HDBSCAN

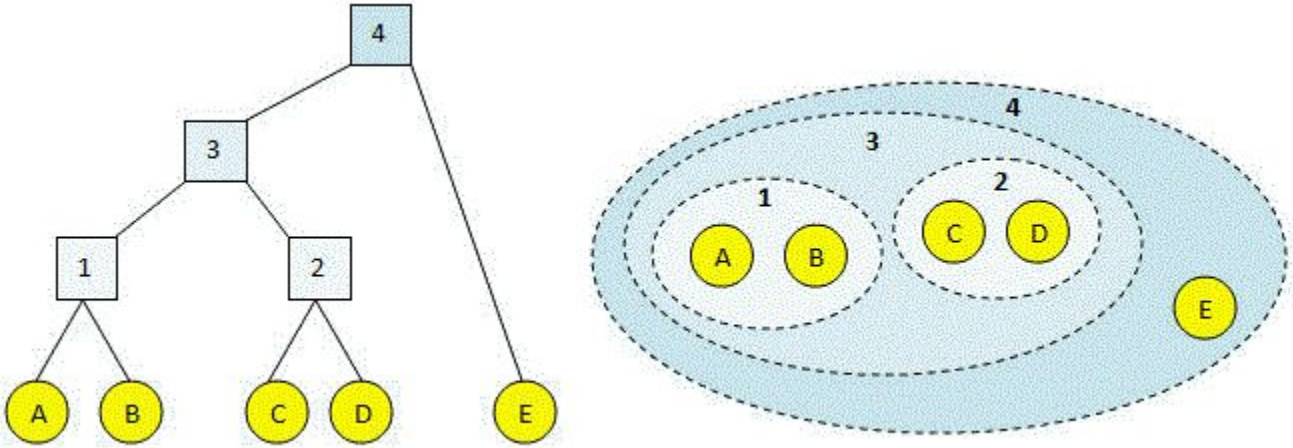
O Estado da Arte de Algoritmos de Agrupamento.
Um Estudo Sobre a Evolução do DBSCAN.

Sumário

- SINGLE LINKAGE
- HDBSCAN
- HDBSCAN - IMPLEMENTAÇÃO EM PYTHON

SINGLE LINKAGE - O QUE É?

É um dos vários métodos de agrupamento **hierárquico**, o qual se baseia na **distância mínima** para realizar as junções (merge) de clusters (ou elementos) que possuem, no caso deste algoritmo, a menor distância.



SINGLE LINKAGE - COMO FUNCIONA?

Os Passos do Algoritmos São:

- **Calcular as distâncias** (nesse caso, a euclidiana) de todos os elementos e criar a **matriz de distância**.
- Procurar a **menor distância** entre os elementos (ou clusters) e fazer a **junção** (merge).
- **Recalcular a matriz de distância** de acordo com o novo cluster. (prevalece a menor distância)
- **Repetir os passos 2 e 3** até que todos os elementos estejam hierarquizados.

SINGLE LINKAGE - PASSO 1

Nesse passo, é necessário calcular a distância (euclidiana) entre todos os elementos e, posteriormente, criar a matriz de distância.

	X	Y
P1	7	83
P2	85	14
P3	66	89
P4	49	64
P5	80	46

	P1	P2	P3	P4	P5
P1	0.0	0.0	0.0	0.0	0.0
P2	104.0	0.0	0.0	0.0	0.0
P3	59.0	77.0	0.0	0.0	0.0
P4	46.0	62.0	30.0	0.0	0.0
P5	82.0	32.0	45.0	36.0	0.0

SINGLE LINKAGE - PASSO 2

Nesse passo, é procurado a menor distância entre elementos (ou cluster) e é realizada a junção entre eles.

