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


```
# Instalacion de la libreria
!pip install patool

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

```
Requirement already satisfied: folium in /usr/local/lib/python3.7/dist-packages (0.12
Requirement already satisfied: pyLDAvis in /usr/local/lib/python3.7/dist-packages (3.
Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: quandl in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: quantecon in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: setuptools 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: attrs>=17 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-pack
```

```
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: funcy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (fro
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: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: inflection>=0.3.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: sympy 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
```

Import de la librerias de Trabajo

```
#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 20:28:27--  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.1’

Datos_de_calidad_de 100%[===================>]   2.44M  2.12MB/s    in 1.2s

2022-11-16 20:28:29 (2.12 MB/s) - ‘Datos_de_calidad_del_agua_de_5000_sitios_de_monitored

patool: Extracting Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip ...
patool: running /usr/bin/7z x -o./Unpack_1uo_r9os -- Datos_de_calidad_del_agua_de_5000_s
patool: ... Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo.zip extracted to `Datc
'Datos_de_calidad_del_agua_de_5000_sitios_de_monitoreo'
```

Lectura dataset monitoreo de aguas subterraneas

```
#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	ASIENTC
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 ROMC
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA P/

5 rows × 57 columns

Analisis de los Datos

```
#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
```

```
#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	

```
#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)


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()
```

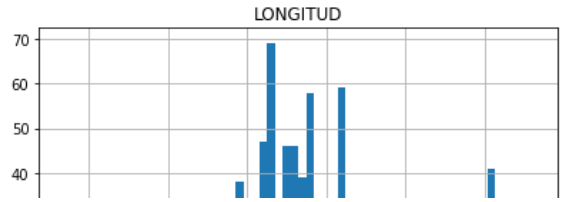
	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	
0	DLAGU6	POZO SAN GIL	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	ASIENTOS	
1	DLAGU6516	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	AGUASCALIENTES	
2	DLAGU7	POZO COSIO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	COSIO	A
3	DLAGU9	POZO EL SALITRILLO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

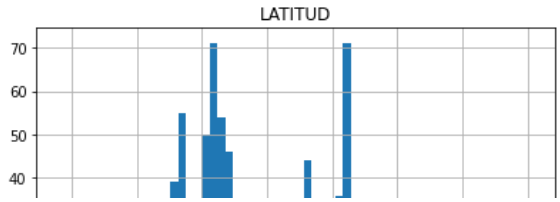
```
#Eliminando los registros que tengan datos nulos
df_sin_nulos.dropna(inplace=True)
df_sin_nulos.shape
```

```
(0, 57)
```

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

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7ff955fd6ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff955f4dc50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff955d8df10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff955d52550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff955d13fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff955cca190>]],
      dtype=object)
```





Analisis de Variables

[ ] ↳ *2 celdas ocultas*

Variables Categóricas

[ ] ↳ *3 celdas ocultas*

Variables numéricas

[ ] ↳ *6 celdas ocultas*

Escalamiento de las variables

```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline

#class sklearn.impute.SimpleImputer(*, missing_values=nan, strategy='mean', fill_value=None,

cat_pipeline = Pipeline(steps = [
    ("imputer", SimpleImputer(strategy='most_frequent'))
])

num_pipeline = Pipeline(steps = [
    ("imputer", SimpleImputer(strategy='median'))
])


from sklearn.compose import ColumnTransformer
```

```
transformer = ColumnTransformer(transformers=[
    ("cat", cat_pipeline, lista_cat),
    ("num", num_pipeline, lista_num)
], remainder='passthrough');


df_imp_num
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	COLI_FEC_NMP/
0	229.990	940.0	603.6000	0.9766	213.7320	
1	231.990	608.0	445.4000	0.9298	185.0514	
2	204.920	532.0	342.0000	1.8045	120.7190	
3	327.000	686.0	478.6000	1.1229	199.8790	
4	309.885	1841.0	1179.0000	0.2343	476.9872	
...	...	...	...	...	...	
1063	231.045	2350.0	1545.8000	0.2000	752.0960	
1064	256.000	529.0	297.0000	0.2000	273.0000	
1065	330.690	2600.0	1873.0000	0.7574	660.2126	
1066	193.140	873.0	690.6667	0.7108	406.3680	
1067	263.070	817.0	495.0000	0.4002	362.5440	

1068 rows × 14 columns

