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_2007.csv")
data
```

Out[2]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2007-12-01 01:00:00 | NaN | 2.86 | NaN | NaN | NaN | 282.200012 | 1054.000000 | NaN | 4.030000 | 156. |
| **1** | 2007-12-01 01:00:00 | NaN | 1.82 | NaN | NaN | NaN | 86.419998 | 354.600006 | NaN | 3.260000 | 80. |
| **2** | 2007-12-01 01:00:00 | NaN | 1.47 | NaN | NaN | NaN | 94.639999 | 319.000000 | NaN | 5.310000 | 53. |
| **3** | 2007-12-01 01:00:00 | NaN | 1.64 | NaN | NaN | NaN | 127.900002 | 476.700012 | NaN | 4.500000 | 105. |
| **4** | 2007-12-01 01:00:00 | 4.64 | 1.86 | 4.26 | 7.98 | 0.57 | 145.100006 | 573.900024 | 3.49 | 52.689999 | 106. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **225115** | 2007-03-01 00:00:00 | 0.30 | 0.45 | 1.00 | 0.30 | 0.26 | 8.690000 | 11.690000 | 1.00 | 42.209999 | 6. |
| **225116** | 2007-03-01 00:00:00 | NaN | 0.16 | NaN | NaN | NaN | 46.820000 | 51.480000 | NaN | 22.150000 | 5. |
| **225117** | 2007-03-01 00:00:00 | 0.24 | NaN | 0.20 | NaN | 0.09 | 51.259998 | 66.809998 | NaN | 18.540001 | 13. |
| **225118** | 2007-03-01 00:00:00 | 0.11 | NaN | 1.00 | NaN | 0.05 | 24.240000 | 36.930000 | NaN | NaN | 6. |
| **225119** | 2007-03-01 00:00:00 | 0.53 | 0.40 | 1.00 | 1.70 | 0.12 | 32.360001 | 47.860001 | 1.37 | 24.150000 | 10. |

225120 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225120 entries, 0 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     225120 non-null  object 
 1   BEN      68885 non-null   float64
 2   CO       206748 non-null  float64
 3   EBE      68883 non-null   float64
 4   MXY      26061 non-null   float64
 5   NMHC     86883 non-null   float64
 6   NO_2     223985 non-null  float64
 7   NOx      223972 non-null  float64
 8   OXY      26062 non-null   float64
 9   O_3      211850 non-null  float64
 10  PM10     222588 non-null  float64
 11  PM25     68870 non-null   float64
 12  PXY      26062 non-null   float64
 13  SO_2     224372 non-null  float64
 14  TCH      87026 non-null   float64
 15  TOL      68845 non-null   float64
 16  station  225120 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.2+ MB
```

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

Out[4]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2007-12-01 01:00:00 | 0.00 | 2.86 | 0.00 | 0.00 | 0.00 | 282.200012 | 1054.000000 | 0.00 | 4.030000 | 156. |
| **1** | 2007-12-01 01:00:00 | 0.00 | 1.82 | 0.00 | 0.00 | 0.00 | 86.419998 | 354.600006 | 0.00 | 3.260000 | 80. |
| **2** | 2007-12-01 01:00:00 | 0.00 | 1.47 | 0.00 | 0.00 | 0.00 | 94.639999 | 319.000000 | 0.00 | 5.310000 | 53. |
| **3** | 2007-12-01 01:00:00 | 0.00 | 1.64 | 0.00 | 0.00 | 0.00 | 127.900002 | 476.700012 | 0.00 | 4.500000 | 105. |
| **4** | 2007-12-01 01:00:00 | 4.64 | 1.86 | 4.26 | 7.98 | 0.57 | 145.100006 | 573.900024 | 3.49 | 52.689999 | 106. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **225115** | 2007-03-01 00:00:00 | 0.30 | 0.45 | 1.00 | 0.30 | 0.26 | 8.690000 | 11.690000 | 1.00 | 42.209999 | 6. |
| **225116** | 2007-03-01 00:00:00 | 0.00 | 0.16 | 0.00 | 0.00 | 0.00 | 46.820000 | 51.480000 | 0.00 | 22.150000 | 5. |
| **225117** | 2007-03-01 00:00:00 | 0.24 | 0.00 | 0.20 | 0.00 | 0.09 | 51.259998 | 66.809998 | 0.00 | 18.540001 | 13. |
| **225118** | 2007-03-01 00:00:00 | 0.11 | 0.00 | 1.00 | 0.00 | 0.05 | 24.240000 | 36.930000 | 0.00 | 0.000000 | 6. |
| **225119** | 2007-03-01 00:00:00 | 0.53 | 0.40 | 1.00 | 1.70 | 0.12 | 32.360001 | 47.860001 | 1.37 | 24.150000 | 10. |

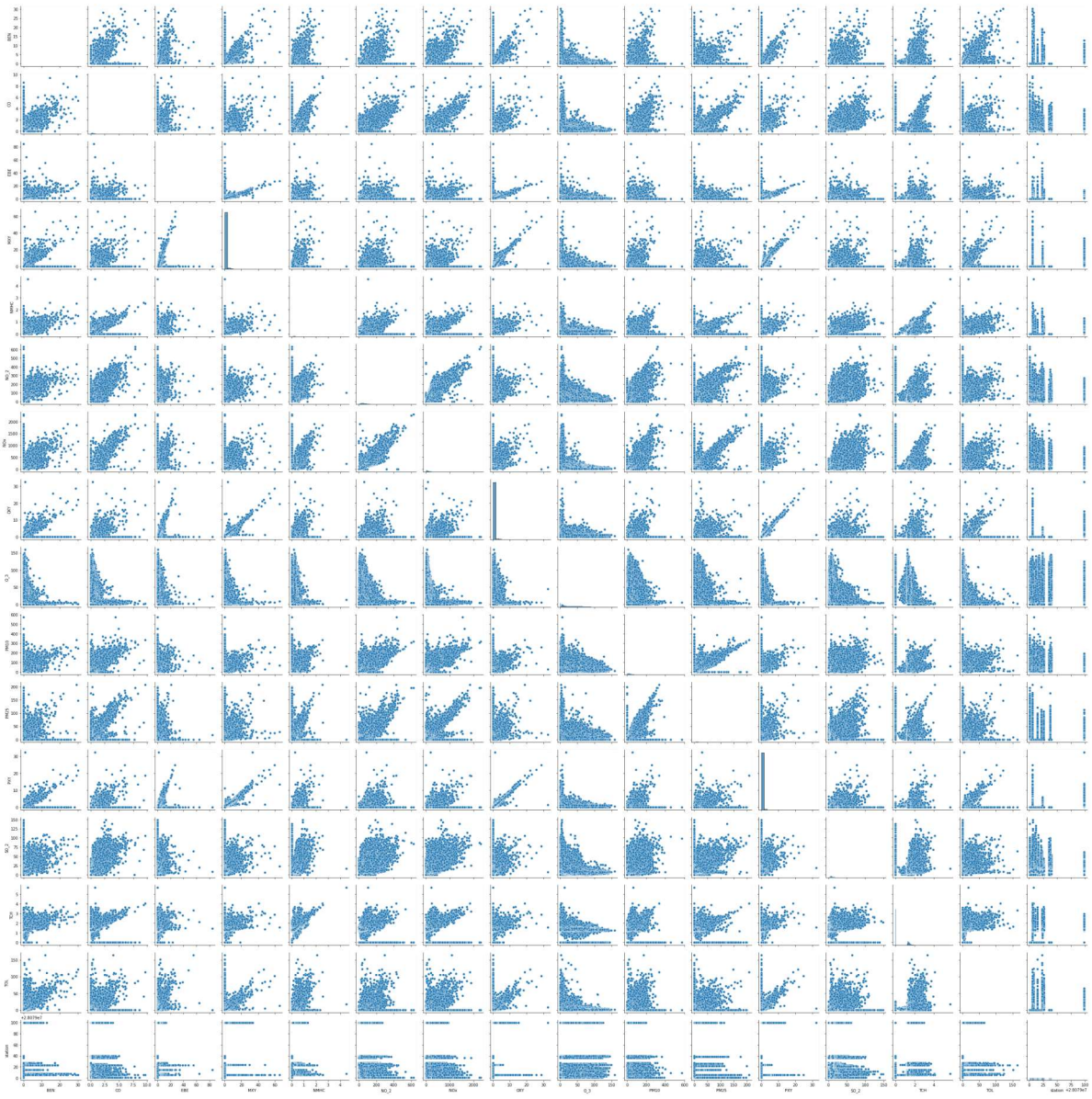
225120 rows × 17 columns

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

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

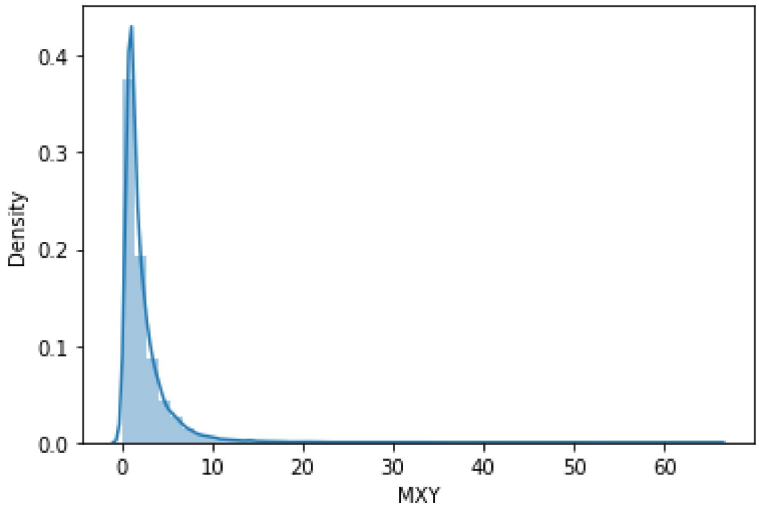
Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x1d25f444cd0>
```



In [7]:
```
sns.distplot(data["MXY"])
```

```
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[7]:
```
<AxesSubplot:xlabel='MXY', ylabel='Density'>
```




# MODEL BUILDING

## Linear Regression

In [8]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [9]:
```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
y=df1[['MXY']]
```

In [10]:
```
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 [11]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

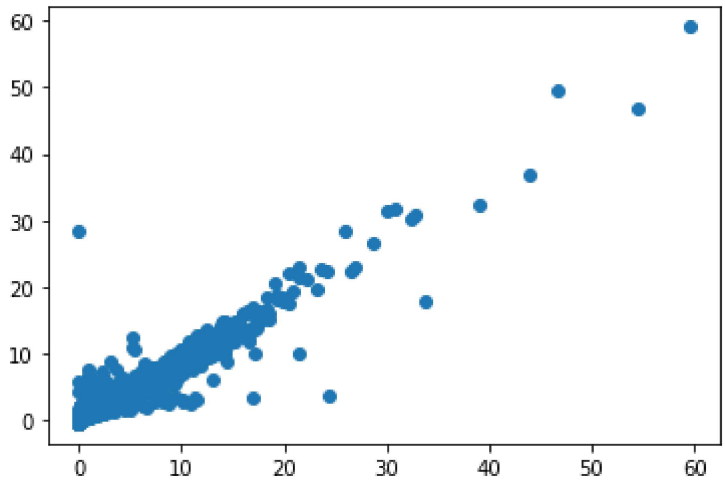
Out[11]:
```
LinearRegression()
```

In [12]:
```
print(lr.intercept_)
```

```
[60137.00665273]
```

In [13]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]:
```
<matplotlib.collections.PathCollection at 0x1d27ad03ca0>
```




In [14]:
```
print(lr.score(x_test,y_test))
```

