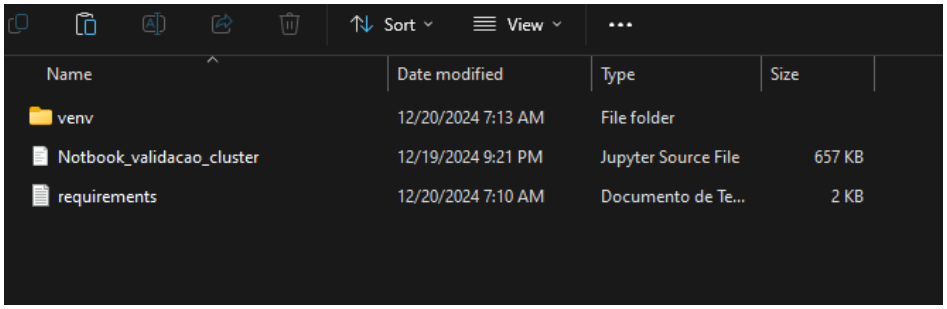
1. Ambiente de desenvolvimento:




```
(venv) C:\Users\guilh\OneDrive\Área de Trabalho\Tabalho validacao cluster>python --version
Python 3.11.2
```

```
(venv) C:\Users\guilh\OneDrive\Área de Trabalho\Tabalho validacao cluster>pip freeze
asttokens==2.4.1
click==8.1.3
colorama==0.4.6
comm==0.2.2
contourpy==1.2.0
cycler==0.12.1
dash==2.9.1
dash-bootstrap-components==1.4.1
dash-core-components==2.0.0
dash-html-components==2.0.0
dash-table==5.0.0
debugpy==1.8.1
decorator==5.1.1
distlib==0.3.9
et-xmlfile==1.1.0
executing==2.0.1
fastjsonschema==2.20.0
filelock==3.16.1
Flask==2.2.3
fonttools==4.50.0
ipykernel==6.29.3
ipython==8.22.2
ipywidgets==8.1.2
itsdangerous==2.1.2
jedi==0.19.1
Jinja2==3.1.2
joblib==1.3.2
jupyter_client==8.6.1
jupyter_core==5.7.2
jupyterlab_widgets==3.0.10
kiwisolver==1.4.5
kmodes==0.12.2
MarkupSafe==2.1.2
matplotlib==3.8.3
matplotlib-inline==0.1.6
nest-asyncio==1.6.0
numpy==1.24.2
openpyxl==3.1.3
packaging==24.0
pandas==1.5.3
parso==0.8.3
pillow==10.2.0
platformdirs==4.2.0
plotly==5.13.1
prompt-toolkit==3.0.43
psutil==5.9.8
pure-eval==0.2.2
Pygments==2.17.2
pyparsing==3.1.2
python-dateutil==2.8.2
pytz==2023.2
pywin32==306
pyzmq==25.1.2
rpds-py==0.21.0
scikit-learn==1.4.1.post1
scipy==1.12.0
seaborn==0.13.2
Send2Trash==1.8.3
six==1.16.0
sniffio==1.3.1
soupsieve==2.6
stack-data==0.6.3
tenacity==8.2.2
threadpoolctl==3.3.0
tinycss2==1.4.0
tornado==6.4
traitlets==5.14.2
types-python-dateutil==2.9.0.20241003
uri-template==1.3.0
urllib3==2.2.3
virtualenv==20.27.1
wcwidth==0.2.13
webcolors==24.11.1
```

2. Escolha da Base:

Foi utilizado a base de dados **Sentiment140 dataset with 1.6 million tweets** do Kaggle no link: https://www.kaggle.com/datasets/kazanova/sentiment140/data

Essa base contém tweets variados, categorizados de acordo com os sentimentos: -2 para negativos, 0 para neutros e 2 para positivos. O primeiro objetivo deste trabalho é testar duas abordagens para agrupar os tweets. A primeira consiste em identificar agrupamentos temáticos utilizando o algoritmo K-means, combinado com o método do cotovelo, visando a criação de clusters genéricos que melhor representem as diferentes temáticas dos tweets.

O segundo objetivo é agrupar os tweets de forma a mapear o comportamento dos usuários com base nos tweets uns dos outros. Com isso, pretendemos desenvolver um sistema de recomendação de tweets, no qual usuários com tweets no mesmo agrupamento podem se recomendar mutuamente dentro de seu cluster.

Será utilizada apenas a coluna 'text'. Então iniciaremos o tratamento removendo pontuações, marcações, links, números e espaços desnecessários.

```
import re
def limpar_texto(texto):
    texto = re.sub(r'@\w+', '', texto)  # Remove menções (@usuario)
    texto = re.sub(r'http\S+|www.\S+', '', texto)  # Remove URLs
    texto = re.sub(r'#\w+', '', texto)  # Remove hashtags
    texto = re.sub(r'[^\w\s]', '', texto)  # Remove pontuações
    texto = re.sub(r'\d+', '', texto)  # Remove números
    texto = texto.lower().strip()  # Converte para minúsculas e remove espaços extras
    return texto

tweets_pretratados = [limpar_texto(tweet) for tweet in lista_de_textos]

tweets_pretratados
```

Após o tratamento, criamos um dicionário de frequência de palavras, para entender a distribuição das palavras nos textos:

```
from collections import Counter, defaultdict

palavras = []
for tweet in tweets_pretratados:
    palavras.extend(tweet.split())

# Criar um DataFrame para contar a frequência de cada palavra
df_palavras = pd.DataFrame(palavras, columns=["palavra"])
frequencia_palavras = df_palavras["palavra"].value_counts()
frequencia_palavras = pd.DataFrame(frequencia_palavras, columns=["count"])

# Exibir o dicionário
print(frequencia_palavras.head(60))
```

	index	count
0	count	64960.000000
1	mean	18.257127
2	std	346.964601
3	min	1.000000
4	25%	1.000000
5	50%	1.000000
6	75%	2.000000
7	max	36556.000000

Quartil 1 (Q1): 1
Quartil 3 (Q3): 2
IQR: 1
Limite Superior: 3.5

Observamos que 75% das palavras se repetem no máximo em dois tweets, o que sugere que essas palavras têm baixa relevância para nossos agrupamentos. Por isso, adotaremos o limite superior de frequência de palavras (maior que 4) para determinar quais termos serão mantidos no texto.

palavra	count
i	36556
the	32195
to	31185
a	24750
you	22664
and	18323
my	15821
for	14442
it	14088
is	13532
in	12095
of	11539
on	10841
im	9589
me	8935
that	8781
so	8097
just	7915
with	7835
good	7697
have	7668
be	6813
its	6076
day	5909

Ao analisar as palavras mais frequentes, verificamos que muitas delas são termos genéricos, o que poderia enviesar os agrupamentos, tornando-os predominantemente genéricos. Assim, definimos o limite superior de frequência como a média (18) somada ao desvio padrão (347). Dessa forma, as palavras que permanecerão no texto terão uma frequência inferior a 365.

```
frequencia_palavras = frequencia_palavras.reset_index()
frequencia_palavras=frequencia_palavras[(frequencia_palavras['count']>4) & (frequencia_palavras['count']<360)]
len(frequencia_palavras)
```

```
9873
```

```
def remover_palavras(texto, palavras_para_remover):
    palavras = texto.split()  # Divide em palavras
    palavras_filtradas = [palavra if palavra in palavras_para_remover else '' for palavra in palavras]
    return " ".join(palavras_filtradas)

palavras_para_remover = set(list(frequencia_palavras['palavra']))
tweets_pretratados = [remover_palavras(tweet,palavras_para_remover) for tweet in tweets_pretratados]
tweets_pretratados
```

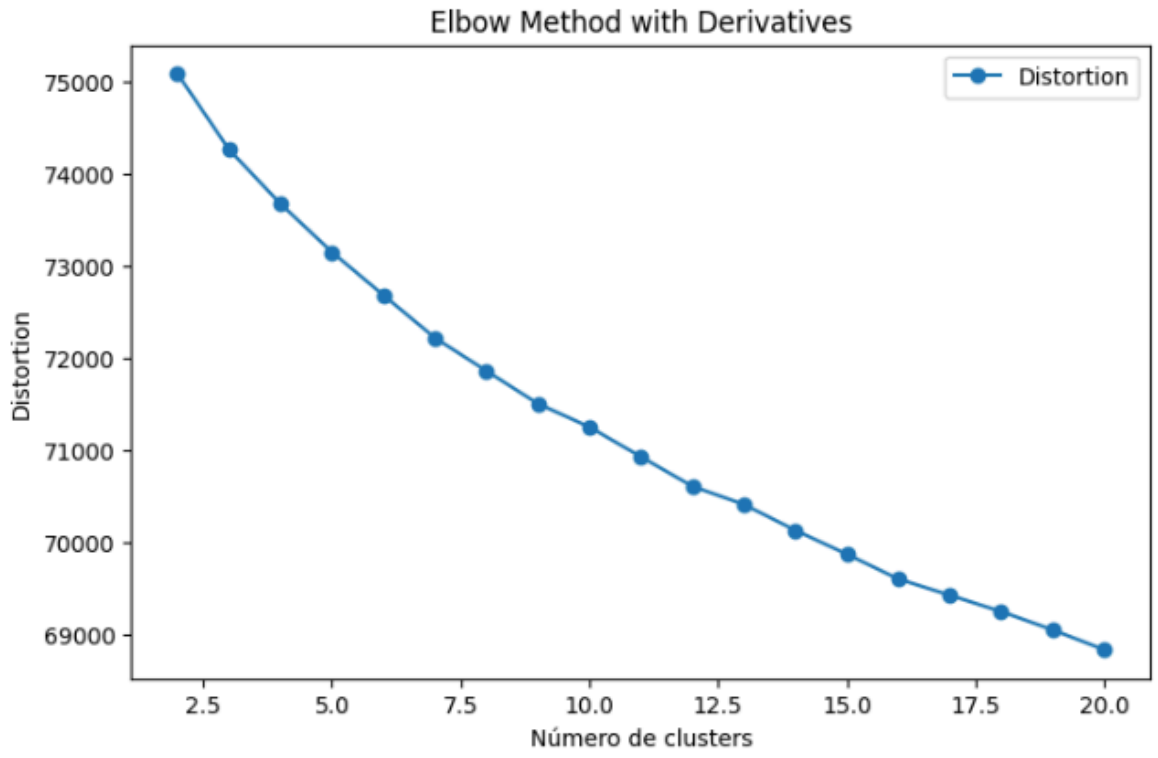
Com isso, mantivemos 9.873 palavras no dicionário, e as demais foram removidas.

Para a vetorização dos textos, optamos pelo uso do modelo Transformer. O Transformer é uma arquitetura de rede neural baseada em atenção, projetada para processar sequências de dados, como texto, de maneira mais eficiente do que modelos anteriores, como RNNs e LSTMs. Utilizamos um modelo pré-treinado, o SBERT, que é baseado no BERT e é especialmente adequado para tarefas de similaridade semântica e comparação de sentenças. Ele usa a arquitetura Transformer para gerar embeddings (representações vetoriais) de sentenças, preservando seu significado e contexto.

```
model = SentenceTransformer('all-MiniLM-L6-v2')

embeddings = model.encode(list(df['texto_tratado']))
```

3. Clusterização



Ao aplicar o método K-means e o método do cotovelo, plotamos o gráfico de distâncias e identificamos que, em algumas transições de clusters, a mudança na inclinação da curva é mais acentuada, o que pode ser observado pela derivada. Escolhemos o número 7 de clusters, pois a inclinação entre o cluster 7 e o 8 segue uma tendência mais linear em comparação com as distâncias entre outros clusters:

2 para 3: -821.89,
3 para 4: -588.97
4 para 5: -524.08
5 para 6: -467.70
6 para 7: -463.19
7 para 8: -358.50
8 para 9: -356.07
9 pra 10: -252.50

Embora os clusters 9 e 10 apresentam boas características de agrupamento, porém optamos por algo mais genérico neste estágio inicial.

