Find and plot all the time series, histograms, violin plot, see distribution of data sets, cross-correlation of the data. Also, analyze other plots taught in EDA lecture

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from matplotlib import pyplot
```

In [8]:

```
electric_motor=pd.read_csv('electric_motor.csv')
electric_motor.head()
```

Out[8]:

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521

In [9]:

```
electric_motor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 998070 entries, 0 to 998069
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   ambient         998070 non-null  float64
 1   coolant         998070 non-null  float64
 2   u_d             998070 non-null  float64
 3   u_q             998070 non-null  float64
 4   motor_speed     998070 non-null  float64
 5   torque          998070 non-null  float64
 6   i_d             998070 non-null  float64
 7   i_q             998070 non-null  float64
 8   pm              998070 non-null  float64
 9   stator_yoke     998070 non-null  float64
 10  stator_tooth    998070 non-null  float64
 11  stator_winding  998070 non-null  float64
 12  profile_id      998070 non-null  int64  
dtypes: float64(12), int64(1)
memory usage: 99.0 MB
```

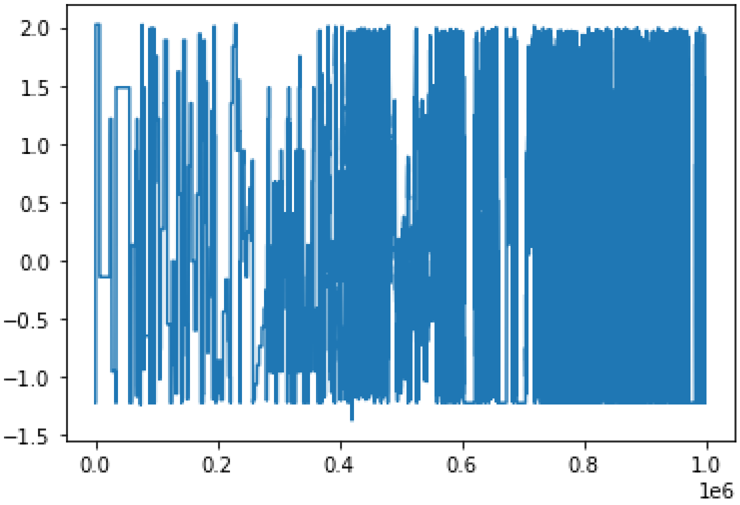
In [10]:
```
electric_motor.describe()
```

Out[10]:

	ambient	coolant	u_d	u_q	motor_speed	torq
count	998070.000000	998070.000000	998070.000000	998070.000000	998070.000000	998070.0000
mean	-0.003905	0.004723	0.004780	-0.005690	-0.006336	-0.0033
std	0.993127	1.002423	0.997878	1.002330	1.001229	0.9979
min	-8.573954	-1.429349	-1.655373	-1.861463	-1.371529	-3.3459
25%	-0.599385	-1.037925	-0.826359	-0.927390	-0.951892	-0.2669
50%	0.266157	-0.177187	0.267542	-0.099818	-0.140246	-0.1872
75%	0.686675	0.650709	0.358491	0.852625	0.853584	0.5471
max	2.967117	2.649032	2.274734	1.793498	2.024164	3.0169

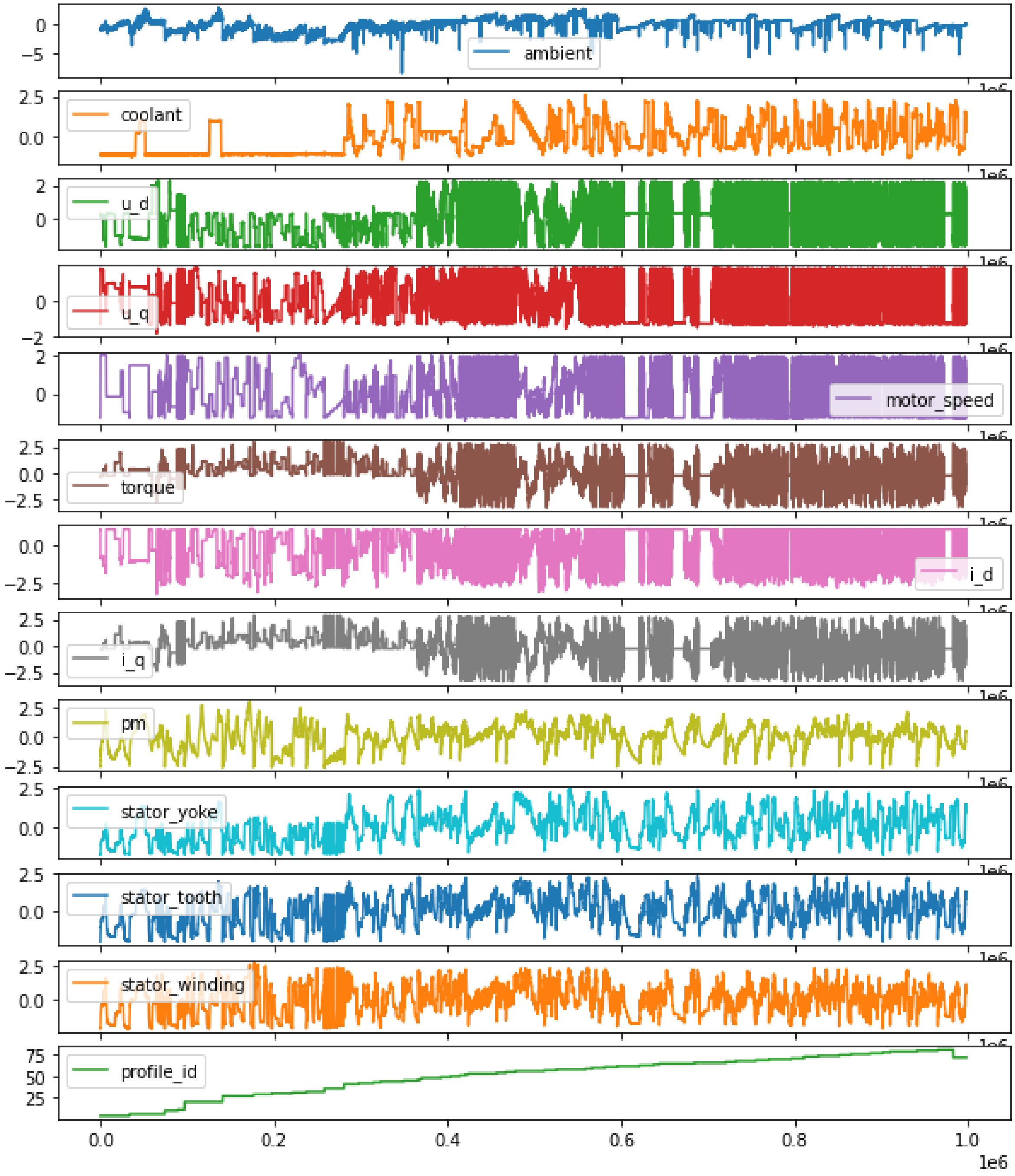
In [11]:
```
electric_motor['motor_speed'].plot()
```

Out[11]:
```
<AxesSubplot:>
```



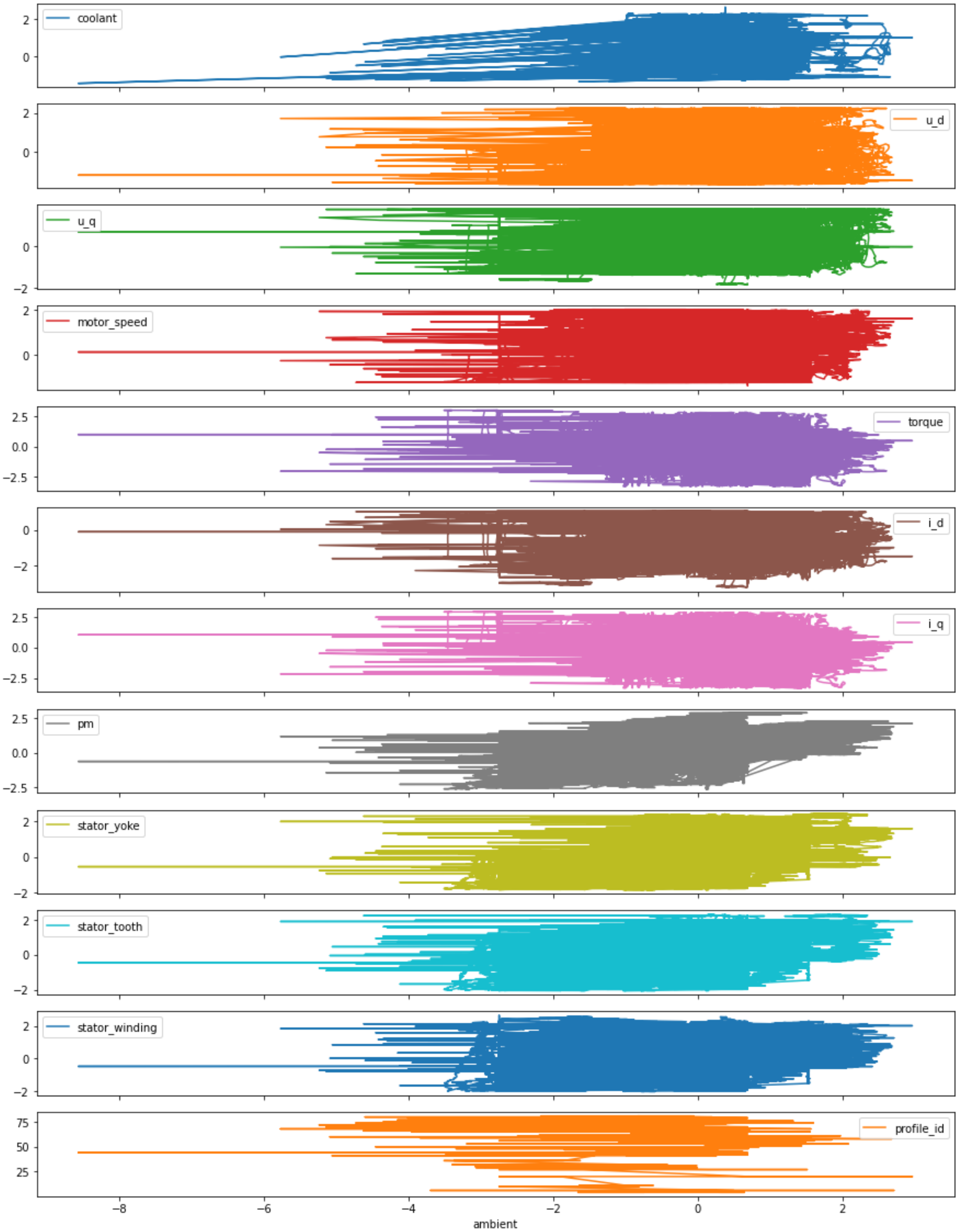
In [12]:
```
electric_motor.plot(subplots=True, figsize=(10, 12))
```

Out[12]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```



Time Series Plots

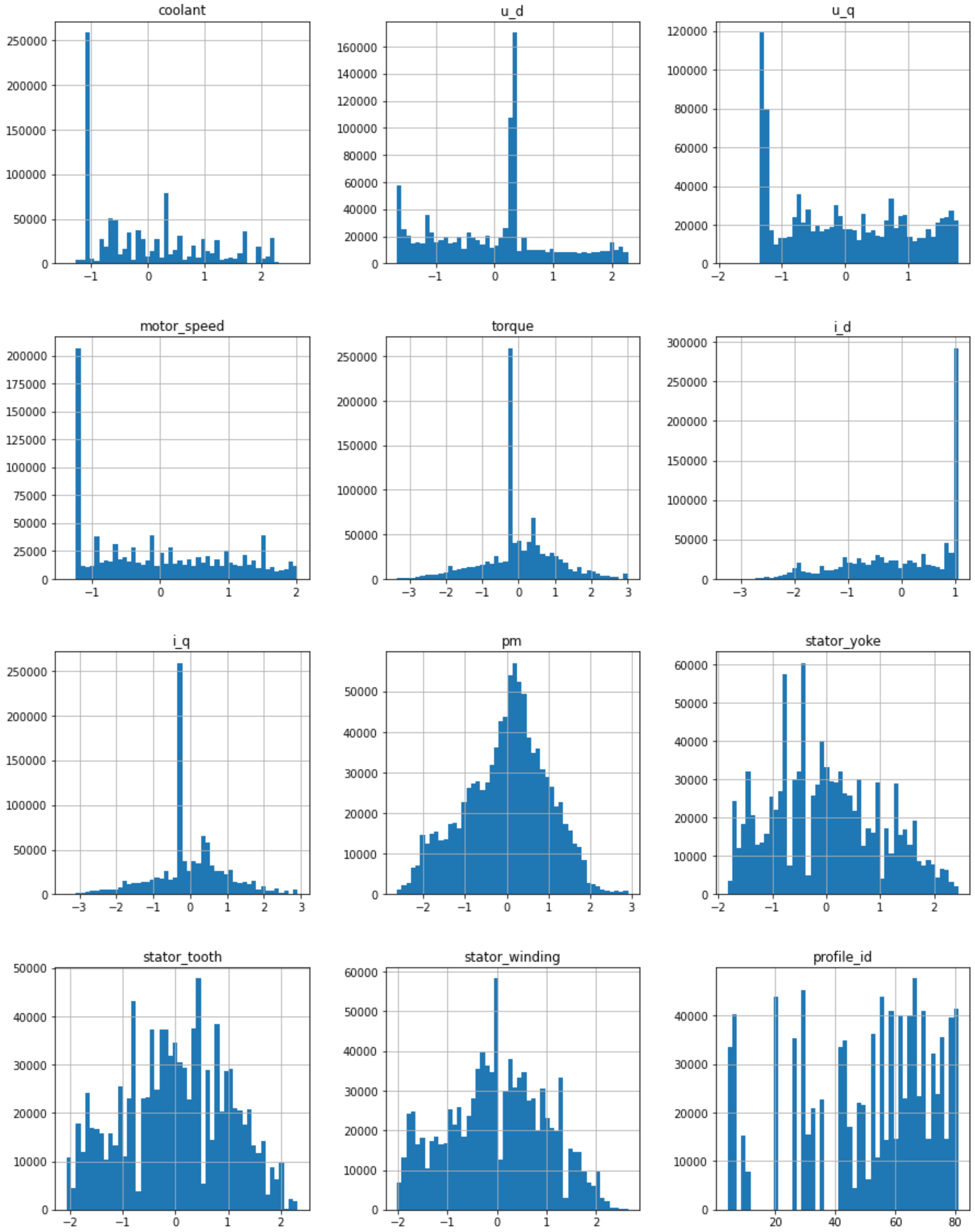
In [27]:
```
df.plot(subplots=True, figsize=(15,20))
plt.show()
```



histograms

In [26]:

```
df.hist(bins=50, figsize=(15,20))
plt.show()
```



violin plot

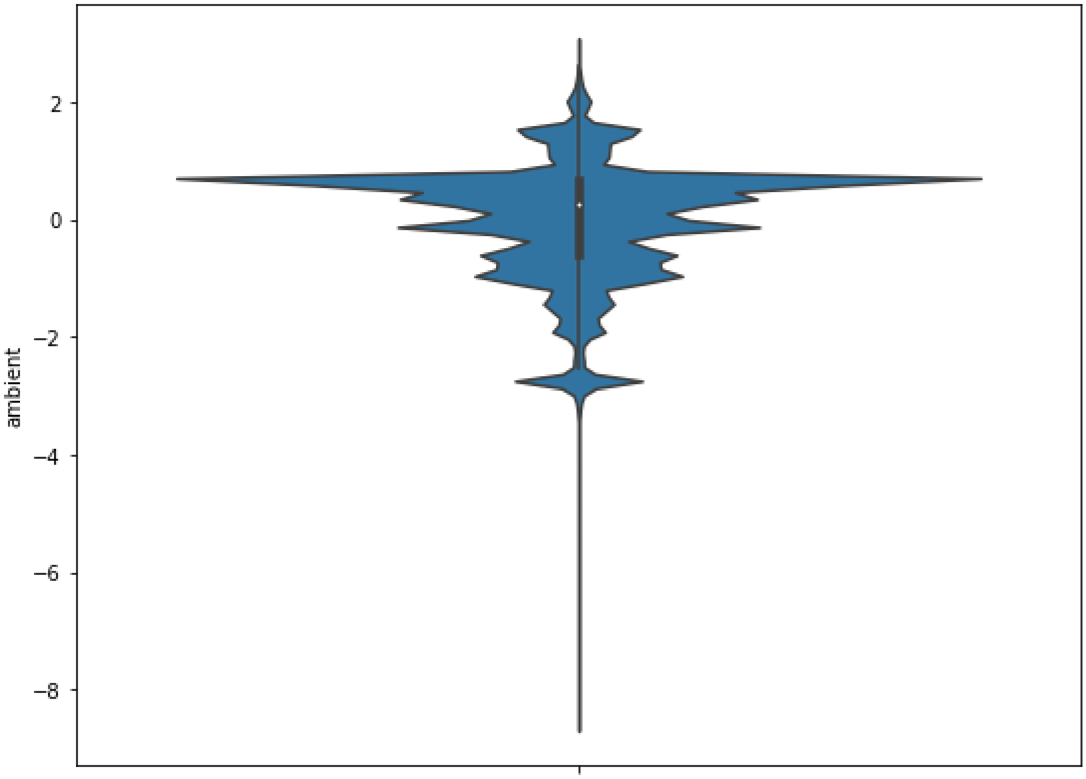
In [53]:

```
plt.figure(figsize=(16,12))
for i in range(len(electric_motor.columns)):
 plt.subplot(3, 4, i+1)
 sns.violinplot(electric_motor[electric_motor.columns[i]])
plt.title('violinplot for electric_motor')
plt.show()
```

```
C:\Users\91830\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
C:\Users\91830\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
C:\Users\91830\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
C:\Users\91830\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
```

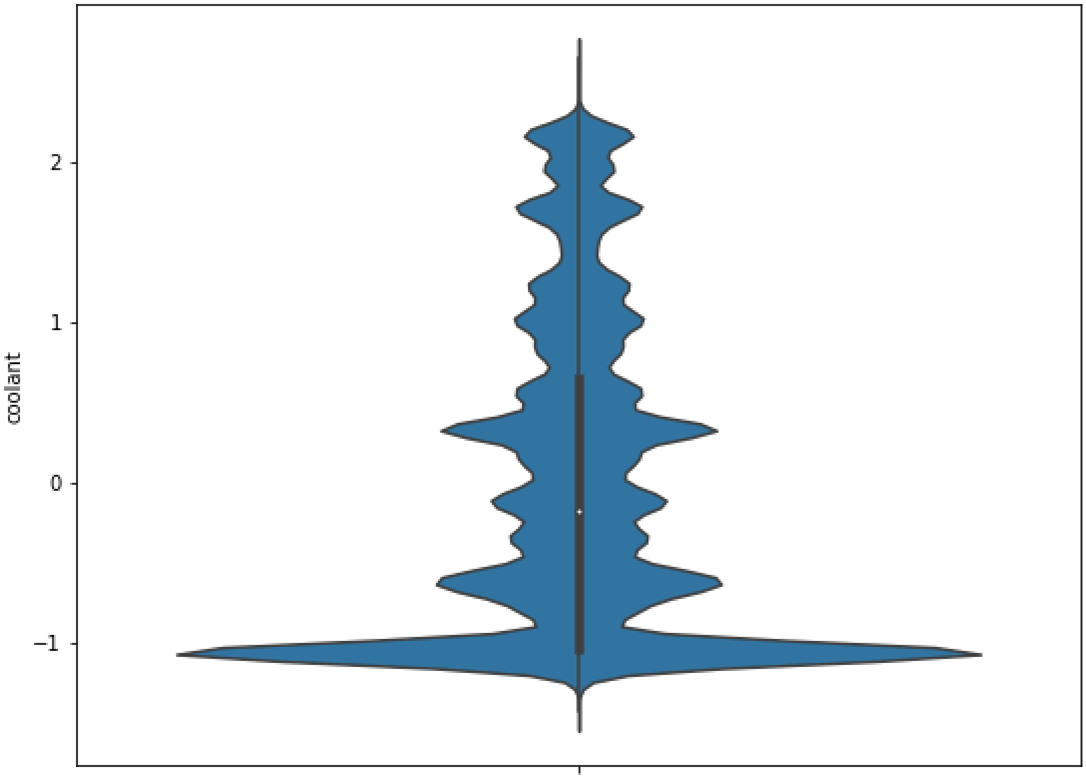
In [17]:
```
fig, ax = pyplot.subplots(figsize =(9, 7))
sns.violinplot( ax = ax, y = electric_motor["ambient"])
```

Out[17]:
```
<AxesSubplot:ylabel='ambient'>
```



In [18]:
```
fig, ax = pyplot.subplots(figsize =(9, 7))
sns.violinplot( ax = ax, y = electric_motor["coolant"])
```

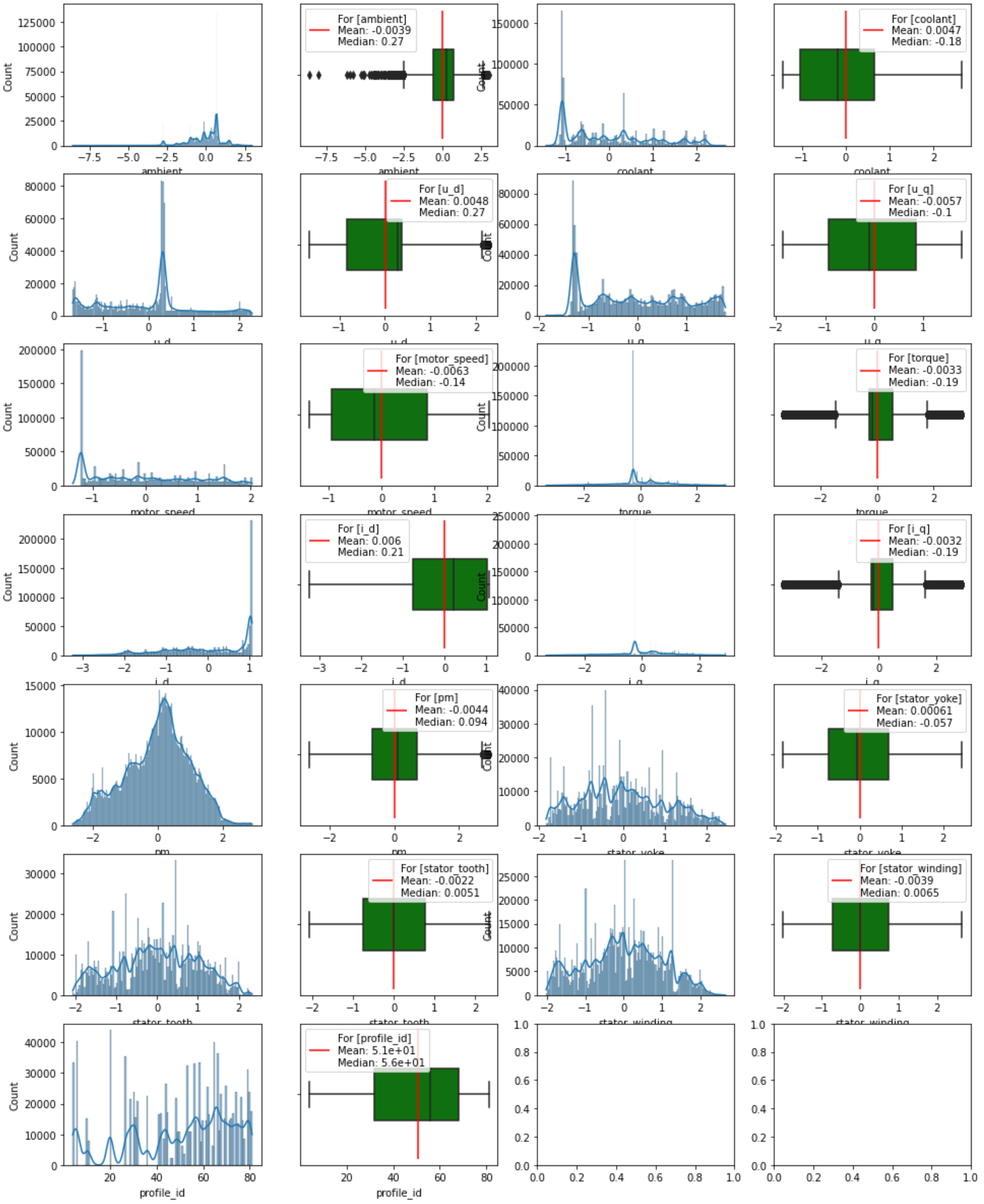
Out[18]:
```
<AxesSubplot:ylabel='coolant'>
```



Distribution of data sets

In [19]:

