7

 # Ciencia y analítica de datos

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: 14/11/2022

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 grai
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
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
```

```
Collecting qeds
  Downloading qeds-0.7.0.tar.gz (24 kB)
Collecting fiona
  Downloading Fiona-1.8.22-cp37-cp37m-manylinux2014_x86_64.whl (16.7 MB)
     |████████████████████████████████| 16.7 MB 468 kB/s
Collecting geopandas
  Downloading geopandas-0.10.2-py2.py3-none-any.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 51.7 MB/s
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: folium in /usr/local/lib/python3.7/dist-packages
Collecting pyLDAvis
  Downloading pyLDAvis-3.3.1.tar.gz (1.7 MB)
     |████████████████████████████████| 1.7 MB 51.2 MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
    Preparing wheel metadata ... done
Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
Collecting quandl
  Downloading Quandl-3.7.0-py2.py3-none-any.whl (26 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (
Collecting quantecon
  Downloading quantecon-0.5.3-py3-none-any.whl (179 kB)
     |████████████████████████████████| 179 kB 63.9 MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas_datareader in /usr/local/lib/python3.7/dis
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-pack
Collecting cligj>=0.5
  Downloading cligj-0.7.2-py3-none-any.whl (7.1 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-package
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packag
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packa
Collecting pyproj>=2.2.0
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 50.6 MB/s
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dis
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-pack
```

```
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_aq
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

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

```
<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_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)
```

```
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

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	
1065	330.690	2600.0	NaN	1873	0.7574	660.2126	
1066	193.140	873.0	NaN	690.6667	0.7108	406.368	
1067	263.070	817.0	NaN	495	0.4002	362.544	

1068 rows × 15 columns

```
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
```

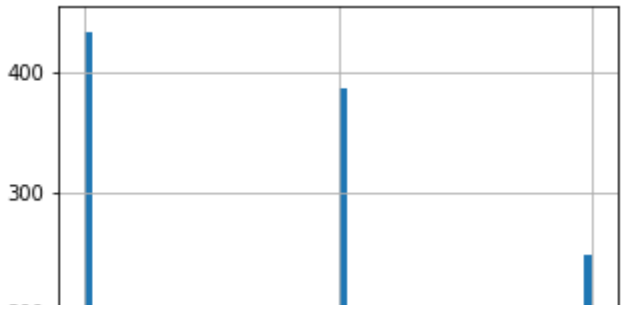
```
0.4343      1
0.4002      1
Name: FLUORUROS_mg/L, Length: 862, dtype: int64
 nombre de la columna ------------------------DUR_mg/L
 sumatoria por valores uncos--------------------
<20         26
121.512      6
53.8542      4
109.56       4
428.27       3
            ..
51.454       1
103          1
24.8725      1
71.6184      1
362.544      1
Name: DUR_mg/L, Length: 889, dtype: int64
 nombre de la columna ------------------------COLI_FEC_NMP/100_mL
 sumatoria por valores uncos--------------------
<1.1    737
10       37
40       28
20       19
31       13
        ...
1607      1
175       1
2247      1
530       1
1658      1
Name: COLI_FEC_NMP/100_mL, Length: 125, dtype: int64
 nombre de la columna ------------------------N_NO3_mg/L
 sumatoria por valores uncos--------------------
<0.02       65
0.096        3
0.163        2
1.955        2
0.147        2
            ..
0.7694       1
1.2477       1
0.0497       1
0.1972       1
0.811876     1
Name: N_NO3_mg/L, Length: 995, dtype: int64
 nombre de la columna ------------------------AS_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.01     815
0.0135      4
0.0201      3
0.0217      3
0.0154      3
          ...
0.027       1
0.0334      1
0.0376      1
0.0208      1
```

```
0.0200      1
0.1397      1
Name: AS_TOT_mg/L, Length: 209, dtype: int64
 nombre de la columna ------------------------CD_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.003     1066
0.0056        1
0.03211       1
Name: CD_TOT_mg/L, dtype: int64
 nombre de la columna ------------------------CR_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.005     854
0.005        7
0.0051       6
0.0053       5
0.0052       4
           ...
0.02508      1
0.01684      1
0.01874      1
0.03963      1
0.01341      1
Name: CR_TOT_mg/L, Length: 168, dtype: int64
 nombre de la columna ------------------------HG_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.0005    968
0.0006      13
0.0005       9
0.00086      4
0.00051      3
           ...
0.00216      1
0.0014       1
0.00168      1
0.00135      1
0.00062      1
Name: HG_TOT_mg/L, Length: 61, dtype: int64
 nombre de la columna ------------------------PB_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.005     1038
0.01225       1
0.00709       1
0.00596       1
0.046         1
0.005         1
0.00744       1
0.00644       1
0.00619       1
0.00703       1
0.0133        1
0.00734       1
0.00557       1
0.00777       1
0.01075       1
0.0116        1
0.0399        1
0.00556       1
```

```
0.00859       1
0.0086        1
0.00769       1
0.00737       1
0.00818       1
0.00813       1
0.01117       1
0.0152        1
0.0219        1
0.0809        1
0.0135        1
0.049         1
0.0053        1
Name: PB_TOT_mg/L, dtype: int64
 nombre de la columna ------------------------MN_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.0015    545
0.0017      12
0.0021      10
0.0016       9
0.003        8
           ...
0.0056       1
0.0193       1
0.00445      1
0.0208       1
0.0242       1
Name: MN_TOT_mg/L, Length: 362, dtype: int64
 nombre de la columna ------------------------FE_TOT_mg/L
 sumatoria por valores uncos--------------------
<0.025    401
0.0288      4
0.0492      4
0.0471      3
0.0564      3
          ...
0.1118      1
0.0565      1
0.3947      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.hist(bins = 60, figsize=(5,5))
y['SEMAFORO'].hist(bins = 60, figsize=(5,5))
```

```
<class 'pandas.core.frame.DataFrame'>
<matplotlib.axes._subplots.AxesSubplot at 0x7f29de8bbdd0>
```




```
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()
```

```
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: '
  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: '
  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: '
/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: '
/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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1068 entries, 0 to 1067
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ALC_mg/L             1064 non-null   float64
 1   CONDUCT_mS/cm        1062 non-null   float64
 2   SDT_mg/L             0 non-null      float64
 3   SDT_M_mg/L           1066 non-null   float64
 4   FLUORUROS_mg/L       1068 non-null   float64
 5   DUR_mg/L             1067 non-null   float64
 6   COLI_FEC_NMP/100_mL  1068 non-null   float64
 7   N_NO3_mg/L           1067 non-null   float64
 8   AS_TOT_mg/L          1068 non-null   float64
 9   CD_TOT_mg/L          1068 non-null   float64
 10  CR_TOT_mg/L          1068 non-null   float64
 11  HG_TOT_mg/L          1068 non-null   float64
 12  PB_TOT_mg/L          1068 non-null   float64
 13  MN_TOT_mg/L          1068 non-null   float64
 14  FE_TOT_mg/L          1068 non-null   float64
dtypes: float64(15)
memory usage: 125.3 KB
```

```
# la columna  SDT_mg/L             0 non-null      float64, esta vacia, asi que nos la

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
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
```

```
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	

	COLI_FEC_NMP/100_mL	N_NO3_mg/L	AS_TOT_mg/L	CD_TOT_mg/L	\
count	1068.000000	1068.000000	1068.000000	1068.000000	
mean	355.490356	4.317663	0.019618	0.003030	
std	2052.457014	8.341504	0.035209	0.000894	
min	1.100000	0.020000	0.010000	0.003000	
25%	1.100000	0.650932	0.010000	0.003000	
50%	1.100000	2.080932	0.010000	0.003000	
75%	13.250000	5.200047	0.010000	0.003000	
max	24196.000000	121.007813	0.452200	0.032110	

