In [31]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import clustermap
from sklearn.preprocessing import MinMaxScaler
%matplotlib notebook
```

In [32]:

```
cmm = pd.read_excel('CMMData.xlsx')
cmm
```

Out[32]:

| | Unnamed: 0 | Unnamed: 1 | Unnamed: 2 | Unnamed: 3 | ID @100 | ID @55 | 38 dia @200 | 42 dia @140 | 42 dia @80 | Base angle F | ... | 162mm taper F | 1( |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | NaN | NaN | 30.000 | 30.000 | 38.000 | 42.000 | 42.000 | 70.000 | ... | 162.000 | 16 |
| **1** | NaN | NaN | NaN | NaN | 0.300 | 0.300 | 0.400 | 0.400 | 0.400 | 1.000 | ... | 1.000 | |
| **2** | NaN | NaN | NaN | Part ID | -0.300 | -0.300 | -0.400 | -0.400 | -0.400 | -1.000 | ... | -1.000 | - |
| **3** | 31/10/2017 | 14:52:40 | 57.0 | 1 | 29.850 | 29.550 | 38.040 | 41.970 | 42.156 | 70.262 | ... | 161.584 | 16 |
| **4** | 31/10/2017 | 15:10:33 | 58.0 | 2 | 29.853 | 29.564 | 38.036 | 41.963 | 42.154 | 70.181 | ... | 161.672 | 16 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **79** | 06/11/2017 | 09:28:58 | 133.0 | 77 | 29.855 | 29.565 | 38.023 | 41.958 | 42.145 | 69.780 | ... | 161.568 | 16 |
| **80** | 06/11/2017 | 09:44:34 | 134.0 | 78 | 29.864 | 29.561 | 38.019 | 41.939 | 42.141 | 70.044 | ... | 161.558 | 16 |
| **81** | 06/11/2017 | 10:00:42 | 135.0 | 79 | 29.861 | 29.568 | 38.014 | 41.964 | 42.145 | 69.914 | ... | 161.552 | 16 |

In [33]:

```
# check cmm datasets
cmm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Unnamed: 0        81 non-null     object 
 1   Unnamed: 1        81 non-null     object 
 2   Unnamed: 2        81 non-null     float64
 3   Unnamed: 3        82 non-null     object 
 4   ID @100           84 non-null     float64
 5   ID @55            84 non-null     float64
 6   38 dia @200       84 non-null     float64
 7   42 dia @140       84 non-null     float64
 8   42 dia @80        84 non-null     float64
 9   Base angle F      84 non-null     float64
 10  Base angle BR     84 non-null     float64
 11  Base angle BL     84 non-null     float64
 12  162mm taper F     84 non-null     float64
 13  162mm taper BR    84 non-null     float64
 14  162mm taper BL    84 non-null     float64
 15  40.5mm taper F    84 non-null     float64
 16  40.5mm taper BR   84 non-null     float64
 17  40.5mm taper BL   84 non-null     float64
 18  Top1              84 non-null     float64
 19  Top2              84 non-null     float64
 20  Top3              84 non-null     float64
 21  Top4              84 non-null     float64
dtypes: float64(19), object(3)
memory usage: 14.6+ KB
```

In [34]:

```
# check null values
cmm.isnull().sum()
# from table below, all values are 0. We can see that there are no NA values in cmm
```

Out[34]:

```
Unnamed: 0          3
Unnamed: 1          3
Unnamed: 2          3
Unnamed: 3          2
ID @100             0
ID @55              0
38 dia @200         0
42 dia @140         0
42 dia @80          0
Base angle F        0
Base angle BR       0
Base angle BL       0
162mm taper F       0
162mm taper BR      0
162mm taper BL      0
40.5mm taper F      0
40.5mm taper BR     0
40.5mm taper BL     0
Top1                0
Top2                0
Top3                0
Top4                0
dtype: int64
```

In [35]:

```
# rename features
cmm.rename(columns={'Unnamed: 3':'Measurement'}, inplace=True)
cmm.drop(columns=['Unnamed: 0'], inplace=True)
cmm.drop(columns=['Unnamed: 1'], inplace=True)
cmm.drop(columns=['Unnamed: 2'], inplace=True)
cmm.iloc[0,0] = 'Nominal value'
cmm.iloc[1,0] = 'Upper error'
cmm.iloc[2,0] = 'Lower error'
cmm
```

Out[35]:

| | Measurement | ID @100 | ID @55 | 38 dia @200 | 42 dia @140 | 42 dia @80 | Base angle F | Base angle BR | Base angle BL | 162mm taper F | 162m tap E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Nominal value | 30.000 | 30.000 | 38.000 | 42.000 | 42.000 | 70.000 | 70.000 | 70.000 | 162.000 | 162.0 |
| **1** | Upper error | 0.300 | 0.300 | 0.400 | 0.400 | 0.400 | 1.000 | 1.000 | 1.000 | 1.000 | 1.0 |
| **2** | Lower error | -0.300 | -0.300 | -0.400 | -0.400 | -0.400 | -1.000 | -1.000 | -1.000 | -1.000 | -1.0 |
| **3** | 1 | 29.850 | 29.550 | 38.040 | 41.970 | 42.156 | 70.262 | 69.925 | 69.862 | 161.584 | 161.6 |
| **4** | 2 | 29.853 | 29.564 | 38.036 | 41.963 | 42.154 | 70.181 | 70.056 | 70.390 | 161.672 | 161.5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **79** | 77 | 29.855 | 29.565 | 38.023 | 41.958 | 42.145 | 69.780 | 69.958 | 69.817 | 161.568 | 161.5 |
| **80** | 78 | 29.864 | 29.561 | 38.019 | 41.939 | 42.141 | 70.044 | 69.606 | 69.864 | 161.558 | 161.6 |
| **81** | 79 | 29.861 | 29.568 | 38.014 | 41.964 | 42.145 | 69.914 | 69.731 | 69.982 | 161.552 | 161.6 |
| **82** | 80 | 29.860 | 29.572 | 38.016 | 41.937 | 42.132 | 69.873 | 69.682 | 69.927 | 161.741 | 161.5 |
| **83** | 81 | 29.864 | 29.566 | 38.001 | 41.937 | 42.122 | 69.667 | 69.663 | 69.952 | 161.625 | 161.4 |

84 rows × 19 columns

As each measurement has different standard, I will transfer the norminal value to the same standard which is 1. Therefore, each measurement values are around 1 with different variance.

**for loop, calculate the norminal values. For each column, I transfer all norminal value to 1 in order to getthe same norminal values. To do that, I divide all values with their corresponding column's norminal values. Bydoing so, all measurement will have the same standard value as 1, and each measurement value differs around 1.**

In [36]:

```
# For loop, standardize norminal values into 1.
for i in range(1, len(cmm.columns)):
    cmm.iloc[:, i] = cmm.iloc[:, i] / cmm.iloc[0,i]
cmm
```

| | ID @100 | ID @55 | 38 dia @200 | 42 dia @140 | 42 dia @80 | Base angle F | Base angle BR | Base angle BL | 162mm taper F | 162mm taper BR | 16 tap |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.0 |
| | 0.010000 | 0.010000 | 0.010526 | 0.009524 | 0.009524 | 0.014286 | 0.014286 | 0.014286 | 0.006173 | 0.006173 | 0.0 |
| | -0.010000 | -0.010000 | -0.010526 | -0.009524 | -0.009524 | -0.014286 | -0.014286 | -0.014286 | -0.006173 | -0.006173 | -0.0 |
| | 0.995000 | 0.985000 | 1.001053 | 0.999286 | 1.003714 | 1.003743 | 0.998929 | 0.998029 | 0.997432 | 0.998019 | 0.9 |
| | 0.995100 | 0.985467 | 1.000947 | 0.999119 | 1.003667 | 1.002586 | 1.000800 | 1.005571 | 0.997975 | 0.997315 | 0.9 |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| | 0.995167 | 0.985500 | 1.000605 | 0.999000 | 1.003452 | 0.996857 | 0.999400 | 0.997386 | 0.997333 | 0.997241 | 0.9 |
| | 0.995467 | 0.985367 | 1.000500 | 0.998548 | 1.003357 | 1.000629 | 0.994371 | 0.998057 | 0.997272 | 0.997562 | 0.9 |
| | 0.995367 | 0.985600 | 1.000368 | 0.999143 | 1.003452 | 0.998771 | 0.996157 | 0.999743 | 0.997235 | 0.997617 | 0.9 |

In [37]:

```
# Build measurement standard data
cmm_measure = cmm.iloc[:3]
cmm_measure
```

Out[37]:

| | Measurement | ID @100 | ID @55 | 38 dia @200 | 42 dia @140 | 42 dia @80 | Base angle F | Base angle BR | Base angle BL |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Nominal value | 1.00 | 1.00 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| **1** | Upper error | 0.01 | 0.01 | 0.010526 | 0.009524 | 0.009524 | 0.014286 | 0.014286 | 0.014286 |
| **2** | Lower error | -0.01 | -0.01 | -0.010526 | -0.009524 | -0.009524 | -0.014286 | -0.014286 | -0.014286 |

In [38]:

```
# Build measure values for each part ID
cmm_part = cmm.iloc[3:].rename(columns={'Measurement':'Part_ID'})
# Set ID index
cmm_part.set_index('Part_ID', inplace=True)
cmm_part
```

Out[38]:

| | ID @100 | ID @55 | 38 dia @200 | 42 dia @140 | 42 dia @80 | Base angle F | Base angle BR | Base angle BL | 162m tape |
|---|---|---|---|---|---|---|---|---|---|
| **Part_ID** | | | | | | | | | |
| **1** | 0.995000 | 0.985000 | 1.001053 | 0.999286 | 1.003714 | 1.003743 | 0.998929 | 0.998029 | 0.9974 |
| **2** | 0.995100 | 0.985467 | 1.000947 | 0.999119 | 1.003667 | 1.002586 | 1.000800 | 1.005571 | 0.9979 |
| **3** | 0.995133 | 0.985267 | 1.000632 | 0.998929 | 1.003357 | 1.000386 | 0.998629 | 0.996871 | 0.9977 |
| **4** | 0.995167 | 0.985433 | 1.000447 | 0.998929 | 1.003238 | 0.996700 | 0.995786 | 0.996971 | 0.9984 |
| **5** | 0.995333 | 0.985700 | 1.000184 | 0.998643 | 1.003214 | 0.999386 | 0.996300 | 0.999857 | 0.9972 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **77** | 0.995167 | 0.985500 | 1.000605 | 0.999000 | 1.003452 | 0.996857 | 0.999400 | 0.997386 | 0.9973 |
| **78** | 0.995467 | 0.985367 | 1.000500 | 0.998548 | 1.003357 | 1.000629 | 0.994371 | 0.998057 | 0.9972 |
| **79** | 0.995367 | 0.985600 | 1.000368 | 0.999143 | 1.003452 | 0.998771 | 0.996157 | 0.999743 | 0.9972 |
| **80** | 0.995333 | 0.985733 | 1.000421 | 0.998500 | 1.003143 | 0.998186 | 0.995457 | 0.998957 | 0.9984 |
| **81** | 0.995467 | 0.985533 | 1.000026 | 0.998500 | 1.002905 | 0.995243 | 0.995186 | 0.999314 | 0.9976 |

81 rows × 18 columns

**To calculate the total error for each part ID. As errors in 18 measures differs in positive and negative values, simply sum the error will cause positive error to cancel out the negative ones. In addition, each measurements' error difference is very small. Thus, I will square each error in 18 measurements to magnify errors, and then take the sum as the total error for each part ID.**

In [39]:

```
cmm_part['total_error'] = 0
for i in range(0,len(cmm_part.columns)-1):
    # calculate the square for each error, minus norminal values 1
    cmm_part.iloc[:,i] = np.square(cmm_part.iloc[:,i]-1)
    cmm_part['total_error'] = cmm_part.iloc[:,i] + cmm_part['total_error'] # calcula

cmm_part
```

Out[39]:

| Part_ID | ID @100 | ID @55 | 38 dia @200 | 42 dia @140 | 42 dia @80 | Base angle F | Base angle BR | Base angle BL | 162mm taper F | 16 tap |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.000025 | 0.000225 | 1.108033e-06 | 5.102041e-07 | 0.000014 | 1.400898e-05 | 1.147959e-06 | 3.886531e-06 | 0.000007 | 0.0( |
| 2 | 0.000024 | 0.000211 | 8.975069e-07 | 7.760771e-07 | 0.000013 | 6.685918e-06 | 6.400000e-07 | 3.104082e-05 | 0.000004 | 0.0( |
| 3 | 0.000024 | 0.000217 | 3.988920e-07 | 1.147959e-06 | 0.000011 | 1.487755e-07 | 1.880816e-06 | 9.787959e-06 | 0.000005 | 0.0( |
| 4 | 0.000023 | 0.000212 | 2.001385e-07 | 1.147959e-06 | 0.000010 | 1.089000e-05 | 1.776020e-05 | 9.172245e-06 | 0.000002 | 0.0( |
| 5 | 0.000022 | 0.000204 | 3.393352e-08 | 1.841837e-06 | 0.000010 | 3.773469e-07 | 1.369000e-05 | 2.040816e-08 | 0.000007 | 0.0( |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

**From the table above, we can see the squared error for each part and the total error for each Part ID.**

In [40]:

```
# Rank total errors
cmm_part.sort_values(by='total_error', inplace=True, ascending=False)
cmm_part[['total_error']].reset_index()
```

Out[40]:

| | Part_ID | total_error |
|---|---|---|
| **0** | 23 | 0.017028 |
| **1** | 28 | 0.016768 |
| **2** | 32 | 0.016551 |
| **3** | 27 | 0.016497 |
| **4** | 21 | 0.016447 |
| **...** | ... | ... |
| **76** | 67 | 0.009870 |
| **77** | 71 | 0.009745 |
| **78** | 66 | 0.009650 |
| **79** | 64 | 0.008820 |
| **80** | 65 | 0.007599 |

81 rows × 2 columns

**From the table above, we can see that part ID 23, 28, 32 have the top three largest total errors. For these five part ID, they may have highest probability of bad quality. To further analyse which sensor parameters most affect these three parts, I will calculate clustermap to visualize the influence of parameters and total_error.**

In [41]:

```
# Read five bad quality parts
part23 = pd.read_csv('Scope0023.csv', encoding='unicode_escape')
part28 = pd.read_csv('Scope0028.csv', encoding='unicode_escape')
part32 = pd.read_csv('Scope0032.csv', encoding='unicode_escape')
```

For sensor machine parameters, there are 99 parameters in total. However, not all parameters' status is 'in use'. I assume that these 'not in use' and 'unknow parameters' are old fasioned sensors which are not used. In this case, I will only analyse quality based on sensor parameters which is 'in use' status.

In [42]:

```
# Check the parameter sheet
data = pd.ExcelFile('ForgedPartDataStructureSummaryv3.xlsx')
data_names = data.sheet_names
data_names
```

Out[42]:

```
['CMM Data Structure', 'Machine Parameters', 'Reordered Machine Parame
ters']
```

In [43]:

```
# Only get 'Machine Parameters' dataset
for par in data_names:
    if par == 'Machine Parameters':
        parameter = pd.read_excel('ForgedPartDataStructureSummaryv3.xlsx', sheet_nam
parameter
```

Out[43]:

| | ID | Classification | Signal Name | Description | Nominal Value | Unit | Notes |
|---|---|---|---|---|---|---|---|
| **0** | 1 | In Use | Timer Tick [ms] | NaN | NaN | ms | NaN |
| **1** | 2 | NaN | Block-Nr | NaN | NaN | NaN | Interpretation of this variable is unknown |
| **2** | 3 | In Use | Power [kW] | Actual value of forging drive power | NaN | kW | NaN |
| **3** | 4 | In Use | Force [kN] | Hammer Force | NaN | kN | NaN |
| **4** | 5 | In Use | A_ges_vibr | NaN | NaN | NaN | Interpretation of this variable is unknown |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **94** | 95 | Not changing | $H1P_Y11 (U11S7) | NaN | NaN | NaN | Interpretation of this variable is unknown |
| **95** | 96 | Not in Use | $U_GH_NOMEXT_2 (U26S1) | Nominal Heater 2 | NaN | NaN | NaN |
| **96** | 97 | Not in Use | $U_GH_HEATON_2 (U26S0) | Digital Signal to On/Off induction heater 2 (d... | NaN | NaN | NaN |
| **97** | 98 | Not in Use | $U_GH_NOMEXT_1 (U25S1) | Nominal Heater 1 | NaN | NaN | No variable data |
| **98** | 99 | In Use | $U_GH_HEATON_1 (U25S0).1 | Digital Signal to On/Off induction heater 1 (d... | NaN | Digital | Digital signal to indictate induction heating ... |

99 rows × 7 columns

In [44]:

```
# Check columns of parameter datasets
parameter.columns
```

Out[44]:

```
Index(['ID', 'Classification', 'Signal Name', 'Description', 'Nominal
Value',
       'Unit', 'Notes'],
      dtype='object')
```

In [45]:

```
# There are many sensor parameter status, I only choose parameter with 'In Use'
parameter['Classification'].unique()
```

Out[45]:

```
array(['In Use', nan, 'Auxiliary Process Measurement', 'Not in Use',
       'Not changing', 'Duplicate',
       'Valid (but does not contian information)'], dtype=object)
```

In [60]:

```
Find signal name which classification is 'in use'
gnal_name = parameter[parameter['Classification'] == 'In Use'] # Select all 'In Use'
gnal_name = signal_name[['Signal Name']] # only choose Signal Name column
gnal_name
```

Out[60]:

| | Signal Name |
|---|---|
| **0** | Timer Tick [ms] |
| **2** | Power [kW] |
| **3** | Force [kN] |
| **4** | A_ges_vibr |
| **5** | Schlagzahl [1/min] |
| **6** | EXZ_pos [deg] |
| **9** | A_ACTpos [mm] |
| **11** | DB_ACTpos [mm] |
| **13** | L_ACTpos [mm] |
| **14** | R_ACTpos [mm] |

In [47]:

```
# get signal_name parameters
signal_name['Signal Name'].unique()
```

Out[47]:

```
array(['Timer Tick [ms]', 'Power [kW]', 'Force [kN]', 'A_ges_vibr',
       'Schlagzahl [1/min]', 'EXZ_pos [deg]', 'A_ACTpos [mm]',
       'DB_ACTpos [mm]', 'L_ACTpos [mm]', 'R_ACTpos [mm]',
       'SBA_ActPos [mm]', 'INDA_ACTpos [deg]', 'A_ACT_Force [kN]',
       'DB_ACT_Force [kN]', 'L_ACTspd [mm/min]', 'R_ACTspd [mm/min]',
       'SBA_NomPos [mm] [mm]', 'A_ACTspd [mm/min]', 'DB_ACTspd [mm/mi
n]',
       'L_NOMpos [mm]', 'R_NOMpos [mm]', 'SBA_OUT [%]', 'A_NOMpos [m
m]',
       'DB_NOMpos [mm]', 'L_OUT [%]', 'R_OUT [%]', 'Feedback SBA [%]',
       'A_OUT [%]', 'DB_OUT [%]', 'L_NOMspd [mm/min]',
       'R_NOMspd [mm/min]', 'Frc_Volt', 'A_NOMspd [mm/min]',
       'DB_NOMspd [mm/min]', 'Feedback L [%]', 'Feedback R [%]',
       'Speed Vn_1 [rpm]', 'NOMforceSPA [kN]', 'IP_ActSpd [mm/min]',
       'IP_ActPos [mm]', 'SPA_OUT [%]', 'Feedback A [%]',
       'Feedback DB [%]', 'IP_NomSpd [mm/min]', 'IP_NomPos',
       'Feedback_SPA [%]', 'ForgingBox_Temp', 'TMP_Ind_U1 [°C]',
       'TMP_Ind_F [°C]', 'IP_Out [%]', 'ACTforceSPA [kN]',
       '$U_GH_HEATON_1 (U25S0).1'], dtype=object)
```

## Drop signal names which parameters are not 'In Use'

## Part23

In [48]:

```
# Select datasets which only 'In Use' signal name parameters
part23 = part23.loc[:,['Timer Tick [ms]', 'Power [kW]', 'Force [kN]', 'A_ges_vibr',
        'Schlagzahl [1/min]', 'EXZ_pos [deg]', 'A_ACTpos [mm]',
        'DB_ACTpos [mm]', 'L_ACTpos [mm]', 'R_ACTpos [mm]',
        'SBA_ActPos [mm]', 'INDA_ACTpos [deg]', 'A_ACT_Force [kN]',
        'DB_ACT_Force [kN]', 'L_ACTspd [mm/min]', 'R_ACTspd [mm/min]',
        'SBA_NomPos [mm] [mm]', 'A_ACTspd [mm/min]', 'DB_ACTspd [mm/min]',
        'L_NOMpos [mm]', 'R_NOMpos [mm]', 'SBA_OUT [%]', 'A_NOMpos [mm]',
        'DB_NOMpos [mm]', 'L_OUT [%]', 'R_OUT [%]', 'Feedback SBA [%]',
        'A_OUT [%]', 'DB_OUT [%]', 'L_NOMspd [mm/min]',
        'R_NOMspd [mm/min]', 'Frc_Volt', 'A_NOMspd [mm/min]',
        'DB_NOMspd [mm/min]', 'Feedback L [%]', 'Feedback R [%]',
        'Speed Vn_1 [rpm]', 'NOMforceSPA [kN]', 'IP_ActSpd [mm/min]',
        'IP_ActPos [mm]', 'SPA_OUT [%]', 'Feedback A [%]',
        'Feedback DB [%]', 'IP_NomSpd [mm/min]', 'IP_NomPos',
        'Feedback_SPA [%]', 'ForgingBox_Temp', 'TMP_Ind_U1 [°C]',
        'TMP_Ind_F [°C]', 'IP_Out [%]', 'ACTforceSPA [kN]',
        '$U_GH_HEATON_1 (U25S0).1']]
part23
```

| | [ms] | [kW] | [kN] | A_ges_vibr | [1/min] | [deg] | [mm] | [mm] | [mm] |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2599023562 | 46.912811 | 0.0 | 5.971617 | 1207.17628 | 206.282891 | 1199.945 | 499.820 | 94.899990 |
| **1** | 2599023572 | 46.912830 | 0.0 | 5.971617 | 1207.83657 | 278.660820 | 1199.945 | 499.630 | 94.899992 |
| **2** | 2599023582 | 46.912993 | 0.0 | 5.211279 | 1207.21498 | 351.214531 | 1199.940 | 499.375 | 94.899988 |
| **3** | 2599023592 | 46.912733 | 0.0 | 5.313335 | 1206.20648 | 63.548516 | 1199.940 | 499.100 | 94.899978 |
| **4** | 2599023602 | 46.914011 | 0.0 | 5.687249 | 1206.18934 | 136.102227 | 1199.945 | 498.810 | 94.899967 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **17856** | 2599202122 | 46.765512 | 0.0 | 10.161914 | 1207.60470 | 270.311211 | 1199.925 | 500.000 | 94.900134 |
| **17857** | 2599202132 | 46.760248 | 0.0 | 10.161914 | 1206.48264 | 342.820977 | 1199.930 | 500.000 | 94.900141 |
| **17858** | 2599202142 | 46.755093 | 0.0 | 10.161914 | 1206.29344 | 55.111016 | 1199.930 | 500.000 | 94.900144 |
| **17859** | 2599202152 | 46.746974 | 0.0 | 10.161914 | 1206.63813 | 127.664727 | 1199.930 | 500.000 | 94.900148 |
| **17860** | 2599202162 | 46.736237 | 0.0 | 10.161914 | 1205.39208 | 199.954766 | 1199.930 | 500.000 | 94.900152 |

In [49]:

```
# As each value differ, I use normalization to rerange all values into [0,1] field
transfer = MinMaxScaler()
part23_new = transfer.fit_transform(part23)       # Normalize part23 values
part23_new = pd.DataFrame(part23_new, columns = part23.columns)   # Transfer to Data
part23_error = cmm_part['total_error'].iloc[0]    # Get the Part23's total error
part23_new['part23_error'] = part23_error          # Create a new column for part 23's
part23_new.head()
```

Out[49]:

| | Timer Tick [ms] | Power [kW] | Force [kN] | A_ges_vibr | Schlagzahl [1/min] | EXZ_pos [deg] | A_ACTpos [mm] | DB_ACTpos [mm] | L_ACT[ [n |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000000 | 0.013465 | 0.0 | 0.000402 | 0.670256 | 0.573068 | 0.999788 | 0.998589 | 0.999 |
| **1** | 0.000056 | 0.013466 | 0.0 | 0.000402 | 0.722384 | 0.774142 | 0.999788 | 0.997615 | 0.999 |
| **2** | 0.000112 | 0.013470 | 0.0 | 0.000351 | 0.673311 | 0.975705 | 0.999783 | 0.996306 | 0.999 |
| **3** | 0.000168 | 0.013463 | 0.0 | 0.000358 | 0.593692 | 0.176535 | 0.999783 | 0.994896 | 0.999 |
| **4** | 0.000224 | 0.013495 | 0.0 | 0.000383 | 0.592339 | 0.378098 | 0.999788 | 0.993408 | 0.999 |

5 rows × 53 columns

In [50]:

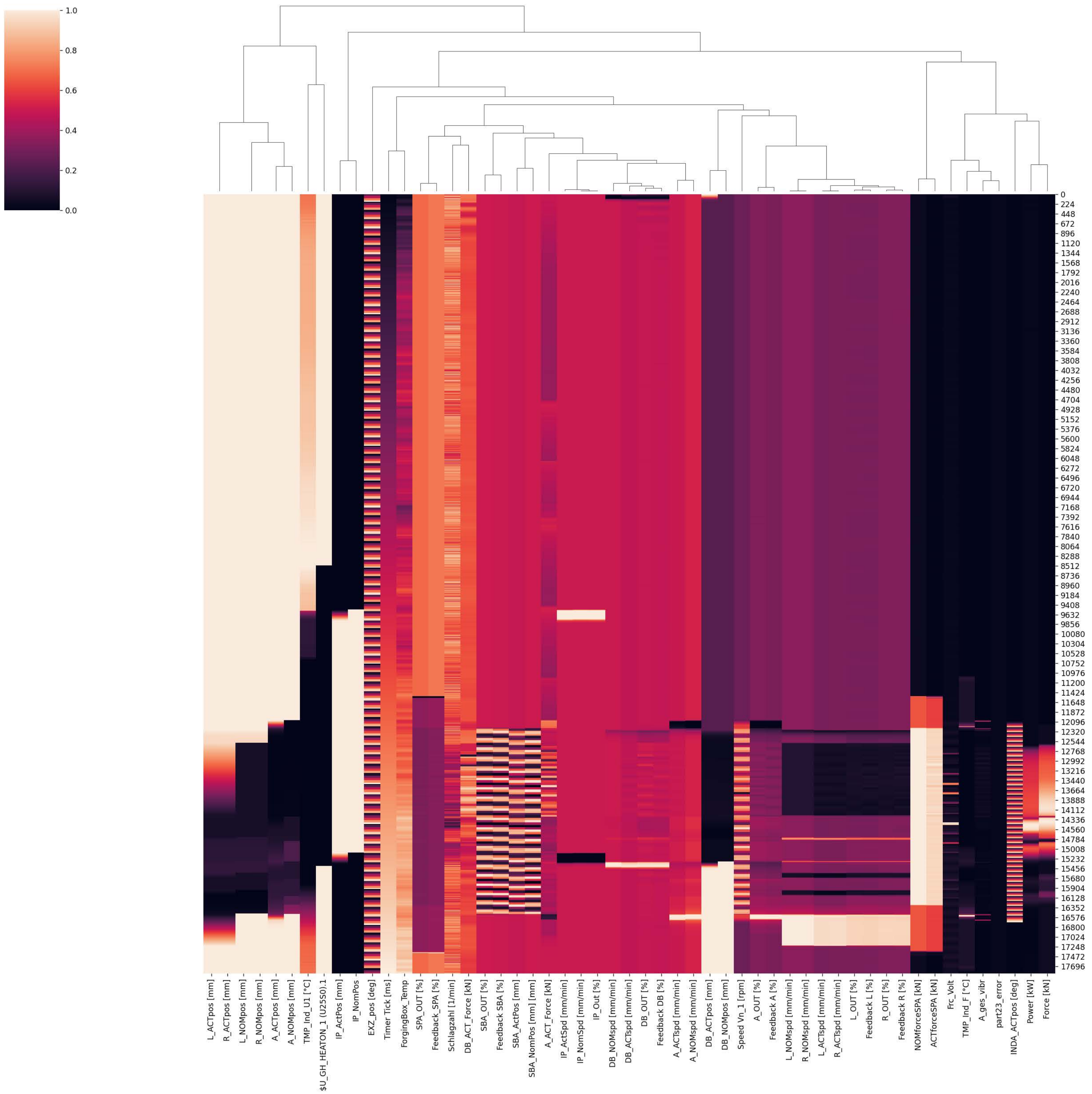
```
x,y= part23_new.iloc[:, 0:-1], part23_new.loc[:, 'part23_error']
```

In [51]:

