

Labor

João Torres

2026-01-05

Para que o *k-means* calcule cada distância entre os pontos, cada valor de da tupla de dados da unidade de análise precisa existir.

Excluir os valores ausentes aqui parece ser a melhor opção, uma vez que a fase de pré-processamento e transformação já agiu sobre os valores ausentes. E caso mais transformações aos dados tenham que ser feitas, isso é algo para a etapa anterior.

| | feature | Valores ausentes | % |
|-----------------------|-----------------------|------------------|----------|
| labor_share_final | labor_share_final | 6498 | 61.05995 |
| productivity_final | productivity_final | 4694 | 44.10825 |
| gini_final | gini_final | 7817 | 73.45424 |
| fdi_net_gdp_final | fdi_net_gdp_final | 2450 | 23.02199 |
| avg_hourly_wage_final | avg_hourly_wage_final | 9698 | 91.12949 |

10067 linhas eliminadas

Esta escolha fez-nos perder 10 mil linhas.

O *k-means* usa o número de clusters como parâmetros e a distância entre pontos como métrica.

O *Ward's method* pode usar o número de clusters ou limites de distância e usa a mesma métrica que o *k-means*.

Vamos ler o que o `scikit-learn` tem para nos ensinar acerca do *k-means*:

O algoritmo *k-means* agrupa dados tentando separar amostras em n grupos de variância igual.

O algoritmo *k-means* divide um conjunto de N amostras X em K clusters disjuntos C , cada um descrito pela média das amostras no *cluster*. As médias são comumente chamadas de «centróides» do *cluster*;

O *k-means* é frequentemente referido como algoritmo de Lloyd. Em termos básicos, o algoritmo tem três etapas. A primeira etapa escolhe os centróides iniciais, sendo o método mais básico escolher amostras do conjunto de dados. Após a inicialização, o *k-means* consiste em um *loop* entre as duas outras etapas. A primeira etapa atribui cada amostra ao seu centroide mais próximo. A segunda etapa cria novos centroides, tomando o valor médio de todas as amostras atribuídas a cada centroide anterior. A diferença entre os centroides antigos e novos é calculada e o algoritmo repete estas duas últimas etapas até que este valor seja inferior a um limite. Em outras palavras, ele repete até que os centroides não se movam significativamente.

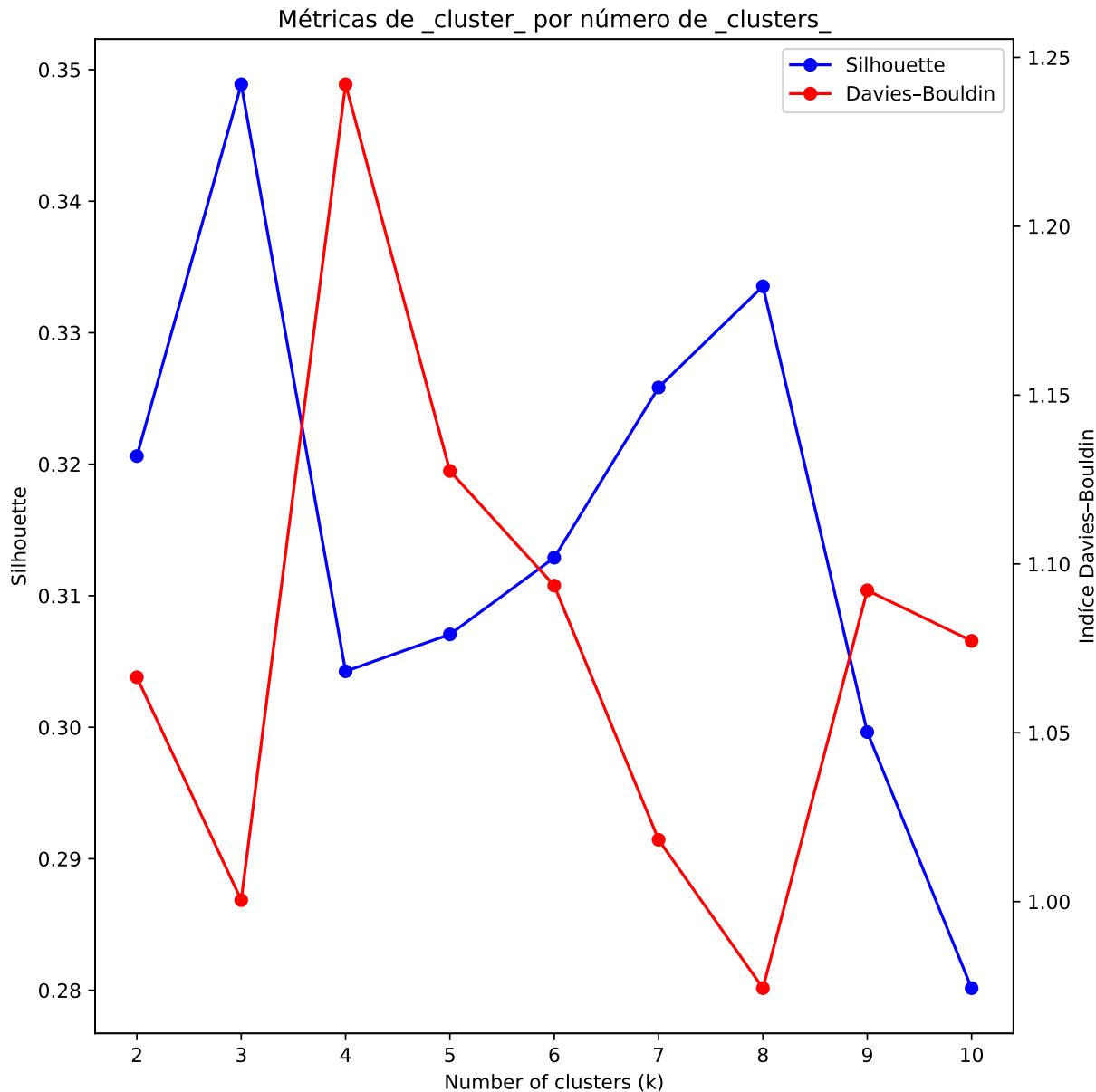
Também é possível encontrar exemplos de como usar a biblioteca para esta tarefa específica.

Todas as variáveis foram normalizadas antes de criar os *clusters* usando o `StandardScaler.fit_transform`. Isso porque queremos que os centróides estejam próximos da média de um *cluster*. Também podemos ter algumas *features* da unidade de análise com escalas muito diferentes umas das outras.

Vamos começar a usar a biblioteca `scikit-learn` para termos acesso ao algoritmo k -means sem o ter que implementar.

Mas antes, temos que decidir qual é o número de *clusters* que nós queremos ter no resultado da análise. Existem uns números que são melhores que outros, e existem algoritmos, como o *silhouette* e o de índice de *Davies–Bouldin* que nos dizem quais são.

Dentro das respostas que o *silhouette* nos dá, devemos escolher o maior valor, ou um dos máximos locais, e evitar números muito grandes sem uma justificação teórica. Para o índice de *Davies–Bouldin*, menor o valor, melhor.



Reparamos que a melhor escolha para número de clusters com certeza é o 3, apesar de que escolher 7 não parece uma opção tão má assim, aliás, na *data warehouse* os países estão divididos em 7 regiões.

Aplicando o algoritmo, conseguimos ver quantos países (juntamente com o ano) é que ficaram em cada um dos clusters. O objetivo é não ficar com clusters pequenos, ou com um enorme que engloba quase tudo.

| | x |
|---|-----|
| 0 | 133 |
| 1 | 409 |
| 2 | 33 |

Conseguimos ver o valor de cada *feature* para cada *centroid*.

| labor_share_final | productivity_final | gini_final | fdi_net_gdp_final | avg_hourly_wage_final | cluster |
|-------------------|--------------------|------------|-------------------|-----------------------|---------|
| 60.40912 | 111607.57 | 35.01466 | -0.0058223 | 23.99842 | 0 |
| 47.58469 | 35789.14 | 42.33052 | 0.0283574 | 407.32417 | 1 |
| 49.35324 | 23538.45 | 38.65000 | 0.0253379 | 7720.74122 | 2 |

Também é possível analisar com que grupo de rendimento e região do globo é que cada centroid se identifica mais, ou de outra forma, como é que um país, dependendo da sua região geográfica e nível de rendimento, fica agrupado com outros.

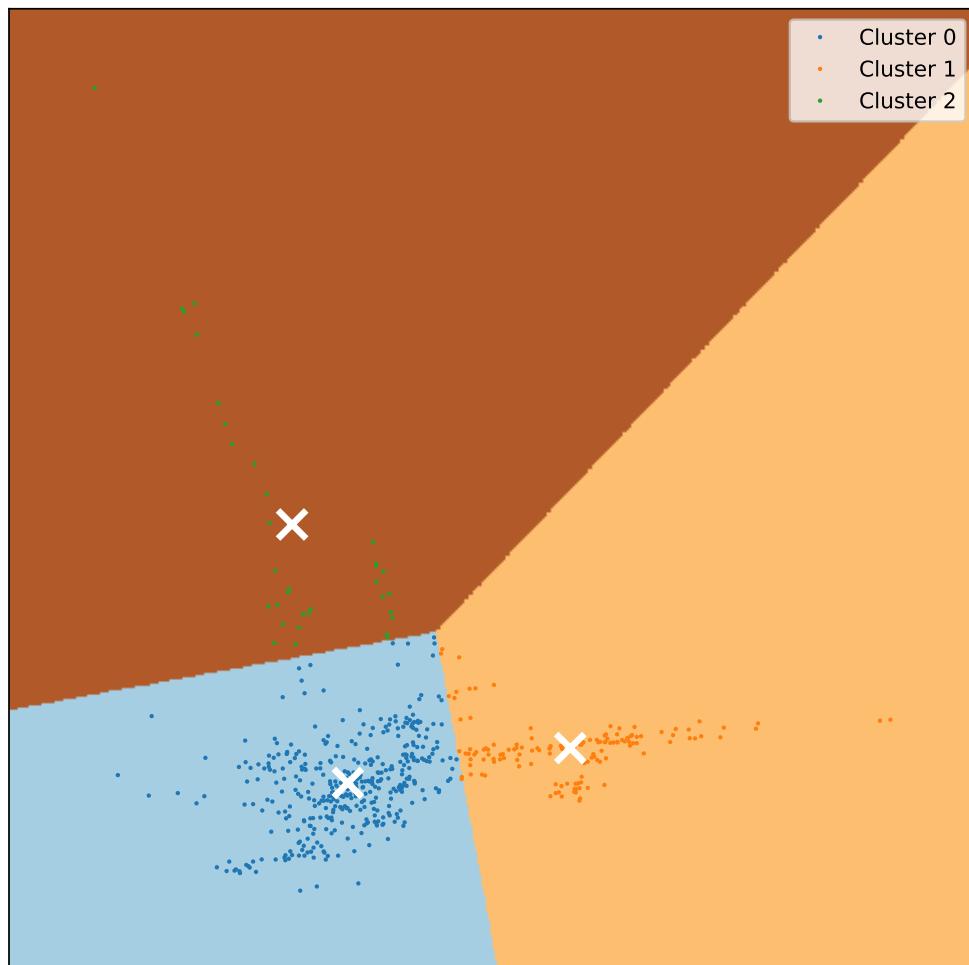
| | High income | Low income | Lower middle income | Upper middle income |
|---|-------------|------------|---------------------|---------------------|
| 0 | 0.9624060 | 0.0000000 | 0.0075188 | 0.0300752 |
| 1 | 0.1446078 | 0.0490196 | 0.2647059 | 0.5416667 |
| 2 | 0.0000000 | 0.0303030 | 0.4242424 | 0.5454545 |

| | East Asia & Pacific | Europe & Central Asia | Latin America & Caribbean | Middle East & North Africa | North America | South Asia | Sub-Saharan Africa |
|---|---------------------|-----------------------|---------------------------|----------------------------|---------------|------------|--------------------|
| 0 | 0.0000000 | 0.6766917 | 0.0300752 | 0.0000000 | 0.2857143 | 0.0000000 | 0.0075188 |
| 1 | 0.1246944 | 0.1295844 | 0.5354523 | 0.0684597 | 0.0000000 | 0.0342298 | 0.1075795 |
| 2 | 0.6969697 | 0.0000000 | 0.2727273 | 0.0000000 | 0.0000000 | 0.0000000 | 0.0303030 |

Vamos criar uma visualização, usando como base o exemplo “Visualize the results on PCA-reduced data”.

```
## KMeans(n_clusters=np.int64(3), n_init=20)
## (-4.563880357651497, 6.710042047616408)
## (-2.673281992085398, 9.48497381266387)
## ([], [])
## ([], [])
```

K-means clustering on country-year clusters (PCA-reduced data)
Centroids are marked with white cross



Um modelo de agrupamento k-means foi aplicado às observações por país e ano utilizando indicadores macroeconómicos e laborais padronizados. O número de agrupamentos foi selecionado através de uma análise de *silhouette*, e os agrupamentos resultantes foram interpretados comparando a composição dos agrupamentos com as classificações regionais e de rendimento existentes.