ML TEST

February 2, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df=pd.read_table('https://archive.ics.uci.edu/ml/machine-learning-databases/
↪00291/airfoil_self_noise.dat',header=None,names=['frequency','attack␣
↪angle','chord length','free stream velocity','thickness','pressure level'])
```

[3]:
```
df.head()
```

[3]:

	frequency	attack angle	chord length	free stream velocity	thickness	pressure level
0	800	0.0	0.3048	71.3	0.002663	126.201
1	1000	0.0	0.3048	71.3	0.002663	125.201
2	1250	0.0	0.3048	71.3	0.002663	125.951
3	1600	0.0	0.3048	71.3	0.002663	127.591
4	2000	0.0	0.3048	71.3	0.002663	127.461

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1503 entries, 0 to 1502
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   frequency             1503 non-null   int64
 1   attack angle          1503 non-null   float64
 2   chord length          1503 non-null   float64
 3   free stream velocity  1503 non-null   float64
```

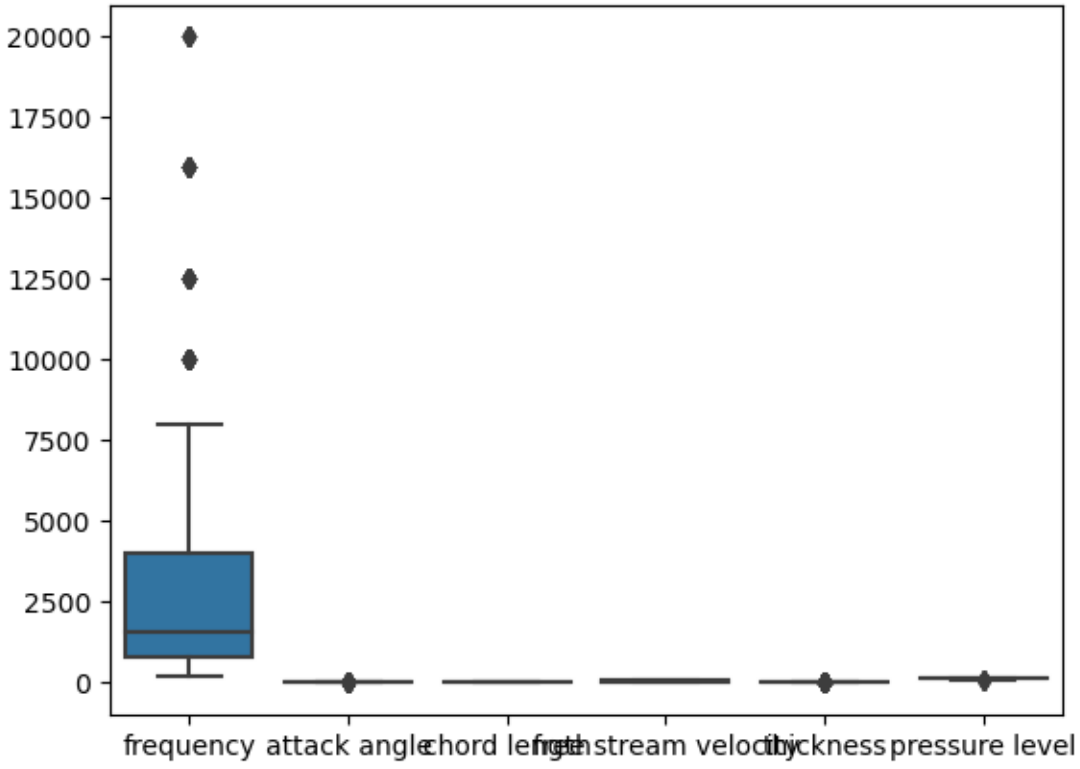
```
 4   thickness             1503 non-null   float64
 5   pressure level        1503 non-null   float64
dtypes: float64(5), int64(1)
memory usage: 70.6 KB
```

[5]:
```
df.isna().sum()
```

[5]:
```
frequency              0
attack angle           0
chord length           0
free stream velocity   0
thickness              0
pressure level         0
dtype: int64
```

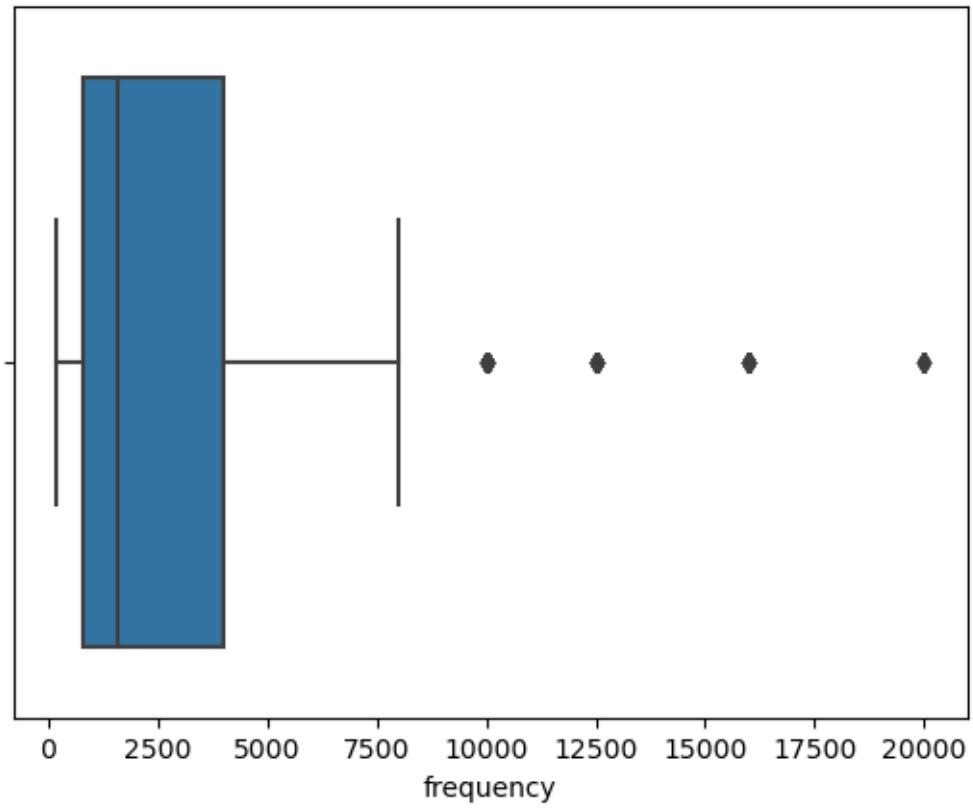
[6]:
```
sns.boxplot(data=df)
```

[6]:
```
<AxesSubplot:>
```



[7]:
```
sns.boxplot(x='frequency',data=df)
```

[7]:
```
<AxesSubplot:xlabel='frequency'>
```



[8]:
```
from scipy.stats import skew
```

[9]:
```
colname=df.select_dtypes(['int64','float64']).columns
```

[10]:
```
colname
```

[10]:
```
Index(['frequency', 'attack angle', 'chord length', 'free stream velocity',
       'thickness', 'pressure level'],
      dtype='object')
```

