

# Tecnológico de Estudios Superiores de Monterrey

## Ciencia y analítica de datos

### Equipo 57

### Integrantes:

Juan Carlos Villamil Rojas A0

Axel Alejandro Tlatoa Villavicencio A01363351

### ▼ Análisis de aguas subterráneas

```
import numpy as np
%matplotlib inline
import matplotlib
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
from shapely.geometry import Point
import pandas as pd
import plotly.express as px
import seaborn as sns
from sklearn import preprocessing
from sklearn.cluster import KMeans
from google.colab import drive
import warnings
import math
from sklearn.datasets import make_blobs
from yellowbrick.cluster import KElbowVisualizer
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from shapely.geometry import Point
import requests, zipfile
from io import BytesIO
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, MinMaxScaler, Normalizer, St
```

```
!pip install geopandas
import geopandas as gpd
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public>  
Requirement already satisfied: geopandas in /usr/local/lib/python3.7/dist-packages (0.10.4)  
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages (1.7.1)  
Requirement already satisfied: pandas>=0.25.0 in /usr/local/lib/python3.7/dist-packages (1.1.5)  
Requirement already satisfied: fiona>=1.8 in /usr/local/lib/python3.7/dist-packages (1.8.2)  
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-packages (3.0.9)  
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-packages (0.7.2)  
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (57.5.0)  
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages (from fiona) (2.5.0)  
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (7.1.2)  
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from fiona) (2021.10.8)  
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (from fiona) (21.2.0)  
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (from fiona) (1.16.0)  
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/dist-packages (from fiona) (1.1.1)  
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2.8.2)  
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (1.21.0)  
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2021.3)

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

data = pd.read_csv('https://raw.githubusercontent.com/PosgradoMNA/actividades-de-aprendizaje-2021/main/actividades-de-aprendizaje-2021.csv')

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



```
data.describe()
```

	LONGITUD	LATITUD	PERIODO	ALC_mg/L	CONDUCT_mS/cm	SDT_mg/L
count	1068.000000	1068.000000	1068.0	1064.000000	1062.000000	0.0
mean	-101.891007	23.163618	2020.0	235.633759	1138.953013	NaN
std	6.703263	3.887670	0.0	116.874291	1245.563674	NaN
min	-116.664250	14.561150	2020.0	26.640000	50.400000	NaN
25%	-105.388865	20.212055	2020.0	164.000000	501.750000	NaN
50%	-102.174180	22.617190	2020.0	215.527500	815.000000	NaN
75%	-98.974716	25.510285	2020.0	292.710000	1322.750000	NaN
max	-86.864120	32.677713	2020.0	1650.000000	18577.000000	NaN



```
data.shape
```

(1068, 57)

```
data.info()
```

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

```

51 CALIDAD_CR          1068 non-null object
52 HG_TOT_mg/L        1068 non-null object
53 CALIDAD_HG          1068 non-null object
54 PB_TOT_mg/L         1068 non-null object
55 CALIDAD_PB          1068 non-null object
56 MN_TOT_mg/L         1068 non-null object
57 CALIDAD_MN          1068 non-null object
58 FE_TOT_mg/L         1068 non-null object
59 CALIDAD_FE          1068 non-null object
60 SEMAFORO            1068 non-null object
61 CONTAMINANTES       634 non-null object
62 CUMPLE_CON_ALC       1068 non-null object
63 CUMPLE_CON_COND      1068 non-null object
64 CUMPLE_CON_SDT_ra    1068 non-null object
65 CUMPLE_CON_SDT_salín 1068 non-null object
66 CUMPLE_CON_FLUO      1068 non-null object
67 CUMPLE_CON_DUR       1068 non-null object
68 CUMPLE_CON_CF        1068 non-null object
69 CUMPLE_CON_NO3       1068 non-null object
70 CUMPLE_CON_AS        1068 non-null object
71 CUMPLE_CON_CD        1068 non-null object
72 CUMPLE_CON_CR        1068 non-null object
73 CUMPLE_CON_HG        1068 non-null object
74 CUMPLE_CON_PB        1068 non-null object
75 CUMPLE_CON_MN        1068 non-null object
76 CUMPLE_CON_FE        1068 non-null object

```

```
dtypes: float64(5), int64(1), object(51)
```

```
memory usage: 475.7+ KB
```

Existe una gran cantidad de variables de entrada de tipo *objeto* y en menor proporción datos numéricos de tipo *flotante*. La variable de entrada **SDT\_mg/L** no posee ningún valor relevante, se procede a eliminar la columna:

```
data.isna().any()
```

```

CLAVE                False
SITIO                False
ORGANISMO_DE_CUENCA  False
ESTADO              False
MUNICIPIO            False
ACUIFERO             False
SUBTIPO              False
LONGITUD             False
LATITUD              False
PERIODO              False
ALC_mg/L             True
CALIDAD_ALC          True
CONDUCT_mS/cm        True
CALIDAD_CONDUCT      True
SDT_mg/L             True
SDT_M_mg/L           True
CALIDAD_SDT_ra       True
CALIDAD_SDT_salín    True

```

```
ELIQUORIOS mg/l      False
```

```

FLUORURO_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

```

```
data.mean()
```

```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
LONGITUD      -101.891007
LATITUD       23.163618
PERIODO       2020.000000
ALC_mg/L      235.633759
CONDUCT_mS/cm 1138.953013
SDT_mg/L      NaN
dtype: float64

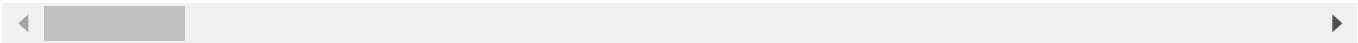
```

```
data.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_salín', '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_salín', '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')
```

```
data.tail()
```

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	ACUIFERO
1063	OCRBR5101M1	L-310 (COMUNIDAD SAN MANUEL)	RIO BRAVO	NUEVO LEON	LINARES	CITRICOLA SUF
1064	OCRBR5102M1	L-305 (EJIDO OJO DE AGUA LAS CRUCESITAS)	RIO BRAVO	NUEVO LEON	LINARES	CITRICOLA SUF
1065	OCRBR5105M2	HACIENDA MEXIQUITO POZO 01	RIO BRAVO	NUEVO LEON	CADEREYTA JIMENEZ	CITRICOLA NORTE
1066	OCRBR5106M1	COMUNIDAD LOS POCITOS	RIO BRAVO	NUEVO LEON	GALEANA	NAVIDAD POTOSI RAICES
1067	OCRBR5109M1	COMUNIDAD LA REFORMA	RIO BRAVO	NUEVO LEON	GALEANA	NAVIDAD POTOSI RAICES



```
data.size
```

60876

```
data.columns.shape
```

```
(57,)
```

Existen datos perdidos, hay una variedad métodos para completar los datos, y puesto que la información es muy relevante para determinar la calidad del agua en zonas subterráneas se tendrá que evaluar el cómo proceder en cuanto a su limpieza, de modo que se evaluará el mejor tratamiento a implementar.

```
data_copy = data.copy()
```

```
data_copy.isnull().sum(axis=1).value_counts()
```

```
1    627
2    427
4      9
3      5
dtype: int64
```

```
data_copy.dropna(axis = 1,thresh = data.shape[0]/2, inplace = True) # Se elimina la columna q
print(f'Cambio de dimensión, una columna menos: {data_copy.shape[1]}')
```

```
Cambio de dimensión, una columna menos: 56
```

```
column_names = [data_copy.columns]
```

```
# Transformación de nombres de variables de entrada para su mejor manipulación
data_copy.columns = ['C' + str(n+1) for n in range(data_copy.shape[1])]
data_copy.head(3)
```

C1 C2 C3 C4 C5

Evaluamos las siguientes columnas:

U DLAGUB SAN GIL SANTIAGO AGUASCALIENTES ASIENFOS CHICAGO

```
print('Valor máximo de calidad del agua para alcalinidad:', data_copy[['C11']].max())
print('Valor mínimo de calidad del agua para alcalinidad:', data_copy[['C11']].min())
print('Delta:', data_copy[['C11']].max() - data_copy[['C11']].min())
print('Mediana de calidad del agua para alcalinidad:', data_copy[['C11']].median())
print('Moda de calidad del agua para alcalinidad:', data_copy[['C11']].mode())
print('Promedio de calidad del agua para alcalinidad:', data_copy[['C11']].mean())
print('Número de datos faltantes:', data_copy[['C11']].isnull().sum())
print(f'Porcentaje de sesgo por manipulación {(4/data.shape[0])*100}%')
```

```
Valor máximo de calidad del agua para alcalinidad: C11    1650.0
dtype: float64
Valor mínimo de calidad del agua para alcalinidad: C11    26.64
dtype: float64
Delta: C11    1623.36
dtype: float64
Mediana de calidad del agua para alcalinidad: C11    215.5275
dtype: float64
Moda de calidad del agua para alcalinidad: C11
0    157.62
Promedio de calidad del agua para alcalinidad: C11    235.633759
dtype: float64
Número de datos faltantes: C11    4
dtype: int64
Porcentaje de sesgo por manipulación 0.37453183520599254%
```

Los datos faltantes constituyen un 0.37% de la muestra total, por lo que cambiarlos por la mediana no resultará en un gran sesgo pues no se afecta ni el 1% de los datos totales, la media y la mediana tienen valores similares, con 20.1 de diferencia, se utilizará el valor de mayor magnitud para cubrir el delta entre el valor máximo y el mínimo.

```
data_copy['C11'].fillna(value = data_copy.C11.median().round(3), inplace = True)
```

```
data_copy[['C11']].isnull().sum()
```

```
C11    0
dtype: int64
```

```
print('Clasificación de variable:', data_copy[['C12']].dtypes)
print('Número de datos faltantes:', data_copy[['C12']].isnull().sum())
print(f'Porcentaje de sesgo por manipulación {(4/data.shape[0])*100}%')
print('Primeros 2 renglones:', data_copy[['C12']].head(2))
print('Moda:', data_copy[['C12']].mode())
```

```
Clasificación de variable: C12    object
```



```

dtype: object
Número de datos faltantes: C12    4
dtype: int64
Porcentaje de sezo por manipulación 0.37453183520599254%
Primeros 2 renglones:    C12
0  Alta
1  Alta
Moda:    C12
0  Alta

```

Notamos que es de tipo objeto o en mayor detalle es equivalente a str, luego se transformará usando *LabelEncoder* por ahora solamente llenamos esos espacios vacíos con la moda.

```
data_copy['C12'].fillna(value = data_copy.C12.mode()[0], inplace = True)
```

```
data_copy[['C12']].isnull().sum()
```

```

C12    0
dtype: int64

```

Hay que hacer notar que las columnas: *CONDUCT\_mS/cm* o *C13* y *CALIDAD\_CONDUC* o *C14* son básicamente la medición y su clasificación/interpretación en relación a su uso en zonas de riego. Se mide la conductividad del agua haciendo uso del *SI* de unidades (Siemens por metro). La conductividad se define como la capacidad que tienen los materiales de dejar pasar corriente y es el inverso de la resistividad o la resistencia que presentan los materiales para dejar pasar corriente através, matemáticamente interpretada como:

$$\sigma = \mathcal{S} \frac{\ell}{A} \left[ \frac{S}{m} \right]$$

$\sigma$  la conductividad,  $\mathcal{S}$  conductancia (inverso de la resistencia),  $A$  el área transversal,  $\ell$  la longitud del material.

Teniendo esto en cuenta podemos intuir que la escala indicada tiene un efecto directo en la eficiencia de las aguas de riego para el uso doméstico, la escala indica que:

Calidad del agua para Conductividad	Criterio Característico
Excelente para riego	$\sigma \leq 250$
Buena para riego	$250 < \sigma \leq 750$
Permisible para riego	$750 < \sigma \leq 2000$
Dudosa para riego	$2000 < \sigma \leq 3000$
Indeseable para riego	$\sigma > 3000$

Entonces ambas columnas deben de tener resultados coherentes, debido a su escala y su critero.

```

print('Clasificación de variable:', data_copy[['C13']].dtypes)
print('Clasificación de variable:', data_copy[['C14']].dtypes)
print('Número de datos faltantes:', data_copy[['C13']].isnull().sum())
print('Número de datos faltantes:', data_copy[['C14']].isnull().sum())
print(f'Porcentaje de sesgo por manipulación {(6/data.shape[0])*100}%')
print('Valor máximo de calidad del agua conductividad:', data_copy[['C13']].max())
print('Valor mínimo de calidad del agua conductividad:', data_copy[['C13']].min())
print('Delta:', data_copy[['C13']].max() - data_copy[['C13']].min())
print('Mediana de calidad del agua conductividad', data_copy[['C13']].median())
print('Moda de calidad del agua conductividad:', data_copy[['C14']].mode())
print('Promedio de calidad del agua conductividad:', data_copy[['C13']].mean())

```

```

Clasificación de variable: C13      float64
dtype: object
Clasificación de variable: C14      object
dtype: object
Número de datos faltantes: C13      6
dtype: int64
Número de datos faltantes: C14      6
dtype: int64
Porcentaje de sesgo por manipulación 0.5617977528089888%
Valor máximo de calidad del agua conductividad: C13      18577.0
dtype: float64
Valor mínimo de calidad del agua conductividad: C13      50.4
dtype: float64
Delta: C13      18526.6
dtype: float64
Mediana de calidad del agua conductividad C13      815.0
dtype: float64
Moda de calidad del agua conductividad:      C14
0  Permisible para riego
Promedio de calidad del agua conductividad: C13      1138.953013
dtype: float64

```

No importa mucho si utilizamos la mediana o la media puesto que el rango en caso de sustitución permanece dentro del criterio *Permisible para riego* y no afectamos ni el uno por ciento de los datos, por ende ya sabemos que la siguiente columna tendrá que tener este criterio en los datos faltantes.

```

data_copy['C13'].fillna(value = data_copy.C13.median(), inplace = True)
data_copy['C14'].fillna(value = data_copy.C14.mode()[0], inplace = True)
data_copy[['C13']].isnull().sum(), data_copy[['C14']].isnull().sum()

(C13      0
 dtype: int64, C14      0
 dtype: int64)

```

De la misma forma esto sucede con las columnas de Calidad de agua para sólidos disueltos totales (agrícola y salinización): *SDT\_M\_mg/L* - escala, *CALIDAD\_SDT\_ra* - criterio para riego

agrícola, *CALIDAD\_SDT\_salin* - criterio de salinización.

### Riego Agrícola:

Calidad del agua para sólidos disueltos totales	Criterio Característico R.A.
Excelente para riego	$SDT \leq 500$
Cultivos sensibles	$500 < SDT \leq 1000$
Cultivos con manejo especial	$1000 < SDT \leq 2000$
Cultivos tolerantes	$2000 < SDT \leq 5000$
Indeseable para riego	$SDT > 5000$

### Salinización:

Calidad del agua para sólidos disueltos totales	Criterio Característico Salin
Potable - Dulce	$SDT \leq 1000$
Ligeramente salobres	$1000 < SDT \leq 2000$
Salobres	$2000 < SDT \leq 10000$
Salinas	$SDT > 10000$

```
print('Clasificación de variable:', data_copy[['C15']].dtypes)
print('Clasificación de variable:', data_copy[['C16']].dtypes)
print('Clasificación de variable:', data_copy[['C17']].dtypes)
```

```
Clasificación de variable: C15    object
dtype: object
Clasificación de variable: C16    object
dtype: object
Clasificación de variable: C17    object
dtype: object
```

```
data_copy['C15'] = data_copy['C15'].str.replace('<', '').astype(np.float64)
```

```
print('Clasificación de variable:', data_copy[['C15']].dtypes)
print('Número de datos faltantes:', data_copy[['C15']].isnull().sum())
print('Número de datos faltantes:', data_copy[['C16']].isnull().sum())
print('Número de datos faltantes:', data_copy[['C16']].isnull().sum())
print(f'Porcentaje de sezo por manipulación {(2/data.shape[0])*100}%')
print('Valor máximo de calidad del agua SDT:', data_copy[['C15']].max())
print('Valor mínimo de calidad del agua SDT:', data_copy[['C15']].min())
print('Delta:', data_copy[['C15']].max() - data_copy[['C15']].min())
print('Mediana de calidad del agua SDT', data_copy[['C15']].median())
print('Moda de calidad del agua SDT:', data_copy[['C16']].mode())
print('Moda de calidad del agua SDT:', data_copy[['C17']].mode())
print('Promedio de calidad del agua SDT:', data_copy[['C15']].mean())
```

```
Clasificación de variable: C15    float64
```

```

dtype: object
Número de datos faltantes: C15      2
dtype: int64
Número de datos faltantes: C16      2
dtype: int64
Número de datos faltantes: C16      2
dtype: int64
Porcentaje de sezo por manipulación 0.18726591760299627%
Valor máximo de calidad del agua SDT: C15      82170.0
dtype: float64
Valor mínimo de calidad del agua SDT: C15      25.0
dtype: float64
Delta: C15      82145.0
dtype: float64
Mediana de calidad del agua SDT C15      550.4
dtype: float64
Moda de calidad del agua SDT:                                C16
0  Excelente para riego
Moda de calidad del agua SDT:                                C17
0  Potable - Dulce
Promedio de calidad del agua SDT: C15      896.101567
dtype: float64

```

Si se toma la mediana y o la media no afectan los criterios porque los valores caen dentro de los rangos de: *Cultivos Sensibles* y *Potable-Dulce*

```

data_copy['C15'].fillna(value = data_copy.C15.mean(), inplace = True)
data_copy['C16'].fillna(value = 'Cultivos sensibles', inplace = True)
data_copy['C17'].fillna(value = data_copy.C17.mode()[0], inplace = True)
data_copy[['C15']].isnull().sum(),data_copy[['C16']].isnull().sum(),data_copy[['C17']].isnull

(C15      0
 dtype: int64, C16      0
 dtype: int64, C17      0
 dtype: int64)

```

```
data_copy.isna().any()
```

```

C1      False
C2      False
C3      False
C4      False
C5      False
C6      False
C7      False
C8      False
C9      False
C10     False
C11     False
C12     False
C13     False
C14     False

```

```
C15    False
C16    False
C17    False
C18    False
C19    False
C20     True
C21     True
C22    False
C23    False
C24     True
C25     True
C26    False
C27    False
C28    False
C29    False
C30    False
C31    False
C32    False
C33    False
C34    False
C35    False
C36    False
C37    False
C38    False
C39    False
C40    False
C41     True
C42    False
C43    False
C44    False
C45    False
C46    False
C47    False
C48    False
C49    False
C50    False
C51    False
C52    False
C53    False
C54    False
C55    False
C56    False
dtype: bool
```

```
print('Clasificación de variable:', data_copy[['C20']].dtypes)
print('Clasificación de variable:', data_copy[['C21']].dtypes)
```

```
Clasificación de variable: C20    object
dtype: object
Clasificación de variable: C21    object
dtype: object
```

