ENTREGA: RETO-> ENTREGA 2 -> CLASIFICACIÓN-ENSAMBLES Y PRESENTACIÓN EJECUTIVA

ALUMNOS: DANIEL LÓPEZ AGUILAR A01121284
JESÚS CHRISTOPHER LÓPEZ GONZÁLEZ A01201236

MAESTRA: MARÍA DE LA PAZ RICO FERNÁNDEZ

MATERIA: CIENCIA Y ANALÍTICA DE DATOS (GPO 10)

PARTE UNO DEL PROYECTO

- Para realizar la limpieza de datos y transformaciones geográficas utilizamos los siguientes paquetes de Python.

- `pandas`
- `tqdm`
- `numpy`
- `matplotlib`
- `geopandas`
- `shapely`
- `qeds`
- `google.colab`
- `io`

LIMPIEZA DE DATOS

```
RangeIndex:
4141 entries,
0 to 4140 Data
columns (total 56
columns)
```

```
dtypes:
float64(7),
geometry(1),
object(48)
```

```
dtypes:
float64(21)
geometry(1)
object(34)
```

- Primero verificamos si había valores nulos en el dataframe de cuerpos de agua, lo hicimos con el método de .isnull() y si hay valores nulos.
- Revisamos cada dimensión, para entender el tipo de dato, la cantidad de entradas y poder movernos a transformarlos antes de revisarlo estadísticamente.
- Luego se cambian algunos tipos de variables y asignación de datos según lo observado en el análisis de datos anterior.
- En el nuevo Dataframe se llenaron los valores nulos con promedios y se lograron ahora si describir estadísticamente los datos.

```
[75] dataset1.describe()
```

	LONGITUD	LATITUD	PERIODO	DBO_mg/L	DQO_mg/L
count	3493.000000	3493.000000	3493.0	2581.000000	2581.000000
mean	-100.359969	21.046992	2020.0	16.839057	64.311326
std	6.122773	3.893696	0.0	65.151129	149.835921
min	-117.124030	14.534910	2020.0	1.900000	9.900000
25%	-103.882310	18.396070	2020.0	1.900000	11.870000
50%	-99.795530	20.148980	2020.0	2.630000	27.010000
75%	-96.860230	22.828930	2020.0	10.000000	57.000000
max	-86.732150	32.706500	2020.0	1500.000000	2871.250000

8 rows × 21 columns

GEO LOCALIZACIÓN

- Para construir la tabla que luego se usa para hacer el pago, se buscan las coordinadas según la latitud y longitud registrada en los datos y se construye el data frame para geopandas y de ahí usar las coordenadas para la operación espacial que se necesita al construir el mapa.
- También nos traemos el df donde esta la información de geolocalización para luego hacer el query de México

```
latlong=dataset[["LATITUD","LONGITUD"]]
```

	LATITUD	LONGITUD
0	22.24730	-102.33911
1	22.90473	-109.84290

```
[14] dataset["Coordinates"] = list(zip(dataset.LONGITUD, dataset.LATITUD))
     dataset["Coordinates"] = dataset["Coordinates"].apply(Point)
     dataset.head()
```

Coordinates
POINT (-102.33911 22.2473)
POINT (-109.8429

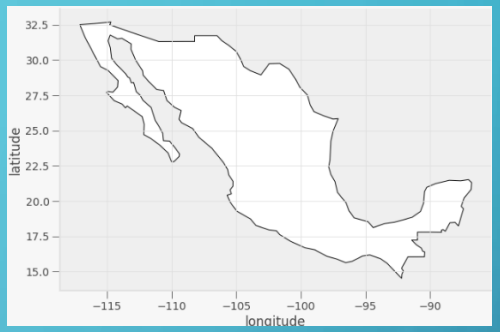
```
world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world = world.set_index("iso_a3")
```

iso_a3	pop_est	continent	name	gdp_md_est	geometry
FJI	920938	Oceania	Fiji	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...

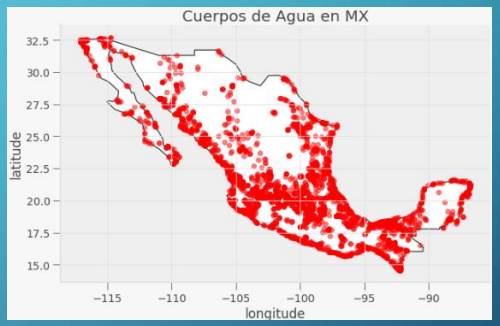
GEO LOCALIZACIÓN

- Luego, con la información construida ya en la tabla y con la información del df de las coordenadas en la tabla de mundo, podemos comenzar a construir el mapa para localizar espacialmente los cuerpos de agua mediante sus coordenadas, en el mapa de México.
- Primero nos traemos un mapa en blanco de México y luego se hace un plot de las coordenadas sobre dicho mapa.
- De hecho si solo haces el plot, los cuerpos de agua están tan distribuidos, que se casi hacen el contorno de México.

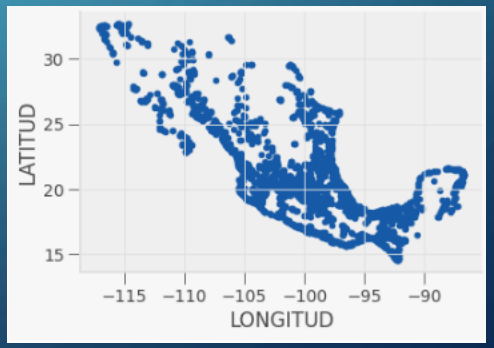
```
world.query
```




```
gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.set_title('Cuerpos de Agua en MX')
```

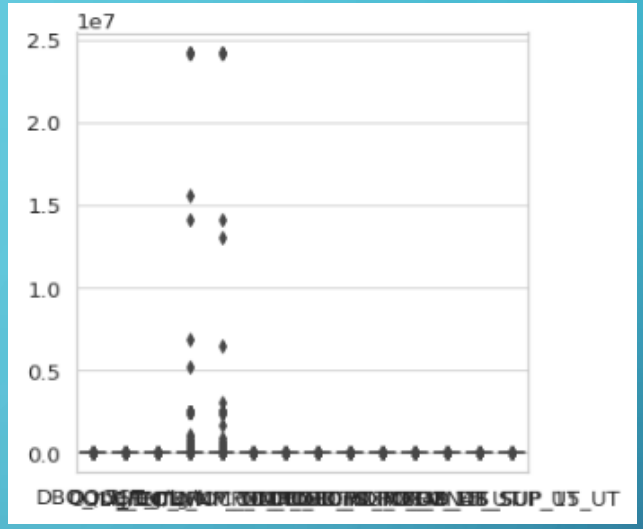


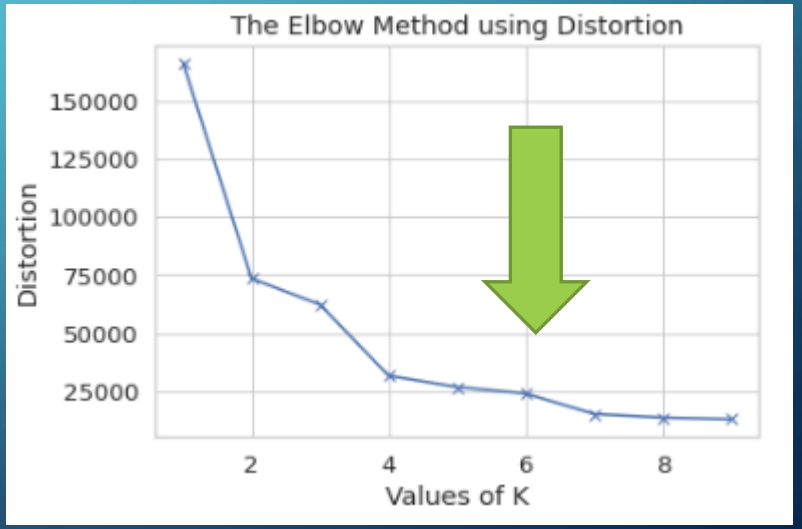

```
latlong.plot.scatter("LONGITUD","LATITUD")
```



DETERMINACIÓN DE CLUSTERS

- Ya con la información limpia, nos movemos a determinar el número de clusters.
- Al realizar un análisis gráfico, observamos que lso clusters parecen estár muy separados entre las observaciones.
- Aplicaremos entonces el “elbow method”, para determinar el número de clusters más eficiente para poder comparar. El número óptimo de clusters es entonces 6





PARTE DOS DEL PROYECTO

- Para realizar los procesos de predicción de datos, utilizamos los siguientes paquetes de Python.

- `seaborn`
- `sklearn.model_selection`
- `sklearn.ensemble`
- `sklearn.datasets`
- `sklearn.metrics`
- `sklearn.tree`
- `sklearn.model_selection`
- `sklearn.model_selection`
- `sklearn.metrics`
- `sklearn.tree`

MAPPING DE DATOS Y SPLIT

- Continuamos con el proceso de limpieza de datos y revisión, como en la parte uno, también se reemplazaron datos que estaban vacíos.
- Este proceso se llevó a cabo para tener el mapping de los datos listados y poder armar luego el semáforo.
- Del Dataframe mapeado, se generan las variables X y Y para luego pasar a los classifiers tanto de random forest, como de decisión tree.

MAPPING

```
d = {'Verde': 0, 'Amarillo': 1, 'Rojo': 2}
dataset['SEMAFORO_'] = dataset['SEMAFORO'].map(d)
d = {'LERMA SANTIAGO PACIFICO': 0, 'FRONTERA SUR': 1, 'PACIFICO SUR': 2,'BAL
dataset['ORGANISMO_DE_CUENCA_'] = dataset['ORGANISMO_DE_CUENCA'].map(d)
d = {'RIO': 0, 'PRESA': 1, 'LAGUNA': 2,'OCEANO-MAR': 3, 'LAGO': 4, 'BAHIA':
dataset['SUBTIPO_'] = dataset['SUBTIPO'].map(d)
d = {'LOTICO': 0, 'COSTERO': 1, 'LENTICO': 2}
dataset['GRUPO_'] = dataset['GRUPO'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_DBO_'] = dataset['CUMPLE_CON_DBO'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_DQO_'] = dataset['CUMPLE_CON_DQO'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_SST_'] = dataset['CUMPLE_CON_SST'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_CF_'] = dataset['CUMPLE_CON_CF'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_E_COLI_'] = dataset['CUMPLE_CON_E_COLI'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_ENTEROC_'] = dataset['CUMPLE_CON_ENTEROC'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_OD_'] = dataset['CUMPLE_CON_OD'].map(d)
d = {'SI': 0, 'NO': 1, 'ND': 2}
dataset['CUMPLE_CON_TOX_'] = dataset['CUMPLE_CON_TOX'].map(d)
```

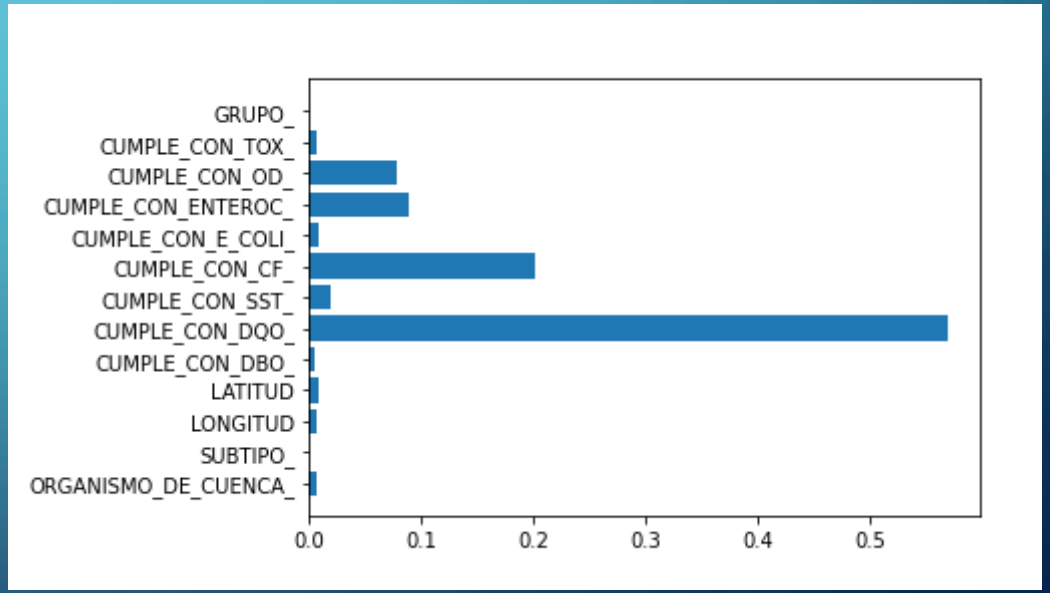
DATA SPLIT

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.3)
RF = RandomForestClassifier(random_state=42)
DT = DecisionTreeClassifier(random_state=42)
```

DECISION TREE

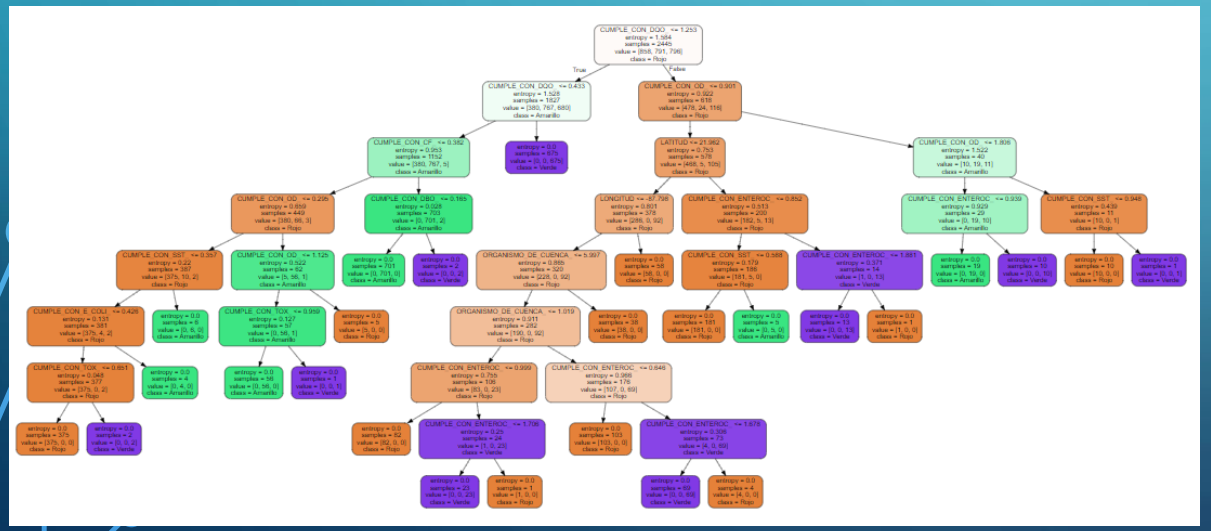
- La aplicación de decisión tree se aplica sobre el dataframe en el que ya se han mapeado los datos para las decisiones.
- Generamos las “features of importance”, utilizando este método y llegamos al array que las muestra como valores.
- Realizamos una gráfica para ver cuál es más grande y con la imagen ya podemos determinar claramente las dimensiones que tienen más importancia.

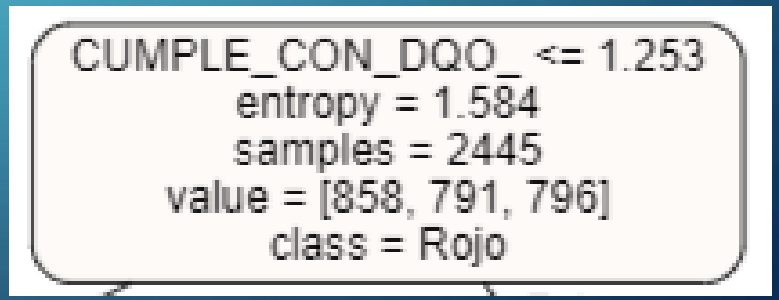
```
array([0.00695266, 0.        , 0.00663253,
0.00771975, 0.00511157, 0.56957125,
0.01884504, 0.201886  , 0.00827181,
0.0894475 , 0.07904013, 0.00652177, 0.
])
```



DECISION TREE

- De la aplicación del método para árboles de decisión se genera la gráfica donde se puede ir viendo la evaluación que el modelo fue haciendo de las dimensiones y como la de más importancia que esta hasta arriba muestra lso valores de decisión más altos.

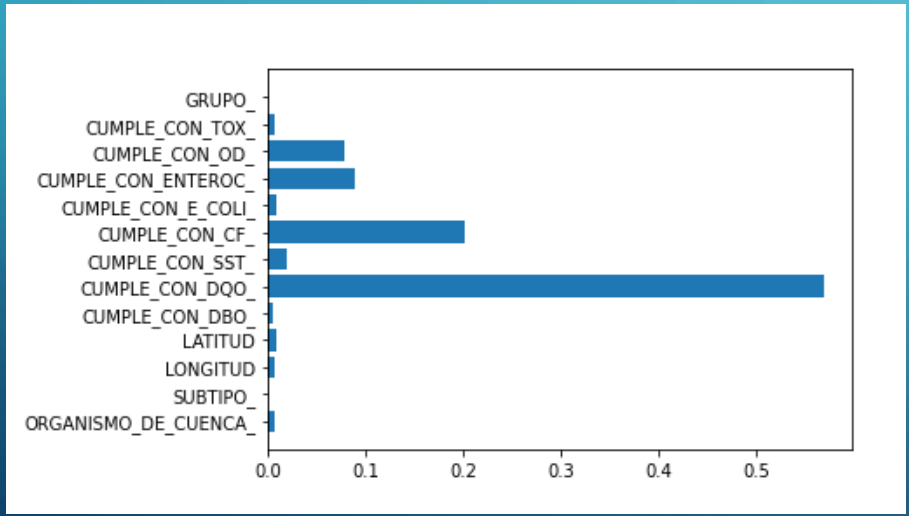




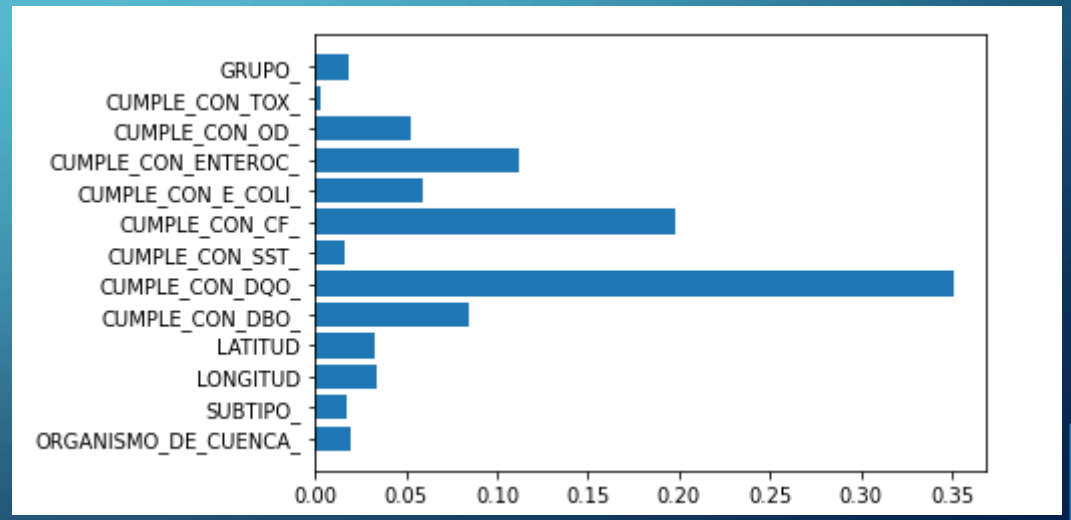
DECISION TREE VS RANDOM FOREST

- Aplicamos después el proceso de Random Forest al mismo set de datos y básicamente tenemos el mismo resultado en cuanto a las dimensiones más significativas:

Decision Tree



Random Forest



ANALISIS EXTRA DE CHI-SQUEARED TEST

- Luego de toda la limpieza y revisión de datos para determinar clusters y dimensiones más importantes, buscamos un algoritmo que nos dejar validar si existían correlaciones fuertes entre las variables categóricas de localidad representada por el estado y la clasificación del agua, que luego del análisis de random forest encontramos era CUMPLE CON DBO

Este análisis nos arrojó un Ch12 bajo, y un p-value de 96%, lo cuál nos dice que estas variables no son estadísticamente significativas entre ellas.

Es decir la locación no esta correlacionada con la toxicidad del agua.

```
print("Chi2 :", chi2)
print("p-value :",p)
print("degree for freedom: ",dof)
```

```
Chi2 : 0.0019360269360269287
p-value : 0.9649041598805919
degree for freedom:  1
```

CONCLUSIONES

- Recomendamos usar nuestro decision tree debido a que el performance es muy similar a random forest. A pesar de que pudiera llegar a generar un posible overfit, es un modelo mas simple, menos costoso y fácil de explicar.
- La limpieza y entendimiento de datos son clave para poder realizar una correcta explicación estadística de los modelos para seleccionar el análisis correcto a aplicar entre las variables.