Vijay(P3) 3/08/2023

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\madrid_2003.csv")
df
```

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL	stat
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	24.299999	NaN	NaN	28079(
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	14.230000	1.55	NaN	28079(
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	17.879999	NaN	NaN	28079(
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	24.900000	NaN	NaN	28079(
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	18.750000	NaN	NaN	28079(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.870000	1.27	1.00	28079(
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50	8.360000	NaN	0.88	28079(
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN	5.330000	1.55	NaN	28079(
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN	6.830000	1.27	NaN	28079(
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.060000	1.32	5.67	28079(

243984 rows × 16 columns

In [10]:
```
df=df.dropna()
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33010 entries, 5 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     33010 non-null  object 
 1   BEN      33010 non-null  float64
 2   CO       33010 non-null  float64
 3   EBE      33010 non-null  float64
 4   MXY      33010 non-null  float64
 5   NMHC     33010 non-null  float64
 6   NO_2     33010 non-null  float64
 7   NOx      33010 non-null  float64
 8   OXY      33010 non-null  float64
 9   O_3      33010 non-null  float64
 10  PM10     33010 non-null  float64
 11  PXY      33010 non-null  float64
 12  SO_2     33010 non-null  float64
 13  TCH      33010 non-null  float64
 14  TOL      33010 non-null  float64
 15  station  33010 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.3+ MB
```

In [13]:
```
data=df[['EBE', 'MXY', 'PXY']]
data
```

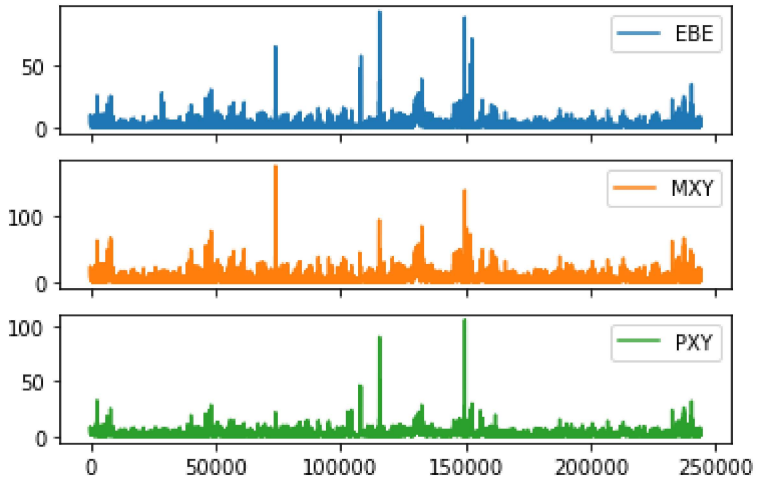
Out[13]:

	EBE	MXY	PXY
5	9.83	21.49	7.94
23	3.43	7.08	2.62
27	5.75	10.88	4.24
33	10.63	24.73	8.93
51	3.20	7.08	2.70
...	...	...	...
243955	3.07	9.38	3.48
243957	3.88	10.86	3.89
243961	4.53	10.88	4.13
243979	2.01	3.17	1.20
243983	2.15	6.41	2.43

33010 rows × 3 columns

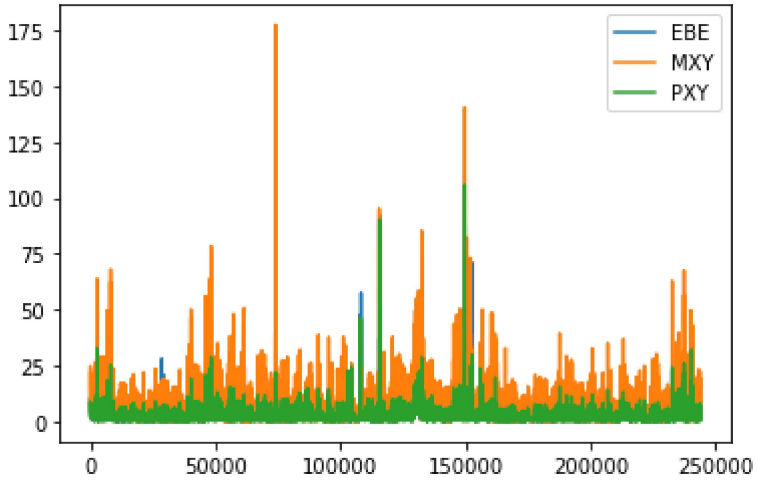
In [14]:
```
data.plot.line(subplots=True)
```

Out[14]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [15]:
```
data.plot.line()
```

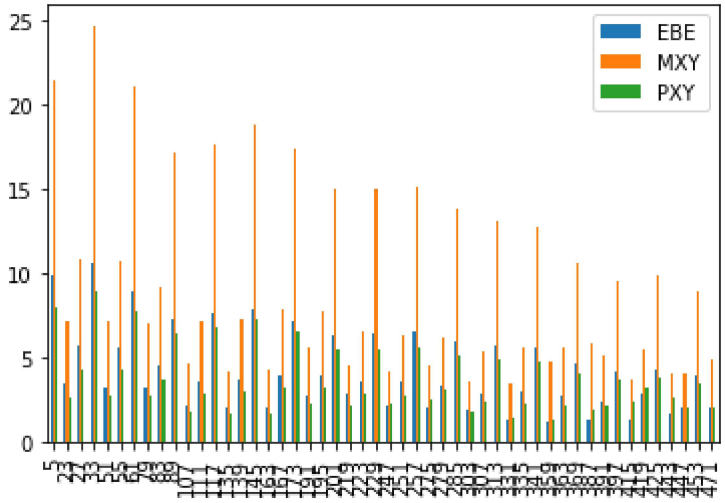
Out[15]:
```
<AxesSubplot:>
```



In [16]:
```
b=data[0:50]
```

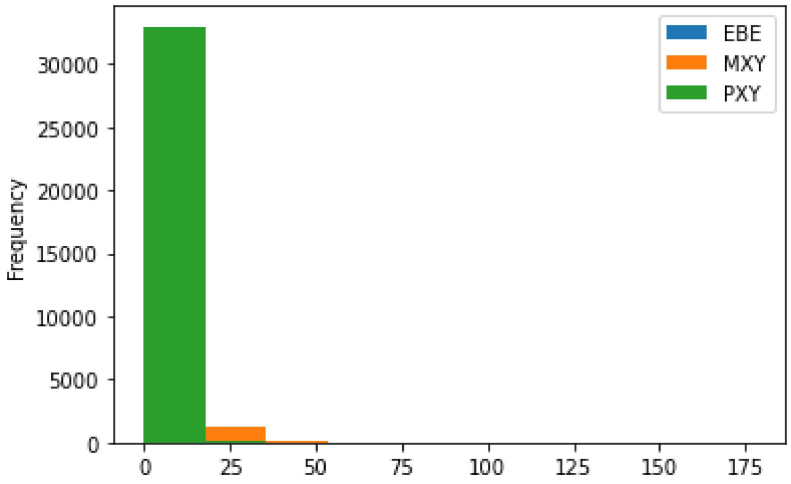
In [18]:
```
b.plot.bar()
```

Out[18]: <AxesSubplot:>



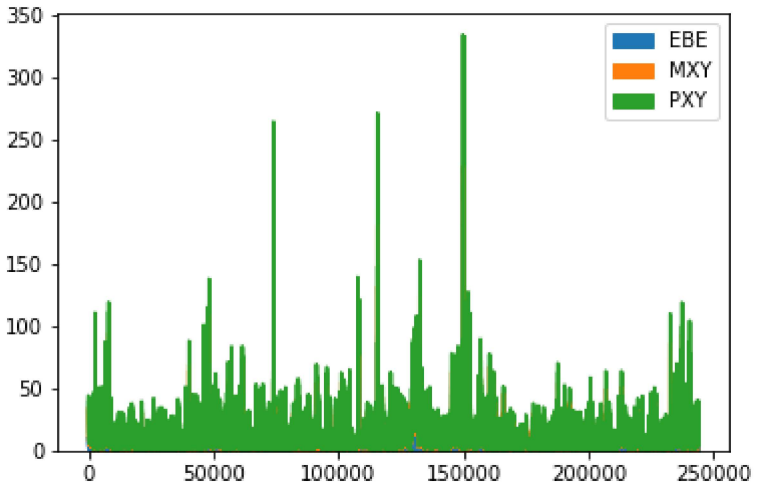
In [19]:
```
data.plot.hist()
```

Out[19]: <AxesSubplot:ylabel='Frequency'>



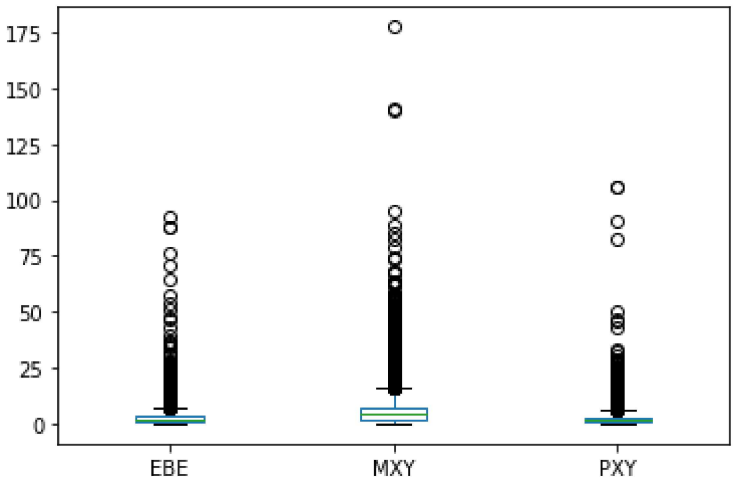
In [20]:
```
data.plot.area()
```

Out[20]: <AxesSubplot:>



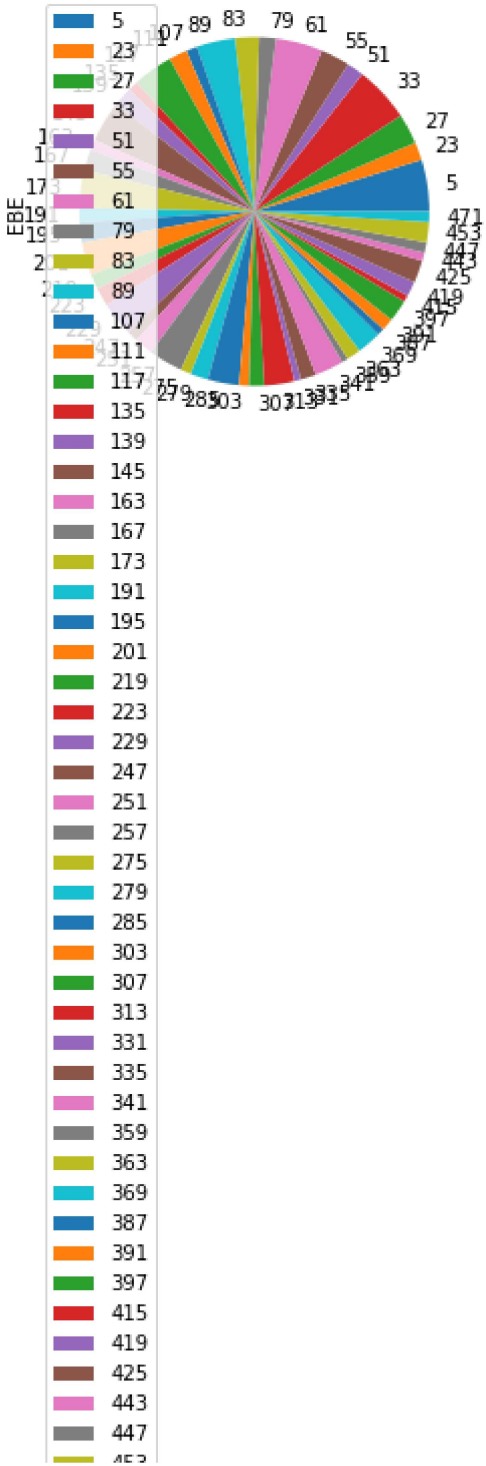
In [21]:
```
data.plot.box()
```

Out[21]: <AxesSubplot:>



In [22]:
```
b.plot.pie(y='EBE' )
```

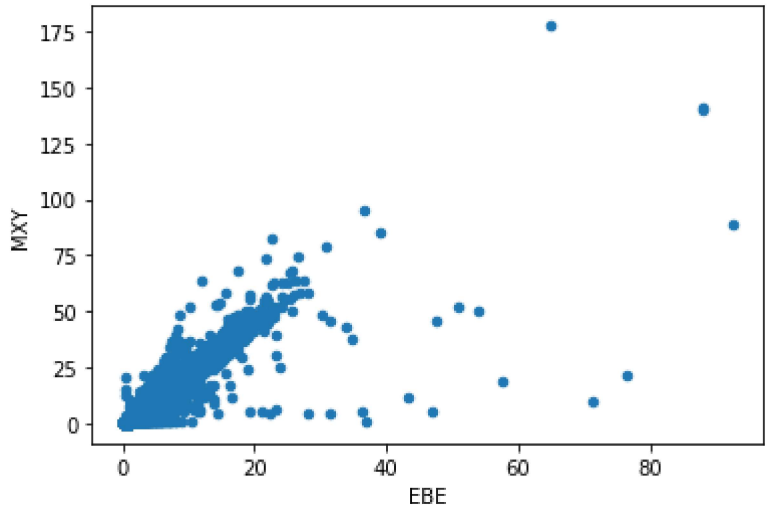
Out[22]:
```
<AxesSubplot:ylabel='EBE'>
```


EBE
5
23
27
33
51
55
61
79
83
89
107
111
117
135
139
145
163
167
173
191
195
201
219
223
229
247
251
257
275
279
285
303
307
313
331
335
341
359
363
369
387
391
397
415
419
425
443
447

471

In [23]:
```
data.plot.scatter(x='EBE' ,y='MXY')
```

Out[23]: <AxesSubplot:xlabel='EBE', ylabel='MXY'>



In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33010 entries, 5 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     33010 non-null  object 
 1   BEN      33010 non-null  float64
 2   CO       33010 non-null  float64
 3   EBE      33010 non-null  float64
 4   MXY      33010 non-null  float64
 5   NMHC     33010 non-null  float64
 6   NO_2     33010 non-null  float64
 7   NOx      33010 non-null  float64
 8   OXY      33010 non-null  float64
 9   O_3      33010 non-null  float64
 10  PM10     33010 non-null  float64
 11  PXY      33010 non-null  float64
 12  SO_2     33010 non-null  float64
 13  TCH      33010 non-null  float64
 14  TOL      33010 non-null  float64
 15  station  33010 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.3+ MB
```

In [26]:
```
df.describe()
```

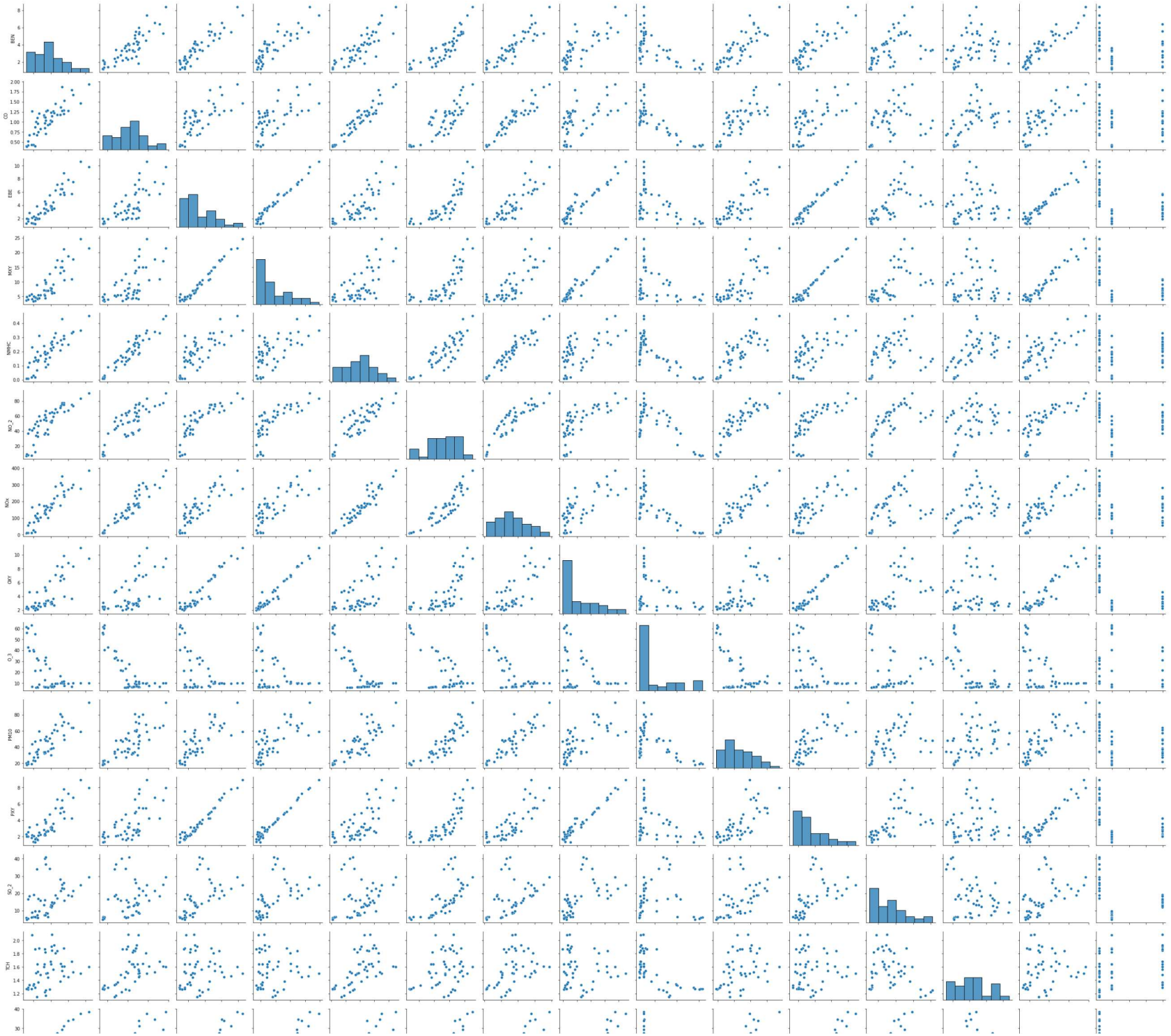
Out[26]:

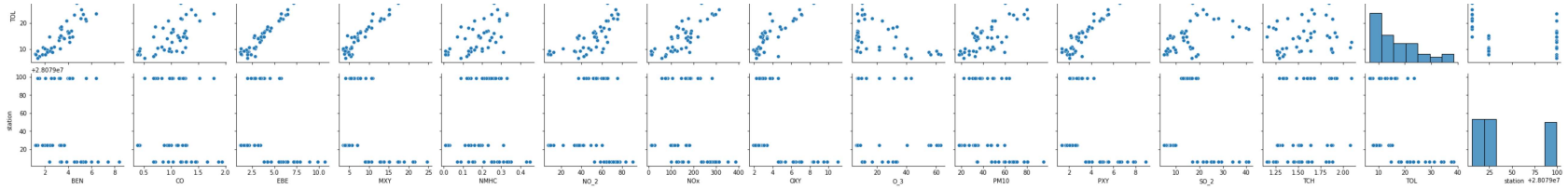
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000
mean	2.192633	0.759868	2.639726	5.838414	0.137177	57.328049	120.153676	2.684084	36.914485
std	2.064160	0.545999	2.825194	6.267296	0.127863	31.811082	104.521700	2.717832	28.988487
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.130000
25%	0.900000	0.430000	1.010000	1.880000	0.060000	34.529999	49.070000	1.000000	12.230000
50%	1.610000	0.620000	1.890000	4.070000	0.110000	55.105000	92.779999	1.790000	30.400000
75%	2.810000	0.930000	3.300000	7.530000	0.170000	76.160004	160.100006	3.340000	54.349998
max	66.389999	7.920000	92.589996	177.600006	2.180000	342.700012	1246.000000	88.180000	178.699997

In [27]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [28]:
```
sns.pairplot(df1[0:50])
```