	P1	P2	P3	P4	P5
P1	0.0	0.0	0.0	0.0	0.0
P2	104.0	0.0	0.0	0.0	0.0
P3	59.0	77.0	0.0	0.0	0.0
P4	46.0	62.0	30.0	0.0	0.0
P5	82.0	32.0	45.0	36.0	0.0

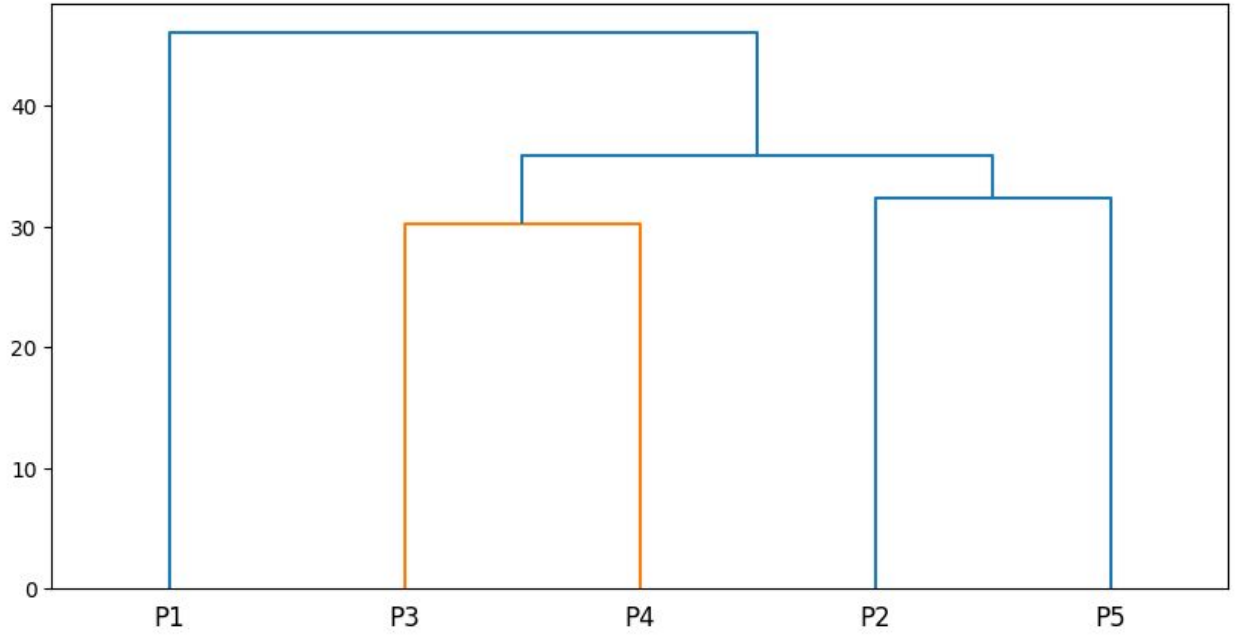
SINGLE LINKAGE - PASSO 3

Nesse passo, com a junção feita, é recalculada a matriz de distância, a qual preserva-se a menor distância.

	P1	P2	P3,P4	P5
P1	0	0	0	0
P2	104	0	0	0
P3,P4	46	62	0	0
P5	82	32	36	0

SINGLE LINKAGE - PASSO 4

Nesse passo, é feita a repetição dos passos 2 e 3 até que todos os pontos estejam hierarquizados. Segue abaixo o dendograma finalizado.

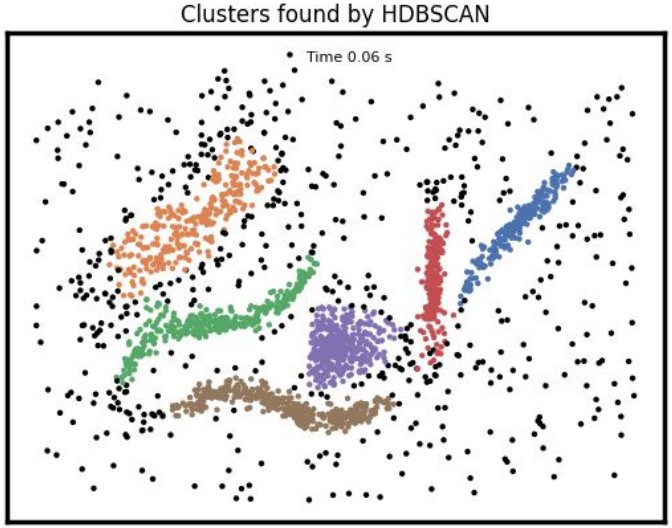


SINGLE LINKAGE - COMPLEXIDADE

O single Linkage tem complexidade **O(n^2)**, visto que calcula as distâncias de um cada ponto com cada ponto.

