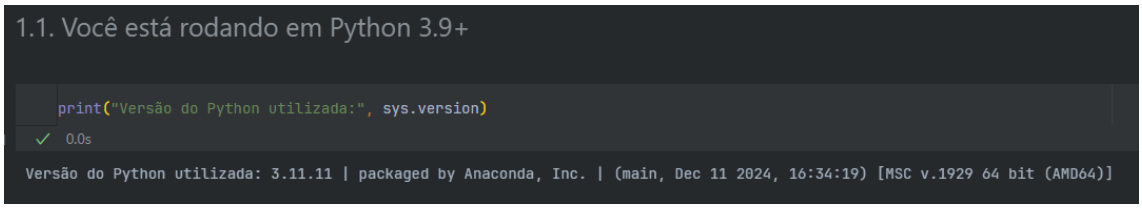
Instituto Infnet	**Avaliação**	Nota: Visto do Professor:

MIT em Inteligência Artificial, Machine Learning e Deep Learning	
Nome	Mateus Teixeira Ramos da Silva
Link do repositório	https://github.com/GitMateusTeixeira/ml_clustering/tree/main/infnet_validation_pd
Matéria	Validação de modelos de clusterização
Prazo	16.12.2024

Parte 1. Infraestrutura

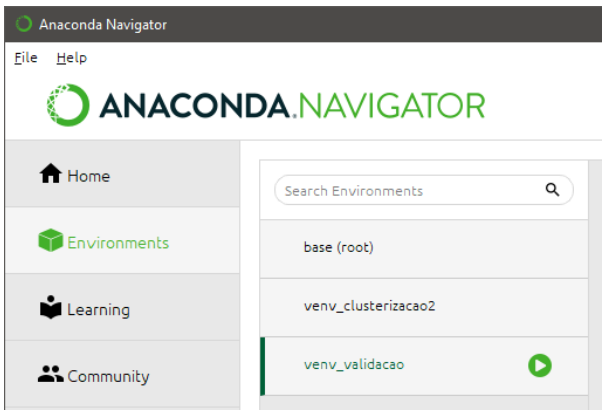
1. Você está rodando em Python 3.9+

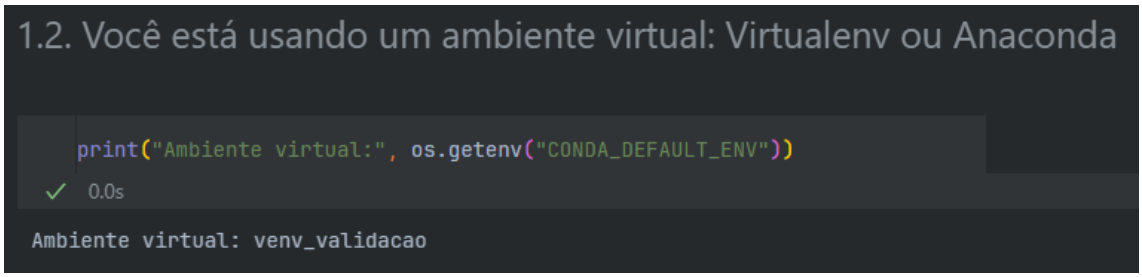
R: Requerimento atendido no ponto “1.1. Você está rodando em Python 3.9+” do arquivo.ipynb.



2. Você está usando um ambiente virtual: Virtualenv ou Anaconda

R: Requerimento atendido no ponto “1.2. Você está usando um ambiente virtual: Virtualenv ou Anaconda” do arquivo.ipynb. O nome do ambiente é ‘venv_validacao’.





3. Todas as bibliotecas usadas nesse exercício estão instaladas em um ambiente virtual específico

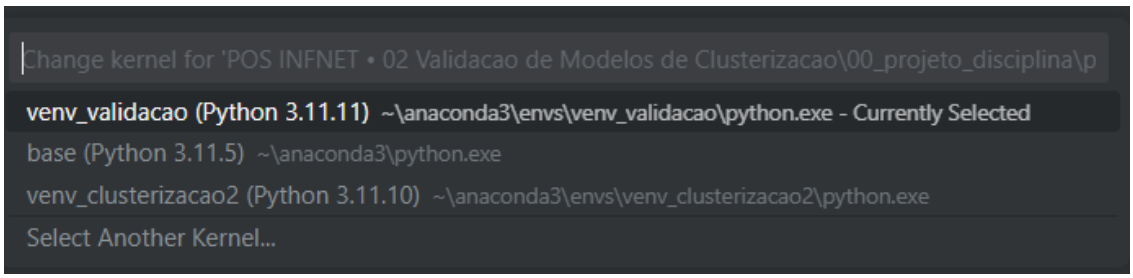
R: Requisito atendido no arquivo 'requirements.txt', presente no repositório (https://github.com/GitMateusTeixeira/ml_clustering/tree/main/infnet_validation_pd).

4. Gere um arquivo de requerimentos (requirements.txt) com os pacotes necessários. É necessário se certificar que a versão do pacote está disponibilizada.

R: Requisito atendido no arquivo 'requirements.txt', presente no repositório (https://github.com/GitMateusTeixeira/ml_clustering/tree/main/infnet_validation_pd).

5. Tire um printscreen do ambiente que será usado rodando em sua máquina.

R: Arquivo no repositório (https://github.com/GitMateusTeixeira/ml_clustering/tree/main/infnet_validation_pd)



6. Disponibilize os códigos gerados, assim como os artefatos acessórios (requirements.txt) e instruções em um repositório GIT público. (se isso não for feito, o diretório com esses arquivos deverá ser enviado compactado no moodle).

R: Link do repositório no GitHub:
https://github.com/GitMateusTeixeira/ml_clustering/tree/main/infnet_validation_pd

Parte 2. Escolha de base de dados

1. Escolha uma base de dados para realizar o trabalho. Essa base será usada em um problema de clusterização.

R: Foi escolhida uma base de dados sobre transações bancárias realizadas em um banco Indiano, com as seguintes informações:

- *id_transacao (TransactionID)*: Id da transferência feita

- *id_cliente (CustomerID)*: Id do cliente

- *idade_calculada (CustomerDOB)*: Idade do cliente no período da transferência

- *genero (CustGender)*: Gênero sexual do cliente – será utilizado para visualização

- *localizazao (CustLocation)*: Localização do cliente – será utilizado para visualização

- *saldo (CustAccountBalance)*: Saldo da conta do cliente após a transferência em rúpias indianas (INR)

- *data_transacao (TransactionDate)*: Data da transferência – foi utilizado unicamente para calcular a idade dos clientes

- *hora_transacao (TransactionTime)*: Hora da transferência em timestamp Unix (o número de segundos que se passaram desde a data da coluna anterior) – não será utilizado

- *quantia_transacao (INR) (TransactionAmount (INR))*: Valor da transferência em rúpias indianas (INR)

2. Escreva a justificativa para a escolha de dados, dando sua motivação e objetivos.

R: Foi escolhida uma base de dados que simulam mais de 50 mil transações de clientes feitas em um banco indiano. Essa base foi escolhida pois simula bem um problema da vida real, com um dataset cheio de dados nulos, que precisam de um tratamento prévio, dando um aspecto mais orgânico para a clusterização.

O objetivo é traçar o perfil preponderante dos clientes, através da clusterização das transações realizadas.

3. Mostre através de gráficos a faixa dinâmica das variáveis que serão usadas nas tarefas de clusterização. Analise os resultados mostrados. O que deve ser feito com os dados antes da etapa de clusterização?

R: Foi demonstrada por boxplot e por histplot no ponto *'2.3'*.

A análise gráfica realizada no código demonstra a presença de alguns dados cujo valor se encontra bem distante da maioria (como na coluna 'saldo' e 'quantia_transacao'), o que pode (ou não) representar dados outliers e a necessidade de tratamento dos mesmos.

Embora estejam tratados na pipeline 'tratamento.py', deve-se adaptar as colunas categóricas e padronizar as colunas numéricas para que os dados fiquem compatíveis com os modelos que irão ser utilizados.

A rigor, em razão do dataset apresentar dados de transferências bancárias e haver saldos com valores bem dispares (de 0 a 17 milhões de rúpias indianas), não são valores nulos. Deve-se, portanto, padronizar os dados, de modo que amenize a diferença entre eles, sem, contudo, substituí-los de nenhuma forma (pela média, mediana, ...).

