Imports necessary for this notebook

In [1]:
```python
import numpy as np
import pandas as pd
from scipy import stats
import seaborn as sns
```

Let's read the data and take a look

In [2]:
```python
dataset_path = '../data/dataset_20221127.csv'
df = pd.read_csv(dataset_path)
df
```

Out[2]:

	state	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c
0	0	0.717573	0.714725	0.704585	0.694305	0.687011	0.683188	0.682031	0.6822
1	0	0.719402	0.717330	0.708022	0.698305	0.691041	0.686480	0.683860	0.6823
2	0	0.723090	0.721113	0.711632	0.701274	0.692945	0.687295	0.683872	0.6820
3	0	0.729627	0.726485	0.715613	0.704214	0.695404	0.689836	0.686908	0.6856
4	0	0.714636	0.714388	0.706626	0.698127	0.691818	0.688184	0.686566	0.6860
...	...	...	...	...	...	...	...	...	
495	2	0.722422	0.720920	0.712699	0.704530	0.699261	0.697420	0.698031	0.6996
496	2	0.726260	0.726227	0.725950	0.725444	0.724643	0.723600	0.722467	0.7214
497	2	0.726733	0.724837	0.715531	0.705948	0.699105	0.695240	0.693513	0.6929
498	2	0.747386	0.745534	0.735654	0.724924	0.716467	0.710962	0.708025	0.7069
499	2	0.740270	0.738251	0.728985	0.719566	0.713096	0.710130	0.709747	0.7105

500 rows × 2001 columns

How many rows do we have for each state?

In [3]:
```python
df.groupby(['state']).size()
```

Out[3]:
```
state
0    300
1    100
2    100
dtype: int64
```

three times more working engines and an equal number of each type of broken engines

Are there missing or NAN values?

In [4]:
```python
nan_count = df.isna().sum().sum()
print(nan_count, "nan values found!")
```

```
23 nan values found!
```

how many NANs per label?

In [5]:
```python
na_df = df.isna()
na_df['state'] = df['state']
na_df.groupby('state').sum().sum(axis=1)
```

Out[5]:
```
state
0    13
1     3
2     7
dtype: int64
```

removing NAN values will get us an uneven quantity of the two type of broken engines

but for now let just remove this values

In [6]:
```python
not_na_df = df.dropna()
print(not_na_df.isna().sum().sum(), "nan values found!")
```

```
0 nan values found!
```

Some statistical information about the features

firstly, the label

In [7]:
```python
df['state'] = df['state'].astype('category')
df['state'].describe()
```

Out[7]:
```
count     500
unique      3
top         0
freq      300
Name: state, dtype: int64
```

now for current and tension

In [8]:
```python
labels = [0, 1, 2]
features = ['current', 'voltage']
label_split = []
for label in labels:
    feat_df_3d = not_na_df[not_na_df['state'] == label].drop(columns=['state
    feat_df_3d.columns = pd.MultiIndex.from_tuples([(features[c[0] == 't'],
    feat_df = pd.concat([feat_df_3d[c].unstack().reset_index()[0] for c in f
    feat_df.columns = features
    feat_df['state'] = label
    feat_df['state'] = feat_df['state'].astype('category')
    label_split.append(feat_df)
label_mult_index_df = pd.concat(label_split, axis=1, keys=labels)
label_mult_index_df.describe()
```

Out[8]:

	0		1			
	current	voltage	current	voltage	current	
count	288000.000000	288000.000000	97000.000000	97000.000000	94000.000000	94000.
mean	0.689264	22.044635	0.890812	21.665591	0.702334	22.
std	0.032326	1.025083	0.100502	2.721973	0.041531	2
min	0.622967	0.234658	0.710802	2.365598	0.595338	1.
25%	0.676121	21.379161	0.802473	19.589508	0.678091	21
50%	0.688727	21.863059	0.871379	20.672930	0.708244	21
75%	0.705339	22.533723	0.972040	23.531722	0.730281	22
max	11.554880	111.454900	3.658796	35.565650	7.564789	42

here we can see current and voltage details for each label

some data seems off, on label 1 the current has a max value of 1214, it is probably an outlier

on label 0, data seems to be more packed than on the other features

lets do some plotting to better analyze this hypothesis

In [9]:

```
label_df = pd.concat(label_split, ignore_index=True)
label_df['state'] = label_df['state'].astype('category')
label_df
```

Out[9]:

	current	voltage	state
0	0.717573	24.521245	0
1	0.719402	24.476110	0
2	0.729627	24.547889	0
3	0.714636	24.573959	0
4	0.709658	24.551453	0
...	...	...	...
478995	0.653173	21.514769	2
478996	0.692476	20.711599	2
478997	0.711744	23.732403	2
478998	0.669758	21.217464	2
478999	0.721182	22.908746	2

479000 rows × 3 columns

In [10]:

```
label_df_melted = label_df.melt(id_vars='state', value_vars=features, var_na
label_df_melted
```

Out[10]:

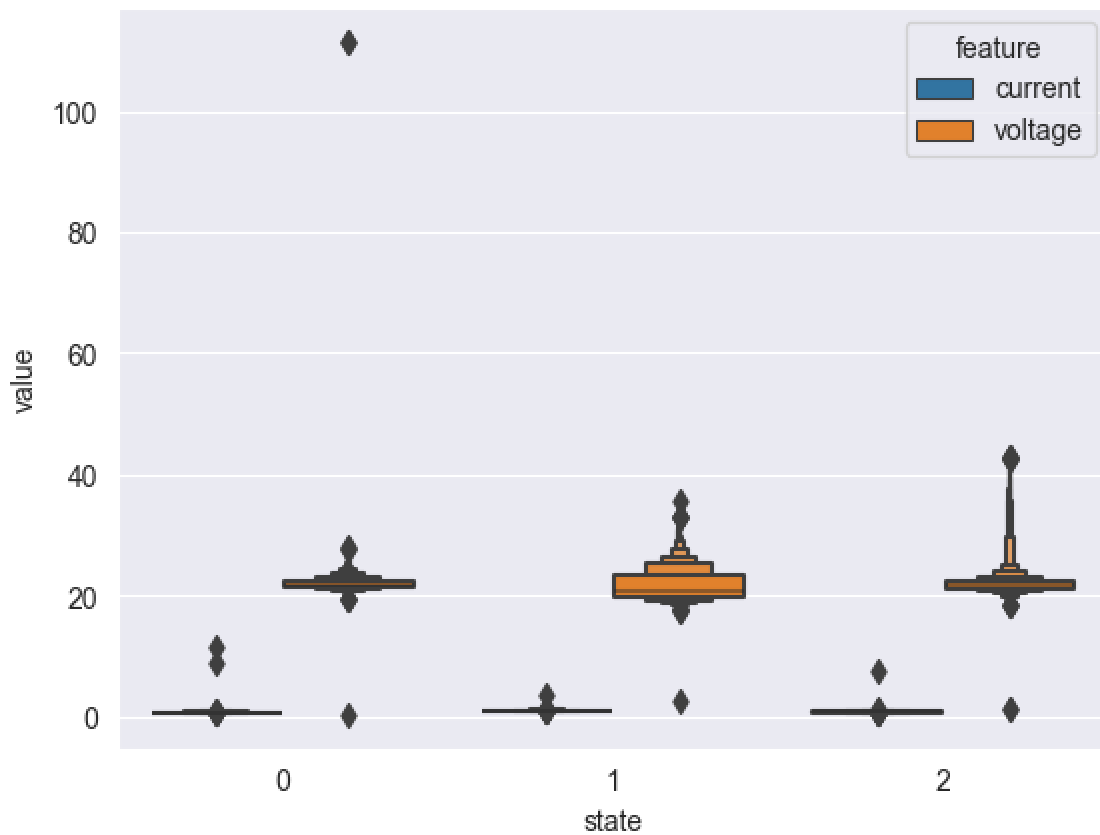
	state	feature	value
0	0	current	0.717573
1	0	current	0.719402
2	0	current	0.729627
3	0	current	0.714636
4	0	current	0.709658
...	...	...	...
957995	2	voltage	21.514769
957996	2	voltage	20.711599
957997	2	voltage	23.732403
957998	2	voltage	21.217464
957999	2	voltage	22.908746

958000 rows × 3 columns

In [11]:

```
sns.boxenplot(data=label_df_melted, x='state', y='value', hue='feature')
```

Out[11]: <AxesSubplot: xlabel='state', ylabel='value'>



we can clearly see some outliers

lets remake this plot removing data too far from the standard deviation

In [12]:
```
label_df_melted_clean = label_df_melted[(np.abs(stats.zscore(label_df_meltec
label_df_melted_clean
```

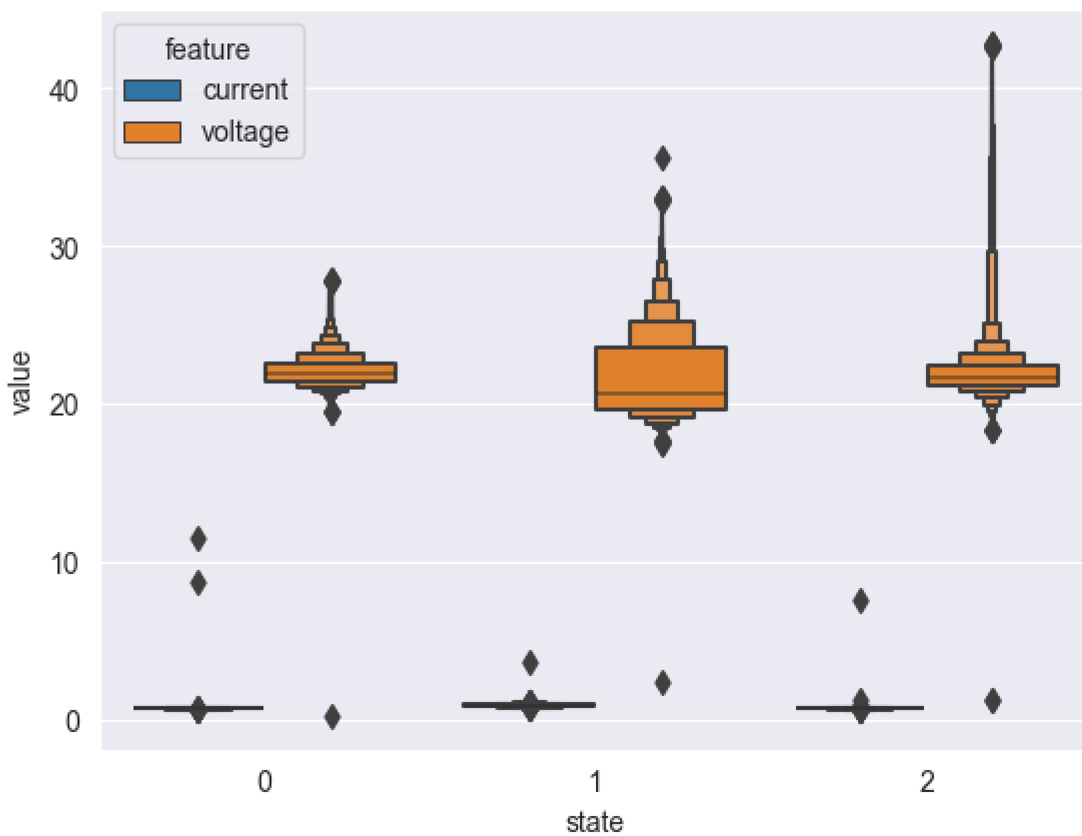
Out[12]:

	state	feature	value
0	0	current	0.717573
1	0	current	0.719402
2	0	current	0.729627
3	0	current	0.714636
4	0	current	0.709658
...	...	...	...
957995	2	voltage	21.514769
957996	2	voltage	20.711599
957997	2	voltage	23.732403
957998	2	voltage	21.217464
957999	2	voltage	22.908746

957998 rows × 3 columns

In [13]:
```
sns.boxenplot(data=label_df_melted_clean, x='state', y='value', hue='feature
```

Out[13]: <AxesSubplot: xlabel='state', ylabel='value'>



let's check the features separately

In [14]:
```
label_df_clean = label_df[(np.abs(stats.zscore(label_df[features])) < 5).all
label_df_clean
```

