04

A validação relativa

Transcrição

Temos três valores para para interpretar com diferentes métricas, mas como faremos isso?

Na métrica de silhouette sabemos que o valor tem um intervalo que vai de -1 até 1, portanto se o cluster obter um valor positivo isso é um bom sinal.

No caso da métrica de Davies Bouldin, quanto mais próximo de 0 o valor, melhor. Já para Calinski, é melhor que o valor seja o mais alto possível.

É difícil fazer qualquer avaliação se considerarmos apenas o valor cru, e nesse ponto que apresentamos a **avaliação relativa**.

Compararmos o valor das métricas com base em configurações diferentes do nosso algorítimo. A ideia é que assim ajustemos os parâmetros do algorítimo para obter os melhores resultados.

Mudaremos o número de clusters, afinal esse é o parâmetro que possui mais impacto. Criaremos um método que será reponsável pela execução do `kmeans` e calcular as três métricas de uma vez.

O nome desse método será `clustering_algorithm()` que receberá como parâmetro o número de clusters que queremos e nosso dataset. Passaremos, ainda, o número máximo de execuções e de interações. Feito isso, estamos prontos para calcular cada uma das métricas.

```
def clustering_algorithm(n_clusters, dataset):
    kmeans = Kmeans(n_clusters=n_clusters, n_init=10, max_iter=300)
    labels = kmeans.fit_predict(dataset)
    s = metrics.silhouette_score(dataset, labels, metric='euclidean')
    dbs = metrics.davies_bouldin_score(dataset, labels)
    calinski = metrics.calinski_harabasz_score(dataset, labels)
    redurn s, dbs, calinski
```

Feito isso, chamaremos a função com diferentes números de cluster. Começaremos com 3 clusters.

```
s1, dbs1, calinski1 = clustering_algorithm(3, values)
print(s1, dbs1, calinski1)
```

Como resultado teremos os valores 0,32; 1.3; 3526 Faremos o mesmo procedimento, dessa vez com 5 clusters.

```
s2, dbs2, calinski2 = clustering_algorithm(5, values)
print(s2, dbs2, calinski2)
```

Os novos valores obtidos serão 0.36; 1,0; 3421.

Observemos o valor de silhouette para 3 e 5 clusters: 0,32 e 0,36. A ideia da métricia de silhouette é aumentar, então obtivemos um melhor resultado com 5 clusters. Mas não temos uma diferença vertiginosa em nenhuma das métricas.

Faremos agora o mesmo processo para 10 clusters.

```
s3, dbs3, calinski3 = clustering_algorithm(10, values)
print(s3, dbs3, calinski3)
```

Os valores obtidos serão de 0.3; 1.1 ; 3018. Para a métrica de bouldin encontramos um meio termo entre o melhor e pior resultado até agora. Se executarmos o mesmo algorítimo com 20 e 50 clusters, veremos que o caso da métrica de calinski a qualidade decai muito, já bauldin parece manter alguma consistência.

Selecionaremos uma desses configurações para seguir adiante e realizar outras validações.

Utilizaremos a configuração de 5 clusters, pois obtemos um resultado bom no índice de silhouette, que é bastante popular, então é interessante utilizá-lo como medidor principal.