In [49]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [50]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\madrid_2013.csv")
data
```

Out[50]:

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

209880 rows × 14 columns

In [51]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      50462 non-null   float64
 2   CO       87018 non-null   float64
 3   EBE      50463 non-null   float64
 4   NMHC     25935 non-null   float64
 5   NO       209108 non-null  float64
 6   NO_2     209108 non-null  float64
 7   O_3      121858 non-null  float64
 8   PM10     104339 non-null  float64
 9   PM25     51980 non-null   float64
 10  SO_2     86970 non-null   float64
 11  TCH      25935 non-null   float64
 12  TOL      50317 non-null   float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [52]:
```
df=data.fillna(value=0)
df
```

Out[52]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2013-11-01 01:00:00	0.0	0.6	0.0	0.0	135.0	74.0	0.0	0.0	0.0	7.0	0.0	0.0	2
1	2013-11-01 01:00:00	1.5	0.5	1.3	0.0	71.0	83.0	2.0	23.0	16.0	12.0	0.0	8.3	2
2	2013-11-01 01:00:00	3.9	0.0	2.8	0.0	49.0	70.0	0.0	0.0	0.0	0.0	0.0	9.0	2
3	2013-11-01 01:00:00	0.0	0.5	0.0	0.0	82.0	87.0	3.0	0.0	0.0	0.0	0.0	0.0	2
4	2013-11-01 01:00:00	0.0	0.0	0.0	0.0	242.0	111.0	2.0	0.0	0.0	12.0	0.0	0.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	0.0	0.4	0.0	0.0	8.0	39.0	52.0	0.0	0.0	0.0	0.0	0.0	2
209876	2013-03-01 00:00:00	0.0	0.4	0.0	0.0	1.0	11.0	0.0	6.0	0.0	2.0	0.0	0.0	2
209877	2013-03-01 00:00:00	0.0	0.0	0.0	0.0	2.0	4.0	75.0	0.0	0.0	0.0	0.0	0.0	2
209878	2013-03-01 00:00:00	0.0	0.0	0.0	0.0	2.0	11.0	52.0	0.0	0.0	0.0	0.0	0.0	2
209879	2013-03-01 00:00:00	0.0	0.0	0.0	0.0	1.0	10.0	75.0	3.0	0.0	0.0	0.0	0.0	2

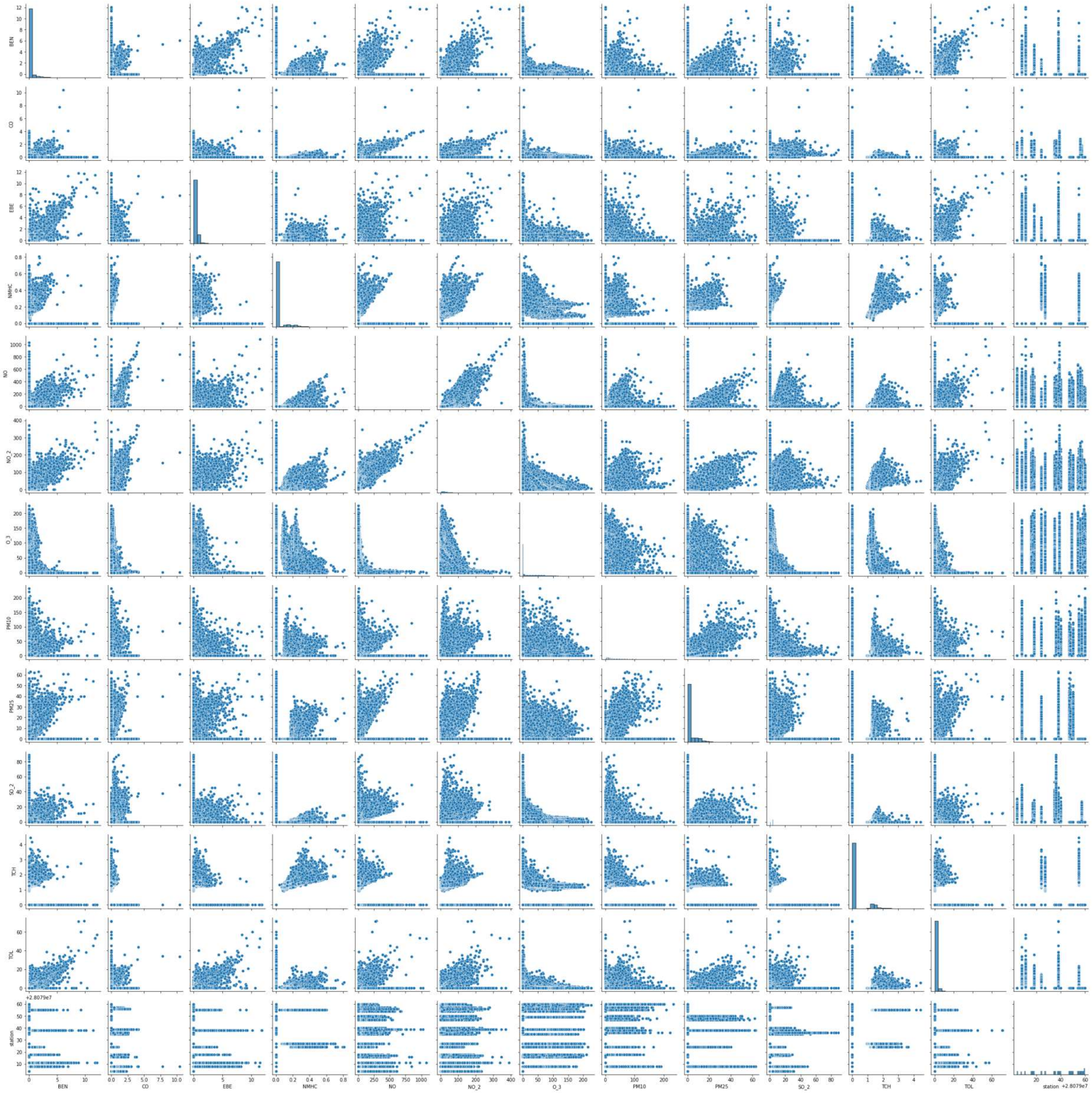
209880 rows × 14 columns

In [53]:
```
df.columns
```

Out[53]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [54]:
```
sns.pairplot(df)
```

Out[54]:
```
<seaborn.axisgrid.PairGrid at 0x1f81f5e0d00>
```



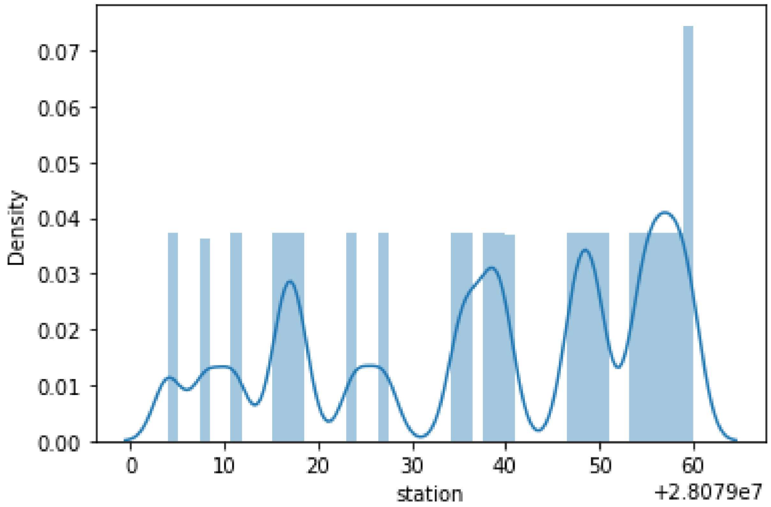
In [55]:

```
sns.distplot(data["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[55]:

```
<AxesSubplot:xlabel='station', ylabel='Density'>
```



MODEL BUILDING

Linear Regression

In [57]:

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

In [58]:

```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df1['station']
```

In [59]:

```
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)
```

In [60]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[60]:

```
LinearRegression()
```

In [61]:

```
print(lr.intercept_)
```

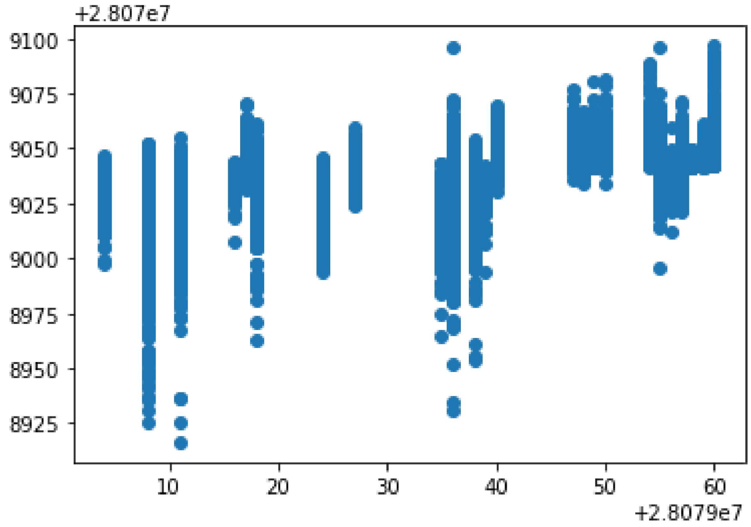
```
28079043.051533587
```

In [62]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[62]:

```
<matplotlib.collections.PathCollection at 0x1f8855bdac0>
```



In [63]:

```
print(lr.score(x_test,y_test))
```

```
0.29815273412055265
```

Ridge Regression

In [17]:

```
from sklearn.linear_model import Ridge
```

In [18]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[18]:

```
Ridge(alpha=10)
```

In [19]:

```
rr.score(x_test,y_test)
```

Out[19]:

```
0.2990637039599836
```

Lasso Regression

In [20]:

```
from sklearn.linear_model import Lasso
```

In [21]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[21]:
```
Lasso(alpha=10)
```

In [22]:
```
la.score(x_test,y_test)
```

Out[22]:
```
0.14425099333511227
```

Elastic Regression

In [23]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

```
[-6.55959003e-01 -4.26230045e-01 -7.95802910e-01 -0.00000000e+00
  2.91478073e-02 -3.41943221e-02 -1.14131981e-03  2.78342122e-01
 -2.46922131e-01 -1.31701512e+00 -0.00000000e+00 -1.69251706e+00]
```

In [25]:
```
print(en.predict(x_test))
```

```
[28079035.77703039 28079035.11076136 28079040.3825461  ...
 28079032.62400804 28079038.38271381 28079041.36218826]
```

In [26]:
```
print(en.score(x_test,y_test))
```

```
0.2418437372373553
```

Logistic Regression

In [27]:
```
from sklearn.linear_model import LogisticRegression
```

In [28]:
```
feature_matrix=df1.iloc[:,0:14]
target_vector=df1.iloc[:,-1]
```

In [29]:
```
feature_matrix.shape
```

Out[29]:
```
(209928, 13)
```

In [30]:
```
target_vector.shape
```

Out[30]:
```
(209928,)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [33]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[33]:
```
LogisticRegression()
```

In [36]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13]]
```

In [37]:
```
prediction=logr.predict(observation)
print(observation)
```

```
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]]
```

In [38]:
```
logr.classes_
```

Out[38]:
```
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 [39]:
```
logr.score(fs,target_vector)
```

Out[39]:
```
0.9833323806257384
```

Random Forest

In [40]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [41]:

```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df1['station']
```

In [42]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.70)
```

In [43]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[43]:

```
RandomForestClassifier()
```

In [44]:

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

In [45]:

```
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[45]:

```
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 [46]:

```
grid_search.best_score_
```

Out[46]:

```
0.7565975419987933
```

In [47]:

```
rfc_best=grid_search.best_estimator_
```

In [48]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

Out[48]:
```
[Text(2222.7000000000003, 1993.2, 'SO_2 <= 0.5\ngini = 0.958\nsamples = 39
927\nvalue = [2502, 2640, 2607, 2536, 2708, 2675, 2644, 2669, 2692\n2559,
2474, 2484, 2677, 2724, 2550, 2674, 2714, 2597\n2579, 2756, 2668, 2687, 25
12, 2650]'),
 Text(1171.8000000000002, 1630.8000000000002, 'O_3 <= 0.5\ngini = 0.929\ns
amples = 23321\nvalue = [8, 16, 2607, 2536, 10, 3, 16, 2669, 13, 2, 17\n24
84, 14, 2724, 2550, 2674, 2714, 2597, 2579, 2756\n1, 2687, 2512, 2650]'),
 Text(595.2, 1268.4, 'BEN <= 0.05\ngini = 0.805\nsamples = 8427\nvalue =
[8, 4, 2607, 4, 8, 0, 13, 4, 2, 2, 17, 2, 14\n2724, 2550, 8, 2714, 5, 257
9, 20, 1, 15, 27, 3]'),
 Text(297.6, 906.0, 'TCH <= 0.625\ngini = 0.685\nsamples = 5184\nvalue =
[8, 1, 54, 4, 8, 0, 8, 4, 2, 2, 16, 2, 14\n2724, 2550, 8, 2714, 5, 30, 20,
1, 15, 27, 3]'),
 Text(148.8, 543.5999999999999, 'NO <= 5.5\ngini = 0.682\nsamples = 5156\n
value = [8, 1, 54, 4, 8, 0, 8, 1, 2, 2, 16, 2, 14\n2724, 2550, 8, 2714, 5,
0, 20, 1, 12, 27, 3]'),
 Text(74.4, 181.19999999999982, 'gini = 0.666\nsamples = 2304\nvalue = [5,
1, 31, 1, 5, 0, 8, 0, 1, 0, 16, 1, 13\n1550, 1198, 8, 728, 5, 0, 17, 0, 8,
20, 1]'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.657\nsamples = 385
```

Results

The best model is Logistic Regression 0.9833323806257384

In [ ]:
```

```