Librerias y funciones

In [1]:
```
# Librerias
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import plotly.express as px

import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
```

In [2]:
```
def plot_machine_failures(machine):
    """
    Función de visualización de las variables y los fallos que ha tenido la máquina
    """
    max_volt = machine['volt'].max()
    min_volt = machine['volt'].min()

    max_rotate = machine['rotate'].max()
    min_rotate = machine['rotate'].min()

    max_pressure = machine['pressure'].max()
    min_pressure = machine['pressure'].min()

    max_vibration = machine['vibration'].max()
    min_vibration = machine['vibration'].min()



    fig, ax = plt.subplots(4, figsize=(20, 15))
    ax[0].plot(machine['datetime'], machine['volt'], linewidth=0.3)
    ax[0].set_ylabel('volt')

    ax[1].plot(machine['datetime'], machine['rotate'], linewidth=0.3)
    ax[1].set_ylabel('rotate')

    ax[2].plot(machine['datetime'], machine['pressure'], linewidth=0.3)
    ax[2].set_ylabel('pressure')

    ax[3].plot(machine['datetime'], machine['vibration'], linewidth=0.3)
    ax[3].set_ylabel('vibration')

    for index, row in machine.iterrows():
        if row['comp1'] == 1:
            ax[0].vlines(x=row['datetime'], ymin=min_volt, ymax= max_volt, linestyles='dashed', color='red')
            ax[1].vlines(x=row['datetime'], ymin=min_rotate, ymax= max_rotate, linestyles='dashed', color='red'
            ax[2].vlines(x=row['datetime'], ymin=min_pressure, ymax= max_pressure, linestyles='dashed', color='
            ax[3].vlines(x=row['datetime'], ymin=min_vibration, ymax= max_vibration, linestyles='dashed', color
        if row['comp2'] == 1:
            ax[0].vlines(x=row['datetime'], ymin=min_volt, ymax= max_volt, linestyles='dashed', color='orange')
            ax[1].vlines(x=row['datetime'], ymin=min_rotate, ymax= max_rotate, linestyles='dashed', color='oran
            ax[2].vlines(x=row['datetime'], ymin=min_pressure, ymax= max_pressure, linestyles='dashed', color='
            ax[3].vlines(x=row['datetime'], ymin=min_vibration, ymax= max_vibration, linestyles='dashed', color
        if row['comp3'] == 1:
            ax[0].vlines(x=row['datetime'], ymin=min_volt, ymax= max_volt, linestyles='dashed', color='black')
            ax[1].vlines(x=row['datetime'], ymin=min_rotate, ymax= max_rotate, linestyles='dashed', color='blac
            ax[2].vlines(x=row['datetime'], ymin=min_pressure, ymax= max_pressure, linestyles='dashed', color='
            ax[3].vlines(x=row['datetime'], ymin=min_vibration, ymax= max_vibration, linestyles='dashed', color
        if row['comp4'] == 1:
            ax[0].vlines(x=row['datetime'], ymin=min_volt, ymax= max_volt, linestyles='dashed', color='gray')
            ax[1].vlines(x=row['datetime'], ymin=min_rotate, ymax= max_rotate, linestyles='dashed', color='gray
            ax[2].vlines(x=row['datetime'], ymin=min_pressure, ymax= max_pressure, linestyles='dashed', color='
            ax[3].vlines(x=row['datetime'], ymin=min_vibration, ymax= max_vibration, linestyles='dashed', color

    plt.show()

def calculate_RUL_cycle(df, column='failure', pre1=3, pre2=24):
    """
    Función para calcular el RUL de los ciclos hasta el fallo de la máquina.

    Se obtiene una lista de dataframes con de cada ciclo de trabajo (hasta el fallo)
    """
    df_fail = df[df[column] == 1]
    temp = []
    initial = 0
    df_rul = df.copy()
    df_rul = df_rul.assign(volt_3h_mean=0, rotate_3h_mean=0, pressure_3h_mean=0, vibration_3h_mean=0,
                           volt_24h_mean=0, rotate_24h_mean=0, pressure_24h_mean=0, vibration_24h_mean=0,
                           error1_count=0, error2_count=0, error3_count=0, error4_count=0, error5_count=0, RUL=
    for row_fail in df_fail.index:
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('volt_3h_mean')] = df_rul.iloc[initial:row_fail+
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('rotate_3h_mean')] = df_rul.iloc[initial:row_fai
```

```
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('pressure_3h_mean')] = df_rul.iloc[initial:row_f
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('vibration_3h_mean')] = df_rul.iloc[initial:row_

        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('volt_24h_mean')] = df_rul.iloc[initial:row_fail
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('rotate_24h_mean')] = df_rul.iloc[initial:row_fa
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('pressure_24h_mean')] = df_rul.iloc[initial:row_
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('vibration_24h_mean')] = df_rul.iloc[initial:row

        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('error1_count')] = df_rul.iloc[initial:row_fail+
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('error2_count')] = df_rul.iloc[initial:row_fail+
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('error3_count')] = df_rul.iloc[initial:row_fail+
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('error4_count')] = df_rul.iloc[initial:row_fail+
        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('error5_count')] = df_rul.iloc[initial:row_fail+

        df_rul.iloc[initial:row_fail+1, df_rul.columns.get_loc('RUL')] = pd.Series(range(row_fail, -1, -1))

        temp.append(df_rul.iloc[initial:row_fail+1])
        initial = row_fail + 1
    df_rul = df_rul.drop(df.index[row_fail+1:len(df_rul)])
    return temp

def cut_cycle(cycle_lst, min_cycle=0, threshold = 480, flat = False):
    """
    Función para que permite "aplanar" la lista obtenida de calculate_RUL_cycle
    Tambíen permite normalizar los ciclos de trabajo según el número minimo de muestras
    que este por encima del threshold. Este normalizado se aplica a todos los ciclos de la lista

    Se obteniene una lista solamente con los ciclos que cumplan con el mínimo de muestras
    También se obtniene el mínimo de muestras por ciclo encontrado
    """
    machine_m3_rul_cycle_norm = []

    if flat == False:
        cycle_lst = [item for items in cycle_lst for item in items]

    if min_cycle == 0:
        min_cycle = min(len(df) for df in cycle_lst if len(df) > threshold)

    for df in cycle_lst:
        if len(df) >= min_cycle:
            machine_m3_rul_cycle_norm.append(df.tail(min_cycle))
    return machine_m3_rul_cycle_norm, min_cycle
```

Carga de datos

In [3]:
```
# DATA_DIR = "C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/Data 2"
DATA_DIR = "C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/Data 2"


dataset = pd.read_csv(f"{DATA_DIR}/Dataset.csv")
dataset['datetime'] = pd.to_datetime(dataset['datetime'])
dataset.head()
```

Out[3]:

	datetime	machineID	volt	rotate	pressure	vibration	error1	error2	error3	error4	error5	comp1	comp2	comp3	comp4
0	2015-01-01 06:00:00	1	176.217853	418.504078	113.077935	45.087686	0	0	0	0	0	0	0	0	0
1	2015-01-01 06:00:00	2	176.558913	424.624162	76.005332	43.767049	0	0	0	0	0	0	0	0	0
2	2015-01-01 06:00:00	3	185.482043	461.211137	87.453199	28.216864	0	0	0	0	0	0	0	0	0
3	2015-01-01 06:00:00	4	169.710847	463.646727	95.929877	38.400372	0	0	0	0	0	0	0	0	0
4	2015-01-01 06:00:00	5	165.082899	452.283576	84.591722	40.298803	0	0	0	0	0	0	0	0	0

In [4]:
```
# Lectura de datos
DATA_DIR = "C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/Data"

telemetry_df = pd.read_csv(f"{DATA_DIR}/PdM_telemetry.csv")
errors_df = pd.read_csv(f"{DATA_DIR}/PdM_errors.csv")
maint_df = pd.read_csv(f"{DATA_DIR}/PdM_maint.csv")
failures_df = pd.read_csv(f"{DATA_DIR}/PdM_failures.csv")
machines_df = pd.read_csv(f"{DATA_DIR}/PdM_machines.csv")
```

Separación de datasets por modelo e ID

Separación de datasets por modelo e ID

Se ha seleccionado las màquinas del mismo modelo y con la misma edad para considerar-los como un único conjunto

In [19]:
```
machine_m3_15 = dataset[(dataset['model']=='model3') & (dataset['machineID']==15)].copy().reset_index().drop('i
machine_m3_33 = dataset[(dataset['model']=='model3') & (dataset['machineID']==33)].copy().reset_index().drop('i
machine_m3_43 = dataset[(dataset['model']=='model3') & (dataset['machineID']==43)].copy().reset_index().drop('i
machine_m3_45 = dataset[(dataset['model']=='model3') & (dataset['machineID']==45)].copy().reset_index().drop('i
machine_m3_52 = dataset[(dataset['model']=='model3') & (dataset['machineID']==52)].copy().reset_index().drop('i
machine_m3_79 = dataset[(dataset['model']=='model3') & (dataset['machineID']==79)].copy().reset_index().drop('i

machine_m3 = [machine_m3_15, machine_m3_33, machine_m3_43, machine_m3_45, machine_m3_52]
```

In [20]:
```
machine_m3_79.head()
```

Out[20]:

	datetime	machineID	volt	rotate	pressure	vibration	error1	error2	error3	error4	error5	comp1	comp2	comp3	comp4
0	2015-01-01 06:00:00	79	194.167651	535.302327	84.071434	35.654640	0	0	0	0	0	0	0	0	0
1	2015-01-01 07:00:00	79	180.989438	429.031142	96.245928	37.163490	0	0	0	0	0	0	0	0	0
2	2015-01-01 08:00:00	79	208.452257	374.095525	106.340279	37.617635	0	0	0	0	0	0	0	0	0
3	2015-01-01 09:00:00	79	192.161379	438.784836	94.309427	47.383279	0	0	0	0	0	0	0	0	0
4	2015-01-01 10:00:00	79	207.822427	391.115326	103.079753	39.579032	0	0	0	0	0	0	0	0	0

In [11]:
```
machine_m3_79.columns
```

Out[11]:
```
Index(['datetime', 'machineID', 'volt', 'rotate', 'pressure', 'vibration',
       'error1', 'error2', 'error3', 'error4', 'error5', 'comp1', 'comp2',
       'comp3', 'comp4', 'failure', 'model', 'age'],
      dtype='object')
```

Correlaciones

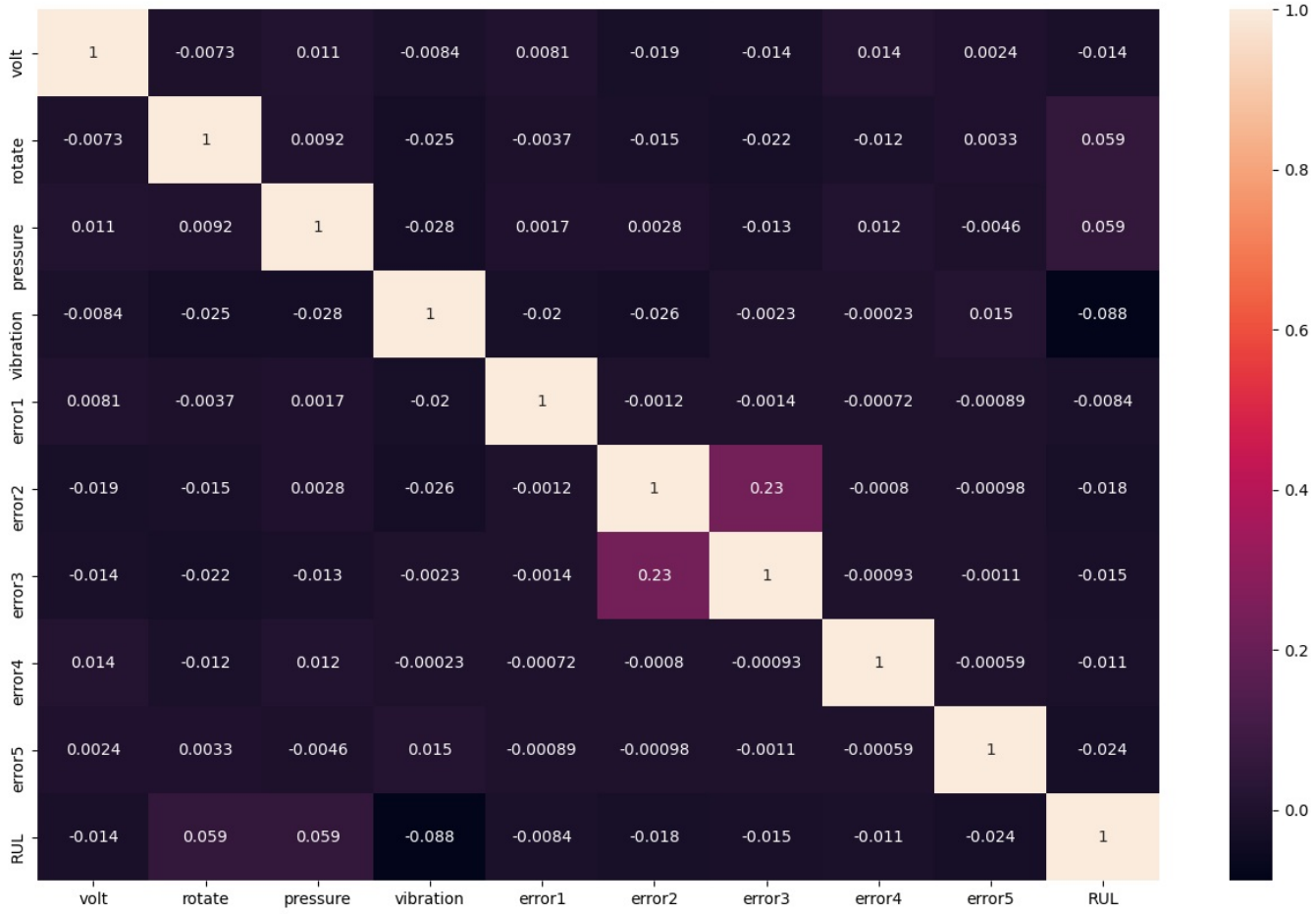
In [25]:
```
# Generamos matriz lista  de dataframes con los ciclos y rul
machine_m3_rul_cycle = []
for machine in machine_m3:
    machine_m3_rul_cycle.append(calculate_RUL_cycle(machine))

# Cortamos los filtros por el mínimo de muestras que queremos sobre la lista de ciclos
min_cycle = 0
muestras = 500

machine_m3_rul_cycle_cut, min_cycle = cut_cycle(machine_m3_rul_cycle, min_cycle, muestras)

# Realizamos lo mismo para la otra
machine_m3_79_rul_cycle = calculate_RUL_cycle(machine_m3_79)
machine_m3_79_rul_cycle_cut, _ = cut_cycle(machine_m3_79_rul_cycle, min_cycle, 500, True)

# Concatenamos todos los dataframes de las listas para generar un dataframe único
machine_m3_rul_cycle_cut = pd.concat(machine_m3_rul_cycle_cut)
machine_m3_79_rul_cycle_cut = pd.concat(machine_m3_79_rul_cycle_cut)

temp_l = [pd.concat(machine_m3_rul_cycle[0]), pd.concat(machine_m3_rul_cycle[1]), pd.concat(machine_m3_rul_cycl
machine_m3_rul_cycle_uncut = pd.concat(temp_l)
machine_m3_79_rul_cycle_uncut = pd.concat(machine_m3_79_rul_cycle)

machine_m3_79_rul_cycle_uncut = machine_m3_79_rul_cycle_uncut[['datetime', 'volt', 'rotate',
                                                               'pressure', 'vibration','error1', 'error2',
                                                               'error3', 'error4', 'error5', 'model', 'RUL']]
machine_m3_79_rul_cycle_uncut[:22]
```

Out[25]:

	datetime	volt	rotate	pressure	vibration	error1	error2	error3	error4	error5	model	RUL
0	2015-01-01 06:00:00	194.167651	535.302327	84.071434	35.654640	0	0	0	0	0	model3	21
1	2015-01-01 07:00:00	180.989438	429.031142	96.245928	37.163490	0	0	0	0	0	model3	20
2	2015-01-01 08:00:00	208.452257	374.095525	106.340279	37.617635	0	0	0	0	0	model3	19
3	2015-01-01 09:00:00	192.161379	438.784836	94.309427	47.383279	0	0	0	0	0	model3	18
4	2015-01-01 10:00:00	207.822427	391.115326	103.079753	39.579032	0	0	0	0	0	model3	17
5	2015-01-01 11:00:00	193.647095	484.526874	95.397495	32.340949	0	0	0	0	0	model3	16
6	2015-01-01 12:00:00	201.603444	318.811141	110.102601	38.413517	0	0	0	0	0	model3	15
7	2015-01-01 13:00:00	213.373747	489.746066	105.192482	31.554710	0	0	0	0	0	model3	14
8	2015-01-01 14:00:00	184.497121	421.289094	94.277212	37.787211	0	0	0	0	0	model3	13
9	2015-01-01 15:00:00	206.497032	486.085091	84.973548	37.514714	0	0	0	0	0	model3	12
10	2015-01-01 16:00:00	206.557544	476.343880	103.187476	38.246337	0	0	0	0	0	model3	11
11	2015-01-01 17:00:00	184.162078	473.900598	82.287550	32.353643	0	0	0	0	0	model3	10
12	2015-01-01 18:00:00	220.870512	388.498630	101.861546	41.342239	0	0	0	0	0	model3	9
13	2015-01-01 19:00:00	210.927495	465.994665	85.270919	37.968711	0	0	0	0	0	model3	8
14	2015-01-01 20:00:00	177.076815	435.148457	101.058159	38.907672	0	0	0	0	0	model3	7
15	2015-01-01 21:00:00	148.141444	410.707959	90.994411	40.245877	0	0	0	0	0	model3	6
16	2015-01-01 22:00:00	174.140050	506.239512	120.878279	56.343024	0	0	0	0	0	model3	5
17	2015-01-01 23:00:00	173.807056	459.251784	99.110984	35.496824	0	0	0	0	0	model3	4
18	2015-01-02 00:00:00	201.189930	453.431673	96.072694	40.861517	0	0	0	0	0	model3	3
19	2015-01-02 01:00:00	203.072918	505.047943	101.052282	47.169353	0	0	0	0	0	model3	2
20	2015-01-02 02:00:00	206.128690	456.912575	97.775535	34.639112	0	0	0	0	0	model3	1
21	2015-01-02 03:00:00	206.934554	423.353299	110.745533	35.708642	0	0	0	0	0	model3	0

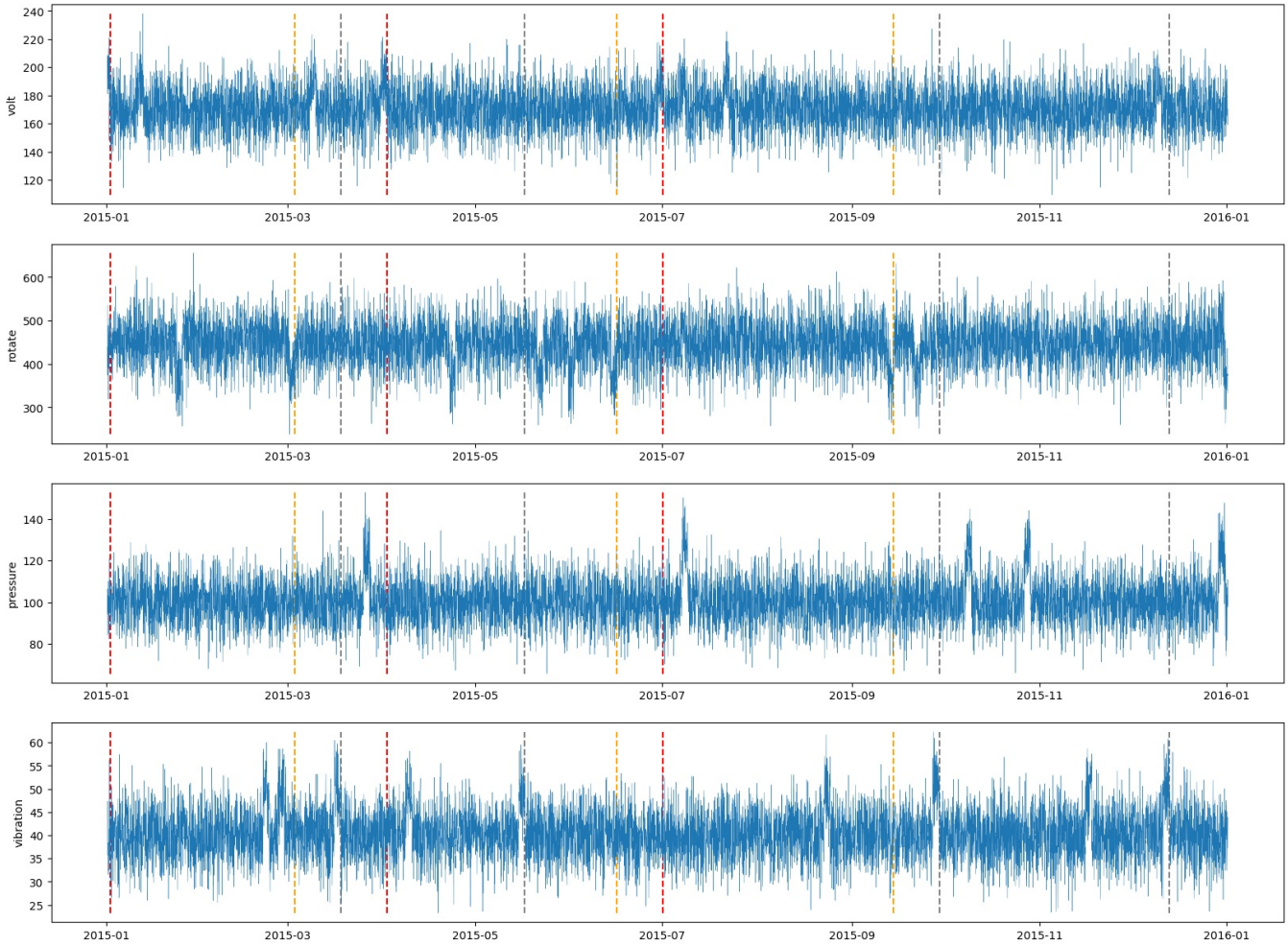
In [26]:
```
#All variables correlations
plt.figure(figsize=(16, 10))
sns.heatmap(machine_m3_79_rul_cycle_uncut.corr(), annot=True)
```

Out[26]:
```
<AxesSubplot:>
```



Variables de las máquinas junto a los fallos de componentes

In [7]:
```
plot_machine_failures(machine_m3_79)
```



Generación de sub-Datasets con RUL y ciclos normalizados

En este sección añadimos cortamos los datos por ciclos de trabajo, es decir, desde su correcto funcionamiento hasta su próximo fallo. Una vez obtenido estos cyclos de trabajo añadiremos la columna RUL y State.

La columna RUL sera un valor decreciente por hora de màquina trabajado, eso significa que el primer valor del RUL de la máquina sera el máximo y este ira decreciendo a medida que pasen las horas, hasta llegar a 0 que es donde termina el ciclo con un fallo.

La columna State serà un valor categorico que permitirá saber si la máquina dejará de funcionar en las próximas horas, para hello tenemos el estado "Normal" = 0 i el estado "Warning" = 1. En este caso se considerará estado "Warning" si la máquina tendra un fallo en las próximas 72h, es decir 3 dias.

In [8]:
```
# Generamos matriz lista  de dataframes con los ciclos y rul
machine_m3_rul_cycle = []
for machine in machine_m3:
    machine_m3_rul_cycle.append(calculate_RUL_cycle(machine))

# Cortamos los filtros por el mínimo de muestras que queremos sobre la lista de ciclos
min_cycle = 0
muestras = 500

machine_m3_rul_cycle_cut, min_cycle = cut_cycle(machine_m3_rul_cycle, min_cycle, muestras)

# Realizamos lo mismo para la otra
machine_m3_79_rul_cycle = calculate_RUL_cycle(machine_m3_79)
machine_m3_79_rul_cycle_cut, _ = cut_cycle(machine_m3_79_rul_cycle, min_cycle, 500, True)
```

In [9]:
```
# Concatenamos todos los dataframes de las listas para generar un dataframe único
machine_m3_rul_cycle_cut = pd.concat(machine_m3_rul_cycle_cut)
machine_m3_79_rul_cycle_cut = pd.concat(machine_m3_79_rul_cycle_cut)

temp_l = [pd.concat(machine_m3_rul_cycle[0]), pd.concat(machine_m3_rul_cycle[1]), pd.concat(machine_m3_rul_cycl
machine_m3_rul_cycle_uncut = pd.concat(temp_l)
machine_m3_79_rul_cycle_uncut = pd.concat(machine_m3_79_rul_cycle)
```

In [10]:
```
machine_m3_79_rul_cycle_uncut
```

Out[10]:

	datetime	machineID	volt	rotate	pressure	vibration	error1	error2	error3	error4	...	volt_24h_mean	rotate_24h_mean
0	2015-01-01 06:00:00	79	194.167651	535.302327	84.071434	35.654640	0	0	0	0	...	194.167651	535.302327
1	2015-01-01 07:00:00	79	180.989438	429.031142	96.245928	37.163490	0	0	0	0	...	187.578545	482.166735
2	2015-01-01 08:00:00	79	208.452257	374.095525	106.340279	37.617635	0	0	0	0	...	194.536449	446.142998
3	2015-01-01 09:00:00	79	192.161379	438.784836	94.309427	47.383279	0	0	0	0	...	193.942681	444.303458
4	2015-01-01 10:00:00	79	207.822427	391.115326	103.079753	39.579032	0	0	0	0	...	196.718630	433.665831
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8300	2015-12-13 02:00:00	79	186.760444	433.024336	95.642356	57.879780	0	0	0	0	...	170.637997	451.481347
8301	2015-12-13 03:00:00	79	168.397729	394.397016	100.890146	55.963847	0	0	0	0	...	170.454144	448.149948
8302	2015-12-13 04:00:00	79	175.738158	449.913107	115.431667	43.352411	0	0	0	0	...	170.059111	447.504322
8303	2015-12-13 05:00:00	79	193.264043	489.664303	87.368971	47.780141	0	0	0	0	...	171.020446	448.102066
8304	2015-12-13 06:00:00	79	166.747733	530.176778	80.825280	45.902295	0	0	0	0	...	170.384790	453.838179

8305 rows × 32 columns

In [11]:
```
# Añadimos la columna State
machine_m3_rul_cycle_cut['State'] = np.where(machine_m3_rul_cycle_cut['RUL'] <= 72, 1, 0)
machine_m3_79_rul_cycle_cut['State'] = np.where(machine_m3_79_rul_cycle_cut['RUL'] <= 72, 1, 0)

# Añadimos la columna State
machine_m3_rul_cycle_uncut['State'] = np.where(machine_m3_rul_cycle_uncut['RUL'] <= 72, 1, 0)
machine_m3_79_rul_cycle_uncut['State'] = np.where(machine_m3_79_rul_cycle_uncut['RUL'] <= 72, 1, 0)
```

Datasets generados

machine_m3_rul_cycle_cut

Dataset de las máquinas de modelo 3, con su rul calculado y seleccionado los ciclos normalizados

In [12]:
```
machine_m3_rul_cycle_cut
```

Out[12]:

	datetime	machineID	volt	rotate	pressure	vibration	error1	error2	error3	error4	...	rotate_24h_mean	pressure_24h_
2977	2015-05-05 07:00:00	15	158.809093	519.491804	89.943126	45.862285	0	0	0	0	...	463.744623	100.0
2978	2015-05-05 08:00:00	15	133.369103	500.095396	90.904882	30.850557	0	0	0	0	...	464.839572	99.5
2979	2015-05-05 09:00:00	15	160.381456	439.088244	99.668807	37.620321	0	0	0	0	...	461.226552	99.5
2980	2015-05-05 10:00:00	15	172.735191	550.954690	112.886843	45.702312	0	0	0	0	...	466.948133	99.8
2981	2015-05-05 11:00:00	15	171.673821	498.732058	95.322091	39.311040	0	0	0	0	...	467.525822	99.1
...	...	...	...	...	...	...	...	...	...	...	...	...	
8708	2015-12-30 02:00:00	52	166.161254	515.775881	94.914844	48.511206	0	0	0	0	...	453.138098	96.8
8709	2015-12-30 03:00:00	52	179.275310	418.558892	81.251350	54.070050	0	0	0	0	...	454.023081	95.8
8710	2015-12-30 04:00:00	52	198.275985	477.652781	100.145070	42.757978	0	0	0	0	...	455.883615	95.4
8711	2015-12-30 05:00:00	52	182.156321	534.983527	82.099477	45.139395	0	0	0	0	...	459.872017	94.7
8712	2015-12-30 06:00:00	52	157.309819	362.760911	100.335067	56.812017	0	0	0	0	...	457.491174	93.9

17280 rows × 33 columns

machine_m3_79_rul_cycle_cut

Dataset de las máquinas de modelo 3 e ID 79, con su rul calculado y seleccionado los ciclos normalizados

In [13]:
```
machine_m3_79_rul_cycle_cut
```

Out[13]:

	datetime	machineID	volt	rotate	pressure	vibration	error1	error2	error3	error4	...	rotate_24h_mean	pressure_24h_
745	2015-02-01 07:00:00	79	171.585268	480.714388	123.036718	36.549834	0	0	0	0	...	461.654747	101.4
746	2015-02-01 08:00:00	79	180.671569	401.104200	98.973284	35.288739	0	0	0	0	...	454.048837	101.5
747	2015-02-01 09:00:00	79	182.152843	462.262032	89.172008	43.614592	0	0	0	0	...	450.610435	101.5
748	2015-02-01 10:00:00	79	187.582362	421.302176	107.973283	42.696285	0	0	0	0	...	449.933323	101.6
749	2015-02-01 11:00:00	79	183.456793	354.812587	113.297576	38.556803	0	0	0	0	...	443.197615	101.9
...	...	...	...	...	...	...	...	...	...	...	...	...	
8300	2015-12-13 02:00:00	79	186.760444	433.024336	95.642356	57.879780	0	0	0	0	...	451.481347	96.9
8301	2015-12-13 03:00:00	79	168.397729	394.397016	100.890146	55.963847	0	0	0	0	...	448.149948	96.3
8302	2015-12-13 04:00:00	79	175.738158	449.913107	115.431667	43.352411	0	0	0	0	...	447.504322	97.0
8303	2015-12-13 05:00:00	79	193.264043	489.664303	87.368971	47.780141	0	0	0	0	...	448.102066	96.3
8304	2015-12-13 06:00:00	79	166.747733	530.176778	80.825280	45.902295	0	0	0	0	...	453.838179	95.4

3600 rows × 33 columns

Dataset con ciclos cortados

In [14]:
```
machine_m3_rul_train = machine_m3_rul_cycle_cut.drop(['datetime', 'machineID', 'model', 'age', 'comp1', 'comp2'
machine_m3_79_rul = machine_m3_79_rul_cycle_cut.drop(['datetime', 'machineID', 'model', 'age', 'comp1', 'comp2'

machine_m3_rul_train.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/train_data.csv', e
machine_m3_79_rul.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/test_data.csv', encod
```

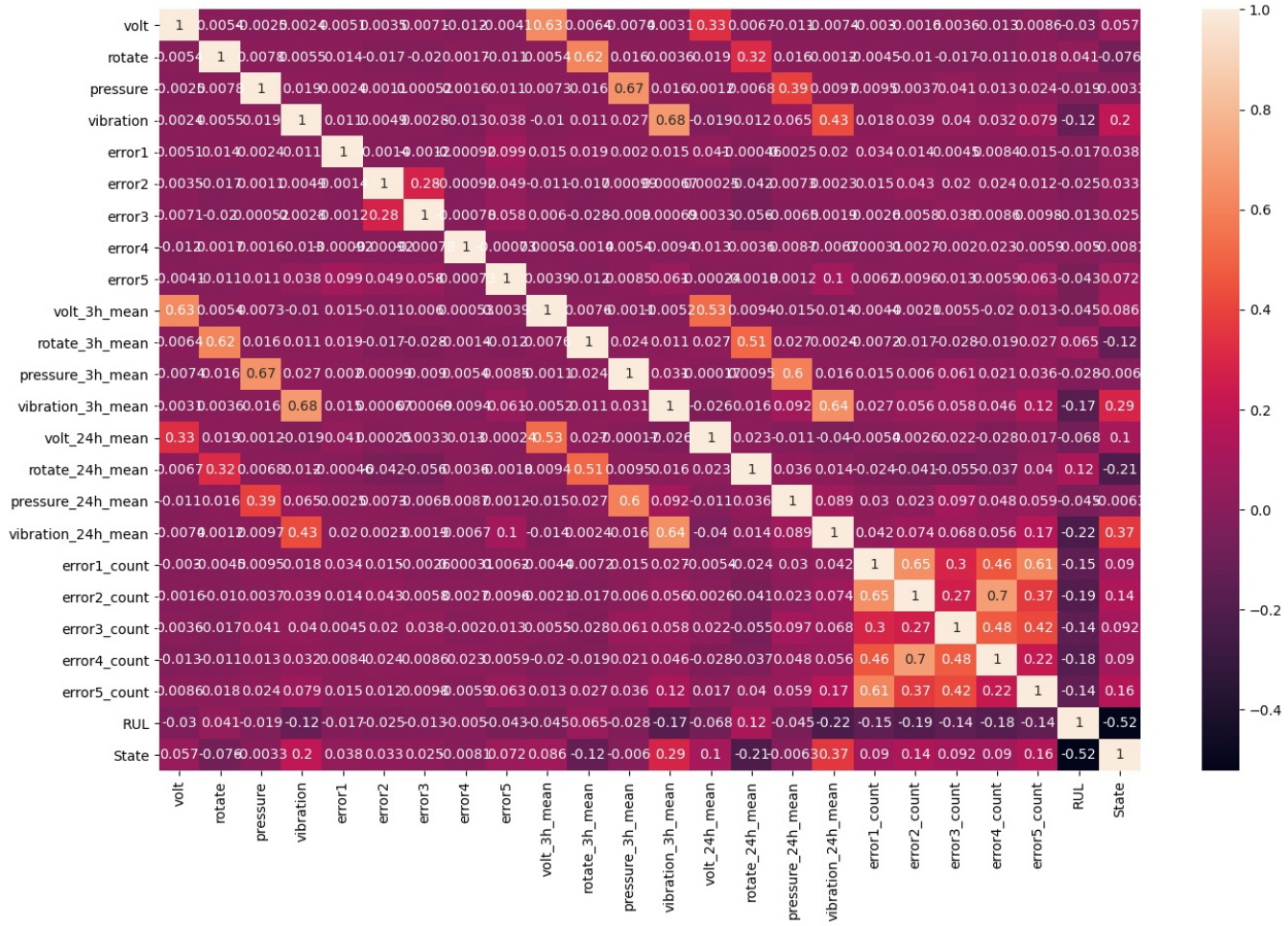
Dataset con ciclos sin cortar

In [15]:
```
machine_m3_rul_train_uncut = machine_m3_rul_cycle_uncut.drop(['datetime', 'machineID', 'model', 'age', 'comp1',
machine_m3_79_rul_uncut = machine_m3_79_rul_cycle_uncut.drop(['datetime', 'machineID', 'model', 'age', 'comp1',

machine_m3_rul_train_uncut.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/train_data_u
machine_m3_79_rul_uncut.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/test_data_uncut
```

Correlaciones

In [16]:
```
#All variables correlations
plt.figure(figsize=(16, 10))
sns.heatmap(machine_m3_rul_train.corr(), annot=True)
```

Out[16]:
```
<AxesSubplot:>
```



Train/Test Datasets

In [17]:
```
machine_m3_rul_train = machine_m3_rul_cycle_cut.drop(['datetime', 'machineID', 'model', 'age', 'comp1', 'comp2'
machine_m3_79_rul = machine_m3_79_rul_cycle_cut.drop(['datetime', 'machineID', 'model', 'age', 'comp1', 'comp2'
```

In [18]:
```
X_train = machine_m3_rul_train.loc[:, machine_m3_rul_train.columns != 'RUL']
y_train = machine_m3_rul_train.loc[:, machine_m3_rul_train.columns == 'RUL']
X_test = machine_m3_79_rul.loc[:, machine_m3_79_rul.columns != 'RUL']
y_test = machine_m3_79_rul.loc[:, machine_m3_79_rul.columns == 'RUL']

print(f"X_train: {len(X_train)}")
print(f"y_train: {len(y_train)}")
print(f"X_test: {len(X_test)}")
print(f"y_test: {len(y_test)}")
```

```
X_train: 17280
y_train: 17280
X_test: 3600
y_test: 3600
```

In [19]:
```
X_train.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/X_train.csv', encoding='utf-8',
y_train.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/y_train.csv', encoding='utf-8',
X_test.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/X_test.csv', encoding='utf-8', i
y_test.to_csv('C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets/y_test.csv', encoding='utf-8', i
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js