[11]:
```
df[colname]
```

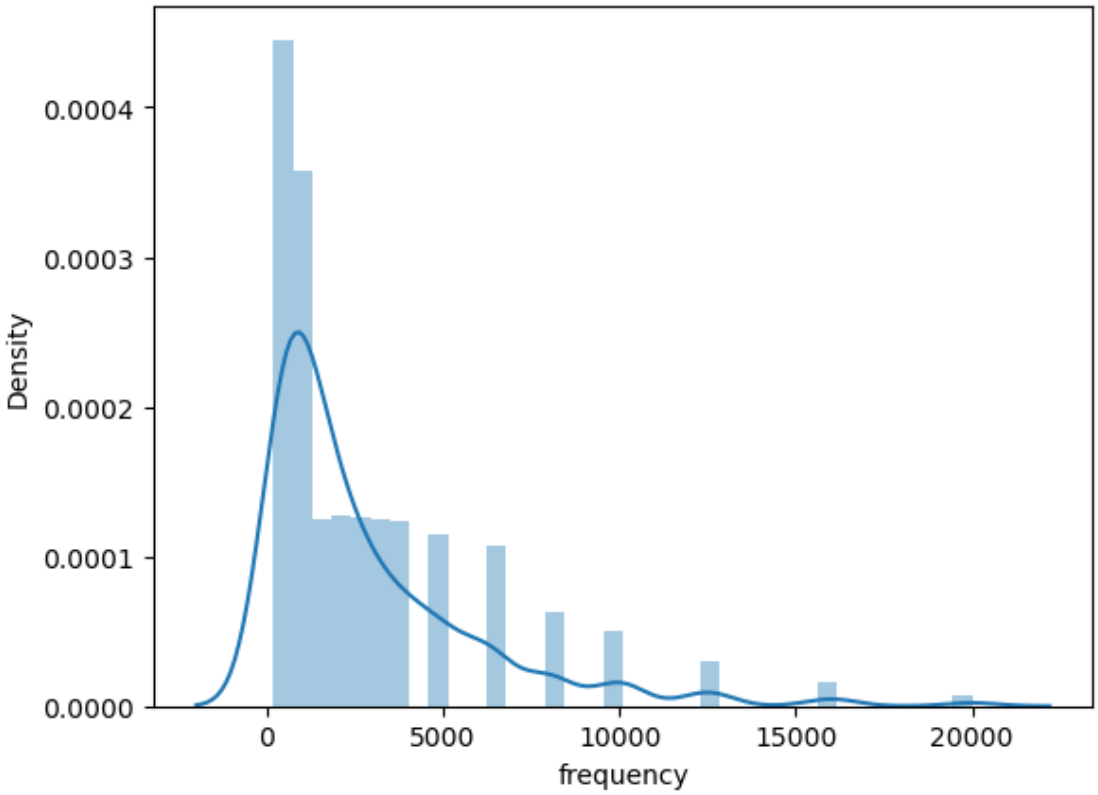
[11]:

	frequency	attack angle	chord length	free stream velocity	thickness	\
0	800	0.0	0.3048	71.3	0.002663	
1	1000	0.0	0.3048	71.3	0.002663	
2	1250	0.0	0.3048	71.3	0.002663	
3	1600	0.0	0.3048	71.3	0.002663	
4	2000	0.0	0.3048	71.3	0.002663	
…	…	…	…	…	…	
1498	2500	15.6	0.1016	39.6	0.052849	
1499	3150	15.6	0.1016	39.6	0.052849	

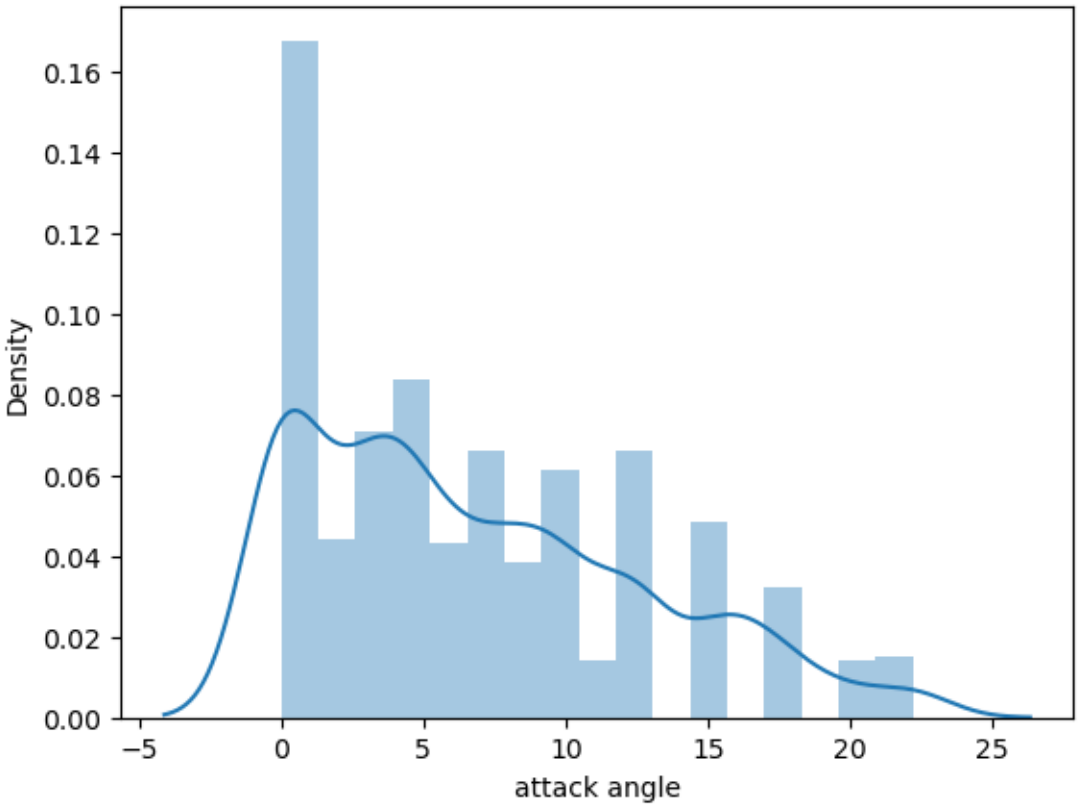
```
1500       4000          15.6        0.1016                 39.6   0.052849
1501       5000          15.6        0.1016                 39.6   0.052849
1502       6300          15.6        0.1016                 39.6   0.052849

      pressure level
0             126.201
1             125.201
2             125.951
3             127.591
4             127.461
…               …
1498          110.264
1499          109.254
1500          106.604
1501          106.224
1502          104.204

[1503 rows x 6 columns]
```

[12]:
```
for i in df[colname]:
    print(i)
    print(skew(df[i]))
    sns.distplot(df[i])
    plt.show()
```

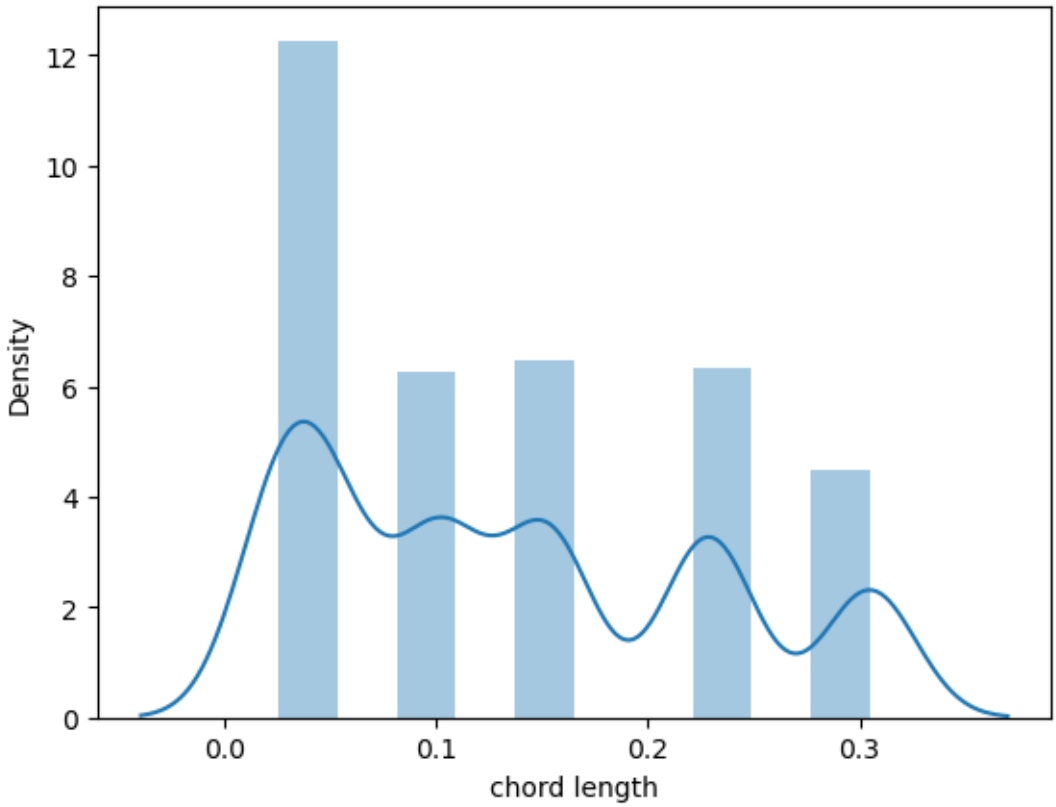
```
frequency
2.1349509268138207
```



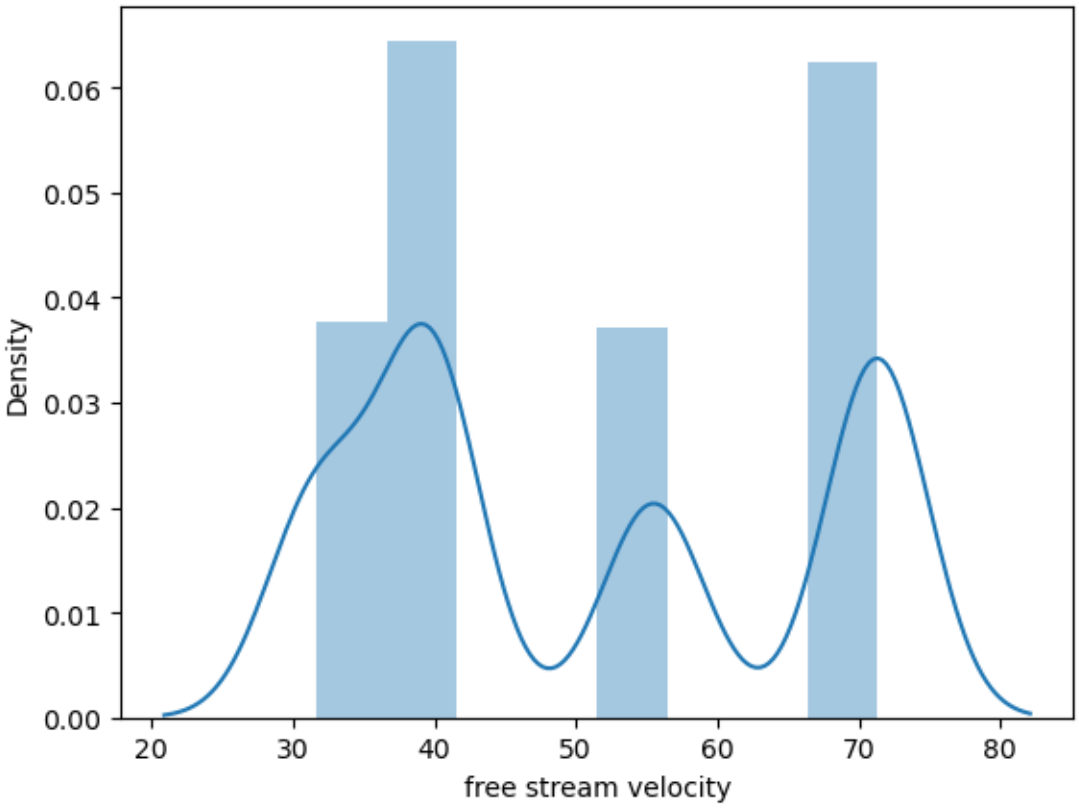

```
attack angle
0.6884764219408198
```



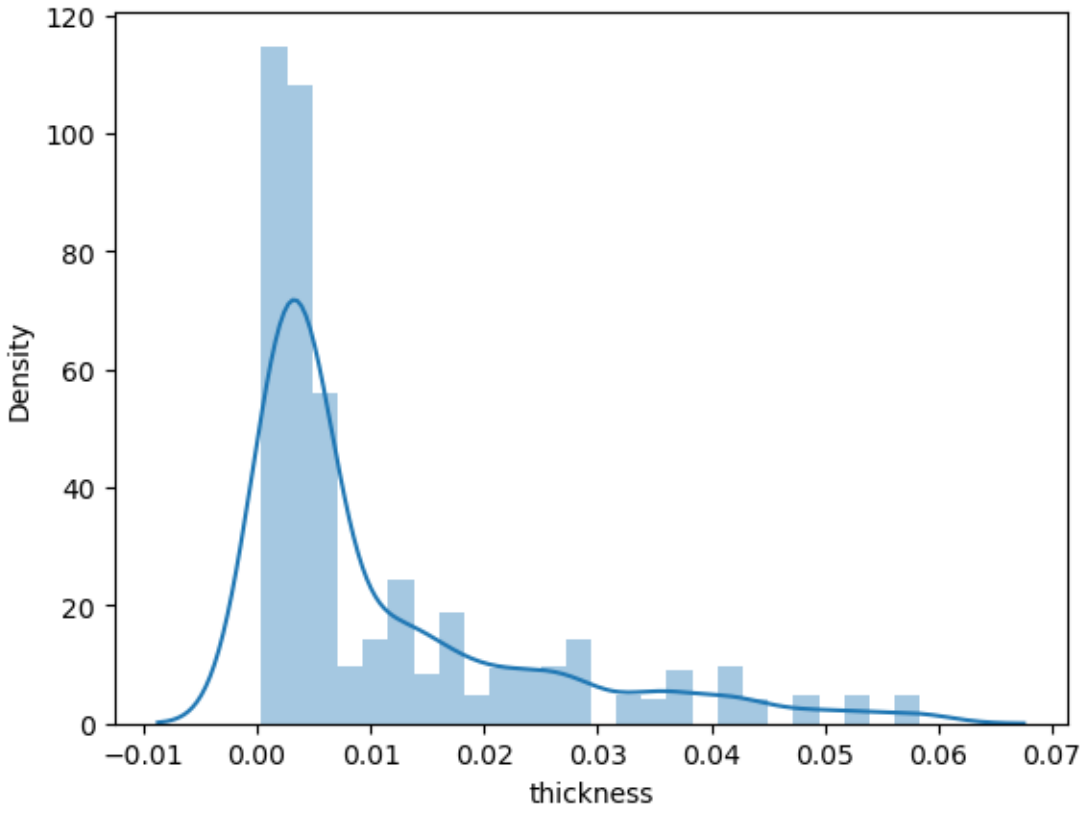

```
chord length
0.45700080866491105
```



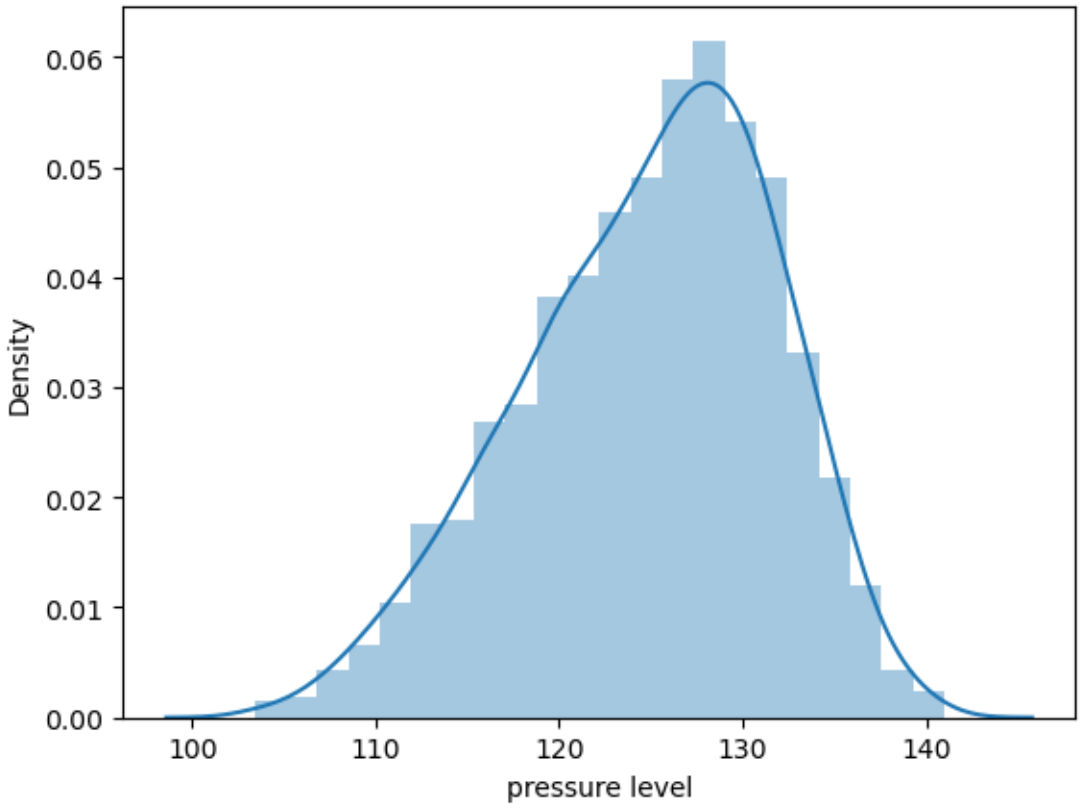

```
free stream velocity
0.2356169672566666
```




```
thickness
1.7004653179096092
```




```
pressure level
-0.4185339558133514
```



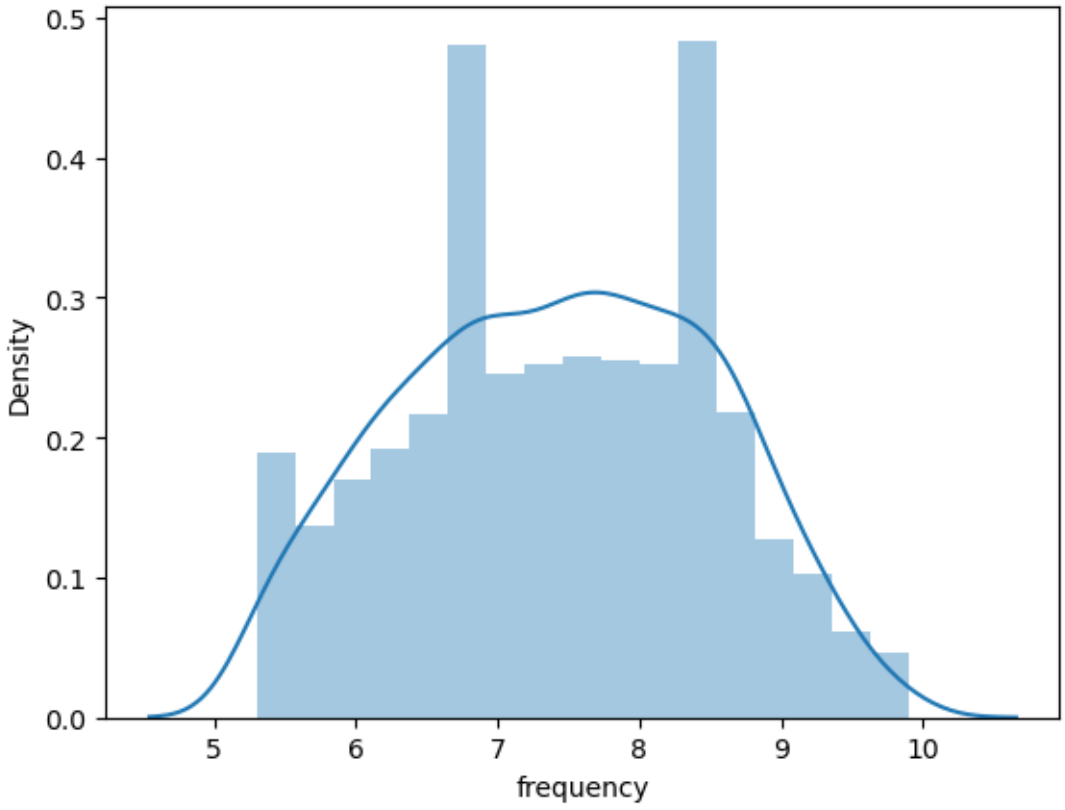
[13]:
```
df.corr().style.background_gradient()
```

[13]:
```
<pandas.io.formats.style.Styler at 0x133a427c9d0>
```

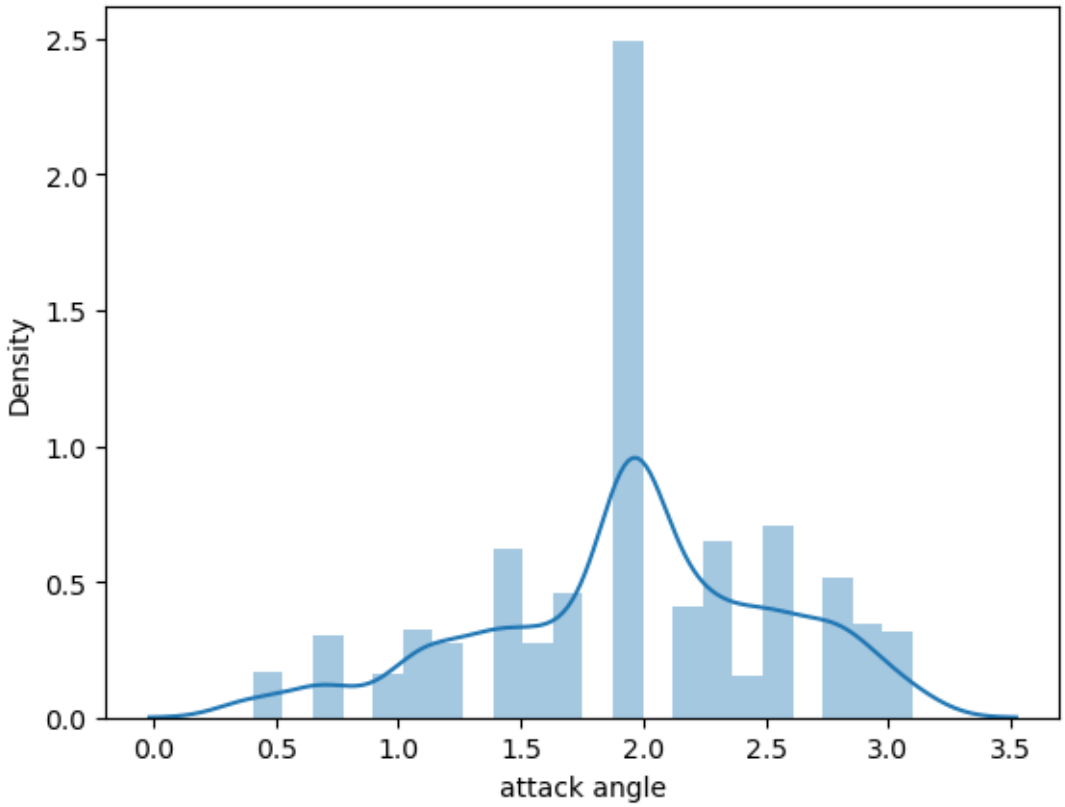
[14]:
```
df[['frequency', 'attack angle', 'chord length', 'free stream velocity',
       'thickness']]=np.log(df[['frequency', 'attack angle', 'chord length',␣
 ↪'free stream velocity',
       'thickness']])
```

[21]:
```
for i in df[colname]:
    print(i)
    print(skew(df[i]))
    sns.distplot(df[i])
    plt.show()
```

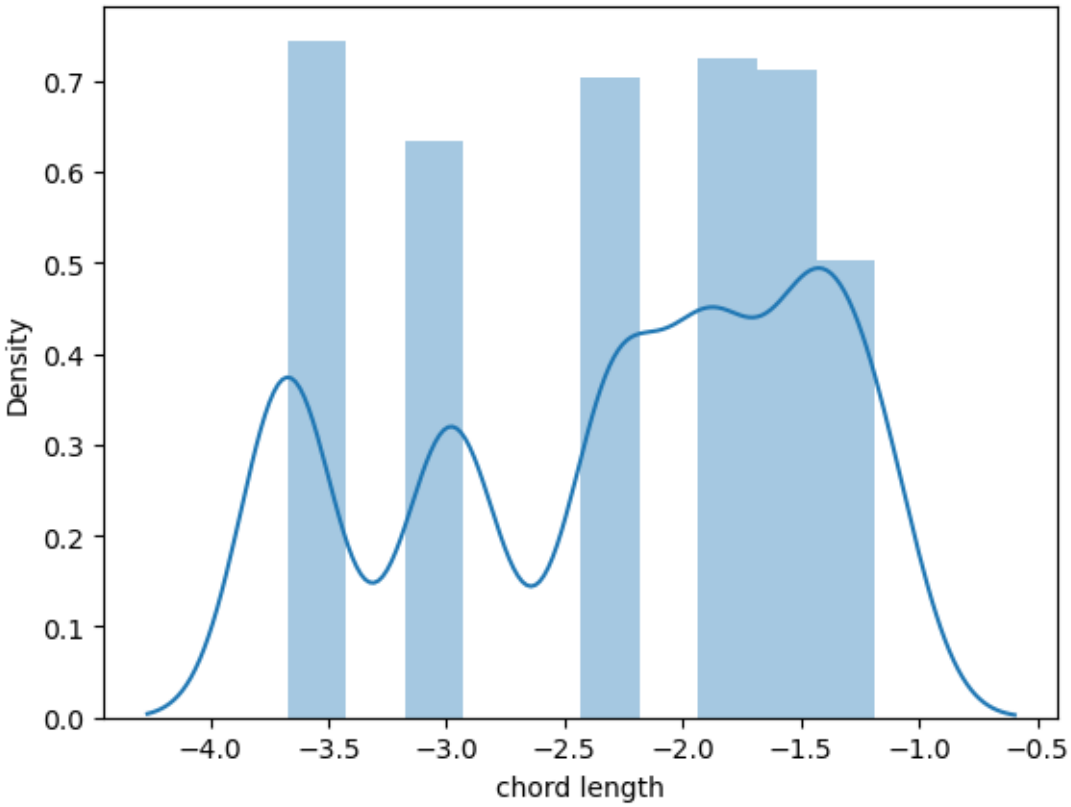
```
frequency
-0.03175638453081654
```



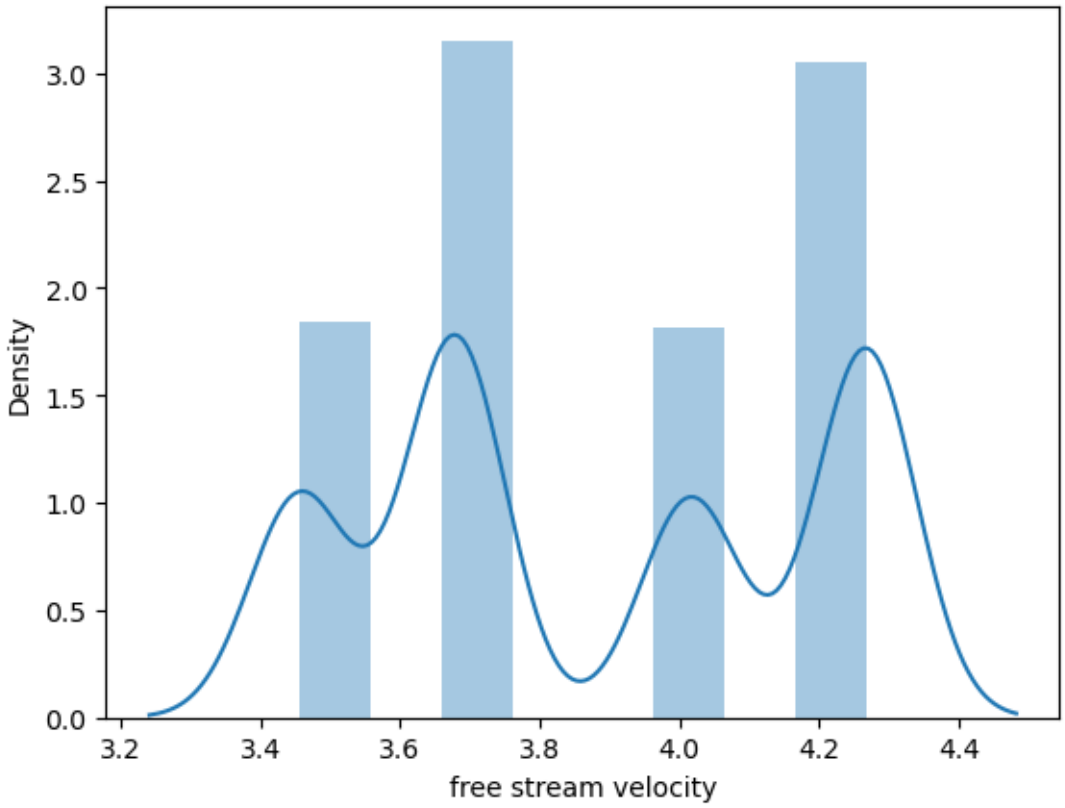

```
attack angle
-0.38378063686531994
```



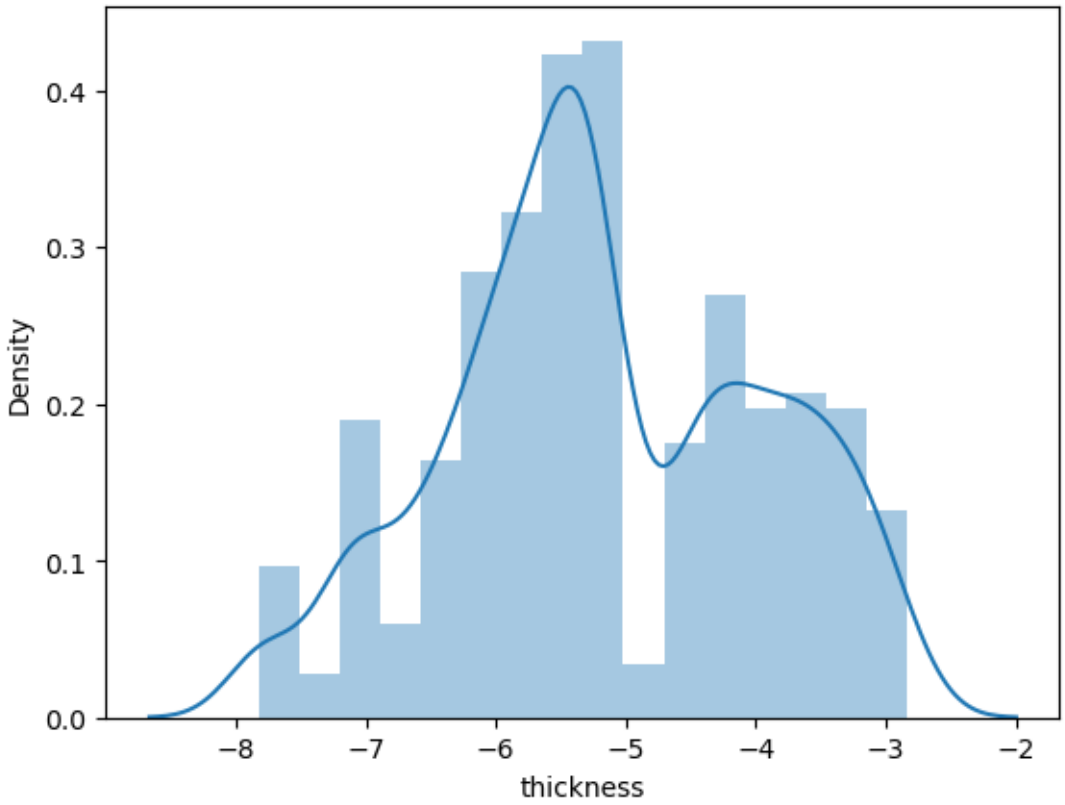

```
chord length
-0.3984640338055838
```



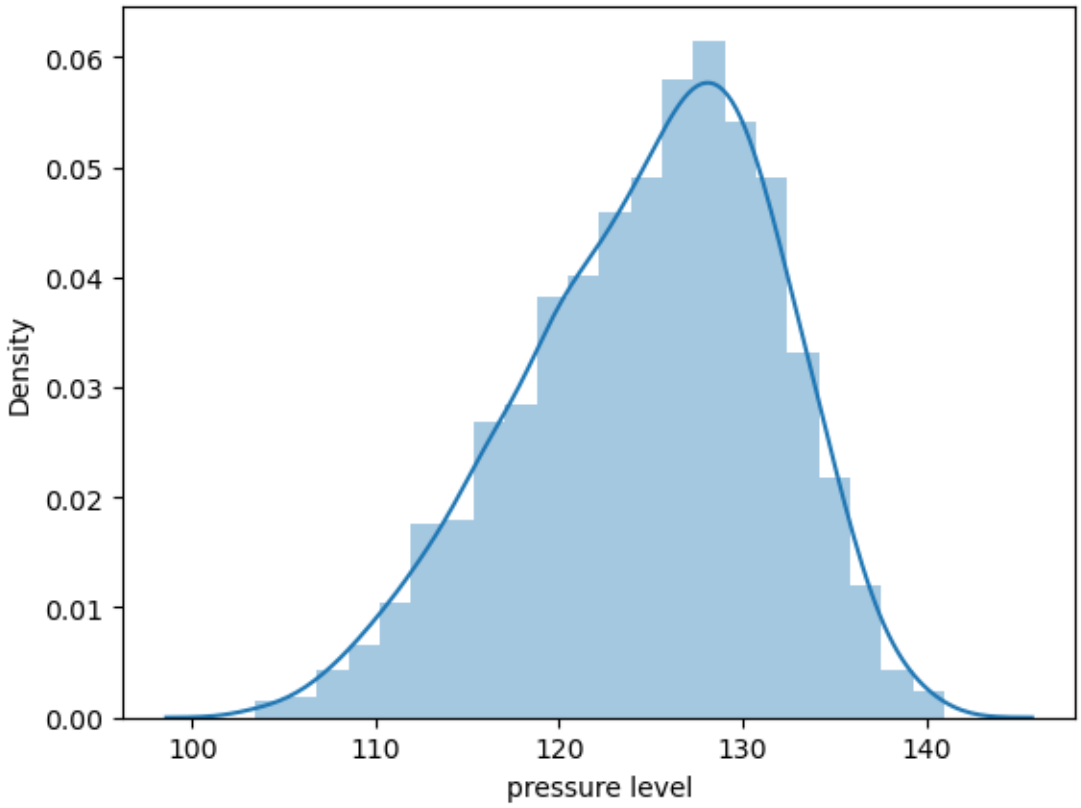

```
free stream velocity
0.039157975558672495
```




```
thickness
-0.007610865788289084
```




```
pressure level
-0.4185339558133514
```



[16]:
```
df['attack angle'].unique()
```

[16]:
```
array([      -inf, 0.40546511, 1.09861229, 1.38629436, 0.69314718,
       1.66770682, 1.98787435, 0.99325177, 1.68639895, 1.97408103,
       2.29253476, 2.53369681, 1.43508453, 2.12823171, 2.41591378,
       2.73436751, 2.98061864, 1.56861592, 2.2512918 , 2.54160199,
       2.85647021, 3.10009229, 1.19392247, 1.90210753, 2.18605128,
       2.50959926, 2.74727091])
```

[17]:
```
df.replace([np.inf,-np.inf],np.nan,inplace=True)
```

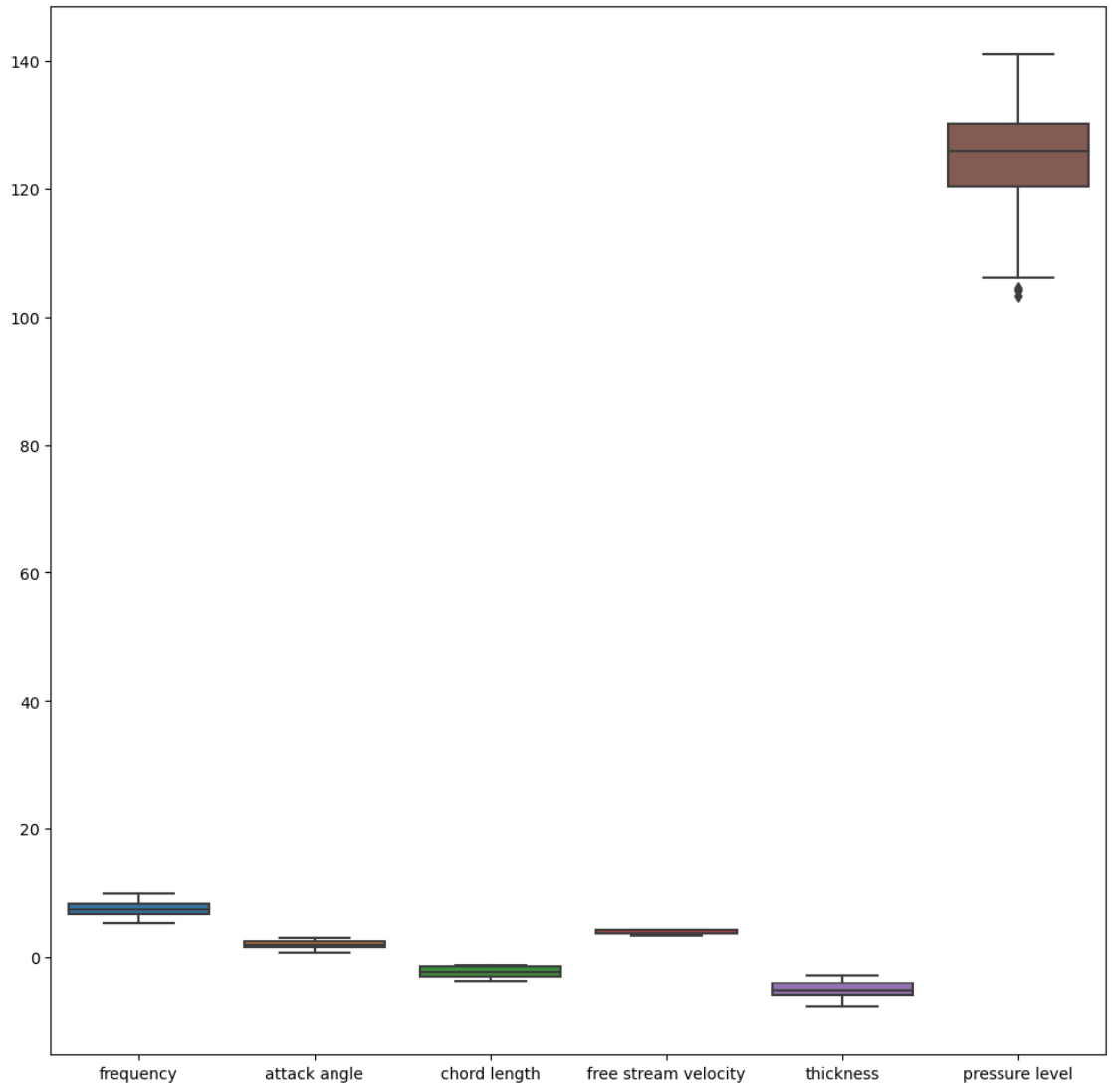
[18]:
```
from sklearn.impute import SimpleImputer
```

[19]:
```
si=SimpleImputer(missing_values=np.nan,strategy='mean')
```

[20]:
```
df[['attack angle']]=si.fit_transform(df[['attack angle']])
```

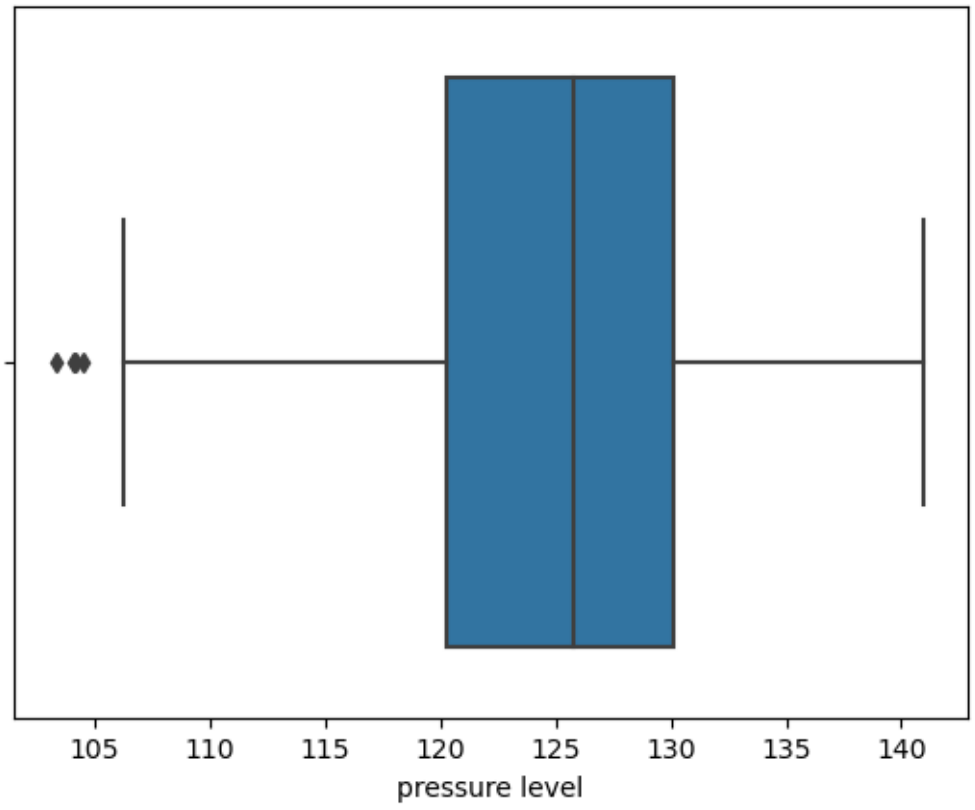
[92]:
```
plt.figure(figsize=(12,12))
sns.boxplot(data=df)
```

[92]:
```
<AxesSubplot:>
```



[74]:
```
sns.boxplot(x='pressure level',data=df)
```

[74]:
```
<AxesSubplot:xlabel='pressure level'>
```



[75]:
```
def outlier(data):
    outlier=[]
    q1,q3=np.percentile(data,[25,75])
    iqr=q3-q1
    lw=q1-1.5*iqr
    uw=q3+1.5*iqr
    for i in data:
        if i<lw or i>uw:
            outlier.append(i)
    return outlier
```

[76]:
```
outlier(df['pressure level'])
```

[76]:
```
[104.5, 104.13, 103.38, 104.204]
```

[ ]:
```

```

[77]:
```
np.where(df['pressure level']==104.204)
```

[77]:
```
(array([1463], dtype=int64),)
```

[79]:
```
#df.drop([687,688,689,1467],axis=0,inplace=True)
```

[ ]:
```
#median = float(df['attack angle'].median())
#df['attack angle'] = np.where(df['attack angle'] > median, median, df['attack␣
↪angle'])
```

[ ]:
```

```

[ ]:
```
#df['attack angle']=df.replace(df['attack angle']==0.
↪405465108108164 4,0,inplace=True)
```

[ ]:
```
#df['attack angle'].value_counts()
```

[62]:
```
df.reset_index()
```

[62]:

	index	frequency	attack angle	chord length	free stream velocity	thickness	pressure level
0	4	7.600902	1.948634	-1.188099	4.266896	-5.928163	127.461
1	5	7.824046	1.948634	-1.188099	4.266896	-5.928163	125.571
2	6	8.055158	1.948634	-1.188099	4.266896	-5.928163	125.201
3	7	8.294050	1.948634	-1.188099	4.266896	-5.928163	123.061
4	8	8.517193	1.948634	-1.188099	4.266896	-5.928163	121.301
…	…	…	…	…	…	…	…
1463	1498	7.824046	2.747271	-2.286712	3.678829	-2.940322	110.264
1464	1499	8.055158	2.747271	-2.286712	3.678829	-2.940322	109.254
1465	1500	8.294050	2.747271	-2.286712	3.678829	-2.940322	106.604
1466	1501	8.517193	2.747271	-2.286712	3.678829	-2.940322	106.224
1467	1502	8.748305	2.747271	-2.286712	3.678829	-2.940322	104.204

[1468 rows x 7 columns]

[ ]:
```

```

[ ]:
```
df.head()
```

[ ]:
```
df.head()
```

[81]:
```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

[82]:
```
x
```

[82]:
```
      frequency  attack angle  chord length  free stream velocity  thickness
4      7.600902      1.948634     -1.188099              4.266896  -5.928163
5      7.824046      1.948634     -1.188099              4.266896  -5.928163
6      8.055158      1.948634     -1.188099              4.266896  -5.928163
7      8.294050      1.948634     -1.188099              4.266896  -5.928163
8      8.517193      1.948634     -1.188099              4.266896  -5.928163
…         …             …             …                     …         …
1498   7.824046      2.747271     -2.286712              3.678829  -2.940322
1499   8.055158      2.747271     -2.286712              3.678829  -2.940322
1500   8.294050      2.747271     -2.286712              3.678829  -2.940322
1501   8.517193      2.747271     -2.286712              3.678829  -2.940322
1502   8.748305      2.747271     -2.286712              3.678829  -2.940322

[1464 rows x 5 columns]
```

[83]:
```
y
```

[83]:
```
4       127.461
5       125.571
6       125.201
7       123.061
8       121.301
          …
1498    110.264
1499    109.254
1500    106.604
1501    106.224
1502    104.204
Name: pressure level, Length: 1464, dtype: float64
```

[84]:
```
from sklearn.model_selection import train_test_split
```

[85]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)
```

[86]:
```
from sklearn.linear_model import LinearRegression
```

[87]:
```
linreg=LinearRegression()
```

[88]:
```
linreg.fit(xtrain,ytrain)
```

[88]:
```
LinearRegression()
```

[89]:
```
ypred=linreg.predict(xtest)
```

[90]:
```
from sklearn.metrics import r2_score
```

[91]:
```
r2=r2_score(ytest,ypred)
print(f'accurcy:{r2}')
```

```
accurcy:0.458115108199583
```

[ ]:
```

```

[93]:
```
train=linreg.score(xtrain,ytrain)
test=linreg.score(xtest,ytest)
print(train)
print(test)
```

```
0.43435216891285067
0.458115108199583
```

[ ]:
```

```