Open in Colab
(https://colab.research.google.com/github/PosgradoMNA/actividades-del-projecto-equipo-13/blob/main/Reto/Reto_Entrega_2_Clasificaci%C3%B3n_ensambles_y_presentacion_ejecutiva_JuanSebastianC

Reto: Parte 2 Clasificación-ensambles y presentación ejecutiva

Ciencia y analítica de datos

Profesor: María de la Paz Rico Fernández

Juan Sebastián Ortega Briones A01794327

Equipo 13

18 de Noviembre del 2022

Carga librerias y datos

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as color
import seaborn as sns
%matplotlib inline
```

In [2]:
```
pd.set_option("display.max_columns", 57)
pd.set_option("display.max_rows", 100)
```

Uso de Base de datos de Aguas Subterraneas

In [3]:
```
df=pd.read_csv("https://raw.githubusercontent.com/PosgradoMNA/actividades-del-projecto-equipo-13/main/Reto/Datos_de_calidad_del_agua_de_sitios_de_monitoreo_de_aguas_subterraneas_2020.csv", encoding = 'ISO-8859-1')  #Importa datos de Aguas subterraneas
```

In [4]:
```
df.head()
```

Out[4]:

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	
0	DLAGU6	POZO SAN GIL	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	ASIENTOS	
1	DLAGU6516	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	AGUASCALIENTES	
2	DLAGU7	POZO COSIO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	COSIO	AGU
3	DLAGU9	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON DE ROMOS	AGU
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA PAZ	T

In [5]:
```
df.shape
```

Out[5]:
```
(1068, 57)
```

Limpieza de datos

In [6]:
```
#Array de solo las columnas numericas
datos_numericos=['ALC_mg/L', 'CONDUCT_mS/cm', 'SDT_M_mg/L', 'FLUORUROS_mg/L', 'DUR_mg/L', 'COLI_FEC_NMP/100_mL', 'N_NO3_mg/L', 'AS_TOT_mg/L', 'CD_TOT_mg/L', 'CR_TOT_mg/L', 'HG_TOT_mg/L', 'PB_TOT_mg/L', 'MN_TOT_mg/L', 'FE_TOT_mg/L']
```

In [7]:
```
#Remplaza datos numericos que incluyen el simbolo < por 0
df.replace(to_replace=r'[<]\w+', value=0, regex=True, inplace=True)
```

In [8]:
```
#La columna SDT_mg/L no contiene datos y la de CONTAMINANTES tiene muchos Nan seran eliminadas
df.drop(['SDT_mg/L','CONTAMINANTES'], axis=1, inplace=True)
```

In [9]:
```
df.dropna(inplace=True)
```

In [10]:
```
#Convierte columnas que contienen datos números de tipo objeto a tipo flotante
df[datos_numericos]=df[datos_numericos].astype('float')
```

Clasificación

In [11]:
```
categorias=df['SEMAFORO'].unique()
categorias
```

Out[11]:
```
array(['Verde', 'Rojo', 'Amarillo'], dtype=object)
```

In [12]:
```
#Cambia la columna del semaforo de un string a un numero para poder comp
arar con kameans
y=df['SEMAFORO'].apply(lambda x: categorias.tolist().index(x))
```

In [13]:
```
X=df[datos_numericos].values
columnas=df[datos_numericos].columns
```

In [14]:
```
columnas
```

Out[14]:
```
Index(['ALC_mg/L', 'CONDUCT_mS/cm', 'SDT_M_mg/L', 'FLUORUROS_mg/L', 'DU
R_mg/L',
       'COLI_FEC_NMP/100_mL', 'N_NO3_mg/L', 'AS_TOT_mg/L', 'CD_TOT_mg/
L',
       'CR_TOT_mg/L', 'HG_TOT_mg/L', 'PB_TOT_mg/L', 'MN_TOT_mg/L',
       'FE_TOT_mg/L'],
      dtype='object')
```

In [15]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [16]:
```
tree_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
tree_clf.fit(X, y)
```

Out[16]:
```
DecisionTreeClassifier(max_depth=4, random_state=42)
```

Importancia de las variables

In [17]:
```
importancia_clf=pd.DataFrame(tree_clf.feature_importances_,columns=['imp
ortancia'])
importancia_clf['nombres']=columnas
importancia_clf.sort_values(by='importancia', ascending=False)
```

Out[17]:

	importancia	nombres
3	0.332748	FLUORUROS_mg/L
4	0.276398	DUR_mg/L
6	0.199842	N_NO3_mg/L
13	0.131057	FE_TOT_mg/L
5	0.038022	COLI_FEC_NMP/100_mL
7	0.021931	AS_TOT_mg/L
0	0.000000	ALC_mg/L
1	0.000000	CONDUCT_mS/cm
2	0.000000	SDT_M_mg/L
8	0.000000	CD_TOT_mg/L
9	0.000000	CR_TOT_mg/L
10	0.000000	HG_TOT_mg/L
11	0.000000	PB_TOT_mg/L
12	0.000000	MN_TOT_mg/L

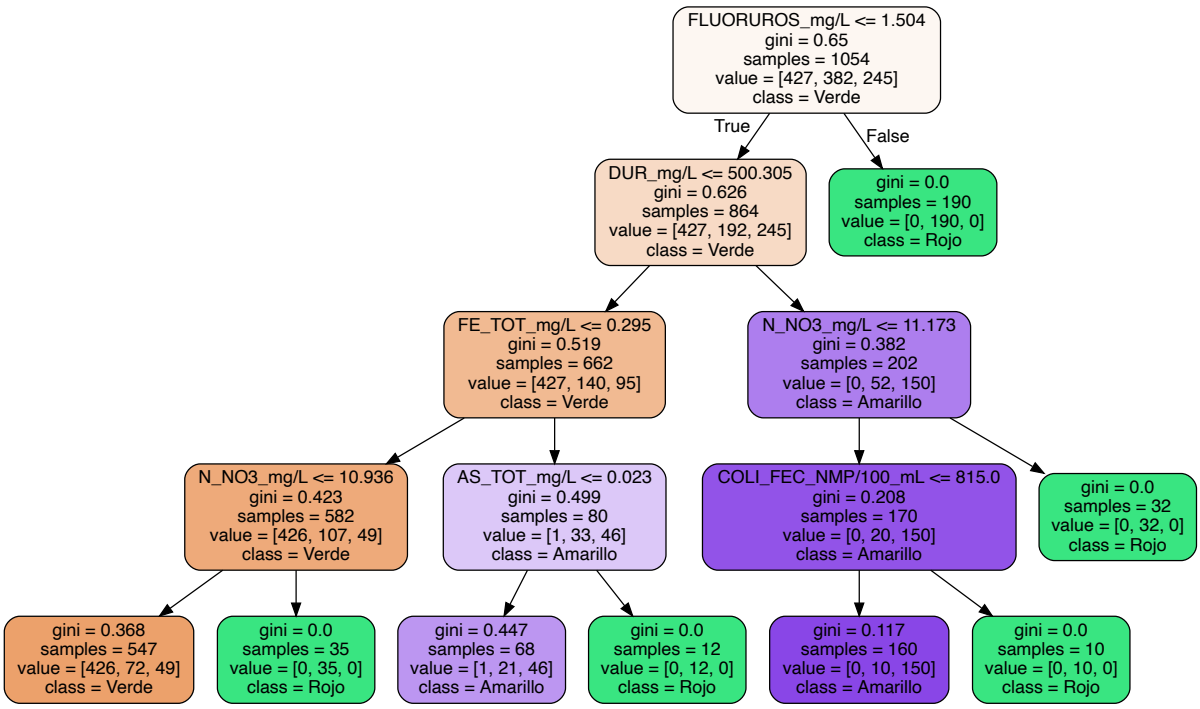
In [18]:
```
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=str( "aguas_subterraneas.dot"),  # path differs in the
 book
        feature_names=columnas,
        class_names=categorias,
        rounded=True,
        filled=True
    )
```

In [19]:
```
from graphviz import Source

