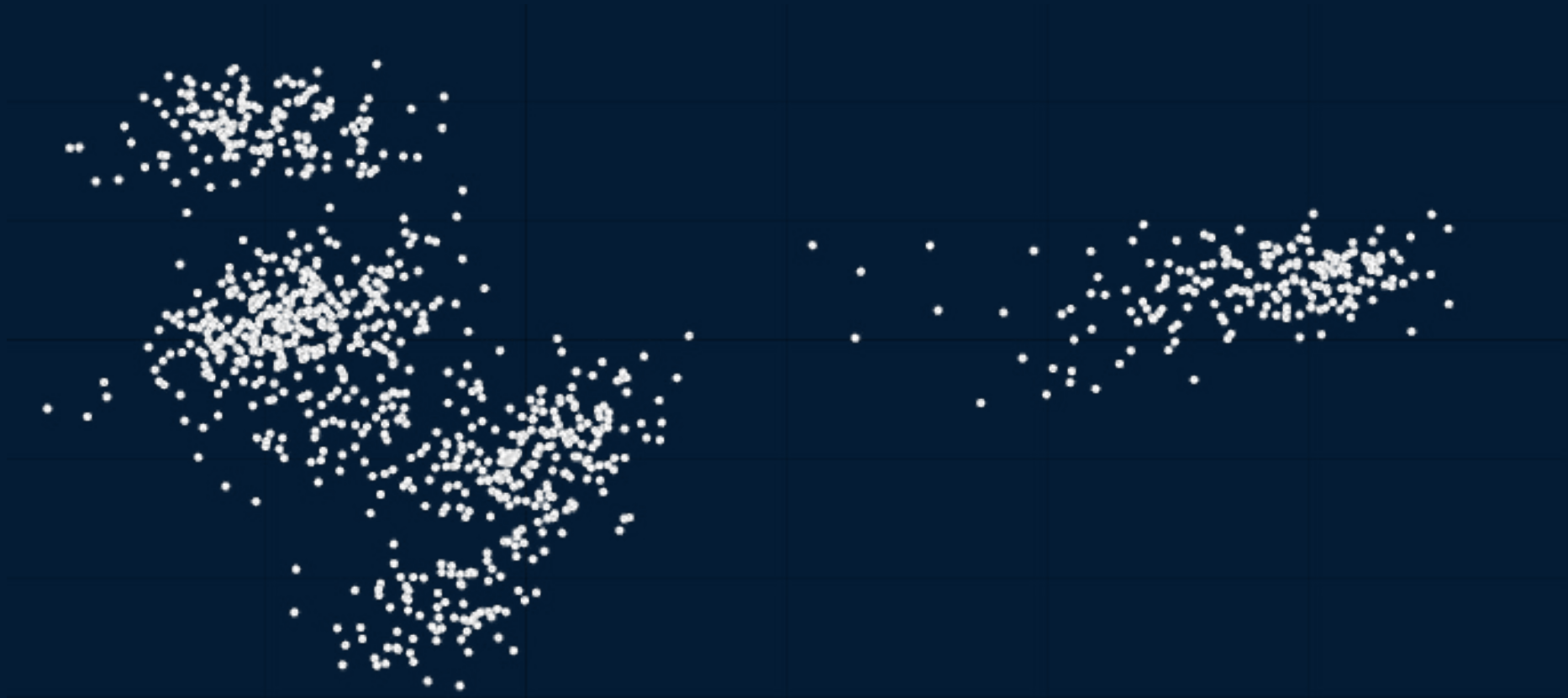
Vinícius Menezes Monte

Paulo Diego De Meneses

O Algoritmo

O que é?

O DBSCAN é um algoritmo de agrupamento de dados de **alta densidade**. Proposto em 1996 por Martin Ester, Hans-Peter Kriegel, Jörg Sander, e Xiaowei Xu .



Qual a motivação?

Detecção de **outliers/ruidos** de grupos de dados de tamanhos variados.

Como funciona?

O algoritmo necessita de 3 parâmetros principais.

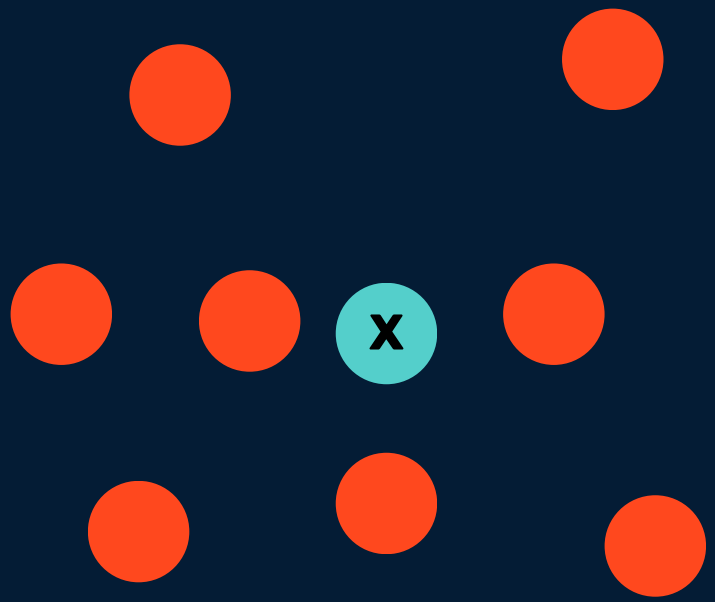
- D -> Conjunto de dados que será usado para o agrupamento
- epsilon -> O raio de cada vizinhança a ser considerada
- minPoints -> A densidade a ser considerada