```
flierprops = dict(markerfacecolor='g', color='g', alpha=0.5)
n_cols = 4
n_rows = int(np.ceil(electric_motor.shape[-1]*2 / n_cols))
fig, axes = plt.subplots(n_rows, n_cols, figsize=(4 * n_cols, 3 * n_rows))
for i, (col) in enumerate(list(electric_motor.columns)):
 mean = electric_motor[col].mean()
 median = electric_motor[col].median()
 sns.histplot(electric_motor[col], ax=axes.flatten()[2*i], kde=True)
 sns.boxplot(x=electric_motor[col], orient='h', ax=axes.flatten()[2*i+1], color='g')
 axes.flatten()[2*i+1].vlines(mean, ymin = -1, ymax = 1, color='r', label=f"For [{col}]\nMean: {mean:.2}\nMedian: {median:.2}")
 axes.flatten()[2*i+1].legend()
```


Count
ambient
For [ambient]
Mean: -0.0039
Median: 0.27
ambient
Count
coolant
For [coolant]
Mean: 0.0047
Median: -0.18
coolant
Count
u_d
For [u_d]
Mean: 0.0048
Median: 0.27
u_d
Count
u_q
For [u_q]
Mean: -0.0057
Median: -0.1
u_q
Count
motor_speed
For [motor_speed]
Mean: -0.0063
Median: -0.14
motor_speed
Count
torque
For [torque]
Mean: -0.0033
Median: -0.19
torque
Count
i_d
For [i_d]
Mean: 0.006
Median: 0.21
i_d
Count
i_q
For [i_q]
Mean: -0.0032
Median: -0.19
i_q
Count
pm
For [pm]
Mean: -0.0044
Median: 0.094
pm
Count
stator_yoke
For [stator_yoke]
Mean: 0.00061
Median: -0.057
stator_yoke
Count
stator_tooth
For [stator_tooth]
Mean: -0.0022
Median: 0.0051
stator_tooth
Count
stator_winding
For [stator_winding]
Mean: -0.0039
Median: 0.0065
stator_winding
Count
profile_id
For [profile_id]
Mean: 5.1e+01
Median: 5.6e+01
profile_id

Cross-correlation of the data

In [37]:

```
import pandas as pd
from pandas import DataFrame
df = pd.read_csv('electric_motor.csv', index_col = 'ambient',
parse_dates=True)
print(df.describe())
```

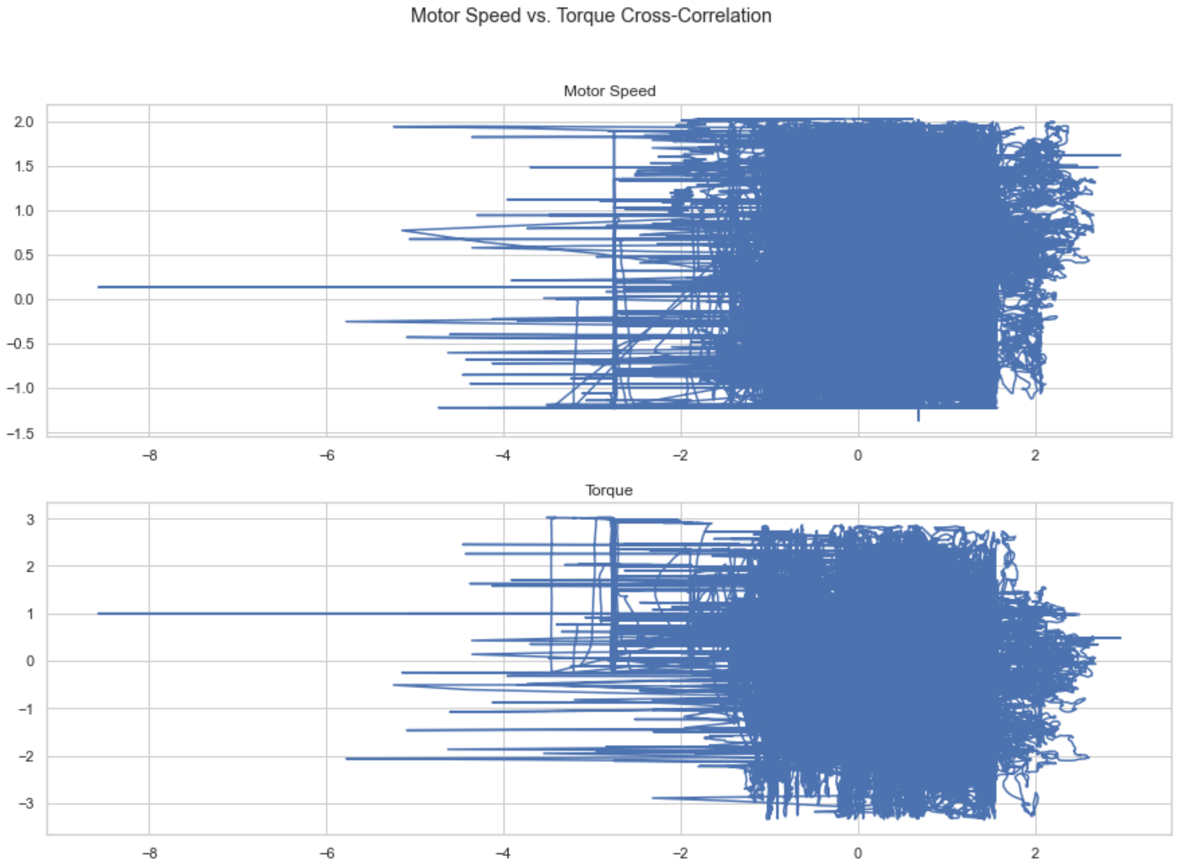
```
             coolant            u_d            u_q    motor_speed  \
count  998070.000000  998070.000000  998070.000000  998070.000000   
mean        0.004723       0.004780      -0.005690      -0.006336   
std         1.002423       0.997878       1.002330       1.001229   
min        -1.429349      -1.655373      -1.861463      -1.371529   
25%        -1.037925      -0.826359      -0.927390      -0.951892   
50%        -0.177187       0.267542      -0.099818      -0.140246   
75%         0.650709       0.358491       0.852625       0.853584   
max         2.649032       2.274734       1.793498       2.024164   

              torque            i_d            i_q             pm  \
count  998070.000000  998070.000000  998070.000000  998070.000000   
mean       -0.003333       0.006043      -0.003194      -0.004396   
std         0.997907       0.998994       0.997912       0.995686   
min        -3.345953      -3.245874      -3.341639      -2.631991   
25%        -0.266917      -0.756296      -0.257269      -0.672308   
50%        -0.187246       0.213935      -0.190076       0.094367   
75%         0.547171       1.013975       0.499260       0.680691   
max         3.016971       1.060937       2.914185       2.917456   

         stator_yoke   stator_tooth  stator_winding     profile_id  
count  998070.000000  998070.000000   998070.000000  998070.000000  
mean        0.000609      -0.002208       -0.003935      50.732001  
std         1.001049       0.999597        0.998343      22.073125  
min        -1.834688      -2.066143       -2.019973       4.000000  
25%        -0.747265      -0.761951       -0.725622      32.000000  
50%        -0.057226       0.005085        0.006536      56.000000  
75%         0.697344       0.772239        0.725660      68.000000  
max         2.449158       2.326668        2.653781      81.000000  
```

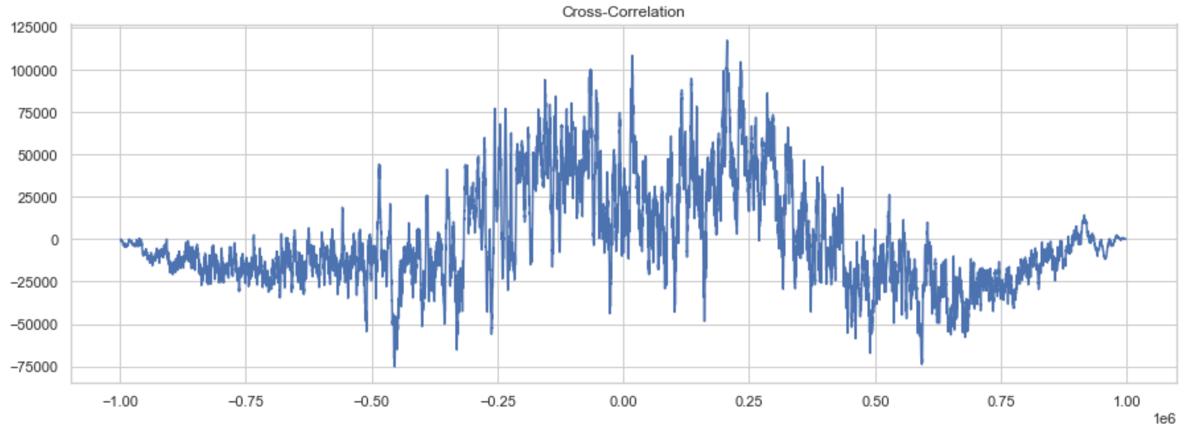
In [32]:
```
print(df.corr())
```

```
                 coolant       u_d       u_q  motor_speed    torque       i_d
\
coolant         1.000000  0.178708  0.027851    -0.033332 -0.189617  0.108489
u_d             0.178708  1.000000 -0.027472    -0.233828 -0.821325  0.358517
u_q             0.027851 -0.027472  1.000000     0.716897 -0.037262 -0.181855
motor_speed    -0.033332 -0.233828  0.716897     1.000000  0.024517 -0.722914
torque         -0.189617 -0.821325 -0.037262     0.024517  1.000000 -0.239059
i_d             0.108489  0.358517 -0.181855    -0.722914 -0.239059  1.000000
i_q            -0.186060 -0.796586 -0.026348     0.006323  0.996560 -0.204230
pm              0.430548 -0.082564  0.101236     0.332419 -0.072905 -0.299227
stator_yoke     0.874037  0.041428  0.106186     0.182564 -0.092207 -0.179903
stator_tooth    0.689003 -0.066256  0.149017     0.333909 -0.011055 -0.387712
stator_winding  0.509298 -0.150714  0.125445     0.393154  0.080981 -0.539924
profile_id      0.499669  0.300771 -0.122260    -0.166225 -0.257446  0.142368

                     i_q        pm  stator_yoke  stator_tooth  stator_winding
\
coolant        -0.186060  0.430548     0.874037      0.689003        0.509298
u_d            -0.796586 -0.082564     0.041428     -0.066256       -0.150714
u_q            -0.026348  0.101236     0.106186      0.149017        0.125445
motor_speed     0.006323  0.332419     0.182564      0.333909        0.393154
torque          0.996560 -0.072905    -0.092207     -0.011055        0.080981
i_d            -0.204230 -0.299227    -0.179903     -0.387712       -0.539924
i_q             1.000000 -0.086486    -0.098753     -0.025191        0.060949
pm             -0.086486  1.000000     0.695014      0.768352        0.729561
stator_yoke    -0.098753  0.695014     1.000000      0.949898        0.845144
stator_tooth   -0.025191  0.768352     0.949898      1.000000        0.965633
stator_winding  0.060949  0.729561     0.845144      0.965633        1.000000
profile_id     -0.256127  0.156735     0.398021      0.280994        0.181908

                profile_id
coolant           0.499669
u_d               0.300771
u_q              -0.122260
motor_speed      -0.166225
torque           -0.257446
i_d               0.142368
i_q              -0.256127
pm                0.156735
stator_yoke       0.398021
stator_tooth      0.280994
stator_winding    0.181908
profile_id        1.000000
```

Cross Correlation

In [43]:
```
from scipy.signal import correlate

fig, axs = plt.subplots(2, figsize=(15, 10))

axs[0].plot(df['motor_speed'])
axs[0].set_title('Motor Speed')

axs[1].plot(df['torque'])
axs[1].set_title('Torque')

fig.suptitle('Motor Speed vs. Torque Cross-Correlation')

corr = correlate(df['motor_speed'], df['torque'])
lags = np.arange(-len(df)+1, len(df))

fig, axs = plt.subplots(1, figsize=(15, 5))

axs.plot(lags, corr)
axs.set_title('Cross-Correlation')

plt.show()
```

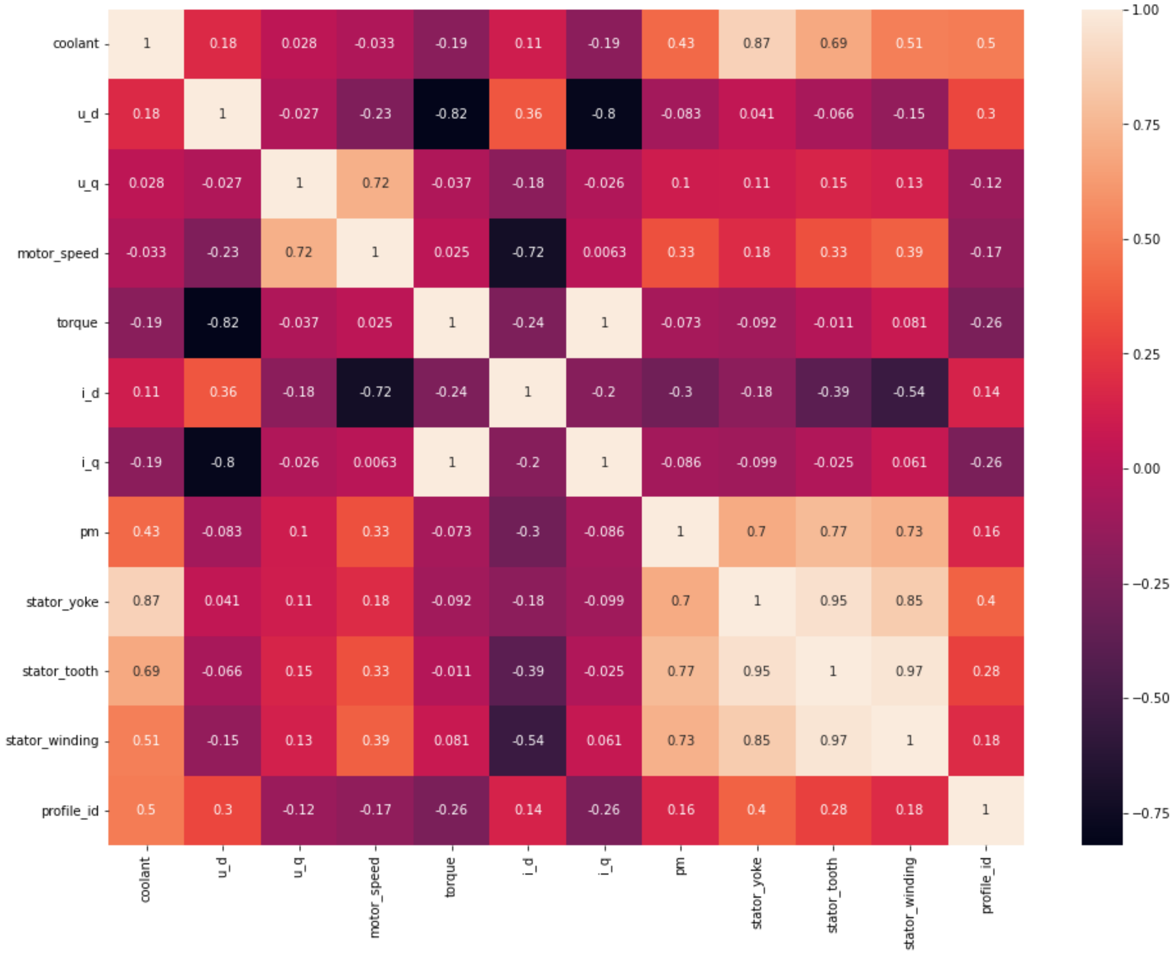




Heatmap

```
In [22]: plt.figure(figsize=(16,12))
         sns.heatmap(df.corr(),annot=True)
```

```
Out[22]: <AxesSubplot:>
```



In [52]:
```
electric_motor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 998070 entries, 0 to 998069
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   ambient         998070 non-null  float64
 1   coolant         998070 non-null  float64
 2   u_d             998070 non-null  float64
 3   u_q             998070 non-null  float64
 4   motor_speed     998070 non-null  float64
 5   torque          998070 non-null  float64
 6   i_d             998070 non-null  float64
 7   i_q             998070 non-null  float64
 8   pm              998070 non-null  float64
 9   stator_yoke     998070 non-null  float64
 10  stator_tooth    998070 non-null  float64
 11  stator_winding  998070 non-null  float64
 12  profile_id      998070 non-null  int64  
dtypes: float64(12), int64(1)
memory usage: 99.0 MB
```

In [ ]: