03/08/2023 (P16)

In [254]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [255]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\madrid_2016.csv")
df
```

Out[255]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28079008
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0	28079011
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN	28079056
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN	28079057
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN	28079058
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN	28079059
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN	28079060

209496 rows × 14 columns

In [256]:
```
df=df.dropna()
```

In [257]:
```
df.columns
```

Out[257]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [258]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16932 non-null  object 
 1   BEN      16932 non-null  float64
 2   CO       16932 non-null  float64
 3   EBE      16932 non-null  float64
 4   NMHC     16932 non-null  float64
 5   NO       16932 non-null  float64
 6   NO_2     16932 non-null  float64
 7   O_3      16932 non-null  float64
 8   PM10     16932 non-null  float64
 9   PM25     16932 non-null  float64
 10  SO_2     16932 non-null  float64
 11  TCH      16932 non-null  float64
 12  TOL      16932 non-null  float64
 13  station  16932 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

In [259]:
```
data=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
data
```

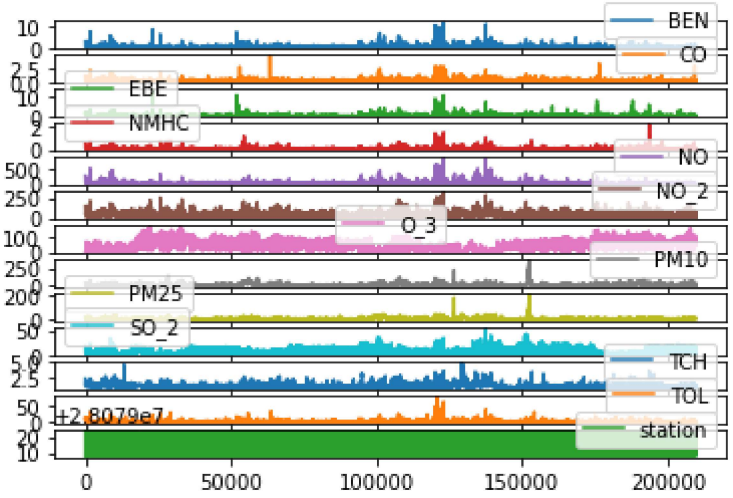
Out[259]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28079008
6	2016-11-01 01:00:00	0.7	0.8	0.4	0.13	57.0	66.0	3.0	23.0	15.0	4.0	1.35	5.0	28079024
25	2016-11-01 02:00:00	2.7	1.0	2.1	0.40	139.0	114.0	4.0	37.0	21.0	14.0	2.30	15.0	28079008
30	2016-11-01 02:00:00	0.7	0.7	0.4	0.13	48.0	59.0	3.0	23.0	15.0	3.0	1.35	5.0	28079024
49	2016-11-01 03:00:00	1.7	0.8	1.4	0.25	53.0	90.0	4.0	31.0	19.0	10.0	1.95	10.7	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209430	2016-06-30 22:00:00	0.1	0.2	0.1	0.02	1.0	5.0	97.0	19.0	12.0	2.0	1.15	0.2	28079024
209449	2016-06-30 23:00:00	0.6	0.4	0.3	0.15	14.0	63.0	54.0	29.0	13.0	16.0	1.48	1.9	28079008
209454	2016-06-30 23:00:00	0.1	0.2	0.1	0.02	1.0	7.0	91.0	16.0	9.0	2.0	1.15	0.3	28079024
209473	2016-07-01 00:00:00	0.6	0.4	0.3	0.16	11.0	68.0	45.0	24.0	14.0	16.0	1.50	1.9	28079008
209478	2016-07-01 00:00:00	0.1	0.2	0.1	0.02	1.0	6.0	89.0	16.0	9.0	2.0	1.15	0.2	28079024

16932 rows × 14 columns

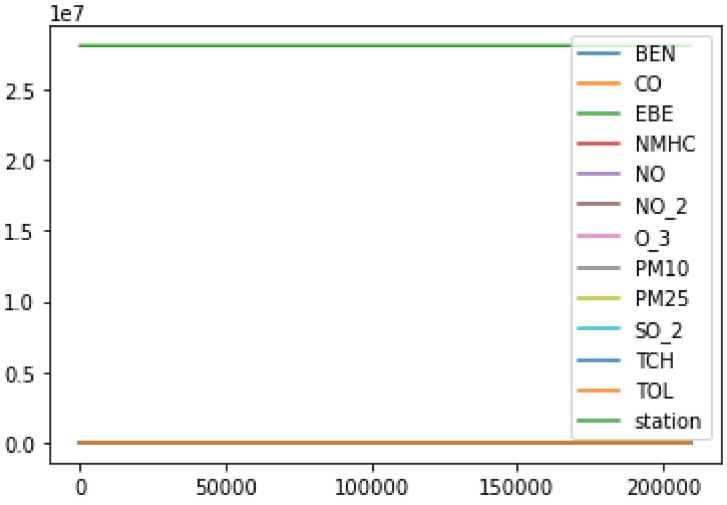
In [260]:
```
data.plot.line(subplots=True)
```

Out[260]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```



In [261]:
```
data.plot.line()
```

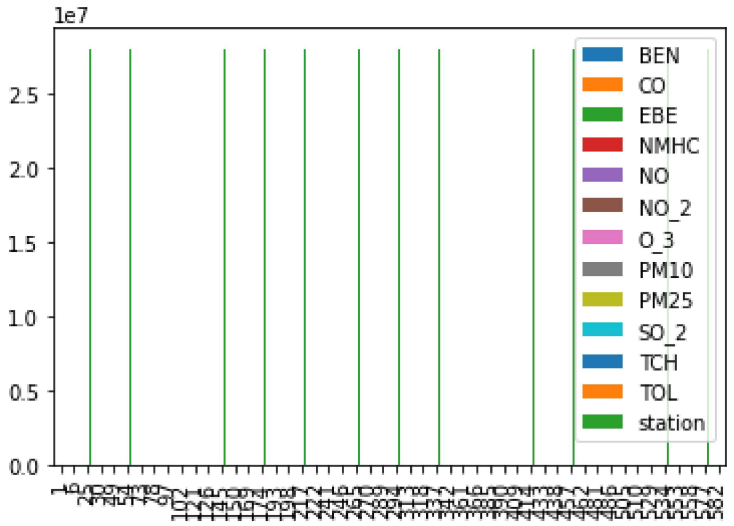
Out[261]:
```
<AxesSubplot:>
```



In [262]:
```
b=data[0:50]
```

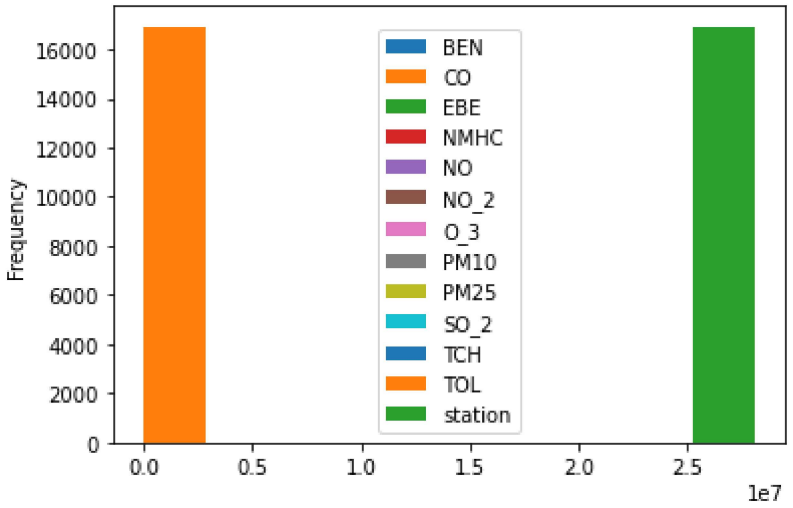
In [263]:
```
b.plot.bar()
```

Out[263]: <AxesSubplot:>



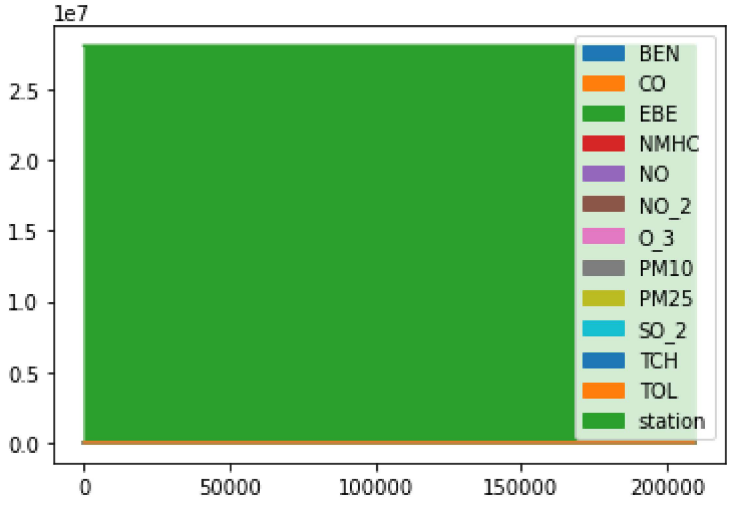
In [264]:
```
data.plot.hist()
```

Out[264]: <AxesSubplot:ylabel='Frequency'>



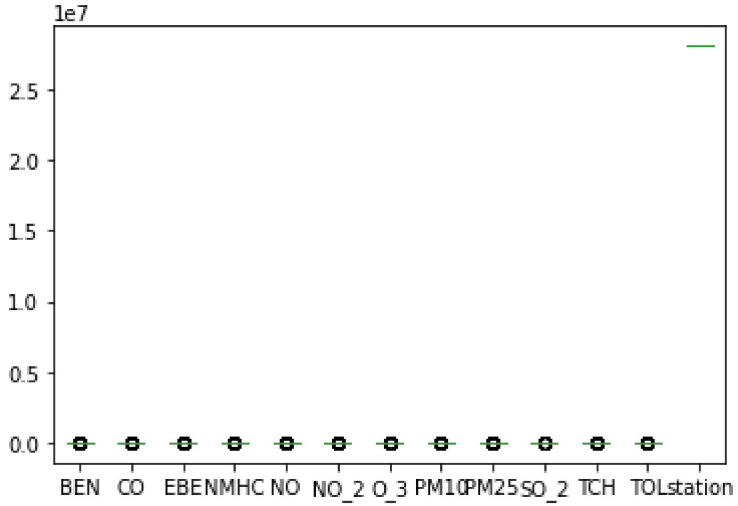
In [265]:
```
data.plot.area()
```

Out[265]: <AxesSubplot:>



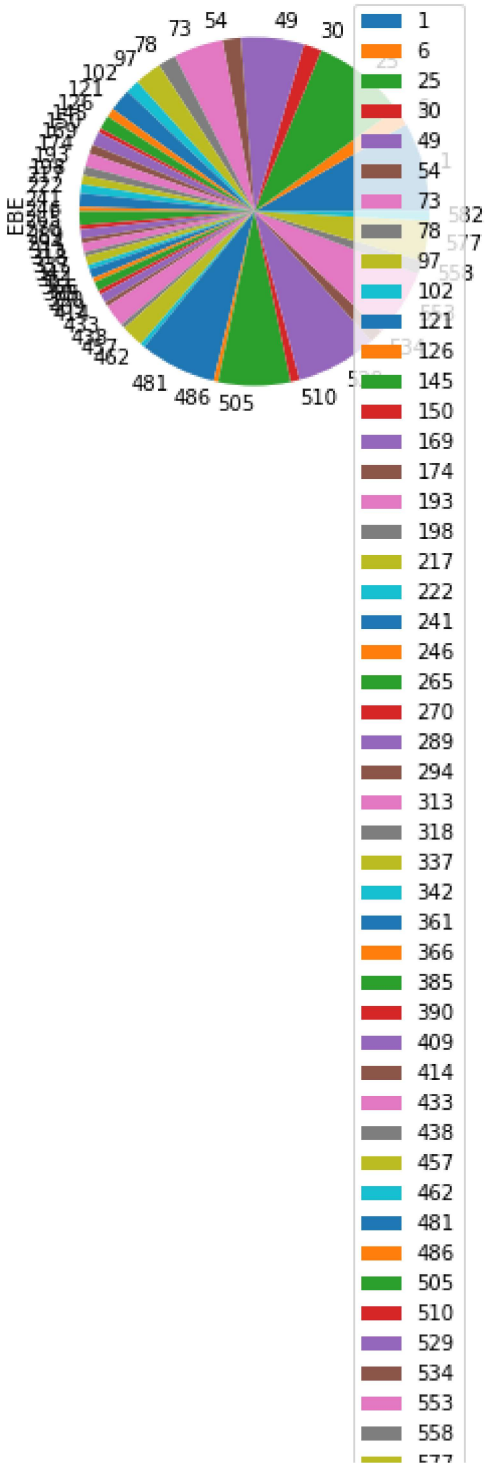
In [266]:
```
data.plot.box()
```

Out[266]: <AxesSubplot:>



In [267]:
```
b.plot.pie(y='EBE' )
```

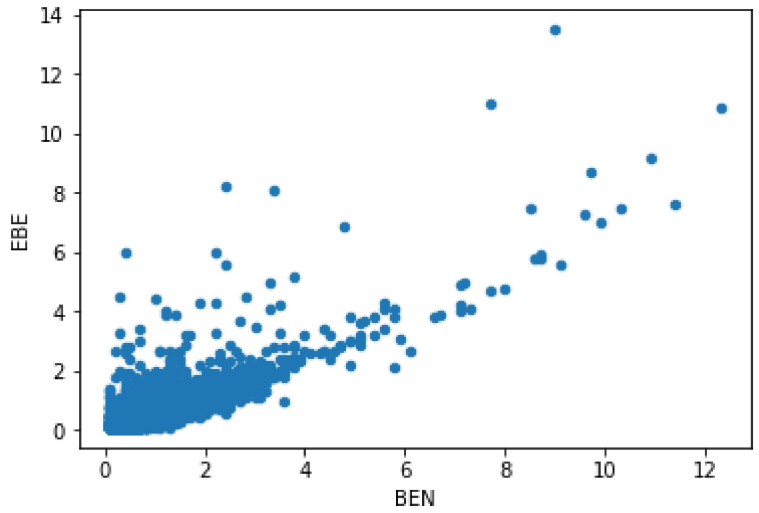
Out[267]:
```
<AxesSubplot:ylabel='EBE'>
```


EBE
73
54
49
30
78
97
102
121
462
481
486
505
510
1
6
25
30
49
54
73
78
97
102
121
126
145
150
169
174
193
198
217
222
241
246
265
270
289
294
313
318
337
342
361
366
385
390
409
414
433
438
457
462
481
486
505
510
529
534
553
558

582

In [268]:
```
data.plot.scatter(x='BEN' ,y='EBE')
```

Out[268]: <AxesSubplot:xlabel='BEN', ylabel='EBE'>



In [269]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16932 non-null  object 
 1   BEN      16932 non-null  float64
 2   CO       16932 non-null  float64
 3   EBE      16932 non-null  float64
 4   NMHC     16932 non-null  float64
 5   NO       16932 non-null  float64
 6   NO_2     16932 non-null  float64
 7   O_3      16932 non-null  float64
 8   PM10     16932 non-null  float64
 9   PM25     16932 non-null  float64
 10  SO_2     16932 non-null  float64
 11  TCH      16932 non-null  float64
 12  TOL      16932 non-null  float64
 13  station  16932 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

In [270]:
```
df.describe()
```

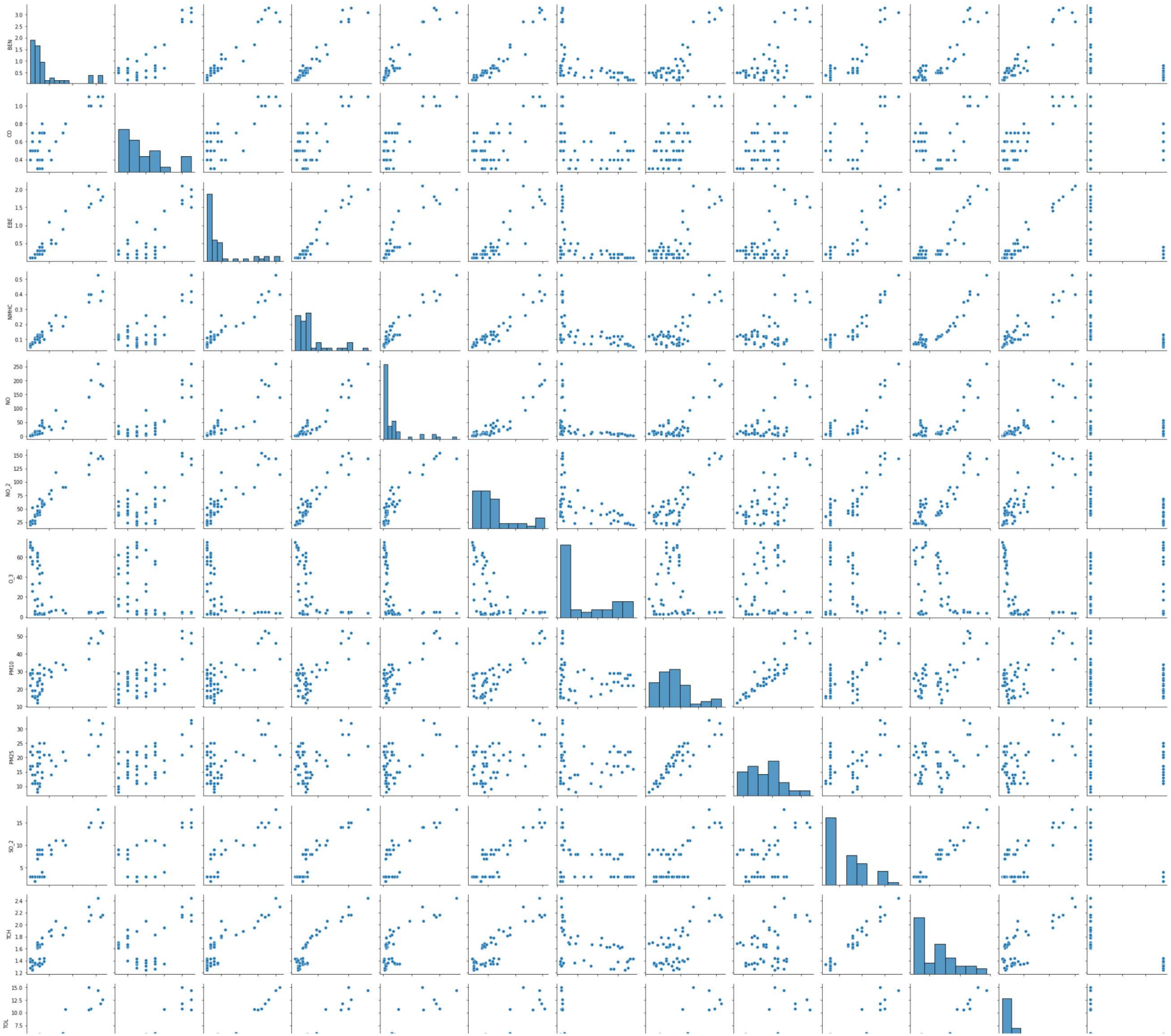
Out[270]:

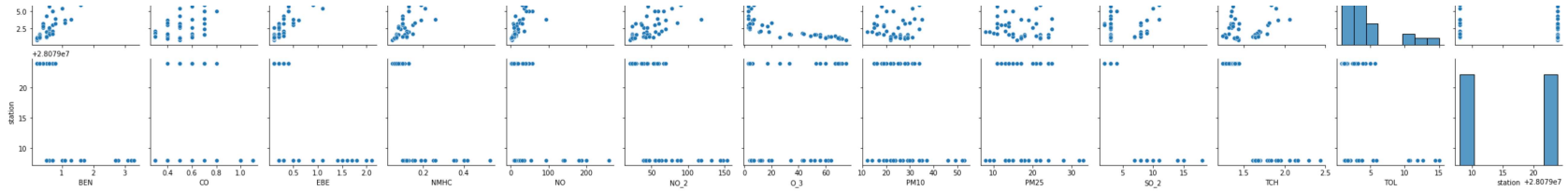
	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25
count	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000
mean	0.537970	0.349941	0.298955	0.099913	20.815734	39.373376	48.118474	19.248110	10.186215
std	0.599479	0.203807	0.450204	0.079850	40.986063	31.170307	32.560277	18.509093	8.418790
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	0.200000	0.200000	0.100000	0.050000	1.000000	14.000000	21.000000	9.000000	5.000000
50%	0.400000	0.300000	0.200000	0.090000	7.000000	34.000000	46.000000	15.000000	8.000000
75%	0.700000	0.400000	0.300000	0.120000	23.000000	58.000000	69.000000	24.000000	14.000000
max	12.300000	4.500000	13.500000	2.210000	829.000000	319.000000	181.000000	367.000000	215.000000

In [271]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [272]:
```
sns.pairplot(df1[0:50])
```

Out[272]:
```
<seaborn.axisgrid.PairGrid at 0x1e55fae6670>
```

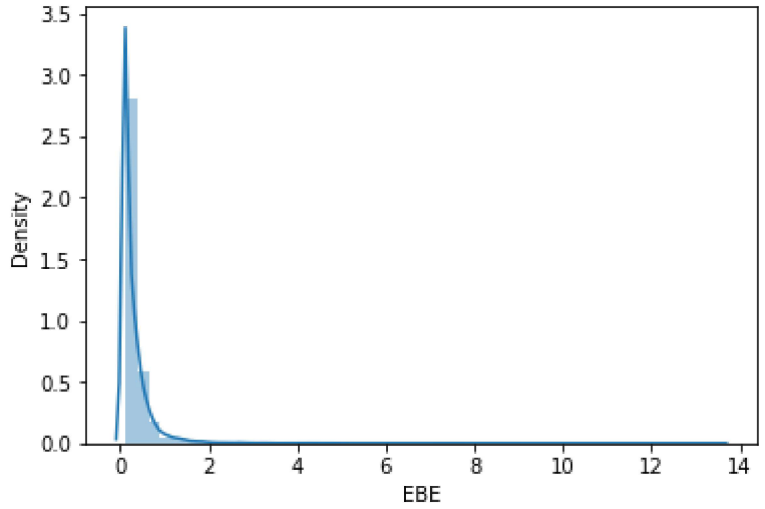

BEN
CO
EBE
NMHC
NO
NO_2
O_3
PM10
PM25
SO_2
TCH
TOL



In [273]:
```
sns.distplot(df1['EBE'])
```

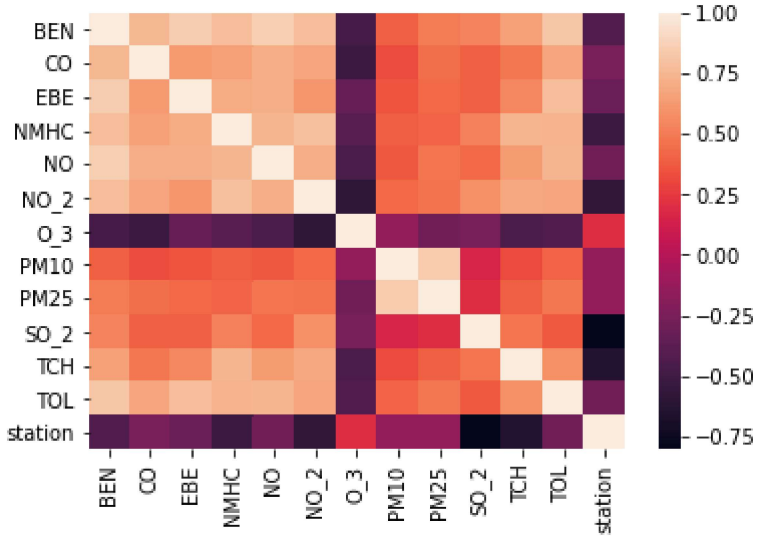
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[273]: <AxesSubplot:xlabel='EBE', ylabel='Density'>



In [274]:
```
sns.heatmap(df1.corr())
```

Out[274]: <AxesSubplot:>



In [275]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [276]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [277]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[277]: LinearRegression()

In [278]:
```
lr.intercept_
```

Out[278]: 28079042.04808828

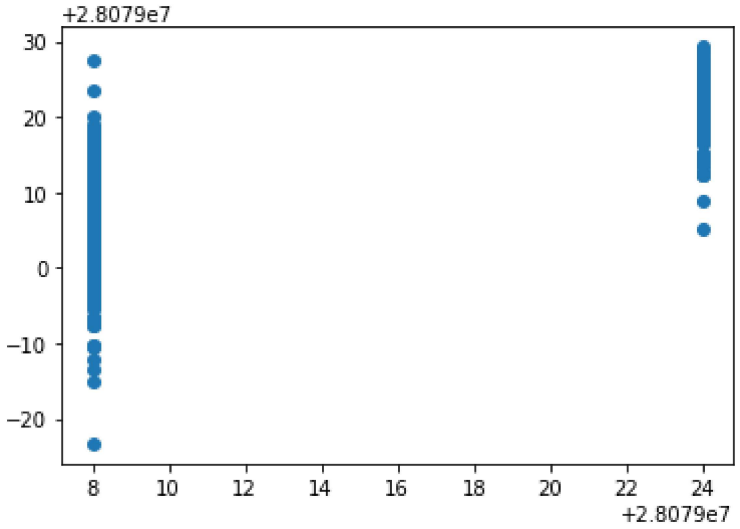
In [279]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[279]:

	Co-efficient
BEN	-1.724141
CO	4.076826
EBE	0.543921
NMHC	2.203850
NO	0.067210
NO_2	-0.066656
O_3	-0.025120
PM10	-0.013344
PM25	0.111345
SO_2	-0.810748
TCH	-14.001716
TOL	0.162649

In [280]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[280]: <matplotlib.collections.PathCollection at 0x1e56d941d30>



In [281]:
```
lr.score(x_test,y_test)
```

Out[281]: 0.833199408628595

In [282]:
```
lr.score(x_train,y_train)
```

Out[282]: 0.8255880256657826

In [283]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [284]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[284]: Ridge(alpha=10)

In [285]:
```
rr.score(x_test,y_test)
```

Out[285]: 0.8326063003770431

In [286]:
```
rr.score(x_train,y_train)
```

Out[286]:
```
0.8254905264011146
```

In [287]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[287]:
```
Lasso(alpha=10)
```

In [288]:
```
la.score(x_train,y_train)
```

Out[288]:
```
0.6436081188413332
```

In [289]:
```
la.score(x_test,y_test)
```

Out[289]:
```
0.6541778487499761
```

In [290]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[290]:
```
ElasticNet()
```

In [291]:
```
en.coef_
```

Out[291]:
```
array([-0.        ,  0.        , -0.        , -0.        ,  0.04478262,
       -0.10548136, -0.02160223,  0.00204987,  0.05733523, -0.87178597,
       -0.03350665,  0.        ])
```

In [292]:
```
en.intercept_
```

Out[292]:
```
28079026.20429332
```

In [293]:
```
prediction=en.predict(x_test)
```

In [294]:
```
en.score(x_test,y_test)
```

Out[294]:
```
0.7101269118542957
```

In [295]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
3.3045877482192076
18.55072742261088
4.307055539763898
```

In [296]:
```
from sklearn.linear_model import LogisticRegression
```

In [297]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [298]:
```
feature_matrix.shape
```

Out[298]:
```
(16932, 12)
```

In [299]:
```
target_vector.shape
```

Out[299]:
```
(16932,)
```

In [300]:
```
from sklearn.preprocessing import StandardScaler
```

In [301]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [302]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[302]:
```
LogisticRegression(max_iter=10000)
```

In [303]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12]]
```

In [304]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [305]:
```
logr.classes_
```

Out[305]:
```
array([28079008, 28079024], dtype=int64)
```

In [306]:
```
logr.score(fs,target_vector)
```

Out[306]:
```
0.996161115048429
```

In [307]:
```
logr.predict_proba(observation)[0][0]
```

Out[307]:
```
1.0
```

In [308]:
```
logr.predict_proba(observation)
```

Out[308]:
```
array([[1.00000000e+00, 1.38109307e-55]])
```

In [309]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [310]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[310]:
```
RandomForestClassifier()
```

In [311]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [312]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[312]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [313]:
```
grid_search.best_score_
```

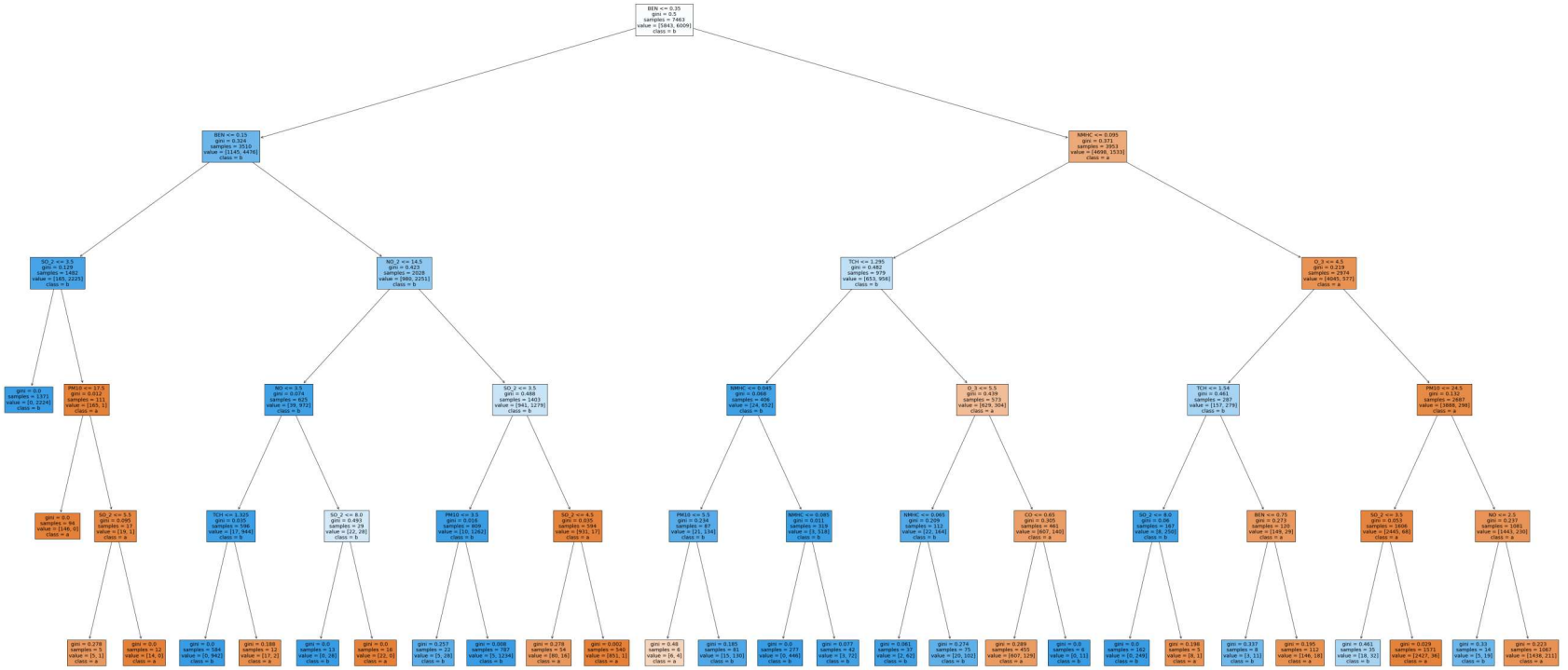
Out[313]:
```
0.994937563280459
```

In [314]:
```
rfc_best=grid_search.best_estimator_
```

In [315]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],filled=True)
```

Out[315]:
```
[Text(1901.3333333333335, 1993.2, 'BEN <= 0.35\ngini = 0.5\nsamples = 7463\nvalue = [5843, 6009]\nclass =
b'),
 Text(661.3333333333334, 1630.8000000000002, 'BEN <= 0.15\ngini = 0.324\nsamples = 3510\nvalue = [1145, 447
6]\nclass = b'),
 Text(165.33333333333334, 1268.4, 'SO_2 <= 3.5\ngini = 0.129\nsamples = 1482\nvalue = [165, 2225]\nclass =
b'),
 Text(82.66666666666667, 906.0, 'gini = 0.0\nsamples = 1371\nvalue = [0, 2224]\nclass = b'),
 Text(248.0, 906.0, 'PM10 <= 17.5\ngini = 0.012\nsamples = 111\nvalue = [165, 1]\nclass = a'),
 Text(165.33333333333334, 543.5999999999999, 'gini = 0.0\nsamples = 94\nvalue = [146, 0]\nclass = a'),
 Text(330.6666666666667, 543.5999999999999, 'SO_2 <= 5.5\ngini = 0.095\nsamples = 17\nvalue = [19, 1]\nclass
= a'),
 Text(248.0, 181.19999999999982, 'gini = 0.278\nsamples = 5\nvalue = [5, 1]\nclass = a'),
 Text(413.33333333333337, 181.19999999999982, 'gini = 0.0\nsamples = 12\nvalue = [14, 0]\nclass = a'),
 Text(1157.3333333333335, 1268.4, 'NO_2 <= 14.5\ngini = 0.423\nsamples = 2028\nvalue = [980, 2251]\nclass =
b'),
 Text(826.6666666666667, 906.0, 'NO <= 3.5\ngini = 0.074\nsamples = 625\nvalue = [39, 972]\nclass = b'),
 Text(661.3333333333334, 543.5999999999999, 'TCH <= 1.325\ngini = 0.035\nsamples = 596\nvalue = [17, 944]\nc
lass = b'),
 Text(578.6666666666667, 181.19999999999982, 'gini = 0.0\nsamples = 584\nvalue = [0, 942]\nclass = b'),
 Text(744.0, 181.19999999999982, 'gini = 0.188\nsamples = 12\nvalue = [17, 2]\nclass = a'),
 Text(992.0, 543.5999999999999, 'SO_2 <= 8.0\ngini = 0.493\nsamples = 29\nvalue = [22, 28]\nclass = b'),
 Text(909.3333333333334, 181.19999999999982, 'gini = 0.0\nsamples = 13\nvalue = [0, 28]\nclass = b'),
 Text(1074.6666666666667, 181.19999999999982, 'gini = 0.0\nsamples = 16\nvalue = [22, 0]\nclass = a'),
 Text(1488.0, 906.0, 'SO_2 <= 3.5\ngini = 0.488\nsamples = 1403\nvalue = [941, 1279]\nclass = b'),
 Text(1322.6666666666667, 543.5999999999999, 'PM10 <= 3.5\ngini = 0.016\nsamples = 809\nvalue = [10, 1262]\n
class = b'),
 Text(1240.0, 181.19999999999982, 'gini = 0.257\nsamples = 22\nvalue = [5, 28]\nclass = b'),
 Text(1405.3333333333335, 181.19999999999982, 'gini = 0.008\nsamples = 787\nvalue = [5, 1234]\nclass = b'),
 Text(1653.3333333333335, 543.5999999999999, 'SO_2 <= 4.5\ngini = 0.035\nsamples = 594\nvalue = [931, 17]\nc
lass = a'),
 Text(1570.6666666666667, 181.19999999999982, 'gini = 0.278\nsamples = 54\nvalue = [80, 16]\nclass = a'),
 Text(1736.0, 181.19999999999982, 'gini = 0.002\nsamples = 540\nvalue = [851, 1]\nclass = a'),
 Text(3141.3333333333335, 1630.8000000000002, 'NMHC <= 0.095\ngini = 0.371\nsamples = 3953\nvalue = [4698, 1
533]\nclass = a'),
 Text(2480.0, 1268.4, 'TCH <= 1.295\ngini = 0.482\nsamples = 979\nvalue = [653, 956]\nclass = b'),
 Text(2149.3333333333335, 906.0, 'NMHC <= 0.045\ngini = 0.068\nsamples = 406\nvalue = [24, 652]\nclass =
b'),
 Text(1984.0, 543.5999999999999, 'PM10 <= 5.5\ngini = 0.234\nsamples = 87\nvalue = [21, 134]\nclass = b'),
 Text(1901.3333333333335, 181.19999999999982, 'gini = 0.48\nsamples = 6\nvalue = [6, 4]\nclass = a'),
 Text(2066.666666666667, 181.19999999999982, 'gini = 0.185\nsamples = 81\nvalue = [15, 130]\nclass = b'),
 Text(2314.666666666667, 543.5999999999999, 'NMHC <= 0.085\ngini = 0.011\nsamples = 319\nvalue = [3, 518]\nc
lass = b'),
 Text(2232.0, 181.19999999999982, 'gini = 0.0\nsamples = 277\nvalue = [0, 446]\nclass = b'),
```

```
 Text(2397.3333333333335, 181.19999999999982, 'gini = 0.077\nsamples = 42\nvalue = [3, 72]\nclass = b'),
 Text(2810.666666666667, 906.0, 'O_3 <= 5.5\ngini = 0.439\nsamples = 573\nvalue = [629, 304]\nclass = a'),
 Text(2645.3333333333335, 543.5999999999999, 'NMHC <= 0.065\ngini = 0.209\nsamples = 112\nvalue = [22, 164]
\nclass = b'),
 Text(2562.666666666667, 181.19999999999982, 'gini = 0.061\nsamples = 37\nvalue = [2, 62]\nclass = b'),
 Text(2728.0, 181.19999999999982, 'gini = 0.274\nsamples = 75\nvalue = [20, 102]\nclass = b'),
 Text(2976.0, 543.5999999999999, 'CO <= 0.65\ngini = 0.305\nsamples = 461\nvalue = [607, 140]\nclass = a'),
 Text(2893.3333333333335, 181.19999999999982, 'gini = 0.289\nsamples = 455\nvalue = [607, 129]\nclass = a'),
 Text(3058.666666666667, 181.19999999999982, 'gini = 0.0\nsamples = 6\nvalue = [0, 11]\nclass = b'),
 Text(3802.666666666667, 1268.4, 'O_3 <= 4.5\ngini = 0.219\nsamples = 2974\nvalue = [4045, 577]\nclass =
a'),
 Text(3472.0, 906.0, 'TCH <= 1.54\ngini = 0.461\nsamples = 287\nvalue = [157, 279]\nclass = b'),
 Text(3306.666666666667, 543.5999999999999, 'SO_2 <= 8.0\ngini = 0.06\nsamples = 167\nvalue = [8, 250]\nclas
s = b'),
 Text(3224.0, 181.19999999999982, 'gini = 0.0\nsamples = 162\nvalue = [0, 249]\nclass = b'),
 Text(3389.3333333333335, 181.19999999999982, 'gini = 0.198\nsamples = 5\nvalue = [8, 1]\nclass = a'),
 Text(3637.3333333333335, 543.5999999999999, 'BEN <= 0.75\ngini = 0.273\nsamples = 120\nvalue = [149, 29]\nc
lass = a'),
 Text(3554.666666666667, 181.19999999999982, 'gini = 0.337\nsamples = 8\nvalue = [3, 11]\nclass = b'),
 Text(3720.0, 181.19999999999982, 'gini = 0.195\nsamples = 112\nvalue = [146, 18]\nclass = a'),
 Text(4133.333333333334, 906.0, 'PM10 <= 24.5\ngini = 0.132\nsamples = 2687\nvalue = [3888, 298]\nclass =
a'),
 Text(3968.0, 543.5999999999999, 'SO_2 <= 3.5\ngini = 0.053\nsamples = 1606\nvalue = [2445, 68]\nclass =
a'),
 Text(3885.3333333333335, 181.19999999999982, 'gini = 0.461\nsamples = 35\nvalue = [18, 32]\nclass = b'),
 Text(4050.666666666667, 181.19999999999982, 'gini = 0.029\nsamples = 1571\nvalue = [2427, 36]\nclass = a'),
 Text(4298.666666666667, 543.5999999999999, 'NO <= 2.5\ngini = 0.237\nsamples = 1081\nvalue = [1443, 230]\nc
lass = a'),
 Text(4216.0, 181.19999999999982, 'gini = 0.33\nsamples = 14\nvalue = [5, 19]\nclass = b'),
 Text(4381.333333333334, 181.19999999999982, 'gini = 0.223\nsamples = 1067\nvalue = [1438, 211]\nclass =
a')]
```



Conclusion

```
linear Regression=0.8255880256657826

Ridge Regression=0.8254905264011146

Lasso Regression=0.6436081188413332

ElasticNet Regression=0.7101269118542957

Logistic Regression=0.996161115048429

Random Forest=0.994937563280459

Logistic Regression is Suitable for this Dataset=0.996161115048429
```