In [1]:

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

In [117]:

```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_2004.csv")
df
```

Out[117]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.860001
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	NaN
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	NaN
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	NaN
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.860000
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	NaN
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.040000
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001	NaN
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.959999

245496 rows × 17 columns

In [118]:
```
df1 = df.fillna(0)
df1
```

Out[118]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	F
0	2004-08-01 01:00:00	0.00	0.66	0.00	0.00	0.00	89.550003	118.900002	0.00	40.020000	39.990002	25.860001	(
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	0.000000	3
2	2004-08-01 01:00:00	0.00	1.02	0.00	0.00	0.00	93.389999	138.600006	0.00	20.860001	49.480000	0.000000	(
3	2004-08-01 01:00:00	0.00	0.53	0.00	0.00	0.00	87.290001	105.000000	0.00	36.730000	31.070000	0.000000	(
4	2004-08-01 01:00:00	0.00	0.17	0.00	0.00	0.00	34.910000	35.349998	0.00	86.269997	54.080002	0.000000	(
...	...	...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.860000	(
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	0.00	97.139999	146.899994	2.34	7.740000	37.689999	0.000000	2
245493	2004-06-01 00:00:00	0.00	0.00	0.00	0.00	0.13	102.699997	132.600006	0.00	17.809999	22.840000	12.040000	(
245494	2004-06-01 00:00:00	0.00	0.00	0.00	0.00	0.09	82.599998	102.599998	0.00	0.000000	45.630001	0.000000	(
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.959999	2

245496 rows × 17 columns

In [119]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      65158 non-null   float64
 2   CO       226043 non-null  float64
 3   EBE      56781 non-null   float64
 4   MXY      39867 non-null   float64
 5   NMHC     107630 non-null  float64
 6   NO_2     243280 non-null  float64
 7   NOx      243283 non-null  float64
 8   OXY      39882 non-null   float64
 9   O_3      233811 non-null  float64
 10  PM10     234655 non-null  float64
 11  PM25     58145 non-null   float64
 12  PXY      39891 non-null   float64
 13  SO_2     243402 non-null  float64
 14  TCH      107650 non-null  float64
 15  TOL      64914 non-null   float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ MB
```

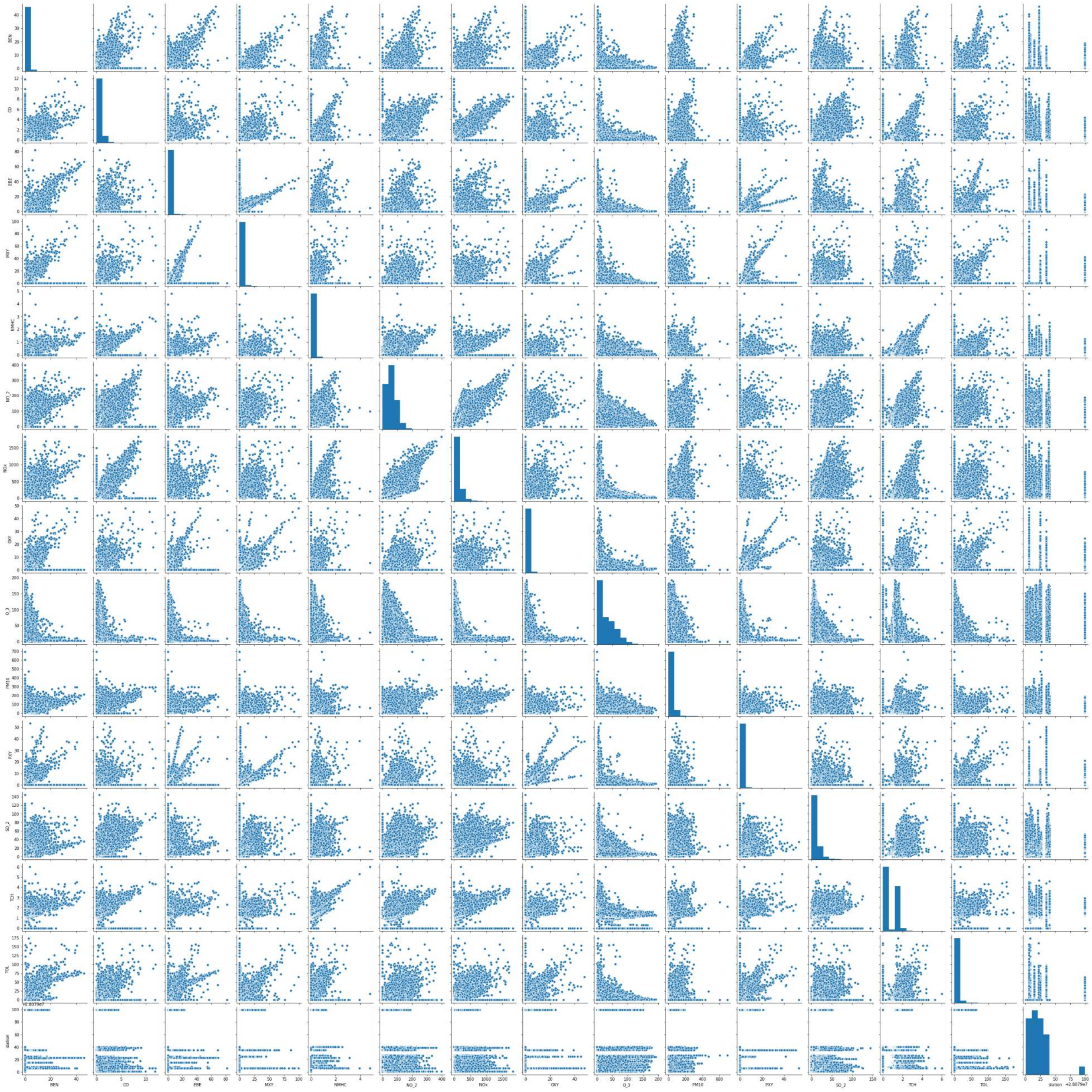
In [120]:
```
df.columns
```

Out[120]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [121]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

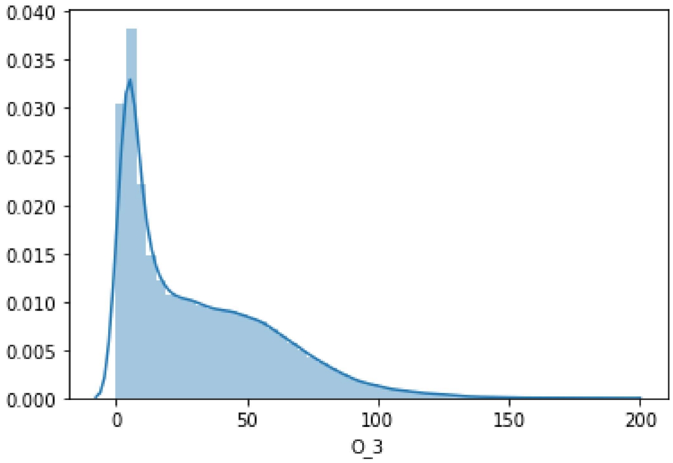
In [122]:
```
sns.pairplot(df2)
```

Out[122]:
```
<seaborn.axisgrid.PairGrid at 0x233803edf40>
```



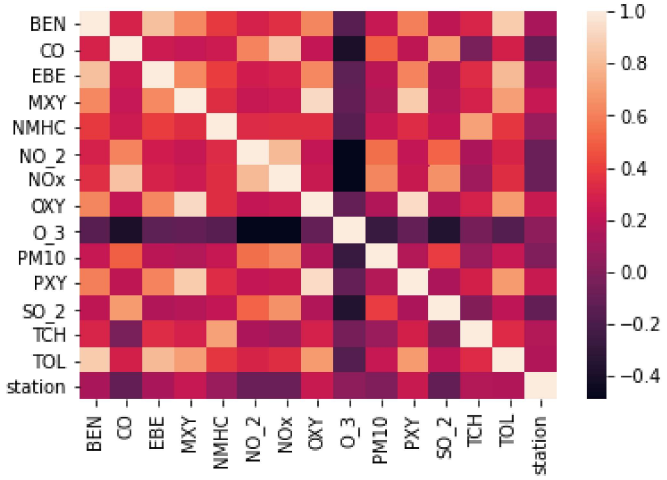
```
In [123]: sns.distplot(df2['O_3'])
```

```
Out[123]: <matplotlib.axes._subplots.AxesSubplot at 0x233cb0e97f0>
```



```
In [124]: sns.heatmap(df2.corr())
```

```
Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x233cb0da760>
```



Linear Regression

```
In [125]: x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
                 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
          y = df2['O_3']
```