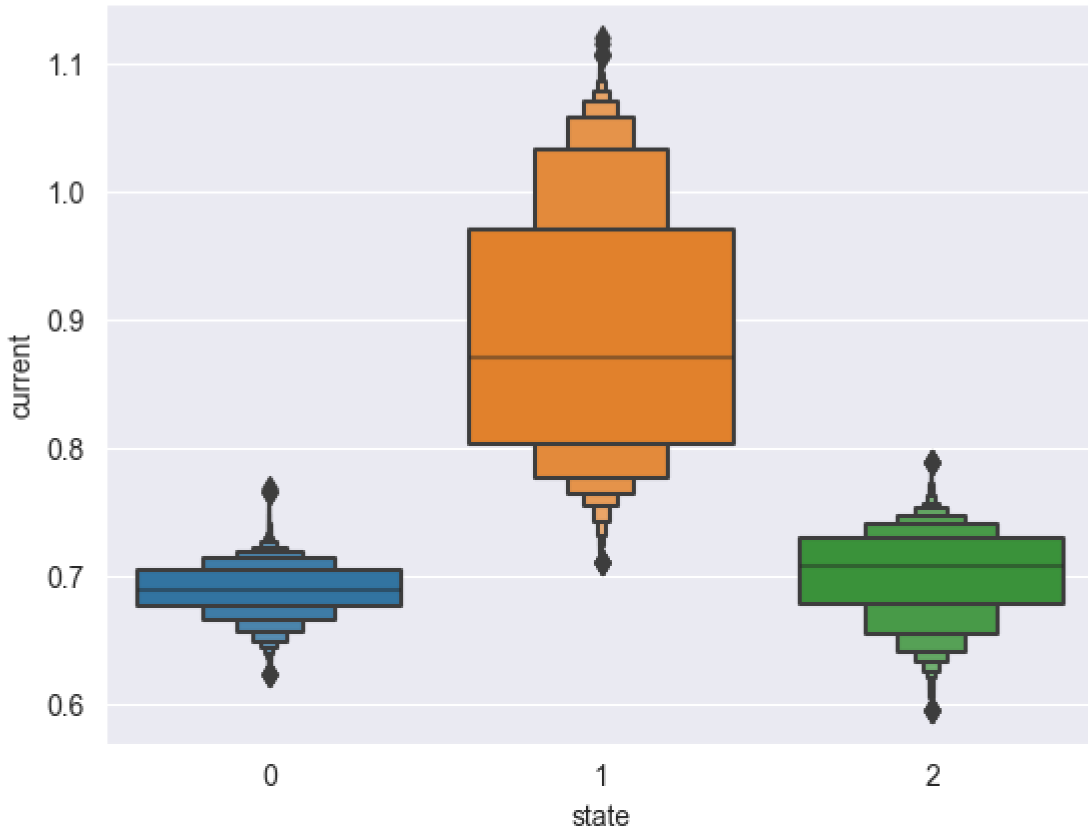
Out[14]:

	current	voltage	state
0	0.717573	24.521245	0
1	0.719402	24.476110	0
2	0.729627	24.547889	0
3	0.714636	24.573959	0
4	0.709658	24.551453	0
...	...	...	...
478995	0.653173	21.514769	2
478996	0.692476	20.711599	2
478997	0.711744	23.732403	2
478998	0.669758	21.217464	2
478999	0.721182	22.908746	2

477311 rows × 3 columns

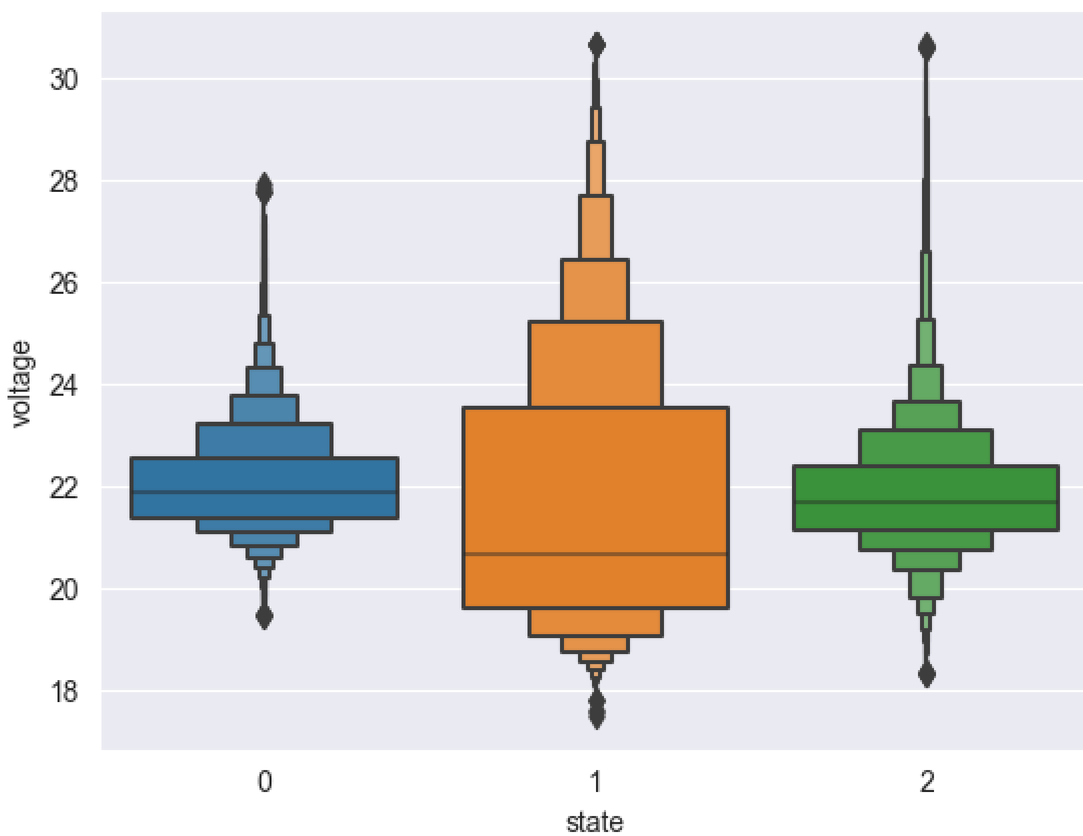
In [15]:
```
sns.boxenplot(data=label_df_clean, x='state', y='current')
```

Out[15]:
```
<AxesSubplot: xlabel='state', ylabel='current'>
```



In [16]:
```
sns.boxenplot(data=label_df_clean, x='state', y='voltage')
```

Out[16]: <AxesSubplot: xlabel='state', ylabel='voltage'>



we can make several observations based on the above plots

label 0 is the most grouped data

label 2 has similar mean to label 0

label 1 has the greatest std

relation between mean current and mean voltage per line

In [18]:
```
mean_feat_df = df.drop(columns=['state'])
mean_feat_df.columns = pd.MultiIndex.from_tuples([(features[c[0] == 't'], c)
mean_feat_df = mean_feat_df.groupby(level=0, axis=1).mean()
mean_feat_df.columns = [f"mean_{c}" for c in mean_feat_df.columns]
mean_feat_df['state'] = df['state']
mean_feat_df
```

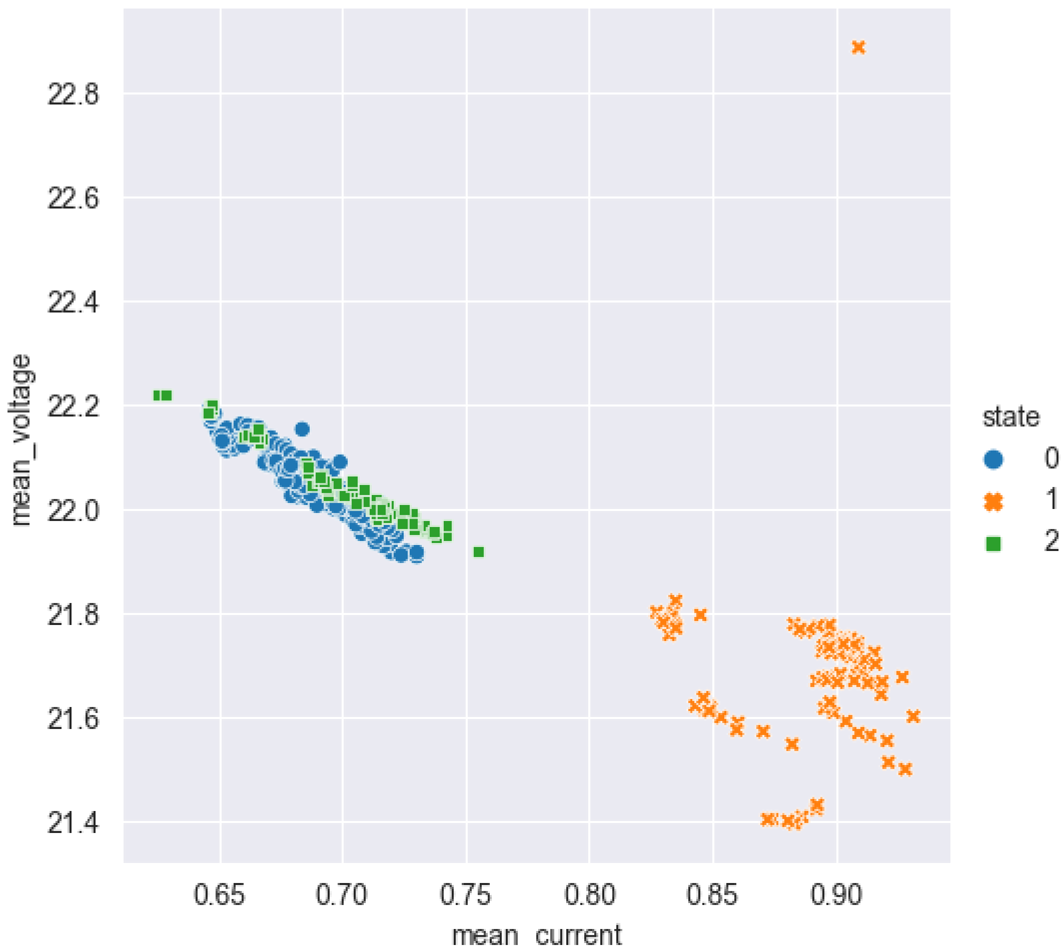
Out[18]:

	mean_current	mean_voltage	state
0	0.683705	22.151397	0
1	0.679833	22.066898	0
2	0.690525	22.070464	0
3	0.692794	22.012264	0
4	0.680013	22.076911	0
...	...	...	...
495	0.685177	22.086528	2
496	0.692129	22.053290	2
497	0.690922	22.061742	2
498	0.686268	22.079134	2
499	0.708508	22.038410	2

500 rows × 3 columns

In [19]:
```
sns.relplot(data=mean_feat_df, x='mean_current', y='mean_voltage', hue='stat
```

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x14a19b310>
```



we can easily see there is a clear distinction on defect type 1

both mean_voltage and mean_average are way off the normal

however there is no clear distinction from defect type 2 and normal

Random sample of each feature

In [20]:

```
label_0_df = df[df['state'] == labels[0]].drop(columns=['state']).sample(n=1
label_0_df.columns = pd.MultiIndex.from_tuples([(features[c[0] == 't'], c.sp
label_0_df = label_0_df.transpose()
label_0_df = label_0_df.melt(ignore_index=False, var_name='ori_line')
label_0_df = label_0_df.reset_index(names=['feature', 'ori_column'])
label_0_df
```

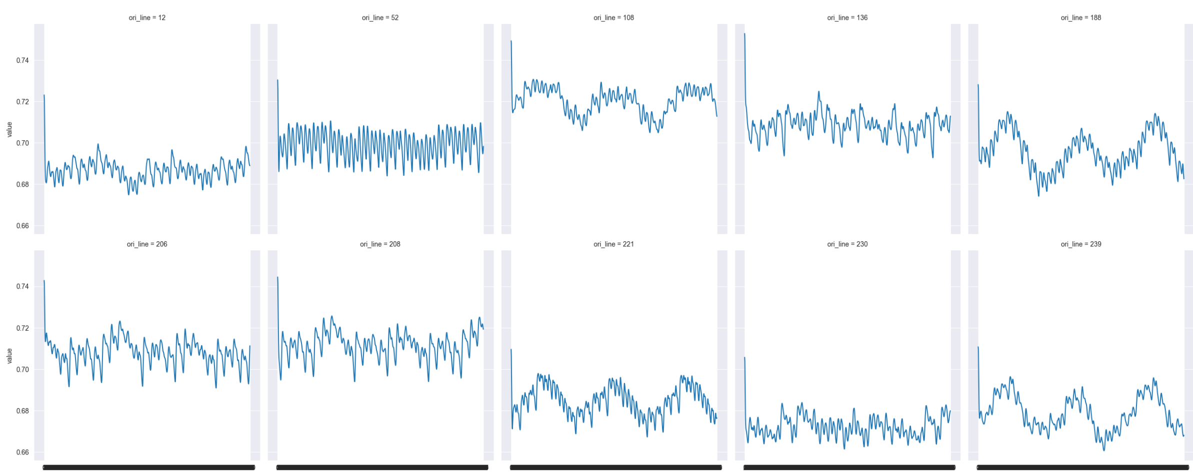
Out[20]:

	feature	ori_column	ori_line	value
0	current	0	208	0.744507
1	current	1	208	0.741720
2	current	2	208	0.731674
3	current	3	208	0.721409
4	current	4	208	0.713688
...	...	...	...	...
19995	voltage	995	108	21.750851
19996	voltage	996	108	21.798354
19997	voltage	997	108	21.935375
19998	voltage	998	108	22.162565
19999	voltage	999	108	22.403317

20000 rows × 4 columns

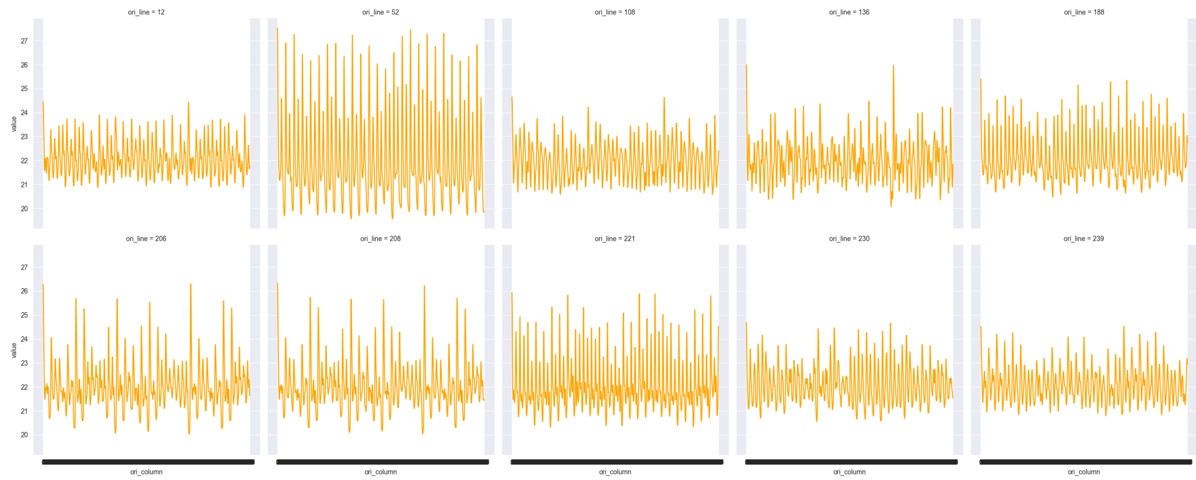
In [21]:
```
sns.relplot(data=label_0_df[label_0_df['feature'] == 'current'], x='ori_colu
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x14a102e50>



In [22]:
```
sns.relplot(data=label_0_df[label_0_df['feature'] == 'voltage'], x='ori_colu
```

Out[22]: <seaborn.axisgrid.FacetGrid at 0x14a342c40>



above we can see ten random samples, for each of them we plot the one thousand measurements made

they all belong to label 0 and there doesn't seem to be any pattern