Maestría en Inteligencia Artificial Aplicada

Curso: Ciencia y Analitica de Datos

Tecnológico de Monterrey

Profesor Titular: Maria de la Paz Rico Fernandez

Profesor Tutor: Juan Miguel Meza Méndez

Equipo 170

Freddy Armendariz Herrera - A01793672

Samuel Elias Flores Gonzalez - A01793668

Reto - Entrega 1 - Limpieza, Analisis, Visualizaicon, Kmeans

Fecha: 13 de Noviembre del 2022

```
# Incluye aquí todos módulos, librerías y paquetes que requieras.

# Descargar Data Set
# ==============================
import requests, zipfile
from io import BytesIO

# Tratamiento de datos
# ==============================
import numpy as np
import pandas as pd

# Graficos
# ==============================
import seaborn as sns
import matplotlib.pyplot as plt

# Preprocesado y modelado
# ==============================
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
```

```
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_validate
from sklearn.metrics import make_scorer
from tabulate import tabulate
from sklearn.model_selection import GridSearchCV
from sklearn.inspection import permutation_importance
from sklearn.preprocessing import LabelEncoder

# Geopandas
# ==============================
! pip install qeds fiona geopandas xgboost gensim folium pyLDAvis descartes

import geopandas as gpd
from shapely.geometry import Point


# Kmeans
# ==============================
from sklearn.cluster import KMeans


# Mapa
# ==============================
import folium
```

```
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Collecting cligj>=0.5
  Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (
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
Collecting pyproj>=2.2.0
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 43.8 MB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist
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
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from
Collecting funcy
  Downloading funcy-1.17-py2.py3-none-any.whl (33 kB)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
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: 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: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packa
Collecting inflection>=0.3.1
  Downloading inflection-0.5.1-py2.py3-none-any.whl (9.5 kB)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.7
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: qeds, pyLDAvis, sklearn
  Building wheel for qeds (setup.py) ... done
  Created wheel for qeds: filename=qeds-0.7.0-py3-none-any.whl size=27812 sha256=d6ef
  Stored in directory: /root/.cache/pip/wheels/fc/8c/52/0cc036b9730b75850b9845770780f
  Building wheel for pyLDAvis (PEP 517) ... done
  Created wheel for pyLDAvis: filename=pyLDAvis-3.3.1-py2.py3-none-any.whl size=13689
  Stored in directory: /root/.cache/pip/wheels/c9/21/f6/17bcf2667e8a68532ba2fbf6d5c72
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=2344 sh
  Stored in directory: /root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b310477667
Successfully built qeds pyLDAvis sklearn
Installing collected packages: munch, inflection, cligj, click-plugins, sklearn, quan
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.22 funcy-1.17 geopan
```

Eleccion de una base de datos

```
# Extraccion de la carpeta comprimida
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()

# Lectura del csv como dataframe
path = "Datos_de_calidad_del_agua_2020/Datos_de_calidad_del_agua_de_sitios_de_monitoreo_de_ag
df=pd.read_csv(path, encoding="latin1")

# Datos de la calidad de aguas superficiales
df.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	

5 rows × 57 columns

Limpieza de los datos

```
# Se verifica la cantidad de datos nulos en cada columna
df.isna().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
```

```
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
```

```
# Se descartan las columnas de CONTAMINANTES y SDT_mg/L ya que la mayor parte de sus datos so
df.drop(["CONTAMINANTES","SDT_mg/L"], inplace=True, axis=1)
```

Las demas columnas presentaban un 6 datos nulos como maximo, estos se pueden considerar despreciables, por lo que se procede a eliminarlos.

```
#Eliminamos los datos NaN
df.dropna(inplace = True)
```

```
#Se corrobora si quedo algún dato vacío, False = No hay datos nulos
df.isna().values.any()
```

```
False
```

Al analizar el set de datos, se puede inferir que los datos categoricos son dependientes de los datos numericos, es decir, hacen referencia a ellos, provocando asi una redundancia en los

mismos, por lo tanto se procede a eliminar estas columnas y utilizar solo las numericas.

