7

Ciencia y analítica de datos

Double-click (or enter) to edit

Entrega: 1 Limpieza, análisis, visualización y kmeans

Nombre: Fernando Anaya Delgado

Matricula: A01793832

Nombre: Christian Emilio Saldana Lopez

Matricula: A00506509

Equipo:95 Profesora: María de la Paz Rico Fernández Fecha: 16/11/2022

```
# This is formatted as code
```

Conclusión del análisis.

Después de realizar el estudio eliminando las columnas innecesarios e información irrelevante, como los valores nulos y flotantes se procede a dividir cuales son las variables categóricas y numéricas para poder comenzar nuestro análisis.

Se puede observar que las diferentes graficas que llegan a tener sentido pero sin correlación o cruce de información adecuado como la latitud y longitud es imposible realizar el estudio correcto.

Una vez correlacionadas estas y ubicadas en el mapa nos auxiliamos con Kmeans para realizar un análisis a profundidad que nos permita identificar si existe una correlación entra la calidad del agua y su ubicación geográfica teniendo como conclusión que:

La calidad del agua y su geografía no están relacionadas y que la mala calidad de la misma puede deberse a otros factores.

Limpieza, análisis, visualización y agrupamiento. En esta base de datos encontraras:

Aguas subterraneas. Aguas superficiales. Elige una base de datos, ya sea la de aguas superficiales o la de aguas subterraneas.

Limpieza de base de datos. Explorar cada datos (auxiliate de describe(), mean(), plot, boxplot de pandas): Identificando tendencias centrales promedio, media y mediana de los datos. Identificar medidas de dispersión, máximo, mínimo . Identificar medidas de posición no centrales , los cuartiles , outliers.
Identificar correlaciones. Preparar los datos Realizar análisis para encontrar si existe una relación entre la calidad del agua y su ubicación geográfica a través de K- means. Mostrar resultados de agrupamiento de latitudes y longitudes con K means en el mapa de México.

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

```
import os
import math #Esta libreria la usamos lara el ramsey y el Mape
import matplotlib.pyplot as plt
import requests, zipfile #Librerira para zip de nuestros origen de datos
from io import BytesIO

from imblearn.metrics import geometric_mean_score, classification_report_imbalanced
from google.colab import drive

from sklearn.model_selection import learning_curve, validation_curve
from sklearn.preprocessing import QuantileTransformer #Esta libreria la usamos al graf
from sklearn.preprocessing import power_transform #esta igual
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split #Para hacer las particiones
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report, make_scorer
from sklearn.model_selection import  cross_validate,  RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler
from sklearn.preprocessing import FunctionTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier #Esta tambien la usamoie en el eje
from sklearn import svm, datasets
from sklearn.model_selection import GridSearchCV
from sklearn import tree
from sklearn.dummy import DummyRegressor
from sklearn.linear_model import LinearRegression
from sklearn.compose import TransformedTargetRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RepeatedKFold



#Instalamos libreria para trabajar con mapas
! pip install qeds fiona geopandas xgboost gensim folium pyLDAvis descartes
```

```
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packag
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-pac
```

```
Collecting pyproj>=2.2.0
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 46.3 MB/s
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-pacl
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
Collecting funcy
  Downloading funcy-1.17-py2.py3-none-any.whl (33 kB)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/l
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-pacl
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (fi
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/lo
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pacl
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-pa
Collecting inflection>=0.3.1
  Downloading inflection-0.5.1-py2.py3-none-any.whl (9.5 kB)
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/pytho
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dis
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.7/dist-pacl
Building wheels for collected packages: qeds, pyLDAvis, sklearn
  Building wheel for qeds (setup.py) ... done
  Created wheel for qeds: filename=qeds-0.7.0-py3-none-any.whl size=27812 sha256=
  Stored in directory: /root/.cache/pip/wheels/fc/8c/52/0cc036b9730b75850b984577
  Building wheel for pyLDAvis (PEP 517) ... done
  Created wheel for pyLDAvis: filename=pyLDAvis-3.3.1-py2.py3-none-any.whl size=1
  Stored in directory: /root/.cache/pip/wheels/c9/21/f6/17bcf2667e8a68532ba2fbf6c
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=23
  Stored in directory: /root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b3104
Successfully built qeds pyLDAvis sklearn
Installing collected packages: munch, inflection, cligj, click-plugins, sklearn,
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.22 funcy-1.17 ge
```

```
import geopandas as gpd
from shapely.geometry import Point
```

```
#Datos_de_calidad_del_agua_2020/Datos_de_calidad_del_agua_de_sitios_de_monitoreo_de_ac
url = 'http://201.116.60.46/Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip'
req = requests.get(url)
zipfile.ZipFile(BytesIO(req.content)).extractall('unzipped_zip/')
df_sub=pd.read_csv('unzipped_zip/Datos_de_calidad_del_agua_2020/Datos_de_calidad_del_a
df_sub.head()
```

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO
0	DLAGU6	POZO SAN GIL	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	ASIENTOS
1	DLAGU6516	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	AGUASCALIENTES
2	DLAGU7	POZO COSIO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	COSIO
3	DLAGU9	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON DE ROMOS
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA PAZ

5 rows × 57 columns

```
#Limpieza de datos
df_aguas = df_sub.copy()
df_aguas.describe()
```