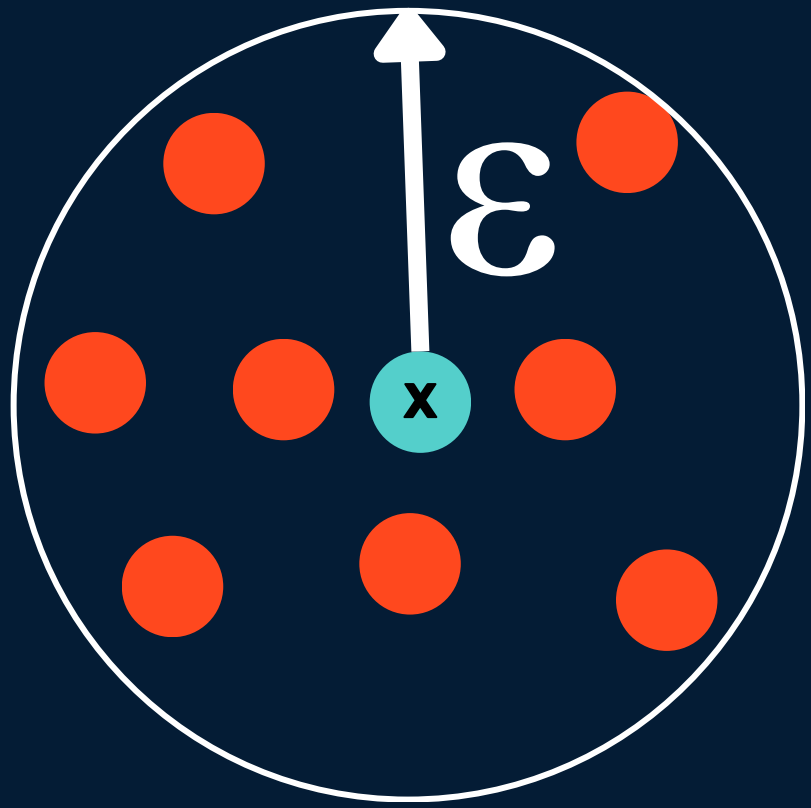
```
DBSCAN (SetOfPoints, Eps, MinPts)

// SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point,
            ClusterId, Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

ε – Épsilon

Épsilon é raio da vizinhança de um dado ponto.

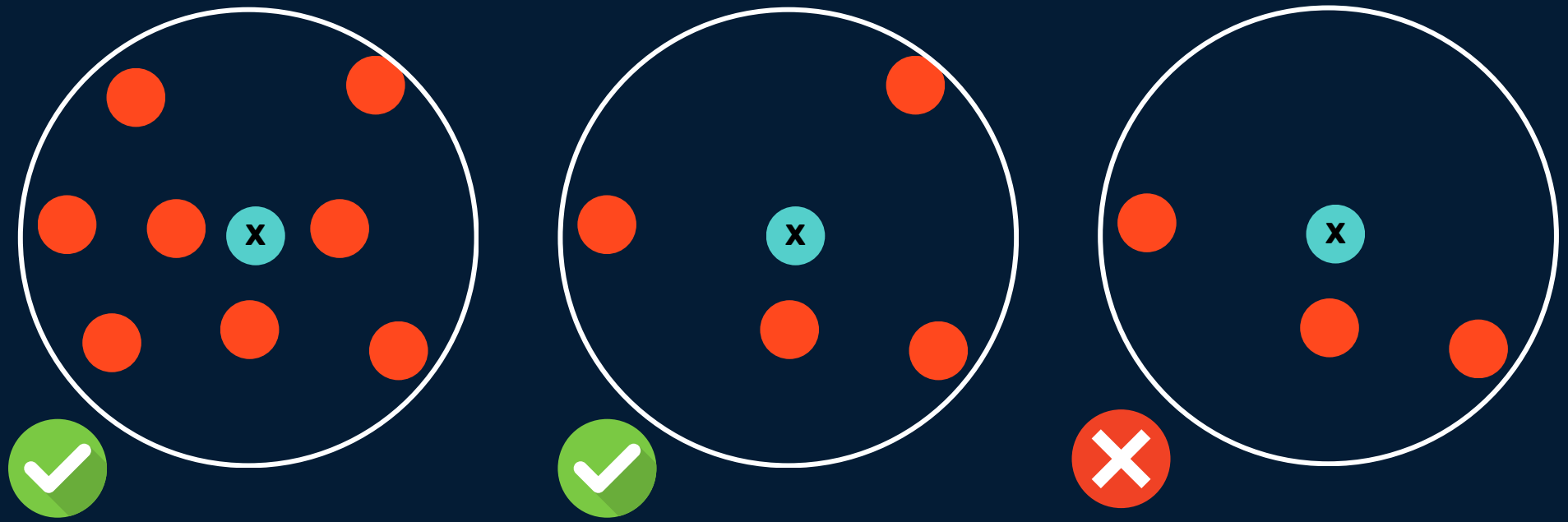




MinPts

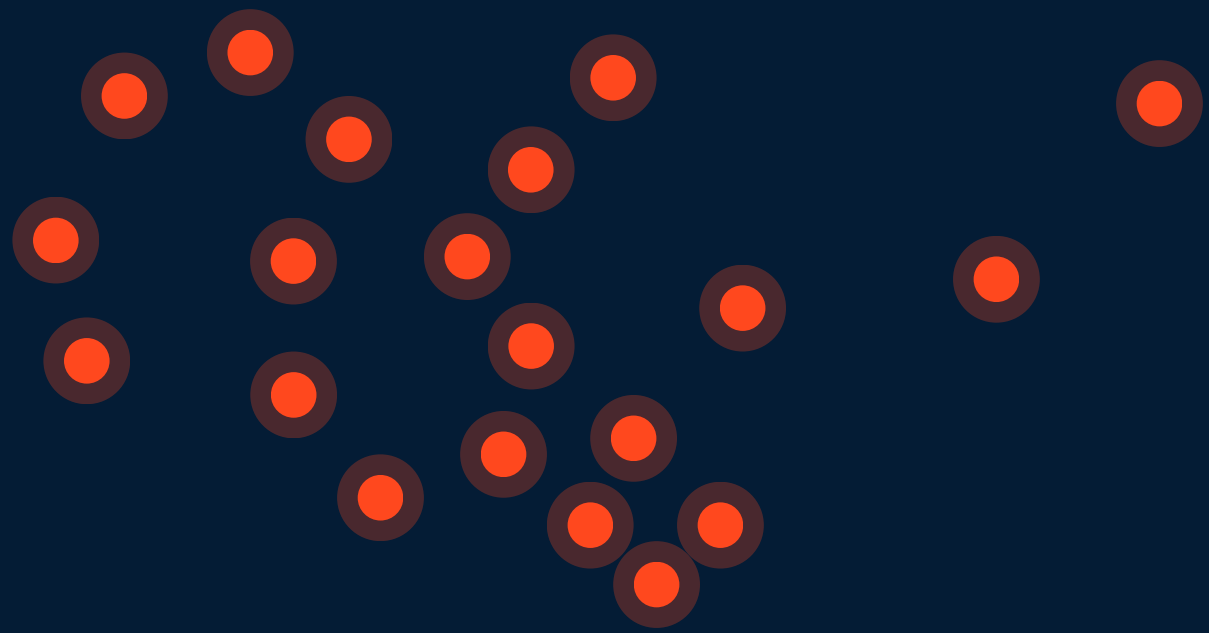
MinPts é o limite de densidade. Se uma vizinhança tiver pelo menos MinPts pontos, ele será considerava densa e poderá fazer parte de um cluster.

MinPts = 5



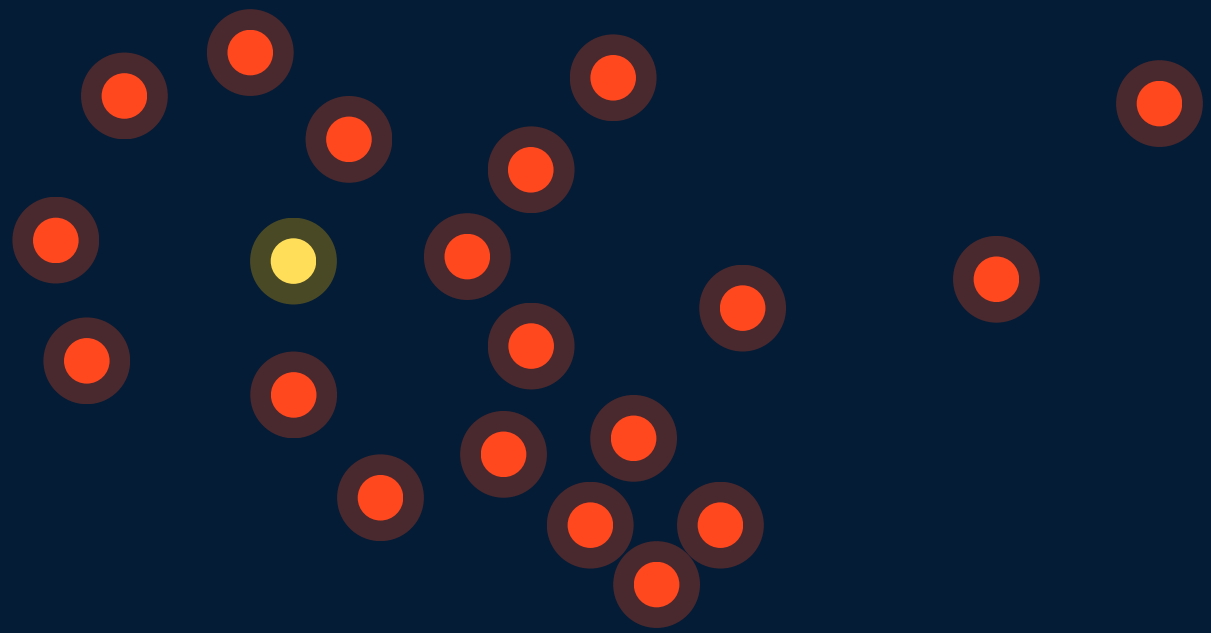
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



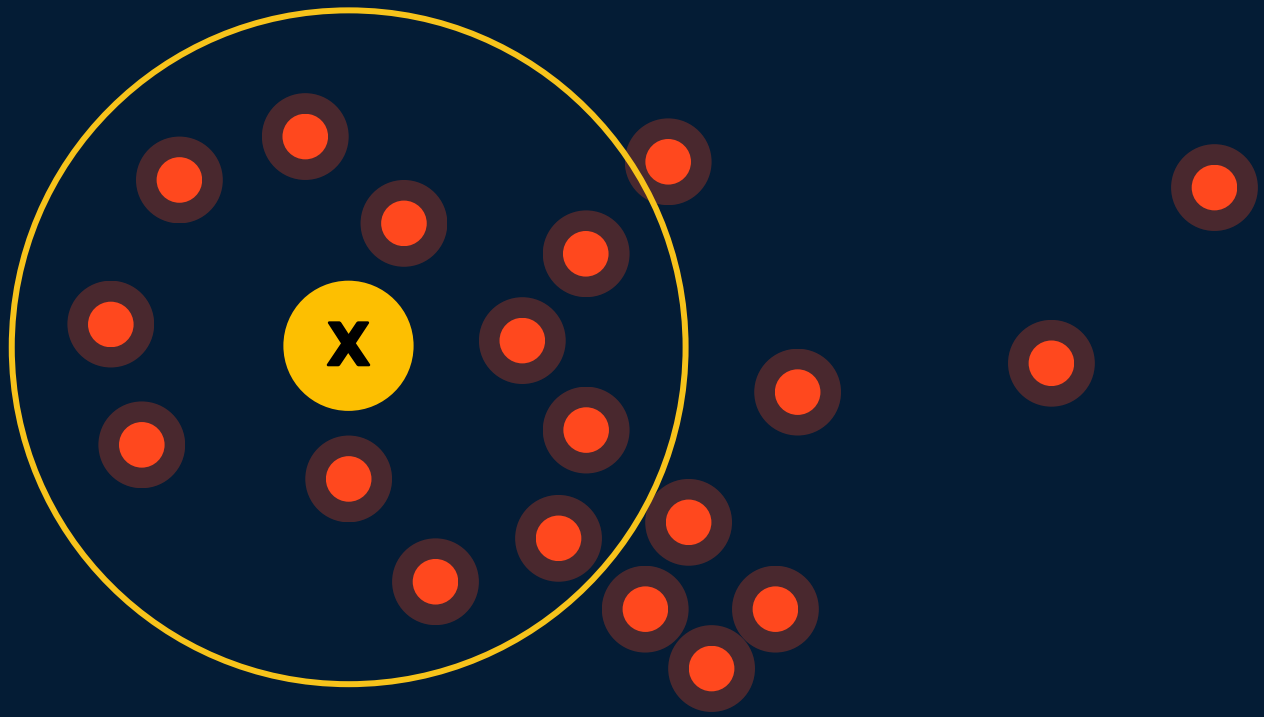
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



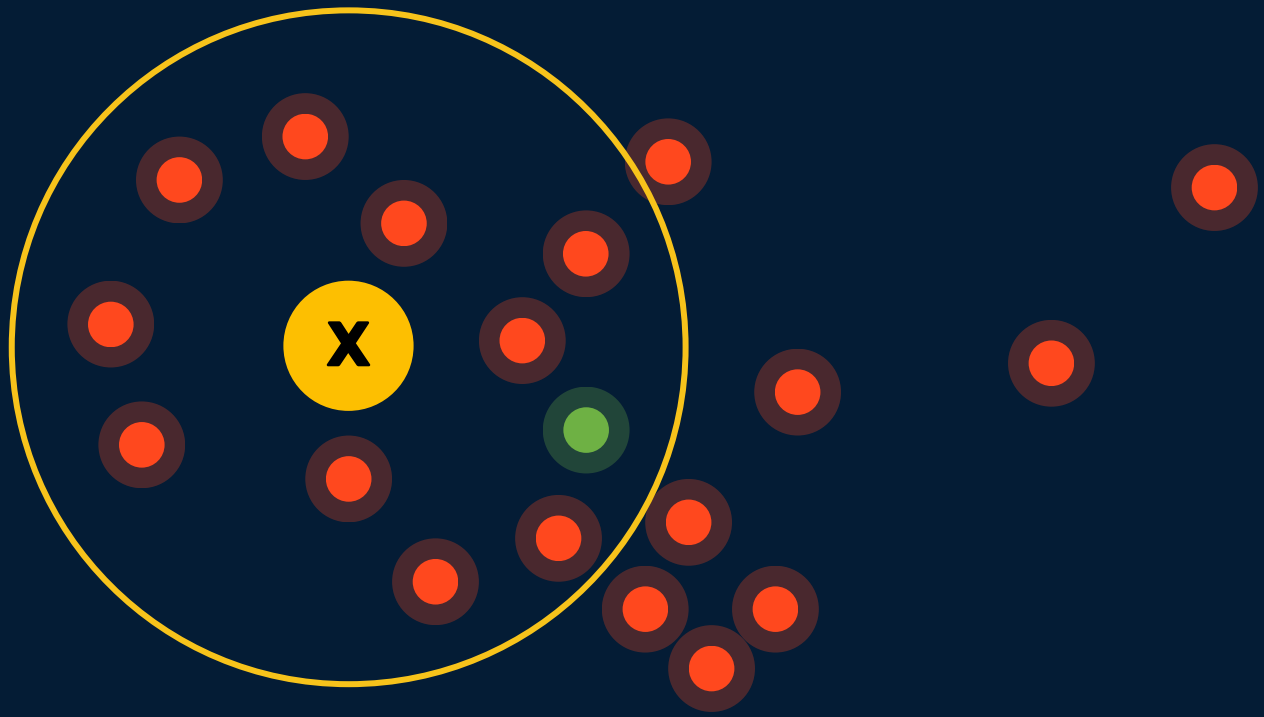
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



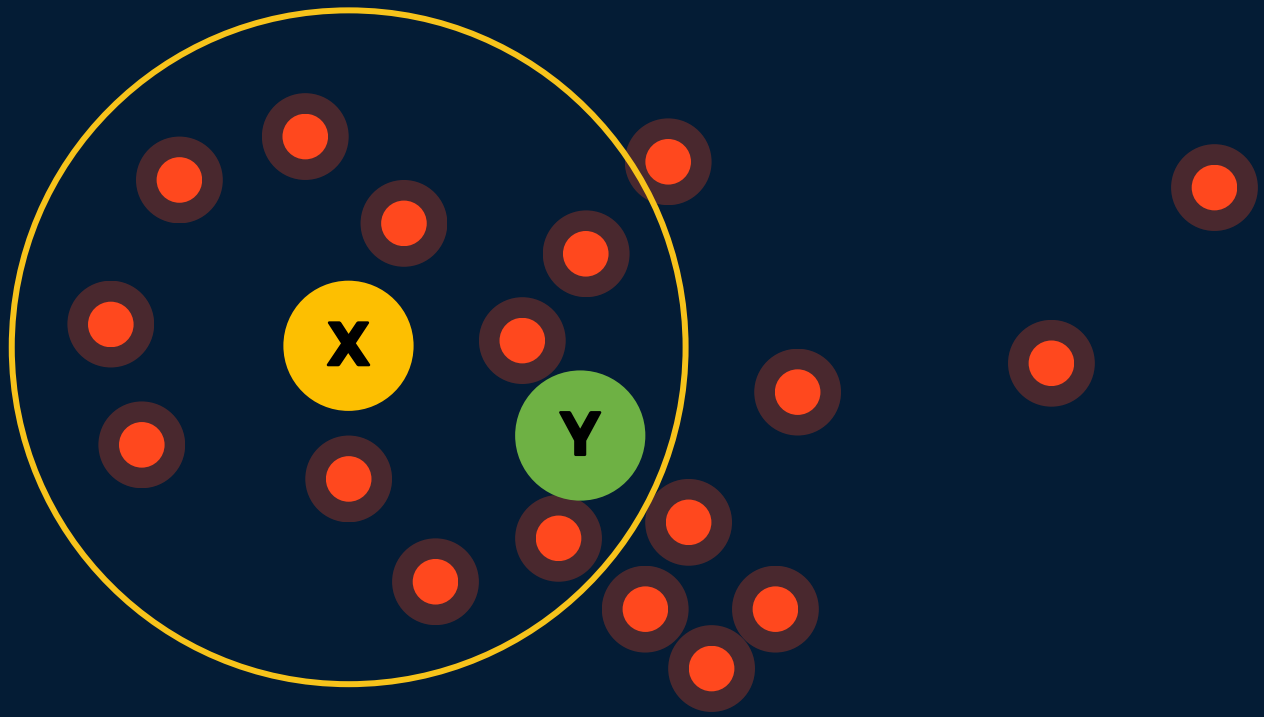
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



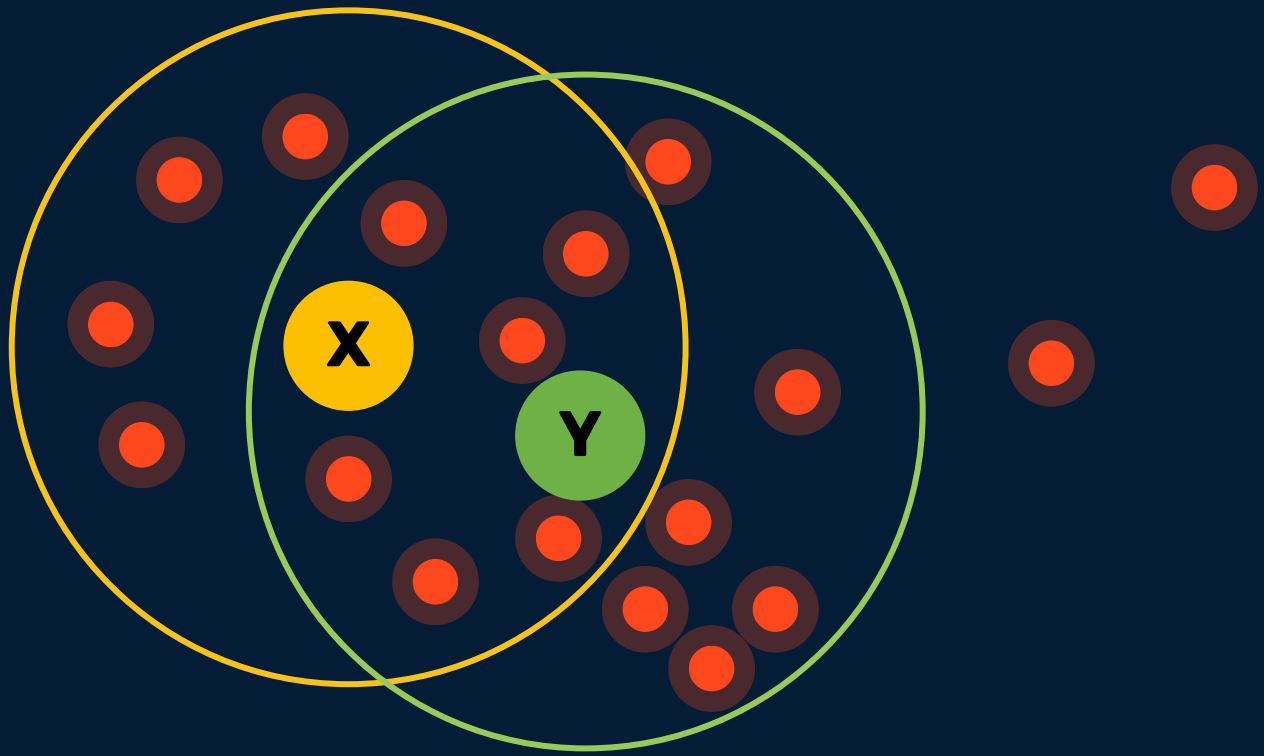
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



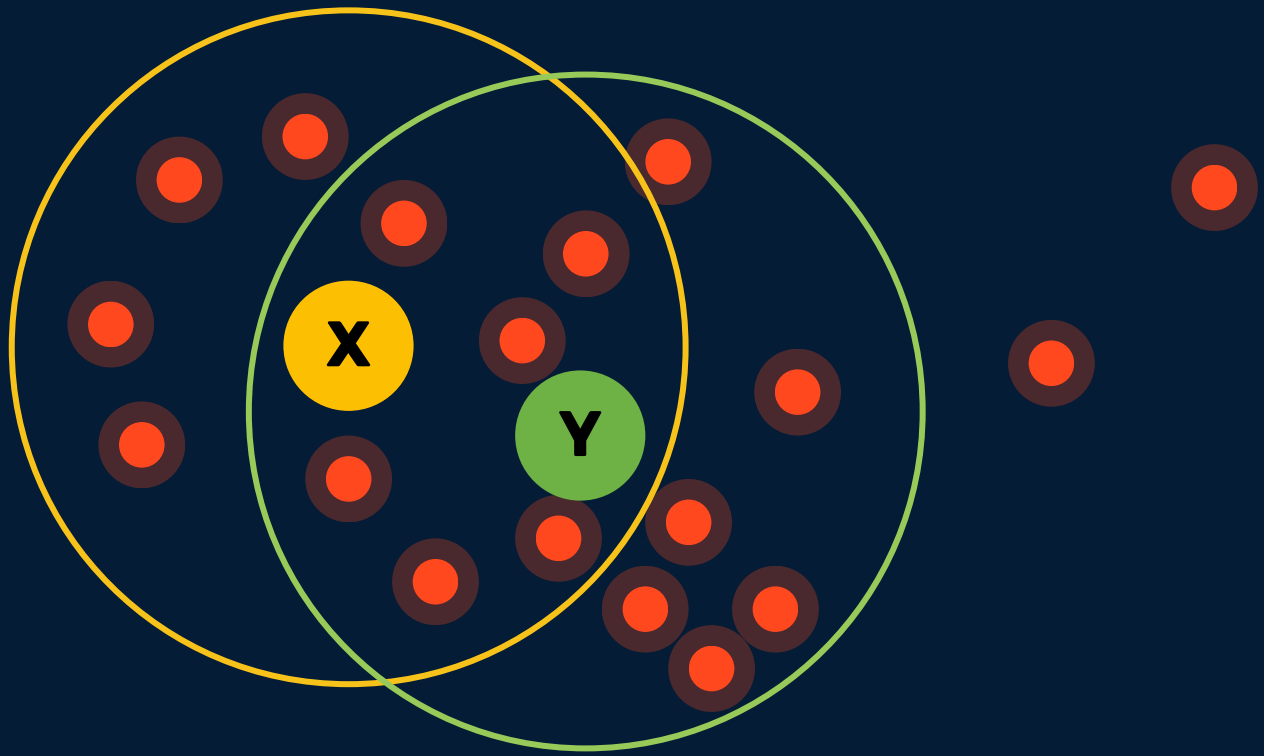
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



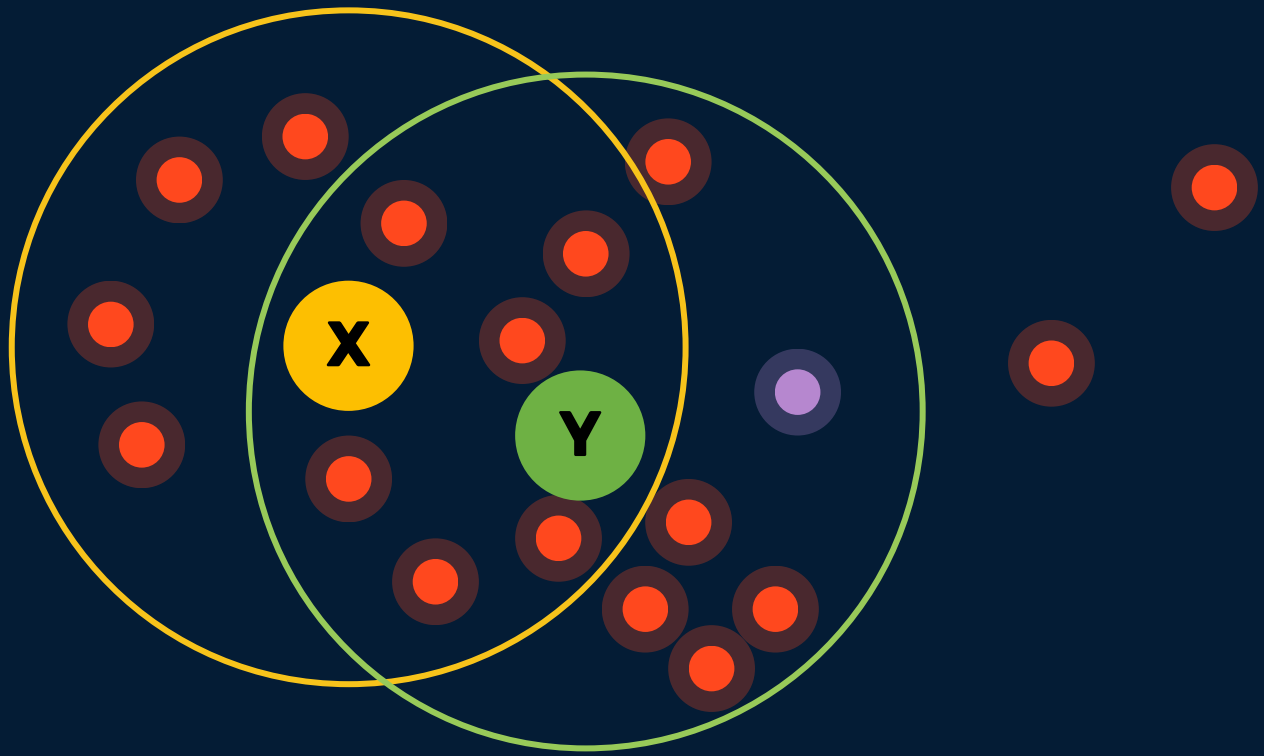
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



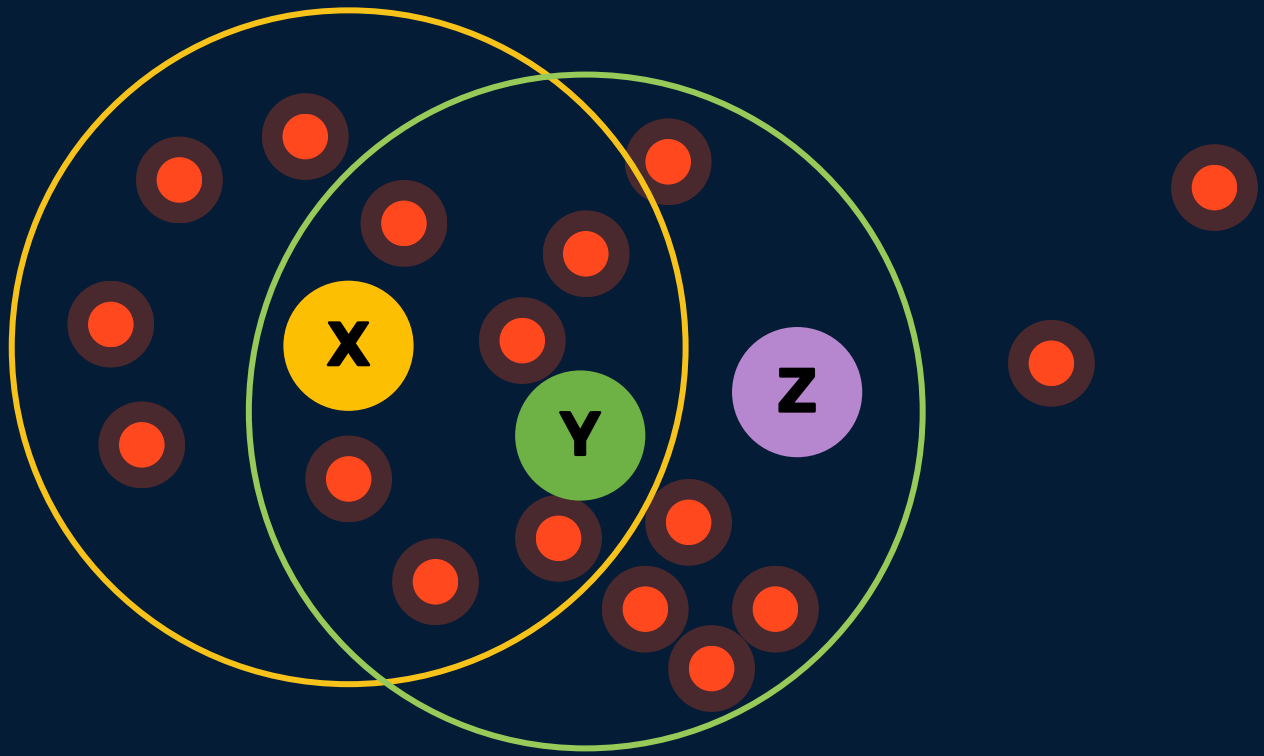
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



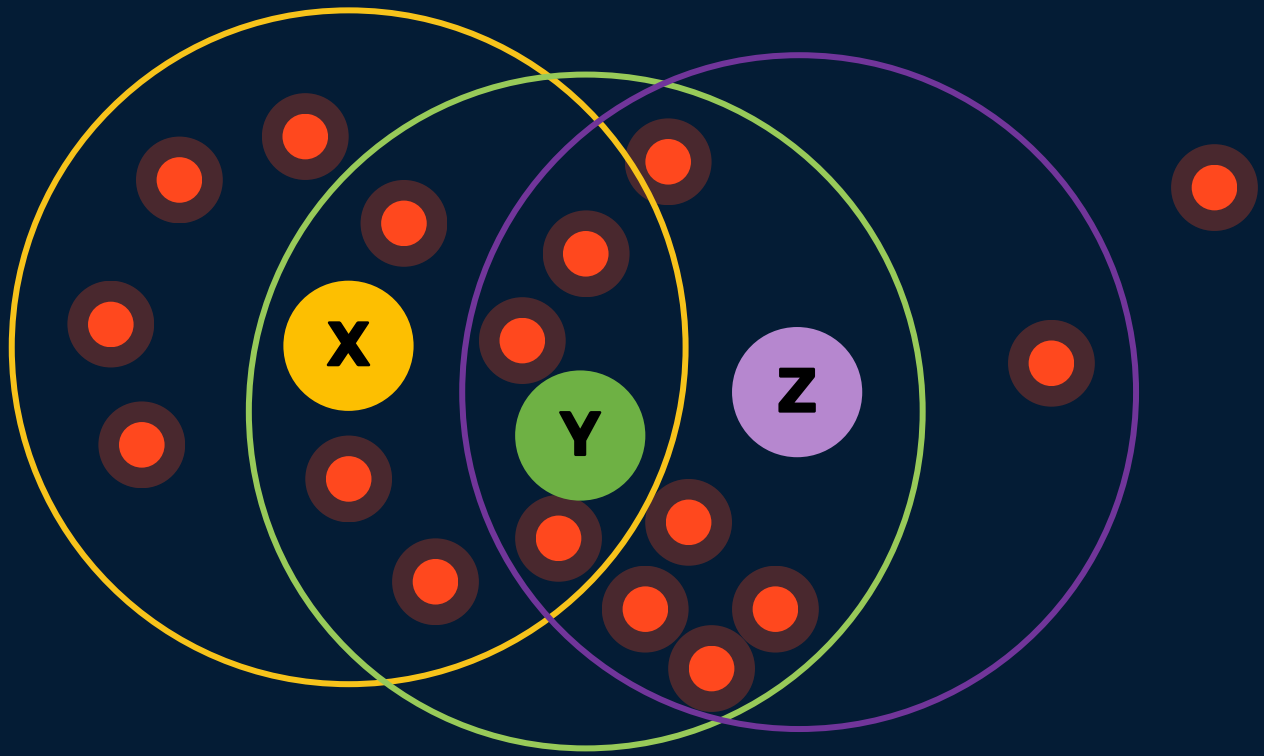
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



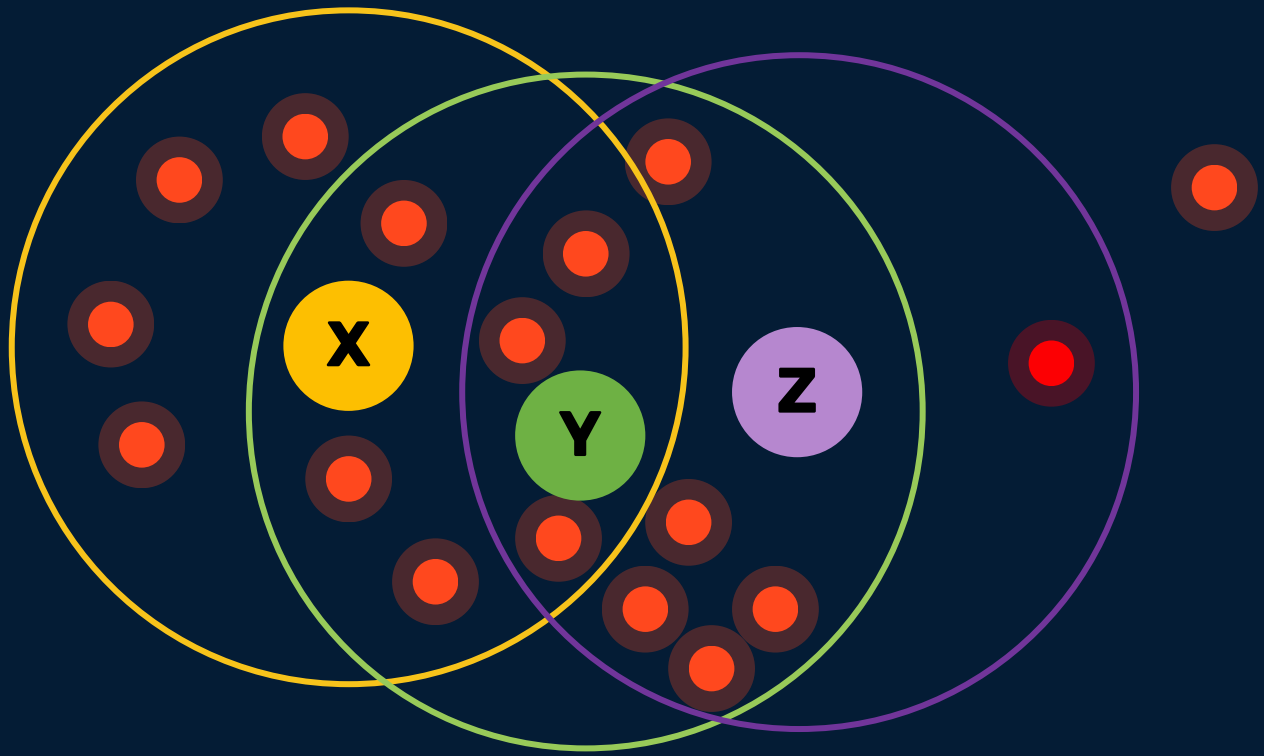
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



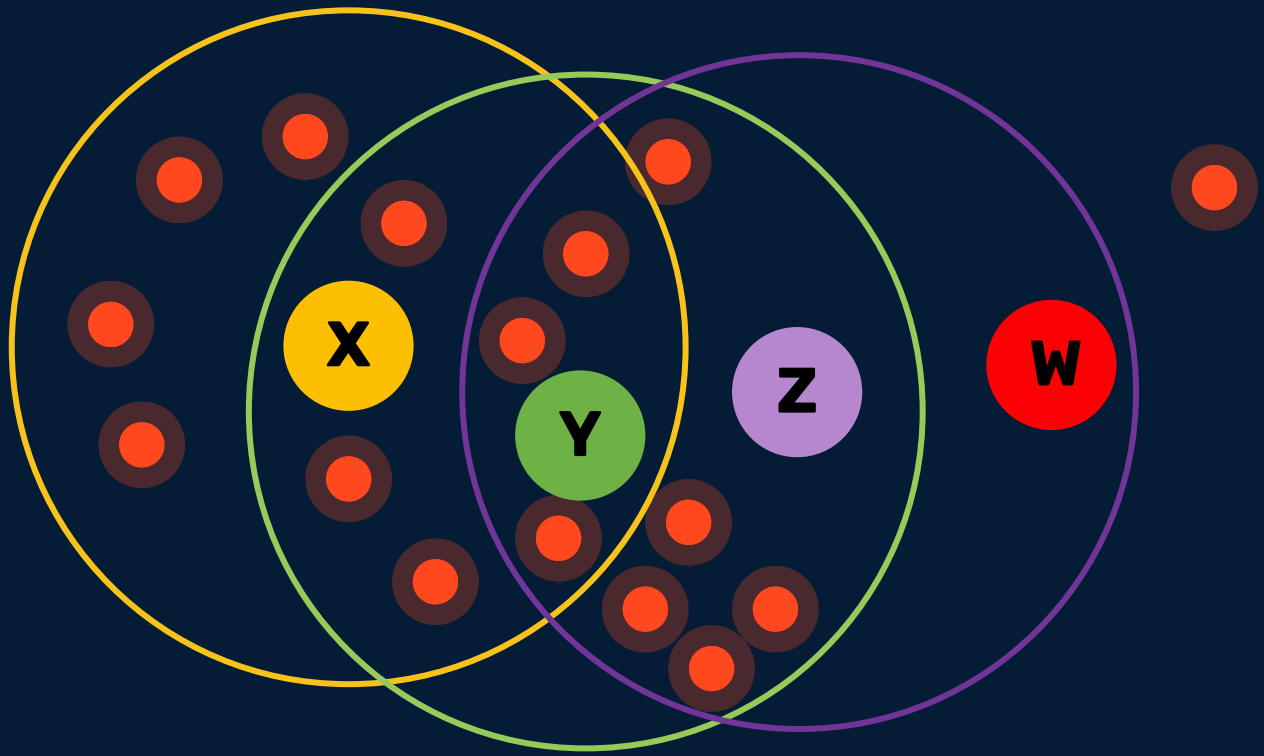
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



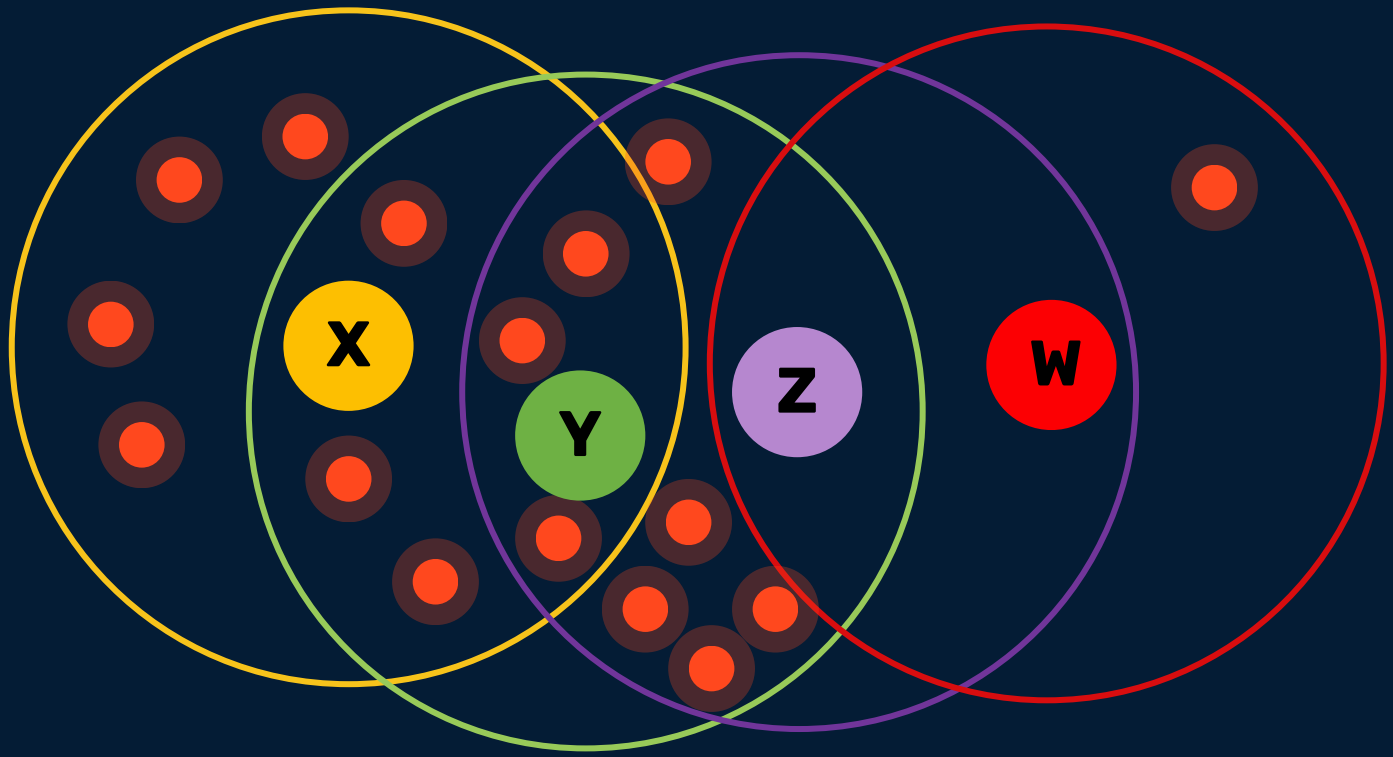
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



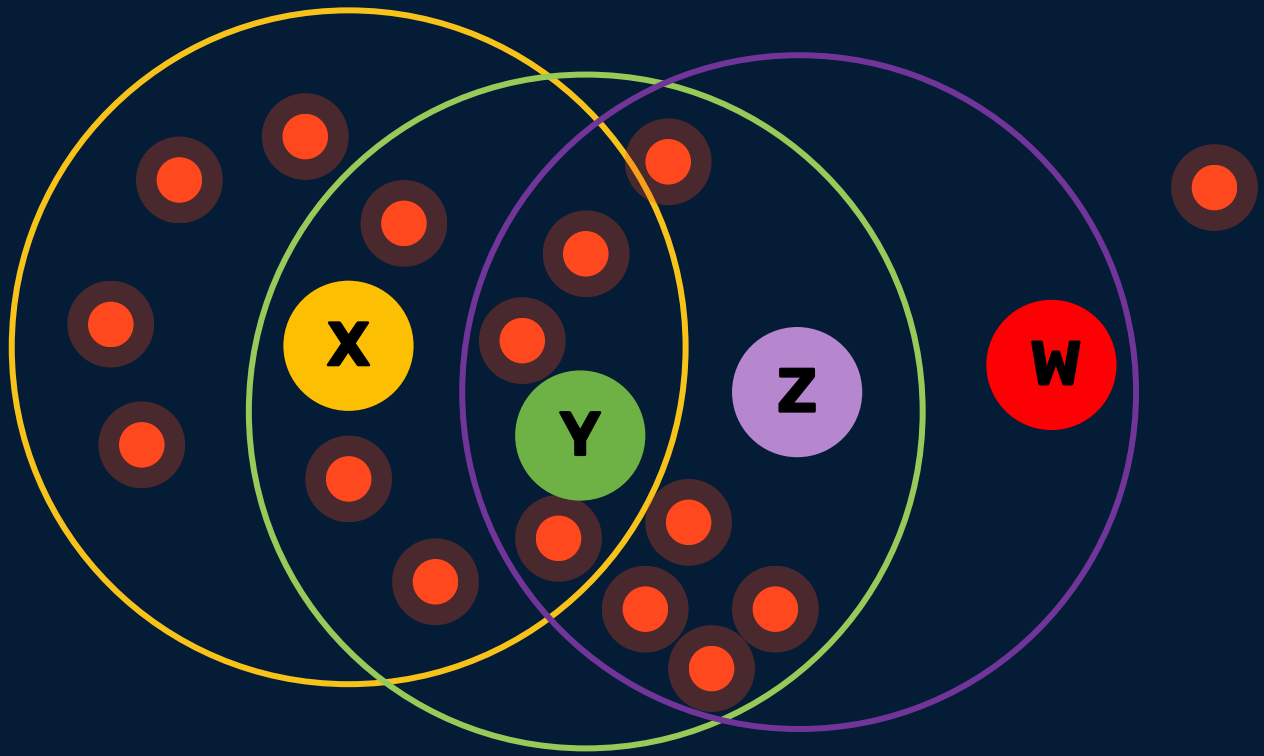
Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



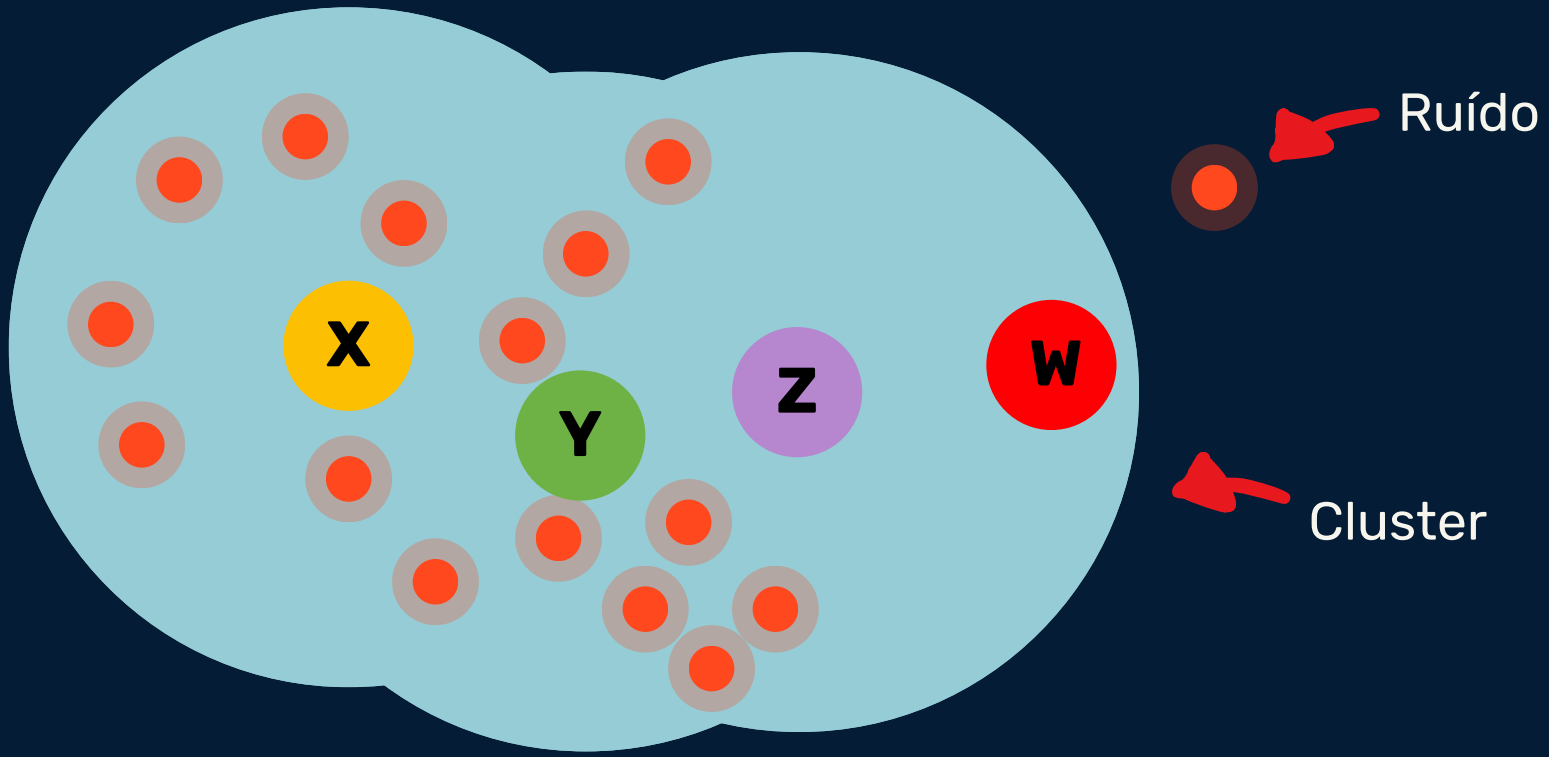
Procedimento

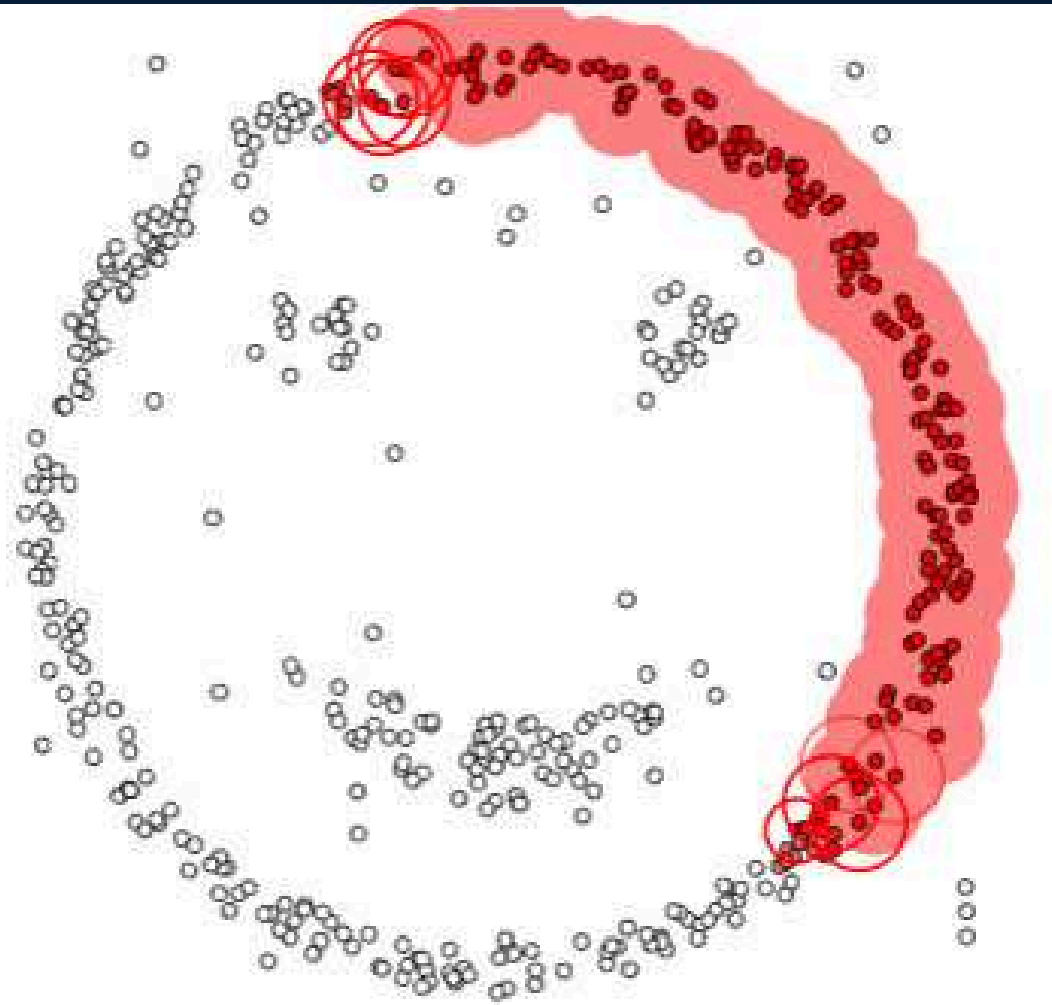
Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.



Procedimento