```
col_num = ['LONGITUD','LATITUD','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_mg/L','PB_TOT_mg/L'
df_new = df[col_num]
df_new.head()
```

	LONGITUD	LATITUD	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	C
0	-102.02210	22.20887	229.990	940.0	603.6	0.9766	213.732	
1	-102.20075	21.99958	231.990	608.0	445.4	0.9298	185.0514	
2	-102.28801	22.36685	204.920	532.0	342	1.8045	120.719	
3	-102.29449	22.18435	327.000	686.0	478.6	1.1229	199.879	
4	-110.24480	23.45138	309.885	1841.0	1179	0.2343	476.9872	

```
# Se comprueba la cantidad de datos nulos por columna y su tipo de dato
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1054 entries, 0 to 1067
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   LONGITUD             1054 non-null   float64
 1   LATITUD              1054 non-null   float64
 2   ALC_mg/L             1054 non-null   float64
 3   CONDUCT_mS/cm        1054 non-null   float64
 4   SDT_M_mg/L           1054 non-null   object 
 5   FLUORUROS_mg/L       1054 non-null   object 
 6   DUR_mg/L             1054 non-null   object 
 7   COLI_FEC_NMP/100_mL  1054 non-null   object 
 8   N_NO3_mg/L           1054 non-null   object 
 9   AS_TOT_mg/L          1054 non-null   object 
 10  CD_TOT_mg/L          1054 non-null   object 
 11  CR_TOT_mg/L          1054 non-null   object 
 12  HG_TOT_mg/L          1054 non-null   object 
 13  PB_TOT_mg/L          1054 non-null   object 
 14  MN_TOT_mg/L          1054 non-null   object 
 15  FE_TOT_mg/L          1054 non-null   object 
 16  SEMAFORO             1054 non-null   object 
dtypes: float64(4), object(13)
memory usage: 148.2+ KB
```

Se aprecia como en la mayor parte de las columnas son de tipo string (object) aunque son numericas, y esto se debe a que incluyen el simbolo <.

```
# Conversion de tipo de dato de la columna y eliminacion del <
col = ['ALC_mg/L','CONDUCT_mS/cm','SDT_M_mg/L','FLUORUROS_mg/L','DUR_mg/L','COLI_FEC_NMP/100_
       'AS_TOT_mg/L','CD_TOT_mg/L','CR_TOT_mg/L','HG_TOT_mg/L','PB_TOT_mg/L','MN_TOT_mg/L','F
for name in col:
  df_new[name] = df_new[name].astype(str)
  df_new[name] = df_new[name].replace("<","", regex=True)
  df_new[name]= df_new[name].astype(float)

df_new.head()
```

```
/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_
  """
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: 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_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: 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_
  import sys
```

	LONGITUD	LATITUD	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	C
0	-102.02210	22.20887	229.990	940.0	603.6	0.9766	213.7320	
1	-102.20075	21.99958	231.990	608.0	445.4	0.9298	185.0514	
2	-102.28801	22.36685	204.920	532.0	342.0	1.8045	120.7190	
3	-102.29449	22.18435	327.000	686.0	478.6	1.1229	199.8790	
4	-110.24480	23.45138	309.885	1841.0	1179.0	0.2343	476.9872	

Se prosigue dividiendo este conjunto de datos en tres partes dependiendo de su categoria.

```
df_location = df_new[["LONGITUD","LATITUD"]] # Localizacion
```

```
df_sust = df_new.drop(["LONGITUD","LATITUD","SEMAFORO"], axis=1) # Sustancias contaminantes
y = pd.DataFrame(df_new["SEMAFORO"])# Semaforo
y
```

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

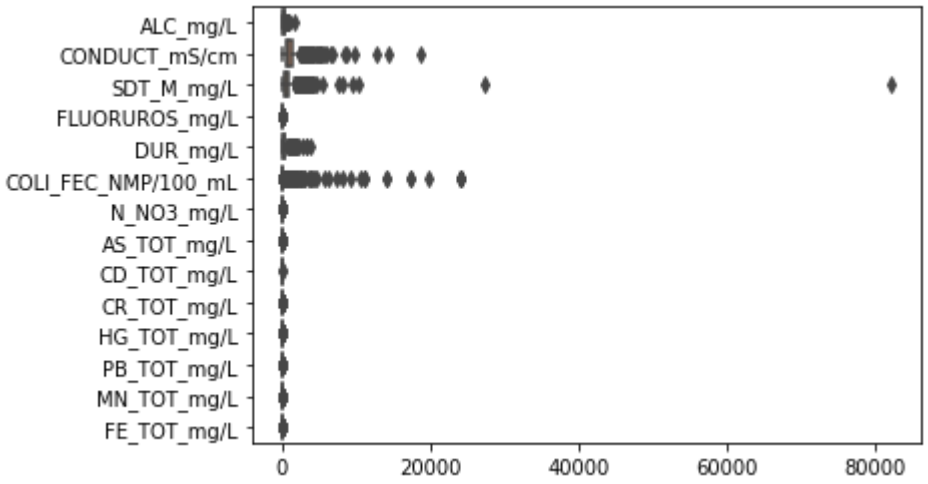
1054 rows × 1 columns

Exploracion de los datos

