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.

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 Quantos países existem no dataset? 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? Realize o pré-processamento adequado dos dados.

```
from scipy import stats
from sklearn.preprocessing import StandardScaler
import numpy as np
from ydata_profiling import ProfileReport
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import AgglomerativeClustering
%matplotlib inline
import seaborn as sns; sns.set()  # for plot styling
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import pairwise_distances
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_samples, silhouette_score
```

```
data = 'C:/Users/laiss/OneDrive/Arquivos/analista de dados/aprendizado não - supervisionado/arquivo p o projeto/Country-data.csv'
```

```
df = pd.read_csv(data)
```

```
df.head()
```

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

```
df['country'].unique # tem 167 paises
```

```
<bound method Series.unique of 0              Afghanistan
1                  Albania
2                  Algeria
3                   Angola
4      Antigua and Barbuda
              ...
162                Vanuatu
163              Venezuela
164                Vietnam
165                  Yemen
166                 Zambia
Name: country, Length: 167, dtype: object>
```

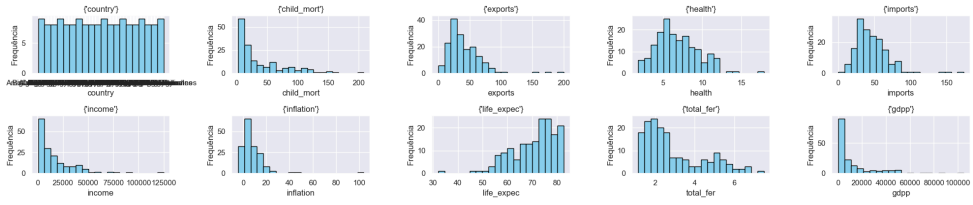
```
#df_profile = ProfileReport(df) # é possível verificar que o dataframe está limpo.
#df_profile
```

```
Variáveis = df.columns
Variáveis
```

```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

```
#com os histogramas por variável poderemos verificar os dados visualmente, e verificar a presença de outliers.
plt.figure(figsize=(20, 20))
for x, y in enumerate(Variáveis):
    plt.subplot(10, 5, x+1)
    plt.hist(df[y], bins=20, color='skyblue', edgecolor='black')
    plt.title({y})
    plt.xlabel(y)
    plt.ylabel('Frequência')

plt.tight_layout() # ajusta o layout dos subplots para não ter sobreposição
plt.show()
```




```
df_1 = df[df.columns[1:]] # devemos padronizar só as colunas numéricas.
df_1.head()
```

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

```
Scaler = StandardScaler().fit(df_1) # padronização dos dados pelo scaler
Scaler
```

```
StandardScaler()
```

```
X = Scaler.transform(df_1)# poderiamos usar a raiz quadrada para diminuir a interferÊncia de outliers se preciso fosse.
```

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

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

2- Para os resultados, do K-Médias: Interprete cada um dos clusters obtidos citando:

Qual a distribuição das dimensões em cada grupo; O país, de acordo com o algoritmo, melhor representa o seu agrupamento. Justifique.

KMeans

```
kmeans_ = KMeans(n_clusters= 3).fit(X) # o problema já diz quantos clusters
kmeans_
```

```
KMeans(n_clusters=3)
```

```
labels = kmeans_.labels_ # para verificar os grupos
labels
```

```
array([2, 0, 0, 2, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0,
       0, 1, 0, 2, 2, 0, 2, 1, 0, 2, 2, 0, 0, 0, 2, 2, 2, 0, 2, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 2, 2, 0, 0, 1, 1, 2, 2, 0, 1, 2, 1, 0, 0, 2, 2, 0,
       2, 0, 1, 0, 0, 0, 2, 1, 1, 1, 0, 1, 0, 0, 2, 2, 1, 0, 2, 0, 0, 2,
       2, 0, 0, 1, 0, 2, 2, 0, 0, 2, 1, 2, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
       1, 1, 2, 2, 1, 0, 2, 0, 0, 0, 0, 0, 1, 1, 0, 0, 2, 0, 0, 2, 0, 0,
       2, 1, 1, 1, 0, 2, 1, 1, 0, 0, 2, 0, 1, 1, 0, 2, 0, 2, 2, 0, 0, 0,
       0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2])
```

```
uniq, counts = np.unique(labels, return_counts= True) # dimensão de cada grupo
```

```
uniq
```

```
array([0, 1, 2])
```

```
counts
```

```
array([85, 36, 46], dtype=int64)
```

```
#calculando silhueta
```

```
y_kmeans = kmeans_.predict(X)
```

```
silhueta_kmeans =  silhouette_score(X, y_kmeans)
silhueta_kmeans
```

```
0.28437730261801497
```

```
df_1['cluster'] = labels
df_1
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster
0	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553	2
1	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090	0
2	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460	0
3	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530	2
4	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200	0
...	...	...	...	...	...	...	...	...	...	...
162	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970	0
163	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500	0
164	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310	0
165	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310	2
166	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1460	2

167 rows × 10 columns

```
group_0 = df_1.groupby('cluster').get_group(0)
group_0
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster
1	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090	0
2	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460	0
4	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200	0
5	14.5	18.9	8.10	16.0	18700	20.90	75.8	2.37	10300	0
6	18.1	20.8	4.40	45.3	6700	7.77	73.3	1.69	3220	0
...	...	...	...	...	...	...	...	...	...	...
160	10.6	26.3	8.35	25.4	17100	4.91	76.4	2.08	11900	0
161	36.3	31.7	5.81	28.5	4240	16.50	68.8	2.34	1380	0
162	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970	0
163	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500	0
164	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310	0

85 rows × 10 columns

```
group_1 = df_1.groupby('cluster').get_group(1)
group_1
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster
7	4.8	19.8	8.73	20.9	41400	1.160	82.0	1.93	51900	1
8	4.3	51.3	11.00	47.8	43200	0.873	80.5	1.44	46900	1
11	8.6	69.5	4.97	50.9	41100	7.440	76.0	2.16	20700	1
15	4.5	76.4	10.70	74.7	41100	1.880	80.0	1.86	44400	1
23	10.5	67.4	2.84	28.0	80600	16.700	77.1	1.84	35300	1
29	5.6	29.1	11.30	31.0	40700	2.870	81.3	1.63	47400	1
42	3.6	50.2	5.97	57.5	33900	2.010	79.9	1.42	30800	1
43	3.4	66.0	7.88	62.9	28300	-1.430	77.5	1.51	19800	1
44	4.1	50.5	11.40	43.6	44000	3.220	79.5	1.87	58000	1
53	3.0	38.7	8.95	37.4	39800	0.351	80.0	1.87	46200	1
54	4.2	26.8	11.90	28.1	36900	1.050	81.4	2.03	40600	1
58	4.2	42.3	11.60	37.1	40400	0.758	80.1	1.39	41800	1
60	3.9	22.1	10.30	30.7	28700	0.673	80.4	1.48	26900	1
68	2.6	53.4	9.40	43.3	38800	5.470	82.0	2.20	41900	1
73	4.2	103.0	9.19	86.5	45700	-3.220	80.4	2.05	48700	1
74	4.6	35.0	7.63	32.9	29600	1.770	81.4	3.03	30600	1
75	4.0	25.2	9.53	27.2	36200	0.319	81.7	1.46	35800	1
77	3.2	15.0	9.49	13.6	35800	-1.900	82.8	1.39	44500	1
82	10.8	66.7	2.63	30.4	75200	11.200	78.2	2.21	38500	1
91	2.8	175.0	7.77	142.0	91700	3.620	81.3	1.63	105000	1
98	6.8	153.0	8.65	154.0	28300	3.830	80.3	1.36	21100	1
110	4.5	72.0	11.90	63.6	45500	0.848	80.7	1.79	50300	1
111	6.2	30.3	10.10	28.0	32300	3.730	80.9	2.17	33700	1
114	3.2	39.7	9.48	28.5	62300	5.950	81.0	1.95	87800	1
122	3.9	29.9	11.00	37.4	27200	0.643	79.8	1.39	22500	1
123	9.0	62.3	1.81	23.8	125000	6.980	79.5	2.07	70300	1
133	2.8	200.0	3.96	174.0	72100	-0.046	82.7	1.15	46600	1
134	7.0	76.3	8.79	77.8	25200	0.485	75.5	1.43	16600	1
135	3.2	64.3	9.41	62.9	28700	-0.987	79.5	1.57	23400	1
138	4.1	49.4	6.93	46.2	30400	3.160	80.1	1.23	22100	1
139	3.8	25.5	9.54	26.8	32500	0.160	81.9	1.37	30700	1
144	3.0	46.2	9.63	40.7	42900	0.991	81.5	1.98	52100	1
145	4.5	64.0	11.50	53.3	55500	0.317	82.2	1.52	74600	1
157	8.6	77.7	3.66	63.6	57600	12.500	76.5	1.87	35000	1
158	5.2	28.2	9.64	30.8	36200	1.570	80.3	1.92	38900	1
159	7.3	12.4	17.90	15.8	49400	1.220	78.7	1.93	48400	1

```
group_2 = df_1.groupby('cluster').get_group(2)
group_2
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster
0	90.2	10.00	7.58	44.9	1610	9.440	56.2	5.82	553	2
3	119.0	62.30	2.85	42.9	5900	22.400	60.1	6.16	3530	2
17	111.0	23.80	4.10	37.2	1820	0.885	61.8	5.36	758	2
25	116.0	19.20	6.74	29.6	1430	6.810	57.9	5.87	575	2
26	93.6	8.92	11.60	39.2	764	12.300	57.7	6.26	231	2
28	108.0	22.20	5.13	27.0	2660	1.910	57.3	5.11	1310	2
31	149.0	11.80	3.98	26.5	888	2.010	47.5	5.21	446	2
32	150.0	36.80	4.53	43.5	1930	6.390	56.5	6.59	897	2
36	88.2	16.50	4.51	51.7	1410	3.870	65.9	4.75	769	2
37	116.0	41.10	7.91	49.6	609	20.800	57.5	6.54	334	2
38	63.9	85.10	2.46	54.7	5190	20.700	60.4	4.95	2740	2
40	111.0	50.60	5.30	43.3	2690	5.390	56.3	5.27	1220	2
49	111.0	85.80	4.48	58.9	33700	24.900	60.9	5.21	17100	2
50	55.2	4.79	2.66	23.3	1420	11.600	61.7	4.61	482	2
55	63.7	57.70	3.50	18.9	15400	16.600	62.9	4.08	8750	2
56	80.3	23.80	5.69	42.7	1660	4.300	65.5	5.71	562	2
59	74.7	29.50	5.22	45.9	3060	16.600	62.2	4.27	1310	2
63	109.0	30.30	4.93	43.2	1190	16.100	58.0	5.34	648	2
64	114.0	14.90	8.50	35.2	1390	2.970	55.6	5.05	547	2
66	208.0	15.30	6.91	64.7	1500	5.450	32.1	3.33	662	2
72	36.9	39.40	8.41	34.1	12700	16.600	67.2	4.56	4500	2
80	62.2	20.70	4.75	33.6	2480	2.090	62.8	4.37	967	2
81	62.7	13.30	11.30	79.9	1730	1.520	60.7	3.84	1490	2
84	78.9	35.40	4.47	49.3	3980	9.200	63.8	3.15	1140	2
87	99.7	39.40	11.10	101.0	2380	4.150	46.5	3.30	1170	2
88	89.3	19.10	11.80	92.6	700	5.470	60.8	5.02	327	2
93	62.2	25.00	3.77	43.0	1390	8.790	60.8	4.60	413	2
94	90.5	22.80	6.59	34.9	1030	12.100	53.1	5.31	459	2
97	137.0	22.80	4.98	35.1	1870	4.370	59.5	6.55	708	2
99	97.4	50.70	4.41	61.2	3320	18.900	68.2	4.98	1200	2
106	101.0	31.50	5.21	46.2	918	7.640	54.5	5.56	419	2
108	56.0	47.80	6.78	60.7	8460	3.560	58.6	3.60	5190	2
112	123.0	22.20	5.16	49.1	814	2.550	58.8	7.49	348	2
113	130.0	25.30	5.07	17.4	5150	104.000	60.5	5.84	2330	2
116	92.1	13.50	2.20	19.4	4280	10.900	65.3	3.85	1040	2
126	63.6	12.00	10.50	30.0	1350	2.610	64.6	4.51	563	2
129	66.8	24.90	5.66	40.3	2180	1.850	64.0	5.06	1000	2
132	160.0	16.80	13.10	34.5	1220	17.200	55.0	5.20	399	2
137	53.7	28.60	8.94	27.4	12000	6.350	54.3	2.59	7280	2
142	76.7	19.70	6.32	17.2	3370	19.600	66.3	4.88	1480	2
147	71.9	18.70	6.01	29.1	2090	9.250	59.3	5.43	702	2
149	62.6	2.20	9.12	27.8	1850	26.500	71.1	6.23	3600	2
150	90.3	40.20	7.65	57.3	1210	1.180	58.7	4.87	488	2
155	81.0	17.10	9.01	28.6	1540	10.600	56.8	6.15	595	2
165	56.3	30.00	5.18	34.4	4480	23.600	67.5	4.67	1310	2
166	83.1	37.00	5.89	30.9	3280	14.000	52.0	5.40	1460	2

```
mean_group_0 = group_0.mean()
mean_group_0
```

```
child_mort       22.287059
exports          40.283400
health            6.225647
imports          47.518422
income        12317.294118
inflation         7.616424
life_expec       72.629412
total_fer         2.314235
gdpp           6484.847059
cluster           0.000000
dtype: float64
```

```
mean_group_1 = group_1.mean()
mean_group_1
```

```
child_mort        5.000000
exports          58.738889
health            8.807778
imports          51.491667
income        45672.222222
inflation         2.671250
life_expec       80.127778
total_fer         1.752778
gdpp          42494.444444
cluster           1.000000
dtype: float64
```

```
mean_group_2 = group_2.mean()
mean_group_2
```

```
child_mort      93.841304
exports         28.837174
health           6.346957
imports         42.128261
income        3738.978261
inflation       12.087065
life_expec      59.232609
total_fer        5.054348
gdpp          1826.130435
cluster          2.000000
dtype: float64
```

```
dist_cent = kmeans_.fit_transform(X)**2
```

```
dist_cent
```

```
array([[ 32.32583067,   2.23926538,  20.69581242],
       [ 35.55139405,   8.59356967,   2.23919859],
       [ 21.91308229,   6.74120859,   4.92685423],
       [ 22.75650988,   6.88865391,  26.0012458 ],
       [ 39.71431679,  10.005383  ,   0.87967823],
       [ 20.84703053,  10.68975636,   6.00255236],
       [ 31.31536372,   7.57443982,   4.39777975],
       [ 51.32366147,  24.79875646,   6.90958518],
       [ 54.39706819,  26.13284816,   5.17407551],
       [ 24.31188817,   6.58045012,   4.50394561],
       [ 42.54746489,  11.37008712,   0.91344832],
       [ 36.97758243,  14.73179257,   2.01545973],
       [ 30.17704059,   5.33571221,   8.83084694],
       [ 41.2765053 ,  10.50134435,   0.76819053],
       [ 27.08498586,   9.91344793,   3.18679104],
       [ 55.82174297,  27.13341283,   6.21449104],
       [ 38.10420214,   6.62189294,   3.29204393],
       [ 38.81212322,   2.45728688,  19.37337126],
       [ 33.38066787,   5.52150307,   4.42194979],
       [ 27.11346354,   2.94723443,   5.36580684],
       [ 44.83507568,  14.12294012,   4.18493372],
       [ 30.22122619,   2.94042909,   7.5386512 ],
       [ 33.28321551,  10.84620124,   5.58898489],
       [ 41.72346723,  31.91750359,  13.85877102],
       [ 40.0476428 ,   9.82387565,   1.22428524],
       [ 34.76249532,   2.73996013,  22.64549402],
       [ 34.97472046,   6.66228769,  24.99337282],
       [ 34.98223674,   3.46812154,   5.70653847],
       [ 37.82520944,   2.29201535,  19.9572224 ],
       [ 51.29599382,  25.33641697,   6.33043943],
       [ 38.45014029,   6.12482508,   4.78166366],
       [ 45.91825353,   8.2544968 ,  34.26166229],
       [ 40.05029234,   6.28978754,  30.42469552],
       [ 32.40962969,  11.21847857,   1.30928778],
       [ 32.13655403,   8.9276664 ,   4.25163042],
       [ 36.18894294,   9.45057554,   4.46070641],
       [ 34.13026512,   1.32500456,  13.80230634],
       [ 25.11530407,   5.08622045,  25.90969356],
       [ 23.13194333,   7.4723288 ,  18.26274986],
       [ 38.07354622,  13.40074483,   3.51490124],
       [ 35.26323891,   2.25872741,  19.35375814],
       [ 40.96912406,  11.57983947,   0.91777504],
       [ 44.62305808,  17.5736414 ,   1.23430865],
       [ 48.25042635,  16.43577145,   1.28236125],
       [ 54.37316447,  28.33866483,   7.50835752],
       [ 31.68075813,   5.15967656,   3.37068656],
       [ 31.18457502,   6.63264458,   3.05899091],
       [ 26.24648841,   3.97502814,   6.10232546],
       [ 36.12252213,   6.47681798,   2.77305126],
       [ 23.45188219,  11.19417891,  20.65255121],
       [ 26.93443618,   3.53459162,  17.00356839],
       [ 43.7435049 ,  14.62155983,   2.08957957],
       [ 35.05058334,   5.37173552,   5.02687359],
       [ 49.93987475,  21.60844711,   3.61561184],
       [ 51.72531763,  23.44770357,   6.21425253],
       [ 21.25022788,   4.13073906,  11.61312203],
       [ 33.97153391,   1.23273695,  14.77213446],
       [ 33.65430818,   8.78200237,   3.53251453],
```

```
dist_cent_1 = dist_cent.sum(axis= 1)
```

```
dist_cent_1
```

```
array([ 55.26090846,  46.38416231,  33.5811451 ,  55.64640959,
        50.59937801,  37.53933926,  43.28758329,  83.03200312,
        85.70399186,  35.3962839 ,  54.83100033,  53.72483473,
        44.34359974,  52.54604018,  40.18522483,  89.16964685,
        48.01813901,  60.64278137,  43.32412073,  35.42650482,
        63.14294952,  40.70030649,  49.71840163,  87.49974184,
        51.09580369,  60.14794948,  66.63038097,  44.15689674,
        60.07444719,  82.96285021,  49.35662902,  88.43441262,
        76.7647754 ,  44.93739604,  45.31585084,  50.10022489,
        49.25757602,  56.11121808,  48.86702199,  54.98919229,
        56.87572445,  53.46673856,  63.43100813,  65.96855905,
        90.22018681,  40.21112125,  40.87621051,  36.32384201,
        45.37239137,  55.29861232,  47.4725962 ,  60.4546443 ,
        45.44919246,  75.1639337 ,  81.38727373,  36.99408898,
        49.97640532,  45.96882508,  82.93385054,  33.61309611,
        68.6485768 ,  47.67632835,  38.89659004,  46.78789253,
        63.6859672 ,  46.48915138, 141.05888954,  66.03101707,
        68.6152006 ,  38.92812943,  37.84570477,  37.51068328,
        32.21382443, 112.69438678,  58.06756695,  74.07206726,
        38.73181118,  87.76630246,  43.0575484 ,  30.98383885,
        47.24735618,  66.25344671,  86.48913521,  43.43973503,
        38.09373813,  55.65534065,  56.45692448,  90.84110776,
        74.95083939,  41.51594659,  55.9745254 , 313.23101627,
        50.87069997,  42.62815935,  50.85306511,  56.86318395,
        56.19434427,  72.20790422, 176.28766977,  39.88054635,
        51.78234971,  74.50013259,  59.03095436,  34.28757842,
        55.61177344,  45.21244217,  55.88102257,  59.2701042 ,
        45.37167344,  34.31384129,  97.99748613,  64.58253991,
        80.09844294, 218.4147173 , 127.65093884,  49.95061849,
        47.88453803,  57.94869249,  40.88767779,  44.41756402,
        42.09999905,  52.75213362,  68.3372207 , 190.41318629,
        46.57379166,  39.33053089,  55.06356216,  45.98125425,
        42.5668436 ,  48.39250053,  52.88704994,  93.5504416 ,
        84.70962667, 300.17468551,  69.48803806,  70.08006976,
```

```
       51.24361812,  48.29873818,  58.26305619,  71.606306  ,
       38.19161334,  44.50330161,  36.63935043,  37.83778429,
       82.79348881, 124.96585623,  35.36231312,  46.53708895,
       54.34015054,  47.53752454,  55.69101189,  46.46550027,
       46.56598556,  44.44234131,  52.64967987,  56.56620745,
       39.13213051,  71.48685418,  70.00728478, 135.88791346,
       47.84636779,  31.38575061,  44.22462468,  44.77044897,
       48.72934633,  29.91263248,  49.22001827])
```

```
df['cluster'] = labels
df['dist'] = dist_cent_1
df
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster	dist
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553	2	55.260908
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090	0	46.384162
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460	0	33.581145
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530	2	55.646410
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200	0	50.599378
...	...	...	...	...	...	...	...	...	...	...	...	...
162	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970	0	44.224625
163	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500	0	44.770449
164	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310	0	48.729346
165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310	2	29.912632
166	Zambia	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1460	2	49.220018

167 rows × 12 columns

```
Países = df.loc[df.groupby('cluster')['dist'].idxmin()]
Países
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster	dist
79	Kazakhstan	21.5	44.2	4.29	29.9	20100	19.50	68.4	2.60	9070	0	30.983839
11	Bahrain	8.6	69.5	4.97	50.9	41100	7.44	76.0	2.16	20700	1	53.724835
165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310	2	29.912632

são divididos em 3 grupos, como se fossem países desenvolvidos, em desenvolvimento e subdesenvolvidos. A menor distancia do centroide de cada grupo representa o país que melhor representa o grupo . Países = desenvolvidos = gropo 0 = Israel/ Países = em desenvolvimento = gropo 2 = suriname/Países = subdesenvolvidos= gropo 1 = Iraque.

Pronto, agora é analisar os dados e responder

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

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

Clusterização hieráquica

```
cluster_hier = AgglomerativeClustering(n_clusters=3).fit(X)
cluster_hier
```

```
AgglomerativeClustering
AgglomerativeClustering(n_clusters=3)
```

```
y_ch = cluster_hier.fit_predict(X)
y_ch
```

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

```
silhueta_hier = silhouette_score(X, y_ch)
silhueta_hier
```

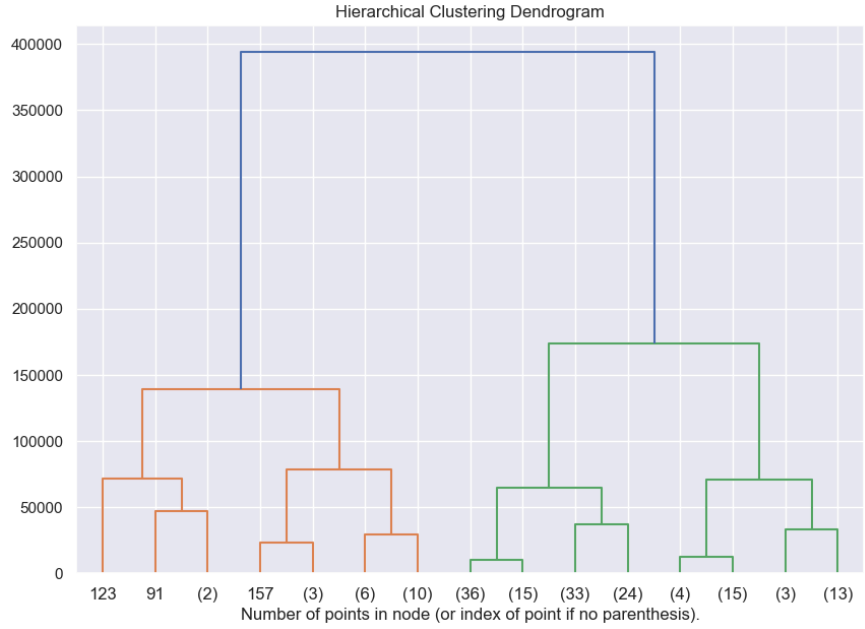
```
0.24563001303300652
```

```
# faz a silhueta para verificar a comparação. e a comparação do kmeans e clusterização hierarquica ver se os grupos sao iguais?
#plotando a cluterização
```

```
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
```

```
model = AgglomerativeClustering(distance_threshold=0, n_clusters=None)
model = model.fit(df_1)
plt.figure(figsize=(10, 7))
plt.title('Hierarchical Clustering Dendrogram')
# plot the top three levels of the dendrogram
plot_dendrogram(model, truncate_mode='level', p=3)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```




```
silhueta_hier
```

```
0.24563001303300652
```

```
silhueta_kmeans
```

```
0.28437730261801497
```

Um ótimo método para avaliar a qualidade da clusterização é o da silhoute_score, como vimos nos resultados, o método Kmeans, por estar mais perto de 1, é o mais indicado para a separação dos grupos.

Comparado os dois métodos, percebemos que são duas abordagens diferentes. O kmeans divide em quantidade específica de clusters, e tem por objetivo diminuir as distancias para o centro do cluster. Já a clusterização hieráquica os clusters são criados formando uma árvore de clusters, que se torna mais flexivel para a formação dos grupos do que o kmeans. Suas semelhanças são que podem ser aplicados a vários domínios, os dois tem o objetivos de separar em grupos similiares e os dois são técnicas de aprendizado de máquina não supervisionado.

Escolha de algoritmos

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

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.

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

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

1- Resposta: - verificando os clusters com a ajuda do método do cotovelo fazer a escolha do número de clusters - execução do k-means com o numero de cluster escolhido - Inicialize os centróides - Atribua pontos aos clusters - Atualize os centróides - Verifique a convergência, fazer esses processos até chegar na convergência

3- Resposta: Os centroides são calculados por média de todos os pontos atribuidos, então se tem um ponto muito distante, pode distorcer a média, e criando centroides que não representa a maioria dos pontos. O método também usa a distância euclidiada para verificar a similaridade dos pontos, então outliers pode influenciar a formação dos clusters.

4- Respostas: O algoritmo DBSCAN é mais sensível à presença de outliers por causa de sua abordagem baseada em densidade para a formação de clusters. O DBSCAN agrupa pontos com base na quantidade local dos pontos. Ele identifica regiões de alta densidade como clusters e pontos isolados como ruído (outliers). Como os outliers geralmente não têm uma densidade em comparação com os pontos em clusters, o DBSCAN tende a informar como ruído. Ele também não pede a especificação do número de clusters a priori. Ele identifica naturalmente os clusters com base na densidade dos pontos.Isso significa que os outliers não interferem na determinação do número de clusters, pois são tratados como ruído e não como clusters separados.