```
lista_num_new = list(['ALC_mg/L','CONDUCT_mS/cm','SDT_M_mg/L','FLUORUROS_mg/L','DUR_mg/L','CO
                      'N_NO3_mg/L','AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_mg/L','P


for name in lista_num_new:
  mediana = df_imp_num[name].median()
  df_imp_num[name]= df_imp_num[name].replace(np.nan, mediana)

df_imp_num.info()
```

```
<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            1068 non-null   float64
 1   CONDUCT_mS/cm       1068 non-null   float64
 2   SDT_M_mg/L          1068 non-null   float64
 3   FLUORUROS_mg/L      1068 non-null   float64
```

```
 4   DUR_mg/L             1068 non-null   float64
 5   COLI_FEC_NMP/100_mL  1068 non-null   float64
 6   N_NO3_mg/L           1068 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
```

Estadistica Descriptiva

```
df_imp_num.describe().transpose()
```

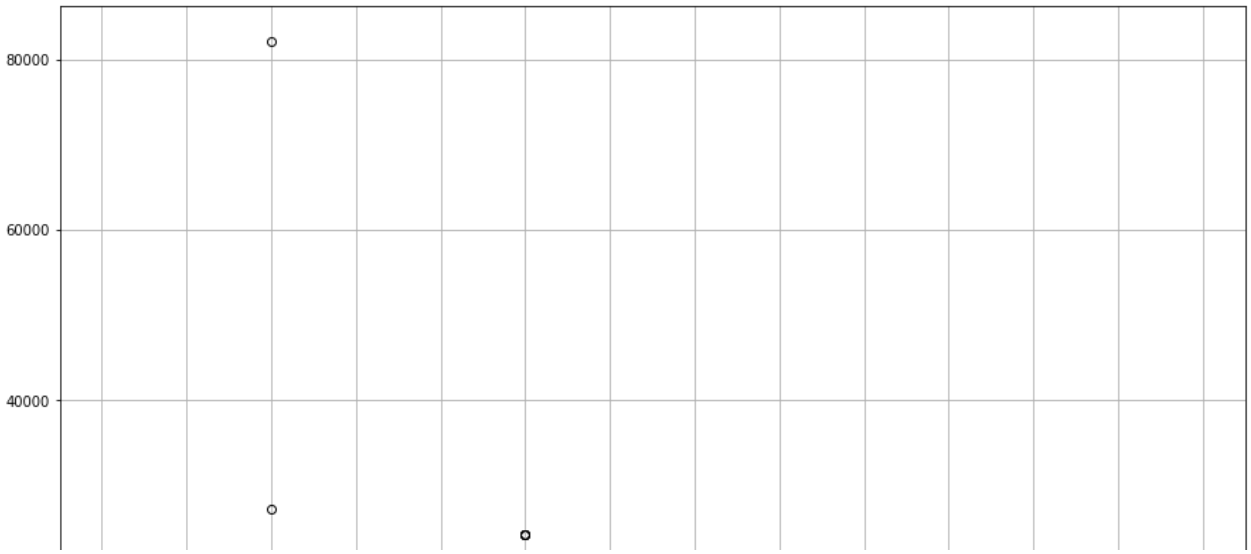
	count	mean	std	min	25%	
ALC_mg/L	1068.0	235.558455	116.661485	26.6400	164.048750	215.52
CONDUCT_mS/cm	1068.0	1137.133052	1242.292889	50.4000	505.500000	815.00
SDT_M_mg/L	1068.0	895.454185	2748.991295	25.0000	337.700000	550.40
FLUORUROS_mg/L	1068.0	1.075600	1.924278	0.2000	0.267175	0.50
DUR_mg/L	1068.0	347.842003	359.514579	20.0000	121.274100	245.33
COLI_FEC_NMP/100_mL	1068.0	355.490356	2052.457014	1.1000	1.100000	1.10
N_NO3_mg/L	1068.0	4.317663	8.341504	0.0200	0.650932	2.08
AS_TOT_mg/L	1068.0	0.019618	0.035209	0.0100	0.010000	0.01
CD_TOT_mg/L	1068.0	0.003030	0.000894	0.0030	0.003000	0.00
CR_TOT_mg/L	1068.0	0.012476	0.154435	0.0040	0.004000	0.00
HG_TOT_mg/L	1068.0	0.000467	0.000479	0.0004	0.000400	0.00
PB_TOT_mg/L	1068.0	0.004310	0.003342	0.0040	0.004000	0.00
MN_TOT_mg/L	1068.0	0.072478	0.376512	0.0015	0.001500	0.00
FE_TOT_mg/L	1068.0	0.410387	5.537974	0.0250	0.025000	0.04

Nivel de correlación de las variables

