Importing Libraries

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

Importing Datasets

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs(Dataset)\ma
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN	2
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	2
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0	2
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN	2
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN	2
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN	2
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN	2
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN	2

209496 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
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 [6]:
```
data=df[['BEN', 'TOL', 'TCH']]
data
```

Out[6]:

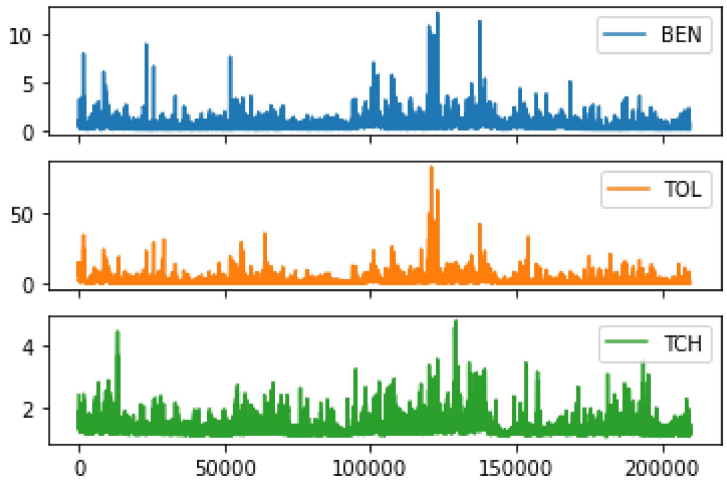
	BEN	TOL	TCH
1	3.1	14.4	2.44
6	0.7	5.0	1.35
25	2.7	15.0	2.30
30	0.7	5.0	1.35
49	1.7	10.7	1.95
...	...	...	...
209430	0.1	0.2	1.15
209449	0.6	1.9	1.48
209454	0.1	0.3	1.15
209473	0.6	1.9	1.50
209478	0.1	0.2	1.15

16932 rows × 3 columns

Line chart

In [7]:
```
data.plot.line(subplots=True)
```

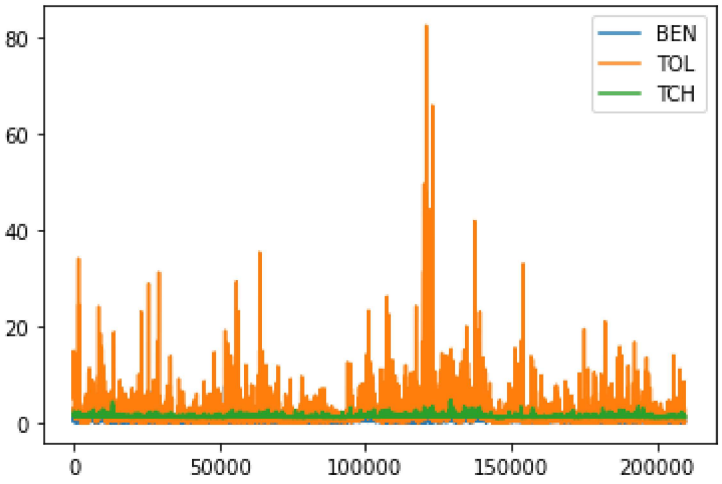
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:
```
data.plot.line()
```

Out[8]: <AxesSubplot:>

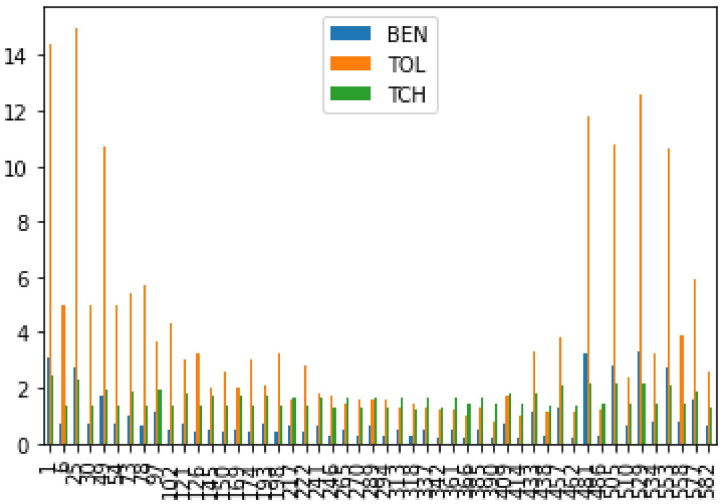


Bar chart

In [9]:
```
b=data[0:50]
```

In [10]:
```
b.plot.bar()
```

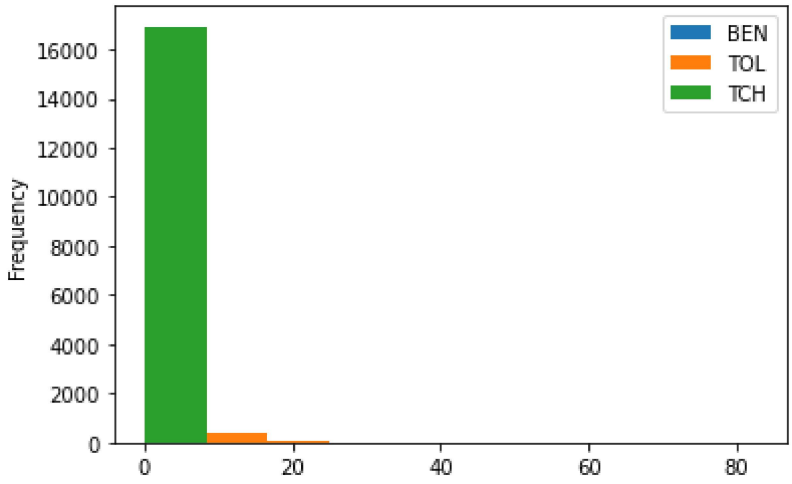
Out[10]: <AxesSubplot:>



Histogram

In [11]:
```
data.plot.hist()
```

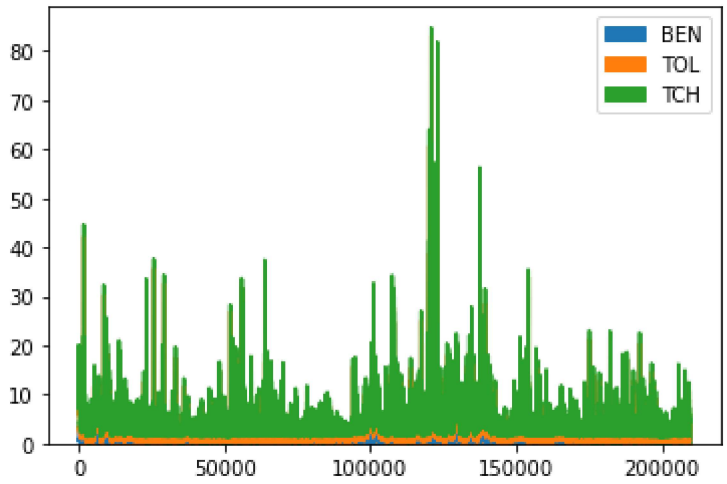
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:
```
data.plot.area()
```

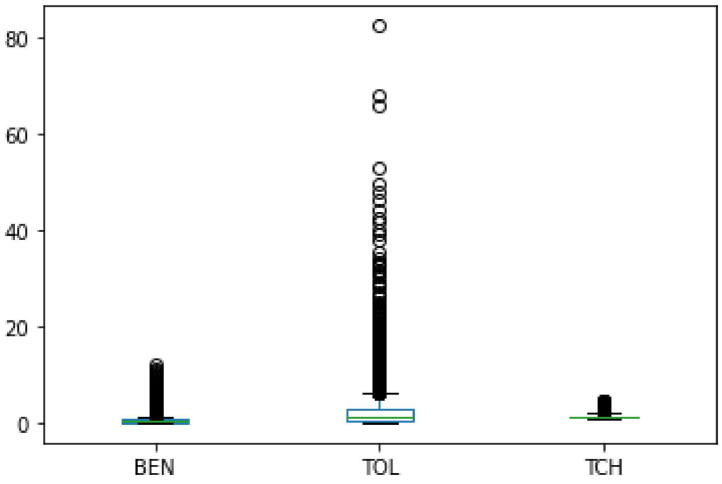
Out[12]: <AxesSubplot:>



Box chart

In [13]:
```
data.plot.box()
```

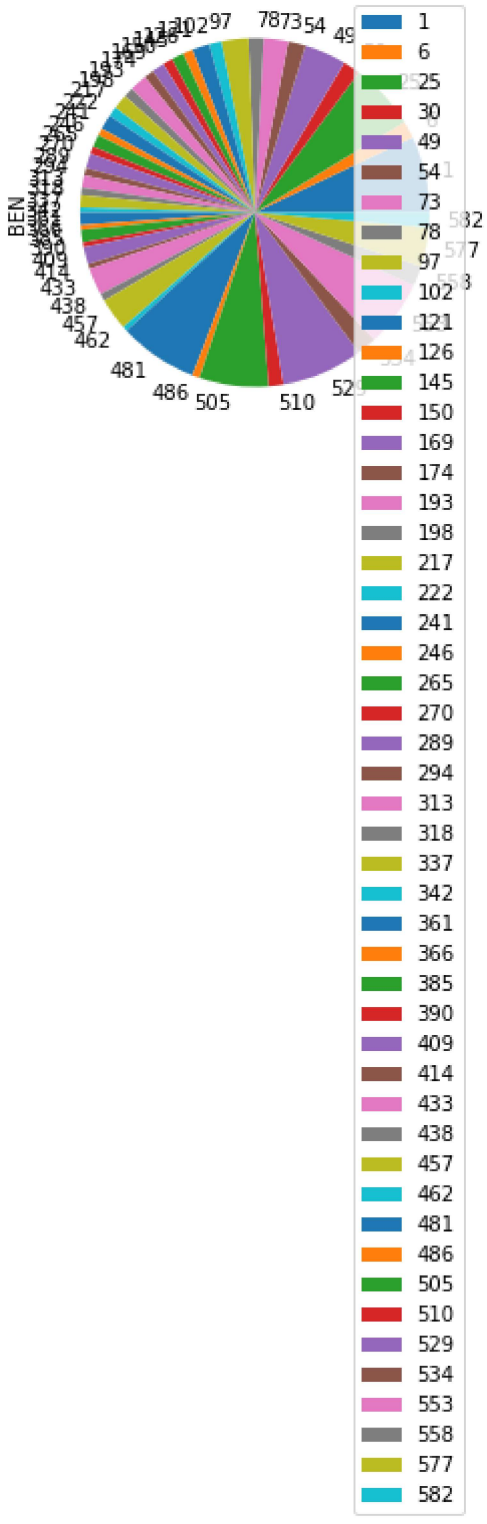
Out[13]: <AxesSubplot:>



Pie chart

In [14]:
```
b.plot.pie(y='BEN' )
```

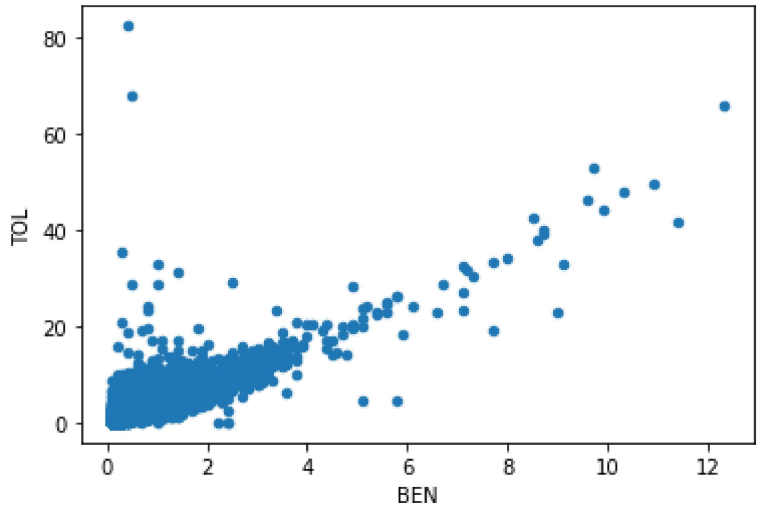
Out[14]: <AxesSubplot:ylabel='BEN'>



Scatter chart

In [15]:
```
data.plot.scatter(x='BEN' ,y='TOL')
```

Out[15]: <AxesSubplot:xlabel='BEN', ylabel='TOL'>



In [16]:
```
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 [17]:
```
df.describe()
```

Out[17]:

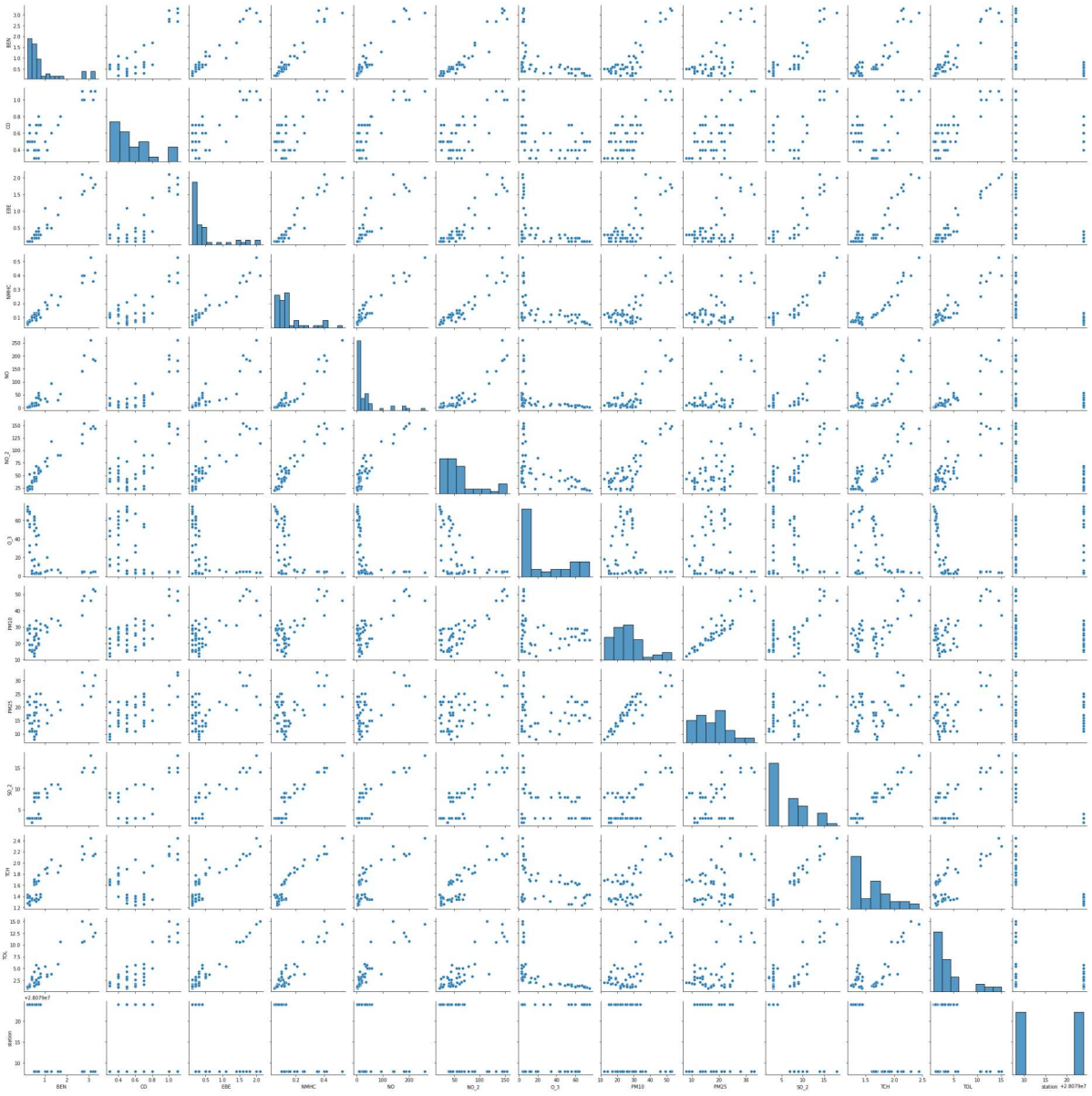
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	169
mean	0.537970	0.349941	0.298955	0.099913	20.815734	39.373376	
std	0.599479	0.203807	0.450204	0.079850	40.986063	31.170307	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.200000	0.200000	0.100000	0.050000	1.000000	14.000000	
50%	0.400000	0.300000	0.200000	0.090000	7.000000	34.000000	
75%	0.700000	0.400000	0.300000	0.120000	23.000000	58.000000	
max	12.300000	4.500000	13.500000	2.210000	829.000000	319.000000	1

In [18]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25
       'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [19]:
```
sns.pairplot(df1[0:50])
```

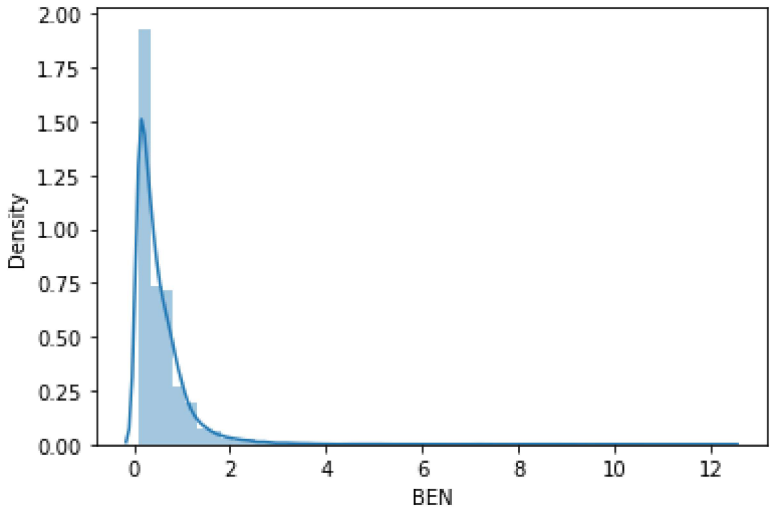
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1b232bba220>
```



In [20]:
```
sns.distplot(df1['BEN'])
```

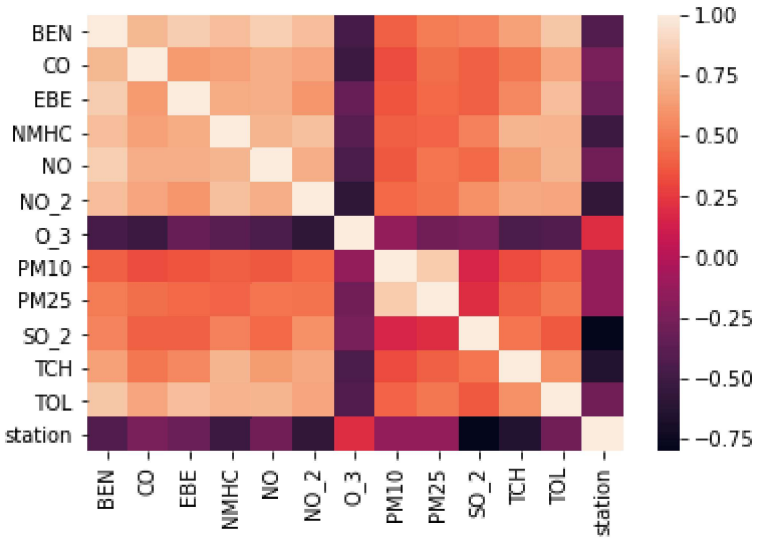
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



In [21]:
```
sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [23]:
```
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)
```

Linear Regression

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
lr.intercept_
```

Out[25]:
```
28079041.999899916
```

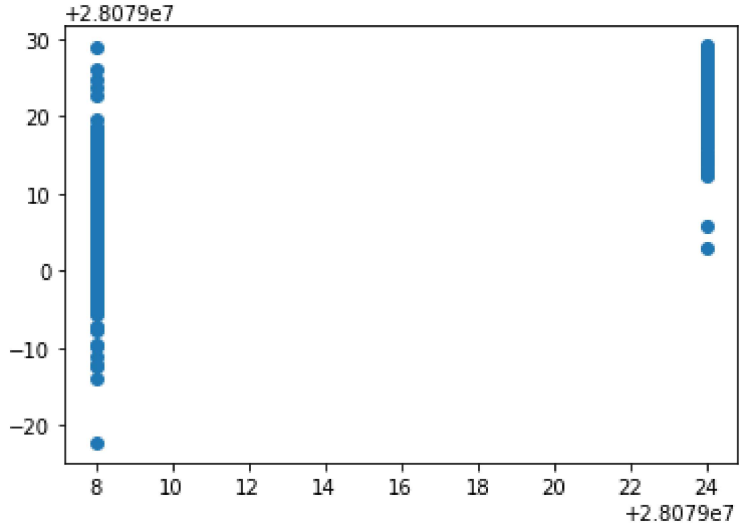
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-1.807121
CO	4.467319
EBE	0.581989
NMHC	1.262170
NO	0.067188
NO_2	-0.067192
O_3	-0.024761
PM10	-0.012139
PM25	0.103008
SO_2	-0.803779
TCH	-13.998236
TOL	0.205204