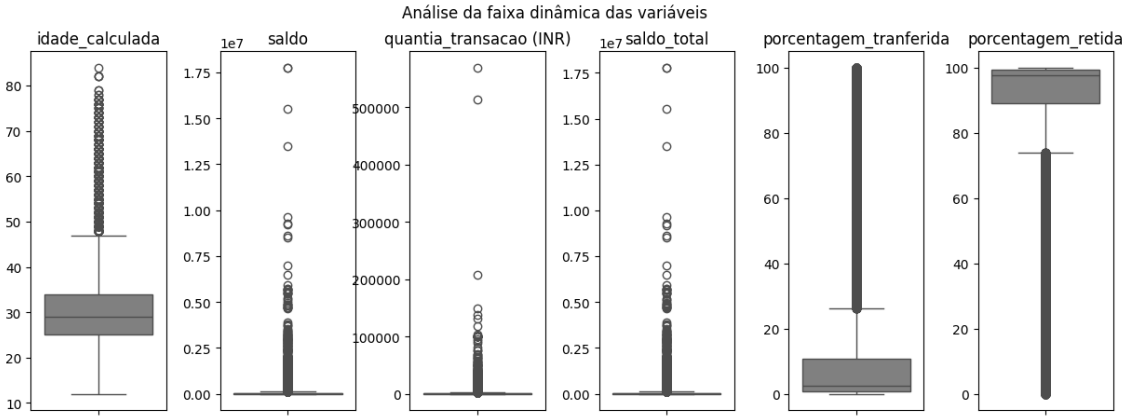
	coluna	tipo	únicos	null_soma	media	desvio	minimo	25%	mediana	75%	maximo
0	saldo	float64	13997	0	104331.201033	457364.148862	0.0	4698.99	16609.17	56051.97	17772978.13

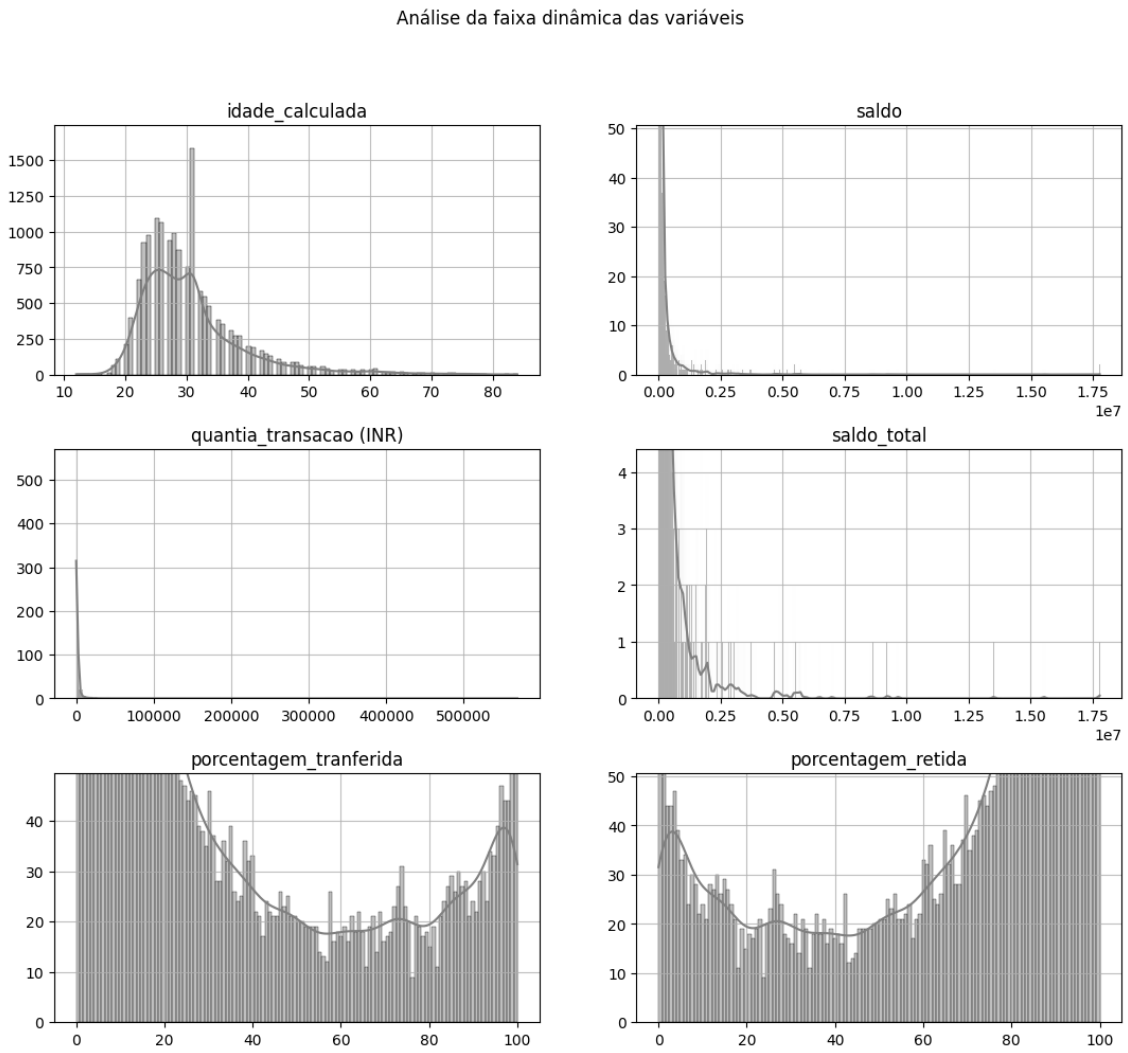
Antes da etapa de clusterização deve-se ainda realizar uma análise exploratória dos dados, a fim de verificar se o dataset possui algum dado nulo, duplicado, incorreto ou ainda incompatível com os modelos de clusterização (vale dizer que esses modelos trabalham com dados numéricos). Tudo no intuito de deixar os dados compatíveis com os modelos que irão ser utilizados.

Para esse dataset em específico, observou-se que a coluna 'saldo' trata-se do saldo que o cliente possuía antes da transferência, assim, se somarmos 'saldo' com 'quantia_transacao' obteremos o ‘saldo total’. A partir daí, podemos retirar a porcentagem que os clientes transferiram de suas contas e a porcentagem que ficou retida, possibilitando uma análise de quem movimenta mais a conta (transferindo quase seu saldo inteiro e não deixando nada na conta) e quem retem mais o dinheiro (transferindo pouco do saldo total) – que seria o público-alvo para o oferecimento de produtos de investimento, por exemplo.

A conversão em porcentagem permite ainda uma padronização de escalas, pois a coluna 'idade' já se encontra padronizada entre 10 e 85 (anos) enquanto que 'saldo' e 'quantia_transacao' ficaram também entre 0 e 100, possibilitando uma clusterização com qualidade.

A distribuição da faixa dinâmica já denota que a maioria dos clientes possuem entre 20 e 40 anos e retem cerca de 80% a 100% do seu saldo em conta, realizando transferências baixas, a rigor.





4. Realize o pré-processamento adequado dos dados. Descreva os passos necessários.

R: Analisando os principais métodos de escalonamento e normalização de dados, poderíamos eleger o RobustScaler, pois ele lida bem com outliers, em sua maioria.

Todavia, ao analisar melhor o dataset e perceber que a coluna 'saldo' trata-se da quantia restante após a transferência do valor de 'quantia_transacao' pode-se calcular o saldo total anterior e, assim, determinar, em % o quanto os clientes estão transferindo e retendo em suas contas, que nos auxiliará melhor na análise de perfil de cliente.

Essa abordagem – já realizada no ponto '*2.3.1. Preparar os dados*' – ainda se iguala na escala da coluna 'idade', deixando padronizadas todas as colunas que serão utilizadas para a clusterização.

Veja-se:

- 'idade': varia de 12 a 84 anos

- 'porcentagem_transferida': varia de 0 a 100 %

- 'porcentagem_retida': varia de 0 a 100 %

	coluna	tipo	únicos	null_soma	media	desvio	minimo	25%	mediana	75%	maximo
0	idade_calculada	int64	70	0	30.708309	8.498054	12.0	25.000000	29.000000	34.000000	84.0
1	porcentagem_tranferida	float64	15552	0	13.318031	24.401673	0.0	0.670390	2.512903	10.833552	100.0
2	porcentagem_retida	float64	15554	0	86.675612	24.411228	0.0	89.163964	97.487097	99.329428	100.0

Porém, percebeu-se também que a coluna 'porcentagem_transferida' de saldo é o espelho de 'porcentagem_retida', visto que se uma pessoa transfere 40% de seu saldo inicial, ela retem 60%, por isso, essa coluna não será usada na clusterização. Utilizaremos ‘porcenatgem_retida’ e ‘idade’.

