Proyecto Final - Aprendizaje Automático - Diego Estrada

Problema de interés: Calidad de agua en el Río de la Plata

Vamos a trabajar con la carga de los conjuntos de datos del período 2013 al 2024. Luego realizaremos el proceso de ETL (Extracción, Transformación y Carga) para obtener un conjunto de datos unificado, listo para el análisis exploratorio y la aplicación de algoritmos de predicción.

In [3]:
```
import os
import pandas as pd
import unicodedata

#El siguiente metodo para quitar acentos y espacios en blanco de inicio y fin de las cadenas de string para estandarizar caden
def quitar_acentos(texto):
    if isinstance(texto, str):
        return unicodedata.normalize('NFKD', texto).encode('ASCII', 'ignore').decode('ASCII').lower().strip()
    return texto

#Ruta de los datasets
ruta_base = 'C:\\Users\\destrada\\Ciencia_datos_2A1C\\Cookiecutter_Proyecto_Final\\data\\interim\\'

#Verificamos si la ruta existe y cargamos los archivos
if os.path.exists(ruta_base):
    archivos = [  'agc_y_riodelaplata_2013.csv', 'agc_y_riodelaplata_2014.csv', 'agc_y_riodelaplata_2015.csv', 'agc_y_riodelap
                  'agc_y_riodelaplata_2017.csv', 'agc_z_riodelaplata_2018.csv', 'agc_z_riodelaplata_2019.csv', 'agc_z_riodelap
                  'agc_z_riodelaplata_2021.csv', 'agc_y_riodelaplata_2022.csv', 'agc_y_riodelaplatal_2023.csv','agc_z_riodelap

    dataframes = []
    for archivo in archivos:
        df = pd.read_csv(ruta_base + archivo, encoding='latin1', sep=';')
        #Columnas en minúsculas
        df.columns = map(str.lower, df.columns)

        #Aplicar quitar_acentos solo a columnas de texto
        for col in df.select_dtypes(include=['object']).columns:
```

```
            df[col] = df[col].apply(quitar_acentos)

        dataframes.append(df)

    #Unificamos todos los datasets en uno solo
    df_final = pd.concat(dataframes, ignore_index=True)

#Mostramos las primeras 5 filas
df_final.head()

#esto esta a aprueba para entender ahora porque la columna calidad de agua tiene filas nulas.
ruta_guardado = r'C:\Users\destrada\Ciencia_datos_2A1C\Cookiecutter_Proyecto_Final\data\processed\df_final_Unificado.csv'
#Guardamos el DataFrame en formato CSV
df_final.to_csv(ruta_guardado, index=False)
```

In [4]:
```
#Queremos conocer la cantidad de registros en el DataFrame Unificado
num_registros = df_final.shape[0]
print("Cantidad de registros: " + str(num_registros))

#Cantidad de columnas
num_columnas = df_final.shape[1]
print("Cantidad de columnas: " + str(num_columnas))
```

```
Cantidad de registros: 1622
Cantidad de columnas: 31
```

Preprocesamiento de Datos (Data Preprocessing)

amos a trabajar en el proceso de ETL (Extracción, Transformación y Carga), durante el cual eliminaremos columnas irrelevantes, trataremos valores nulos y errores, y codificaremos las variables categóricas. Todo este proceso tiene como objetivo preparar los datos adecuadamente para asegurar el correcto funcionamiento del algoritmo de predicción.

In [6]:
```
#Validamos si tenemos valores nulos en cada columnas
valores_nulos = df_final.isnull().sum()
print(valores_nulos)
```

```
orden                       0
sitios                      0
codigo                      0
fecha                       3
año                         0
campaña                     0
tem_agua                  312
tem_aire                  932
od                        465
ph                        316
olores                    796
color                     801
espumas                   798
mat_susp                  798
colif_fecales_ufc_100ml   362
escher_coli_ufc_100ml     382
enteroc_ufc_100ml         826
nitrato_mg_l              245
nh4_mg_l                  246
p_total_l_mg_l            768
fosf_ortofos_mg_l         864
dbo_mg_l                  711
dqo_mg_l                  331
turbiedad_ntu             568
hidr_deriv_petr_ug_l      741
cr_total_mg_l             789
cd_total_mg_l             784
clorofila_a_ug_l          729
microcistina_ug_l         802
ica                       430
calidad_de_agua           436
dtype: int64
```

Vamos a eliminar los valores nulos de la columna objetivo 'calidad_de_agua', ya que completar estos datos con métodos alternativos podría introducir errores de categorización que afecten negativamente el desempeño de los modelos de predicción.

In [8]:
```
#Eliminamos los valores Null
df_final = df_final[df_final['calidad_de_agua'].notna() & df_final['ica'].notna()]
```

Luego de revisar los valores nulos, procederemos a eliminar símbolos y cadenas de texto que fueron cargadas incorrectamente. Este paso nos permitirá estandarizar y dar formato adecuado a cada columna.

In [10]:
```
import numpy as np

#Reemplazamos el símbolo '<' por una cadena vacía en todo el DataFrame
df_final = df_final.replace(r'<', '', regex=True)

#Reemplazamos ciertas cadenas por NaN que luego serán tratados
frases_a_reemplazar = ['-', 'nd', 'no miden','falto un frasco']
df_final = df_final.replace(frases_a_reemplazar, np.nan)
```

In [11]:
```
#Vamos a eliminar filas duplicadas, esto incluye quitar todas las filas que sean completamente vacias y repetidas.
df_final = df_final.drop_duplicates()
```

In [12]:
```
#Vamos a quitar columnas que no serán relevante para el modelo
columnas_a_eliminar = ['orden', 'sitios', 'codigo', 'fecha', ]

#Aplicamos la eliminación
df_final = df_final.drop(columns=columnas_a_eliminar)
```

Luego de eliminar las columnas que no aportaban información relevante para el modelo predictivo, se identificaron columnas que contienen valores numéricos de tipo flotante. Para garantizar el correcto funcionamiento del modelo, es necesario tratar los valores nulos, reemplazándolos por la mediana correspondiente de cada una. Esta estrategia permite conservar la distribución de los datos sin verse afectada por valores extremos.

In [14]:
```
#Con este ciclo, iteramos para que cada columna, reemplazamos valores y calcule la mediana
for col in ['od', 'ph', 'tem_agua','escher_coli_ufc_100ml','nitrato_mg_l','dqo_mg_l','dbo_mg_l','cr_total_mg_l',
            'colif_fecales_ufc_100ml','fosf_ortofos_mg_l','hidr_deriv_petr_ug_l','tem_aire','enteroc_ufc_100ml','nh4_mg_l','p_
           'turbiedad_ntu','cd_total_mg_l','clorofila_a_ug_l','microcistina_ug_l']:
    df_final[col] = pd.to_numeric(df_final[col], errors='coerce')
    df_final[col] = df_final[col].fillna(df_final[col].median())
```

Luego procederemos a corregir los valores nulos presentes en las columnas categóricas. Utilizaremos la "moda" de cada columna que seria el valor más frecuente, como estrategia para completar los datos faltantes de manera coherente con la distribución original.

In [16]:
```
#Columnas que queremos procesar
columnas = ['olores', 'color', 'espumas', 'mat_susp']

#Reemplazamos valores nulos por la moda de cada columna
for columna in columnas:
    #Obtenemos la moda (valor más frecuente)
    moda = df_final[columna].mode().iloc[0]
    df_final[columna] = df_final[columna].fillna(moda)
```

In [17]:
```
#Eliminamos los valores Null
df_final = df_final[df_final['calidad_de_agua'].notna()]
```

In [18]:
```
#Volvemos a validar si tenemos valores nulos en cada columna
valores_nulos = df_final.isnull().sum()
print("Valores nulos por columna:\n", valores_nulos)

#Mostramos el tipo de dato de cada columna
tipos_de_dato = df_final.dtypes
print("\nTipo de dato por columna:\n", tipos_de_dato)
```

```
Valores nulos por columna:
 año                        0
campaña                     0
tem_agua                    0
tem_aire                    0
od                          0
ph                          0
olores                      0
color                       0
espumas                     0
mat_susp                    0
colif_fecales_ufc_100ml     0
escher_coli_ufc_100ml       0
enteroc_ufc_100ml           0
nitrato_mg_l                0
nh4_mg_l                    0
p_total_l_mg_l              0
fosf_ortofos_mg_l           0
dbo_mg_l                    0
dqo_mg_l                    0
turbiedad_ntu               0
hidr_deriv_petr_ug_l        0
cr_total_mg_l               0
cd_total_mg_l               0
clorofila_a_ug_l            0
microcistina_ug_l           0
ica                         0
calidad_de_agua             0
dtype: int64

Tipo de dato por columna:
 año                          int64
campaña                     object
tem_agua                   float64
tem_aire                   float64
od                         float64
ph                         float64
olores                      object
color                       object
espumas                     object
mat_susp                    object
```

```
colif_fecales_ufc_100ml    float64
escher_coli_ufc_100ml      float64
enteroc_ufc_100ml          float64
nitrato_mg_l               float64
nh4_mg_l                   float64
p_total_l_mg_l             float64
fosf_ortofos_mg_l          float64
dbo_mg_l                   float64
dqo_mg_l                   float64
turbiedad_ntu              float64
hidr_deriv_petr_ug_l       float64
cr_total_mg_l              float64
cd_total_mg_l              float64
clorofila_a_ug_l           float64
microcistina_ug_l          float64
ica                         object
calidad_de_agua             object
dtype: object
```

Siguiendo con el tratamiento de los datos, el siguiente paso consiste en convertir las variables categóricas a formato numérico. Esta transformación es esencial, ya que los algoritmos de aprendizaje automático requieren datos numéricos para operar correctamente y alcanzar un rendimiento óptimo.

Dentro de las columnas categóricas "olores", "color", "espumas" y "mat_susp" se detectaron valores como "ausente", "ausenca" y "ausencia", los cuales serán necesario estandarizar antes de ser codificados numéricamente.

In [20]:
```
#Columnas categóricas a procesar
columnas_categoricas = ['olores', 'color', 'espumas', 'mat_susp']

#Reemplazamos "ausencia" y "ausenca" por "ausente" en las columnas categóricas
for columna in columnas_categoricas:
    df_final[columna] = df_final[columna].replace("ausencia", "ausente")
    df_final[columna] = df_final[columna].replace("ausenca", "ausente")
    df_final[columna] = df_final[columna].replace("presencia", "presente")
```

Ahora vamos a binarizar las columnas categoricas 'olores', 'color', 'espumas' y 'mat_susp'

In [22]:
```
#Convertimos las columnas binarias 'presente'/'ausente' a 1 y 0
columnas_binarias = ['olores', 'color', 'espumas', 'mat_susp']
```

```
for columna in columnas_binarias:
    #Esta es otra forma de reemplazar los valores
    df_final[columna] = df_final[columna].str.lower().map({'presente': 1, 'ausente': 0})
```

In [23]:
```
#Aplicamos el método One-Hot Encoding a la columna 'campaña', porque es una columna categóricas nominales y tiene las estacion
#Invierno, Otoño, Primavera)
df_final = pd.get_dummies(df_final, columns=['campaña'], drop_first=False)

#Verificamos las creación de las nuevas columnas (campaña_invierno, campaña_otono, campaña_primavera y campaña_verano)
print(df_final.head(3))
```

```
    año  tem_agua  tem_aire   od   ph  olores  color  espumas  mat_susp  \
0  2013      10.3      14.5  0.7  7.9       0      0        0         0
1  2013      10.5      14.5  0.5  7.5       0      0        0         0
2  2013      10.6      14.5  0.5  7.5       0      0        0         0

   colif_fecales_ufc_100ml  ...  cr_total_mg_l  cd_total_mg_l  \
0                    130.0  ...          0.006          0.002
1                    490.0  ...          0.006          0.002
2                     34.8  ...          0.006          0.002

   clorofila_a_ug_l  microcistina_ug_l   ica             calidad_de_agua  \
0              10.0                0.5  62.0             muy deteriorada
1              10.0                0.5  50.0             muy deteriorada
2              10.0                0.5  27.0  extremadamente deteriorada

   campaña_invierno  campaña_otono  campaña_primavera  campaña_verano
0              True          False              False           False
1              True          False              False           False
2              True          False              False           False

[3 rows x 30 columns]
```

In [24]:
```
#Convertimos las columnas campañas a binarias 'True=1, False=0
columnas_binarias = ['campaña_invierno', 'campaña_otono', 'campaña_primavera', 'campaña_verano']

for columna in columnas_binarias:
    df_final[columna] = df_final[columna].astype(int)
```

Aplicamos el método Label-Encoding para la variable objetivo calidad_de_agua, levemente deteriorada = 0, deteriorada = 1, muy deteriorada = 2, y extremadamente deteriorada = 3

In [26]:
```
#Aplicamos Label - Encoding para la columna calidad_de_agua
mapa_calidad = {
    'levemente deteriorada': 0,
    'deteriorada': 1,
    'muy deteriorada': 2,
    'extremadamente deteriorada': 3
}

df_final['calidad_de_agua'] = df_final['calidad_de_agua'].map(mapa_calidad)
```

In [27]:
```
#Queremos conocer la cantidad de registros en el DataFrame Unificado y como quedo con todo el tratamiento que realizamos
num_registros = df_final.shape[0]
print("Cantidad de registros: " + str(num_registros))

#Cantidad de columnas
num_columnas = df_final.shape[1]
print("Cantidad de columnas: " + str(num_columnas))

#Definimos nueva ruta dentro de la estructura de CookieCutter
ruta_guardado = r'C:\Users\destrada\Ciencia_datos_2A1C\Cookiecutter_Proyecto_Final\data\processed\df_final_Unificado.csv'

#Guardamos el DataFrame en formato CSV
df_final.to_csv(ruta_guardado, index=False)

print("El archivo fue guardado correctamente en:", ruta_guardado)
```

```
Cantidad de registros: 1182
Cantidad de columnas: 30
El archivo fue guardado correctamente en: C:\Users\destrada\Ciencia_datos_2A1C\Cookiecutter_Proyecto_Final\data\processed\df_final_Unificado.csv
```

Conclusion: El proceso de carga y tratamiento del conjunto de datos sobre la calidad del agua en el Río de la Plata, correspondiente al período 2013-2024, permitió consolidar una base de datos unificada, limpia, estructurada y lista para su análisis. Durante este trabajo se abordaron problemas comunes en los datos, aplicando técnicas como la estandarización de formatos, imputación de valores nulos, eliminación de duplicados y codificación de variables categóricas. Estas acciones no solo mejoraron la calidad del dataset, sino que también sentaron las

bases para la aplicación efectiva de modelos de aprendizaje automático orientados a predecir y monitorear el estado del agua en la región. Como resultado, se generó un dataset final compuesto por 1.182 registros y 30 variables, preparado para futuros análisis exploratorios y predictivos.