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

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\madrid_2014.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	2
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	2
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	2
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	2
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210019	2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN	2
210020	2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN	2
210021	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN	2
210022	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN	2
210023	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN	2

210024 rows × 14 columns

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210024 entries, 0 to 210023
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210024 non-null  object 
 1   BEN      46703 non-null   float64
 2   CO       87023 non-null   float64
 3   EBE      46722 non-null   float64
 4   NMHC     25021 non-null   float64
 5   NO       209154 non-null  float64
 6   NO_2     209154 non-null  float64
 7   O_3      121681 non-null  float64
 8   PM10     104311 non-null  float64
 9   PM25     51954 non-null   float64
 10  SO_2     87141 non-null   float64
 11  TCH      25021 non-null   float64
 12  TOL      46570 non-null   float64
 13  station  210024 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [4]:
```
df=data.fillna(value=0)
df
```

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2014-06-01 01:00:00	0.0	0.2	0.0	0.00	3.0	10.0	0.0	0.0	0.0	3.0	0.00	0.0	28
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28
2	2014-06-01 01:00:00	0.3	0.0	0.1	0.00	2.0	6.0	0.0	0.0	0.0	0.0	0.00	1.1	28
3	2014-06-01 01:00:00	0.0	0.2	0.0	0.00	1.0	6.0	79.0	0.0	0.0	0.0	0.00	0.0	28
4	2014-06-01 01:00:00	0.0	0.0	0.0	0.00	1.0	6.0	75.0	0.0	0.0	4.0	0.00	0.0	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210019	2014-09-01 00:00:00	0.0	0.5	0.0	0.00	20.0	84.0	29.0	0.0	0.0	0.0	0.00	0.0	28
210020	2014-09-01 00:00:00	0.0	0.3	0.0	0.00	1.0	22.0	0.0	15.0	0.0	6.0	0.00	0.0	28
210021	2014-09-01 00:00:00	0.0	0.0	0.0	0.00	1.0	13.0	70.0	0.0	0.0	0.0	0.00	0.0	28
210022	2014-09-01 00:00:00	0.0	0.0	0.0	0.00	3.0	38.0	42.0	0.0	0.0	0.0	0.00	0.0	28
210023	2014-09-01 00:00:00	0.0	0.0	0.0	0.00	1.0	26.0	65.0	11.0	0.0	0.0	0.00	0.0	28

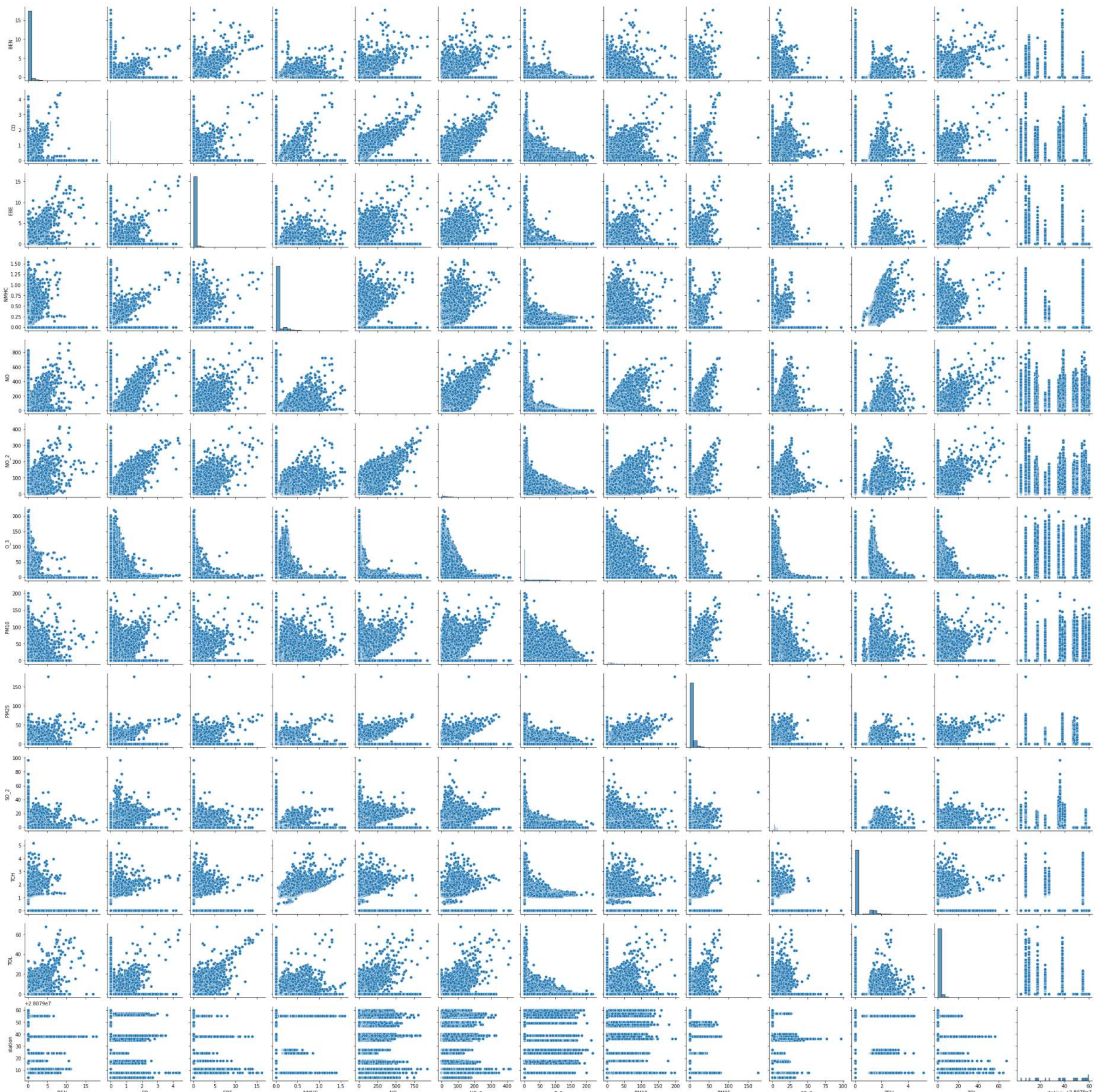
210024 rows × 14 columns

In [5]:
```
df.columns
```

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

In [6]:
```
sns.pairplot(df)
```

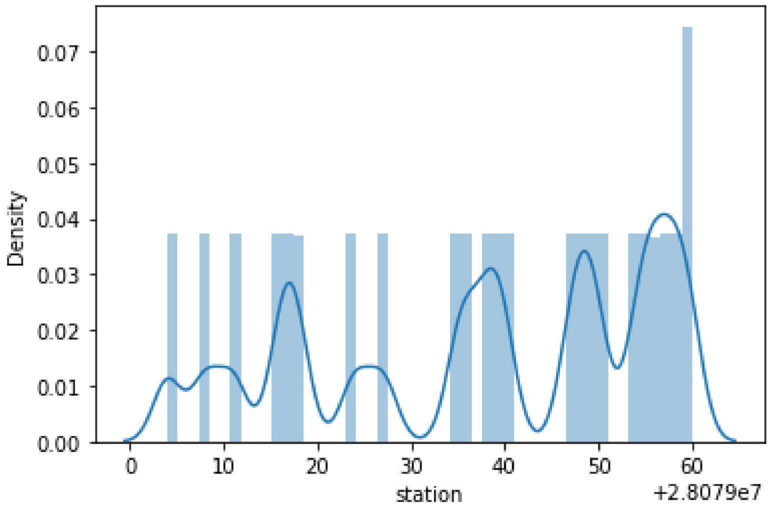
Out[6]: <seaborn.axisgrid.PairGrid at 0x22900044ac0>



In [45]:
```
sns.distplot(data["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[45]: <AxesSubplot:xlabel='station', ylabel='Density'>



MODEL BUILDING

Linear Regression

In [46]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [47]:
```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df1[['station']]
```

In [48]:
```
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 [49]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

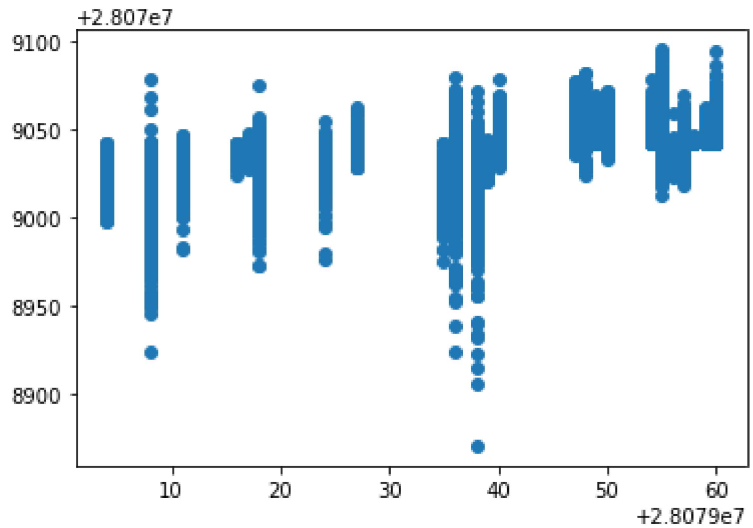
Out[49]: LinearRegression()

In [50]:
```
print(lr.intercept_)
```

```
[28079042.00855549]
```

In [51]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[51]: <matplotlib.collections.PathCollection at 0x2290dbb2ac0>



In [52]:
```
print(lr.score(x_test,y_test))
```

```
0.30805331568051975
```

Ridge Regression

In [53]:
```
from sklearn.linear_model import Ridge
```

In [54]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[54]: Ridge(alpha=10)

In [55]:
```
rr.score(x_test,y_test)
```

Out[55]: 0.3080823444433828

Lasso Regression

In [56]:
```
from sklearn.linear_model import Lasso
```

In [57]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[57]:
```
Lasso(alpha=10)
```

In [58]:
```
la.score(x_test,y_test)
```

Out[58]:
```
0.11705608702010839
```

Elastic Regression

In [59]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[59]:
```
ElasticNet()
```

In [60]:
```
print(en.coef_)
```

```
[-0.68535783 -0.24077958 -0.          -0.          0.03453292 -0.02905656
 -0.01747141  0.28372955 -0.26279892 -1.7989431  -0.39956777 -1.87883152]
```

In [61]:
```
print(en.predict(x_test))
```

```
[28079040.27800945 28079039.76204846 28079043.22049388 ...
 28079044.72021452 28079043.8532663  28079041.58557998]
```

In [62]:
```
print(en.score(x_test,y_test))
```

```
0.2358857348730783
```

Logistic Regression

In [63]:
```
from sklearn.linear_model import LogisticRegression
```

In [64]:
```
feature_matrix=df1.iloc[:,0:14]
target_vector=df1.iloc[:,-1]
```

In [65]:
```
feature_matrix.shape
```

Out[65]:
```
(210024, 13)
```

In [66]:
```
target_vector.shape
```

Out[66]:
```
(210024,)
```

In [67]:
```
from sklearn.preprocessing import StandardScaler
```

In [68]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [69]:
```
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://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[69]:
```
LogisticRegression()
```

In [70]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13]]
```

In [71]:
```
prediction=logr.predict(observation)
print(observation)
```

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

In [72]:
```
logr.classes_
```

Out[72]:
```
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 [73]:
```
logr.score(fs,target_vector)
```

Out[73]:
```
0.975402811107302
```

Random Forest

In [74]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [75]:
```
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 [76]:
```
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 [77]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[77]:
```
RandomForestClassifier()
```

In [78]:
```
parameters={'max_depth':[1,2,3,4,5],
           'min_samples_leaf':[5,10,15,20,25],
           'n_estimators':[10,20,30,40,50]}
```

In [79]:
```
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[79]:
```
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 [80]:
```
grid_search.best_score_
```

Out[80]:
```
0.7358229767737421
```

In [81]:
```
rfc_best=grid_search.best_estimator_
```

In [82]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

```
0, 0, 0, 0]'),
 Text(1913.1428571428569, 543.5999999999999, 'gini = 0.145\nsamples = 31\n
value = [0, 4, 0, 0, 0, 47, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0]'),
 Text(2072.5714285714284, 543.5999999999999, 'TOL <= 9.0\ngini = 0.004\nsa
mples = 1569\nvalue = [0, 5, 0, 0, 0, 2495, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.002\nsamples = 152
9\nvalue = [0, 2, 0, 0, 0, 2433, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0,
0, 0, 0, 0]'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.088\nsamples = 40\n
value = [0, 3, 0, 0, 0, 62, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0]'),
 Text(3407.785714285714, 1630.8000000000002, 'BEN <= 0.05\ngini = 0.833\ns
amples = 9881\nvalue = [0, 2633, 0, 0, 0, 0, 2517, 0, 0, 0, 2594, 0, 0\n25
62, 2700, 0, 2589, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2869.7142857142853, 1268.4, 'PM25 <= 3.5\ngini = 0.694\nsamples = 52
06\nvalue = [0, 78, 0, 0, 0, 0, 153, 0, 0, 0, 124, 0, 0\n2562, 2700, 0, 25
89, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2550.8571428571427, 906.0, 'NO_2 <= 10.5\ngini = 0.545\nsamples = 55
```

Results

The best model is Logistic Regression 0.975402811107302

In [ ]: