

▼ Instituto Tecnológico de Estudios Superiores de Monterrey

Maestría en Inteligencia Artificial Aplicada

Ciencia y Analítica de datos

Nombre del trabajo: Reto -> Entrega 2: Clasificación y ensambles

Nombre del maestro: María de la Paz Rico Fernández

Nombre de los estudiantes: Kevin Brandon Cruz Jorge Fernández Lara

Matrículas: A01794176 A01793062

Fecha de entrega: 18 de noviembre de 2022

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

```
Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: quandl in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: quantecon in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-packages (from qeds)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from qeds)

Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: funcy in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (from smart-open)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from smart-open)
```

```
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/loc
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pa
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
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
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-package
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: inflection>=0.3.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packa
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-p
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (
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-package
```



```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report
from sklearn.metrics import make_scorer
from sklearn.model_selection import RepeatedStratifiedKFold, GridSearchCV
from sklearn.metrics import PrecisionRecallDisplay

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from tqdm import tqdm
%matplotlib inline
import geopandas as gpd

from shapely.geometry import Point
```

```
%matplotlib inline
# activate plot theme
import qeds
qeds.themes.mpl_style();
from sklearn.cluster import KMeans

from statistics import mode

from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.linear_model import Ridge
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

from sklearn.tree import export_graphviz
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
import pydotplus
from IPython.display import Image
```

```
#Como hemos estado trabajando con las aguas superficiales, se vuelve a importar el mismo conj
filename = 'https://raw.githubusercontent.com/PosgradoMNA/actividades-del-proyecto-cad_kbcm_j
df = pd.read_csv(filename, encoding='cp1252')
df.head()
```

	CLAVE	SITIO	ORGANISMO_DE CUENCA	ESTADO	MUNICIPIO	CUENCA	CUEF
0	DLAGU8	PRESA EL SAUCILLO 100M AGUAS ARRIBA DE LA CORTINA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON DE ROMOS	RIO SAN PEDRO	PRE SAU

Vamos a separar las variables dependientes (X) de las independientes (Y).

Como nuestra variable dependiente o de salida corresponde a la de la calidad del agua, es la de "SEMAFORO".

```
# DIA DE LA SEMANA   LADOS   PENINSULA DE BAJA CALIFORNIA  LUGAR  SAN  CUE
```

```
df1 = df.copy()
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4141 entries, 0 to 4140
Data columns (total 55 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   CLAVE            3493 non-null    object 
 1   SITIO             3493 non-null    object 
 2   ORGANISMO_DE CUENCA  3493 non-null    object 
 3   ESTADO            3493 non-null    object 
 4   MUNICIPIO         3493 non-null    object 
 5   CUENCA            3492 non-null    object 
 6   CUERPO DE AGUA   3479 non-null    object 
 7   TIPO              3493 non-null    object 
 8   SUBTIPO           3479 non-null    object 
 9   LONGITUD          3493 non-null    float64
 10  LATITUD            3493 non-null    float64
 11  PERIODO           3493 non-null    float64
 12  DBO_mg/L          2581 non-null    object 
 13  CALIDAD_DBO       2581 non-null    object 
 14  DQO_mg/L          2581 non-null    object 
 15  CALIDAD_DQO       2581 non-null    object 
 16  SST_mg/L          3489 non-null    object 
 17  CALIDAD_SST        3489 non-null    object 
 18  COLI_FEC_NMP_100mL 2582 non-null    object 
 19  CALIDAD_COLI_FEC  2582 non-null    object 
 20  E_COLI_NMP_100mL   2582 non-null    object 
 21  CALIDAD_E_COLI     2582 non-null    object 
 22  ENTEROC_NMP_100mL  904 non-null    object 
 23  CALIDAD_ENTEROC    904 non-null    object 
 24  OD_PORC            1797 non-null    object 
 25  CALIDAD_OD_PORC    1797 non-null    object 
 26  OD_PORC_SUP        1619 non-null    object 
 27  CALIDAD_OD_PORC_SUP 1619 non-null    object 
 28  OD_PORC_MED        487 non-null    object
```

29	CALIDAD_OD_PORC_MED	487	non-null	object
30	OD_PORC_FON	946	non-null	object
31	CALIDAD_OD_PORC_FON	946	non-null	object
32	TOX_D_48_UT	1816	non-null	object
33	CALIDAD_TOX_D_48	1816	non-null	object
34	TOX_V_15_UT	1819	non-null	object
35	CALIDAD_TOX_V_15	1819	non-null	object
36	TOX_D_48_SUP_UT	762	non-null	object
37	CALIDAD_TOX_D_48_SUP	762	non-null	object
38	TOX_D_48_FON_UT	0	non-null	float64
39	CALIDAD_TOX_D_48_FON	0	non-null	float64
40	TOX_FIS_SUP_15_UT	1674	non-null	object
41	CALIDAD_TOX_FIS_SUP_15	1674	non-null	object
42	TOX_FIS_FON_15_UT	0	non-null	float64
43	CALIDAD_TOX_FIS_FON_15	0	non-null	float64
44	SEMAFORO	3493	non-null	object
45	CONTAMINANTES	2226	non-null	object
46	CUMPLE_CON_DBO	3493	non-null	object
47	CUMPLE_CON_DQO	3493	non-null	object
48	CUMPLE_CON_SST	3493	non-null	object
49	CUMPLE_CON_CF	3493	non-null	object
50	CUMPLE_CON_E_COLI	3493	non-null	object
51	CUMPLE_CON_ENTEROC	3493	non-null	object
52	CUMPLE_CON_OD	3493	non-null	object

El número de registros de la variable de salida 'SEMAFORO' es menor a los registros totales, por lo que se presume que si son 4141 registros y existen 3493, 648 registros de 'SEMAFORO' son NaN, que procederemos a eliminar.

```
df1.isna().sum()
```

CLAVE	648
SITIO	648
ORGANISMO_DE_CUENCA	648
ESTADO	648
MUNICIPIO	648
CUENCA	649
CUERPO_DE_AGUA	662
TIPO	648
SUBTIPO	662
LONGITUD	648
LATITUD	648
PERIODO	648
DBO_mg/L	1560
CALIDAD_DBO	1560
DQO_mg/L	1560
CALIDAD_DQO	1560
SST_mg/L	652
CALIDAD_SST	652
COLI_FEC_NMP_100mL	1559
CALIDAD_COLI_FEC	1559
E_COLI_NMP_100mL	1559
CALIDAD_E_COLI	1559
ENTEROC_NMP_100mL	3237

```
CALIDAD_ENTEROC          3237
OD_PORC                  2344
CALIDAD_OD_PORC          2344
OD_PORC_SUP              2522
CALIDAD_OD_PORC_SUP      2522
OD_PORC_MED              3654
CALIDAD_OD_PORC_MED      3654
OD_PORC_FON              3195
CALIDAD_OD_PORC_FON      3195
TOX_D_48_UT              2325
CALIDAD_TOX_D_48          2325
TOX_V_15_UT              2322
CALIDAD_TOX_V_15          2322
TOX_D_48_SUP_UT          3379
CALIDAD_TOX_D_48_SUP      3379
TOX_D_48_FON_UT          4141
CALIDAD_TOX_D_48_FON      4141
TOX_FIS_SUP_15_UT         2467
CALIDAD_TOX_FIS_SUP_15    2467
TOX_FIS_FON_15_UT         4141
CALIDAD_TOX_FIS_FON_15    4141
SEMAFORO                 648
CONTAMINANTES            1915
CUMPLE_CON_DBO           648
CUMPLE_CON_DQ0           648
CUMPLE_CON_SST           648
CUMPLE_CON_CF            648
CUMPLE_CON_E_COLI         648
CUMPLE_CON_ENTEROC        648
CUMPLE_CON_OD             648
CUMPLE_CON_TOX            648
GRUPO                     648
dtype: int64
```

```
df1 = df1.dropna(how = 'all')
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3493 entries, 0 to 3492
Data columns (total 55 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   CLAVE            3493 non-null   object 
 1   SITIO             3493 non-null   object 
 2   ORGANISMO_DE_CUENCA 3493 non-null   object 
 3   ESTADO            3493 non-null   object 
 4   MUNICIPIO         3493 non-null   object 
 5   CUENCA            3492 non-null   object 
 6   CUERPO DE AGUA   3479 non-null   object 
 7   TIPO              3493 non-null   object 
 8   SUBTIPO           3479 non-null   object 
 9   LONGITUD          3493 non-null   float64
 10  LATITUD           3493 non-null   float64
 11  PERIODO           3493 non-null   float64
```

12	DBO_mg/L	2581	non-null	object
13	CALIDAD_DBO	2581	non-null	object
14	DQO_mg/L	2581	non-null	object
15	CALIDAD_DQO	2581	non-null	object
16	SST_mg/L	3489	non-null	object
17	CALIDAD_SST	3489	non-null	object
18	COLI_FEC_NMP_100mL	2582	non-null	object
19	CALIDAD_COLI_FEC	2582	non-null	object
20	E_COLI_NMP_100mL	2582	non-null	object
21	CALIDAD_E_COLI	2582	non-null	object
22	ENTEROC_NMP_100mL	904	non-null	object
23	CALIDAD_ENTEROC	904	non-null	object
24	OD_PORC	1797	non-null	object
25	CALIDAD_OD_PORC	1797	non-null	object
26	OD_PORC_SUP	1619	non-null	object
27	CALIDAD_OD_PORC_SUP	1619	non-null	object
28	OD_PORC_MED	487	non-null	object
29	CALIDAD_OD_PORC_MED	487	non-null	object
30	OD_PORC_FON	946	non-null	object
31	CALIDAD_OD_PORC_FON	946	non-null	object
32	TOX_D_48_UT	1816	non-null	object
33	CALIDAD_TOX_D_48	1816	non-null	object
34	TOX_V_15_UT	1819	non-null	object
35	CALIDAD_TOX_V_15	1819	non-null	object
36	TOX_D_48_SUP_UT	762	non-null	object
37	CALIDAD_TOX_D_48_SUP	762	non-null	object
38	TOX_D_48_FON_UT	0	non-null	float64
39	CALIDAD_TOX_D_48_FON	0	non-null	float64
40	TOX_FIS_SUP_15_UT	1674	non-null	object
41	CALIDAD_TOX_FIS_SUP_15	1674	non-null	object
42	TOX_FIS_FON_15_UT	0	non-null	float64
43	CALIDAD_TOX_FIS_FON_15	0	non-null	float64
44	SEMAFORO	3493	non-null	object
45	CONTAMINANTES	2226	non-null	object
46	CUMPLE_CON_DBO	3493	non-null	object
47	CUMPLE_CON_DQO	3493	non-null	object
48	CUMPLE_CON_SST	3493	non-null	object
49	CUMPLE_CON_CF	3493	non-null	object
50	CUMPLE_CON_E_COLI	3493	non-null	object
51	CUMPLE_CON_ENTEROC	3493	non-null	object
52	CUMPLE_CON_OD	3493	non-null	object

Las variables TOX_D_48_FON_UT , CALIDAD_TOX_D_48_FON , TOX_FIS_FON_15_UT y CALIDAD_TOX_FIS_FON_15 están vacías por lo que se eliminarán.

```
vacias = ['TOX_D_48_FON_UT', 'CALIDAD_TOX_D_48_FON', 'TOX_FIS_FON_15_UT', 'CALIDAD_TOX_FIS_FON_15']
df1.drop(vacias, inplace = True, axis = 1)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3493 entries, 0 to 3492
Data columns (total 51 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   DBO_mg/L        3493 non-null    object 
 1   CALIDAD_DBO     3493 non-null    object 
 2   DQO_mg/L        3493 non-null    object 
 3   CALIDAD_DQO     3493 non-null    object 
 4   SST_mg/L        3493 non-null    object 
 5   CALIDAD_SST     3493 non-null    object 
 6   COLI_FEC_NMP_100mL  3493 non-null    object 
 7   CALIDAD_COLI_FEC 3493 non-null    object 
 8   E_COLI_NMP_100mL  3493 non-null    object 
 9   CALIDAD_E_COLI   3493 non-null    object 
 10  ENTEROC_NMP_100mL 3493 non-null    object 
 11  CALIDAD_ENTEROC 3493 non-null    object 
 12  OD_PORC          3493 non-null    object 
 13  CALIDAD_OD_PORC  3493 non-null    object 
 14  OD_PORC_SUP      3493 non-null    object 
 15  CALIDAD_OD_PORC_SUP 3493 non-null    object 
 16  OD_PORC_MED      3493 non-null    object 
 17  CALIDAD_OD_PORC_MED 3493 non-null    object 
 18  OD_PORC_FON      3493 non-null    object 
 19  CALIDAD_OD_PORC_FON 3493 non-null    object 
 20  TOX_D_48_UT       3493 non-null    object 
 21  CALIDAD_TOX_D_48  3493 non-null    object 
 22  TOX_V_15_UT       3493 non-null    object 
 23  CALIDAD_TOX_V_15  3493 non-null    object 
 24  TOX_D_48_SUP_UT  3493 non-null    object 
 25  CALIDAD_TOX_D_48_SUP 3493 non-null    object 
 26  TOX_D_48_FON_UT  3493 non-null    float64
 27  CALIDAD_TOX_D_48_FON 3493 non-null    float64
 28  TOX_FIS_SUP_15_UT 3493 non-null    object 
 29  CALIDAD_TOX_FIS_SUP_15 3493 non-null    object 
 30  TOX_FIS_FON_15_UT  3493 non-null    float64
 31  CALIDAD_TOX_FIS_FON_15 3493 non-null    float64
 32  SEMAFORO          3493 non-null    object 
 33  CONTAMINANTES     3493 non-null    object 
 34  CUMPLE_CON_DBO    3493 non-null    object 
 35  CUMPLE_CON_DQO    3493 non-null    object 
 36  CUMPLE_CON_SST    3493 non-null    object 
 37  CUMPLE_CON_CF     3493 non-null    object 
 38  CUMPLE_CON_E_COLI 3493 non-null    object 
 39  CUMPLE_CON_ENTEROC 3493 non-null    object 
 40  CUMPLE_CON_OD     3493 non-null    object 
```

0	CLAVE	3493	non-null	object
1	SITIO	3493	non-null	object
2	ORGANISMO_DE_CUENCA	3493	non-null	object
3	ESTADO	3493	non-null	object
4	MUNICIPIO	3493	non-null	object
5	CUENCA	3492	non-null	object
6	CUERPO DE AGUA	3479	non-null	object
7	TIPO	3493	non-null	object
8	SUBTIPO	3479	non-null	object
9	LONGITUD	3493	non-null	float64
10	LATITUD	3493	non-null	float64
11	PERIODO	3493	non-null	float64
12	DBO_mg/L	2581	non-null	object
13	CALIDAD_DBO	2581	non-null	object
14	DQO_mg/L	2581	non-null	object
15	CALIDAD_DQO	2581	non-null	object
16	SST_mg/L	3489	non-null	object
17	CALIDAD_SST	3489	non-null	object
18	COLI_FEC_NMP_100mL	2582	non-null	object
19	CALIDAD_COLI_FEC	2582	non-null	object
20	E_COLI_NMP_100mL	2582	non-null	object
21	CALIDAD_E_COLI	2582	non-null	object
22	ENTEROC_NMP_100mL	904	non-null	object
23	CALIDAD_ENTEROC	904	non-null	object
24	OD_PORC	1797	non-null	object
25	CALIDAD_OD_PORC	1797	non-null	object
26	OD_PORC_SUP	1619	non-null	object
27	CALIDAD_OD_PORC_SUP	1619	non-null	object
28	OD_PORC_MED	487	non-null	object
29	CALIDAD_OD_PORC_MED	487	non-null	object
30	OD_PORC_FON	946	non-null	object
31	CALIDAD_OD_PORC_FON	946	non-null	object
32	TOX_D_48_UT	1816	non-null	object
33	CALIDAD_TOX_D_48	1816	non-null	object
34	TOX_V_15_UT	1819	non-null	object
35	CALIDAD_TOX_V_15	1819	non-null	object
36	TOX_D_48_SUP_UT	762	non-null	object
37	CALIDAD_TOX_D_48_SUP	762	non-null	object
38	TOX_FIS_SUP_15_UT	1674	non-null	object
39	CALIDAD_TOX_FIS_SUP_15	1674	non-null	object
40	SEMAFORO	3493	non-null	object
41	CONTAMINANTES	2226	non-null	object
42	CUMPLE_CON_DBO	3493	non-null	object
43	CUMPLE_CON_DQO	3493	non-null	object
44	CUMPLE_CON_SST	3493	non-null	object
45	CUMPLE_CON_CF	3493	non-null	object
46	CUMPLE_CON_E_COLI	3493	non-null	object
47	CUMPLE_CON_ENTEROC	3493	non-null	object
48	CUMPLE_CON_OD	3493	non-null	object
49	CUMPLE_CON_TOX	3493	non-null	object
50	GRUPO	3493	non-null	object

Dónde:

0. **CLAVE:** es categórico Identificador del registro
1. **SITIO:** es categórica sobre ubicación
2. **ORGANISMO_DE_CUENCA:** categórica que identifica al responsable de administrar y preservar las aguas nacionales en cada una de las trece regiones hidrológico-administrativas en que se ha dividido el país
3. **ESTADO:** categórica sobre el estado en que está ubicado el sitio de muestreo
4. **MUNICIPIO:** categórica sobre el municipio en que está ubicado el sitio de muestreo
5. **CUENCA:** categórica sobre el nombre de la cuenca donde se localiza el sitio de monitoreo
6. **CUERPO DE AGUA:** categórica sobre el nombre del cuerpo de agua donde se localiza el sitio de monitoreo
7. **TIPO:** Categórica sobre el tipo de cuerpo de agua donde se encuentra el sitio de muestreo
8. **SUBTIPO:** Categórica sobre el subtipo de cuerpo de agua donde se encuentra el sitio de muestreo
9. **LONGITUD:** numérica longitud
10. **LATITUD:** numérica latitud
11. **PERIODO:** categórica Año en que se realizo el muestreo
12. **DBO_mg/L:** numérica Valor de la Demanda Bioquímica de Oxígeno, en miligramos por litro
13. **CALIDAD_DBO:** categórica Clasificación de la calidad del agua de acuerdo con el indicador Demanda Bioquímica de Oxígeno
14. **DQO_mg/L:** numérica Valor de la Demanda Química de Oxígeno, en miligramos por litro
15. **CALIDAD_DQO:** categórica Clasificación de la calidad del agua de acuerdo con el indicador Demanda Química de Oxígeno
16. **SST_mg/L:** numérica Valor de los Sólidos Suspensos Totales, en miligramos por litro
17. **CALIDAD_SST:** categórica Clasificación de la calidad del agua de acuerdo con el indicador Sólidos Suspensos Totales
18. **COLI_FEC_NMP_100mL:** numérica Valor de los Coliformes Fecales, en número más probable por cien mililitros
19. **CALIDAD_COLI_FEC:** categórica Clasificación de la calidad del agua de acuerdo con el indicador Coliformes Fecales
20. **E_COLI_NMP_100mL:** numérica Valor de Escherichia coli, en número más probable por cien mililitros
21. **CALIDAD_E_COLI:** categórica Clasificación de la calidad del agua de acuerdo con el indicador Escherichia coli
22. **ENTEROC_NMP_100mL:** numérica Valor de Enterococos fecales, en número más probable por cien mililitros
23. **CALIDAD_ENTEROC:** categórica Indica si cumple con la calidad de Excelente o Buena calidad, para el Indicador Enterococos fecales

24. **OD_PORC**: numérica Valor de Porcentaje de saturacion de oxigeno disuelto, en cuerpos loticos
25. **CALIDAD_OD_PORC**: categórica de saturación de oxígeno
26. **OD_PORC_SUP**: numérica Valor de Porcentaje de saturacion de oxigeno disuelto superficial
27. **CALIDAD_OD_PORC_SUP**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador Porcentaje de saturaci n de Oxigeno disuelto, superficial
28. **OD_PORC_MED**: numérica Valor de Porcentaje de saturacion de oxigeno disuelto medio
29. **CALIDAD_OD_PORC_MED**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador Porcentaje de saturacion de oxigeno disuelto, medio
30. **OD_PORC_FON**: numérica Valor de Porcentaje de saturacion de oxigeno disuelto en fondo
31. **CALIDAD_OD_PORC_FON**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador Porcentaje de saturacion de oxigeno disuelto, en fondo
32. **TOX_D_48_UT**: numérica Valor de Toxicidad, Dafnia magna, 48 horas, Unidades de Toxicidad, en cuerpos loticos
33. **CALIDAD_TOX_D_48**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador Toxicidad, Dafnia magna, 48 horas
34. **TOX_V_15_UT**: numérica Valor de Toxicidad, Vibrio Fisher, 15 minutos, Unidades de Toxicidad, en cuerpos loticos
35. **CALIDAD_TOX_V_15**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador Toxicidad, Vibrio Fisher, 15 minutos
36. **TOX_D_48_SUP_UT**: numérica Valor de Toxicidad, Dafnia magna 48 horas, superficial, Unidades de Toxicidad
37. **CALIDAD_TOX_D_48_SUP**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador de Toxicidad, Dafnia magna 48 horas, superficial
38. **TOX_D_48_FON_UT**: numérica Valor de Toxicidad, Dafnia magna, 48 horas, Unidades de Toxicidad, en cuerpos loticos
39. **CALIDAD_TOX_D_48_FON**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador, Toxicidad, Dafnia magna 48 horas, en fondo
40. **TOX_FIS_SUP_15_UT**: numérica Valor de Toxicidad, Vibrio Fisher, 15 minutos, superficial, Unidades de Toxicidad
41. **CALIDAD_TOX_FIS_SUP_15**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador de Toxicidad, Vibrio Fisher, 15 minutos, superficial
42. **TOX_FIS_FON_15_UT**: numérica Valor de Toxicidad, Vibrio Fisher, 15 minutos, en fondo, Unidades de Toxicidad
43. **CALIDAD_TOX_FIS_FON_15**: categórica Clasificacion de la calidad del agua de acuerdo con el indicador Toxicidad, Vibrio Fisher, 15 minutos, en fondo
44. **SEMAFORO**: categórica Indica el nivel de contaminacion de acuerdo a los contaminantes presentes

45. **CONTAMINANTES:** string Contaminantes presentes en incumplimiento (Contaminados).
46. **CUMPLE_CON_DBO:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente, Buena calidad o Aceptable, para el Indicador Demanda Bioquimica de Oxigeno
47. **CUMPLE_CON_DQO:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente, Buena calidad o Aceptable, para el Indicador Demanda Quimica de Oxigeno
48. **CUMPLE_CON_SST:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente, Buena calidad o Aceptable, para el Indicador Solidos Suspendidos Totales
49. **CUMPLE_CON_CF:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente, Buena calidad o Aceptable, para el Indicador Coliformes Fecales
50. **CUMPLE_CON_E_COLI:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente, Buena calidad o Aceptable, para el Indicador Escherichia coli
51. **CUMPLE_CON_ENTEROC:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente o Buena calidad, para el Indicador Enterococos fecales
52. **CUMPLE_CON_OD:** binaria sobre si cumple criterio de Indica si cumple con la calidad de Excelente, Buena calidad o Aceptable, para el Indicador Porcentaje de saturaci n de Oxigeno disuelto
53. **CUMPLE_CON_TOX:** binaria sobre si cumple criterio de Indica si cumple con la calidad de No toxic, Toxicidad baja, o Toxicidad moderada, para el Indicador Toxicidad aguda
54. **GRUPO:** categ rico Grupo del cuerpo de agua

Con base en la definici n de cada atributo, no nos son de utilidad algunos para el desarrollo del modelo, como lo son:

- Clave
- Sitio
- ORGANISMO_DE_CUENCA
- ESTADO
- MUNICIPIO
- CUENCA
- CUERPO DE AGUA
- TIPO
- SUBTIPO
- PERIODO
- CONTAMINANTES

```
eliminar = ['CLAVE', 'SITIO', 'ORGANISMO_DE_CUENCA', 'ESTADO', 'MUNICIPIO', 'CUENCA', 'CUERPO DE AG  
df1.drop(eliminar, inplace = True, axis =1)  
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3493 entries, 0 to 3492
Data columns (total 40 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   LONGITUD         3493 non-null    float64 
 1   LATITUD          3493 non-null    float64 
 2   DBO_mg/L         2581 non-null    object  
 3   CALIDAD_DBO      2581 non-null    object  
 4   DQO_mg/L         2581 non-null    object  
 5   CALIDAD_DQO      2581 non-null    object  
 6   SST_mg/L         3489 non-null    object  
 7   CALIDAD_SST      3489 non-null    object  
 8   COLI_FEC_NMP_100mL 2582 non-null    object  
 9   CALIDAD_COLI_FEC 2582 non-null    object  
 10  E_COLI_NMP_100mL 2582 non-null    object  
 11  CALIDAD_E_COLI    2582 non-null    object  
 12  ENTEROC_NMP_100mL 904 non-null    object  
 13  CALIDAD_ENTEROC   904 non-null    object  
 14  OD_PORC          1797 non-null    object  
 15  CALIDAD_OD_PORC   1797 non-null    object  
 16  OD_PORC_SUP       1619 non-null    object  
 17  CALIDAD_OD_PORC_SUP 1619 non-null    object  
 18  OD_PORC_MED       487 non-null    object  
 19  CALIDAD_OD_PORC_MED 487 non-null    object  
 20  OD_PORC_FON       946 non-null    object  
 21  CALIDAD_OD_PORC_FON 946 non-null    object  
 22  TOX_D_48_UT        1816 non-null    object  
 23  CALIDAD_TOX_D_48    1816 non-null    object  
 24  TOX_V_15_UT        1819 non-null    object  
 25  CALIDAD_TOX_V_15    1819 non-null    object  
 26  TOX_D_48_SUP_UT    762 non-null    object  
 27  CALIDAD_TOX_D_48_SUP 762 non-null    object  
 28  TOX_FIS_SUP_15_UT  1674 non-null    object  
 29  CALIDAD_TOX_FIS_SUP_15 1674 non-null    object  
 30  SEMAFORO          3493 non-null    object  
 31  CUMPLE_CON_DBO     3493 non-null    object  
 32  CUMPLE_CON_DQO     3493 non-null    object  
 33  CUMPLE_CON_SST     3493 non-null    object  
 34  CUMPLE_CON_CF      3493 non-null    object  
 35  CUMPLE_CON_E_COLI   3493 non-null    object  
 36  CUMPLE_CON_ENTEROC 3493 non-null    object  
 37  CUMPLE_CON_OD      3493 non-null    object  
 38  CUMPLE_CON_TOX     3493 non-null    object  
 39  GRUPO              3493 non-null    object 

dtypes: float64(2), object(38)
memory usage: 1.1+ MB
```

df1.head()

	LONGITUD	LATITUD	DBO_mg/L	CALIDAD_DBO	DQO_mg/L	CALIDAD_DQO	SST_mg/L	CALIDAD_SST
0	-102.33911	22.24730	6	Buena calidad	54.08	Contaminada	13.75	Excelente
1	-109.84290	22.90473	NaN	NaN	NaN	NaN	<10	Excelente
2	-109.86442	22.89880	NaN	NaN	NaN	NaN	<10	Excelente
3	-109.88604	22.89609	NaN	NaN	NaN	NaN	13.9667	Excelente

Algunos valores corresponden a evaluaciones métricas, así que con el fin de poder generar un análisis automatizado, se cambiarán a números.

```
Var_conc = ['DBO_mg/L', 'DQO_mg/L', 'SST_mg/L', 'COLI_FEC_NMP_100mL', 'E_COLI_NMP_100mL', 'ENTEROC_100CFU', 'OD_PORC_FON', 'TOX_D_48_UT', 'TOX_V_15_UT', 'TOX_D_48_SUP_UT', 'TOX_FIS_SUP_15_UT']
```

```
lista = list()
for lis in Var_conc:
    df1[lis] = df1[lis].astype('str')
    df1[lis] = df1[lis].str.replace('<1', '1')
    df1[lis] = df1[lis].str.replace('<2', '2')
    df1[lis] = df1[lis].str.replace('<3', '3')
    df1[lis] = df1[lis].str.replace('<10', '10')
    df1[lis] = df1[lis].astype('float')

cumplir = ["CUMPLE_CON_DBO", "CUMPLE_CON_DQO", "CUMPLE_CON_SST", "CUMPLE_CON_CF", "CUMPLE_CON_E_COLI"]

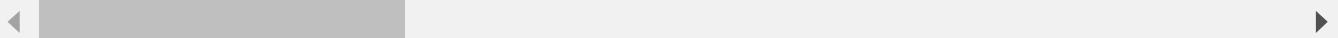
for name in cumplir:
    if name == 'CUMPLE_CON_DBO':
        df1[name]= df1[name].astype('str')
        df1[name]= df1[name].replace('ND', 'SI')
    if name == 'CUMPLE_CON_DQO':
        df1[name]= df1[name].astype('str')
        df1[name]= df1[name].replace('ND', 'SI')
    if name == 'CUMPLE_CON_SST':
        df1[name]= df1[name].astype('str')
        df1[name]= df1[name].replace('ND', 'SI')
    if name == 'CUMPLE_CON_CF':
        df1[name]= df1[name].astype('str')
        df1[name]= df1[name].replace('ND', 'SI')
    if name == 'CUMPLE_CON_E_COLI':
        df1[name]= df1[name].astype('str')
        df1[name]= df1[name].replace('ND', 'SI')
    if name == 'CUMPLE_CON_ENTEROC':
        df1[name]= df1[name].astype('str')
        df1[name]= df1[name].replace('ND', 'SI')
    if name == 'CUMPLE_CON_OD':
        df1[name]= df1[name].astype('str')
```

```
df1[name]= df1[name].replace('ND', 'SI')
if name == 'CUMPLE_CON_TOX':
    df1[name]= df1[name].astype('str')
    df1[name]= df1[name].replace('ND', 'SI')
```

```
df1.describe(include = 'all')
```

	LONGITUD	LATITUD	DBO_mg/L	CALIDAD_DBO	DQO_mg/L	CALIDAD_DQO	%
count	3493.000000	3493.000000	2581.000000		2581	2581.000000	2581 348
unique	Nan	Nan	Nan		5	Nan	5
top	Nan	Nan	Nan	Excelente		Nan	Contaminada
freq	Nan	Nan	Nan		1330	Nan	790
mean	-100.359969	21.046992	16.886481		Nan	64.332985	Nan 10
std	6.122773	3.893696	65.140268		Nan	149.828058	Nan 44
min	-117.124030	14.534910	2.000000		Nan	10.000000	Nan 1
25%	-103.882310	18.396070	2.000000		Nan	11.870000	Nan 1
50%	-99.795530	20.148980	2.630000		Nan	27.010000	Nan 2
75%	-96.860230	22.828930	10.000000		Nan	57.000000	Nan 5
max	-86.732150	32.706500	1500.000000		Nan	2871.250000	Nan 943

11 rows × 40 columns



```
df1.head()
```

	LONGITUD	LATITUD	DBO_mg/L	CALIDAD_DBO	DQO_mg/L	CALIDAD_DQO	SST_mg/L	CALIDAD_
0	-102.33911	22.24730	6.0	Buena calidad	54.08	Contaminada	13.7500	Excelente
1	-109.84290	22.90473	Nan	Nan	Nan	Nan	10.0000	Excelente
2	-109.86442	22.89880	Nan	Nan	Nan	Nan	10.0000	Excelente
3	-109.88604	22.89609	Nan	Nan	Nan	Nan	13.9667	Excelente
4	-109.89657	22.87694	Nan	Nan	Nan	Nan	10.0000	Excelente

5 rows × 40 columns



Para los valores faltantes de los contaminantes, realizaremos la imputación media, pero para no generar un sesgo mayor, realizaremos una división por su clasificación de semáforo de forma previa.

```
ndf_vd = df1[df1.SEMAFORO == 'Verde'].copy()
ndf_am = df1[df1.SEMAFORO == 'Amarillo'].copy()
ndf_rj = df1[df1.SEMAFORO == 'Rojo'].copy()
ndf_var = [ndf_vd, ndf_am, ndf_rj]
```

Ahora realizaremos la imputación media comenzando con las Potables, o sea las verdes.

```
vd = list()
for color in Var_conc:
    promedio = ndf_vd[color].mean()
    ndf_vd[color] = ndf_vd[color].replace(np.nan, promedio)
    vd.append([color, promedio])

ndf_vd.describe()
```

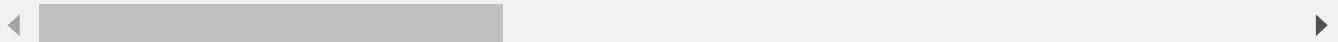
	LONGITUD	LATITUD	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100m
count	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000
mean	-100.181832	21.397310	3.372192	18.241230	24.810056	281.205881
std	7.865871	4.472317	2.026717	5.873656	21.477673	180.914638
min	-117.124030	14.541280	2.000000	10.000000	10.000000	3.000000
25%	-105.523060	17.951895	2.000000	17.641000	10.000000	230.000000
50%	-99.368000	20.686420	3.372192	18.241230	16.000000	281.205881
75%	-93.848050	24.147350	3.372192	18.241230	31.660000	281.205881
max	-86.732150	32.706500	21.500000	40.000000	147.000000	991.000000

Repetimos el procedimiento con las amarillas:

```
am = list()
for color in Var_conc:
    ndf_am[color] = ndf_am[color].replace(np.nan, promedio)
    am.append([color, promedio])
```

```
ndf_am.describe()
```

	LONGITUD	LATITUD	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100mL
count	1135.000000	1135.000000	1135.000000	1135.000000	1135.000000	1135.000000
mean	-100.409841	21.063824	3.734693	18.100690	93.376443	11736.058150
std	4.888005	3.609725	3.783072	9.529662	385.109968	24779.728296
min	-115.498450	14.534910	1.000000	1.000000	10.000000	1.000000
25%	-103.890050	18.298855	2.000000	10.000000	10.000000	1650.000000
50%	-99.800420	20.152120	2.000000	15.190000	23.400000	3255.000000
75%	-97.134095	23.703455	4.080000	25.200000	68.000000	24000.000000
max	-86.805810	32.663990	26.110000	40.000000	9408.000000	241960.000000

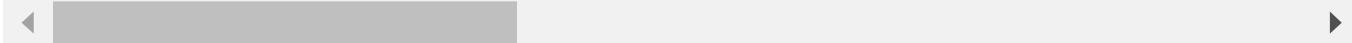


Finalmente con el semáforo rojo.

```
rj = list()
for color in Var_conc:
    promedio = ndf_rj[color].mean()
    ndf_rj[color] = ndf_rj[color].replace(np.nan, promedio)
    rj.append([color, promedio])
```

```
ndf_rj.describe()
```

	LONGITUD	LATITUD	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100mL
count	1091.000000	1091.000000	1091.000000	1091.000000	1091.000000	1.091000e+01
mean	-100.514960	20.622648	40.030949	144.593989	201.451951	2.490353e+01
std	4.828816	3.385247	96.436763	210.479424	673.105512	1.789631e+01
min	-117.107890	14.615670	2.000000	14.400000	10.000000	3.000000e+00
25%	-102.274535	18.970835	6.475000	55.305000	20.000000	8.450000e+00
50%	-99.923030	19.902140	18.000000	91.000000	44.000000	2.400000e+00
75%	-98.173810	20.986190	40.030949	144.593989	126.000000	4.747500e+00
max	-88.170970	32.666080	1500.000000	2871.250000	9430.000000	2.419600e+01



Ahora es necesario conocer las escalas para cada calidad, por lo que tomaremos los valores de referencia del documento proporcionado:

```
calidad_ref = pd.read_csv('https://raw.githubusercontent.com/PosgradoMNA/actividades-del-proj  
calidad_ref
```

	CALIDAD DEL AGUA PARA TOXICIDAD	CRITERIO	DESCRIPCION
0	No tóxico	TOX menor a 1	Agua no contaminada. Toxicidad no detectable.
1	Toxicidad baja	TOX mayor o igual a 1 y menor o igual a 1.33	Toxicidad baja
2	Toxicidad moderada	TOX mayor de 1.33 y menor a 5	Toxicidad moderada
3	Toxicidad alta	TOX mayor o igual a 5	Toxicidad alta
4	CALIDAD DEL AGUA PARA SST	CRITERIO	DESCRIPCION
5	Excelente	SST menor o igual a 25	Clase de excepción, muy buena calidad.
6	Buena calidad	SST mayor de 25 y menor o igual a 75	Aguas superficiales con bajo contenido de soli...
7	Aceptable	SST mayor de 75 y menor o igual a 150	Aguas superficiales con indicio de contaminaci...
8	Contaminada	SST mayor de 150 y menor o igual a 400	Aguas superficiales de mala calidad con descarr...
9	Fuertemente contaminada	SST mayor de 400	Aguas superficiales con fuerte impacto de descarr...
10	CALIDAD DEL AGUA PARA ENTEROCOCCOS FECALES	CRITERIO	DESCRIPCION

```
calidades = ['CALIDAD_DBO', 'CALIDAD_DQO', 'CALIDAD_SST', 'CALIDAD_COLI_FEC', 'CALIDAD_E_COLI', 'CALIDAD_OD_PORC_FON', 'CALIDAD_TOX_D_48', 'CALIDAD_TOX_V_15', 'CALIDAD_TOX_D_48_SU']
```

11 Aguas superficiales con alta calidad

Los valores de referencia no se encuentran en un formato que permita su evaluación de forma automatizada, por lo que los valores se comparan de forma manual.

```
ndf_cal_ver = ndf_vd.loc[:,['DBO_mg/L', 'DQO_mg/L', 'SST_mg/L', 'COLI_FEC_NMP_100mL', 'E_COLI_NMP', 'OD_PORC_FON', 'TOX_D_48_UT', 'TOX_V_15_UT', 'TOX_D_48_SUP_UT', 'TOX_FIS_SUP_15_UT']]
ndf_cal_ver.describe(include='all')
```

	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100mL	E_COLI_NMP_100mL	ENTEI
count	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	
mean	3.372192	18.241230	24.810056	281.205882	91.069853	
std	2.026717	5.873656	21.477673	180.914635	89.053108	
min	2.000000	10.000000	10.000000	3.000000	3.000000	

Con la comparación manual tenemos las siguientes escalas para los verdes:

- DBO_mg/L Buena Calidad
- DQO_mg/L Buena Calidad
- SST_mg/L Excelente
- COLI_FEC_NMP_100mL Aceptable
- E_COLI_NMP_100mL Excelente
- ENTEROC_NMP_100mL Excelente
- OD_PORC Excelente
- OD_PORC_SUP Excelente
- OD_PORC_MED Excelente
- OD_PORC_FON Excelente
- TOX_D_48_UT Toxicidad Baja
- TOX_V_15_UT Toxicidad Baja
- TOX_D_48_SUP_UT Toxicidad Baja
- TOX_D_48_FON_UT NaN
- TOX_FIS_SUP_15_UT Toxicidad Baja
- TOX_FIS_FON_15_UT NaN

```
for name in calidades:
    if name == 'CALIDAD_DBO':
        ndf_vd[name] = ndf_vd[name].astype('str')
        ndf_vd[name] = ndf_vd[name].replace('nan', 'Buena Calidad')
    if name == 'CALIDAD_DQO':
        ndf_vd[name] = ndf_vd[name].astype('str')
        ndf_vd[name] = ndf_vd[name].replace('nan', 'Buena Calidad')
    if name == 'CALIDAD_SST':
        ndf_vd[name] = ndf_vd[name].astype('str')
        ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
    if name == 'CALIDAD_COLI_FEC':
        ndf_vd[name] = ndf_vd[name].astype('str')
        ndf_vd[name] = ndf_vd[name].replace('nan', 'Aceptable')
    if name == 'CALIDAD_E_COLI':
        ndf_vd[name] = ndf_vd[name].astype('str')
        ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
    if name == 'CALIDAD_ENTEROC':
```

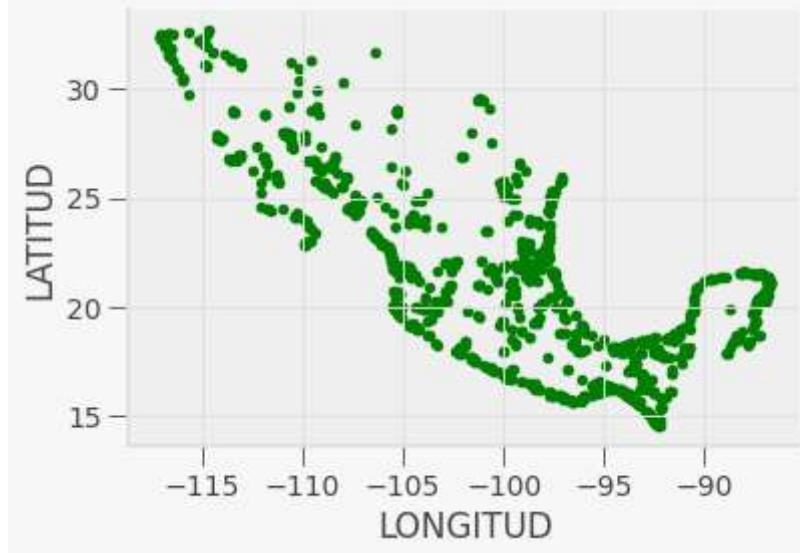
```

ndf_vd[name] = ndf_vd[name].astype('str')
ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC_SUP':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC_MED':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC_FON':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Excelente')
if name == 'CALIDAD_TOX_D_48':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Toxicidad Baja')
if name == 'CALIDAD_TOX_V_15':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Toxicidad Baja')
if name == 'CALIDAD_TOX_D_48_SUP':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Toxicidad Baja')
if name == 'CALIDAD_TOX_FIS_SUP_15':
    ndf_vd[name] = ndf_vd[name].astype('str')
    ndf_vd[name] = ndf_vd[name].replace('nan', 'Toxicidad Baja')

```

```
ndf_vd.plot.scatter('LONGITUD', 'LATITUD', c='green')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3beee9990>
```



Para el caso del semáforo amarillo:

```
ndf_cal_am = ndf_vd.loc[:,['DBO_mg/L','DQO_mg/L','SST_mg/L','COLI_FEC_NMP_100mL','E_COLI_NMP_100mL','OD_PORC_FON','TOX_D_48_UT','TOX_V_15_UT','TOX_D_48_SUP_UT','TOX_FIS_SUP_15_UT']]
ndf_cal_am.describe(include='all')
```

	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100mL	E_COLI_NMP_100mL	ENTEI
count	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	
mean	3.372192	18.241230	24.810056	281.205882	91.069853	
std	2.026717	5.873656	21.477673	180.914635	89.053108	
min	2.000000	10.000000	10.000000	3.000000	3.000000	
25%	2.000000	17.641000	10.000000	230.000000	41.000000	
50%	3.372192	18.241230	16.000000	281.205882	91.069853	
75%	3.372192	18.241230	31.660000	281.205882	91.069853	
max	21.500000	40.000000	147.000000	991.000000	750.000000	

Con la comparación manual tenemos la siguiente escala para los amarillos:

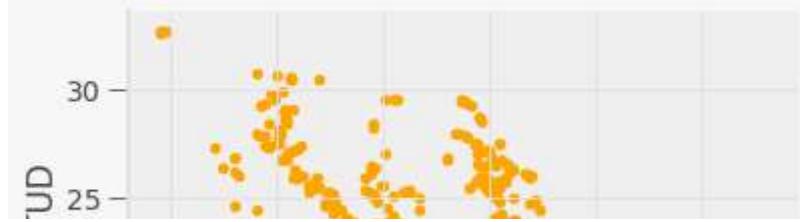
- DBO_mg/L Contaminada
- DQO_mg/L Contaminada
- SST_mg/L Contaminada
- COLI_FEC_NMP_100mL Fuertemente contaminada
- E_COLI_NMP_100mL Fuertemente contaminada
- ENTEROC_NMP_100mL Fuertemente contaminada
- OD_PORC_Aceptable
- OD_PORC_SUP Excelente
- OD_PORC_MED Buena Calidad
- OD_PORC_FON Buena Calidad
- TOX_D_48_UT Toxicidad Baja
- TOX_V_15_UT Toxicidad moderada
- TOX_D_48_SUP_UT Toxicidad Baja
- TOX_D_48_FON_UT NaN
- TOX_FIS_SUP_15_UT Toxicidad Baja
- TOX_FIS_FON_15_UT NaN

```
for name in calidades:
    if name == 'CALIDAD_DBO':
        ndf_am[name] = ndf_am[name].astype('str')
        ndf_am[name] = ndf_am[name].replace('nan', 'Buena Calidad')
    if name == 'CALIDAD_DQO':
```

```
ndf_am[name] = ndf_am[name].astype('str')
ndf_am[name] = ndf_am[name].replace('nan', 'Buena Calidad')
if name == 'CALIDAD_SST':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Aceptable')
if name == 'CALIDAD_COLI_FEC':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Fuertemente contaminada')
if name == 'CALIDAD_E_COLI':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Fuertemente contaminada')
if name == 'CALIDAD_ENTEROC':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC_SUP':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Excelente')
if name == 'CALIDAD_OD_PORC_MED':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Buena Calidad')
if name == 'CALIDAD_OD_PORC_FON':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Buena Calidad')
if name == 'CALIDAD_TOX_D_48':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Toxicidad Baja')
if name == 'CALIDAD_TOX_V_15':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Toxicidad Baja')
if name == 'CALIDAD_TOX_D_48_SUP':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Toxicidad Baja')
if name == 'CALIDAD_TOX_FIS_SUP_15':
    ndf_am[name] = ndf_am[name].astype('str')
    ndf_am[name] = ndf_am[name].replace('nan', 'Toxicidad Baja')

ndf_am.plot.scatter('LONGITUD','LATITUD',c='orange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3befb2850>
```



