Tecnológico de Monterrey

RETO PARTE 2 – Clasificación-ensambles y presentación ejecutiva

CIENCIA Y ANALITICA DE DATOS

Profesor: María de La Paz Rico

EQUIPO 12

Guillermo Alfonso Muñiz Hermosillo - A01793101

José Ramiro Adán Charles - A00174646

ENLACE COLLAB:

https://colab.research.google.com/drive/1slTP7eW-ttA_flQZhqa2go3xnFCSDQc7?usp=sharing

ENLACE GITHUB NOTEBOOK:

https://github.com/PosgradoMNA/actividades-del-projecto-cad_equipo_12/blob/main/Reto/Parte2/Reto_Parte2.ipynb

ENLACE GITHUB PRESENTACION EJECUTIVA:

https://github.com/PosgradoMNA/actividades-del-projecto-cad_equipo_12/blob/main/Reto/Parte2/Presentacion%20Ejecutiva%20Reto2.pptx

NOTEBOOK

RETO - PARTE 2

CIENCIA Y ANALITICA DE DATOS

EQUIPO 12

Guillermo Alfonso Muñiz Hermosillo - A01793101

José Ramiro Adán Charles - A00174646

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, cross_validate, GridSearc
from sklearn.linear_model import LinearRegression, Ridge, Lasso


from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz


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

import matplotlib.pyplot as plt

import seaborn as sns
sns.set()

import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

In [2]:
```
from pathlib import Path

IMAGES_PATH = Path() / "images" / "decision_trees"
IMAGES_PATH.mkdir(parents=True, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = IMAGES_PATH / f"{fig_id}.{fig_extension}"
```

```
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

In [3]:
```
plt.rc('font', size=14)
plt.rc('axes', labelsize=14, titlesize=14)
plt.rc('legend', fontsize=14)
plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)
```

In [4]:
```
df = pd.read_csv('AguasSubterraneas_Clean.csv')

print('Shape de Datos de Aguas Subterraneas', df.shape)
```

```
Shape de Datos de Aguas Subterraneas (1068, 53)
```

In [5]:
```
numericasSubte = ['ALC_mg/L','CONDUCT_mS/cm','SDT_M_mg/L','FLUORUROS_mg/L','DUR
points = ['LONGITUD', 'LATITUD']
NASubte = ['CLAVE','SITIO','PERIODO','SDT_mg/L']
```

In [6]:
```
df[['SEMAFORO']].head()
```

Out[6]:

	SEMAFORO
0	Verde
1	Verde
2	Rojo
3	Verde
4	Rojo

In [7]:
```
le = preprocessing.LabelEncoder()
le.fit(df['SEMAFORO'])
classes = le.classes_

classes
```

Out[7]:
```
array(['Amarillo', 'Rojo', 'Verde'], dtype=object)
```

In [8]:
```
x = le.transform(df['SEMAFORO'])
df['SEMAFORO'] = x
df[['SEMAFORO']].head()
```

Out[8]:

	SEMAFORO
0	2
1	2
2	1
3	2
4	1

In [9]:
```
y = df['SEMAFORO']
```

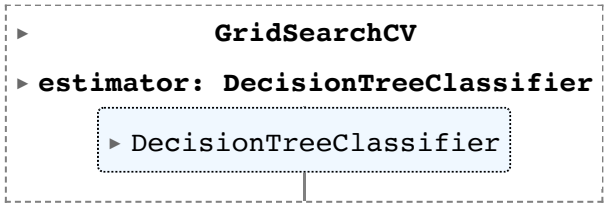
```
X = df[numericasSubte]

X_train, Xtest, y_train, ytest = train_test_split(X, y, train_size=0.75, random

print("X Entrenamiento-Validacion", X_train.shape)
print("X Prueba", Xtest.shape)

print("Y Entrenamiento", y_train.shape)
print("Y Prueba", ytest.shape)
```

```
X Entrenamiento-Validacion (801, 14)
X Prueba (267, 14)
Y Entrenamiento (801,)
Y Prueba (267,)
```

DECISION TREE

In [10]:
```
#HACEMOS UNA ITERACION PARA ENCONTRAR LOS MEJORES VALORES ENTRE POSIBLE PARAMET
params = {
    'max_leaf_nodes': list(range(2, 100)),
    'max_depth': list(range(1, 7)),
    'min_samples_split': [2, 3, 4]
}
modelo=DecisionTreeClassifier(random_state=42)
grid_search_dt = GridSearchCV(estimator=modelo,
                              param_grid=params,
                              cv=3)

grid_search_dt.fit(X_train, y_train)
```

Out[10]:



In [11]:
```
#VISUALIZAMOS LOS PARAMETROS Y SUS VALORES OPTIMOS ENCONTRADOS
grid_search_dt.best_estimator_
```

Out[11]:
```
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_state=4
2)
```

In [12]:
```
tree_clf = DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_state=
tree_clf.fit(X_train, y_train)

export_graphviz(
        tree_clf,
        out_file=str(IMAGES_PATH / "aguasSub.dot"),
        feature_names=X.columns,
        class_names=classes,
        rounded=True,
```

```
        filled=True
    )
```

In [13]:
```
from graphviz import Source

Source.from_file(IMAGES_PATH / "aguasSub.dot")
```

Out[13]: b'

\n'

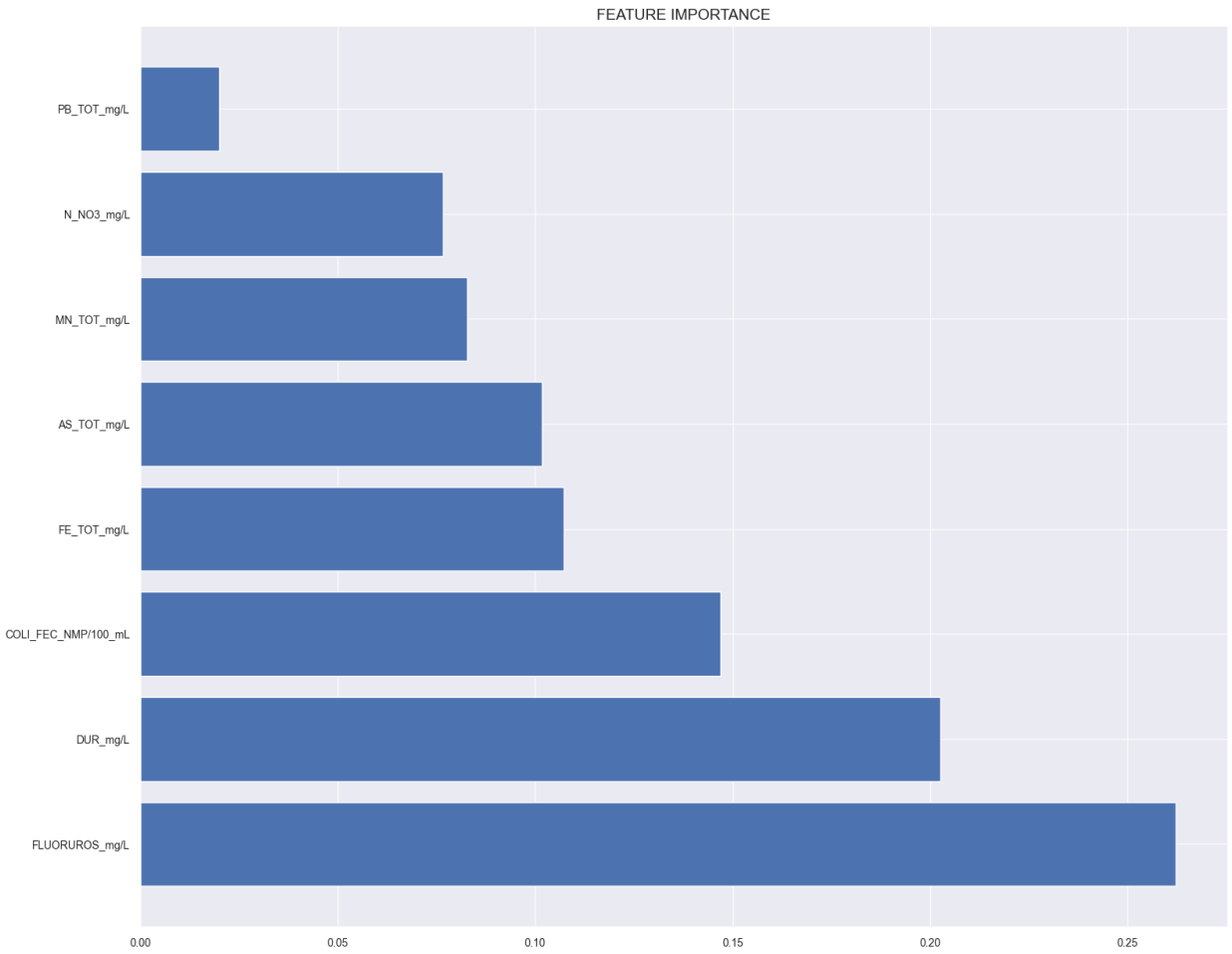
OBTENER LAS FEATURE IMPORTANCE DE DECISION TREE

In [15]:
```
ftImpTree = pd.DataFrame({'FEATURE': X.columns, 'IMPORTANCE': tree_clf.feature_
ftImpTree = ftImpTree.sort_values('IMPORTANCE', ascending=False)
ftImpTree = ftImpTree.loc[ftImpTree['IMPORTANCE'] > 0, ]
ftImpTree
```

Out[15]:

	FEATURE	IMPORTANCE
3	FLUORUROS_mg/L	0.262220
4	DUR_mg/L	0.202551
5	COLI_FEC_NMP/100_mL	0.146987
13	FE_TOT_mg/L	0.107219
7	AS_TOT_mg/L	0.101651
12	MN_TOT_mg/L	0.082706
6	N_NO3_mg/L	0.076630
11	PB_TOT_mg/L	0.020035

In [16]:
```
plt.figure(figsize=(18, 15))
plt.barh(
    ftImpTree['FEATURE'],
    ftImpTree['IMPORTANCE']
  )

plt.title('FEATURE IMPORTANCE')
plt.show()
```



OBTENEMOS EL REPORTE DE CLASIFICACION PARA DECISION TREE

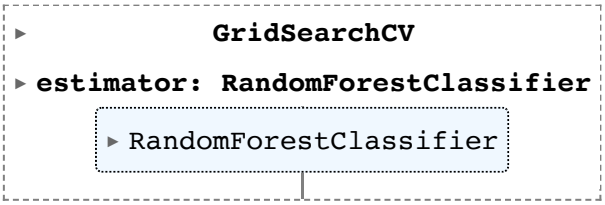
In [294…
```
y_pred = tree_clf.predict(Xtest)
print(classification_report(ytest, y_pred, target_names=classes))
```

```
              precision    recall  f1-score   support

    Amarillo       0.95      0.74      0.83        81
        Rojo       0.96      0.95      0.96       113
       Verde       0.86      0.99      0.92       127

    accuracy                           0.91       321
   macro avg       0.92      0.89      0.90       321
weighted avg       0.92      0.91      0.91       321
```

RANDOM FOREST

In [18]:
```
#HACEMOS UNA ITERACION PARA ENCONTRAR LOS MEJORES VALORES ENTRE POSIBLE PARAMET
from sklearn.ensemble import RandomForestClassifier
params = {
    'n_estimators': [500],
    'max_leaf_nodes': [64, 128, 256]
}
modelo = RandomForestClassifier(n_jobs=-1, random_state=42)
grid_search_rf = GridSearchCV(estimator=modelo,
                              param_grid=params,
                              cv=3)

grid_search_rf.fit(X_train, y_train)
```

Out[18]:



In [19]:
```
#VISUALIZAMOS LOS PARAMETROS Y SUS VALORES OPTIMOS ENCONTRADOS
grid_search_rf.best_estimator_
```

Out[19]:
```
RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1,
                       random_state=42)
```

In [20]:
```
rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=64,
                                 n_jobs=-1, random_state=42)
rnd_clf.fit(X_train, y_train)
```

Out[20]:
```
RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1,
                       random_state=42)
```

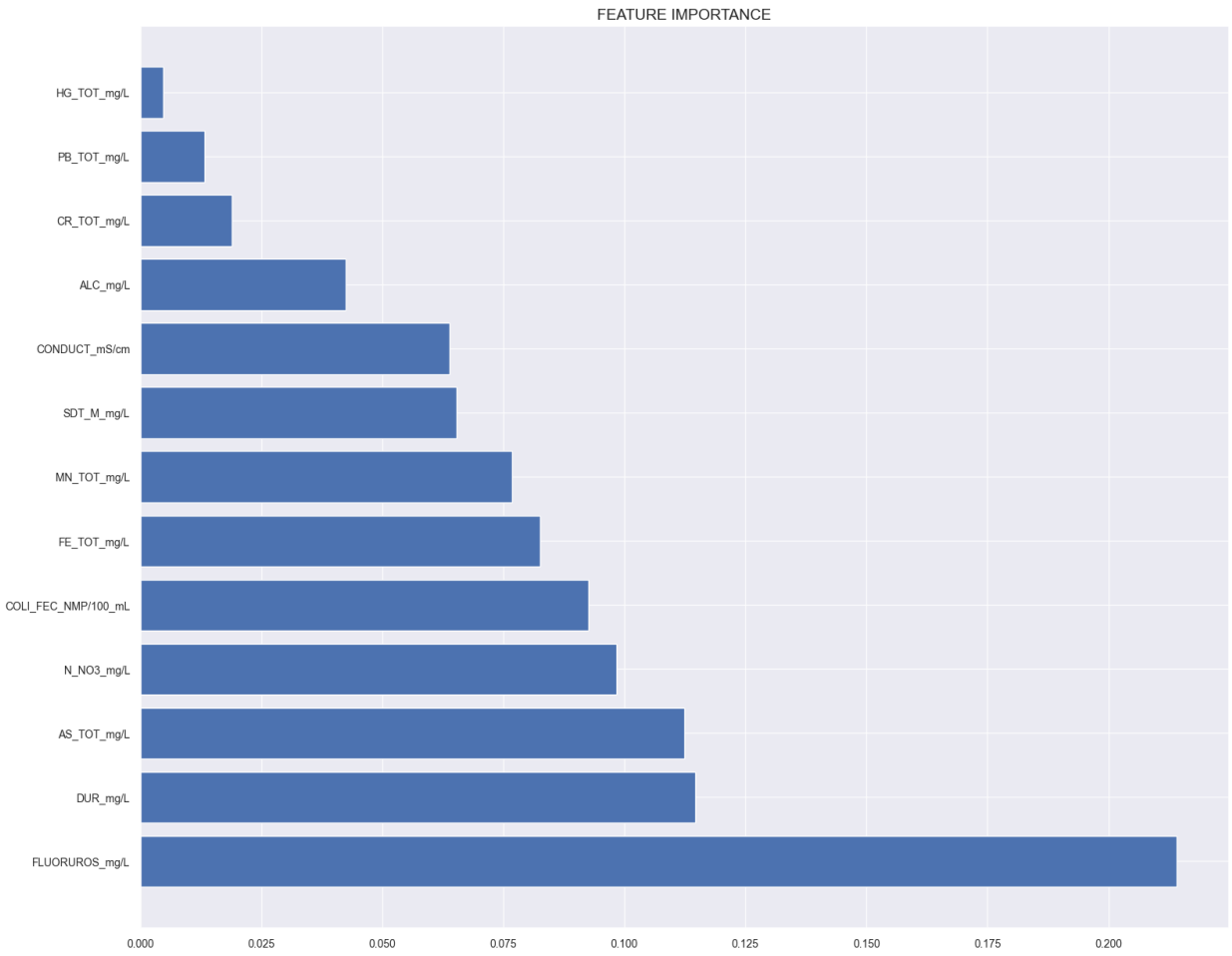
OBTENEMOS LAS FEATURE IMPORTANCE DE RANDOM FOREST

In [21]:
```
ftImpDf = pd.DataFrame({'FEATURE': X.columns, 'IMPORTANCE': rnd_clf.feature_imp
ftImpDf = ftImpDf.sort_values('IMPORTANCE', ascending=False)
ftImpDf = ftImpDf.loc[ftImpDf['IMPORTANCE'] > 0, ]
ftImpDf
```

Out[21]:

	FEATURE	IMPORTANCE
3	FLUORUROS_mg/L	0.214121
4	DUR_mg/L	0.114724
7	AS_TOT_mg/L	0.112350
6	N_NO3_mg/L	0.098399
5	COLI_FEC_NMP/100_mL	0.092577
13	FE_TOT_mg/L	0.082550
12	MN_TOT_mg/L	0.076810
2	SDT_M_mg/L	0.065316
1	CONDUCT_mS/cm	0.063902
0	ALC_mg/L	0.042387
9	CR_TOT_mg/L	0.018900
11	PB_TOT_mg/L	0.013232
10	HG_TOT_mg/L	0.004732

In [22]:
```
plt.figure(figsize=(18, 15))
plt.barh(
    ftImpDf['FEATURE'],
    ftImpDf['IMPORTANCE']
  )

plt.title('FEATURE IMPORTANCE')
plt.show()
```



OBTENEMOS EL REPORTE DE CLASIFICACION PARA RANDOM FOREST

In [24]:
```
from sklearn.metrics import classification_report
y_pred = rnd_clf.predict(Xtest)
print(classification_report(ytest, y_pred, target_names=classes))
```

```
              precision    recall  f1-score   support

    Amarillo       0.89      1.00      0.94        65
        Rojo       0.99      0.93      0.96        86
       Verde       0.99      0.97      0.98       116

    accuracy                           0.96       267
   macro avg       0.96      0.97      0.96       267
weighted avg       0.97      0.96      0.96       267
```

NOTAMOS QUE CON LOS DATOS DE PRUEBA, RANDOM FOREST ENTREGA UN MEJOR MODELO

COMPARACION DE CLASIFICADORES

Dividimos nuestro conjunto de entrenamiento en datos de entrenamiento y validacion

In [25]:
```
X_train2, X_valid, y_train2, y_valid = train_test_split(X_train, y_train, trair

print("X Entrenamiento", X_train2.shape)
print("X Validacion", X_valid.shape)

print("Y Entrenamiento", y_train2.shape)
print("Y Validacion", y_valid.shape)
```

```
X Entrenamiento (560, 14)
X Validacion (241, 14)
Y Entrenamiento (560,)
Y Validacion (241,)
```

Entrenamos nuestros modelos

In [26]:
```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import LinearSVC
from sklearn.neural_network import MLPClassifier

extra_trees_clf = ExtraTreesClassifier(n_estimators=5, random_state=42)
svm_clf = LinearSVC(max_iter=100, tol=20, random_state=42)
mlp_clf = MLPClassifier(random_state=42,max_iter= 10000)

estimators = [tree_clf, rnd_clf, extra_trees_clf, svm_clf, mlp_clf]
for estimator in estimators:
    print("Training the", estimator)
    estimator.fit(X_train2, y_train2)
```

```
Training the DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_sta
te=42)
Training the RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_job
s=-1,
                       random_state=42)
Training the ExtraTreesClassifier(n_estimators=5, random_state=42)
Training the LinearSVC(max_iter=100, random_state=42, tol=20)
Training the MLPClassifier(max_iter=10000, random_state=42)
```

In [27]:
```
print('----------------------------------------')
for estimator in estimators:
    print(estimator)
    print(estimator.score(X_valid, y_valid))
    print('----------------------------------------')
```

```
----------------------------------------
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_state=42)
0.8962655601659751
----------------------------------------
RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1,
                       random_state=42)
0.9543568464730291
----------------------------------------
ExtraTreesClassifier(n_estimators=5, random_state=42)
0.8630705394190872
----------------------------------------
LinearSVC(max_iter=100, random_state=42, tol=20)
0.8381742738589212
----------------------------------------
MLPClassifier(max_iter=10000, random_state=42)
0.91701244813278
----------------------------------------
```

REPORTE DE CLASIFICACION PARA TODOS NUESTROS MODELOS

In [28]:
```
from sklearn.metrics import classification_report

for estimator in estimators:
    print('-------------------------------------------------------')
    print(estimator)
    y_pred = estimator.predict(X_train2)
    print(classification_report(y_train2, y_pred, target_names=classes))
```

```
--------------------------------------------------------
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_state=42)
              precision    recall  f1-score   support

    Amarillo       0.96      0.83      0.89       130
        Rojo       1.00      0.94      0.97       206
       Verde       0.88      1.00      0.94       224

    accuracy                           0.94       560
   macro avg       0.95      0.92      0.93       560
weighted avg       0.94      0.94      0.94       560

--------------------------------------------------------
RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1,
                       random_state=42)
              precision    recall  f1-score   support

    Amarillo       1.00      1.00      1.00       130
        Rojo       1.00      1.00      1.00       206
       Verde       1.00      1.00      1.00       224

    accuracy                           1.00       560
   macro avg       1.00      1.00      1.00       560
weighted avg       1.00      1.00      1.00       560

--------------------------------------------------------
ExtraTreesClassifier(n_estimators=5, random_state=42)
              precision    recall  f1-score   support

    Amarillo       1.00      1.00      1.00       130
        Rojo       1.00      1.00      1.00       206
       Verde       1.00      1.00      1.00       224

    accuracy                           1.00       560
   macro avg       1.00      1.00      1.00       560
weighted avg       1.00      1.00      1.00       560

--------------------------------------------------------
LinearSVC(max_iter=100, random_state=42, tol=20)
              precision    recall  f1-score   support

    Amarillo       0.73      0.85      0.79       130
        Rojo       0.95      0.81      0.87       206
       Verde       0.86      0.91      0.89       224

    accuracy                           0.86       560
   macro avg       0.85      0.85      0.85       560
weighted avg       0.87      0.86      0.86       560

--------------------------------------------------------
MLPClassifier(max_iter=10000, random_state=42)
              precision    recall  f1-score   support

    Amarillo       1.00      0.99      1.00       130
        Rojo       1.00      1.00      1.00       206
       Verde       1.00      1.00      1.00       224
```

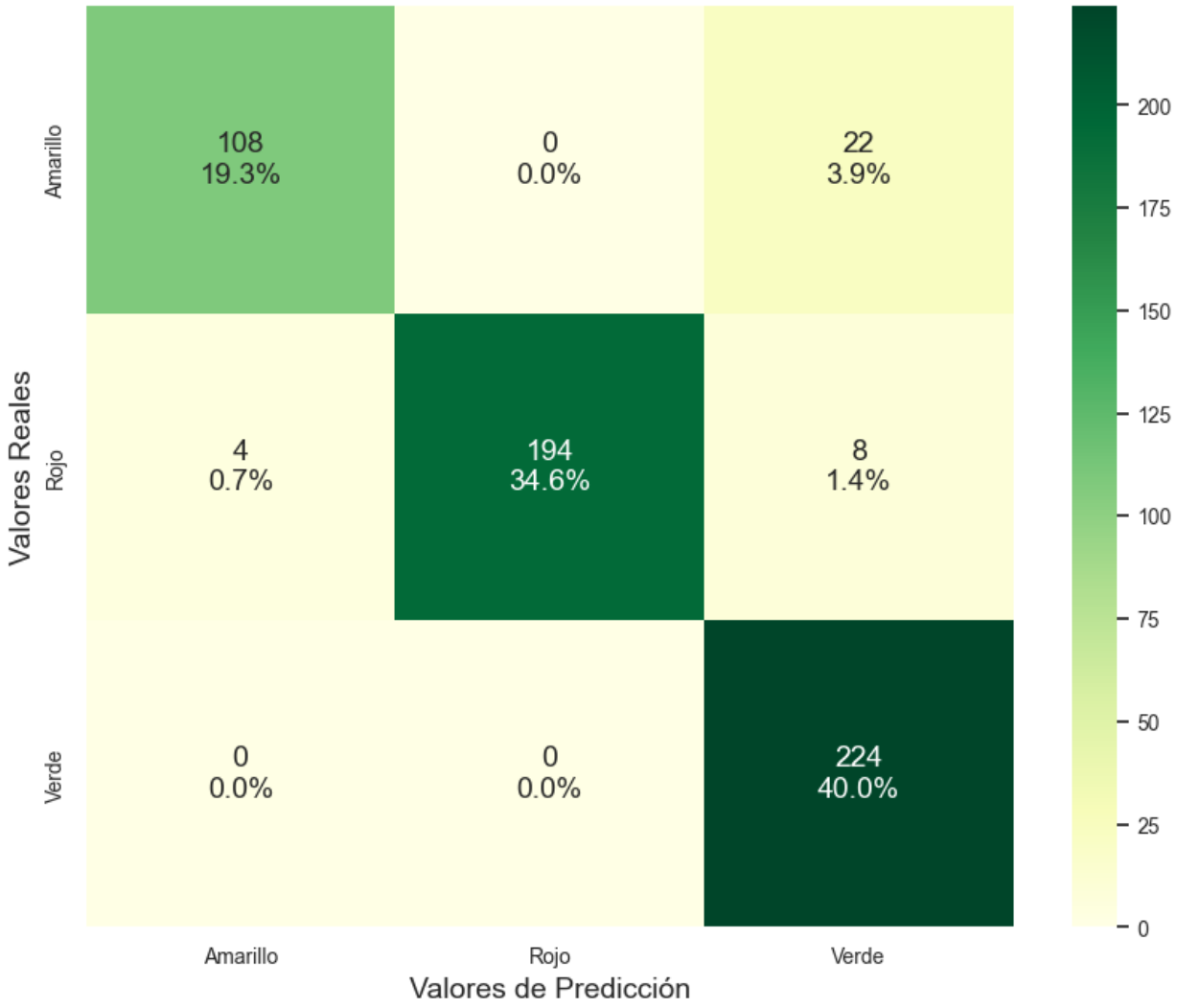
```
              precision    recall  f1-score   support

    accuracy                           1.00       560
   macro avg       1.00      1.00      1.00       560
weighted avg       1.00      1.00      1.00       560
```

CONFUSION MATRIX

In [29]:
```
def mi_cm(yreal, ypred):
    cm = confusion_matrix(yreal, ypred)
    labels = ['Amarillo', 'Rojo', 'Verde']
    frec = ["{0:0.0f}".format(value) for value in cm.flatten()]
    porc = ["{0:.1%}".format(value) for value in cm.flatten()/np.sum(cm)]

    etiquetas = [f"{v1}\n{v2}" for v1, v2 in zip(frec,porc)]
    etiquetas = np.asarray(etiquetas).reshape(3,3)

    plt.figure(figsize=(10,8))
    ax = sns.heatmap(cm, annot=etiquetas,xticklabels = labels, yticklabels=labe
    ax.set(ylabel="Valores Reales", xlabel="Valores de Predicción")
    plt.show()
```

In [30]:
```
from sklearn.metrics import confusion_matrix

for estimator in estimators:
    print('-------------------------------------------------------')
    print(estimator)
    y_pred = estimator.predict(X_train2)
    mi_cm(y_train2, y_pred)
```

```
-------------------------------------------------------
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_state=42)
```



```
RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1,
                       random_state=42)
```

```
ExtraTreesClassifier(n_estimators=5, random_state=42)
```

```
LinearSVC(max_iter=100, random_state=42, tol=20)
```

```
MLPClassifier(max_iter=10000, random_state=42)
```

CONCLUSIONES FINALES

Con base en los resultados del reporte de clasificación y de las matriz de confusion pensamos que el mejor modelo clasificatorio es el Decision Tree porque brinda buenas métricas de salida y sin overfitting, tal como se muestra en los demás.

Al menos para ese conjunto de datos, el modelo del Decision Tree es el adecuado, pudiendo incluso mejorarse intentando con un abanico más amplio de datos y de los parámetros evaluados.

PRESENTACION EJECUTIVA

ANALISIS DE AGUAS SUBTERRANEAS EN MEXICO.

CIENCIA Y ANALITICA DE DATOS

EQUIPO 12

GUILLERMO ALFONSO MUÑIZ HERMOSILLO – A01793101

JOSE RAMIRO ADÁN CHARLES – A00174646

México enfrenta contaminación del agua subterránea

- En los ultimos años se ha vuelto de vital importancia el conocer la problemática de la extraccion del agua subterranea en nuestro pais con el fin de suministrar a la poblacion de este vital liquido.

- Actualmente, se tienen reportes de que mas de 20 estados enfrentan contaminacion en sus mantos acuiferos, causando que este liquido no cumpla con los estandares para su suministro. [1]

SITUACION ACTUAL

- Existen 634 cuerpos de agua contaminados, por tan solo 434 no contaminados.
- Muchos de estos se encuentran cerca de las zonas mas pobladas.
- El peligro a la salud es inminente.

ANALISIS DE DATOS

- Para nuestro estudio, tomamos el conjunto de datos correspondientes a 5000 sitios de monitoreo de contaminantes acuifereos en Mexico.
- Dicho conjunto de datos se compone de 2 bases de datos enfocadas en Aguas superficiales y Aguas Subterraneas, en nuestro estudio nos enfocamos en este ultimo.
- El estudio se divide en 2 partes:
  - **PARTE 1: Limpieza, análisis, visualización y kmeans**
  - **PARTE 2: Clasificacion y ensambles.**

PARTE 1

- CONJUNTO DE DATOS:

  Nuestro conjunto de datos consta de 1068 registros(filas) y 57 variables(columnas) las cuales contienen la informacion relativa a las aguas subterraneas de Mexico y su estado de contaminacion asi como otros datos de localizacion de dichos registros.

- DESCRIPCION DE DATOS:

  Para comenzar la limpieza de nuestros datos, se comenzó definiendo diferentes subconjuntos de datos:

  1. Variables Categoricas: Aquellas que contienen valores definidos dentro de una categoria. Dentro de este subconjunto encontramos 28 variables/caracteristicas.
  2. Variables Numericas: Aquellas variables que contienen valores en escala numerica. Dentro de este subconjunto encontramos 14 variables/caracteristicas
  3. Variables Binarias: Variables que contienen solamente 2 valores posibles. Dentro de este subconjunto contamos con 9 variables.
  4. Variables de ubicación: Variables que describen una coordenada. Solo 2 variables se encuentran en este subconjunto
  5. Variables no relevantes: Variables que estan vacias o no creemos que aportaran algun valor relevante a nuestro modelo. 4 variables se encuentran en este subconjunto

LIMPIEZA DE DATOS

Despues de haber definido nuestros datos se llevaron a cabo los siguientes 2 pasos:

1. **Definir Pipelines de imputacion** para cada uno de estos subconjuntos. Es decir, que acciones se iban a tomar con los datos faltantes. Dichas acciones fueron:

A. Variables Categoricas: Imputacion por Moda, sustituir por el valor que mas veces aparece en la columna.

```
simpleImput = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
x = simpleImput.fit_transform(dfSubterraneas[categoricasSubte])
```

B. Variables Numericas: Estandarizacion de tipo de datos seguida de imputacion por mediana (valor medio de todo el conjunto de datos)

```
simpleImputnum = SimpleImputer(missing_values=np.nan, strategy='median')
xnum = simpleImputnum.fit_transform(dfSubterraneas[numericasSubte])
```

C. Variables No Relevantes: Se eliminaron estas variables de nuestro conjunto de datos

```
dfSubterraneas.drop(dfSubterraneas[NASubte], axis=1, inplace=True)
dfSubterraneas.shape
```

```
(1068, 53)
```

El resto de subconjuntos no contaban con valores faltantes, por lo que no se requirio imputacion de valores faltantes

LIMPIEZA DE DATOS

2. **Definir Pipelines de transformacion** para cada uno de los subconjuntos que creimos necesario. Las acciones llevadas acabo fueron:

A. Variables Categoricas: Codificacion de Cadenas con el objetivo de tener valores numericos codificados correspondientes a cada

```
le = preprocessing.LabelEncoder()
le.fit(dfSubterraneas[categoricasSubte])
```

B. Variables Numericas: Escalamiento estandar para las variables, con el objetivo de no perder informacion y no tener variables desbalanceadas

```
scal = StandardScaler()
xscale = scal.fit_transform(dfSubterraneas[numericasSubte])
```

C. Variables Binarias: Transformacion One Hot encoding con drop binario.

