In [2]:

```python
#iberias generales
import numpy as np
import pandas as pd

#librerias graficas
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go

#modelo
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture

#
from yellowbrick.cluster import KElbowVisualizer
from scipy.cluster.hierarchy import dendrogram, linkage


#preprocesamiento
from sklearn.preprocessing import StandardScaler

#otros
import warnings
warnings.filterwarnings('ignore')
```

# Carga de datos

In [2]:

```python
#Carga de dataset
df = pd.read_csv('https://raw.githubusercontent.com/SaulSebastian/dataset/ma
```

# Analisis

In [3]:

```python
#Muestreo aleatoio
df.sample(5)
```

Out[3]:

| | cases | deaths | countriesAndTerritories | continentExp | Cumulative_number_for_1 19_ |
|---|---|---|---|---|---|
| **345** | 0 | 0 | Brazil | America | |
| **33578** | 15 | 0 | Malta | Europe | |
| **2531** | 0 | 0 | Philippines | Asia | |
| **51280** | 0 | 0 | Bermuda | America | |
| **34112** | 0 | 0 | Brunei Darussalam | Asia | |

In [4]:

```python
#Visualización de los tipos de datos
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61900 entries, 0 to 61899
Data columns (total 7 columns):
 #   Column                                                      Non-Null Co
unt  Dtype
---  ------                                                      -----------
---  -----
 0   cases                                                       61900 non-n
ull  int64
 1   deaths                                                      61900 non-n
ull  int64
 2   countriesAndTerritories                                     61900 non-n
ull  object
 3   continentExp                                                61900 non-n
ull  object
 4   Cumulative_number_for_14_days_of_COVID-19_cases_per_100000  59021 non-n
ull  float64
 5   PORC_SOBREPESO_F                                            61900 non-n
ull  float64
 6   PORC_SOBREPESO_M                                            61900 non-n
ull  float64
dtypes: float64(3), int64(2), object(2)
memory usage: 3.3+ MB
```

In [5]:

```python
#Verificar la existencia y cantidad de null's
df.isna().sum()
```

Out[5]:

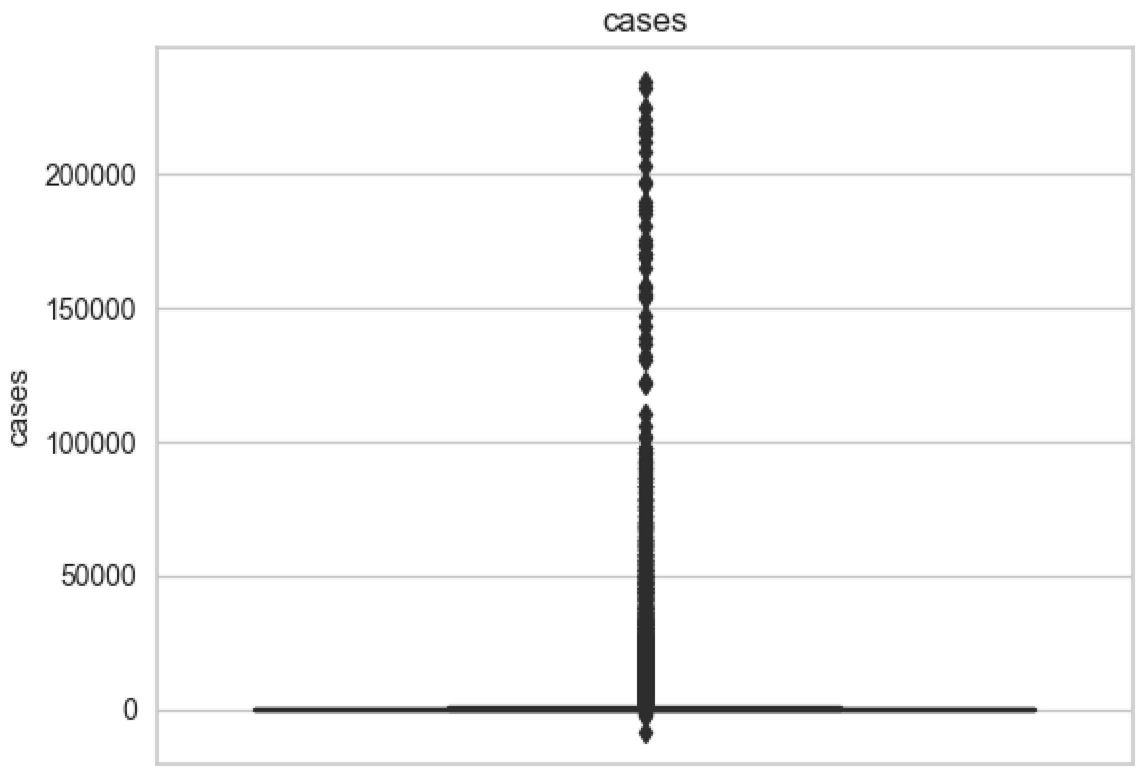
```
cases                                                            0
deaths                                                           0
countriesAndTerritories                                          0
continentExp                                                     0
Cumulative_number_for_14_days_of_COVID-19_cases_per_100000    2879
PORC_SOBREPESO_F                                                 0
PORC_SOBREPESO_M                                                 0
dtype: int64
```

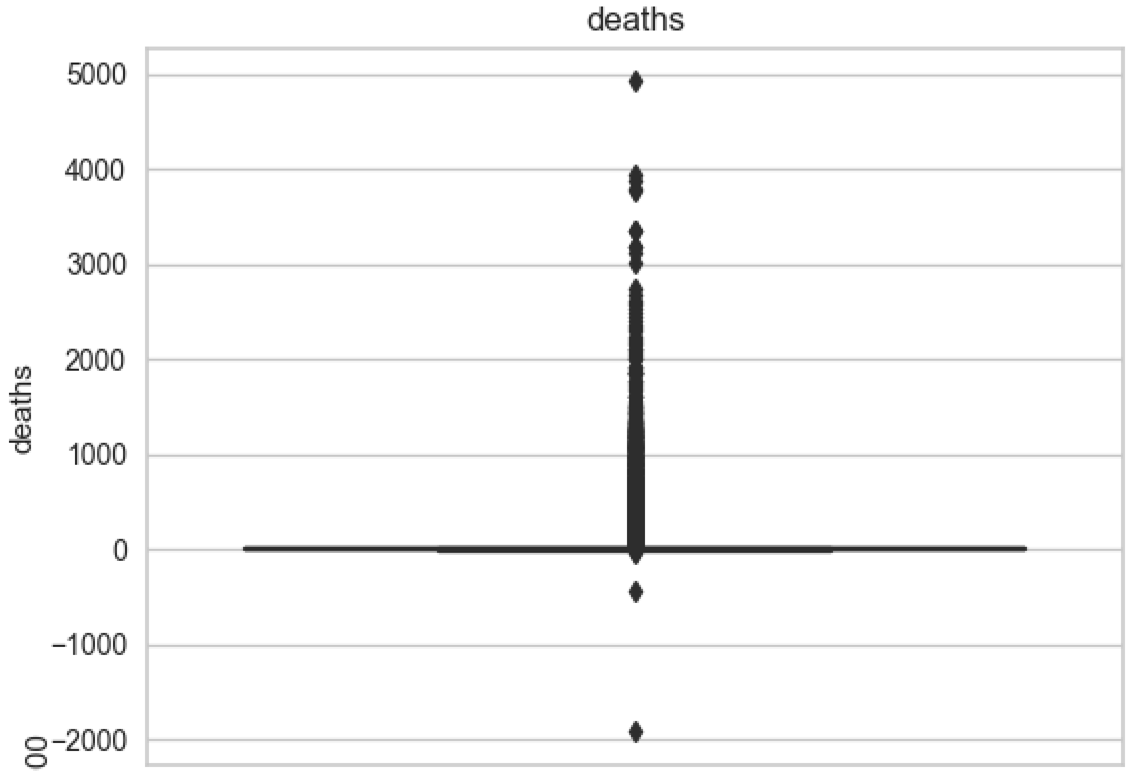
In [6]:

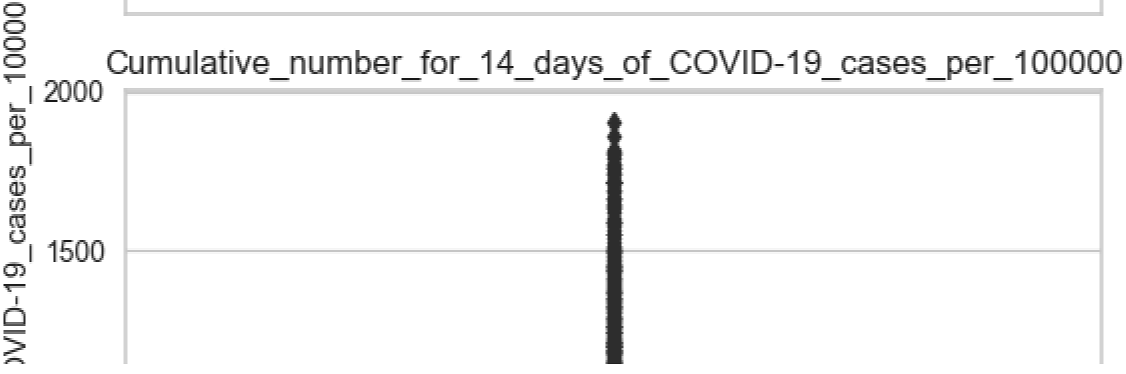
```python
#Extraer variables numericas
var_num = [var for var in df.columns if df[var].dtypes != 'O']
var_cat = [var for var in df.columns if df[var].dtypes == 'O']
```

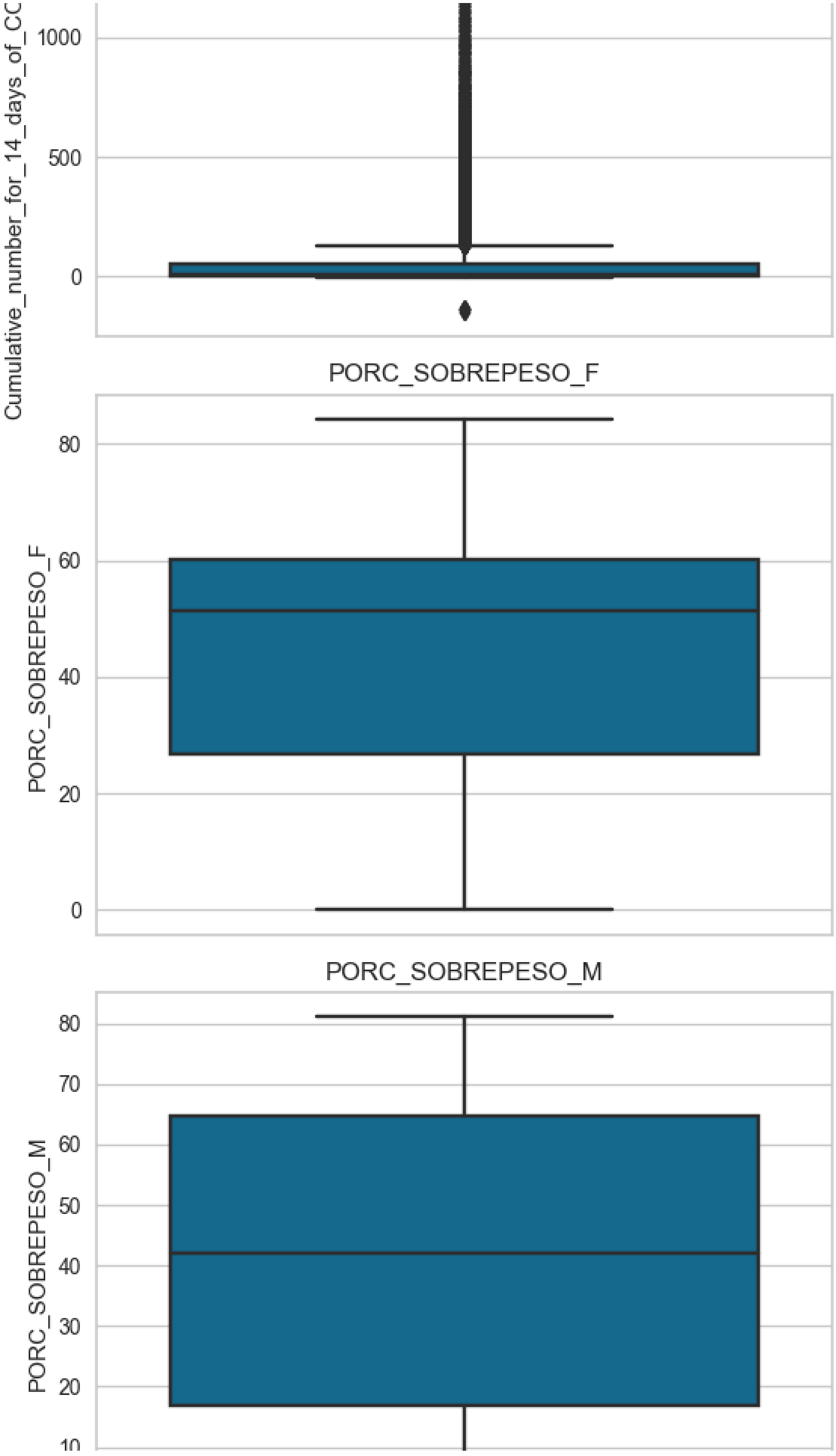
In [7]:

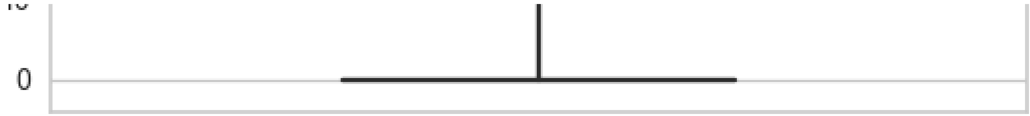
```python
# Crear la figura con subplots
fig, axs = plt.subplots(len(var_num), 1, figsize=(6, 4*len(var_num)))
axs = axs.flatten()

# Crear los gráficos de caja para las variables numéricas
for i, col in enumerate(var_num):
    sns.boxplot( y=col, data= df, ax=axs[i])
    axs[i].set_title(f"{col}")

# Ajustar los subplots y mostrar la figura
fig.tight_layout()
plt.show()
```


cases
cases
200000
150000
100000
50000
0


deaths
deaths
5000
4000
3000
2000
1000
0
−1000
−2000

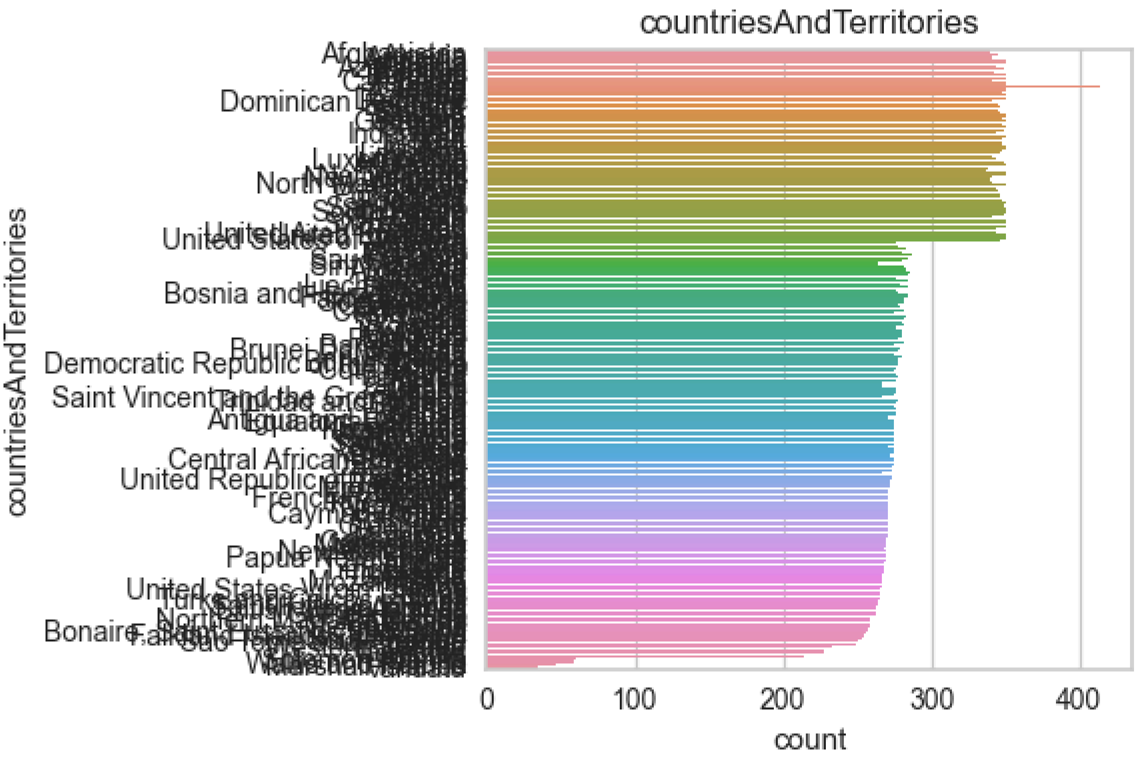

Cumulative_number_for_14_days_of_COVID-19_cases_per_100000
OVID-19_cases_per_100000
2000
1500

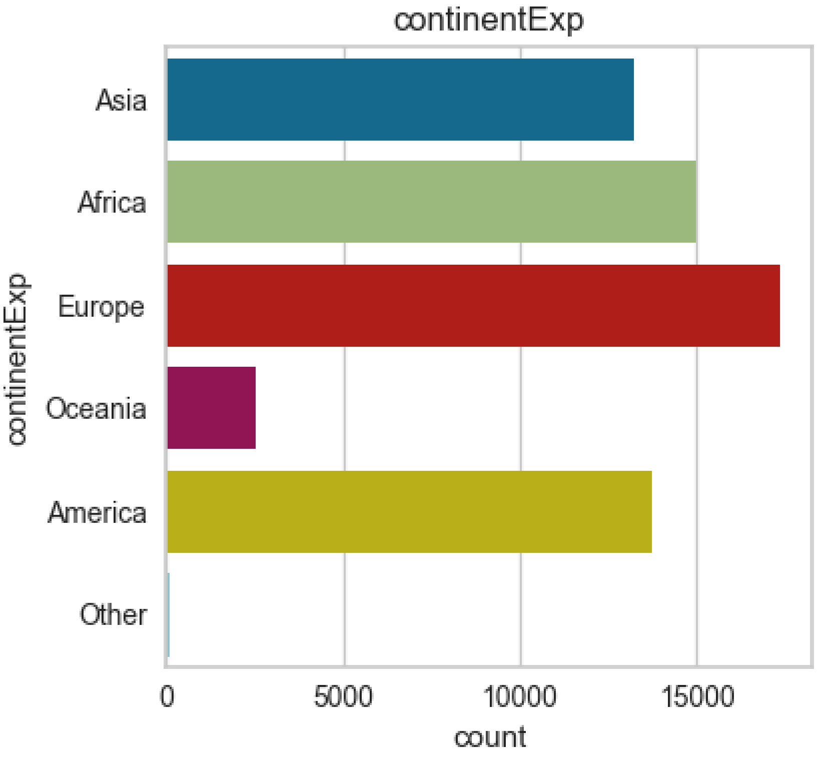

Cumulative_number_for_14_days_of_CC
1000
500
0
PORC_SOBREPESO_F
PORC_SOBREPESO_F
80
60
40
20
0
PORC_SOBREPESO_M
PORC_SOBREPESO_M
80
70
60
50
40
30
20
10



In [8]:

```
# Crear la figura con subplots
fig, axs = plt.subplots(len(var_cat), 1, figsize=(6, 4*len(var_cat)))
axs = axs.flatten()

# Crear los gráficos de caja para las variables numéricas
for i, col in enumerate(var_cat):
    sns.countplot( y=col, data= df, ax=axs[i])
    axs[i].set_title(f"{col}")

# Ajustar los subplots y mostrar la figura
fig.tight_layout()
plt.show()
```





## Modificación y Transformación

- en base al análisis anterior, se observan valores null's y outliers en las variables de casos y muertes

In [9]:

```
df_0 = df.dropna().reset_index().copy()
```

In [10]:

```
df_1 = df_0[['cases'
           , 'deaths'
           , 'Cumulative_number_for_14_days_of_COVID-19_cases_per_100000'
           , 'PORC_SOBREPESO_F'
           , 'PORC_SOBREPESO_M']].copy()
```

In [11]:

```
df_1.sample(5)
```

Out[11]:

| | cases | deaths | Cumulative_number_for_14_days_of_COVID-19_cases_per_100000 | PORC_SOBREPESO_F |
|---|---|---|---|---|
| **44127** | 33 | 0 | 2.185363 | 0.0 |
| **16710** | 0 | 0 | 0.000000 | 56.3 |
| **46558** | 15186 | 222 | 312.392809 | 61.0 |
| **10730** | 76 | 0 | 82.989878 | 67.2 |
| **56651** | 2043 | 30 | 354.339601 | 53.3 |

In [12]:

```
# escalado
escalado = StandardScaler()
df_scaler = escalado.fit_transform(df_1)
```

## Funciones adicionales

In [13]:

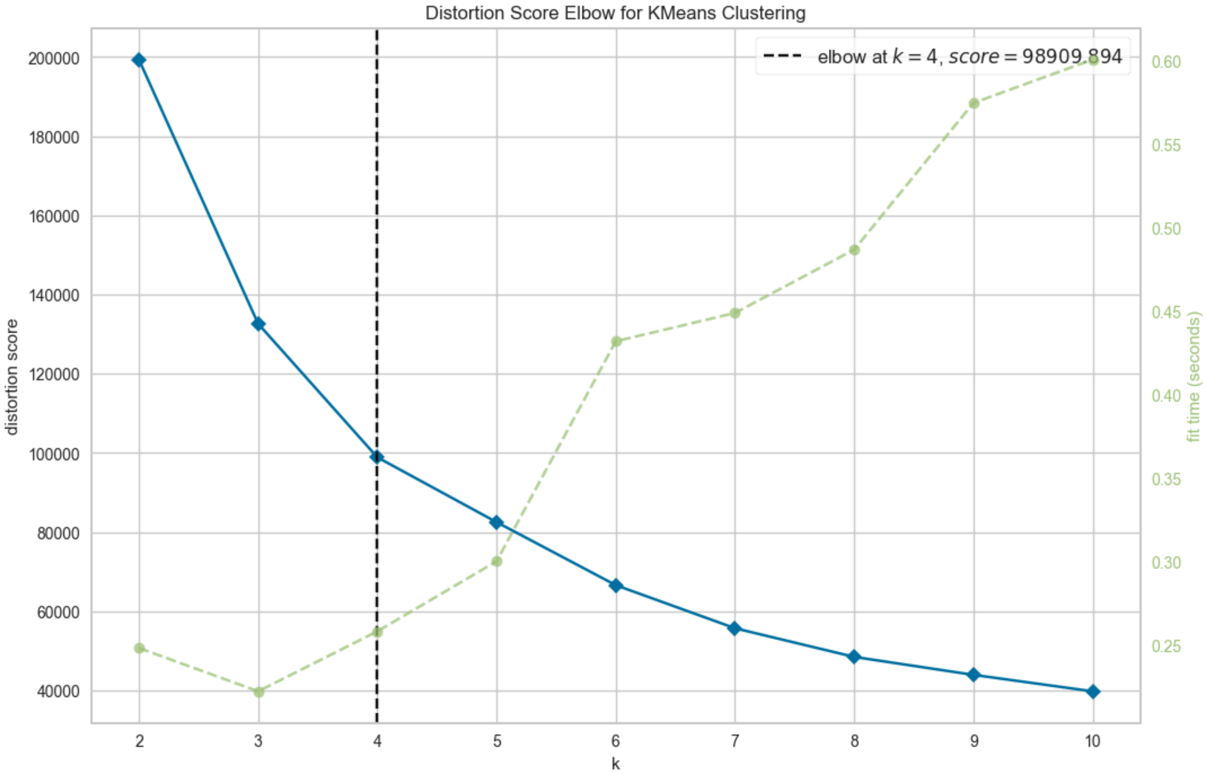
```
# Crea la función para calcular el centroide de los clusters
def calcular_centroide(x, y):
    return (np.sum(x)/len(x), np.sum(y)/len(y))# Crea la función para calcul
```

In [14]:

```
# Crear la función para Graficar las agrupaciones
def plot_graficos_clusters(atributos, predicciones, nombre_algoritmo):

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728','#9467bd', '#8c564b
    markers = ['*', '^', 'X', '+', 'D', 'H', 'o', '+', 's', 'v']

    plt.figure(figsize=(12,6))
    plt.title('Pronósticos de algoritmos' + nombre_algoritmo)

    for i in np.unique(predicciones):
                x = atributos[predicciones == i, 0]
                y = atributos[predicciones == i, 1]
                centroide = calcular_centroide(x,y)

                if i < 0:
                    nombre =  'Sin Cluster ' + str(i)
                    plt.scatter(x, y, s = 100, c = 'gray', label = nombre)
                    plt.scatter(centroide[0],  centroide[1], marker = marker
                else:
                    nombre = 'Cluster ' + str(i+1)
                    plt.scatter(x, y, s = 100, c = colors[i], label =  nombr
                    plt.scatter(centroide[0],  centroide[1], marker = marker

    plt.xlabel('Limite 1')
    plt.ylabel('Limite 2')
    plt.legend()
    plt.show()
```

# Modelo KNN

In [15]:
```
plt.figure(figsize=(12, 8))
g = KElbowVisualizer(KMeans(random_state=111), k=10)
g.fit(df_scaler)
g.show()
```



Out[15]:
```
<Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xl
abel='k', ylabel='distortion score'>
```

In [16]:
```
#Aplicamos el resultado de la tecnica del codo
grupos = KMeans(n_clusters=4, random_state= 111)
```

In [17]:
```
#Entrenamos el modelo
grupos.fit(df_scaler)
```

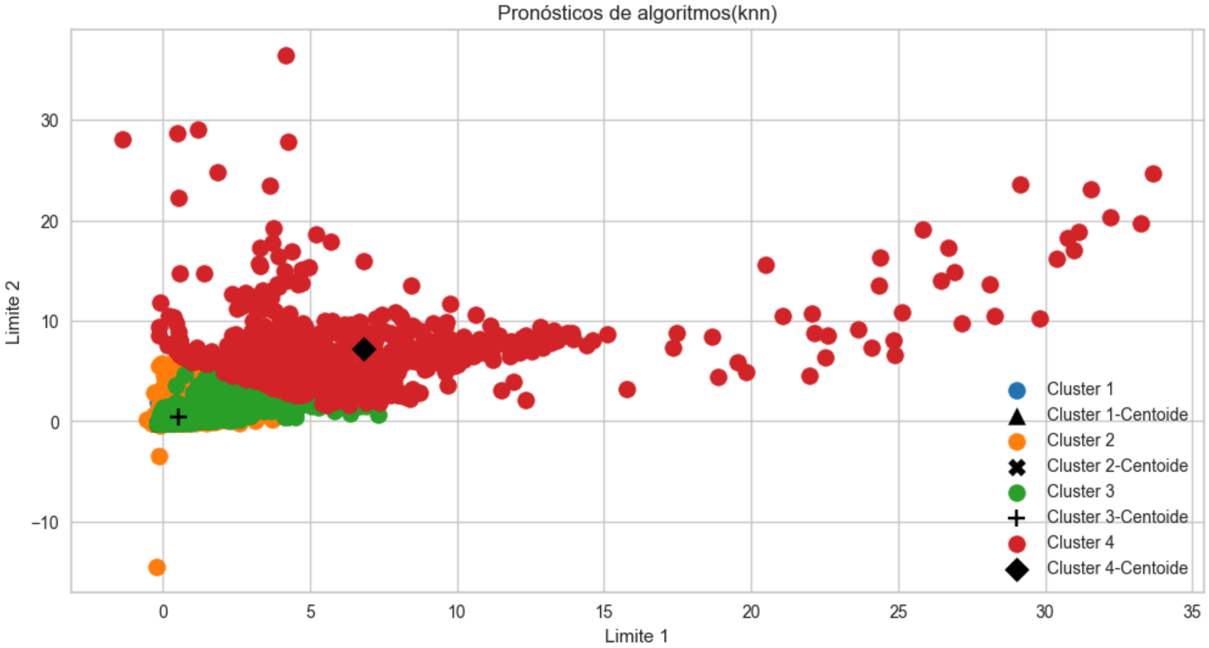
Out[17]:
```
▾            KMeans
KMeans(n_clusters=4, random_state=111)
```

In [18]:
```
#Obtenemos las etiquetas de grupos
```

```
pred_knn = grupos.predict(df_scaler)
```

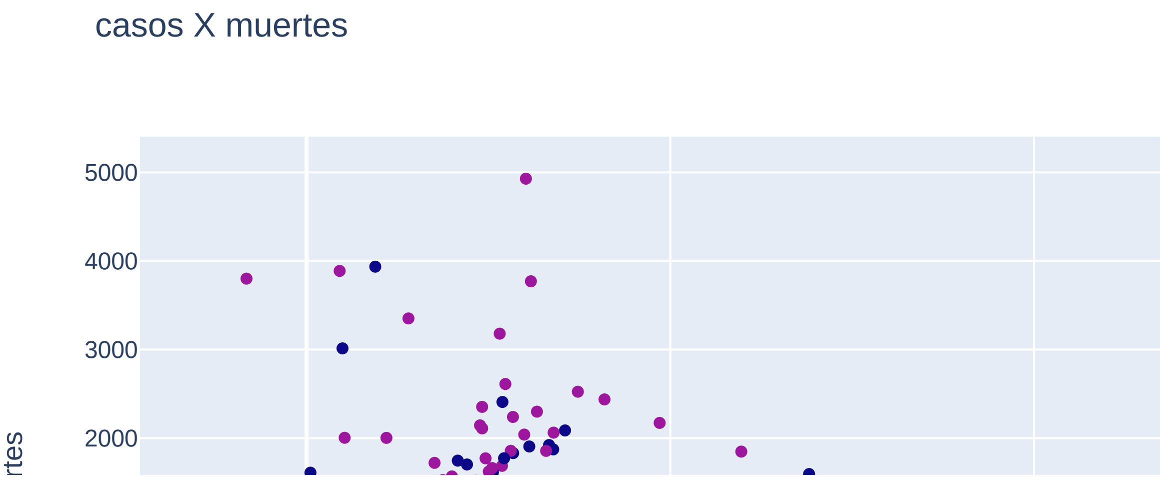
In [19]:

```
plot_graficos_clusters(df_scaler ,pred_knn, '(knn)')
```

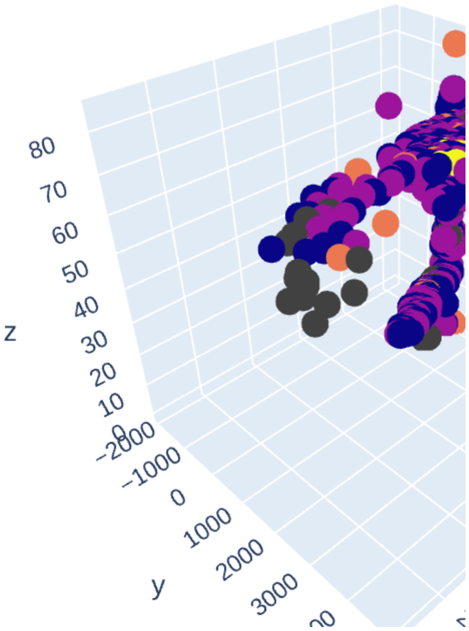


In [20]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['cases']
                         , y = df['deaths']
                         , mode = 'markers'
                         , marker = dict(color = pred_knn.astype(np.float64)
                         , text = pred_knn
                        )
             )
fig.update_layout(title = 'casos X muertes')
fig.update_xaxes(title = 'casos')
fig.update_yaxes(title = 'muertes')
fig.show()
```

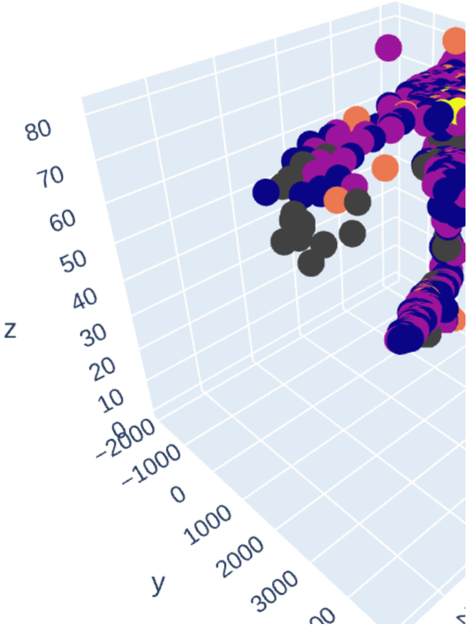


```
In [21]: #casos x muertes x indice de masa corpoal femenino
         fig = go.Figure()
         fig.add_trace(go.Scatter3d(x=df['cases'],
                                  y = df['deaths'], z = df['PORC_SOBREPESO_F'],
                                  mode = 'markers',
                                   marker = dict(color = pred_knn.astype(np.float64))
                                 text = [0, 1, 2]))
         fig.show()
```



In [22]:

```
#casos x muertes x indice de masa corpoal masculino
fig = go.Figure()
fig.add_trace(go.Scatter3d(x=df['cases'],
                         y = df['deaths'], z = df['PORC_SOBREPESO_M'],
                         mode = 'markers',
                          marker = dict(color = pred_knn.astype(np.float64))
                        text = [0, 1, 2]))
fig.show()
```



## GaussianMixture

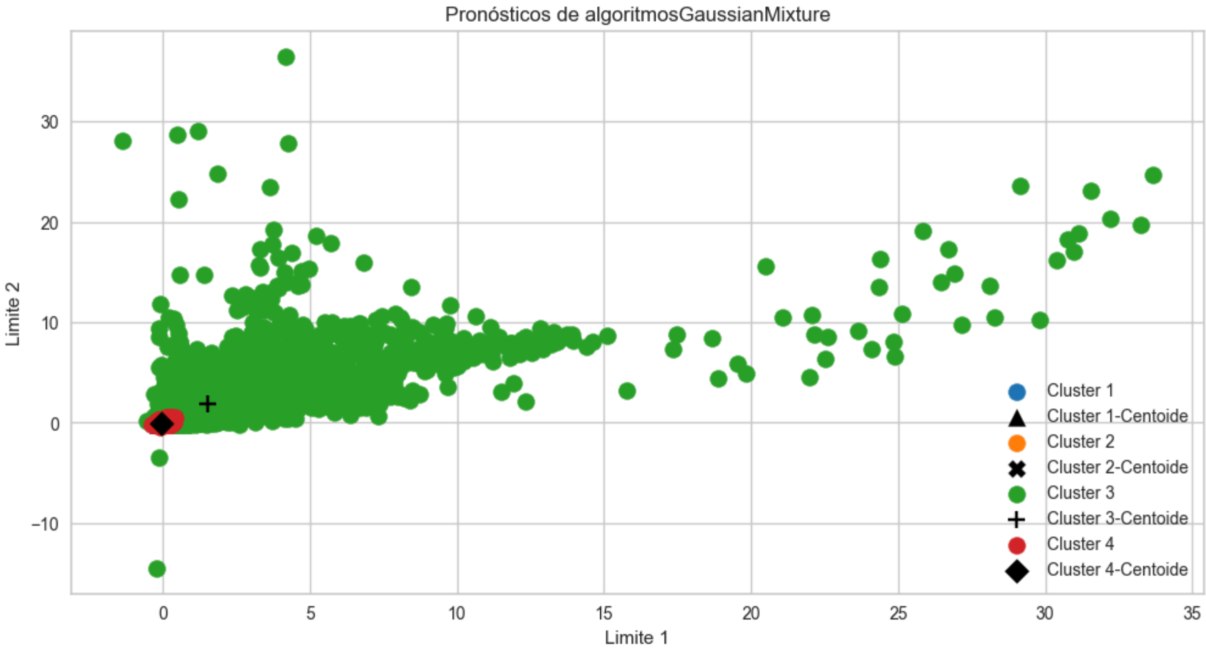
In [23]:
```
# Crear una instancia de Gaussian Mixture Model
gmm = GaussianMixture(n_components= 4, random_state= 111)
```

In [24]:
```
# Ajustar el modelo a los datos
gmm.fit(df_scaler)
```

Out[24]:
```
▾                    GaussianMixture

GaussianMixture(n_components=4, random_state=111)
```

In [25]:
```
# Obtener las etiquetas de los clusters
pred_gmm = gmm.predict(df_scaler)
```

In [26]:
```
plot_graficos_clusters(df_scaler ,pred_gmm, 'GaussianMixture')
```



In [27]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['cases'], y = df['deaths'],
                                                        mode = 'markers',
                                                        marker = dict(color
                                                        text = pred_gmm))

fig.update_layout(title = 'casos X muertes')
fig.update_xaxes(title = 'casos')
fig.update_yaxes(title = 'muertes')
fig.show()
```