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("madrid_2017.csv").fillna(1)
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH
0	2017-06-01 01:00:00	1.0	1.0	0.3	1.0	1.00	4.0	38.0	1.0	1.0	1.0	1.0	5.0	1.0
1	2017-06-01 01:00:00	0.6	1.0	0.3	0.4	0.08	3.0	39.0	1.0	71.0	22.0	9.0	7.0	1.4
2	2017-06-01 01:00:00	0.2	1.0	1.0	0.1	1.00	1.0	14.0	1.0	1.0	1.0	1.0	1.0	1.0
3	2017-06-01 01:00:00	1.0	1.0	0.2	1.0	1.00	1.0	9.0	1.0	91.0	1.0	1.0	1.0	1.0
4	2017-06-01 01:00:00	1.0	1.0	1.0	1.0	1.00	1.0	19.0	1.0	69.0	1.0	1.0	2.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210115	2017-08-01 00:00:00	1.0	1.0	0.2	1.0	1.00	1.0	27.0	1.0	65.0	1.0	1.0	1.0	1.0
210116	2017-08-01 00:00:00	1.0	1.0	0.2	1.0	1.00	1.0	14.0	1.0	1.0	73.0	1.0	7.0	1.0
210117	2017-08-01 00:00:00	1.0	1.0	1.0	1.0	1.00	1.0	4.0	1.0	83.0	1.0	1.0	1.0	1.0
210118	2017-08-01 00:00:00	1.0	1.0	1.0	1.0	1.00	1.0	11.0	1.0	78.0	1.0	1.0	1.0	1.0
210119	2017-08-01 00:00:00	1.0	1.0	1.0	1.0	1.00	1.0	14.0	1.0	77.0	60.0	1.0	1.0	1.0

210120 rows × 16 columns

Data Cleaning and Data Preprocessing

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

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

Out[4]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      210120 non-null  float64
 2   CH4      210120 non-null  float64
 3   CO       210120 non-null  float64
 4   EBE      210120 non-null  float64
 5   NMHC     210120 non-null  float64
 6   NO       210120 non-null  float64
 7   NO_2     210120 non-null  float64
 8   NOx      210120 non-null  float64
 9   O_3      210120 non-null  float64
 10  PM10     210120 non-null  float64
 11  PM25     210120 non-null  float64
 12  SO_2     210120 non-null  float64
 13  TCH      210120 non-null  float64
 14  TOL      210120 non-null  float64
 15  station  210120 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 27.3+ MB
```

In [6]:
```
data=df[['CO' ,'station']]
data
```

Out[6]:

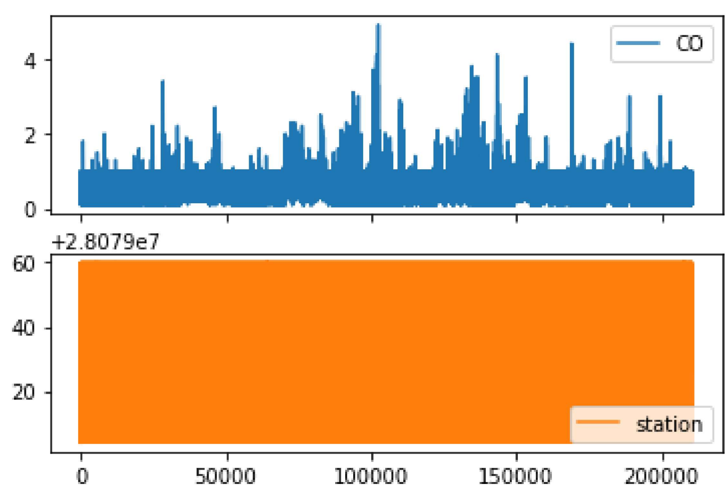
	CO	station
0	0.3	28079004
1	0.3	28079008
2	1.0	28079011
3	0.2	28079016
4	1.0	28079017
...	...	...
210115	0.2	28079056
210116	0.2	28079057
210117	1.0	28079058
210118	1.0	28079059
210119	1.0	28079060

210120 rows × 2 columns

Line chart

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

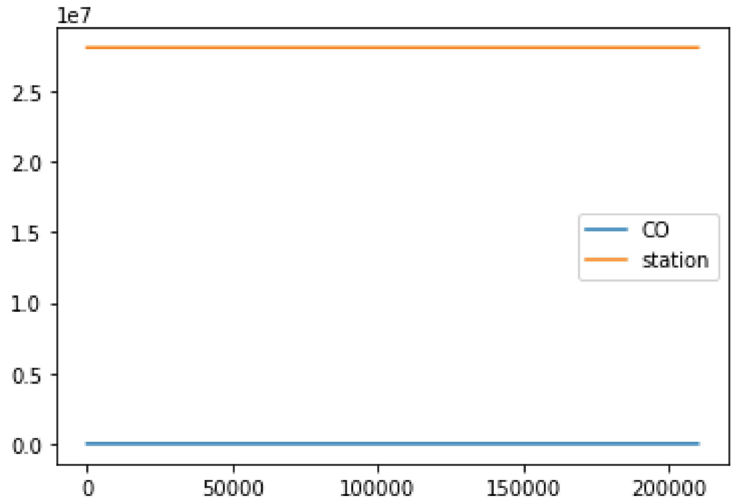
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

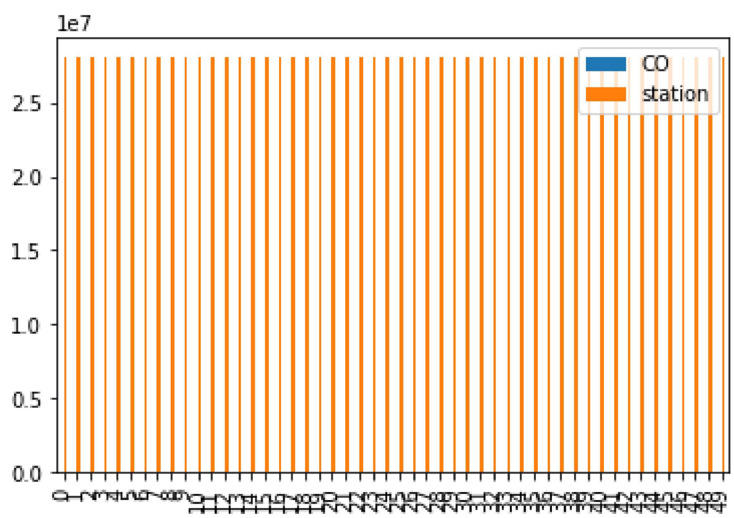


Bar chart

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

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

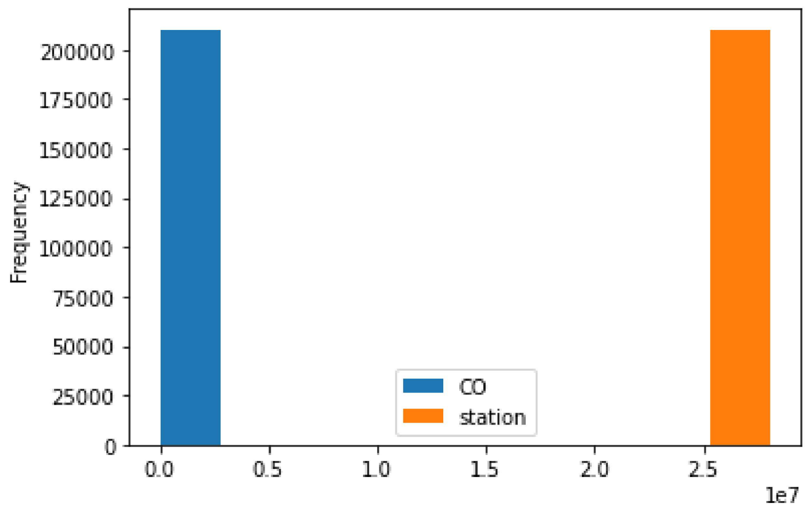
Out[10]: <AxesSubplot:>



Histogram

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

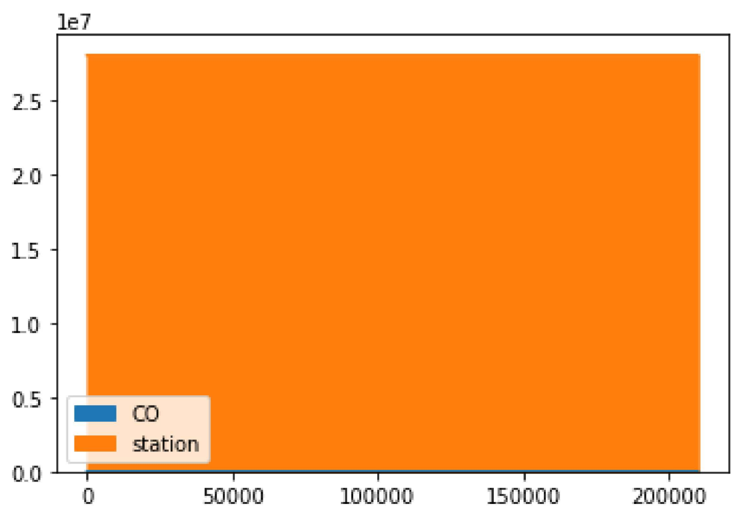
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

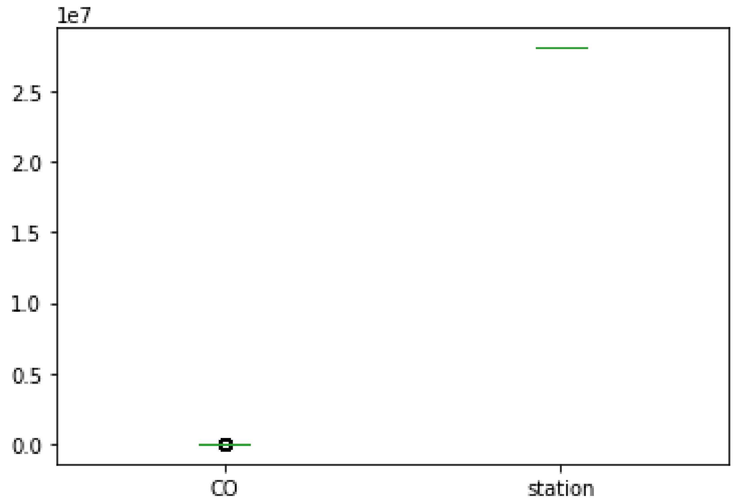
Out[12]: <AxesSubplot:>



Box chart

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

Out[13]: <AxesSubplot:>



Pie chart

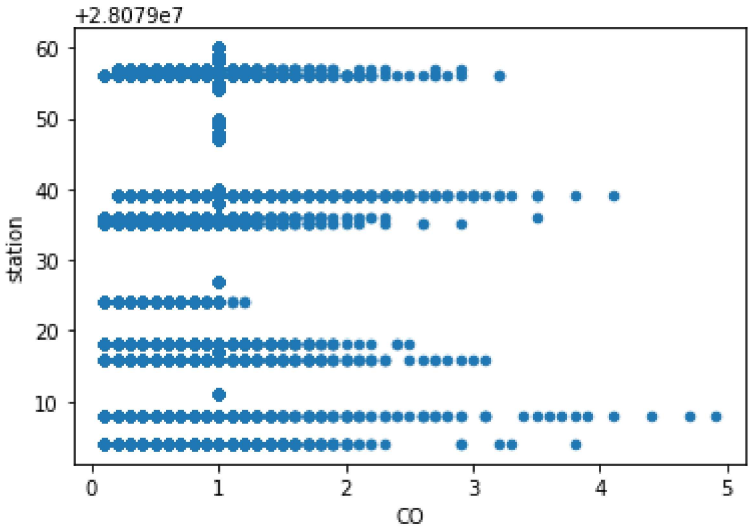
In [14]:
```
b.plot.pie(y='station' )
```

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

Scatter chart

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

Out[15]:
```
<AxesSubplot:xlabel='CO', ylabel='station'>
```



In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      210120 non-null  float64
 2   CH4      210120 non-null  float64
 3   CO       210120 non-null  float64
 4   EBE      210120 non-null  float64
 5   NMHC     210120 non-null  float64
 6   NO       210120 non-null  float64
 7   NO_2     210120 non-null  float64
 8   NOx      210120 non-null  float64
 9   O_3      210120 non-null  float64
 10  PM10     210120 non-null  float64
 11  PM25     210120 non-null  float64
 12  SO_2     210120 non-null  float64
 13  TCH      210120 non-null  float64
```

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

Out[17]:

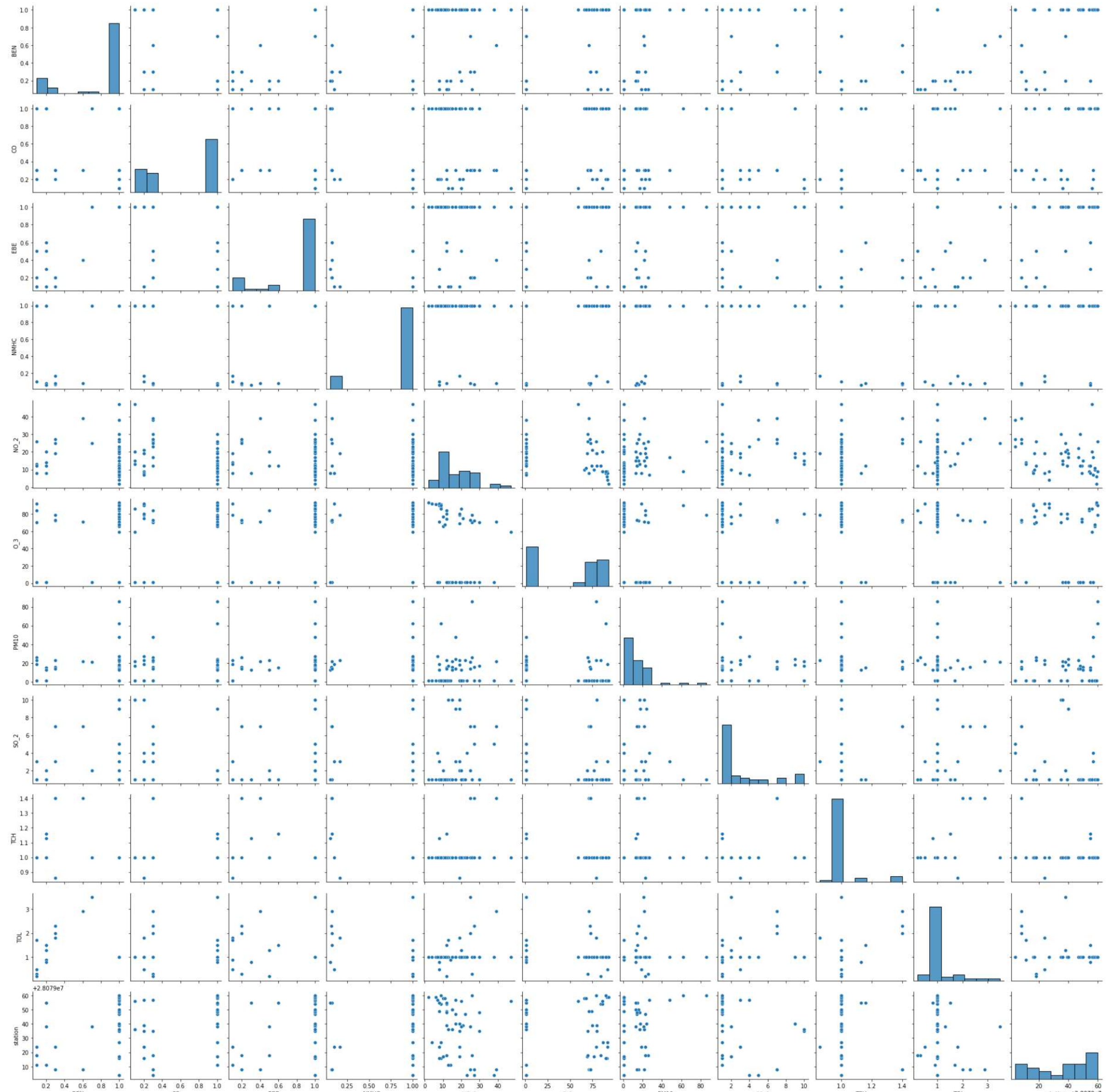
	BEN	CH4	CO	EBE	NMHC	N
count	210120.000000	210120.000000	210120.000000	210120.000000	210120.000000	210120.00000
mean	0.903367	1.009799	0.736606	0.856069	0.894275	23.29667
std	0.415995	0.065702	0.356031	0.417742	0.286557	50.2613
min	0.100000	1.000000	0.100000	0.100000	0.000000	1.00000
25%	1.000000	1.000000	0.300000	1.000000	1.000000	2.00000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	6.00000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	20.00000
max	19.600000	3.630000	4.900000	38.299999	4.400000	973.00000

In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

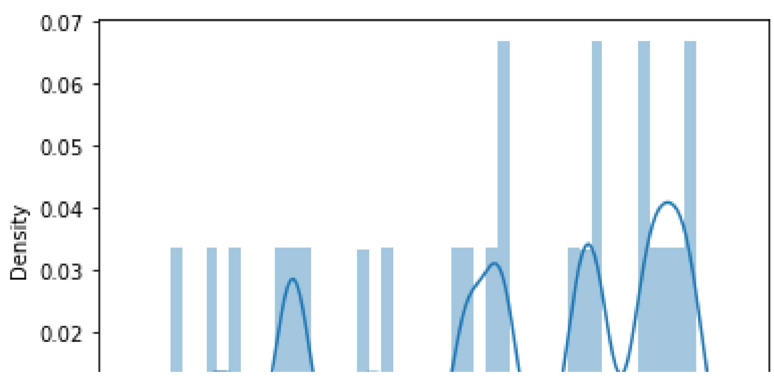
Out[19]: <seaborn.axisgrid.PairGrid at 0x1ae52b5b250>



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

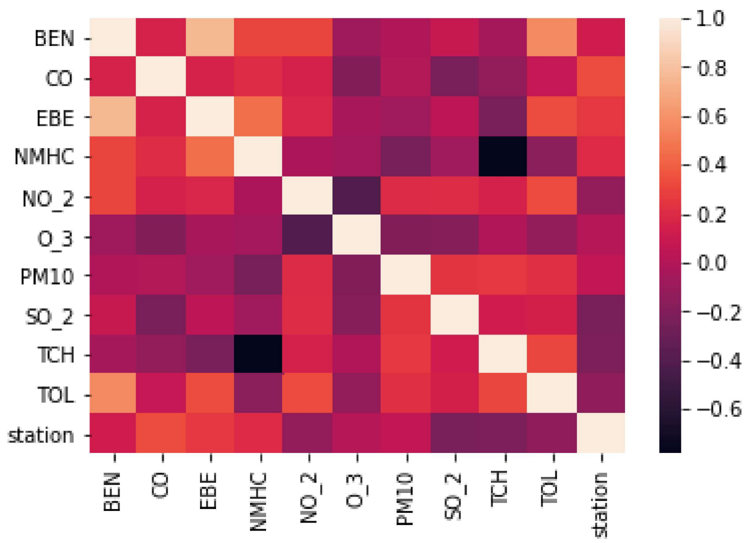
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[20]: <AxesSubplot:xlabel='station', 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_2', 'O_3',
       'PM10', '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]:
```
28079048.29312048
```

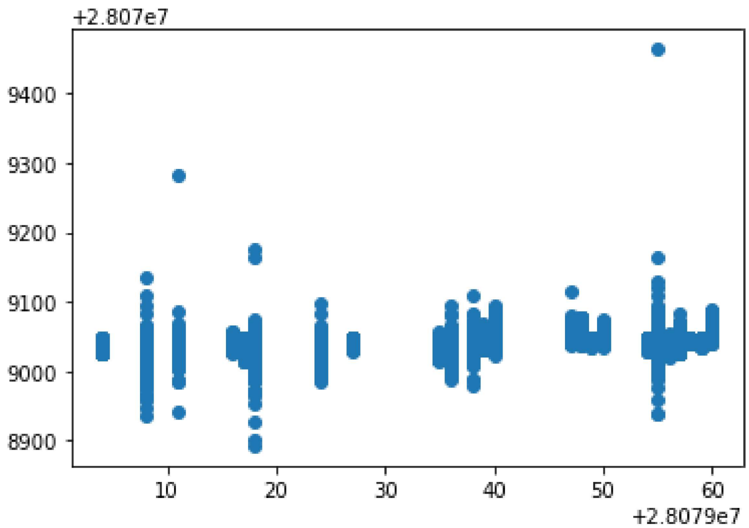
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	5.283567
CO	14.520638
EBE	13.180195
NMHC	-11.862988
NO_2	-0.083729
O_3	-0.002715
PM10	0.253691
SO_2	-0.636507
TCH	-19.277328
TOL	-2.456508

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

Out[27]: <matplotlib.collections.PathCollection at 0x1ae5f88b0d0>



ACCURACY

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

Out[28]: 0.3048526510319767

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

Out[29]: 0.3059238247577176

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]: 0.3048569414042449

In [33]:
```
rr.score(x_train,y_train)
```
Out[33]: 0.3059225891971954

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

In [35]:
```
la.score(x_train,y_train)
```
Out[35]: 0.058554267110469405

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```
Out[36]: 0.05765364356354252

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([ 1.41071731,  2.08097256,  2.58112421,  0.42682043, -0.05934952,
       -0.02051917,  0.1908996 , -0.8727883 , -0.        , -1.06113847])
```

In [39]:
```
en.intercept_
```
Out[39]: 28079037.85476781

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

In [41]:
```
en.score(x_test,y_test)
```
Out[41]: 0.16513257854706354

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)))
```
```
13.622986632450322
259.9651686333126
16.123435385590522
```

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]:
```
(210120, 10)
```

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

Out[46]:
```
(210120,)
```

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)
```

```
[28079018]
```

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

Out[52]:
```
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 [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.6336855130401675
```

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

Out[54]:
```
3.1869195039911524e-203
```

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

Out[55]:
```
array([[3.18691950e-203, 2.22165855e-010, 4.62826902e-188,
        1.06625793e-151, 3.29027331e-092, 1.00000000e+000,
        2.21819034e-026, 4.03862272e-193, 1.48367814e-096,
        5.12259815e-077, 3.86118695e-047, 1.24733680e-161,
        5.76936555e-092, 3.74086730e-199, 3.15398013e-198,
        1.68143596e-199, 1.64434869e-201, 6.74349755e-192,
        2.71552113e-129, 8.96658400e-151, 1.98748253e-083,
        4.27215238e-205, 6.55150631e-197, 9.90226143e-095]])
```

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 [*]:
```
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)
```

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

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

In [*]:
```
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'
```

Conclusion

Accuracy

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

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

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

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

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

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

Logistic Regression is suitable for this dataset