Importing Libraries

In [182]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [244]:
```
df=pd.read_csv(r"C:\Users\user\Desktop\csvs_per_year\csvs_per_year\madrid_2006.
df
```

Out[244]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490

230568 rows × 17 columns

Data Cleaning and Data Preprocessing

In [245]:
```
df=df.dropna()
```

In [246]:
```
df.columns
```

Out[246]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [247]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24758 non-null  object 
 1   BEN      24758 non-null  float64
 2   CO       24758 non-null  float64
 3   EBE      24758 non-null  float64
 4   MXY      24758 non-null  float64
 5   NMHC     24758 non-null  float64
 6   NO_2     24758 non-null  float64
 7   NOx      24758 non-null  float64
 8   OXY      24758 non-null  float64
 9   O_3      24758 non-null  float64
 10  PM10     24758 non-null  float64
 11  PM25     24758 non-null  float64
 12  PXY      24758 non-null  float64
 13  SO_2     24758 non-null  float64
 14  TCH      24758 non-null  float64
 15  TOL      24758 non-null  float64
 16  station  24758 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

In [248]:
```
data=df[['CO' ,'station']]
data
```

Out[248]:

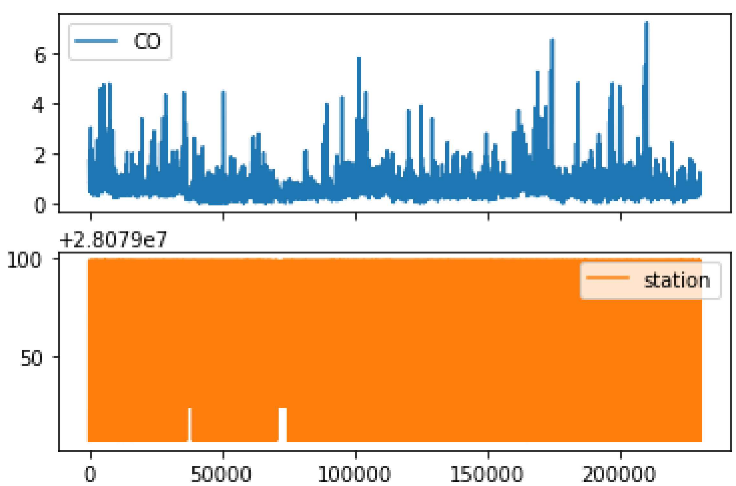
	CO	station
5	1.69	28079006
22	0.79	28079024
25	1.47	28079099
31	0.85	28079006
48	0.79	28079024
...	...	...
230538	0.40	28079024
230541	0.94	28079099
230547	1.06	28079006
230564	0.32	28079024
230567	0.74	28079099

24758 rows × 2 columns

Line chart

In [249]:
```
data.plot.line(subplots=True)
```

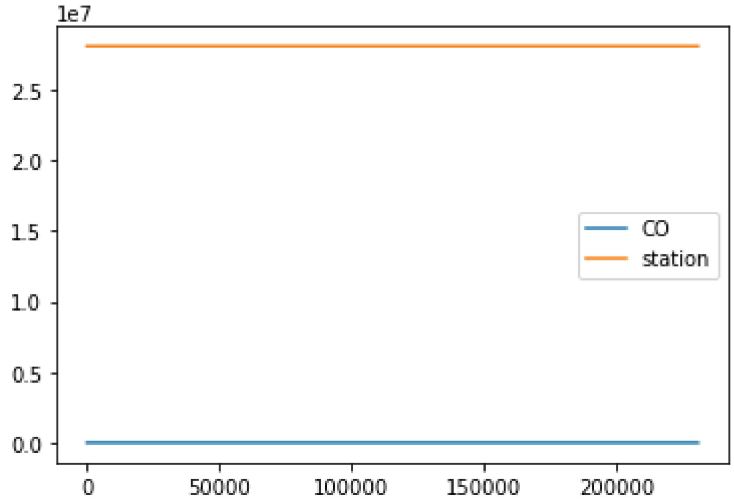
Out[249]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

In [250]:
```
data.plot.line()
```

Out[250]: <AxesSubplot:>

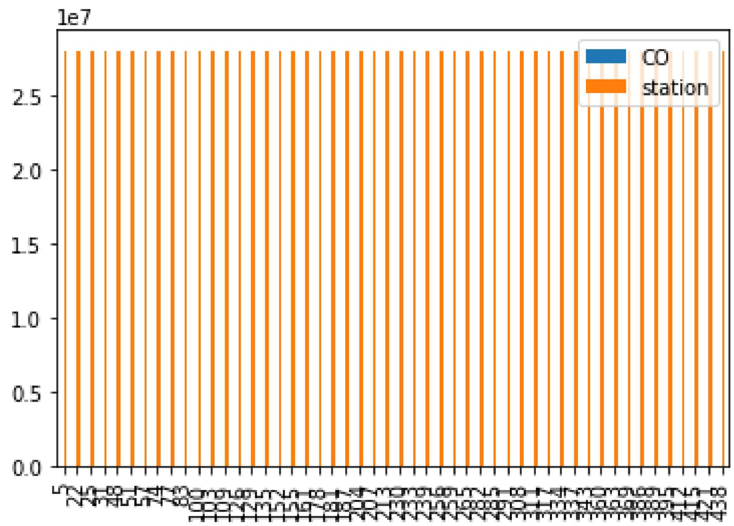


Bar chart

In [251]:
```
b=data[0:50]
```

In [252]:
```
b.plot.bar()
```

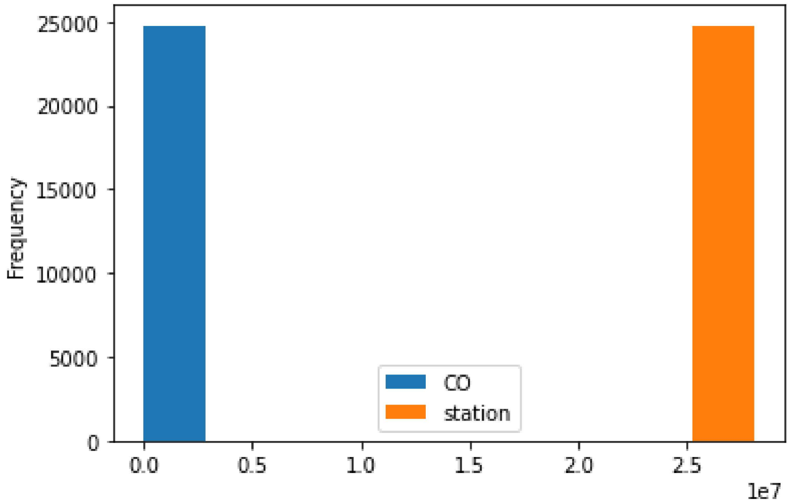
Out[252]: <AxesSubplot:>



Histogram

In [253]:
```
data.plot.hist()
```

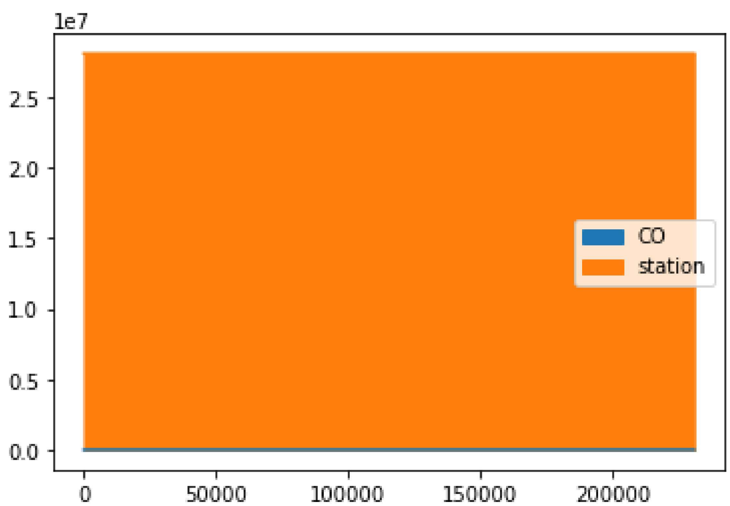
Out[253]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [254]:
```
data.plot.area()
```

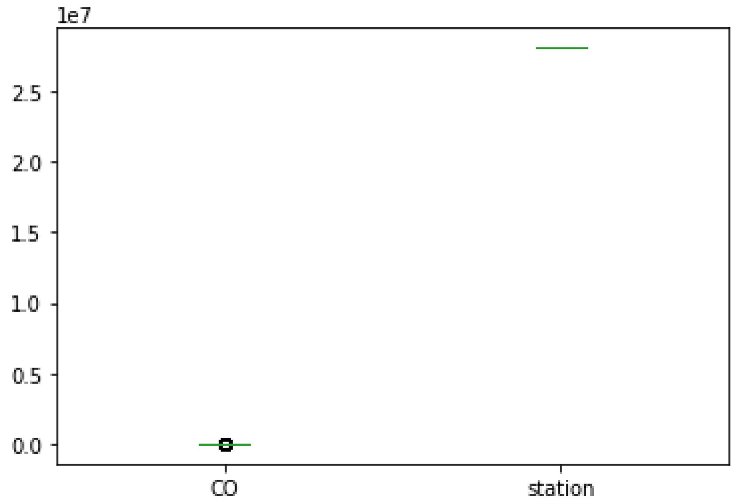
Out[254]: <AxesSubplot:>



Box chart

In [255]:
```
data.plot.box()
```

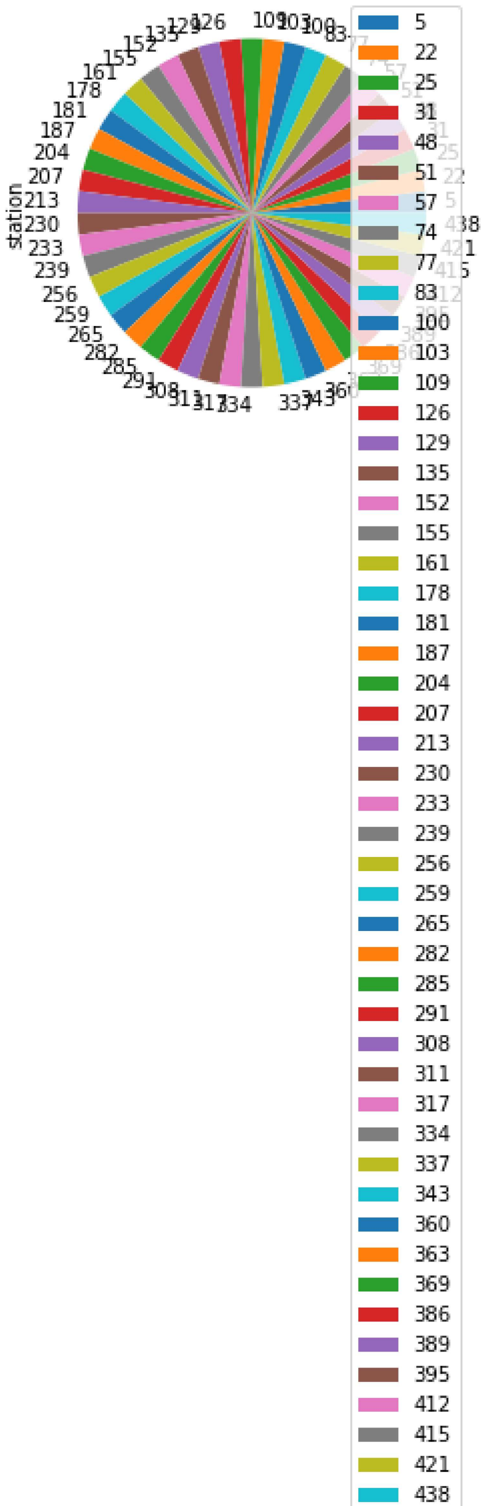
Out[255]: <AxesSubplot:>



Pie chart

In [256]:
```
b.plot.pie(y='station' )
```

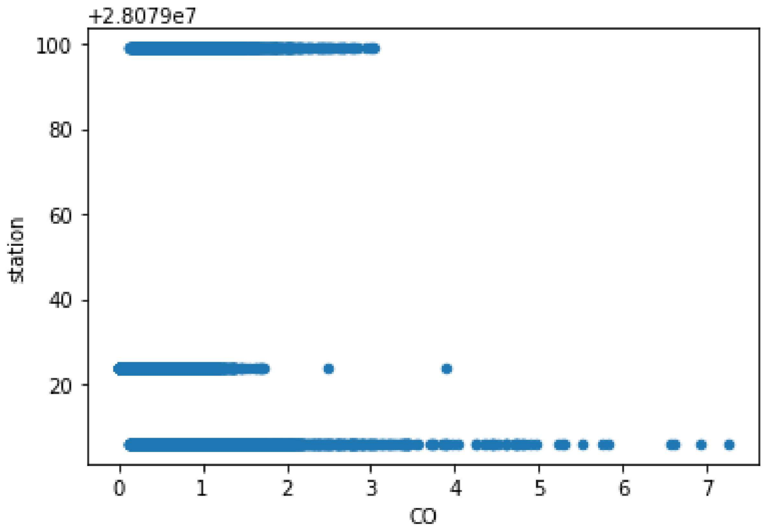
Out[256]: <AxesSubplot:ylabel='station'>



Scatter chart

In [257]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[257]: <AxesSubplot:xlabel='CO', ylabel='station'>



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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24758 non-null  object 
 1   BEN      24758 non-null  float64
 2   CO       24758 non-null  float64
 3   EBE      24758 non-null  float64
 4   MXY      24758 non-null  float64
 5   NMHC     24758 non-null  float64
 6   NO_2     24758 non-null  float64
 7   NOx      24758 non-null  float64
 8   OXY      24758 non-null  float64
 9   O_3      24758 non-null  float64
 10  PM10     24758 non-null  float64
 11  PM25     24758 non-null  float64
 12  PXY      24758 non-null  float64
 13  SO_2     24758 non-null  float64
 14  TCH      24758 non-null  float64
 15  TOL      24758 non-null  float64
 16  station  24758 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

In [259]:
```
df.describe()
```

Out[259]:

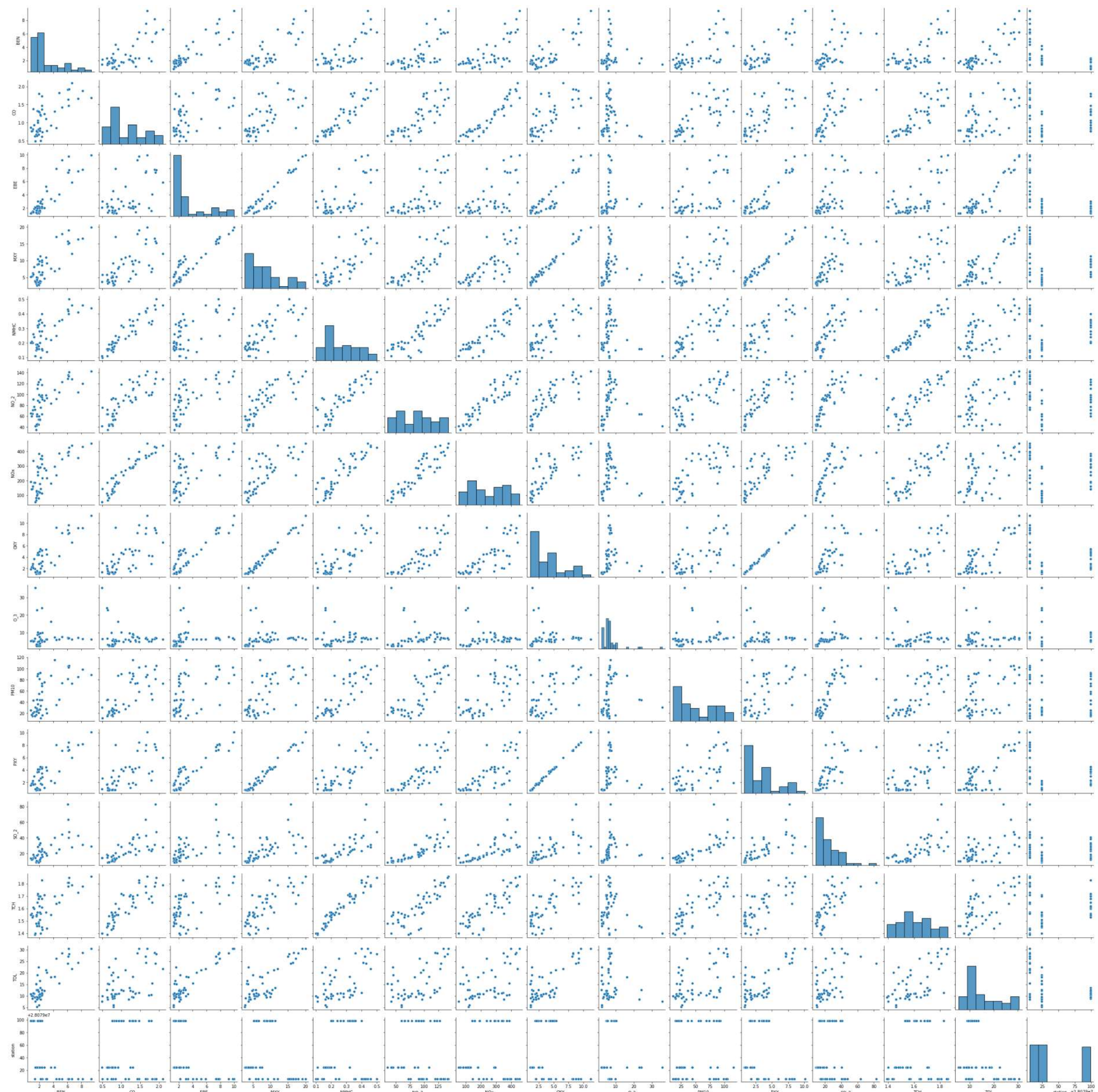
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	247
mean	1.350624	0.600713	1.824534	3.835034	0.176546	58.333481	1
std	1.541636	0.419048	1.868939	4.069036	0.126683	40.529382	1
min	0.110000	0.000000	0.170000	0.150000	0.000000	1.680000	
25%	0.450000	0.360000	0.810000	1.060000	0.100000	28.450001	
50%	0.850000	0.500000	1.130000	2.500000	0.150000	52.959999	
75%	1.680000	0.720000	2.160000	5.090000	0.220000	79.347498	1
max	45.430000	7.250000	57.799999	66.900002	2.020000	461.299988	16

In [260]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [261]:
```
sns.pairplot(df1[0:50])
```

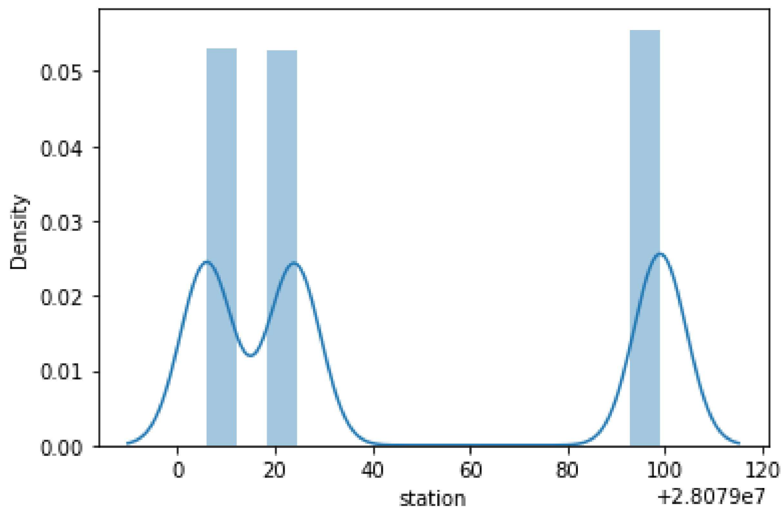
Out[261]: <seaborn.axisgrid.PairGrid at 0x1cd3fd1d250>



In [262]:
```
sns.distplot(df1['station'])
```

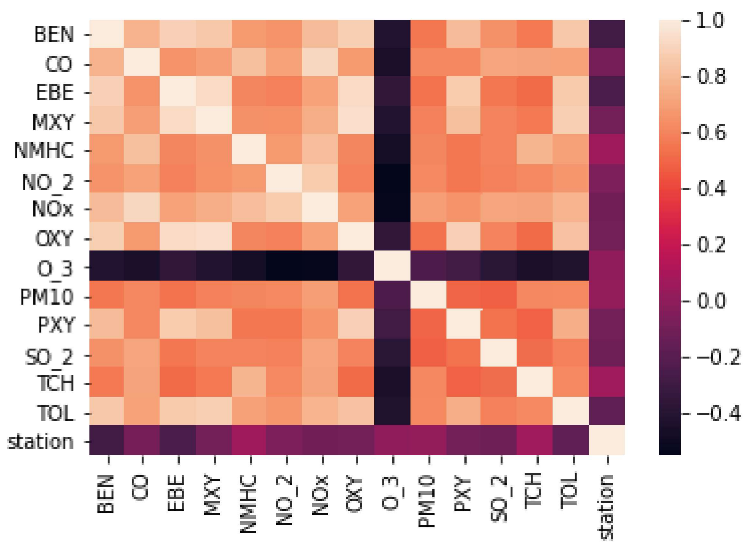
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[262]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [263]:
```
sns.heatmap(df1.corr())
```

Out[263]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [264]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [265]:
```
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 [266]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[266]:
```
LinearRegression()
```

In [267]:
```
lr.intercept_
```

Out[267]:
```
28079016.076544955
```

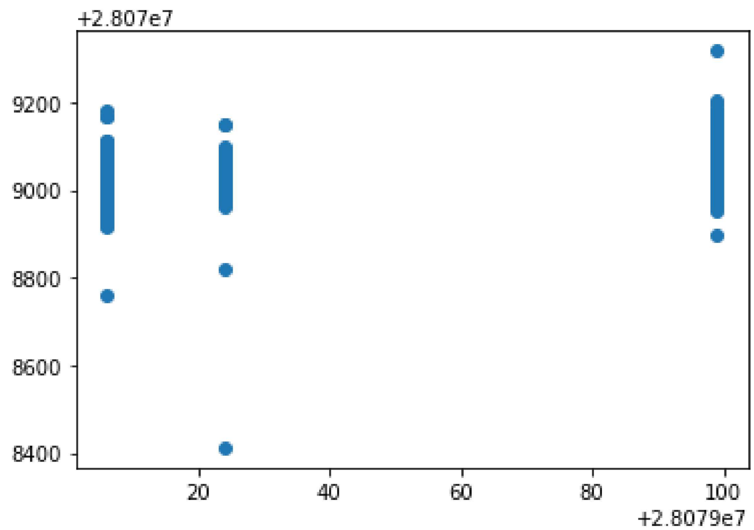
In [268]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[268]:

	Co-efficient
BEN	-19.319885
CO	-10.507243
EBE	-22.394434
MXY	4.401645
NMHC	127.460987
NO_2	-0.039672
NOx	0.005303
OXY	15.813423
O_3	-0.053057
PM10	0.145343
PXY	5.295290
SO_2	-0.666623
TCH	21.728755
TOL	-0.543613

In [269]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[269]: <matplotlib.collections.PathCollection at 0x1cd2b2097f0>



ACCURACY

In [270]:
```
lr.score(x_test,y_test)
```

Out[270]: 0.37949271754913305

In [271]:
```
lr.score(x_train,y_train)
```

Out[271]: 0.3988690718023816

Ridge and Lasso

In [272]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [273]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[273]: Ridge(alpha=10)

Accuracy(Ridge)

In [274]:
```
rr.score(x_test,y_test)
```
Out[274]: 0.37848656387129054

In [275]:
```
rr.score(x_train,y_train)
```
Out[275]: 0.3982325289376347

In [276]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[276]: Lasso(alpha=10)

In [277]:
```
la.score(x_train,y_train)
```
Out[277]: 0.060198845996248807

Accuracy(Lasso)

In [278]:
```
la.score(x_test,y_test)
```
Out[278]: 0.06197759766733968

In [279]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[279]: ElasticNet()

In [280]:
```
en.coef_
```
Out[280]:
```
array([-8.72324106,  0.        , -8.93980792,  3.42386809,  0.43797286,
       -0.02487847,  0.01421054,  3.51377261, -0.12351481,  0.30466397,
        2.37345184, -0.45507151,  0.60599212, -1.01345425])
```

In [281]:
```
en.intercept_
```
Out[281]: 28079052.271592375

In [282]:
```
prediction=en.predict(x_test)
```

In [283]:
```
en.score(x_test,y_test)
```
Out[283]: 0.23668401711102205

Evaluation Metrics

In [284]:
```
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)))
```

```
32.100659269189556
1253.4459470112943
35.404038569226735
```

Logistic Regression

In [285]:
```
from sklearn.linear_model import LogisticRegression
```

In [286]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [287]:
```
feature_matrix.shape
```

Out[287]:
```
(24758, 14)
```

In [288]:
```
target_vector.shape
```

Out[288]:
```
(24758,)
```

In [289]:
```
from sklearn.preprocessing import StandardScaler
```

In [290]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [291]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[291]:
```
LogisticRegression(max_iter=10000)
```

In [292]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [293]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [294]:
```
logr.classes_
```

Out[294]:
```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [295]:
```
logr.score(fs,target_vector)
```

Out[295]:
```
0.8741416915744405
```

In [296]:
```
logr.predict_proba(observation)[0][0]
```

Out[296]:
```
3.5557727473608076e-15
```

In [297]:
```
logr.predict_proba(observation)
```

Out[297]:
```
array([[3.55577275e-15, 7.80743173e-29, 1.00000000e+00]])
```

Random Forest

In [298]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [299]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[299]:
```
RandomForestClassifier()
```

In [300]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [301]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[301]:
```
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 [302]:
```
grid_search.best_score_
```

Out[302]:
```
0.8765147143681478
```

In [303]:
```
rfc_best=grid_search.best_estimator_
```

In [304]:
```
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',
```

```
 Text(906.75, 181.19999999999982, 'gini = 0.155\nsamples = 40\nvalue = [54,
5, 0]\nclass = a'),
 Text(1046.25, 181.19999999999982, 'gini = 0.62\nsamples = 5\nvalue = [5,
2, 3]\nclass = a'),
 Text(1674.0, 1268.4, 'EBE <= 0.535\ngini = 0.581\nsamples = 3320\nvalue =
[493, 2385, 2374]\nclass = b'),
 Text(1395.0, 906.0, 'NOx <= 87.61\ngini = 0.212\nsamples = 699\nvalue = [5
1, 979, 78]\nclass = b'),
 Text(1255.5, 543.5999999999999, 'NOx <= 42.83\ngini = 0.153\nsamples = 651
\nvalue = [14, 958, 72]\nclass = b'),
 Text(1185.75, 181.19999999999982, 'gini = 0.062\nsamples = 524\nvalue =
[5, 817, 22]\nclass = b'),
 Text(1325.25, 181.19999999999982, 'gini = 0.438\nsamples = 127\nvalue =
[9, 141, 50]\nclass = b'),
 Text(1534.5, 543.5999999999999, 'MXY <= 0.885\ngini = 0.549\nsamples = 48
\nvalue = [37, 21, 6]\nclass = a'),
 Text(1464.75, 181.19999999999982, 'gini = 0.307\nsamples = 26\nvalue = [2
7, 5, 1]\nclass = a'),
 Text(1604.25, 181.19999999999982, 'gini = 0.604\nsamples = 22\nvalue = [1
```

Conclusion

Accuracy

Linear Regression:0.4012427537752702

Ridge Regression:0.3996835768174142

Lasso Regression:0.3996835768174142

ElasticNet Regression:0.23600965343781022

Logistic Regression:0.8741416915744405

Random Forest:0.8748413156376227