



###Maestría en Inteligencia Artificial Aplicada

###Curso: Ciencia y Analítica de Datos

###Profesora: Dra. María de la Paz Rico Fernández

###Alumno: Francisco Javier Ramírez Arias

###Matrícula: A01316379

###Alumno: Jesús Angel Rincón Ruiz

###Matrícula: A01793960

###Objetivo:

###Implementar conocimientos adquiridos a lo largo de curso en el desarrollo de un proyecto con datos reales.

*#Carga de Librerías*

```
import numpy as np
import matplotlib as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import matplotlib
```

```
from sklearn import metrics
from sklearn.metrics import r2_score
```

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

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

```
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)
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-
packages (0.90)
```

```
Requirement already satisfied: gensim in
/usr/local/lib/python3.7/dist-packages (3.6.0)
```

```
Requirement already satisfied: folium in
```

```
/usr/local/lib/python3.7/dist-packages (0.12.1.post1)
Collecting pyLDavis
  Downloading pyLDavis-3.3.1.tar.gz (1.7 MB)
  Preparing metadata (from pyproject.toml) ... done
Requirement already satisfied: etadatas in /usr/local/lib/python3.7/dist-packages (1.1.0)
Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-packages (1.1.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from qeds) (1.3.5)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from qeds) (2.23.0)
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 (from qeds) (1.7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from qeds) (1.21.6)
Collecting quantecon
  Downloading quantecon-0.5.3-py3-none-any.whl (179 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from qeds) (3.2.2)
Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packages (from qeds) (6.0.1)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packages (from qeds) (3.0.10)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from qeds) (5.5.0)
Requirement already satisfied: pandas_datareader in /usr/local/lib/python3.7/dist-packages (from qeds) (0.9.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from qeds) (1.0.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (from qeds) (0.11.2)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (from qeds) (0.12.2)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (from fiona) (22.1.0)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (from fiona) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from fiona) (2022.9.24)
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (from fiona) (7.1.2)
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-packages (from fiona) (57.4.0)
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Collecting pyproj>=2.2.0
```

Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010\_x86\_64.whl (6.3 MB)  
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages (from geopandas) (1.8.5.post1)  
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->qeds) (2.8.2)  
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->qeds) (2022.6)  
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-packages (from gensim) (5.2.1)  
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packages (from folium) (2.11.3)  
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-packages (from folium) (0.6.0)  
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2>=2.9->folium) (2.0.1)  
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from pyLDAvis) (1.2.0)  
Collecting funcy  
 Downloading funcy-1.17-py2.py3-none-any.whl (33 kB)  
Collecting sklearn  
 Downloading sklearn-0.0.post1.tar.gz (3.6 kB)  
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (from pyLDAvis) (2.8.4)  
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from pyLDAvis) (0.16.0)  
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->qeds) (0.11.0)  
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->qeds) (3.0.9)  
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->qeds) (1.4.4)  
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib->qeds) (4.1.1)  
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-packages (from openpyxl->qeds) (1.1.0)  
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (from pandas\_datareader->qeds) (4.9.1)  
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->qeds) (2.10)  
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->qeds) (3.0.4)  
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->qeds) (1.24.3)  
Requirement already satisfied: tenacity>=6.2.0 in

```

/usr/local/lib/python3.7/dist-packages (from plotly->qeds) (8.1.0)
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-packages (from quandl->qeds) (9.0.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-
packages (from quantecon->qeds) (1.7.1)
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-
packages (from quantecon->qeds) (0.56.4)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in
/usr/local/lib/python3.7/dist-packages (from numba->quantecon->qeds)
(0.39.1)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from numba->quantecon->qeds)
(4.13.0)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata-
>numba->quantecon->qeds) (3.10.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn->qeds)
(3.1.0)
Requirement already satisfied: patsy>=0.5 in
/usr/local/lib/python3.7/dist-packages (from statsmodels->qeds)
(0.5.3)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.7/dist-packages (from sympy->quantecon->qeds)
(1.2.1)
Building wheels for collected packages: qeds, pyLDAvis, sklearn
  Building wheel for qeds (setup.py) ... e=qeds-0.7.0-py3-none-any.whl
size=27812
sha256=d3955ddd5ae4ab3e20e026278957bb1b6edd628e642193784752c455dc60f17
a
    Stored in directory:
/root/.cache/pip/wheels/fc/8c/52/0cc036b9730b75850b9845770780f8d05ed08
ff38a67cbaa29
  Building wheel for pyLDAvis (PEP 517) ... e=pyLDAvis-3.3.1-py2.py3-
none-any.whl size=136898
sha256=1f88ec95f52aeef4ac55fb1839a44336ba8ac63f1894dc90800e6d859790914
8
    Stored in directory:
/root/.cache/pip/wheels/c9/21/f6/17bcf2667e8a68532ba2fbf6d5c72fdf4c7f7
d9abfa4852d2f
  Building wheel for sklearn (setup.py) ... e=sklearn-0.0.post1-py3-
none-any.whl size=2344
sha256=6ald7c65e48add8feab83561ae6ca6dcde2ad3563a7b0425b94af592e890fd6
b
    Stored in directory:
/root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b310477667c1fca5c5
1c3ae4124a003
Successfully built qeds pyLDAvis sklearn

```

Installing collected packages: munch, inflection, cligj, click-plugins, sklearn, quantecon, quandl, pyproj, fancy, fiona, qeds, pyLDAvis, geopandas  
 Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.22 fancy-1.17 geopandas-0.10.2 inflection-0.5.1 munch-2.5.0 pyLDAvis-3.3.1 pyproj-3.2.1 qeds-0.7.0 quandl-3.7.0 quantecon-0.5.3 sklearn-0.0.post1

#Parte 1

*#Lectura de la Base de Datos de Aguas Subterraneas*

```
df =
pd.read_csv('/content/sample_data/AguasSubterraneas_2020.csv',encoding
='cp1252')
df.sample(10)
```

	CLAVE	SITIO \
273	DLEST6147	SITIO No. 11 NORIA IMSS
289	DLGUA1036	RODEO DE AYALA
882	OCNOR6506	URES.
878	OCNOR6264	LA MORA
821	OCNOR4036	EL REAL DEL CATORCE
192	DLDUR654	POZO PRESIDIOS
557	DLZAC2613	POZO 524 AGUA POTABLE RANCHO NUEVO (SUSTITUTO ...
870	OCNOR4224M1	POZO QUITOVAC
209	DLDUR691	POZO 3 PEÑON BLANCO
498	DLSAN5384	EL HUIZACHE CNA-11-248

	ORGANISMO_DE_CUENCA	ESTADO \
273	LERMA SANTIAGO PACIFICO	MEXICO
289	LERMA SANTIAGO PACIFICO	GUANAJUATO
882	NOROESTE	SONORA
878	NOROESTE	SONORA
821	NOROESTE	SONORA
192	CUENCAS CENTRALES DEL NORTE	DURANGO
557	CUENCAS CENTRALES DEL NORTE	ZACATECAS
870	NOROESTE	SONORA
209	CUENCAS CENTRALES DEL NORTE	DURANGO
498	CUENCAS CENTRALES DEL NORTE	SAN LUIS POTOSI

	MUNICIPIO	ACUIFERO	SUBTIPO \
273	ZINACANTEPEC	VALLE DE TOLUCA	NORIA
289	PENJAMO	PENJAMO-ABASOLO	POZO
882	URES	RIO SONORA	POZO
878	BANAMICHI	RIO SONORA	POZO
821	HERMOSILLO	COSTA DE HERMOSILLO	POZO
192	TEPEHUANES	TEPEHUANES-SANTIAGO	POZO
557	ZACATECAS	BENITO JUAREZ	POZO
870	GENERAL PLUTARCO ELIAS CALLES	SONOYTA-PUERTO PEÑASCO	POZO
209	PEÑON BLANCO	PEÑON BLANCO	POZO
498	SOLEDAD DE GRACIANO SANCHEZ	SAN LUIS POTOSI	POZO

	LONGITUD	LATITUD	PERIODO	...	CUMPLE_CON_DUR	CUMPLE_CON_CF
\ 273	-99.733256	19.290975	2020	...	SI	SI
289	-101.625050	20.485330	2020	...	SI	SI
882	-110.382940	29.424050	2020	...	SI	SI
878	-110.204800	29.977650	2020	...	SI	SI
821	-111.055560	28.961570	2020	...	SI	SI
192	-105.628800	25.282450	2020	...	SI	SI
557	-102.736960	22.754910	2020	...	SI	SI
870	-112.750556	31.515833	2020	...	SI	SI
209	-104.027790	24.790060	2020	...	SI	SI
498	-100.853056	22.236667	2020	...	SI	SI

	CUMPLE_CON_N03	CUMPLE_CON_AS	CUMPLE_CON_CD	CUMPLE_CON_CR
CUMPLE_CON_HG \ 273	SI	SI	SI	SI
SI				
289	SI	NO	SI	SI
SI				
882	SI	SI	SI	SI
SI				
878	SI	SI	SI	SI
SI				
821	SI	SI	SI	SI
SI				
192	SI	NO	SI	SI
SI				
557	SI	NO	SI	SI
SI				
870	SI	NO	SI	SI
SI				
209	SI	NO	SI	SI
SI				
498	SI	SI	SI	SI
SI				

	CUMPLE_CON_PB	CUMPLE_CON_MN	CUMPLE_CON_FE
273	SI	SI	SI

289	SI	SI	SI
882	SI	SI	SI
878	SI	SI	SI
821	SI	SI	SI
192	SI	SI	SI
557	SI	SI	SI
870	SI	SI	SI
209	SI	SI	SI
498	SI	SI	SI

[10 rows x 57 columns]

*#¿Que tipo de dato son las variables del conjunto de Datos?*  
df.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_salín	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_N03_mg/L	object
CALIDAD_N_N03	object
AS_TOT_mg/L	object
CALIDAD_AS	object
CD_TOT_mg/L	object
CALIDAD_CD	object
CR_TOT_mg/L	object
CALIDAD_CR	object
HG_TOT_mg/L	object
CALIDAD_HG	object
PB_TOT_mg/L	object

CALIDAD_PB	object
MN_TOT_mg/L	object
CALIDAD_MN	object
FE_TOT_mg/L	object
CALIDAD_FE	object
SEMAFORO	object
CONTAMINANTES	object
CUMPLE_CON_ALC	object
CUMPLE_CON_COND	object
CUMPLE_CON_SDT_ra	object
CUMPLE_CON_SDT_salín	object
CUMPLE_CON_FLUO	object
CUMPLE_CON_DUR	object
CUMPLE_CON_CF	object
CUMPLE_CON_N03	object
CUMPLE_CON_AS	object
CUMPLE_CON_CD	object
CUMPLE_CON_CR	object
CUMPLE_CON_HG	object
CUMPLE_CON_PB	object
CUMPLE_CON_MN	object
CUMPLE_CON_FE	object
dtype:	object

*#¿Cuántas variables de cada tipo de dato tenemos en nuestro Dataset?*

```
df.dtypes.value_counts()
```

```
object      51
float64      5
int64        1
dtype: int64
```

*#Revisemos cual es la dimensión de nuestro Dataset.*

```
df.shape
```

```
(1068, 57)
```

###Este archivo contiene la calidad del agua de **1,068 sitios subterráneos** en México; calificando su calidad con base en cada uno de los Indicadores y sus respectivas escalas.

###Se incluyen las coordenadas geográficas y datos generales de ubicación de cada sitio.

*#Confirmemos si en nuestros Dataset tenemos valores nulos/faltantes*

```
df.isnull().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
```

```
#Veamos cuantos valores nulos tenemos por variable
```

```
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_salín  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

```

*#En total tendríamos 1532 datos nulos.*

```
df.isnull().sum().sum()
```

```
1532
```

*#Mostramos las columnas para posteriormente revisar los diferentes tipos de valores que se presentan en algunas de ellas.*

```
df.columns
```

```

Index(['CLAVE', 'SITIO', 'ORGANISMO_DE_CUENCA', 'ESTADO', 'MUNICIPIO',
      'ACUIFERO', 'SUBTIPO', 'LONGITUD', 'LATITUD', 'PERIODO',
      'ALC_mg/L',
      'CALIDAD_ALC', 'CONDUCT_mS/cm', 'CALIDAD_CONDUCT', '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["PERIODO"].value_counts()
```

```
2020      1068
Name: PERIOD0, dtype: int64
```

```
df["CONTAMINANTES"].value_counts()

FLU0,      78
DT,        65
FLU0,AS,   51
CF,        31
AS,        31
..
ALC,CONDUCT,SDT_ra,SDT_salin,DT,N03,      1
ALC,CONDUCT,SDT_ra,SDT_salin,FLU0,DT,AS,MN,FE,  1
PB,MN,FE,  1
ALC,AS,FE,  1
ALC,DT,N03,  1
Name: CONTAMINANTES, Length: 126, dtype: int64
```

###Observamos que nuestra variable **Valor de Alcalinidad Total en miligramos por litro(SDT\_mg/L)** cuenta con **1068 valores nulos**, esto implica el 100% de las muestras por lo que se eliminara de nuestro Dataset.

###La variable **Contaminantes** cuenta con 434 valores nulos sin embargo esto es debido a que estas muestras(Sitios Subterraneos) no presentan algun tipo de contaminación que se describa con esta variable, por lo que se mantiene en el Dataset.

###Las Variable **ALC\_mg/L** presentan 4 Sitios Subterraneos sin valor, por consecuencia **CALIDAD\_ALC** presenta los mismos datos nulos, la cantidad de datos nulos no es representativo para la muestra por lo que seran eliminados.

###Se presenta este mismo caso para las variables **CONDUCT\_mS/cm, CALIDAD\_CONDUCT, CONDUCT\_mS/cm,CALIDAD\_CONDUCT,SDT\_M\_mg/L CALIDAD\_SDT\_ra,CALIDAD\_SDT\_salin, DUR\_mg/L, CALIDAD\_DUR** que seran eliminadas sus muestras con valores nulos.

###La columna **"FECHA"** presenta el mismo valor para todas las muestras por lo que no sera relevante en nuestro analisis y sera eliminada.

*#Agregaremos un valor a la Variable Contaminantes a aquellos valores nulos que indican que no hay presencia de contaminantes  
#para posteriormente eliminar los valores nulos del resto del Dataset.*

```
df["CONTAMINANTES"].fillna("OK", inplace= True)
```

*#Observamos que los valores nulos de la Columna "CONTAMINANTES" fueron llenados con "OK" como indicación que no hay presencia de Contaminantes.*

```
df["CONTAMINANTES"].value_counts()
```

```
OK      434
FLU0,    78
```

```

DT, 65
FLUO,AS, 51
CF, 31
...
ALC,FLUO,AS,FE, 1
ALC,CONDUCT,SDT_ra,SDT_salín,FLUO,DT,AS,MN,FE, 1
PB,MN,FE, 1
ALC,AS,FE, 1
ALC,DT,N03, 1
Name: CONTAMINANTES, Length: 127, dtype: int64

```

```

#Eliminamos la columna "SDT_mg/L"
Clean_df= df.drop(["SDT_mg/L"],axis=1)

```

```

#Eliminamos las muestras con datos nulos.
Clean_df.dropna(inplace=True)

```

```

#Comprobamos que nuestro Dataset se encuentra libre de Datos nulos y
Variables que al momento no se consideran de relevancia para el
análisis.
Clean_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 0
CALIDAD_ALC 0
CONDUCT_mS/cm 0
CALIDAD_CONDUCT 0
SDT_M_mg/L 0
CALIDAD_SDT_ra 0
CALIDAD_SDT_salín 0
FLUORUROS_mg/L 0
CALIDAD_FLUO 0
DUR_mg/L 0
CALIDAD_DUR 0
COLI_FEC_NMP/100_mL 0
CALIDAD_COLI_FEC 0
N_N03_mg/L 0
CALIDAD_N_N03 0
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    0
CUMPLE_CON_ALC   0
CUMPLE_CON_COND  0
CUMPLE_CON_SDT_ra 0
CUMPLE_CON_SDT_salín 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

```

```
Clean_df.shape
```

```
(1054, 56)
```

###Ya con nuestro Dataset limpio continuaremos explorando

*#Visualisemos la distribución de los sitios subterráneos por su ubicación(Estado)*

```
gb = df.groupby('ESTADO').apply(len)
gb
```

```

ESTADO
AGUASCALIENTES    14
BAJA CALIFORNIA    31
BAJA CALIFORNIA SUR 49
CAMPECHE          25
CHIAPAS           21
CHIHUAHUA         35
COAHUILA DE ZARAGOZA 59
COLIMA            26
DISTRITO FEDERAL   2
DURANGO           121

```

GUANAJUATO	41
GUERRERO	5
HIDALGO	37
JALISCO	33
MEXICO	24
MICHOACAN DE OCAMPO	27
MORELOS	11
NAYARIT	8
NUEVO LEON	15
OAXACA	20
PUEBLA	23
QUERETARO ARTEAGA	6
QUINTANA ROO	15
SAN LUIS POTOSI	47
SINALOA	32
SONORA	103
TABASCO	13
TAMAULIPAS	25
TLAXCALA	24
VERACRUZ DE IGNACIO DE LA LLAVE	16
YUCATAN	85
ZACATECAS	75

dtype: int64

*#Subtipo de cuerpo de agua donde se encuentra el sitio de muestreo*

```
geb = df.groupby('SUBTIPO').apply(len)
geb
```

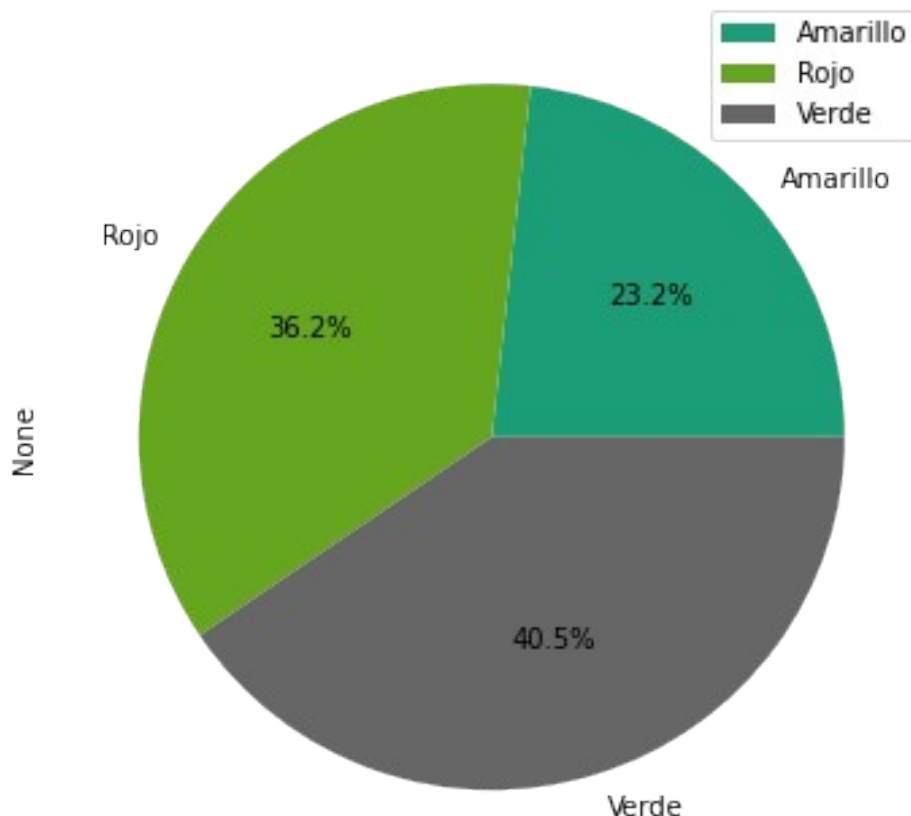
SUBTIPO	
BOMBEO CENOTE	1
CENOTE	7
DESCARGA	1
MANANTIAL	12
NORIA	3
POZO	1039
POZO NORIA	4
Pozo	1

dtype: int64

*#Veamos la proporción de nuestra variable "SEMAFORO" la cual nos indica el nivel de contaminacion de acuerdo a los contaminantes presentes*

```
gsb = Clean_df.groupby('SEMAFORO').apply(len)
gsb.plot(kind='pie', title = 'Proporción de Aguas Subterráneas de México',
cmap='Dark2', autopct="%.1f%%", figsize = (10,6), legend=True);
```