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1b23ec55f70>
```



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```

Out[28]:
```
0.826130159659848
```

In [29]:
```
lr.score(x_train,y_train)
```

Out[29]:
```
0.8286608817866095
```

Ridge and Lasso

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[31]:
```
Ridge(alpha=10)
```

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```

Out[32]:
```
0.8256727930423138
```

In [33]:
```
rr.score(x_train,y_train)
```

Out[33]:
```
0.828577461543129
```

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]:
```
Lasso(alpha=10)
```

In [35]:
```
la.score(x_test,y_test)
```

Out[35]:
```
0.6431475578851213
```

Accuracy(Lasso)

In [36]:
```
la.score(x_train,y_train)
```

Out[36]:
```
0.6467262929917432
```

Accuracy(Elastic Net)

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[37]:
```
ElasticNet()
```

In [38]:
```
en.coef_
```

Out[38]:
```
array([-0.        ,  0.        , -0.        , -0.        ,  0.048158  ,
       -0.10696444, -0.02103236,  0.00315771,  0.04814422, -0.86268439,
       -0.03113773,  0.        ])
```

In [39]:
```
en.intercept_
```

Out[39]:
```
28079026.225688875
```

In [40]:
```
prediction=en.predict(x_test)
```

In [41]:
```
en.score(x_test,y_test)
```

Out[41]:
```
0.7049603330703134
```

Evaluation Metrics

In [42]:
```
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)))
```

```
3.352243386644606
18.8716480631203
4.344151017531538
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [45]:
```
feature_matrix.shape
```

Out[45]:
```
(16932, 10)
```

In [46]:
```
target_vector.shape
```

Out[46]:
```
(16932,)
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [52]:
```
logr.classes_
```

Out[52]:
```
array([28079008, 28079024], dtype=int64)
```

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.9923812898653437
```

In [54]:
```
logr.predict_proba(observation)[0][0]
```

Out[54]:
```
1.0
```

In [55]:
```
logr.predict_proba(observation)
```

Out[55]:
```
array([[1.0000000e+00, 1.6336121e-46]])
```

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]:
```
RandomForestClassifier()
```

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [59]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
grid_search.fit(x_train,y_train)
```

Out[59]:
```
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 [60]:
```
grid_search.best_score_
```

Out[60]:
```
0.9946000674991562
```

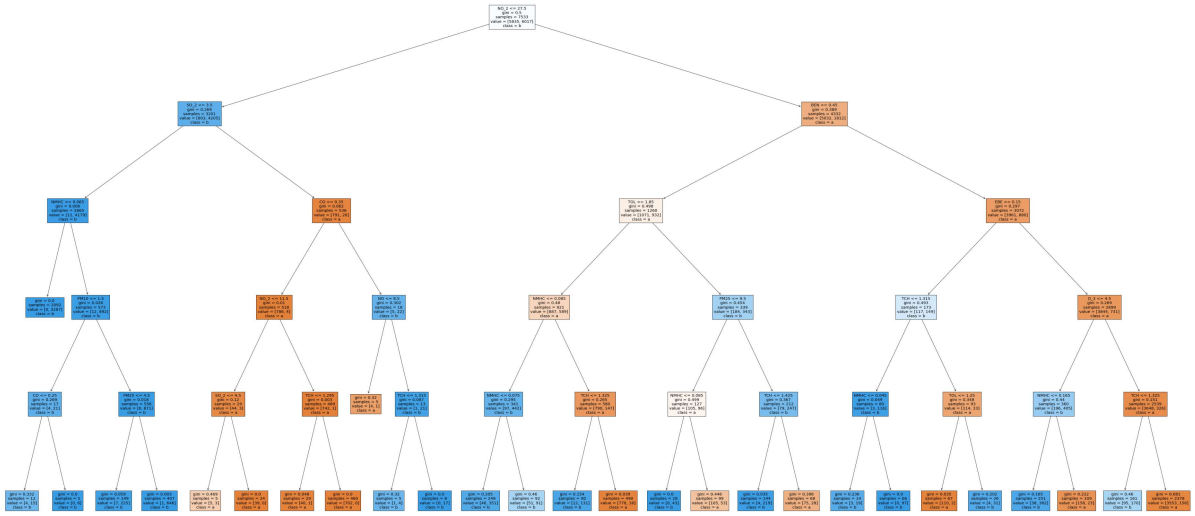
In [61]:
```
rfc_best=grid_search.best_estimator_
```

In [62]:
```
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'
```

Out[62]:
```
[Text(1920.8076923076922, 1993.2, 'NO_2 <= 27.5\ngini = 0.5\nsamples = 7533\n
value = [5835, 6017]\nclass = b'),
 Text(751.1538461538461, 1630.8000000000002, 'SO_2 <= 3.5\ngini = 0.269\nsamp
les = 3201\nvalue = [803, 4205]\nclass = b'),
 Text(257.53846153846155, 1268.4, 'NMHC <= 0.065\ngini = 0.006\nsamples = 266
5\nvalue = [12, 4179]\nclass = b'),
 Text(171.69230769230768, 906.0, 'gini = 0.0\nsamples = 2092\nvalue = [0, 328
7]\nclass = b'),
 Text(343.38461538461536, 906.0, 'PM10 <= 1.5\ngini = 0.026\nsamples = 573\nv
alue = [12, 892]\nclass = b'),
 Text(171.69230769230768, 543.5999999999999, 'CO <= 0.25\ngini = 0.269\nsampl
es = 17\nvalue = [4, 21]\nclass = b'),
 Text(85.84615384615384, 181.19999999999982, 'gini = 0.332\nsamples = 12\nval
ue = [4, 15]\nclass = b'),
 Text(257.53846153846155, 181.19999999999982, 'gini = 0.0\nsamples = 5\nvalue
= [0, 6]\nclass = b'),
 Text(515.0769230769231, 543.5999999999999, 'PM25 <= 4.5\ngini = 0.018\nsampl
es = 556\nvalue = [8, 871]\nclass = b'),
 Text(429.23076923076917, 181.19999999999982, 'gini = 0.059\nsamples = 149\nv
alue = [7, 225]\nclass = b'),
 Text(600.9230769230769, 181.19999999999982, 'gini = 0.003\nsamples = 407\nva
lue = [1, 646]\nclass = b'),
 Text(1244.7692307692307, 1268.4, 'CO <= 0.35\ngini = 0.062\nsamples = 536\nv
alue = [791, 26]\nclass = a'),
 Text(1030.1538461538462, 906.0, 'NO_2 <= 11.5\ngini = 0.01\nsamples = 518\nv
alue = [786, 4]\nclass = a'),
 Text(858.4615384615383, 543.5999999999999, 'SO_2 <= 4.5\ngini = 0.12\nsample
s = 29\nvalue = [44, 3]\nclass = a'),
 Text(772.6153846153845, 181.19999999999982, 'gini = 0.469\nsamples = 5\nvalu
e = [5, 3]\nclass = a'),
 Text(944.3076923076923, 181.19999999999982, 'gini = 0.0\nsamples = 24\nvalue
= [39, 0]\nclass = a'),
 Text(1201.8461538461538, 543.5999999999999, 'TCH <= 1.295\ngini = 0.003\nsam
ples = 489\nvalue = [742, 1]\nclass = a'),
 Text(1116.0, 181.19999999999982, 'gini = 0.048\nsamples = 29\nvalue = [40,
1]\nclass = a'),
 Text(1287.6923076923076, 181.19999999999982, 'gini = 0.0\nsamples = 460\nval
ue = [702, 0]\nclass = a'),
 Text(1459.3846153846152, 906.0, 'NO <= 8.5\ngini = 0.302\nsamples = 18\nvalu
e = [5, 22]\nclass = b'),
 Text(1373.5384615384614, 543.5999999999999, 'gini = 0.32\nsamples = 5\nvalue
= [4, 1]\nclass = a'),
 Text(1545.230769230769, 543.5999999999999, 'TCH <= 1.315\ngini = 0.087\nsamp
les = 13\nvalue = [1, 21]\nclass = b'),
 Text(1459.3846153846152, 181.19999999999982, 'gini = 0.32\nsamples = 5\nvalu
e = [1, 4]\nclass = b'),
 Text(1631.0769230769229, 181.19999999999982, 'gini = 0.0\nsamples = 8\nvalue
= [0, 17]\nclass = b'),
 Text(3090.461538461538, 1630.8000000000002, 'BEN <= 0.45\ngini = 0.389\nsamp
les = 4332\nvalue = [5032, 1812]\nclass = a'),
 Text(2403.6923076923076, 1268.4, 'TOL <= 1.85\ngini = 0.498\nsamples = 1260
\nvalue = [1071, 932]\nclass = a'),
 Text(2060.3076923076924, 906.0, 'NMHC <= 0.085\ngini = 0.48\nsamples = 921\n
value = [887, 589]\nclass = a'),
 Text(1888.6153846153845, 543.5999999999999, 'NMHC <= 0.075\ngini = 0.295\nsa
mples = 341\nvalue = [97, 442]\nclass = b'),
 Text(1802.7692307692307, 181.19999999999982, 'gini = 0.205\nsamples = 249\nv
```

```
alue = [46, 351]\nclass = b'),
 Text(1974.4615384615383, 181.19999999999982, 'gini = 0.46\nsamples = 92\nval
ue = [51, 91]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'TCH <= 1.325\ngini = 0.265\nsamples = 580\n
value = [790, 147]\nclass = a'),
 Text(2146.153846153846, 181.19999999999982, 'gini = 0.154\nsamples = 90\nval
ue = [12, 131]\nclass = b'),
 Text(2317.846153846154, 181.19999999999982, 'gini = 0.039\nsamples = 490\nva
lue = [778, 16]\nclass = a'),
 Text(2747.076923076923, 906.0, 'PM25 <= 9.5\ngini = 0.454\nsamples = 339\nva
lue = [184, 343]\nclass = b'),
 Text(2575.3846153846152, 543.5999999999999, 'NMHC <= 0.085\ngini = 0.499\nsa
mples = 127\nvalue = [105, 96]\nclass = a'),
 Text(2489.5384615384614, 181.19999999999982, 'gini = 0.0\nsamples = 28\nvalu
e = [0, 43]\nclass = b'),
 Text(2661.230769230769, 181.19999999999982, 'gini = 0.446\nsamples = 99\nval
ue = [105, 53]\nclass = a'),
 Text(2918.7692307692305, 543.5999999999999, 'TCH <= 1.425\ngini = 0.367\nsam
ples = 212\nvalue = [79, 247]\nclass = b'),
 Text(2832.9230769230767, 181.19999999999982, 'gini = 0.035\nsamples = 144\nv
alue = [4, 219]\nclass = b'),
 Text(3004.6153846153843, 181.19999999999982, 'gini = 0.396\nsamples = 68\nva
lue = [75, 28]\nclass = a'),
 Text(3777.230769230769, 1268.4, 'EBE <= 0.15\ngini = 0.297\nsamples = 3072\n
value = [3961, 880]\nclass = a'),
 Text(3433.8461538461534, 906.0, 'TCH <= 1.315\ngini = 0.493\nsamples = 173\n
value = [117, 149]\nclass = b'),
 Text(3262.1538461538457, 543.5999999999999, 'NMHC <= 0.045\ngini = 0.049\nsa
mples = 80\nvalue = [3, 116]\nclass = b'),
 Text(3176.307692307692, 181.19999999999982, 'gini = 0.236\nsamples = 14\nval
ue = [3, 19]\nclass = b'),
 Text(3347.9999999999995, 181.19999999999982, 'gini = 0.0\nsamples = 66\nvalu
e = [0, 97]\nclass = b'),
 Text(3605.5384615384614, 543.5999999999999, 'TOL <= 1.25\ngini = 0.348\nsamp
les = 93\nvalue = [114, 33]\nclass = a'),
 Text(3519.6923076923076, 181.19999999999982, 'gini = 0.035\nsamples = 67\nva
lue = [110, 2]\nclass = a'),
 Text(3691.3846153846152, 181.19999999999982, 'gini = 0.202\nsamples = 26\nva
lue = [4, 31]\nclass = b'),
 Text(4120.615384615385, 906.0, 'O_3 <= 4.5\ngini = 0.269\nsamples = 2899\nva
lue = [3844, 731]\nclass = a'),
 Text(3948.9230769230767, 543.5999999999999, 'NMHC <= 0.165\ngini = 0.44\nsam
ples = 360\nvalue = [196, 405]\nclass = b'),
 Text(3863.076923076923, 181.19999999999982, 'gini = 0.165\nsamples = 251\nva
lue = [38, 382]\nclass = b'),
 Text(4034.7692307692305, 181.19999999999982, 'gini = 0.222\nsamples = 109\nv
alue = [158, 23]\nclass = a'),
 Text(4292.307692307692, 543.5999999999999, 'TCH <= 1.325\ngini = 0.151\nsamp
les = 2539\nvalue = [3648, 326]\nclass = a'),
 Text(4206.461538461538, 181.19999999999982, 'gini = 0.46\nsamples = 161\nval
ue = [95, 170]\nclass = b'),
 Text(4378.153846153846, 181.19999999999982, 'gini = 0.081\nsamples = 2378\nv
alue = [3553, 156]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.8286608817866095

Ridge Regression:0.828577461543129

Lasso Regression:0.6467262929917432

ElasticNet Regression:0.7049603330703134

Logistic Regression:0.9923812898653437

Random Forest:0.9946000674991562

Random Forest is suitable for this dataset