Parte 3. Clusterização

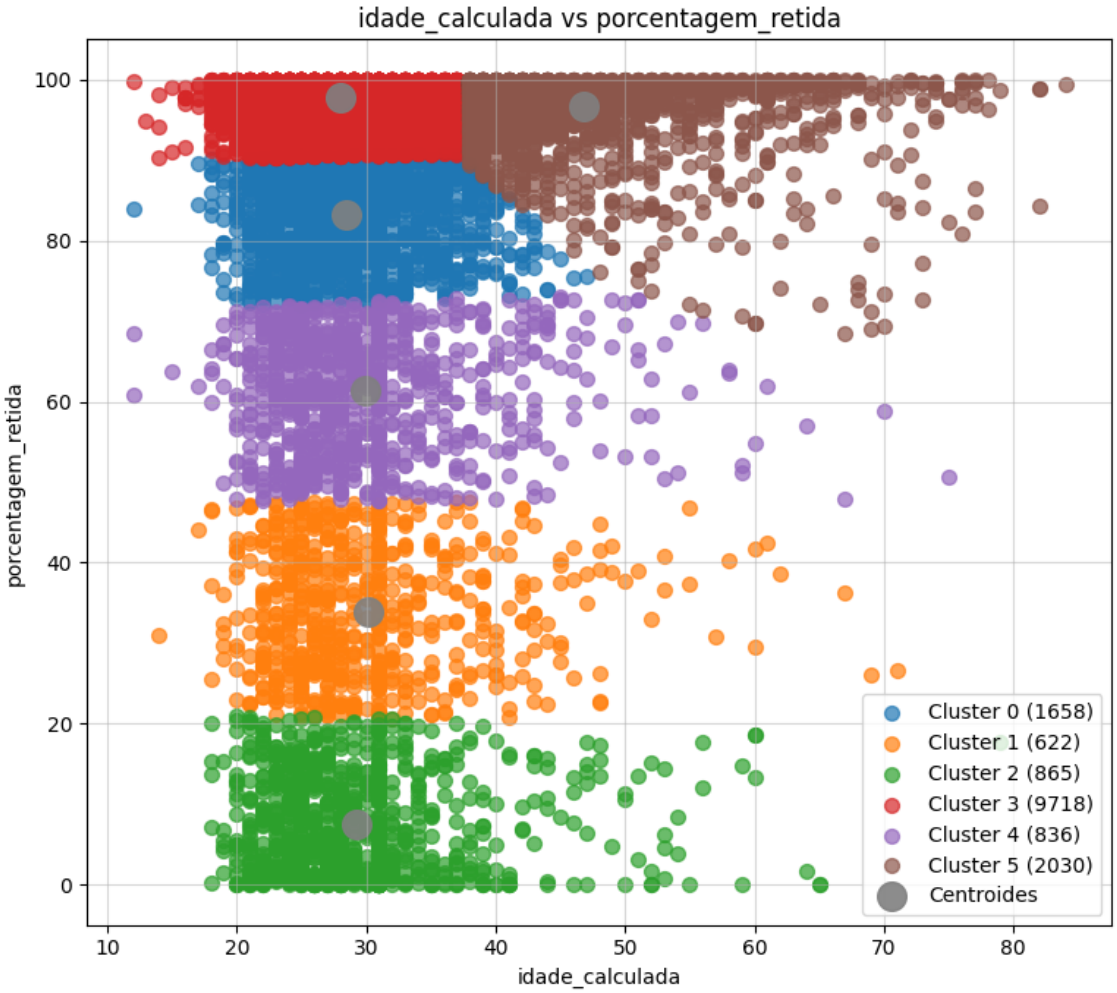
1. Realizar o agrupamento dos dados, escolhendo o número ótimo de clusters. Para tal, use o índice de silhueta e as técnicas:

a. K-Médias

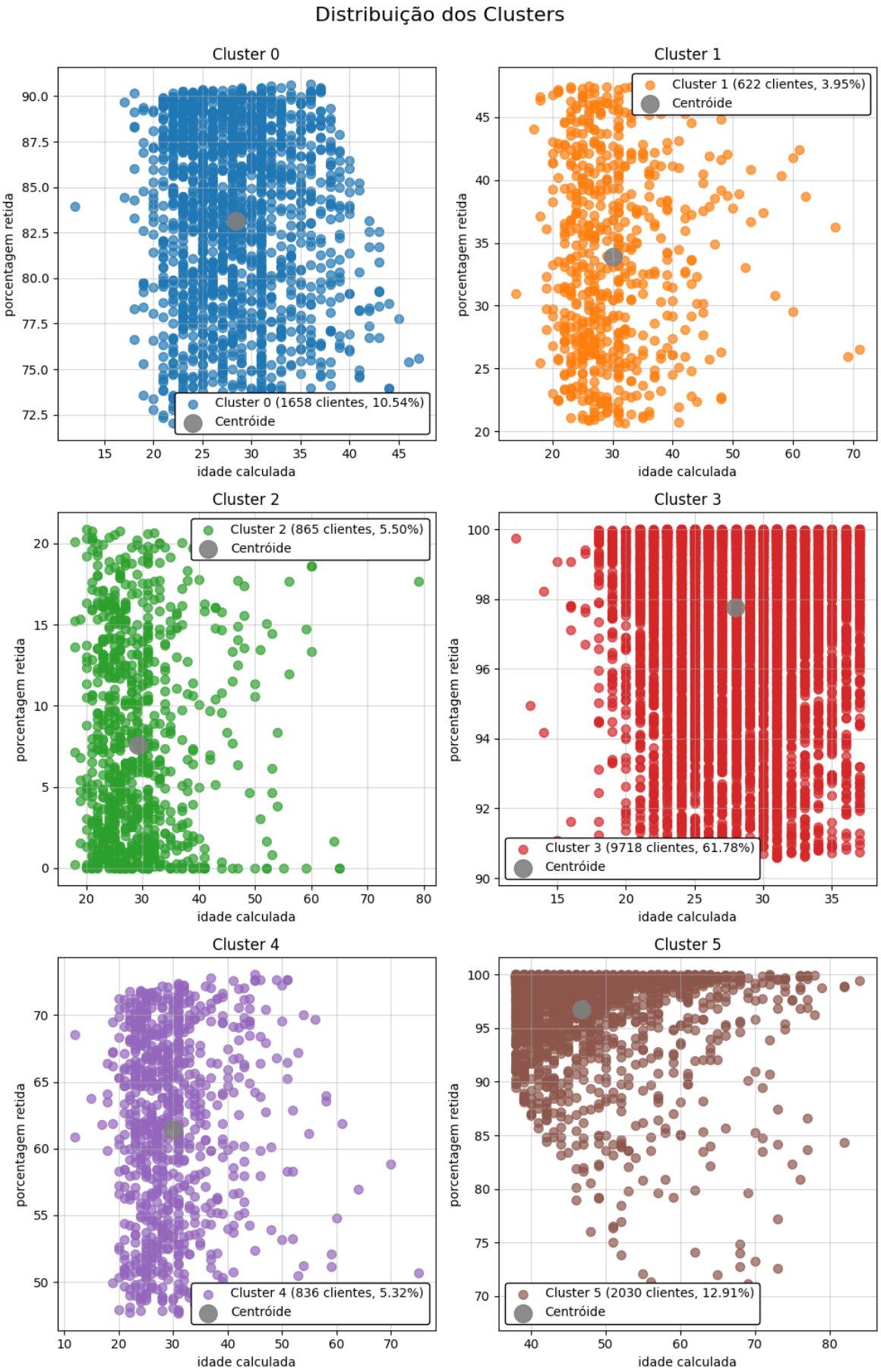
R: Foi implementado um Grid Search para determinar o número ótimo de ‘*k*’, no ponto ‘*3.2.1. Gráficos do índice da silhueta para o K-Means’*. Para isso, fora realizado testes no número de ‘*k*’ (de 2 a 15).

Embora o melhor score da silhueta tenha resultado em ‘*k=2’*, a seguir, veremos que não é a melhor solução para esse dataset. Isso revela que esses índices são usados para termos um ‘norte’ de como seguir, mas não determina o número específico de clusters.

O modelo foi implementado (no ponto '*3.1.a.1. Implementando o modelo do K-Means com base nos resultados da silhueta (3.2.)*) utilizando '*k*=6', considerando que o 'Cluster 3' (em vermelho) possui dados altamente concentrados, sendo o maior grupo com (9.718 – cerca de 62% da base de dados), porém em um local bem concentrado (em '*3.1.a.2. Análise gráfica do K-Means*'), não indicando a necessidade de mais grupos:



Plotando gráficos isolados de cada cluster – vide '*3.1.a.2. Análise gráfica do K-Means*', é possível ver esse fenômeno melhor (abaixo):



No final, imprimiu-se as características gerais de cada cluster:

Tabela de características dos Clusters:

	Cluster	Clientes	Genero	Faixa Etária	Top 3 Localizações	Porcentagem de Retenção
0	Cluster 0	1658	M (72%)	22 a 31	MUMBAI (149), BANGALORE (129), DELHI (111)	72% a 90%
1	Cluster 1	622	M (78%)	22 a 33	NEW DELHI (64), MUMBAI (56), DELHI (46)	20% a 47%
2	Cluster 2	865	M (78%)	21 a 31	MUMBAI (77), BANGALORE (62), NEW DELHI (57)	0% a 20%
3	Cluster 3	9718	M (72%)	22 a 31	MUMBAI (977), NEW DELHI (825), BANGALORE (761)	90% a 100%
4	Cluster 4	836	M (72%)	22 a 31	MUMBAI (64), BANGALORE (61), DELHI (53)	47% a 73%
5	Cluster 5	2030	M (75%)	38 a 48	MUMBAI (248), BANGALORE (170), NEW DELHI (166)	68% a 100%

b. DBScan

R: Para o DBSCAN, foi implementado um Grid Search para determinar o número ótimo de '*eps*' e '*min_samples*'.

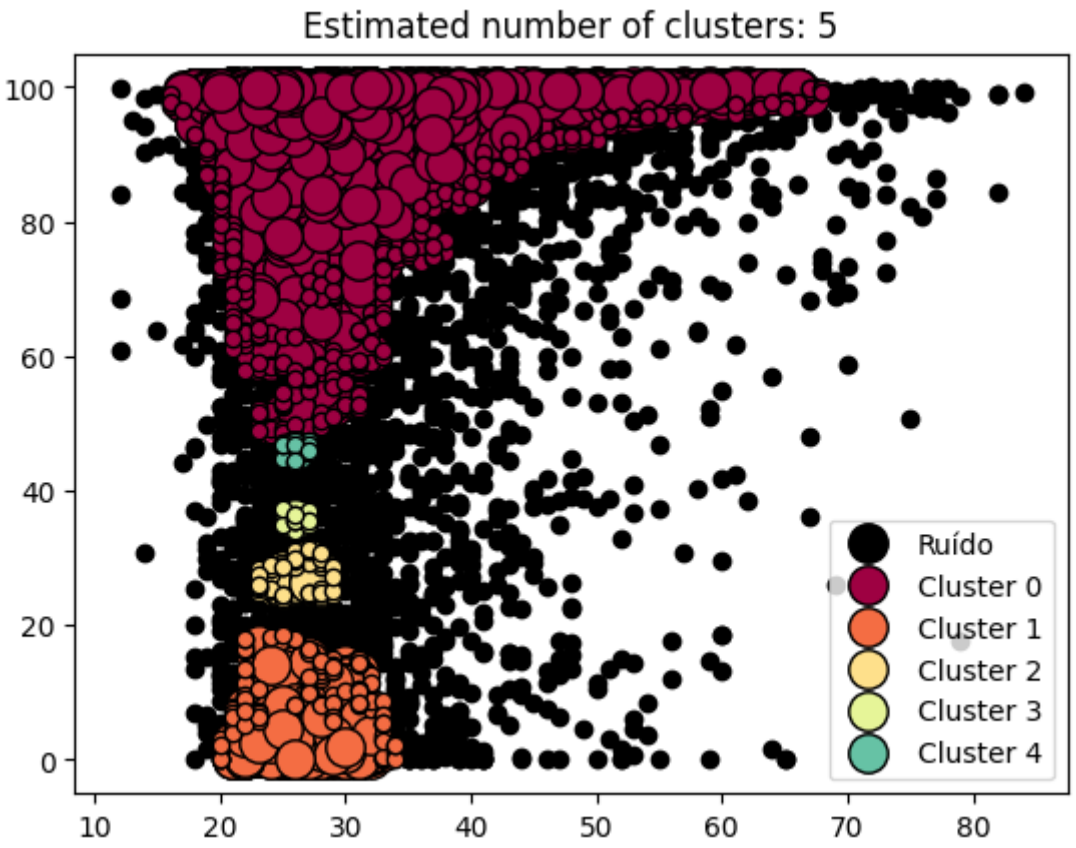
O modelo proposto foi o mais compatível com o K-Means, possibilitando a melhor comparação possível entre os modelos, com os parâmetros:

metric = 'euclidean' (método de cálculo das distâncias dos dados)

p=2 (potência para as métricas da fórmula de distância Minkowski – '2' para Euclidiana)

Como será tratado mais a frente, foram realizadas diversas pesquisas para determinar os melhores parâmetros, considerando que fora realizado testes no número de '*eps*' (de 0,8 a 1,8) e o '*min_samples*' (de 50 a 150).

O resultado da pesquisa será analisado na pergunta a seguir, mas pode-se perceber que os grupos formados são compatíveis com os grupos do K-Means, a exemplo, o 'Cluster 0' do DBSCAN abrangeu todos os clusters mais densos do K-Means e, por isso, propícios à criação de grupos com o DBSCAN.



No final, também foi impresso o perfil preponderante de cada cluster:

Tabela de características dos Clusters DBSCAN:

	Cluster	Clientes	Genero	Faixa Etária	Top 3 Localizações	Porcentagem de Retenção
0	Cluster 0	13598	M (73%)	22 a 31	MUMBAI (1377), NEW DELHI (1103), BANGALORE (1066)	48% a 100%
1	Cluster 1	620	M (78%)	22 a 31	MUMBAI (55), BANGALORE (46), DELHI (40)	0% a 18%
2	Cluster 2	89	M (80%)	23 a 29	NEW DELHI (8), DELHI (7), MUMBAI (6)	24% a 31%
3	Cluster 3	20	M (85%)	25 a 27	NEW DELHI (4), BANGALORE (3), NAVI MUMBAI (1)	34% a 37%
4	Cluster 4	20	M (75%)	25 a 27	NEW DELHI (5), DELHI (2), MUMBAI (2)	44% a 47%

2. Com os resultados em mão, descreva o processo de mensuração do índice de silhueta. Mostre o gráfico e justifique o número de clusters escolhidos.

R: O índice de silhueta é uma métrica utilizada para avaliar a qualidade de um agrupamento. Ele mede o quão bem um determinado ponto está atribuído ao seu cluster em comparação com outros clusters, refletindo tanto a coesão dentro do cluster (medida interna) quanto a separação entre clusters (medida externa).

A coesão mede a distância média de um ponto para todos os pontos no mesmo cluster.

A separação mede a distância média de um ponto para todos os pontos do cluster mais próximo (que não seja o cluster ao qual ele pertence).

O índice da silhueta considera a fórmula: ‘separação – coesão’

Dessa forma:

- Se a separação for maior que a coesão, o resultado do índice da silhueta será positivo, indicando uma boa distribuição dos clusters;

- Se a separação for parecida com a coesão, o resultado do índice da silhueta será próximo a zero, indicando que os dados estão no limítrofe dos clusters;

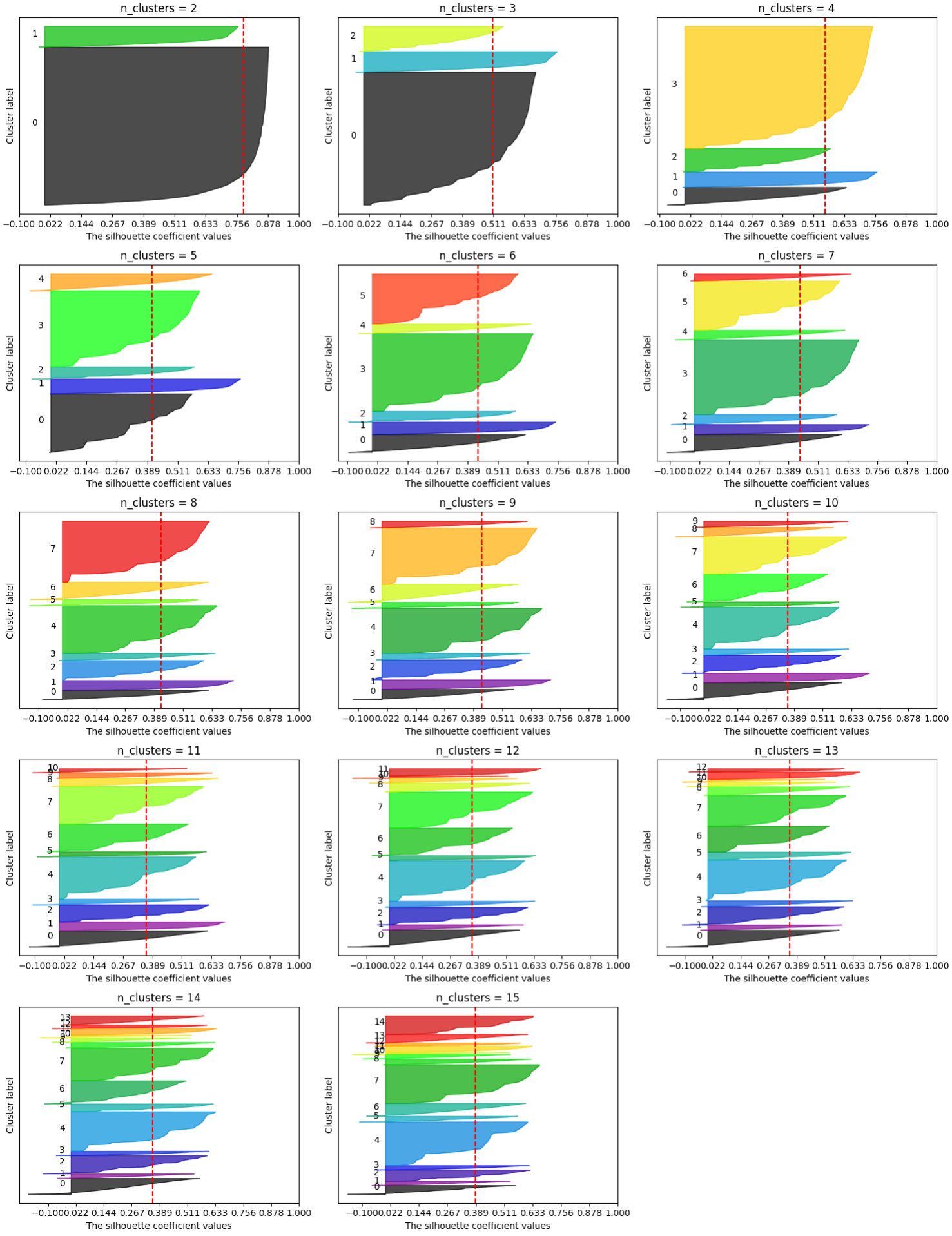
- Se a separação for menor que a coesão, o resultado do índice da silhueta será negativo, indicando uma má distribuição dos clusters.

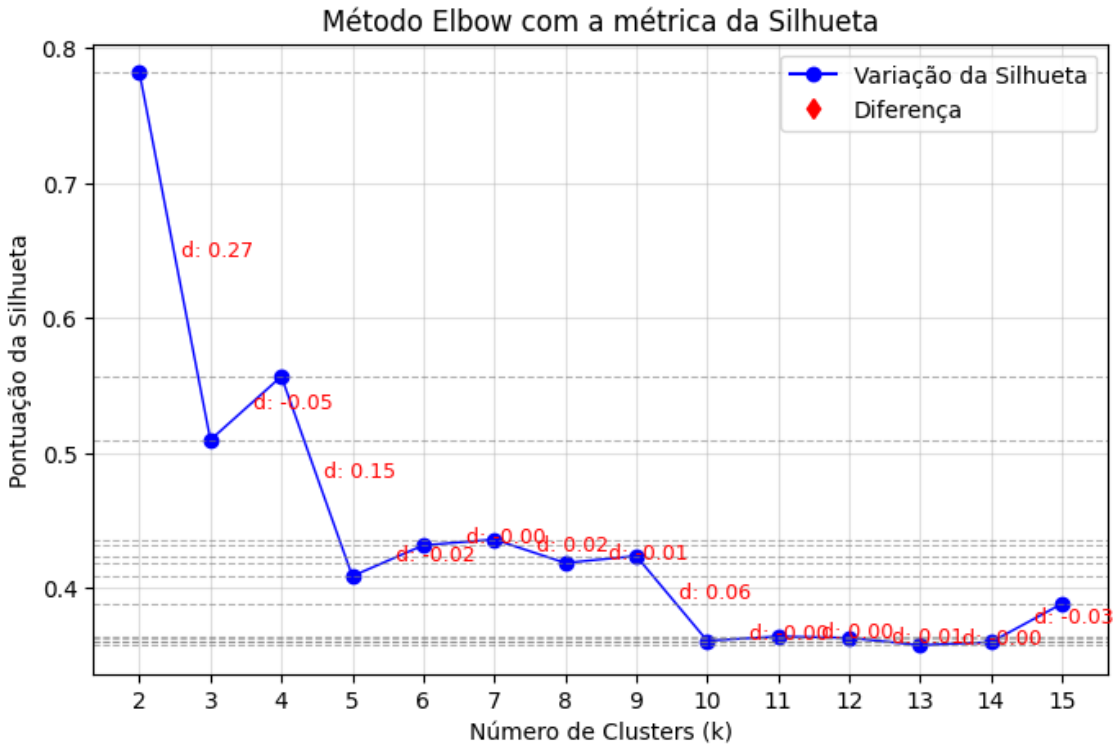
Por fim, o índice de silhueta médio, calcula a média dos valores de cada cluster, seguindo a mesma ideia de avaliação (<0, =0 e >0).

Os gráficos estão demonstrados nos pontos:

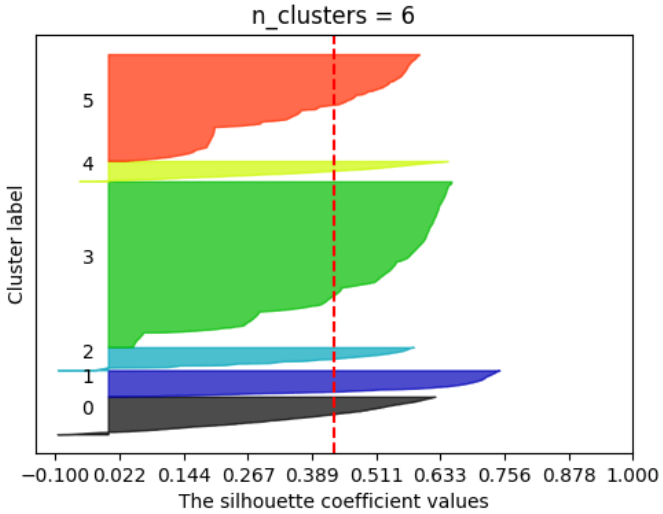
‘*3.2.1. Gráficos do índice da silhueta para o K-Means*’:

Silhouette Analysis for KMeans with 'lloyd' Algorithm





Para o K-Means, foi escolhido k=6, pois é um número onde o índice de silhueta apresenta poucos negativos e um bom resultado (0,43):



Para o DBSCAN foram realizadas diversas buscas de silhueta para encontrar o melhor intervalo possível.

Num primeiro momento foi feito o seguinte intervalo: '*eps*' de 0,8 a 1,8 e '*samples*' de 10 a 100 (de 10 em 10), onde os dez melhores resultados foram esses:

```
eps_range = np.arange(0.8, 1.8, 0.1)
samples_range = np.arange(10, 100, 10)

avaliar_dbscan(dados_padronizados, eps_range, samples_range)
```

✓ 5m 51.6s

```
As 10 melhores pontuações foram:
eps         samples     score       clusters
1.60        30          0.595409    4
1.50        40          0.586906    3
1.20        30          0.575628    3
1.40        40          0.573572    3
1.70        20          0.569989    5
1.60        20          0.559152    6
1.60        60          0.514704    2
1.50        60          0.506376    2
1.40        60          0.498059    2
1.40        10          0.488330    16
```

A partir disso, fora realizada uma nova busca, dessa vez com '*eps*' de 1,2 a 1,8 e '*samples*' de 20 a 70 (de 5 em 5), onde os dez melhores resultados foram esses:

```
eps_range = np.arange(1.2, 1.8, 0.1)
samples_range = np.arange(20, 70, 5)

avaliar_dbscan(dados_padronizados, eps_range, samples_range)
```

✓ 5m 18.8s

```
As 10 melhores pontuações foram:
eps         samples     score       clusters
1.80        35          0.633910    2
1.60        30          0.595409    4
1.60        30          0.595409    4
1.80        20          0.595180    5
1.70        45          0.588857    4
1.50        40          0.586906    3
1.50        40          0.586906    3
1.60        45          0.584076    3
1.20        30          0.575628    3
1.40        40          0.573572    3
```

