Projeto de Disciplina de Algoritmos de Clusterizacao [24E4_2]

Aluna: Rachel Reuters

PARTE 1- INFRAESTRUTURA

In [194…

```
import platform
import sys

print("Python version:", sys.version)
print("Platform:", platform.platform())
print("Architecture:", platform.architecture())
print("Processor:", platform.processor())
print("System:", platform.system())
print("Machine:", platform.machine())
print("Release:", platform.release())
print("Node:", platform.node())
```

```
Python version: 3.9.20 (main, Oct  3 2024, 07:38:01) [MSC v.1929 64 bit (AMD64)]
Platform: Windows-10-10.0.22621-SP0
Architecture: ('64bit', 'WindowsPE')
Processor: Intel64 Family 6 Model 158 Stepping 10, GenuineIntel
System: Windows
Machine: AMD64
Release: 10
Node: DESKTOP-4U6C9N4
```

In [195…

```
import subprocess

def get_conda_info():
    result = subprocess.run(['conda', 'info'], stdout=subprocess.PIPE)
    print(result.stdout.decode('utf-8'))

print("Conda Info:")
get_conda_info()
```

```
Conda Info:

     active environment : py39
    active env location : C:\Users\belch\anaconda3\envs\py39
            shell level : 1
       user config file : C:\Users\belch\.condarc
 populated config files : C:\Users\belch\.condarc
          conda version : 24.5.0
    conda-build version : 24.5.1
         python version : 3.12.4.final.0
                 solver : libmamba (default)
       virtual packages : __archspec=1=skylake
                          __conda=24.5.0=0
                          __cuda=12.6=0
                          __win=0=0
       base environment : C:\Users\belch\anaconda3  (writable)
      conda av data dir : C:\Users\belch\anaconda3\etc\conda
  conda av metadata url : None
           channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/win-64
                          https://repo.anaconda.com/pkgs/r/noarch
                          https://repo.anaconda.com/pkgs/msys2/win-64
                          https://repo.anaconda.com/pkgs/msys2/noarch
                          https://conda.anaconda.org/conda-forge/win-64
                          https://conda.anaconda.org/conda-forge/noarch
          package cache : C:\Users\belch\anaconda3\pkgs
                          C:\Users\belch\.conda\pkgs
                          C:\Users\belch\AppData\Local\conda\conda\pkgs
       envs directories : C:\Users\belch\anaconda3\envs
                          C:\Users\belch\.conda\envs
                          C:\Users\belch\AppData\Local\conda\conda\envs
               platform : win-64
             user-agent : conda/24.5.0 requests/2.32.2 CPython/3.12.4 Windows/11 Win
dows/10.0.22631 solver/libmamba conda-libmamba-solver/24.1.0 libmambapy/1.5.8 aau/0.
4.4 c/. s/. e/.
          administrator : False
             netrc file : None
           offline mode : False
```

In [196…

```python
with open('requirements.txt', 'r') as file:
    requirements = file.read()

print(requirements)
```

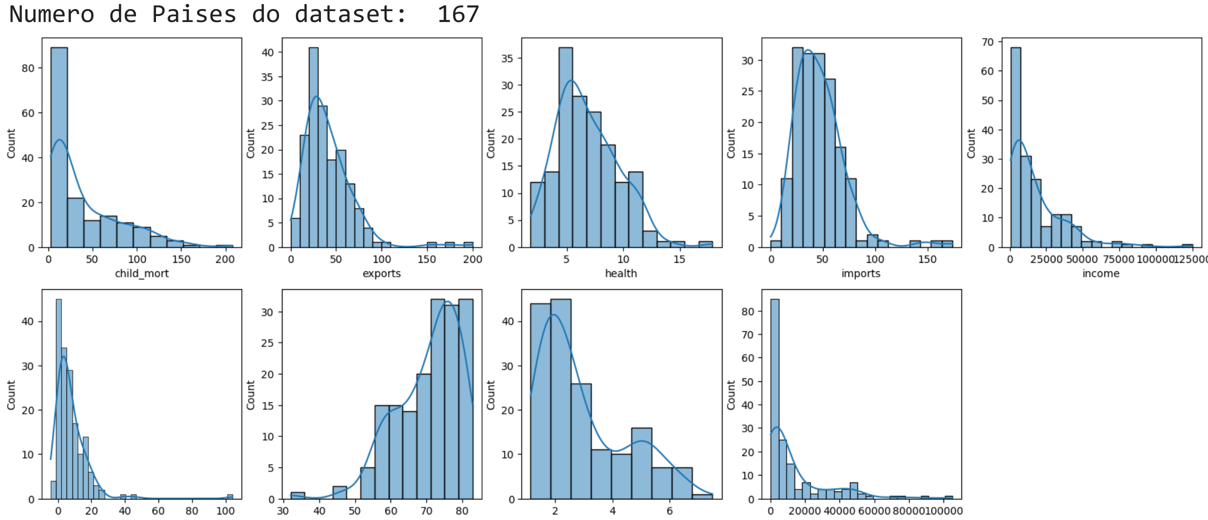
```
matplotlib==3.9.2
numpy==1.26.4
pandas==2.2.1
scikit-learn==1.5.2
scipy==1.13.1
seaborn==0.13.2
scikit-learn-extra==0.3.0
```

PARTE 2 - ESCOLHA DA BASE DE DADOS

In [197…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot  as plt
import seaborn as sns

#PARTE 2.1
#Baixe os dados disponibilizados na plataforma Kaggle
dataset = pd.read_csv('country-data.csv', sep=',', decimal='.')

nome_paises = dataset['country'].to_list()

#PARTE 2.2
#Quantos países existem no dataset?

print("Numero de Paises do dataset: ", dataset.shape[0])

#Remover a coluna de nome
dataset.drop(dataset.columns[0], axis=1,inplace=True )

#PARTE 2.3
#Mostre através de gráficos a faixa dinâmica das variáveis que serão usadas nas tar
# Analise os resultados mostrados. O que deve ser feito com os dados antes da etapa

#Podemos observar que temos uma coluna de string referente ao nome do pais.
#Acredito que podemos desconsiderar a coluna de nome do pais. Pois ela so acaba ser
#Para os demais atributos, precisamos verificar se existem valores faltantes.
#Tambem 'e interessante verificar se existem valores duplicados (para esse caso nao
#As outras 9 colunas todas precisam ser normalizadas pois estao com diferentes faix
#'e necessario  que todos estejam na mesma faixa de valores, para que nao exista di


plt.figure(figsize=(20, 8))

for index, value in enumerate(dataset.columns):
    plt.subplot(2, 5,index+1)
    sns.histplot(dataset[value], kde=True)
```

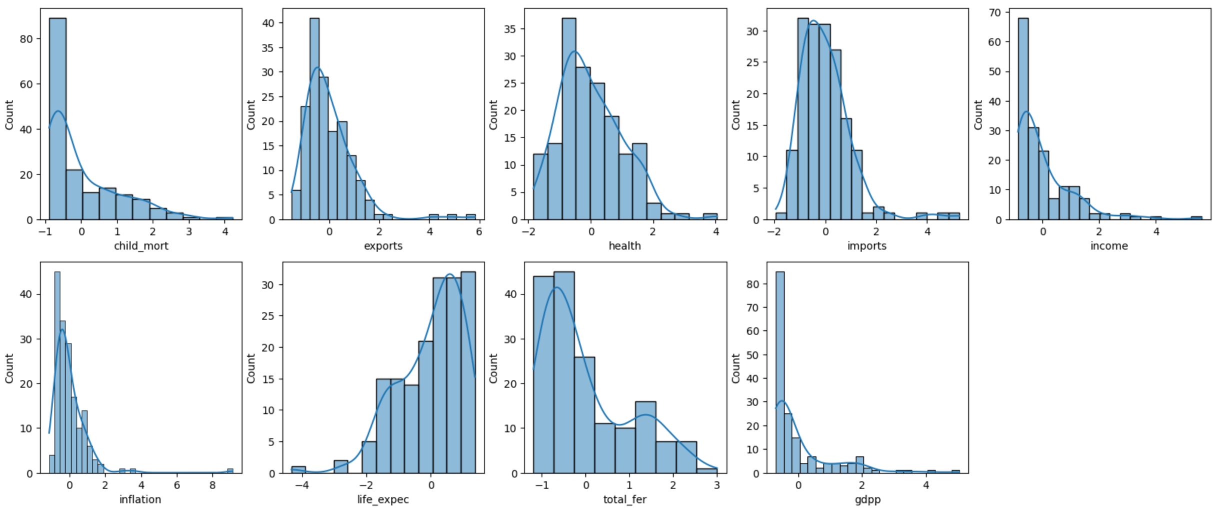
```
Numero de Paises do dataset:  167
```



In [198…

```python
#PARTE 2.4
#Realize o pré-processamento adequado dos dados.

#Primeiro verificar se tem nulos
print("Dados faltantes:")
print(dataset.isna().sum())

#Depois verificar se tem duplicados
print("Dados duplicados:")
print(dataset.duplicated().sum())


print("Verificando os tipos de cada atributo:")
print(dataset.dtypes)
#Em seguida 'e necessario normalizar os dados para que todos os atributos fiquem na

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(dataset)
dataset_transformed = scaler.transform(dataset)

dataframe_transformed  = pd.DataFrame(dataset_transformed)
#Plotando nova distribuicao
plt.figure(figsize=(20, 8))

for index, value in enumerate(dataset.columns):
    plt.subplot(2, 5,index+1)
    sns.histplot(dataframe_transformed[index], kde=True )
    plt.xlabel(dataset.columns[index])
```

```
Dados faltantes:
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
Dados duplicados:
0
Verificando os tipos de cada atributo:
child_mort    float64
exports       float64
health        float64
imports       float64
income          int64
inflation     float64
life_expec    float64
total_fer     float64
gdpp            int64
dtype: object
```



PARTE 3 - CLUSTERIZAÇÃO

In [199…

```python
#Parte 3.1.a
#Agrupamento em 3 grupos  usando K-MEDIAS
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3,random_state=22)
model= kmeans.fit(dataset_transformed)
y_kmeans = kmeans.predict(dataset_transformed)

#Parte 3.2
#Interpretacao e distribuicao dos resultados
resultado_k_means = pd.DataFrame(y_kmeans)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
resultado_k_means = resultado_k_means.rename(columns={0: 'Categoria'})
resultado_k_means['Pais'] = nome_paises

resultado_k_means = resultado_k_means.sort_values(by='Categoria')

print(resultado_k_means)

#Como ja sabemos como alguns paises se classificam,
#podemos chegar a conclusao que as categorias estao separando em nivel de desenvolv
#Grupo 0 = em desenvolvimento
#Grupo 1 = sub-desenvolvidos
#Grupo 2 = desenvolvidos

#Para verificar o pais que mais representa o agrupamento podemos
#verificar a distancia de cada pais para o centro do cluster
#o que tiver a menor distancia e o que mais representa
distances = kmeans.transform(dataset_transformed)
distancias = pd.DataFrame(distances)

mais_proximos = []
mais_proximos = pd.DataFrame(mais_proximos)
mais_proximos['Categoria']=list(set(resultado_k_means['Categoria']))

coluna =0
```

```python
for center in range(len(kmeans.cluster_centers_)):
    distancia_coluna = distancias[coluna].tolist()
    _index = np.argmin(distancia_coluna)
    mais_proximos.at[coluna, 'Pais que + representa'] = nome_paises[_index]
    mais_proximos.at[coluna, 'Distancia'] = distancias[coluna][_index]
    coluna+=1

print("Pontos mais próximos dos centroides:\n", mais_proximos)
```

	Categoria	Pais
83	0	Kyrgyz Republic
119	0	Peru
67	0	Hungary
120	0	Philippines
65	0	Guyana
121	0	Poland
124	0	Romania
62	0	Guatemala
61	0	Grenada
125	0	Russia
127	0	Samoa
57	0	Georgia
128	0	Saudi Arabia
130	0	Serbia
131	0	Seychelles
52	0	Fiji
51	0	Estonia
100	0	Mauritius
48	0	El Salvador
47	0	Egypt
69	0	India
46	0	Ecuador
70	0	Indonesia
118	0	Paraguay
96	0	Maldives
95	0	Malaysia
101	0	Micronesia, Fed. Sts.
102	0	Moldova
92	0	Macedonia, FYR
103	0	Mongolia
90	0	Lithuania
89	0	Libya
104	0	Montenegro
105	0	Morocco
86	0	Lebanon
85	0	Latvia
107	0	Myanmar
109	0	Nepal
79	0	Kazakhstan
78	0	Jordan
76	0	Jamaica
115	0	Oman
117	0	Panama
71	0	Iran
45	0	Dominican Republic
136	0	Solomon Islands
140	0	Sri Lanka
19	0	Bolivia
18	0	Bhutan
156	0	Ukraine
16	0	Belize
160	0	Uruguay
14	0	Belarus
13	0	Barbados
12	0	Bangladesh

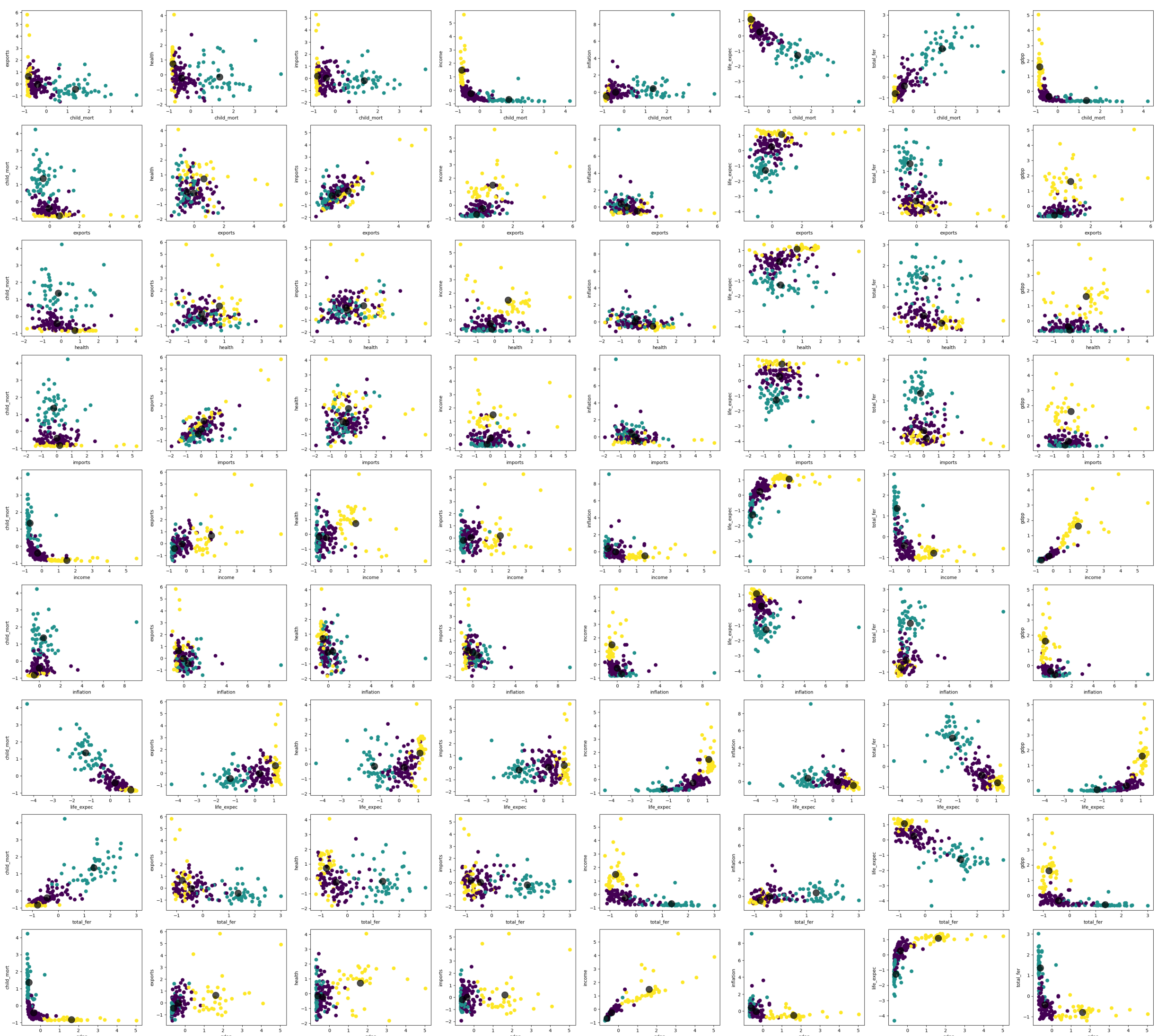
161	0	Uzbekistan
10	0	Bahamas
9	0	Azerbaijan
162	0	Vanuatu
163	0	Venezuela
6	0	Armenia
5	0	Argentina
4	0	Antigua and Barbuda
164	0	Vietnam
2	0	Algeria
1	0	Albania
22	0	Brazil
154	0	Turkmenistan
20	0	Bosnia and Herzegovina
153	0	Turkey
33	0	Chile
24	0	Bulgaria
148	0	Thailand
146	0	Tajikistan
143	0	Suriname
34	0	China
39	0	Costa Rica
30	0	Cape Verde
151	0	Tonga
27	0	Cambodia
41	0	Croatia
141	0	St. Vincent and the Grenadines
152	0	Tunisia
35	0	Colombia
142	1	Sudan
147	1	Tanzania
129	1	Senegal
132	1	Sierra Leone
126	1	Rwanda
108	1	Namibia
112	1	Niger
149	1	Timor-Leste
113	1	Nigeria
150	1	Togo
116	1	Pakistan
137	1	South Africa
155	1	Uganda
106	1	Mozambique
99	1	Mauritania
0	1	Afghanistan
97	1	Mali
59	1	Ghana
56	1	Gambia
55	1	Gabon
50	1	Eritrea
49	1	Equatorial Guinea
40	1	Cote d'Ivoire
38	1	Congo, Rep.
37	1	Congo, Dem. Rep.
36	1	Comoros
32	1	Chad

31	1	Central African Republic
28	1	Cameroon
26	1	Burundi
25	1	Burkina Faso
21	1	Botswana
17	1	Benin
3	1	Angola
63	1	Guinea
64	1	Guinea-Bissau
166	1	Zambia
94	1	Malawi
80	1	Kenya
72	1	Iraq
81	1	Kiribati
165	1	Yemen
84	1	Lao
93	1	Madagascar
66	1	Haiti
87	1	Lesotho
88	1	Liberia
68	2	Iceland
110	2	Netherlands
82	2	Kuwait
29	2	Canada
91	2	Luxembourg
157	2	United Arab Emirates
158	2	United Kingdom
159	2	United States
15	2	Belgium
11	2	Bahrain
8	2	Austria
7	2	Australia
23	2	Brunei
145	2	Switzerland
111	2	New Zealand
42	2	Cyprus
122	2	Portugal
123	2	Qatar
60	2	Greece
73	2	Ireland
58	2	Germany
74	2	Israel
75	2	Italy
54	2	France
144	2	Sweden
53	2	Finland
98	2	Malta
134	2	Slovak Republic
135	2	Slovenia
77	2	Japan
44	2	Denmark
138	2	South Korea
139	2	Spain
43	2	Czech Republic
114	2	Norway
133	2	Singapore

```
Pontos mais próximos dos centroides:
    Categoria Pais que + representa  Distancia
0           0              Jamaica   0.734379
1           1               Guinea   0.829088
2           2              Iceland   0.731764
```

In [200…

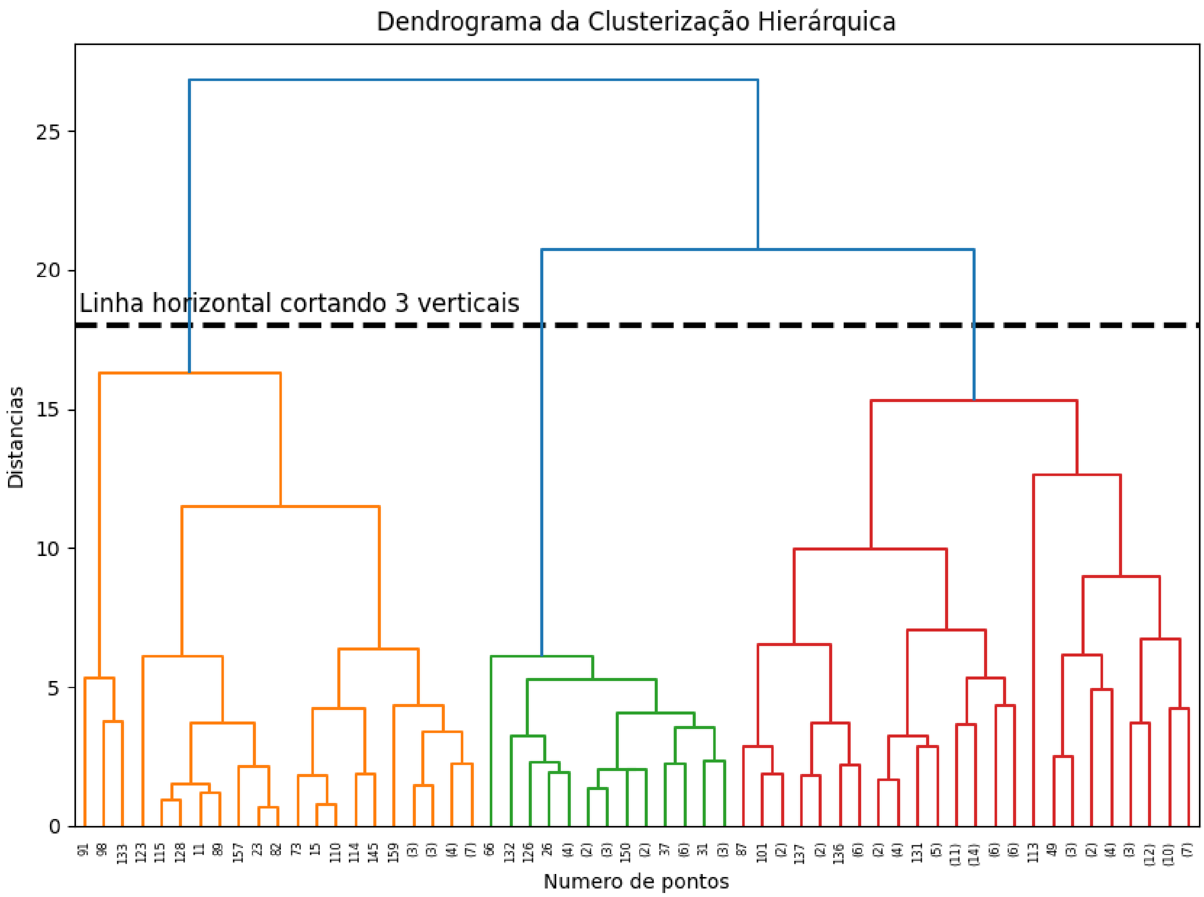
```
centers = kmeans.cluster_centers_

plt.figure(figsize=(50, 40))
index=0
for col1 in dataframe_transformed.columns:
    for col2 in dataframe_transformed.columns:
        if col1 != col2:
            index+=1
            plt.subplot(9,9,index)
            plt.scatter(dataframe_transformed[col1], dataframe_transformed[col2], c
            plt.xlabel(dataset.columns[col1])
            plt.ylabel(dataset.columns[col2])
            plt.scatter(centers[:, col1], centers[:, col2], c='black', s=200, alpha
    index+=1
```



In [201…

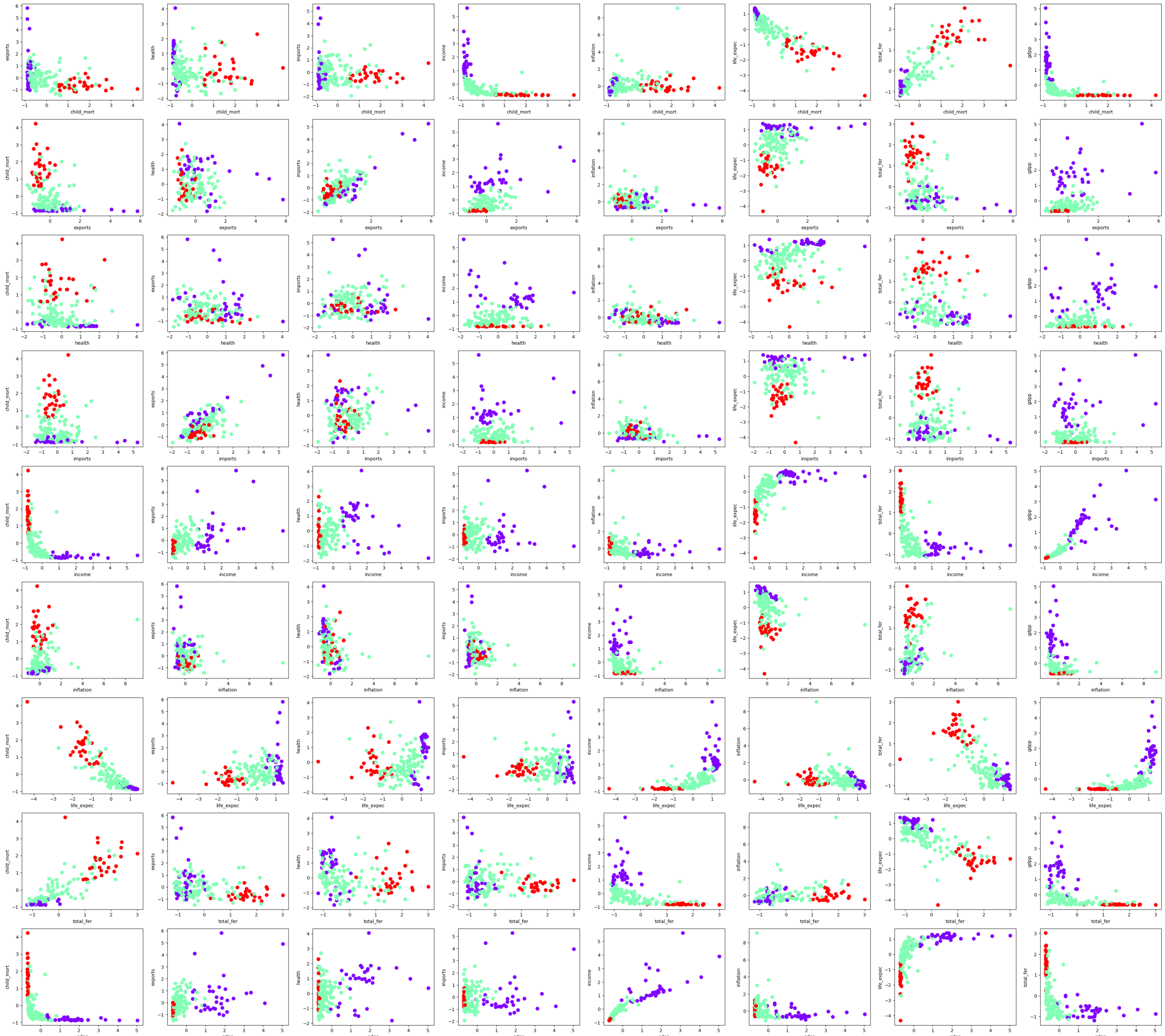
```
#Parte 3.1.b
#Agrupamento em 3 grupos  usando Clusterizacao Hierarquica
```

```python
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram


def plot_dendrogram(model, **kwargs):
    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
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    return dendrogram(linkage_matrix, **kwargs)

#Parte 3.3  Para os resultados da Clusterização Hierárquica, apresente o dendograma
cluster = AgglomerativeClustering(distance_threshold=0, n_clusters=None)
model = cluster.fit(dataset_transformed)

plt.figure(figsize=(10, 7))
plt.title('Dendrograma da Clusterização Hierárquica')
plt.ylabel('Distancias')
plt.hlines(y=18,xmin=0,xmax=2000,lw=3,linestyles='--',colors='black' )
plt.text(x=2,y=18.5,s='Linha horizontal cortando 3 verticais',fontsize=12)
#plt.grid(True)
dendro=plot_dendrogram(model, truncate_mode='level', p=6)
plt.xlabel("Numero de pontos")
plt.show()
```



In [202…

```
#Resultado da clusterizacao
cluster = AgglomerativeClustering(n_clusters=3, metric='euclidean', linkage='ward',

cluster_result = cluster.fit_predict(dataset_transformed)

plt.figure(figsize=(50, 40))
index=0
for col1 in dataframe_transformed.columns:
    for col2 in dataframe_transformed.columns:
        if col1 != col2:
            index+=1
            plt.subplot(9,9,index)
            plt.scatter(dataframe_transformed[col1], dataframe_transformed[col2], c
            plt.xlabel(dataset.columns[col1])
            plt.ylabel(dataset.columns[col2])
    index+=1
```



In [203…

```
#3.3 Para os resultados da Clusterização Hierárquica interprete os resultados
resultado_hierarquico = pd.DataFrame(cluster_result)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
resultado_hierarquico = resultado_hierarquico.rename(columns={0: 'Categoria'})
resultado_hierarquico['Pais'] = nome_paises

resultado_hierarquico = resultado_hierarquico.sort_values(by='Categoria')

print(resultado_hierarquico)

#Como ja sabemos como alguns paises se classificam,
#podemos chegar a conclusao que as categorias estao separando em nivel de desenvolv
#Grupo 0 = desenvolvidos
#Grupo 1 = em desenvolvimento
#Grupo 2 = sub-desenvolvidos
```

	Categoria	Pais
111	0	New Zealand
89	0	Libya
91	0	Luxembourg
29	0	Canada
60	0	Greece
159	0	United States
115	0	Oman
114	0	Norway
23	0	Brunei
58	0	Germany
157	0	United Arab Emirates
77	0	Japan
110	0	Netherlands
54	0	France
44	0	Denmark
53	0	Finland
98	0	Malta
158	0	United Kingdom
123	0	Qatar
122	0	Portugal
133	0	Singapore
145	0	Switzerland
11	0	Bahrain
82	0	Kuwait
73	0	Ireland
68	0	Iceland
8	0	Austria
7	0	Australia
74	0	Israel
144	0	Sweden
75	0	Italy
128	0	Saudi Arabia
139	0	Spain
15	0	Belgium
96	1	Maldives
81	1	Kiribati
95	1	Malaysia
148	1	Thailand
87	1	Lesotho
84	1	Lao
149	1	Timor-Leste
85	1	Latvia
92	1	Macedonia, FYR
90	1	Lithuania
99	1	Mauritania
86	1	Lebanon
88	1	Liberia
165	1	Yemen
100	1	Mauritius
83	1	Kyrgyz Republic
102	1	Moldova
140	1	Sri Lanka
138	1	South Korea
137	1	South Africa
136	1	Solomon Islands

135	1	Slovenia
134	1	Slovak Republic
143	1	Suriname
131	1	Seychelles
130	1	Serbia
127	1	Samoa
125	1	Russia
124	1	Romania
101	1	Micronesia, Fed. Sts.
121	1	Poland
119	1	Peru
118	1	Paraguay
117	1	Panama
116	1	Pakistan
113	1	Nigeria
146	1	Tajikistan
109	1	Nepal
108	1	Namibia
107	1	Myanmar
105	1	Morocco
104	1	Montenegro
103	1	Mongolia
120	1	Philippines
151	1	Tonga
71	1	Iran
78	1	Jordan
35	1	Colombia
34	1	China
33	1	Chile
160	1	Uruguay
30	1	Cape Verde
161	1	Uzbekistan
27	1	Cambodia
162	1	Vanuatu
163	1	Venezuela
24	1	Bulgaria
22	1	Brazil
21	1	Botswana
20	1	Bosnia and Herzegovina
38	1	Congo, Rep.
19	1	Bolivia
164	1	Vietnam
16	1	Belize
14	1	Belarus
13	1	Barbados
12	1	Bangladesh
10	1	Bahamas
9	1	Azerbaijan
6	1	Armenia
5	1	Argentina
4	1	Antigua and Barbuda
3	1	Angola
2	1	Algeria
1	1	Albania
18	1	Bhutan
79	1	Kazakhstan

39	1	Costa Rica
41	1	Croatia
76	1	Jamaica
72	1	Iraq
141	1	St. Vincent and the Grenadines
70	1	Indonesia
69	1	India
67	1	Hungary
152	1	Tunisia
65	1	Guyana
153	1	Turkey
154	1	Turkmenistan
62	1	Guatemala
61	1	Grenada
156	1	Ukraine
59	1	Ghana
55	1	Gabon
52	1	Fiji
51	1	Estonia
50	1	Eritrea
49	1	Equatorial Guinea
48	1	El Salvador
47	1	Egypt
46	1	Ecuador
45	1	Dominican Republic
43	1	Czech Republic
42	1	Cyprus
57	1	Georgia
142	1	Sudan
155	2	Uganda
150	2	Togo
147	2	Tanzania
0	2	Afghanistan
129	2	Senegal
17	2	Benin
25	2	Burkina Faso
26	2	Burundi
28	2	Cameroon
31	2	Central African Republic
32	2	Chad
36	2	Comoros
37	2	Congo, Dem. Rep.
40	2	Cote d'Ivoire
132	2	Sierra Leone
56	2	Gambia
64	2	Guinea-Bissau
66	2	Haiti
80	2	Kenya
93	2	Madagascar
94	2	Malawi
97	2	Mali
106	2	Mozambique
112	2	Niger
126	2	Rwanda
63	2	Guinea
166	2	Zambia

In [204…

```
#3.4 - Compare os dois resultados, aponte as semelhanças e diferenças e interprete.
# Para comparar corretamente, temos que colocar os nomes das categorias de forma eq
#Atualmente temos:
#Hierarquico
#Grupo 0 = desenvolvidos
#Grupo 1 = em desenvolvimento
#Grupo 2 = sub-desenvolvidos

#Kmeans
#Grupo 0 = em desenvolvimento
#Grupo 1 = sub-desenvolvidos
#Grupo 2 = desenvolvidos

resultado_hierarquico = resultado_hierarquico.replace({0: 2, 1: 0, 2: 1})


resultado_comparativo = resultado_hierarquico.sort_values(by='Pais')

resultado_comparativo['K_means'] = resultado_k_means.sort_values(by='Pais')['Catego

resultado_comparativo.rename(columns={'Categoria':'Clusterizacao Hierarquica'},inpl

resultado_comparativo = resultado_comparativo[['Pais', 'K_means', 'Clusterizacao Hi

resultado_comparativo
```

Out[204]:

	Pais	K_means	Clusterizacao Hierarquica
0	Afghanistan	1	1
1	Albania	0	0
2	Algeria	0	0
3	Angola	1	0
4	Antigua and Barbuda	0	0
5	Argentina	0	0
6	Armenia	0	0
7	Australia	2	2
8	Austria	2	2
9	Azerbaijan	0	0
10	Bahamas	0	0
11	Bahrain	2	2
12	Bangladesh	0	0
13	Barbados	0	0
14	Belarus	0	0
15	Belgium	2	2
16	Belize	0	0
17	Benin	1	1
18	Bhutan	0	0
19	Bolivia	0	0
20	Bosnia and Herzegovina	0	0
21	Botswana	1	0
22	Brazil	0	0
23	Brunei	2	2
24	Bulgaria	0	0
25	Burkina Faso	1	1
26	Burundi	1	1
27	Cambodia	0	0
28	Cameroon	1	1
29	Canada	2	2
30	Cape Verde	0	0
31	Central African Republic	1	1

	Pais	K_means	Clusterizacao Hierarquica
32	Chad	1	1
33	Chile	0	0
34	China	0	0
35	Colombia	0	0
36	Comoros	1	1
37	Congo, Dem. Rep.	1	1
38	Congo, Rep.	1	0
39	Costa Rica	0	0
40	Cote d'Ivoire	1	1
41	Croatia	0	0
42	Cyprus	2	0
43	Czech Republic	2	0
44	Denmark	2	2
45	Dominican Republic	0	0
46	Ecuador	0	0
47	Egypt	0	0
48	El Salvador	0	0
49	Equatorial Guinea	1	0
50	Eritrea	1	0
51	Estonia	0	0
52	Fiji	0	0
53	Finland	2	2
54	France	2	2
55	Gabon	1	0
56	Gambia	1	1
57	Georgia	0	0
58	Germany	2	2
59	Ghana	1	0
60	Greece	2	2
61	Grenada	0	0
62	Guatemala	0	0
63	Guinea	1	1

	Pais	K_means	Clusterizacao Hierarquica
64	Guinea-Bissau	1	1
65	Guyana	0	0
66	Haiti	1	1
67	Hungary	0	0
68	Iceland	2	2
69	India	0	0
70	Indonesia	0	0
71	Iran	0	0
72	Iraq	1	0
73	Ireland	2	2
74	Israel	2	2
75	Italy	2	2
76	Jamaica	0	0
77	Japan	2	2
78	Jordan	0	0
79	Kazakhstan	0	0
80	Kenya	1	1
81	Kiribati	1	0
82	Kuwait	2	2
83	Kyrgyz Republic	0	0
84	Lao	1	0
85	Latvia	0	0
86	Lebanon	0	0
87	Lesotho	1	0
88	Liberia	1	0
89	Libya	0	2
90	Lithuania	0	0
91	Luxembourg	2	2
92	Macedonia, FYR	0	0
93	Madagascar	1	1
94	Malawi	1	1
95	Malaysia	0	0

	Pais	K_means	Clusterizacao Hierarquica
96	Maldives	0	0
97	Mali	1	1
98	Malta	2	2
99	Mauritania	1	0
100	Mauritius	0	0
101	Micronesia, Fed. Sts.	0	0
102	Moldova	0	0
103	Mongolia	0	0
104	Montenegro	0	0
105	Morocco	0	0
106	Mozambique	1	1
107	Myanmar	0	0
108	Namibia	1	0
109	Nepal	0	0
110	Netherlands	2	2
111	New Zealand	2	2
112	Niger	1	1
113	Nigeria	1	0
114	Norway	2	2
115	Oman	0	2
116	Pakistan	1	0
117	Panama	0	0
118	Paraguay	0	0
119	Peru	0	0
120	Philippines	0	0
121	Poland	0	0
122	Portugal	2	2
123	Qatar	2	2
124	Romania	0	0
125	Russia	0	0
126	Rwanda	1	1
127	Samoa	0	0

	Pais	K_means	Clusterizacao Hierarquica
128	Saudi Arabia	0	2
129	Senegal	1	1
130	Serbia	0	0
131	Seychelles	0	0
132	Sierra Leone	1	1
133	Singapore	2	2
134	Slovak Republic	2	0
135	Slovenia	2	0
136	Solomon Islands	0	0
137	South Africa	1	0
138	South Korea	2	0
139	Spain	2	2
140	Sri Lanka	0	0
141	St. Vincent and the Grenadines	0	0
142	Sudan	1	0
143	Suriname	0	0
144	Sweden	2	2
145	Switzerland	2	2
146	Tajikistan	0	0
147	Tanzania	1	1
148	Thailand	0	0
149	Timor-Leste	1	0
150	Togo	1	1
151	Tonga	0	0
152	Tunisia	0	0
153	Turkey	0	0
154	Turkmenistan	0	0
155	Uganda	1	1
156	Ukraine	0	0
157	United Arab Emirates	2	2
158	United Kingdom	2	2
159	United States	2	2

	Pais	K_means	Clusterizacao Hierarquica
160	Uruguay	0	0
161	Uzbekistan	0	0
162	Vanuatu	0	0
163	Venezuela	0	0
164	Vietnam	0	0
165	Yemen	1	0
166	Zambia	1	1

In [205…

```
#Verificando o percentual de similaridade entre os 2 algoritmos:

import difflib

kmeans_lista = resultado_comparativo['K_means'].tolist()
hierarquico_lista = resultado_comparativo['Clusterizacao Hierarquica'].tolist()

sm=difflib.SequenceMatcher(None,kmeans_lista,hierarquico_lista)
similaridade= sm.ratio()*100

# Print the result
print("Similaridade entre os resultados dos algoritmos : {:.2f}".format(similaridad
```

```
Similaridade entre os resultados dos algoritmos : 83.83
```

PARTE 4 - ESCOLHA DE ALGORITMOS

In [206…

```
#Parte 4.1-Escreva em tópicos as etapas do algoritmo de K-médias até sua convergênc
#1) Inicializa com o numero K de clusters em que cada centroide inicial 'e aleatori
#   Por isso para que haja reprodutibilidade 'e importante configurar o seed inicia
#2) Para cada ponto do dado calcula a distancia ate os K centroides e vincula os po
#   pros seus correspondentes  clusters.
#3) Em cada K cluster calcula a media dos pontos que estao dentro do cluster e atua
# dos centroide para essas medias
#4) O processo e repetido ate que os centroides comecem a nao mudar ou mudar muito
```

In [207…

```
#Parte 4.2-Refaça o algoritmo apresentado na questão 1 a fim de garantir que o clus
#  pelo dado mais próximo  ao seu baricentro em todas as iterações do algoritmo.

# Para isso utilizarei o algoritmo do Kmedoide que utiliza um ponto real do conjunt
#  que minimiza a soma das distâncias entre ele e todos os outros pontos do cluster
from sklearn_extra.cluster import KMedoids

kmedoide = KMedoids(n_clusters=3, random_state=22, init='k-medoids++')
kmedoide_model= kmedoide.fit(dataset_transformed)

y_kmedoide = kmedoide.predict(dataset_transformed)

resultado_kmedoide = pd.DataFrame(y_kmedoide)
```

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
resultado_kmedoide = resultado_kmedoide.rename(columns={0: 'Categoria'})
resultado_kmedoide['Pais'] = nome_paises

resultado_kmedoide = resultado_kmedoide.sort_values(by='Categoria')

print(resultado_kmedoide)

#Como ja sabemos como alguns paises se classificam,
#podemos chegar a conclusao que as categorias estao separando em nivel de desenvolv
#Grupo 0 = desenvolvidos
#Grupo 1 = sub-desenvolvidos
#Grupo 2 = em desenvolvimento

#Para verificar o pais que mais representa o agrupamento podemos
#verificar a distancia de cada pais para o centro do cluster
#o que tiver a menor distancia e o que mais representa
distances_kmedoide = kmedoide.transform(dataset_transformed)
distances_kmedoide = pd.DataFrame(distances_kmedoide)

mais_proximos_kmedoide = []
mais_proximos_kmedoide = pd.DataFrame(mais_proximos)
mais_proximos_kmedoide['Categoria']=list(set(resultado_kmedoide['Categoria']))

coluna =0
for center in range(len(kmedoide.cluster_centers_)):
    distancia_coluna = distances_kmedoide[coluna].tolist()
    _index = np.argmin(distancia_coluna)
    mais_proximos_kmedoide.at[coluna, 'Pais que + representa'] = nome_paises[_index
    mais_proximos_kmedoide.at[coluna, 'Distancia'] = distances_kmedoide[coluna][_in
    coluna+=1

print("Pontos mais próximos dos centroides no kmedoide:\n", mais_proximos_kmedoide)
```

	Categoria	Pais
131	0	Seychelles
122	0	Portugal
60	0	Greece
58	0	Germany
54	0	France
53	0	Finland
130	0	Serbia
51	0	Estonia
100	0	Mauritius
133	0	Singapore
134	0	Slovak Republic
135	0	Slovenia
44	0	Denmark
43	0	Czech Republic
42	0	Cyprus
41	0	Croatia
121	0	Poland
138	0	South Korea
67	0	Hungary
117	0	Panama
98	0	Malta
104	0	Montenegro
96	0	Maldives
95	0	Malaysia
91	0	Luxembourg
90	0	Lithuania
86	0	Lebanon
85	0	Latvia
110	0	Netherlands
111	0	New Zealand
114	0	Norway
77	0	Japan
75	0	Italy
74	0	Israel
73	0	Ireland
68	0	Iceland
139	0	Spain
102	0	Moldova
7	0	Australia
8	0	Austria
164	0	Vietnam
15	0	Belgium
158	0	United Kingdom
144	0	Sweden
145	0	Switzerland
4	0	Antigua and Barbuda
29	0	Canada
13	0	Barbados
20	0	Bosnia and Herzegovina
148	0	Thailand
159	0	United States
10	0	Bahamas
24	0	Bulgaria
123	1	Qatar
89	1	Libya

125	1	Russia
79	1	Kazakhstan
11	1	Bahrain
157	1	United Arab Emirates
82	1	Kuwait
113	1	Nigeria
115	1	Oman
128	1	Saudi Arabia
23	1	Brunei
49	1	Equatorial Guinea
163	1	Venezuela
103	1	Mongolia
108	2	Namibia
160	2	Uruguay
161	2	Uzbekistan
107	2	Myanmar
112	2	Niger
106	2	Mozambique
105	2	Morocco
162	2	Vanuatu
109	2	Nepal
141	2	St. Vincent and the Grenadines
119	2	Peru
156	2	Ukraine
137	2	South Africa
136	2	Solomon Islands
142	2	Sudan
143	2	Suriname
146	2	Tajikistan
147	2	Tanzania
132	2	Sierra Leone
149	2	Timor-Leste
150	2	Togo
129	2	Senegal
127	2	Samoa
151	2	Tonga
126	2	Rwanda
152	2	Tunisia
124	2	Romania
153	2	Turkey
120	2	Philippines
154	2	Turkmenistan
155	2	Uganda
140	2	Sri Lanka
118	2	Paraguay
116	2	Pakistan
0	2	Afghanistan
83	2	Kyrgyz Republic
99	2	Mauritania
39	2	Costa Rica
38	2	Congo, Rep.
37	2	Congo, Dem. Rep.
36	2	Comoros
35	2	Colombia
34	2	China
33	2	Chile

32	2	Chad
31	2	Central African Republic
30	2	Cape Verde
28	2	Cameroon
27	2	Cambodia
26	2	Burundi
40	2	Cote d'Ivoire
25	2	Burkina Faso
21	2	Botswana
19	2	Bolivia
18	2	Bhutan
17	2	Benin
16	2	Belize
14	2	Belarus
12	2	Bangladesh
9	2	Azerbaijan
6	2	Armenia
5	2	Argentina
3	2	Angola
2	2	Algeria
1	2	Albania
22	2	Brazil
45	2	Dominican Republic
46	2	Ecuador
47	2	Egypt
97	2	Mali
94	2	Malawi
93	2	Madagascar
92	2	Macedonia, FYR
88	2	Liberia
87	2	Lesotho
84	2	Lao
165	2	Yemen
81	2	Kiribati
80	2	Kenya
78	2	Jordan
76	2	Jamaica
72	2	Iraq
71	2	Iran
70	2	Indonesia
69	2	India
66	2	Haiti
65	2	Guyana
64	2	Guinea-Bissau
63	2	Guinea
62	2	Guatemala
61	2	Grenada
59	2	Ghana
57	2	Georgia
56	2	Gambia
55	2	Gabon
52	2	Fiji
50	2	Eritrea
48	2	El Salvador
101	2	Micronesia, Fed. Sts.
166	2	Zambia

```
Pontos mais próximos dos centroides no kmedoide:
    Categoria Pais que + representa  Distancia
0           0              Slovenia        0.0
1           1          Saudi Arabia        0.0
2           2             Guatemala        0.0
```

In [208…

```
resultado_comparativo['KMedoide']=resultado_kmedoide.sort_values(by='Pais')['Catego

resultado_comparativo['KMedoide']= resultado_comparativo['KMedoide'].replace({0: 2,
print(resultado_comparativo)
#Verificando o percentual de similaridade entre os 3 algoritmos (2 a 2):

import difflib

kmeans_lista = resultado_comparativo['K_means'].tolist()
KMedoide_lista = resultado_comparativo['KMedoide'].tolist()
hierarquico_lista=  resultado_comparativo['Clusterizacao Hierarquica'].tolist()

sm=difflib.SequenceMatcher(None,kmeans_lista,KMedoide_lista)
similaridade= sm.ratio()*100

# Print the result
print("Similaridade entre KMEANS e KMEDOIDE : {:.2f}".format(similaridade))

sm=difflib.SequenceMatcher(None,hierarquico_lista,KMedoide_lista)
similaridade= sm.ratio()*100
print("Similaridade entre HIERARQUICO e KMEDOIDE : {:.2f}".format(similaridade))
```

	Pais	K_means	Clusterizacao Hierarquica	\
0	Afghanistan	1	1	
1	Albania	0	0	
2	Algeria	0	0	
3	Angola	1	0	
4	Antigua and Barbuda	0	0	
5	Argentina	0	0	
6	Armenia	0	0	
7	Australia	2	2	
8	Austria	2	2	
9	Azerbaijan	0	0	
10	Bahamas	0	0	
11	Bahrain	2	2	
12	Bangladesh	0	0	
13	Barbados	0	0	
14	Belarus	0	0	
15	Belgium	2	2	
16	Belize	0	0	
17	Benin	1	1	
18	Bhutan	0	0	
19	Bolivia	0	0	
20	Bosnia and Herzegovina	0	0	
21	Botswana	1	0	
22	Brazil	0	0	
23	Brunei	2	2	
24	Bulgaria	0	0	
25	Burkina Faso	1	1	
26	Burundi	1	1	
27	Cambodia	0	0	
28	Cameroon	1	1	
29	Canada	2	2	
30	Cape Verde	0	0	
31	Central African Republic	1	1	
32	Chad	1	1	
33	Chile	0	0	
34	China	0	0	
35	Colombia	0	0	
36	Comoros	1	1	
37	Congo, Dem. Rep.	1	1	
38	Congo, Rep.	1	0	
39	Costa Rica	0	0	
40	Cote d'Ivoire	1	1	
41	Croatia	0	0	
42	Cyprus	2	0	
43	Czech Republic	2	0	
44	Denmark	2	2	
45	Dominican Republic	0	0	
46	Ecuador	0	0	
47	Egypt	0	0	
48	El Salvador	0	0	
49	Equatorial Guinea	1	0	
50	Eritrea	1	0	
51	Estonia	0	0	
52	Fiji	0	0	
53	Finland	2	2	
54	France	2	2	

55	Gabon	1	0
56	Gambia	1	1
57	Georgia	0	0
58	Germany	2	2
59	Ghana	1	0
60	Greece	2	2
61	Grenada	0	0
62	Guatemala	0	0
63	Guinea	1	1
64	Guinea-Bissau	1	1
65	Guyana	0	0
66	Haiti	1	1
67	Hungary	0	0
68	Iceland	2	2
69	India	0	0
70	Indonesia	0	0
71	Iran	0	0
72	Iraq	1	0
73	Ireland	2	2
74	Israel	2	2
75	Italy	2	2
76	Jamaica	0	0
77	Japan	2	2
78	Jordan	0	0
79	Kazakhstan	0	0
80	Kenya	1	1
81	Kiribati	1	0
82	Kuwait	2	2
83	Kyrgyz Republic	0	0
84	Lao	1	0
85	Latvia	0	0
86	Lebanon	0	0
87	Lesotho	1	0
88	Liberia	1	0
89	Libya	0	2
90	Lithuania	0	0
91	Luxembourg	2	2
92	Macedonia, FYR	0	0
93	Madagascar	1	1
94	Malawi	1	1
95	Malaysia	0	0
96	Maldives	0	0
97	Mali	1	1
98	Malta	2	2
99	Mauritania	1	0
100	Mauritius	0	0
101	Micronesia, Fed. Sts.	0	0
102	Moldova	0	0
103	Mongolia	0	0
104	Montenegro	0	0
105	Morocco	0	0
106	Mozambique	1	1
107	Myanmar	0	0
108	Namibia	1	0
109	Nepal	0	0
110	Netherlands	2	2

111	New Zealand	2	2
112	Niger	1	1
113	Nigeria	1	0
114	Norway	2	2
115	Oman	0	2
116	Pakistan	1	0
117	Panama	0	0
118	Paraguay	0	0
119	Peru	0	0
120	Philippines	0	0
121	Poland	0	0
122	Portugal	2	2
123	Qatar	2	2
124	Romania	0	0
125	Russia	0	0
126	Rwanda	1	1
127	Samoa	0	0
128	Saudi Arabia	0	2
129	Senegal	1	1
130	Serbia	0	0
131	Seychelles	0	0
132	Sierra Leone	1	1
133	Singapore	2	2
134	Slovak Republic	2	0
135	Slovenia	2	0
136	Solomon Islands	0	0
137	South Africa	1	0
138	South Korea	2	0
139	Spain	2	2
140	Sri Lanka	0	0
141	St. Vincent and the Grenadines	0	0
142	Sudan	1	0
143	Suriname	0	0
144	Sweden	2	2
145	Switzerland	2	2
146	Tajikistan	0	0
147	Tanzania	1	1
148	Thailand	0	0
149	Timor-Leste	1	0
150	Togo	1	1
151	Tonga	0	0
152	Tunisia	0	0
153	Turkey	0	0
154	Turkmenistan	0	0
155	Uganda	1	1
156	Ukraine	0	0
157	United Arab Emirates	2	2
158	United Kingdom	2	2
159	United States	2	2
160	Uruguay	0	0
161	Uzbekistan	0	0
162	Vanuatu	0	0
163	Venezuela	0	0
164	Vietnam	0	0
165	Yemen	1	0
166	Zambia	1	1

	KMedoide
0	0
1	0
2	0
3	0
4	2
5	0
6	0
7	2
8	2
9	0
10	2
11	1
12	0
13	2
14	0
15	2
16	0
17	0
18	0
19	0
20	2
21	0
22	0
23	1
24	2
25	0
26	0
27	0
28	0
29	2
30	0
31	0
32	0
33	0
34	0
35	0
36	0
37	0
38	0
39	0
40	0
41	2
42	2
43	2
44	2
45	0
46	0
47	0
48	0
49	1
50	0
51	2
52	0
53	2

54	2
55	0
56	0
57	0
58	2
59	0
60	2
61	0
62	0
63	0
64	0
65	0
66	0
67	2
68	2
69	0
70	0
71	0
72	0
73	2
74	2
75	2
76	0
77	2
78	0
79	1
80	0
81	0
82	1
83	0
84	0
85	2
86	2
87	0
88	0
89	1
90	2
91	2
92	0
93	0
94	0
95	2
96	2
97	0
98	2
99	0
100	2
101	0
102	2
103	1
104	2
105	0
106	0
107	0
108	0
109	0

110	2
111	2
112	0
113	1
114	2
115	1
116	0
117	2
118	0
119	0
120	0
121	2
122	2
123	1
124	0
125	1
126	0
127	0
128	1
129	0
130	2
131	2
132	0
133	2
134	2
135	2
136	0
137	0
138	2
139	2
140	0
141	0
142	0
143	0
144	2
145	2
146	0
147	0
148	2
149	0
150	0
151	0
152	0
153	0
154	0
155	0
156	0
157	1
158	2
159	2
160	0
161	0
162	0
163	1
164	2
165	0

```
166         0
Similaridade entre KMEANS e KMEDOIDE : 17.96
Similaridade entre HIERARQUICO e KMEDOIDE : 17.96
```

In [209…

```
#Parte 4.3-  O algoritmo de K-médias é sensível a   outliers nos dados. Explique.
# Pois a logica do algoritmo do Kmeans se basea no calculo das medias, entao
# a presença de outliers pode deslocar os centroides para longe dos verdadeiros cen
# Alem disso, com um provavel deslocamento dos centroides, alguns pontos podem ser
#  clusters que nao representam bem a sua localizacao real nos dados.
```

In [210…

```
#Parte 4.4 - Por que o algoritmo de DBScan é mais robusto à presença de outliers?
# O DBScan identifica pontos que nao pertencem a nenhum cluster como ruído. Isso oc
# pois o algoritmo do dbscan 'e baseado na densidade de pontos numa determinada are
# Logo, um outlier nao vai ter pontos proximos, logo nao tera vizinhos proximos.
# Isso 'e, esses pontos nao vao influenciar a formacao dos clusters.
```

In [211…

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=1.2, min_samples=4)
dbscan_model= dbscan.fit(dataframe_transformed)

dbscan_labels = dbscan.labels_

dbscan_labels
```

Out[211]:

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