HDBSCAN

O HDBSCAN (*Hierarchical Density-based Spatial Clustering of Applications with Noise)* é um algoritmo de agrupamento derivado da junção entre o DBSCAN (Density-based) e o SINGLE LINKAGE (Hierarchical).



HDBSCAN - DEFINIÇÕES

- **Core Distance** (Distância do Núcleo) - é a distância do elemento x até atingir o último ponto necessário para obter o número mínimo de pontos dentro da hiperesfera.
- **Mutual Reachability Distance** (Distância de Alcance Mútuo) - é relação entre dois elementos seguida pela fórmula:

  $$Dmrd(x,y) = \max \{ coredistance(x), coredistance(y), distance(x,y) \}$$

- **Mutual Reachability Graph** (Grafo de Alcance Mútuo) - é um grafo com os elementos sendo as vértices e os pesos nas arestas sendo a Mutual Reachability Distance.

HDBSCAN - PROPOSIÇÃO 3.4

PROPOSITION 3.4. *Let* $\mathbf{X}$ *be a set of* n *objects described in a metric space by* $n \times n$ *pairwise distances. The clustering of this data obtained by DBSCAN* w.r.t* m_{pts} *and some value* ε *is identical to the one obtained by first running Single-Linkage over the transformed space of mutual reachability distances (w.r.t* m_{pts}*), then, cutting the resulting dendrogram at level* ε *of its scale, and treating all resulting singletons with* $d_{\text{core}}(\mathbf{x}_p) > \varepsilon$ *as a single class representing "Noise".*

HDBSCAN - PROPOSIÇÃO 3.4

- Caso transformemos o espaço de distâncias através da Mutual Reachability Distance é possível criar um DBSCAN hierárquico.
- O HDBSCAN será o DBSCAN com todos os valores de epsilon.

HDBSCAN - PARAMETROS

- O único parâmetro necessário para o agrupamento é o MinPoints (Número mínimo de pontos).

HDBSCAN - ALGORITMO

O algoritmo do HDBSCAN é dividido em algumas partes importantes, sendo elas:

- Criação da HDBSCAN* hierarchy.
- Extração de Cluster Estáveis.
- Detecção de Outliers

HDBSCAN - Hierarchy

Para extrair a HDBSCAN* Hierarchy é necessário seguir os passos abaixo:

- Calcular as coredistance para todos os elementos, de acordo com o MinPoints.
- Calcular a Minimum Spanning Tree (MST) do Mutual Reachability Graph (G).
- Obter a MST estendida.
- Extrair a Hierarchy da MST estendida.

HDBSCAN - Hierarchy - MST

- Dado o Mutual Reachability Graph é possível obter a Minimum Spanning Tree, que é o grafo com caminho que passe por todos os elementos que possui o menor custo.

HDBSCAN - Hierarchy - MST Estendida

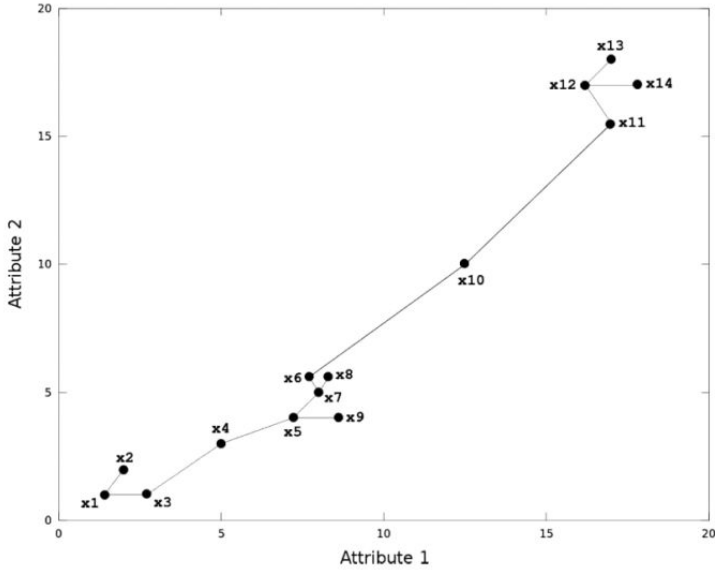
- Dada a MST é possível chegar na MST estendida adicionando self loops em cada vértice com o Mutual Reachability Distance como peso.

HDBSCAN - Hierarchy - Extração

Para a extração da Hierarchy é necessário realizar alguns passos com a MST estendidas, sendo eles:

- Inicialmente, pela raiz atribua a todos os elementos o mesmo cluster.
- Remover, iterativamente, todas as arestas em ordem decrescente.
    - Antes de cada remoção, definir o valor da escala do dendograma do nível corrente como sendo o peso das arestas a ser eliminadas.
    - Após cada remoção, atribuir os clusters dos componentes conectados que restaram; atribuir uma nova etiqueta de agrupamento a um componente se este ainda tiver pelo menos uma aresta; ou atribuir ruído.

HDBSCAN - Hierarchy - Resultado



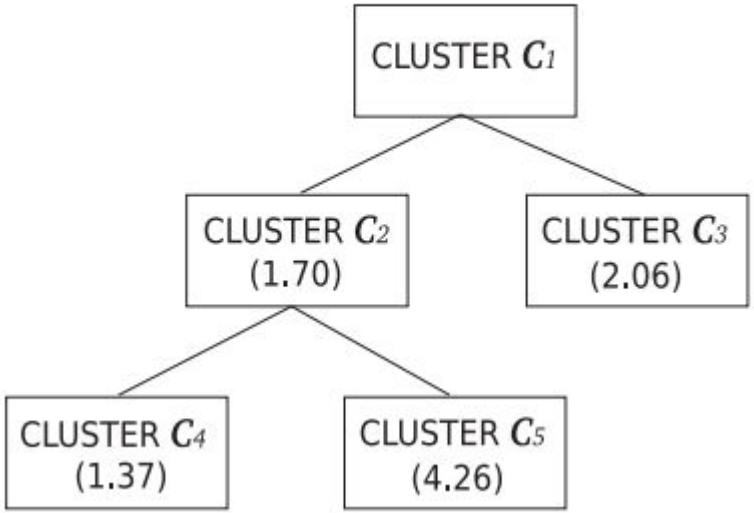
$\mathbf{x}_1$	1	2	2	4	4	4	4	0	0	0	0
$\mathbf{x}_2$	1	2	2	4	4	4	4	4	4	0	0
$\mathbf{x}_3$	1	2	2	4	4	4	4	0	0	0	0
$\mathbf{x}_4$	1	2	2	0	0	0	0	0	0	0	0
$\mathbf{x}_5$	1	2	2	5	5	5	0	0	0	0	0
$\mathbf{x}_6$	1	2	2	5	5	5	5	5	5	5	0
$\mathbf{x}_7$	1	2	2	5	5	5	5	5	5	5	0
$\mathbf{x}_8$	1	2	2	5	5	5	5	5	5	5	0
$\mathbf{x}_9$	1	2	2	5	5	5	0	0	0	0	0
$\mathbf{x}_{10}$	1	2	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{11}$	1	3	3	3	0	0	0	0	0	0	0
$\mathbf{x}_{12}$	1	3	3	3	3	0	0	0	0	0	0
$\mathbf{x}_{13}$	1	3	3	3	3	3	3	3	0	0	0
$\mathbf{x}_{14}$	1	3	3	3	3	0	0	0	0	0	0
ε	7.1	6.51	3.04	1.7	1.6	1.4	1.3	1.28	1.22	0.67	0

HDBSCAN - Estabilidade - Definição

- $\lambda = 1 / \varepsilon$
- A estabilidade de um cluster (S) é definida de acordo com a fórmula:

$$S(\mathbf{C}_i) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \left(\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i) - \lambda_{\min}(\mathbf{C}_i) \right)$$

HDBSCAN - Estabilidade - Resultado



HDBSCAN - Estabilidade - Escolha dos Clusters

- Cada nó de cluster é processado (menos a raiz), a estabilidade total (St) de cada cluster se dá:

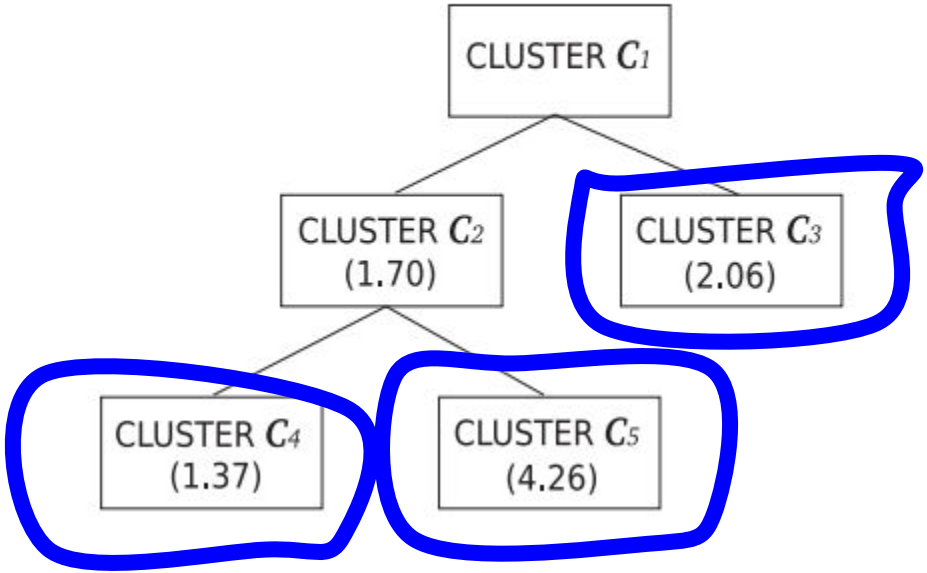
$$\hat{S}(\mathbf{C}_i) = \begin{cases} S(\mathbf{C}_i), & \text{if } \mathbf{C}_i \text{ is a leaf node} \\ \max\{S(\mathbf{C}_i), \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})\} & \text{if } \mathbf{C}_i \text{ is an internal node} \end{cases}$$

HDBSCAN - Estabilidade - Escolha dos Clusters

- Inicialize δ=1 para cada nó de cluster.
- Inicialize St = S, Estabilidade total = Estabilidade do Cluster.

2. Starting from the deepest levels, do bottom-up (except for the root):
   2.1 If $S(\mathbf{C}_i) < \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$, set $\hat{S}(\mathbf{C}_i) = \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$ and set $\delta_i = 0$.
   2.2 Else: set $\hat{S}(\mathbf{C}_i) = S(\mathbf{C}_i)$ and set $\delta_{(.)} = 0$ for all clusters in $\mathbf{C}_i$'s subtrees.

HDBSCAN - Estabilidade - Escolha dos Clusters



HDBSCAN - Detecção de Outliers

- A detecção de outliers é feita pela GLOSH (Global-Local Outliers Score from Hierarchies).
- A GLOSH é feita para cada elemento seguindo a fórmula:

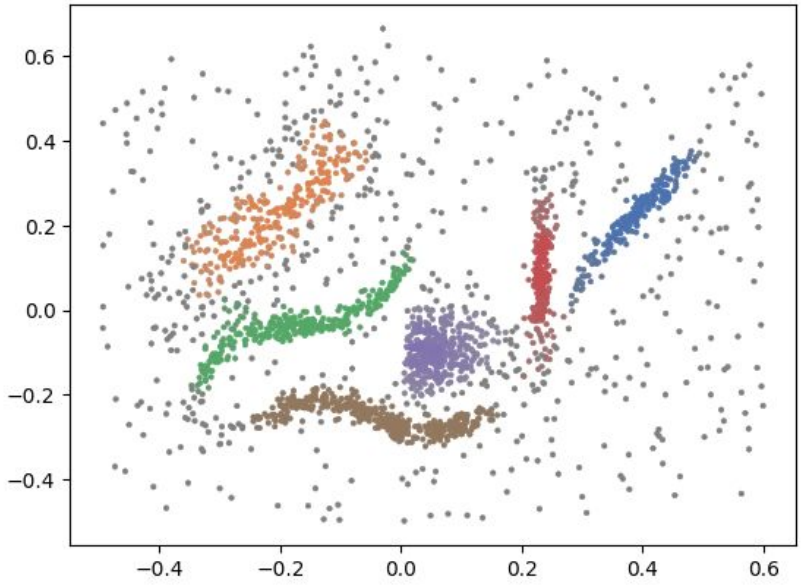
$$\mathrm{GLOSH}(\mathbf{x}_i) = \frac{\lambda_{\max}(\mathbf{x}_i) - \lambda(\mathbf{x}_i)}{\lambda_{\max}(\mathbf{x}_i)} = \frac{\frac{1}{\varepsilon_{\max}(\mathbf{x}_i)} - \frac{1}{\varepsilon(\mathbf{x}_i)}}{\frac{1}{\varepsilon_{\max}(\mathbf{x}_i)}} = 1 - \frac{\varepsilon_{\max}(\mathbf{x}_i)}{\varepsilon(\mathbf{x}_i)}$$

- Quanto mais próximo de 1, mais outlier é o elemento. Quanto mais próximo de 0, mais inlier (centro denso do cluster).

Implementação em Python

```
clusterer = hdbscan.HDBSCAN(min_cluster_size=15).fit(data)
color_palette = sns.color_palette('deep', 8)
cluster_colors = [color_palette[x] if x >= 0
                  else (0.5, 0.5, 0.5)
                  for x in clusterer.labels_]
cluster_member_colors = [sns.desaturate(x, p) for x, p in
                         zip(cluster_colors, clusterer.probabilities_)]
plt.scatter(*data.T, s=7, linewidth=0.3, c=cluster_member_colors, alpha=1)
```

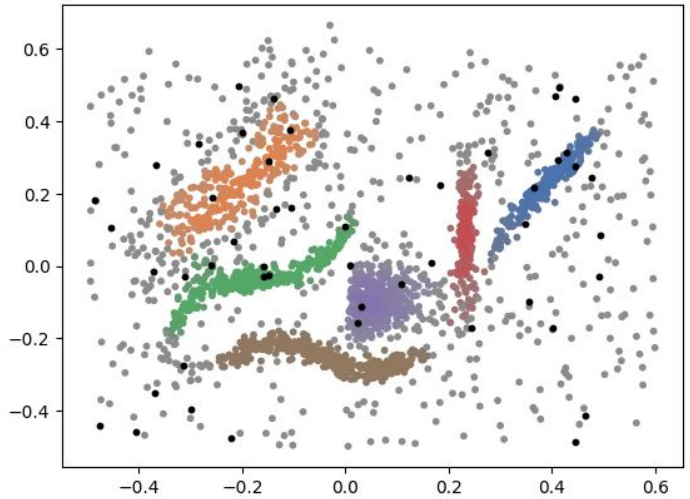
```
<matplotlib.collections.PathCollection at 0x1ccb2a63b50>
```



Implementação em Python

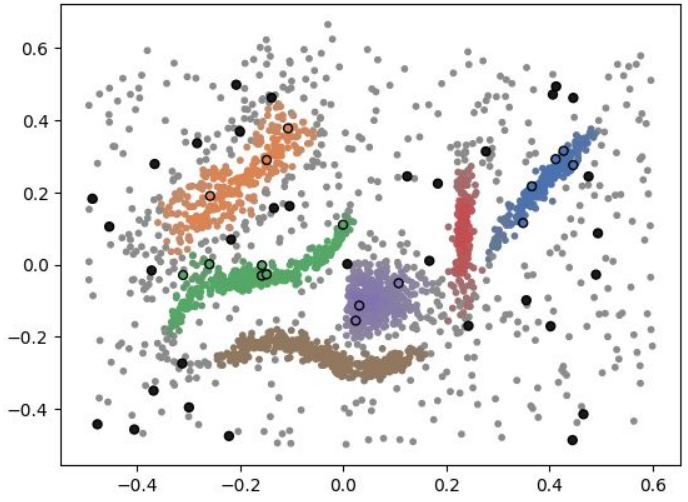
```
test_points = np.random.random(size=(50, 2)) - 0.5

colors = [sns.desaturate(pal[col], sat) for col, sat in zip(clusterer.labels_,
                                                            clusterer.probabilities_)]
plt.scatter(data.T[0], data.T[1], c=colors,s=10,alpha=1);
plt.scatter(*test_points.T, c='black', s=10,alpha=1)
```

```
<matplotlib.collections.PathCollection at 0x1cccf4521d0>
```



```
colors = [sns.desaturate(pal[col], sat) for col, sat in zip(clusterer.labels_,
                                                            clusterer.probabilities_)]
test_colors = [pal[col] if col >= 0 else (0.1, 0.1, 0.1) for col in test_labels]
plt.scatter(data.T[0], data.T[1], c=colors,s=10,alpha=1);
plt.scatter(*test_points.T, c=test_colors, s=25, linewidths=1, edgecolors='k',alpha=1)
```

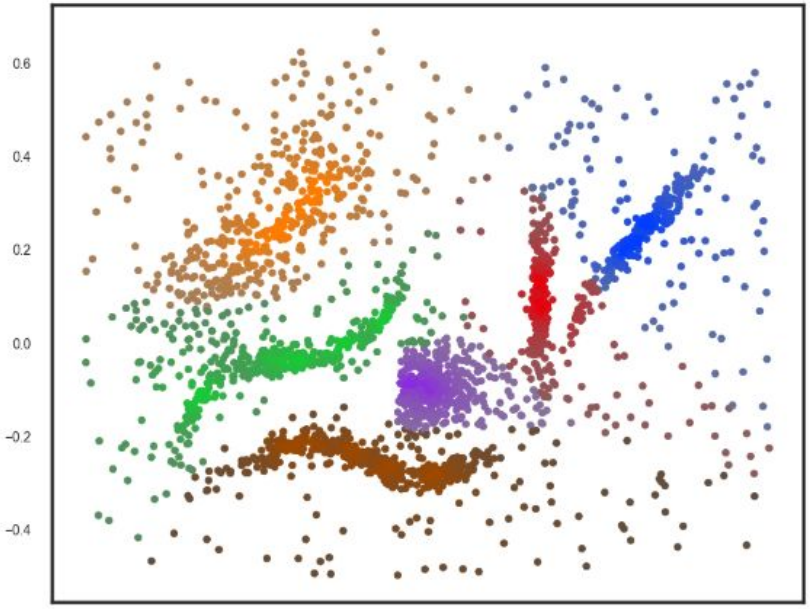
```
<matplotlib.collections.PathCollection at 0x1cccf496800>
```



Implementação em Python

```
exemplar_dict = {c:exemplars(c,tree) for c in tree._select_clusters()}
colors = np.empty((data.shape[0], 3))
for x in range(data.shape[0]):
    membership_vector = dist_membership_vector(x, exemplar_dict, data)
    color = np.argmax(membership_vector)
    saturation = membership_vector[color]
    colors[x] = sns.desaturate(pal[color], saturation)
plt.figure(figsize=(10, 8))
plt.tick_params(labelsize=10)
plt.scatter(data.T[0], data.T[1], c=colors, **plot_kwds)
```

```
C:\Users\brain\AppData\Local\Temp\ipykernel_8116\3033185213.py:6: RuntimeWarning: divide by zero encountered in divide
  result = 1./dist_vector(point, exemplar_dict, data)
<matplotlib.collections.PathCollection at 0x1ccd8a5e0e0>
```



Referências

- https://hdbscan.readthedocs.io/en/latest/index.html
- https://repositorio.usp.br/bitstream/handle/BDPI/51005/2709770.pdf?sequence=1
- https://www.analyticsvidhya.com/blog/2021/06/single-link-hierarchical-clustering-clearly-explained/
- https://github.com/hiperbrainer/HDBSCAN