Os pontos das vizinhanças densas dão origem a mais vizinhanças densas, e esses pontos são agrupados. Quando deixa de ser possível criar mais vizinhanças, um novo ponto de partida é tomado.

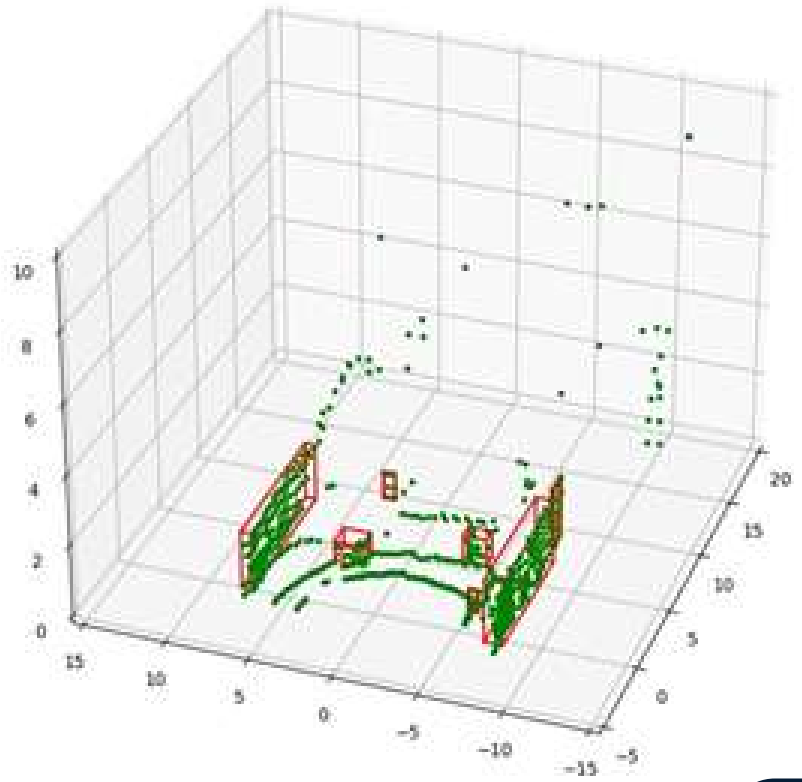




epsilon = 1.00
minPoints = 4

Onde é usado?

Usado para clusterizar os pontos de um LiDAR.



Estrutura de dados usado

Para o conjunto de dados foi usado um Array Nx2, descrito como:

- pontos[i][0] -> Valor x do ponto i.
- pontos[i][1] -> Valor y do ponto i