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

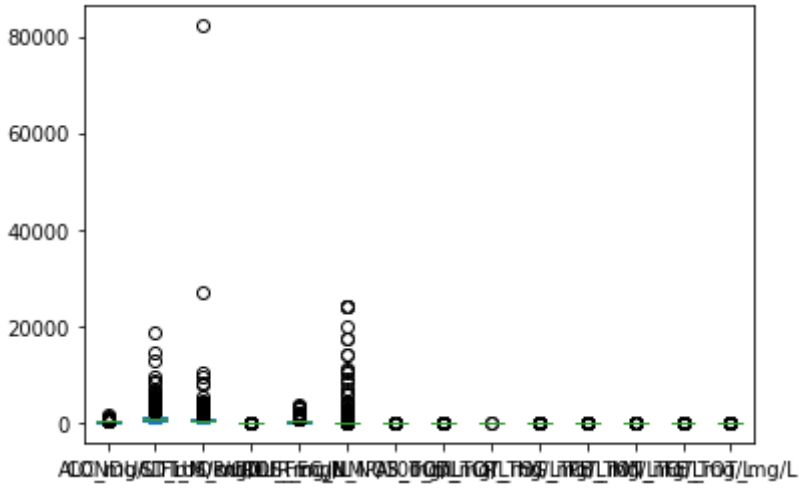
Preparar los datos

std	0.154435	0.000479	0.003342	0.376512	5.537974

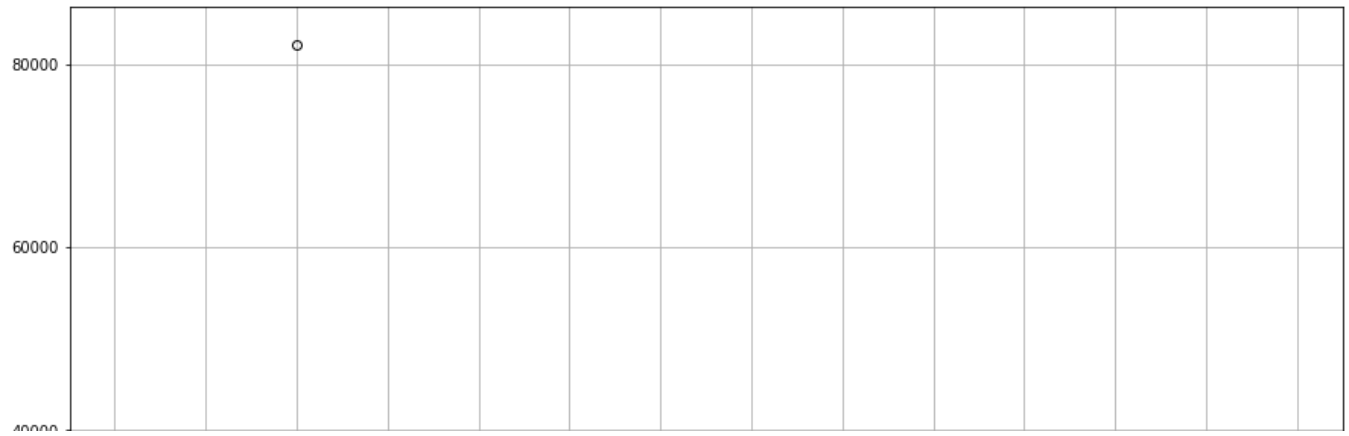
```
#Matriz de Correlación
df_limpio.corr()
```

```
#Vemos los outliers en la gráfica, pero necesitamos acercarnos
df_limpio.plot.box()
```

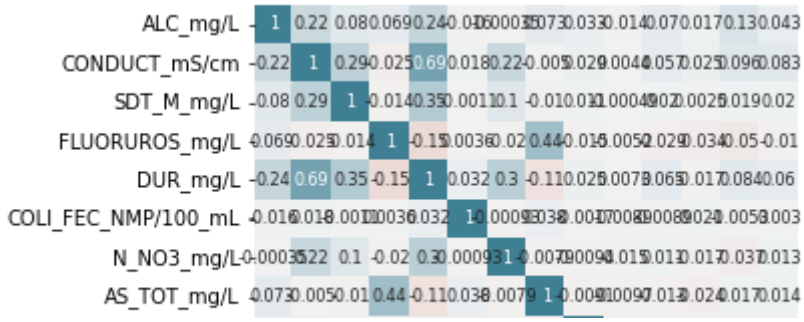
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f29e77e8d10>
```




```
#Opcion 2
outliers = df_limpio.boxplot(figsize = (15,10),showmeans = True)
outliers.plot()
plt.xticks(rotation=90)
plt.show()
```

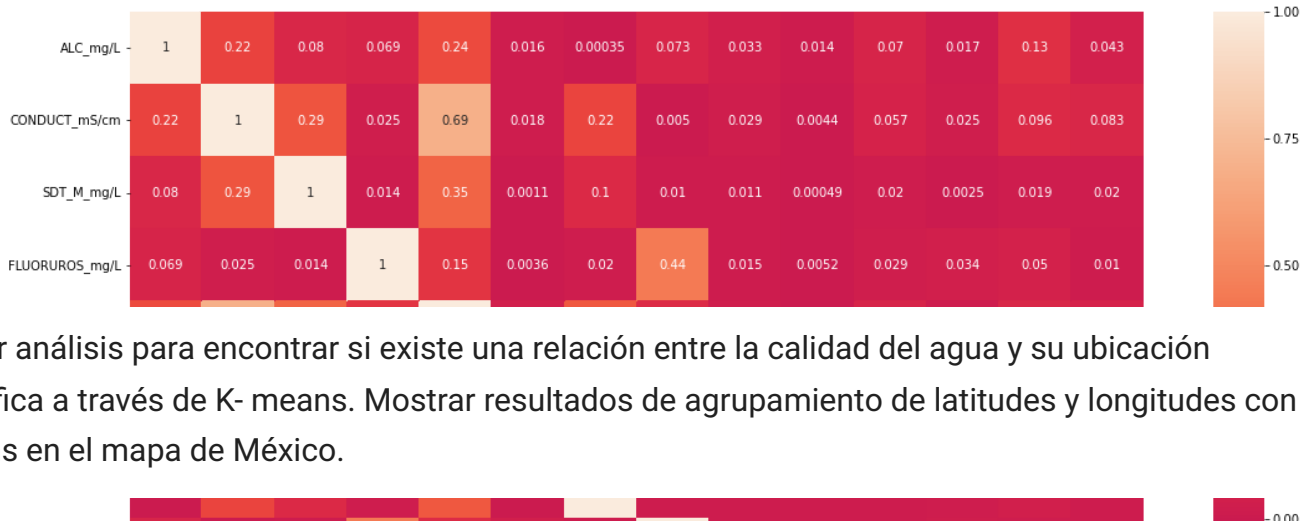



```
#Gráfica de correlación a color

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 0x7f29e4897d50>
```



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
```

y

```
0        Verde
1        Verde
2         Rojo
3        Verde
4         Rojo
         ...
1063      Rojo
1064      Rojo
1065      Rojo
1066     Verde
1067     Verde
Name: SEMAFORO, Length: 1068, dtype: object
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f29e48b5e50>
```


```
#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/stab
  This is separate from the ipykernel package so we can avoid doing imports unti
/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/stab
  after removing the cwd from sys.path.
```

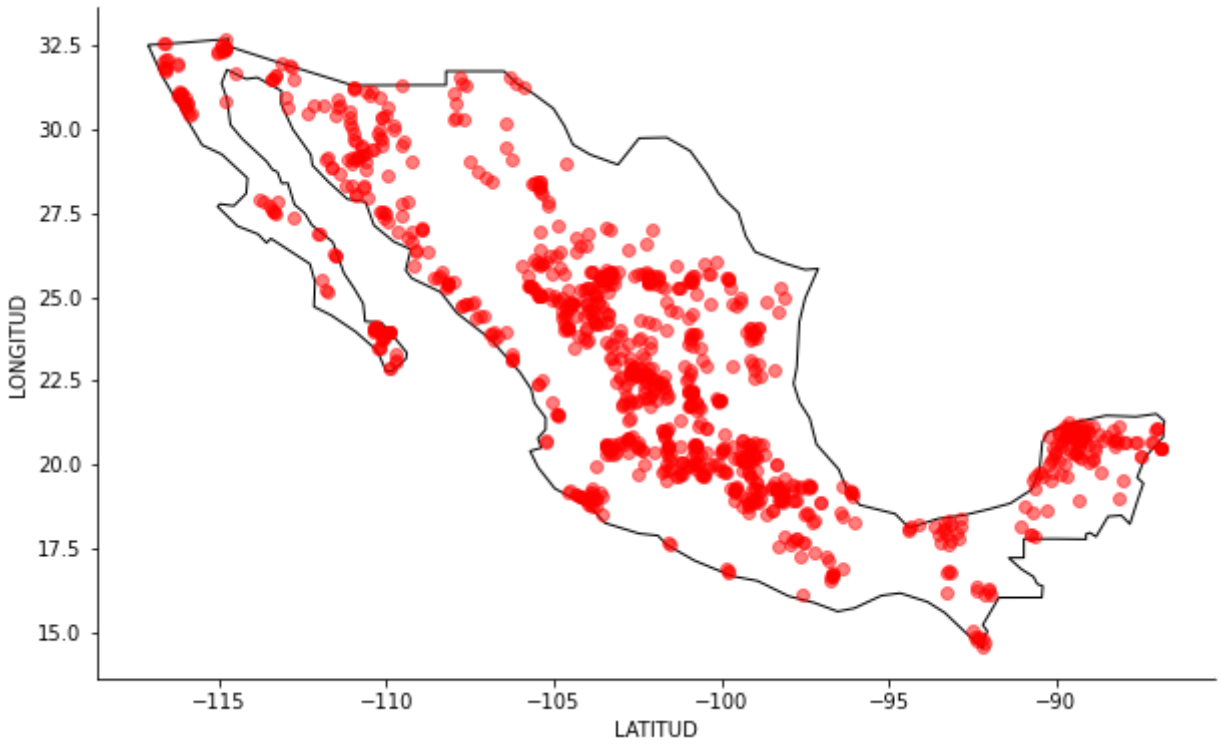
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.0221 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.2448 23.45138)

```
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
```

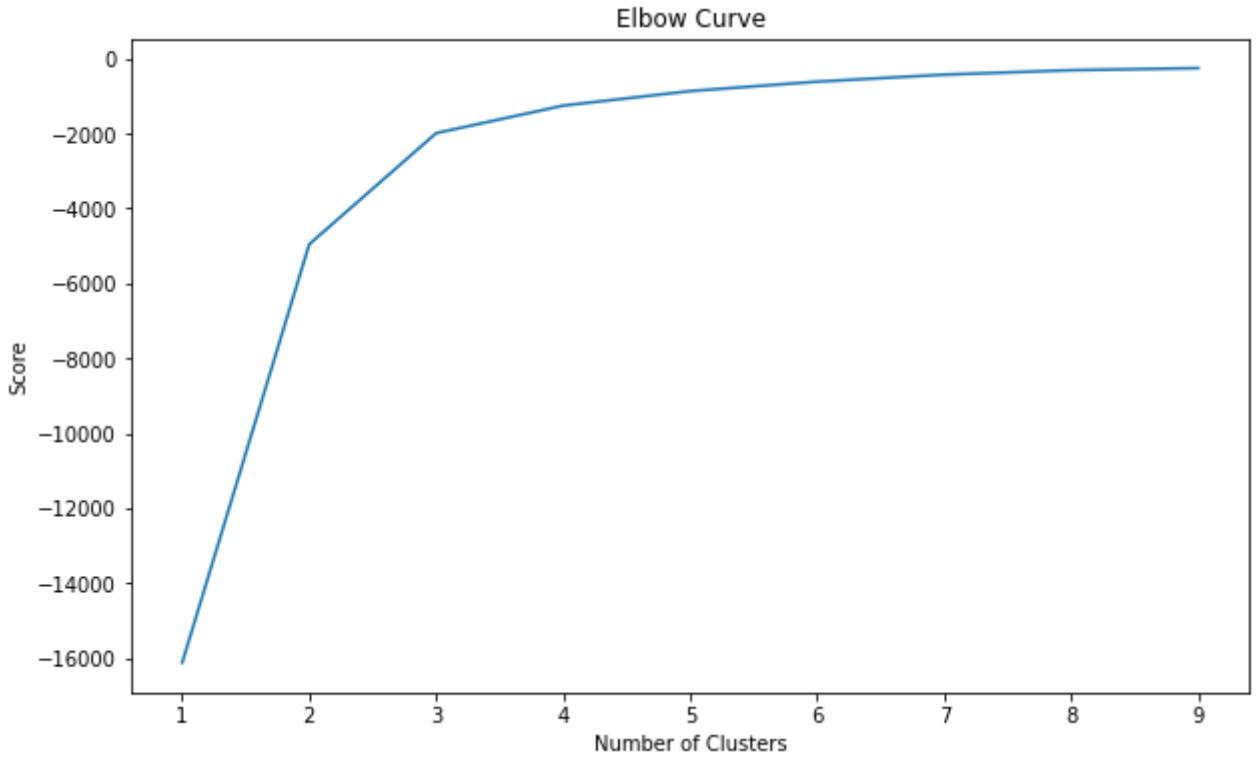
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.02210 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.24480 23.45138)
...	...	...	...
1063	-99.54191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

```
numero_de_closters = range(1,10)
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=3).fit(X)
centroids = kmeans.cluster_centers_
labels = kmeans.predict(X)
C = kmeans.cluster_centers_

C_DF = pd.DataFrame(C)
```

```
C_DF["Coordinates"] = list(zip(C_DF[0], C_DF[1]))
C_DF["Coordinates"] = C_DF["Coordinates"].apply(Point)

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

	0	1	Coordinates
0	-90.728470	19.473362	POINT (-90.72847 19.47336)
1	-110.794488	28.438202	POINT (-110.79449 28.43820)
2	-101.722127	22.254226	POINT (-101.72213 22.25423)

```
#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())
```

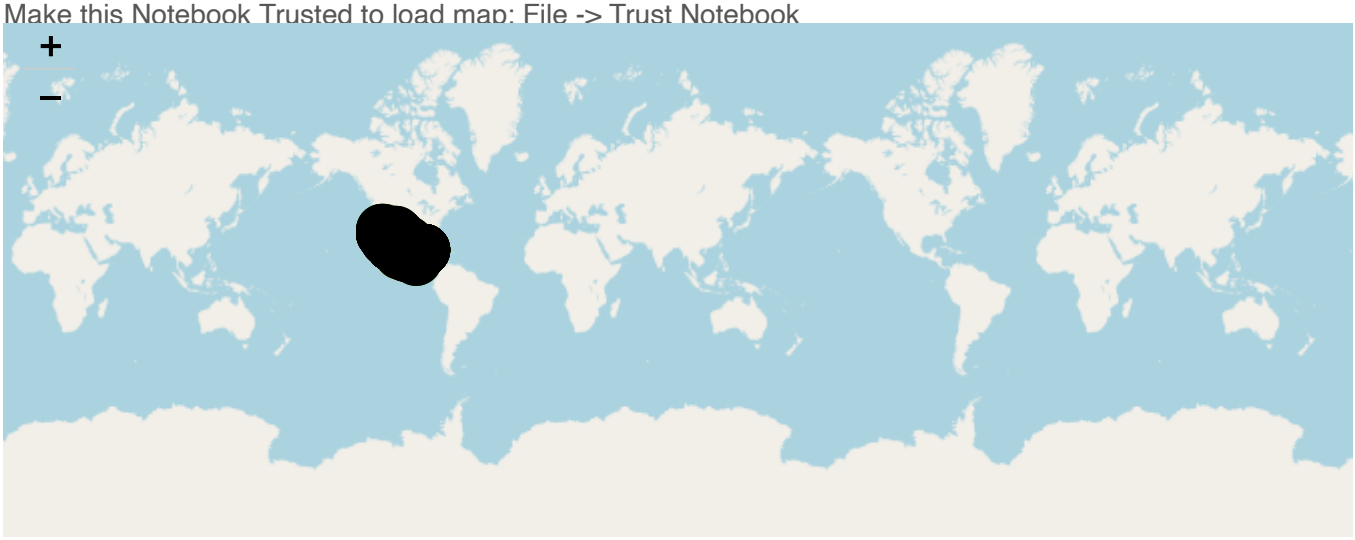
```
  SEMAFORO codigo_color ingles
0    Verde        Verde  Verde
1    Verde        Verde  Verde
2     Rojo         Rojo   Rojo
3    Verde        Verde  Verde
4     Rojo         Rojo   Rojo
    LONGITUD   LATITUD                  COORDENADAS
0 -102.02210  22.20887  POINT (-102.02210 22.20887)
1 -102.20075  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	codigo_color	ingles	SEMAPHORE
0	Verde	Verde	Verde	green
1	Verde	Verde	Verde	green
2	Rojo	Rojo	Rojo	red
3	Verde	Verde	Verde	green
4	Rojo	Rojo	Rojo	red
...	...	...	...	...
1063	Rojo	Rojo	Rojo	red
1064	Rojo	Rojo	Rojo	red

```
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
```

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



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

```
DataFrame_Final['COLOR'] = y['SEMAFORO']
DataFrame_Final.head()

'''
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') '''
```

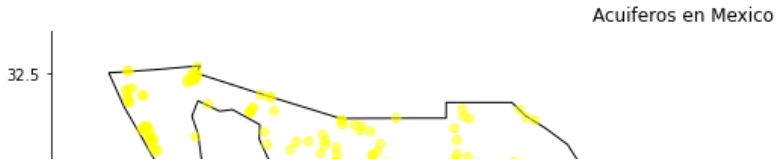
```
'\nfor i in y:\n  y.codigo_color[i] = y[i].str.replace('Verde','1') \n  y[i] = y
[i].str.replace('Amarillo','2') \n  y[i] = y[i].str.replace('Rojo','3') '
```

```
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
```

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

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

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




```
#La propuesta de rigo
#1 Asignar un numero de cluster a cada punto
#2 Sacarle la moda  cada cluster
#3 Graficar cada cluster con su color predominante en la media de los puntos que tiene

# Como hacer que el cada cluster imprima el color que le corresponda
# Como agrupar colores
# Sera que se puede psa

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()
```

```
#https://paratodomexico.com/geografia-de-mexico/hidrografia-de-mexico/acuiferos-de-mex
```

CAMPO	DESCRIPCION
CLAVE	Clave del sitio de monitoreo
SITIO	Nombre del sitio de muestreo
ORGANISMO_DE_CUENCA	Nombre del Organismo de Cuenca de agua donde se localiza el sitio de monitoreo
ESTADO	Estado donde se encuentra el sitio de muestreo
MUNICIPIO	Municipio donde se encuentra el sitio de muestreo
ACUIFERO	Acuífero donde se encuentra el sitio de muestreo
SUBTIPO	Subtipo de cuerpo de agua donde se encuentra el sitio de muestreo
LONGITUD	Coordenada de longitud
LATITUD	Coordenada de latitud
ANIO/PERIODO	Anio o periodo en que se realizo el muestreo
ALC_mg/L	Valor de Alcalinidad Total, en miligramos por litro
AS_TOT_mg/L	Valor de Arsenico Total, en miligramos por litro
CD_TOT_mg/L	Valor de Cadmio Total, en miligramos por litro
COLI_FEC_NMP/100_mL	Valor de Coliformes Fecales, en Numero Mas Probable por 100 mililitros
CONDUCT_mS/cm	Valor de Conductividad en microSiemens por centimetro
CR_TOT_mg/L	Valor de Cromo Total, en miligramos por litro
DUR_mg/L	Valor de Dureza Total, en miligramos por litro
FE_TOT_mg/L	Valor de Hierro Total, en miligramos por litro
FLUORUROS_mg/L	Valor de Fluoruros Totales (F-), en miligramos por litro
HG_TOT_mg/L	Valor de Mercurio Total, en miligramos por litro
MN_TOT_mg/L	Valor de Manganeso Total, en miligramos por litro
N_NO3_mg/L	Valor de Nitrogeno de Nitratos, en miligramos por litro
PB_TOT_mg/L	Valor de Plomo Total, en miligramos por litro
SDT_M_mg/L	Valor de Solidos Disueltos Totales-Medidos, en miligramos por litro
SDT_mg/L	Valor de Solidos Disueltos Totales, en miligramos por litro
CALIDAD_ALC	Clasificacion de la calidad del agua de acuerdo con el indicador Alcalinidad Total
CALIDAD_AS	Clasificacion de la calidad del agua de acuerdo con el indicador Arsenico Total
CALIDAD_CD	Clasificacion de la calidad del agua de acuerdo con el indicador Cadmio Total
CALIDAD_COLI_FEC	Clasificacion de la calidad del agua de acuerdo con el indicador Coliformes Fecales
CALIDAD_CONDUC	Clasificacion de la calidad del agua de acuerdo con el indicador Conductividad
CALIDAD_CR	Clasificacion de la calidad del agua de acuerdo con el indicador Cromo Total
CALIDAD_DUR	Clasificacion de la calidad del agua de acuerdo con el indicador Dureza Total
CALIDAD_FE	Clasificacion de la calidad del agua de acuerdo con el indicador Hierro Total
CALIDAD_FLUO	Clasificacion de la calidad del agua de acuerdo con el indicador Fluoruros Totales
CALIDAD_HG	Clasificacion de la calidad del agua de acuerdo con el indicador Mercurio Total
CALIDAD_MN	Clasificacion de la calidad del agua de acuerdo con el indicador Manganeso Total
CALIDAD_N_NO3	Clasificacion de la calidad del agua de acuerdo con el indicador Nitrogeno de Nitratos
CALIDAD_PB	Clasificacion de la calidad del agua de acuerdo con el indicador Plomo Total
CALIDAD_SDT_ra	Clasificacion de la calidad del agua de acuerdo con el indicador de los Solidos Disueltos Totales (Riego agricola)
CALIDAD_SDT_salin	Clasificacion de la calidad del agua de acuerdo con el indicador de los Solidos Disueltos Totales (Salinizacion)
CUMPLE_CON_ALC	Indica si cumple con la calidad de Baja, Media, o Alta para el Indicador Alcalinidad Total
CUMPLE_CON_AS	Indica si cumple con la calidad de Potable - Excelente o Apta como FAAP, para el Indicador Arsenico Total
CUMPLE_CON_CD	Indica si cumple con la calidad de Potable - Excelente, para el Indicador Cadmio Total
CUMPLE_CON_CF	Indica si cumple con la calidad de Potable - Excelente, o Buena calidad, Aceptable, para el Indicador Coliformes Fecales
CUMPLE_CON_COND	Indica si cumple con la calidad de Excelente para riego, Buena para riego, o Permisible para riego, para el Indicador Conduc
CUMPLE_CON_CR	Indica si cumple con la calidad de Potable - Excelente, para el Indicador Cromo Total
CUMPLE_CON_DUR	Indica si cumple con la calidad de Potable - Suave, Potable - Moderadamente suave, o Potable - Dura, para el Indicador Dur
CUMPLE_CON_FE	Indica si cumple con la calidad de Potable - Excelente, para el Indicador Hierro Total
CUMPLE_CON_FLUO	Indica si cumple con la calidad de Baja, Media, o Potable - Optima, para el Indicador Fluoruros Totales
CUMPLE_CON_HG	Indica si cumple con la calidad de Potable - Excelente, para el Indicador Mercurio Total
CUMPLE_CON_MN	Indica si cumple con la calidad de Potable - Excelente, para el Indicador Manganeso Total
CUMPLE_CON_NO3	Indica si cumple con la calidad de Potable - Excelente o Potable - Buena calidad, para el Indicador Nitrogeno de Nitratos
CUMPLE_CON_PB	Indica si cumple con la calidad de Potable - Excelente, para el Indicador Plomo Total
CUMPLE_CON_SDT_ra	Indica si cumple con la calidad de Excelente para riego, Cultivos sensibles o Cultivos con manejo especial, para el Indicador
CUMPLE_CON_SDT_salin	Indica si cumple con la calidad de Potable - Dulce o Ligeramente salobres, para el Indicador Solidos Disueltos Totales (Salir
CONTAMINANTES	Contaminantes presentes en incumplimiento (Contaminados)
SEMAFORO	Indica el nivel de contaminacion de acuerdo a los contaminantes presentes
ND	No disponible