```
data_copy[['C20']].isnull().sum(),data_copy[['C21']].isnull().sum()
```

```
(C20      1
 dtype: int64, C21      1
 dtype: int64)
```

```
data_copy[['C20']].head(3)
```

	C20
0	213.732
1	185.0514
2	120.719

```
data_copy['C20'] = data_copy['C20'].str.replace('<', '').astype(np.float64)
```

```
print('Promedio de calidad del agua DUR:', data_copy[['C20']].mean())
print('Mediana de calidad del agua DUR', data_copy[['C20']].median())
print('Moda de calidad del agua DUR:', data_copy[['C21']].mode())
```

```
Promedio de calidad del agua DUR: C20      347.938073
dtype: float64
Mediana de calidad del agua DUR C20      245.3358
dtype: float64
Moda de calidad del agua DUR:          C21
0 Potable - Dura
```

La media y la mediana caen dentro del mismo rango.

Calidad del agua para dureza	Criterio Característico
Potable - Suave	$DUR \leq 60$
Potable - Moderadamente suave	$60 < DUR \leq 120$
Potable - Dura	$120 < DUR \leq 500$
Muy dura e indeseable usos industrial y doméstico	$DUR > 120$

```
data_copy['C20'].fillna(value = data_copy.C20.mean(), inplace = True)
data_copy['C21'].fillna(value = data_copy.C21.mode()[0], inplace = True)
```

```
data_copy[['C20']].isnull().sum(), data_copy[['C21']].isnull().sum()
```

```
(C20      0
 dtype: int64, C21      0
 dtype: int64)
```

```
data_copy.isna().any()
```

```
C1      False
```

```
C2      False
C3      False
C4      False
C5      False
C6      False
C7      False
C8      False
C9      False
C10     False
C11     False
C12     False
C13     False
C14     False
C15     False
C16     False
C17     False
C18     False
C19     False
C20     False
C21     False
C22     False
C23     False
C24     True
C25     True
C26     False
C27     False
C28     False
C29     False
C30     False
C31     False
C32     False
C33     False
C34     False
C35     False
C36     False
C37     False
C38     False
C39     False
C40     False
C41     True
C42     False
C43     False
C44     False
C45     False
C46     False
C47     False
C48     False
C49     False
C50     False
C51     False
C52     False
C53     False
C54     False
C55     False
C56     False
dtype: bool
```


```
data_copy[['C24']].isnull().sum(),data_copy[['C25']].isnull().sum()
```

```
(C24      1
 dtype: int64, C25      1
 dtype: int64)
```

```
print('Clasificación de variable:', data_copy[['C24']].dtypes)
print('Clasificación de variable:', data_copy[['C25']].dtypes)
```

```
Clasificación de variable: C24      object
dtype: object
Clasificación de variable: C25      object
dtype: object
```

```
data_copy[['C24']].head(3)
```

	C24 
0	4.184656
1	5.75011
2	1.449803

```
data_copy['C24'] = data_copy['C24'].str.replace('<', '').astype(np.float64)
```

```
print('Promedio de calidad del agua SDT:', data_copy[['C24']].mean())
print('Mediana de calidad del agua SDT', data_copy[['C24']].median())
print('Moda de calidad del agua DUR:', data_copy[['C25']].mode())
```

```
Promedio de calidad del agua SDT: C24      4.319759
dtype: float64
Mediana de calidad del agua SDT C24      2.080932
dtype: float64
Moda de calidad del agua DUR:                                C25
0 Potable - Excelente
```

Calidad del agua para nitrógeno de nitratos	Criterio Característico
---	-------------------------

Potable - Excelente	$N/NO_3 \leq 5$
Potable - Buena calidad	$5 < N/NO_3 \leq 11$
No apta como FAAP	$N/NO_3 > 11$

```
data_copy['C24'].fillna(value = data_copy.C24.mean(), inplace = True)
data_copy['C25'].fillna(value = data_copy.C25.mode()[0], inplace = True)
```

```
data_copy[['C41']].isnull().sum()
```



```
C41      434
dtype: int64
```

```
# Re transformamos a los nombres de las columnas originales:
```

```
data_copy.columns = column_names
```

```
data_copy.isna().any()
```

CLAVE	False
SITIO	False
ORGANISMO_DE_CUENCA	False
ESTADO	False
MUNICIPIO	False
ACUIFERO	False
SUBTIPO	False
LONGITUD	False
LATITUD	False
PERIODO	False
ALC_mg/L	False
CALIDAD_ALC	False
CONDUCT_mS/cm	False
CALIDAD_CONDUC	False
SDT_M_mg/L	False
CALIDAD_SDT_ra	False
CALIDAD_SDT_salin	False
FLUORUROS_mg/L	False
CALIDAD_FLUO	False
DUR_mg/L	False
CALIDAD_DUR	False
COLI_FEC_NMP/100_mL	False
CALIDAD_COLI_FEC	False
N_NO3_mg/L	False
CALIDAD_N_NO3	False
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
```

```
index = data_copy.index[0]
data_copy.iloc[index - 4: index + 4, :]
```

CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	ACUIFERO	SUBTIPO	LONGITUD	LATITUD
-------	-------	---------------------	--------	-----------	----------	---------	----------	---------



```
data_copy[data_copy == "<2"].count() / data_copy.count()
```

```
LATITUD      0.0
LONGITUD     0.0
SEMAFORO     0.0
SEMAFORO_Type 0.0
dtype: float64
```

```
data_copy = data[['LATITUD','LONGITUD','SEMAFORO']]
data_copy = data_copy.dropna()
data_copy['SEMAFORO'] = data_copy['SEMAFORO'].replace({'Verde':1,'Amarillo':2,'Rojo':3})
data_copy.head()
```

	LATITUD	LONGITUD	SEMAFORO
0	22.20887	-102.02210	1
1	21.99958	-102.20075	1
2	22.36685	-102.28801	3
3	22.18435	-102.29449	1
4	23.45138	-110.24480	3

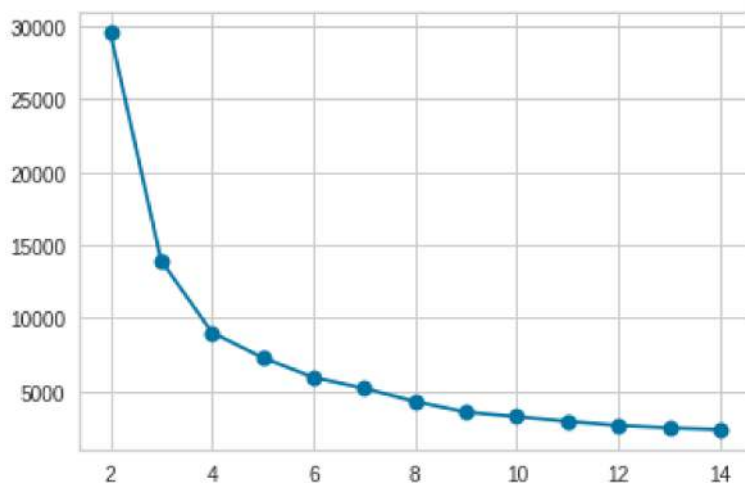
```
clusters = range(2,15)
inertias = []
labels = []
```

```

for k in clusters:
    m = KMeans(n_clusters=k)
    m.fit(data_copy)
    labels.append(m.labels_)
    inertias.append(m.inertia_)

plt.plot(clusters, inertias, 'bo-', markersize=8)
plt.show()

```



```

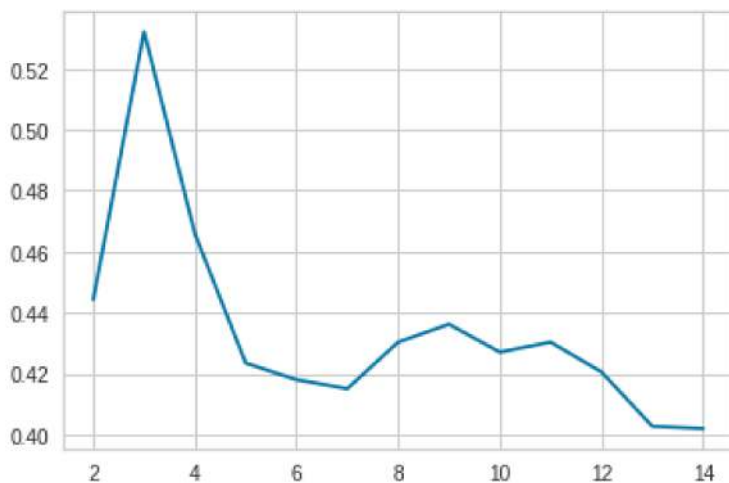
from sklearn.metrics import silhouette_score

ss = []
for l in labels:
    ss.append(silhouette_score(data_copy, labels=l))

plt.plot(clusters,ss)

```

[<matplotlib.lines.Line2D at 0x7f065d301b50>]



```

kmodelo = KMeans(n_clusters=7)
kmodelo.fit(data_copy)

```

```
centroide = kmodelo.cluster_centers_  
centroide
```

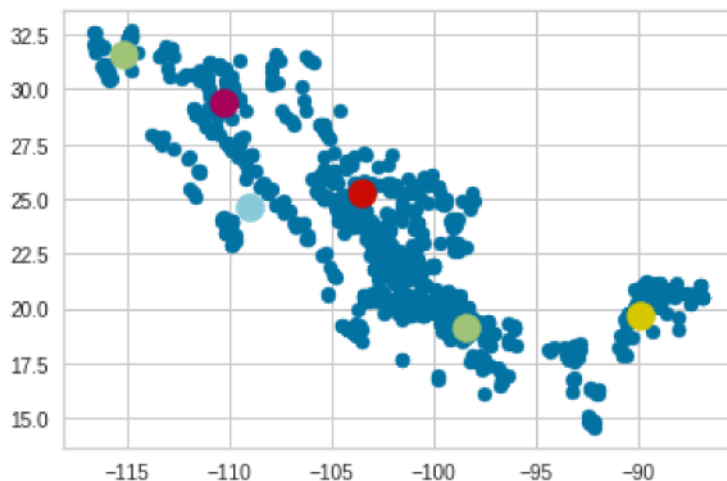
```
array([[ 19.13256277, -98.43223172,  1.64117647,  0.64117647],  
       [ 25.28486143, -103.50138428,  2.37656904,  1.37656904],  
       [ 29.3734628 , -110.30176434,  1.70873786,  0.70873786],  
       [ 19.68862526, -89.98552806,  1.70886076,  0.70886076],  
       [ 24.64584432, -109.07278654,  1.91139241,  0.91139241],  
       [ 21.44959177, -101.78433665,  2.01492537,  1.01492537],  
       [ 31.59499227, -115.20662469,  2.05882353,  1.05882353]])
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-293-525f1533a3d0> in <module>  
----> 1 world.name.unique
```

```
NameError: name 'world' is not defined
```

SEARCH STACK OVERFLOW

```
plt.scatter(data_copy.LONGITUD, data_copy.LATITUD)  
for i in range(len(centroide)):  
    plt.scatter(centroide[i][1],centroide[i][0], label = i, s= 200)  
plt.show()
```



```
data_copy  
data_copy["COORDENADAS"] = list(zip(data_copy.LONGITUD, data_copy.LATITUD))  
data_copy["COORDENADAS"] = data_copy["COORDENADAS"].apply(Point)  
data_copy.head()
```

	LATITUD	LONGITUD	SEMAFORO	SEMAFORO_Type	COORDENADAS
0	22.20887	-102.02210	1	0	POINT (-102.0221 22.20887)
1	21.99958	-102.20075	1	0	POINT (-102.20075 21.99958)
2	22.36685	-102.28801	3	2	POINT (-102.28801 22.36685)
3	22.18435	-102.29449	1	0	POINT (-102.29449 22.18435)



```
puntos = gpd.GeoDataFrame(data_copy, 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', cmap='Blues')
```

```
gax.set_xlabel('LATITUD')
```

```
gax.set_ylabel('LONGITUD')
```

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

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

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

```
puntos
```

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



```
plt.figure(figsize=(12,8))
sns.heatmap(data_copy.corr(), annot=True, cmap='Dark2_r', linewidths = 2)
plt.show()
```



```

categorias = preprocessing.LabelEncoder()
categorias.fit(data['SEMAFORO'])
categorias_name_mapping = dict(zip(categorias.classes_, categorias.transform(categorias.class
print(categorias_name_mapping)

```

```

# Encode labels in column 'species'.
data['SEMAFORO']= categorias.fit_transform(data['SEMAFORO'])
data['SEMAFORO'].unique

```

```

{'Amarillo': 0, 'Rojo': 1, 'Verde': 2}
<bound method Series.unique of 0      2
1      2
2      1
3      2
4      1
..
1063    1
1064    1
1065    1
1066    2
1067    2
Name: SEMAFORO, Length: 1068, dtype: int64>

```

```

Datos = data.select_dtypes(include=["float"])
X = Datos.columns
X.shape

```

```

(5,)

```

```

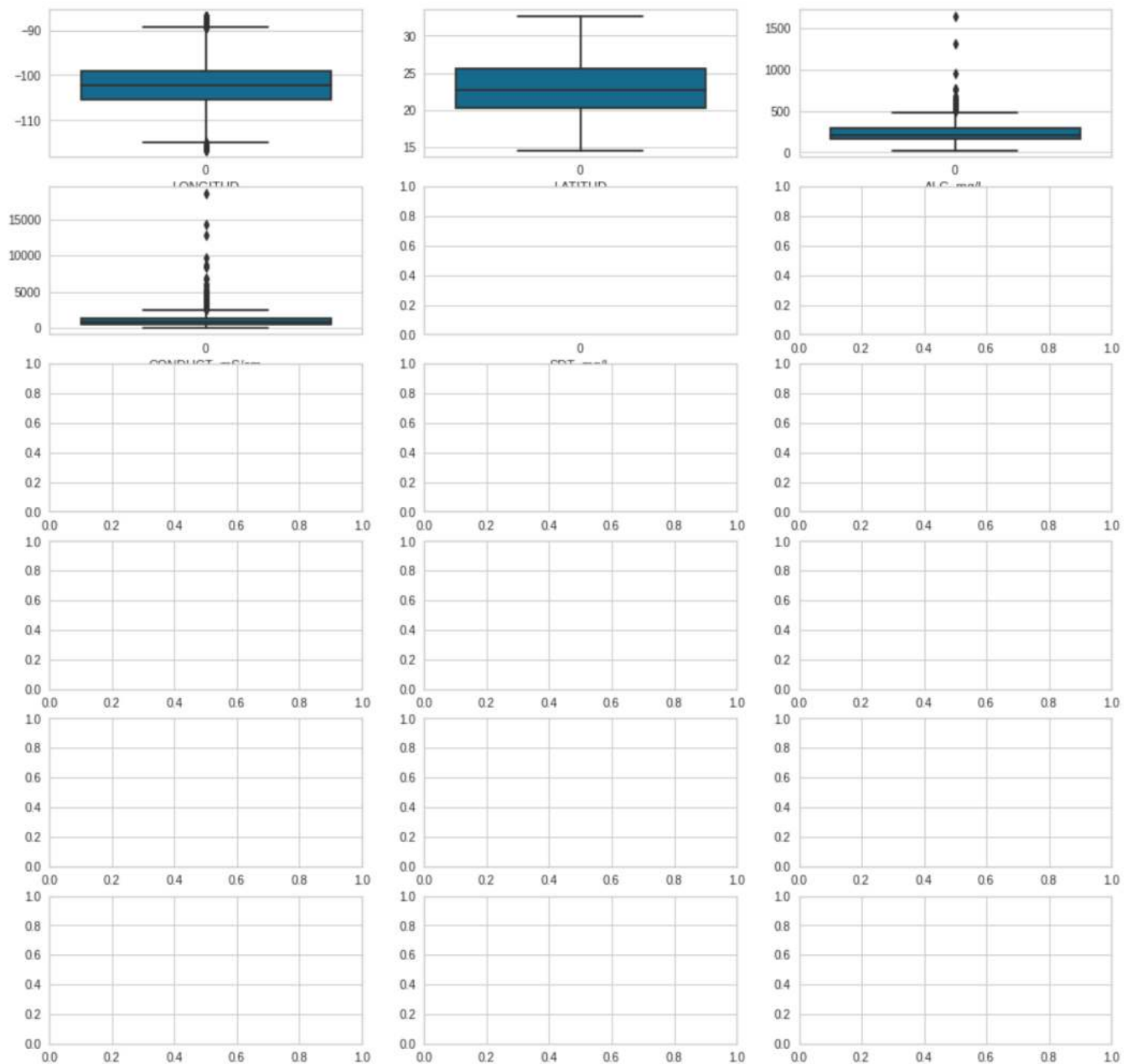
fig, axes = plt.subplots(6, 3, figsize=(16, 16))

```

```

for name, ax in zip(X, axes.flatten()):
    sns.boxplot(data=data[name], orient='v', ax=ax).set(xlabel=name)

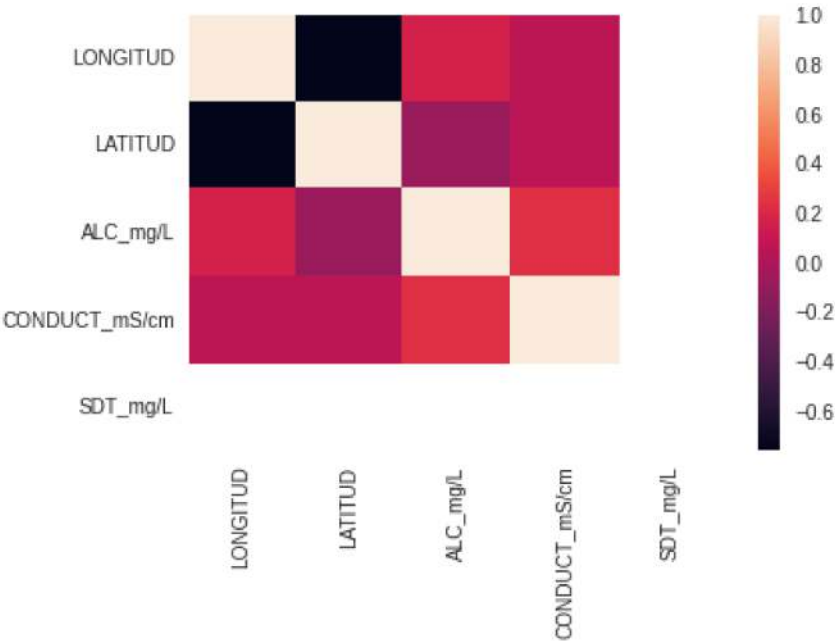
```



```
Relacion = Datos.corr()
sns.heatmap(Relacion)
```



<matplotlib.axes.\_subplots.AxesSubplot at 0x7f065cbc8390>



Productos pagados de Colab - [Cancela los contratos aquí](#)

✓ 0 s se ejecutó 00:03

