Importing Libraries

In [70]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [71]:
```
df=pd.read_csv("madrid_2013.csv").fillna(1)
df
```

Out[71]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2013-11-01 01:00:00	1.0	0.6	1.0	1.0	135.0	74.0	1.0	1.0	1.0	7.0	1.0	1.0	28
1	2013-11-01 01:00:00	1.5	0.5	1.3	1.0	71.0	83.0	2.0	23.0	16.0	12.0	1.0	8.3	28
2	2013-11-01 01:00:00	3.9	1.0	2.8	1.0	49.0	70.0	1.0	1.0	1.0	1.0	1.0	9.0	28
3	2013-11-01 01:00:00	1.0	0.5	1.0	1.0	82.0	87.0	3.0	1.0	1.0	1.0	1.0	1.0	28
4	2013-11-01 01:00:00	1.0	1.0	1.0	1.0	242.0	111.0	2.0	1.0	1.0	12.0	1.0	1.0	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	1.0	0.4	1.0	1.0	8.0	39.0	52.0	1.0	1.0	1.0	1.0	1.0	28
209876	2013-03-01 00:00:00	1.0	0.4	1.0	1.0	1.0	11.0	1.0	6.0	1.0	2.0	1.0	1.0	28
209877	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	2.0	4.0	75.0	1.0	1.0	1.0	1.0	1.0	28
209878	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	2.0	11.0	52.0	1.0	1.0	1.0	1.0	1.0	28
209879	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	1.0	10.0	75.0	3.0	1.0	1.0	1.0	1.0	28

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

In [72]:
```
df=df.dropna()
```

In [73]:
```
df.columns
```

Out[73]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [74]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 24.0+ MB
```

In [75]:
```
data=df[['CO' ,'station']]
data
```

Out[75]:

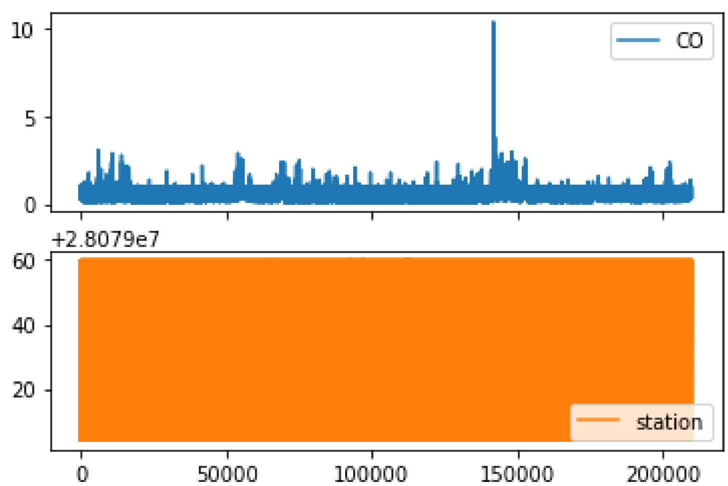
	CO	station
0	0.6	28079004
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

Line chart

In [76]:
```
data.plot.line(subplots=True)
```

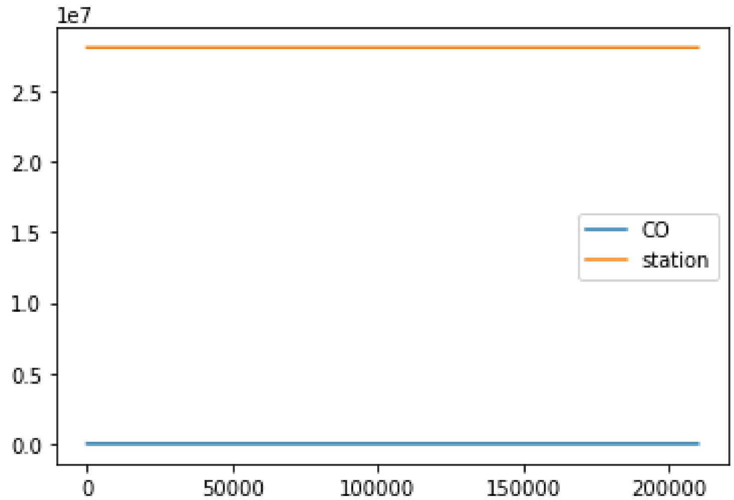
Out[76]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [77]:
```
data.plot.line()
```

Out[77]: <AxesSubplot:>

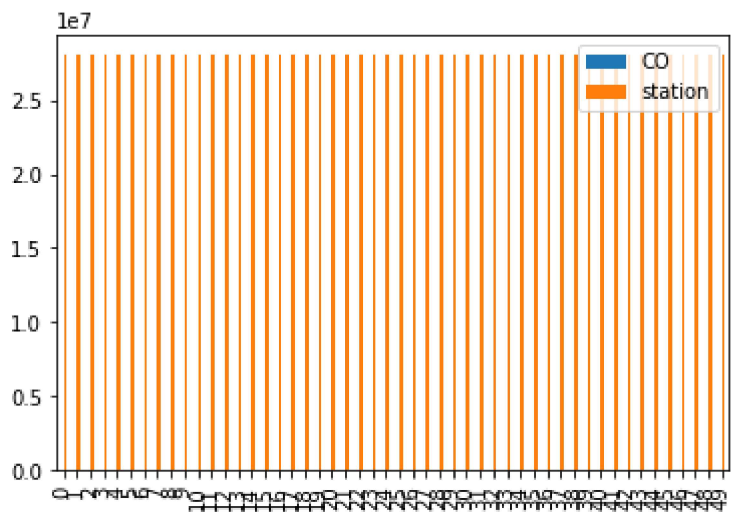


Bar chart

In [78]:
```
b=data[0:50]
```

In [79]:
```
b.plot.bar()
```

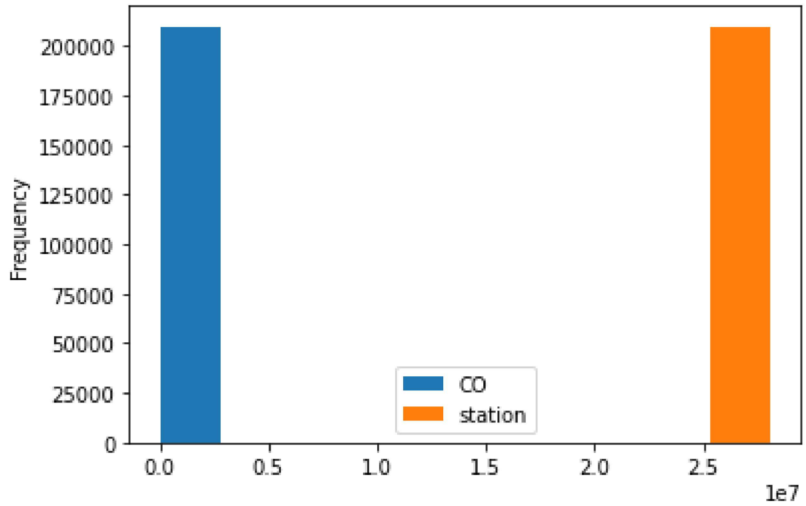
Out[79]: <AxesSubplot:>



Histogram

In [80]:
```
data.plot.hist()
```

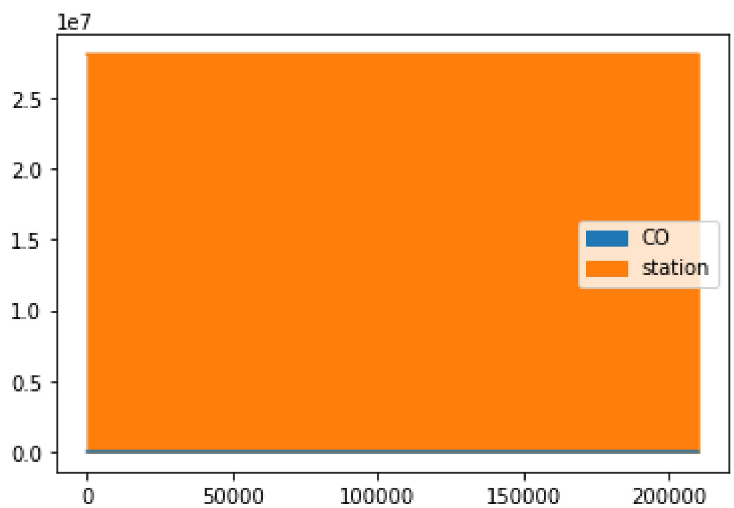
Out[80]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [81]:
```
data.plot.area()
```

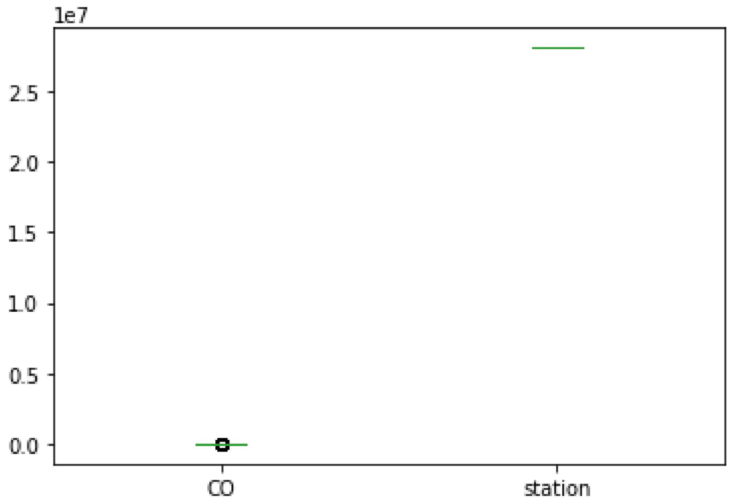
Out[81]: <AxesSubplot:>



Box chart

In [82]:
```
data.plot.box()
```

Out[82]: <AxesSubplot:>



Pie chart

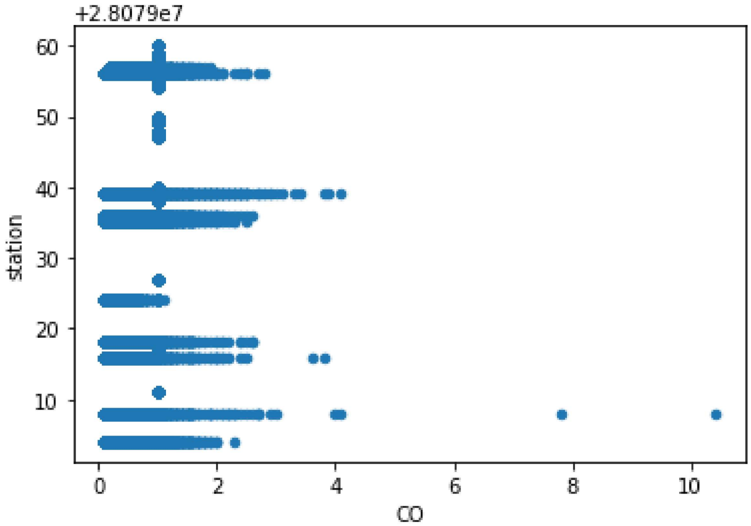
In [83]:
```
b.plot.pie(y='station' )
```

Out[83]: <AxesSubplot:ylabel='station'>

Scatter chart

In [84]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[84]:
```
<AxesSubplot:xlabel='CO', ylabel='station'>
```



In [85]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   date     209880 non-null  object
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64
dtypes: float64(12), int64(1), object(1)
```

In [86]:
```
df.describe()
```

Out[86]:

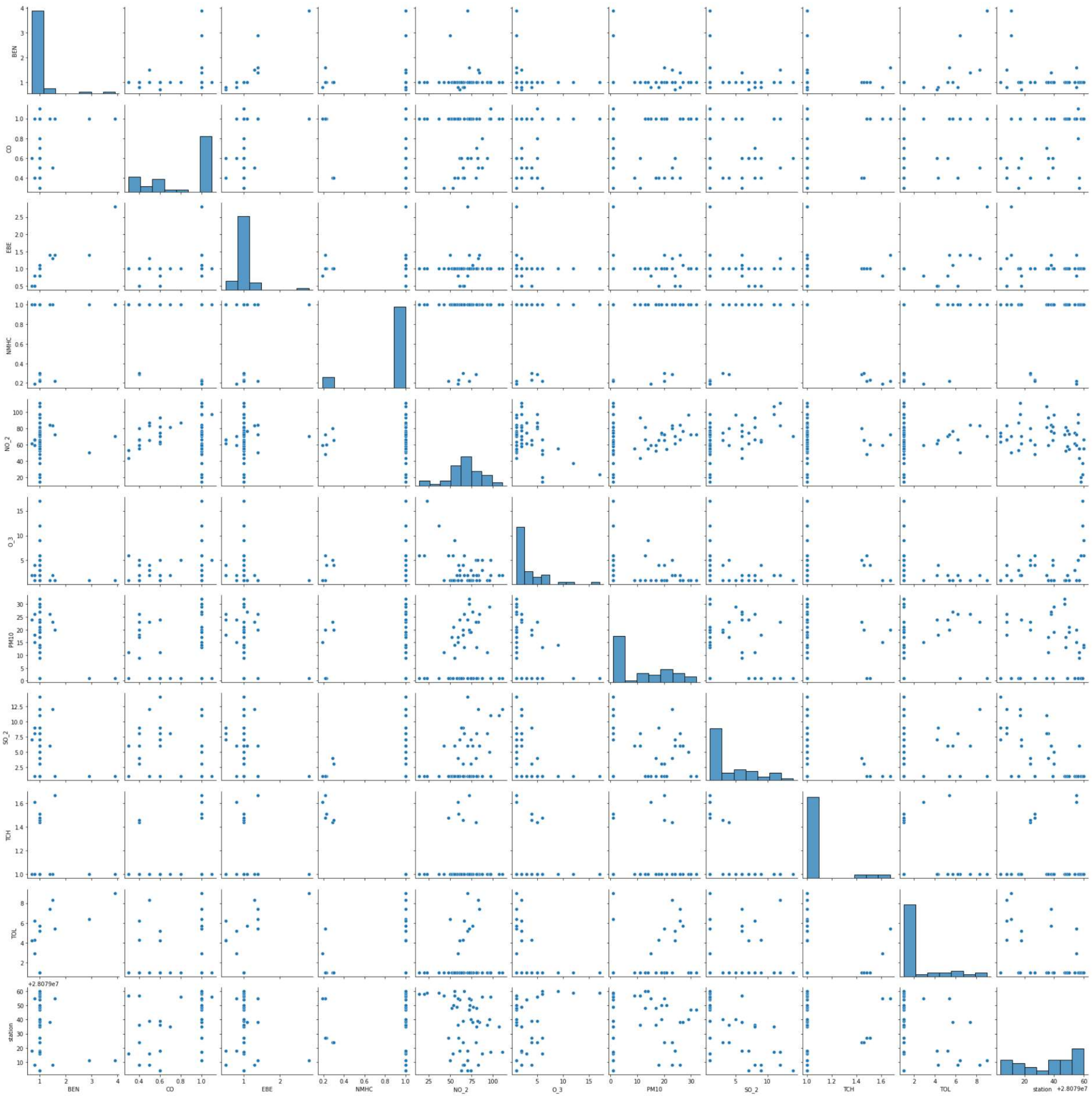
	BEN	CO	EBE	NMHC	NO	NO_
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.00000
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.58640
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.86658
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.00000
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.00000
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.00000
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.00000
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.00000

In [87]:
```
df1=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [88]:
```
sns.pairplot(df1[0:50])
```

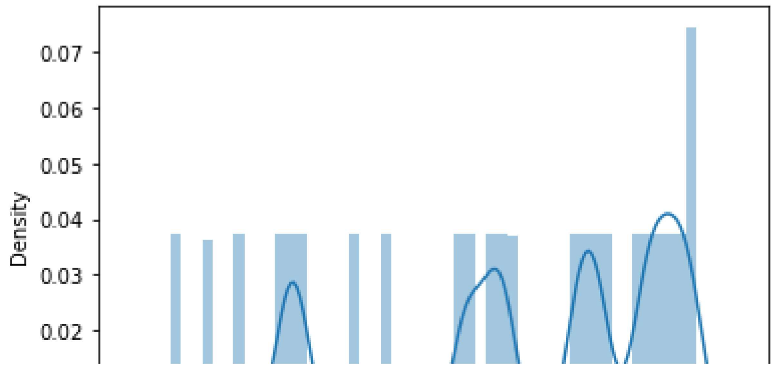
Out[88]: <seaborn.axisgrid.PairGrid at 0x21084f226a0>



In [89]:
```
sns.distplot(df1['station'])
```

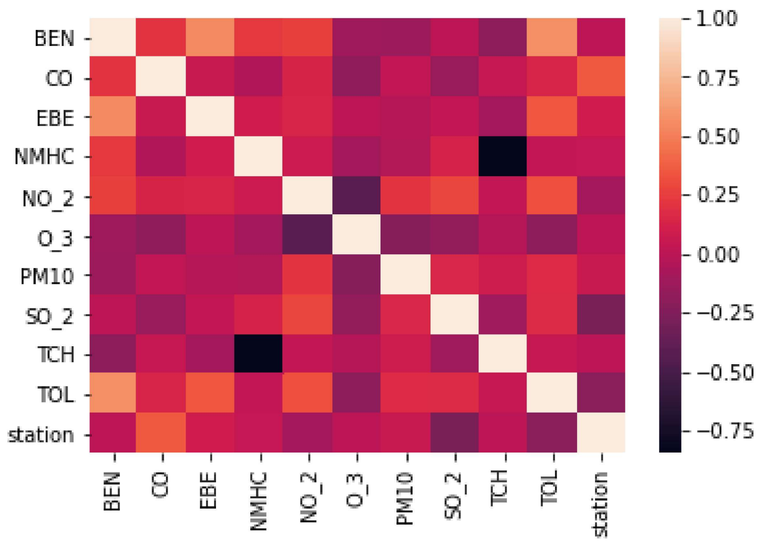
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio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

Out[89]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [90]:
```
sns.heatmap(df1.corr())
```

Out[90]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [91]:
```
x=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [92]:
```
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 [93]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[93]: LinearRegression()

In [94]:
```
lr.intercept_
```

Out[94]: 28078974.171740144

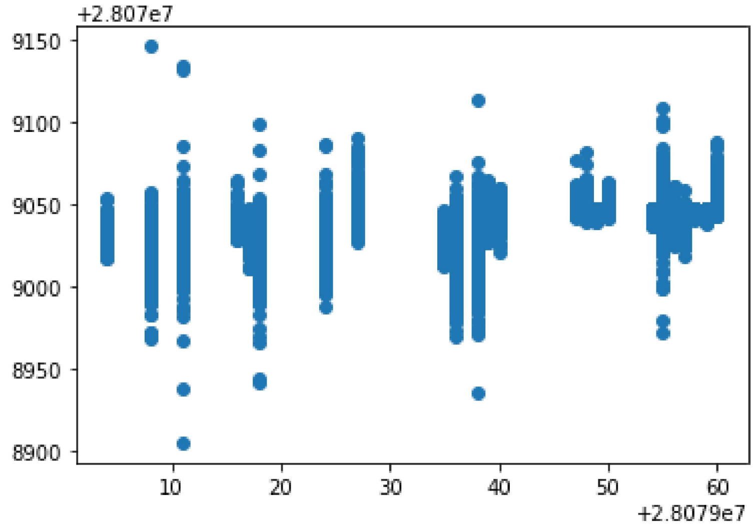
In [95]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[95]:

	Co-efficient
BEN	2.179739
CO	18.374955
EBE	10.083815
NMHC	18.720052
NO_2	-0.055955
O_3	0.009840
PM10	0.206652
SO_2	-0.933468
TCH	27.406878
TOL	-3.694933

In [96]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[96]: <matplotlib.collections.PathCollection at 0x21091bd7610>



ACCURACY

In [97]:
```
lr.score(x_test,y_test)
```

Out[97]: 0.29798024782200694

In [98]:
```
lr.score(x_train,y_train)
```

Out[98]: 0.3005822000263991

Ridge and Lasso

In [99]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [100]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[100]: Ridge(alpha=10)

Accuracy(Ridge)

In [101]:
```
rr.score(x_test,y_test)
```

Out[101]: 0.29800039047740845

In [102]:
```
rr.score(x_train,y_train)
```
Out[102]: 0.300578924013424

Accuracy(Lasso)

In [103]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[103]: Lasso(alpha=10)

In [104]:
```
la.score(x_train,y_train)
```
Out[104]: 0.04519637644952712

ElasticNet

In [105]:
```
la.score(x_test,y_test)
```
Out[105]: 0.04452577535469704

In [106]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[106]: ElasticNet()

In [107]:
```
en.coef_
```
Out[107]:
```
array([ 0.40223113,  2.69520561,  0.5174599 ,  0.        , -0.02040017,
       -0.01561821,  0.16279299, -1.27581233, -0.        , -1.66911887])
```

In [108]:
```
en.intercept_
```
Out[108]: 28079039.963054292

In [109]:
```
prediction=en.predict(x_test)
```

In [110]:
```
en.score(x_test,y_test)
```
Out[110]: 0.15303243515292386

Evaluation Metrics

In [111]:
```
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)))
```

```
13.733731326439136
262.79454649641025
16.21093909976872
```

Logistic Regression

In [112]:
```
from sklearn.linear_model import LogisticRegression
```

In [113]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [114]:
```
feature_matrix.shape
```

Out[114]:
```
(209880, 10)
```

In [115]:
```
target_vector.shape
```

Out[115]:
```
(209880,)
```

In [116]:
```
from sklearn.preprocessing import StandardScaler
```

In [117]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [118]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[118]:
```
LogisticRegression(max_iter=10000)
```

In [121]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [122]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [123]:
```
logr.classes_
```

Out[123]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [124]:
```
logr.score(fs,target_vector)
```
Out[124]:
```
0.6612921669525443
```

In [125]:
```
logr.predict_proba(observation)[0][0]
```
Out[125]:
```
9.49253547859177e-217
```

In [126]:
```
logr.predict_proba(observation)
```
Out[126]:
```
array([[9.49253548e-217, 6.03969072e-001, 1.69773000e-169,
        1.44179094e-134, 1.71060740e-074, 3.96021369e-001,
        9.55808997e-006, 5.22717178e-089, 5.48319507e-081,
        1.32436170e-079, 1.07294134e-076, 3.50636612e-129,
        1.69529056e-079, 3.82520459e-158, 4.22872970e-161,
        3.57928159e-187, 2.10845766e-164, 8.33937392e-188,
        1.12752042e-082, 7.42692411e-129, 7.66872499e-080,
        6.30044443e-191, 4.32093567e-191, 3.26054498e-071]])
```

Random Forest

In [127]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [128]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[128]:
```
RandomForestClassifier()
```

In [129]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [130]:
```
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[130]:
```
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 [131]:
```
grid_search.best_score_
```
Out[131]:
```
0.6921165836260176
```

In [132]:
```
rfc_best=grid_search.best_estimator_
```

In [133]:
```
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[133]:
```
[Text(2480.0, 1993.2, 'SO_2 <= 1.5\ngini = 0.958\nsamples = 92867\nvalue =
[6087, 5963, 6200, 6122, 6172, 6108, 6139, 6071, 6153\n6021, 6094, 6162, 61
29, 6265, 6136, 6252, 6120, 6171\n6106, 6035, 5939, 6179, 5970, 6322]\nclas
s = x'),
 Text(1550.0, 1630.8000000000002, 'CO <= 0.95\ngini = 0.936\nsamples = 5823
8\nvalue = [11, 49, 6200, 6122, 71, 40, 1851, 6071, 8, 3158, 7\n6162, 900,
6265, 6136, 6252, 6120, 6171, 6106, 6035\n30, 6179, 5970, 6322]\nclass =
x'),
 Text(806.0, 1268.4, 'PM10 <= 1.5\ngini = 0.773\nsamples = 14369\nvalue =
[3, 11, 0, 6017, 0, 33, 1831, 0, 6, 3156, 0, 5961\n0, 0, 0, 0, 0, 0, 0, 574
2, 20, 0, 0, 0]\nclass = d'),
 Text(496.0, 906.0, 'CO <= 0.15\ngini = 0.668\nsamples = 11200\nvalue = [3,
0, 0, 6017, 0, 6, 22, 0, 6, 2, 0, 5961, 0\n0, 0, 0, 0, 0, 0, 5742, 0, 0, 0,
0]\nclass = d'),
 Text(248.0, 543.5999999999999, 'O_3 <= 71.5\ngini = 0.482\nsamples = 788\n
value = [0, 0, 0, 39, 0, 0, 0, 0, 0, 0, 0, 406, 0, 0\n0, 0, 0, 0, 0, 788,
0, 0, 0, 0]\nclass = t'),
 Text(124.0, 181.19999999999982, 'gini = 0.453\nsamples = 441\nvalue = [0,
0, 0, 34, 0, 0, 0, 0, 0, 0, 0, 181, 0, 0\n0, 0, 0, 0, 0, 480, 0, 0, 0, 0]\n
```

Conclusion

Accuracy

In [134]:
```
lr.score(x_train,y_train)
```

Out[134]: 0.3005822000263991

In [135]:
```
rr.score(x_train,y_train)
```

Out[135]: 0.300578924013424

In [136]:
```
la.score(x_train,y_train)
```

Out[136]: 0.04519637644952712

In [137]:
```
en.score(x_test,y_test)
```

Out[137]: 0.15303243515292386

In [138]:
```
logr.score(fs,target_vector)
```

Out[138]: 0.6612921669525443

In [139]:
```
grid_search.best_score_
```

Out[139]:
```
0.6921165836260176
```

Linear Regression is suitable for this dataset