UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

DATOS MASIVOS

Shamir Alejandro Rodriguez Ojeda

Matricula: 1423061

Herramienta: Colab

```
##############################################################################
###                   PIA - SHAMIR A. RODRIGUEZ OJEDA                       ##
###                     Maestria en Ciencia de Datos                        ##
###                              12/04/2024                                 ##
###                                                                         ##
##############################################################################
```

Instalación Librerias

```
!pip install --q https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
!pip install --q pyod       # Detección de atipicos +30 algoritmos
!pip install --q catboost   # Catboost
!pip install --q imblearn   # Balanceo de clases
```

```
    - 17.8 MB 26.6 MB/s 0:00:01
 Preparing metadata (setup.py) ... done
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 104.8/104.8 kB 1.4 MB/s eta 0:00:00
 Preparing metadata (setup.py) ... done
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 686.1/686.1 kB 7.5 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 5.4 MB/s eta 0:00:00
 Building wheel for ydata-profiling (setup.py) ... done
 Building wheel for htmlmin (setup.py) ... done
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.0/165.0 kB 4.1 MB/s eta 0:00:00
 Preparing metadata (setup.py) ... done
 Building wheel for pyod (setup.py) ... done
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 MB 8.7 MB/s eta 0:00:00
```

```
from google.colab import drive
from pathlib import Path
```

```
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
```

```python
from google.colab import drive
from pathlib import Path
```

```python
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
```

```
<ipython-input-3-c03a24be5f20>:3: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport
```

```python
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```python
from pyod.models.lof import LOF
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import average_precision_score
from sklearn import preprocessing
from sklearn.model_selection import GridSearchCV


from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from imblearn.over_sampling import ADASYN
from collections import Counter

from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from catboost import CatBoostClassifier, Pool


import seaborn as sns
import matplotlib.pyplot as plt
from string import ascii_letters

```

```python

import seaborn as sns
import matplotlib.pyplot as plt
from string import ascii_letters

import warnings
warnings.filterwarnings("ignore")
```

```python
SEED=111
```

Funciones

```python
def grafica_correla(df, paleta="Greens"):
  import seaborn as sns
  import matplotlib.pyplot as plt
  from string import ascii_letters
  fig, ax = plt.subplots(figsize=(6, 6))
  mask = np.zeros_like(df.corr())
  mask[np.triu_indices_from(mask)] = 1
  sns.heatmap(df.corr(), mask= mask, cmap=paleta)
```

```python
def refusion_dimension(df, umbral=0.95):
  corr_df = df.corr().abs()
  mask = np.triu(np.ones_like(corr_df, dtype=bool))
  correlacion = df.corr()
  tri_df = correlacion.mask(mask)
  to_drop = [c for c in tri_df.columns if any(tri_df[c] > umbral)]
  return to_drop
```

Para poder realizar la entraga siguiendo el formato del archivo TRAIN, se requiere saber las variables iniciales dadas para el caso de uso.

```
etiquetas= train.iloc[:,:-7].columns
len(etiquetas)
```

```
27
```

Se creo un reporte previo para la exploración de los datos EDA. El archivo html tiene varios analisis sobre los datos iniciales.

```
profile = ProfileReport(train, title='Pandas Profiling Report', html={'style':{'full_width':False}})
```

```
profile.to_notebook_iframe()
```

```
profile.to_file(output_file=link_lectura+name_report)
```

```
Export report to file: 100% 1/1 [00:00<00:00,  1.36it/s]
```

```
profile
```

```
Summarize dataset: 100% 619/619 [02:47<00:00,  2.18it/s, Completed]
Generate report structure: 100% 1/1 [00:23<00:00, 23.02s/it]
Render HTML: 100% 1/1 [00:20<00:00, 20.99s/it]
```

Vale la pena verificar cuantos nulos tiene el dataframe dado que muchas veces es/será un dolor de cabeza para la creación de un modelo de ML

```
train.isnull().sum()
```

```
X_Minimum                0
X_Maximum                0
Y_Minimum                0
Y_Maximum                0
Pixels_Areas             0
X_Perimeter              0
Y_Perimeter              0
Sum_of_Luminosity        0
Minimum_of_Luminosity    0
Maximum_of_Luminosity    0
Length_of_Conveyer       0
TypeOfSteel_A300         0
TypeOfSteel_A400         0
Steel_Plate_Thickness    0
Edges_Index              0
Empty_Index              0
Square_Index             0
Outside_X_Index          0
Edges_X_Index            0
Edges_Y_Index            0
Outside_Global_Index     0
LogOfAreas               0
Log_X_Index              0
Log_Y_Index              0
Orientation_Index        0
Luminosity_Index         0
SigmoidOfAreas           0
Pastry                   0
Z_Scratch                0
K_Scatch                 0
Stains                   0
Dirtiness                0
Bumps                    0
Other_Faults             0
dtype: int64
```

```
train.describe()
```

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximum_of_Luminosity	...	Orientation_Index	Luminosity_Index	SigmoidOfAreas	Pa
count	1455.000000	1455.000000	1.455000e+03	1.455000e+03	1455.000000	1455.000000	1455.000000	1.455000e+03	1455.000000	1455.000000	...	1455.000000	1455.000000	1455.000000	1455.00
mean	570.035052	616.308155	1.678894e+06	1.678950e+06	1506.098282	113.644674	85.914089	2.088553e+05	84.851168	130.021993	...	0.087881	-0.130914	0.582273	0.08
std	516.641171	493.543539	1.826262e+06	1.826279e+06	5540.549250	331.232990	457.996747	5.333202e+05	32.375864	18.461963	...	0.499567	0.151795	0.340529	0.27
min	0.000000	4.000000	6.712000e+03	6.724000e+03	2.000000	2.000000	1.000000	2.500000e+02	0.000000	37.000000	...	-0.973300	-0.998900	0.119000	0.00
25%	56.000000	194.000000	5.020040e+05	5.020045e+05	82.500000	15.000000	13.000000	9.374000e+03	63.000000	124.000000	...	-0.295200	-0.195200	0.243200	0.00
50%	435.000000	466.000000	1.223690e+06	1.223696e+06	172.000000	26.000000	25.000000	1.898600e+04	90.000000	127.000000	...	0.098800	-0.132600	0.496300	0.00
75%	1049.500000	1070.000000	2.183652e+06	2.183658e+06	794.000000	82.500000	83.000000	8.168850e+04	106.000000	140.000000	...	0.509850	-0.066100	0.999800	0.00
max	1705.000000	1713.000000	1.298766e+07	1.298769e+07	152655.000000	10449.000000	18152.000000	1.159141e+07	203.000000	247.000000	...	0.991700	0.642100	1.000000	1.00

8 rows × 34 columns

```
train.head()
```

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximum_of_Luminosity	...	Orientation_Index	Luminosity_Index	SigmoidOfAreas	Pastry	Z_Scratch
1309	1343	1351	46714	46725	56	11	11	5729	90	117	...	0.2727	-0.2007	0.1934	0	0
1132	354	370	409908	409916	116	19	10	12174	82	126	...	-0.3750	-0.1801	0.2792	0	0
1230	1056	1077	4364066	4364076	106	35	22	10747	89	116	...	-0.5238	-0.2079	0.3509	0	0
793	816	822	899659	899663	18	8	6	2384	127	141	...	0.0000	0.0047	0.1449	0	0
1762	78	94	1523381	1523431	463	69	54	60518	118	141	...	0.6800	0.0213	0.9830	0	0

5 rows × 34 columns

Se va a proceder a eliminar las clases individuales del DF train para despues concadenar la columna **target**

```
train.drop(mask, axis=1, inplace=True)
train
```

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximum_of_Luminosity	...	Outside_X_Index	Edges_X_Index	Edges_Y_Index
1309	1343	1351	46714	46725	56	11	11	5729	90	117	...	0.0059	0.7273	1.0000
1132	354	370	409908	409916	116	19	10	12174	82	126	...	0.0118	0.8421	1.0000
1230	1056	1077	4364066	4364076	106	35	22	10747	89	116	...	0.0154	0.6000	0.4545
793	816	822	899659	899663	18	8	6	2384	127	141	...	0.0044	0.7500	1.0000
1762	78	94	1523381	1523431	463	69	54	60518	118	141	...	0.0118	0.2319	0.9259
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
955	677	686	3379334	3379344	69	12	10	7774	95	132	...	0.0054	0.7500	1.0000
1702	114	134	115356	115418	532	75	73	54451	94	111	...	0.0144	0.2667	0.8493
194	109	121	560425	560445	124	18	20	13549	97	124	...	0.0089	0.6667	1.0000
1335	373	386	1058240	1058279	308	29	39	37264	113	132	...	0.0095	0.4483	1.0000
827	39	178	2627229	2627391	7998	558	343	889727	49	140	...	0.0993	0.2491	0.4723

1455 rows × 27 columns

UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

Reversa de One-Hot Encoding

Se crea la columna target que toma el one-hot encoding y se hace el proceso de reversa.

```
train['target'] = classes.apply(lambda x: x.idxmax(), axis = 1)
train
```

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximum_of_Luminosity	...	Edges_X_Index	Edges_Y_Index	Outside_Global_Index	LogOfAreas	Log_X_Index
1309	1343	1351	46714	46725	56	11	11	5729	90	117	...	0.7273	1.0000	1.0	1.7482	0.9031
1132	364	370	409908	409918	116	19	10	12174	82	126	...	0.8421	1.0000	0.0	2.0645	1.2041
1338	1056	1077	4364056	4364076	106	35	22	10747	89	116	...	0.6000	0.4545	0.0	2.0253	1.3222
783	816	822	899659	899665	18	8	6	2384	127	141	...	0.7500	1.0000	0.5	1.2553	0.7782
1762	78	94	1523381	1523431	463	69	54	60518	118	141	...	0.2319	0.9259	1.0	2.6656	1.2041
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
955	677	686	3379334	3379344	69	12	10	7774	95	132	...	0.7500	1.0000	1.0	1.8388	0.9542
1792	114	134	115356	115418	532	76	73	54451	94	111	...	0.2667	0.8493	1.0	2.7259	1.3010
194	109	121	560425	560445	124	18	20	13549	97	124	...	0.6667	1.0000	1.0	2.0934	1.0792
1335	373	386	1058240	1058279	308	29	39	37264	113	132	...	0.4483	1.0000	1.0	2.4885	1.1139
827	39	178	2627229	2627391	7998	558	343	889727	49	140	...	0.2491	0.4723	1.0	3.9030	2.1430

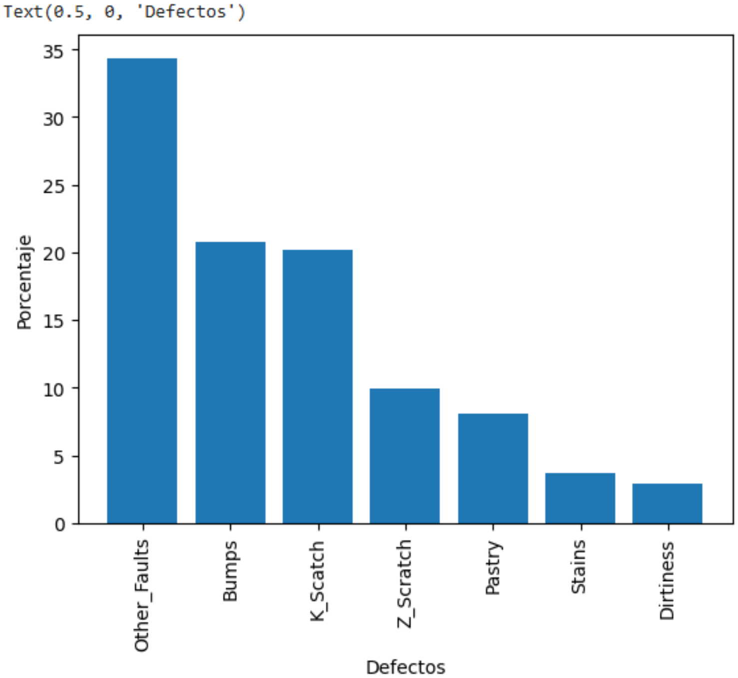
1455 rows × 28 columns

```
aux
```

	class	count
0	Other_Faults	34.364261
1	Bumps	20.756014
2	K_Scatch	20.206186
3	Z_Scratch	9.965636
4	Pastry	8.109966
5	Stains	3.711340
6	Dirtiness	2.886598

```
aux= ((train['target'].value_counts()/train.shape[0])*100).rename_axis('class').reset_index()
plt.bar(aux['class'], aux['count'], )
plt.xticks(rotation=90)
plt.ylabel('Porcentaje')
plt.xlabel('Defectos')
```

```
Text(0.5, 0, 'Defectos')
```



Nulos

En celdas anteriores como previo a la carga de los datos se exploro rapidamente los valores nulos ahora se procederá a revisar si existe al menos 1 en todo el DF.

```
train.isnull().sum().sum()
```

```
0
```

Uno de los problemas tipicos al trabajar con datos es que los datos vengan en otro formato, es decir cuando uno ve el dato es un numero 1 pero el sistema lo ha leido como texto '1' causando complicaciones en los modelos.

formato de las variables (tipo)

```
train.dtypes
```

```
X_Minimum                 int64
X_Maximum                 int64
Y_Minimum                 int64
Y_Maximum                 int64
Pixels_Areas              int64
X_Perimeter               int64
Y_Perimeter               int64
Sum_of_Luminosity         int64
Minimum_of_Luminosity     int64
Maximum_of_Luminosity     int64
Length_of_Conveyer        int64
TypeOfSteel_A300          int64
TypeOfSteel_A400          int64
Steel_Plate_Thickness     int64
Edges_Index             float64
Empty_Index             float64
Square_Index            float64
Outside_X_Index         float64
Edges_X_Index           float64
```

Duplicados

Validación de los duplicados en el DF

```
aux=train.shape[0]
train.drop_duplicates(keep='first',inplace=True)
aux==train.shape[0]
```

```
True
```

Al parecer no existe duplicados en el DF, dado que la cantidad de columnas antes y despues de hacer la operación sigue siendo la misma

Atipicos

Local Outlier Factor

```
clf = LOF(contamination=0.1)
clf.fit(train.drop('target',axis=1))
atipicos = clf.predict(train.drop('target',axis=1))   ## etiquetas binarias (0: no atipicos, 1: atipicos)

atipicos=atipicos!=1
train=train.iloc[atipicos,:]
train.shape
```

```
(1322, 28)
```

Alta Correlación

```
correlacion = train.drop('target',axis=1).corr().unstack()
correla_ordena= correlacion.sort_values(kind="quicksort")

```

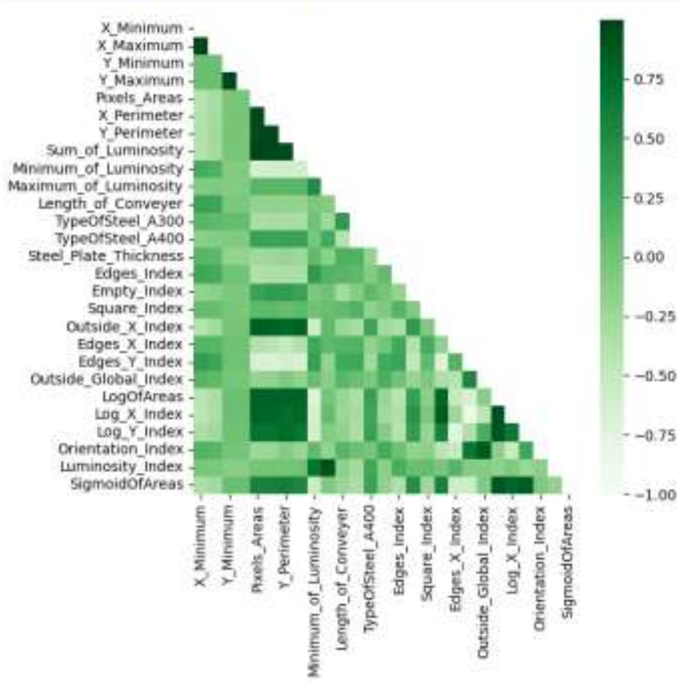
Alta Correlación

```
correlacion = train.drop('target',axis=1).corr().unstack()
correla_ordena= correlacion.sort_values(kind="quicksort")

correla_ordena=correla_ordena[(correla_ordena > 0.90) &( correla_ordena<0.999)]
correla_ordena
```

```
LogOfAreas              Log_X_Index              0.901216
Log_X_Index             LogOfAreas               0.901216
Maximum_of_Luminosity   Luminosity_Index         0.913880
Luminosity_Index        Maximum_of_Luminosity    0.913880
Y_Perimeter             Pixels_Areas             0.954729
Pixels_Areas            Y_Perimeter              0.954729
Y_Perimeter             Sum_of_Luminosity        0.958623
Sum_of_Luminosity       Y_Perimeter              0.958623
X_Perimeter             Pixels_Areas             0.974513
Pixels_Areas            X_Perimeter              0.974513
Sum_of_Luminosity       X_Perimeter              0.976020
X_Perimeter             Sum_of_Luminosity        0.976020
Y_Perimeter             X_Perimeter              0.977837
X_Perimeter             Y_Perimeter              0.977837
X_Maximum               X_Minimum                0.991427
X_Minimum               X_Maximum                0.991427
Sum_of_Luminosity       Pixels_Areas             0.997331
Pixels_Areas            Sum_of_Luminosity        0.997331
dtype: float64
```

```
grafica_correla(train.iloc[:,:-1])
```



```
[ ]  to_drop = refusion_dimension(train.drop('target',axis=1))
     to_drop
```

```
['X_Minimum', 'Y_Minimum', 'Pixels_Areas', 'X_Perimeter', 'Y_Perimeter']
```

```
[ ]  train = train.drop(to_drop, axis=1)
     train.shape
```

```
(1322, 23)
```

```
[ ]  X=train.drop('target', axis=1).values
     y=train['target'].values

     X.shape, y.shape
```

```
((1322, 22), (1322,))
```

Transformación de datos

Para poder trabajar con algumos modelos de ML se quiere hacer un encoding para las variables, normalización/estadarización segun corresponda.

```
[ ]  label_encoder = preprocessing.LabelEncoder()
     y=label_encoder.fit_transform(y)
     y
```

```
array([3, 0, 0, ..., 6, 3, 2])
```

```
[ ]  X=preprocessing.scale(X)
     X
```

```
array([[ 1.45284097, -0.90049171, -0.41018559, ...,  0.4166186 ,
        -0.46622624, -1.05573526],
       [-0.52164727, -0.6998491 , -0.39501208, ..., -0.92790963,
        -0.3318459 , -0.79539686],
       [ 0.90135292,  1.48458886, -0.39837167, ..., -1.23679615,
        -0.51319413, -0.57784136],
       ...,
       [-1.02281707, -0.61669185, -0.3917749 , ...,  0.68087434,
        -0.1120101 , -0.4358386 ],
       [-0.48944359, -0.34166807, -0.33594248, ...,  1.23450361,
         0.48552579,  0.77452253],
       [-0.90809145,  0.52517332,  1.67101813, ...,  0.14530497,
        -0.01089868,  1.39168837]])
```

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(
         X, y,test_size=0.2,random_state=SEED,stratify=y,shuffle=True)
```

Catboost

Se usará una estrategia para que Catboost sea la proporción que tienen las clases

```
classes = np.unique(y_train)
weights = compute_class_weight(class_weight='balanced', classes=classes, y=y_train)
class_weights = dict(zip(classes, weights))
```

```
train_dataset = Pool(X_train, y_train)

rscv = StratifiedKFold(n_splits=3, shuffle=True, random_state=SEED)

model = CatBoostClassifier(
                              learning_rate=0.01
                              ,depth=6
                              ,n_estimators=100
                              ,loss_function='MultiClass'
                              ,train_dir= 'crossentropy'
                              ,early_stopping_rounds=200
                              ,bagging_temperature = 1
                              ,metric_period = 100
                              ,langevin=True
                              ,random_seed =SEED
                              ,class_weights =class_weights

                             )

model.fit(train_dataset,  verbose=False, plot=True)
```

```
<catboost.core.CatBoostClassifier at 0x79932acf3880>
```

Datos de Entrenamiento

```
y_pred = model.predict(X_train)
print(classification_report(y_train,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.62      0.71      0.66       235
           1       0.39      1.00      0.56        28
           2       0.95      0.89      0.92       199
           3       0.77      0.37      0.50       362
           4       0.39      0.80      0.53        85
           5       0.76      0.91      0.83        43
           6       0.71      0.90      0.79       105

    accuracy                           0.67      1057
   macro avg       0.66      0.80      0.68      1057
weighted avg       0.72      0.67      0.66      1057
```

Datos de test

```
y_pred = model.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.55      0.71      0.62        59
           1       0.35      1.00      0.52         7
           2       1.00      0.82      0.90        50
           3       0.75      0.30      0.43        91
           4       0.36      0.62      0.46        21
           5       0.73      1.00      0.85        11
           6       0.60      0.92      0.73        26

    accuracy                           0.62       265
   macro avg       0.62      0.77      0.64       265
weighted avg       0.69      0.62      0.61       265
```

HyperTunning para Catboost

Balanceo de clases

Se usara una libreria para hacer el balanceo de clases, se probo una estrategia de decirle al algoritmo la proporción de las clases y se esperaba que este resultado fuera bueno. Dado que el supuesto es falso se procedio a hacer un balanceo de clases con estrategia más robustas.

```python
oversample = ADASYN(n_neighbors=6, random_state=SEED)
X_balance, y_balance = oversample.fit_resample(X_test, y_test)

clases_balance= pd.DataFrame.from_dict(Counter(y_balance), orient='index').reset_index()
clases_balance.columns=['defectos','porcentaje']
clases_balance['porcentaje']=round((clases_balance['porcentaje']/len(X_balance))*100,2)
clases_balance
```

	defectos	porcentaje
0	2	14.37
1	3	14.22
2	0	15.62
3	4	14.37
4	6	13.12
5	1	14.06
6	5	14.22

Cuando se entreno el modelo baseline usando CatBoost se uso una estrategia para decirle al algoritmo como estaban distribuidas las clases. Para hacer el hypertunning no vamos a usar está estrategia sino que por el contrario usaremos una estrategia de balanceo de clases por lo cual es conveniente que el modelo base no tenga está opción impuesta.

```python
model = CatBoostClassifier(#iterations=500,
                           learning_rate=0.01
                           ,depth=6
                           ,n_estimators=100
                           ,loss_function='MultiClass'
                           ,train_dir= 'crossentropy'
                           ,early_stopping_rounds=200
                           ,metric_period = 100
                           ,random_seed =SEED
                           #,task_type="GPU"
                           )

model.fit(train_dataset,  verbose=False, plot=True)

parameters = {'depth'         : [5,6,7,9, 10],
              'learning_rate' : [0.01,0.02,0.03,0.04],
              'n_estimators'    : [75,100,120]
               }

Grid_CBC_balance = GridSearchCV(estimator=model
                                ,param_grid = parameters
                                ,cv = rscv
                                ,scoring="accuracy"
                                ,n_jobs=-1)

Grid_CBC_balance.fit(X_balance, y_balance)
```

```
0:      learn: 1.8742504        total: 48.8ms   remaining: 5.8s
100:    learn: 0.4070332        total: 6.79s    remaining: 1.28s
119:    learn: 0.3480472        total: 7.59s    remaining: 0us
```

```
▸ GridSearchCV
▸ estimator: CatBoostClassifier
  ▸ CatBoostClassifier
```

```
from google.colab import output
output.enable_custom_widget_manager()
```

Support for third party widgets will remain active for the duration of the session. To disable support:

```
from google.colab import output
output.disable_custom_widget_manager()
```

Datos de Test

```
Grid_CBC_balance.best_params_,Grid_CBC_balance.best_estimator_
```

```
({'depth': 10, 'learning_rate': 0.04, 'n_estimators': 120},
 <catboost.core.CatBoostClassifier at 0x7f2af28cd650>)
```

```
y_pred = Grid_CBC_balance.best_estimator_.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        88
           1       0.85      1.00      0.92        11
           2       1.00      1.00      1.00        75
           3       1.00      0.94      0.97       136
           4       0.94      1.00      0.97        32
           5       1.00      1.00      1.00        16
           6       0.97      1.00      0.99        39

    accuracy                           0.98       397
   macro avg       0.96      0.99      0.97       397
weighted avg       0.98      0.98      0.98       397
```

Validación del modelo

Ahora que se tiene el modelo entrenado, se usarán el otro conjunto de datos suministrados para validarlo.

```
test=pd.read_csv(link_lectura+name_file_test, sep='\t',index_col=0)
print("La dimension del DataFrame de test es {}".format(test.shape))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-c5b05c631370> in <module>()
----> 1 test=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Tecnoap/Faults.test', sep='\t',index_col=0)
      2 print("La dimension del DataFrame de test es {}".format(test.shape))

NameError: name 'pd' is not defined
```

```
test
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-4e1243bd22c6> in <module>()
----> 1 test

NameError: name 'test' is not defined
```

Se coloca un procedimiento para garantizar que el nombre de las columnas en los dos archivos entregados por la empresa sean similares, excluyendo de DF de train la variable objetivo.

```
test= test[etiquetas]
test
```

```
XX_test=test.values
XX_test.shape
```

```
XX_test=preprocessing.scale(XX_test)
```

```
yy_pred = Grid_CBC_balance.best_estimator_.predict(XX_test)
defectos= label_encoder.inverse_transform(yy_pred)
defectos
```

```
test['target']=defectos
test
```

```
test.to_csv(link_lectura+name_user+"/"+name_file_exist)
```