```
# preprocesado y limpieza del dataset, balanceo de clases
# gerardo herrera
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files und

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 5GB to the current directory (/kaggle/working/) that gets preserved as
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the
```

```
/kaggle/input/pump-sensor-data/sensor.csv
```

```
sensor = pd.read_csv('../input/pump-sensor-data/sensor.csv')
```

```
print(sensor.shape)
```

```
(220320, 55)
```

```
sensor['machine_status'].value_counts()
```

```
NORMAL        205836
RECOVERING     14477
BROKEN             7
Name: machine_status, dtype: int64
```

```
#Show the number of missing (NAN, NaN, na) data for each column
sensor.isnull().sum()
```

```
Unnamed: 0            0
timestamp             0
sensor_00         10208
sensor_01           369
sensor_02            19
sensor_03            19
sensor_04            19
sensor_05            19
sensor_06          4798
```

```
sensor_07           5451
sensor_08           5107
sensor_09           4595
sensor_10             19
sensor_11             19
sensor_12             19
sensor_13             19
sensor_14             21
sensor_15         220320
sensor_16             31
sensor_17             46
sensor_18             46
sensor_19             16
sensor_20             16
sensor_21             16
sensor_22             41
sensor_23             16
sensor_24             16
sensor_25             36
sensor_26             20
sensor_27             16
sensor_28             16
sensor_29             72
sensor_30            261
sensor_31             16
sensor_32             68
sensor_33             16
sensor_34             16
sensor_35             16
sensor_36             16
sensor_37             16
sensor_38             27
sensor_39             27
sensor_40             27
sensor_41             27
sensor_42             27
sensor_43             27
sensor_44             27
sensor_45             27
sensor_46             27
sensor_47             27
sensor_48             27
sensor_49             27
sensor_50          77017
sensor_51          15383
machine_status         0
dtype: int64
```

```
# sensores con unidades conocidas, version dos de npantawee
#X=sensor[['sensor_00', 'sensor_01', 'sensor_02', 'sensor_03','sensor_04', 'sensor_11', 'sens
```

```
# borrar todos los demas sensores
```

```
sensor.drop(['sensor_05'], axis=1, inplace=True); sensor.drop(['sensor_06'], axis=1, inplace=
sensor.drop(['sensor_07'], axis=1, inplace=True); sensor.drop(['sensor_08'], axis=1, inplace=

sensor.drop(['sensor_09'], axis=1, inplace=True); sensor.drop(['sensor_10'], axis=1, inplace=

sensor.drop(['sensor_12'], axis=1, inplace=True); sensor.drop(['sensor_13'], axis=1, inplace=

# sensor sin data en en el dataset, segun esta rota la comunicacion
sensor.drop(['sensor_15'], axis=1, inplace=True);

sensor.drop(['sensor_24'], axis=1, inplace=True); sensor.drop(['sensor_29'], axis=1, inplace=

sensor.drop(['sensor_32'], axis=1, inplace=True); sensor.drop(['sensor_33'], axis=1, inplace=

sensor.drop(['sensor_34'], axis=1, inplace=True); sensor.drop(['sensor_35'], axis=1, inplace=

sensor.drop(['sensor_36'], axis=1, inplace=True); sensor.drop(['sensor_37'], axis=1, inplace=

sensor.drop(['sensor_38'], axis=1, inplace=True); sensor.drop(['sensor_39'], axis=1, inplace=

sensor.drop(['sensor_40'], axis=1, inplace=True); sensor.drop(['sensor_41'], axis=1, inplace=

sensor.drop(['sensor_42'], axis=1, inplace=True); sensor.drop(['sensor_43'], axis=1, inplace=

sensor.drop(['sensor_45'], axis=1, inplace=True); sensor.drop(['sensor_46'], axis=1, inplace=

sensor.drop(['sensor_47'], axis=1, inplace=True); sensor.drop(['sensor_48'], axis=1, inplace=

sensor.drop(['sensor_49'], axis=1, inplace=True)

# dataset con solo sensores con unidades conocidas
print(sensor.shape)
```

```
    (220320, 27)
```

```
# numero de instancias por clase
sensor['machine_status'].value_counts()
```

```
    NORMAL        205836
    RECOVERING     14477
    BROKEN            7
    Name: machine_status, dtype: int64
```

```
sensor.describe()
```

	Unnamed: 0	sensor_00	sensor_01	sensor_02	sensor_03	sen
count	220320.000000	210112.000000	219951.000000	220301.000000	220301.000000	220301.
mean	110159.500000	2.372221	47.591611	50.867392	43.752481	590.
std	63601.049991	0.412227	3.296666	3.666820	2.418887	144.
min	0.000000	0.000000	0.000000	33.159720	31.640620	2.
25%	55079.750000	2.438831	46.310760	50.390620	42.838539	626.
50%	110159.500000	2.456539	48.133678	51.649300	44.227428	632.
75%	165239.250000	2.499826	49.479160	52.777770	45.312500	637.
max	220319.000000	2.549016	56.727430	56.032990	48.220490	800.

8 rows × 25 columns

```
#Show the number of missing (NAN, NaN, na) data for each column
sensor.isnull().sum()
```

```
    Unnamed: 0            0
    timestamp             0
    sensor_00         10208
    sensor_01           369
    sensor_02            19
    sensor_03            19
    sensor_04            19
    sensor_11            19
    sensor_14            21
    sensor_16            31
    sensor_17            46
    sensor_18            46
    sensor_19            16
    sensor_20            16
    sensor_21            16
    sensor_22            41
    sensor_23            16
    sensor_25            36
    sensor_26            20
    sensor_27            16
    sensor_28            16
    sensor_30           261
    sensor_31            16
    sensor_44            27
    sensor_50         77017
    sensor_51         15383
    machine_status        0
    dtype: int64
```

```
index_names = sensor[ sensor['machine_status'] == "" ].index

# drop these row indexes
# from dataFrame
sensor.drop(index_names, inplace = True)
```

```
print(sensor.shape)
```

```
(220320, 27)
```

```
#Show the number of missing (NAN, NaN, na) data for each column
sensor.isnull().sum()
```

```
Unnamed: 0            0
timestamp             0
sensor_00         10208
sensor_01           369
sensor_02            19
sensor_03            19
sensor_04            19
sensor_11            19
sensor_14            21
sensor_16            31
sensor_17            46
sensor_18            46
sensor_19            16
sensor_20            16
sensor_21            16
sensor_22            41
sensor_23            16
sensor_25            36
sensor_26            20
sensor_27            16
sensor_28            16
sensor_30           261
sensor_31            16
sensor_44            27
sensor_50         77017
sensor_51         15383
machine_status        0
dtype: int64
```

```
# hacer un dataset balanceado de recovering=14000 y de normal=14000, descartar broken
```

```
sensor['machine_status'].value_counts()
```

```
NORMAL        205836
RECOVERING     14477
BROKEN             7
Name: machine_status, dtype: int64
```

```
# cuenta las instancias de la clase recovering

sensor[sensor.machine_status == "RECOVERING"].shape[0]
```

```
14477
```

```
sensor.sort_values(by='machine_status', ascending=False, na_position='first')
```

	Unnamed: 0	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	ser
17780	17780	2018-04-13 08:20:00	NaN	46.310760	49.001740	42.534721	3.104745	1
134357	134357	2018-07-03 07:17:00	NaN	32.725693	33.463539	32.378471	3.451967	0
134349	134349	2018-07-03 07:09:00	NaN	32.725693	33.463539	32.378471	3.567708	0
134350	134350	2018-07-03 07:10:00	NaN	32.725693	33.463539	32.378471	3.145254	0
134351	134351	2018-07-03 07:11:00	NaN	32.725693	33.463539	32.378471	3.451967	0
...	...	...	...	...	...	...	...	
17155	17155	2018-04-12 21:55:00	0.000000	53.342010	52.821180	43.402775	202.526031	3
69318	69318	2018-05-19 03:18:00	2.258796	47.265630	52.734370	43.446178	200.115738	43
128040	128040	2018-06-28 22:00:00	0.364005	40.190970	45.225690	40.190971	201.368622	1
166440	166440	2018-07-25 14:00:00	2.318808	45.833332	52.994790	43.880210	420.503448	50
24510	24510	2018-04-18 00:30:00	1.093982	42.534720	47.699650	41.449650	206.038757	30

220320 rows × 27 columns

```
sensor.head()
```

	Unnamed: 0	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_1
0	0	2018-04-01 00:00:00	2.465394	47.09201	53.2118	46.310760	634.3750	47.5242
1	1	2018-04-01 00:01:00	2.465394	47.09201	53.2118	46.310760	634.3750	47.5242
2	2	2018-04-01 00:02:00	2.444734	47.35243	53.2118	46.397570	638.8889	48.1772
3	3	2018-04-01 00:03:00	2.460474	47.09201	53.1684	46.397568	628.1250	48.6560
4	4	2018-04-01 00:04:00	2.445718	47.13541	53.2118	46.397568	636.4583	49.0629

5 rows × 27 columns

```
#index_names = sensor[ (sensor.machine_status == "RECOVERING") & ( sensor['machine_status'].v

#index_names = sensor[ (sensor.machine_status == "RECOVERING") & (sensor.value_counts()>20000

# drop these given row
# indexes from dataFrame
#sensor.drop(index_names, inplace = True)
```

```
sensor[sensor.machine_status == "RECOVERING"].shape[0]
```

```
14477
```

```
print(len(sensor))
```

```
220320
```

```
dios77 = sensor.sort_values(by=['machine_status'])
dios77.head(3)
#dios77.tail(3)
```

	Unnamed: 0	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	ser
24510	24510	2018-04-18 00:30:00	1.093982	42.53472	47.69965	41.449650	206.038757	30
128040	128040	2018-06-28 22:00:00	0.364005	40.19097	45.22569	40.190971	201.368622	1
69318	69318	2018-05-19 03:18:00	2.258796	47.26563	52.73437	43.446178	200.115738	43

```
dios77.tail(3)
```

	Unnamed: 0	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sen
25858	25858	2018-04-18 22:58:00	NaN	38.671880	39.019100	35.243050	3.608217	0.
25848	25848	2018-04-18 22:48:00	NaN	38.585070	38.932290	35.112846	3.608217	0.
131489	131489	2018-07-01 07:29:00	NaN	33.897568	36.979164	33.810764	3.029514	0.

3 rows × 27 columns

```
dios77['machine_status'].value_counts()
```

```
NORMAL        205836
RECOVERING     14477
BROKEN             7
Name: machine_status, dtype: int64
```

```
# se esta eliminado las instancias de la clase broken para poder balancear el dataset
dios77 = dios77[7:]
#dios77 = dios77.drop(index<=6)
```

```
dios77['machine_status'].value_counts()
```

```
NORMAL        205836
RECOVERING     14477
Name: machine_status, dtype: int64
```

```
# vamos a balancear las clases, quedarnos con la misma cantidad de cada una
# vamos a balancear las clases, quedarnos con la misma cantidad de cada una
```

```
# se separa en un dataset indepediente la clase NORMAL
normal = dios77[:205836]
normal['machine_status'].value_counts()
```

```
NORMAL    205836
Name: machine_status, dtype: int64
```

```
# se separa en un dataset indepediente la clase RECOVERING
recov = dios77[205836:]
recov['machine_status'].value_counts()
```

```
RECOVERING    14477
Name: machine_status, dtype: int64
```

```
# se vuelve a unir todas las clases para verificar que no se pierde ninguna instancia
#vertical_stack = pd.concat([survey_sub, survey_sub_last10], axis=0)
vertical_stack = pd.concat([normal, recov], axis=0)
vertical_stack['machine_status'].value_counts()
```

```
NORMAL        205836
RECOVERING     14477
Name: machine_status, dtype: int64
```

```
from random import randrange
#np.random.seed(randrange(100))

#remove_n = 190798

#drop_indices = np.random.choice(normal.index, remove_n, replace=False)
#normal = normal.drop(drop_indices)
```

```
# se elimina instancias del dataset con formato invalido en varias columnas, de la clase RECO
recov=recov.dropna(subset=['sensor_01', 'sensor_02','sensor_03','sensor_04','sensor_11','sens
```

```
recov['machine_status'].value_counts()
```

```
RECOVERING    14372
Name: machine_status, dtype: int64
```

```
normal.isnull().sum()
```

```
Unnamed: 0          0
timestamp           0
sensor_00          14
sensor_01         339
```

```
sensor_02            14
sensor_03            14
sensor_04            14
sensor_11            14
sensor_14            21
sensor_16            31
sensor_17            46
sensor_18            46
sensor_19            16
sensor_20            16
sensor_21            16
sensor_22            41
sensor_23            16
sensor_25            36
sensor_26            20
sensor_27            16
sensor_28            16
sensor_30           261
sensor_31            16
sensor_44            22
sensor_50         76936
sensor_51         12384
machine_status        0
dtype: int64
```

```
# se elimina instancias del dataset con formato invalido en varias columnas, de la clase NORM
normal=normal.dropna(subset=['sensor_01','sensor_30','sensor_22','sensor_51'])
normal['machine_status'].value_counts()
```

```
NORMAL    192877
Name: machine_status, dtype: int64
```

```
count_row = normal.shape[0]
#normal.count
#print (normal.count)
print (count_row)
```

```
192877
```

```
# eliminamos instancias al azar hasta tener 14000 de la clase NORMAL
from random import randrange
import math
np.random.seed(randrange(100))

#remove_n = 477
remove_n = int(math.fabs(count_row - 14000))
#df = pd.DataFrame({"a":[1,2,3,4], "b":[5,6,7,8]})
drop_indices = np.random.choice(normal.index, remove_n, replace=False)
normal = normal.drop(drop_indices)
normal['machine_status'].value_counts()
```

```
     NORMAL    14000
     Name: machine_status, dtype: int64
```

```
# eliminamos instancias al azar hasta tener 14000 de la clase RECOVERING
from random import randrange
np.random.seed(randrange(100))

remove_n = 372
#df = pd.DataFrame({"a":[1,2,3,4], "b":[5,6,7,8]})
drop_indices = np.random.choice(recov.index, remove_n, replace=False)
recov = recov.drop(drop_indices)
recov['machine_status'].value_counts()
```

```
     RECOVERING    14000
     Name: machine_status, dtype: int64
```

```
# se unen las clases en un dataset ahora balanceado
juntos_stack = pd.concat([normal, recov], axis=0)
juntos_stack['machine_status'].value_counts()
```

```
     RECOVERING    14000
     NORMAL        14000
     Name: machine_status, dtype: int64
```

```
dios77= juntos_stack
```

```
dios77['machine_status'].value_counts()
```

```
     RECOVERING    14000
     NORMAL        14000
     Name: machine_status, dtype: int64
```

```
# se exporta a csv para luego usar en el entrenamiento de los modelos
dios77.to_csv('dios77-ordenado_status_sin_broken_balanced_28k_antes.csv',index=False)
```