Proporción de Aguas Subterráneas de México



*#Describamos nuestros datos numericos.*

```
Clean_df.describe()
```

	LONGITUD	LATITUD	PERIODO	ALC_mg/L	CONDUCT_mS/cm
count	1054.000000	1054.000000	1054.0	1054.000000	1054.000000
mean	-101.848270	23.161796	2020.0	234.695266	1142.726471
std	6.697568	3.875005	0.0	111.147849	1248.990617
min	-116.664250	14.561150	2020.0	26.640000	110.000000
25%	-105.385170	20.224857	2020.0	164.257500	506.000000
50%	-102.170665	22.640705	2020.0	215.825000	820.000000
75%	-98.971268	25.508770	2020.0	292.930000	1328.000000
max	-86.864120	32.677713	2020.0	1650.000000	18577.000000

*#Notamos que la función describe dejó fuera algunas otras variables numericas por incluir el simbolo de desigualdad "<" como 'SDT\_M\_mg/L', 'DUR\_mg/L', etc.*

*#Cambiemos este tipo de valor a numerico.*

```
Clean_df['SDT_M_mg/L'] = Clean_df['SDT_M_mg/L'].str.replace('<', '')
Clean_df['SDT_M_mg/L'] = pd.to_numeric(Clean_df['SDT_M_mg/L'])
```



```

Clean_df['FLUORUROS_mg/L'] =
Clean_df['FLUORUROS_mg/L'].str.replace('<', '')
Clean_df['FLUORUROS_mg/L'] = pd.to_numeric(Clean_df['FLUORUROS_mg/L'])

Clean_df['DUR_mg/L'] = Clean_df['DUR_mg/L'].str.replace('<', '')
Clean_df['DUR_mg/L'] = pd.to_numeric(Clean_df['DUR_mg/L'])

Clean_df['COLI_FEC_NMP/100_mL'] =
Clean_df['COLI_FEC_NMP/100_mL'].str.replace('<', '')
Clean_df['COLI_FEC_NMP/100_mL'] =
pd.to_numeric(Clean_df['COLI_FEC_NMP/100_mL'])

Clean_df['N_NO3_mg/L'] = Clean_df['N_NO3_mg/L'].str.replace('<', '')
Clean_df['N_NO3_mg/L'] = pd.to_numeric(Clean_df['N_NO3_mg/L'])

Clean_df['AS_TOT_mg/L'] = Clean_df['AS_TOT_mg/L'].str.replace('<', '')
Clean_df['AS_TOT_mg/L'] = pd.to_numeric(Clean_df['AS_TOT_mg/L'])

Clean_df['CD_TOT_mg/L'] = Clean_df['CD_TOT_mg/L'].str.replace('<', '')
Clean_df['CD_TOT_mg/L'] = pd.to_numeric(Clean_df['CD_TOT_mg/L'])

Clean_df['CR_TOT_mg/L'] = Clean_df['CR_TOT_mg/L'].str.replace('<', '')
Clean_df['CR_TOT_mg/L'] = pd.to_numeric(Clean_df['CR_TOT_mg/L'])

Clean_df['HG_TOT_mg/L'] = Clean_df['HG_TOT_mg/L'].str.replace('<', '')
Clean_df['HG_TOT_mg/L'] = pd.to_numeric(Clean_df['HG_TOT_mg/L'])

Clean_df['PB_TOT_mg/L'] = Clean_df['PB_TOT_mg/L'].str.replace('<', '')
Clean_df['PB_TOT_mg/L'] = pd.to_numeric(Clean_df['PB_TOT_mg/L'])

Clean_df['MN_TOT_mg/L'] = Clean_df['MN_TOT_mg/L'].str.replace('<', '')
Clean_df['MN_TOT_mg/L'] = pd.to_numeric(Clean_df['MN_TOT_mg/L'])

Clean_df['FE_TOT_mg/L'] = Clean_df['FE_TOT_mg/L'].str.replace('<', '')
Clean_df['FE_TOT_mg/L'] = pd.to_numeric(Clean_df['FE_TOT_mg/L'])

```

*#Describamos nuevamente nuestros datos numericos.*

```
Clean_df.describe()
```

	LONGITUD	LATITUD	PERIODO	ALC_mg/L	
CONDUCT_mS/cm \					
count	1054.000000	1054.000000	1054.0	1054.000000	1054.000000
mean	-101.848270	23.161796	2020.0	234.695266	1142.726471
std	6.697568	3.875005	0.0	111.147849	1248.990617

min	-116.664250	14.561150	2020.0	26.640000	110.000000
25%	-105.385170	20.224857	2020.0	164.257500	506.000000
50%	-102.170665	22.640705	2020.0	215.825000	820.000000
75%	-98.971268	25.508770	2020.0	292.930000	1328.000000
max	-86.864120	32.677713	2020.0	1650.000000	18577.000000

	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	COLI_FEC_NMP/100_mL
\count	1054.000000	1054.000000	1054.000000	1054.000000
mean	896.945797	1.078547	349.893584	359.734156
std	2765.757924	1.931204	360.960153	2065.705773
min	101.200000	0.200000	20.000000	1.100000
25%	338.050000	0.269475	121.512000	1.100000
50%	551.400000	0.506950	245.994450	1.100000
75%	915.600000	1.142400	455.617200	10.750000
max	82170.000000	34.803300	3810.692200	24196.000000

	N_N03_mg/L	AS_TOT_mg/L	CD_TOT_mg/L	CR_TOT_mg/L	HG_TOT_mg/L
\count	1054.000000	1054.000000	1054.000000	1054.000000	1054.000000
mean	4.321651	0.019504	0.00303	0.013353	0.000557
std	8.378332	0.035051	0.00090	0.155412	0.000470
min	0.020000	0.010000	0.00300	0.005000	0.000500
25%	0.651667	0.010000	0.00300	0.005000	0.000500
50%	2.082916	0.010000	0.00300	0.005000	0.000500
75%	5.190385	0.010000	0.00300	0.005000	0.000500
max	121.007813	0.452200	0.03211	5.003200	0.014150

	PB_TOT_mg/L	MN_TOT_mg/L	FE_TOT_mg/L
count	1054.000000	1054.000000	1054.000000
mean	0.005285	0.072960	0.412234
std	0.003276	0.378856	5.574307
min	0.005000	0.001500	0.025000
25%	0.005000	0.001500	0.025000
50%	0.005000	0.001500	0.046900
75%	0.005000	0.009830	0.172275
max	0.080900	8.982000	178.615000

*#Dejemos fuera las variables de "LONGITUD", "LATITUD" y "PERIODO" para seguir explorando nuestros datos*

```
df_explore =
Clean_df[["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"]].copy()
```

*#Media de nuestros datos numéricos*

```
df_explore.mean()
```

ALC_mg/L	234.695266
CONDUCT_mS/cm	1142.726471
SDT_M_mg/L	896.945797
FLUORUROS_mg/L	1.078547
DUR_mg/L	349.893584
COLI_FEC_NMP/100_mL	359.734156
N_NO3_mg/L	4.321651
AS_TOT_mg/L	0.019504
CD_TOT_mg/L	0.003030
CR_TOT_mg/L	0.013353
HG_TOT_mg/L	0.000557
PB_TOT_mg/L	0.005285
MN_TOT_mg/L	0.072960
FE_TOT_mg/L	0.412234
dtype:	float64

*#Mediana de nuestros datos numéricos*

```
df_explore.median()
```

ALC_mg/L	215.825000
CONDUCT_mS/cm	820.000000
SDT_M_mg/L	551.400000
FLUORUROS_mg/L	0.506950
DUR_mg/L	245.994450
COLI_FEC_NMP/100_mL	1.100000
N_NO3_mg/L	2.082916
AS_TOT_mg/L	0.010000
CD_TOT_mg/L	0.003000
CR_TOT_mg/L	0.005000

```
HG_TOT_mg/L          0.000500
PB_TOT_mg/L          0.005000
MN_TOT_mg/L          0.001500
FE_TOT_mg/L          0.046900
dtype: float64
```

*#Valores maximos de nuestros datos númericos*  
df\_explore.max()

```
ALC_mg/L              1650.000000
CONDUCT_mS/cm         18577.000000
SDT_M_mg/L            82170.000000
FLUORUROS_mg/L        34.803300
DUR_mg/L              3810.692200
COLI_FEC_NMP/100_mL   24196.000000
N_NO3_mg/L            121.007813
AS_TOT_mg/L           0.452200
CD_TOT_mg/L           0.032110
CR_TOT_mg/L           5.003200
HG_TOT_mg/L           0.014150
PB_TOT_mg/L           0.080900
MN_TOT_mg/L           8.982000
FE_TOT_mg/L           178.615000
dtype: float64
```

*#Valores minimos de nuestros datos númericos*  
df\_explore.min()

```
ALC_mg/L              26.6400
CONDUCT_mS/cm         110.0000
SDT_M_mg/L            101.2000
FLUORUROS_mg/L        0.2000
DUR_mg/L              20.0000
COLI_FEC_NMP/100_mL   1.1000
N_NO3_mg/L            0.0200
AS_TOT_mg/L           0.0100
CD_TOT_mg/L           0.0030
CR_TOT_mg/L           0.0050
HG_TOT_mg/L           0.0005
PB_TOT_mg/L           0.0050
MN_TOT_mg/L           0.0015
FE_TOT_mg/L           0.0250
dtype: float64
```

*#Valores desviación estandar de nuestros datos númericos*  
df\_explore.std()

```
ALC_mg/L              111.147849
CONDUCT_mS/cm         1248.990617
SDT_M_mg/L            2765.757924
FLUORUROS_mg/L        1.931204
```

```

DUR_mg/L          360.960153
COLI_FEC_NMP/100_mL  2065.705773
N_NO3_mg/L        8.378332
AS_TOT_mg/L       0.035051
CD_TOT_mg/L       0.000900
CR_TOT_mg/L       0.155412
HG_TOT_mg/L       0.000470
PB_TOT_mg/L       0.003276
MN_TOT_mg/L       0.378856
FE_TOT_mg/L       5.574307
dtype: float64

```

```

#Media de nuestros datos nmericos
df_explore.var()

```

```

ALC_mg/L          1.235384e+04
CONDUCT_mS/cm     1.559978e+06
SDT_M_mg/L        7.649417e+06
FLUORURO5_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

```

```

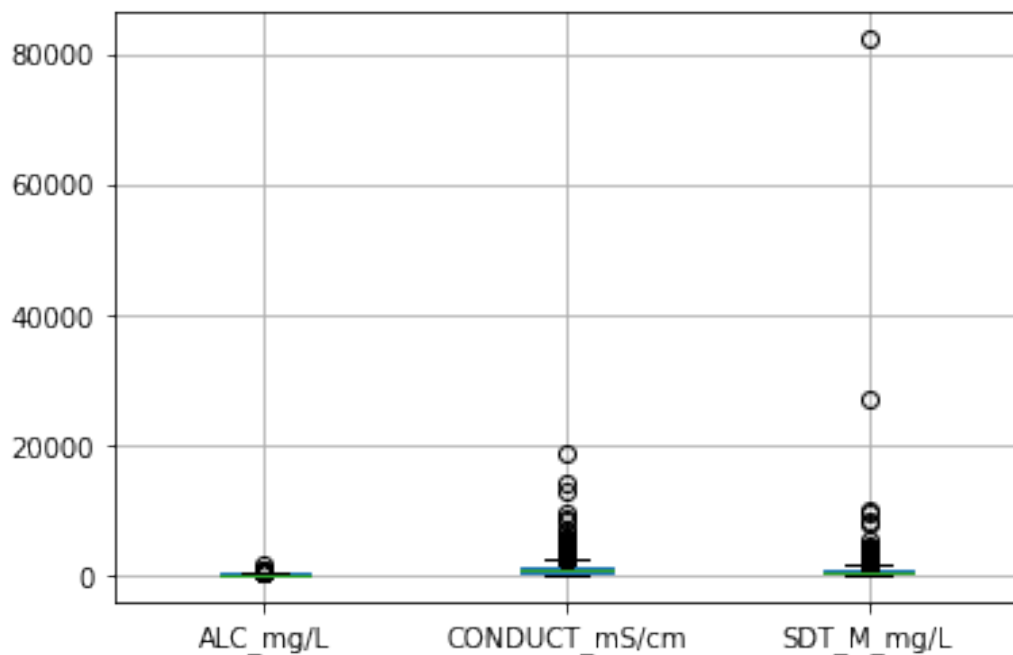
#Explemos nuestros datos a traves de Boxplot
b_plot = df_explore.boxplot(column =
["ALC_mg/L", "CONDUCT_mS/cm", "SDT_M_mg/L",])
b_plot.plot()
#plot.show()

```

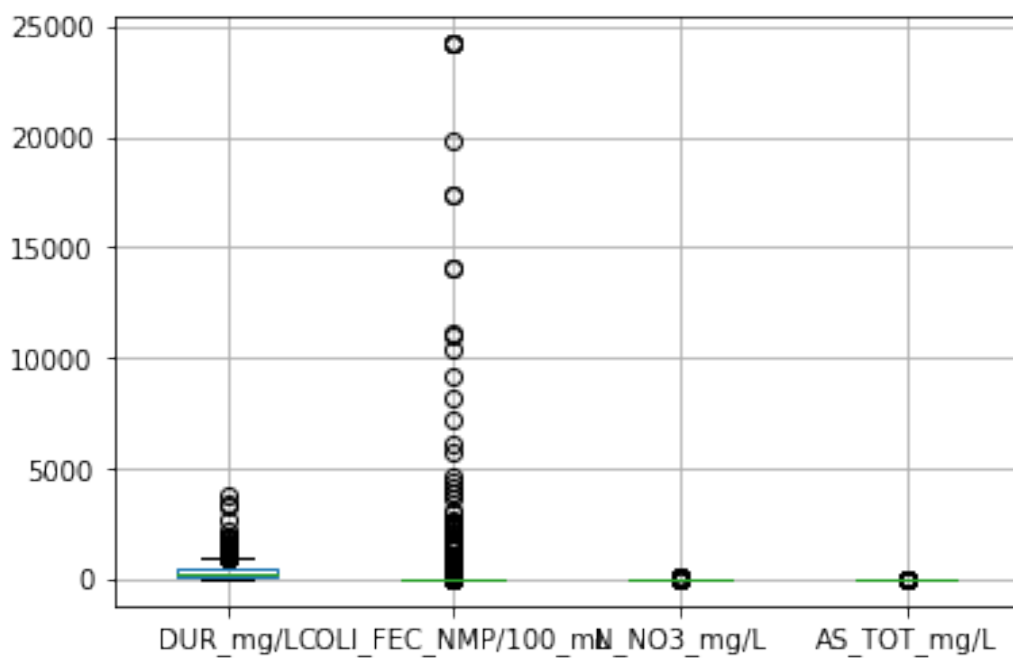
```

[]

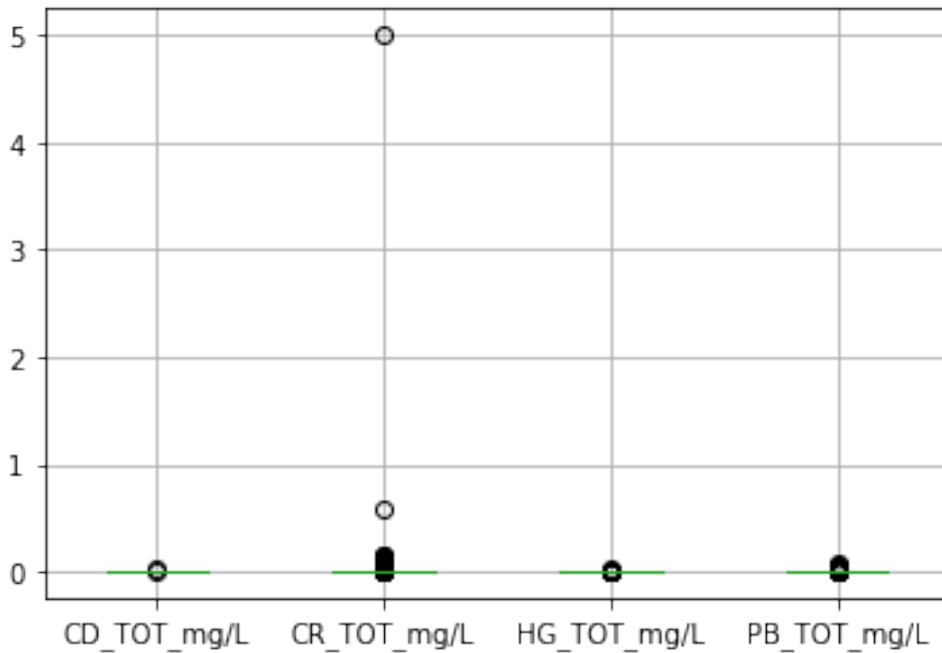
```



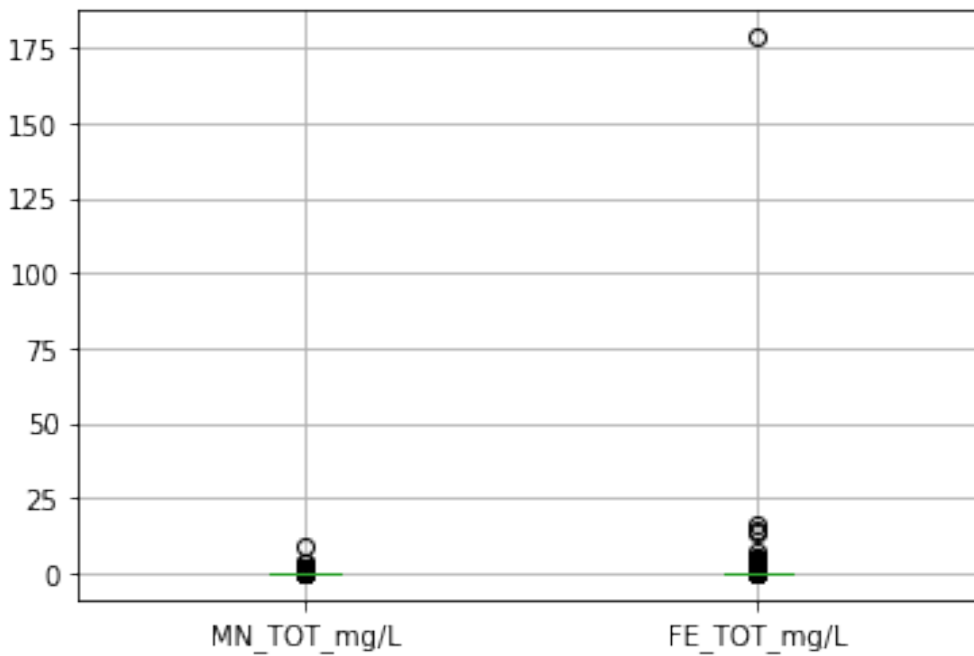
```
boxplot =
df_explore.boxplot(column=["DUR_mg/L", "COLI_FEC_NMP/100_mL", "N_NO3_mg/L", "AS_TOT_mg/L"])
```



```
boxplot =
df_explore.boxplot(column=["CD_TOT_mg/L", "CR_TOT_mg/L", "HG_TOT_mg/L", "PB_TOT_mg/L"])
```



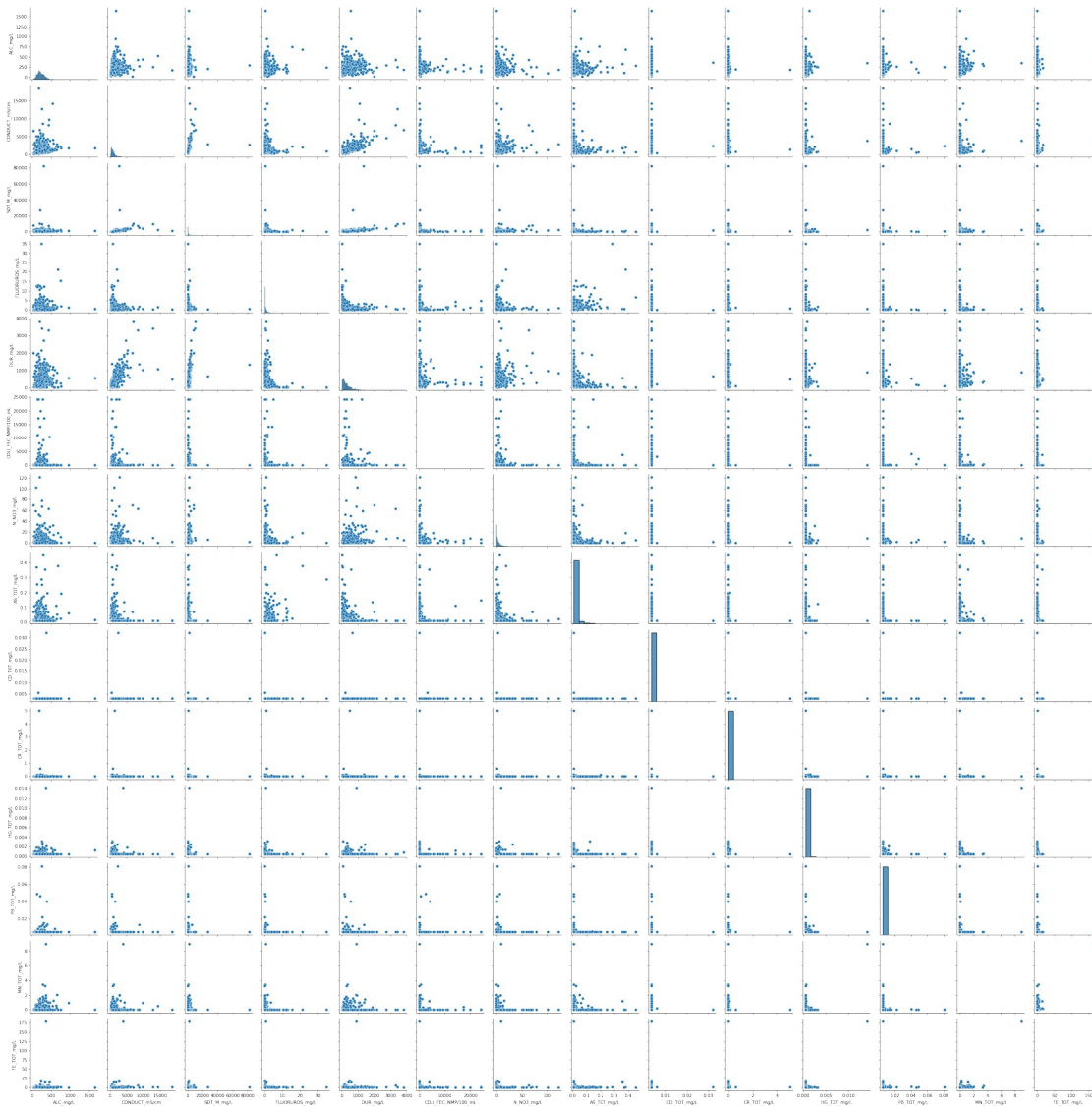
```
boxplot = df_explore.boxplot(column=["MN_TOT_mg/L", "FE_TOT_mg/L"])
```



*#Graficas de dispersión de las diferentes variables, estas  
#nos permiten identificar visualmente la correlación de las  
#variables o la dispersión.*

```
sns.pairplot(df_explore)
```

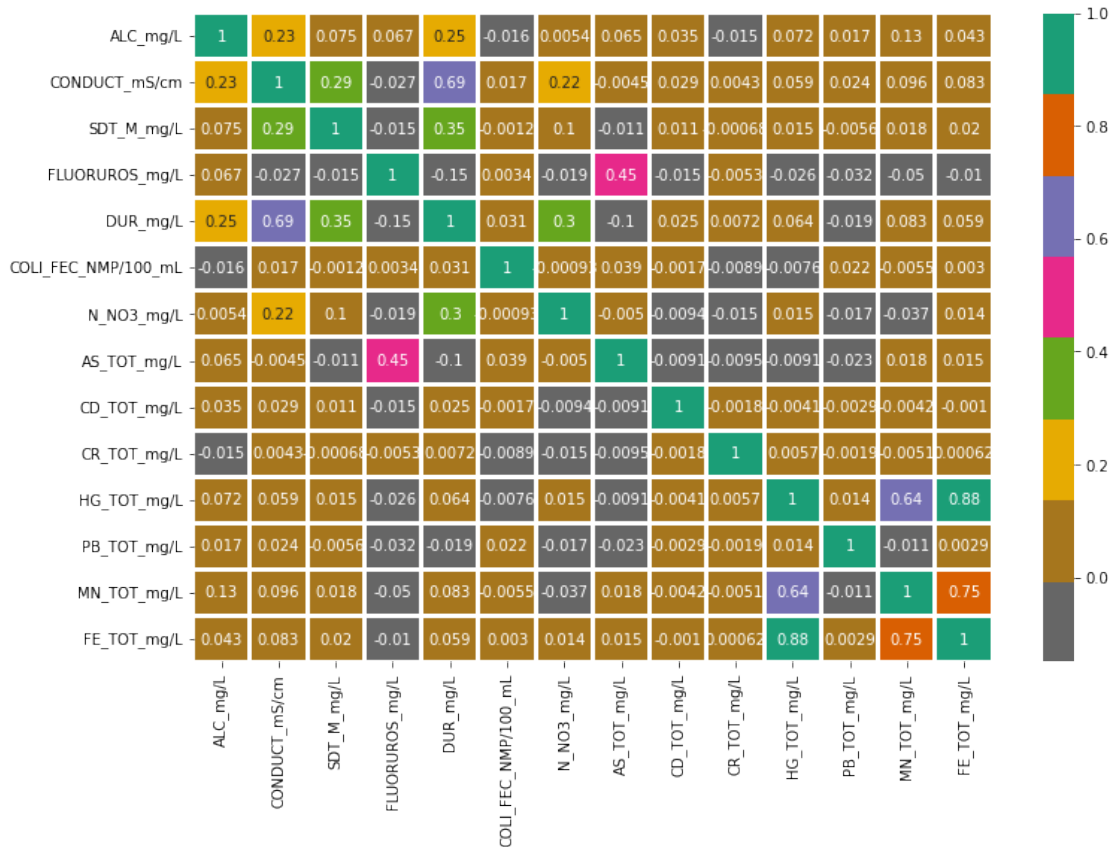
```
<seaborn.axisgrid.PairGrid at 0x7ff6fc5b8a10>
```



*#Mapa de correlaciones nos permite identificar  
#que variables se encuentran correlacionadas.*

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
sns.heatmap(df_explore.corr(), annot=True, cmap='Dark2_r', linewidths
= 2)
plt.show()
```





*#Llamado a librerias para el graficado de las  
#coordenadas de las localidades de las aguas subterraneas*

```
import geopandas
from geopy.geocoders import Nominatim
import geopandas as gpd
from shapely.geometry import Point
import qeds
qeds.themes.mpl_style();
```

*#Latitudes y Longitudes*

```
latlong = Clean_df[["LATITUD", "LONGITUD"]]
latlong["Coordinates"] = list(zip(latlong.LONGITUD, latlong.LATITUD))
latlong["Coordinates"] = latlong["Coordinates"].apply(Point)
latlong.head()
```

```
/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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
after removing the cwd from sys.path.

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

```
gdf = gpd.GeoDataFrame(latlong, geometry="Coordinates")
gdf.head()
```

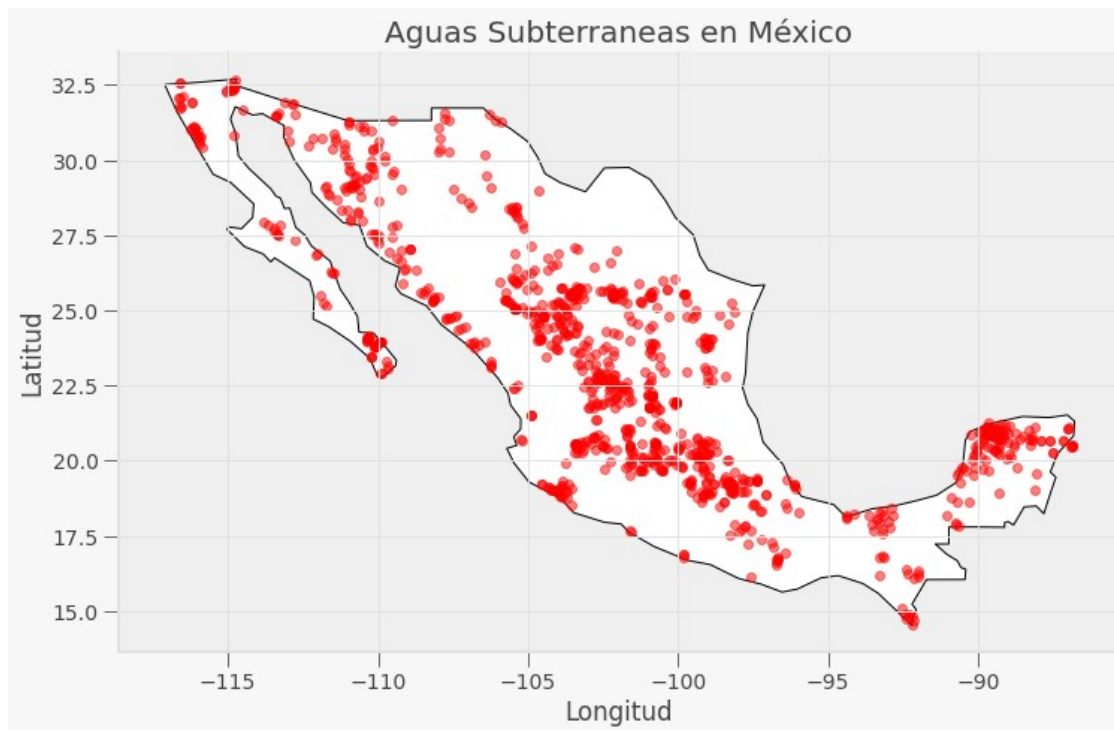
	LATITUD	LONGITUD	Coordinates
0	22.20887	-102.02210	POINT (-102.02210 22.20887)
1	21.99958	-102.20075	POINT (-102.20075 21.99958)
2	22.36685	-102.28801	POINT (-102.28801 22.36685)
3	22.18435	-102.29449	POINT (-102.29449 22.18435)
4	23.45138	-110.24480	POINT (-110.24480 23.45138)

```
world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world = world.set_index("iso_a3")
```

```
import matplotlib.pyplot as plt
fig, gax = plt.subplots(figsize=(12,12))
```

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

```
gdf.plot(ax=gax, color='red', alpha = 0.5)
gax.set_xlabel('Longitud')
gax.set_ylabel('Latitud')
gax.set_title('Aguas Subterráneas en México')
gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)
plt.show()
```



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

```
! pip install haversine
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting haversine

Downloading haversine-2.7.0-py2.py3-none-any.whl (6.9 kB)

Installing collected packages: haversine

Successfully installed haversine-2.7.0

*#Librerías para el análisis de K-MEANS*

```
import scipy.spatial
from haversine import haversine
from sklearn.cluster import KMeans
```

*#Revisión de los datos de latitud y longitud*

```
latlong = Clean_df[["LATITUD", "LONGITUD"]]
latlong.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 1054 entries, 0 to 1067
```

```
Data columns (total 2 columns):
```

#	Column	Non-Null Count	Dtype
0	LATITUD	1054 non-null	float64
1	LONGITUD	1054 non-null	float64

```
dtypes: float64(2)
memory usage: 24.7 KB
```

```
#Ajuste del modelo
```

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(latlong)
```

```
KMeans(n_clusters=3)
```

```
#Centro obtenidos por el análisis de K-MEANS
```

```
centers = kmeans.cluster_centers_
print("'kmeans' model intances is trained and the cluster centroids
are stored in 'centers'")
centers
```

```
'kmeans' model intances is trained and the cluster centroids are
stored in 'centers'
```

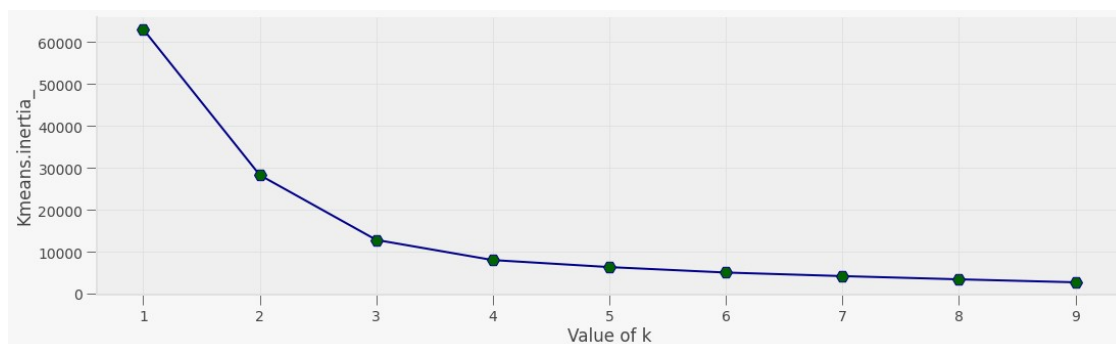
```
array([[ 22.27162356, -101.71558109],
       [ 28.42037512, -110.74089614],
       [ 19.47516461, -90.69843377]])
```

```
#Método de Codo, para obtener el valor de K-MEANS "Optimo"
```

```
sum_square = {}
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k).fit(latlong)
    sum_square[k] = kmeans.inertia_
```

```
fig,ax = plt.subplots(figsize=(18,5))
ax.plot(list(sum_square.keys()),
list(sum_square.values()),ls='-',marker='H', color='DarkBlue', lw=2,
markersize=12,markerfacecolor = 'DarkGreen')
ax.set_xlabel("Value of k")
ax.set_ylabel("Kmeans.inertia_")
```

```
Text(0, 0.5, 'Kmeans.inertia_')
```



```
#Las coordenadas se encuentran etiquetadas a un determinado "Cluster"
#esto como resultado del análisis
```

```
clustering = KMeans(n_clusters = 3, max_iter= 10000)
clustering.fit(latlong)
```

```
Optimal_centers = pd.DataFrame(clustering.cluster_centers_)
print("Los centroides se encuentran en las siguientes coordenadas:\n",
      clustering.cluster_centers_)
latlong["Cluster"] = clustering.labels_
latlong
```

Los centroides se encuentran en las siguientes coordenadas:

```
[[ 28.42037512 -110.74089614]
 [ 19.47516461 -90.69843377]
 [ 22.27162356 -101.71558109]]
```

```
/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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
import sys

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

[1054 rows x 3 columns]

Optimal\_centers

	0	1
0	28.420375	-110.740896
1	19.475165	-90.698434
2	22.271624	-101.715581

```
latlong["SEMAFORO"] = Clean_df['SEMAFORO']
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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

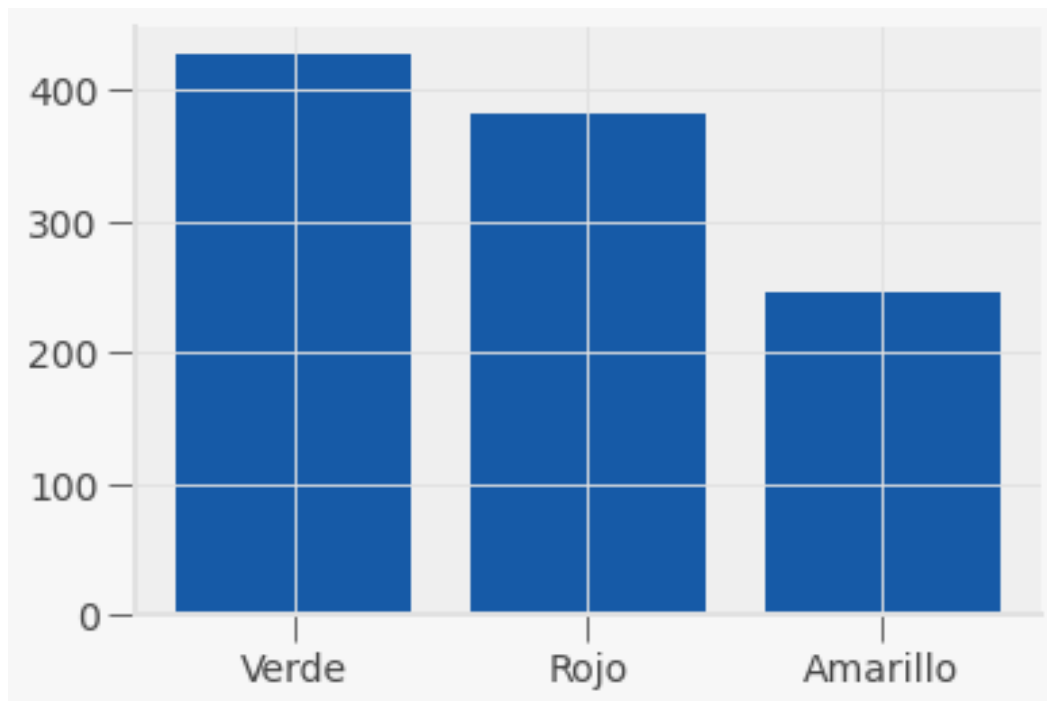
```
"""Entry point for launching an IPython kernel.
```

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

```
[1054 rows x 4 columns]
```

*#Relación de la calidad del agua y su ubicación geográfica*

```
x_values = latlong['SEMAFORO'].unique()
y_values = latlong['SEMAFORO'].value_counts().tolist()
plt.bar(x_values, y_values)
plt.show()
plt.close('all')
y_values
```

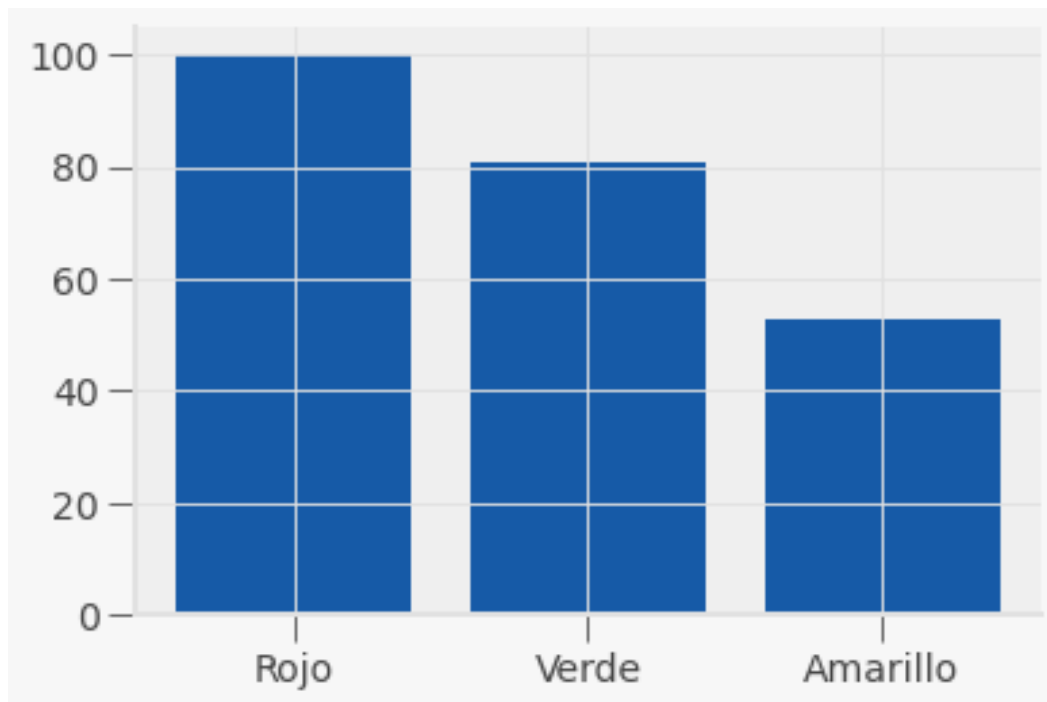


```
[427, 382, 245]
```

```
rslt_df_clt_0 = latlong[latlong['Cluster'] == 0]  
rslt_df_clt_1 = latlong[latlong['Cluster'] == 1]  
rslt_df_clt_2 = latlong[latlong['Cluster'] == 2]
```

*#Esta grafica de barras muestra el semaforo del "Cluster #0"*

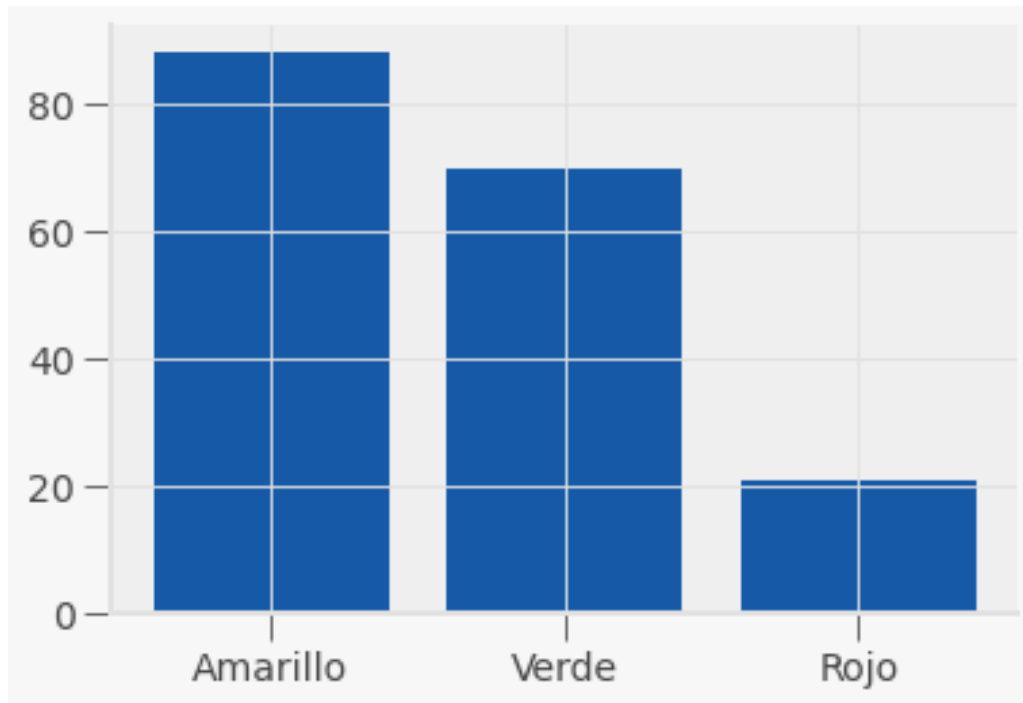
```
x_values = rslt_df_clt_0['SEMAFORO'].unique()  
y_values = rslt_df_clt_0['SEMAFORO'].value_counts().tolist()  
plt.bar(x_values, y_values)  
plt.show()  
plt.close('all')  
y_values
```



```
[100, 81, 53]
```

*#Esta grafica de barras muestra el semaforo del "Cluster #1"*

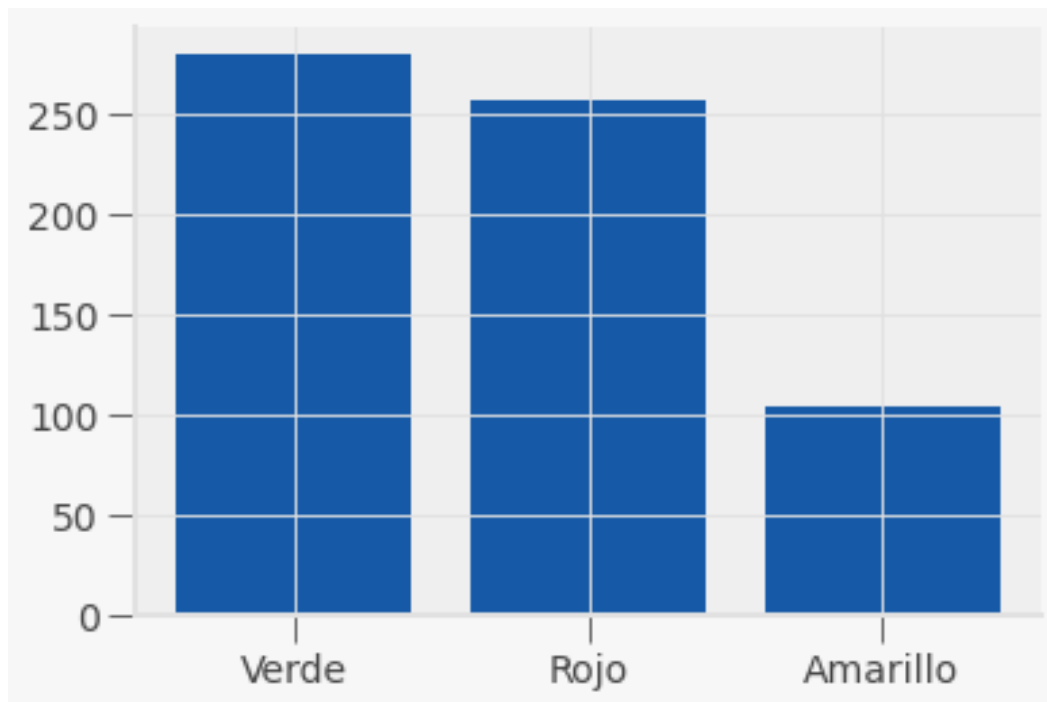
```
x_values = rslt_df_clt_1['SEMAFORO'].unique()  
y_values = rslt_df_clt_1['SEMAFORO'].value_counts().tolist()  
plt.bar(x_values, y_values)  
plt.show()  
plt.close('all')  
y_values
```



[88, 70, 21]

```
#Esta grafica de barras muestra el semaforo del "Cluster #2"  
x_values = rslt_df_clt_2['SEMAFORO'].unique()  
y_values = rslt_df_clt_2['SEMAFORO'].value_counts().tolist()  
plt.bar(x_values, y_values)  
plt.show()  
plt.close('all')  
y_values
```





```
[280, 257, 104]
```

```
latlong.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 1054 entries, 0 to 1067
```

```
Data columns (total 4 columns):
```

#	Column	Non-Null Count	Dtype
0	LATITUD	1054 non-null	float64
1	LONGITUD	1054 non-null	float64
2	Cluster	1054 non-null	int32
3	SEMAFORO	1054 non-null	object

```
dtypes: float64(2), int32(1), object(1)
```

```
memory usage: 37.1+ KB
```

```
#Grafica de dispersion que muestra las diferentes "Cluster" y su  
respectivo "Centro".
```

```
colors = ["#DF2020", "#81DF20", "#2095DF"]
```

```
latlong["colormap"] = latlong.Cluster.map({0:colors[0], 1:colors[1],  
2:colors[2]})
```

```
Lat = pd.DataFrame(latlong["LATITUD"])
```

```
Long = pd.DataFrame(latlong["LONGITUD"])
```

```
Clus = pd.DataFrame(latlong["colormap"])
```

```
plt.scatter(latlong.LONGITUD, latlong.LATITUD, c=latlong.colormap)
```

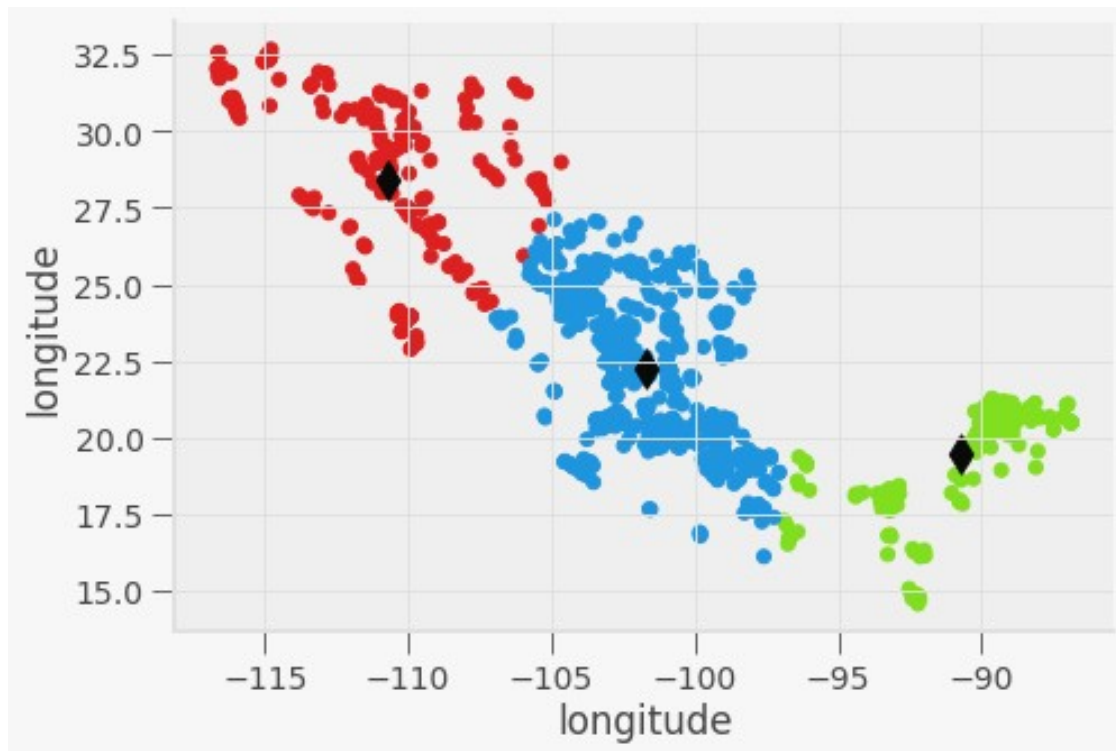
```
plt.scatter(Optimal_centers[1], Optimal_centers[0], marker = "d",
```

```
s=150, color= "#0A0A0A")
plt.gcf().set_size_inches((7.5,5))
```

```
plt.xlabel("longitude")
plt.ylabel("longitude")
plt.show()
```

/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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
This is separate from the ipykernel package so we can avoid doing imports until



```
#Ubicación geográfica de los "Centros de Cada Cluster"
geolocator = Nominatim(user_agent="geoapiExercises")
location_1 = geolocator.reverse(" 22.271624, -101.715581")
print(location_1)
location_2 = geolocator.reverse(" 19.475165, -90.698434")
print(location_2)
location_3 = geolocator.reverse("28.420375, -110.740896")
print(location_3)
```

Pinos, Zacatecas, México  
 Ciudad del Sol, Champotón, Campeche, México  
 Guaymas, Sonora, México

## #Parte 2

*#Preparación de los datos para el entrenamiento de los algoritmos de "Arbol de Decisión" y "Bosque Aleatorio"*

```
df_train_test =  
Clean_df[["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", "SEMAFORO"]].copy()  
df_train_test
```

	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L	\
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	

	COLI_FEC_NMP/100_mL	N_NO3_mg/L	AS_TOT_mg/L	CD_TOT_mg/L
CR_TOT_mg/L \				
0	1.1	4.184656	0.0161	0.003
0.005				
1	1.1	5.750110	0.0134	0.003
0.005				
2	1.1	1.449803	0.0370	0.003
0.005				
3	1.1	1.258597	0.0154	0.003
0.005				
4	291.0	15.672251	0.0100	0.003
0.005				
...	...	...	...	...
...				
1063	1.1	14.615488	0.0100	0.003
0.005				
1064	1.1	77.392000	0.0100	0.003
0.005				
1065	620.0	36.477104	0.0100	0.003
0.005				
1066	1.1	0.020000	0.0100	0.003
0.005				

```

1067          1.1      0.811876      0.0100      0.003
0.005

```

```

      HG_TOT_mg/L  PB_TOT_mg/L  MN_TOT_mg/L  FE_TOT_mg/L  SEMAFORO
0      0.0005      0.005      0.00150      0.08910      Verde
1      0.0005      0.005      0.00150      0.02500      Verde
2      0.0005      0.005      0.00150      0.02500      Rojo
3      0.0005      0.005      0.00150      0.02500      Verde
4      0.0005      0.005      0.00150      0.02500      Rojo
...
1063      0.0005      0.005      0.00150      0.02500      Rojo
1064      0.0005      0.005      0.00709      0.07578      Rojo
1065      0.0005      0.005      0.02420      0.21290      Rojo
1066      0.0005      0.005      0.01200      0.17860      Verde
1067      0.0005      0.005      0.00150      0.02500      Verde

```

```
[1054 rows x 15 columns]
```

```

#Datos Preparados para el entrenamiento de los
#modelos de Arbol de Decisión y Bosque Aleatorio

```

```

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

```

```

#Codificación de los "Targets"

```

```

df_train_test['SEMAFORO']=
label_encoder.fit_transform(df_train_test['SEMAFORO'])
df_train_test

```

```

      ALC_mg/L  CONDUCT_mS/cm  SDT_M_mg/L  FLUORUROS_mg/L  DUR_mg/L  \
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

```

```

      COLI_FEC_NMP/100_mL  N_NO3_mg/L  AS_TOT_mg/L  CD_TOT_mg/L
CR_TOT_mg/L  \
0          1.1      4.184656      0.0161      0.003
0.005
1          1.1      5.750110      0.0134      0.003
0.005
2          1.1      1.449803      0.0370      0.003
0.005
3          1.1      1.258597      0.0154      0.003

```

0.005				
4	291.0	15.672251	0.0100	0.003
0.005				
...	...	...	...	...
...				
1063	1.1	14.615488	0.0100	0.003
0.005				
1064	1.1	77.392000	0.0100	0.003
0.005				
1065	620.0	36.477104	0.0100	0.003
0.005				
1066	1.1	0.020000	0.0100	0.003
0.005				
1067	1.1	0.811876	0.0100	0.003
0.005				

	HG_TOT_mg/L	PB_TOT_mg/L	MN_TOT_mg/L	FE_TOT_mg/L	SEMAFORO
0	0.0005	0.005	0.00150	0.08910	2
1	0.0005	0.005	0.00150	0.02500	2
2	0.0005	0.005	0.00150	0.02500	1
3	0.0005	0.005	0.00150	0.02500	2
4	0.0005	0.005	0.00150	0.02500	1
...	...	...	...	...	...
1063	0.0005	0.005	0.00150	0.02500	1
1064	0.0005	0.005	0.00709	0.07578	1
1065	0.0005	0.005	0.02420	0.21290	1
1066	0.0005	0.005	0.01200	0.17860	2
1067	0.0005	0.005	0.00150	0.02500	2

[1054 rows x 15 columns]

*#Variables a utilizar para el entrenamiento y variable de salida*

X =

```
df_train_test[["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"]]
```

y = df\_train\_test[["SEMAFORO"]]

print(X.shape)

print(y.shape)

(1054, 14)

(1054, 1)

*#Division de los datos en entrenamiento y prueba*

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

```

random_state=0, train_size = .80)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)

(843, 14)
(843, 1)
(211, 14)
(211, 1)

from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Crea el Arbol de Decisión
mdl_dt = DecisionTreeClassifier()

# Entrena el Clasificador de Arbol de Decisión
clf = mdl_dt.fit(X_train,y_train)

#Realiza Predicciones con los Datos de Prueba
y_pred = mdl_dt.predict(X_test)

#Metricas del Modelo de Arboles de Decision
target_names = ['clase 0', 'clase 1', 'clase 2']
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred,
target_names=target_names))

Accuracy: 0.985781990521327

```

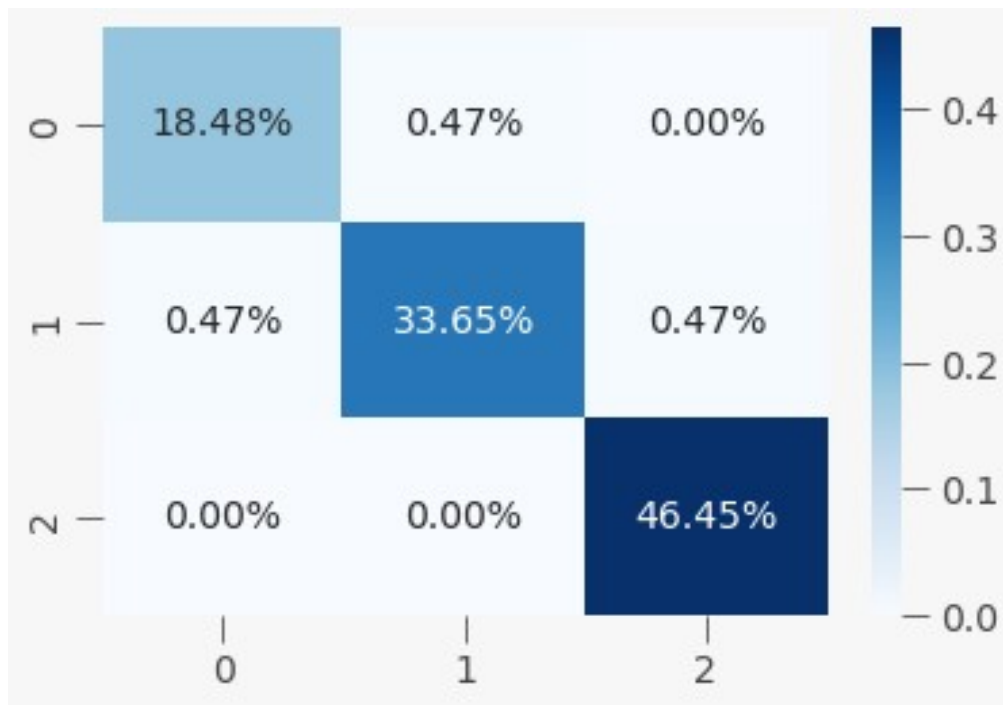
	precision	recall	f1-score	support
clase 0	0.97	0.97	0.97	40
clase 1	0.99	0.97	0.98	73
clase 2	0.99	1.00	0.99	98
accuracy			0.99	211
macro avg	0.98	0.98	0.98	211
weighted avg	0.99	0.99	0.99	211

```

#Matriz de Confusion del Modelo de Arboles de Decision
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
fmt='.2%', cmap='Blues')

<matplotlib.axes._subplots.AxesSubplot at 0x7ff6d597fc90>

```



*#Import Random Forest Model*

```
from sklearn.ensemble import RandomForestClassifier
```

*#Crea el Bosque Aleatorio*

```
mdl_rf=RandomForestClassifier(n_estimators=100)
```

*#Entrena el Clasificador de Bosque Aleatorio*

```
mdl_rf.fit(X_train,y_train)
```

*#Realiza Predicciones con los Datos de Prueba*

```
y_pred=mdl_rf.predict(X_test)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
```

*#Metricas del Modelo de Bosques Aleatorios*

```
target_names = ['clase 0', 'clase 1', 'clase 2']
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred,
target_names=target_names))
```

```
Accuracy: 0.9715639810426541
```

	precision	recall	f1-score	support
clase 0	0.95	0.97	0.96	40
clase 1	0.97	0.96	0.97	73

clase 2	0.98	0.98	0.98	98
accuracy			0.97	211
macro avg	0.97	0.97	0.97	211
weighted avg	0.97	0.97	0.97	211

```
#Matriz de Confusion del Modelo de Bosque Aleatorios
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
            fmt='.2%', cmap='Blues')
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7ff6d59b7410>



```
#Análisis de Características de Importancia
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from matplotlib import pyplot
# Conjunto de Datos
X, y = make_classification(n_samples=1000, n_features=14,
                          n_informative=5, n_redundant=5, random_state=1)
# Definición del modelo
model = DecisionTreeClassifier()
# Entrenamiento del modelo
model.fit(X, y)
# Importancia de las características
importance = model.feature_importances_
# Resumen de las características de importancia
for i,v in enumerate(importance):
```



```

    print('Feature: %0d, Score: %.5f' % (i,v))
# Graficado de las características en una grafica de barras
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()

```

```

Feature: 0, Score: 0.13077
Feature: 1, Score: 0.02305
Feature: 2, Score: 0.31446
Feature: 3, Score: 0.16901
Feature: 4, Score: 0.06828
Feature: 5, Score: 0.01115
Feature: 6, Score: 0.00512
Feature: 7, Score: 0.00857
Feature: 8, Score: 0.01987
Feature: 9, Score: 0.00812
Feature: 10, Score: 0.02312
Feature: 11, Score: 0.17730
Feature: 12, Score: 0.01812
Feature: 13, Score: 0.02306

```

