

Agrupamento

MOHAMMED ASHOUR, ANTONIO MOSCOSO SÁNCHEZ,
NICOLÁS VILELA PÉREZ

10 de decembro do 2022

1. Introdución

Horas dedicadas fóra de clase: 6

Antes que nada realizouse un procesado dos conxuntos de datos empregados para a eliminación de posibles valores atípicos ou *outliers* presentes no dataset. Para isto usouse a seguinte función en Python:

```
1 def remove_outliers(df):
2     """Function to remove outliers"""
3
4     # Calculate the Q3 and Q1
5     Q3 = df.quantile(0.75, numeric_only=True)
6     Q1 = df.quantile(0.25, numeric_only=True)
7
8     # Calculate the IQR
9     IQR = Q3 - Q1
10
11    # Calculate the upper and lower limits of the dataframe
12    upper_limit = Q3 + 1.5 * IQR
13    lower_limit = Q1 - 1.5 * IQR
14
15    # Add index of rows that are outliers to a list
16    indexesToDelete = []
17    for i in range(len(df)):
18        for j in range(len(df.columns)):
19            if df.iloc[i, j] > upper_limit[j] or df.iloc[i, j] <
lower_limit[j]:
20                indexesToDelete.append(i)
21
22    # Drop from the dataframe the rows whose indexes are in the
list
23    df.drop(indexesToDelete, inplace=True)
24
25    return df
```

Empregando o método estatístico de eliminación de valores atípicos elimináñanse todos aqueles datos que difiran do resto dos datos almacenados.

Todo o **código** empregado para a elaboración deste informe está **dispoñible no arquivo adxunto** `codigo.py`.

2. Conxunto de datos dun arquivo CSV

Para este apartado empregouse o conxunto de datos da seguinte ligazón:
https://www.dropbox.com/s/k2vs1tvqoe6u70y/AMD_clustering_0.csv.gz?dl=0

Cabe destacar que se tratou de empregar a totalidade dos datos para a execución dos tres algoritmos dos cales se falará a continuación. Non obstante, debido a problemas computacionais de memoria para certos algoritmos, e tamén para facilitar a visualización, decidiuse seleccionar un subconxunto de datos do dataset orixinal. Neste caso, empregarase o 1% inicial dos datos orixinais.

2.1. K-means

A primeira tarefa que se realizará antes de executar o algoritmo será estudar o número de clusters óptimo para o conxunto de datos que tratamos. Para isto executaranse as seguintes liñas de código:

```
1 import matplotlib.pyplot as plt
2 from sklearn.cluster import KMeans
3
4 wcss = []
5 for i in range(1, 11):
6     modelo = KMeans(n_clusters=i)
7     agrup = modelo.fit(csv_data_sample)
8     wcss.append(agrup.inertia_)
9 plt.plot(range(1, 11), wcss)
10 plt.title('Método do cóbado para K-means')
11 plt.xlabel('Número de clusters')
12 plt.ylabel('Custo')
```

En primeiro lugar almacénase o custo de cálculo dos clusters, cun número de clúster que vai desde 1 ata 10. A continuación, represéntase graficamente este custo:

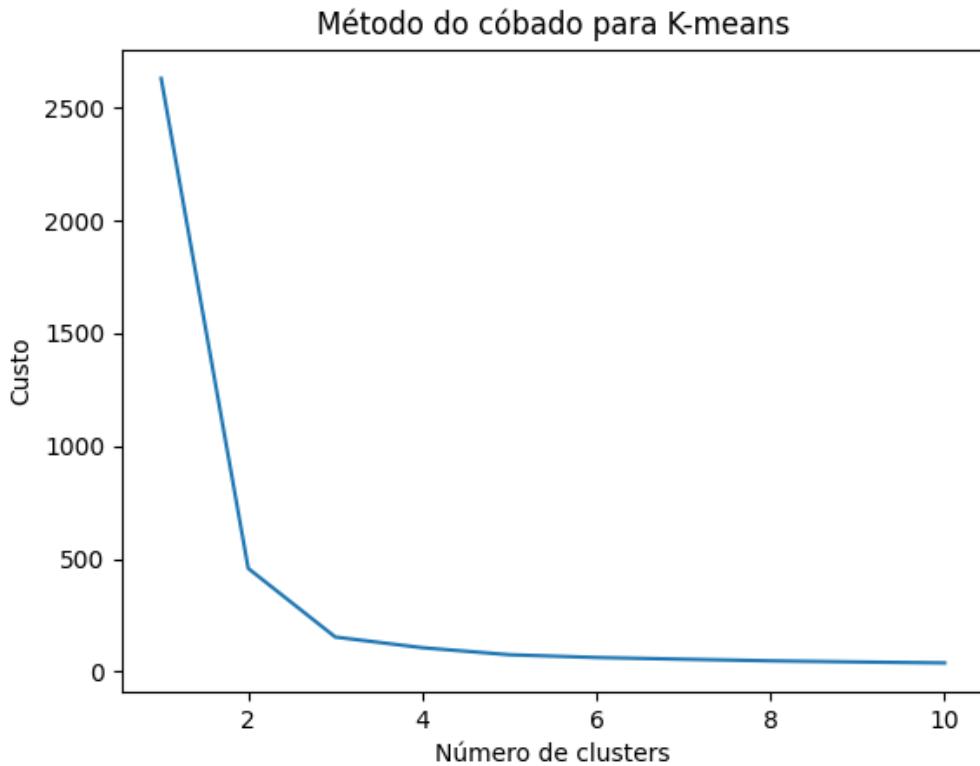


Figura 1: Método do cóbado para o algoritmo K-means sobre os datos do CSV

A partir do obtido na gráfica, e aplicando o método do cóbado, podemos concluir que o número óptimo de agrupamentos é de 3, pois a partir deste valor a mellora de custos comeza a diminuir, formándose un cóbado na gráfica.

Tras a execución do algoritmo K-means de agrupamento dos datos, co número de clusters óptimo, obtívose o seguinte resultado:

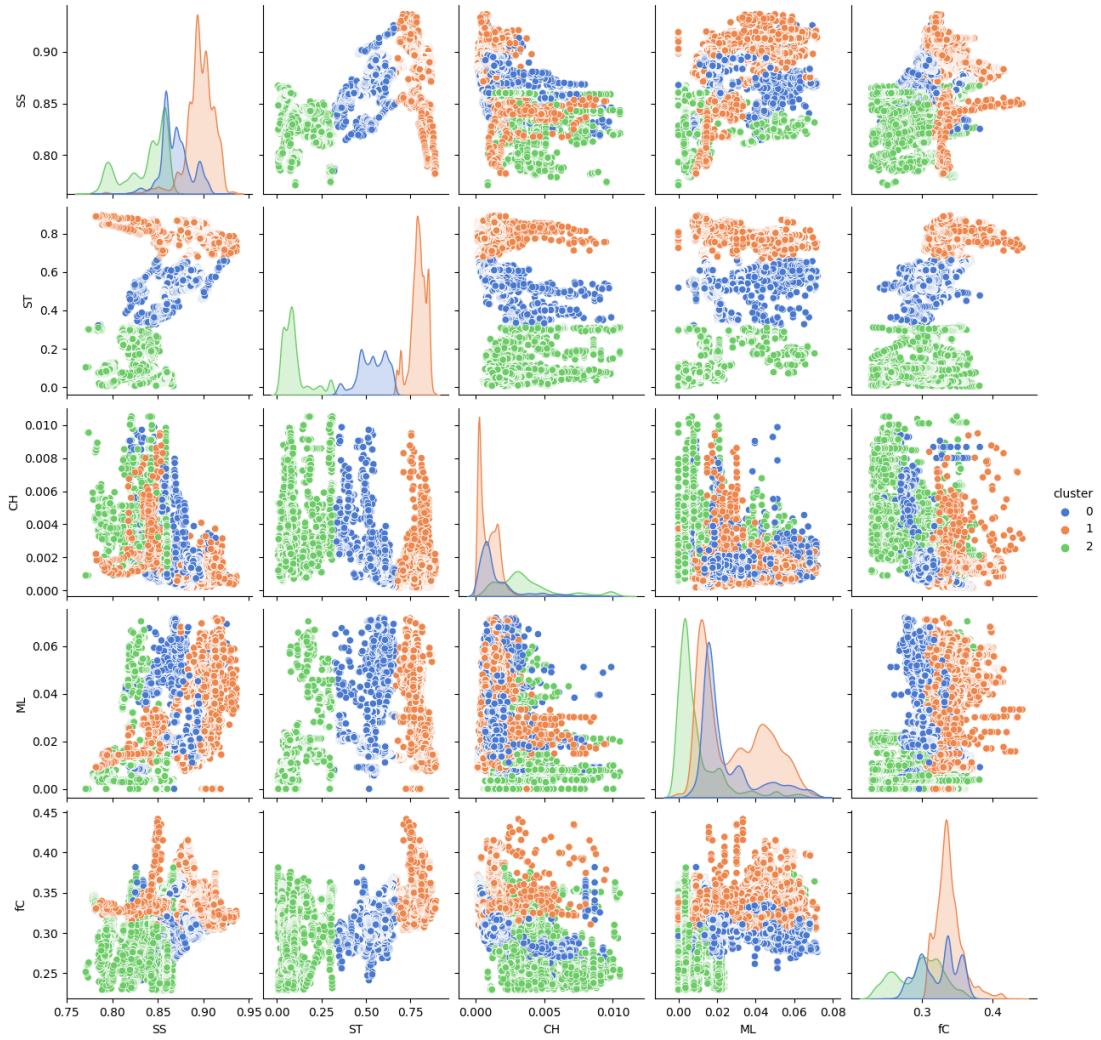


Figura 2: Agrupamento a través do algoritmo K-means sobre os datos do CSV con $k=3$

Á vista dos resultados da figura 2 pódese comprobar que os resultados obtidos tras a execución do algoritmo K-means son 3 agrupamentos para o arquivo de datos empregado, tal como se indicou na creación do modelo. O número de clusters escolleuse empregando o método do cóbado exposto anteriormente.

O algoritmo K-means consta de 3 etapas: inicialización, asignación dos exemplos aos centroides e actualización dos centroides:

1. Inicialización: co número de grupos k escollido, establecense aleatoriamente k centroides
2. Asignación dos exemplos aos centroides: cada valor do conxunto de datos asígnase ao seu centroide máis próximo.
3. Actualización dos centroides: para cada grupo, actualízase a posición do centroide tomando como novo centroide a poición media dos valores que pertencen ao grupo.

Nas representacións por pares obtidas pode observarse claramente a distribución en 3 agrupamentos. Estes agrupamentos poden diferenciarse máis claramente para algunas dimensíóns que para outras.

2.2. SOM

En primeiro lugar, deberase obter un número óptimo de clusters. Para iso, empregarase o método do cóbado para saber o susodito mellor número:

```
1 import matplotlib.pyplot as plt
2 from sklearn_som.som import SOM
3
4 wcss = []
5 for i in range(1, 11):
6     data_som = SOM(m=i, n=1, dim=len(csv_data_sample.columns))
7     data_som.fit(data)
8     wcss.append(data_som.inertia_)
9 plt.plot(range(1, 11), wcss)
10 plt.title('Método do cóbado para SOM')
11 plt.xlabel('Número de clusters')
12 plt.ylabel('Custo')
```

Mediante o código anterior almacénase nunha lista o custo de obtención dos clusters en función do seu número. Despois da execución do código anterior, obtense o seguinte gráfico:

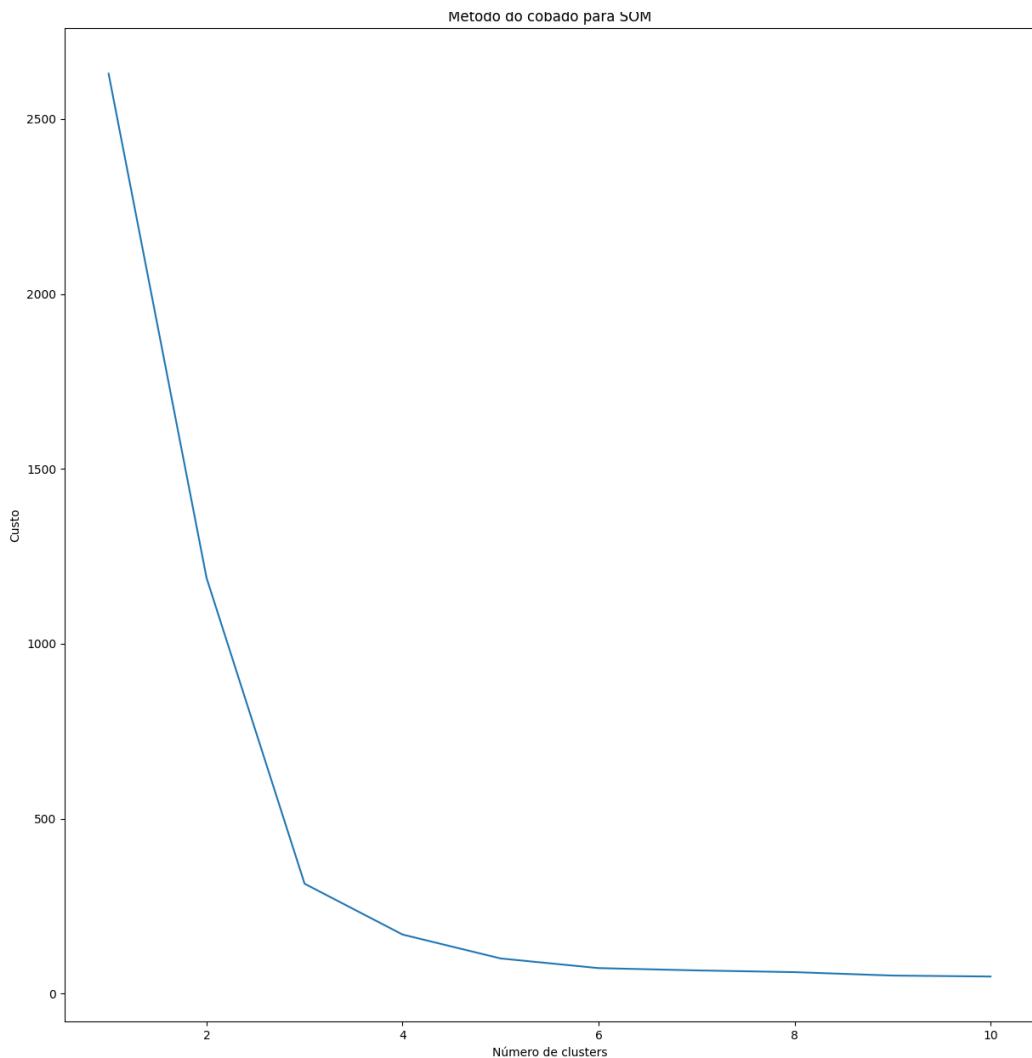


Figura 3: Método do cóbado para o algoritmo SOM sobre os dados do CSV

Póde-se comprobar mediante a figura 3 que un valor que pode ser óptimo para o número de clusters é 3, pois vese que a tendencia de baixada do custo no gráfico disminue a partir dese valor.

Despois de saber o número de grupos que se emplegará, execútase o algoritmo SOM para agrupar os datos, obtendo os seguintes resultados:

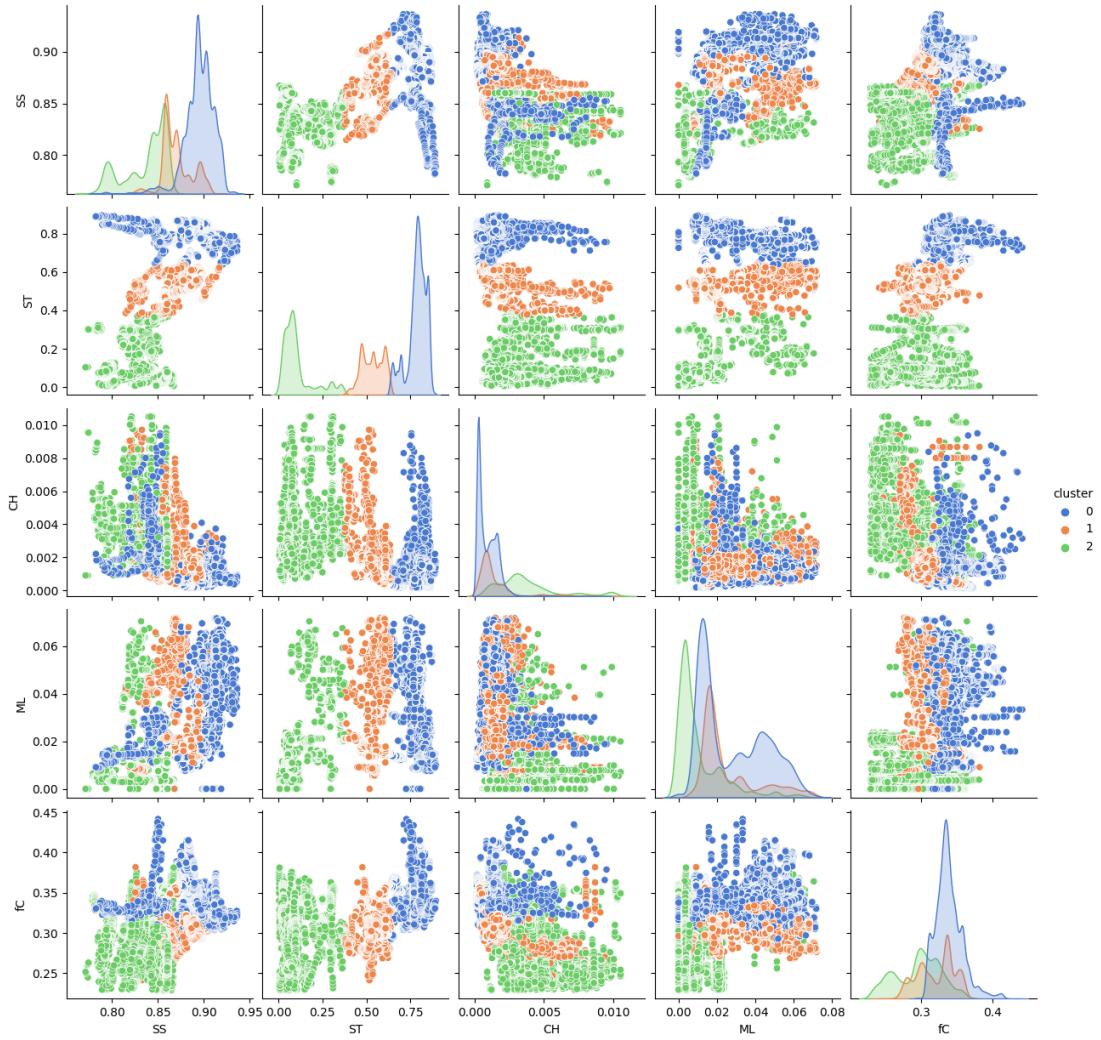


Figura 4: Agrupamento a través do algoritmo Kmeans sobre os datos do CSV con $m=3$

Á vista dos resultados da figura 4 pódese comprobar que os resultados obtidos tras a execución do algoritmo SOM son moi parecidos aos obtidos en K-means (figura 2). Isto é debido a que ambos algoritmos gardan unha forte similitude en canto a asociar un punto determinado a un clúster en función da proximidade á neurona (no caso do SOM, a "neurona gañadora" que asocia o punto ao clúster da mesma). Non obstante, a diferenza do método K-means, cando se actualiza o nodo central ou centroide, este actualiza os centroides veciños en lugar de unicamente actualizarse a si mesmo. Polo tanto, cada vez que se asocia un punto ao centroide determinado, actualízanse os demais nodos centrais dos clusters veciños, obtendo así uns resultados notoriamente similares.

En consecuencia, o algoritmo SOM está composto por etapas moi similares ás anteriormente explicadas no algoritmo K-means, unicamente modificando a fase de actualización de centroides, pois, como xa se explicou, en lugar de actualizar o único centroide asociado a cada grupo, actualízanse os centroides veciños.

2.3. DBSCAN

O algoritmo DBSCAN ten dous parámetros principais para os cales hai que indicar o seu valor:

- **Épsilon (eps):** especifica o cerca que deben estar os puntos entre si para ser considerados parte do mesmo grupo ou clúster.
- **Puntos mínimos (minPts):** especifica o número mínimo de puntos para formar unha rexión densa.

Para determinar estes parámetros seguiuse o seguinte procedemento:

1. Estimación de minPts

Non existe unha forma automática de determinar este parámetro, senón que se adoita empregar o coñecemento e familiarización co dominio do conxunto de datos. Neste caso non hai moito coñecemento sobre o dominio, polo cal se vai empregar unha regla xeral que di que o valor do parámetro será igual ao dobre das dimensíons do conxunto de datos. Neste caso, o conxunto de datos ten 5 dimensíons, polo cal **minPts = $2 \cdot 5 = 10$** .

2. Aplicación do método do cóbado para estimar eps

Unha vez coñecido o valor estimado de **minPts**, aplicarase o método do cóbado ao conxunto de datos para estimar **eps**. Emprégase o seguinte código:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from sklearn.neighbors import NearestNeighbors
4
5 neighbors = NearestNeighbors(n_neighbors=estimated_minPts)
6 neighbors_fit = neighbors.fit(csv_data_sample)
7 distances, _ = neighbors_fit.kneighbors(csv_data_sample)
8 distances = np.sort(distances, axis=0)
9 distances = distances[:,1]
10 plt.xticks(np.arange(28000, 30000, 500))
11 plt.xlim(28000, 30000)
12 plt.yticks(np.arange(0.0, 0.01, 0.001))
13 plt.ylim(0.0, 0.01)
14 plt.plot(distances)
```

Con este código obtense a seguinte gráfica:

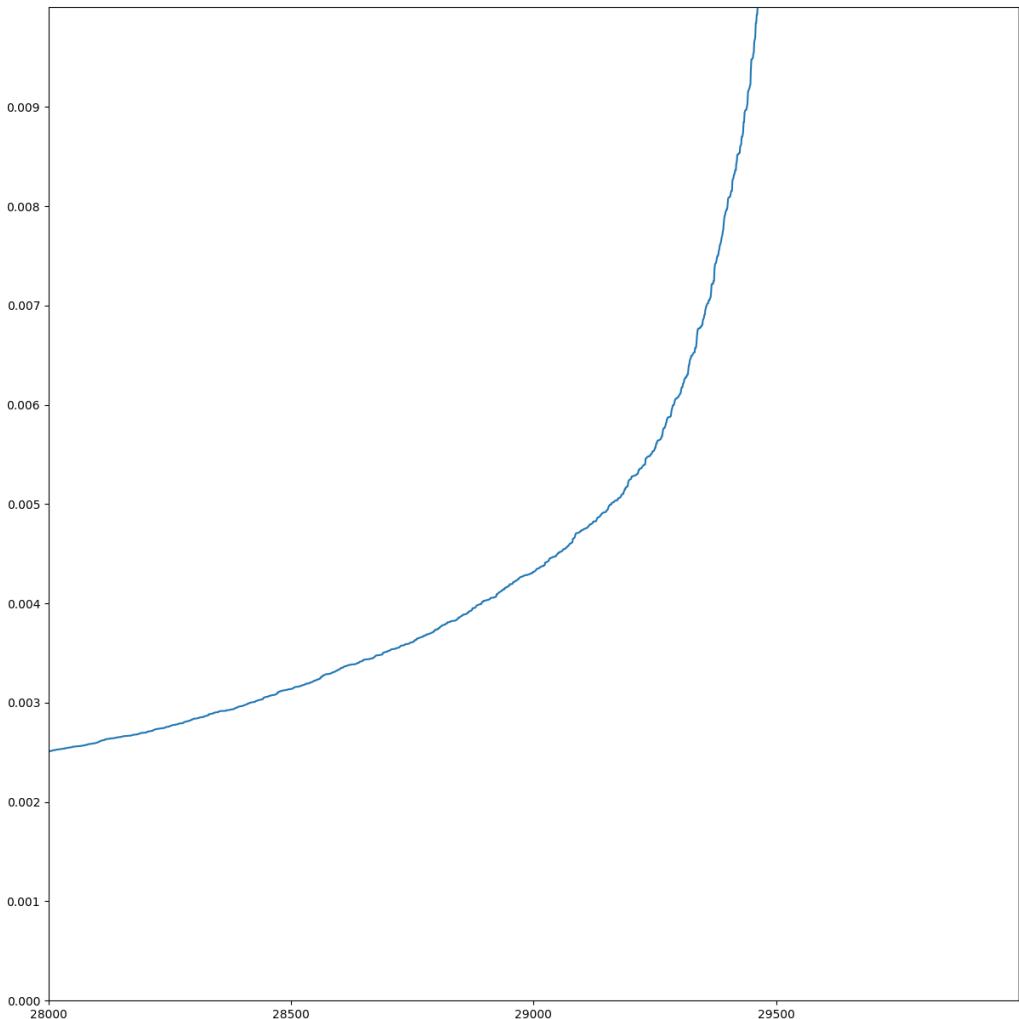


Figura 5: Método do cóbado para o algoritmo DBSCAN sobre os datos do CSV

A partir da gráfica 5, pódese aproximar o punto de máxima curvatura, de forma que **eps = 0.005**.

Tras a execución do algoritmo DBSCAN de agrupamento dos datos coa estimación paramétrica anterior, obtívose o seguinte resultado:

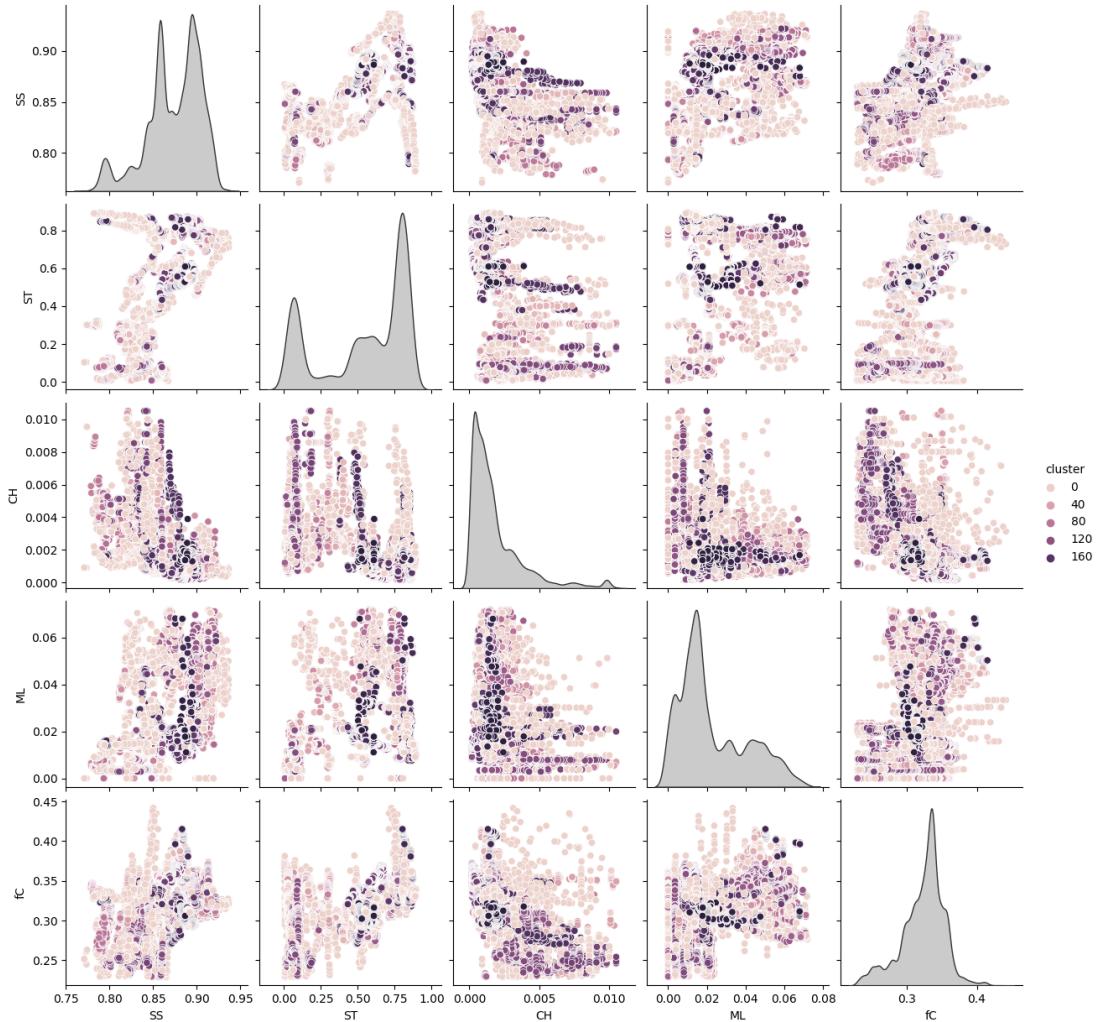


Figura 6: Agrupamento a través do algoritmo DBSCAN sobre os datos do CSV con $\text{eps}=0.005$ e $\text{minPts}=10$

Á vista dos resultados da figura 6 pódese comprobar que os clusters obtidos tras a execución do algoritmo DBSCAN son moi pouco diferenciados en comparación cos obtidos con K-means (figura 2) ou SOM (figura 4). Isto seguramente sexa debido ao descoñecemento do conxunto de datos, xa que sen saber moito acerca do conxunto de datos non é trivial determinar o parámetro `minPts`, como xa se dixo anteriormente. A maiores disto, o parámetro `eps` vén determinado a partir dos puntos mínimos (`minPts`) e aplicando o método do cóbado, polo cal se arrastra o erro.

O algoritmo comeza por un punto arbitrario que non fose visitado. Todos os veciños que hai a unha distancia menor ou igual a `eps` son visitados, e se esa e -veciñanza contén un número maior ou igual de puntos que `minPts`, créase un clúster sobre devandita veciñanza. Do contrario, o punto é etiquetado como ruído.

Se un punto se inclúe na parte densa dun clúster, a súa e -veciñanza tamén forma parte dese clúster. Desta forma, todos os puntos de dita veciñanza se engaden

ao clúster, ao igual que as e -veciñanzas destes puntos que sexan o suficientemente densas.

Este proceso continua ata construír completamente un clúster densamente conectado. Entón, un novo punto non visitado visítase e procésase co obxectivo de descubrir outro clúster ou ruído.

2.4. Análise de resultados

Pódese comprobar que para os diferentes algoritmos de agrupación, para algúns pares de variables fai unha boa agrupación (véxase ST e ML), mentres que para outras fai unha agrupación dunha calidade degradada (por exemplo, entre SS e CH). Isto é debido a que, para a realización dos diferentes grupos, é máis fácil facer grupos cando os puntos están previamente localizados en zonas da gráfica heteroxéneas, isto é, facer unha "agrupación previa" para que así o algoritmo que vaia realizar a clusterización poida agrupar os puntos dun modo correcto (dito doutro modo, que existe unha correlación entre ese par de atributos). En caso de que os puntos estén dispostos nunha forma homoxénea, isto é, estén demasiado mesturados nun espazo reducido, a agrupación que se vaia realizar vai ser de peor calidade.

3. Conxunto de datos dunha base de datos

Para este apartado empregouse unha base de datos na cal se importou o conxunto de datos da seguinte ligazón:

https://www.dropbox.com/s/71ybnugxdbelcn6/Databank_World_Development_Indicators.csv?dl=0

Os metadatos asociados ao conxunto de datos están dispoñibles na seguinte ligazón:

https://www.dropbox.com/s/ynobjtvfuvg5s13/Databank_World_Development_Indicators_Metadata.csv?dl=0

O primeiro que hai que facer desde Python é conectarse á base de datos e obter o conxunto de datos propiamente dito:

```
1 import sqlalchemy
2 import pandas as pd
3
4 # Connect to the PostgreSQL database
5 db_engine = sqlalchemy.create_engine("postgresql+psycopg2://
6                                     postgres:postgres@172.17.0.2:5432/P7")
7 connection = db_engine.connect()
8
9 # Get the data from the database
10 db_df = pd.read_sql_query('SELECT * FROM
11                           databank_world_deployment_indicators', connection)
```

```

10
11 # Close the database connection
12 connection.close()
13 db_engine.dispose()

```

Unha vez obtido o conxunto de datos, bórranse as columnas que non son necesarias (`time`, `time_code`, `country_name` e `country_code`); úsase a función `interpolate` do paquete `pandas` para o recheo de valores nulos dunha columna determinada realizando unha interpolación. Con isto conséguese rechear estes valores dunha forma máis elegante; e, por último, bórranse os valores atípicos coa función mencionada na páxina 1.

3.1. K-means

Igual que co conxunto de datos anterior, empregamos o método do cóbado para obter o número óptimo de clústers. A gráfica que se obtén é a seguinte:

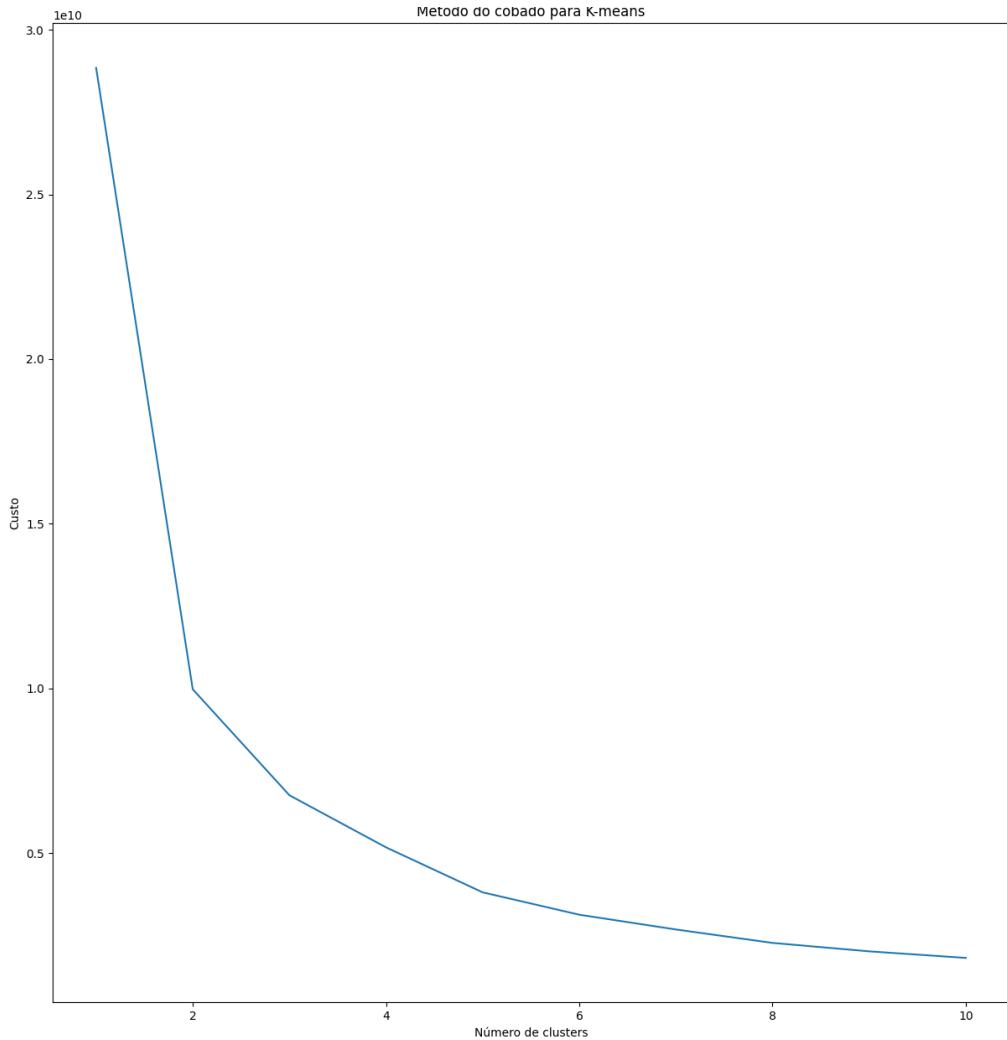


Figura 7: Método do cóbado para o algoritmo K-means sobre os datos da base de datos

Despois de analizar a gráfica, podemos concluir que o número óptimo de clusters é 5, polo que procederemos a executar o algoritmo definindo este valor. O proceso de execución e obtención dos resultados é análogo ao do apartado anterior. Despois da execución, obtemos o seguinte:

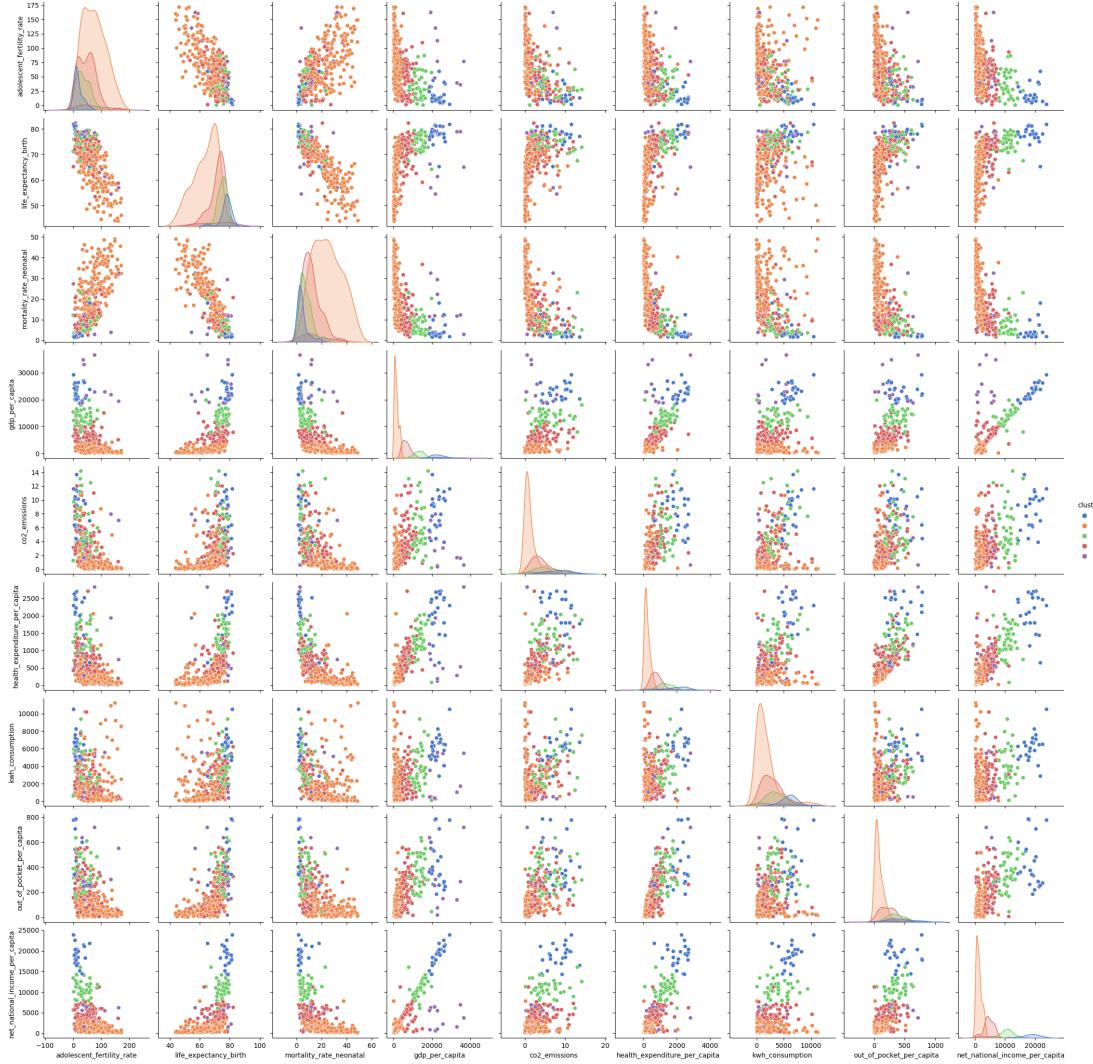


Figura 8: Agrupamento a través do algoritmo K-means sobre os datos da base de datos con $k=5$

3.2. SOM

Tal e como se fixo en apartados anteriores, realizarase o método do cóbado para a obtención do número óptimo de grupos a realizar. Tras executar o código relativo a este método, obtívose o seguinte gráfico:

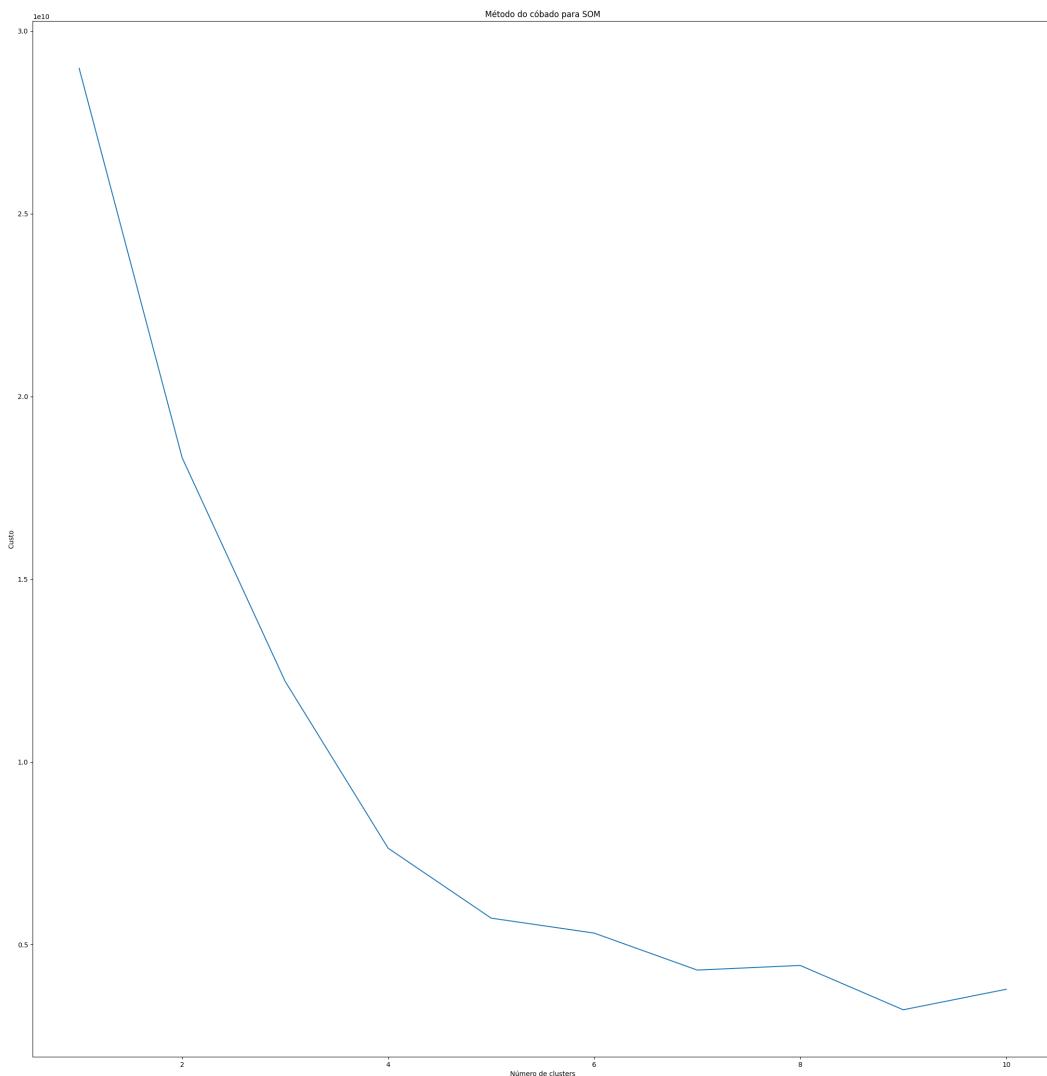


Figura 9: Método do cóbado para o algoritmo SOM sobre os datos da base de datos

Pódese comprobar que un valor que parece axeitado para establecelo como número de clusters é 5, pois a partir dese valor, estabilízase o decrecemento do custo na gráfica. Sabendo esto, procédere a realizar o agrupamento con 5 clusters:

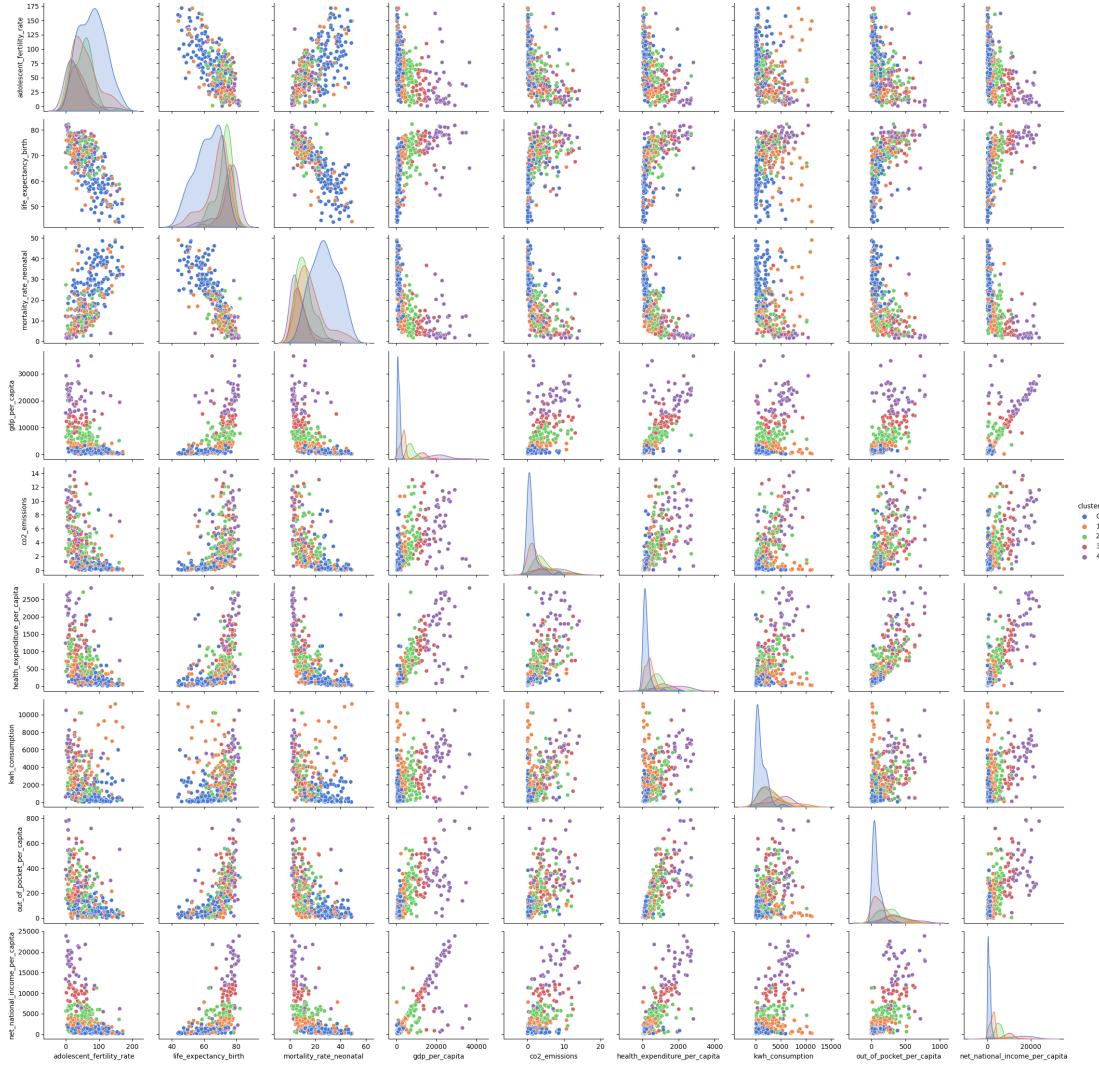


Figura 10: Agrupamento a través do algoritmo SOM sobre os datos da base de datos con $m=5$

Á vista dos resultados obtidos na figura 10, pódese comprobar que sucede algo parecido ao que ocorría con este mesmo algoritmo para o CSV: para algúns pares de variables realiza unha boa clusterización, mentres que para outros, aparece un agrupamento demasiado homoxéneo. Isto foi explicado na páxina 10.

3.3. DBSCAN

Para este conxunto de datos determináronse os parámetros do algoritmo seguindo o procedemento explicado na páxina 8. Así, os valores dos parámetros son os seguintes: **eps = 3000** e **minPts = $2 \cdot 9 = 18$** . A gráfica tras aplicar o método do cóbado para este conxunto de datos é a seguinte:

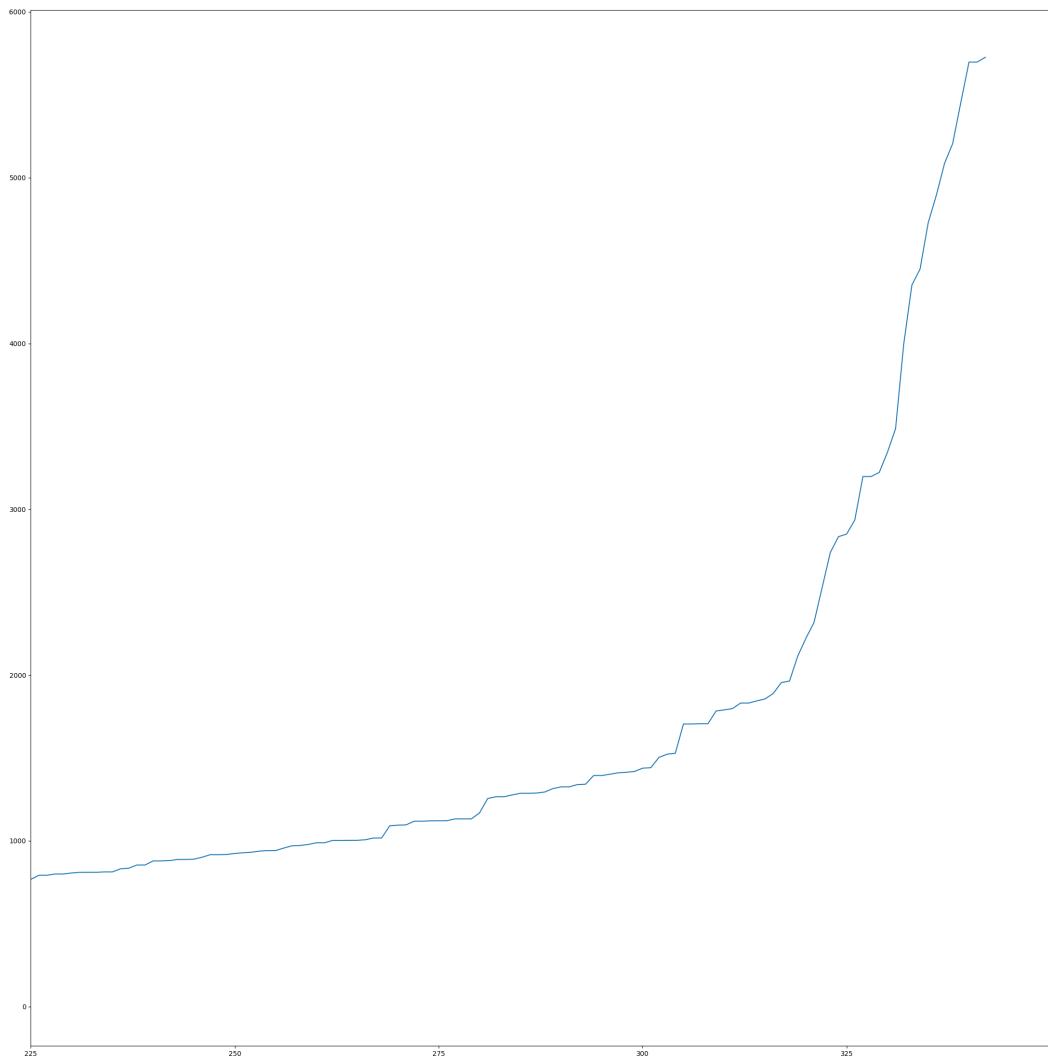


Figura 11: Método do cóbado para o algoritmo DBSCAN sobre os datos da base de datos

Tras a execución do algoritmo DBSCAN de agrupamento dos datos coa estimación paramétrica anterior, obtívose o seguinte resultado:

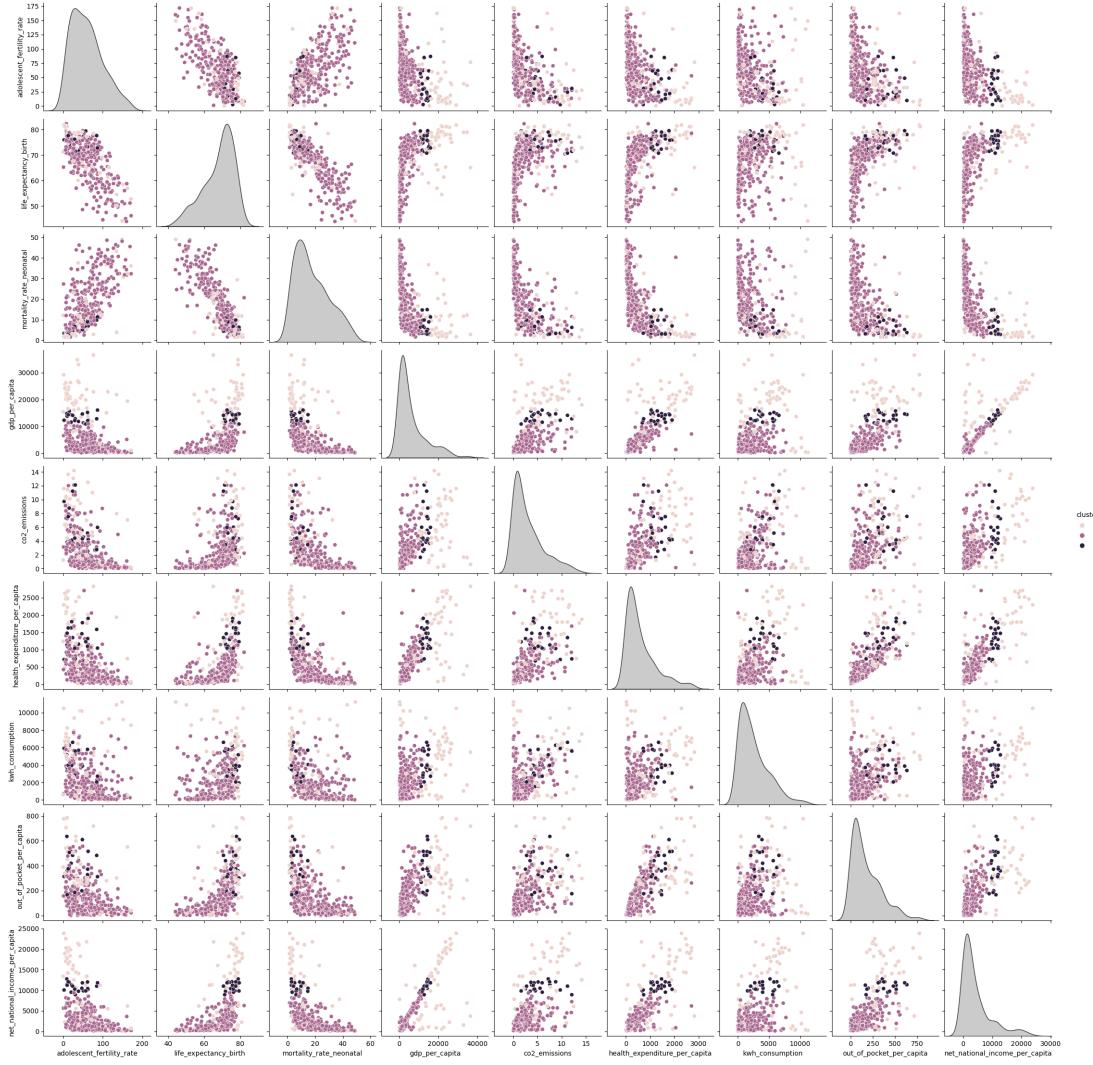


Figura 12: Agrupamento a través do algoritmo DBSCAN sobre os datos da base de datos con $\text{eps}=3000$ e $\text{minPts}=18$

Á vista dos resultados da figura 12 pódese comprobar que os datos obtidos tras a execución do algoritmo DBSCAN teñen menos clusters que con K-means (figura 8) ou SOM (figura 10). Non obstante, estos clusters son máis diferenciados que os obtidos con este algoritmo para o conxunto de datos anterior (figura 6).

3.4. Análise de resultados

Á vista dos resultados obtidos, pódese comprobar que existen algunas gráficas nas que a agrupación aparece dun xeito máis diferenciado que noutras, indicando que os valores están máis correlacionados. A continuación expóñense algúns exemplos:

- Na gráfica que relaciona o PIB per cápita coa taxa de natalidade en adolescentes (entre 15 e 19 anos), pódese observar que os países con PIB per cápita

máis baixo teñen unha taxa de natalidade en adolescentes máis alta que o resto.

- Na gráfica que relaciona o PIB per cápita coa esperanza de vida, pódese observar que os países con PIB per cápita máis baixo teñen unha esperanza de vida é máis baixa, e viceversa.
- Na gráfica que relaciona o PIB per cápita co ingreso nacional neto per cápita, pódese observar unha correlación moi estreita entre os datos, describindo unha tendencia praticamente linear. Algúns dos datos desvíanse da tendencia, probablemente debido á interpolación realizada no preprocessamento cos datos nulos.

Pódese observar tamén que existen indicadores que son mellores discriminantes que outros, xa que as agrupacións que se obteñen a partir destes teñen grupos moito más diferenciados para calquera dos tres algoritmos empregados.

Estes indicadores discriminantes son `gdp_per_capita` (PIB per cápita, en USD) e `net_national_income_per_capita` (ingreso nacional neto per cápita, en USD).

Pódese apreciar claramente esta última observación nas figuras [8](#), [10](#) e [12](#), correspondentes aos agrupamentos a través dos algoritmos K-means, SOM e DBSCAN, respectivamente.