Após agrupar os textos, listamos as 100 palavras com mais frequência e tiramos as seguintes conclusões:

Cluster 4
Contexto: Viagens e Atividades ao Ar Livre. Palavras como "beach", "trip", "vacation", "sunny", "airport", "holiday", "august", "swimming", "london", "lake", "park" e "tour" sugerem um tema relacionado a viagens, clima e atividades ao ar livre.

Cluster 3
Contexto: Emoções e Conversas Cotidianas. Palavras como "hate", "mind", "head", "damn", "laugh", "care", "hell", "fuck", "sad", "aww", "wrong", "proud", "sleeping" e "fine" indicam um cluster focado em emoções, opiniões e expressões pessoais cotidianas.

Cluster 6
Contexto: Entretenimento e Cultura Pop. Palavras como "songs", "rock", "movies", "concert", "fan", "season", "episode", "guitar", "disney", "radio", "dance" e "youtube" apontam para um tema de música, filmes, televisão e cultura pop.

Cluster 0
Contexto: Comida e Bebidas. Palavras como "breakfast", "tea", "chocolate", "pizza", "cake", "drinks", "cooking", "wine", "salad", "burger", "bacon", "cookies" e "ice cream" claramente sugerem um foco em comida e bebidas.

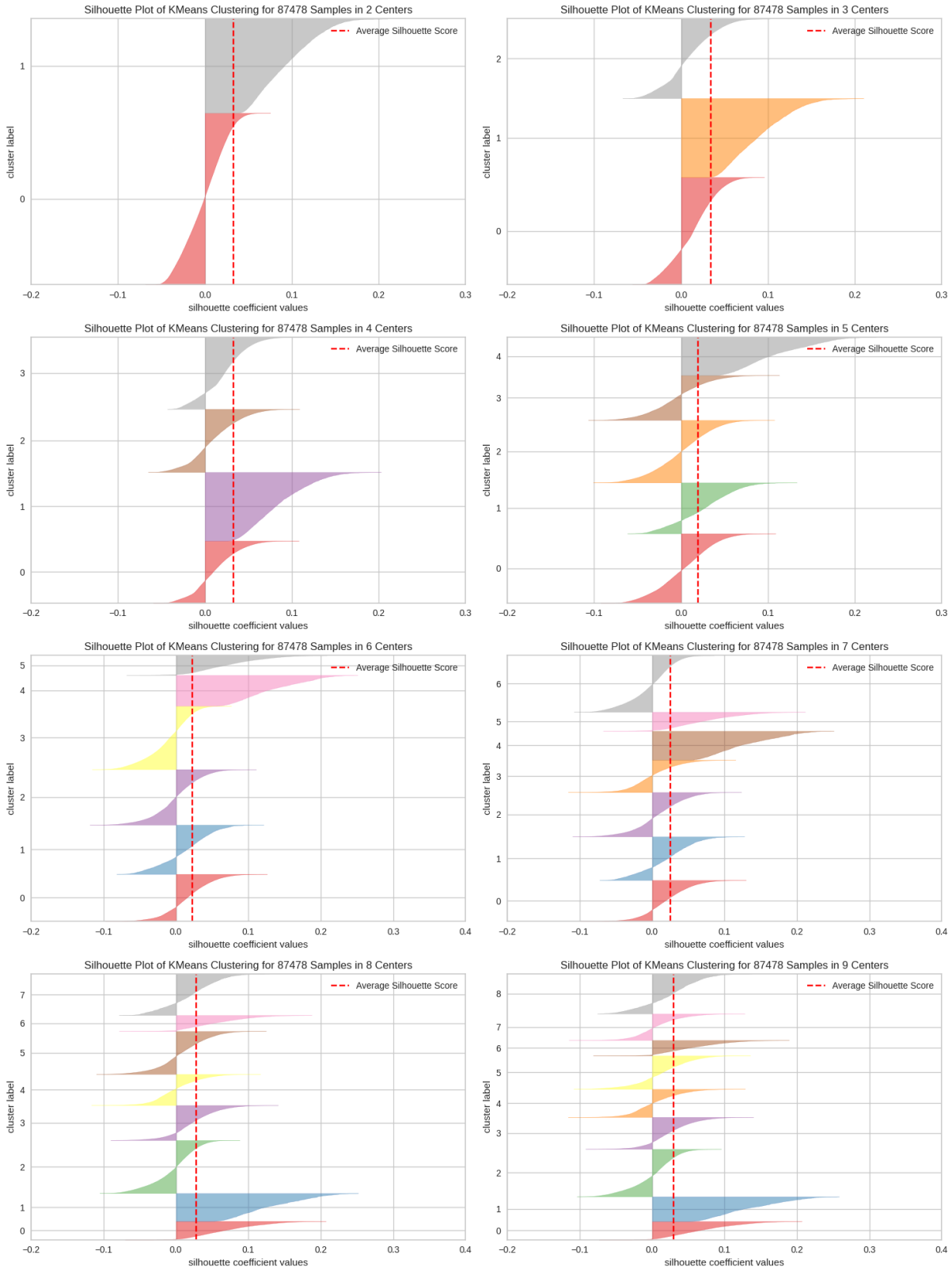
Cluster 2
Contexto: Tecnologia e Redes Sociais. Palavras como "facebook", "email", "website", "online", "internet", "profile", "app", "google", "photos", "laptop", "tweeting" e "media" sugerem um tema relacionado a tecnologia, redes sociais e comunicação digital.