	LONGITUD	LATITUD	PERIODO	ALC_mg/L	CONDUCT_mS/cm	SDT_mg/L
count	1068.000000	1068.000000	1068.0	1064.000000	1062.000000	0.0
mean	-101.891007	23.163618	2020.0	235.633759	1138.953013	NaN
std	6.703263	3.887670	0.0	116.874291	1245.563674	NaN
min	-116.664250	14.561150	2020.0	26.640000	50.400000	NaN
25%	-105.388865	20.212055	2020.0	164.000000	501.750000	NaN
50%	-102.174180	22.617190	2020.0	215.527500	815.000000	NaN
75%	-98.974716	25.510285	2020.0	292.710000	1322.750000	NaN
max	-86.864120	32.677713	2020.0	1650.000000	18577.000000	NaN

```
df_aguas.info()
#'O' solo significa “objeto”
```

```
1   SITIO                  1068 non-null   object
2   ORGANISMO_DE_CUENCA    1068 non-null   object
3   ESTADO                 1068 non-null   object
4   MUNICIPIO              1068 non-null   object
5   ACUIFERO               1068 non-null   object
6   SUBTIPO                1068 non-null   object
7   LONGITUD               1068 non-null   float64
8   LATITUD                1068 non-null   float64
9   PERIODO                1068 non-null   int64
10  ALC_mg/L               1064 non-null   float64
11  CALIDAD_ALC            1064 non-null   object
12  CONDUCT_mS/cm          1062 non-null   float64
13  CALIDAD_CONDUC         1062 non-null   object
14  SDT_mg/L               0 non-null      float64
15  SDT_M_mg/L             1066 non-null   object
16  CALIDAD_SDT_ra         1066 non-null   object
17  CALIDAD_SDT_salin      1066 non-null   object
18  FLUORUROS_mg/L         1068 non-null   object
19  CALIDAD_FLUO           1068 non-null   object
20  DUR_mg/L               1067 non-null   object
21  CALIDAD_DUR            1067 non-null   object
22  COLI_FEC_NMP/100_mL    1068 non-null   object
23  CALIDAD_COLI_FEC       1068 non-null   object
24  N_NO3_mg/L             1067 non-null   object
25  CALIDAD_N_NO3          1067 non-null   object
26  AS_TOT_mg/L            1068 non-null   object
27  CALIDAD_AS             1068 non-null   object
28  CD_TOT_mg/L            1068 non-null   object
29  CALIDAD_CD             1068 non-null   object
30  CR_TOT_mg/L            1068 non-null   object
31  CALIDAD_CR             1068 non-null   object
32  HG_TOT_mg/L            1068 non-null   object
33  CALIDAD_HG             1068 non-null   object
34  PB_TOT_mg/L            1068 non-null   object
35  CALIDAD_PB             1068 non-null   object
36  MN_TOT_mg/L            1068 non-null   object
37  CALIDAD_MN             1068 non-null   object
38  FE_TOT_mg/L            1068 non-null   object
39  CALIDAD_FE             1068 non-null   object
40  SEMAFORO               1068 non-null   object
41  CONTAMINANTES          634 non-null    object
42  CUMPLE_CON_ALC         1068 non-null   object
43  CUMPLE_CON_COND        1068 non-null   object
44  CUMPLE_CON_SDT_ra      1068 non-null   object
45  CUMPLE_CON_SDT_salin   1068 non-null   object
46  CUMPLE_CON_FLUO        1068 non-null   object
47  CUMPLE_CON_DUR         1068 non-null   object
48  CUMPLE_CON_CF          1068 non-null   object
49  CUMPLE_CON_NO3         1068 non-null   object
50  CUMPLE_CON_AS          1068 non-null   object
51  CUMPLE_CON_CD          1068 non-null   object
52  CUMPLE_CON_CR          1068 non-null   object
53  CUMPLE_CON_HG          1068 non-null   object
54  CUMPLE_CON_PB          1068 non-null   object
55  CUMPLE_CON_MN          1068 non-null   object
56  CUMPLE_CON_FE          1068 non-null   object
```

```
dtypes: float64(5), int64(1), object(51)
memory usage: 475.7+ KB
```

```
df_aguas.columns
```

```
Index(['CLAVE', 'SITIO', 'ORGANISMO_DE_CUENCA', 'ESTADO', 'MUNICIPIO',
       'ACUIFERO', 'SUBTIPO', 'LONGITUD', 'LATITUD', 'PERIODO', 'ALC_mg/L',
       'CALIDAD_ALC', 'CONDUCT_mS/cm', 'CALIDAD_CONDUC', 'SDT_mg/L',
       'SDT_M_mg/L', 'CALIDAD_SDT_ra', 'CALIDAD_SDT_salin', 'FLUORUROS_mg/L',
       'CALIDAD_FLUO', 'DUR_mg/L', 'CALIDAD_DUR', 'COLI_FEC_NMP/100_mL',
       'CALIDAD_COLI_FEC', 'N_NO3_mg/L', 'CALIDAD_N_NO3', 'AS_TOT_mg/L',
       'CALIDAD_AS', 'CD_TOT_mg/L', 'CALIDAD_CD', 'CR_TOT_mg/L', 'CALIDAD_CR',
       'HG_TOT_mg/L', 'CALIDAD_HG', 'PB_TOT_mg/L', 'CALIDAD_PB', 'MN_TOT_mg/L',
       'CALIDAD_MN', 'FE_TOT_mg/L', 'CALIDAD_FE', 'SEMAFORO', 'CONTAMINANTES',
       'CUMPLE_CON_ALC', 'CUMPLE_CON_COND', 'CUMPLE_CON_SDT_ra',
       'CUMPLE_CON_SDT_salin', 'CUMPLE_CON_FLUO', 'CUMPLE_CON_DUR',
       'CUMPLE_CON_CF', 'CUMPLE_CON_NO3', 'CUMPLE_CON_AS', 'CUMPLE_CON_CD',
       'CUMPLE_CON_CR', 'CUMPLE_CON_HG', 'CUMPLE_CON_PB', 'CUMPLE_CON_MN',
       'CUMPLE_CON_FE'],
      dtype='object')
```

```
df_aguas.isnull().sum()
```

```
CLAVE                     0
SITIO                     0
ORGANISMO_DE_CUENCA       0
ESTADO                    0
MUNICIPIO                 0
ACUIFERO                  0
SUBTIPO                   0
LONGITUD                  0
LATITUD                   0
PERIODO                   0
ALC_mg/L                  4
CALIDAD_ALC               4
CONDUCT_mS/cm             6
CALIDAD_CONDUC            6
SDT_mg/L               1068
SDT_M_mg/L                2
CALIDAD_SDT_ra            2
CALIDAD_SDT_salin         2
FLUORUROS_mg/L            0
CALIDAD_FLUO              0
DUR_mg/L                  1
CALIDAD_DUR               1
COLI_FEC_NMP/100_mL       0
CALIDAD_COLI_FEC          0
N_NO3_mg/L                1
CALIDAD_N_NO3             1
AS_TOT_mg/L               0
CALIDAD_AS                0
CD_TOT_mg/L               0
CALIDAD_CD                0
CR_TOT_mg/L               0
CALIDAD_CR                0
```

```
HG_TOT_mg/L                0
CALIDAD_HG                 0
PB_TOT_mg/L                0
CALIDAD_PB                 0
MN_TOT_mg/L                0
CALIDAD_MN                 0
FE_TOT_mg/L                0
CALIDAD_FE                 0
SEMAFORO                   0
CONTAMINANTES            434
CUMPLE_CON_ALC             0
CUMPLE_CON_COND            0
CUMPLE_CON_SDT_ra          0
CUMPLE_CON_SDT_salin       0
CUMPLE_CON_FLUO            0
CUMPLE_CON_DUR             0
CUMPLE_CON_CF              0
CUMPLE_CON_NO3             0
CUMPLE_CON_AS              0
CUMPLE_CON_CD              0
CUMPLE_CON_CR              0
CUMPLE_CON_HG              0
CUMPLE_CON_PB              0
CUMPLE_CON_MN              0
CUMPLE_CON_FE              0
dtype: int64
```

```
df_aguas.isna().sum().sort_values(ascending=False) #Se ordenan los campos nulos de for
```

```
SDT_mg/L                1068
CONTAMINANTES            434
CALIDAD_CONDUC             6
CONDUCT_mS/cm              6
ALC_mg/L                   4
CALIDAD_ALC                4
CALIDAD_SDT_ra             2
SDT_M_mg/L                 2
CALIDAD_SDT_salin          2
CALIDAD_N_NO3              1
CALIDAD_DUR                1
N_NO3_mg/L                 1
DUR_mg/L                   1
CUMPLE_CON_COND            0
CUMPLE_CON_ALC             0
SEMAFORO                   0
CALIDAD_FE                 0
FE_TOT_mg/L                0
CALIDAD_MN                 0
CUMPLE_CON_SDT_ra          0
CUMPLE_CON_SDT_salin       0
CLAVE                      0
CUMPLE_CON_FLUO            0
CUMPLE_CON_DUR             0
CALIDAD_PB                 0
```

```
CUMPLE_CON_CF              0
CUMPLE_CON_NO3             0
CUMPLE_CON_AS              0
CUMPLE_CON_CD              0
CUMPLE_CON_CR              0
CUMPLE_CON_HG              0
CUMPLE_CON_PB              0
CUMPLE_CON_MN              0
MN_TOT_mg/L                0
CD_TOT_mg/L                0
PB_TOT_mg/L                0
CALIDAD_HG                 0
ORGANISMO_DE_CUENCA        0
ESTADO                     0
MUNICIPIO                  0
ACUIFERO                   0
SUBTIPO                    0
LONGITUD                   0
LATITUD                    0
PERIODO                    0
FLUORUROS_mg/L             0
CALIDAD_FLUO               0
COLI_FEC_NMP/100_mL        0
CALIDAD_COLI_FEC           0
AS_TOT_mg/L                0
CALIDAD_AS                 0
SITIO                      0
CALIDAD_CD                 0
CR_TOT_mg/L                0
CALIDAD_CR                 0
HG_TOT_mg/L                0
CUMPLE_CON_FE              0
dtype: int64
```

```
# Limpiamos los datos y dividiremos la informacion usando variablescategoricas y numer

columnas_numericas = ['ALC_mg/L','CONDUCT_mS/cm','SDT_mg/L','SDT_M_mg/L','FLUORUROS_mg
                      'N_NO3_mg/L','AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_m

df_limpio = df_aguas[['ALC_mg/L','CONDUCT_mS/cm','SDT_mg/L','SDT_M_mg/L','FLUORUROS_mg
                      'N_NO3_mg/L','AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_m

#Ahora que gano la imputacion pues vamos a analizar como se va  a hacer ahorita en un
df_limpio
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_mg/L	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	CO
0	229.990	940.0	NaN	603.6	0.9766	213.732	
1	231.990	608.0	NaN	445.4	0.9298	185.0514	
2	204.920	532.0	NaN	342	1.8045	120.719	
3	327.000	686.0	NaN	478.6	1.1229	199.879	
4	309.885	1841.0	NaN	1179	0.2343	476.9872	
...	...	...	...	...	...	...	
1063	231.045	2350.0	NaN	1545.8	<0.2	752.096	
1064	256.000	529.0	NaN	297	<0.2	273	

```
#Se verifican los valores de las columnas
print(df_limpio['SDT_mg/L'].unique())
print(df_limpio['SDT_mg/L'].value_counts())
```

```
[nan]
Series([], Name: SDT_mg/L, dtype: int64)
```

```
columnas_numericas = ['ALC_mg/L','CONDUCT_mS/cm','SDT_mg/L','SDT_M_mg/L','FLUORUROS_mg
                      'N_NO3_mg/L','AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_m
for i in columnas_numericas:
  print(" nombre de la columna -------------------------" + i)
  print(" sumatoria por valores uncos--------------------")
  print(df_limpio[i].value_counts())

'''for i in df.columns:
    print("nombre de la columna", df[i].column)
    print("Valores unicos", df[i].unique())'''
```

```
 nombre de la columna ------------------------ALC_mg/L
 sumatoria por valores uncos--------------------
157.620    5
193.815    4
195.360    4
204.765    4
257.850    4
          ..
341.000    1
151.000    1
106.000    1
99.000     1
256.000    1
Name: ALC_mg/L, Length: 816, dtype: int64
 nombre de la columna ------------------------CONDUCT_mS/cm
 sumatoria por valores uncos--------------------
777.0    6
300.0    4
412.0    4
454.0    4
308.0    4
        ..
826.0    1
876.0    1
373.0    1
733.0    1
817.0    1
Name: CONDUCT_mS/cm, Length: 801, dtype: int64
 nombre de la columna ------------------------SDT_mg/L
 sumatoria por valores uncos--------------------
Series([], Name: SDT_mg/L, dtype: int64)
 nombre de la columna ------------------------SDT_M_mg/L
 sumatoria por valores uncos--------------------
496         4
320         4
292         4
317         4
380         4
           ..
148         1
224         1
392         1
1736        1
690.6667    1
Name: SDT_M_mg/L, Length: 925, dtype: int64
 nombre de la columna ------------------------FLUORUROS_mg/L
 sumatoria por valores uncos--------------------
<0.2      162
0.466       3
0.5202      3
0.4993      2
0.482       2
          ...
1.6185      1
0.6045      1
0.7042      1
```

```
0.4343      1
```

```
y= pd.DataFrame(df_aguas['SEMAFORO'])
y
print(type(y))
#Dataframe visto en lista y la generacion de un semaforo para realizar graficas
y['SEMAFORO'].hist(bins = 60, figsize=(5,5))
```

```
<class 'pandas.core.frame.DataFrame'>
<matplotlib.axes._subplots.AxesSubplot at 0x7fbd38da9510>
```

```
columnas_numericas = ['ALC_mg/L','CONDUCT_mS/cm','SDT_mg/L','SDT_M_mg/L','FLUORUROS_mg
                      'N_NO3_mg/L','AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_m


for name in columnas_numericas:
  df_limpio[name] = df_limpio[name].astype('str')
  df_limpio[name] = df_limpio[name].str.replace('<25','25')
  df_limpio[name] = df_limpio[name].str.replace('<0.2','0.2')
  df_limpio[name] = df_limpio[name].str.replace('<20','20')
  df_limpio[name] = df_limpio[name].str.replace('<1.1','1.1')
  df_limpio[name] = df_limpio[name].str.replace('<0.02','0.02')
  df_limpio[name] = df_limpio[name].str.replace('<0.01','0.01')
  df_limpio[name] = df_limpio[name].str.replace('<0.003','0.003')
  df_limpio[name] = df_limpio[name].str.replace('<0.005','0.004')
  df_limpio[name] = df_limpio[name].str.replace('<0.0005','0.0004')
  df_limpio[name] = df_limpio[name].str.replace('<0.0015','0.0015')
  df_limpio[name] = df_limpio[name].str.replace('<0.025','0.025')
  df_limpio[name]= df_limpio[name].astype('float')

df_limpio.info()
```

```
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: FutureWarning: T
  if sys.path[0] == '':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  if sys.path[0] == '':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: FutureWarning: T
  del sys.path[0]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  del sys.path[0]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: FutureWarning: T

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: FutureWarning: T
  from ipykernel import kernelapp as app
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  from ipykernel import kernelapp as app
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16: FutureWarning: T
  app.launch_new_instance()
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  app.launch_new_instance()
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: FutureWarning: T
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18: FutureWarning: T
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:19: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
```

```
# la columna  SDT_mg/L             0 non-null      esta vacia, asi que se elimina

df_limpio.drop('SDT_mg/L', axis=1, inplace=True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  errors=errors,
```

```
print(df_limpio.columns)
print(df_limpio.info())
print(df_limpio.isnull().sum())
```

```
Index(['ALC_mg/L', 'CONDUCT_mS/cm', 'SDT_M_mg/L', 'FLUORUROS_mg/L', 'DUR_mg/L',
       'COLI_FEC_NMP/100_mL', 'N_NO3_mg/L', 'AS_TOT_mg/L', 'CD_TOT_mg/L',
       'CR_TOT_mg/L', 'HG_TOT_mg/L', 'PB_TOT_mg/L', 'MN_TOT_mg/L',
       'FE_TOT_mg/L'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1068 entries, 0 to 1067
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
```

```
 ---  ------                --------------  -----
  0   ALC_mg/L              1064 non-null   float64
  1   CONDUCT_mS/cm         1062 non-null   float64
  2   SDT_M_mg/L            1066 non-null   float64
  3   FLUORUROS_mg/L        1068 non-null   float64
  4   DUR_mg/L              1067 non-null   float64
  5   COLI_FEC_NMP/100_mL   1068 non-null   float64
  6   N_NO3_mg/L            1067 non-null   float64
  7   AS_TOT_mg/L           1068 non-null   float64
  8   CD_TOT_mg/L           1068 non-null   float64
  9   CR_TOT_mg/L           1068 non-null   float64
  10  HG_TOT_mg/L           1068 non-null   float64
  11  PB_TOT_mg/L           1068 non-null   float64
  12  MN_TOT_mg/L           1068 non-null   float64
  13  FE_TOT_mg/L           1068 non-null   float64
dtypes: float64(14)
memory usage: 116.9 KB
None
ALC_mg/L               4
CONDUCT_mS/cm          6
SDT_M_mg/L             2
FLUORUROS_mg/L         0
DUR_mg/L               1
COLI_FEC_NMP/100_mL    0
N_NO3_mg/L             1
AS_TOT_mg/L            0
CD_TOT_mg/L            0
CR_TOT_mg/L            0
HG_TOT_mg/L            0
PB_TOT_mg/L            0
MN_TOT_mg/L            0
FE_TOT_mg/L            0
dtype: int64
```

```
#Usamos moda y mediana

df_limpio

columnas_numericas_new= ['ALC_mg/L','CONDUCT_mS/cm','SDT_M_mg/L','FLUORUROS_mg/L','DUR
                         'N_NO3_mg/L','AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_m
'''
for name in columnas_numericas_new:
  mean = df_limpio[name].mean()
  df_limpio[name]= df_limpio[name].replace(np.nan, mean)'''

for name in columnas_numericas_new:
  moda_telas = df_limpio[name].mode()
  df_limpio[name]= df_limpio[name].replace(np.nan, moda_telas)
'''

for name in columnas_numericas_new:
  mediana = df_limpio[name].median()
  df_limpio[name]= df_limpio[name].replace(np.nan, mediana)
```

```
df_limpio.info()
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopy
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
```

```
print(df_limpio.describe())
df_limpio.describe().T
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	\
count	1068.000000	1068.000000	1068.000000	1068.000000	1068.000000	
mean	235.558455	1137.133052	895.454185	1.075600	347.842003	
std	116.661485	1242.292889	2748.991295	1.924278	359.514579	
min	26.640000	50.400000	25.000000	0.200000	20.000000	
25%	164.048750	505.500000	337.700000	0.267175	121.274100	
50%	215.527500	815.000000	550.400000	0.503500	245.335800	
75%	292.423750	1321.250000	915.900000	1.139850	453.930000	
max	1650.000000	18577.000000	82170.000000	34.803300	3810.692200	

Identificar medidas de posición no centrales , los cuartiles , outliers. Identificar correlaciones.

Preparar los datos

min	1.100000	0.020000	0.010000	0.003000

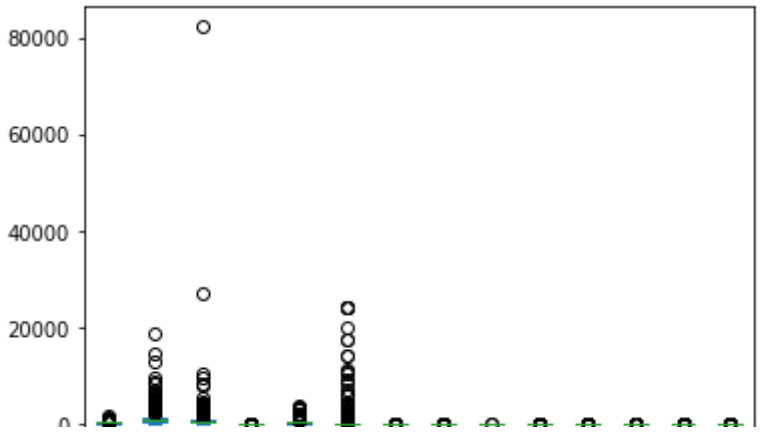
```
#Matriz de Correlación
df_limpio.corr()
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg
ALC_mg/L	1.000000	0.217212	0.079572	0.068860	0.2434
CONDUCT_mS/cm	0.217212	1.000000	0.286244	-0.025071	0.6926
SDT_M_mg/L	0.079572	0.286244	1.000000	-0.013709	0.3472
FLUORUROS_mg/L	0.068860	-0.025071	-0.013709	1.000000	-0.1495
DUR_mg/L	0.243404	0.692656	0.347211	-0.149549	1.0000
COLI_FEC_NMP/100_mL	-0.016338	0.018021	-0.001102	0.003564	0.0317
N_NO3_mg/L	-0.000346	0.219881	0.101522	-0.019672	0.3015
AS_TOT_mg/L	0.073458	-0.005047	-0.010092	0.444079	-0.1064
CD_TOT_mg/L	0.032706	0.029083	0.010807	-0.015123	0.0250
CR_TOT_mg/L	-0.014234	0.004436	-0.000494	-0.005205	0.0073
HG_TOT_mg/L	0.069779	0.057007	0.020332	-0.028597	0.0649
PB_TOT_mg/L	0.016989	0.024816	0.002517	-0.034191	-0.0173
MN_TOT_mg/L	0.129942	0.095940	0.018963	-0.049742	0.0838
FE_TOT_mg/L	0.043454	0.083172	0.020103	-0.009994	0.0597

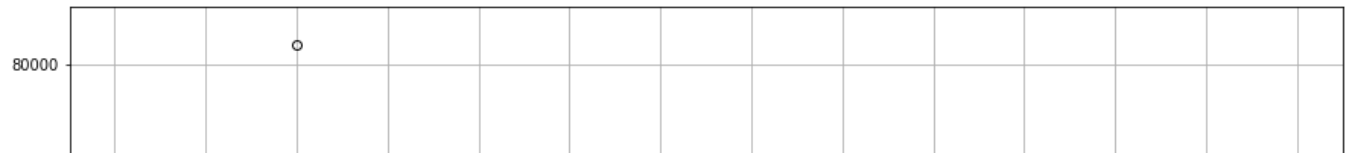
CR_TOT_mg/L	1068.0	0.012476	0.154435	0.0040	0.004000	0.004000

```
#Vemos los outliers en la gráfica, pero necesitamos acercarnos

df_limpio.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbd3866f850>
```




```
outliers = df_limpio.boxplot(figsize = (15,10),showmeans = True)
outliers.plot()
plt.xticks(rotation=90)
plt.show()
```




```
#Ahora vemos la grafica de correlación a color

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5, 5))

mi_correlacion = df_limpio.corr()

sns.heatmap(
    mi_correlacion,
    annot     = True,
    cbar      = False,
    annot_kws = {"size": 8},
    vmin      = -1,
    vmax      = 1,
    center    = 0,
    cmap      = sns.diverging_palette(20, 220, n=200),
    square    = True,
    ax        = ax
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right',
)

ax.tick_params(labelsize = 10)
```


```
correlacion = df_limpio.corr().abs()

f, ax = plt.subplots(figsize = (20,15))

sns.heatmap(correlacion, vmax = 1, vmin = -1, square = True, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbd35829110>
```

Realizar análisis para encontrar si existe una relación entre la calidad del agua y su ubicación geográfica a través de K- means. Mostrar resultados de agrupamiento de latitudes y longitudes con K means en el mapa de México.

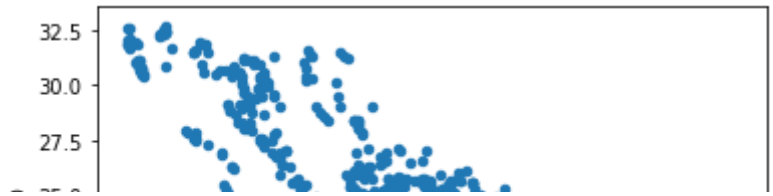

```
#Se verifican las variables relacionadas a la calidad del agua y su ubicacion geografi

df_ubicacion = df_aguas[['LONGITUD','LATITUD']]
df_ubicacion

#Se verifican las variables de calidad del agua basadas en nuestro semaforo
y
```

	SEMAFORO
0	Verde
1	Verde
2	Rojo
3	Verde
4	Rojo
...	...
1063	Rojo
1064	Rojo
1065	Rojo
1066	Verde
1067	Verde

1068 rows × 1 columns

```
df_ubicacion.plot.scatter('LONGITUD','LATITUD')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbd33d8c450>
```




```
#Se crea la columna de coordenadas
df_ubicacion
df_ubicacion["COORDENADAS"] = list(zip(df_ubicacion.LONGITUD, df_ubicacion.LATITUD))
df_ubicacion["COORDENADAS"] = df_ubicacion["COORDENADAS"].apply(Point)
df_ubicacion.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  after removing the cwd from sys.path.
```

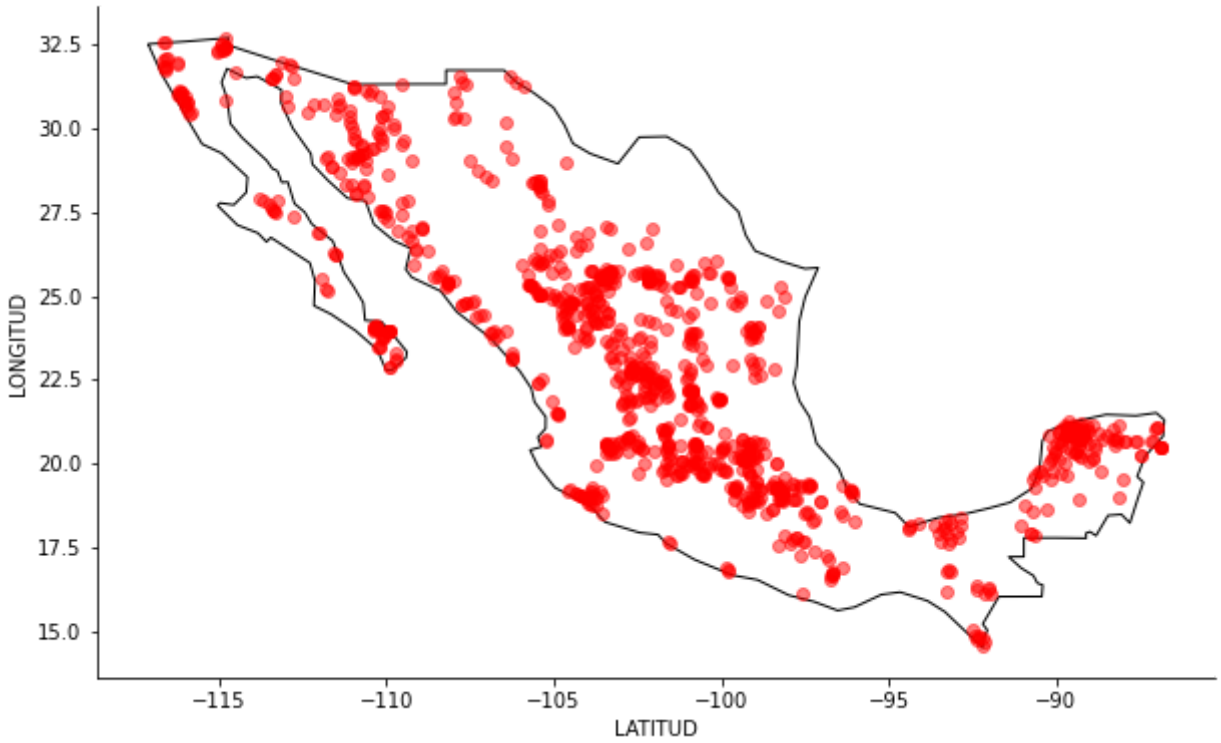
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.0221 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.2448 23.45138)

```
puntos_en_mapa = gpd.GeoDataFrame(df_ubicacion, geometry="COORDENADAS")

world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))

world = world.set_index("iso_a3")
world.name.unique()
fig, gax = plt.subplots(figsize=(10,10))

world.query("name == 'Mexico'").plot(ax=gax, edgecolor='black',color='white')

gax.set_xlabel('LATITUD')
gax.set_ylabel('LONGITUD')
```

```
gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)

puntos_en_mapa.plot(ax=gax, color='red', alpha = 0.5)
puntos_en_mapa
```

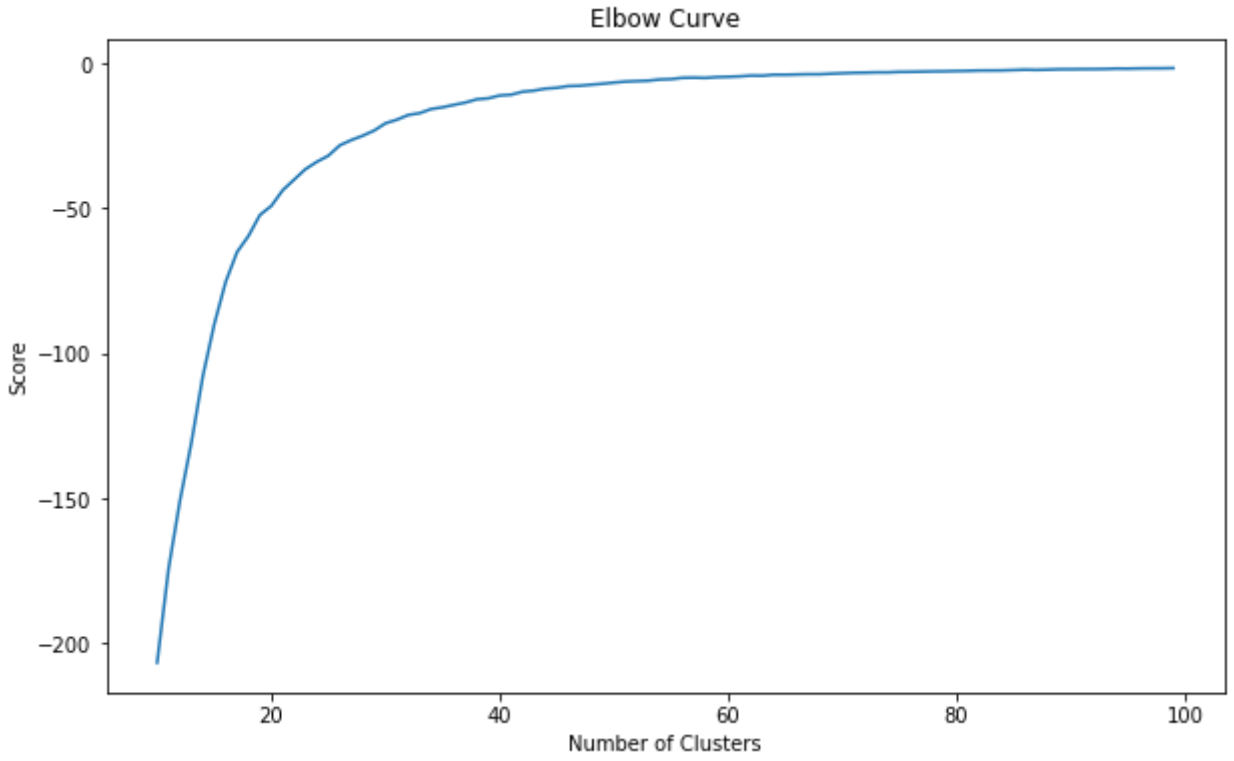
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.02210 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.24480 23.45138)
...	...	...	...
1063	-99.54191	24.76036	POINT (-99.54191 24.76036)
1064	-99.70099	24.78280	POINT (-99.70099 24.78280)
1065	-99.82249	25.55197	POINT (-99.82249 25.55197)
1066	-100.32683	24.80118	POINT (-100.32683 24.80118)
1067	-100.73302	25.09380	POINT (-100.73302 25.09380)

1068 rows × 3 columns




```
# Se agrupa la información por color o por ubicación
```

```
from sklearn.cluster import KMeans

#Se realiza el analisis usando 3 clusters

numero_de_closters = range(10,100)
mi_kmeans = [KMeans(n_clusters=i) for i in numero_de_closters]
Y_axis = df_ubicacion[['LATITUD']]
X_axis = df_ubicacion[['LONGITUD']]
calulo_kmeans = [mi_kmeans[i].fit(Y_axis).score(Y_axis) for i in range(len(mi_kmeans))


plt.figure(figsize=(10,6))
plt.plot(numero_de_closters, calulo_kmeans)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```




```
X = df_sub[['LONGITUD', 'LATITUD']]


kmeans = KMeans(n_clusters=20).fit(X)
centroids = kmeans.cluster_centers_ #Sacamos los centroides
labels = kmeans.predict(X)
C = kmeans.cluster_centers_
```

```
C_DF = pd.DataFrame(C) #Se convierte en DataFrame
C_DF["Coordinates"] = list(zip(C_DF[0], C_DF[1]))
C_DF["Coordinates"] = C_DF["Coordinates"].apply(Point) #Puntos para gráficar

puntos_centroides = gpd.GeoDataFrame(C_DF, geometry="Coordinates")
puntos_centroides
```

	0	1	Coordinates
0	-100.079438	24.779429	POINT (-100.07944 24.77943)
1	-112.903746	31.309819	POINT (-112.90375 31.30982)
2	-89.627607	20.535159	POINT (-89.62761 20.53516)
3	-103.764117	19.947591	POINT (-103.76412 19.94759)
4	-104.772299	24.824705	POINT (-104.77230 24.82470)
5	-97.649457	18.478355	POINT (-97.64946 18.47835)
6	-92.761627	17.004097	POINT (-92.76163 17.00410)
7	-110.195427	23.861448	POINT (-110.19543 23.86145)
8	-110.957923	27.322830	POINT (-110.95792 27.32283)
9	-102.470490	22.665601	POINT (-102.47049 22.66560)
10	-99.255412	19.807499	POINT (-99.25541 19.80750)
11	-103.108104	25.542966	POINT (-103.10810 25.54297)
12	-115.780432	31.622931	POINT (-115.78043 31.62293)
13	-107.281006	30.941352	POINT (-107.28101 30.94135)
14	-107.534814	24.697606	POINT (-107.53481 24.69761)
15	-87.659191	20.587765	POINT (-87.65919 20.58776)
16	-105.773238	28.333006	POINT (-105.77324 28.33301)
17	-101.285214	20.229017	POINT (-101.28521 20.22902)
18	-110.668940	29.844702	POINT (-110.66894 29.84470)
19	-100.408034	22.323348	POINT (-100.40803 22.32335)

```
len(labels)
```

```
1068
```

```
#Obtenermos el número de elementos por cluster
df_sub['SEMAFORO'].value_counts()
```

```
Verde       434
Rojo        387
Amarillo    247
Name: SEMAFORO, dtype: int64
```

```
print(y.head())
print(df_ubicacion.head())
```

```
  SEMAFORO
0    Verde
1    Verde
2     Rojo
3    Verde
4     Rojo
    LONGITUD   LATITUD                   COORDENADAS
0 -102.02210  22.20887  POINT (-102.02210 22.20887)
1 -102.20075  21.99958  POINT (-102.20075 21.99958)
2 -102.28801  22.36685  POINT (-102.28801 22.36685)
3 -102.29449  22.18435  POINT (-102.29449 22.18435)
4 -110.24480  23.45138  POINT (-110.24480 23.45138)
```

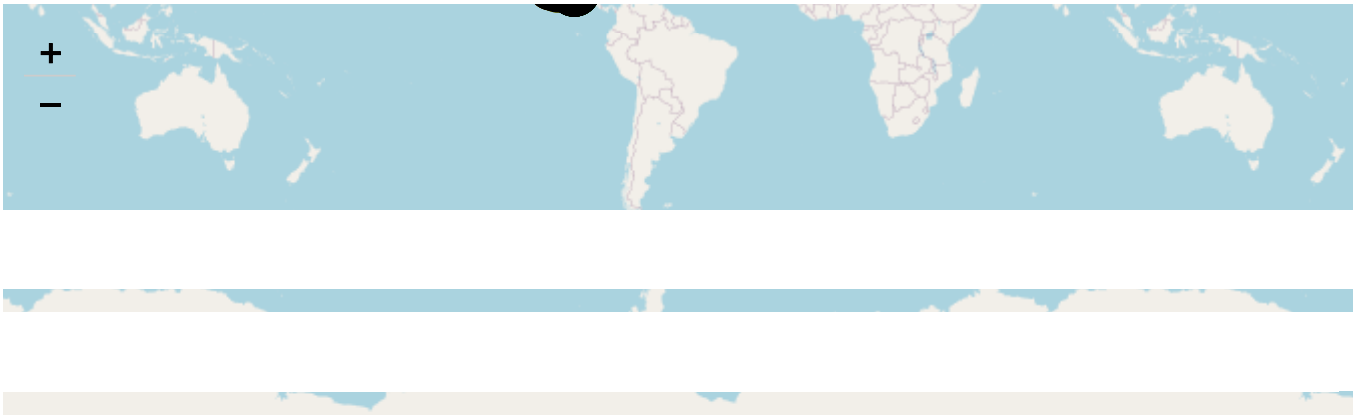
```
y['SEMAPHORE'] = y['SEMAFORO'].replace(to_replace = "Verde", value = "green")
y['SEMAPHORE'].replace(to_replace = "Rojo", value = "red", inplace=True)
y['SEMAPHORE'].replace(to_replace = "Amarillo", value = "yellow", inplace=True)
y
```

	SEMAFORO	SEMAPHORE
0	Verde	green
1	Verde	green
2	Rojo	red
3	Verde	green
4	Rojo	red
...	...	...
1063	Rojo	red
1064	Rojo	red
1065	Rojo	red
1066	Verde	green
1067	Verde	green

1068 rows × 2 columns

```
puntos_en_mapa['LATITUDYLONGITUD'] = puntos_en_mapa['LATITUD'] + puntos_en_mapa['LONGI
diccionario_semaforo = dict(zip(puntos_en_mapa.LATITUDYLONGITUD, y.SEMAPHORE))
```

```
diccionario_semaforo

import folium
lat = puntos_en_mapa.iloc[0]['LATITUD']
lng = puntos_en_mapa.iloc[0]['LONGITUD']
map = folium.Map(location=[lng, lat], zoom_start=1)
for _, row in puntos_en_mapa.iterrows():
    folium.CircleMarker(
        location=[row["LATITUD"], row["LONGITUD"]],
        radius=12,
        weight=2,
        fill=True,
        fill_color=diccionario_semaforo[row["LATITUDYLONGITUD"]],
        color=diccionario_semaforo[row["LATITUDYLONGITUD"]]
    ).add_to(map)
color='black'
for _, row in puntos_en_mapa.iterrows():
    folium.CircleMarker(
        location=[row[1], row[0]],
        radius=12,
        weight=2,
        fill=True,
        fill_color=color,
        color=color
    ).add_to(map)
map
```



```
'''
for i in y:
  y.codigo_color[i] = y[i].str.replace('Verde','1')
  y[i] = y[i].str.replace('Amarillo','2')
  y[i] = y[i].str.replace('Rojo','3') '''
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-37-df7d50dcf274> in <module>
      2 #y= df_aguas['SEMAFORO']
      3 #DataFrame_Final = pd.concat([df_ubicacion, y], ignore_index=True)
----> 4 DataFrame_Final['COLOR'] = y['SEMAFORO']
      5 DataFrame_Final.head()
      6

NameError: name 'DataFrame_Final' is not defined
```

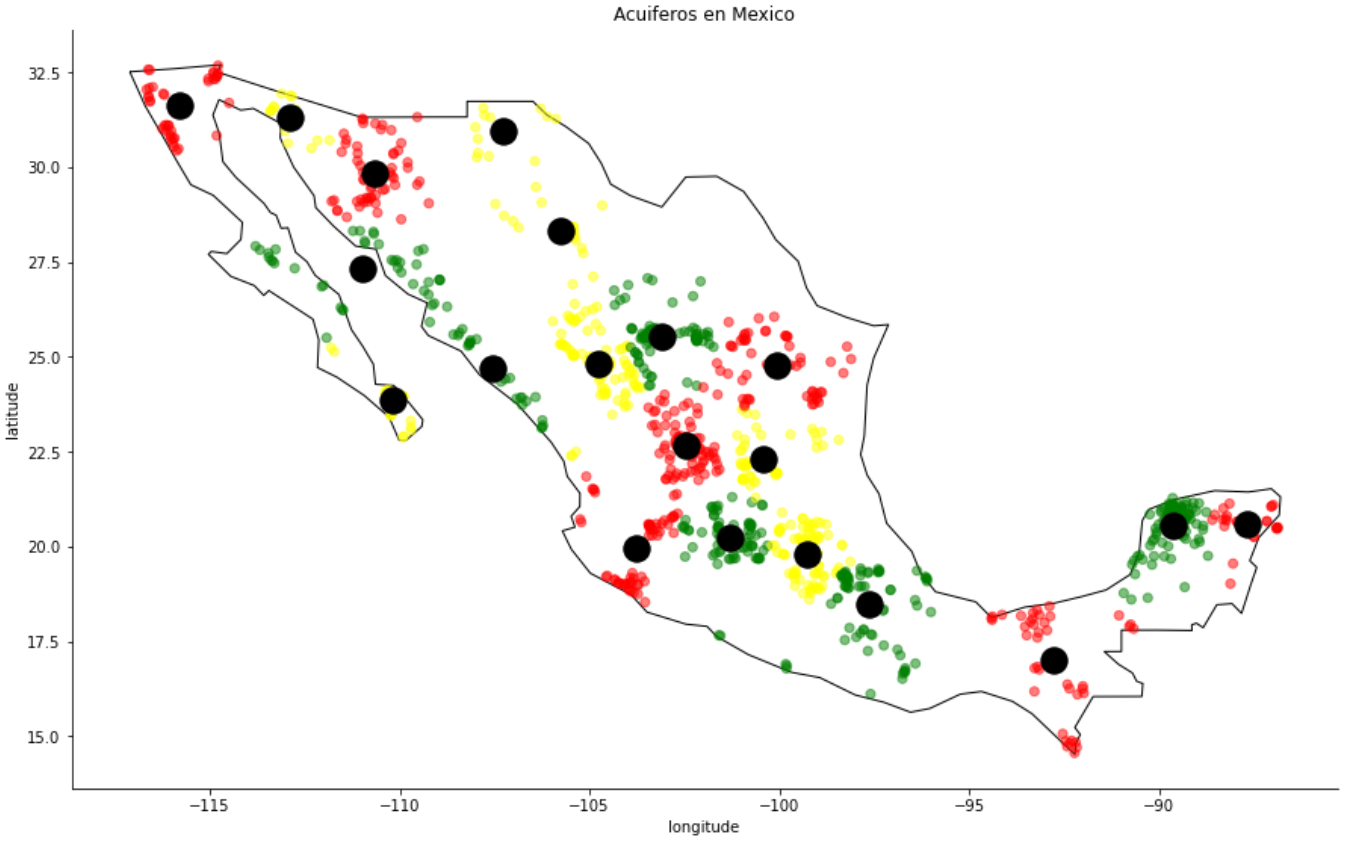
SEARCH STACK OVERFLOW

```
df_aguas['CALIDAD_COLI_FEC'].value_counts()
```

```
Potable - Excelente        739
Buena calidad              208
Aceptable                   60
Contaminada                 49
Fuertemente contaminada     12
Name: CALIDAD_COLI_FEC, dtype: int64
```

```
#Ploteamos los centroids

fig, gax = plt.subplots(figsize=(15,10))
colores = ['red','yellow','green','red','yellow','green','red','yellow','green','red',
color_asig = []

for row in labels:
  color_asig.append(colores[row])
```

```
world.query("name == 'Mexico'").plot(ax = gax, edgecolor='black', color='white') #Sele

puntos_en_mapa.plot(ax=gax, color=color_asig, alpha = 0.5)
puntos_centroides.plot(ax=gax, color='black', alpha = 1, markersize = 300)

gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.set_title('Acuiferos en Mexico')

gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)

plt.show()
```




```
from sklearn.preprocessing import LabelEncoder

lbe = LabelEncoder()
```

```
df_limpio_ASubterraneas["SEMAFORO_Type"]= lbe.fit_transform(df_limpio_ASubterraneas["S
df_limpio_ASubterraneas["SEMAFORO_Type"].unique()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-40-348a3d6ffd95> in <module>
     11
     12 lbe = LabelEncoder()
---> 13 df_limpio_ASubterraneas["SEMAFORO_Type"]=
lbe.fit_transform(df_limpio_ASubterraneas["SEMAFORO"])
     14 df_limpio_ASubterraneas["SEMAFORO_Type"].unique()

NameError: name 'df_limpio_ASubterraneas' is not defined
```

SEARCH STACK OVERFLOW

```
#Se asigna un número de cluster a cada punto en el mapa

puntos_en_mapa['COLOR']= y['SEMAFORO']
puntos_en_mapa['CLUSTER'] = labels
puntos_en_mapa
```

	LONGITUD	LATITUD	COORDENADAS	LATITUDYLONGITUD	COLOR	CLUSTER
0	-102.02210	22.20887	POINT (-102.02210 22.20887)	-79.81323	Verde	9
1	-102.20075	21.99958	POINT (-102.20075 21.99958)	-80.20117	Verde	9
2	-102.28801	22.36685	POINT (-102.28801 22.36685)	-79.92116	Rojo	9
3	-102.29449	22.18435	POINT (-102.29449 22.18435)	-80.11014	Verde	9
4	-110.24480	23.45138	POINT (-110.24480 23.45138)	-86.79342	Rojo	7
...	...	...	...	...	...	...
1063	-99.54191	24.76036	POINT (-99.54191 24.76036)	-74.78155	Rojo	0
1064	-99.70099	24.78280	POINT (-99.70099 24.78280)	-74.91819	Rojo	0

```
nuevo_dataset = puntos_en_mapa[puntos_en_mapa.CLUSTER == 0].copy()
nuevo_dataset.shape
```

```
(55, 6)
```

```
#Se obtiene la media a cada Cluster
```

```
lista_de_modas=[]

for i in range(0,20): # Ciclo de 20  veces
  nuevo_dataset = pd.DataFrame() #Creamos un dataframe hueco
  nuevo_dataset = puntos_en_mapa[puntos_en_mapa.CLUSTER == i].copy()
  moda = nuevo_dataset['COLOR'].mode()[0] #Se va creando la moda de cada color
  lista_de_modas.append(moda)

len(lista_de_modas)
```

```
20
```

```
#Sumamos la columna de la moda por Cluster al data ser de Cluster
puntos_centroides['MODA'] = lista_de_modas
puntos_centroides
```

	0	1	Coordinates	MODA

```
lista_gringa = []

for i in range(0,20):
  if lista_de_modas[i] == 'Verde':
    lista_gringa.append('green')
  if lista_de_modas[i] == 'Rojo':
    lista_gringa.append('red')
  if lista_de_modas[i] == 'Amarillo':
    lista_gringa.append('yellow')

len(lista_gringa)
```

```
20
```

```
 9  -102.470490  22.665601  POINT (-102.47049 22.66560)    Rojo
```

```
lista_gringa_individual = []

for i in range(0,1068):
  if puntos_en_mapa.COLOR[i] == 'Verde':
    lista_gringa_individual.append('green')
  if puntos_en_mapa.COLOR[i] == 'Rojo':
    lista_gringa_individual.append('red')
  if puntos_en_mapa.COLOR[i] == 'Amarillo':
    lista_gringa_individual.append('yellow')

len(lista_gringa_individual)
```

```
1068
```

```
18  -110.668940  29.844702  POINT (-110.66894 29.84470)    Verde
```

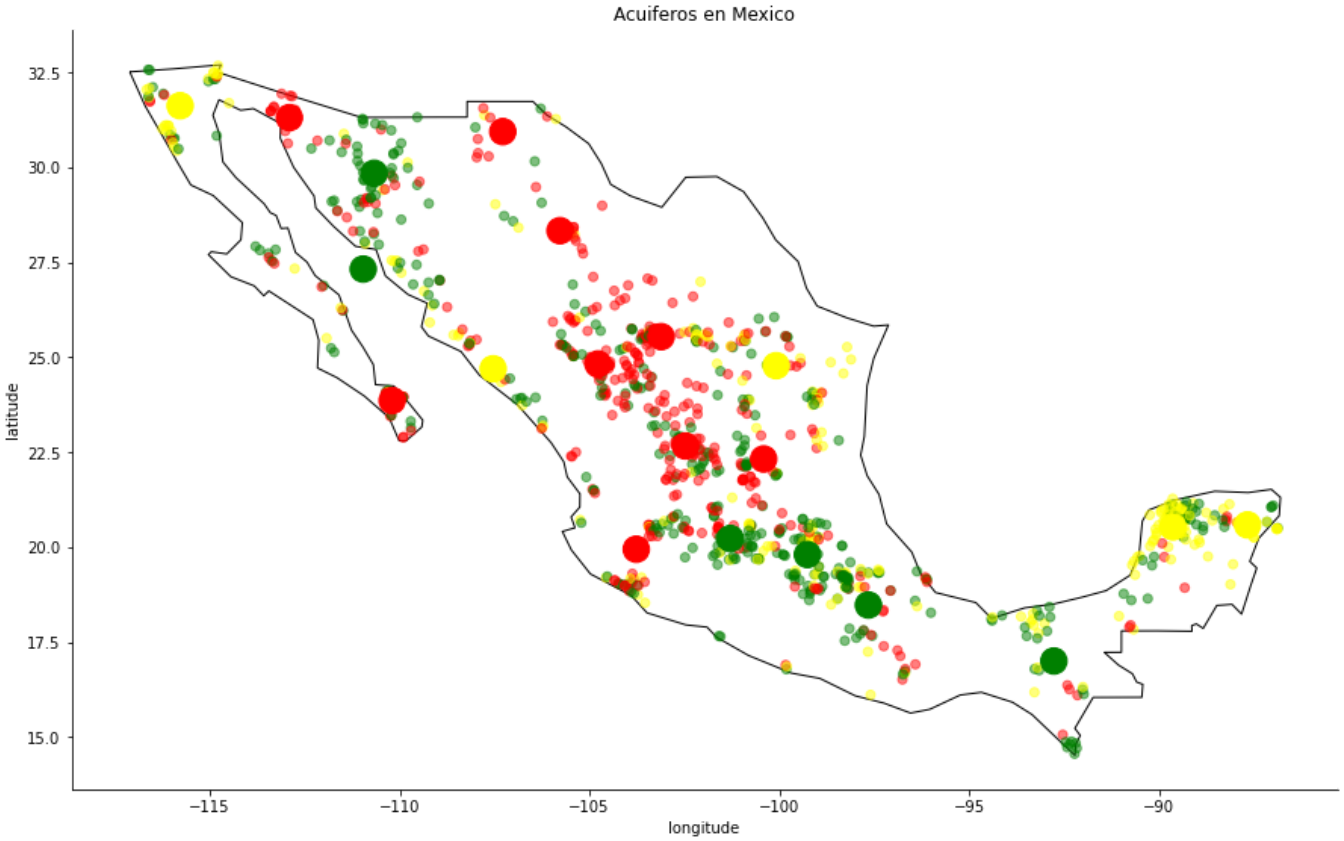
```
print(puntos_centroides)
print(puntos_en_mapa)
```

```
             0          1                  Coordinates      MODA
0  -100.079438  24.779429  POINT (-100.07944 24.77943)  Amarillo
1  -112.903746  31.309819  POINT (-112.90375 31.30982)      Rojo
2   -89.627607  20.535159   POINT (-89.62761 20.53516)  Amarillo
3  -103.764117  19.947591  POINT (-103.76412 19.94759)      Rojo
4  -104.772299  24.824705  POINT (-104.77230 24.82470)      Rojo
5   -97.649457  18.478355   POINT (-97.64946 18.47835)     Verde
6   -92.761627  17.004097   POINT (-92.76163 17.00410)     Verde
7  -110.195427  23.861448  POINT (-110.19543 23.86145)      Rojo
8  -110.957923  27.322830  POINT (-110.95792 27.32283)     Verde
9  -102.470490  22.665601  POINT (-102.47049 22.66560)      Rojo
10  -99.255412  19.807499   POINT (-99.25541 19.80750)     Verde
11 -103.108104  25.542966  POINT (-103.10810 25.54297)      Rojo
12 -115.780432  31.622931  POINT (-115.78043 31.62293)  Amarillo
13 -107.281006  30.941352  POINT (-107.28101 30.94135)      Rojo
14 -107.534814  24.697606  POINT (-107.53481 24.69761)  Amarillo
```

```
15  -87.659191  20.587765   POINT (-87.65919 20.58776)  Amarillo
16 -105.773238  28.333006  POINT (-105.77324 28.33301)      Rojo
17 -101.285214  20.229017  POINT (-101.28521 20.22902)     Verde
18 -110.668940  29.844702  POINT (-110.66894 29.84470)     Verde
19 -100.408034  22.323348  POINT (-100.40803 22.32335)      Rojo
       LONGITUD   LATITUD                  COORDENADAS  LATITUDYLONGITUD  \
0    -102.02210  22.20887  POINT (-102.02210 22.20887)         -79.81323
1    -102.20075  21.99958  POINT (-102.20075 21.99958)         -80.20117
2    -102.28801  22.36685  POINT (-102.28801 22.36685)         -79.92116
3    -102.29449  22.18435  POINT (-102.29449 22.18435)         -80.11014
4    -110.24480  23.45138  POINT (-110.24480 23.45138)         -86.79342
...         ...       ...                          ...               ...
1063  -99.54191  24.76036   POINT (-99.54191 24.76036)         -74.78155
1064  -99.70099  24.78280   POINT (-99.70099 24.78280)         -74.91819
1065  -99.82249  25.55197   POINT (-99.82249 25.55197)         -74.27052
1066 -100.32683  24.80118  POINT (-100.32683 24.80118)         -75.52565
1067 -100.73302  25.09380  POINT (-100.73302 25.09380)         -75.63922

      COLOR  CLUSTER
0     Verde        9
1     Verde        9
2      Rojo        9
3     Verde        9
4      Rojo        7
...     ...      ...
1063   Rojo        0
1064   Rojo        0
1065   Rojo        0
1066  Verde        0
1067  Verde        0

[1068 rows x 6 columns]
```

```
Se plotean los centroides

ig, gax = plt.subplots(figsize=(15,10))
olor_asig = []
olor_individual = puntos_en_mapa['COLOR']

or row in range(0,len(lista_gringa)):
 color_asig.append(lista_gringa[row])

orld.query("name == 'Mexico'").plot(ax = gax, edgecolor='black', color='white') #filtra

untos_en_mapa.plot(ax=gax, color=lista_gringa_individual, alpha = 0.5)
untos_centroides.plot(ax=gax, color=color_asig, alpha = 1, markersize = 300)

ax.set_xlabel('longitude')
ax.set_ylabel('latitude')
ax.set_title('Acuiferos en Mexico')

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

```
lt.show()
```




```
len(color_asig)
```

```
20
```

Double-click (or enter) to edit

```
# send back into dataframe and display it
df_sub_copy = df_sub.copy(deep=True)
df_sub_copy['CLUSTER'] = labels
# display the number of mamber each clustering
_clusters = df_sub_copy.groupby('CLUSTER')['CLUSTER'].count()
print(_clusters)
```