

## 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+

```
(py39) C:\Users\alan echer\Documents\25E4-infnet-algoritmos-de-inteligencia-artificial-para-clusterizacao>python --version
Python 3.9.23

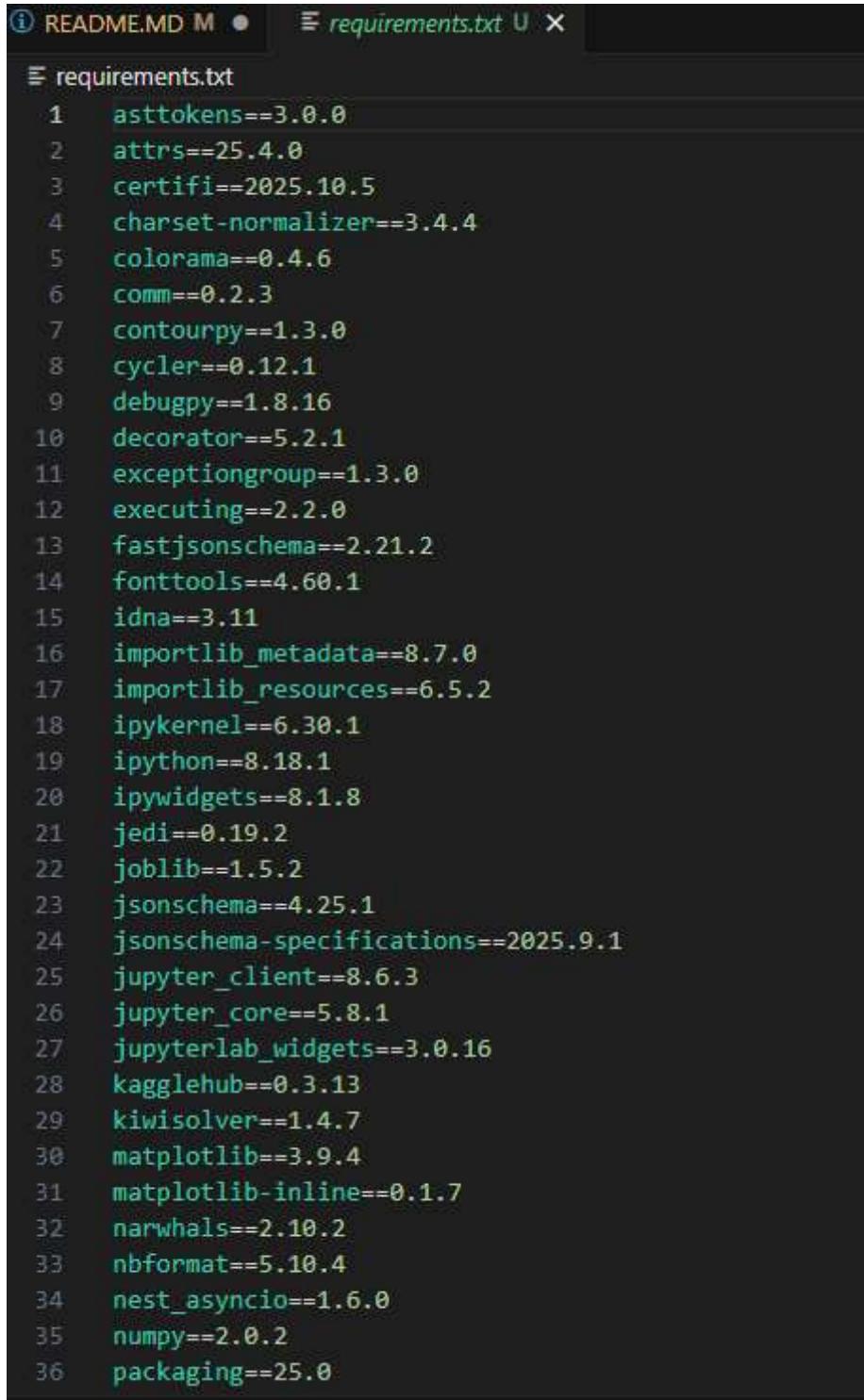
(py39) C:\Users\alan echer\Documents\25E4-infnet-algoritmos-de-inteligencia-artificial-para-clusterizacao>
```

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

```
(base) C:\Users\alan echer\Documents\25E4-infnet-algoritmos-de-inteligencia-artificial-para-clusterizacao>conda activate
py39

(py39) C:\Users\alan echer\Documents\25E4-infnet-algoritmos-de-inteligencia-artificial-para-clusterizacao>
```

3. Todas as bibliotecas usadas nesse exercícios estão instaladas em um ambiente virtual específico anaconda - py39
4. Gere um arquivo de requerimentos (requirements.txt) com os pacotes necessários. É necessário se certificar que a versão do pacote está disponibilizada.



The screenshot shows a terminal window with two tabs: 'README.MD' and 'requirements.txt'. The 'requirements.txt' tab is active, displaying a list of Python package dependencies. The list includes packages like asttokens, attrs, certifi, charset-normalizer, colorama, comm, contourpy, cycler, debugpy, decorator, exceptiongroup, executing, fastjsonschema, fonttools, idna, importlib\_metadata, importlib\_resources, ipykernel, ipython, ipywidgets, jedi, joblib, jsonschema, jsonschema-specifications, jupyter\_client, jupyter\_core, jupyterlab\_widgets, kagglehub, kiwisolver, matplotlib, matplotlib-inline, narwhals, nbformat, nest\_asyncio, numpy, and packaging.

```
① README.MD M ● ② requirements.txt U X  
③ requirements.txt  
1 asttokens==3.0.0  
2 attrs==25.4.0  
3 certifi==2025.10.5  
4 charset-normalizer==3.4.4  
5 colorama==0.4.6  
6 comm==0.2.3  
7 contourpy==1.3.0  
8 cycler==0.12.1  
9 debugpy==1.8.16  
10 decorator==5.2.1  
11 exceptiongroup==1.3.0  
12 executing==2.2.0  
13 fastjsonschema==2.21.2  
14 fonttools==4.60.1  
15 idna==3.11  
16 importlib_metadata==8.7.0  
17 importlib_resources==6.5.2  
18 ipykernel==6.30.1  
19 ipython==8.18.1  
20 ipywidgets==8.1.8  
21 jedi==0.19.2  
22 joblib==1.5.2  
23 jsonschema==4.25.1  
24 jsonschema-specifications==2025.9.1  
25 jupyter_client==8.6.3  
26 jupyter_core==5.8.1  
27 jupyterlab_widgets==3.0.16  
28 kagglehub==0.3.13  
29 kiwisolver==1.4.7  
30 matplotlib==3.9.4  
31 matplotlib-inline==0.1.7  
32 narwhals==2.10.2  
33 nbformat==5.10.4  
34 nest_asyncio==1.6.0  
35 numpy==2.0.2  
36 packaging==25.0
```

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

The screenshot shows a Jupyter Notebook interface with two scatter plots. The left plot is titled 'Comparação visual entre K-Médias e K-Medóides' and the right is 'K-MEDÓIDES (Medídeis em preto)'. Both plots have 'child\_mort' on the x-axis and 'health' on the y-axis, ranging from 0.0 to 1.0. The plots show data points colored by cluster and centroids marked with stars and crosses. The notebook code at the top includes imports for pandas, numpy, and matplotlib, and defines functions for K-Medoids and K-Medians clustering.

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).

<https://github.com/echer/25E4-infnet-algoritmos-de-inteligencia-artificial-para-clusterizacao.git>

## 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>

```
# PARTE 2 - 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
path = kagglehub.dataset_download("rohan0301/unsupervised-learning-on-country-data")
```

2. Quantos países existem no dataset?

The screenshot shows a Jupyter Notebook cell with the following code: '# PARTE 2 - 2 Quantos países existem no dataset?' followed by 'print('Qtde Países no dataset: ', df['country'].count())'. The output shows 'Qtde Países no dataset: 167'.

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?\\

1. Não há valores nulos no conjunto de dados\\
2. Tem 167 países no total\\
3. Todas as variáveis são numéricas, exceto a coluna 'country'. \\

4. Foi observado grandes diferenças nas escalas, os dados possuem variação alta, presença de outliers e diferença de escala entre variáveis.

```
# Visualizando as primeiras linhas e informações básicas
print("Primeiras linhas do dataset:")
print(df.head())
print("\nInformações do dataset:")
print(df.info())
print("\nEstatísticas descritivas:")
print(df.describe())
print("\nValores nulos por coluna:")
print(df.isnull().sum())

142] ✓ 0.0s
```

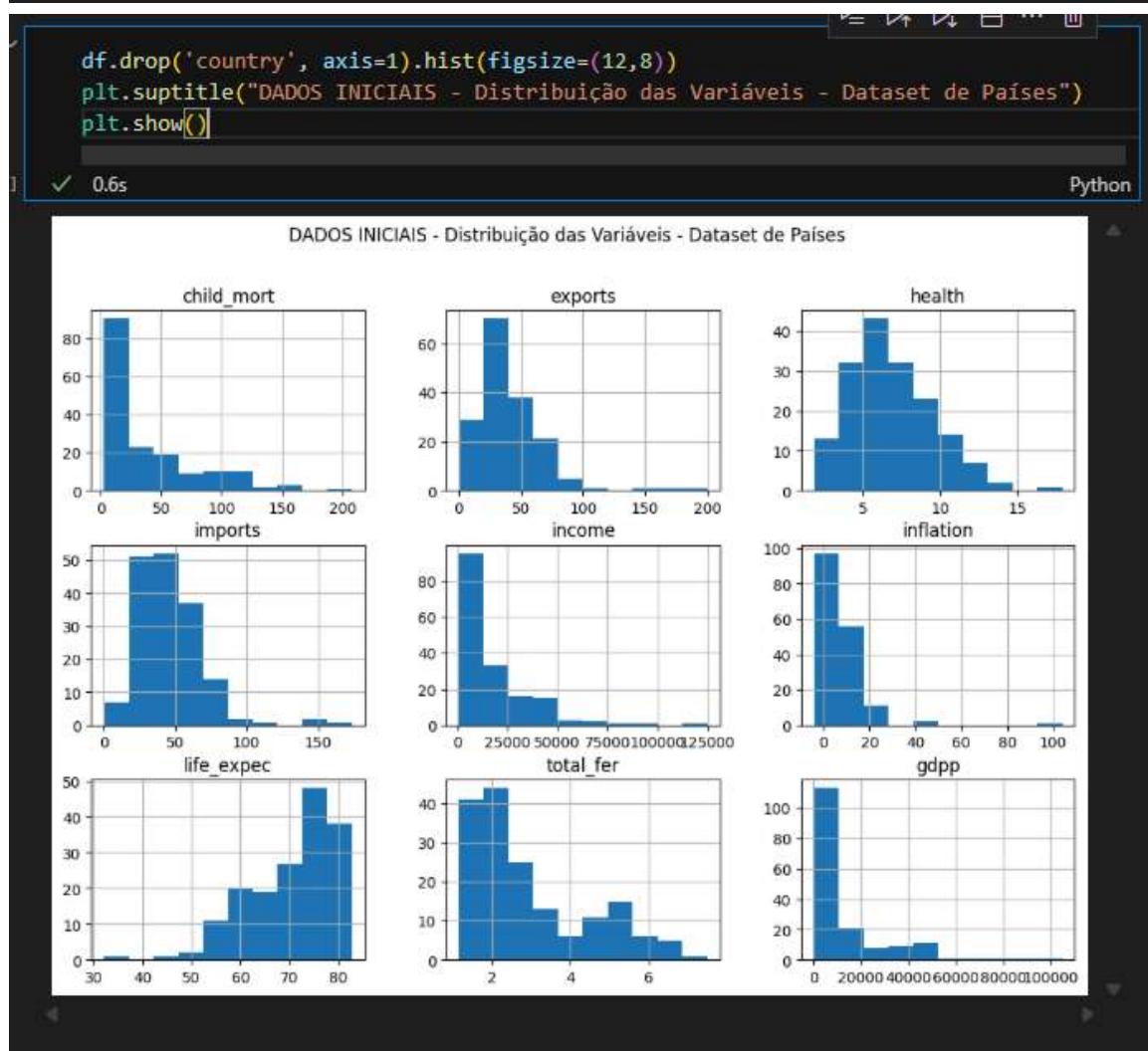
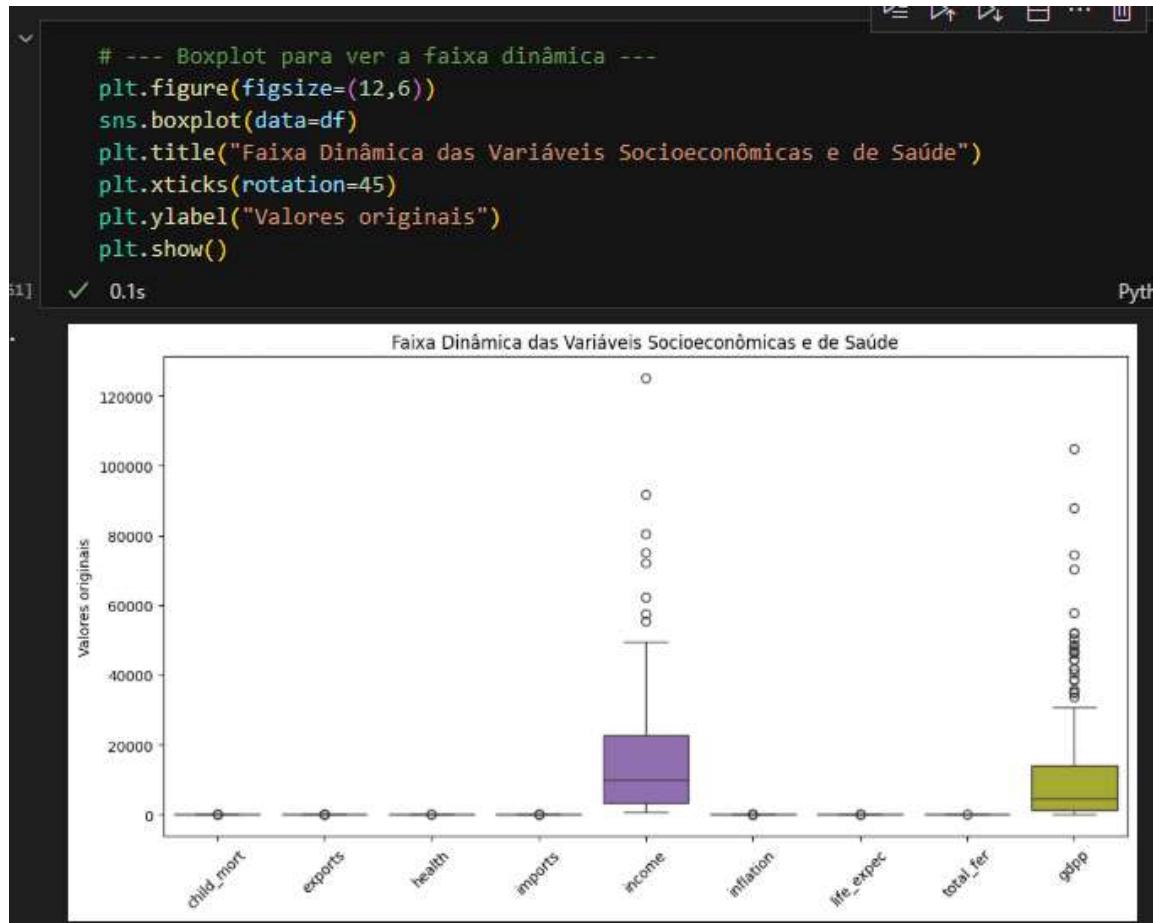
.. Primeiras linhas do dataset:

	country	child_mort	exports	health	imports	income	\
0	Afghanistan	90.2	10.0	7.58	44.9	1610	
1	Albania	16.6	28.0	6.55	48.6	9930	
2	Algeria	27.3	38.4	4.17	31.4	12900	
3	Angola	119.0	62.3	2.85	42.9	5900	
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	

	inflation	life_expec	total_fer	gdpp
0	9.44	56.2	5.82	553
1	4.49	76.3	1.65	4090
2	16.10	76.5	2.89	4460
3	22.40	60.1	6.16	3530
4	1.44	76.8	2.13	12200

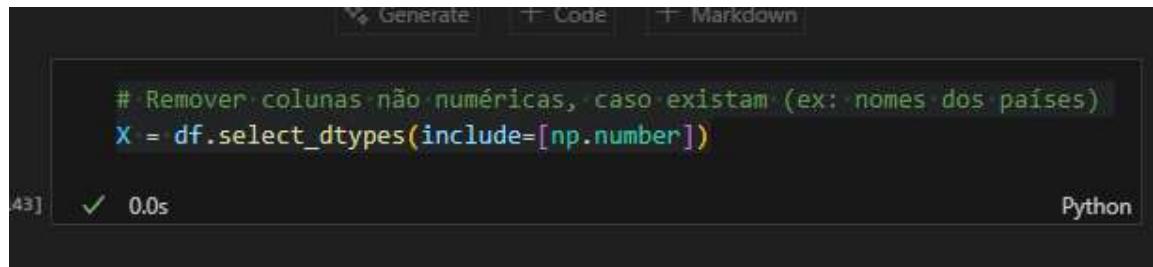
Informações do dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   country     167 non-null    object 
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 ...
 life_expec    0                
 total_fer    0                
 gdpp        0               
```



4. Realize o pré-processamento adequado dos dados.\

## 1. Remocao das colunas nao numericas.

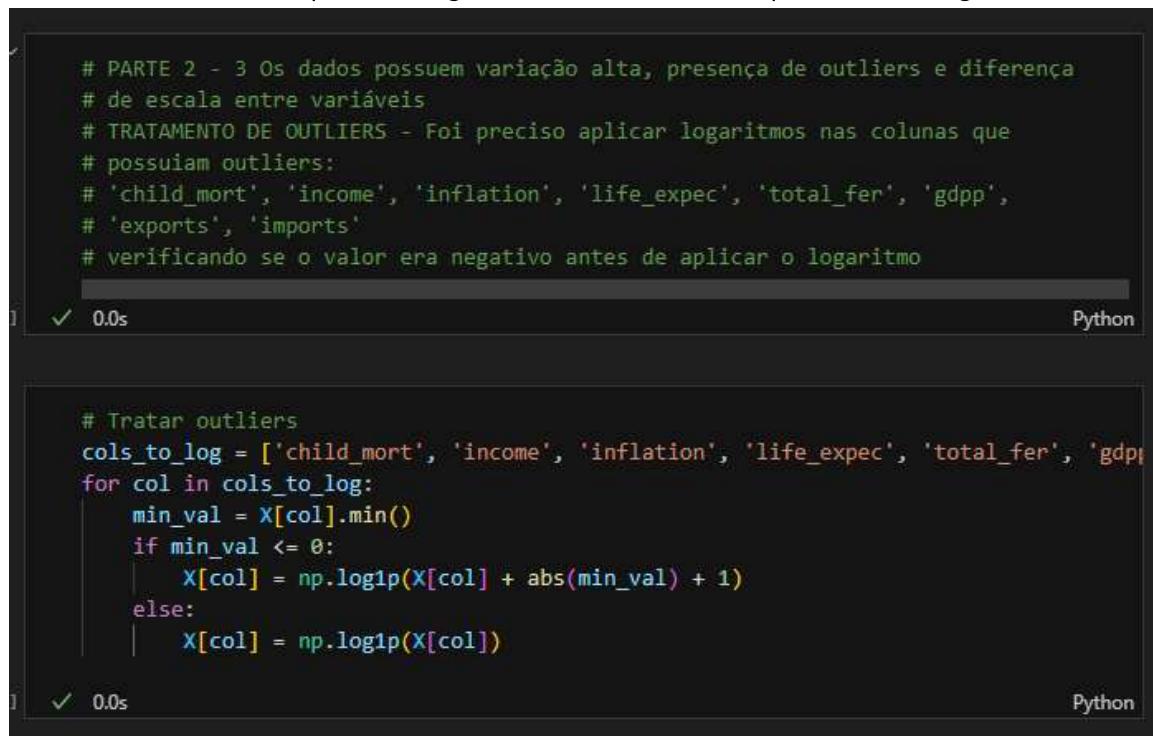


```
# Remover colunas não numéricas, caso existam (ex: nomes dos países)
X = df.select_dtypes(include=[np.number])
```

431 ✓ 0.0s Python

\

## 2. Tratamento de outliers aplicando logaritmos com tratamento para valores negativos.



```
# PARTE 2 - 3 Os dados possuem variação alta, presença de outliers e diferença
# de escala entre variáveis
# TRATAMENTO DE OUTLIERS - Foi preciso aplicar logaritmos nas colunas que
# possuam outliers:
# 'child_mort', 'income', 'inflation', 'life_expec', 'total_fer', 'gdpp',
# 'exports', 'imports'
# verificando se o valor era negativo antes de aplicar o logaritmo
```

1 ✓ 0.0s Python

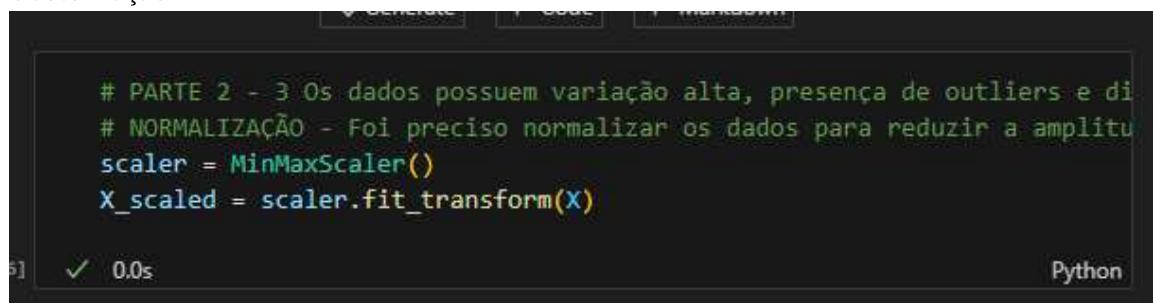
  

```
# Tratar outliers
cols_to_log = ['child_mort', 'income', 'inflation', 'life_expec', 'total_fer', 'gdpp']
for col in cols_to_log:
    min_val = X[col].min()
    if min_val <= 0:
        X[col] = np.log1p(X[col] + abs(min_val) + 1)
    else:
        X[col] = np.log1p(X[col])
```

1 ✓ 0.0s Python

\

## 3. Redução da escala de valores para melhorar o resultado na aplicação dos modelos de clusterização.

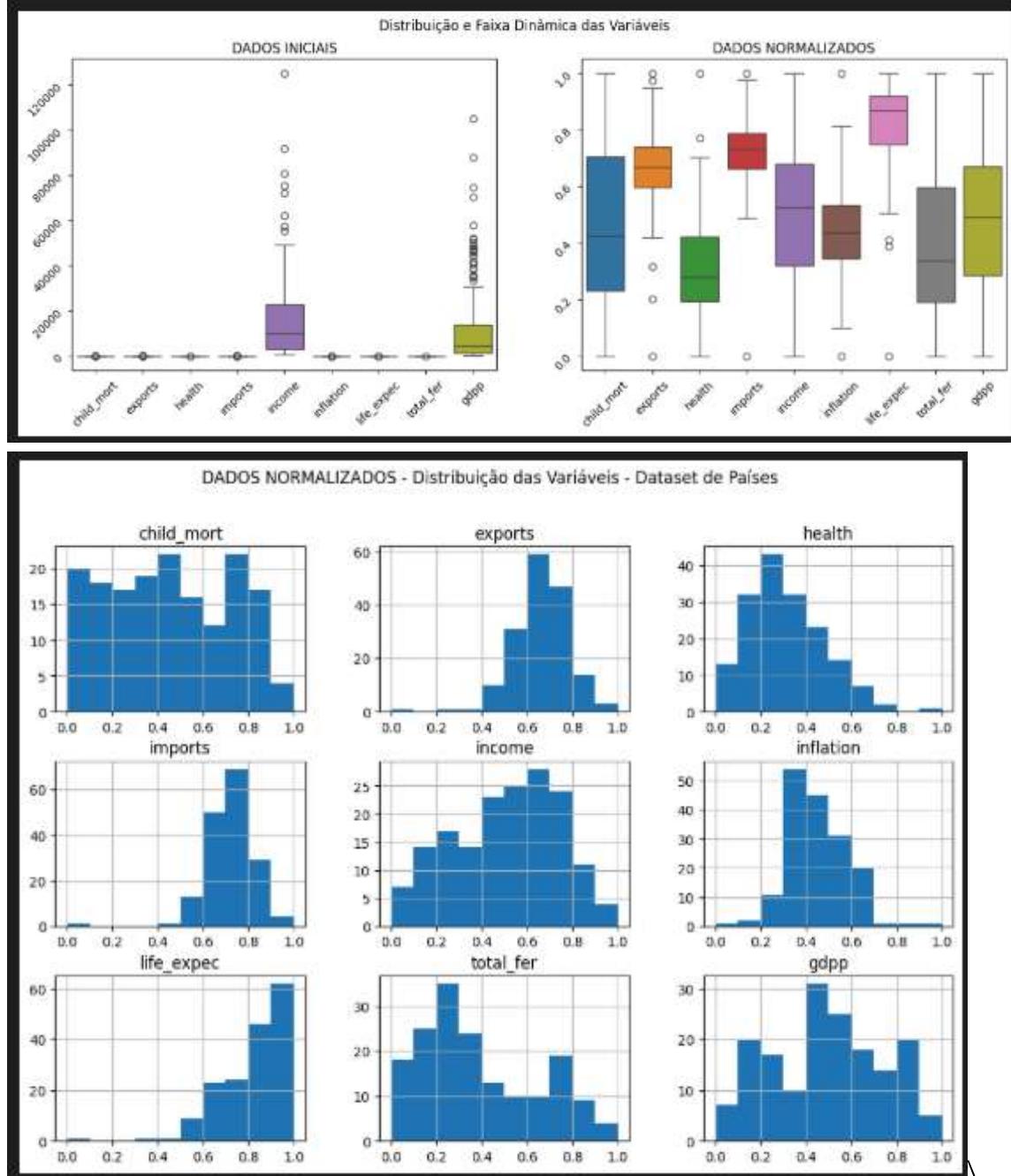


```
# PARTE 2 - 3 Os dados possuem variação alta, presença de outliers e di
# NORMALIZAÇÃO - Foi preciso normalizar os dados para reduzir a amplitude
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

6] ✓ 0.0s Python

\

#### 4. Resultados obtidos em comparação com os valores iniciais com a redução da escala.

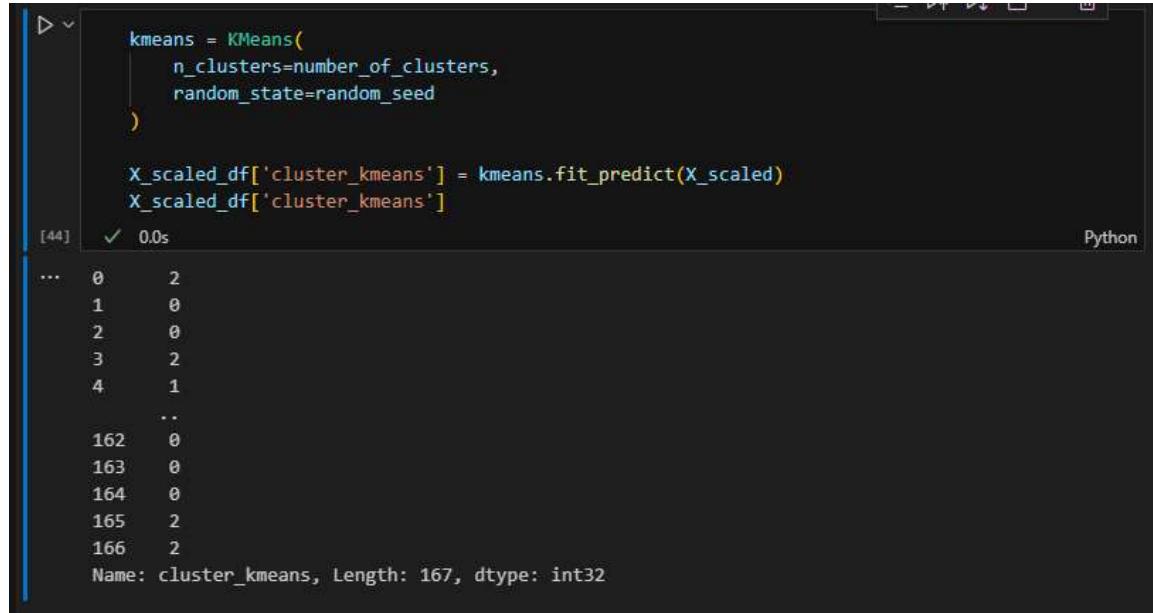


### Parte 3 - Clusterização

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

1. Realizar o agrupamento dos países em 3 grupos distintos. Para tal, use:

## 1. K-Médias



```

kmeans = KMeans(
    n_clusters=number_of_clusters,
    random_state=random_seed
)

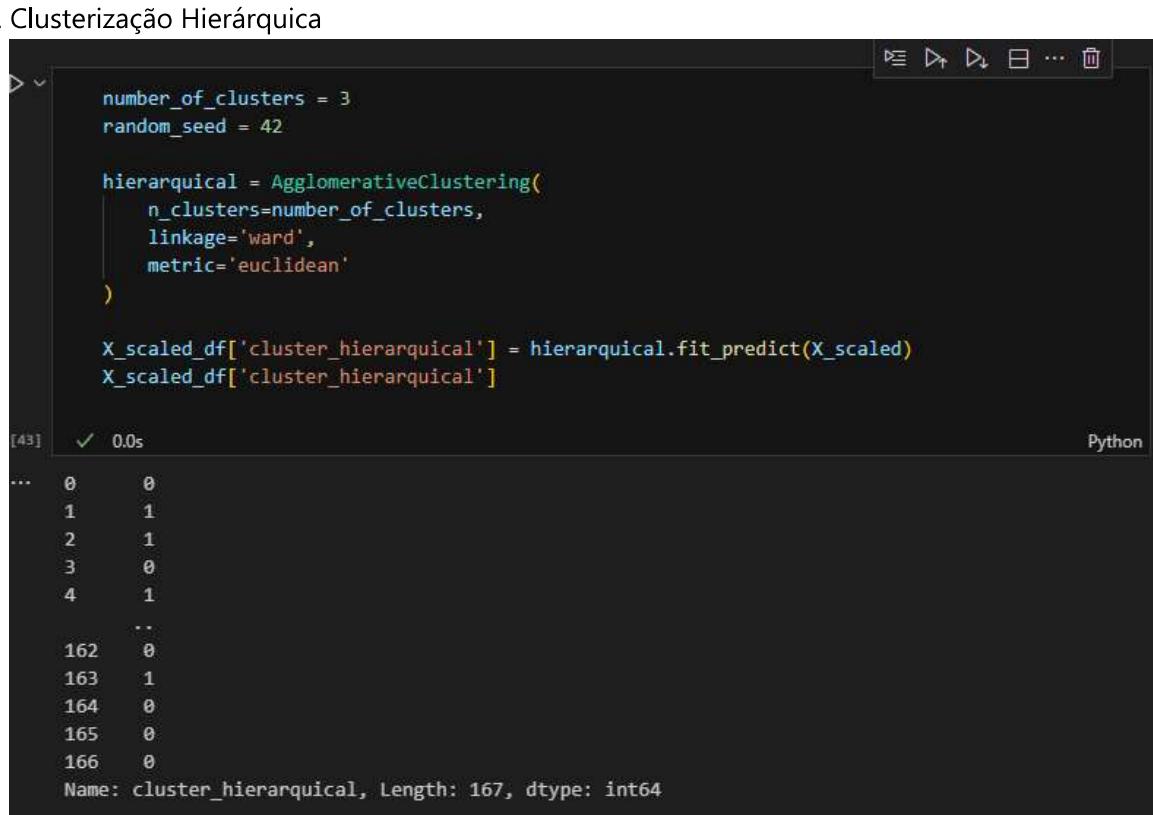
X_scaled_df['cluster_kmeans'] = kmeans.fit_predict(X_scaled)
X_scaled_df['cluster_kmeans']

[44]    ✓  0.0s                                         Python

...
0      2
1      0
2      0
3      2
4      1
...
162     0
163     0
164     0
165     2
166     2
Name: cluster_kmeans, Length: 167, dtype: int32

```

## 2. Clusterização Hierárquica



```

number_of_clusters = 3
random_seed = 42

hierarchical = AgglomerativeClustering(
    n_clusters=number_of_clusters,
    linkage='ward',
    metric='euclidean'
)

X_scaled_df['cluster_hierarquical'] = hierarchical.fit_predict(X_scaled)
X_scaled_df['cluster_hierarquical']

[43]    ✓  0.0s                                         Python

...
0      0
1      1
2      1
3      0
4      1
...
162     0
163     1
164     0
165     0
166     0
Name: cluster_hierarquical, Length: 167, dtype: int64

```

2. Para os resultados, do K-Médias:

1. Interprete cada um dos clusters obtidos citando:

### 1. Qual a distribuição das dimensões em cada grupo



O algoritmo K-Means conseguiu separar os países em três níveis de desenvolvimento socioeconômico:

Cluster 0: países emergentes, com indicadores intermediários.

Cluster 1: países desenvolvidos, com alta renda, PIB e expectativa de vida.

Cluster 2: países subdesenvolvidos, com baixa renda e alta mortalidade.\

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

```

# Obter os centróides
centroids = kmeans.cluster_centers_

# Calcular a distância de cada ponto ao centróide do seu cluster
distances = []
for i, row in enumerate(X_scaled):
    cluster_label = X_scaled_df.loc[i, 'cluster_kmeans']
    centroid = centroids[cluster_label]
    distance = np.linalg.norm(row - centroid)
    distances.append(distance)

X_scaled_df['distance_to_centroid'] = distances

X_scaled_df['country'] = df['country']
# Encontrar o país mais representativo de cada cluster
representative_countries = X_scaled_df.loc[
    X_scaled_df.groupby('cluster_kmeans')['distance_to_centroid'].idxmin(),
    ['country', 'cluster_kmeans', 'distance_to_centroid']]
X_scaled_df = X_scaled_df.drop('country', axis=1)

print("Países que melhor representam cada cluster:\n")
print(representative_countries)

```

[62] ✓ 0.0s

... Países que melhor representam cada cluster:

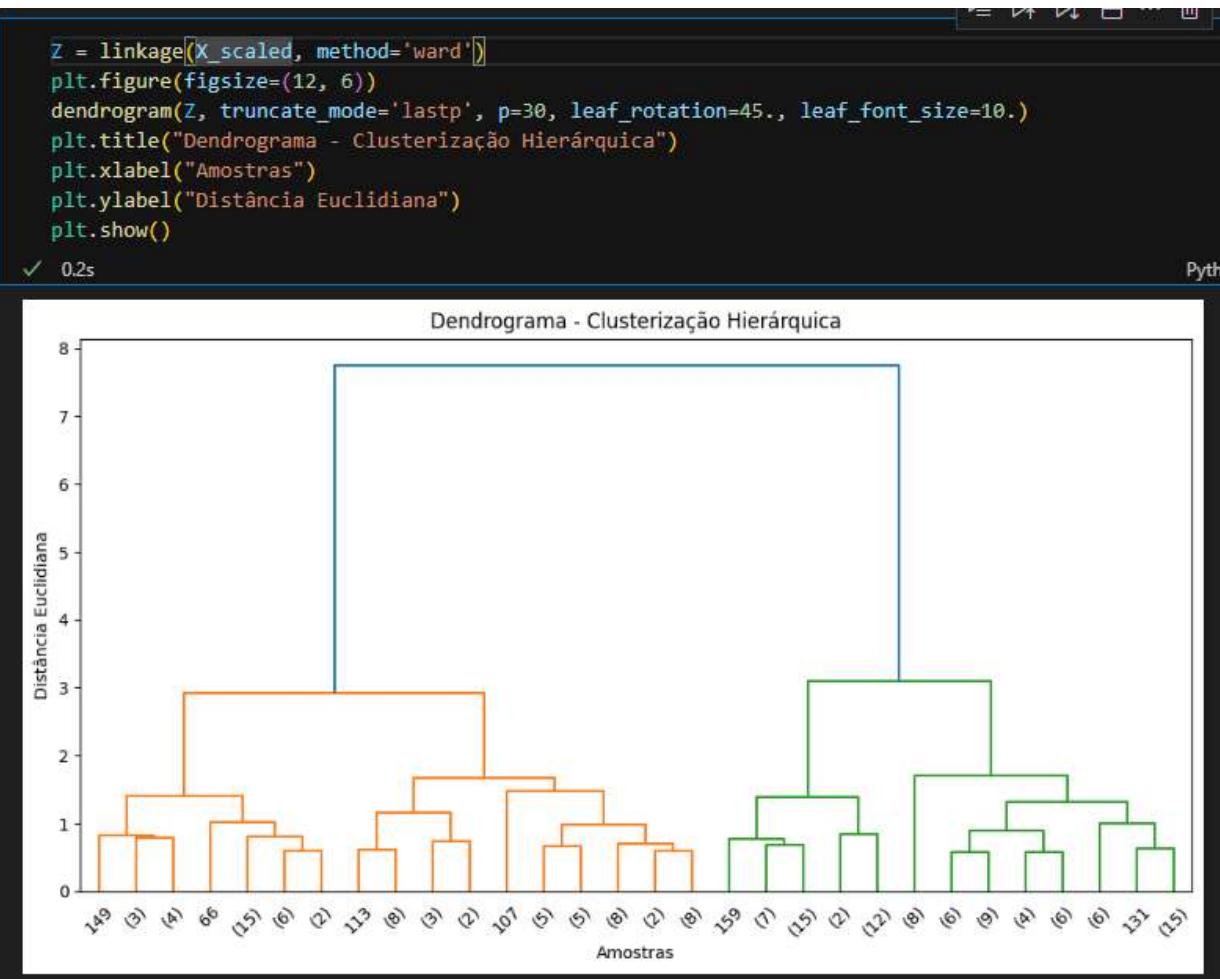
	country	cluster_kmeans	distance_to_centroid
118	Paraguay	0	0.128643
41	Croatia	1	0.145154
147	Tanzania	2	0.098071

Cluster 0: O Paraguai foi considerado pelo algoritmo por possuir valores medianos de PIB, mortalidade e expectativa de vida de acordo com seu centróide.

Cluster 1: A Croácia foi considerado pelo algoritmo por possuir indicadores socioeconômicos muito próximos do centróide de países desenvolvidos.

Cluster 2: A Tanzânia foi considerado pelo algoritmo por representar a média de países com desafios sociais e econômicos de acordo com seu centróide.\

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



Analisando o dendograma podemos identificar que temos duas grandes estruturas hierárquicas diferentes entre si, olhando no topo do gráfico elas se juntam apenas próximo a distância euclidiana 8, demonstrando essa grande diferença, um grupo deve ter características socioeconómicas semelhantes e de nível mais elevado enquanto o outro deve ter indicadores menores com baixa renda e saúde. Olhando mais abaixo na distância euclidiana de 0 a 3 temos países bem semelhantes entre si separados em grupos menores com suas particularidades e diferenças de grupos.

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



Ambos os algoritmos conseguiram separar os dados em 3 clusters diferentes, lembrando que a separação dos dados em clusters leva em conta todas as colunas que foram passadas para análises, porém montamos o gráfico para comparação utilizando apenas duas colunas health x child\_mort, tendo assim uma visão de como foi feito a separação olhando deste ponto de vista. Olhando para os dados de gasto em saúde e morte infantil, temos no primeiro terço o kmeans criando o cluster 1 enquanto do outro lado no hierárquico temos o cluster 2, o Kmeans se alongou mais nos dados e pegou uma quantidade maior enquanto o hierárquico se reteve em uma faixa de dados um pouco menor para separação essa faixa de dados é representativa de países desenvolvidos com baixa mortalidade infantil e alto gasto em saúde. Agora observando o segundo terço de dados temos no kmeans o cluster 0 e no hierárquico temos o cluster 1, nesse caso temos países intermediários com mortalidade moderada e gastos típicos de economias emergentes, com o hierárquico temos o início do grupo em iniciando em mortalidade infantil aproximada de 0.2 e terminando logo após 0.4 enquanto no kmeans temos o grupo iniciando em meados de 0.3 e terminando em 0.7 uma diferença até bem significativa na divisão dos dados, lembrando que os dados foram divididos utilizando todas as colunas numéricas do dataset e não apenas as duas colunas analisadas. E no último terço dos dados temos ali no kmeans um grupo bem menor enquanto no hierárquico temos um grupo bem maior de países, que representam países com alta mortalidade infantil.

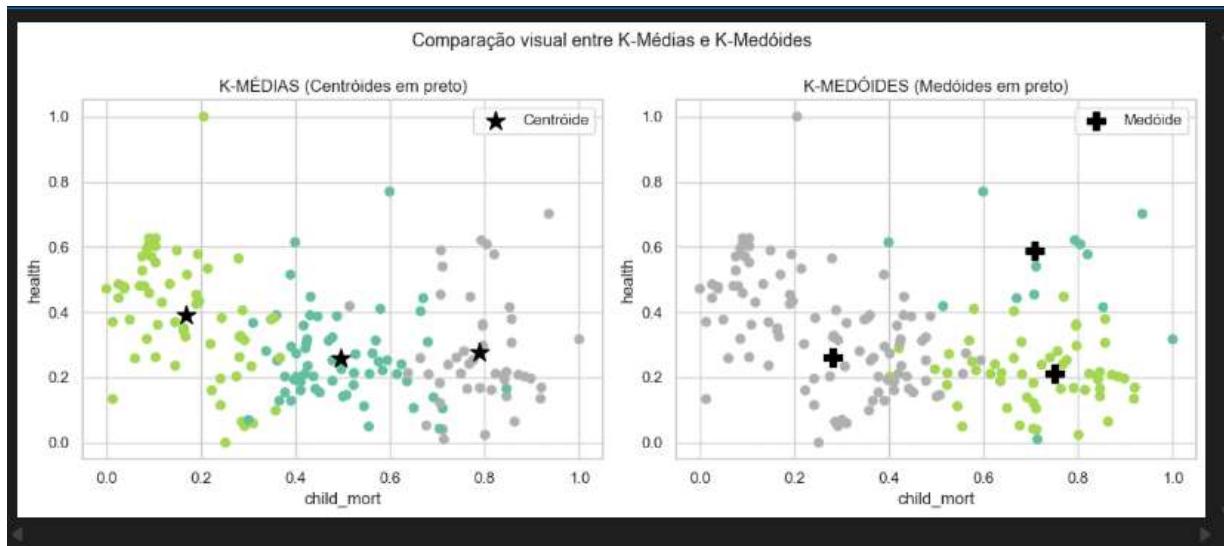
## Parte 4 - Escolha de algoritmos

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

O primeiro passo é definir o número de clusters para o algoritmo formar, depois o algoritmo irá

selecionar aleatoriamente centroides iniciais aleatórios, em seguida as amostras de dados são atribuídas ao centroide do cluster mais próximo utilizando a distância euclidiana, após essa atribuição é recalculado os centroides com base como a média das amostras pertencentes a cada grupo, este processo de atribuição e atualização dos centroides é repetido de forma a atingir a convergência, quando não há mais mudanças significativas na posição dos centróides ou dos clusters.

- 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.



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

O k-means é sensível a outlier de dados pois calcula o centroide de cada cluster utilizando a média dos pontos pertencentes a ele, quando há valores distantes extremos no conjunto de dados isso acaba impactando e puxando a média para fora da região central, deslocando o centróide, podendo gerar agrupamentos incorretos ou pouco representativos.

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

O algoritmo DBScan utiliza uma técnica diferente baseado na densidade de pontos próximos no espaço, um ponto é considerado core se tiver um número minímo de vizinhos a um certo raio de distância, enquanto um ponto que estiver muito distante de uma região densa de vizinhos são simplesmente rotulados como ruidos e não influenciam na formação dos clusters.

Abaixo o código do notebook executado na integra:

```
In [32]: import kagglehub
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage

# PARTE 2 - 1 Baixe os dados disponibilizados na plataforma Kaggle sobre dados s
# 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-dat
path = kagglehub.dataset_download("rohan0301/unsupervised-learning-on-country-da
print(path)
# Carregar os dados
df = pd.read_csv(path+"\Country-data.csv", sep = ",")

# PARTE 2 - 2 Quantos países existem no dataset?
print('Qtde Países no dataset: ', df['country'].count())
```

C:\Users\alan echer\.cache\kagglehub\datasets\rohan0301\unsupervised-learning-on-country-data\versions\2  
Qtde Países no dataset: 167

```
In [33]: # Visualizando as primeiras Linhas e informações básicas
print("Primeiras linhas do dataset:")
print(df.head())
print("\nInformações do dataset:")
print(df.info())
print("\nEstatísticas descritivas:")
print(df.describe())
print("\nValores nulos por coluna:")
print(df.isnull().sum())
```

Primeiras linhas do dataset:

	country	child_mort	exports	health	imports	income	\
0	Afghanistan	90.2	10.0	7.58	44.9	1610	
1	Albania	16.6	28.0	6.55	48.6	9930	
2	Algeria	27.3	38.4	4.17	31.4	12900	
3	Angola	119.0	62.3	2.85	42.9	5900	
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	
	inflation	life_expec	total_fer	gdpp			
0	9.44	56.2	5.82	553			
1	4.49	76.3	1.65	4090			
2	16.10	76.5	2.89	4460			
3	22.40	60.1	6.16	3530			
4	1.44	76.8	2.13	12200			

Informações do dataset:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 167 entries, 0 to 166

Data columns (total 10 columns):

#	Column	Non-Null Count	Dtype
0	country	167	non-null
1	child_mort	167	non-null
2	exports	167	non-null
3	health	167	non-null
4	imports	167	non-null
5	income	167	non-null
6	inflation	167	non-null
7	life_expec	167	non-null
8	total_fer	167	non-null
9	gdpp	167	non-null

dtypes: float64(7), int64(2), object(1)

memory usage: 13.2+ KB

None

Estatísticas descritivas:

	child_mort	exports	health	imports	income	\
count	167.000000	167.000000	167.000000	167.000000	167.000000	
mean	38.270060	41.108976	6.815689	46.890215	17144.688623	
std	40.328931	27.412010	2.746837	24.209589	19278.067698	
min	2.600000	0.109000	1.810000	0.065900	609.000000	
25%	8.250000	23.800000	4.920000	30.200000	3355.000000	
50%	19.300000	35.000000	6.320000	43.300000	9960.000000	
75%	62.100000	51.350000	8.600000	58.750000	22800.000000	
max	208.000000	200.000000	17.900000	174.000000	125000.000000	

	inflation	life_expec	total_fer	gdpp
count	167.000000	167.000000	167.000000	167.000000
mean	7.781832	70.555689	2.947964	12964.155689
std	10.570704	8.893172	1.513848	18328.704809
min	-4.210000	32.100000	1.150000	231.000000
25%	1.810000	65.300000	1.795000	1330.000000
50%	5.390000	73.100000	2.410000	4660.000000
75%	10.750000	76.800000	3.880000	14050.000000
max	104.000000	82.800000	7.490000	105000.000000

Valores nulos por coluna:

country	0
child_mort	0
exports	0

```
health      0
imports     0
income      0
inflation   0
life_expec  0
total_fer   0
gdpp        0
dtype: int64
```

In [34]: *# Remover colunas não numéricas, caso existam (ex: nomes dos países)*  
`X = df.select_dtypes(include=[np.number])`

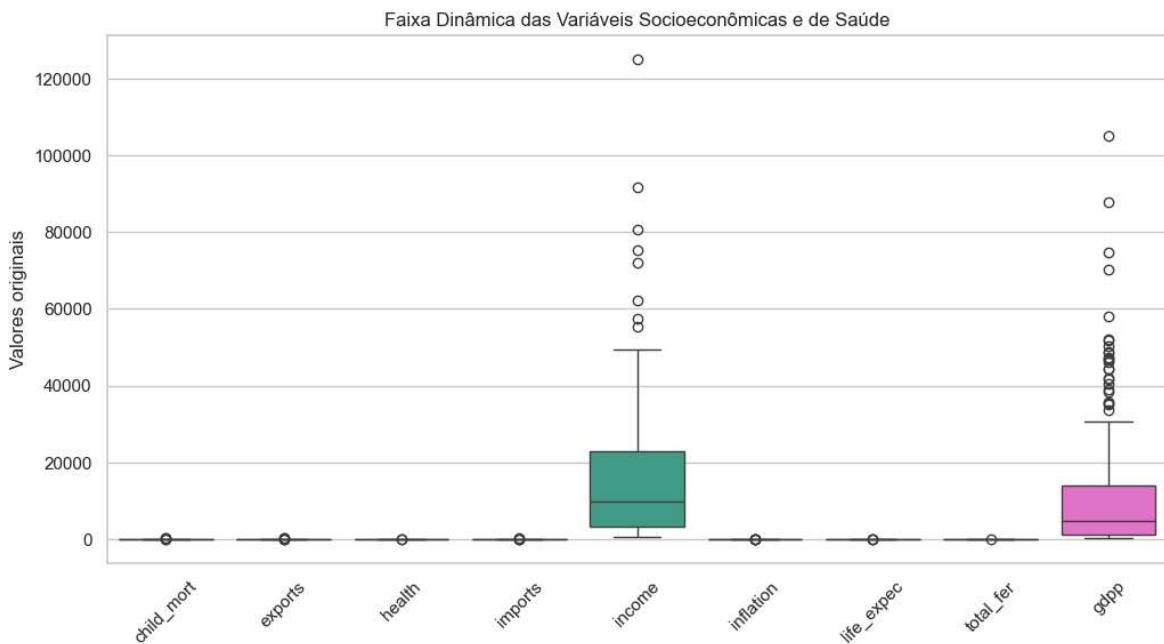
In [35]: *# PARTE 2 - 3 Os dados possuem variação alta, presença de outliers e diferença de escala entre variáveis*  
*# TRATAMENTO DE OUTLIERS - Foi preciso aplicar logaritmos nas colunas que possuíam outliers:*  
*# 'child\_mort', 'income', 'inflation', 'life\_expec', 'total\_fer', 'gdpp',*  
*# 'exports', 'imports'*  
*# verificando se o valor era negativo antes de aplicar o Logaritmo*

In [36]: *# Tratar outliers*  
`cols_to_log = ['child_mort', 'income', 'inflation', 'life_expec', 'total_fer', 'exports', 'imports']`  
`for col in cols_to_log:`  
 `min_val = X[col].min()`  
 `if min_val <= 0:`  
 `X[col] = np.log1p(X[col] + abs(min_val) + 1)`  
 `else:`  
 `X[col] = np.log1p(X[col])`

In [37]: *# PARTE 2 - 3 Os dados possuem variação alta, presença de outliers e diferença de escala entre variáveis*  
*# NORMALIZAÇÃO - Foi preciso normalizar os dados para reduzir a amplitude dos valores*  
`scaler = MinMaxScaler()`  
`X_scaled = scaler.fit_transform(X)`

In [38]: `X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)`

In [39]: *# --- Boxplot para ver a faixa dinâmica ---*  
`plt.figure(figsize=(12,6))`  
`sns.boxplot(data=df)`  
`plt.title("Faixa Dinâmica das Variáveis Socioeconômicas e de Saúde")`  
`plt.xticks(rotation=45)`  
`plt.ylabel("Valores originais")`  
`plt.show()`

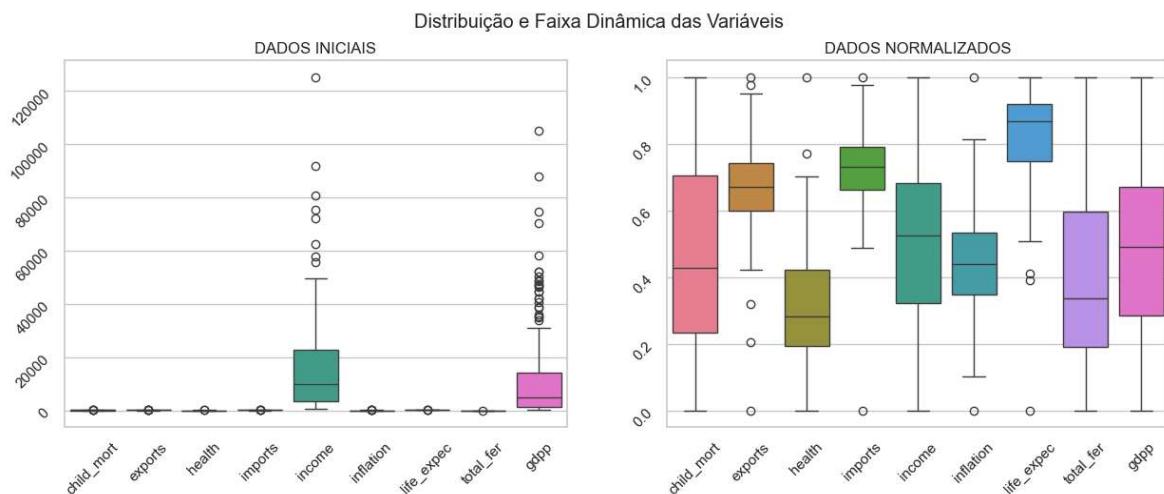


```
In [40]: f, ax = plt.subplots(1, 2, figsize=(15,5))

sns.boxplot(data=df.drop('country', axis=1), ax=ax[0])
ax[0].set_title('DADOS INICIAIS')
ax[0].tick_params(labelrotation=45)

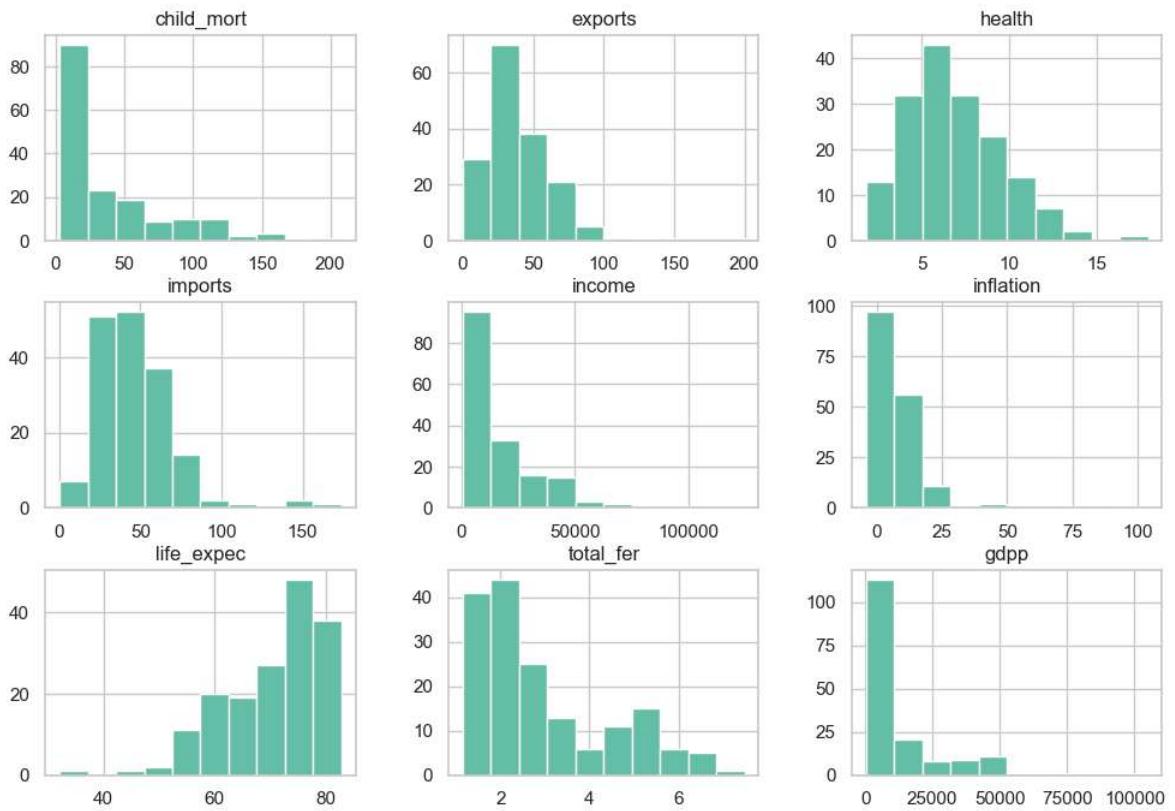
sns.boxplot(data=X_scaled_df, ax=ax[1])
ax[1].set_title('DADOS NORMALIZADOS')
ax[1].tick_params(labelrotation=45)

plt.suptitle('Distribuição e Faixa Dinâmica das Variáveis')
plt.show()
```



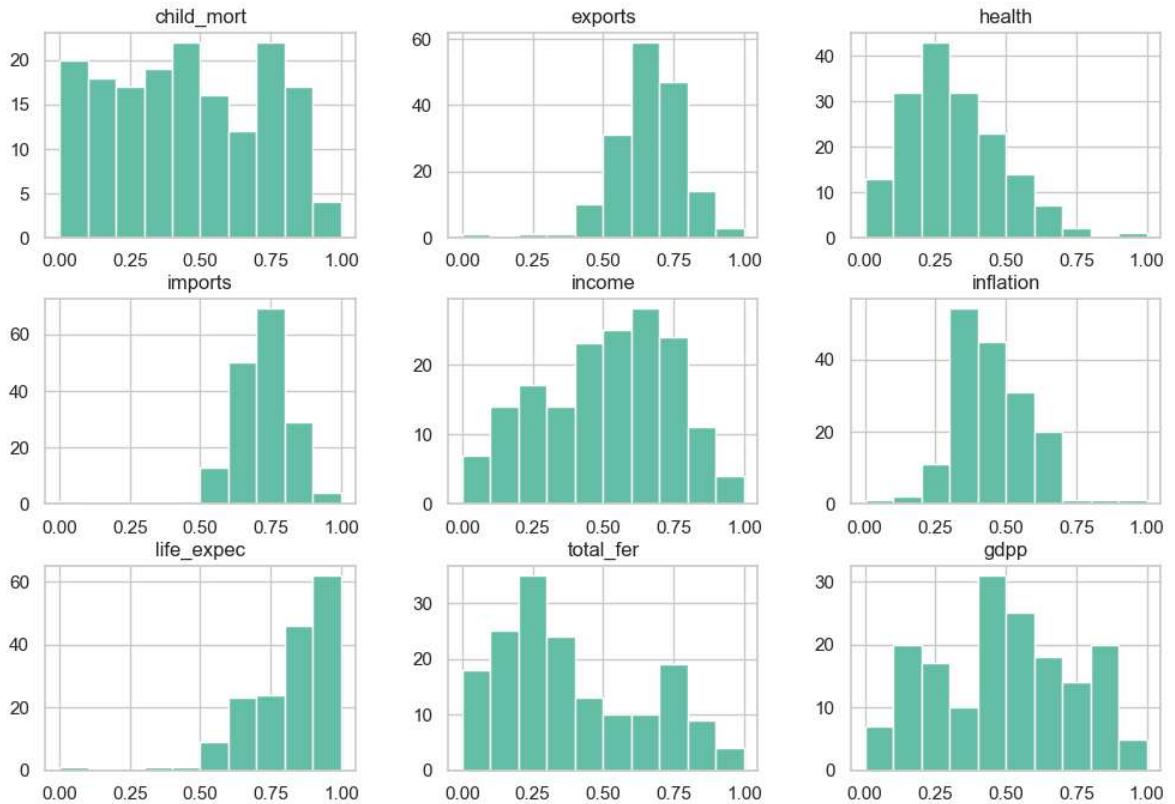
```
In [41]: df.drop('country', axis=1).hist(figsize=(12,8))
plt.suptitle("DADOS INICIAIS - Distribuição das Variáveis - Dataset de Países")
plt.show()
```

## DADOS INICIAIS - Distribuição das Variáveis - Dataset de Países



```
In [42]: X_scaled_df.hist(figsize=(12,8))
plt.suptitle("DADOS NORMALIZADOS - Distribuição das Variáveis - Dataset de Países")
plt.show()
```

## DADOS NORMALIZADOS - Distribuição das Variáveis - Dataset de Países



```
In [43]: number_of_clusters = 3
random_seed = 42
```

```

hierarquical = AgglomerativeClustering(
    n_clusters=number_of_clusters,
    linkage='ward',
    metric='euclidean'
)

X_scaled_df['cluster_hierarquical'] = hierarquical.fit_predict(X_scaled)
X_scaled_df['cluster_hierarquical']

```

Out[43]:

0	0
1	1
2	1
3	0
4	1
..	
162	0
163	1
164	0
165	0
166	0

Name: cluster\_hierarquical, Length: 167, dtype: int64

In [44]:

```

kmeans = KMeans(
    n_clusters=number_of_clusters,
    random_state=random_seed
)

X_scaled_df['cluster_kmeans'] = kmeans.fit_predict(X_scaled)
X_scaled_df['cluster_kmeans']

```

Out[44]:

0	2
1	0
2	0
3	2
4	1
..	
162	0
163	0
164	0
165	2
166	2

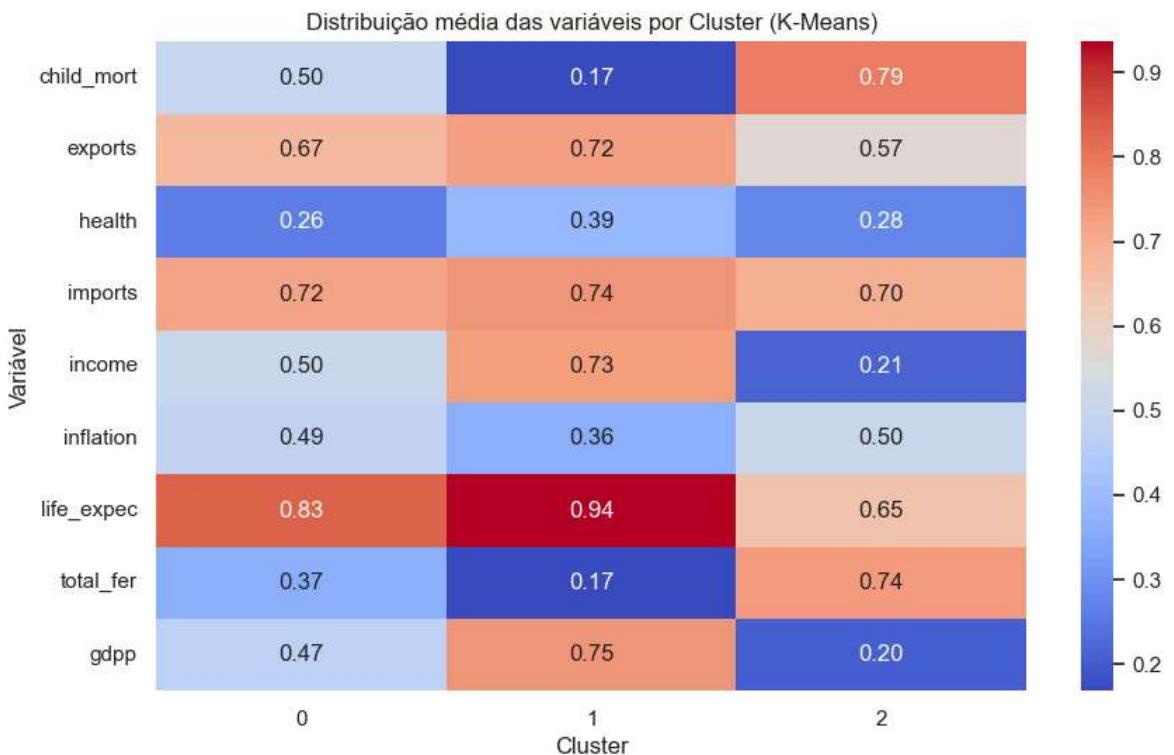
Name: cluster\_kmeans, Length: 167, dtype: int32

In [45]:

```

# Médias por cluster (distribuição central das dimensões)
means = X_scaled_df.drop('cluster_hierarquical', axis=1).groupby('cluster_kmeans')
plt.figure(figsize=(10,6))
sns.heatmap(means, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Distribuição média das variáveis por Cluster (K-Means)")
plt.ylabel("Variável")
plt.xlabel("Cluster")
plt.show()

```



```
In [46]: # Obter os centróides
centroids = kmeans.cluster_centers_

# Calcular a distância de cada ponto ao centróide do seu cluster
distances = []
for i, row in enumerate(X_scaled):
    cluster_label = X_scaled_df.loc[i, 'cluster_kmeans']
    centroid = centroids[cluster_label]
    distance = np.linalg.norm(row - centroid)
    distances.append(distance)

X_scaled_df['distance_to_centroid'] = distances

X_scaled_df['country'] = df['country']
# Encontrar o país mais representativo de cada cluster
representative_countries = X_scaled_df.loc[
    X_scaled_df.groupby('cluster_kmeans')['distance_to_centroid'].idxmin(),
    ['country', 'cluster_kmeans', 'distance_to_centroid']]
X_scaled_df = X_scaled_df.drop('country', axis=1)

print("Países que melhor representam cada cluster:\n")
print(representative_countries)
```

Países que melhor representam cada cluster:

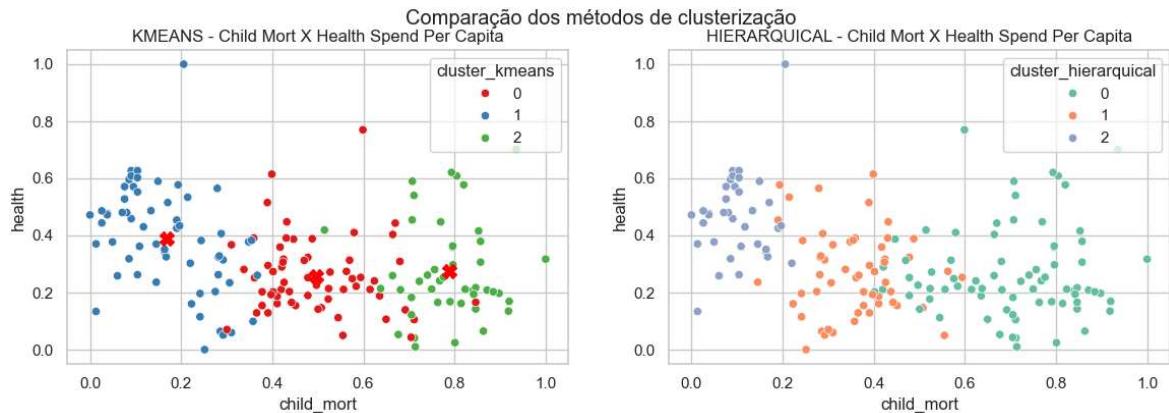
	country	cluster_kmeans	distance_to_centroid
118	Paraguay	0	0.128643
41	Croatia	1	0.145154
147	Tanzania	2	0.098071

```
In [47]: f, ax = plt.subplots(1, 2, figsize=(14,4))
x_column = 'child_mort'
y_column = 'health'
pos_for_x_column = 0
pos_for_y_column = 2
sns.scatterplot(data=X_scaled_df, x=x_column, y=y_column, hue=X_scaled_df['cluster_kmeans'],
                 palette='Set1', ax=ax[0])
```

```

ax[0].set_title('KMEANS - Child Mort X Health Spend Per Capita')
ax[0].plot(
    [x[pos_for_x_column] for x in kmeans.cluster_centers_],
    [x[pos_for_y_column] for x in kmeans.cluster_centers_],
    'X',
    color='red',
    markersize=10,
)
sns.scatterplot(data=X_scaled_df, x=x_column, y=y_column, hue=X_scaled_df['cluster_kmeans'],
                 palette='Set2', ax=ax[1])
ax[1].set_title('HIERARQUICAL - Child Mort X Health Spend Per Capita')
plt.suptitle('Comparação dos métodos de clusterização')
plt.show()

```



```

In [48]: print("\nDistribuição dos clusters (K-Means):")
print(X_scaled_df['cluster_kmeans'].value_counts())

print("\nDistribuição dos clusters (Hierárquico):")
print(X_scaled_df['cluster_hierarquical'].value_counts())

```

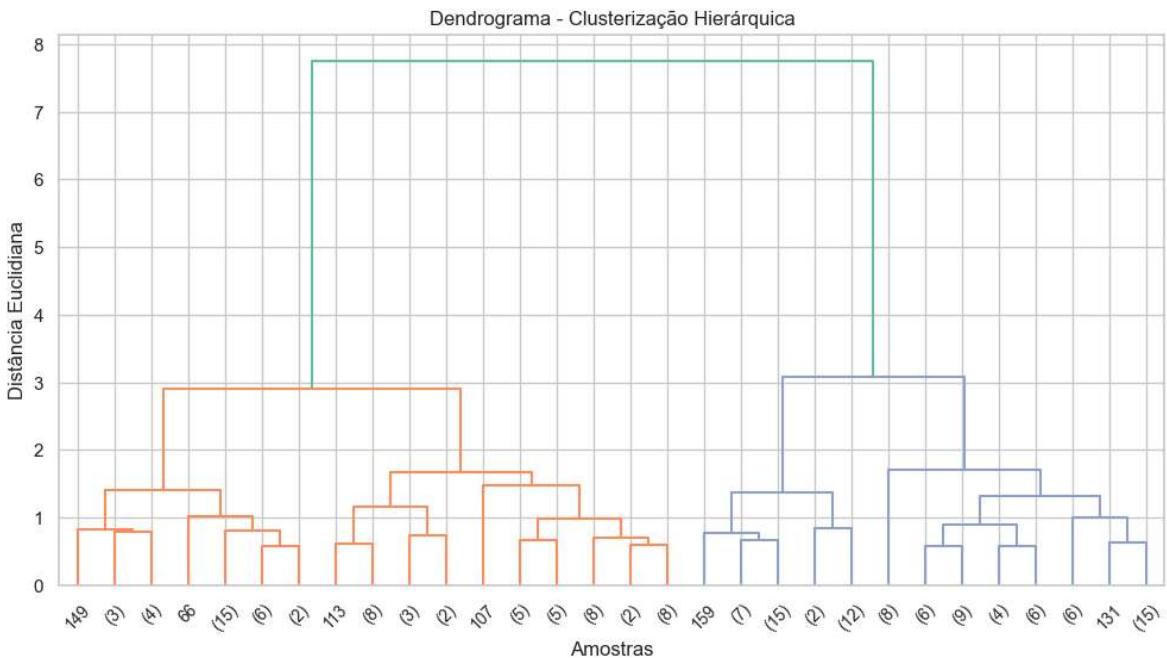
Distribuição dos clusters (K-Means):  
 cluster\_kmeans  
 0 61  
 1 61  
 2 45  
 Name: count, dtype: int64

Distribuição dos clusters (Hierárquico):  
 cluster\_hierarquical  
 0 75  
 1 55  
 2 37  
 Name: count, dtype: int64

```

In [49]: Z = linkage(X_scaled, method='ward')
plt.figure(figsize=(12, 6))
dendrogram(Z, truncate_mode='lastp', p=30, leaf_rotation=45., leaf_font_size=10.)
plt.title("Dendrograma - Clusterização Hierárquica")
plt.xlabel("Amostras")
plt.ylabel("Distância Euclidiana")
plt.show()

```



```
In [50]: pd.crosstab(X_scaled_df['cluster_kmeans'], X_scaled_df['cluster_hierarquical'])
```

```
Out[50]: cluster_hierarquical    0    1    2
```

		cluster_kmeans		
		0	30	31
cluster_kmeans		0	30	31
	0	30	31	0
	1	0	24	37
	2	45	0	0

```
In [51]: import numpy as np
import pandas as pd
from sklearn.metrics import pairwise_distances

def k_medoids(X, k, max_iter=100, random_state=42):
    np.random.seed(random_state)

    # 1 Inicialização: escolher aleatoriamente k pontos como medóides iniciais
    medoid_indices = np.random.choice(len(X), k, replace=False)
    medoids = X[medoid_indices]

    for it in range(max_iter):
        # 2 Atribuição: calcular distância de cada ponto a cada medóide
        distances = pairwise_distances(X, medoids)
        labels = np.argmin(distances, axis=1)

        new_medoids = np.copy(medoids)

        # 3 Atualização: para cada cluster, escolher novo medóide
        for i in range(k):
            cluster_points = X[labels == i]
            if len(cluster_points) == 0:
                continue

            # Calcular matriz de distâncias dentro do cluster
            intra_dist = pairwise_distances(cluster_points)
            total_dist = intra_dist.sum(axis=1)
```

```

# Escolher o ponto que minimiza a soma das distâncias
new_medoid = cluster_points[np.argmin(total_dist)]
new_medoids[i] = new_medoid

# 4 Verificar convergência
if np.all(medoids == new_medoids):
    print(f"Convergência alcançada na iteração {it+1}.")
    break

medoids = new_medoids

return labels, medoids

```

```

In [52]: X = X_scaled_df.drop('cluster_hierarquical', axis=1).drop('cluster_kmeans', axis=1)

medoids_labels, medoids = k_medoids(X, k=3)

df['cluster_medoide'] = medoids_labels
print(df.head())

```

Convergência alcançada na iteração 2.

	country	child_mort	exports	health	imports	income	\
0	Afghanistan	90.2	10.0	7.58	44.9	1610	
1	Albania	16.6	28.0	6.55	48.6	9930	
2	Algeria	27.3	38.4	4.17	31.4	12900	
3	Angola	119.0	62.3	2.85	42.9	5900	
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	

	inflation	life_expec	total_fer	gdpp	cluster_medoide
0	9.44	56.2	5.82	553	1
1	4.49	76.3	1.65	4090	2
2	16.10	76.5	2.89	4460	2
3	22.40	60.1	6.16	3530	1
4	1.44	76.8	2.13	12200	2

```

In [53]: # -----
# 5 Visualização: comparação Lado a Lado
# -----
sns.set(style="whitegrid", palette="Set2")
fig, axes = plt.subplots(1, 2, figsize=(12,5))

# Para facilitar a visualização, usaremos apenas duas variáveis:
x_idx, y_idx = 0, 2 # exemplo: child_mort vs health

# K-MEANS
axes[0].scatter(X_scaled[:, x_idx], X_scaled[:, y_idx], c=X_scaled_df['cluster_kmeans'])
axes[0].scatter(centroids[:, x_idx], centroids[:, y_idx], marker='*', c='black')
axes[0].set_title("K-MÉDIAS (Centróides em preto)")
axes[0].set_xlabel(df.drop(columns=['country']).columns[x_idx])
axes[0].set_ylabel(df.drop(columns=['country']).columns[y_idx])
axes[0].legend()

# K-MEDÓIDES
axes[1].scatter(X_scaled[:, x_idx], X_scaled[:, y_idx], c=medoids_labels, cmap='viridis')
axes[1].scatter(medoids[:, x_idx], medoids[:, y_idx], marker='P', c='black', s=100)
axes[1].set_title("K-MEDÓIDES (Medóides em preto)")
axes[1].set_xlabel(df.drop(columns=['country']).columns[x_idx])
axes[1].set_ylabel(df.drop(columns=['country']).columns[y_idx])
axes[1].legend()

```

```
plt.suptitle("Comparação visual entre K-Médias e K-Medóides", fontsize=13)
plt.tight_layout()
plt.show()
```

Comparação visual entre K-Médias e K-Medóides