Cluster 1
Contexto: Interações Pessoais e Conversações. Palavras como "aww", "fine", "agree", "congratulations", "dear", "hugs", "thankyou", "sweetie", "exciting", "hoping", "update" e "works" apontam para um cluster focado em interações pessoais, emoções positivas e respostas.

Cluster 5
Contexto: Tarefas e Rotina. Palavras como "vote", "class", "homework", "question", "office", "meeting", "starting", "running", "plan", "team", "interview" e "learning" sugerem um tema de trabalho, estudos e atividades rotineiras.

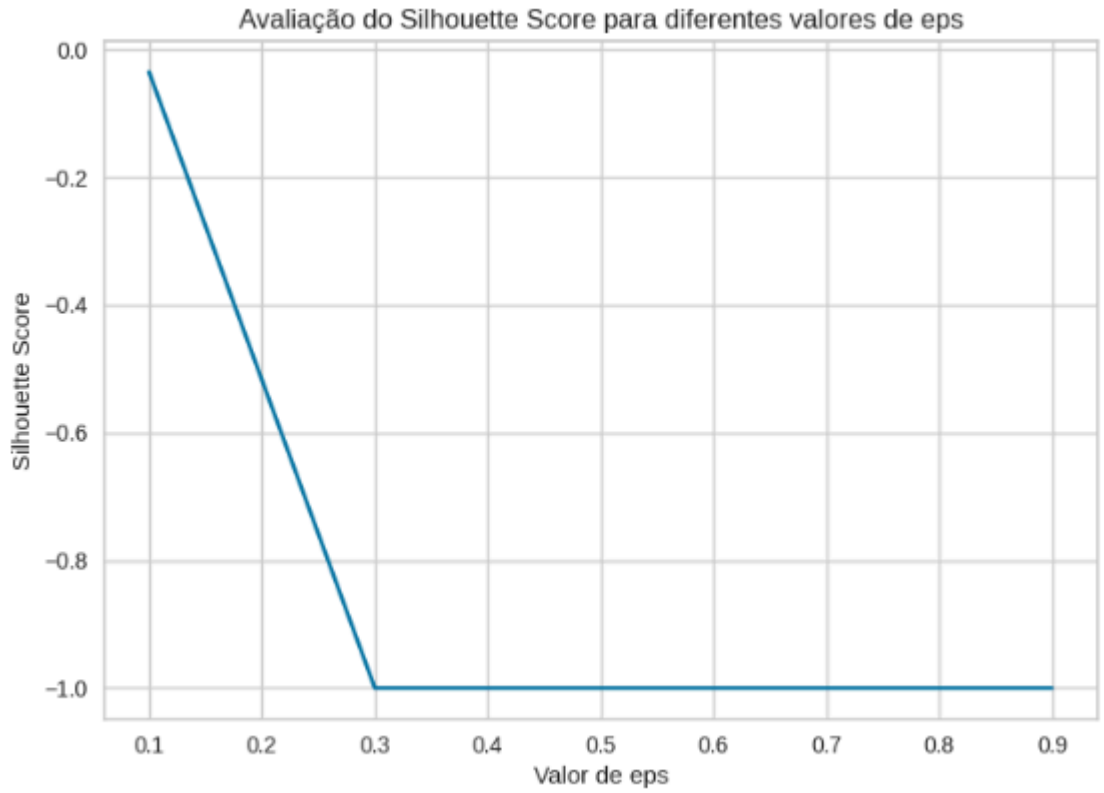
Para nosso primeiro objetivo, vamos utilizar o método da silhueta, que refere-se a um método de interpretação e validação da consistência dentro de agrupamentos de dados, para escolher o melhor agrupamento para nosso sistema de recomendação e qual será o método de agrupamento.

Aplicando a silhueta no K Means, obtivemos esses resultados:



Estaria na avaliação da média escolheria o número de clusters 4 ou 9, além da média ser maior, com 4 cluster temos menores valores negativos e prováveis erros de cluster.

Após diversas tentativas, não conseguimos adquirir nenhum cluster ótimo com o DBscan, com o índice de silhueta com essa visualização.



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

Resposta: Não é um bom método para avaliar o DBSCAN. E a utilização do método se torna inviável.

Outros índices para avaliação:
Índice de Dunn para K-Médias: 0.434
Índice de Dunn para DBSCAN: 0.0878
Coeficiente de V-Measure para K-Médias: 1.00
Coeficiente de V-Measure para DBSCAN: 0.18

O Índice de Dunn é maior para o K-Means, indicando que:
Os clusters gerados pelo K-Means têm maior separação entre clusters e menor compactação intracluster em comparação com o DBSCAN.

O DBSCAN obteve um valor menor, o que pode indicar que:
Alguns clusters são muito espalhados (baixa compactação).
As distâncias entre clusters diferentes não são suficientemente altas.

A V-Measure para o K-Means é perfeita (1.0), o que sugere que:
Os clusters formados pelo K-Means coincidem exatamente com as classes verdadeiras (ground truth). Isso é possível se você usou os próprios rótulos de K-Means como referência no cálculo, ou se os dados tinham clusters bem definidos
.
A V-Measure para o DBSCAN é baixa (0.1818), indicando:
Os clusters gerados pelo DBSCAN têm baixa correspondência com as classes verdadeiras. Isso pode ocorrer se o valor de eps ou min_samples não foi bem ajustado.

5.Medidas de Similaridade:

Passos para agrupar as 10 séries temporais em 3 grupos com base na correlação cruzada:

- Pré-tratamento das séries temporais: retirada de valores duplicados, tratamento de valores nulos e etc.
- Modelagem dos dados: geração de colunas novas, modelar os dados em outra arquitetura, etc
- Escolha de um algoritmo de clusterização: K-means, clusterização hierárquica ou DBSCANS.

- Aplicação do algoritmo de clusterização: Aplicar um algoritmo de clusterização (por exemplo, K-means ou DBSCAN) usando a matriz de similaridade para identificar os 3 grupos.
- O número de clusters deve ser 3, conforme o critério da questão.
- Avaliação do agrupamento: Verificar os resultados do agrupamento observando as séries temporais dentro de cada grupo e a similaridade dentro de cada grupo.

Algoritmo de Clusterização Escolhido na hipótese acima: **K-means**

Justificativa:

O K-means é simples e eficiente para agrupamento, especialmente quando o número de clusters (neste caso, 3) é conhecido de antemão.

Embora o K-means tenha a limitação de ser sensível à inicialização e à forma das distribuições dos dados, ele tende a ser eficaz em cenários onde o número de clusters é pequeno e as distribuições dos dados não são muito distorcidas.