```
sns.clustermap(data=part23_new, pivot_kws=None, method='average',

figsize=(5, 5), cbar_kws=None,
 row_cluster=False, col_cluster=True)
```

# The Clustermap of Part ID.23



In [52]: **Part28**

```
# Select datasets which only 'In Use' signal name parameters
part28 = part28.loc[:,['Timer Tick [ms]', 'Power [kW]', 'Force [kN]', 'A_ges_vibr',
        'Schlagzahl [1/min]', 'EXZ_pos [deg]', 'A_ACTpos [mm]',
        'DB_ACTpos [mm]', 'L_ACTpos [mm]', 'R_ACTpos [mm]',
        'SBA_ActPos [mm]', 'INDA_ACTpos [deg]', 'A_ACT_Force [kN]',
        'DB_ACT_Force [kN]', 'L_ACTspd [mm/min]', 'R_ACTspd [mm/min]',
        'SBA_NomPos [mm] [mm]', 'A_ACTspd [mm/min]', 'DB_ACTspd [mm/min]',
        'L_NOMpos [mm]', 'R_NOMpos [mm]', 'SBA_OUT [%]', 'A_NOMpos [mm]',
        'DB_NOMpos [mm]', 'L_OUT [%]', 'R_OUT [%]', 'Feedback SBA [%]',
        'A_OUT [%]', 'DB_OUT [%]', 'L_NOMspd [mm/min]',
        'R_NOMspd [mm/min]', 'Frc_Volt', 'A_NOMspd [mm/min]',
        'DB_NOMspd [mm/min]', 'Feedback L [%]', 'Feedback R [%]',
        'Speed Vn_1 [rpm]', 'NOMforceSPA [kN]', 'IP_ActSpd [mm/min]',
        'IP_ActPos [mm]', 'SPA_OUT [%]', 'Feedback A [%]',
        'Feedback DB [%]', 'IP_NomSpd [mm/min]', 'IP_NomPos',
        'Feedback_SPA [%]', 'ForgingBox_Temp', 'TMP_Ind_U1 [°C]',
        'TMP_Ind_F [°C]', 'IP_Out [%]', 'ACTforceSPA [kN]',
        '$U_GH_HEATON_1 (U25S0).1']]
part28
```

Out[52]:

| | Timer Tick [ms] | Power [kW] | Force [kN] | A_ges_vibr | Schlagzahl [1/min] | EXZ_pos [deg] | A_ACTpos [mm] | DB_ACTpos [mm] | L_ACTpos [mm] |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2599919189 | 46.007757 | 0.0 | 5.753715 | 1205.25764 | 8.309258 | 1199.965 | 499.925 | 94.899937 |
| **1** | 2599919199 | 46.010688 | 0.0 | 5.753715 | 1206.04694 | 80.731133 | 1199.965 | 499.745 | 94.899938 |
| **2** | 2599919209 | 46.012026 | 0.0 | 5.753715 | 1205.00891 | 152.933281 | 1199.965 | 499.465 | 94.899938 |
| **3** | 2599919219 | 46.012928 | 0.0 | 5.112676 | 1204.83110 | 225.355156 | 1199.965 | 499.255 | 94.899939 |
| **4** | 2599919229 | 46.013953 | 0.0 | 0.536051 | 1204.60876 | 297.601250 | 1199.965 | 498.965 | 94.899943 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **17868** | 2600097869 | 45.919296 | 0.0 | 5.107856 | 1203.67931 | 31.072930 | 1199.965 | 500.000 | 94.899969 |
| **17869** | 2600097879 | 45.916606 | 0.0 | 5.107856 | 1204.42075 | 103.362969 | 1199.965 | 500.000 | 94.899973 |
| **17870** | 2600097889 | 45.910914 | 0.0 | 5.107856 | 1204.92432 | 175.521172 | 1199.965 | 500.000 | 94.899980 |

In [53]:

```
# As each value differ, I use normalization to rerange all values into [0,1] field
transfer = MinMaxScaler()
part28_new = transfer.fit_transform(part28)       # Normalize part28 values
part28_new = pd.DataFrame(part28_new, columns = part23.columns)           # Transfe
part28_error = cmm_part['total_error'].iloc[1]   # Get the Part28's total error, whi
part28_new['part28_error'] = part28_error        # Create a new column for part 28's
part28_new.head()
```

Out[53]:

| | Timer Tick [ms] | Power [kW] | Force [kN] | A_ges_vibr | Schlagzahl [1/min] | EXZ_pos [deg] | A_ACTpos [mm] | DB_ACTpos [mm] | L_ACTp [m |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000000 | 0.013173 | 0.0 | 0.000408 | 0.725166 | 0.023074 | 0.999808 | 0.998718 | 0.999 |
| **1** | 0.000056 | 0.013245 | 0.0 | 0.000408 | 0.789084 | 0.224271 | 0.999808 | 0.997794 | 0.999 |
| **2** | 0.000112 | 0.013277 | 0.0 | 0.000408 | 0.705024 | 0.424857 | 0.999808 | 0.996358 | 0.999 |
| **3** | 0.000168 | 0.013299 | 0.0 | 0.000363 | 0.690625 | 0.626053 | 0.999808 | 0.995281 | 0.999 |
| **4** | 0.000224 | 0.013324 | 0.0 | 0.000038 | 0.672620 | 0.826761 | 0.999808 | 0.993793 | 0.999 |

5 rows × 53 columns

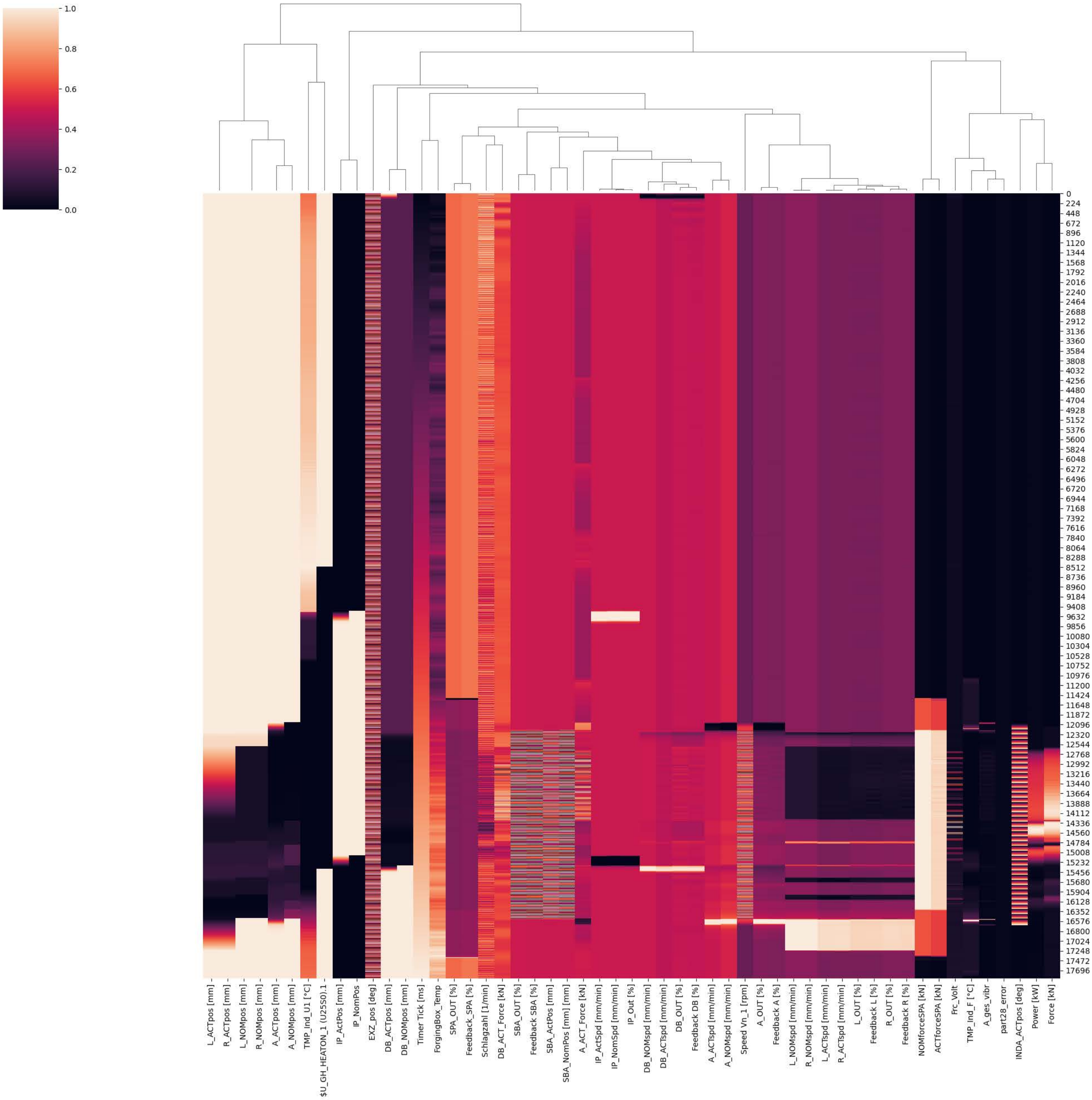
In [54]:

```
x,y= part28_new.iloc[:, 0:-1], part28_new.loc[:, 'part28_error']
```

In [55]:

```
sns.clustermap(data=part28_new, pivot_kws=None, method='average',
               figsize=(5, 5), cbar_kws=None,
                row_cluster=False, col_cluster=True)
```

## The Cluster of Part ID.28



In [56]:

```
# Select datasets which only 'In Use' signal name parameters
part32 = part32.loc[:,['Timer Tick [ms]', 'Power [kW]', 'Force [kN]', 'A_ges_vibr',
        'Schlagzahl [1/min]', 'EXZ_pos [deg]', 'A_ACTpos [mm]',
        'DB_ACTpos [mm]', 'L_ACTpos [mm]', 'R_ACTpos [mm]',
        'SBA_ActPos [mm]', 'INDA_ACTpos [deg]', 'A_ACT_Force [kN]',
        'DB_ACT_Force [kN]', 'L_ACTspd [mm/min]', 'R_ACTspd [mm/min]',
        'SBA_NomPos [mm] [mm]', 'A_ACTspd [mm/min]', 'DB_ACTspd [mm/min]',
        'L_NOMpos [mm]', 'R_NOMpos [mm]', 'SBA_OUT [%]', 'A_NOMpos [mm]',
        'DB_NOMpos [mm]', 'L_OUT [%]', 'R_OUT [%]', 'Feedback SBA [%]',
        'A_OUT [%]', 'DB_OUT [%]', 'L_NOMspd [mm/min]',
        'R_NOMspd [mm/min]', 'Frc_Volt', 'A_NOMspd [mm/min]',
        'DB_NOMspd [mm/min]', 'Feedback L [%]', 'Feedback R [%]',
        'Speed Vn_1 [rpm]', 'NOMforceSPA [kN]', 'IP_ActSpd [mm/min]',
        'IP_ActPos [mm]', 'SPA_OUT [%]', 'Feedback A [%]',
        'Feedback DB [%]', 'IP_NomSpd [mm/min]', 'IP_NomPos',
        'Feedback_SPA [%]', 'ForgingBox_Temp', 'TMP_Ind_U1 [°C]',
        'TMP_Ind_F [°C]', 'IP_Out [%]', 'ACTforceSPA [kN]',
        '$U_GH_HEATON_1 (U25S0).1']]
part32
```

Out[56]:

| | Timer Tick [ms] | Power [kW] | Force [kN] | A_ges_vibr | Schlagzahl [1/min] | EXZ_pos [deg] | A_ACTpos [mm] | DB_ACTpos [mm] | L_ACTpo [mm |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2600637294 | 46.215480 | 0.000000 | 5.224182 | 1208.07322 | 183.035820 | 1199.975 | 499.980 | 94.90005 |
| **1** | 2600637304 | 46.216380 | 0.000000 | 5.429102 | 1207.52203 | 255.589531 | 1199.980 | 499.895 | 94.90005 |
| **2** | 2600637314 | 46.219487 | 0.000000 | 5.429102 | 1206.78169 | 327.967461 | 1199.980 | 499.725 | 94.90004 |
| **3** | 2600637324 | 46.220170 | 0.000000 | 5.429102 | 1205.83432 | 40.477227 | 1199.980 | 499.440 | 94.90004 |
| **4** | 2600637334 | 46.222089 | 0.000000 | 5.429102 | 1206.89631 | 113.030937 | 1199.980 | 499.220 | 94.90004 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **17869** | 2600815984 | 46.420413 | 34.599609 | 0.068260 | 1212.41813 | 137.772148 | 1199.955 | 500.000 | 94.89996 |
| **17870** | 2600815994 | 46.420047 | 34.599609 | 0.076315 | 1211.92815 | 210.545586 | 1199.955 | 500.000 | 94.89996 |
| **17871** | 2600816004 | 46.421076 | 34.599609 | 0.083133 | 1211.00230 | 283.055352 | 1199.955 | 500.000 | 94.89996 |

In [57]:

```
# As each value differ, I use normalization to rerange all values into [0,1] field
transfer = MinMaxScaler()
part32_new = transfer.fit_transform(part32)      # Normalize part32 values
part32_new = pd.DataFrame(part32_new, columns = part32.columns)             # Transfe
part32_error = cmm_part['total_error'].iloc[2]   # Get the Part32's total error, whi
part32_new['part32_error'] = part32_error        # Create a new column for part 32's
part32_new.head()
```

Out[57]:

| | Timer Tick [ms] | Power [kW] | Force [kN] | A_ges_vibr | Schlagzahl [1/min] | EXZ_pos [deg] | A_ACTpos [mm] | DB_ACTpos [mm] | L_ACT [n |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000000 | 0.004504 | 0.0 | 0.000362 | 0.364892 | 0.508427 | 0.999818 | 0.999026 | 0.999 |
| **1** | 0.000056 | 0.004526 | 0.0 | 0.000376 | 0.315285 | 0.710064 | 0.999823 | 0.998590 | 0.999 |
| **2** | 0.000112 | 0.004601 | 0.0 | 0.000376 | 0.248655 | 0.911212 | 0.999823 | 0.997718 | 0.999 |
| **3** | 0.000168 | 0.004618 | 0.0 | 0.000376 | 0.163393 | 0.112237 | 0.999823 | 0.996257 | 0.999 |
| **4** | 0.000224 | 0.004665 | 0.0 | 0.000376 | 0.258971 | 0.313874 | 0.999823 | 0.995128 | 0.999 |

5 rows × 53 columns

In [58]:

```
x,y= part32_new.iloc[:, 0:-1], part32_new.loc[:, 'part32_error']
```

In [59]:

```
sns.clustermap(data=part32_new, pivot_kws=None,
              figsize=(5, 5), cbar_kws=None,
               row_cluster=False, col_cluster=True)
```

# The Cluster of Part ID.32