```
onehot = OneHotEncoder(drop='if_binary', handle_unknown='ignore')
xonebin = onehot.fit_transform(dfSubterraneas[binariasSubte]).toarray()
dfSubterraneas[binariasSubte] = xonebin
dfSubterraneas[binariasSubte].head()
```

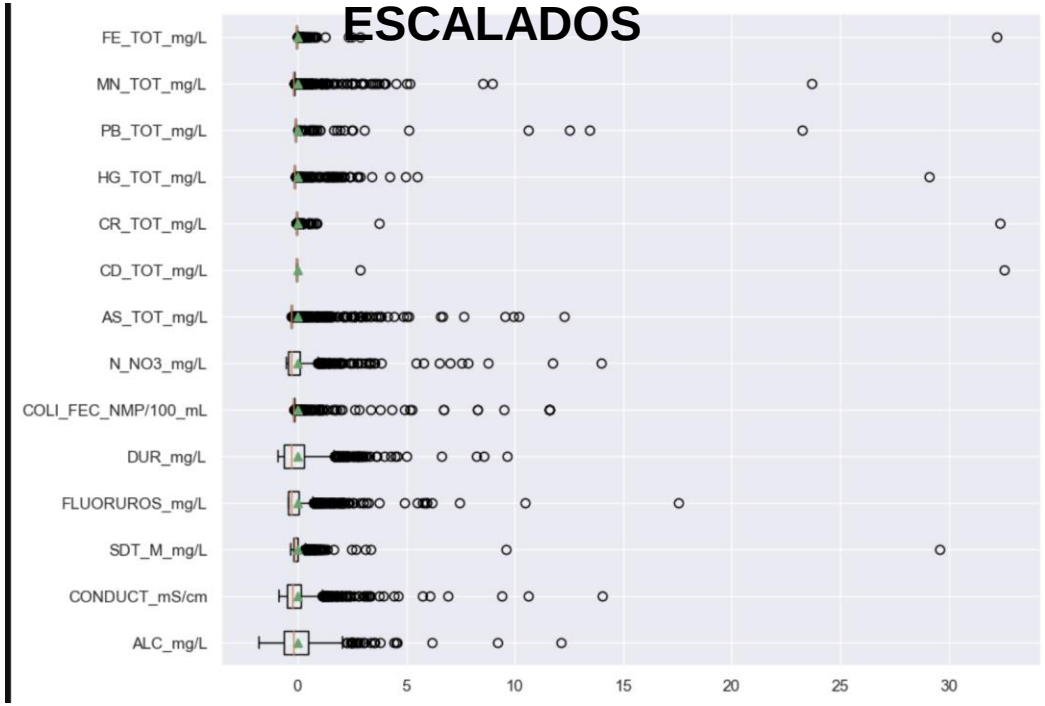
	CUMPLE_CON_FLUO	CUMPLE_CON_CF	CUMPLE_CON_AS	CUMPLE_CON_CD	CUMPLE_CON_CR	CUMPLE_CON_HG	CUMPLE_CON_PB	CUMPLE_CON_MN	CUMPLE_CON_FE
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

El resto de subconjuntos no contaban con valores para escalar.

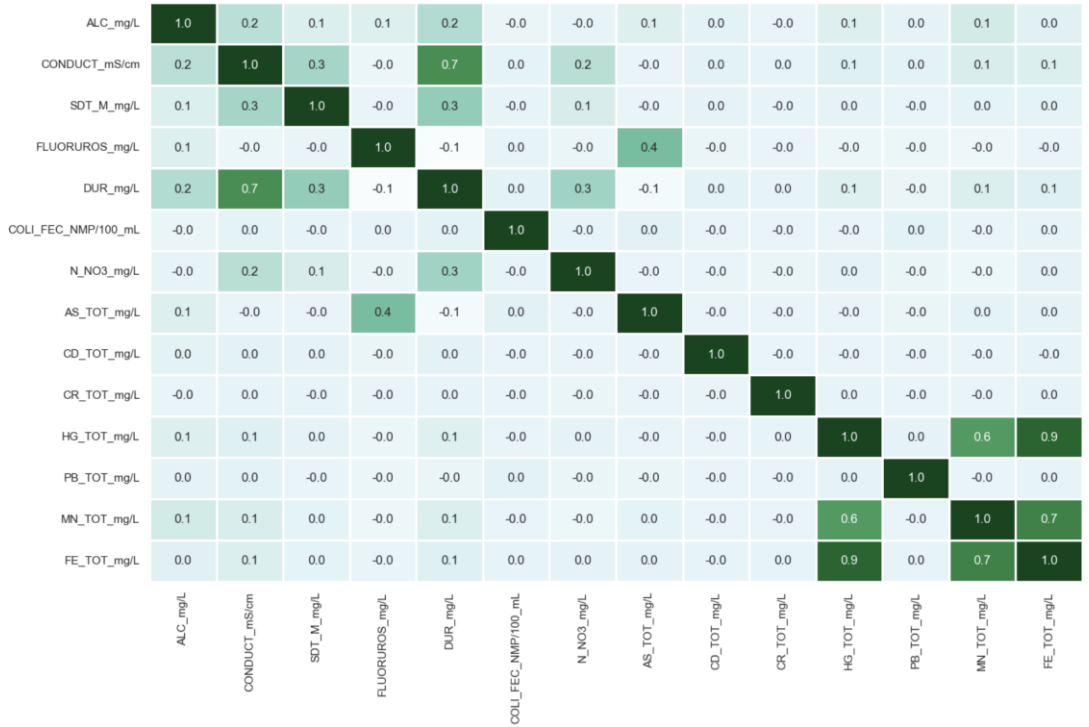
ANALISIS DE DATOS

- Una vez realizado nuestra limpieza de datos, procedimos a el analisis de estos. En los cuales pudimos observar como se relacionan nuestros datos entre ellos y como era conveniente el escalamiento y transformacion de algunos de ellos .

DISTRIBUCION DE DATOS ESCALADOS

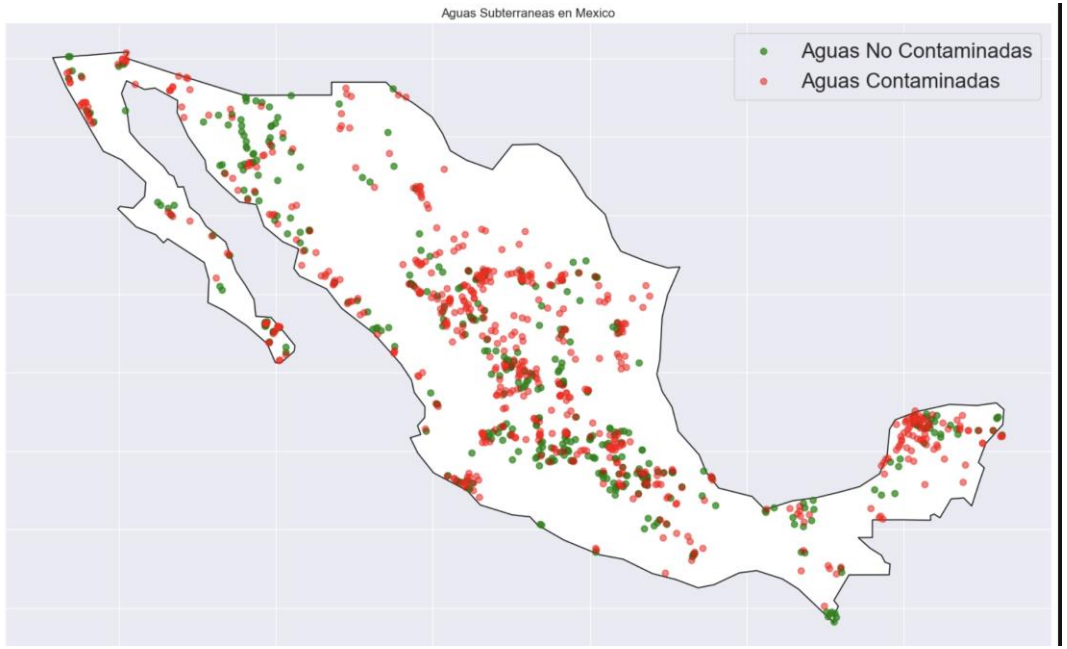


MATRIZ DE CORRELACION

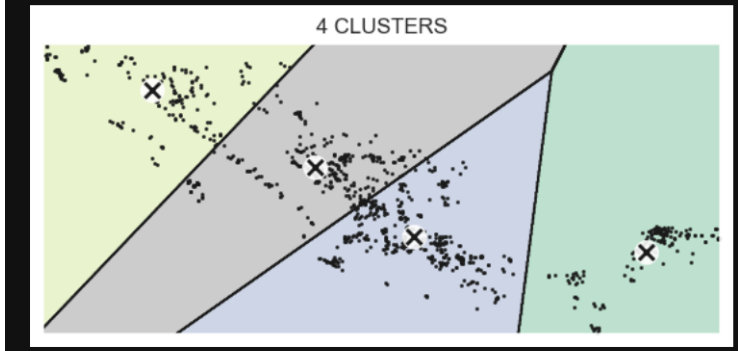


KMEANS

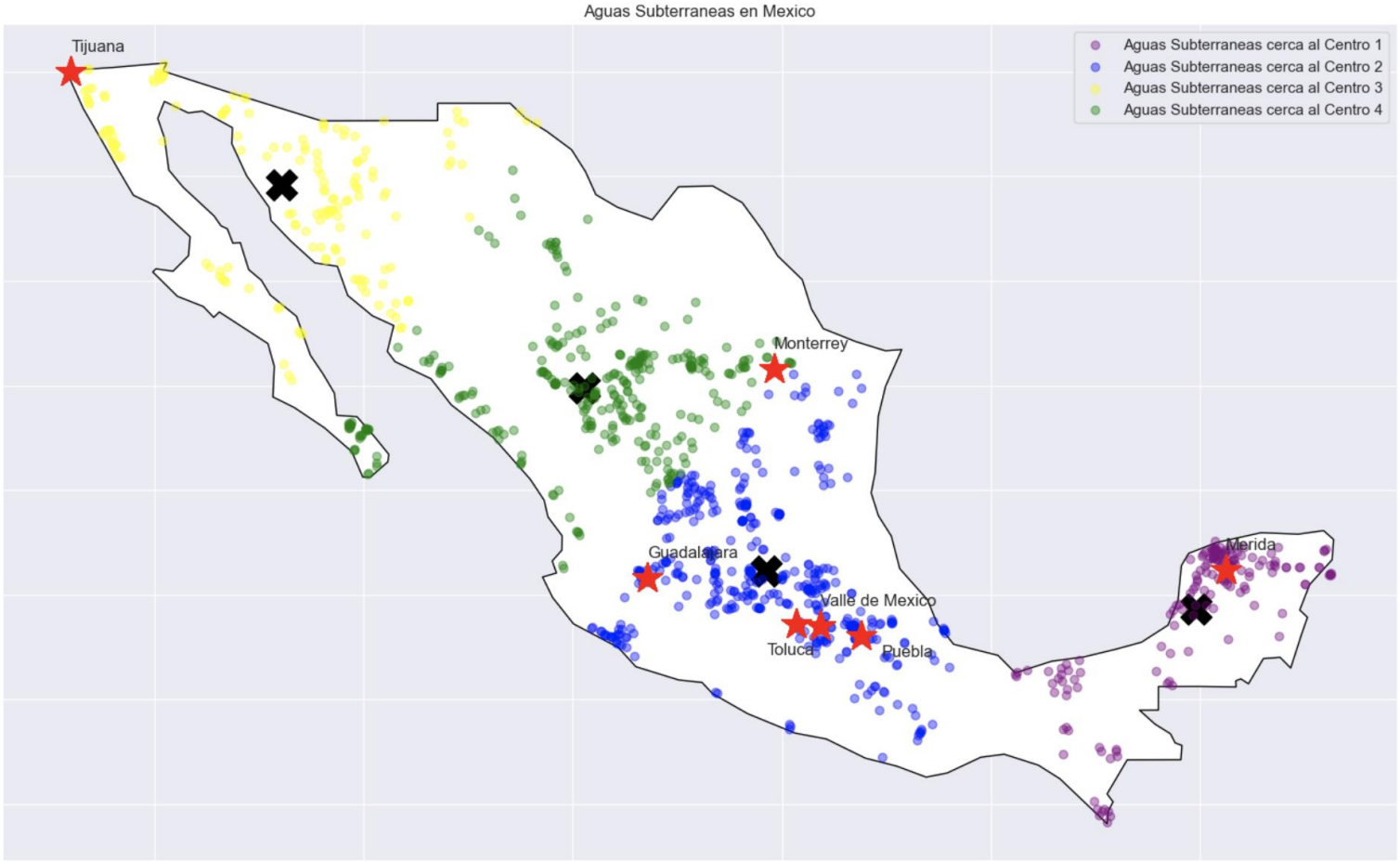
- Con el objetivo de identificar si existia una relacion entre la calidad del agua y su ubicación geografica llevamos acabo un analisis de agrupamiento por clusters, en base a la ubicación de los registros de agua contaminada. Encontrando que un numero de 4 clusters es el modelo que explica mucho porcentaje de la contaminacion




```
]: kmeans_final = KMeans(n_clusters=4, init="random", n_init=1, max_iter=10,
                         random_state=42)
   kmeans_final.fit(XKmeans)
   plt.figure(figsize=(15, 10))
   plt.subplot(321)
   plot_decision_boundaries(kmeans_final, XKmeans, show_centroids=False,
                            show_ylabels=False, show_xlabels=False)
   plot_centroids(kmeans_final.cluster_centers_)
   plt.title("4 CLUSTERS")

]: Text(0.5, 1.0, '4 CLUSTERS')
```



- Finalmente con el objetivo de encontrar otra relacion, mapeamos nuestros cuerpos de agua contaminadas agrupadas en 4 clusters, cada uno de diferente color y su centro. Asi mismo decidimos agregar las zonas mas pobladas de nuestro pais con el objetivo de encontrar una relacion con la contaminacion subyacente en dichos cuerpos subterraneos.



Como podemos observar se ve una posible relacion entre el numero de personas o ciudades importantes y el numero de aguas subterraneas contaminadas, si bien la relacion usando K-means no es del todo clara, podemos mediante la graficacion de las ciudades importantes ver que hay muchas aguas contaminadas alrededor de ellas, lo que indica que la actividad humana e industrial en las zonas puede ser un factor determinante para la contaminacion acuifera.

PARTE 2

- ENCONTRANDO UN MODELO DE CLASIFICACION

Del conjunto de datos, consideramos la columna SEMAFORO como una variable de salida, y las variables numéricas descritas en la Parte 1 (diferentes mediciones) como variables de entrada para usarlas en encontrar un modelo de clasificación predictivo siguiente los siguientes pasos:

1. Generación de particiones de entrenamiento y prueba
2. Codificación del semáforo a numérico
3. Exploración de modelos Decision Tree y Random Forest
    1. Iteración de mejores parámetros
    2. Analizando las variables de mayor importancia
    3. Comparando los resultados entre ambos
4. Comparación contra modelos EXTRATREE, LINEARSVC Y MLPCLASSIFIER
5. Visualización de resultados
6. Conclusiones Finales

Particiones de entrenamiento y etiquetado de variable de salida

```
X_train, Xtest, y_train, ytest = train_test_split(X, y, train_size=0.75, random_state=42)

print("X Entrenamiento-Validacion", X_train.shape)
print("X Prueba", Xtest.shape)

print("Y Entrenamiento", y_train.shape)
print("Y Prueba", ytest.shape)
```

```
X Entrenamiento-Validacion (801, 14)
X Prueba (267, 14)
Y Entrenamiento (801,)
Y Prueba (267,)
```

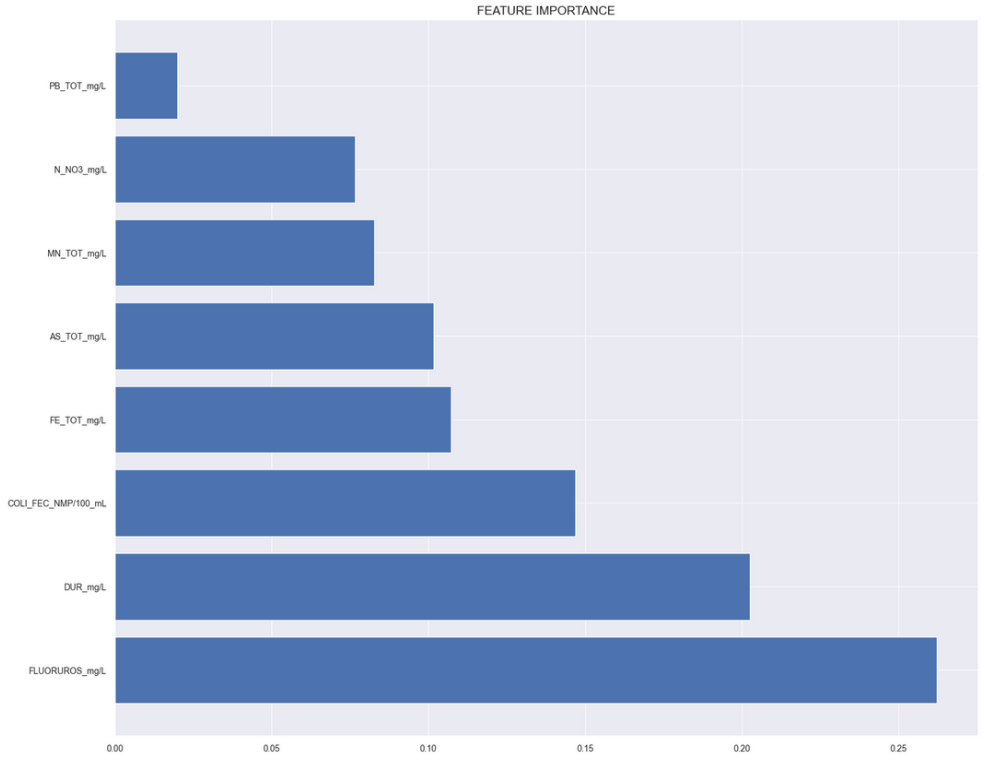
```
le = preprocessing.LabelEncoder()
le.fit(df['SEMAFORO'])
classes = le.classes_

classes
```

```
array(['Amarillo', 'Rojo', 'Verde'], dtype=object)
```

```
x = le.transform(df['SEMAFORO'])
df['SEMAFORO'] = x
df[['SEMAFORO']].head()
```

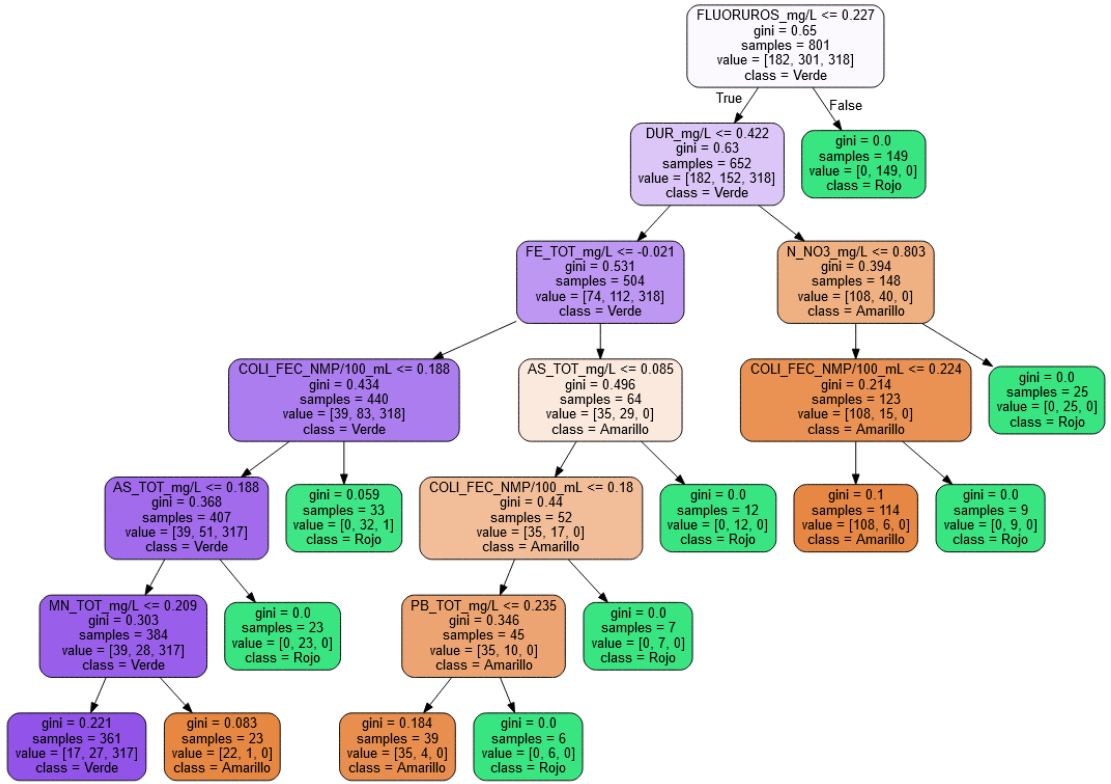
Decision Tree

```
#HACEMOS UNA ITERACION PARA ENCONTRAR LOS MEJORES VALORES ENTRE POSIBLE PARAMETROS
params = {
    'max_leaf_nodes': list(range(2, 100)),
    'max_depth': list(range(1, 7)),
    'min_samples_split': [2, 3, 4]
}
modelo=DecisionTreeClassifier(random_state=42)
grid_search_dt = GridSearchCV(estimator=modelo,
                              param_grid=params,
                              cv=3)

grid_search_dt.fit(X_train, y_train)
```



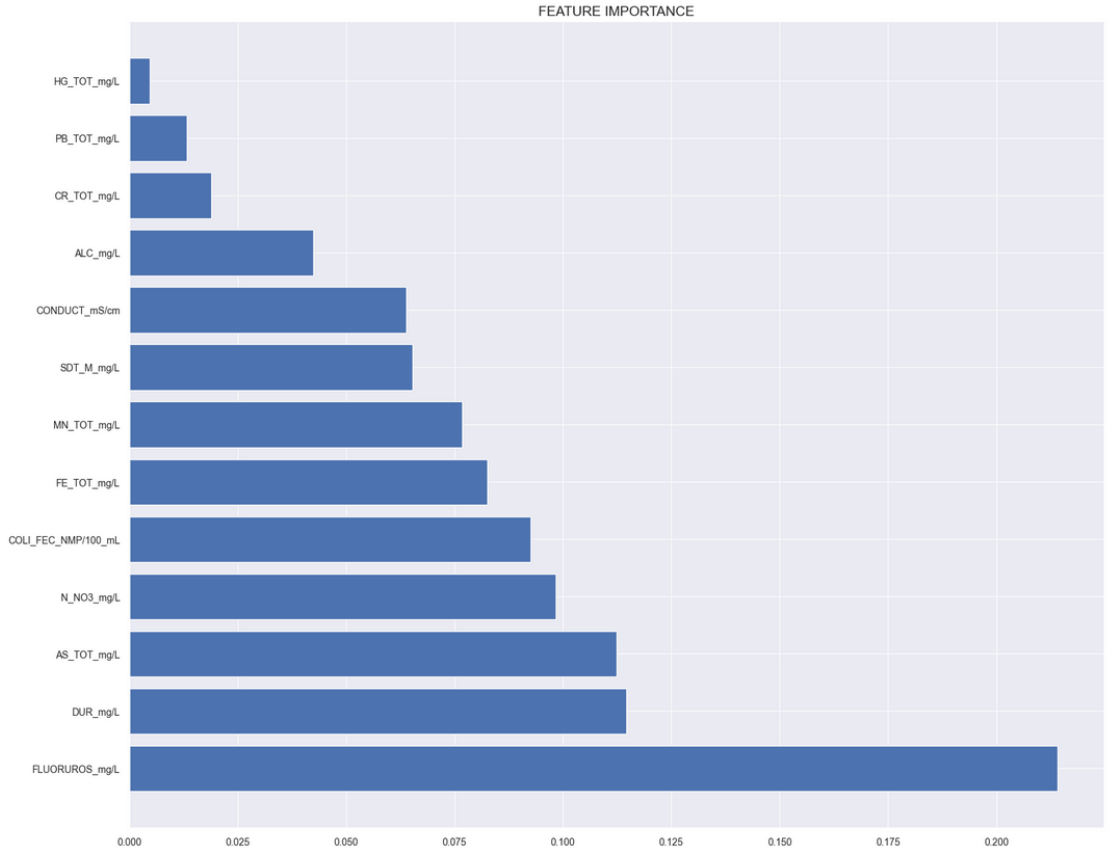
Mejores parámetros encontrados:

DecisionTreeClassifier(**max_depth=6**, max_leaf_nodes=12, random_state=42)



Random Forest

```
#HACEMOS UNA ITERACION PARA ENCONTRAR LOS MEJORES VALORES ENTRE POSIBLE PARAMETROS
from sklearn.ensemble import RandomForestClassifier
params = {
    'n_estimators': [500],
    'max_leaf_nodes': [64, 128, 256]
}
modelo = RandomForestClassifier(n_jobs=-1, random_state=42)
grid_search_rf = GridSearchCV(estimator=modelo,
                              param_grid=params,
                              cv=3)
```



Mejores parámetros encontrados:

RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1, random_state=42)

Reporte de Clasificacion
Decision Tree vs Random Forest

Decision Tree

	precision	recall	f1-score	support
Amarillo	0.95	0.74	0.83	81
Rojo	0.96	0.95	0.96	113
Verde	0.86	0.99	0.92	127
accuracy			0.91	321
macro avg	0.92	0.89	0.90	321
weighted avg	0.92	0.91	0.91	321

Random Forest

	precision	recall	f1-score	support
Amarillo	0.89	1.00	0.94	65
Rojo	0.99	0.93	0.96	86
Verde	0.99	0.97	0.98	116
accuracy			0.96	267
macro avg	0.96	0.97	0.96	267
weighted avg	0.97	0.96	0.96	267

El reporte muestra que Random Forest obtiene mejores puntajes en cuanto predicción de cada clase, es de extrañar el 1.00 mostrado para la clase Amarillo del Recall

Comparación contra modelos EXTRATREE, LINEARSVC Y MLPCLASSIFIER

Generando particion de validación para comparar todos los modelos.

```
X_train2, X_valid, y_train2, y_valid = train_test_split(X_train, y_train, train_size=0.70, random_state=42)

print("X Entrenamiento", X_train2.shape)
print("X Validacion", X_valid.shape)

print("Y Entrenamiento", y_train2.shape)
print("Y Validacion", y_valid.shape)
```

```
X Entrenamiento (560, 14)
X Validacion (241, 14)
Y Entrenamiento (560,)
Y Validacion (241,)
```

Iteración a través de todos los modelos.

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import LinearSVC
from sklearn.neural_network import MLPClassifier

extra_trees_clf = ExtraTreesClassifier(n_estimators=5, random_state=42)
svm_clf = LinearSVC(max_iter=100, tol=20, random_state=42)
mlp_clf = MLPClassifier(random_state=42,max_iter= 10000)

estimators = [tree_clf, rnd_clf, extra_trees_clf, svm_clf, mlp_clf]
for estimator in estimators:
    print("Training the", estimator)
    estimator.fit(X_train2, y_train2)
```

Usando los modelos Decision Tree y Random Forest que se encontraron con los mejores parámetros

Comparación de todos los modelos con Reporte de Clasificación

---

DecisionTreeClassifier(max_depth=6, max_leaf_nodes=12, random_state=42)

	precision	recall	f1-score	support
Amarillo	0.96	0.83	0.89	130
Rojo	1.00	0.94	0.97	206
Verde	0.88	1.00	0.94	224
accuracy			0.94	560
macro avg	0.95	0.92	0.93	560
weighted avg	0.94	0.94	0.94	560

RandomForestClassifier(max_leaf_nodes=64, n_estimators=500, n_jobs=-1, random_state=42)

	precision	recall	f1-score	support
Amarillo	1.00	1.00	1.00	130
Rojo	1.00	1.00	1.00	206
Verde	1.00	1.00	1.00	224
accuracy			1.00	560
macro avg	1.00	1.00	1.00	560
weighted avg	1.00	1.00	1.00	560

ExtraTreesClassifier(n_estimators=5, random_state=42)

	precision	recall	f1-score	support
Amarillo	1.00	1.00	1.00	130
Rojo	1.00	1.00	1.00	206
Verde	1.00	1.00	1.00	224
accuracy			1.00	560
macro avg	1.00	1.00	1.00	560
weighted avg	1.00	1.00	1.00	560

LinearSVC(max_iter=100, random_state=42, tol=20)

	precision	recall	f1-score	support
Amarillo	0.73	0.85	0.79	130
Rojo	0.95	0.81	0.87	206
Verde	0.86	0.91	0.89	224
accuracy			0.86	560
macro avg	0.85	0.85	0.85	560
weighted avg	0.87	0.86	0.86	560

MLPClassifier(max_iter=10000, random_state=42)

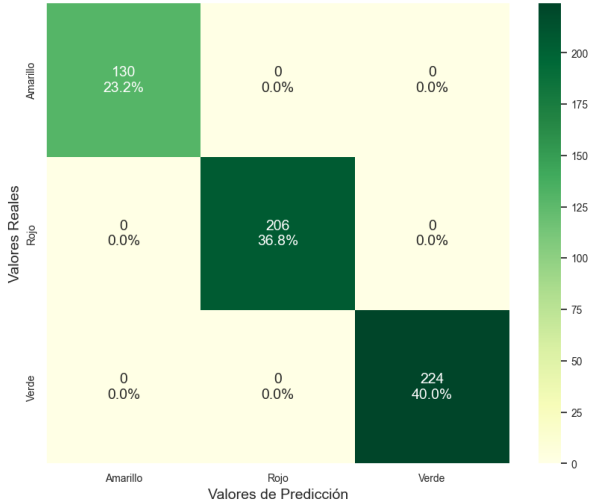
	precision	recall	f1-score	support
Amarillo	1.00	0.99	1.00	130
Rojo	1.00	1.00	1.00	206
Verde	1.00	1.00	1.00	224
accuracy			1.00	560
macro avg	1.00	1.00	1.00	560
weighted avg	1.00	1.00	1.00	560

Son de notar los resultados 1.0 que los modelos ExtraTree, LinearSVC y MLP muestran, es como si indicaran sobre entrenamiento.

Comparación de todos lo modelos con Matriz de Confusión

Decision Tree

Random Forest



Extra Tree Classifier

Linear SVC

MLP Classifier

CONCLUSIONES FINALES

- Con base en los resultados del reporte de clasificación y de las matriz de confusion pensamos que el mejor modelo clasificatorio es el Decision Tree porque brinda buenas métricas de salida y sin overfitting, tal como se muestra en los demás.
- Al menos para ese conjunto de datos, el modelo del Decision Tree es el adecuado, pudiendo incluso mejorarse intentando con un abanico más amplio de datos y de los parámetros evaluados.
- En cuanto al conocimiento que se puede obtener con este análisis respecto a la contaminación del agua, detectamos que la medición en las concentraciones de los fluoruros, materiales duros y arsénico están muy por encima de todos los demás y son los que mas influyen en el valor del semáforo.

BIBLIOGRAFIA ADICIONAL

- *[1] México enfrenta contaminación del agua subterránea*. (2015, febrero 3). Teorema Ambiental. http://www.teorema.com.mx/contaminacion_/mexico-enfrenta-contaminacion-del-agua-subterranea/