Out[28]:
```
<seaborn.axisgrid.PairGrid at 0x1b133924460>
```

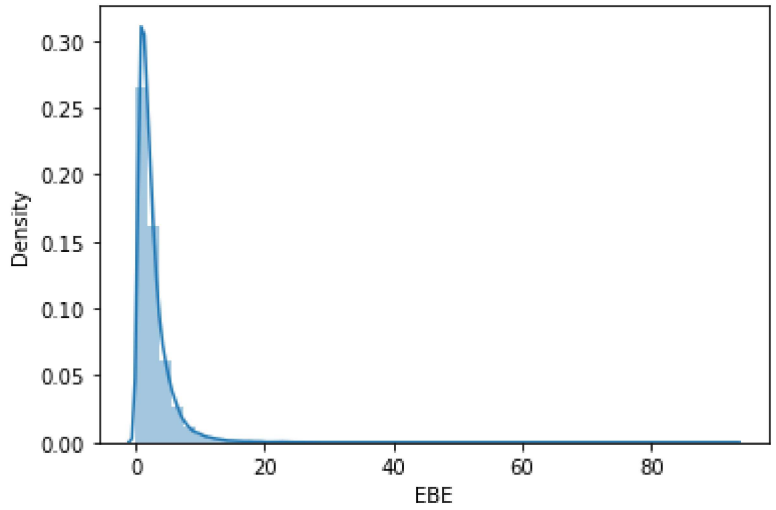

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



In [29]:
```
sns.distplot(df1['EBE'])
```

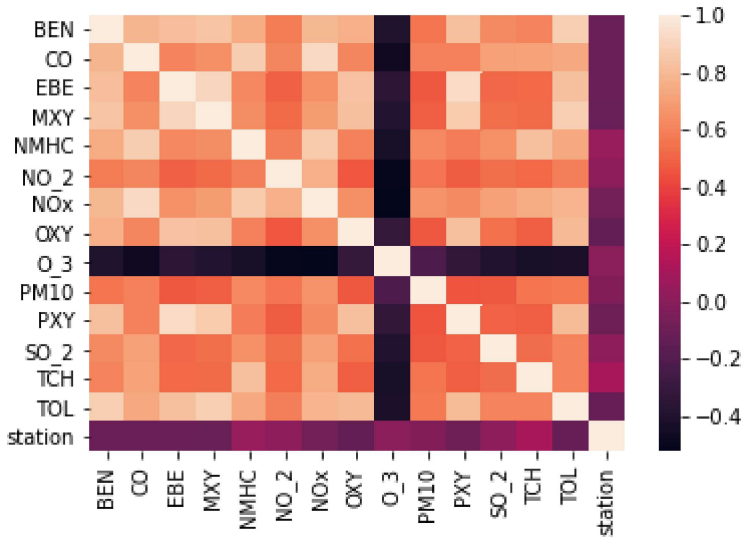
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[29]: <AxesSubplot:xlabel='EBE', ylabel='Density'>



In [31]:
```
sns.heatmap(df1.corr())
```

Out[31]: <AxesSubplot:>



In [32]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [33]:
```
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 [34]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[34]: LinearRegression()

In [35]:
```
lr.intercept_
```

Out[35]: 28078999.693834357

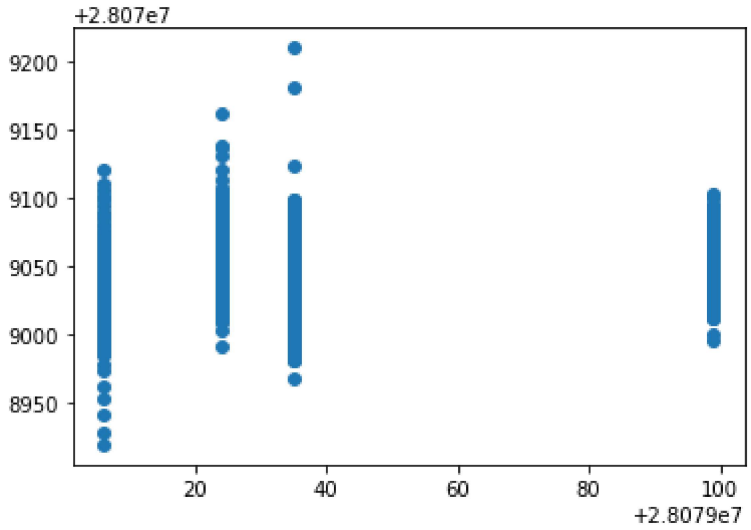
In [36]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[36]:

	Co-efficient
BEN	1.269026
CO	-38.475082
EBE	-1.499653
MXY	0.069212
NMHC	154.440710
NO_2	0.159445
NOx	-0.072799
OXY	-1.193262
O_3	-0.012126
PM10	-0.072790
PXY	1.638311
SO_2	0.881308
TCH	36.867399
TOL	-0.826314

In [37]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[37]: <matplotlib.collections.PathCollection at 0x1b1446ee940>



In [38]:
```
lr.score(x_test,y_test)
```

Out[38]: 0.1687896410853008

In [39]:
```
lr.score(x_train,y_train)
```

Out[39]: 0.1790831213614923

In [40]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [41]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[41]: Ridge(alpha=10)

In [42]:
```
rr.score(x_test,y_test)
```

Out[42]: 0.1678366928792907

In [43]:
```
rr.score(x_train,y_train)
```

Out[43]:
```
0.1780093328816681
```

In [44]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[44]:
```
Lasso(alpha=10)
```

In [45]:
```
la.score(x_train,y_train)
```

Out[45]:
```
0.0363881269887365
```

In [46]:
```
la.score(x_test,y_test)
```

Out[46]:
```
0.03420563530990339
```

In [47]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[47]:
```
ElasticNet()
```

In [48]:
```
en.coef_
```

Out[48]:
```
array([-0.        , -0.28886968,  0.07744697, -0.0537006 ,  0.12892798,
        0.14167371, -0.06796021, -1.15272212, -0.04460086,  0.0589397 ,
        0.24690545,  0.78121411,  1.62363296, -0.44187145])
```

In [49]:
```
en.intercept_
```

Out[49]:
```
28079038.03252416
```

In [50]:
```
prediction=en.predict(x_test)
```

In [51]:
```
en.score(x_test,y_test)
```

Out[51]:
```
0.04750469706113003
```

In [52]:
```
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)))
```

```
29.00655883615497
1178.8179033562951
34.333917681445776
```

In [53]:
```
from sklearn.linear_model import LogisticRegression
```

In [54]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [57]:
```
feature_matrix.shape
```

Out[57]:
```
(33010, 14)
```

In [58]:
```
target_vector.shape
```

Out[58]:
```
(33010,)
```

In [59]:
```
from sklearn.preprocessing import StandardScaler
```

In [60]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [61]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[61]:
```
LogisticRegression(max_iter=10000)
```

In [62]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [63]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079035]
```

In [64]:
```
logr.classes_
```

Out[64]:
```
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

In [65]:
```
logr.score(fs,target_vector)
```

Out[65]:
```
0.7584974250227204
```

In [66]:
```
logr.predict_proba(observation)[0][0]
```

Out[66]:
```
2.3306153265290618e-23
```

In [67]:
```
logr.predict_proba(observation)
```

Out[67]:
```
array([[2.33061533e-23, 1.44436075e-55, 1.00000000e+00, 6.68457491e-16]])
```

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [69]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[69]:
```
RandomForestClassifier()
```

In [70]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [ ]:
```
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)
```

In [72]:
```
grid_search.best_score_
```

Out[72]:
```
0.7290431072600644
```

In [75]:
```
rfc_best=grid_search.best_estimator_
```

In [77]:
```
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[77]:
```
[Text(2222.7000000000003, 1993.2, 'NO_2 <= 26.305\ngini = 0.75\nsamples = 14637\nvalue = [5408, 5858, 5911,
5930]\nclass = d'),
 Text(1171.8000000000002, 1630.8000000000002, 'MXY <= 2.245\ngini = 0.496\nsamples = 2514\nvalue = [498, 27
21, 397, 352]\nclass = b'),
 Text(595.2, 1268.4, 'SO_2 <= 7.235\ngini = 0.364\nsamples = 2073\nvalue = [78, 2584, 351, 276]\nclass =
b'),
 Text(297.6, 906.0, 'MXY <= 1.095\ngini = 0.192\nsamples = 1720\nvalue = [42, 2454, 115, 126]\nclass = b'),
 Text(148.8, 543.5999999999999, 'PXY <= 1.31\ngini = 0.065\nsamples = 1339\nvalue = [1, 2047, 44, 26]\nclas
s = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.05\nsamples = 1327\nvalue = [1, 2047, 27, 26]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 12\nvalue = [0, 0, 17, 0]\nclass =
c'),
 Text(446.40000000000003, 543.5999999999999, 'PXY <= 0.725\ngini = 0.524\nsamples = 381\nvalue = [41, 407,
71, 100]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.271\nsamples = 237\nvalue = [24, 331, 23, 12]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.693\nsamples = 144\nvalue = [17, 76, 48, 88]\nclass
= d'),
 Text(892.8000000000001, 906.0, 'NMHC <= 0.035\ngini = 0.684\nsamples = 353\nvalue = [36, 130, 236, 150]\nc
lass = c'),
```

Conclusion

linear Regression=0.1790831213614923

Ridge Regression=0.1780093328816681

Lasso Regression=0.1780093328816681

ElasticNet Regression=0.04750469706113003

Logistic Regression=0.7584974250227204

Random Forest=0.7290431072600644

Logistic Regression is Suitable for this Dataset=0.758497425022720