```
In [126]: 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 [127]: from sklearn.linear_model import LinearRegression

          lr=LinearRegression()
          lr.fit(x_train,y_train)
```

```
Out[127]: LinearRegression()
```

In [128]:
```
print(lr.intercept_)
```

```
52.575372173875785
```

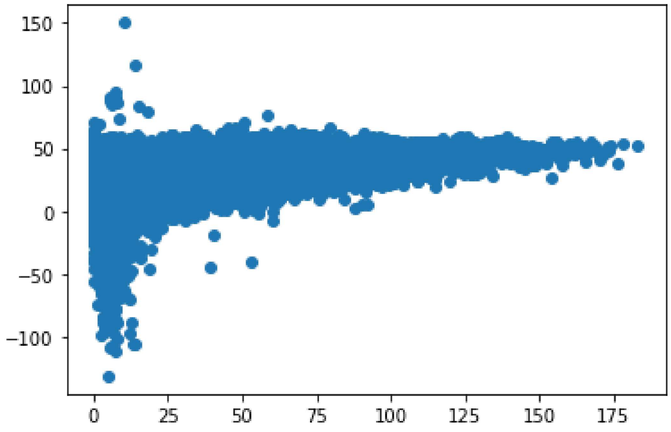
In [129]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[129]:

	Co-effecient
BEN	1.445308
CO	4.346790
EBE	0.357843
MXY	-0.205967
NMHC	9.029511
NO_2	-0.214476
NOx	-0.092395
OXY	0.500563
PM10	0.080688
PXY	0.537244
SO_2	-0.216725
TCH	-0.353907
TOL	-0.502155

In [130]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[130]:
```
<matplotlib.collections.PathCollection at 0x23343f02a30>
```



In [131]:
```
print(lr.score(x_test,y_test))
```

```
0.27848502134774233
```

In [132]:
```
lr.score(x_train,y_train)
```

Out[132]:
```
0.27536834626506734
```

Ridge Lasso

```
In [133]: from sklearn.linear_model import Ridge,Lasso
```

```
In [134]: rr=Ridge(alpha=10)
          rr.fit(x_train,y_train)
          rr.score(x_test,y_test)

Out[134]: 0.27848660273191406
```

```
In [135]: predict2=(rr.predict(x_test))
```

```
In [136]: la=Lasso(alpha=10)
          la.fit(x_train,y_train)

Out[136]: Lasso(alpha=10)
```

```
In [137]: la.score(x_test,y_test)

Out[137]: 0.2707189798655314
```

Elastic Net regression

```
In [138]: from sklearn.linear_model import ElasticNet
          en=ElasticNet()
          en.fit(x_train,y_train)

Out[138]: ElasticNet()
```

```
In [139]: print(en.coef_)

          [ 0.3556417   0.          0.3480369   0.09368703  0.         -0.22289699
           -0.07147375  0.06541692  0.07677801  0.00629613 -0.13864582  0.
           -0.18489377]
```

```
In [140]: print(en.intercept_)

          52.7536621756676
```

```
In [141]: print(en.score(x_test,y_test))

          0.2746437307982825
```

```
In [142]: print(en.score(x_train,y_train))

          0.2702125917708419
```

Logistic Regression

```
In [143]: from sklearn.linear_model import LogisticRegression
```

```
In [144]: feature_matrix=df2.iloc[:,0:5]
          target_vector=df2.iloc[:,-1]
```

```
In [145]: from sklearn.preprocessing import StandardScaler
```

In [146]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [147]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Convergen
ceWarning: 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/st
able/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[147]:
```
LogisticRegression()
```

In [148]:
```
df2.shape
```

Out[148]:
```
(245496, 15)
```

In [149]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [150]:
```
print(predication)
```

```
[28079035]
```

In [151]:
```
logr.classes_
```

Out[151]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079026, 28079027, 28079035, 28079036,
       28079038, 28079039, 28079040, 28079099], dtype=int64)
```

In [152]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(feature_matrix,target_vector,test_size=0.30
```

In [153]:
```
print(logr.score(x_test,y_test))
```

```
0.08650490841695067
```

In [154]:
```
print(logr.score(x_train,y_train))
```

```
0.0847905404225852
```

Conclusion ¶

Linear Regression is bestfit model

The Score x_test,y_test is 0.27848502134774233 and x_train,y_train score is 0.27536834626506734

In [ ]:

In [ ]: