Maestría en Inteligencia Artificial Aplicada

- Curso: Ciencia y analítica de datos
- Tecnológico de Monterrey
- Prof Maria Paz Rico
- Reto_Entrega1

Nombres y matrículas de los integrantes del equipo:

- Andres Javier Galindo Vargas - A01793927
- Carlos Jesús Peñaloza Julio - A01793931


```
!pip install patool
```

```
! pip install qeds fiona geopandas xgboost gensim folium pyLDAvis descartes
```

```
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages
Collecting pyproj>=2.2.0
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 52.0 MB/s
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (fro
Collecting funcy
  Downloading funcy-1.17-py2.py3-none-any.whl (33 kB)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
Requirement already satisfied: cycler>=0.10 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/local
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (from p
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packag
```

```
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packag
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 (from
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.7
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.7/dist-packages
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=dc46
  Stored in directory: /root/.cache/pip/wheels/fc/8c/52/0cc036b9730b75850b9845770780f
  Building wheel for pyLDAvis (PEP 517) ... done
  Created wheel for pyLDAvis: filename=pyLDAvis-3.3.1-py2.py3-none-any.whl size=13689
  Stored in directory: /root/.cache/pip/wheels/c9/21/f6/17bcf2667e8a68532ba2fbf6d5c72
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=2344 sh
  Stored in directory: /root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b310477667
Successfully built qeds pyLDAvis sklearn
Installing collected packages: munch, inflection, cligj, click-plugins, sklearn, quan
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.22 funcy-1.17 geopan
```

```
#Librerías
import patoolib as pt
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import geopandas as gpd
from shapely.geometry import Point

from sklearn.cluster import KMeans

import folium # plotting library
from folium import plugins

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors
```

```
#Bajar los datos: Base de datos de calidad de agua
!wget 'http://201.116.60.46/Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip'
pt.extract_archive('Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip')
```

```
--2022-11-16 22:12:43--  http://201.116.60.46/Datos_de_calidad_del_agua_de_5000_sitios_d
Connecting to 201.116.60.46:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2556825 (2.4M) [application/x-zip-compressed]
Saving to: ‘Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip’

Datos_de_calidad_de 100%[===================>]   2.44M  2.06MB/s    in 1.2s

2022-11-16 22:12:45 (2.06 MB/s) - ‘Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo

patool: Extracting Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip ...
patool: running /usr/bin/7z x -o./Unpack_uyu_4ppu -- Datos_de_calidad_del_agua_de_5000_s
patool: ... Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip extracted to `Dato
'Datos_de_calidad_del_agua_2020'
```

```
#Leer archivo
file = 'Datos_de_calidad_del_agua_2020/Datos_de_calidad_del_agua_de_sitios_de_monitoreo_de_ag
df = pd.read_csv(file, encoding = 'latin1')
df.head()
```

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIP
0	DLAGU6	POZO SAN GIL	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	ASIENTO
1	DLAGU6516	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	AGUASCALIENTI
2	DLAGU7	POZO COSIO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	COS
3	DLAGU9	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON I ROMO
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA P

5 rows × 57 columns

```
#El conjunto de datos muestra la siguiente información:

#shape
print('Shape: ', df.shape)
#columns
print('Columns: ',df.columns)
#dtypes
print('Dtypes: ',df.dtypes)
```

```
Shape:  (1068, 57)
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')
Dtypes:  CLAVE                   object
SITIO                    object
ORGANISMO_DE_CUENCA      object
ESTADO                   object
MUNICIPIO                object
ACUIFERO                 object
SUBTIPO                  object
LONGITUD                float64
LATITUD                 float64
PERIODO                   int64
ALC_mg/L                float64
CALIDAD_ALC              object
CONDUCT_mS/cm           float64
CALIDAD_CONDUC           object
SDT_mg/L                float64
SDT_M_mg/L               object
CALIDAD_SDT_ra           object
CALIDAD_SDT_salin        object
FLUORUROS_mg/L           object
CALIDAD_FLUO             object
DUR_mg/L                 object
CALIDAD_DUR              object
COLI_FEC_NMP/100_mL      object
CALIDAD_COLI_FEC         object
N_NO3_mg/L               object
CALIDAD_N_NO3            object
AS_TOT_mg/L              object
CALIDAD_AS               object
CD_TOT_mg/L              object
CALIDAD_CD               object
CR_TOT_mg/L              object
CALIDAD_CR               object
HG_TOT_mg/L              object
CALIDAD_HG               object
PB_TOT_mg/L              object
CALIDAD_PB               object
MN_TOT_mg/L              object
CALIDAD_MN               object
FE_TOT_mg/L              object
CALIDAD_FE               object
SEMAFORO                 object
CONTAMINANTES            object
```

```
#Revisamos cuántos datos nulos tenemos en el conjunto
df.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
```

```
#Teniendo en cuenta los tipos de variables del conjunto de datos, procedemos a revisar la est

df.describe().transpose()
```

	count	mean	std	min	25%	50%	
LONGITUD	1068.0	-101.891007	6.703263	-116.66425	-105.388865	-102.17418	
LATITUD	1068.0	23.163618	3.887670	14.56115	20.212055	22.61719	
PERIODO	1068.0	2020.000000	0.000000	2020.00000	2020.000000	2020.00000	2
ALC_mg/L	1064.0	235.633759	116.874291	26.64000	164.000000	215.52750	
CONDUCT_mS/cm	1062.0	1138.953013	1245.563674	50.40000	501.750000	815.00000	1
SDT_mg/L	0.0	NaN	NaN	NaN	NaN	NaN	

La estadística descriptiva como sabemos, nos toma las variables numéricas, de las mostradas podemos ver que la Alcalinidad tiene una media importante en 235 mg/L, valor cercano al segundo cuartil.

```
#Dado que se identificaron datos nulos, se procede a ordenarlos para un mejor análisis

df.isna().sum().sort_values(ascending=False)
```

```
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
```

LIMPIEZA DE DATOS NULOS

```
#Total de registros vs registros nulos
print('El total de datos es de: ' + str(df.shape[0]) +
      '\nEl total de datos nulos es de: ' + str(df.isna().sum().sum()))
#Porcentaje de registros nulos
#Porcentaje de registros nulos
print('Los datos nulos representan el ' + str(round(df.isna().sum().sum()/df.shape[0] * 100,2
```

```
El total de datos es de: 1068
El total de datos nulos es de: 1532
Los datos nulos representan el 143.45% del total de los valores
```

```
pd.set_option('display.max_columns', None)
```

```
pd.set_option('display.max_columns', None)


datos_nul = df[df.isnull().any(axis=1)].shape[0]


print('El total de datos es de: ' + str(df.shape[0]) +
      '\nEl total de datos nulos es de: ' + str(datos_nul))
#Porcentaje de registros nulos
print('Los datos nulos representan el ' + str(round(datos_nul/df.shape[0] * 100,2)) + '% del
```

```
El total de datos es de: 1068
El total de datos nulos es de: 1068
Los datos nulos representan el 100.0% del total de los valores
```

```
df.isna().any()
```

```
CLAVE                  False
SITIO                  False
ORGANISMO_DE_CUENCA    False
ESTADO                 False
MUNICIPIO              False
ACUIFERO               False
SUBTIPO                False
LONGITUD               False
LATITUD                False
PERIODO                False
ALC_mg/L                True
CALIDAD_ALC             True
CONDUCT_mS/cm           True
CALIDAD_CONDUC          True
SDT_mg/L                True
SDT_M_mg/L              True
CALIDAD_SDT_ra          True
CALIDAD_SDT_salin       True
FLUORUROS_mg/L         False
CALIDAD_FLUO           False
DUR_mg/L                True
CALIDAD_DUR             True
COLI_FEC_NMP/100_mL    False
CALIDAD_COLI_FEC       False
N_NO3_mg/L              True
CALIDAD_N_NO3           True
AS_TOT_mg/L            False
CALIDAD_AS             False
CD_TOT_mg/L            False
CALIDAD_CD             False
CR_TOT_mg/L            False
CALIDAD_CR             False
HG_TOT_mg/L            False
CALIDAD_HG             False
PB_TOT_mg/L            False
CALIDAD_PB             False
MN_TOT_mg/L            False
CALIDAD_MN             False
```

```
FE_TOT_mg/L             False
CALIDAD_FE              False
SEMAFORO                False
CONTAMINANTES            True
CUMPLE_CON_ALC          False
CUMPLE_CON_COND         False
CUMPLE_CON_SDT_ra       False
CUMPLE_CON_SDT_salin    False
CUMPLE_CON_FLUO         False
CUMPLE_CON_DUR          False
CUMPLE_CON_CF           False
CUMPLE_CON_NO3          False
CUMPLE_CON_AS           False
CUMPLE_CON_CD           False
CUMPLE_CON_CR           False
CUMPLE_CON_HG           False
CUMPLE_CON_PB           False
CUMPLE_CON_MN           False
CUMPLE_CON_FE           False
dtype: bool
```

```
df_imp = df.copy()
df_sin_nulos = df.copy()
```

```
df_sin_nulos.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	A
3	DLAGU9	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON DE ROMOS	A
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA PAZ	

```
#Eliminando los registros que tengan datos nulos
df_sin_nulos.dropna(inplace=True)
df_sin_nulos.shape
```

```
(0, 57)
```

```
#Imputación de datos
df.hist(bins = 60, figsize=(15,15))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fad8d1e9850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fad8d097e50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fad8d05c490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fad8d012a90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fad8cfd50d0>,
```

VARIABLES CATEGÓRICAS Y NUMÉRICAS

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1068 entries, 0 to 1067
Data columns (total 57 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   CLAVE                 1068 non-null   object 
 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
```

```
df.isna().any()
```

```
CLAVE                  False
SITIO                  False
ORGANISMO_DE_CUENCA    False
ESTADO                 False
MUNICIPIO              False
ACUIFERO               False
SUBTIPO                False
LONGITUD               False
LATITUD                False
PERIODO                False
ALC_mg/L                True
CALIDAD_ALC             True
CONDUCT_mS/cm           True
CALIDAD_CONDUC          True
SDT_mg/L                True
SDT_M_mg/L              True
CALIDAD_SDT_ra          True
CALIDAD_SDT_salin       True
FLUORUROS_mg/L         False
CALIDAD_FLUO           False
DUR_mg/L                True
CALIDAD_DUR             True
COLI_FEC_NMP/100_mL    False
CALIDAD_COLI_FEC       False
N_NO3_mg/L              True
CALIDAD_N_NO3           True
AS_TOT_mg/L            False
CALIDAD_AS             False
CD_TOT_mg/L            False
CALIDAD_CD             False
CR_TOT_mg/L            False
CALIDAD_CR             False
HG_TOT_mg/L            False
CALIDAD_HG             False
PB_TOT_mg/L            False
CALIDAD_PB             False
MN_TOT_mg/L            False
CALIDAD_MN             False
```

```
FE_TOT_mg/L             False
CALIDAD_FE              False
SEMAFORO                False
CONTAMINANTES            True
CUMPLE_CON_ALC          False
CUMPLE_CON_COND         False
CUMPLE_CON_SDT_ra       False
CUMPLE_CON_SDT_salin    False
CUMPLE_CON_FLUO         False
CUMPLE_CON_DUR          False
CUMPLE_CON_CF           False
CUMPLE_CON_NO3          False
CUMPLE_CON_AS           False
CUMPLE_CON_CD           False
CUMPLE_CON_CR           False
CUMPLE_CON_HG           False
CUMPLE_CON_PB           False
CUMPLE_CON_MN           False
CUMPLE_CON_FE           False
dtype: bool
```

VARIABLES CATEGÓRICAS

[ ] ↳ *3 celdas ocultas*

VARIABLES NUMÉRICAS

[ ] ↳ *18 celdas ocultas*

K-MEANS

Realizar análisis para encontrar si existe una relación entre la calidad del agua y su ubicación geográfica a través de K- means.

UBICACIÓN GEOGRÁFICA

```
#Basados en la latitud y longitud se creará un dataframe y posteriormente se presentarán las
latlong=df[['LONGITUD','LATITUD']]
latlong["COORDENADAS"] = list(zip(latlong.LONGITUD, latlong.LATITUD))
latlong["COORDENADAS"] = latlong["COORDENADAS"].apply(Point)
latlong.head()
```

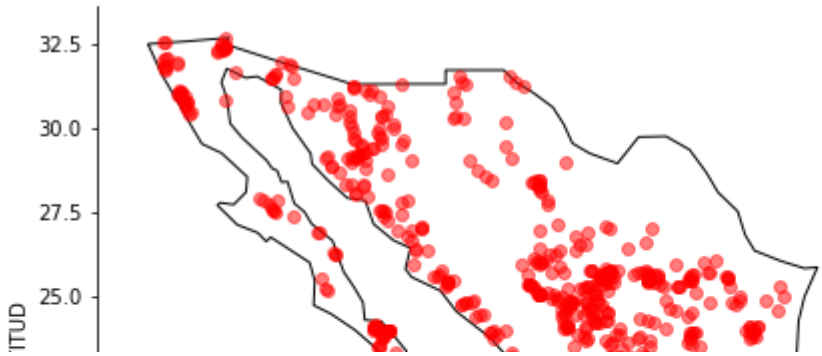
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

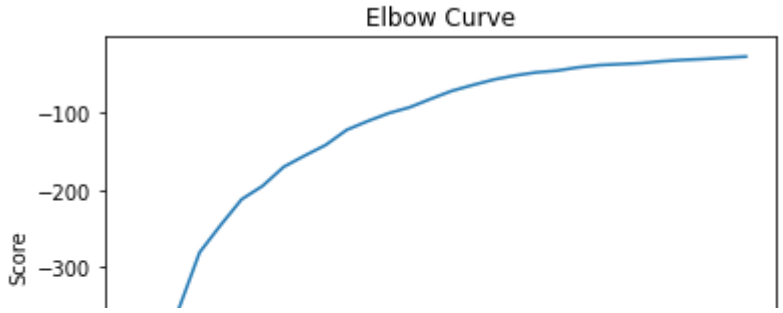
```
ubicacion_geografica = gpd.GeoDataFrame(latlong, geometry="COORDENADAS")

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))

# By only plotting rows in which the continent is 'South America' we only plot SA.
world.query("name == 'Mexico'").plot(ax=gax, edgecolor='black',color='white')

# By the way, if you haven't read the book 'longitude' by Dava Sobel, you should...
gax.set_xlabel('LONGITUD')
gax.set_ylabel('LATITUD')

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

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

	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




```
K_clusters = range(10,40)
kmeans = [KMeans(n_clusters=i) for i in K_clusters]

Y_axis = latlong[['LONGITUD']]
X_axis = latlong[['LATITUD']]
score = [kmeans[i].fit(Y_axis).score(Y_axis) for i in range(len(kmeans))]

plt.plot(K_clusters, score)
plt.xlabel('Número de Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```




```
#Coordenadas halladas:
X = latlong[["LONGITUD","LATITUD"]]
X
```

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

1068 rows × 2 columns

```
#Aplicamos K-means
kmeans = KMeans(n_clusters=25).fit(X)
centroides = kmeans.cluster_centers_
labels = kmeans.predict(X)
c_c = kmeans.cluster_centers_
mdf = pd.DataFrame(c_c)


mdf["Coordenadas"] = list(zip(mdf[0], mdf[1]))
mdf["Coordenadas"] = mdf["Coordenadas"].apply(Point)


geo_mdf = gpd.GeoDataFrame(mdf, geometry="Coordenadas")
geo_mdf
```

	0	1	Coordenadas
0	-102.433561	22.580590	POINT (-102.43356 22.58059)
1	-87.659191	20.587765	POINT (-87.65919 20.58776)
2	-110.698251	29.812006	POINT (-110.69825 29.81201)
3	-97.751935	19.077840	POINT (-97.75194 19.07784)
4	-105.312119	25.554020	POINT (-105.31212 25.55402)
5	-91.998388	15.997558	POINT (-91.99839 15.99756)
6	-115.780432	31.622931	POINT (-115.78043 31.62293)
7	-101.607999	25.386561	POINT (-101.60800 25.38656)
8	-101.141321	20.304250	POINT (-101.14132 20.30425)
9	-110.146688	23.819670	POINT (-110.14669 23.81967)
10	-89.627607	20.535159	POINT (-89.62761 20.53516)
11	-103.352602	20.683322	POINT (-103.35260 20.68332)
12	-106.395956	29.427649	POINT (-106.39596 29.42765)
13	-100.716044	22.454333	POINT (-100.71604 22.45433)
14	-112.903746	31.309819	POINT (-112.90375 31.30982)
15	-99.309391	19.832809	POINT (-99.30939 19.83281)
16	-109.731105	27.319170	POINT (-109.73110 27.31917)
17	-99.139244	24.194012	POINT (-99.13924 24.19401)
18	-112.671813	27.005056	POINT (-112.67181 27.00506)
19	-104.122289	24.235305	POINT (-104.12229 24.23530)
20	-107.438890	24.602789	POINT (-107.43889 24.60279)
21	-97.549928	17.196103	POINT (-97.54993 17.19610)
22	-103.811394	18.922499	POINT (-103.81139 18.92250)
23	-103.568548	25.673262	POINT (-103.56855 25.67326)
24	-93.420789	17.873379	POINT (-93.42079 17.87338)

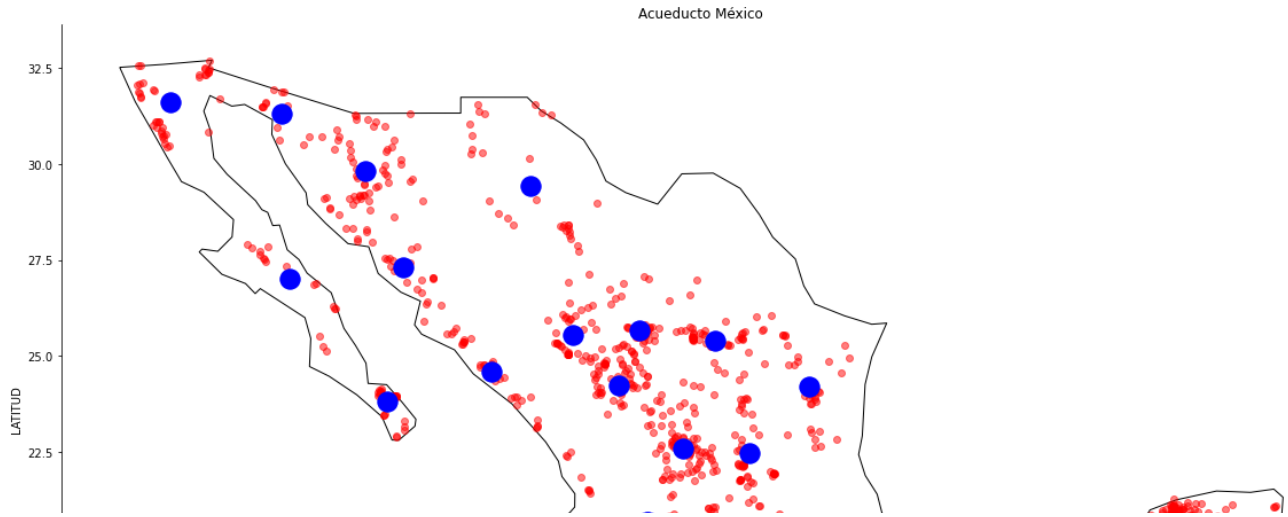
Realizar análisis para encontrar si existe una relación entre la calidad del agua y su ubicación geográfica a través de K- means.

```
#La variable categórica que nos permite conocer la calidad del agua es CALIDAD_COLI_FEC
#Procedemos entonces a contar la clasificación desde fuertemente contaminada hasta excelente
```

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

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

```
fig, gax = plt.subplots(figsize=(20,20))

world.query("name == 'Mexico'").plot(ax = gax, edgecolor='black', color='white')
ubicacion_geografica.plot(ax=gax, color='red', alpha = 0.5)
geo_mdf.plot(ax=gax, color='blue', alpha = 1, markersize = 300)

#Graficamos
gax.set_xlabel('LONGITUD')
gax.set_ylabel('LATITUD')
gax.set_title('Acueducto México')

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

plt.show()
```



Hasta la ubicación de los clusters de acuerdo con el resultado de k-means podríamos considerar que son los puntos más relevantes en cuanto a ubicación de los acuiferos y la respectiva calidad del agua.

Vemos importante en este punto, revisar la validación que se hace con la variable categórica "SEMAFORO" y una variable relacionada con la calidad del agua como lo es "CALIDAD_COLI_FEC".

```
#Revisaremos cómo está el conteo de la variable "SEMAFORO"
plt.figure(figsize=(8, 6))
sns.countplot(df['SEMAFORO'], palette='Set2')
plt.xlabel("'SEMAFORO'")
plt.title("Count Plot of 'SEMAFORO'")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
Text(0.5, 1.0, "Count Plot of 'SEMAFORO'")
```

```
df['SEMAFORO'].value_counts()
```

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

Aquí importa ver la relación de las coordenadas con el color y el cluster correspondiente por lo que adicionamos esa columna al dataframe.


```
latlong['COLOR']= df['SEMAFORO']
latlong['CLUSTER'] = labels
latlong
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
```

Para poder ubicar las coordenadas en el mapa de México y relacionarla con el valor en el semáforo, es necesario traducir al inglés los valores de la columna "SEMAFORO" incluyendo también el encabezado.

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
```

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


print(df['SEMAPHORE'].head())
print(latlong.head())
```

```
0    green
1    green
2      red
3    green
4      red
Name: SEMAPHORE, dtype: object
    LONGITUD   LATITUD                  COORDENADAS  COLOR  CLUSTER
0 -102.02210  22.20887  POINT (-102.02210 22.20887)  Verde        0
1 -102.20075  21.99958  POINT (-102.20075 21.99958)  Verde        0
2 -102.28801  22.36685  POINT (-102.28801 22.36685)   Rojo        0
3 -102.29449  22.18435  POINT (-102.29449 22.18435)  Verde        0
4 -110.24480  23.45138  POINT (-110.24480 23.45138)   Rojo        9
```

Para hacernos una idea de la relación de las dos variables mencionadas "SEMAFORO" y "CALIDAD_COLI_FEC", imprimimos un dataframe con estas y adicionamos la latitud y longitud.

```
df_types = df[['CALIDAD_COLI_FEC','SEMAPHORE','LONGITUD','LATITUD']]
df_types.head(20)
```

	CALIDAD_COLI_FEC	SEMAPHORE	LONGITUD	LATITUD
0	Potable - Excelente	green	-102.022100	22.208870
1	Potable - Excelente	green	-102.200750	21.999580
2	Potable - Excelente	red	-102.288010	22.366850
3	Potable - Excelente	green	-102.294490	22.184350
4	Aceptable	red	-110.244800	23.451380
5	Contaminada	red	-110.220670	23.464930
6	Buena calidad	green	-110.213960	23.474600
7	Aceptable	red	-109.907306	22.890500
8	Buena calidad	green	-110.088778	23.799861
9	Contaminada	red	-110.054722	23.824722

Ahora miramos cuál es la ocurrencia entre la clasificación de la calidad del agua y los semáforos

```
types = pd.get_dummies(df_types['SEMAPHORE'])
types['CALIDAD_COLI_FEC'] = df_types['CALIDAD_COLI_FEC']
types = types.groupby('CALIDAD_COLI_FEC').sum().reset_index()
types.head()
```

	CALIDAD_COLI_FEC	green	red	yellow
0	Aceptable	21.0	22.0	17.0
1	Buena calidad	77.0	72.0	59.0
2	Contaminada	0.0	49.0	0.0
3	Fuertemente contaminada	0.0	12.0	0.0
4	Potable - Excelente	336.0	232.0	171.0

Teniendo en cuenta el análisis de la relación de las dos variables, procederemos a confirmar la cantidad de clusters.

```
codes = types[['CALIDAD_COLI_FEC']]
types.drop('CALIDAD_COLI_FEC', axis=1, inplace=True)

# K-means clustering
kmeans = KMeans(n_clusters=5, random_state=0).fit(types)
```

```
codes['cluster'] = kmeans.labels_
codes.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: 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/user_
  after removing the cwd from sys.path.
```

	CALIDAD_COLI_FEC	cluster
0	Aceptable	3
1	Buena calidad	2
2	Contaminada	1
3	Fuertemente contaminada	4
4	Potable - Excelente	0

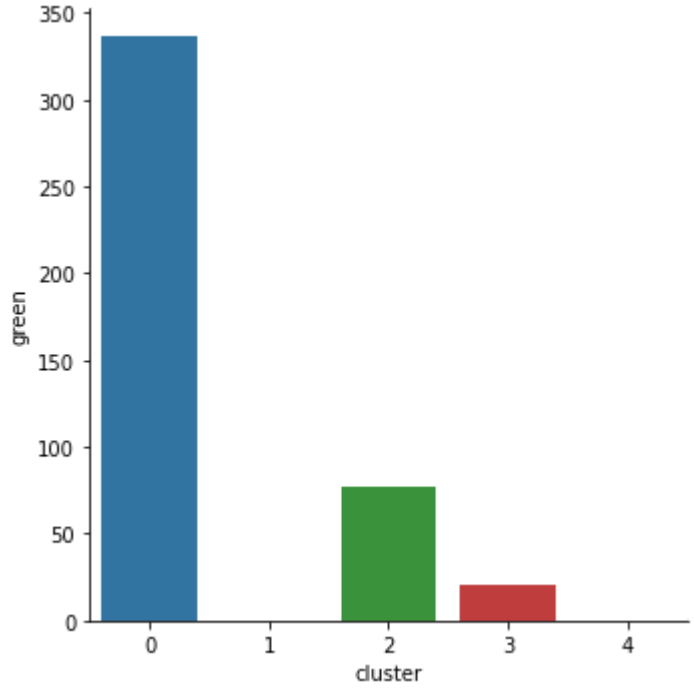
```
types['cluster']  = kmeans.labels_
types.head()
```

	green	red	yellow	cluster
0	21.0	22.0	17.0	3
1	77.0	72.0	59.0	2
2	0.0	49.0	0.0	1
3	0.0	12.0	0.0	4
4	336.0	232.0	171.0	0

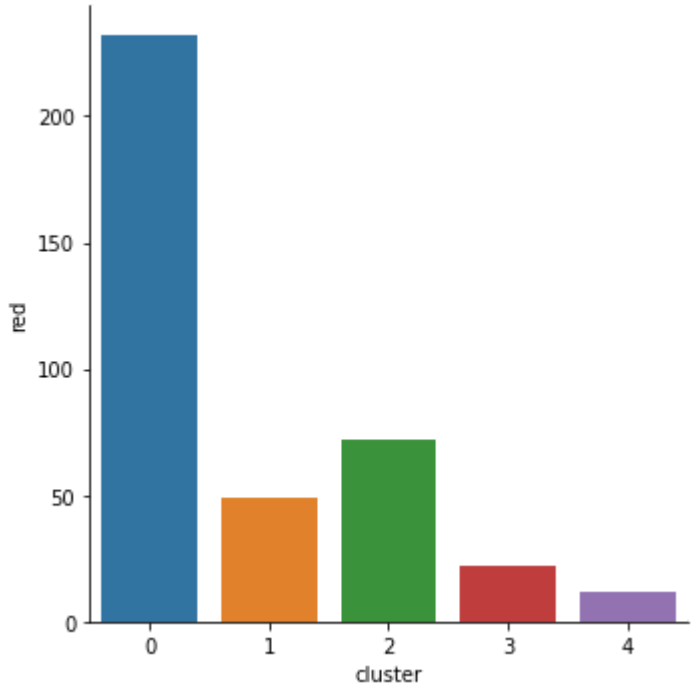
```
#Media de los primeros 5 clusters
types.groupby('cluster').mean()
```

	green	red	yellow
cluster			
0	336.0	232.0	171.0
1	0.0	49.0	0.0
2	77.0	72.0	59.0
3	21.0	22.0	17.0
4	0.0	12.0	0.0

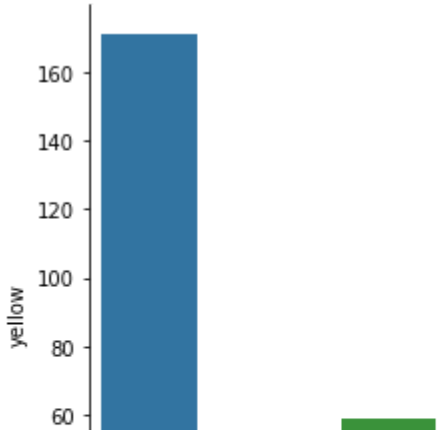
```
sns.catplot(x='cluster', y='green', data=types, kind='bar');
```




```
sns.catplot(x='cluster', y='red', data=types, kind='bar');
```




```
sns.catplot(x='cluster', y='yellow', data=types, kind='bar');
```



Se va a definir el dataframe acuiferos con las variables:

'SITIO', 'ORGANISMO_DE_CUENCA', 'ACUIFERO', 'LONGITUD', 'LATITUD', 'MUNICIPIO', 'ESTADO'

El objetivo es analizar el número de ocurrencias o el conteo por acuifero según su ubicación geográfica.

```
acuiferos = df[['SITIO', 'ORGANISMO_DE_CUENCA','ACUIFERO','LONGITUD','LATITUD','MUNICIPIO','E
acuiferos.head()
```

	SITIO	ORGANISMO_DE_CUENCA	ACUIFERO	LONGITUD	LATITUD	MUNI
0	POZO SAN GIL	LERMA SANTIAGO PACIFICO	VALLE DE CHICALOTE	-102.02210	22.20887	ASIE
1	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	VALLE DE CHICALOTE	-102.20075	21.99958	AGUASCALIE
2	POZO COSIO	LERMA SANTIAGO PACIFICO	VALLE DE AGUASCALIENTES	-102.28801	22.36685	C
3	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	VALLE DE AGUASCALIENTES	-102.29449	22.18435	RINCC RC

```
# Hallando el número de ocurrencias por Acuifero según los primeros 5 clusters
number_of_occurences = pd.DataFrame(acuiferos['ACUIFERO'].value_counts())
number_of_occurences.reset_index(inplace=True)
number_of_occurences.columns = ['ACUIFERO', 'Count']
number_of_occurences.head()
```

	ACUIFERO	Count
0	PENINSULA DE YUCATAN	119

```
number_of_occurences = number_of_occurences.merge(acuiferos.drop_duplicates())
```

2	ALTO ATOYAC	19

```
number_of_occurences.head()
```

	ACUIFERO	Count	SITIO	ORGANISMO_DE_CUENCA	LONGITUD	LATITUD	MUNICI
0	PENINSULA DE YUCATAN	119	POZO DEL SISTEMA DE AGUA POTABLE DE CANDELARIA...	PENINSULA DE YUCATAN	-91.04672	18.18680	CANDELA
1	PENINSULA DE YUCATAN	119	POZO DEL SISTEMA DE AGUA POTABLE DE CHULBAC	PENINSULA DE YUCATAN	-90.55914	19.74566	CAMPEC
	PENINSULA		POZO 1 ORIENTE DE				

Procedemos a graficar sobre el mapa de México, definiendo previamente un diccionario con las coordenadas y el semáforo.

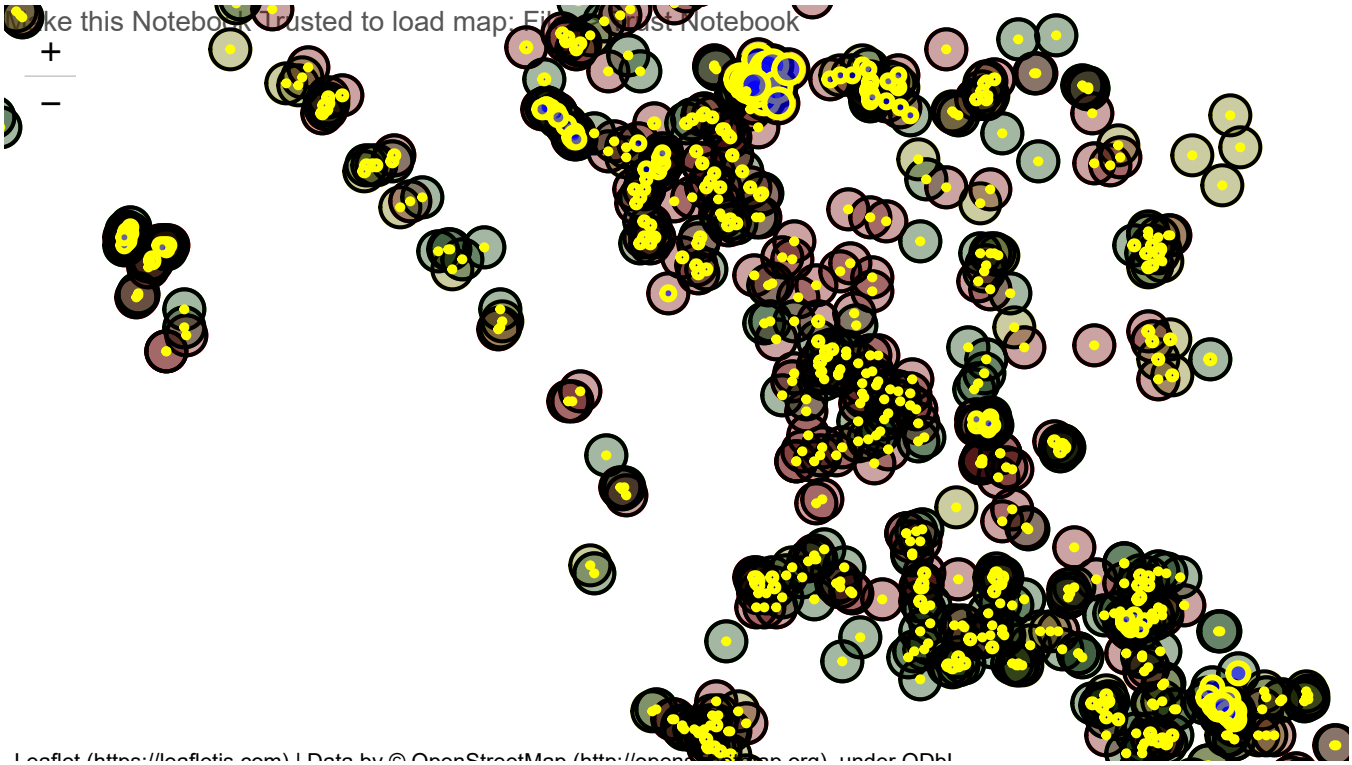
```
#ubicacion_geografica['LATITUDYLONGITUD'] = ubicacion_geografica['LATITUD'] + ubicacion_geogr
#dicc_semaforo = dict(zip(ubicacion_geografica.LATITUDYLONGITUD, df.SEMAPHORE))
#dicc_semaforo

latlong['LATITUDYLONGITUD'] = latlong['LATITUD'] + latlong['LONGITUD']
dicc_semaforo = dict(zip(latlong.LATITUDYLONGITUD, df.SEMAPHORE))
dicc_semaforo

import folium

#lat = latlong.iloc[0]['LATITUD']
#lng = latlong.iloc[0]['LONGITUD']
#map = folium.Map(location=[lng, lat], zoom_start=5)
#folium.Map( location= (19.419444, -99.145556), zoom_start=10 )
Mexico_map=folium.Map(location=[19.432, -99.13], zoom_start=6) # Mapa de México centrado

for _, row in latlong.iterrows():
    folium.CircleMarker(
        location=[row["LATITUD"], row["LONGITUD"]],
        radius=10,
        weight=2,
```

```
            fill=True,
            fill_color=dicc_semaforo[row["LATITUDYLONGITUD"]],
            color=dicc_semaforo[row["LATITUDYLONGITUD"]]
        ).add_to(Mexico_map)
    color='black'
    for _, row in latlong.iterrows():
        folium.CircleMarker(
            location=[row[1], row[0]],
            radius=10,
            weight=2,
            fill=True,
            fill_color=color,
            color=color
        ).add_to(Mexico_map)
    Mexico_map
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: 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/user_
  """
```

En el mapa anterior lo que hicimos fue ubicar las coordenadas de los acuiferos y relacionarlos con el comportamiento del semáforo, a continuación se hará la combinación de lo obtenido con la librería folium para ubicar las ocurrencias o participación identificada para cada acuifero, todo esto después de analizar la cantidad de clusters y la cobertura que cada uno podría tener.

```
occurences = folium.map.FeatureGroup()
n_mean = number_of_occurences['Count'].mean()
for lat, lng, number, city, state in zip(number_of_occurences['LATITUD'],
                                         number_of_occurences['LONGITUD'],
                                         number_of_occurences['Count'],
                                         number_of_occurences['MUNICIPIO'],
                                         number_of_occurences['ESTADO'],):
      occurences.add_child(
        folium.vector_layers.CircleMarker(
            [lat, lng],
            radius=number/n_mean*5, # define how big you want the circle markers to be
            color='yellow',
            fill=True,
            fill_color='blue',
            fill_opacity=0.6,
            tooltip = str(number)+','+str(city) +','+ str(state)
        )
    )

Mexico_map.add_child(occurences)
```



Este último gráfico si bien no contiene puntualmente la distribución de los clusters como lo vimos con anterioridad, se consideró la representación de los acuiferos, es así como en la Península de Yucatán obligatoriamente debemos contar con un cluster, el k-means marcó 2 de los 25 que tentativamente se pueden tener según la estimación. El otro cluster significativo, se puede apreciar cerca al estado Coahuila y Durango, seguido de Tlaxcala.