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_2016.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN

209496 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209496 entries, 0 to 209495
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209496 non-null  object 
 1   BEN      50755 non-null   float64
 2   CO       85999 non-null   float64
 3   EBE      50335 non-null   float64
 4   NMHC     25970 non-null   float64
 5   NO       208614 non-null  float64
 6   NO_2     208614 non-null  float64
 7   O_3      121197 non-null  float64
 8   PM10     102892 non-null  float64
 9   PM25     52165 non-null   float64
 10  SO_2     86023 non-null   float64
 11  TCH      25970 non-null   float64
 12  TOL      50662 non-null   float64
 13  station  209496 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	2016-11-01 01:00:00	0.0	0.7	0.0	0.00	153.0	77.0	0.0	0.0	0.0	7.0	0.00	0.0	2
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	2
2	2016-11-01 01:00:00	5.9	0.0	7.5	0.00	297.0	139.0	0.0	0.0	0.0	0.0	0.00	26.0	2
3	2016-11-01 01:00:00	0.0	1.0	0.0	0.00	154.0	113.0	2.0	0.0	0.0	0.0	0.00	0.0	2
4	2016-11-01 01:00:00	0.0	0.0	0.0	0.00	275.0	127.0	2.0	0.0	0.0	18.0	0.00	0.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209491	2016-07-01 00:00:00	0.0	0.2	0.0	0.00	2.0	29.0	73.0	0.0	0.0	0.0	0.00	0.0	2
209492	2016-07-01 00:00:00	0.0	0.3	0.0	0.00	1.0	29.0	0.0	36.0	0.0	5.0	0.00	0.0	2
209493	2016-07-01 00:00:00	0.0	0.0	0.0	0.00	1.0	19.0	71.0	0.0	0.0	0.0	0.00	0.0	2
209494	2016-07-01 00:00:00	0.0	0.0	0.0	0.00	6.0	17.0	85.0	0.0	0.0	0.0	0.00	0.0	2
209495	2016-07-01 00:00:00	0.0	0.0	0.0	0.00	2.0	46.0	61.0	34.0	0.0	0.0	0.00	0.0	2

209496 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')
```

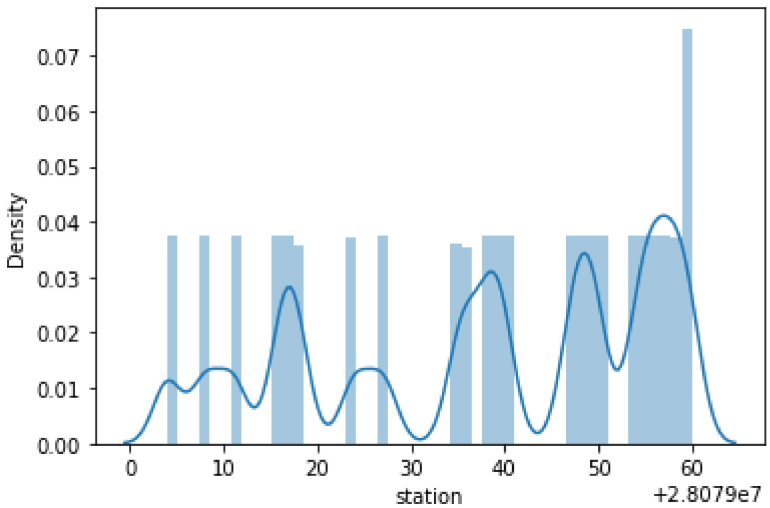
In [6]:
```
sns.pairplot(df)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x18223fe4730>

In [8]:
```
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[8]:
```
<AxesSubplot:xlabel='station', ylabel='Density'>
```



MODEL BUILDING

Linear Regression

In [10]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [11]:
```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df1[['station']]
```

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

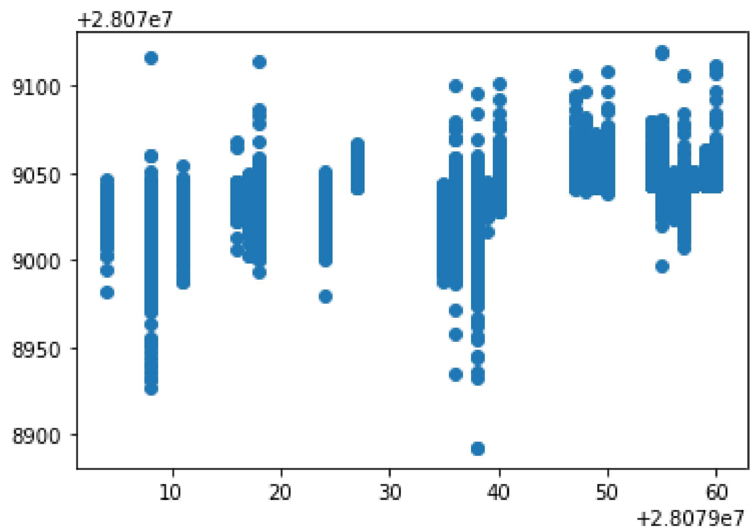
Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
[28079043.29340011]
```

In [15]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x18233e09a00>
```



In [16]:
```
print(lr.score(x_test,y_test))
```

```
0.32782218723180634
```

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]:
```
0.3273460296426014
```

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]:
```
0.16202886515402581
```

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_)
```

```
[-0.91880846 -0.35807898 -0.          0.          0.036925   -0.04430166
 -0.0021193   0.21257535 -0.26272722 -1.30830156 -0.55258298 -1.39872323]
```

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

```
[28079018.92979379 28079042.72678536 28079041.68022207 ...
 28079032.68586449 28079031.16860852 28079035.06668555]
```

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

```
0.24487292196921295
```

Logistic Regression

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

In [28]:
```
feature_matrix=df1.iloc[:,0:15]
target_vector=df1.iloc[:,-1]
```

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

Out[29]:
```
(209496, 13)
```

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

Out[30]:
```
(209496,)
```

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://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[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]:
```
0.9837944399893077
```

Random Forest

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

In [44]:
```
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', 'station']]
y=df1['station']
```

In [45]:
```
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 [46]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[46]:
```
RandomForestClassifier()
```

In [47]:
```
parameters={'max_depth':[1,2,3,4,5],
           'min_samples_leaf':[5,10,15,20,25],
           'n_estimators':[10,20,30,40,50]}
```

In [48]:
```
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[48]:
```
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 [49]:
```
grid_search.best_score_
```

Out[49]:
```
0.9550661914460286
```

In [50]:
```
rfc_best=grid_search.best_estimator_
```

In [51]:
```
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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 2639,
6]'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 1685\n
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 2727]'),
 Text(818.4000000000001, 906.0, 'station <= 28079049.0\ngini = 0.673\nsamp
les = 5013\nvalue = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 80, 0, 0, 2637\n2531,
0, 2568, 0, 0, 0, 0, 0, 0, 0]'),
 Text(744.0, 543.5999999999999, 'NO <= 85.5\ngini = 0.515\nsamples = 3382
\nvalue = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 80, 0, 0, 2637\n2531, 0, 0, 0, 0,
0, 0, 0, 0, 0]'),
 Text(669.6, 181.19999999999982, 'gini = 0.513\nsamples = 3186\nvalue =
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 65, 0, 0, 2432\n2446, 0, 0, 0, 0, 0, 0, 0,
0, 0]'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.468\nsamples = 196
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0, 205\n85, 0, 0, 0, 0, 0,
0, 0, 0, 0]'),
 Text(892.8000000000001, 543.5999999999999, 'gini = 0.0\nsamples = 1631\nv
alue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 2568, 0, 0, 0, 0,
```

Results

The best model is Logistic Regression 0.9837944399893077

In [ ]: