IAN MIRANDA DE SOUZA

Projeto da Disciplina
Algoritmos de Inteligência Artificial para clusterização
24E4_2

RIO DE JANEIRO
2024

Projeto da Disciplina

Algoritmos de Inteligência Artificial para clusterização - 24E4_2

> Importação das bibliotecas necessárias para execução desse notebook

In [349…
```
import subprocess
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score, silhouette_samples, euclidean_distances
from sklearn_extra.cluster import KMedoids

from scipy.cluster import hierarchy as sch
```

Parte 1 - Infraestrutura

Para as questões a seguir, você deverá executar códigos em um notebook Jupyter, rodando em ambiente local, certifique-se que:

1) Você está rodando em Python 3.9+

In [350…
```
print("Resposta:")
print(f"Versão do Python: {sys.version_info.major}.{sys.version_info.minor}.{sys.version_in
```

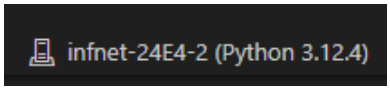
```
Resposta:
Versão do Python: 3.12.4
```

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

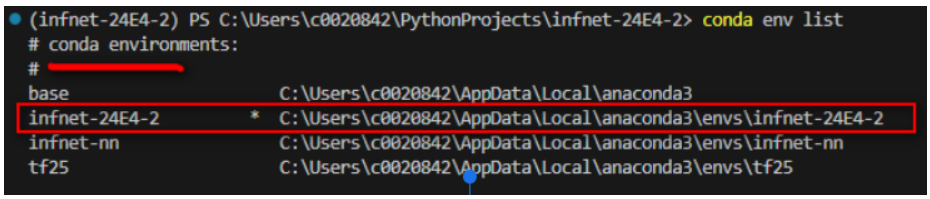
In [351…
```
print("Resposta:")
print(sys.version)
```

```
Resposta:
3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 64 bit (AMD6
4)]
```

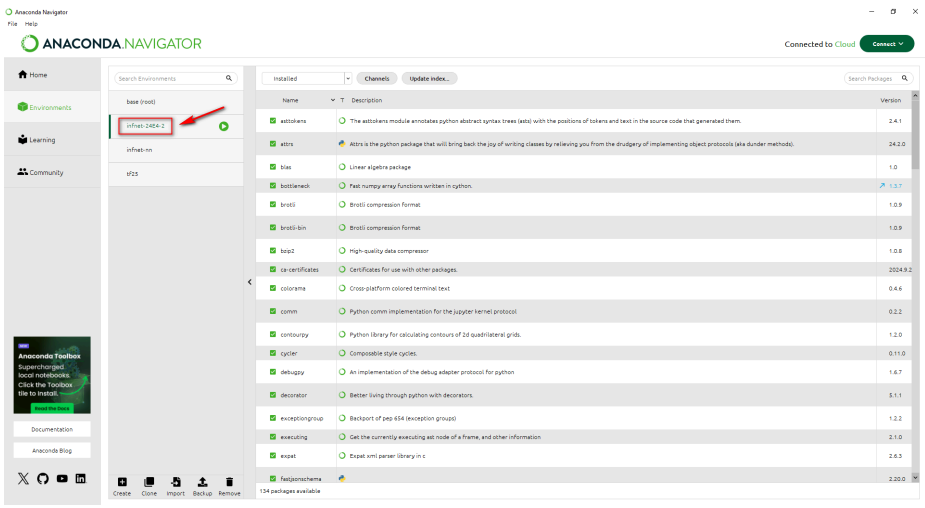
> **Evidências:**
> Print Screen do ambiente Anaconda:
>
> 1. Ambiente virtual no Visual Studio Code



2. Ambiente virtual do Anaconda no Visual Studio Code



3. Ambiente Virtual do Anaconda Navigator



3) Todas as bibliotecas usadas nesse exercícios estão instaladas em um ambiente virtual específico

In [352…

```
installed_packages = subprocess.check_output(['conda', 'list']).decode()
print("Resposta:")
print("Lista de todas as bibliotecas instaladas no ambiente Anaconda para execução desse no
print(installed_packages)
```

Resposta:
Lista de todas as bibliotecas instaladas no ambiente Anaconda para execução desse notebook:

```
# packages in environment at C:\Users\c0020842\AppData\Local\anaconda3\envs\infnet-24E4-2:
#
# Name                    Version                   Build  Channel
anyio                     4.6.2.post1        pyhd8ed1ab_0    conda-forge
argon2-cffi               23.1.0             pyhd8ed1ab_0    conda-forge
argon2-cffi-bindings      21.2.0          py312h2bbff1b_0
arrow                     1.3.0              pyhd8ed1ab_0    conda-forge
asttokens                 2.4.1              pyhd8ed1ab_0    conda-forge
async-lru                 2.0.4              pyhd8ed1ab_0    conda-forge
attrs                     24.2.0             pyh71513ae_0    conda-forge
babel                     2.16.0             pyhd8ed1ab_0    conda-forge
beautifulsoup4            4.12.3             pyha770c72_0    conda-forge
blas                      1.0                         mkl
bleach                    6.2.0              pyhd8ed1ab_0    conda-forge
bottleneck                1.3.7           py312he558020_0
brotli                    1.0.9                h2bbff1b_8
brotli-bin                1.0.9                h2bbff1b_8
brotli-python             1.0.9           py312hd77b12b_8
bzip2                     1.0.8                h2bbff1b_6
ca-certificates           2024.9.24            haa95532_0
cached-property           1.5.2                hd8ed1ab_1    conda-forge
cached_property           1.5.2              pyha770c72_1    conda-forge
certifi                   2024.8.30          pyhd8ed1ab_0    conda-forge
cffi                      1.17.1          py312h827c3e9_0
charset-normalizer        3.4.0              pyhd8ed1ab_0    conda-forge
colorama                  0.4.6              pyhd8ed1ab_0    conda-forge
comm                      0.2.2              pyhd8ed1ab_0    conda-forge
contourpy                 1.2.0           py312h59b6b97_0
cycler                    0.11.0             pyhd3eb1b0_0
debugpy                   1.6.7           py312hd77b12b_0
decorator                 5.1.1              pyhd8ed1ab_0    conda-forge
defusedxml                0.7.1              pyhd8ed1ab_0    conda-forge
entrypoints               0.4                pyhd8ed1ab_0    conda-forge
exceptiongroup            1.2.2              pyhd8ed1ab_0    conda-forge
executing                 2.1.0              pyhd8ed1ab_0    conda-forge
expat                     2.6.3                h5da7b33_0
fonttools                 4.51.0          py312h2bbff1b_0
fqdn                      1.5.1              pyhd8ed1ab_0    conda-forge
freetype                  2.12.1               ha860e81_0
glib                      2.78.4               hd77b12b_0
glib-tools                2.78.4               hd77b12b_0
greenlet                  3.1.1                    pypi_0    pypi
gst-plugins-base          1.18.5               h9e645db_0
gstreamer                 1.18.5               hd78058f_0
h11                       0.14.0             pyhd8ed1ab_0    conda-forge
h2                        4.1.0              pyhd8ed1ab_0    conda-forge
hpack                     4.0.0              pyh9f0ad1d_0    conda-forge
httpcore                  1.0.6              pyhd8ed1ab_0    conda-forge
httpx                     0.27.2             pyhd8ed1ab_0    conda-forge
hyperframe                6.0.1              pyhd8ed1ab_0    conda-forge
icc_rt                    2022.1.0             h6049295_2
icu                       58.2                 ha925a31_3
idna                      3.10               pyhd8ed1ab_0    conda-forge
importlib-metadata        8.5.0              pyha770c72_0    conda-forge
importlib_metadata        8.5.0                hd8ed1ab_0    conda-forge
importlib_resources       6.4.5              pyhd8ed1ab_0    conda-forge
intel-openmp              2023.1.0         h59b6b97_46320
ipykernel                 6.29.5             pyh4bbf305_0    conda-forge
ipython                   8.28.0             pyh7428d3b_0    conda-forge
isoduration               20.11.0            pyhd8ed1ab_0    conda-forge
jedi                      0.19.1             pyhd8ed1ab_0    conda-forge
jinja2                    3.1.4              pyhd8ed1ab_0    conda-forge
joblib                    1.4.2           py312haa95532_0
jpeg                      9e                   h827c3e9_3
json5                     0.9.28             pyhff2d567_0    conda-forge
```

```
jsonpointer               2.0                      py_0    conda-forge
jsonschema                4.23.0                pyhd8ed1ab_0    conda-forge
jsonschema-specifications 2024.10.1             pyhd8ed1ab_0    conda-forge
jsonschema-with-format-nongpl 4.23.0                 hd8ed1ab_0    conda-forge
jupyter-lsp               2.2.5                 pyhd8ed1ab_0    conda-forge
jupyter_client            7.4.9                 pyhd8ed1ab_0    conda-forge
jupyter_core              5.7.2              py312haa95532_0
jupyter_events            0.10.0                pyhd8ed1ab_0    conda-forge
jupyter_server            2.14.2                pyhd8ed1ab_0    conda-forge
jupyter_server_terminals  0.5.3                 pyhd8ed1ab_0    conda-forge
jupyterlab                4.2.5                 pyhd8ed1ab_0    conda-forge
jupyterlab_pygments       0.3.0                 pyhd8ed1ab_1    conda-forge
jupyterlab_server         2.27.3                pyhd8ed1ab_0    conda-forge
kiwisolver                1.4.4              py312hd77b12b_0
krb5                      1.21.3                   hdf4eb48_0    conda-forge
lcms2                     2.12                     h83e58a3_0
lerc                      3.0                      hd77b12b_0
libbrotlicommon           1.0.9                    h2bbff1b_8
libbrotlidec              1.0.9                    h2bbff1b_8
libbrotlienc              1.0.9                    h2bbff1b_8
libclang                  12.0.0             default_h627e005_2
libdeflate                1.17                     h2bbff1b_1
libffi                    3.4.4                    hd77b12b_1
libglib                   2.78.4                   ha17d25a_0
libiconv                  1.16                     h2bbff1b_3
libogg                    1.3.5                    h2bbff1b_1
libpng                    1.6.39                   h8cc25b3_0
libsodium                 1.0.20                   hc70643c_0    conda-forge
libtiff                   4.5.1                    hd77b12b_0
libvorbis                 1.3.7                    he774522_0
libwebp-base              1.3.2                    h2bbff1b_0
lz4-c                     1.9.4                    h2bbff1b_1
markupsafe                3.0.2                 pyhe1237c8_0    conda-forge
matplotlib                3.9.2              py312haa95532_0
matplotlib-base           3.9.2              py312hbdc63d0_0
matplotlib-inline         0.1.7                 pyhd8ed1ab_0    conda-forge
mistune                   3.0.2                 pyhd8ed1ab_0    conda-forge
mkl                       2023.1.0            h6b88ed4_46358
mkl-service               2.4.0              py312h2bbff1b_1
mkl_fft                   1.3.10             py312h827c3e9_0
mkl_random                1.2.7              py312h0158946_0
nbclient                  0.10.0                pyhd8ed1ab_0    conda-forge
nbconvert-core            7.16.4                pyhd8ed1ab_1    conda-forge
nbformat                  5.10.4                pyhd8ed1ab_0    conda-forge
nest-asyncio              1.6.0                 pyhd8ed1ab_0    conda-forge
notebook                  7.2.2                 pyhd8ed1ab_0    conda-forge
notebook-shim             0.2.4                 pyhd8ed1ab_0    conda-forge
numexpr                   2.8.7              py312h96b7d27_0
numpy                     1.26.4             py312hfd52020_0
numpy-base                1.26.4             py312h4dde369_0
openjpeg                  2.5.2                    hae555c5_0
openssl                   3.4.0                    h2466b09_0    conda-forge
overrides                 7.7.0                 pyhd8ed1ab_0    conda-forge
packaging                 24.1                  pyhd8ed1ab_0    conda-forge
pandas                    2.2.2              py312h0158946_0
pandocfilters             1.5.0                 pyhd8ed1ab_0    conda-forge
parso                     0.8.4                 pyhd8ed1ab_0    conda-forge
pcre2                     10.42                    h0ff8eda_1
pickleshare               0.7.5                    py_1003    conda-forge
pillow                    10.4.0             py312h827c3e9_0
pip                       24.2               py312haa95532_0
pkgutil-resolve-name      1.3.10                pyhd8ed1ab_1    conda-forge
platformdirs              4.3.6                 pyhd8ed1ab_0    conda-forge
playwright                1.48.0                      pypi_0    pypi
plotly                    5.24.1                      pypi_0    pypi
ply                       3.11               py312haa95532_1
prometheus_client         0.21.0                pyhd8ed1ab_0    conda-forge
```

```
prompt-toolkit            3.0.48             pyha770c72_0    conda-forge
psutil                    5.9.0           py312h2bbff1b_0
pure_eval                 0.2.3              pyhd8ed1ab_0    conda-forge
pybind11-abi              5                    hd3eb1b0_0
pycparser                 2.22               pyhd8ed1ab_0    conda-forge
pyee                      12.0.0                   pypi_0    pypi
pygments                  2.18.0             pyhd8ed1ab_0    conda-forge
pyparsing                 3.1.2           py312haa95532_0
pyqt                      5.15.10         py312hd77b12b_0
pyqt5-sip                 12.13.0         py312h2bbff1b_0
pysocks                   1.7.1              pyh0701188_6    conda-forge
python                    3.12.4               h14ffc60_1
python-dateutil           2.9.0              pyhd8ed1ab_0    conda-forge
python-fastjsonschema     2.20.0             pyhd8ed1ab_0    conda-forge
python-json-logger        2.0.7              pyhd8ed1ab_0    conda-forge
python-tzdata             2023.3             pyhd3eb1b0_0
pytz                      2024.1          py312haa95532_0
pywin32                   305             py312h2bbff1b_0
pywinpty                  2.0.10          py312h5da7b33_0
pyyaml                    6.0.2           py312h827c3e9_0
pyzmq                     24.0.1          py312h2bbff1b_0
qt-main                   5.15.2               he8e5bd7_7
referencing               0.35.1             pyhd8ed1ab_0    conda-forge
requests                  2.32.3             pyhd8ed1ab_0    conda-forge
rfc3339-validator         0.1.4              pyhd8ed1ab_0    conda-forge
rfc3986-validator         0.1.1              pyh9f0ad1d_0    conda-forge
rpds-py                   0.20.1                   pypi_0    pypi
scikit-learn              1.5.1           py312h0158946_0
scikit-learn-extra        0.3.0                    pypi_0    pypi
scipy                     1.13.1          py312hbb039d4_0
seaborn                   0.13.2          py312haa95532_0
send2trash                1.8.3              pyh5737063_0    conda-forge
setuptools                75.1.0          py312haa95532_0
sip                       6.7.12          py312hd77b12b_0
six                       1.16.0             pyh6c4a22f_0    conda-forge
sniffio                   1.3.1              pyhd8ed1ab_0    conda-forge
soupsieve                 2.5                pyhd8ed1ab_1    conda-forge
sqlite                    3.45.3               h2bbff1b_0
stack_data                0.6.2              pyhd8ed1ab_0    conda-forge
tbb                       2021.8.0             h59b6b97_0
tenacity                  9.0.0                    pypi_0    pypi
terminado                 0.18.1             pyh5737063_0    conda-forge
threadpoolctl             3.5.0           py312hfc267ef_0
tinycss2                  1.4.0              pyhd8ed1ab_0    conda-forge
tk                        8.6.14               h0416ee5_0
tomli                     2.1.0              pyhff2d567_0    conda-forge
tornado                   6.4.1           py312h827c3e9_0
traitlets                 5.14.3             pyhd8ed1ab_0    conda-forge
types-python-dateutil     2.9.0.20241003     pyhff2d567_0    conda-forge
typing-extensions         4.12.2               hd8ed1ab_0    conda-forge
typing_extensions         4.12.2             pyha770c72_0    conda-forge
typing_utils              0.1.0              pyhd8ed1ab_0    conda-forge
tzdata                    2024b                h04d1e81_0
ucrt                      10.0.22621.0         h57928b3_1    conda-forge
unicodedata2              15.1.0          py312h2bbff1b_0
uri-template              1.3.0              pyhd8ed1ab_0    conda-forge
urllib3                   2.2.3              pyhd8ed1ab_0    conda-forge
vc                        14.40                h2eaa2aa_1
vc14_runtime              14.40.33810         hcc2c482_22    conda-forge
vs2015_runtime            14.40.33810         h3bf8584_22    conda-forge
wcwidth                   0.2.13             pyhd8ed1ab_0    conda-forge
webcolors                 24.8.0             pyhd8ed1ab_0    conda-forge
webencodings              0.5.1              pyhd8ed1ab_2    conda-forge
websocket-client          1.8.0              pyhd8ed1ab_0    conda-forge
wheel                     0.44.0          py312haa95532_0
win_inet_pton             1.1.0              pyh7428d3b_7    conda-forge
winpty                    0.4.3                        4    conda-forge
```

```
xz                        5.4.6               h8cc25b3_1
yaml                      0.2.5               h8ffe710_2    conda-forge
zeromq                    4.3.5               ha9f60a1_6    conda-forge
zipp                      3.20.2            pyhd8ed1ab_0    conda-forge
zlib                      1.2.13              h8cc25b3_1
zstandard                 0.23.0          py312h4fc1ca9_1
zstd                      1.5.6               h8880b57_0
```

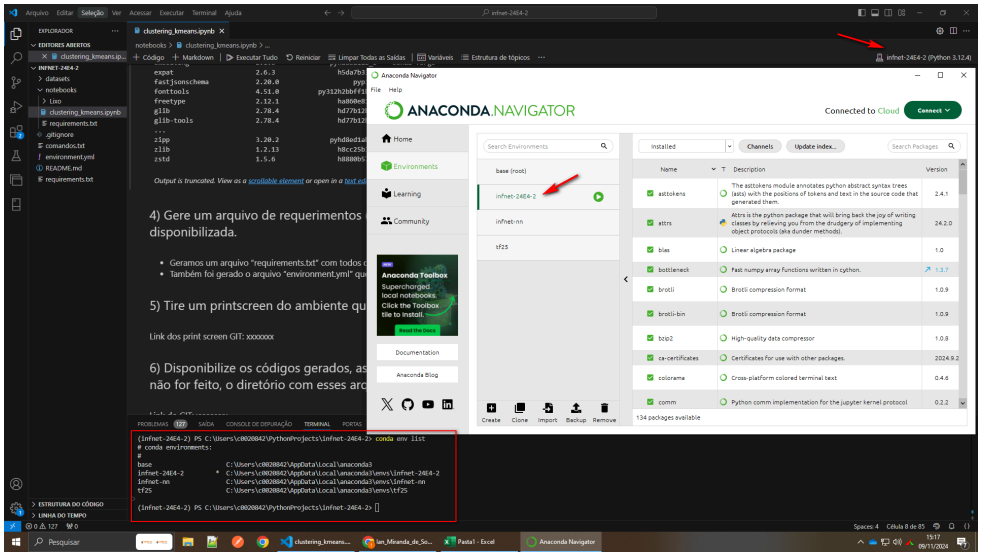
4) Gere um arquivo de requerimentos (requirements.txt) com os pacotes necessários.
É necessário se certificar que a versão do pacote está disponibilizada.

> Resposta:
>
> - Foi gerado um arquivo *"requirements.txt"* com todos os pacotes necessários e suas respectivas versões.
> https://github.com/ianmsouza/cluster_analysis_country_data/blob/main/requirements.txt
> - Também foi gerado o arquivo *"environment.yml"* que é específico para o ambiente virtual Anaconda, utilizado neste notebook.
> https://github.com/ianmsouza/cluster_analysis_country_data/blob/main/environment.yml

> Comandos utilizados para geração dos arquivos:
>
> - *pip freeze > requirements.txt*
> - *conda env export > environment.yml*

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

> Resposta:
> Print Screen da máquina com o ambiente Anaconda iniciado no Visual Studio Code.



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).

> Link do GitHub e README:
> https://github.com/ianmsouza/cluster_analysis_country_data

Parte 2 - Escolha de base de dados

Para as questões a seguir, usaremos uma base de dados e faremos a análise exploratória dos dados, antes da clusterização.

1) Baixe os dados disponibilizados na plataforma Kaggle sobre dados sócio-econômicos e de saúde que determinam o índice de desenvolvimento de um país.

Esses dados estão disponibilizados através do link: https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data

> Resposta:
> Importação da base de dados

In [353…
```
# Carregando os dados
df = pd.read_csv(r'C:\Users\c0020842\PythonProjects\infnet-24E4-2\datasets\Country-data.csv

colunas_traduzidas = ['país (Country)', 'mortalidade_infantil (child_mort)', 'exportações (
                      'renda (income)', 'inflação (inflation)', 'expectativa_de_vida (life_

# Atualiza o DataFrame com os novos nomes de coluna:
df.columns = colunas_traduzidas
```

> Quantidade de linhas e colunas no dataset

In [354…
```
df.shape
```

Out[354… `(167, 10)`

> Amostra dos dados presentes no dataset

In [355…
```
df
```

Out[355…

	país (Country)	mortalidade_infantil (child_mort)	exportações (exports)	saúde (health)	importações (imports)	renda (income)	inflação (inflation)	ex
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	
...	...	...	...	...	...	...	...	
162	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	
163	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	
164	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	
165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	
166	Zambia	83.1	37.0	5.89	30.9	3280	14.00	

167 rows × 10 columns

> Informações detalhadas de cada coluna

In [356…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   país (Country)                    167 non-null    object 
 1   mortalidade_infantil (child_mort)  167 non-null    float64
 2   exportações (exports)             167 non-null    float64
 3   saúde (health)                    167 non-null    float64
 4   importações (imports)             167 non-null    float64
 5   renda (income)                    167 non-null    int64  
 6   inflação (inflation)              167 non-null    float64
 7   expectativa_de_vida (life_expec)  167 non-null    float64
 8   taxa_fertilidade (total_fer)      167 non-null    float64
 9   PIB (gdpp)                        167 non-null    int64  
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

> Estatísticas descritivas de cada coluna

In [357…

```
df.describe()
```

Out[357...

	mortalidade_infantil (child_mort)	exportações (exports)	saúde (health)	importações (imports)	renda (income)	inflação (inflation)	expe
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	
mean	38.270060	41.108976	6.815689	46.890215	17144.688623	7.781832	
std	40.328931	27.412010	2.746837	24.209589	19278.067698	10.570704	
min	2.600000	0.109000	1.810000	0.065900	609.000000	-4.210000	
25%	8.250000	23.800000	4.920000	30.200000	3355.000000	1.810000	
50%	19.300000	35.000000	6.320000	43.300000	9960.000000	5.390000	
75%	62.100000	51.350000	8.600000	58.750000	22800.000000	10.750000	
max	208.000000	200.000000	17.900000	174.000000	125000.000000	104.000000	

A tabela gerada pelo método df.describe() apresenta estatísticas descritivas para um conjunto de dados com nove variáveis:

1. child_mort (mortalidade infantil)
2. exports (exportações como % do PIB)
3. health (investimento em saúde como % do PIB)
4. imports (importações como % do PIB)
5. income (renda per capita)
6. inflation (taxa de inflação)
7. life_expec (expectativa de vida)
8. total_fer (taxa de fertilidade total)
9. gdpp (PIB per capita).

No conjunto de dados analisados, foram calculadas as principais métricas estatísticas, que ajudam a descrever e interpretar cada variável.

Essas métricas são:

- Count (Contagem): Cada variável possui 167 entradas válidas, representando a quantidade de registros observados para cada indicador.
- Mean (Média): A média da mortalidade infantil é de 38,27, enquanto a expectativa de vida média é de 70,56 anos. Em termos econômicos, a renda média per capita é de 17.144,69 e o PIB per capita médio é de 12.964,16.
- Std (Desvio Padrão): A medida de dispersão, ou desvio-padrão, é mais elevada nas variáveis de renda e - PIB per capita, com valores de 19.278,07 e 18.328,70, respectivamente. Isso indica uma maior variação nos níveis de renda e PIB entre os países da amostra.
- Mínimo e Máximo: A mortalidade infantil varia de 2,6 a 208, enquanto a expectativa de vida varia de 32,1 a 82,8 anos. Já para a renda, observa-se uma amplitude significativa, de 609 até 125.000. Esses valores extremos refletem as grandes desigualdades entre os países analisados.
- Quartis: Os quartis fornecem insights sobre a distribuição dos dados. Por exemplo, para a variável de mortalidade infantil, o primeiro quartil é 8,25 e o terceiro quartil é 62,1, o que revela uma ampla variação e possíveis desigualdades. Os quartis mostram também como a maioria dos países está posicionada entre valores intermediários, enquanto poucos países apresentam valores extremos.

> Pode-se observar que os valores médios estão geralmente acima da mediana (50%), o que indica que a distribuição dos dados pode ser positivamente assimétrica.

2) Quantos países existem no dataset?

In [358…
```
print("Resposta:")
print(f"Total de países: {df['país (Country)'].nunique()}")
```

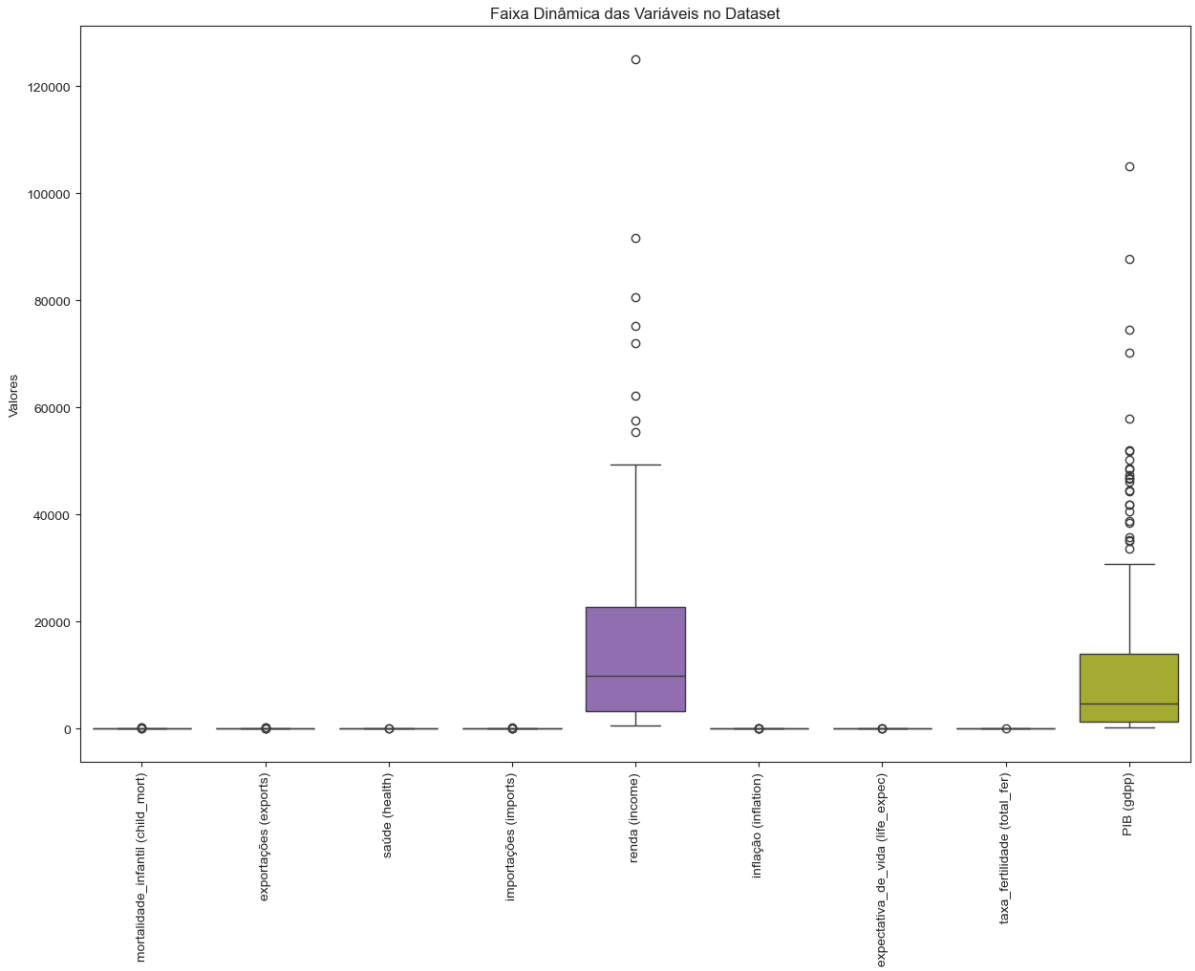
```
Resposta:
Total de países: 167
```

> O comando *nunique* foi utilizado para obter a quantidade real de países distintos no conjunto de dados, garantindo que apenas valores únicos fossem considerados em vez de contar todas as linhas, que poderiam incluir registros duplicados.

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?

> **Faixa dinâmica das variáveis no dataset**

In [359…
```
# Cria gráfico boxplot para visualizar a faixa dinâmica das variáveis
plt.figure(figsize=(15, 10))
sns.boxplot(data=df)
plt.ylabel("Valores")
plt.title("Faixa Dinâmica das Variáveis no Dataset")
plt.xticks(rotation=90)
plt.show()
```

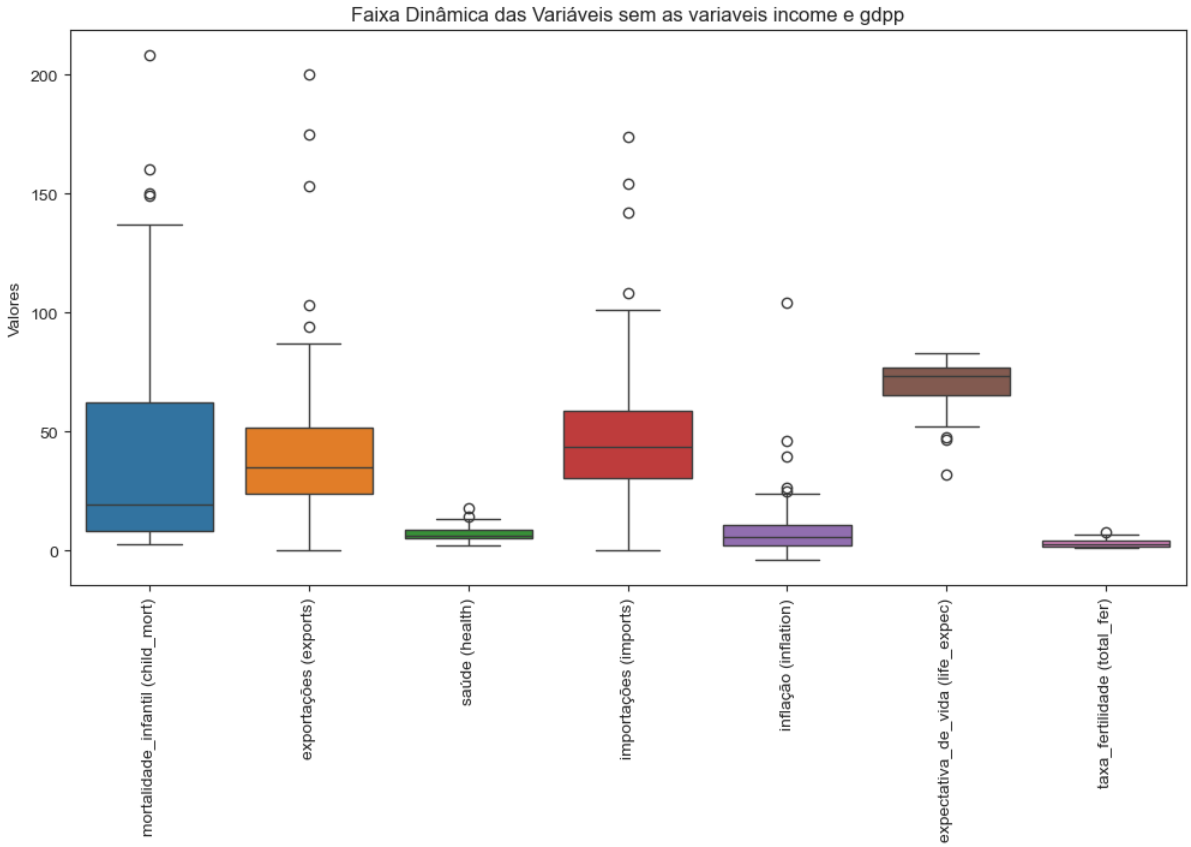


> Na faixa dinâmica das variáveis, as colunas "renda (income)" e "PIB (gdpp)" possuem valores significativamente maiores em comparação com as outras variáveis, e elas também contêm outliers (valores extremos).
> Além disso, devido ao fato de que essas colunas terem os valores muito altos, pode ser difícil visualizar com detalhes as outras colunas do gráfico.
>
> Para facilitar a visualização dos valores das demais variáveis, foi criado o gráfico "Faixa Dinâmica das Variáveis" sem incluir as variáveis income e gdpp. Esse gráfico permite identificar tanto a faixa dinâmica quanto os outliers das outras variáveis.
> No gráfico anterior, conseguimos visualizar apenas os outliers das variáveis "renda (income)" e "PIB (gdpp)".

> **Faixa dinâmica das variáveis sem as variáveis income e gdpp**

In [360…

```
df_copy = df.drop(columns=['renda (income)', 'PIB (gdpp)'])
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_copy)
plt.ylabel("Valores")
plt.title("Faixa Dinâmica das Variáveis sem as variaveis income e gdpp")
plt.xticks(rotation=90)
plt.show()
```

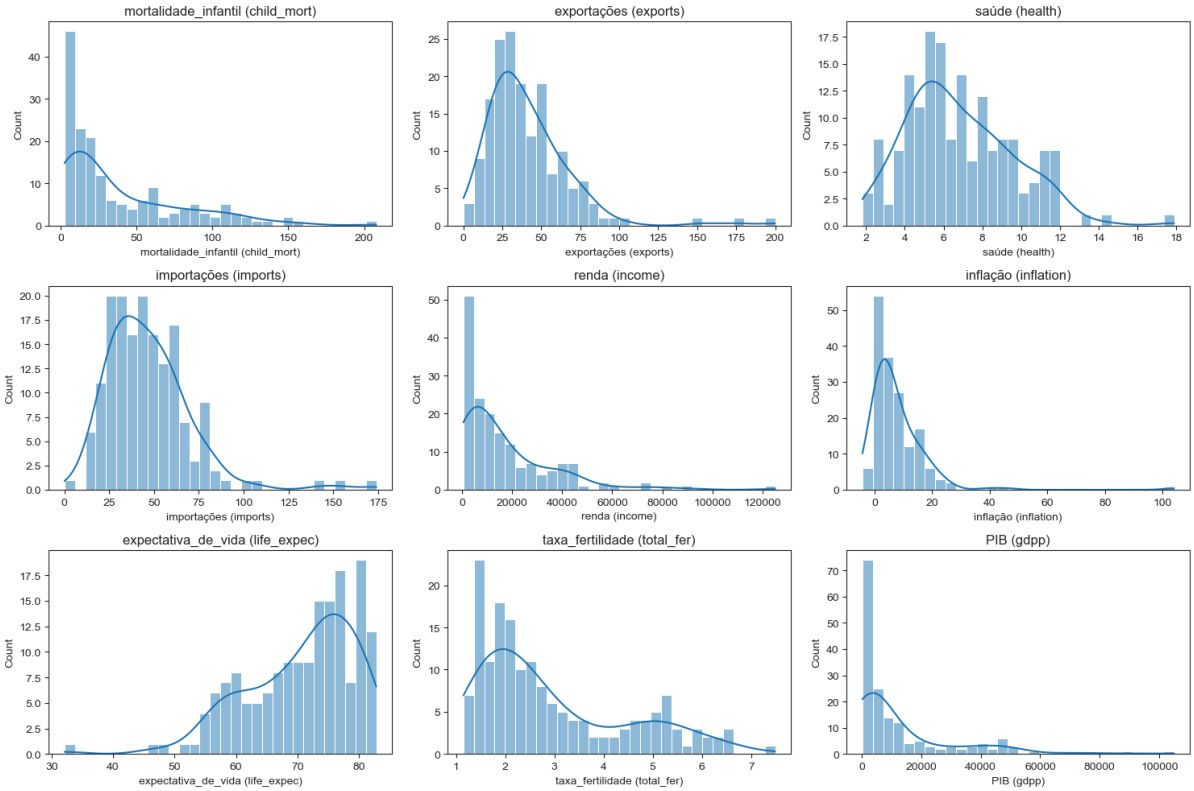


> **Gráfico de histograma de todas as colunas**

> A análise do histograma revela que a maioria dos países presentes no conjunto de dados exibe características indicativas de nações subdesenvolvidas ou em desenvolvimento.

In [361…

```
numeric_features = ['mortalidade_infantil (child_mort)', 'exportações (exports)', 'saúde (h
                    'renda (income)', 'inflação (inflation)', 'expectativa_de_vida (life_

plt.figure(figsize=(15, 10))
for i, feature in enumerate(numeric_features, 1):
    plt.subplot(3, 3, i)
    sns.histplot(df[feature], kde=True, bins=30)
    plt.title(feature)
plt.tight_layout()
plt.show()
```

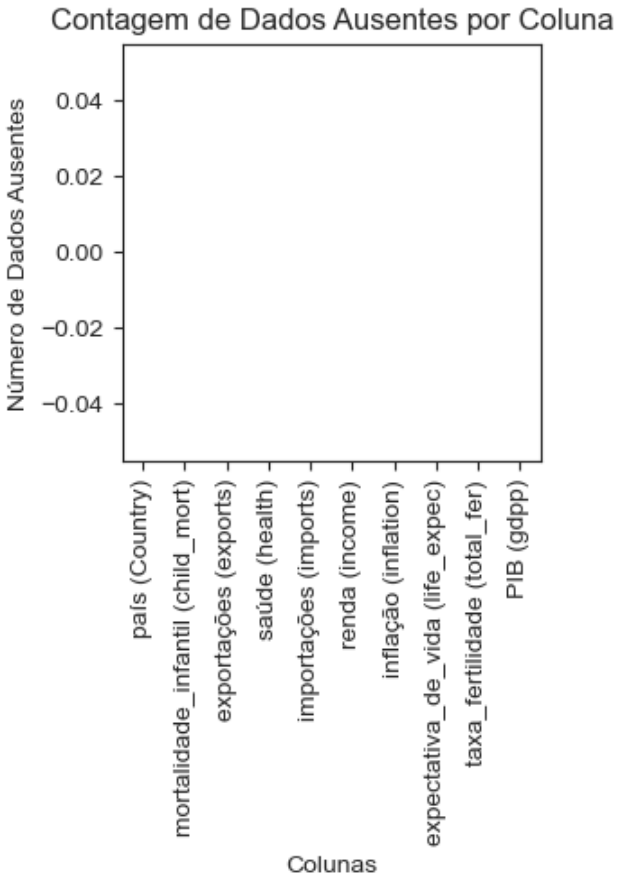


> **Antes da etapa de clusterização**, é importante realizar as etapas de pré-processamento e preparação de dados para garantir que os dados estejam em um formato adequado para a análise de clusters.
>
> **Etapas a serem realizadas antes da clusterização:**
>
> - **Limpeza de dados:** Verificar a presença de valores ausentes e, se encontrados, removê-los ou tratá-los adequadamente para evitar impactos negativos na análise.
> - **Detecção e tratamento de outliers:** Identificar outliers e decidir se devem ser transformados, removidos ou mantidos, considerando seu impacto potencial na qualidade da clusterização.
> - **Normalização ou padronização:** Aplicar técnicas de normalização ou padronização para garantir que todas as variáveis estejam na mesma escala, evitando que alguma variável domine o processo de clusterização.
> - **Redução de Dimensionalidade:** Utilizar técnicas como PCA (Análise de Componentes Principais) para reduzir o número de variáveis, mantendo a maior parte da variabilidade dos dados.
> - **Seleção de Características:** Escolher as variáveis mais relevantes para a análise de clusterização.
> - **Transformação de Dados:** Aplicar transformações logarítmicas ou de raiz quadrada, se necessário, para lidar com distribuições assimétricas.

4) Realize o pré-processamento adequado dos dados.

> **Limpeza de dados**
>
> - Verificação de dados ausentes

In [362…
```
missing_data = df.isnull().sum()
missing_data.plot(kind='bar', figsize=(3,3))
plt.title("Contagem de Dados Ausentes por Coluna")
plt.xlabel("Colunas")
plt.ylabel("Número de Dados Ausentes")
plt.show();
print("Dados Ausentes:")
```




```
Dados Ausentes:
```

> Verificamos a presença de dados ausentes em todas as variáveis do dataset e confirmamos que não há nenhum valor ausente.

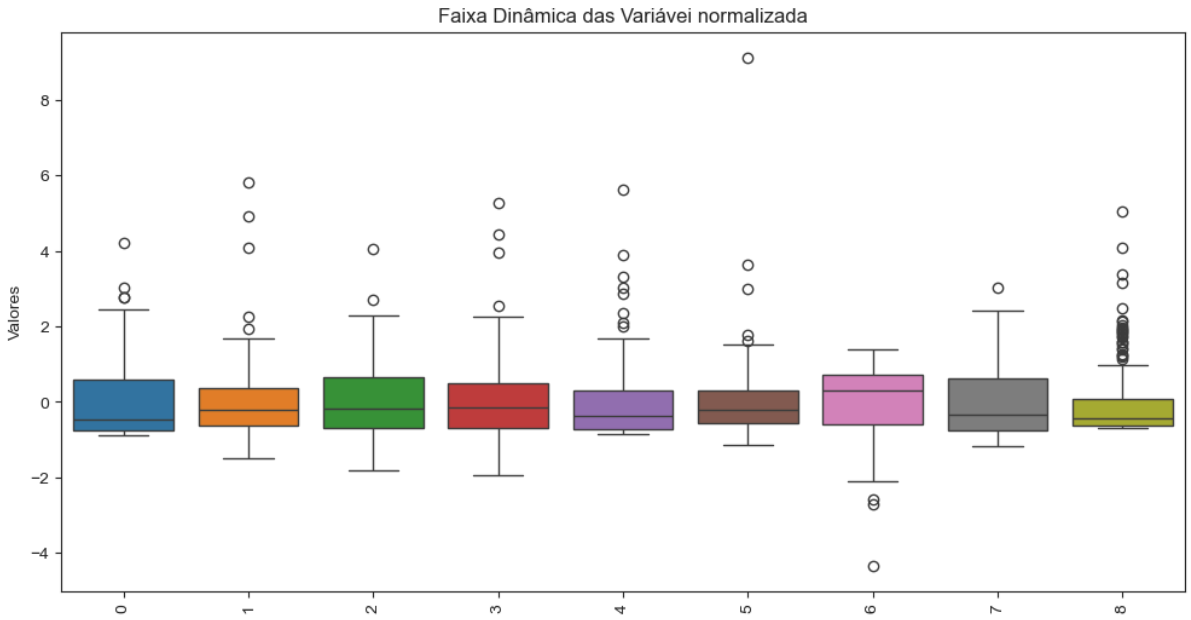
> **Normalização dos dados**

In [363…
```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[numeric_features])

df_scaled
```

Out[363…
```
array([[ 1.29153238, -1.13827979,  0.27908825, ..., -1.61909203,
         1.90288227, -0.67917961],
       [-0.5389489 , -0.47965843, -0.09701618, ...,  0.64786643,
        -0.85997281, -0.48562324],
       [-0.27283273, -0.09912164, -0.96607302, ...,  0.67042323,
        -0.0384044 , -0.46537561],
       ...,
       [-0.37231541,  1.13030491,  0.0088773 , ...,  0.28695762,
        -0.66120626, -0.63775406],
       [ 0.44841668, -0.40647827, -0.59727159, ..., -0.34463279,
         1.14094382, -0.63775406],
       [ 1.11495062, -0.15034774, -0.33801514, ..., -2.09278484,
         1.6246091 , -0.62954556]])
```

> O gráfico da "*Faixa Dinâmica das Variávei normalizada*" podemos ver as variaveis na mesma faixa, sobre os outliers neste momento não estaremos aplicado nenhum tratamento neles.

In [364…
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_scaled)
plt.ylabel("Valores")
plt.title('Faixa Dinâmica das Variávei normalizada')
plt.xticks(rotation=90)
plt.show()
```



Parte 3 - Clusterização

Para os dados pré-processados da etapa anterior você irá:

1) Realizar o agrupamento dos países em 3 grupos distintos. Para tal, use:
a) K-Médias
b) Clusterização Hierárquica

> Resposta:

In [365…
```
# Convertendo dados normalizados para DataFrame mantendo informação dos países
df_scaled = pd.DataFrame(df_scaled, columns=numeric_features)
df_scaled['país (Country)'] = df['país (Country)'].values

# 1) Agrupamento dos países em 3 grupos distintos
#    a) K-Médias (K-Means)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(df_scaled[numeric_features])
df_scaled['Cluster_KMeans'] = kmeans_labels

#    b) Clusterização Hierárquica
```

```
hierarchical = AgglomerativeClustering(n_clusters=3)
hierarchical_labels = hierarchical.fit_predict(df_scaled[numeric_features])
df_scaled['Cluster_Hierarchical'] = hierarchical_labels
```

```
c:\Users\c0020842\AppData\Local\anaconda3\envs\infnet-24E4-2\Lib\site-packages\sklearn\clust
er\_kmeans.py:1429: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than a
vailable threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
```

2) Para os resultados, do K-Médias:
a) Interprete cada um dos clusters obtidos citando:
i) Qual a distribuição das dimensões em cada grupo

> Resposta:

In [366…

```
# 2) Análise dos resultados do K-Médias
#   i) Distribuição das dimensões em cada grupo

for i in range(3):
    cluster_description = df_scaled[df_scaled['Cluster_KMeans'] == i][numeric_features].des
    print(f"Cluster {i + 1} - K-Médias")
    display(cluster_description)
```

```
Cluster 1 - K-Médias
```

	mortalidade_infantil (child_mort)	exportações (exports)	saúde (health)	importações (imports)	renda (income)	inflação (inflation)	expectativa_d (life_
count	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	86.
mean	-0.393282	-0.030584	-0.206179	0.019562	-0.250930	-0.005783	0.
std	0.350117	0.688175	0.791044	0.825407	0.420602	0.741832	0.
min	-0.839884	-1.500192	-1.769403	-1.939940	-0.799401	-1.137852	-1.
25%	-0.660815	-0.519908	-0.709555	-0.594118	-0.543291	-0.507575	-0.
50%	-0.486720	-0.126564	-0.301500	0.072908	-0.348314	-0.175238	0.
75%	-0.226200	0.374720	0.270872	0.554534	-0.036144	0.217589	0.
max	0.649869	1.927969	2.696381	2.531789	1.470079	3.616865	1.