```
0.9208879352770731
```

# Ridge Regression

In [15]:
```
from sklearn.linear_model import Ridge
```

In [16]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[16]:
```
Ridge(alpha=10)
```

In [17]:
```
rr.score(x_test,y_test)
```

Out[17]:
```
0.9208914849638712
```

# Lasso Regression

In [18]:
```
from sklearn.linear_model import Lasso
```

In [19]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[19]:
```
Lasso(alpha=10)
```

In [20]:
```
la.score(x_test,y_test)
```

Out[20]:
```
0.03030484287360058
```

# Elastic Regression

In [21]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[21]:
```
ElasticNet()
```

In [22]:
```
print(en.coef_)
```

```
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 -0.00000000e+00  9.20735284e-04  0.00000000e+00  8.74939038e-05
 -0.00000000e+00  0.00000000e+00  2.86363180e-03  0.00000000e+00
  8.59352857e-02  1.19900944e-02]
```

In [23]:
```
print(en.predict(x_test))
```

```
[-0.06371627  0.22555995  0.11190457 ... -0.00997036  0.31434668
  0.45095764]
```

In [24]:
```
print(en.score(x_test,y_test))
```

```
0.26637250502408616
```

# Logistic Regression

In [25]:
```
from sklearn.linear_model import LogisticRegression
```

In [26]:
```
feature_matrix=df1.iloc[:,0:15]
target_vector=df1.iloc[:,-1]
```

In [27]:
```
feature_matrix.shape
```

Out[27]:
```
(225120, 15)
```

In [28]:
```
target_vector.shape
```

Out[28]:
```
(225120,)
```

In [29]:
```
from sklearn.preprocessing import StandardScaler
```

In [30]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [31]:
```
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[31]:
```
LogisticRegression()
```

In [32]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]]
```

In [33]:
```
prediction=logr.predict(observation)
print(observation)
```

```
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]]
```

In [34]:
```
logr.classes_
```

Out[34]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [35]:
```
logr.score(fs,target_vector)
```

Out[35]:
```
0.892070895522388
```

# Random Forest

In [36]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [37]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
y=df1['station']
```

In [38]:
```
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 [39]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[39]:
```
RandomForestClassifier()
```

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

In [41]:
```
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[41]:
```
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 [42]:
```
grid_search.best_score_
```

Out[42]:
```
0.7915926320777067
```

In [43]:
```
rfc_best=grid_search.best_estimator_
```

In [44]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

```
value = [0, 0, 0, 0, 2, 0, 0, 37, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0]'),
 Text(3061.0285714285715, 181.19999999999982, 'gini = 0.008\nsamples = 154
\nvalue = [0, 0, 0, 0, 1, 0, 0, 259, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0]'),
 Text(3443.657142857143, 543.5999999999999, 'O_3 <= 8.945\ngini = 0.525\ns
amples = 2846\nvalue = [0, 0, 0, 11, 2597, 8, 0, 1664, 0, 0, 182, 0, 0\n0,
0, 0, 7, 15, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3316.114285714286, 181.19999999999982, 'gini = 0.616\nsamples = 488
\nvalue = [0, 0, 0, 7, 350, 3, 0, 310, 0, 0, 96, 0, 0\n0, 0, 0, 0, 4, 0,
0, 0, 0, 0, 0, 0, 0]'),
 Text(3571.2, 181.19999999999982, 'gini = 0.501\nsamples = 2358\nvalue =
[0, 0, 0, 4, 2247, 5, 0, 1354, 0, 0, 86, 0, 0\n0, 0, 0, 7, 11, 0, 0, 0, 0,
0, 0, 0, 0]'),
 Text(4208.914285714286, 1268.4, 'station <= 28079064.0\ngini = 0.875\nsam
ples = 13133\nvalue = [0, 0, 0, 2606, 0, 2483, 0, 0, 0, 0, 2500, 0, 0\n0,
0, 0, 2643, 2627, 0, 2605, 2642, 0, 0, 0, 0\n2784]'),
 Text(4081.3714285714286, 906.0, 'station <= 28079027.0\ngini = 0.857\nsam
ples = 11389\nvalue = [0, 0, 0, 2606, 0, 2483, 0, 0, 0, 0, 2500, 0, 0\n0,
0, 0, 2643, 2627, 0, 2605, 2642, 0, 0, 0, 0\n0]'),
```

# Results

The best model is Ridge Regression 0.9208914849638712

In [ ]: