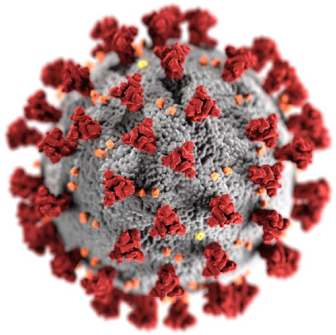
Universidad Oberta de Catalunya

Tipología y Ciclo de vida de los datos

WebScrapping

Detección de patrones existentes en diferentes indicadores con respecto a datos de la COVID-19



Autores:

Enrique Otero Espinosa

Francisco Javier Melchor González

8 de Noviembre de 2020

Contents

1 Contexto

La materia que trata el conjunto de datos generado, corresponde con los datos correspondientes con la pandemia COVID-19 desde que esta se empezó a manifestar en los diferentes países. Además, el conjunto de datos contiene indicadores importantes de los diferentes países (así como la expectance de vida, la fertilidad, el acceso a la electricidad, el número de tests realizados…) que desde nuestro punto de vista, consideramos que pueden llegar a tener cierta influencia en las muertes que ha ocasionado dicha pandemia en los diferentes países

Los sitios webs que se han elegido para extraer esta información, han sido:

- Wolrldometer, sitio web de referencia que proporciona estimaciones y estadísticas en tiempo real para diversos temas.
- WorldBankData, sitio web que sirve como herramienta de análisis y visualización que contiene colecciones de datos de diferentes temas.

Ambas fuentes, proporcionan esta información porque son páginas dedicadas a proporcionar información de diversos temas para que se puedan realizar análisis sobre los mismos.

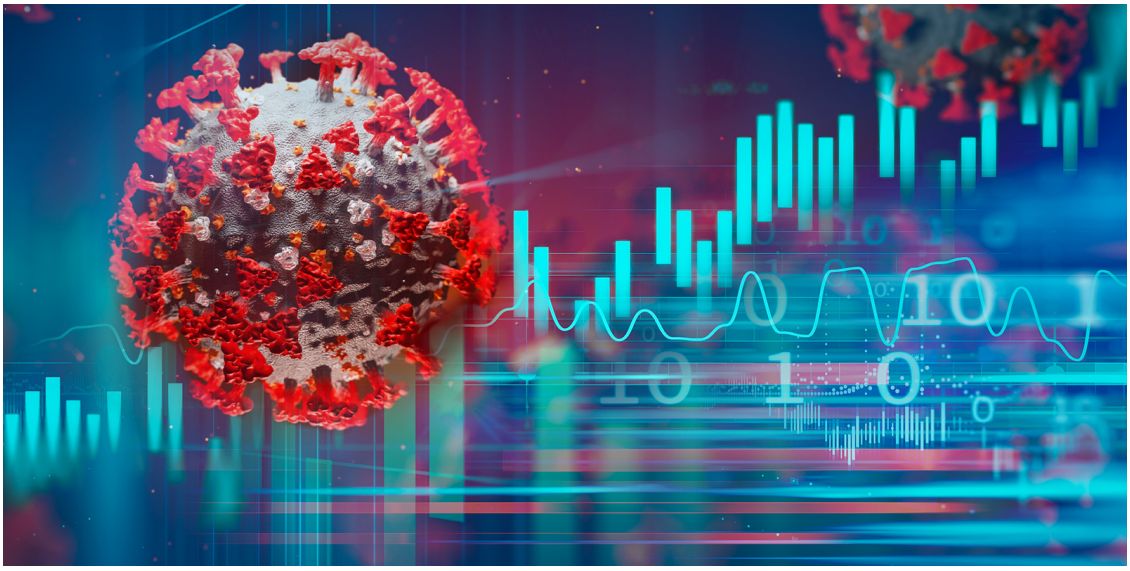
2 Título del dataset

Debido a que el dataset que se quiere conseguir fusiona información de la pandemia COVID-19 en los diferentes países y de los valores de los diferentes indicadores que se consideren a simple vista que pueden llegar a tener una relación con las muertes ocasionadas por la pandemia, el nombre que hemos elegido para el dataset que se va a generar es: **covid19AndWbIndicators**

3 Descripción del dataset

El conjunto de datos generado como parte de este proyecto reúne por un lado las métricas que permiten medir el impacto de la pandemia COVID-19, con respecto a las muertes y a los casos registrados en cada país, y por otro lado un conjunto de indicadores que se han considerado que pueden tener cierta relación con dicho impacto causado.

4 Representación gráfica



5 Contenido:

Los campos finales del dataset son 25 y provienen de las 2 fuentes de datos usadas. Los datos se obtienen en un dia concreto ya que el objetivo del dataset no es realizar análisis de series temporales o tendencias sino análisis estáticos como búsqueda de patrones, clustering o incluso análisis de regresiones lineales. Los campos son los siguientes:

- Campos obtenidos de la página de Worldometer dedicada a datos sobre la COVID-19
  - **ID:** Campo numérico identificador de país.
  - **TotalCases:** Número de contagiados totales de Coronavirus reportados desde el inicio de la pandemia. (float).
  - **TotalDeaths:** Número de fallecimientos totales por Coronavirus reportados desde el inicio de la pandemia. (float).
  - **TotalCases1M:** Número de contagiados por millón de habitantes. (float).
  - **TotalDeaths1M:** Número de fallecimientos por millón de habitantes. (float).
  - **TotalTests:** Número total de tests realizados. (float).
  - **TotalTests1M:** Número de tests realizados por millón de habitantes. (float).
  - **Population:** Población del pais. (float).
  - **Continent:** Continente.
  - **1CaseEvery:** Ratio entre número total de habitantes y número total de contagiados. (float).
  - **1DeathEvery:** Ratio entre número total de habitantes y número total de fallecimientos. (float).
  - **1TestEvery:** Ratio entre número total de habitantes y número total tests realizados. (float).
- Campos obtenidos de la página de WorldBankData, a través de la librería wbdata de Python con la que cuenta la propia página para poder acceder a los datos. Aunque es parametrizable, concretamente el dataset desarrollado en este proyecto, contiene índices recopilados durante

el año 2018, por ser este el último año que contiene datos más completos sobre todas las variables.

- **AccessElectricity:** Porcentaje de la población que tiene acceso a electricidad. (percent)
- **GDP:** PIB - Producto Interior Bruto del país (del inglés Gross Domestic Product). (float).
- **GDPperCap:** Producto Interior Bruto per cápita. (float).
- **LifeExpect:** Esperanza o expectativa de vida al nacer. (float).
- **DrsPer1k:** Número de médicos por cada 1000 habitantes. (float).
- **NewInfecHIV:** Nuevas personas infectadas con HIV desde el año anterior. (float).
- **FertilityRate:** Tasa de fertilidad o tasa de fecundidad. Número de nacimientos por cada mil mujeres en edad fértil (15-49 años) en un año. (float).
- **UrbanPopulation:** Habitantes viviendo en ciudades o población urbana. (float).
- **UrbanPopulationPerc:** Porcentaje de habitantes viviendo en ciudades. (percent)

6 Agradecimientos

Los datos han sido recolectados de dos fuente, como se ha indicado anteriormente: * En el caso de la página **Worldometer**, se ha hecho uso del la técnica de Web Scraping, con el lenguaje de programación Python. Esta página, está dirigida por un equipo internacional de desarrolladores, investigadores y voluntarios con el objetivo de hacer que las estadísticas mundiales estén disponibles en un formato que invite a la reflexión y sea relevante en el tiempo para una amplia audiencia en todo el mundo. Es publicado por una pequeña empresa de medios digitales independiente con sede en los Estados Unidos. Los datos son de dominio público y no poseen una categorización DOI.

- En el caso de la página **WorldBankData** también son de dominio público sin registro en DOI. Según los términos de uso es recomendable, aunque no obligatorio, incluir una cita que incluya el año y los indicadores usados de la forma:

  World Bank, 2018 indicators,

  “EG.ELC.ACCS.ZS”:“AccessElectricity”,“NY.GDP.MKTP.CD”:“GDP”,*

  “NY.GDP.PCAP.CD”:“GDPperCap”,“SH.XPD.CHEX.PC.CD”:“HealthExpenseperCap”,

  “IT.NET.USER.ZS”:“IndividUsingInternet”,“SP.DYN.LE00.IN”:“LifeExpect”,

  “SH.MED.PHYS.ZS”:“DrsPer1k”,“GB.XPD.RSDV.GD.ZS”:“RDExpen”,

  “SH.HIV.INCD”:“NewInfecHIV”,“SP.DYN.TFRT.IN”:“FertilityRate”,

  “per_si_allsi.cov_pop_tot”:“CovSocialInsurance”,

  “SP.URB.TOTL”:“UrbanPopulation”, “SP.URB.TOTL.IN.ZS”:“UrbanPopulationPerc”

  Por otro lado, la librería **wbdata** utilizada para obtener los datos del WorldBankData, ha sido desarrollada por Oliver Sherouse. En la descripción de la librería establece recomendable, aunque no obligatorio incluir una cita de la siguiente forma:

  Sherouse, Oliver (2014). Wbdata. Arlington, VA.
  Available from http://github.com/OliverSherouse/wbdata.13

Por último, no se han encontrado conjuntos de datos o proyectos que contengan un dataset con ambas fuentes de datos como el desarrollado en este proyecto

7 Inspiración

Actualmente, debido a la pandemia COVID-19 estamos pasando por una situación que está marcando un hecho histórico a nivel mundial. Han sido muchas las muertes causadas por la misma, las secuelas dejadas en algunas personas y lo mucho que ha afectado esta a nivel económico a los diferentes países del mundo. Es por ello, que consideramos muy interesante tratar datos con respecto a la misma y relacionarlos con los diferentes indicadores de los diferentes países, para tratar de encontrar así alguna relación existente entre los valores de los diferentes indicadores, y el nivel con el que está afectando la pandemia a cada uno de los países analizados.

La pregunta que se trata de responder realizando esta recolección de datos y este anális son: * ¿Qué indicadores están relacionados con el nivel de afectación de la pandemia en cada país? * ¿Por qué dichos indicadores están relacionados con el nivel de afectación de la pandemia a cada país?

Para tratar así de estudiar la relación y tratar de extraer como conclusión a que se debe que un país tenga más casos que otros (con respecto a los indicadores)

La manera de medier el nivel de afectación de la pandemia que hemos tomado en este proyecto es el número de muertes causadas por la misma en cada país

8 Licencia

La licencia elegida para el dataset desarrollado es la siguiente: * **Released Under CC0: Public Domain License**

Debido a que las fuentes de las cuales se han extraído los datos, son ambas de dominio público.

9 Código

[1]:
```
import wbdata
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
from pylab import rcParams
from sklearn.ensemble import ExtraTreesClassifier
import statistics
from geopy.geocoders import Nominatim
import math
import folium
from folium.plugins import HeatMap
from sklearn import preprocessing
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.cluster import KMeans
import seaborn as sns
```

9.1 Extracción de datos

[2]:
```
url = 'https://www.worldometers.info/coronavirus/'
requests.get(url)
wdweb = requests.get(url)
```

[3]:
```
wdweb
```

[3]:
```
<Response [200]>
```

[4]:
```
wdwebsoup = BeautifulSoup(wdweb.text, 'lxml')
```

[5]:
```
wdwebtable_data = wdwebsoup.find('table', id = 'main_table_countries_yesterday')
```

[6]:
```
headers = []
for i in wdwebtable_data.find_all('th'):
    title = i.text
    headers.append(title)
```

[ ]:
```

```

[7]:
```
covid = pd.DataFrame(columns = headers)
```

[8]:
```
for j in wdwebtable_data.find_all('tr')[1:]:
        row_data = j.find_all('td')
        row = [td.text for td in row_data]
        length = len(covid)
        covid.loc[length] = row
```

[9]:
```
covid.columns =␣
↪['ID','country','TotalCases','NewCases','TotalDeaths','NewDeaths','TotalRecov','NewRecov','
covid.set_index('country', inplace=True, drop=True)
```

[10]:
```
covid.shape
```

[10]:
```
(234, 18)
```

[11]:
```
covid.index
```

[11]:
```
Index(['\nAsia\n', '\nNorth America\n', '\nSouth America\n', '\nEurope\n',
       '\nAfrica\n', '\nOceania\n', '\n\n', 'World', 'China', 'USA',
       …
       'Marshall Islands', 'Wallis and Futuna', 'Total:', 'Total:', 'Total:',
       'Total:', 'Total:', 'Total:', 'Total:', 'Total:'],
      dtype='object', name='country', length=234)
```

[12]:
```
pd.set_option('display.max_rows', None)
covid.index=covid.index.str.replace("\n","")
covid=covid.rename(index={'USA': 'United States'})
```

[13]:
```
data_date = datetime.datetime(2018, 12, 31), datetime.datetime(2018, 12, 31)
# countries = [i['id'] for i in wbdata.get_country(incomelevel='HIC')]
indicators = {"EG.ELC.ACCS.ZS":"AccessElectricity", "NY.GDP.MKTP.CD":"GDP",
              "NY.GDP.PCAP.CD":"GDPperCap", "SH.XPD.CHEX.PC.CD":
 ↪"HealthExpenseperCap",
               "IT.NET.USER.ZS":"IndividUsingInternet", "SP.DYN.LE00.IN":
 ↪"LifeExpect",
               "SH.MED.PHYS.ZS":"DrsPer1k", "GB.XPD.RSDV.GD.ZS":"RDExpen",
               "SH.HIV.INCD":"NewInfecHIV", "SP.DYN.TFRT.IN":"FertilityRate",
              "per_si_allsi.cov_pop_tot":"CovSocialInsurance",
              "SP.URB.TOTL":"UrbanPopulation", "SP.URB.TOTL.IN.ZS":
 ↪"UrbanPopulationPerc"}
wbdf = wbdata.get_dataframe(indicators, country="all", data_date=data_date)
```

[14]:
```
covidandwb_merged = pd.merge(covid,wbdf, on=["country"])
```

[15]:
```
covidandwb_merged.head()
```

[15]:

country	ID	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecov	\
North America		11,960,330	+146,616	361,791	+1,914	7,817,698	
World		49,649,261	+623,277	1,248,195	+9,205	35,261,462	
China	1	86,151	+36	4,634		81,098	
United States	2	10,058,586	+132,540	242,230	+1,248	6,391,208	
India	3	8,460,885	+49,851	125,605	+576	7,818,558	

country	NewRecov	ActiveCases	SeriousCritical	TotalCases1M	…	\
North America	+60,371	3,780,841	21,991		…	
World	+284,386	13,139,604	90,830	6,370	…	
China	+17	419	9	60	…	
United States	+49,508	3,425,148	18,303	30,326	…	
India	+53,795	516,722	8,944	6,110	…	

country	HealthExpenseperCap	IndividUsingInternet	LifeExpect	DrsPer1k	\
North America	None	88.498903	78.886891	NaN	
World	None	NaN	72.563274	NaN	
China	None	NaN	76.704000	NaN	
United States	None	88.498903	78.539024	NaN	
India	None	20.081300	69.416000	0.8571	

	RDExpen	NewInfecHIV	FertilityRate	CovSocialInsurance	\

```
country
North America  2.739569         NaN      1.706032                NaN
World          2.273640   1400000.0      2.414975                NaN
China          2.185680         NaN      1.690000                NaN
United States  2.837660     33000.0      1.729500                NaN
India          0.649980         NaN      2.222000                NaN

               UrbanPopulation  UrbanPopulationPerc
country
North America     2.989531e+08            82.173045
World             4.195080e+09            55.270426
China             8.238276e+08            59.152000
United States     2.687201e+08            82.256000
India             4.602957e+08            34.030000

[5 rows x 31 columns]
```

[16]:
```
covidandwb_merged.shape
```

[16]:
```
(172, 31)
```

[17]:
```
covidandwb = covidandwb_merged
```

9.2 Procesamiento de los datos obtenidos

[18]:
```
covidandwb = covidandwb.drop(["NewCases", "NewDeaths", "TotalRecov", "NewRecov",
                              "ActiveCases", "SeriousCritical"], axis=1)
covidandwb = covidandwb.drop(["World"], axis=0)
covidandwb = covidandwb.drop(["North America"], axis=0)
```

[19]:
```
covidandwb.head()
```

[19]:
```
                ID  TotalCases TotalDeaths TotalCases1M TotalDeaths1M  \
country
China            1      86,151      4,634           60             3
United States    2  10,058,586    242,230       30,326           730
India            3   8,460,885    125,605        6,110            91
Brazil           4   5,632,505    162,035       26,433           760
France           6   1,661,853     39,865       25,440           610

                 TotalTests TotalTests1M      Population      Continent  \
country
China           160,000,000      111,163  1,439,323,776           Asia
United States   155,148,732      467,759    331,685,117  North America
India           115,429,095       83,357  1,384,752,705           Asia
Brazil           21,900,000      102,774    213,089,916  South America
France           17,367,177      265,860     65,324,507         Europe
```

```
              1CaseEvery  … HealthExpenseperCap IndividUsingInternet  \
country                   …
China             16,707  …                None                  NaN
United States         33  …                None            88.498903
India                164  …                None            20.081300
Brazil                38  …                None            70.434283
France                39  …                None            82.043187

               LifeExpect  DrsPer1k  RDExpen NewInfecHIV  FertilityRate  \
country
China           76.704000       NaN  2.18568         NaN         1.6900
United States   78.539024       NaN  2.83766     33000.0         1.7295
India           69.416000    0.8571  0.64998         NaN         2.2220
Brazil          75.672000    2.1643      NaN         NaN         1.7300
France          82.724390    3.2672  2.20002         NaN         1.8800

               CovSocialInsurance  UrbanPopulation  UrbanPopulationPerc
country
China                         NaN      823827650.0                59.152
United States                 NaN      268720071.0                82.256
India                         NaN      460295677.0                34.030
Brazil                  30.869548      181335507.0                86.569
France                        NaN       53870058.0                80.444

[5 rows x 25 columns]
```

[20]:
```
TotalDeathsindex = covidandwb[covidandwb['TotalDeaths'].str.match(' ')].index
covidandwb.loc[TotalDeathsindex, 'TotalDeaths'] = 0
covidandwb['TotalDeaths']=covidandwb['TotalDeaths'].str.replace(",","").
 ↪astype(float)
covidandwb.loc[TotalDeathsindex, 'TotalDeaths1M'] = 0
covidandwb['TotalDeaths1M']=covidandwb['TotalDeaths1M'].str.replace(",","").
 ↪astype(float)
covidandwb['TotalCases1M']=covidandwb['TotalCases1M'].str.replace(",","").
 ↪astype(float)
covidandwb['TotalCases']=covidandwb['TotalCases'].str.replace(",","").
 ↪astype(float)
```

[21]:
```
covidandwb.head()
```

[21]:
```
              ID  TotalCases  TotalDeaths  TotalCases1M  TotalDeaths1M  \
country
China          1     86151.0       4634.0          60.0            3.0
United States  2  10058586.0     242230.0       30326.0          730.0
India          3   8460885.0     125605.0        6110.0           91.0
Brazil         4   5632505.0     162035.0       26433.0          760.0
```

```
France         6   1661853.0      39865.0       25440.0          610.0

               TotalTests TotalTests1M      Population      Continent  \
country
China         160,000,000      111,163  1,439,323,776           Asia
United States 155,148,732      467,759    331,685,117  North America
India         115,429,095       83,357  1,384,752,705           Asia
Brazil         21,900,000      102,774    213,089,916  South America
France         17,367,177      265,860     65,324,507         Europe

              1CaseEvery  … HealthExpenseperCap IndividUsingInternet  \
country                   …
China             16,707  …               None                  NaN
United States         33  …               None            88.498903
India                164  …               None            20.081300
Brazil                38  …               None            70.434283
France                39  …               None            82.043187

              LifeExpect  DrsPer1k  RDExpen NewInfecHIV  FertilityRate  \
country
China          76.704000       NaN  2.18568         NaN         1.6900
United States  78.539024       NaN  2.83766     33000.0         1.7295
India          69.416000    0.8571  0.64998         NaN         2.2220
Brazil         75.672000    2.1643      NaN         NaN         1.7300
France         82.724390    3.2672  2.20002         NaN         1.8800

              CovSocialInsurance  UrbanPopulation  UrbanPopulationPerc
country
China                        NaN      823827650.0               59.152
United States                NaN      268720071.0               82.256
India                        NaN      460295677.0               34.030
Brazil                 30.869548      181335507.0               86.569
France                       NaN       53870058.0               80.444

[5 rows x 25 columns]
```

[22]:
```
covidandwb.dtypes
```

[22]:
```
ID                    object
TotalCases           float64
TotalDeaths          float64
TotalCases1M         float64
TotalDeaths1M        float64
TotalTests            object
TotalTests1M          object
Population            object
Continent             object
```

```
1CaseEvery                object
1DeathEvery               object
1TestEvery                object
AccessElectricity        float64
GDP                      float64
GDPperCap                float64
HealthExpenseperCap       object
IndividUsingInternet     float64
LifeExpect               float64
DrsPer1k                 float64
RDExpen                  float64
NewInfecHIV              float64
FertilityRate            float64
CovSocialInsurance       float64
UrbanPopulation          float64
UrbanPopulationPerc      float64
dtype: object
```

[23]:
```
covidandwb['1DeathEvery']= covidandwb['1DeathEvery'].str.replace(",","")
covidandwb['1DeathEvery']=covidandwb['1DeathEvery'].str.replace(r'^\s*$','NaN')
covidandwb['1DeathEvery']=covidandwb['1DeathEvery'].astype(float)
```

[24]:
```
covidandwb['TotalTests']= covidandwb['TotalTests'].str.replace(",","")
covidandwb['TotalTests']=covidandwb['TotalTests'].str.replace(r'^\s*$','NaN')
covidandwb['TotalTests']=covidandwb['TotalTests'].astype(float)
```

[25]:
```
covidandwb['TotalTests1M']= covidandwb['TotalTests1M'].str.replace(",","")
covidandwb['TotalTests1M']=covidandwb['TotalTests1M'].str.
 ↪replace(r'^\s*$','NaN')
covidandwb['TotalTests1M']=covidandwb['TotalTests1M'].astype(float)
```

[26]:
```
covidandwb['Population']= covidandwb['Population'].str.replace(",","")
covidandwb['Population']=covidandwb['Population'].str.replace(r'^\s*$','NaN')
covidandwb['Population']=covidandwb['Population'].astype(float)
```

[27]:
```
covidandwb['1CaseEvery']= covidandwb['1CaseEvery'].str.replace(",","")
covidandwb['1CaseEvery']=covidandwb['1CaseEvery'].str.replace(r'^\s*$','NaN')
covidandwb['1CaseEvery']=covidandwb['1CaseEvery'].astype(float)
```

[28]:
```
covidandwb['HealthExpenseperCap']= covidandwb['HealthExpenseperCap'].str.
 ↪replace(",","")
covidandwb['HealthExpenseperCap']=covidandwb['HealthExpenseperCap'].str.
 ↪replace(r'^\s*$','NaN')
covidandwb['HealthExpenseperCap']=covidandwb['HealthExpenseperCap'].
 ↪astype(float)
```

[29]:
```
covidandwb['1TestEvery']= covidandwb['1TestEvery'].str.replace(",","")
covidandwb['1TestEvery']=covidandwb['1TestEvery'].str.replace(r'^\s*$','NaN')
covidandwb['1TestEvery']=covidandwb['1TestEvery'].astype(float)
```

[30]:
```
covidandwb.dtypes
```

[30]:
```
ID                      object
TotalCases             float64
TotalDeaths            float64
TotalCases1M           float64
TotalDeaths1M          float64
TotalTests             float64
TotalTests1M           float64
Population             float64
Continent               object
1CaseEvery             float64
1DeathEvery            float64
1TestEvery             float64
AccessElectricity      float64
GDP                    float64
GDPperCap              float64
HealthExpenseperCap    float64
IndividUsingInternet   float64
LifeExpect             float64
DrsPer1k               float64
RDExpen                float64
NewInfecHIV            float64
FertilityRate          float64
CovSocialInsurance     float64
UrbanPopulation        float64
UrbanPopulationPerc    float64
dtype: object
```

[31]:
```
covidandwb.shape
```

[31]:
```
(170, 25)
```

[32]:
```
covidandwb.isnull().sum()
```

[32]:
```
ID                      0
TotalCases              0
TotalDeaths            13
TotalCases1M            0
TotalDeaths1M          13
TotalTests             13
TotalTests1M           13
Population              0
```

```
Continent                 0
1CaseEvery                0
1DeathEvery              13
1TestEvery               13
AccessElectricity         1
GDP                      12
GDPperCap                12
HealthExpenseperCap     170
IndividUsingInternet     97
LifeExpect               10
DrsPer1k                114
RDExpen                 110
NewInfecHIV              66
FertilityRate             9
CovSocialInsurance      164
UrbanPopulation           1
UrbanPopulationPerc       1
dtype: int64
```

[33]:
```
covidandwb.columns
```

[33]:
```
Index(['ID', 'TotalCases', 'TotalDeaths', 'TotalCases1M', 'TotalDeaths1M',
       'TotalTests', 'TotalTests1M', 'Population', 'Continent', '1CaseEvery',
       '1DeathEvery', '1TestEvery', 'AccessElectricity', 'GDP', 'GDPperCap',
       'HealthExpenseperCap', 'IndividUsingInternet', 'LifeExpect', 'DrsPer1k',
       'RDExpen', 'NewInfecHIV', 'FertilityRate', 'CovSocialInsurance',
       'UrbanPopulation', 'UrbanPopulationPerc'],
      dtype='object')
```

Eliminamos: * **CovSocialInsurance** * **RDExpen** * **HealthExpenseperCap** * **IndividUsing-Internet**

Debido a que todas ellas cuentan con más de un 60% de datos nulos

[34]:
```
columns_acceptables = ['ID', 'TotalCases', 'TotalDeaths', 'TotalCases1M',␣
↪'TotalDeaths1M',
       'TotalTests', 'TotalTests1M', 'Population', 'Continent', '1CaseEvery',
       '1DeathEvery', '1TestEvery', 'AccessElectricity', 'GDP', 'GDPperCap',␣
↪'LifeExpect','DrsPer1k', 'NewInfecHIV', 'FertilityRate',
       'UrbanPopulation', 'UrbanPopulationPerc']
```

[35]:
```
covidandwb = covidandwb[columns_acceptables]
covidandwb.isnull().sum()
```

[35]:
```
ID                    0
TotalCases            0
TotalDeaths          13
TotalCases1M          0
```

```
TotalDeaths1M           13
TotalTests              13
TotalTests1M            13
Population               0
Continent                0
1CaseEvery               0
1DeathEvery             13
1TestEvery              13
AccessElectricity        1
GDP                     12
GDPperCap               12
LifeExpect              10
DrsPer1k               114
NewInfecHIV             66
FertilityRate            9
UrbanPopulation          1
UrbanPopulationPerc      1
dtype: int64
```

[36]:
```
def imputationFunct(x, indexColumn):
    if math.isnan(x.iloc[indexColumn]):
        x.iloc[indexColumn] = statistics.median(covidandwb.
↪loc[covidandwb['Continent'] == x.iloc[8]].iloc[:, indexColumn].dropna())
    return x.iloc[indexColumn]
```

[37]:
```
columnsWithNaN = [2,4,5,6,10,11,12,13,14,15,16,17,18,19,20]
```

9.2.1 Imputation Data

[38]:
```
for column in columnsWithNaN:
    covidandwb.iloc[:,[column]] = covidandwb.
↪apply(imputationFunct,axis=1,args=(column,))
```

[39]:
```
covidandwb.isnull().sum()
```

[39]:
```
ID                    0
TotalCases            0
TotalDeaths           0
TotalCases1M          0
TotalDeaths1M         0
TotalTests            0
TotalTests1M          0
Population            0
Continent             0
1CaseEvery            0
1DeathEvery           0
```

```
1TestEvery             0
AccessElectricity      0
GDP                    0
GDPperCap              0
LifeExpect             0
DrsPer1k               0
NewInfecHIV            0
FertilityRate          0
UrbanPopulation        0
UrbanPopulationPerc    0
dtype: int64
```

[40]:
```
covidandwb.head()
```

[40]:

country	ID	TotalCases	TotalDeaths	TotalCases1M	TotalDeaths1M	\
China	1	86151.0	4634.0	60.0	3.0	
United States	2	10058586.0	242230.0	30326.0	730.0	
India	3	8460885.0	125605.0	6110.0	91.0	
Brazil	4	5632505.0	162035.0	26433.0	760.0	
France	6	1661853.0	39865.0	25440.0	610.0	

country	TotalTests	TotalTests1M	Population	Continent	\
China	160000000.0	111163.0	1.439324e+09	Asia	
United States	155148732.0	467759.0	3.316851e+08	North America	
India	115429095.0	83357.0	1.384753e+09	Asia	
Brazil	21900000.0	102774.0	2.130899e+08	South America	
France	17367177.0	265860.0	6.532451e+07	Europe	

country	1CaseEvery	…	1TestEvery	AccessElectricity	GDP	\
		…				
China	16707.0	…	9.0	100.000000	1.389482e+13	
United States	33.0	…	2.0	100.000000	2.052905e+13	
India	164.0	…	12.0	95.235855	2.713165e+12	
Brazil	38.0	…	10.0	100.000000	1.885483e+12	
France	39.0	…	4.0	100.000000	2.787864e+12	

country	GDPperCap	LifeExpect	DrsPer1k	NewInfecHIV	FertilityRate	\
China	9976.676822	76.704000	0.9186	1050.0	1.6900	
United States	62840.020239	78.539024	1.9357	33000.0	1.7295	
India	2005.863005	69.416000	0.8571	1050.0	2.2220	
Brazil	9001.234249	75.672000	2.1643	2300.0	1.7300	
France	41631.090739	82.724390	3.2672	500.0	1.8800	

country	UrbanPopulation	UrbanPopulationPerc

```
country
China              823827650.0              59.152
United States      268720071.0              82.256
India              460295677.0              34.030
Brazil             181335507.0              86.569
France              53870058.0              80.444

[5 rows x 21 columns]
```

[41]:
```
geolocator = Nominatim(user_agent='myapplication')
```

[42]:
```
def getLatitude(x):
    return geolocator.geocode(x[0]).latitude
```

[43]:
```
def getLongitude(x):
    return geolocator.geocode(x[0]).longitude
```

[44]:
```
covidandwb.head()
```

[44]:

country	ID	TotalCases	TotalDeaths	TotalCases1M	TotalDeaths1M	\
China	1	86151.0	4634.0	60.0	3.0	
United States	2	10058586.0	242230.0	30326.0	730.0	
India	3	8460885.0	125605.0	6110.0	91.0	
Brazil	4	5632505.0	162035.0	26433.0	760.0	
France	6	1661853.0	39865.0	25440.0	610.0	

country	TotalTests	TotalTests1M	Population	Continent	\
China	160000000.0	111163.0	1.439324e+09	Asia	
United States	155148732.0	467759.0	3.316851e+08	North America	
India	115429095.0	83357.0	1.384753e+09	Asia	
Brazil	21900000.0	102774.0	2.130899e+08	South America	
France	17367177.0	265860.0	6.532451e+07	Europe	

country	1CaseEvery	…	1TestEvery	AccessElectricity	GDP	\
China	16707.0	…	9.0	100.000000	1.389482e+13	
United States	33.0	…	2.0	100.000000	2.052905e+13	
India	164.0	…	12.0	95.235855	2.713165e+12	
Brazil	38.0	…	10.0	100.000000	1.885483e+12	
France	39.0	…	4.0	100.000000	2.787864e+12	

country	GDPperCap	LifeExpect	DrsPer1k	NewInfecHIV	FertilityRate	\
China	9976.676822	76.704000	0.9186	1050.0	1.6900	
United States	62840.020239	78.539024	1.9357	33000.0	1.7295	

```
India           2005.863005   69.416000    0.8571       1050.0         2.2220
Brazil          9001.234249   75.672000    2.1643       2300.0         1.7300
France         41631.090739   82.724390    3.2672        500.0         1.8800

               UrbanPopulation  UrbanPopulationPerc
country
China             823827650.0               59.152
United States     268720071.0               82.256
India             460295677.0               34.030
Brazil            181335507.0               86.569
France             53870058.0               80.444

[5 rows x 21 columns]
```

[45]:
```
covidandwb = covidandwb.reset_index()
covidandwb.head()
```

[45]:
```
         country ID  TotalCases  TotalDeaths  TotalCases1M  TotalDeaths1M  \
0          China  1     86151.0       4634.0          60.0            3.0
1  United States  2  10058586.0     242230.0       30326.0          730.0
2          India  3   8460885.0     125605.0        6110.0           91.0
3         Brazil  4   5632505.0     162035.0       26433.0          760.0
4         France  6   1661853.0      39865.0       25440.0          610.0

    TotalTests  TotalTests1M    Population      Continent  …  1TestEvery  \
0  160000000.0      111163.0  1.439324e+09           Asia  …         9.0
1  155148732.0      467759.0  3.316851e+08  North America  …         2.0
2  115429095.0       83357.0  1.384753e+09           Asia  …        12.0
3   21900000.0      102774.0  2.130899e+08  South America  …        10.0
4   17367177.0      265860.0  6.532451e+07         Europe  …         4.0

   AccessElectricity           GDP     GDPperCap  LifeExpect  DrsPer1k  \
0         100.000000  1.389482e+13   9976.676822   76.704000    0.9186
1         100.000000  2.052905e+13  62840.020239   78.539024    1.9357
2          95.235855  2.713165e+12   2005.863005   69.416000    0.8571
3         100.000000  1.885483e+12   9001.234249   75.672000    2.1643
4         100.000000  2.787864e+12  41631.090739   82.724390    3.2672

   NewInfecHIV  FertilityRate  UrbanPopulation  UrbanPopulationPerc
0       1050.0         1.6900      823827650.0               59.152
1      33000.0         1.7295      268720071.0               82.256
2       1050.0         2.2220      460295677.0               34.030
3       2300.0         1.7300      181335507.0               86.569
4        500.0         1.8800       53870058.0               80.444

[5 rows x 22 columns]
```

[46]:
```
covidandwb['Latitude'] = covidandwb.apply(getLatitude,axis=1)
```

[47]:
```
covidandwb['Longitude'] = covidandwb.apply(getLongitude,axis=1)
```

[48]:
```
covidandwb.head()
```

[48]:

	country	ID	TotalCases	TotalDeaths	TotalCases1M	TotalDeaths1M	\
0	China	1	86151.0	4634.0	60.0	3.0	
1	United States	2	10058586.0	242230.0	30326.0	730.0	
2	India	3	8460885.0	125605.0	6110.0	91.0	
3	Brazil	4	5632505.0	162035.0	26433.0	760.0	
4	France	6	1661853.0	39865.0	25440.0	610.0	

	TotalTests	TotalTests1M	Population	Continent	…	GDP	\
0	160000000.0	111163.0	1.439324e+09	Asia	…	1.389482e+13	
1	155148732.0	467759.0	3.316851e+08	North America	…	2.052905e+13	
2	115429095.0	83357.0	1.384753e+09	Asia	…	2.713165e+12	
3	21900000.0	102774.0	2.130899e+08	South America	…	1.885483e+12	
4	17367177.0	265860.0	6.532451e+07	Europe	…	2.787864e+12	

	GDPperCap	LifeExpect	DrsPer1k	NewInfecHIV	FertilityRate	\
0	9976.676822	76.704000	0.9186	1050.0	1.6900	
1	62840.020239	78.539024	1.9357	33000.0	1.7295	
2	2005.863005	69.416000	0.8571	1050.0	2.2220	
3	9001.234249	75.672000	2.1643	2300.0	1.7300	
4	41631.090739	82.724390	3.2672	500.0	1.8800	

	UrbanPopulation	UrbanPopulationPerc	Latitude	Longitude
0	823827650.0	59.152	35.000074	104.999927
1	268720071.0	82.256	39.783730	-100.445882
2	460295677.0	34.030	22.351115	78.667743
3	181335507.0	86.569	-10.333333	-53.200000
4	53870058.0	80.444	46.603354	1.888334

[5 rows x 24 columns]

9.3 Representaciones gráficas de los datos

9.3.1 HeatMap

[49]:
```
map_hooray = folium.Map([40.4166359,-3.7059988], zoom_start=3)
```

[50]:
```
HeatMap(data=covidandwb[['Latitude','Longitude','TotalCases1M']].
↪groupby(['Latitude','Longitude']).sum().reset_index().values.tolist()).
↪add_to(map_hooray)
```

[50]:
```
<folium.plugins.heat_map.HeatMap at 0x176b90cedd8>
```

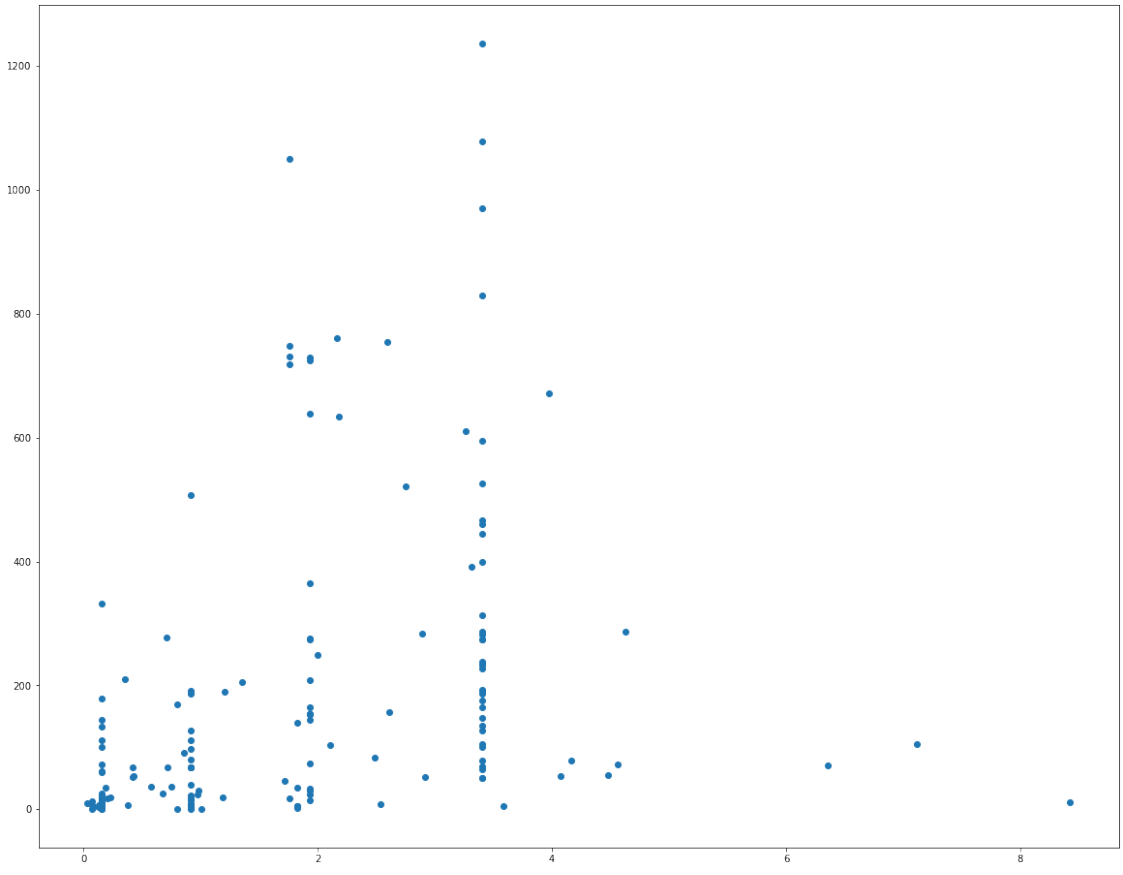
[51]:
```
map_hooray
```

[51]:
```
<folium.folium.Map at 0x176b90ce4e0>
```

[52]:
```
covidandwb.set_index('country', inplace=True, drop=True)
```

9.3.2 Plots of relations

[53]:
```
%matplotlib inline
rcParams['figure.figsize'] = 20,16
plt.scatter(covidandwb['DrsPer1k'], covidandwb['TotalDeaths1M'])
plt.show()
```



[54]:
```
# Recreamos el campo UrbanPopulationPerc
```

[55]:
```
covidandwb['UrbanPopulationPerc'].head()
```

[55]:
```
country
China            59.152
United States    82.256
```

```
India            34.030
Brazil           86.569
France           80.444
Name: UrbanPopulationPerc, dtype: float64
```

[56]:
```
def fx(x, y):
    return x*100/y
covidandwb['UrbanPopulationPerc'] = np.
 ↪vectorize(fx)(covidandwb['UrbanPopulation'], covidandwb['Population'])
```

[57]:
```
covidandwb['UrbanPopulationPerc'].head()
```

[57]:
```
country
China            57.237132
United States    81.016620
India            33.240280
Brazil           85.098117
France           82.465311
Name: UrbanPopulationPerc, dtype: float64
```

[58]:
```
# Exportamos el Datframe final a CSV
```

[59]:
```
csv_path='./covidandwb.csv'
covidandwb.to_csv(csv_path, index=False, header=True)
```

9.3.3 Clusterización con KMeans

[60]:
```
covidkmeans_cols=['TotalDeaths1M', 'TotalTests1M', 'GDPperCap',␣
 ↪'LifeExpect','DrsPer1k', 'NewInfecHIV', 'FertilityRate',
       'UrbanPopulation']
covidkmeans=covidandwb[covidkmeans_cols]
covidkmeans.head()
```

[60]:
```
               TotalDeaths1M  TotalTests1M     GDPperCap  LifeExpect  \
country
China                    3.0      111163.0   9976.676822   76.704000
United States          730.0      467759.0  62840.020239   78.539024
India                   91.0       83357.0   2005.863005   69.416000
Brazil                 760.0      102774.0   9001.234249   75.672000
France                 610.0      265860.0  41631.090739   82.724390

               DrsPer1k  NewInfecHIV  FertilityRate  UrbanPopulation
country
China            0.9186       1050.0         1.6900      823827650.0
United States    1.9357      33000.0         1.7295      268720071.0
India            0.8571       1050.0         2.2220      460295677.0
Brazil           2.1643       2300.0         1.7300      181335507.0
```

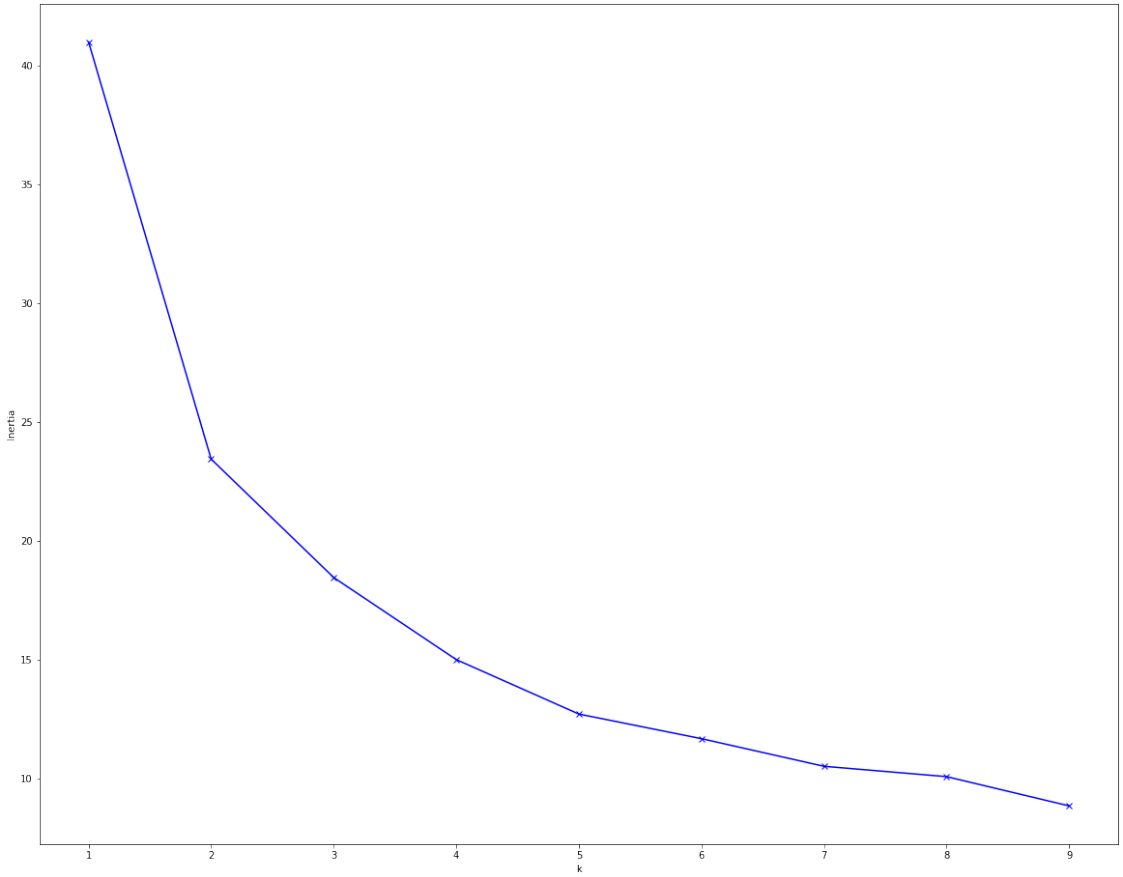
```
France          3.2672        500.0         1.8800       53870058.0
```

[61]:
```
scaler = preprocessing.MinMaxScaler()
features_normal = scaler.fit_transform(covidkmeans)
```

[62]:
```
features_normal
```

[62]:
```
array([[2.36070239e-03, 5.28858340e-02, 5.23015077e-02, …,
        5.00263296e-03, 9.52710896e-02, 1.00000000e+00],
       [5.90110921e-01, 2.22790218e-01, 3.37191152e-01, …,
        1.73249078e-01, 1.02113286e-01, 3.26180371e-01],
       [7.35051580e-02, 3.96373363e-02, 9.34542203e-03, …,
        5.00263296e-03, 1.87424216e-01, 5.58725212e-01],
       …,
       [1.24438120e-01, 9.61023516e-02, 2.92088828e-01, …,
        5.00263296e-03, 1.48969340e-01, 5.24397124e-05],
       [3.97762183e-03, 0.00000000e+00, 1.00558641e-02, …,
        2.63296472e-03, 5.65217391e-01, 1.81605931e-04],
       [3.97762183e-03, 2.64436317e-02, 1.89505462e-02, …,
        2.63296472e-03, 1.43772735e-01, 4.80200695e-05]])
```

[63]:
```
inertia = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(features_normal)
    kmeanModel.fit(features_normal)
    inertia.append(kmeanModel.inertia_)
plt.plot(K, inertia, 'bx-')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.show()
```



[64]:
```
kmeans = KMeans(n_clusters=5).fit(features_normal)
```

[65]:
```
covidkmeans.head()
```

[65]:

country	TotalDeaths1M	TotalTests1M	GDPperCap	LifeExpect	DrsPer1k	NewInfecHIV	FertilityRate	UrbanPopulation
China	3.0	111163.0	9976.676822	76.704000	0.9186	1050.0	1.6900	823827650.0
United States	730.0	467759.0	62840.020239	78.539024	1.9357	33000.0	1.7295	268720071.0
India	91.0	83357.0	2005.863005	69.416000	0.8571	1050.0	2.2220	460295677.0
Brazil	760.0	102774.0	9001.234249	75.672000	2.1643	2300.0	1.7300	181335507.0
France	610.0	265860.0	41631.090739	82.724390	3.2672	500.0	1.8800	53870058.0

[66]:
```
labels = pd.DataFrame(kmeans.labels_) #This is where the label output of the␣
↪KMeans we just ran lives. Make it a dataframe so we can concatenate back to␣
↪the original data
covidkmeans=covidkmeans.assign(Country=covidkmeans.index.
↪get_level_values('country'))
covidkmeans.reset_index(drop=True, inplace=True)
labeledcovidkmeans = pd.concat((covidkmeans,labels), axis=1)
labeledcovidkmeans = labeledcovidkmeans.rename({0:'labels'},axis=1)
```

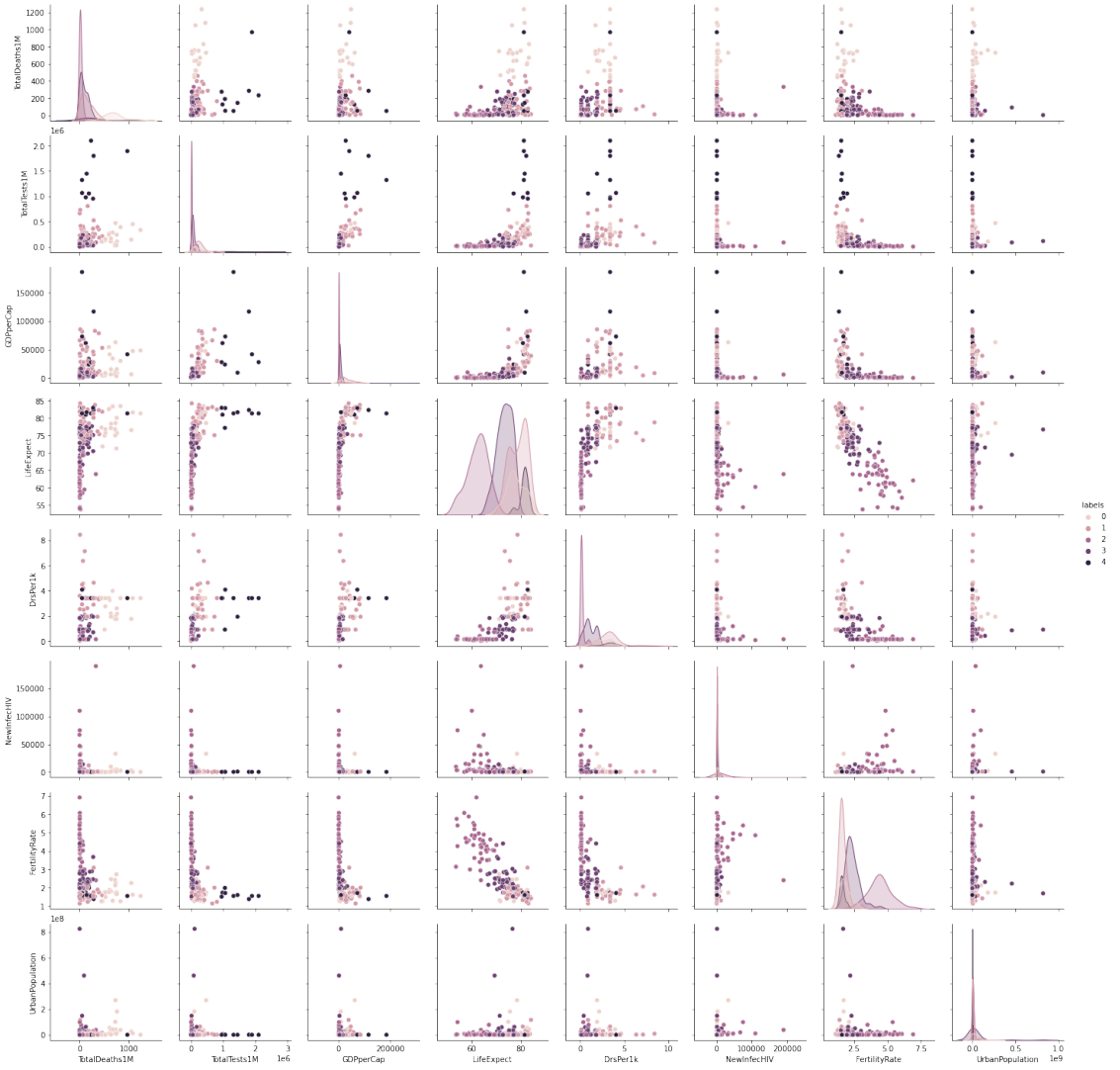
[67]:
```
labeledcovidkmeans.head()
```

[67]:

	TotalDeaths1M	TotalTests1M	GDPperCap	LifeExpect	DrsPer1k	NewInfecHIV	FertilityRate	UrbanPopulation	Country	labels
0	3.0	111163.0	9976.676822	76.704000	0.9186	1050.0	1.6900	823827650.0	China	3
1	730.0	467759.0	62840.020239	78.539024	1.9357	33000.0	1.7295	268720071.0	United States	0
2	91.0	83357.0	2005.863005	69.416000	0.8571	1050.0	2.2220	460295677.0	India	3
3	760.0	102774.0	9001.234249	75.672000	2.1643	2300.0	1.7300	181335507.0	Brazil	0
4	610.0	265860.0	41631.090739	82.724390	3.2672	500.0	1.8800	53870058.0	France	0

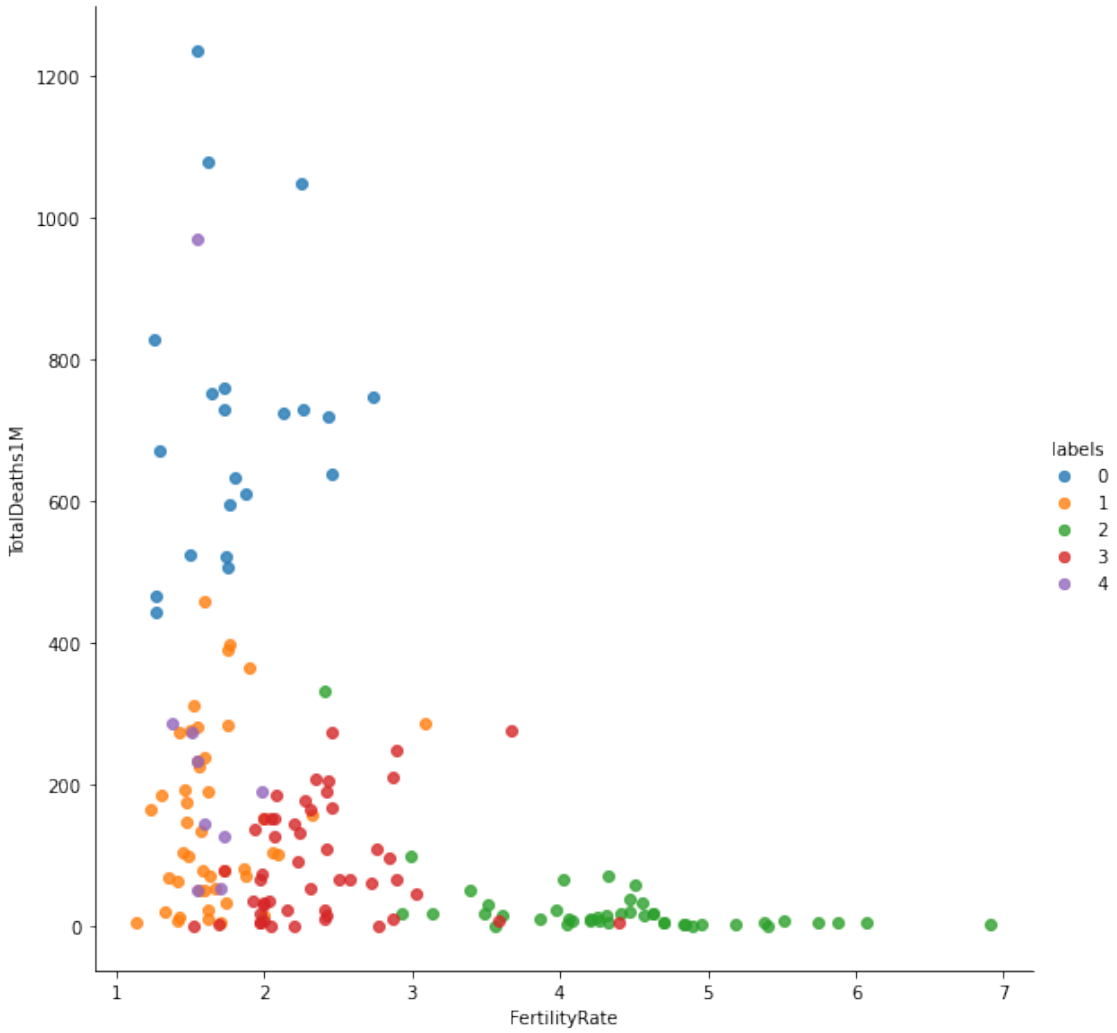
[68]:
```
sns.pairplot(labeledcovidkmeans,hue='labels')
```

[68]:
```
<seaborn.axisgrid.PairGrid at 0x176b989a6d8>
```



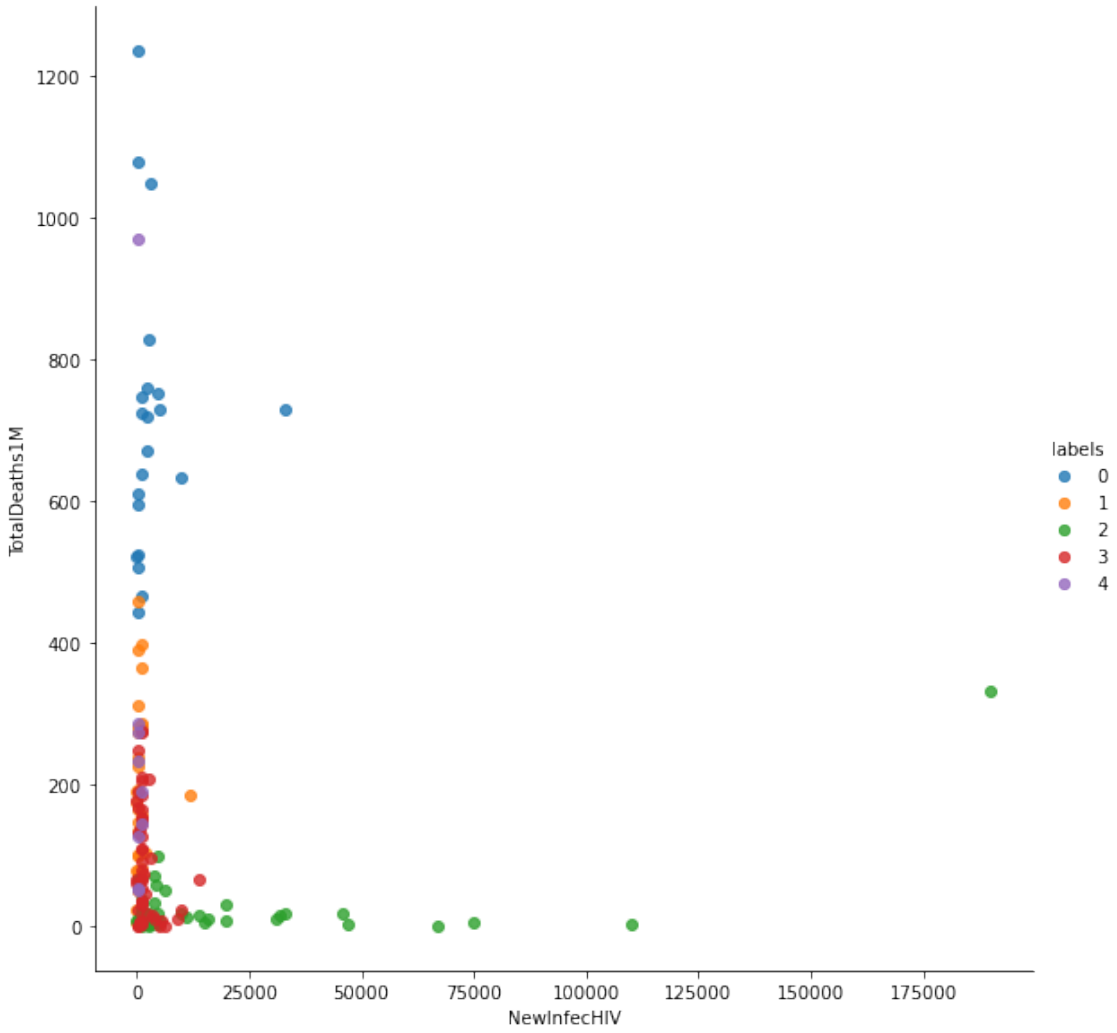
[69]:
```
sns.
↪lmplot(x='FertilityRate',y='TotalDeaths1M',data=labeledcovidkmeans,hue='labels',fit_reg=Fal
↪height=8)
```

[69]: <seaborn.axisgrid.FacetGrid at 0x176bf4edf28>



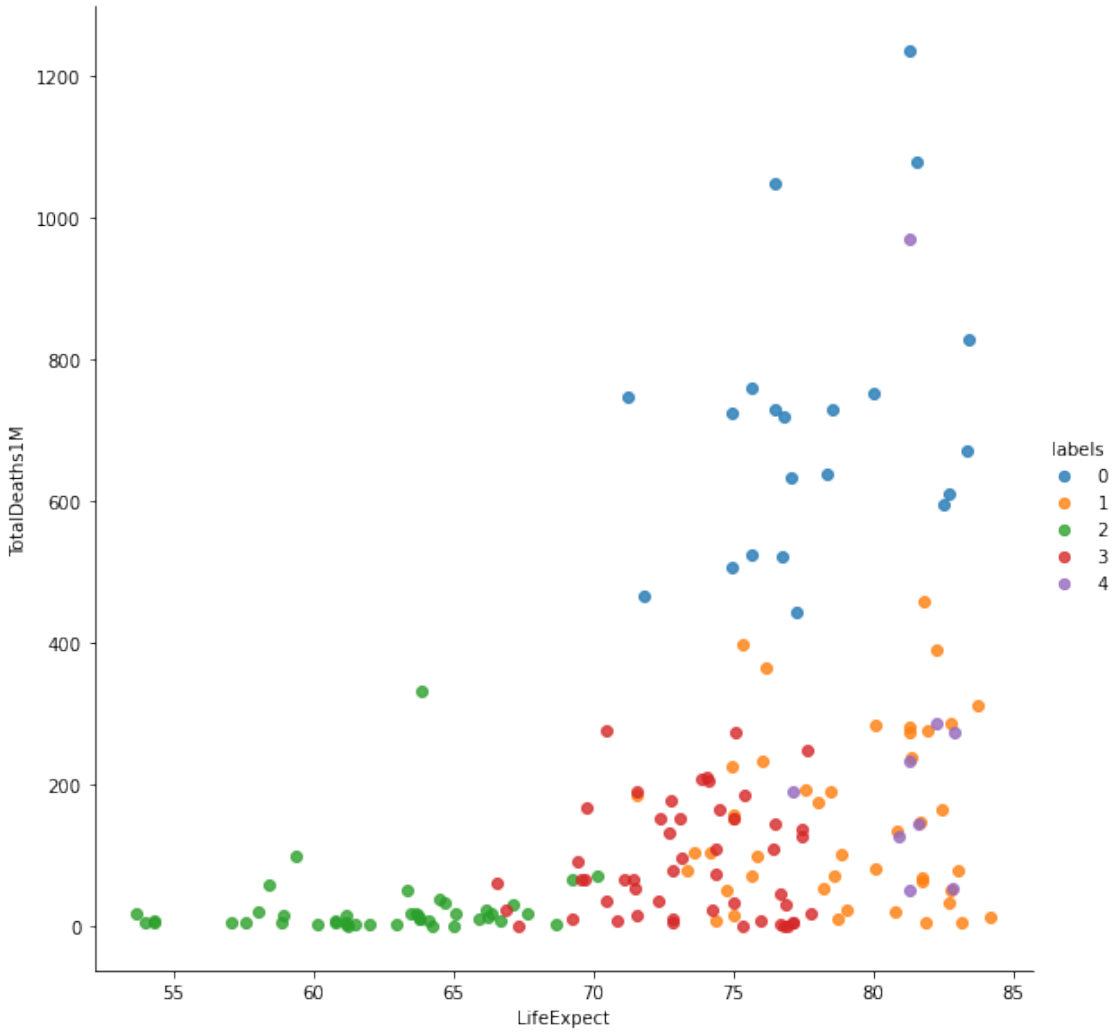
[70]:
```
sns.
↪lmplot(x='NewInfecHIV',y='TotalDeaths1M',data=labeledcovidkmeans,hue='labels',fit_reg=False
```

[70]: <seaborn.axisgrid.FacetGrid at 0x176bf4ed710>



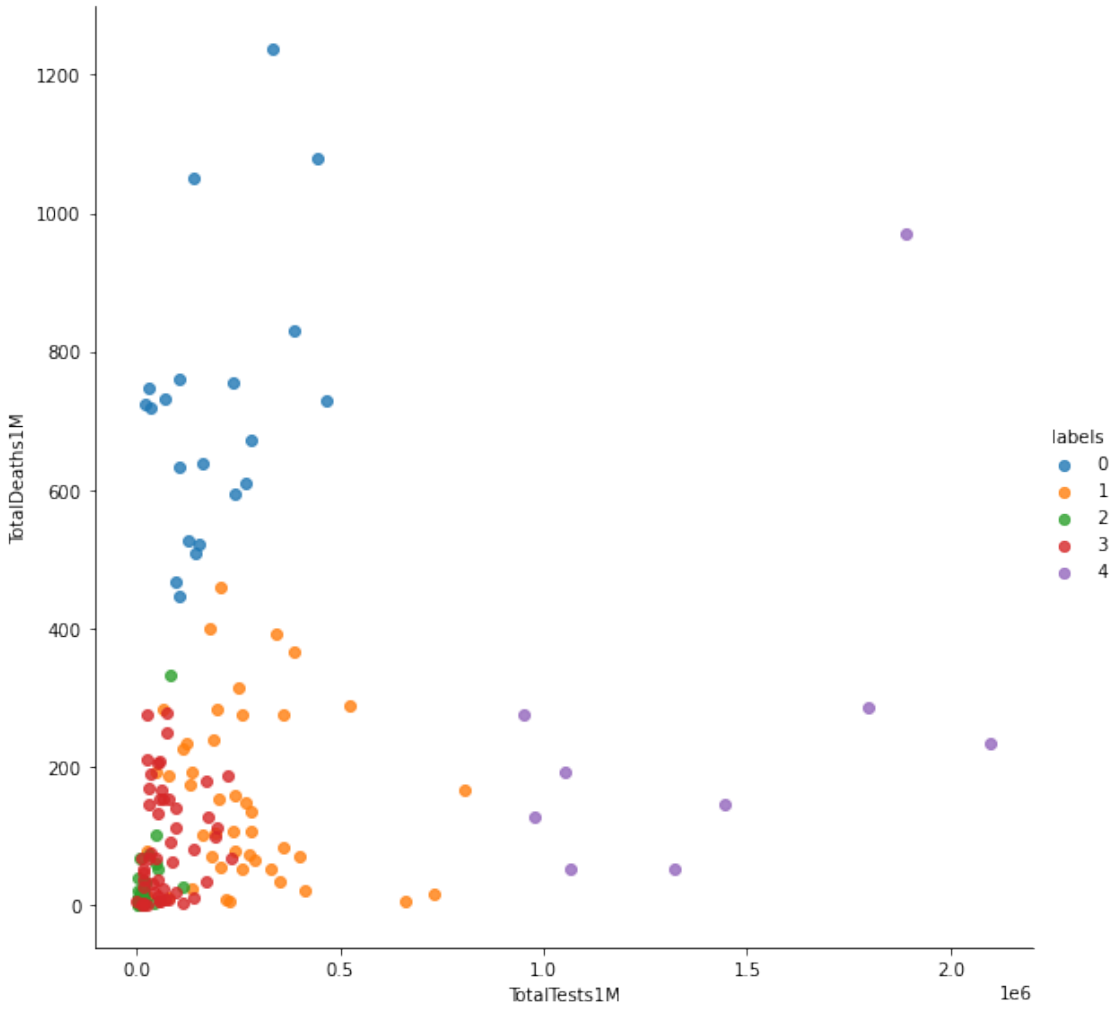
[71]:
```
sns.
↪lmplot(x='LifeExpect',y='TotalDeaths1M',data=labeledcovidkmeans,hue='labels',fit_reg=False,
↪height=8)
```

[71]: <seaborn.axisgrid.FacetGrid at 0x176c0f20080>



[72]:
```
sns.
↪lmplot(x='TotalTests1M',y='TotalDeaths1M',data=labeledcovidkmeans,hue='labels',fit_reg=Fals
↪height=8)
```

[72]: <seaborn.axisgrid.FacetGrid at 0x176c0f9ba90>

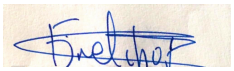
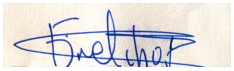
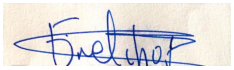
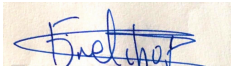
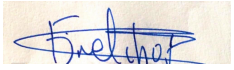
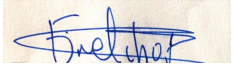
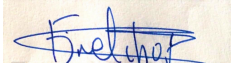
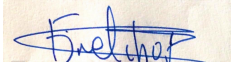


Finalmente, atendiendo a los clustering obtenidos por el modelo, se detecta que hay un grupo de países donde los parámetros parecen indicar que son países subdesarrollados, en los cuales se detectan un menor número de muertes debido a la pandemia de la COVID-19

10 Dataset:

Tras publicar el dataset en Zenodo el DOI obtenido es el siguiente: 10.5281/zenodo.4256839

DOI

Contribuciones	Firma	
Investigación Inicial		
Análisis de herramientas Scrappping		
Codigo python Scrapping y app		
Investigación de Indicadores WorldBank		
Obtención de datos Covid y WorldBank		
Procesado y limpieza de Datos		
Análisis ejemplo Clustering		
Export de DataSet		
Documentacion