A terceira busca, fora realizada com ‘*eps*’ de 1,2 a 1,8 e ‘*samples*’ de 22 a 46 (de 2 em 2), onde os dez melhores resultados foram esses:

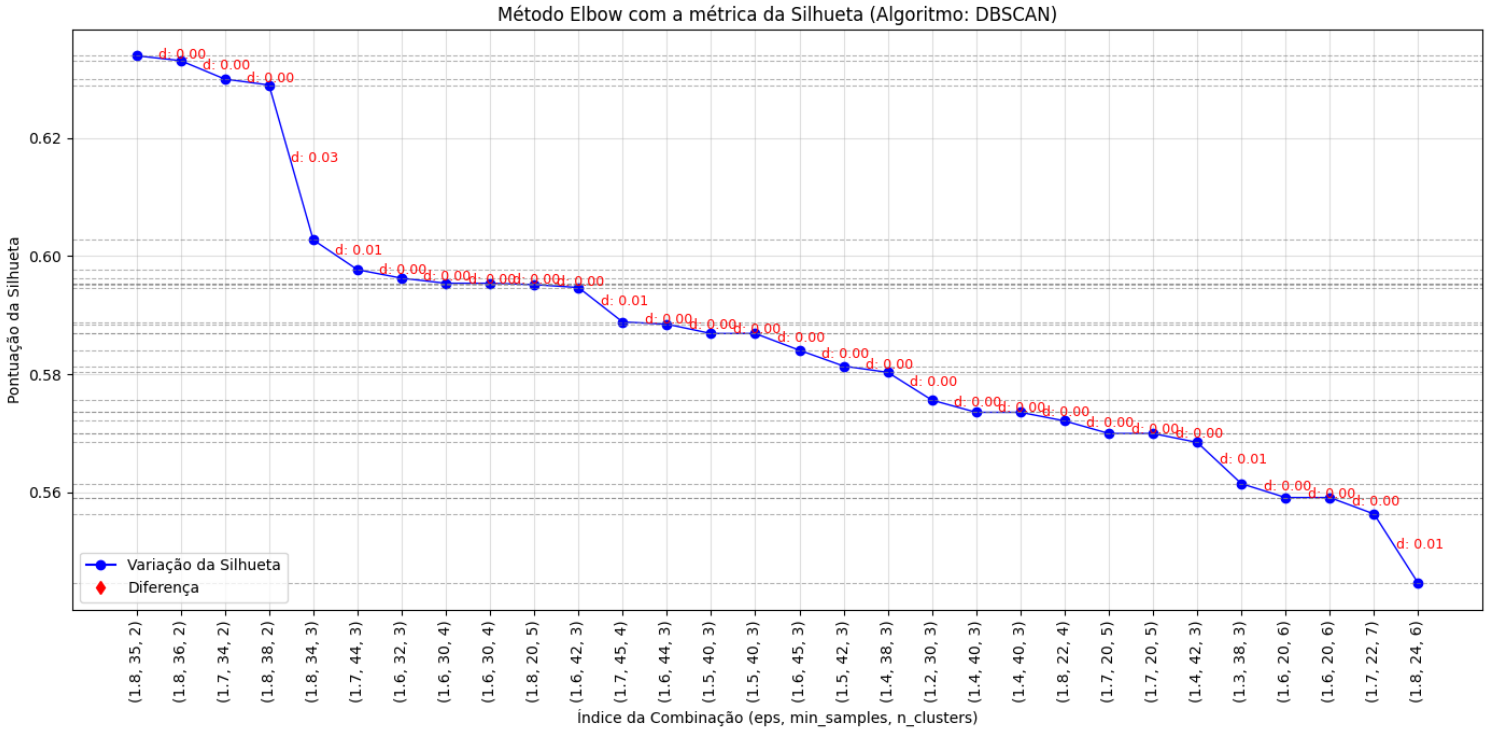
```
eps_range = np.arange(1.2, 1.8, 0.1)
samples_range = np.arange(22, 46, 2)

avaliar_dbscan(dados_padronizados, eps_range, samples_range)
```

✓ 7m 35.5s

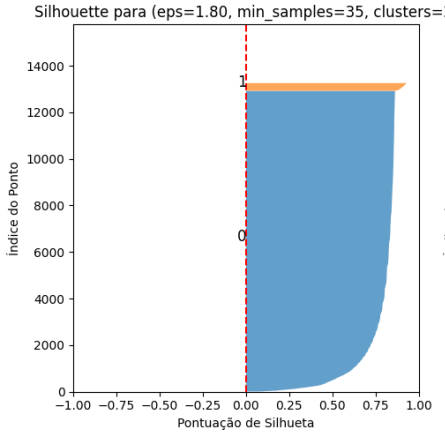
As 10 melhores pontuações foram:

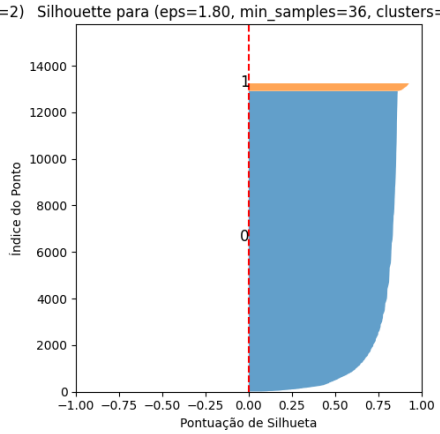
eps	samples	score	clusters
1.80	35	0.633910	2
1.80	36	0.633078	2
1.70	34	0.629957	2
1.80	38	0.628957	2
1.80	34	0.602747	3
1.70	44	0.597703	3
1.60	32	0.596270	3
1.60	30	0.595409	4
1.60	30	0.595409	4
1.80	20	0.595180	5

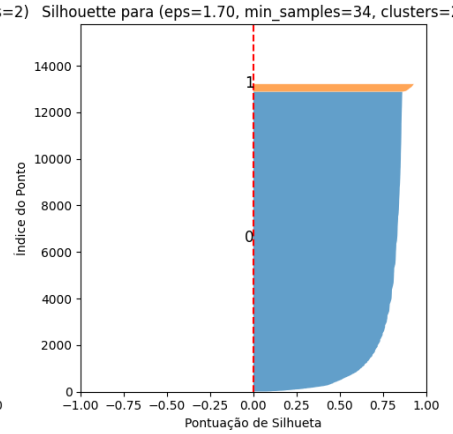
Em seguida, foi impresso um gráfico Elbow com a métrica da silhueta, dos 30 melhores scores, em ordem decrescente:

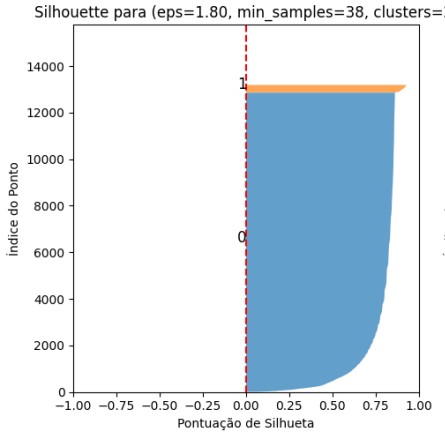


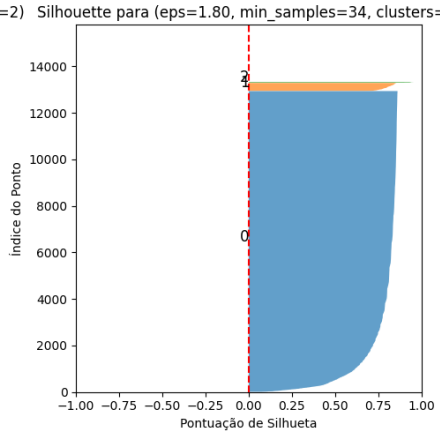
E plota gráficos de silhueta para analisar a qualidade dos clusters, com as 15 melhores combinações.