```
# Se exploran los datos estadisticos de las columnas de las sustancias contaminantes
df_sust.describe()
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	COLI_FEC_NI
count	1054.000000	1054.000000	1054.000000	1054.000000	1054.000000	10

```
# Grafico de cajas
sns.boxplot(data = df_sust, orient="h")
plt.show()
```




```
# Se visualiza la matriz de correlacion
corrs = df_sust.corr()
sns.set(rc = {'figure.figsize':(15,10)})
sns.heatmap(corrs, vmin = -1, vmax = 1, cmap = "BuGn", annot= True, fmt=".3f")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8761d1f1d0>
```


```
# Exploracion de la varianza
print('Varianza correspondiente a cada columna:')
print(df_sust.var())
print('\nTotal de varianza: ',sum(df_sust.var()))
print('\nProporcion de varianza de cada columna:')
print(np.round(df_sust.var()/sum(df_sust.var()),3)*100)
```

```
Varianza correspondiente a cada columna:
ALC_mg/L               1.235384e+04
CONDUCT_mS/cm          1.559978e+06
SDT_M_mg/L             7.649417e+06
FLUORUROS_mg/L         3.729549e+00
DUR_mg/L               1.302922e+05
COLI_FEC_NMP/100_mL    4.267140e+06
N_NO3_mg/L             7.019645e+01
AS_TOT_mg/L            1.228557e-03
CD_TOT_mg/L            8.102546e-07
CR_TOT_mg/L            2.415279e-02
HG_TOT_mg/L            2.206552e-07
PB_TOT_mg/L            1.073068e-05
MN_TOT_mg/L            1.435322e-01
FE_TOT_mg/L            3.107290e+01
dtype: float64

Total de varianza:  13619286.03779883

Proporcion de varianza de cada columna:
ALC_mg/L                0.1
CONDUCT_mS/cm          11.5
SDT_M_mg/L             56.2
FLUORUROS_mg/L          0.0
DUR_mg/L                1.0
COLI_FEC_NMP/100_mL    31.3
N_NO3_mg/L              0.0
AS_TOT_mg/L             0.0
CD_TOT_mg/L             0.0
CR_TOT_mg/L             0.0
HG_TOT_mg/L             0.0
PB_TOT_mg/L             0.0
```

```
MN_TOT_mg/L             0.0
FE_TOT_mg/L             0.0
dtype: float64
```

Segun su estadistica descriptiva y varianza, se aprecia como los valores entre columnas tienen diferentes escalas numericas, lo cual produce un alto indice de varianza. No nos podemos fiar de estos valores debido a las diferentes magnitudes que se presentan.

```
# Escalamiento de los datos
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df_sust)
scaled_df_sust = pd.DataFrame(scaled, columns=df_sust.columns)
scaled_df_sust.head()
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	COLI_FEC_NMP/100_mL	I
0	0.125265	0.044945	0.006122	0.022443	0.051107	0.000000	
1	0.126497	0.026967	0.004194	0.021090	0.043541	0.000000	
2	0.109822	0.022852	0.002934	0.046368	0.026570	0.000000	
3	0.185024	0.031191	0.004599	0.026671	0.047453	0.000000	
4	0.174481	0.093735	0.013133	0.000991	0.120555	0.011982	

```
# Se explora nuevamente la varianza con los datos transformados
print('Varianza correspondiente a cada columna:')
print(scaled_df_sust.var())
print('\nTotal de varianza: ',sum(scaled_df_sust.var()))
print('\nProporcion de varianza de cada columna:')
print(np.round(scaled_df_sust.var()/sum(scaled_df_sust.var()),3)*100)
```

```
Varianza correspondiente a cada columna:
ALC_mg/L               0.004688
CONDUCT_mS/cm          0.004574
SDT_M_mg/L             0.001136
FLUORUROS_mg/L         0.003115
DUR_mg/L               0.009067
COLI_FEC_NMP/100_mL    0.007289
N_NO3_mg/L             0.004795
AS_TOT_mg/L            0.006283
CD_TOT_mg/L            0.000956
CR_TOT_mg/L            0.000967
HG_TOT_mg/L            0.001184
PB_TOT_mg/L            0.001863
MN_TOT_mg/L            0.001780
```

```
FE_TOT_mg/L            0.000974
dtype: float64

Total de varianza:  0.04867157843563119

Proporcion de varianza de cada columna:
ALC_mg/L                9.6
CONDUCT_mS/cm           9.4
SDT_M_mg/L              2.3
FLUORUROS_mg/L          6.4
DUR_mg/L               18.6
COLI_FEC_NMP/100_mL    15.0
N_NO3_mg/L              9.9
AS_TOT_mg/L            12.9
CD_TOT_mg/L             2.0
CR_TOT_mg/L             2.0
HG_TOT_mg/L             2.4
PB_TOT_mg/L             3.8
MN_TOT_mg/L             3.7
FE_TOT_mg/L             2.0
dtype: float64
```

```
# Grafico de cajas
sns.boxplot(data = scaled_df_sust, orient="h")
plt.show()
```

Aun transformando los datos, en el grafico de cajas puede notar como la distribucion de los lados en cada columna no es uniforme y presentad una cantidad grande de outliers.

Relacion entre la calidad del agua y su ubicacion geografica utilizando Kmeans

```
# Graficamos los cuerpos de agua segun los puntos de sus coordenadas

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

```
WARNING:matplotlib.axes._axes:*c* argument looks like a single numeric RGB or RGBA seque
<matplotlib.axes._subplots.AxesSubplot at 0x7f875da183d0>
```

```
#Generamos dataframe con los datos de las coordendas
df_location
df_location["COORDENADAS"] = list(zip(df_location.LONGITUD, df_location.LATITUD))
df_location["COORDENADAS"] = df_location["COORDENADAS"].apply(Point)
```

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

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

```
df_location.head()
```

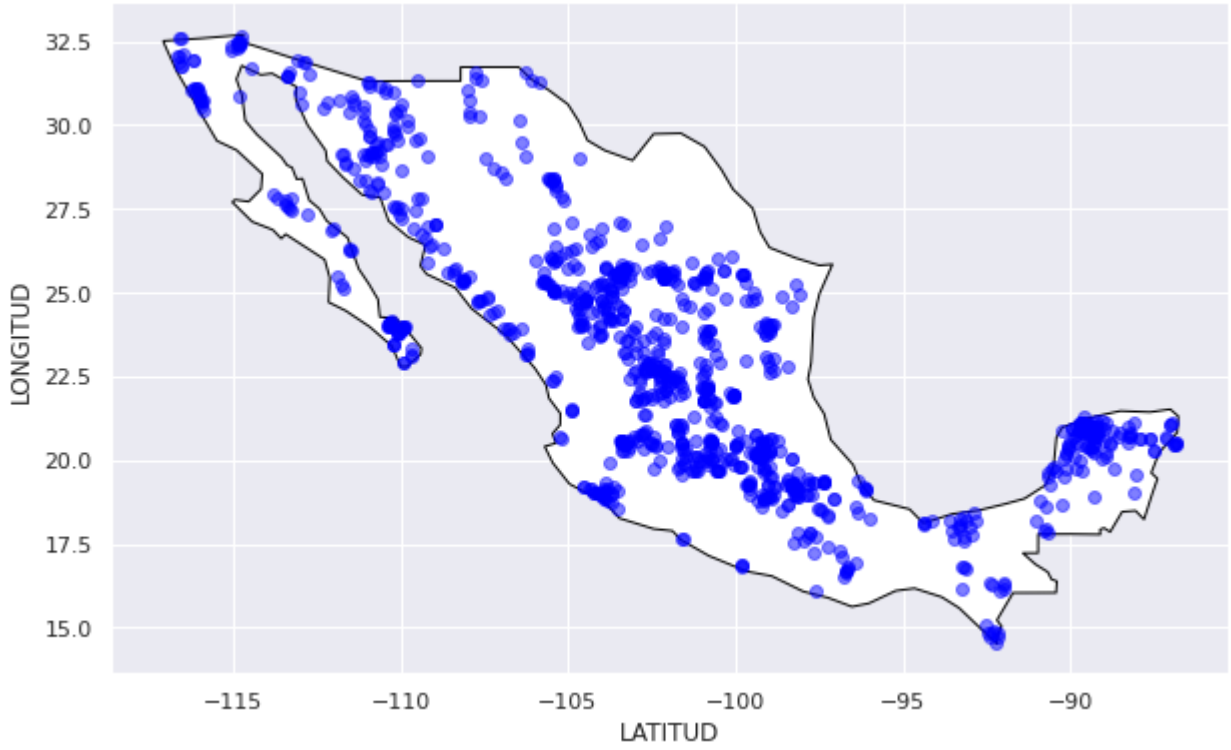
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.0221 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.2448 23.45138)

```
#Creamos Geodataframe

Mapa_Geo_Mex = gpd.GeoDataFrame(df_location, 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)

Mapa_Geo_Mex.plot(ax=gax, color='Blue', alpha = 0.5)
Mapa_Geo_Mex
```

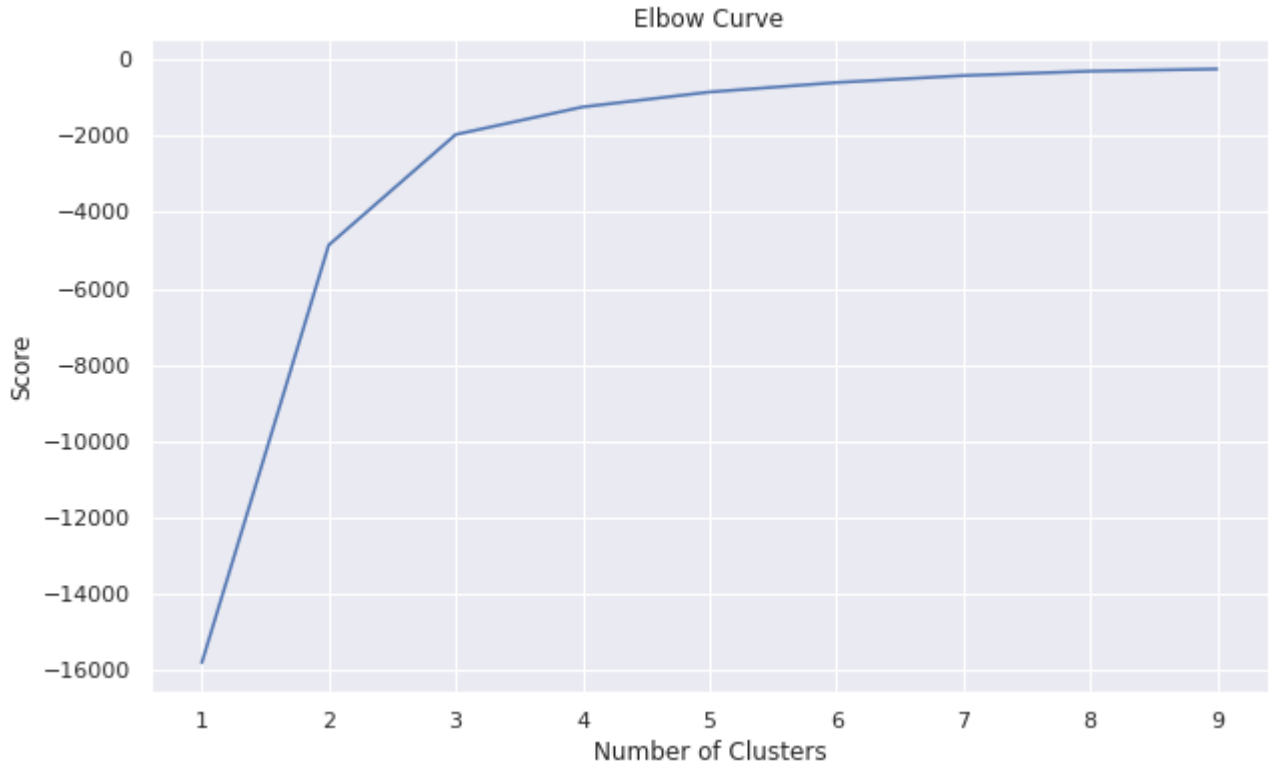
	LONGITUD	LATITUD	COORDENADAS
0	-102.02210	22.20887	POINT (-102.02210 22.20887)
1	-102.20075	21.99958	POINT (-102.20075 21.99958)
2	-102.28801	22.36685	POINT (-102.28801 22.36685)
3	-102.29449	22.18435	POINT (-102.29449 22.18435)
4	-110.24480	23.45138	POINT (-110.24480 23.45138)
...	...	...	...
1063	-99.54191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

1054 rows × 3 columns




```
#Aplicamos Kmeans
```

```
Cluster_num = range(1,10)
mi_kmeans = [KMeans(n_clusters=i) for i in Cluster_num]
Y_axis = df_location[['LATITUD']]
X_axis = df_location[['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(Cluster_num, calulo_kmeans)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')

plt.show()
```




```
# De acuaerdo a la grafica de codo, el mayor cambio en el score se lo llevan los tres primero
# y a partir de estos se podria considerar que los demas son constantes.

X = df[['LONGITUD', 'LATITUD']]

kmeans = KMeans(n_clusters=3).fit(X)
centroids = kmeans.cluster_centers_ # Obtencion de centroides
labels = kmeans.predict(X) # prediccion de color de semaforo segun el centroide mas cercano
# Obtencion de los centroides
C = kmeans.cluster_centers_

df_centroids = pd.DataFrame(C) # Conversion a DataFrame
df_centroids["Coordinates"] = list(zip(df_centroids[0], df_centroids[1])) # Se convierte a li
df_centroids["Coordinates"] = df_centroids["Coordinates"].apply(Point) # Se convierte a punto
```

```
centroids_plot = gpd.GeoDataFrame(df_centroids, geometry="Coordinates")
centroids_plot
```

	0	1	Coordinates
0	-110.740896	28.420375	POINT (-110.74090 28.42038)
1	-101.715581	22.271624	POINT (-101.71558 22.27162)
2	-90.698434	19.475165	POINT (-90.69843 19.47516)

```
# Conteo de cuerpos de agua de acuerdo a su respectivo color de semaforo
df['SEMAFORO'].value_counts()
```

```
Verde       427
Rojo        382
Amarillo    245
Name: SEMAFORO, dtype: int64
```

```
y
```

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

1054 rows × 1 columns

```
# Reemplazo del los nombres del color para que el codigo los respete
y['SEMAFORO_plot'] = y['SEMAFORO'].replace(to_replace = "Verde", value = "green")
y['SEMAFORO_plot'].replace(to_replace = "Rojo", value = "red", inplace=True)
y['SEMAFORO_plot'].replace(to_replace = "Amarillo", value = "yellow", inplace=True)
y
```

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

1054 rows × 2 columns

```
Mapa_Geo_Mex['Coordenada'] = Mapa_Geo_Mex['LATITUD'] + Mapa_Geo_Mex['LONGITUD']

semaforo_plot = dict(zip(Mapa_Geo_Mex.Coordenada, y['SEMAFORO_plot']))


# Para graficar el mapa
lat = Mapa_Geo_Mex.iloc[0]['LATITUD']
lng = Mapa_Geo_Mex.iloc[0]['LONGITUD']
map = folium.Map(location=[lng, lat], zoom_start=1)

for _, row in Mapa_Geo_Mex.iterrows():
    folium.CircleMarker(
        location=[row["LATITUD"], row["LONGITUD"]],
        radius=5,
        weight=1,
        fill=True,
        fill_color=semaforo_plot[row["Coordenada"]],
        color=semaforo_plot[row["Coordenada"]]
    ).add_to(map)
color='black'

for _, row in Mapa_Geo_Mex.iterrows():
    folium.CircleMarker(
        location=[row[1], row[0]],
        radius=5,
        weight=1,
        fill=True,
```

```
            fill_color=color,
            color=color
        ).add_to(map)
    map
```

```
#Graficacion de los puntos respecto a su centroide mas cercano

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

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

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

Mapa_Geo_Mex.plot(ax=gax, color=color_asig, alpha = 0.5)
centroids_plot.plot(ax=gax, color='blue', 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()
```


```
Mapa_Geo_Mex['SEMAFORO']= y['SEMAFORO']
Mapa_Geo_Mex['CLUSTER'] = labels
Mapa_Geo_Mex
```

	LONGITUD	LATITUD	COORDENADAS	Coordenada	SEMAFORO	CLUSTER
0	-102.02210	22.20887	POINT (-102.02210 22.20887)	-79.81323	Verde	1
1	-102.20075	21.99958	POINT (-102.20075 21.99958)	-80.20117	Verde	1
2	-102.28801	22.36685	POINT (-102.28801 22.36685)	-79.92116	Rojo	1
3	-102.29449	22.18435	POINT (-102.29449 22.18435)	-80.11014	Verde	1
4	-110.24480	23.45138	POINT (-110.24480 23.45138)	-86.79342	Rojo	0
...	...	...	...	...	...	...
1063	-99.54191	24.76036	POINT (-99.54191 24.76036)	-74.78155	Rojo	1
1064	-99.70099	24.78280	POINT (-99.70099 24.78280)	-74.91819	Rojo	1
1065	-99.82249	25.55197	POINT (-99.82249 25.55197)	-74.27052	Rojo	1
1066	-100.32683	24.80118	POINT (-100.32683 24.80118)	-75.52565	Verde	1
1067	-100.73302	25.09380	POINT (-100.73302 25.09380)	-75.63922	Verde	1

1054 rows × 6 columns

Resultados de agrupamiento de latitudes y longitudes con Kmeans en el mapa de Mexico

```
labels_semaforo = Mapa_Geo_Mex['SEMAFORO_Plot'].tolist()
Mapa_Geo_Mex['SEMAFORO_Plot']
```

```
0       green
1       green
2         red
3       green
4         red
        ...
1063      red
1064      red
1065      red
1066    green
1067    green
Name: SEMAFORO_Plot, Length: 1054, dtype: object
```

```
fig, gax = plt.subplots(figsize=(15,10))

colores = ['green','yellow','red']

color_asig = []

for j in range(0,1054):
  color_asig.append(labels_semaforo[j])

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

Mapa_Geo_Mex.plot(ax=gax, color=color_asig, alpha = 0.5, legend=True)

plt.scatter(centroids[:, 0], centroids[:, 1], c='black', s=250, alpha=1)

list_names_regions = ["Cluster0", "Cluster1", "Cluster2",  "Cluster3"]
list_stores_cluster = pd.DataFrame(labels).value_counts().to_list()
for i, txt in enumerate(list_stores_cluster):
    plt.annotate(str(txt)+ ", " + list_names_regions[i], (centroids[i,0], centroids[i,1]), xy

gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.set_title('Calidad del agua en México')

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

plt.show()
```


```
Mapa_Geo_Mex["Cluster"] = labels
clusters_dict = {}
for i in range(3):
  if "Cluster"+str(i) not in clusters_dict:
      clusters_dict["Cluster"+str(i)] = []
  clusters_dict["Cluster"+str(i)] = Mapa_Geo_Mex[Mapa_Geo_Mex["Cluster"] == i].groupby(by="SE
```

```
Cluster_Semaforo = pd.DataFrame(clusters_dict)
Cluster_Semaforo.transpose().plot.bar(color={"Amarillo": "yellow", "Rojo": "red", "Verde": "g
```

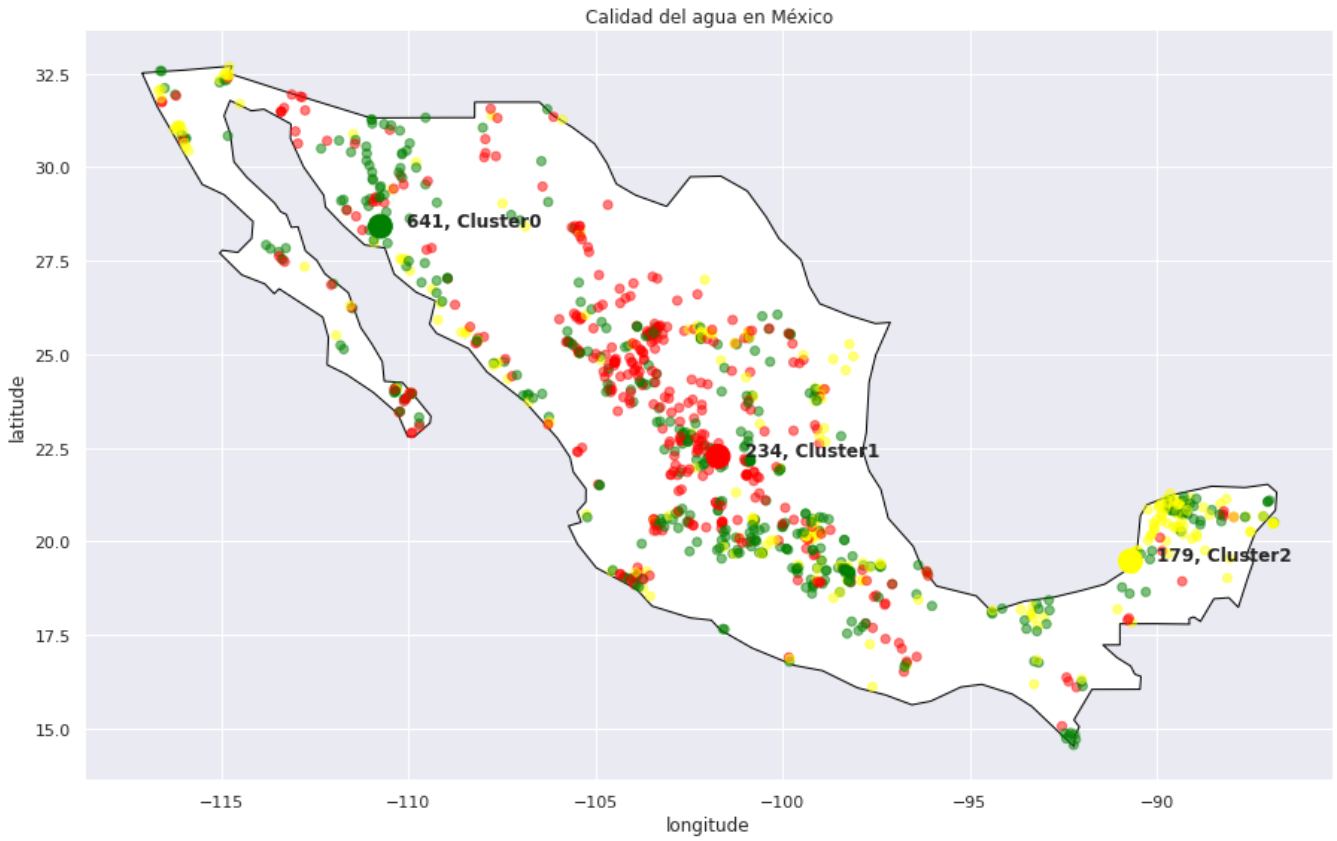
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f875a9f2ed0>
```

```
mode_list=[]

for i in range(0,3):
  df_cluster = pd.DataFrame()
  df_cluster = Mapa_Geo_Mex[Mapa_Geo_Mex.CLUSTER == i].copy()
  moda = df_cluster['SEMAFORO'].mode()[0]
  mode_list.append(moda)

len(mode_list)


centroids_plot['MODA'] = mode_list
centroids_plot
```

	0	1	Coordinates	MODA
0	-110.740896	28.420375	POINT (-110.74090 28.42038)	Verde
1	-101.715581	22.271624	POINT (-101.71558 22.27162)	Rojo
2	-90.698434	19.475165	POINT (-90.69843 19.47516)	Amarillo


```
fig, gax = plt.subplots(figsize=(15,10))

colores = ['green','yellow','red']

color_asig = []

for j in range(0,1054):
  color_asig.append(labels_semaforo[j])

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

Mapa_Geo_Mex.plot(ax=gax, color=color_asig, alpha = 0.5, legend=True)

plt.scatter(centroids[:, 0], centroids[:, 1], c=["green", "red",  "yellow"], s=250, alpha=1)

list_names_regions = ["Cluster0", "Cluster1", "Cluster2",  "Cluster3"]
list_stores_cluster = pd.DataFrame(labels).value_counts().to_list()
for i, txt in enumerate(list_stores_cluster):
    plt.annotate(str(txt)+ ", " + list_names_regions[i], (centroids[i,0], centroids[i,1]), xy

gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.set_title('Calidad del agua en México')
```

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

plt.show()
```




```
Mapa_Geo_Mex.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
Int64Index: 1054 entries, 0 to 1067
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   LONGITUD     1054 non-null   float64 
 1   LATITUD      1054 non-null   float64 
 2   COORDENADAS  1054 non-null   geometry
 3   Coordenada   1054 non-null   float64 
 4   SEMAFORO     1054 non-null   object  
 5   CLUSTER      1054 non-null   int32   
 6   Cluster      1054 non-null   int32   
```

```
dtypes: float64(3), geometry(1), int32(2), object(1)
memory usage: 57.6+ KB
```