RETO 2

AGUAS - SUPERFICIALES

JOSE SANTIAGO RUEDA ANTONIO
A01794118
ROBERTO SANTIAGO OLIVA
A01374957

LIMPIEZA

Realizamos una limpieza del df ya que sin esto no podemos realizar los pasos posteriores:

```
columns = ['DBO_mg/L', 'DQO_mg/L', 'SST_mg/L', 'COLI_FEC_NMP_100mL', 'E_COLI_NMP_100mL',

for i in columns:

  df_superf[i] = df_superf[i].astype('str')
  df_superf[i] = df_superf[i].str.replace('<25','25',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.2','0.2',regex=True)
  df_superf[i] = df_superf[i].str.replace('<1','1',regex=True)
  df_superf[i] = df_superf[i].str.replace('<2','2',regex=True)
  df_superf[i] = df_superf[i].str.replace('<3','3',regex=True)
  df_superf[i] = df_superf[i].str.replace('<20','20',regex=True)
  df_superf[i] = df_superf[i].str.replace('<1.1','1.1',regex=True)
  df_superf[i] = df_superf[i].str.replace('<10','10',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.02','0.02',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.01','0.01',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.003','0.003',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.005','0.004',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.0005','0.0004',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.0015','0.0015',regex=True)
  df_superf[i] = df_superf[i].str.replace('<0.025','0.025',regex=True)
  df_superf[i]= df_superf[i].astype('float')
  mean_a = df_superf[i].mean()
  df_superf[i] = df_superf[i].replace(np.nan,mean_a)
```

```
(df_superf.isnull().sum() * 100 / len(df_superf)).sort_values(ascending=False)
```

LIMPIEZA

Con la limpieza eliminamos algunas columnas:

```
[ ]  columns_to_drop = ['CLAVE','TOX_D_48_FON_UT',
                        'CALIDAD_TOX_D_48_FON',
                         'TOX_FIS_FON_15_UT',
                          'CALIDAD_TOX_FIS_FON_15',
                           'CALIDAD_OD_PORC_MED',
                            'CALIDAD TOX_D_48_SUP',
                             'CALIDAD_ENTEROC',
                             'CALIDAD_OD_PORC_FON',
                             'CALIDAD_OD_PORC_SUP',
                             'CALIDAD_TOX_FIS_SUP_15',
                             'CALIDAD_OD_PORC',
                             'CALIDAD_TOX_D_48',
                             'CALIDAD_TOX_V_15']
     df_superf.drop(columns_to_drop, inplace=True, axis=1)
```

```
[ ]  df_superf.shape
```

```
     (4141, 41)
```

```
[ ]  names = df_superf.columns.to_list()
     types = df_superf.dtypes.to_dict()
     df_super_clean = pd.DataFrame(SimpleImputer(strategy='most_frequent').fit_transform(
         df_superf), columns=names).astype(types)
```

SELECCIÓN DE X, Y

Posterior a eso, seleccionamos las variables independientes y la dependiente:

```
[ ] X = df_super_clean.drop('SEMAFORO', axis=1)
    y = LabelEncoder().fit_transform(df_super_clean['SEMAFORO'])
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, random_state=0)
```

```
[ ] X.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
LONGITUD	4141.0	-101.126668	5.898307	-117.12403	-105.259510	-100.958600	-97.680560	-8.673215e+01
LATITUD	4141.0	20.040593	4.271863	14.53491	16.839720	19.454840	21.998090	3.270650e+01
PERIODO	4141.0	2020.000000	0.000000	2020.00000	2020.000000	2020.000000	2020.000000	2.020000e+03
DBO_mg/L	4141.0	16.886481	51.423215	2.00000	2.000000	13.800000	16.886481	1.500000e+03
DQO_mg/L	4141.0	64.332985	118.277690	10.00000	20.700000	64.332985	64.332985	2.871250e+03
SST_mg/L	4141.0	102.148144	405.576843	10.00000	12.000000	32.000000	102.148144	9.430000e+03
COLI_FEC_NMP_100mL	4141.0	95688.808675	922925.127897	3.00000	1333.000000	24000.000000	95688.808675	2.419600e+07
E_COLI_NMP_100mL	4141.0	79337.687452	830107.982666	3.00000	150.000000	15531.000000	79337.687452	2.419600e+07
ENTEROC_NMP_100mL	4141.0	1086.709845	2010.985728	3.00000	1086.709845	1086.709845	1086.709845	2.419600e+04
OD_PORC	4141.0	66.840289	20.692827	10.00000	66.840289	66.840289	66.840289	2.261000e+02
OD_PORC_SUP	4141.0	81.459975	17.855915	10.00000	81.459975	81.459975	81.459975	2.890000e+02
OD_PORC_MED	4141.0	71.590965	9.035034	10.00000	71.590965	71.590965	71.590965	1.330000e+02
OD_PORC_FON	4141.0	66.869239	13.471635	10.00000	66.869239	66.869239	66.869239	1.460000e+02

PIPELINE

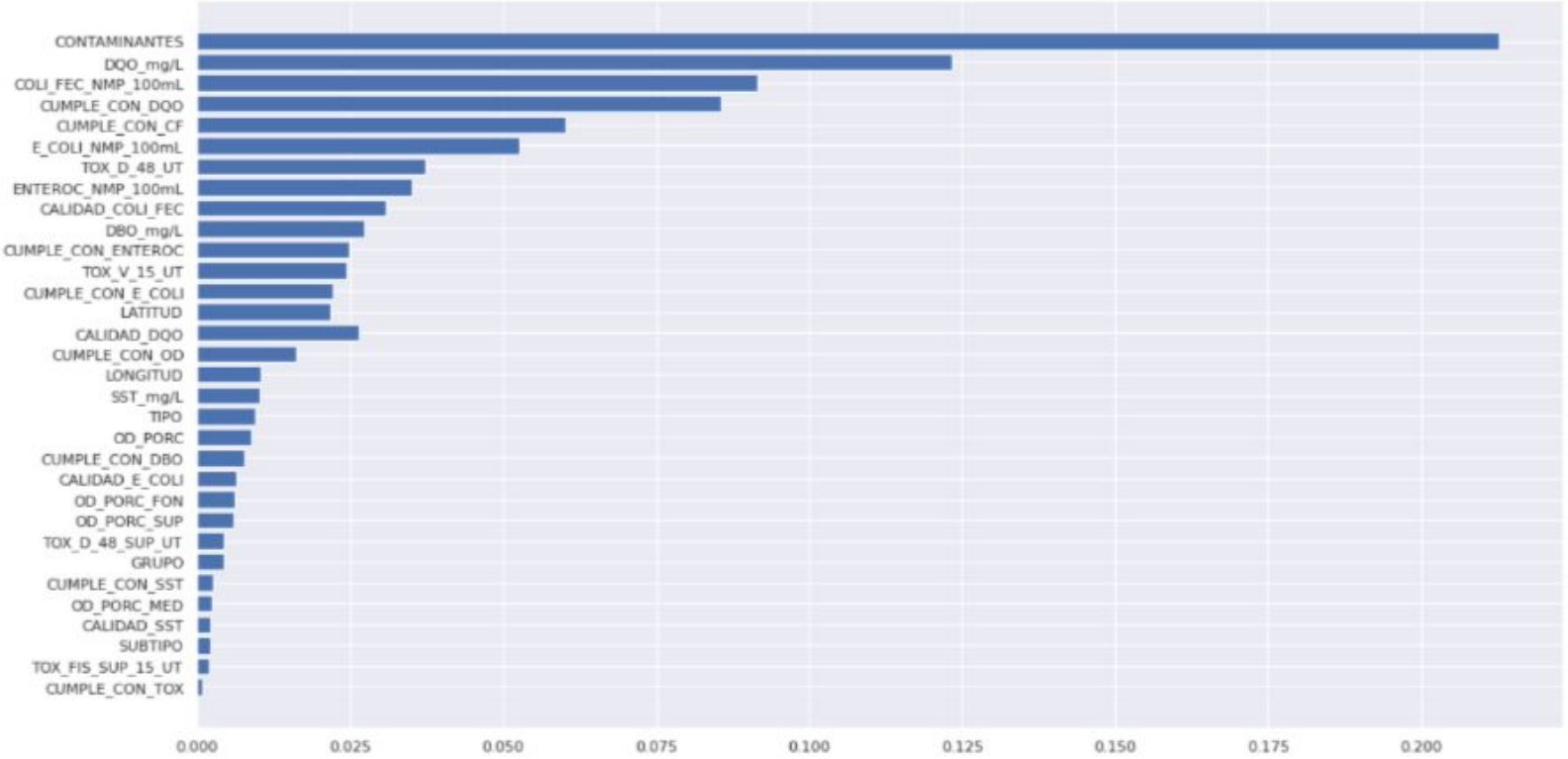
Con todo lo anterior pudimos construir el pipeline:

```
[ ]
    pipe_geo = Pipeline(steps= [('geo', SimpleImputer(strategy='most_frequent'))])
    geo = ['LONGITUD','LATITUD']


    pipe_num = Pipeline(steps= [('num', MinMaxScaler())])
    num = ['DBO_mg/L','DQO_mg/L','SST_mg/L','COLI_FEC_NMP_100mL','E_COLI_NMP_100mL',
           'ENTEROC_NMP_100mL','OD_PORC','OD_PORC_SUP','OD_PORC_MED','OD_PORC_FON','TOX_D_48_UT',
           'TOX_V_15_UT','TOX_D_48_SUP_UT','TOX_FIS_SUP_15_UT']

    pipe_cat = Pipeline(steps= [('cat', OrdinalEncoder())])
    cat_columns = ['TIPO','SUBTIPO', 'CALIDAD_DQO','CALIDAD_DQO','CALIDAD_SST','CALIDAD_COLI_FEC','CALIDAD_E_COLI','GRUPO','CONTAMINANTES']

    pipe_bin = Pipeline(steps= [('bin', OrdinalEncoder())])
    binarias = ['CUMPLE_CON_DBO','CUMPLE_CON_DQO','CUMPLE_CON_SST','CUMPLE_CON_CF','CUMPLE_CON_E_COLI','CUMPLE_CON_ENTEROC','CUMPLE_CON_OD','CUMPLE_CON_TOX']
```

```
[ ] transformada = ColumnTransformer(transformers=[
                                        ('geo_columns', pipe_geo, geo),
                                        ('num', pipe_num, num),
                                        ('cat', pipe_cat, cat_columns),
                                        ('bin',pipe_bin,binarias)])
```

```
[ ] columns = geo + num + cat_columns + binarias
```

FEATURE IMPORTANCE



Posteriormente graficamos el feature importance, denotando que los contaminantes es lo más importante para el análisis de nuestro modelo.

CARACTERÍSTICAS MAS IMPORTANTES

```
def validacion_scores(X,y):
  resultados = list()
  modelos = [RandomForestClassifier(random_state= 0), DecisionTreeClassifier(random_state= 0)]
  nombres = ['RandomForest', 'DecisionTree']
  for nombre,modelo in zip(nombres,modelos):
    KFold = RepeatedKFold(n_splits=8, n_repeats=3, random_state=0)
    score = cross_validate(estimator=modelo, X=transformada.fit_transform(X), y=y, cv=KFold)
    resultados.append({'nombre':nombre,'score':score['test_score'].mean()})
  return resultados
```

```
validacion_scores(X_train,y_train)
```

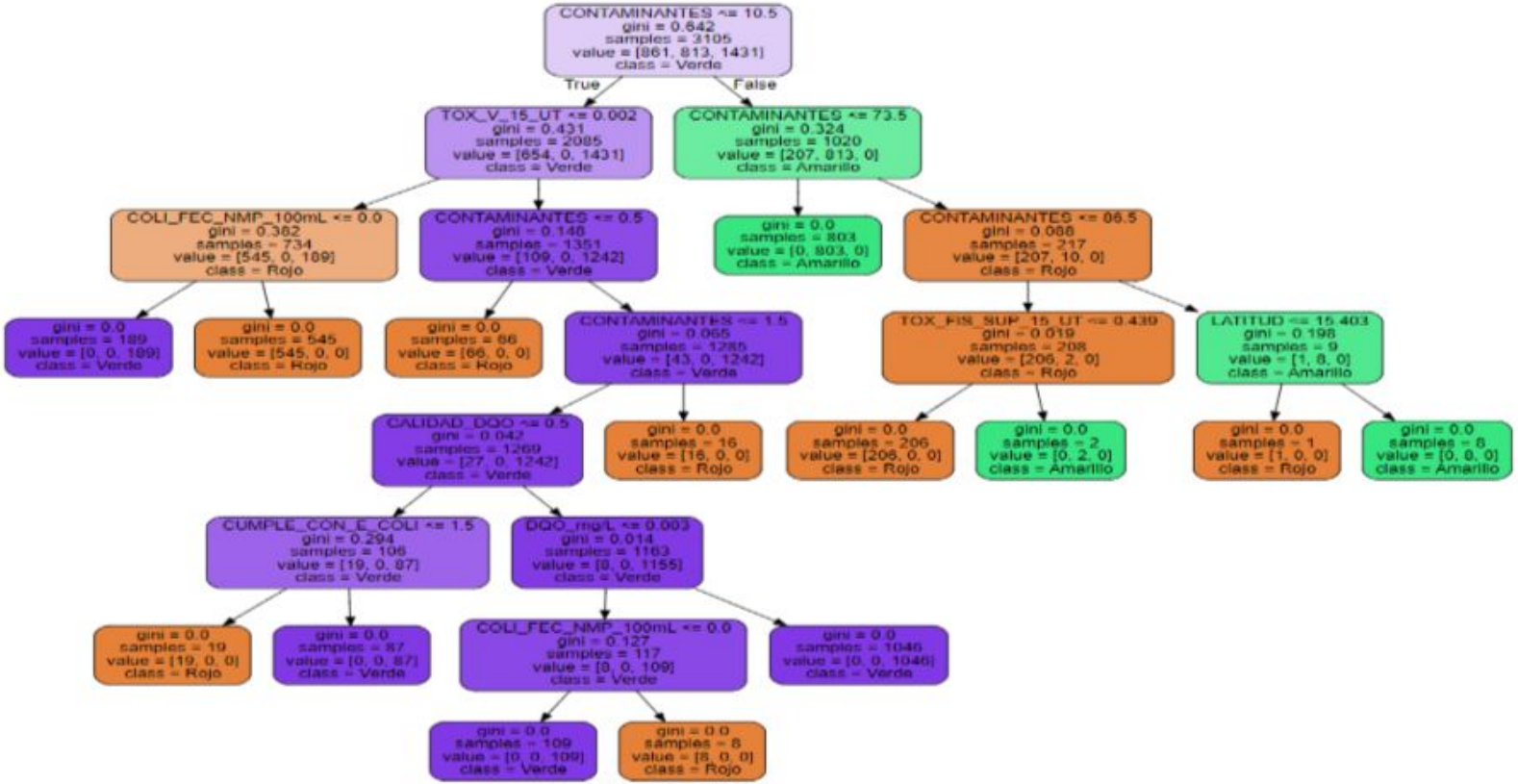
```
[{'nombre': 'RandomForest', 'score': 0.9980675624342972},
 {'nombre': 'DecisionTree', 'score': 0.9974229564748804}]
```

```
from graphviz import Source
```

Basandonos en las características más importantes, construimos un algoritmo con dichas características.

DECISION TREE

Con todo lo anterior pudimos construir nuestro decision tree:



CONCLUSIONES

- Podemos concluir que nuestro modelo quedó algo sobreentrenado debido a que realizamos bastantes transformaciones, además de que si disminuimos las caracteristicas del mismo, este se vuelve menos complejo.

- Por último, podemos concluir que es mejor utilizar un árbol de decisión que un random forest, en el caso de nuestro modelo.

¡Gracias!

EQUIPO 19