Vijay(P11) 3/08/2023

In [67]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [197]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_201
df
```

Out[197]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	28079004
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	28079011
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	28079056
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	28079057
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	28079058
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	28079059
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	28079060

209928 rows × 14 columns

In [198]:
```
df=df.dropna()
```

In [199]:
```
df.columns

```

Out[199]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [200]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16460 non-null  object 
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

In [224]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3
```

Out[224]:

	BEN	CO	station
1	2.5	0.4	28079008
6	0.7	0.3	28079024
25	1.8	0.3	28079008
30	1.0	0.4	28079024
49	1.3	0.2	28079008
...	...	...	...
18457	2.5	0.3	28079008
18462	0.7	0.3	28079024
18481	0.6	0.1	28079008
18486	0.6	0.2	28079024
18505	0.3	0.1	28079008

1500 rows × 3 columns

In [225]:
```
df=df.head(1500)
df
```

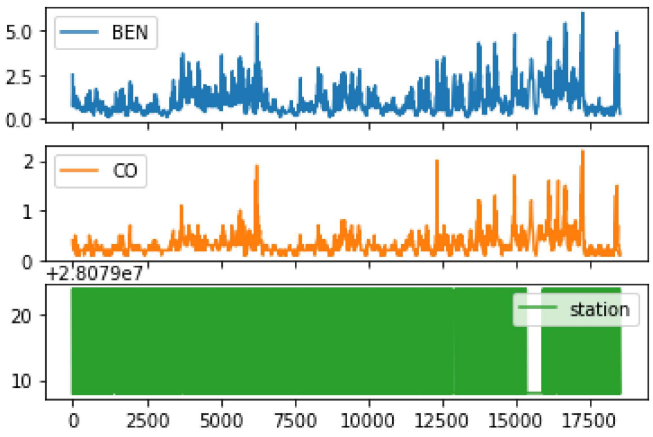
Out[225]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
6	2011-11-01 01:00:00	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.7	28079024
25	2011-11-01 02:00:00	1.8	0.3	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	1.71	7.4	28079008
30	2011-11-01 02:00:00	1.0	0.4	1.3	0.18	31.0	67.0	5.0	25.0	18.0	3.0	1.40	2.9	28079024
49	2011-11-01 03:00:00	1.3	0.2	2.4	0.22	29.0	72.0	3.0	33.0	20.0	8.0	1.75	6.2	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18457	2011-10-03 02:00:00	2.5	0.3	3.7	0.27	30.0	89.0	17.0	34.0	20.0	4.0	1.58	9.9	28079008
18462	2011-10-03 02:00:00	0.7	0.3	1.0	0.14	1.0	41.0	34.0	30.0	19.0	2.0	1.31	2.0	28079024
18481	2011-10-03 03:00:00	0.6	0.1	1.2	0.19	2.0	38.0	48.0	21.0	11.0	3.0	1.36	5.0	28079008
18486	2011-10-03 03:00:00	0.6	0.2	1.1	0.13	1.0	20.0	44.0	26.0	17.0	2.0	1.31	1.4	28079024
18505	2011-10-03 04:00:00	0.3	0.1	1.1	0.18	1.0	29.0	50.0	23.0	13.0	3.0	1.33	4.0	28079008

1500 rows × 14 columns

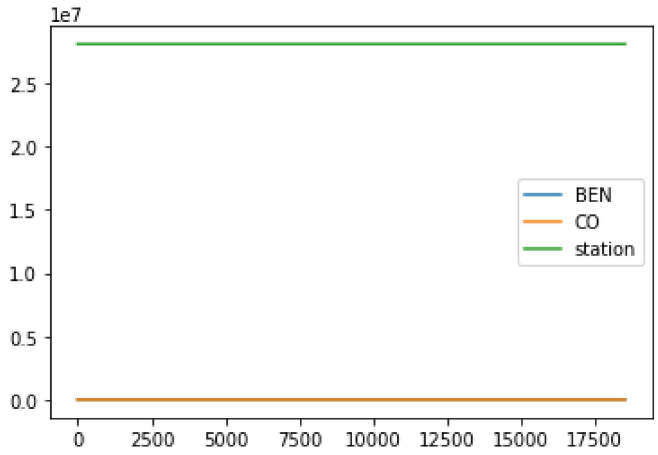
In [226]:
```
data.plot.line(subplots=True)
```

Out[226]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [227]:
```
1 data.plot.line()
2
```

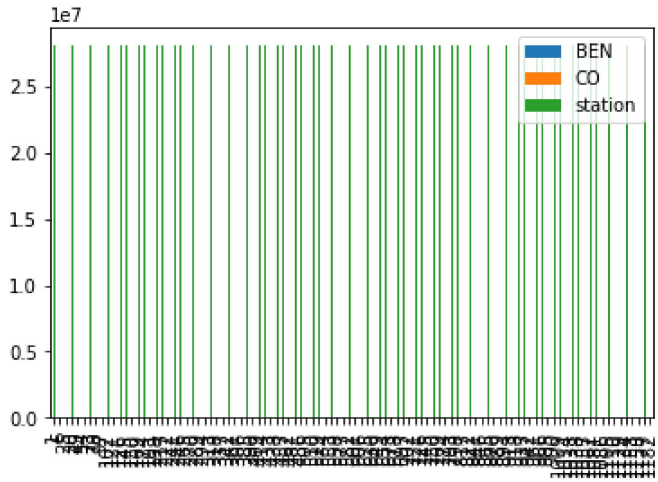
Out[227]: <AxesSubplot:>



In [228]:
```
1 b=data[0:100]
2
```

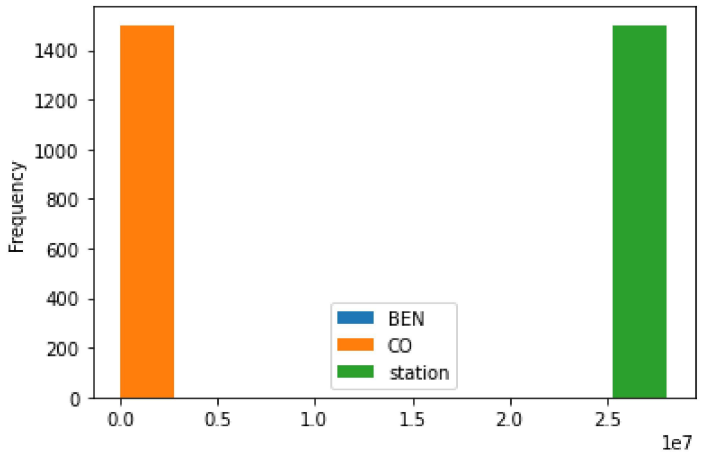
In [229]:
```
1 b.plot.bar()
```

Out[229]: <AxesSubplot:>



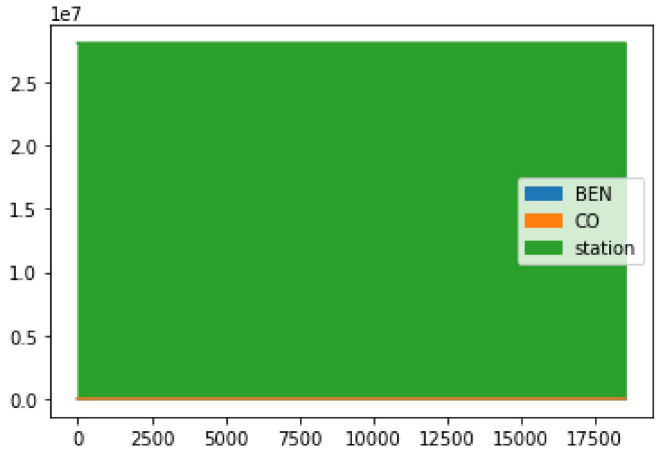
In [230]:
```
1 data.plot.hist()
2
```

Out[230]: <AxesSubplot:ylabel='Frequency'>



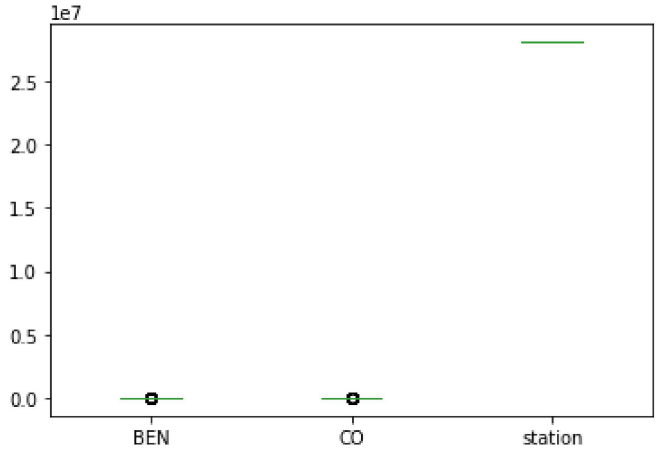
In [231]:
```
1 data.plot.area()
```

Out[231]: <AxesSubplot:>



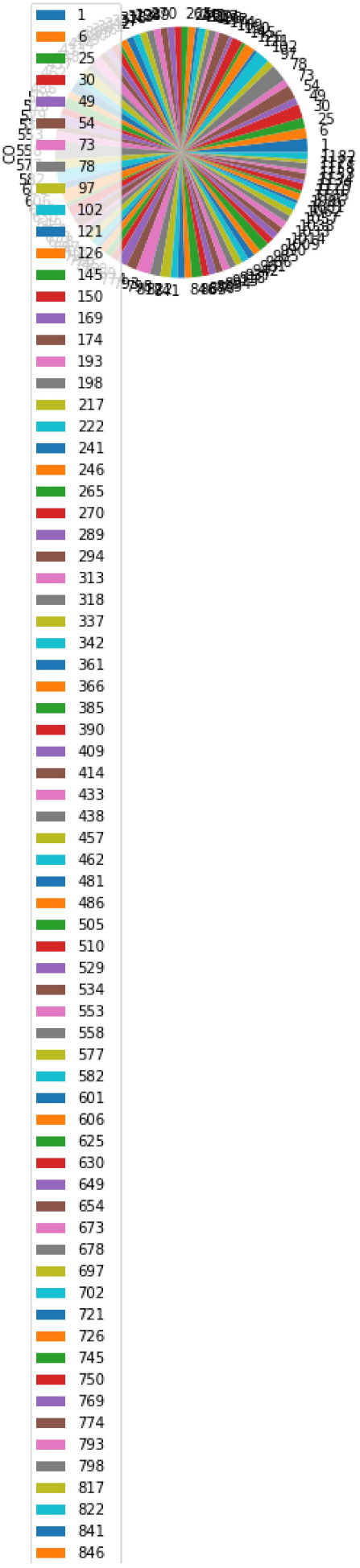
In [232]:
```
1 data.plot.box()
```

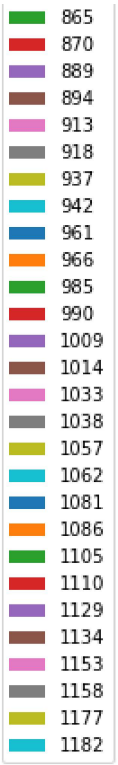
Out[232]: <AxesSubplot:>



In [234]:
```
1 b.plot.pie(y='CO' )
```

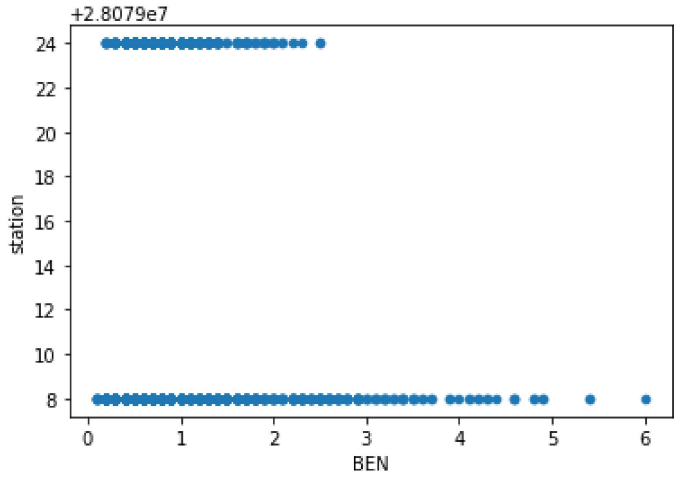
Out[234]:
```
<AxesSubplot:ylabel='CO'>
```


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
577
582
601
606
625
630
649
654
673
678
697
702
721
726
745
750
769
774
793
798
817
822
841
846



In [235]:
```
1 data.plot.scatter(x='BEN' ,y='station')
2
```

Out[235]: <AxesSubplot:xlabel='BEN', ylabel='station'>



In [236]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1500 entries, 1 to 18505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1500 non-null   object 
 1   BEN      1500 non-null   float64
 2   CO       1500 non-null   float64
 3   EBE      1500 non-null   float64
 4   NMHC     1500 non-null   float64
 5   NO       1500 non-null   float64
 6   NO_2     1500 non-null   float64
 7   O_3      1500 non-null   float64
 8   PM10     1500 non-null   float64
 9   PM25     1500 non-null   float64
 10  SO_2     1500 non-null   float64
 11  TCH      1500 non-null   float64
 12  TOL      1500 non-null   float64
 13  station  1500 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 175.8+ KB
```

In [237]:
```
1 df.describe()
2 
```

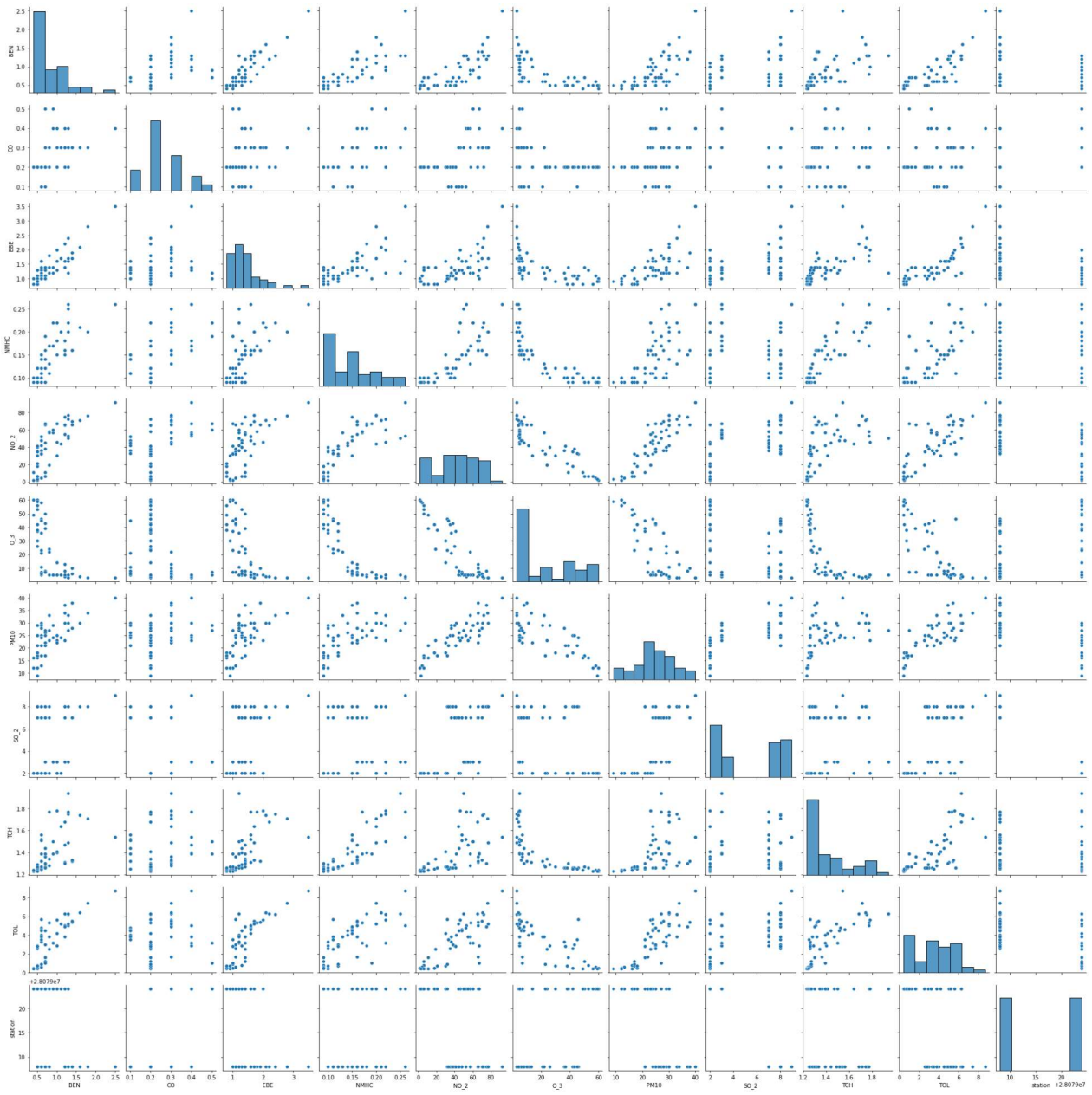
Out[237]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	15
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.0000	1500.000000	
mean	1.025200	0.329600	1.519600	0.170940	31.166000	45.547333	24.0540	21.181333	
std	0.804717	0.222374	1.198388	0.064675	45.640996	28.661439	19.3855	14.284647	
min	0.100000	0.100000	0.300000	0.070000	1.000000	1.000000	2.0000	2.000000	
25%	0.500000	0.200000	0.800000	0.130000	2.000000	23.000000	6.0000	12.000000	
50%	0.700000	0.300000	1.100000	0.155000	15.000000	43.000000	20.0000	18.000000	
75%	1.300000	0.400000	1.700000	0.200000	41.000000	63.000000	38.0000	28.000000	
max	6.000000	2.200000	10.400000	0.630000	415.000000	201.000000	95.0000	174.000000	

In [292]:
```
1 df1=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2',  'O_3',
2  'PM10', 'SO_2', 'TCH', 'TOL','station']]
```

In [293]:
```
1 sns.pairplot(df1[0:50])
```

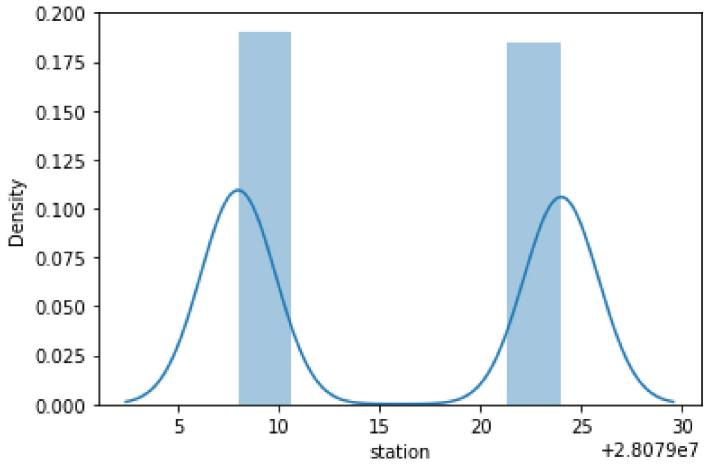
Out[293]: <seaborn.axisgrid.PairGrid at 0x27d44b1a340>



In [294]:
```
1 sns.distplot(df1['station'])
2
```

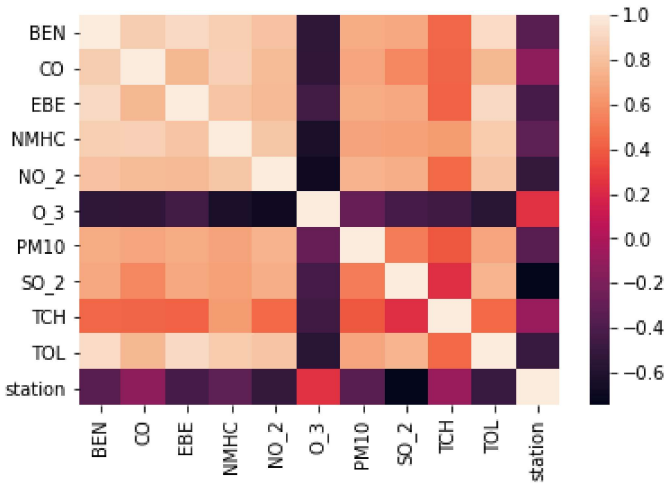
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function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

Out[294]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [295]:
```
1 sns.heatmap(df1.corr())
```

Out[295]: <AxesSubplot:>



In [296]:
```
1 x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2  'PM10', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
4
```

In [297]:
```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
3
```

In [298]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

```

Out[298]:
```
LinearRegression()
```

In [299]:
```
lr.intercept_
```

Out[299]:
```
28079021.83435971
```

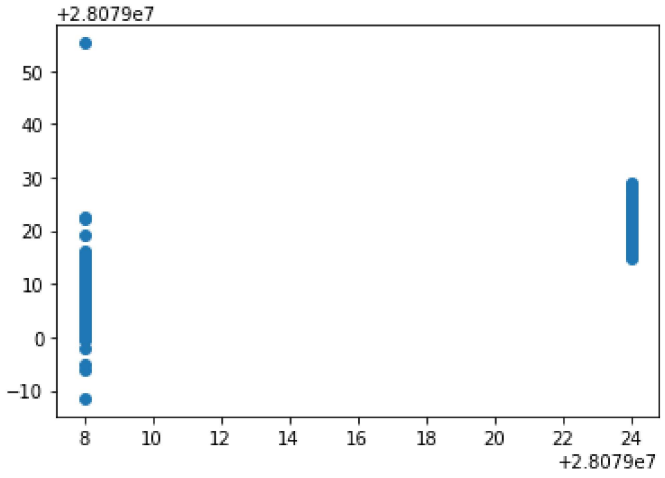
In [300]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[300]:

	Co-efficient
BEN	2.433054
CO	29.711507
EBE	-0.595272
NMHC	-14.402854
NO_2	-0.112989
O_3	-0.018623
PM10	-0.076463
SO_2	-1.448307
TCH	1.977278
TOL	-0.478526

In [301]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[301]:
```
<matplotlib.collections.PathCollection at 0x27d6a4976d0>
```



In [302]:
```
lr.score(x_test,y_test)

```

Out[302]:
```
0.6931217157901892
```

In [303]:
```
lr.score(x_train,y_train)

```

Out[303]: 0.803626173448059

In [304]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [305]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[305]: Ridge(alpha=10)

In [306]:
```
rr.score(x_test,y_test)

```

Out[306]: 0.7209903589483975

In [307]:
```
rr.score(x_train,y_train)

```

Out[307]: 0.7759567429663019

In [308]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[308]: Lasso(alpha=10)

In [309]:
```
la.score(x_train,y_train)

```

Out[309]: 0.40107086127919944

In [310]:
```
la.score(x_test,y_test)

```

Out[310]: 0.44182896260539795

In [311]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[311]: ElasticNet()

In [312]:
```
en.coef_

```

Out[312]:
```
array([ 0.53180933,  0.        ,  0.        ,  0.        , -0.04902033,
       -0.05556661,  0.04157072, -1.42373255,  0.        ,  0.10061326])
```

In [313]:
```
en.intercept_

```

Out[313]: 28079025.99769019

In [314]:
```
prediction=en.predict(x_test)

```

In [315]:
```
en.score(x_test,y_test)
```

Out[315]: 0.5985722320430774

In [316]:
```
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)))

```

```
4.365719159220656
25.629971586081407
5.06260521728501
```

In [317]:
```
from sklearn.linear_model import LogisticRegression

```

In [318]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
 'PM10',  'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']

```

In [319]:
```
feature_matrix.shape

```

Out[319]: (1500, 10)

In [320]:
```
target_vector.shape

```

Out[320]: (1500,)

In [321]:
```
from sklearn.preprocessing import StandardScaler
```

In [322]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [323]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[323]: LogisticRegression(max_iter=10000)

In [324]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]

```

In [325]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079008]
```

In [326]:
```
logr.classes_

```

Out[326]:
```
array([28079008, 28079024], dtype=int64)
```

In [327]:
```
logr.score(fs,target_vector)

```

Out[327]:
```
0.988
```

In [328]:
```
logr.predict_proba(observation)[0][0]

```

Out[328]:
```
1.0
```

In [329]:
```
logr.predict_proba(observation)

```

Out[329]:
```
array([[1.00000000e+00, 2.22414935e-29]])
```

In [330]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [331]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[331]:
```
RandomForestClassifier()
```

In [332]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [333]:
```
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[333]:
```
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')
```

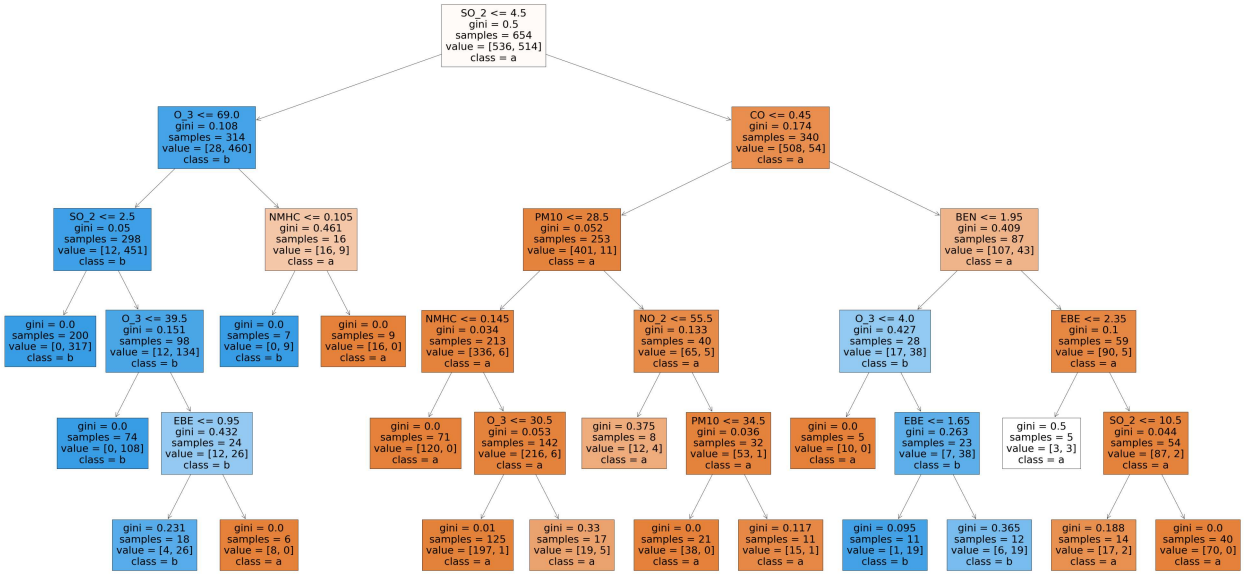
In [334]:
```
grid_search.best_score_
```

Out[334]:
```
0.9761904761904763
```

In [335]:
```
rfc_best=grid_search.best_estimator_
```

In [336]:
```
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']
```

Out[336]:
```
[Text(1767.0, 1993.2, 'SO_2 <= 4.5\ngini = 0.5\nsamples = 654\nvalue = [536, 514]\nclass =
a'),
 Text(744.0, 1630.8000000000002, 'O_3 <= 69.0\ngini = 0.108\nsamples = 314\nvalue = [28, 4
60]\nclass = b'),
 Text(372.0, 1268.4, 'SO_2 <= 2.5\ngini = 0.05\nsamples = 298\nvalue = [12, 451]\nclass =
b'),
 Text(186.0, 906.0, 'gini = 0.0\nsamples = 200\nvalue = [0, 317]\nclass = b'),
 Text(558.0, 906.0, 'O_3 <= 39.5\ngini = 0.151\nsamples = 98\nvalue = [12, 134]\nclass =
b'),
 Text(372.0, 543.5999999999999, 'gini = 0.0\nsamples = 74\nvalue = [0, 108]\nclass = b'),
 Text(744.0, 543.5999999999999, 'EBE <= 0.95\ngini = 0.432\nsamples = 24\nvalue = [12, 26]
\nclass = b'),
 Text(558.0, 181.19999999999982, 'gini = 0.231\nsamples = 18\nvalue = [4, 26]\nclass =
b'),
 Text(930.0, 181.19999999999982, 'gini = 0.0\nsamples = 6\nvalue = [8, 0]\nclass = a'),
 Text(1116.0, 1268.4, 'NMHC <= 0.105\ngini = 0.461\nsamples = 16\nvalue = [16, 9]\nclass =
a'),
 Text(930.0, 906.0, 'gini = 0.0\nsamples = 7\nvalue = [0, 9]\nclass = b'),
 Text(1302.0, 906.0, 'gini = 0.0\nsamples = 9\nvalue = [16, 0]\nclass = a'),
 Text(2790.0, 1630.8000000000002, 'CO <= 0.45\ngini = 0.174\nsamples = 340\nvalue = [508,
54]\nclass = a'),
 Text(2046.0, 1268.4, 'PM10 <= 28.5\ngini = 0.052\nsamples = 253\nvalue = [401, 11]\nclass
= a'),
 Text(1674.0, 906.0, 'NMHC <= 0.145\ngini = 0.034\nsamples = 213\nvalue = [336, 6]\nclass
= a'),
 Text(1488.0, 543.5999999999999, 'gini = 0.0\nsamples = 71\nvalue = [120, 0]\nclass = a'),
 Text(1860.0, 543.5999999999999, 'O_3 <= 30.5\ngini = 0.053\nsamples = 142\nvalue = [216,
6]\nclass = a'),
 Text(1674.0, 181.19999999999982, 'gini = 0.01\nsamples = 125\nvalue = [197, 1]\nclass =
a'),
 Text(2046.0, 181.19999999999982, 'gini = 0.33\nsamples = 17\nvalue = [19, 5]\nclass =
a'),
 Text(2418.0, 906.0, 'NO_2 <= 55.5\ngini = 0.133\nsamples = 40\nvalue = [65, 5]\nclass =
a'),
 Text(2232.0, 543.5999999999999, 'gini = 0.375\nsamples = 8\nvalue = [12, 4]\nclass = a'),
 Text(2604.0, 543.5999999999999, 'PM10 <= 34.5\ngini = 0.036\nsamples = 32\nvalue = [53,
1]\nclass = a'),
 Text(2418.0, 181.19999999999982, 'gini = 0.0\nsamples = 21\nvalue = [38, 0]\nclass = a'),
 Text(2790.0, 181.19999999999982, 'gini = 0.117\nsamples = 11\nvalue = [15, 1]\nclass =
a'),
 Text(3534.0, 1268.4, 'BEN <= 1.95\ngini = 0.409\nsamples = 87\nvalue = [107, 43]\nclass =
a'),
 Text(3162.0, 906.0, 'O_3 <= 4.0\ngini = 0.427\nsamples = 28\nvalue = [17, 38]\nclass =
b'),
 Text(2976.0, 543.5999999999999, 'gini = 0.0\nsamples = 5\nvalue = [10, 0]\nclass = a'),
 Text(3348.0, 543.5999999999999, 'EBE <= 1.65\ngini = 0.263\nsamples = 23\nvalue = [7, 38]
\nclass = b'),
 Text(3162.0, 181.19999999999982, 'gini = 0.095\nsamples = 11\nvalue = [1, 19]\nclass =
b'),
 Text(3534.0, 181.19999999999982, 'gini = 0.365\nsamples = 12\nvalue = [6, 19]\nclass =
b'),
 Text(3906.0, 906.0, 'EBE <= 2.35\ngini = 0.1\nsamples = 59\nvalue = [90, 5]\nclass = a'),
 Text(3720.0, 543.5999999999999, 'gini = 0.5\nsamples = 5\nvalue = [3, 3]\nclass = a'),
 Text(4092.0, 543.5999999999999, 'SO_2 <= 10.5\ngini = 0.044\nsamples = 54\nvalue = [87,
2]\nclass = a'),
 Text(3906.0, 181.19999999999982, 'gini = 0.188\nsamples = 14\nvalue = [17, 2]\nclass =
a'),
 Text(4278.0, 181.19999999999982, 'gini = 0.0\nsamples = 40\nvalue = [70, 0]\nclass = a')]
```



Conclusion

Linear Regression=0.7869626771020186

Ridge Regression=0.759783552235543

Lasso Regression=0.42381706905797656

ElasticNet Regression=0.5793840256257061

Logistic Regression=0.988

Random Forest=0.9780952380952381

Logistic Regression is Suitable for this Dataset

In [ ]:
```
1
```