Source.from_file("aguas_subterraneas.dot")  # path differs in the book
```

Out[19]:



In [20]:
```
!dot -Tpng {"aguas_subterraneas.dot"} -o {"aguas_subterraneas.png"}
```

In [21]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [22]:
```
tree_reg = DecisionTreeRegressor(max_depth=4, random_state=42)
tree_reg.fit(X, y)
```

Out[22]:
```
DecisionTreeRegressor(max_depth=4, random_state=42)
```

In [23]:
```
importancia=pd.DataFrame(tree_reg.feature_importances_,columns=['importancia'])
importancia['nombres']=columnas
importancia.sort_values(by='importancia', ascending=False)
```

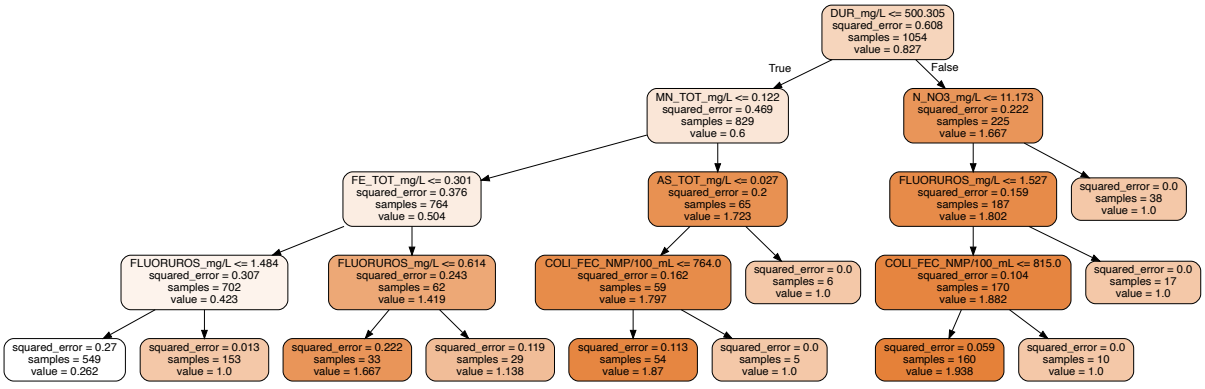
Out[23]:

	importancia	nombres
4	0.434251	DUR_mg/L
12	0.191849	MN_TOT_mg/L
3	0.175533	FLUORUROS_mg/L
13	0.121840	FE_TOT_mg/L
6	0.043786	N_NO3_mg/L
5	0.025294	COLI_FEC_NMP/100_mL
7	0.007447	AS_TOT_mg/L
0	0.000000	ALC_mg/L
1	0.000000	CONDUCT_mS/cm
2	0.000000	SDT_M_mg/L
8	0.000000	CD_TOT_mg/L
9	0.000000	CR_TOT_mg/L
10	0.000000	HG_TOT_mg/L
11	0.000000	PB_TOT_mg/L

In [24]:
```
export_graphviz(
    tree_reg,
    out_file=str("regression_tree.dot"),
    feature_names=columnas,
    rounded=True,
    filled=True
)

Source.from_file("regression_tree.dot")
```

Out[24]:



In [4]:
```
!dot -Tpng {"regression_tree.dot"} -o {"regression_tree.png"}
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "./regression_tree.png", width=100, height=100)
```

```
zsh:1: command not found: dot
```

Out[4]:

Random Forest

In [26]:
```
from sklearn.datasets import make_moons
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=4
2)
print("Datos de entrenamiento: ", round((X_train.shape[0]/X.shape[0])*10
0,2),"%\t", y_train.shape[0])
print("Datos de prueba: \t", round((X_test.shape[0]/X.shape[0])*100,2),
"%\t", y_test.shape[0])
```

```
Datos de entrenamiento:  74.95 %         790
Datos de prueba:         25.05 %         264
```

In [28]:
```
log_clf=LogisticRegression(random_state=42, max_iter=10000)
rnd_clf=RandomForestClassifier(random_state=42)
svm_clf=SVC(probability=True,random_state=42)
```

In [29]:
```
voting_clf=VotingClassifier(estimators=[('lr',log_clf), ('rf', rnd_clf),
('svc', svm_clf)], voting='soft')
```

In [30]:
```
voting_clf.fit(X_train, y_train)
```

Out[30]:
```
VotingClassifier(estimators=[('lr',
                              LogisticRegression(max_iter=10000,
                                                 random_state=42)),
                             ('rf', RandomForestClassifier(random_state
=42)),
                             ('svc', SVC(probability=True, random_state
=42))],
                 voting='soft')
```

In [31]:
```
from sklearn.metrics import accuracy_score
for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
  clf.fit(X_train, y_train)
  y_pred=clf.predict(X_test)
  print(clf.__class__.__name__,accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.8333333333333334
RandomForestClassifier 0.9583333333333334
SVC 0.5984848484848485
VotingClassifier 0.9090909090909091
```

In [32]:
```
prueba=pd.DataFrame({'test':y_test.values.tolist(), 'pred':y_pred.tolist
()} )
prueba
```

Out[32]:

	test	pred
0	1	1
1	2	2
2	0	0
3	2	2
4	0	0
...	...	...
259	1	1
260	1	1
261	1	0
262	1	1
263	0	0

264 rows × 2 columns

In [33]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, cl
assification_report
```

Reporte de clasificación

In [34]:
```
print(classification_report(y_test, y_pred, target_names=categorias))
```

```
              precision    recall  f1-score   support

       Verde       0.91      0.97      0.94       115
        Rojo       0.92      0.91      0.91        88
    Amarillo       0.89      0.80      0.84        61

    accuracy                           0.91       264
   macro avg       0.91      0.89      0.90       264
weighted avg       0.91      0.91      0.91       264
```

Matrix de confusión

In [35]:
```
cf_matrix=confusion_matrix(y_test, y_pred)
cf_matrix
```

Out[35]:
```
array([[111,   2,   2],
       [  4,  80,   4],
       [  7,   5,  49]])
```

In [36]:
```
disp = ConfusionMatrixDisplay(confusion_matrix=cf_matrix, display_labels
=categorias)
fig, ax = plt.subplots(figsize=(10,10))
disp.plot(ax=ax)
plt.show()
```