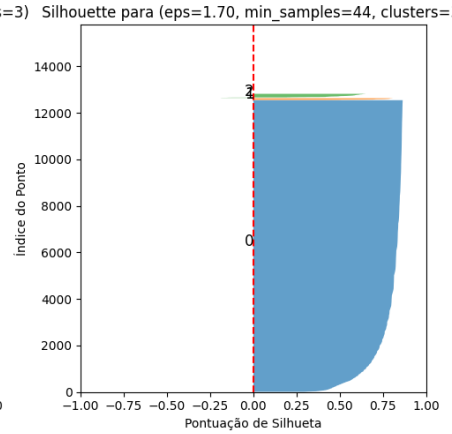

Silhouette para (eps=1.80, min_samples=35, clusters=2)
Índice do Ponto
Pontuação de Silhueta

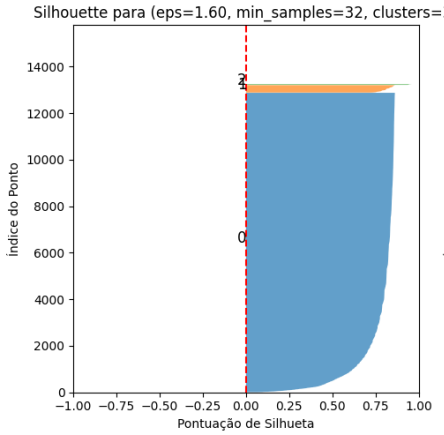

Silhouette para (eps=1.80, min_samples=36, clusters=2)
Índice do Ponto
Pontuação de Silhueta

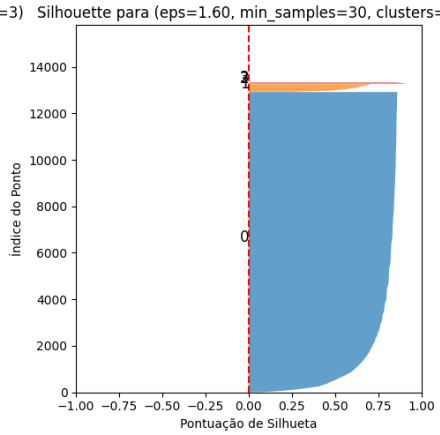

Silhouette para (eps=1.70, min_samples=34, clusters=2)
Índice do Ponto
Pontuação de Silhueta

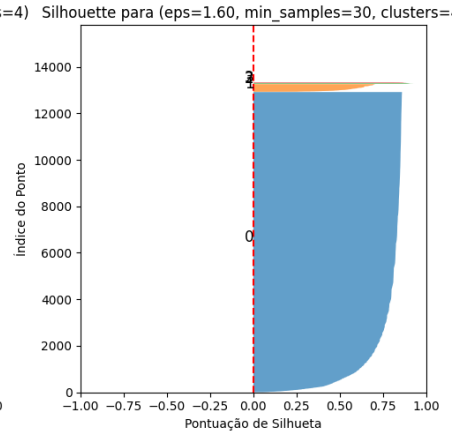

Silhouette para (eps=1.80, min_samples=38, clusters=2)
Índice do Ponto
Pontuação de Silhueta

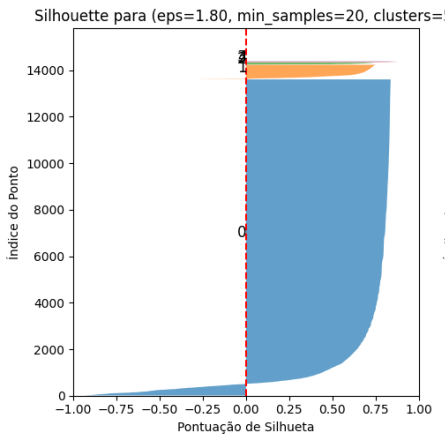

Silhouette para (eps=1.80, min_samples=34, clusters=3)
Índice do Ponto
Pontuação de Silhueta

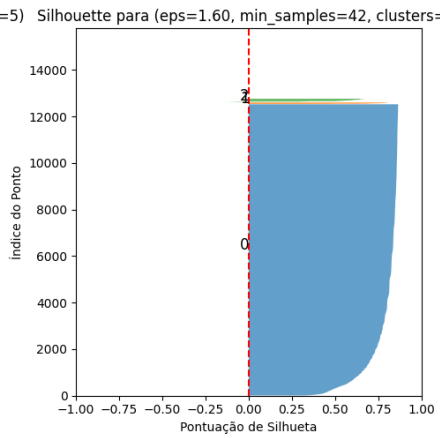

Silhouette para (eps=1.70, min_samples=44, clusters=3)
Índice do Ponto
Pontuação de Silhueta

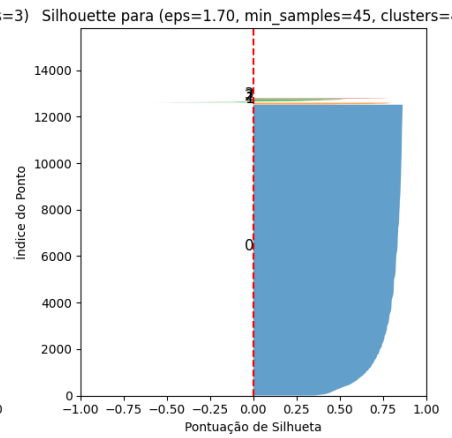

Silhouette para (eps=1.60, min_samples=32, clusters=3)
Índice do Ponto
Pontuação de Silhueta

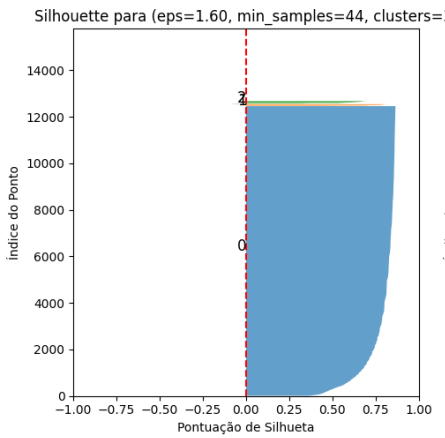

Silhouette para (eps=1.60, min_samples=30, clusters=4)
Índice do Ponto
Pontuação de Silhueta

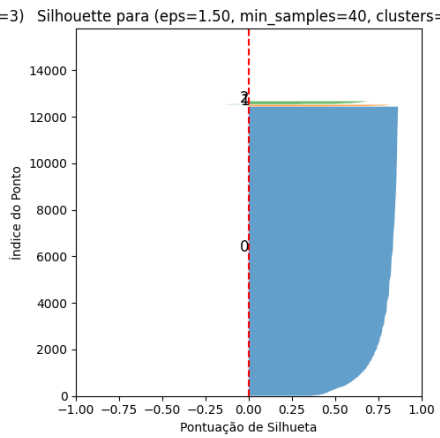

Silhouette para (eps=1.60, min_samples=30, clusters=4)
Índice do Ponto
Pontuação de Silhueta

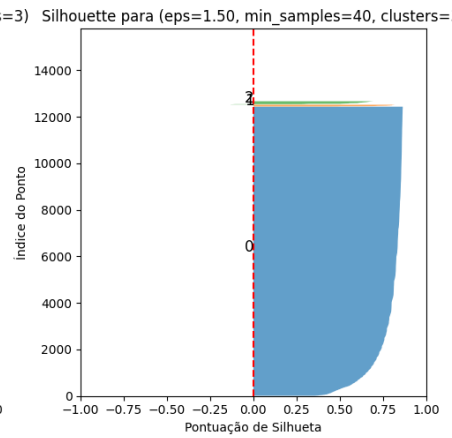

Silhouette para (eps=1.80, min_samples=20, clusters=5)
Índice do Ponto
Pontuação de Silhueta


Silhouette para (eps=1.60, min_samples=42, clusters=3)
Índice do Ponto
Pontuação de Silhueta

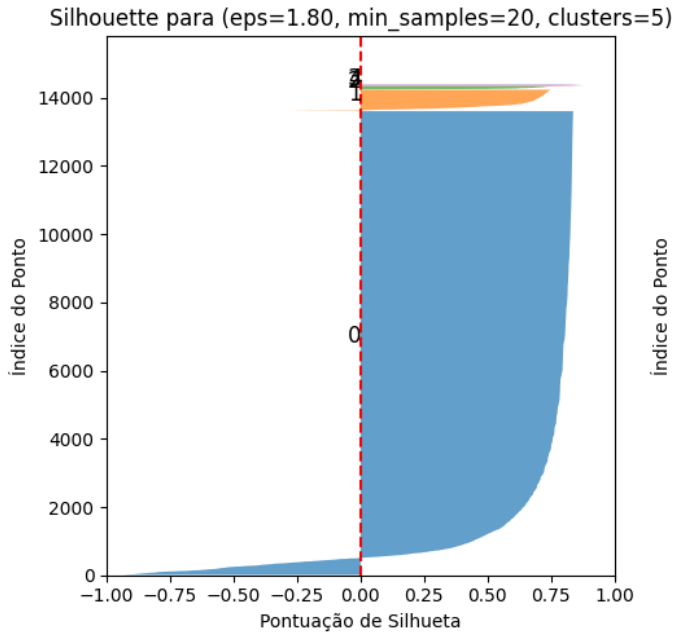

Silhouette para (eps=1.70, min_samples=45, clusters=4)
Índice do Ponto
Pontuação de Silhueta


Silhouette para (eps=1.60, min_samples=44, clusters=3)
Índice do Ponto
Pontuação de Silhueta


Silhouette para (eps=1.50, min_samples=40, clusters=3)
Índice do Ponto
Pontuação de Silhueta


Silhouette para (eps=1.50, min_samples=40, clusters=3)
Índice do Ponto
Pontuação de Silhueta

Com base nessa análise foi escolhido os parâmetros '*eps*' = 1.8, '*min_samples*' = 20, fazendo 5 clusters bem definidos.



3. Compare os dois resultados, aponte as semelhanças e diferenças e interprete.

R: Como primeira análise, percebe-se que o DBSCAN, em razão de necessitar de dois hiperparâmetros ('*eps*' e '*min_samples*'), possui um intervalo de pesquisa bem maior do score de silhueta.

Além disso, o DBSCAN agrupa os ruídos (*noises*) como '-1', fazendo com que tenhamos que tratar esse grupo, limitando a comparação entre os modelos, visto que não é eficaz comparar um grupo do K-Means com ruídos do DBSCAN porque os ruídos estão espalhados por todos os grupos do K-Means, conforme já analisado.

Ao analisar o índice de acurácia, temos uma similaridade de cerca de 11% entre os modelos:

```
print("Acurácia entre K-Means e DBSCAN:", accuracy)
✓ 0.0s
Acurácia entre K-Means e DBSCAN: 0.11005785181571061
```

Ambos os algoritmos podem gerar valores de silhueta positivos quando o agrupamento for bem feito. Se os clusters são bem definidos e os pontos estão bem agrupados, ambos os algoritmos devem exibir um bom resultado de silhueta.

Em razão da necessidade de escolhas de hiperparâmetros, (‘*k* ’ para o K-Means; e ‘*eps*’ e ‘*min_samples*’ para o DBSCAN). A escolha errada pode levar a uma pontuação de silhueta ruim.

Caso o dataset possua outliers, o DBSCAN pode resultar em uma pontuação de silhueta mais alta, pois esses são identificados e excluídos do cálculo. Já o K-Means, sendo sensível a outliers, pode reduzir a pontuação de silhueta se houver muitos pontos distantes do centroide.

O K-Means ainda tende a formar clusters esféricos, o que pode não ser adequado se os dados tiverem clusters de formas irregulares. Já o DBSCAN, por ser um algoritmo baseado em densidade, é mais flexível e pode capturar clusters de formas mais complexas.

Como os resultados de score da silhueta foram positivos (cerca de 0,43 para o K-Means e 0,59 para o DBSCAN), podemos dizer que, embora os dados estejam bem densos, também estão bem distribuídos nos parâmetros propostos. Em geral, pode-se dizer que a comparação entre K-Means e DBSCAN em termos de silhueta depende muito das características dos dados.

4. Escolha mais duas medidas de validação para comparar com o índice de silhueta e analise os resultados encontrados. Observe, para a escolha, medidas adequadas aos algoritmos.

R: Foram escolhidas as métricas de ‘Davies-Bouldin’ e ‘Calinski-Harabasz’.

O Índice de Davies-Bouldin mede a qualidade do agrupamento com base na relação entre a coesão e a separação dos clusters.

Já o Índice de Calinski-Harabasz, também conhecido como Índice de Variância entre e dentro dos clusters, mede a separação global dos clusters em comparação com a compactação interna.

Vale dizer que ambos os índices são compatíveis com o K-Means e o DBSCAN:

```
Comparação das Medidas
        Silhouette Score  Davies-Bouldin Index  Calinski-Harabasz Index
KMeans          0.388260              0.838755              36844.581287
DBSCAN          0.709336              0.313788              10393.253757
```

5. Realizando a análise, responda: A silhueta é um o índice indicado para escolher o número de clusters para o algoritmo de DBScan?

R: Se os dados estiverem bem dispostos e o cluster de ruído for isolado, é possível utilizar o índice de silhueta para escolher os hiperparâmetros necessários (‘*eps*’ e ‘*min_samples*’), avaliar a qualidade de clusterização e ainda para se chegar ao melhor número de clusters do DBSCAN.

Parte 4. Medidas de similaridade

1. Um determinado problema, apresenta 10 séries temporais distintas. Gostaríamos de agrupá-las em 3 grupos, de acordo com um critério de similaridade, baseado no valor máximo de correlação cruzada entre elas. Descreva em tópicos todos os passos necessários.

R: Para realizar o agrupamento de séries temporais, reduzindo a sua dimensionalidade, em primeiro lugar, devemos realizar a preparação correta dos dados. Garantir que todas as séries temporais estão no formato correto para a análise, sem dados nulos ou repetidos.

Além disso, caso possuam escalas diferentes, deve-se realizar a normalização de seus dados, para evitar magnitudes diferentes que afetem os resultados da correlação.

Em segundo lugar, deve-se calcular a correlação cruzada, que mede a similaridade entre duas séries temporais, para isso pode-se utilizar o ‘*numpy.correlate*’ ou ‘*scipy.signal_correlate*’, onde para cada par, deve registrar o número máximo da correlação cruzada, sendo pois, o critério de similaridade.

A partir disso, é possível criar uma matriz de similaridade, utilizando gráficos como o heatmap, por exemplo.

E daí, enfim, agrupar os dados resultantes nos três clusters desejados.

2. Para o problema da questão anterior, indique qual algoritmo de clusterização você usaria. Justifique.

R: Para agrupar dados baseados em uma matriz de similaridade, é recomendável o modelo de agrupamento hierárquico (HCluster), utilizando o dendrograma como ferramenta de visualização.

Isso porque o agrupamento hierárquico cria uma estrutura de árvore, permitindo visualizar todos os níveis de similaridade entre as séries, através das distâncias de suas folhas.

Além disso, o algoritmo hierárquico funciona diretamente com matrizes, como a de correlação cruzada, proposta aqui.

3. Indique um caso de uso para essa solução projetada.

R: Um caso de uso prático para essa solução projetada seria na análise de comportamento de consumidores em diferentes períodos, especialmente em setores como o varejo ou o financeiro.

No caso proposto em tela, "uma loja online deseja analisar os padrões de 10 clientes ao longo de um ano, onde cada cliente possui uma série temporal que representa o número de compras feitas no mês".

A clusterização poderia possibilitar traçar perfis de clientes, de acordo com seus padrões de compra.

4. Sugira outra estratégia para medir a similaridade entre séries temporais. Descreva em tópicos os passos necessários.

R: Uma outra estratégia para medir a similaridade entre séries temporais é a utilização da distância Dynamic Time Warping (DTW).

Essa técnica é particularmente útil quando as séries temporais podem ter defasagens ou variações de amplitude, mas ainda assim têm padrões semelhantes.

Para isso, os passos necessários seriam:

1. Realizar o pré-processamento das séries temporais conforme já tratado nas perguntas anteriores;

2. Calcular a distância de DTW, de forma a medir a distância entre duas séries temporais, tentando “alinhar” os pontos das séries de maneira não linear, minimizando a diferença entre elas;

```
from fastdtw import fastdtw

from scipy.spatial.distance import euclidean



# Transformar as séries em formato adequado para o cálculo (lista de tuplas)
```

```
serie_1 = [(i, serie_1[i]) for i in range(len(serie_1))]

serie_2 = [(i, serie_2[i]) for i in range(len(serie_2))]


# Calcular a distância DTW

distancia, caminho = fastdtw(serie_1, serie_2, dist=euclidean)


# Exibir o valor da distância DTW e o caminho de alinhamento

print(f"Distância DTW: {distancia}")

print(f"Caminho de Alinhamento: {caminho}")
```

3. Construir a matriz de similaridade, onda cada célula contém o valor da distância entre a série ‘a’ e a série ‘b’;

```
import numpy as np
from scipy.spatial.distance import euclidean

# Lista de todas as séries temporais
series = [serie_1, serie_2, serie_3]

# Número de séries temporais
n = len(series)

# Criando uma matriz de similaridade (inicializada com zeros)
similaridade = np.zeros((n, n))

# Calculando as distâncias Euclidianas entre cada par de séries
for i in range(n):
  for j in range(n):
    if i != j:
      similaridade[i][j] = euclidean(series[i], series[j])

# Exibindo a matriz de similaridade
print("Matriz de Similaridade (distância Euclidiana):")
print(similaridade)
```

4. Realizar a clusterização utilizando o agrupamento hierárquico, conforme já tratado nas perguntas anteriores.