```
df_imp_num.corr()
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_m
ALC_mg/L	1.000000	0.217212	0.079572	0.068860	0.243
CONDUCT_mS/cm	0.217212	1.000000	0.286244	-0.025071	0.692
SDT_M_mg/L	0.079572	0.286244	1.000000	-0.013709	0.347
FLUORUROS_mg/L	0.068860	-0.025071	-0.013709	1.000000	-0.149
DUR_mg/L	0.243404	0.692656	0.347211	-0.149549	1.000
COLI_FEC_NMP/100_mL	-0.016338	0.018021	-0.001102	0.003564	0.031
N_NO3_mg/L	-0.000346	0.219881	0.101522	-0.019672	0.301
AS_TOT_mg/L	0.073458	-0.005047	-0.010092	0.444079	-0.106
CD_TOT_mg/L	0.032706	0.029083	0.010807	-0.015123	0.025
CR_TOT_mg/L	-0.014234	0.004436	-0.000494	-0.005205	0.007
HG_TOT_mg/L	0.069779	0.057007	0.020332	-0.028597	0.064
PB_TOT_mg/L	0.016989	0.024816	0.002517	-0.034191	-0.017
MN_TOT_mg/L	0.129942	0.095940	0.018963	-0.049742	0.083
FE_TOT_mg/L	0.043454	0.083172	0.020103	0.009994	0.050

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

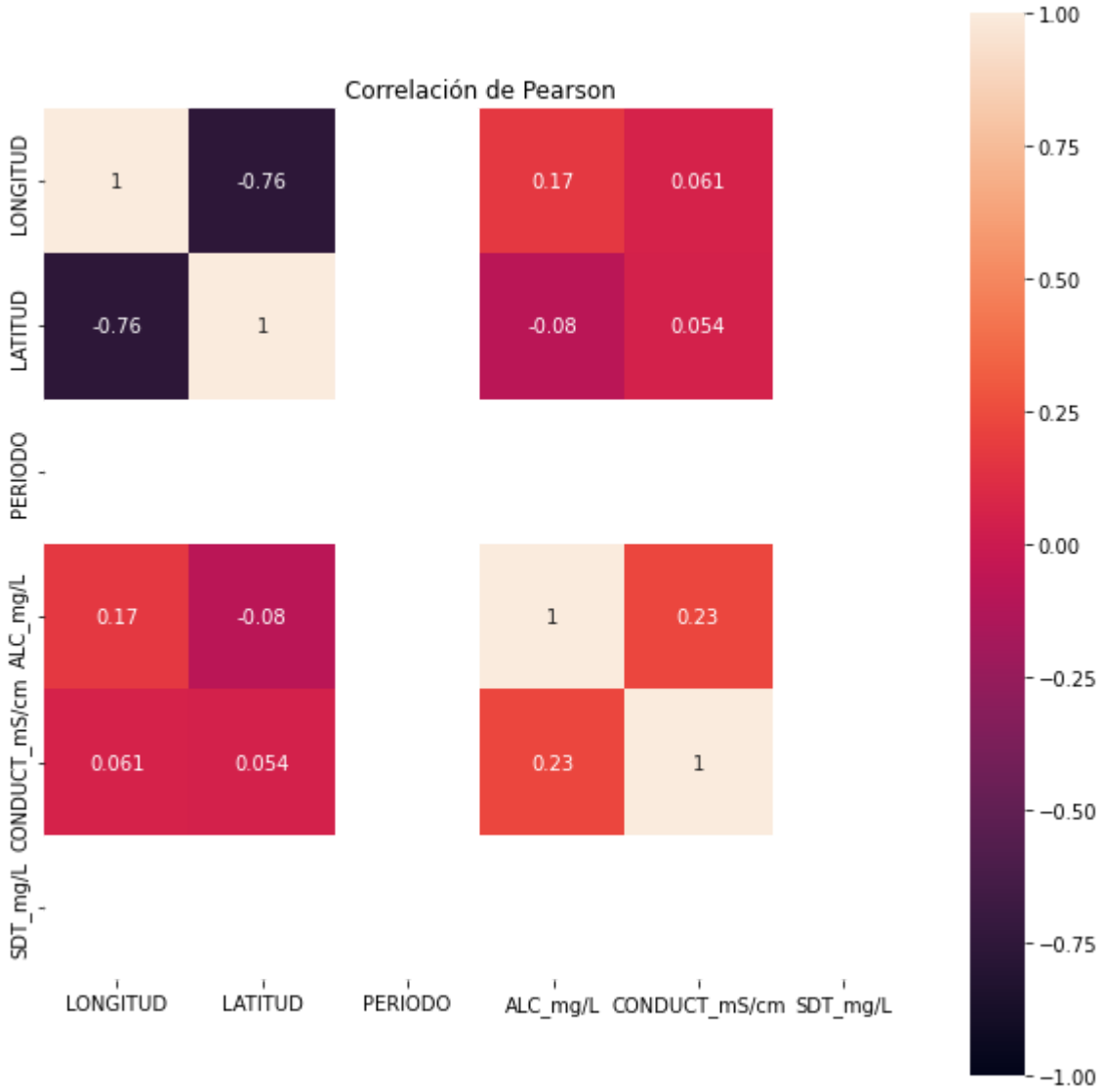



```
#Histograma de correlación
f, ax = plt.subplots(figsize=(10, 10))
tst = df.corr()['ALC_mg/L'].copy()
tst = tst.drop('ALC_mg/L')
tst.sort_values(inplace=True)
tst.plot(kind='bar', alpha=0.6)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff954f7e990>
```


```
#Matriz de correlación

f, ax = plt.subplots(figsize=(10, 10))
plt.title('Correlación de Pearson')
#sns.heatmap(df.astype(float).corr(), linewidths=0.7,vmax=1.0, square=True, annot=True)
corrmat = df.corr()
sns.heatmap(corrmat, vmin = -1, vmax=1, square=True, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff954ee51d0>
```



Algoritmo 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.
```

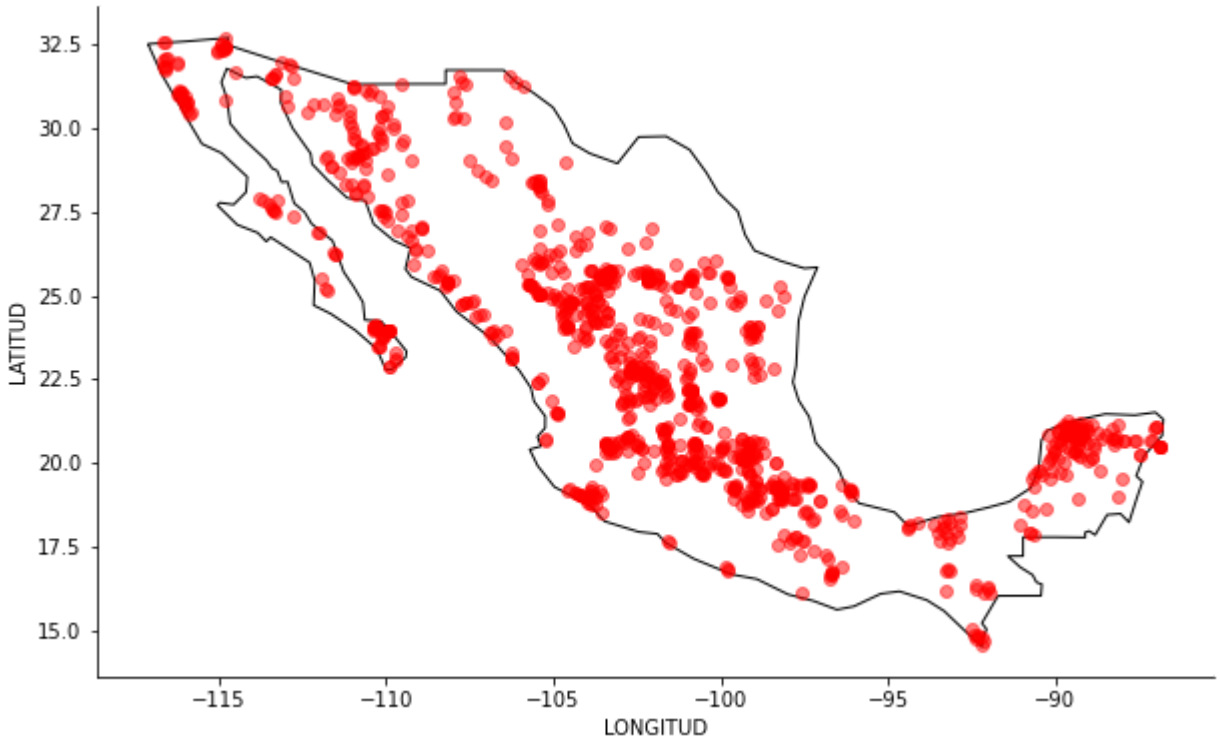
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.0221 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.2448 23.45138)

```
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
```

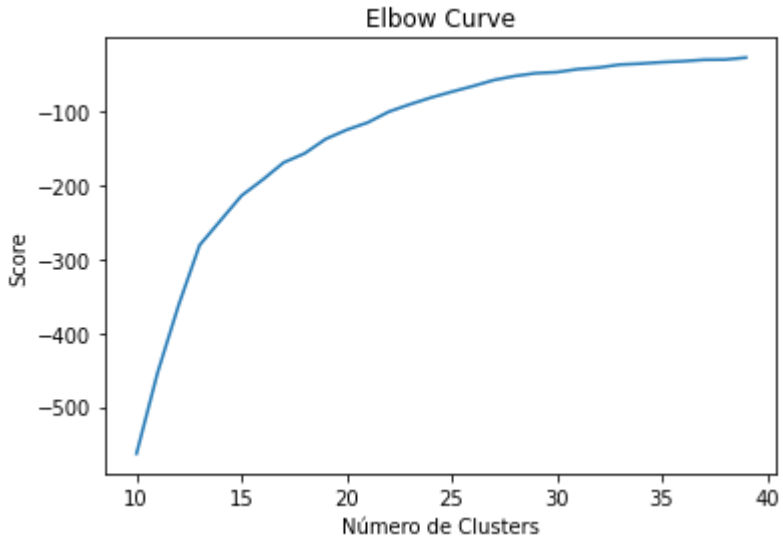
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.02210 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.24480 23.45138)
...	...	...	...
1063	-99.54191	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

```
#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	-98.449027	19.055544	POINT (-98.44903 19.05554)
1	-106.372195	23.435289	POINT (-106.37219 23.43529)
2	-89.627607	20.535159	POINT (-89.62761 20.53516)
3	-110.504722	28.989829	POINT (-110.50472 28.98983)
4	-102.463826	22.560986	POINT (-102.46383 22.56099)

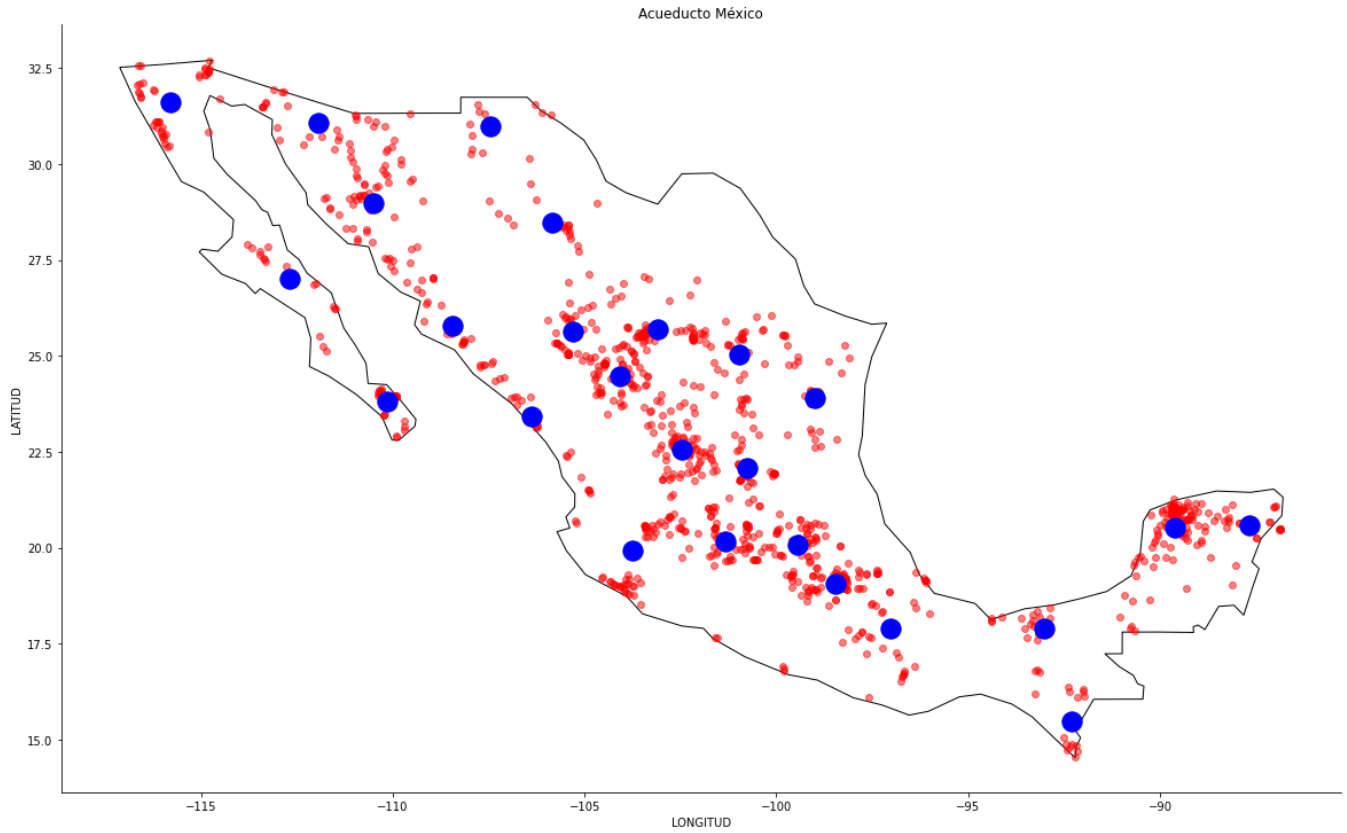
Realizar análisis para encontrar si existe una relación entre la calidad del agua y su ubicación geográfica a través de K- means.

7	-93.018481	17.887457	POINT (-93.01848 17.88746)

```
#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()
```

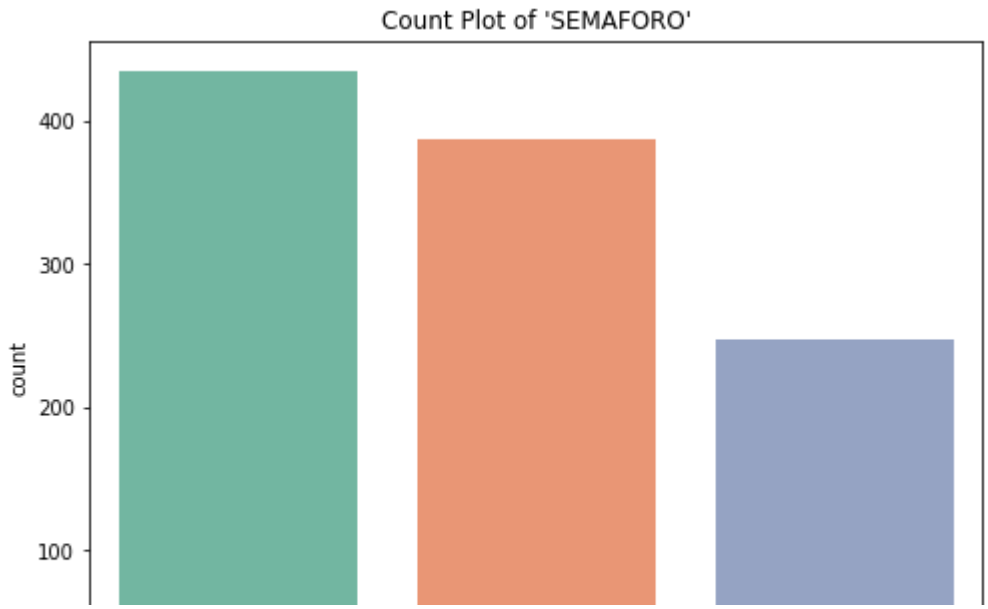


Hasta la ubicación de los clusters de acuerdo con el resultado de k-means podríamos considerar que son los puntos más relevantes en cuanto a ubicación de los acuiferos y la respectiva calidad del agua.

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:
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_
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: 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_
```

	LONGITUD	LATITUD	COORDENADAS	COLOR	CLUSTER
0	-102.02210	22.20887	POINT (-102.02210 22.20887)	Verde	4
1	-102.20075	21.99958	POINT (-102.20075 21.99958)	Verde	4
2	-102.28801	22.36685	POINT (-102.28801 22.36685)	Rojo	4

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.

...	...	...	...	...	...	...
1065	-99.82249	25.55197	POINT (-99.82249 25.55197)	Rojo	10	

```
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        4
1 -102.20075  21.99958  POINT (-102.20075 21.99958)  Verde        4
2 -102.28801  22.36685  POINT (-102.28801 22.36685)   Rojo        4
3 -102.29449  22.18435  POINT (-102.29449 22.18435)  Verde        4
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
10	Aceptable	green	-109.907091	23.946320
11	Potable - Excelente	green	-110.108253	23.807347
12	Buena calidad	red	-109.958920	23.973740
13	Aceptable	red	-110.061100	23.805540
14	Contaminada	red	-110.111850	23.742210
15	Potable - Excelente	green	-111.720090	25.135490
16	Potable - Excelente	green	-111.803800	25.240100
17	Potable - Excelente	yellow	-111.922210	25.504700
18	Aceptable	red	-109.964667	23.969660
19	Buena calidad	green	-111.485700	26.227900

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

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)


# run 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: 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.
```

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

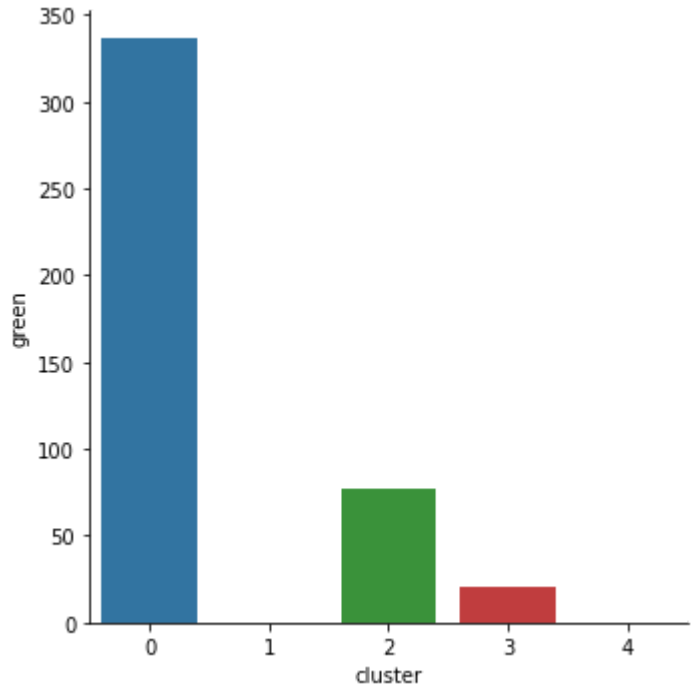
```
types['cluster']  = kmeans.labels_
types.head()
```

	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

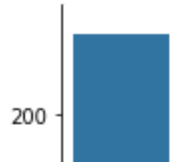
```
types.groupby('cluster').mean()
```

cluster	green	red	yellow
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

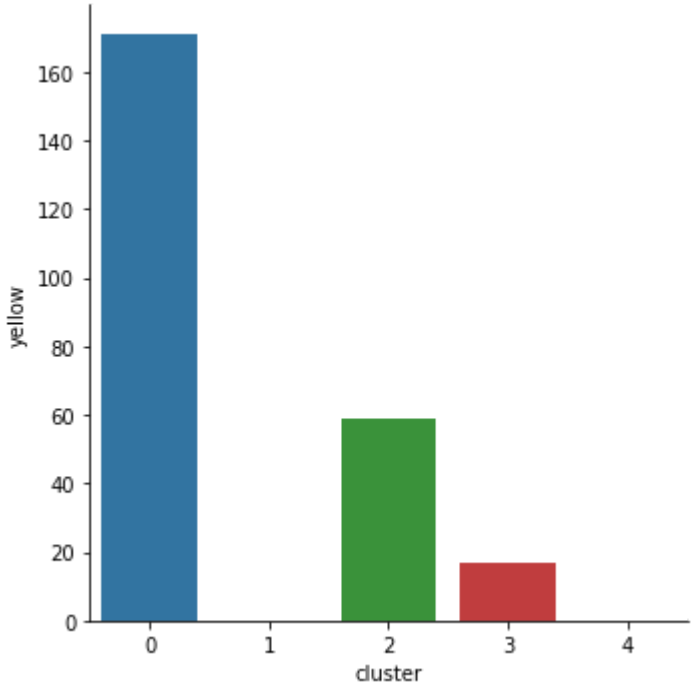
```
sns.catplot(x='cluster', y='green', data=types, kind='bar');
```




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




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




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

	SITIO	ORGANISMO_DE_CUENCA	ACUIFERO	LONGITUD	LATITUD	
0	POZO SAN GIL	LERMA SANTIAGO PACIFICO	VALLE DE CHICALOTE	-102.02210	22.20887	
1	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	VALLE DE CHICALOTE	-102.20075	21.99958	AGUAS
2	POZO COSIO	LERMA SANTIAGO PACIFICO	VALLE DE AGUASCALIENTES	-102.28801	22.36685	

```
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
1	PRINCIPAL-REGION LAGUNERA	28
2	ALTO ATOYAC	19

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

```
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	PENINSULA DE YUCATAN	-90.55914	19.74566	CAMPEC

```
#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)

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(
```

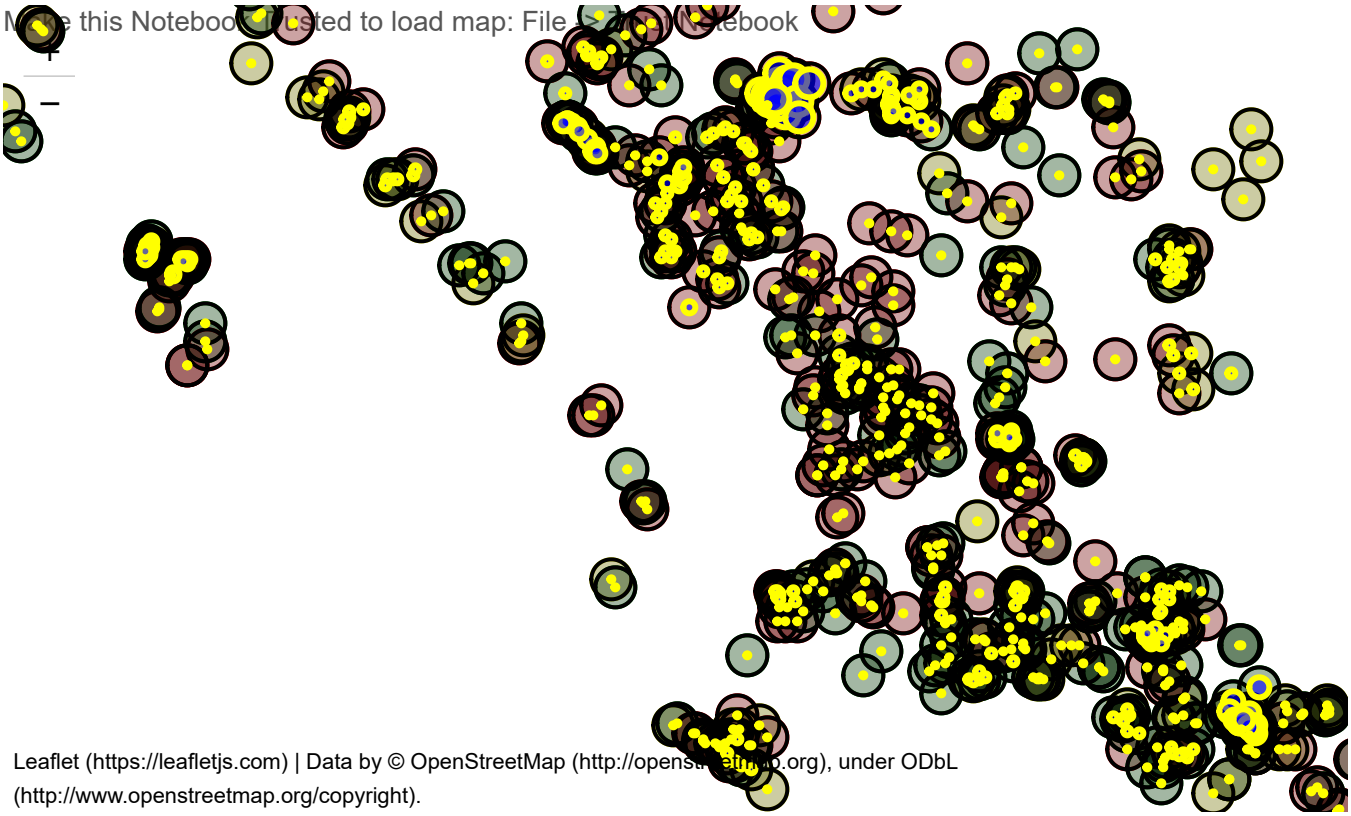
```
            location=[row[1], row[0]],
            radius=10,
            weight=2,
            fill=True,
            fill_color=color,
            color=color
        ).add_to(Mexico_map)
    Mexico_map
```

```
/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_
  """
```

Make this Notebook Trusted to load map: File > Trust Notebook

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)
```




```
map_clusters = folium.Map(location=[19.432, -99.13], zoom_start=6)

# set color scheme for the clusters
x = np.arange(4)
ys = [i + x + (i*x)**2 for i in range(4)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
```

```
for lat, lng, cluster, city, state in zip(number_of_occurences['LATITUD'],
                                          number_of_occurences['LONGITUD'],
                                          number_of_occurences['Cluster'],
                                          number_of_occurences['MUNICIPIO'],
                                          number_of_occurences['ESTADO']):
    #label = folium.Popup(str(city)+ ','+str(state) + '- Cluster ' + str(cluster), parse_html
    folium.vector_layers.CircleMarker(
        [lat, lng],
        radius=5,
        #popup=label,
        tooltip = str(city)+ ','+str(state) + '- Cluster ' + str(cluster),
        color=rainbow,
        fill=True,
        fill_color=rainbow,
        fill_opacity=0.9).add_to(map_clusters)

map_clusters
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in get_loc(self,
key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

                                 4 frames

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Cluster'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in get_loc(self,
key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:
```