```
Cluster 2 - K-Médias
```

	mortalidade_infantil (child_mort)	exportações (exports)	saúde (health)	importações (imports)	renda (income)	inflação (inflation)	expectativa_d (life_
count	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.
mean	-0.827449	0.645080	0.727411	0.190639	1.484243	-0.484921	1.
std	0.054440	1.534250	1.160452	1.526453	1.084898	0.386917	0.
min	-0.887138	-1.050464	-1.827827	-1.379219	0.419105	-1.043915	0.
25%	-0.863511	-0.417455	0.335686	-0.766053	0.796311	-0.695543	1.
50%	-0.847345	0.338130	0.960093	-0.324821	1.217742	-0.625470	1.
75%	-0.808796	0.981200	1.445742	0.663288	1.533814	-0.392289	1.
max	-0.683199	5.813835	4.047436	5.266181	5.611542	0.846206	1.

Cluster 3 - K-Médias

	mortalidade_infantil (child_mort)	exportações (exports)	saúde (health)	importações (imports)	renda (income)	inflação (inflation)	expectativa_d (life_
count	45.000000	45.000000	45.000000	45.000000	45.000000	45.000000	45.
mean	1.413564	-0.457615	-0.187898	-0.189897	-0.707839	0.398989	-1.
std	0.806360	0.672062	0.981480	0.747324	0.282000	1.502663	0.
min	0.383753	-1.423682	-1.685418	-1.230071	-0.860326	-0.654410	-4.
25%	0.709559	-0.889467	-0.841922	-0.716337	-0.819692	-0.371177	-1.
50%	1.294019	-0.633337	-0.553454	-0.273034	-0.795759	0.095661	-1.
75%	1.808842	-0.157666	0.304649	0.099838	-0.719277	0.836717	-0.
max	4.221297	1.635248	2.294716	2.241778	0.861347	9.129718	0.

> Observação: Nas respostas das próximas questões, apresentarei uma análise detalhada da distribuição das dimensões em cada grupo, acompanhada de interpretações claras e fundamentadas para facilitar o entendimento dos resultados.

ii) O país, de acordo com o algoritmo, melhor representa o seu agrupamento. Justifique.

> Resposta:

In [367…

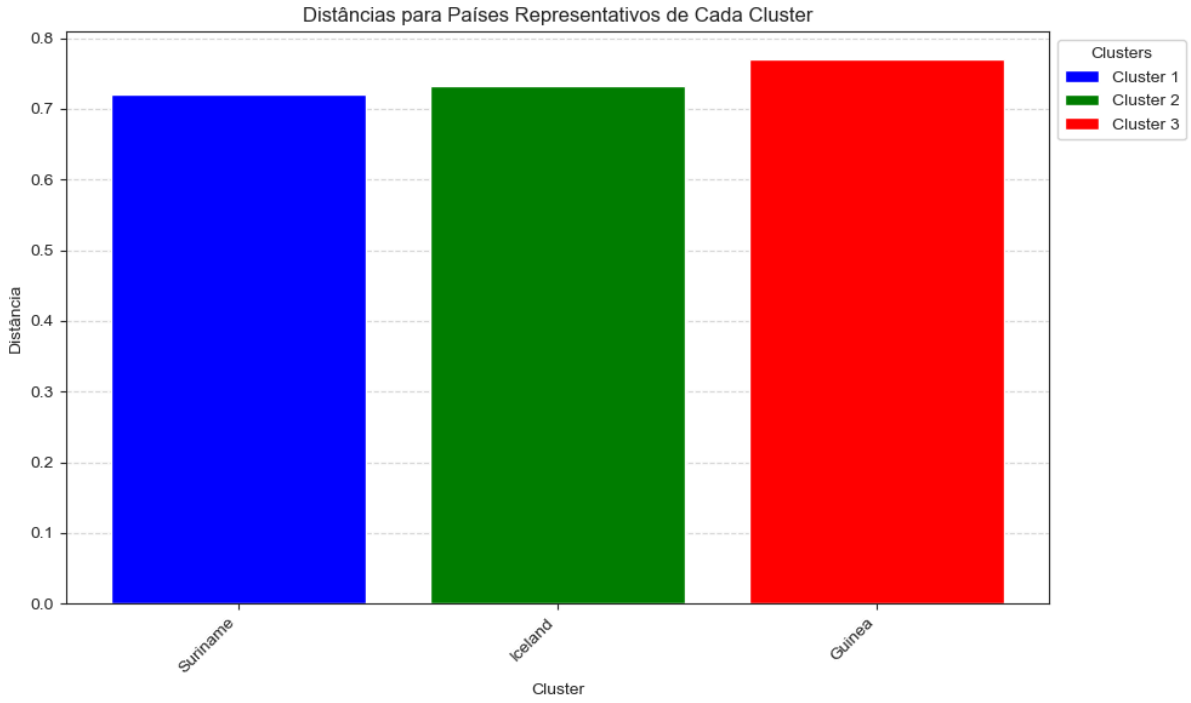
```
# Definindo os centróides a partir do modelo KMeans
centroids = kmeans.cluster_centers_

# Criar uma lista para armazenar as informações de cada cluster
cluster_representatives = []
distancias = []  # Inicializando a lista para armazenar as distâncias

# Calcular as distâncias e identificar o país mais representativo de cada cluster
for i in range(3):
    # Filtrar os dados para o cluster atual
    cluster_data = df_scaled[df_scaled['Cluster_KMeans'] == i][numeric_features]
    centroid = centroids[i]

    # Calcula distância euclidiana entre cada ponto e o centróide
    distances = np.linalg.norm(cluster_data.values - centroid, axis=1)
    min_distance_idx = distances.argmin()

    # Acessar o país representativo
    representative_country = df.iloc[cluster_data.index[min_distance_idx]]

    # Armazenar as informações do país representativo
    cluster_representatives.append({
        "Cluster": f"Cluster {i + 1}",
        "País Representante": representative_country['país (Country)'],
        **representative_country[numeric_features]
    })

    # Adicionar as distâncias à lista
    distancias.append((i, representative_country['país (Country)'], distances[min_distance_

# Criar um DataFrame com os resultados
representatives_df = pd.DataFrame(cluster_representatives)
```

```
# Exibir o DataFrame
print("Resposta:")
display(representatives_df)

# Exibir os países mais representativos para cada cluster
for idx, pais, distancia in distancias:
    print(f"O país mais representativo do cluster {idx + 1} é {pais} (distância: {distancia

# Criação do DataFrame a partir das distâncias
distancias_df = pd.DataFrame(distancias, columns=['Cluster', 'País', 'Distância'])

# Definir as cores para cada cluster
cores = ['blue', 'green', 'red']

# Configuração do gráfico
plt.figure(figsize=(10, 6))
bars = plt.bar(distancias_df['Cluster'], distancias_df['Distância'], color=cores)

# Legenda dos clusters
legendas = [f'Cluster {cluster + 1}' for cluster in distancias_df['Cluster']]
plt.legend(bars, legendas, title="Clusters", bbox_to_anchor=(1, 1), loc='upper left')

# Configurações do gráfico
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xlabel('Cluster')
plt.ylabel('Distância')
plt.title('Distâncias para Países Representativos de Cada Cluster')

# Ajustar as ticks do eixo x para mostrar os países
plt.xticks(distancias_df['Cluster'], distancias_df['País'], rotation=45, ha="right")

# Ajustar layout e mostrar o gráfico
plt.tight_layout()
plt.show()
```

Resposta:

	Cluster	País Representante	mortalidade_infantil (child_mort)	exportações (exports)	saúde (health)	importações (imports)	renda (income)	infla (inflati
0	Cluster 1	Suriname	24.1	52.5	7.01	38.4	14200	
1	Cluster 2	Iceland	2.6	53.4	9.40	43.3	38800	
2	Cluster 3	Guinea	109.0	30.3	4.93	43.2	1190	1

```
O país mais representativo do cluster 1 é Suriname (distância: 0.72)
O país mais representativo do cluster 2 é Iceland (distância: 0.73)
O país mais representativo do cluster 3 é Guinea (distância: 0.77)
```



> Os resultados indicando que os países mais próximos dos centros dos clusters são os seguintes:
>
> - Cluster 1: Suriname
> - Cluster 2: Iceland
> - Cluster 3: Guinea
>
> A proximidade de um país ao centro de um cluster em uma análise de agrupamento (clustering) é um indicativo de que esse país possui características semelhantes às médias das características dos países no cluster.

In [368…

```
kmeans.fit(df_scaled[numeric_features])
clusters = kmeans.predict(df_scaled[numeric_features])
results = df_scaled.copy()
results['cluster'] = clusters
```

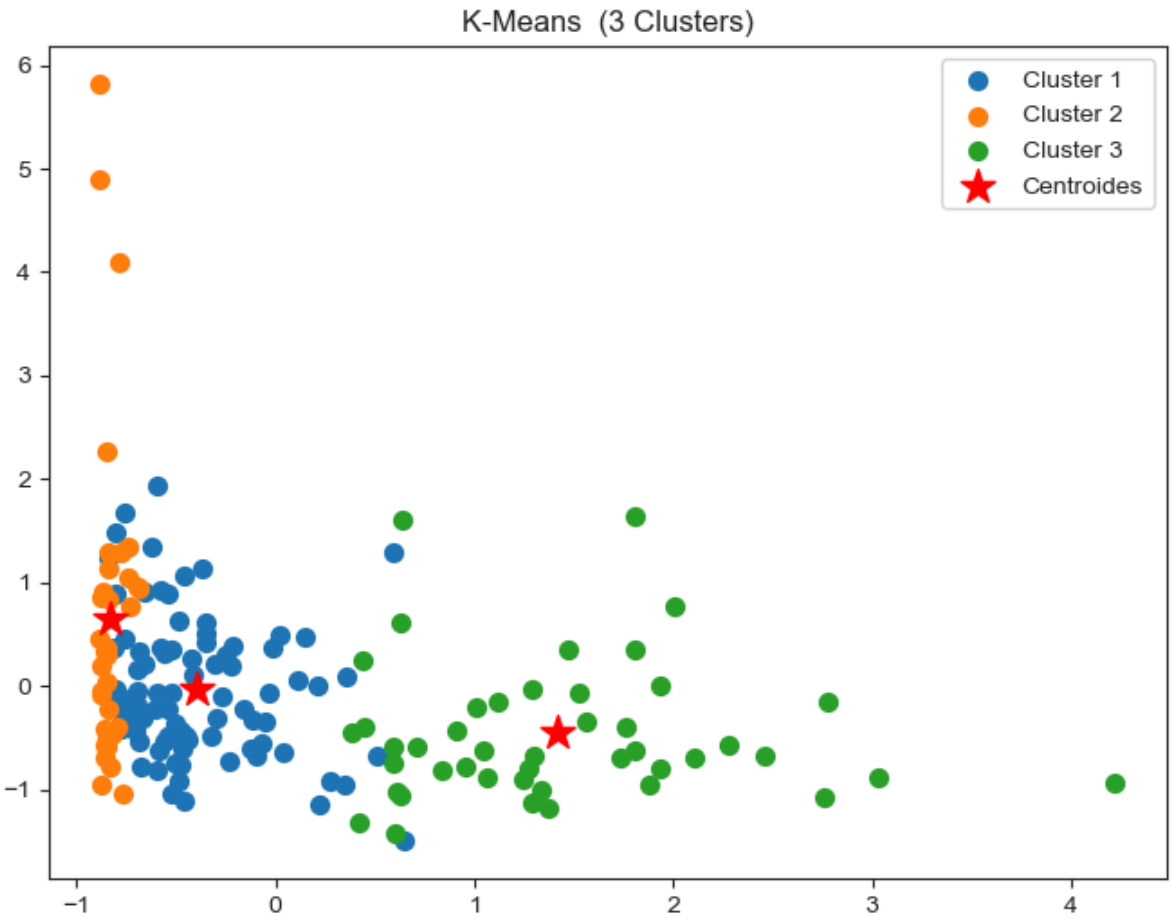
```
c:\Users\c0020842\AppData\Local\anaconda3\envs\infnet-24E4-2\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
```

> No **gráfico K-Means** a seguir, podemos visualizar a distribuição dos clusters formados pelo algoritmo K-Means, que procura particionar o conjunto de dados em grupos distintos.
>
> Esses clusters são representados por diferentes cores ou marcadores, destacando a segmentação identificada pelo algoritmo.

In [369…

```
num_clusters = 3
data_array = df_scaled.values
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
cluster_names = [f'Cluster {i + 1}' for i in range(num_clusters)]
```

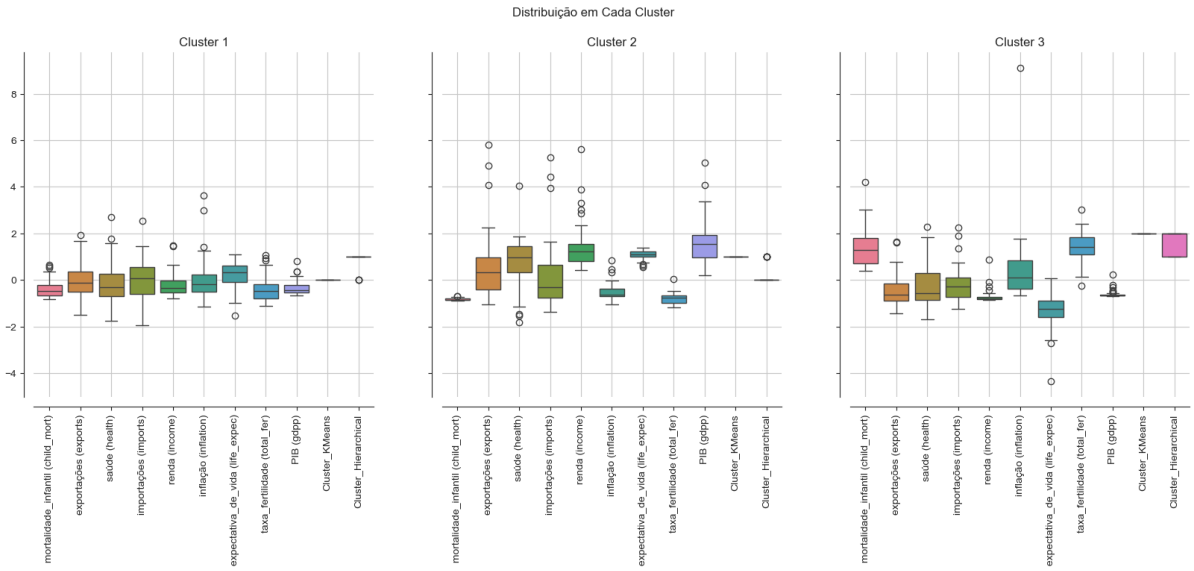
```
plt.figure(figsize=(8, 6))

for cluster_id in range(num_clusters):
    plt.scatter(data_array[labels == cluster_id, 0], data_array[labels == cluster_id, 1], s

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='*', s=200, label='Centroides
plt.title(f'K-Means  ({num_clusters} Clusters)')
plt.legend()
plt.show()
```



> Já no gráfico **"Distribuição em Cada Cluster"**, exploramos a distribuição das variáveis dentro de cada cluster. Cada cluster é analisado separadamente, permitindo uma compreensão mais aprofundada da variação das variáveis dentro de cada grupo

In [370…

```
num_clusters = 3
fig, axes = plt.subplots(1, num_clusters, figsize=(20, 6), sharey=True)
sns.set_style('ticks')
for idx in range(num_clusters):
    sns.boxplot(data=results[results['cluster'] == idx].drop(columns='cluster'), ax=axes[id
    axes[idx].set_title(f'Cluster {idx + 1}')
    axes[idx].grid(True)
    axes[idx].tick_params(axis='x', rotation=90)  # Gira as labels do eixo x
    sns.despine(offset=10)

fig.suptitle('Distribuição em Cada Cluster')
plt.show()
```



> Por fim, o gráfico **"Distribuição dos Clusters com K-Means"** nos proporciona uma visualização geoespacial da distribuição dos clusters no mapa.
>
> Cada país é colorido de acordo com o cluster ao qual foi atribuído pelo algoritmo K-Means. Essa representação espacial é valiosa para identificar padrões de agrupamento geográfico e entender como os clusters estão distribuídos globalmente.

In [371…

```
data = results.copy()
data.insert(0, column='country', value=results.index)

# Converta a coluna 'cluster' para string
data['cluster'] = data['cluster'].astype(str)

data.loc[data['cluster'] == '0', 'cluster'] = 'Cluster 1'
data.loc[data['cluster'] == '1', 'cluster'] = 'Cluster 2'
data.loc[data['cluster'] == '2', 'cluster'] = 'Cluster 3'

fig = px.choropleth(data,
                    locationmode='country names',
                    locations='país (Country)',  # Verifique se o nome da coluna está corre
                    color='cluster',
                    title='Distribuição dos clusters com K-Means',
                    color_discrete_map={'Cluster 1': '#ff5733',
                                        'Cluster 2': '#33ff57',
                                        'Cluster 3': '#5733ff'})

fig.update_geos(fitbounds="locations", visible=True)
fig.update_layout(legend_title_text='Legenda',
                  title_pad_l=260,
                  title_y=0.90,
                  legend=dict(
                      orientation="h",
                      yanchor="bottom",
                      y=0.00,
                      xanchor="right",
                      x=0.5
                  )
                  )

fig.show()
```

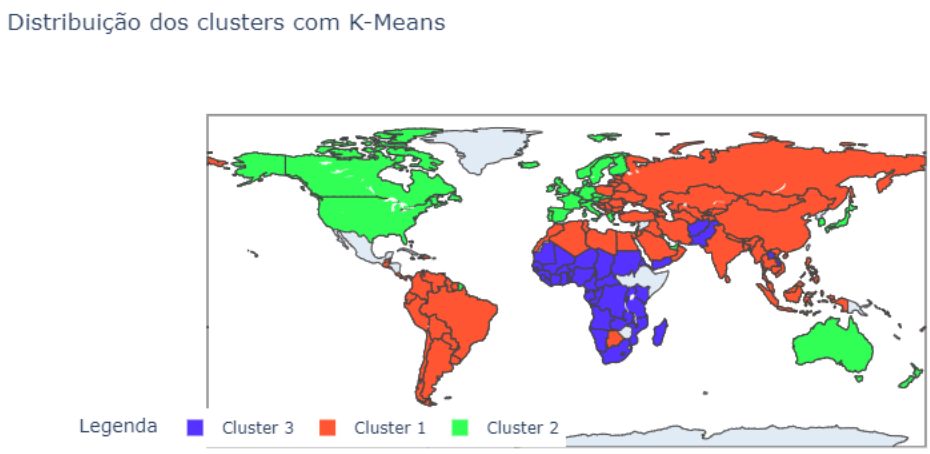
> **Distribuição Global dos Clusters:**

A imagem divide o mundo em três clusters:

- **Cluster 1 (laranja):** Contém grande parte dos países asiáticos, leste europeu, partes da América do Sul e algumas regiões da África, indicando características compartilhadas nesses locais.
- **Cluster 2 (verde):** Agrupa países da América do Norte, partes da Europa Ocidental e Oceania, sugerindo perfis socioeconômicos ou de desenvolvimento semelhantes.
- **Cluster 3 (azul):** Inclui grande parte dos países da África e algumas regiões do Oriente Médio e da Ásia Central, sugerindo similaridades específicas que diferenciam esses países dos outros clusters.

Análise Geográfica e Socioeconômica:

- Os clusters refletem um agrupamento que parece correlacionado com fatores regionais ou econômicos.
- Países de alta renda e com economias desenvolvidas tendem a se agrupar no Cluster 2 (verde), enquanto países em desenvolvimento aparecem no Cluster 1 (laranja) e países com características específicas (talvez níveis socioeconômicos mais baixos) no Cluster 3 (azul).



O **Gráfico de Silhueta** ilustra a qualidade da clusterização, mostrando como as amostras se ajustam a cada cluster.

Cluster 3:

- Valor Mínimo: aproximadamente -0,1
- Valor Máximo: aproximadamente 0,4

O cluster 2 possui amostras com coeficientes de silhueta variando de -0,1 a 0,4. Isso indica que algumas amostras podem não estar bem ajustadas, com possível sobreposição com outros clusters. No entanto, o valor máximo de 0,4 sugere que algumas amostras estão bem representadas dentro do cluster.

Cluster 2:

- Valor Mínimo: 0

- Valor Máximo: aproximadamente 0,5

O cluster 2 exibe coeficientes de silhueta de 0 a 0,5, indicando que suas amostras estão relativamente bem ajustadas e separadas dos outros clusters. O valor máximo de 0,5 revela que algumas amostras possuem uma boa distância dos pontos nos clusters vizinhos.

Cluster 1:

- Valor Mínimo: aproximadamente -0,1
- Valor Máximo: aproximadamente 0,4

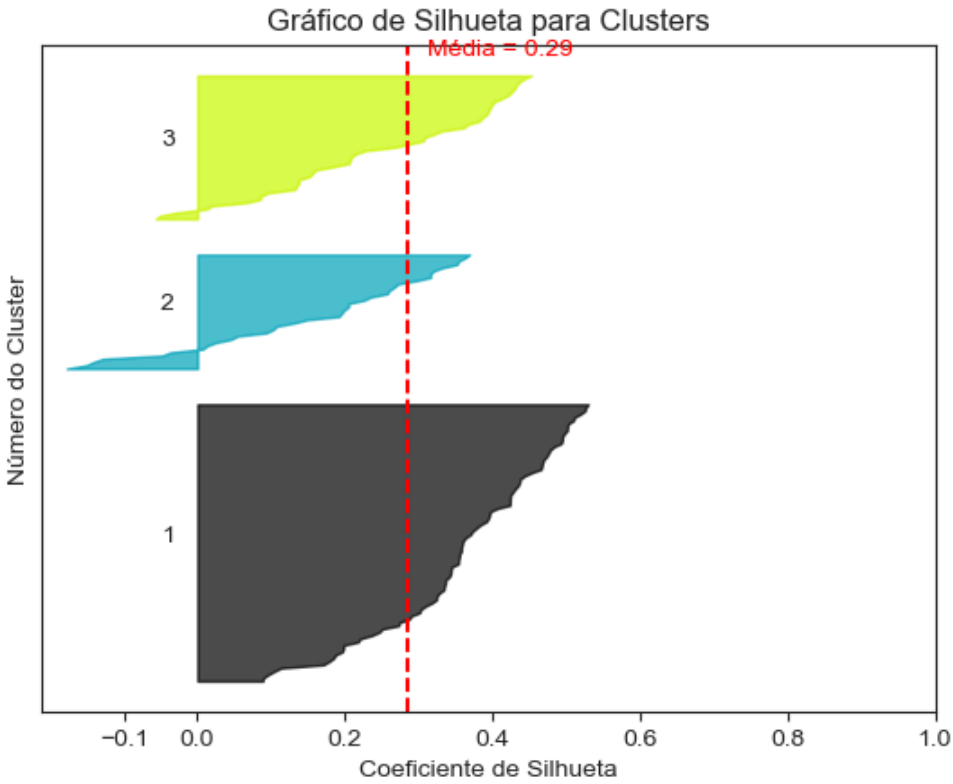
O cluster 1 apresenta coeficientes de silhueta variando de -0,1 a 0,4. Tal como o cluster 3, algumas amostras podem não estar bem ajustadas, mas o valor máximo de 0,4 sugere que há amostras razoavelmente bem agrupadas neste cluster.

Média Geral:

A linha vermelha pontilhada no gráfico indica a média do coeficiente de silhueta (aproximadamente 0,29), o que sugere um agrupamento geral moderado.

In [372…

```
silhouette_avg = silhouette_score(df_scaled[numeric_features], clusters)
sample_silhouette_values = silhouette_samples(df_scaled[numeric_features], clusters)

fig, ax = plt.subplots()
y_lower = 10

for i in range(num_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[clusters == i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i
    color = plt.cm.nipy_spectral(float(i) / num_clusters)
    ax.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, facecol
    ax.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i + 1))
    y_lower = y_upper + 10

ax.set_title("Gráfico de Silhueta para Clusters")
ax.set_xlabel("Coeficiente de Silhueta")
ax.set_ylabel("Número do Cluster")

ax.axvline(x=silhouette_avg, color="red", linestyle="--")
ax.text(silhouette_avg + 0.025, 0.5 * (y_lower + y_upper), 'Média = {:.2f}'.format(silhouet
ax.set_yticks([])
ax.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

plt.show()
```

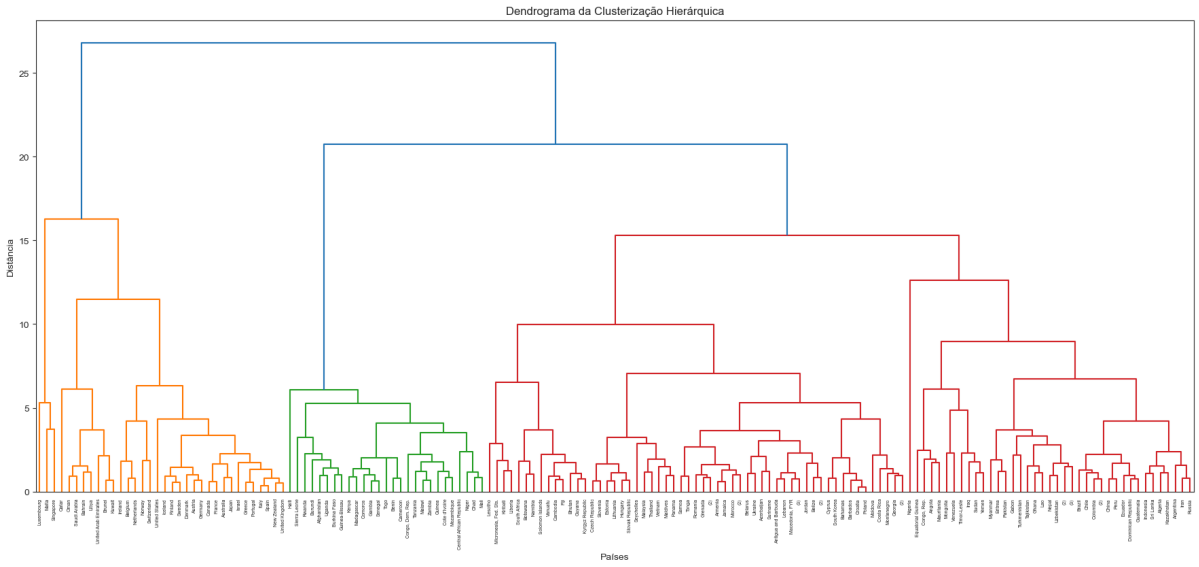


3) Para os resultados da Clusterização Hierárquica, apresente o dendograma e interprete os resultados

> Resposta:

In [373…

```
# 3) Visualização e análise da Clusterização Hierárquica
plt.figure(figsize=(22, 9))

# Cálculo da matriz de ligação para o dendrograma
hierarchical_linkage = sch.linkage(df_scaled[numeric_features], method='ward')

# Configuração do threshold para visualização
distance_threshold = 17

# Geração do dendrograma
dendrogram = sch.dendrogram(
    hierarchical_linkage,
    p=10,  # Limita o número de folhas mostradas
    truncate_mode='level',
    labels=df['país (Country)'].values,
    orientation='top'
)

# Ajuste do modelo hierárquico com threshold de distância
model = AgglomerativeClustering(n_clusters=None,
                                distance_threshold=distance_threshold)
hierarchical_predict = model.fit_predict(df_scaled[numeric_features])

# Configurações finais do gráfico do dendrograma
plt.title('Dendrograma da Clusterização Hierárquica')
plt.xlabel('Países')
plt.ylabel('Distância')
plt.show()
```



Com base no dendrograma da Clusterização Hierárquica anterior, segue algumas observações e interpretações:

1. Estrutura Geral dos Clusters:

- O dendrograma mostra uma estrutura hierárquica onde os países são agrupados em clusters, de acordo com a similaridade entre eles nas variáveis consideradas. Países mais próximos no eixo horizontal são mais semelhantes entre si.
- Observamos três grandes grupos (marcados pelas cores laranja, verde e vermelho) se formando com o threshold de distância configurado para 17.

2. Interpretação dos Grupos Principais:

- Cluster Laranja: Agrupa países com características bastante similares. Eles são unidos em uma altura de distância menor, o que indica uma proximidade entre esses países em termos das variáveis analisadas.
- Cluster Verde: Este grupo representa um segundo conjunto de países, distintos do primeiro grupo (laranja), mas ainda com certa similaridade interna.
- Cluster Vermelho: Este é o maior grupo, onde os países possuem variações maiores entre si, sendo um cluster com maior diversidade interna.

3. Altura dos Ramos:

- A altura dos ramos no dendrograma indica o nível de similaridade. Quanto maior a altura do ponto de união (ou fusão), menos semelhantes os países são. Neste caso, a linha azul que conecta os três clusters principais está em uma altura de aproximadamente 25, sugerindo que esses três clusters principais são bastante distintos entre si.

4. Número de Clusters Definido pelo Threshold:

- O threshold de distância ajustado para 17 ajuda a dividir os países em três grandes clusters, o que parece ser um ponto de corte razoável para esta análise, pois preserva a estrutura de agrupamento sem perder detalhes importantes de agrupamentos menores.

5. Padrões Regionais ou Econômicos:

> - Embora não tenhamos acesso às variáveis exatas analisadas, é possível que os clusters estejam refletindo padrões regionais, econômicos ou sociais entre os países, agrupando países com características socioeconômicas ou de desenvolvimento semelhantes.

4) Compare os dois resultados, aponte as semelhanças e diferenças e interprete

> Resposta:
> Ao comparar os resultados dos algoritmos K-Means e Clustering Hierárquico com três clusters, observamos diferenças distintas em termos de separação e organização dos dados.

In [374…

```
# Criar o DataFrame de comparação entre os métodos de clustering
comparison_df = pd.DataFrame({
    'País': df['país (Country)'],
    'Cluster K-Means': kmeans_labels,
    'Cluster Hierarchical': hierarchical_labels
})

# Ajuste a exibição do DataFrame
comparison_df.index.name = "Índice"
comparison_df.columns.name = "Comparação entre os métodos de clustering"

# Exibir o DataFrame organizado
print("Resposta:")
display(comparison_df)
```

```
Resposta:
```

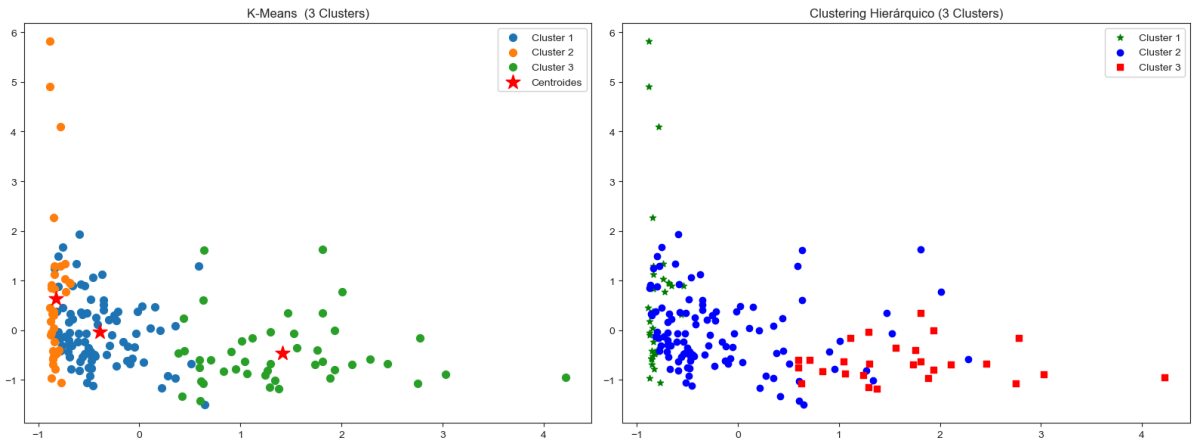
Comparação entre os métodos de clustering Índice	País	Cluster K-Means	Cluster Hierarchical
0	Afghanistan	2	2
1	Albania	0	1
2	Algeria	0	1
3	Angola	2	1
4	Antigua and Barbuda	0	1
...	...	...	...
162	Vanuatu	0	1
163	Venezuela	0	1
164	Vietnam	0	1
165	Yemen	2	1
166	Zambia	2	2

167 rows × 3 columns

> No **gráfico K-Means**, notamos uma clara separação entre os clusters. Cada grupo parece estar bem definido, com limites nítidos e uma distribuição coesa dos pontos dentro de cada cluster. Essa organização sugere que o K-Means conseguiu identificar padrões claros e distintos nos dados.
>
> Por outro lado, ao analisar a **clusterização hierárquica**, percebemos uma sobreposição sutil entre o cluster 2 e o cluster 3. Essa sobreposição indica que, no método hierárquico, os limites entre esses dois clusters podem não ser tão precisos quanto no K-Means. A proximidade desses clusters na hierarquia pode resultar em uma mistura de características, tornando a separação menos definida.
>
> Em resumo, enquanto o K-Means exibe uma melhor separação entre os clusters, a clusterização hierárquica apresenta uma ligeira sobreposição entre os clusters 2 e 3.

In [375…

```
data_array = df_scaled.values
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
cluster_names = [f'Cluster {i + 1}' for i in range(num_clusters)]

fig, ax = plt.subplots(1, 2, figsize=(16, 6))
ax[0].set_title(f'K-Means  ({num_clusters} Clusters)')


for cluster_id in range(num_clusters):
    ax[0].scatter(data_array[labels == cluster_id, 0], data_array[labels == cluster_id, 1],

ax[0].scatter(centroids[:, 0], centroids[:, 1], c='red', marker='*', s=200, label='Centroid
ax[0].legend()

colors = ['g', 'b', 'r', 'c']
markers = ['*', 'o', 's', 'x']

ax[1].set_title('Clustering Hierárquico (3 Clusters)')


for i, cluster in enumerate(np.unique(hierarchical_predict)):
    indices = np.where(hierarchical_predict == cluster)[0]
    color = colors[i % len(colors)]
    marker = markers[i % len(markers)]
    ax[1].scatter(df_scaled.iloc[indices, 0], df_scaled.iloc[indices, 1], c=color, marker=m

ax[1].legend()
plt.tight_layout()
plt.show()
```



Parte 4 - Escolha de algoritmos

1) Escreva em tópicos as etapas do algoritmo de K-médias até sua convergência.

> Resposta:
>
> **Etapas do algoritmo de K-Médias até sua convergência:**
>
> 1. **Escolha do Número de Clusters (K):**
>
> - Definir o número desejado de clusters (K), que é um parâmetro essencial para o funcionamento do algoritmo.
>
> 2. **Inicialização dos Centrôides:**
>
> - Selecionar K pontos aleatórios do dataset como centróides iniciais ou usar métodos como K-Médias++ para escolher centróides mais distantes um do outro, melhorando a eficiência e a convergência do algoritmo.
>
> 3. **Atribuição de Pontos aos Clusters:**
>
> - Para cada ponto de dados, calcular a distância até cada centróide e atribuir o ponto ao cluster cujo centróide esteja mais próximo, normalmente utilizando a distância euclidiana.
>
> 4. **Recalcular os Centrôides:**
>
> - Após a atribuição de todos os pontos, recalcular a posição de cada centróide como a média dos pontos atribuídos a ele.
>
> 5. **Verificação da Convergência:**
>
> - Repetir os passos de atribuição e recalculo dos centróides até que a posição dos centróides não mude significativamente entre as iterações, ou o movimento dos centróides fique abaixo de um limiar pré-determinado.
>   O algoritmo para quando:
> - As posições dos centróides não mudam entre iterações, ou
> - Um limite máximo de iterações é atingido.
>
> 6. **Resultado Final**:
>
> - Quando a convergência é alcançada, o algoritmo termina com clusters bem definidos pelos centróides.

2) O algoritmo de K-médias converge até encontrar os centróides que melhor descrevem os clusters encontrados (até o deslocamento entre as interações dos centróides ser mínimo). Lembrando que o centróide é o baricentro do cluster em questão e não representa, em via de regra, um dado existente na base. Refaça o algoritmo apresentado na questão 1 a fim de garantir que o cluster seja representado pelo dado mais próximo ao seu baricentro em todas as iterações do algoritmo.

Obs: nesse novo algoritmo, o dado escolhido será chamado medóide.

> Resposta:
>
> Diferentemente do K-Means, onde o centroide é representado pela média dos pontos, no K-Medoids o centroide é um ponto de dados real presente nos dados, escolhido dentro de cada cluster como aquele que minimiza a soma das dissimilaridades com os demais pontos.
>
> O K-Medoids usa um ponto real (observado) em vez de uma média para representar o centroide, tornando-se menos sensível a valores extremos. Isso é possível porque a escolha do medoide é menos afetada por outliers, já que utiliza um ponto real que minimiza a dissimilaridade com outros pontos no cluster.
>
> Por outro lado, o K-Means é suscetível à influência de outliers devido à sua abordagem de minimizar a soma dos quadrados das distâncias entre os pontos de dados e os centroides dos clusters. Os outliers podem impactar a posição dos centroides, influenciar a inicialização do algoritmo e distorcer a variância das distâncias, prejudicando a interpretação e segmentação dos dados. Outliers podem até mesmo levar à formação de agrupamentos inadequados, fragmentando clusters e comprometendo a eficácia do agrupamento.
>
> No gráfico K-Medoids Cluster, podemos explorar de forma detalhada a distribuição dos clusters obtidos por meio do algoritmo K-Medoids. Cada ponto no mapa representa um país, colorido de acordo com o cluster ao qual foi atribuído. Além disso, notamos a presença de um marcador em vermelho para cada cluster, indicando o elemento que representa o centróide deste grupo específico.
>
> Ao examinar mais de perto, identificamos que o país "Tanzania" é o elemento representativo do centróide do Cluster 1. Da mesma forma, o país "Tunisia" é o ponto central que caracteriza o Cluster 2, enquanto o Cluster 3 tem como centróide o país "DomFinland".
>
> Esses pontos vermelhos destacam a importância dos países específicos que são considerados os "medóides" ou pontos centrais de seus respectivos clusters. Eles desempenham um papel crucial na definição e representação das características predominantes de cada agrupamento.

In [376…

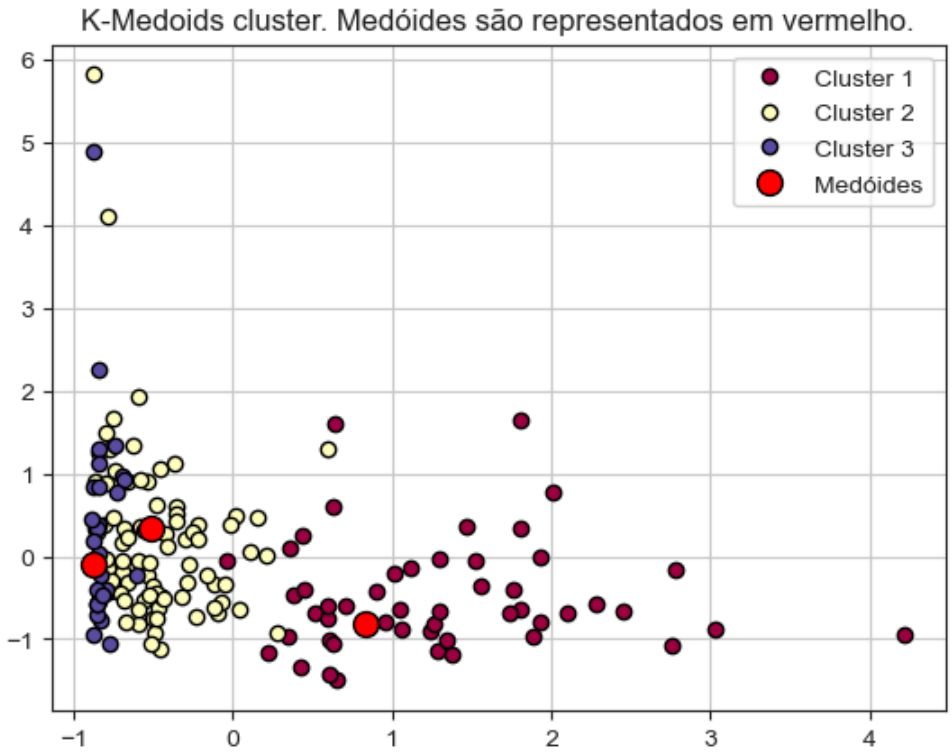
```
# Configurar e ajustar o modelo K-Medoids nos dados
KMedoid = KMedoids(
    n_clusters=3, init='k-medoids++', metric='euclidean', random_state=7
).fit(df_scaled[numeric_features])

# Obter rótulos e medóides
labels = KMedoid.labels_
unique_labels = set(labels)
KMedoid_results = df_scaled.copy()
KMedoid_results['cluster'] = labels

labels
```

Out[376…

```
array([0, 1, 1, 0, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 0, 1, 1, 1, 0,
       1, 2, 1, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 2, 1,
       2, 1, 1, 1, 1, 0, 0, 1, 1, 2, 2, 0, 0, 1, 2, 0, 2, 1, 1, 0, 0, 1,
       0, 1, 2, 0, 1, 1, 0, 2, 2, 2, 1, 2, 1, 1, 0, 0, 2, 1, 0, 1, 1, 0,
       0, 1, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       2, 2, 0, 0, 2, 1, 0, 1, 1, 1, 1, 1, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 2, 1, 0, 1, 2, 1, 1, 0, 1, 2, 2, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 2, 2, 2, 1, 1, 1, 1, 1, 0, 0], dtype=int64)
```

In [377…

```python
# Cores para os clusters
colors = [
    plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))
]

# Plotar os clusters
for k, col in zip(unique_labels, colors):
    class_member_mask = labels == k
    xy = df_scaled[numeric_features].to_numpy()[class_member_mask]
    plt.plot(
        xy[:, 0],
        xy[:, 1],
        "o",
        markerfacecolor=tuple(col),
        markeredgecolor="k",
        markersize=6,
        label=f'Cluster {k + 1}'
    )

# Plotar os medóides
medoid_coords = KMedoid.cluster_centers_
plt.plot(
    medoid_coords[:, 0],
    medoid_coords[:, 1],
    "o",
    markerfacecolor="red",
    markeredgecolor="k",
    markersize=10,
    label='Medóides'
)

# Ajustar título, grid e legenda
plt.title("K-Medoids cluster. Medóides são representados em vermelho.")
plt.grid(True)
plt.legend(loc='upper right')
plt.show()
```



In [378…

```python
# Adicionando a coluna 'Cluster' ao DataFrame df_scaled
df_scaled['Cluster'] = KMedoid.labels_

print("Resposta:")
distancias = []
for idx in range(3):
    # Centroide do cluster atual
    cluster = KMedoid.cluster_centers_[idx, :]

    # Filtrar os dados dos países pertencentes ao cluster atual
    paises = df_scaled[numeric_features][df_scaled['Cluster'] == idx]

    # Calcular as distâncias euclidianas entre cada país e o centroide do cluster
    distances = euclidean_distances(paises, [cluster])

    # Encontrar o índice do país mais próximo ao centroide (medóide)
    closest_idx = distances.argmin()

    # Obter o nome do país representativo usando o índice do DataFrame original
    representative_country = df.loc[paises.index[closest_idx], 'país (Country)']

    # Adicionar as informações de cluster, país e distância à lista
    distancias.append([idx + 1, representative_country, distances[closest_idx][0]])

    # Exibir o país representativo para cada cluster
    print(
        f"O país mais representativo do cluster {idx + 1} é {representative_country} "
        f"(distância: {distances[closest_idx][0]:.2f})"
    )
```

```
Resposta:
O país mais representativo do cluster 1 é Tanzania (distância: 0.00)
O país mais representativo do cluster 2 é Tunisia (distância: 0.00)
O país mais representativo do cluster 3 é Finland (distância: 0.00)
```

> No gráfico **"Distribuição dos Clusters com K-Medoids"**, essa representação espacial é aprimorada, permitindo-nos visualizar a distribuição geográfica dos clusters no mapa.

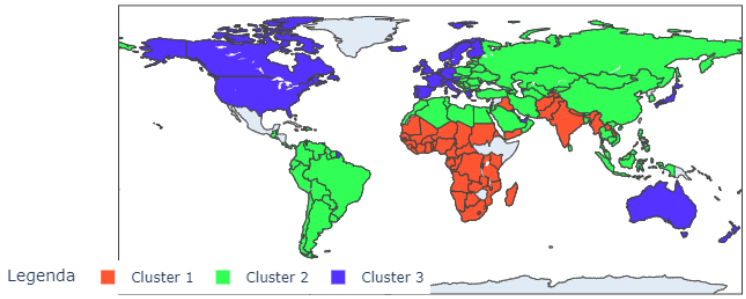
In [379…

```python
# Copiar os resultados do K-Medoids e adicionar a coluna 'country'
data = KMedoid_results.copy()
data.insert(0, column='country', value=results.index)

# Converter a coluna 'cluster' para string para evitar o erro
data['cluster'] = data['cluster'].astype(str)

# Renomear os clusters
data.loc[data['cluster'] == '0', 'cluster'] = 'Cluster 1'
data.loc[data['cluster'] == '1', 'cluster'] = 'Cluster 2'
data.loc[data['cluster'] == '2', 'cluster'] = 'Cluster 3'

# Criar o mapa coroplético
fig = px.choropleth(data,
                    locationmode='country names',
                    locations='país (Country)',  # Certifique-se de que o nome da coluna es
                    title='Distribuição dos clusters com K-Medoids',
                    color='cluster',
                    color_discrete_map={'Cluster 1': '#ff5733',
                                        'Cluster 2': '#33ff57',
                                        'Cluster 3': '#5733ff'})

fig.update_geos(fitbounds="locations", visible=True)
fig.update_layout(legend_title_text='Legenda',
                  title_pad_l=260,
                  title_y=0.90,
                  legend=dict(
                      orientation="h",
                      yanchor="bottom",
                      y=0.00,
                      xanchor="right",
                      x=0.5
                  ))
fig.show()
```

> **Distribuição Global dos Clusters:**
> A imagem indica que, com base nas características utilizadas, os países foram agrupados em três clusters distintos. Esses clusters podem refletir padrões globais específicos, como:
>
> - Cluster 1 (laranja): Agrupamento de países que podem ter semelhanças em algum aspecto econômico, geográfico ou demográfico.
> - Cluster 2 (verde): Pode representar países com níveis intermediários ou um conjunto de características que os diferenciam dos demais clusters.
> - Cluster 3 (azul): Pode representar países com perfis de desenvolvimento mais avançados ou características específicas em relação aos outros clusters.

> **Análise Geográfica e Socioeconômica:**
>
> - A distribuição mostra que países de continentes ou regiões específicas tendem a ser agrupados juntos, sugerindo que compartilham características socioeconômicas ou geopolíticas semelhantes.
> - Por exemplo, países da América do Norte e Europa estão no Cluster 3 (azul), enquanto muitos países africanos estão no Cluster 1 (laranja), o que pode refletir diferentes níveis de desenvolvimento ou outras métricas usadas no modelo.

Distribuição dos clusters com K-Medoids



3) O algoritmo de K-médias é sensível a outliers nos dados. Explique.

O K-Médias é sensível a outliers porque calcula o centróide de cada cluster como a média dos pontos pertencentes ao cluster. Um outlier, sendo um ponto muito distante da maioria dos dados do cluster, tem um grande impacto na média, fazendo com que o centróide se desloque na direção desse ponto extremo. Isso pode resultar em:

- **Clusters mal definidos**, onde o centróide não representa bem a maioria dos pontos do grupo.
- **Mudança indesejada na convergência**, pois o centróide continuará se ajustando devido à presença do outlier.

Esse comportamento faz com que o K-Médias não seja robusto para dados que possuem outliers. Em contraste, algoritmos que utilizam a mediana ou outras métricas menos sensíveis a outliers podem lidar melhor com dados extremos.

4) Por que o algoritmo de DBScan é mais robusto à presença de outliers?

O DBScan (Density-Based Spatial Clustering of Applications with Noise) é mais robusto a outliers porque, ao contrário do K-Médias, ele não exige que todos os pontos sejam atribuídos a um cluster. Em vez disso, o DBScan agrupa pontos densamente conectados e trata pontos isolados (ou com densidade insuficiente) como "ruído" ou "outliers".

Esse método é mais robusto porque:

- **Não força os outliers a fazerem parte de clusters**, deixando-os como pontos de ruído.
- **Agrupa somente pontos com vizinhança densa**, o que reduz a influência de pontos dispersos.
- **Define clusters de forma adaptativa** com base na densidade, tornando-se menos influenciado por outliers que estão longe da massa de dados.

Essas características fazem do DBScan uma escolha robusta para dados com outliers, pois ele os ignora ao formar clusters.