# Reto -> Entrega 2
# Equipo 118

**Diana Valeria García Briones - A01793006**

**Octavio Alberto García Morán - A01213977**

# Descripción de los datos

**Se tiene una base de datos que contiene la información de 1,068 registros de pozos subterráneos que se encuentran en los 32 estados de la república mexicana. Se ve de la siguiente manera:**

| | CLAVE | SITIO | ORGANISMO_DE_CUENCA | ESTADO | MUNICIPIO | ACUIFERO | SUBTIPO | LONGITUD | LATITUD | PERIODO | ... | CUMPLE_CON_DUR | CUMPLE_CON_CF | CUMPLE_CON_NO3 | CUMPLE_CON_AS | CUMPLE_CON_CD | CUMPLE_CON |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | DLAGU6 | POZO SAN GIL | LERMA SANTIAGO PACIFICO | AGUASCALIENTES | ASIENTOS | VALLE DE CHICALOTE | POZO | -102.02210 | 22.20887 | 2020 | ... | SI | SI | SI | SI | SI | |
| 1 | DLAGU6516 | POZO R013 CAÑADA HONDA | LERMA SANTIAGO PACIFICO | AGUASCALIENTES | AGUASCALIENTES | VALLE DE CHICALOTE | POZO | -102.20075 | 21.99958 | 2020 | ... | SI | SI | SI | SI | SI | |
| 2 | DLAGU7 | POZO COSIO | LERMA SANTIAGO PACIFICO | AGUASCALIENTES | COSIO | VALLE DE AGUASCALIENTES | POZO | -102.28801 | 22.36685 | 2020 | ... | SI | SI | SI | NO | SI | |
| 3 | DLAGU9 | POZO EL SALITRILLO | LERMA SANTIAGO PACIFICO | AGUASCALIENTES | RINCON DE ROMOS | VALLE DE AGUASCALIENTES | POZO | -102.29449 | 22.18435 | 2020 | ... | SI | SI | SI | SI | SI | |
| 4 | DLBAJ107 | RANCHO EL TECOLOTE | PENINSULA DE BAJA CALIFORNIA | BAJA CALIFORNIA SUR | LA PAZ | TODOS SANTOS | POZO | -110.24480 | 23.45138 | 2020 | ... | SI | SI | NO | SI | SI | |

5 rows × 57 columns

**Adicionalmente, se agregó al notebook del Colab el diccionario de datos adicionales**

| | CALIDAD DEL AGUA PARA COLIFORMES FECALES_subterraneas | CRITERIO | DESCRIPCION |
|---|---|---|---|
| 0 | Excelente | COLI_FEC menor a 1.1 | Agua potable. Agua no contaminada o condicion ... |
| 1 | Buena calidad | COLI_FEC mayor o igual de 1.1 y menor o igual ... | Aguas para uso recreativo con contacto primari... |
| 2 | Aceptable | COLI_FEC mayor de 200 y menor o igual a 1000 | Aguas con calidad admisible como fuente de aba... |
| 3 | Contaminada | COLI_FEC mayor de 1000 y menor o igual a 10000 | Aguas con contaminacion bacteriologica. Indic... |
| 4 | Fuertemente contaminada | COLI_FEC mayor de 10000 | Aguas con fuerte contaminacion bacteriologica.... |

# Limpieza de los datos (1/2)

**Como primer paso se revisaron las columnas de las base de datos, se logró identificar una columna sin datos y se eliminó de la base de datos. También, se identificó un símbolo en las columnas numéricas por lo que se procedió a eliminarlo.**

```
[ ]  #Eliminar la columna que no contiene datos
     Datos_Limpios.drop('SDT_mg/L', axis=1, inplace=True)
```

```
[ ]  #Eliminamos los simbolos de las columnas numericas
     Col_Num = ['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']

     Datos_Limpios[Col_Num] = Datos_Limpios[Col_Num].replace({'<':''}, regex=True) #Eliminamos el simbolo de < de todas las columnas numericas
```

**Después, se transformaron las columnas al tipo correcto para asegurarse que se podrán realizar cálculos con las variables numéricas.**

```
[ ]  #Despues que eliminamos los simbolos podemos cambiar las columnas a su tipo correcto (numericas)

     for name in Col_Num:
        mediana = Datos_Limpios[name].median()
        Datos_Limpios[name]= Datos_Limpios[name].astype('float')
        Datos_Limpios[name]=Datos_Limpios[name].replace(np.nan, mediana)
```

# Limpieza de los datos (2/2)

**Después, se revisó la base de datos para verificar que no existieran valores nulos que pudieran sesgar la base de datos y se identificaron valores perdidos. Por lo anterior, se procedió a hacer una sustitución de los datos por el promedio correspondiente.**

```
#Remplazamos valores nulos por la media

for name in Col_Num:
   Datos_Limpios[name].replace(np.nan, np.mean)
```

**Se revisó de nuevo la integridad de los datos para asegurar que sean útiles para los siguientes pasos.**

```
#Valiamos la que ya no haya valores nulos en las variables numericas
Datos_Limpios.isnull().sum()
```

```
CLAVE                   0
SITIO                   0
ORGANISMO_DE_CUENCA     0
ESTADO                  0
MUNICIPIO               0
ACUIFERO                0
SUBTIPO                 0
LONGITUD                0
LATITUD                 0
PERIODO                 0
ALC_mg/L                0
CALIDAD_ALC             4
CONDUCT_mS/cm           0
CALIDAD_CONDUC          6
SDT_M_mg/L              0
CALIDAD_SDT_ra          2
CALIDAD_SDT_salin       2
FLUORUROS_mg/L          0
CALIDAD_FLUO            0
DUR_mg/L                0
CALIDAD_DUR             1
COLI_FEC_NMP/100_mL     0
CALIDAD_COLI_FEC        0
N_NO3_mg/L              0
CALIDAD_N_NO3           1
AS_TOT_mg/L             0
CALIDAD_AS              0
CD_TOT_mg/L             0
CALIDAD_CD              0
CR_TOT_mg/L             0
CALIDAD_CR              0
HG_TOT_mg/L             0
CALIDAD_HG              0
PB_TOT_mg/L             0
CALIDAD_PB              0
MN_TOT_mg/L             0
CALIDAD_MN              0
FE_TOT_mg/L             0
CALIDAD_FE              0
SEMAFORO                0
```

# Selección de variables: X y Y

**Como siguiente paso, se procedió a escoger las variables dependientes (Y) y las independientes (X). Por definición, la variable dependiente es el semáforo, ya que es la variable de interés; por otro lado, las variables numéricas serán las independientes.**

```
[ ]  #Validaos el valor de Y
     Y = pd.DataFrame(Datos['SEMAFORO']) #Es nuestro target
     Y.shape

     #Tomamos las variables numericas
     import seaborn as sns
     Datos_Num = Datos_Limpios[Col_Num]
```

```
[ ]  Datos_Num.shape

     (1068, 14)
```

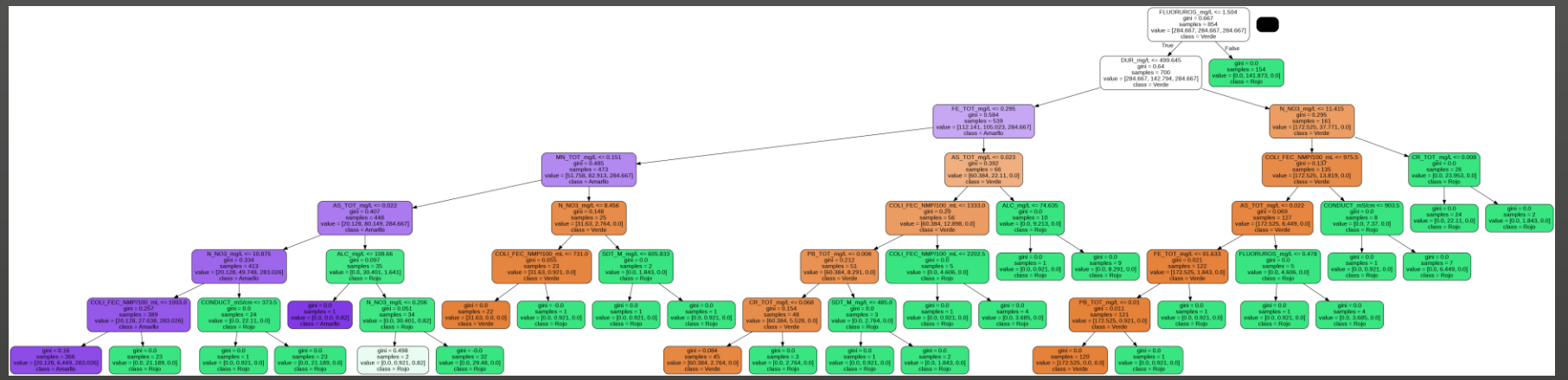
**Adicionalmente, se realizó un Label Encoding y una división de datos procurando que ésta sea balanceada.**

```
[ ]  #Importamos la libreria que vamos a utilizar
     from sklearn import preprocessing

     #Aplicamos el label encoding a la variable Y
     lbe = preprocessing.LabelEncoder() #Salvamos metodo
     Y["ENCODING"]= lbe.fit_transform(Y["SEMAFORO"]) #Creamos un nuevo campo con el valor del encoding
     Y["ENCODING"].unique() #Validamos los valores unicos creados

     array([2, 1, 0])
```

```
[ ]  #Importamos la libreria
     from sklearn.model_selection import train_test_split

     #Asignamos valor a la semilla
     Semilla = 1

     #Dividioms el conjunto en set de entrenamiento y prueba
     x_train, x_test, y_train, y_test = train_test_split(Datos_Num[Datos_Num.columns], Y[['ENCODING']], test_size=0.2, random_state=Semilla, stratify= Y)

     print('X_train:',x_train.shape)
     print('X_test:',x_test.shape)
     print('Y_train:',y_train.shape)
     print('Y_test:',y_test.shape)

     X_train: (854, 14)
     X_test: (214, 14)
     Y_train: (854, 1)
     Y_test: (214, 1)
```

# Construcción de modelos (1/2)

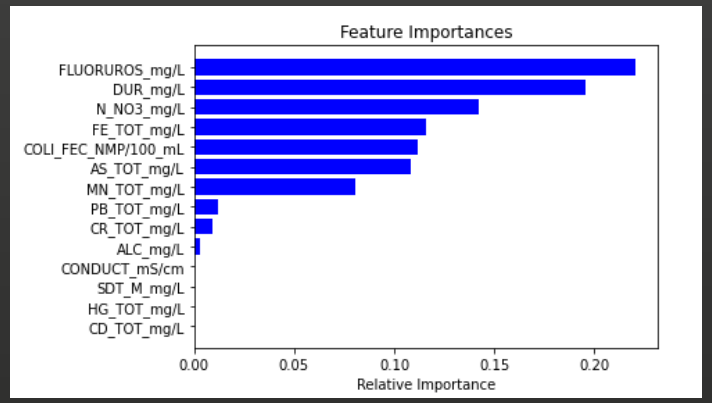
**Primero se procedió con la construcción del Árbol de Decisión, resultando en un árbol con una profundidad de 7 ramas. Las características DUR_mg/L, N_NO3_mg/L y FE_TOT_mg/L determinaron los grupos más grandes.**



**Después, se obtuvo la precisión del árbol y fue de 92.0%; también, se graficó la importancia de cada una de las características, las tres más importantes fueron: FLUORUROS_mg/L, DUR_mg/L y N_NO3_mg/L**

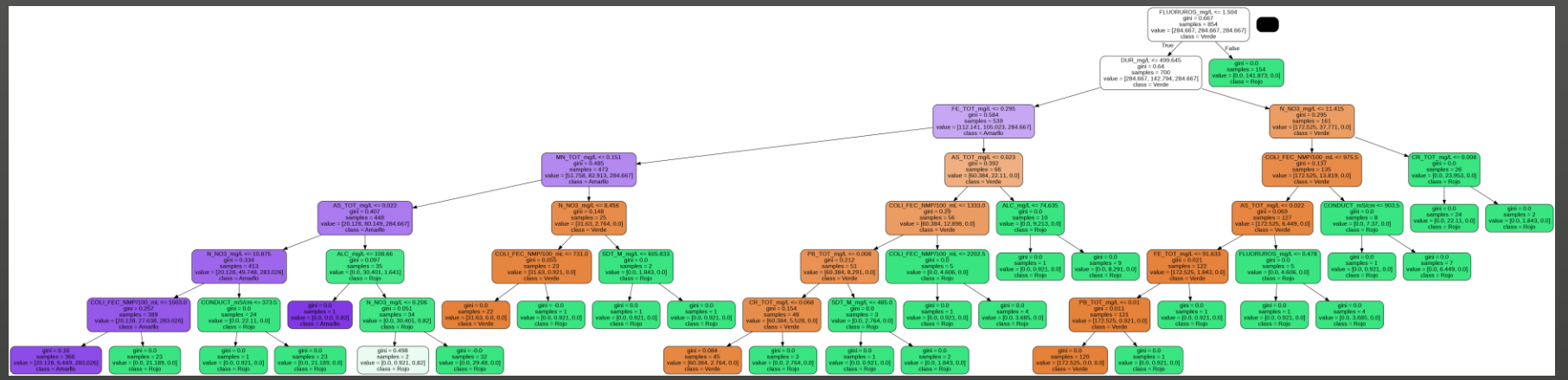
```
[ ]  #Presicion del modelo:
     #from sklearn.metrics import accuracy_score

     accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)
```

```
0.9205607476635514
```



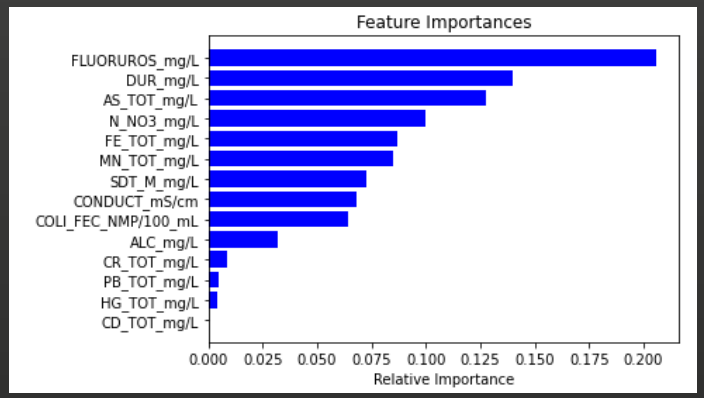
# Construcción de modelos (2/2)

**Después, se procedió con la construcción del Bosque Aleatorio, resultando en un árbol con una profundidad de 7 ramas. Las características FLUORUROS_mg/L/L y COLI_FEC_NMP/100_mL determinaron los grupos más grandes.**



**Después, se obtuvo la precisión del árbol y fue de 93.9%; también, se graficó la importancia de cada una de las características, las tres más importantes fueron: FLUORUROS_mg/L, DUR_mg/L y AS_TOT_mg/L (Variable diferente al árbol de decisión).**

```
[ ]  #Precision del modelo
     y_pred_r = Random_Forest.predict(x_test)
     accuracy_score(y_test, y_pred_r, normalize=True, sample_weight=None)

     0.9392523364485982
```



# Resultados

**Debido a las mínimas diferencias encontradas en las precisiones de los modelos, se analizó cada una de las métricas de votación para tener más información. Se encontró que los resultados son similares, solo cambian en la precisión y en el F1 para la clase 0.**

Votos Soft

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.84 | 0.84 | 49 |
| 1 | 1.00 | 0.90 | 0.95 | 78 |
| 2 | 0.91 | 0.99 | 0.95 | 87 |
| accuracy | | | 0.92 | 214 |
| macro avg | 0.91 | 0.91 | 0.91 | 214 |
| weighted avg | 0.92 | 0.92 | 0.92 | 214 |

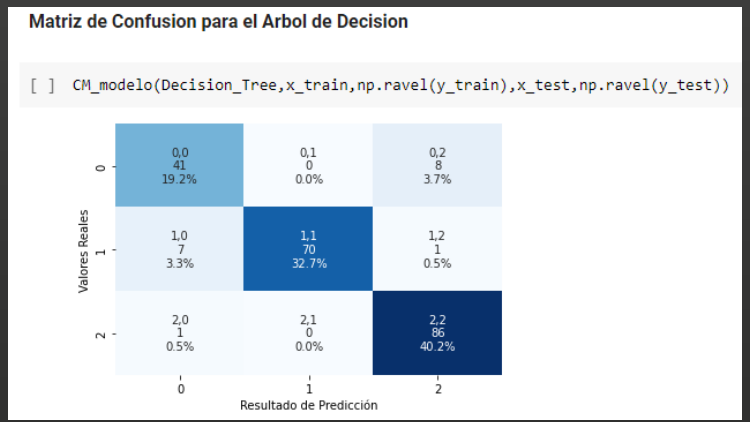
Votos Hard

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.81 | 0.98 | 0.89 | 49 |
| 1 | 1.00 | 0.87 | 0.93 | 78 |
| 2 | 0.98 | 0.98 | 0.98 | 87 |
| accuracy | | | 0.94 | 214 |
| macro avg | 0.93 | 0.94 | 0.93 | 214 |
| weighted avg | 0.95 | 0.94 | 0.94 | 214 |

**Para tener un poco más de claridad sobre esto, se procedió a elaborar la matriz de confusión para ambos.**

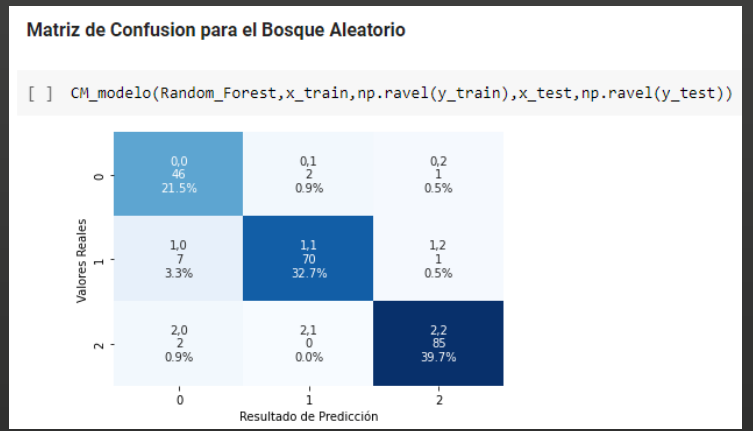
**Matriz de Confusion para el Arbol de Decision**

```
[ ]  CM_modelo(Decision_Tree,x_train,np.ravel(y_train),x_test,np.ravel(y_test))
```




**Matriz de Confusion para el Bosque Aleatorio**

```
[ ]  CM_modelo(Random_Forest,x_train,np.ravel(y_train),x_test,np.ravel(y_test))
```



# Conclusión

**A pesar que la diferencia de precisión parece menor (92.0% vs 93.9%) se puede destacar que esta pequeña diferencia afecta en mayor magnitud a la clase 0.**

**En ambos modelos las clasificaciones de las clases 1 y 2, obtuvimos exactamente los mismos resultados, así como los resultados de aciertos/fallos fueron en las clases, mientras que para la clase 0 se obtuvieron:**

- **Árbol de Decisión: 41 aciertos y 8 fallos.**
- **Bosque Aleatorio: 46 aciertos y 3 fallos.**

**Se recomienda utilizar el bosque aleatorio, pues es el que menos perjudicó a las minorías y por lo tanto es el modelo que tendrá un menor sesgo.**