```
static const float points[NUM_POINTS][NUM_FEATURES] = {
    { 472.431845, 133.637138 },
    // ......
}
```

Para o ids clusters foi usado um Struct, “SetOfPoints” descrito como:

- clusterIds[i] -> Id do cluster onde o ponto “i” está.
- size -> Numero de pontos que já tem cluster.

```
typedef struct {
    int clusterIds[MAX_POINTS];
    int size;
} SetOfPoints;
```

Complexidade Temporal

Por passar por n pontos n vezes.

$$O(n^2)$$

Complexidade Espacial

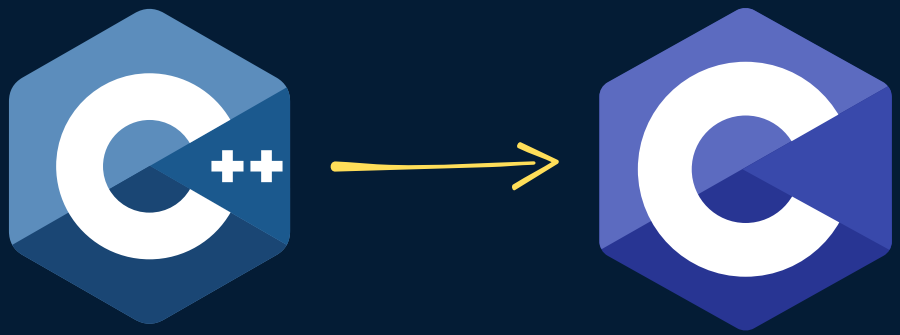
Array necessário para armazenar os clusters IDs.

$$O(n)$$

As Dificuldades

Implementação em C

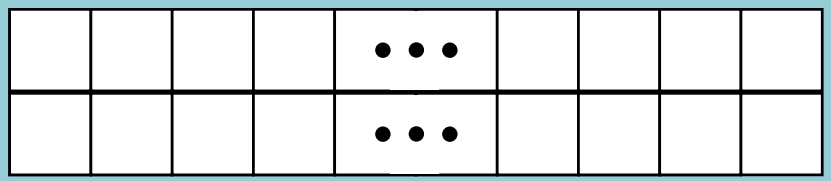
Não existe uma implementação em C na internet



Adaptamos de uma implementação em C++

Dimensão dos dados

Decidir qual conjuntos de dados usar. Difícil achar dados densos, de 2 dimensões ou 3 dimensões.



Optamos por usar um conjunto de dados já bi dimensionados, de um estudo anterior do DBSCAN.

Parâmetros de Ajuste

É desafiador escolher um bom epsilon e um bom minPts.

ε

Usamos um conjunto de dados onde os parâmetros são conhecidos.

Saber se funcionou

Como saber se o algoritmo processou todos os dados?

FIzemos uma contagem de quantos pontos existem em quais clusters, e somamos para ver se dá a quantidade inicial.

Os Testes

Escolha do conjunto de dados

Para teste, foi escolhido o conjunto de dados de sequência de RNA (Hi-Seq) PANCAN, que contém características genéticas de pacientes com 5 tipos de tumores:

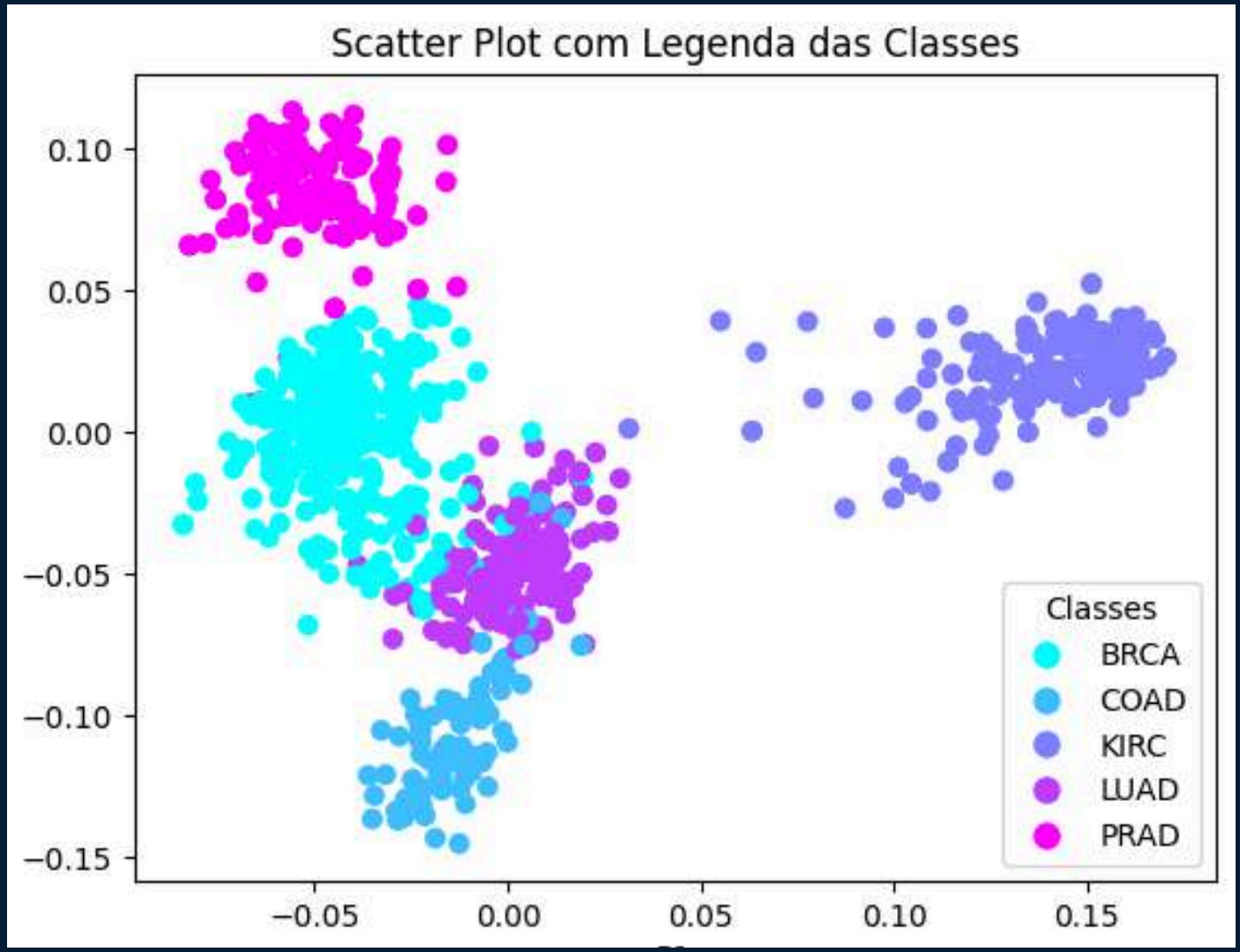
- BRCA
- KIRC
- COAD
- LUAD
- PREAD

	P1	P2
0	-0.056920	0.082045
1	-0.001607	-0.082631
2	-0.066280	0.010366
3	-0.077986	0.066873
4	-0.063931	0.011263
...	...	...
999	-0.056405	0.024994
1000	-0.012920	-0.050042
1001	0.008534	-0.024661
1002	-0.050052	0.080348
1003	-0.044771	0.043976

1004 rows × 2 columns

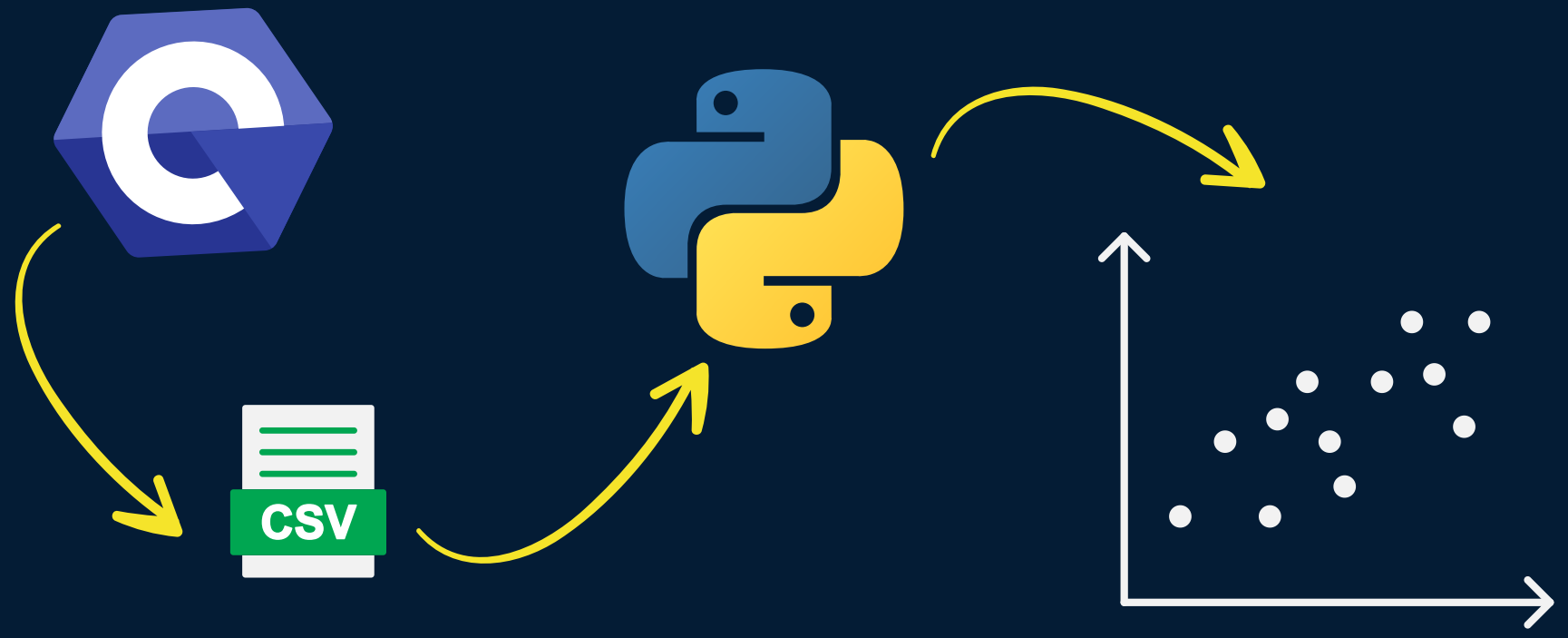
Gráfico do estudo

O gráfico abaixo apresenta o estudo da universidade de medicina.



Execução do Algoritmo em C

Usamos um script para compilar e executar o algoritmo em C, bem como executar alguns passos extras.

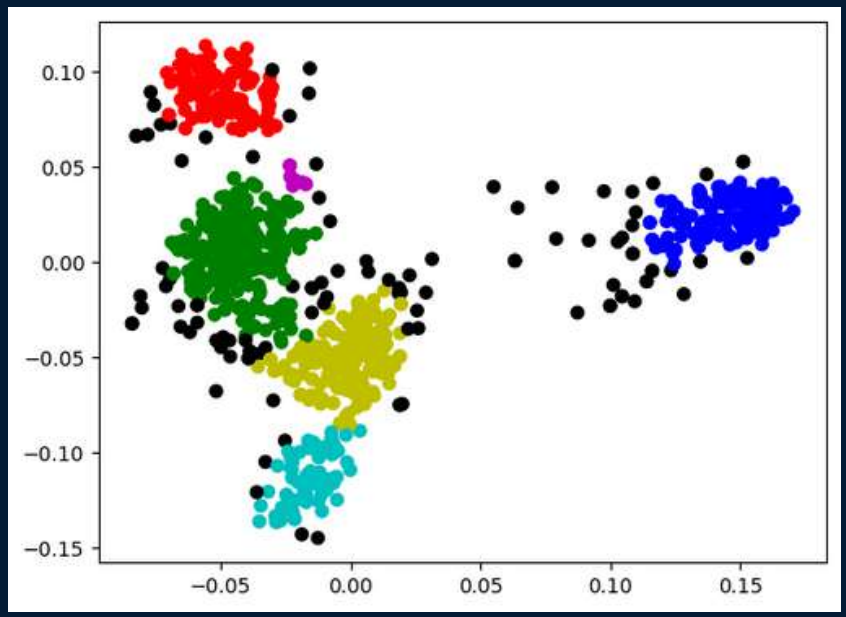


Execução do Algoritmo em Python

Para fins de comparação, foi usado o método DBSCAN da biblioteca Scikit-Learn.

```
from sklearn.cluster import DBSCAN

db_default = DBSCAN(eps = 0.008, min_samples = 10).fit(df_principal)
labels = db_default.labels_
```



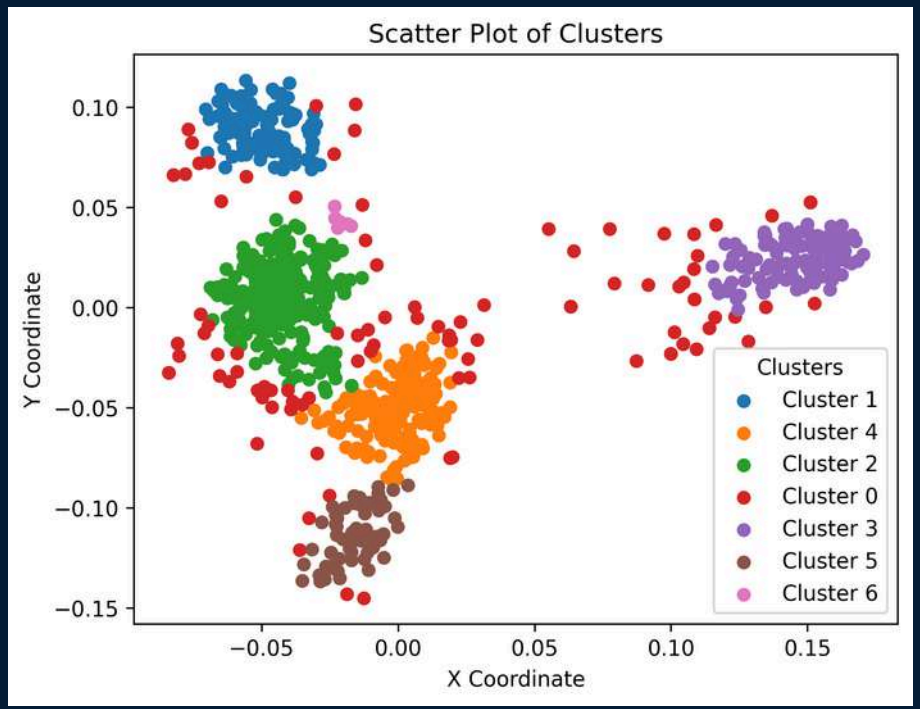
Saída do código

Algumas das saídas são descritas abaixo:

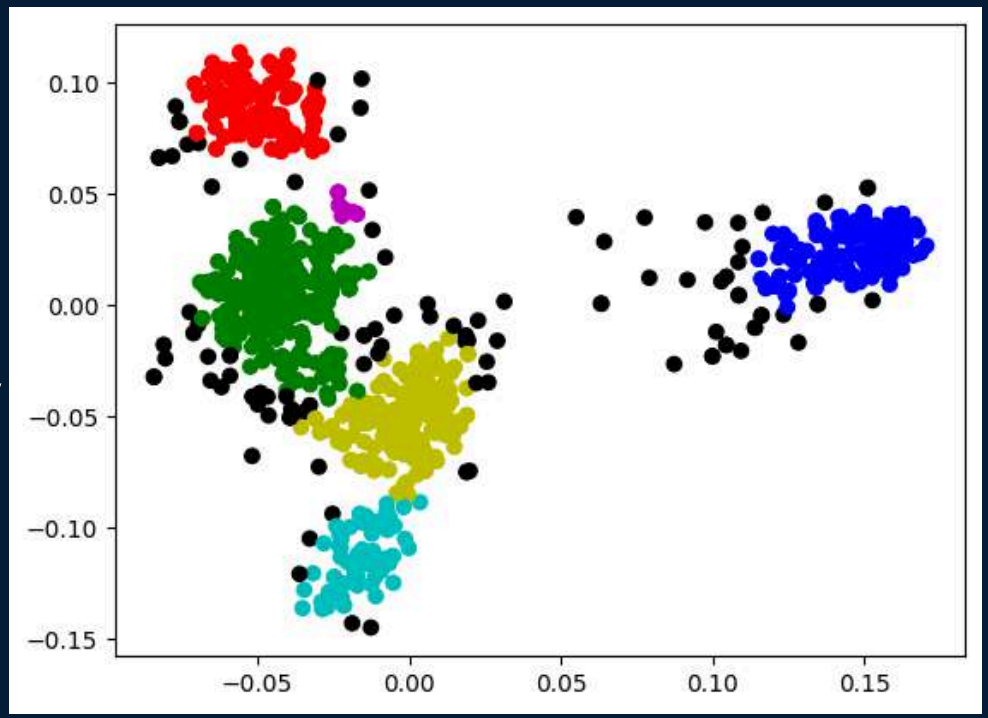
```
Compiling C implementation...
Compilation successful
Running C implementation...
Pontos : 1004, eps : 0.008, MinPoints : 10
Ruido : 109  |  Grupo 1 : 149  |  Grupo 2 : 326  |  Grupo 3 : 145  |  Grupo 4 : 192  |  Grupo 5 : 73  |  Grupo 6 : 10  |
Resultados exportados para: ../dados/clustering_results.csv
C implementation completed successfully
Running Python visualization...
Scatter plot saved as: visual-output/clustering_scatter_plot.png
Python visualization completed successfully
All tasks completed successfully
Pressione qualquer tecla para continuar. . .
```

Comparação

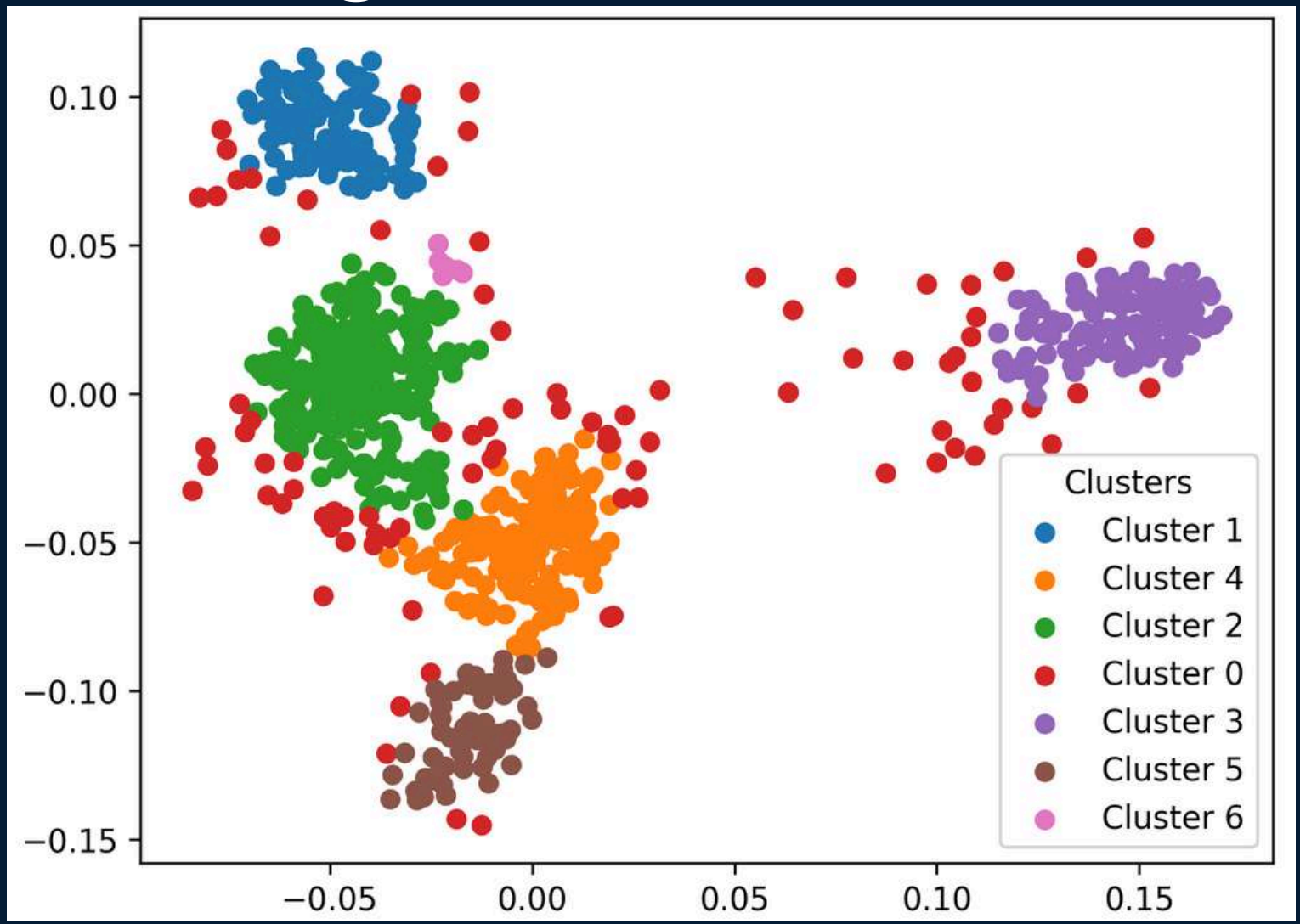
Comparamos a saída do nosso código em C, com a saída do código em python.



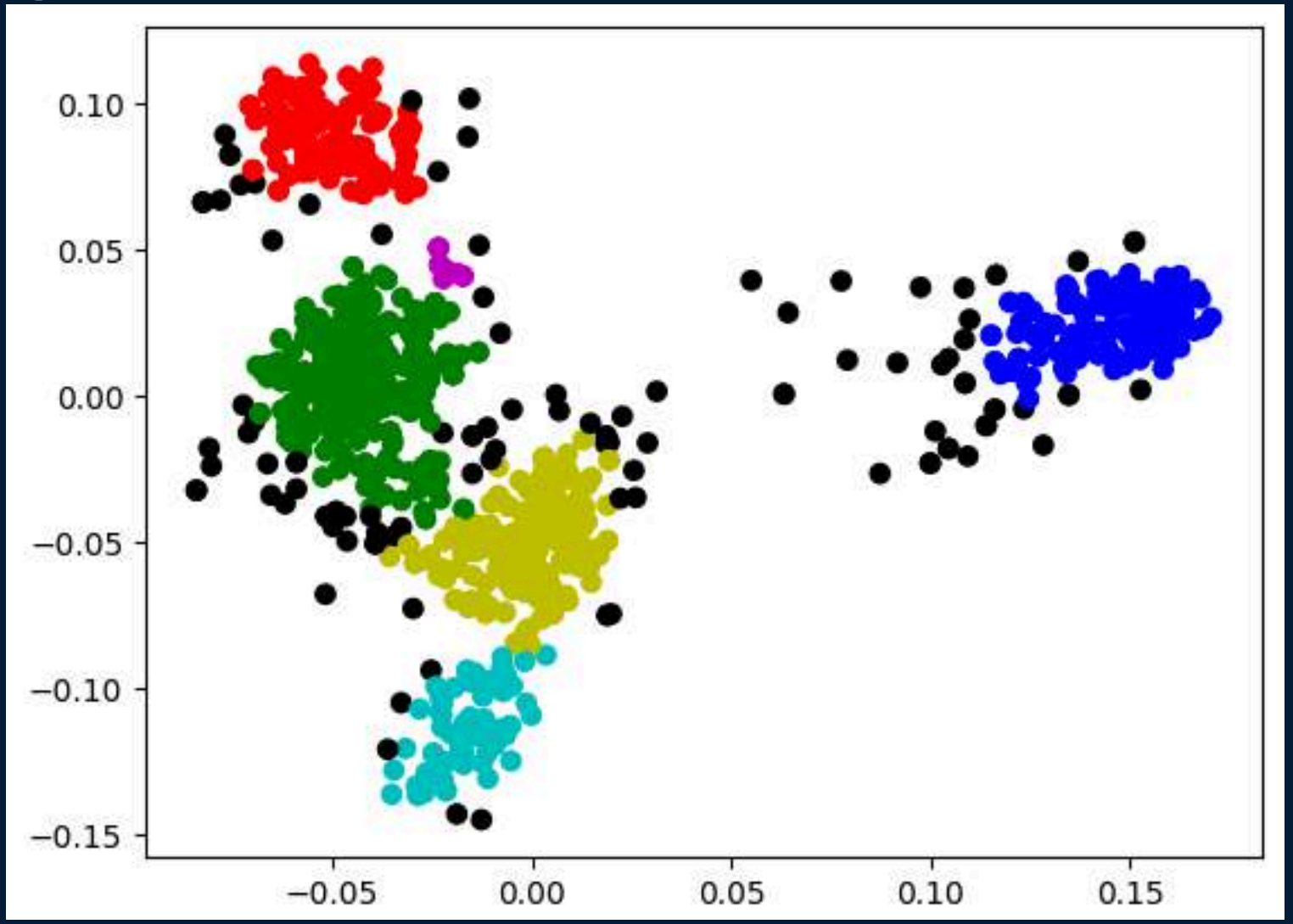
-Vs-



Nosso Algoritmo



Algoritmo do Scikit Learn



Comparação numérica

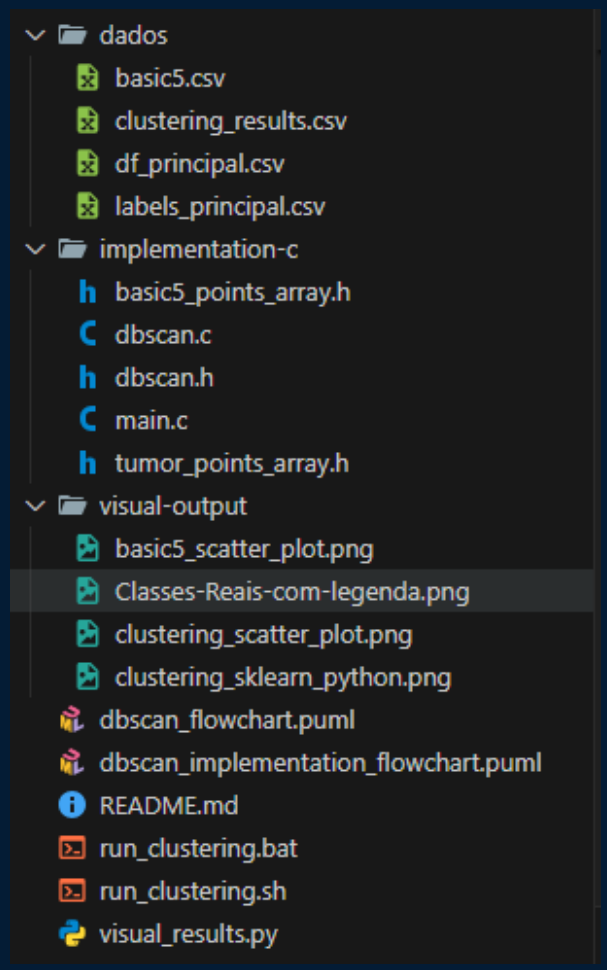
```
Arquivos iguais
      CSV1_Coluna3  CSV2_Coluna3  Iguais
0                1             1    True
1                4             4    True
2                2             2    True
3                0             0    True
4                2             2    True
...            ...           ...     ...
999              2             2    True
1000             4             4    True
1001             4             4    True
1002             1             1    True
1003             2             2    True

[1004 rows x 3 columns]
```

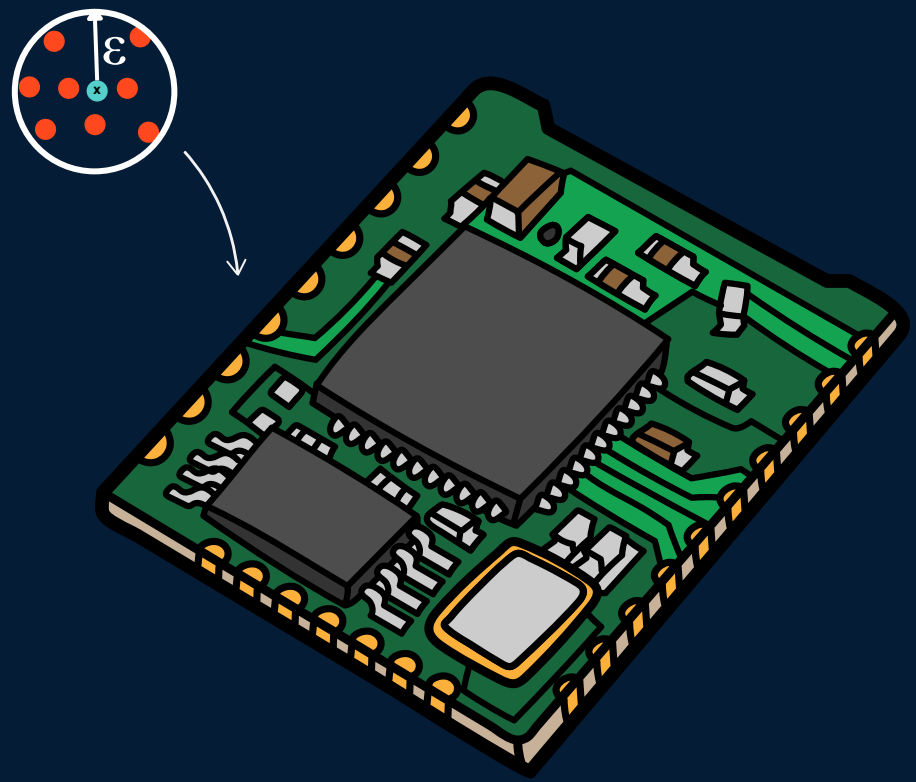
Considerações

Reuso de Software

Para manter o reuso de software, separamos os arquivos como abaixo:



Aprendizado de máquina em embarcados.



Bibliografia

Artigos Online:

- https://gabriellm.medium.com/entendendo-dbscan-770f680d9160
- https://towardsdatascience.com/a-practical-guide-to-dbscan-method-d4ec5ab2bc99
- https://medium.com/@sachinsoni600517/clustering-like-a-pro-a-beginners-guide-to-dbscan-6c8274c362c4
- https://medium.com/analytics-vidhya/all-you-need-to-know-about-the-dbscan-algorithm-f1a35ed8e712

Artigo Original

- https://cdn.aaai.org/KDD/1996/KDD96-037.pdf

Implementação do GitHub

- https://github.com/kelvins/DBSCAN/tree/master

PERGUNTAS?

OBRIGADO!