Para el caso del semáforo rojo:



```
ndf_cal_rj = ndf_vd.loc[:, ['DBO_mg/L', 'DQO_mg/L', 'SST_mg/L', 'COLI_FEC_NMP_100mL', 'E_COLI_NMP_100mL', 'OD_PORC_FON', 'TOX_D_48_UT', 'TOX_V_15_UT', 'TOX_D_48_SUP_UT', 'TOX_FIS_SUP_15_UT']]
ndf_cal_rj.describe(include='all')
```

	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100mL	E_COLI_NMP_100mL	ENTEI
count	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000
mean	3.372192	18.241230	24.810056	281.205882	91.069853	
std	2.026717	5.873656	21.477673	180.914635	89.053108	
min	2.000000	10.000000	10.000000	3.000000	3.000000	
25%	2.000000	17.641000	10.000000	230.000000	41.000000	
50%	3.372192	18.241230	16.000000	281.205882	91.069853	
75%	3.372192	18.241230	31.660000	281.205882	91.069853	
max	21.500000	40.000000	147.000000	991.000000	750.000000	



Con la escala manual tenemos la siguiente escala para los rojos:

- DBO_mg/L Buena Calidad
- DQO_mg/L Buena Calidad
- SST_mg/L Aceptable
- COLI_FEC_NMP_100mL Fuertemente contaminada
- E_COLI_NMP_100mL Fuertemente contaminada
- ENTEROC_NMP_100mL Excelente
- OD_PORC Excelente
- OD_PORC_SUP Excelente
- OD_PORC_MED Buena Calidad
- OD_PORC_FON Buena Calidad
- TOX_D_48_UT Toxicidad Baja
- TOX_V_15_UT Toxicidad Baja
- TOX_D_48_SUP_UT Toxicidad Baja

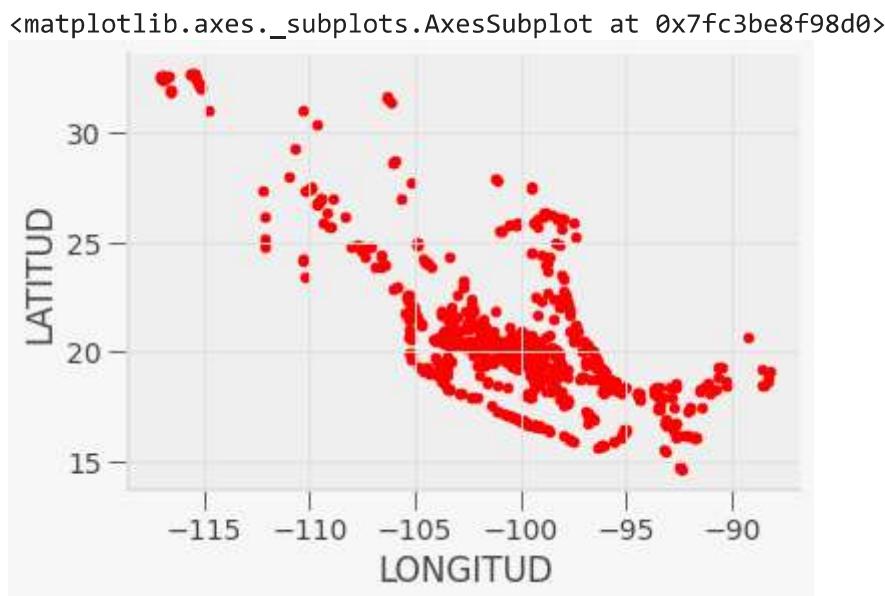
- TOX_D_48_FON_UT NaN
- TOX_FIS_SUP_15_UT Toxicidad Baja
- TOX_FIS_FON_15_UT NaN

```

for name in calidades:
    if name == 'CALIDAD_DBO':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Contaminada')
    if name == 'CALIDAD_DQO':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Contaminada')
    if name == 'CALIDAD_SST':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Contaminada')
    if name == 'CALIDAD_COLI_FEC':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Fuertemente contaminada')
    if name == 'CALIDAD_E_COLI':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Fuertemente contaminada')
    if name == 'CALIDAD_ENTEROC':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Fuertemente contaminada')
    if name == 'CALIDAD_OD_PORC':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Aceptable')
    if name == 'CALIDAD_OD_PORC_SUP':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Excelente')
    if name == 'CALIDAD_OD_PORC_MED':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Buena Calidad')
    if name == 'CALIDAD_OD_PORC_FON':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Buena Calidad')
    if name == 'CALIDAD_TOX_D_48':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Toxicidad Baja')
    if name == 'CALIDAD_TOX_V_15':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Toxicidad Moderada')
    if name == 'CALIDAD_TOX_D_48_SUP':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Toxicidad Baja')
    if name == 'CALIDAD_TOX_FIS_SUP_15':
        ndf_rj[name] = ndf_rj[name].astype('str')
        ndf_rj[name] = ndf_rj[name].replace('nan', 'Toxicidad Baja')

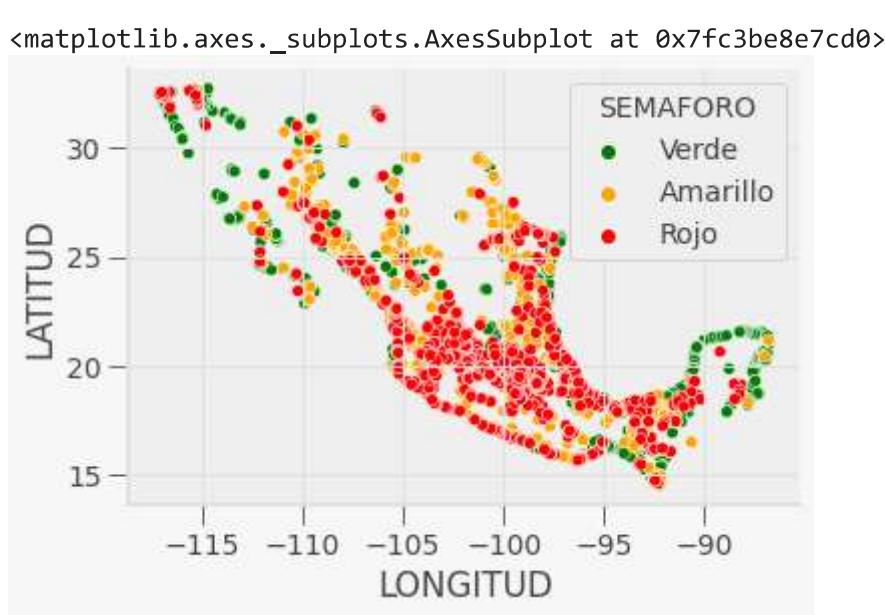
```

```
ndf_rj.plot.scatter('LONGITUD', 'LATITUD', c='red')
```



```
dfproc = pd.concat([ndf_vd, ndf_am, ndf_rj]) #Calidades del agua
```

```
sns.scatterplot(data = dfproc, y="LATITUD", x="LONGITUD", hue="SEMAFORO", palette=['green', 'orange', 'red'])
```



```
dfproc.info()
```

👤 <class 'pandas.core.frame.DataFrame'>
 Int64Index: 3493 entries, 1 to 3490
 Data columns (total 40 columns):

#	Column	Non-Null Count	Dtype
0	LONGITUD	3493 non-null	float64
1	LATITUD	3493 non-null	float64
2	DBO_mg/L	3493 non-null	float64
3	CALIDAD_DBO	3493 non-null	object

```

4 DQO_mg/L           3493 non-null   float64
5 CALIDAD_DQO        3493 non-null   object
6 SST_mg/L           3493 non-null   float64
7 CALIDAD_SST        3493 non-null   object
8 COLI_FEC_NMP_100mL 3493 non-null   float64
9 CALIDAD_COLI_FEC   3493 non-null   object
10 E_COLI_NMP_100mL   3493 non-null   float64
11 CALIDAD_E_COLI    3493 non-null   object
12 ENTEROC_NMP_100mL 3493 non-null   float64
13 CALIDAD_ENTEROC   3493 non-null   object
14 OD_PORC            3493 non-null   float64
15 CALIDAD_OD_PORC   3493 non-null   object
16 OD_PORC_SUP        3493 non-null   float64
17 CALIDAD_OD_PORC_SUP 3493 non-null   object
18 OD_PORC_MED        3493 non-null   float64
19 CALIDAD_OD_PORC_MED 3493 non-null   object
20 OD_PORC_FON        3493 non-null   float64
21 CALIDAD_OD_PORC_FON 3493 non-null   object
22 TOX_D_48_UT         3493 non-null   float64
23 CALIDAD_TOX_D_48    3493 non-null   object
24 TOX_V_15_UT         3493 non-null   float64
25 CALIDAD_TOX_V_15    3493 non-null   object
26 TOX_D_48_SUP_UT    3493 non-null   float64
27 CALIDAD_TOX_D_48_SUP 3493 non-null   object
28 TOX_FIS_SUP_15_UT   3493 non-null   float64
29 CALIDAD_TOX_FIS_SUP_15 3493 non-null   object
30 SEMAFORO            3493 non-null   object
31 CUMPLE_CON_DBO     3493 non-null   object
32 CUMPLE_CON_DQO     3493 non-null   object
33 CUMPLE_CON_SST     3493 non-null   object
34 CUMPLE_CON_CF      3493 non-null   object
35 CUMPLE_CON_E_COLI  3493 non-null   object
36 CUMPLE_CON_ENTEROC 3493 non-null   object
37 CUMPLE_CON_OD      3493 non-null   object
38 CUMPLE_CON_TOX     3493 non-null   object
39 GRUPO               3493 non-null   object

```

dtypes: float64(16), object(24)

memory usage: 1.1+ MB

```
ndf_cal_ver.describe(include='all')
```

	DBO_mg/L	DQO_mg/L	SST_mg/L	COLI_FEC_NMP_100mL	E_COLI_NMP_100mL	ENTEL
count	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000
	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000
	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000	1267.000000

Con lo anterior nuestras variables se encuentran completas.

```
 0.000000  2.020711  3.073000  21.771070  100.814000  05.000100
```

► ENTREGA 2 Kmeans

```
[ ] ↓ 21 celdas ocultas
```

▼ Reto_Entrega_2

Realiza clasificador, con las variables seleccionadas. Toma en consideración si existe un desbalance en las clases y realiza acciones pertinentes (ej. class_weight en clasificador, división de datos con train_test_split).

```
[ ] ↓ 25 celdas ocultas
```

Análisis de resultados con modelo de Decision Trees y Random Forest

- Realiza análisis de exactitudes (accuracies). A través de métricas de exactitud o classification report de scikitlearn.
- Realiza gráfica de confusion matrix

▼ Classification Trees

```
tree_clf = DecisionTreeClassifier(max_depth=4, ccp_alpha=0.01, criterion = 'gini', class_weight='balanced')
tree_clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.01, class_weight='balanced', max_depth=4,
```

```
min_samples_split=4)
```

Se realiza el accuracy

```
y_pred = tree_clf.predict(X_test)
print(tree_clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
DecisionTreeClassifier 1.0
```

```
predicion.append(y_pred)
nombre.append("Decision Tree")
```

```
print(classification_report(y_test, y_pred, labels=[1, 2, 3]))
```

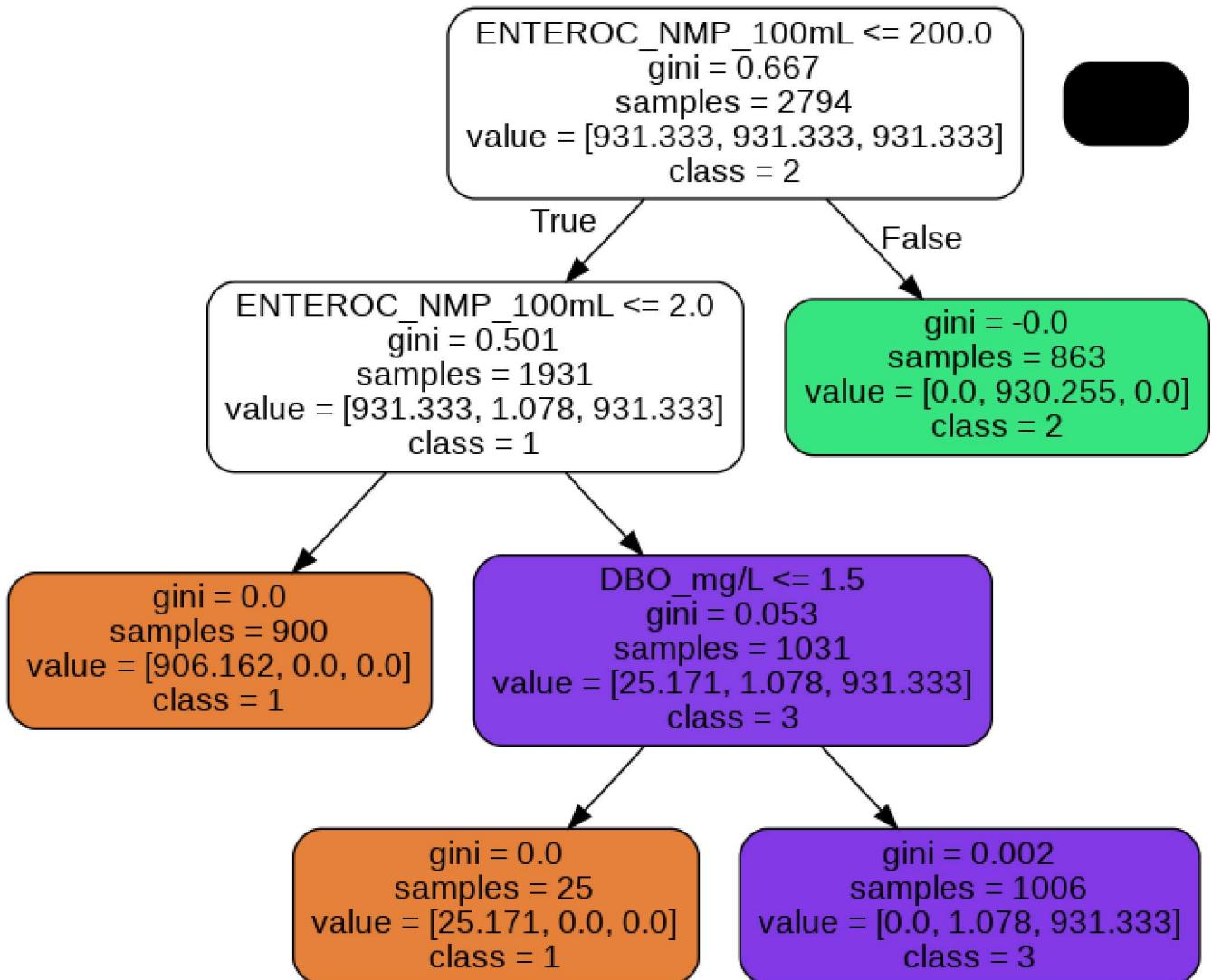
	precision	recall	f1-score	support
1	1.00	1.00	1.00	227
2	1.00	1.00	1.00	262
3	0.00	0.00	0.00	0
micro avg	1.00	1.00	1.00	489
macro avg	0.67	0.67	0.67	489
weighted avg	1.00	1.00	1.00	489

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefined
    _warn_prf(average, modifier, msg_start, len(result))
```

Se obtienen las metricas de desempeño de precision, recall, f1 score

```
dot_data = export_graphviz(tree_clf,
                           out_file=None, #image_path("iris_tree.dot"),
                           feature_names=list(pd.DataFrame(X_train).columns.values),
                           class_names=['1','2','3'],
                           rounded=True,
                           filled=True)
```

```
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



El árbol de decisión generado, nos indica: Si los enterococos fecales son mayores a 200 por cada 100 ml, la calidad del agua será regular (amarillo). Si los enterococos fecales son menores a 2 por cada 100ml la calidad del agua será buena (verde). Si los enterococos fecales son menores a 200 pero mayores a 2 por cada 100 ml, y El oxígeno disuelto es menor a 1.5 , la calidad será buena (verde). Si los enterococos fecales son menores a 200 pero mayores a 2 por cada 100 ml, y el oxígeno disuelto por cada 100 ml es mayor a 1.5 , la calidad será mala (rojo).

► Importancia de las variables

Analiza la importancia de las variables a través de la función feature_importances.

[] ↴ 3 celdas ocultas

▼ Random Forest

```
rnd_clf = RandomForestClassifier(n_estimators=5, max_depth=4, ccp_alpha=0.01,criterion = 'gini')
rnd_clf.fit(X_train, y_train)

RandomForestClassifier(ccp_alpha=0.01, class_weight='balanced', max_depth=4,
                      n_estimators=5)

y_pred = rnd_clf.predict(X_test)
print(rnd_clf.__class__.__name__, accuracy_score(y_test, y_pred))

RandomForestClassifier 1.0

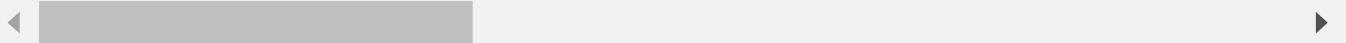
predicion.append(y_pred)
nombre.append("Random Forest")
```

Se realiza el accuracy

```
print(classification_report(y_test, y_pred, labels=[1, 2, 3]))
```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	227
2	1.00	1.00	1.00	262
3	0.00	0.00	0.00	0
micro avg	1.00	1.00	1.00	489
macro avg	0.67	0.67	0.67	489
weighted avg	1.00	1.00	1.00	489

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedVariableWarning: _warn_prf(average, modifier, msg_start, len(result))
```



► Importancia de las variables

Analiza la importancia de las variables a través de la función feature_importances.

[] ↓ 3 celdas ocultas

▼ Realiza gráfica de confusion matrix

```
def mi_cm(yreal, ypred,titulo):
    cm = confusion_matrix(yreal, ypred)
    txt = ['Verdaderos Negativos','Falsos Positivos','Falsos Negativos','Verdaderos Positivos']
    vf = [ '( VN )', '( FP )', '( FN )', '( VP )' ]
    frecuencia = ["{0:0.0f}".format(value) for value in cm.flatten()]
    porcentaje = ["{0:.1%}".format(value) for value in cm.flatten()/np.sum(cm)]

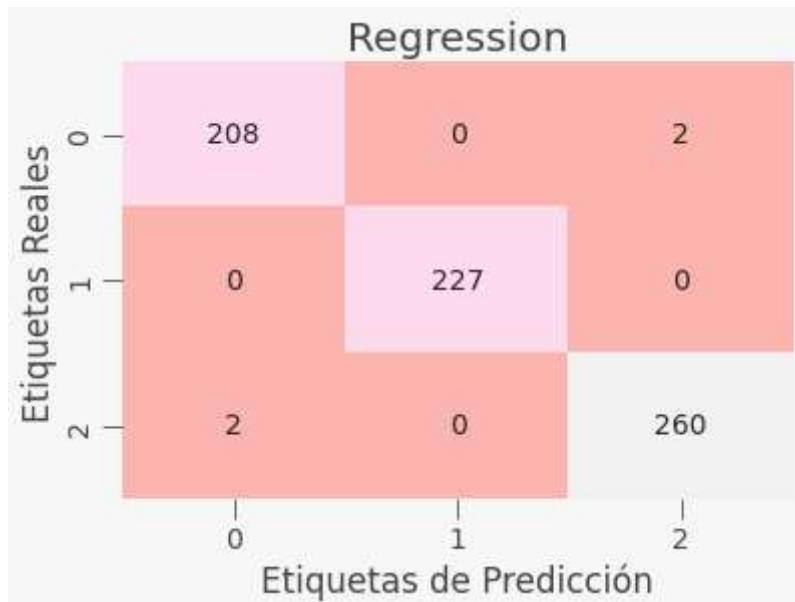
    labels = [f"\n{v1}\n{v2}\n{v3}\n{v4}" for v1, v2, v3, v4 in zip(txt,vf, frecuencia,porcentaje)]
    labels = np.asarray(labels).reshape(2,2)

    plt.figure(figsize=(6,4))
    plt.title(titulo)
    ax = sns.heatmap(cm, annot=True, fmt='', cmap='Pastel1', cbar=False)
    ax.set(ylabel="Etiquetas Reales", xlabel="Etiquetas de Predicción")
    plt.show()

    vn = 100*cm[0,0] / cm.sum()
    fp1 = 100*cm[0,1] / cm.sum()
    fp2 = 100*cm[0,2] / cm.sum()
    fn1 = 100*cm[1,0] / cm.sum()
    fn2 = 100*cm[2,0] / cm.sum()
    fn3 = 100*cm[1,2] / cm.sum()
    fn4 = 100*cm[2,1] / cm.sum()
    vp1 = 100*cm[1,1] / cm.sum()
    vp2 = 100*cm[2,2] / cm.sum()

    print("VN : ",vn)
    print("FP1 : ",fp1)
    print("FP2 : ",fp2)
    print("FN1 : ",fn1)
    print("FN2: ",fn2)
    print("FN3: ",fn3)
    print("FN4: ",fn4)
    print("VP1 : ",vp1)
    print("VP2: ",vp2)

for tupla in zip(predicion,nombre):
    mi_cm(y_test, tupla[0] ,tupla[1] )
```



VN : 29.756795422031473

FP1 : 0.0

FP2 : 0.2861230329041488

FN1 : 0.0

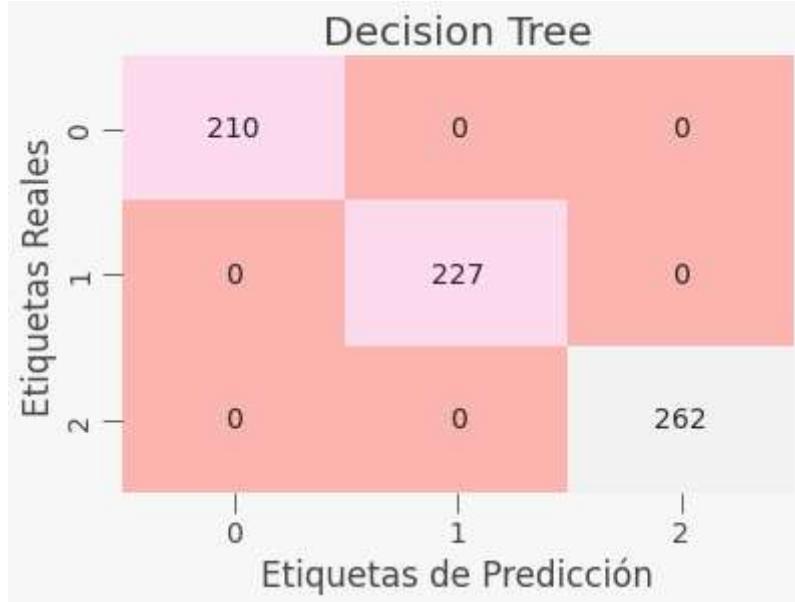
FN2: 0.2861230329041488

FN3: 0.0

FN4: 0.0

VP1 : 32.474964234620884

VP2: 37.19599427753934



VN : 30.042918454935624

FP1 : 0.0

FP2 : 0.0

FN1 : 0.0

FN2: 0.0

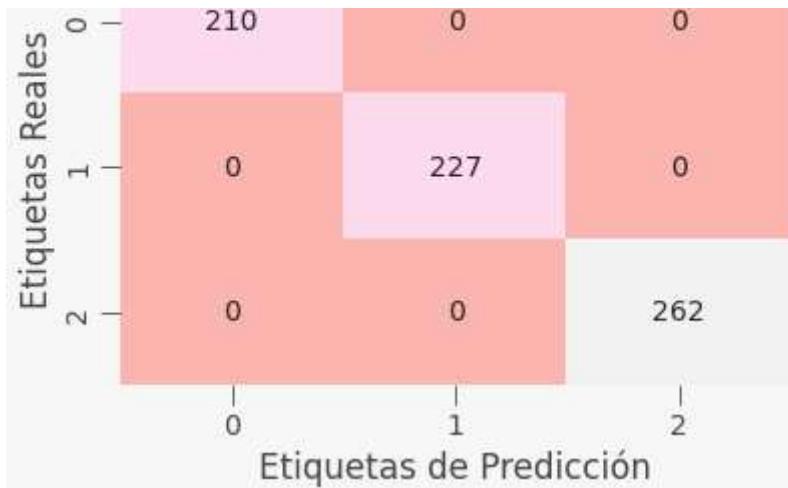
FN3: 0.0

FN4: 0.0

VP1 : 32.474964234620884

VP2: 37.48211731044349





VN : 30.042918454935624

FP1 : 0.0
FP2 : 0.0
FN1 : 0.0
FN2: 0.0
FN3: 0.0

Con los resultados de la matriz de confusión nos indican que el modelo tuvo un resultado sin errores, lo cual no es correcto y nos indica un sobreentrenamiento.

Intentamos realizar el análisis con una clase ya que no tenemos conocimiento aún de como realizar el análisis multiclas, y el ejemplo con el que estamos trabajando considera 3, Verde, Amarillo y Rojo.

▼ Conclusiones

En este segundo reto las conclusiones son:

- Hay una relación de la ubicación respecto a la calidad de agua superficial como se muestra en las graficas elaboradas con kmeans
- La mayoría de aguas superficiales están contaminadas con el color rojo
- El conjunto de datos con el análisis de kmeans está relacionado con la población e industrialización del país y posiblemente una sobre población.
- Al analizar las evaluaciones del decision tree es claro que primero que los enterococos fecales son importantes para la contaminación del agua
- Con el resultado de los tres modelos de clasificación obtenemos métricas de desempeño cercanas al 100%, lo que indica un sobreentrenamiento de la clase positiva, ya que el modelo está aprendiendo de la misma, a pesar del balanceo de las clases sigue con los mismos resultados, entonces se considera una mejora el tratar esto con estrategias de multiclas para disminuir el sobreentrenamiento.
- Para la limpieza de la base de datos eliminamos variables vacías, variables categóricas como ubicación, tipo de cuerpo, etc., para los datos NaN que eran variables continuas separamos