In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [45]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_20
df
```

Out[45]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	I
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	I
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	I
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	I
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	I
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	I
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150

209448 rows × 17 columns

In [46]:
```
df1 = df.fillna(0)
df1
```

Out[46]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	0.00	0.29	0.00	0.0	0.00	25.090000	29.219999	0.0	68.930000	0.000
1	2010-03-01 01:00:00	0.00	0.27	0.00	0.0	0.00	24.879999	30.040001	0.0	0.000000	0.000
2	2010-03-01 01:00:00	0.00	0.28	0.00	0.0	0.00	17.410000	20.540001	0.0	72.120003	0.000
3	2010-03-01 01:00:00	0.38	0.24	1.74	0.0	0.05	15.610000	21.080000	0.0	72.970001	19.410
4	2010-03-01 01:00:00	0.79	0.00	1.32	0.0	0.00	21.430000	26.070000	0.0	0.000000	24.670
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	0.00	0.55	0.00	0.0	0.00	125.000000	219.899994	0.0	25.379999	0.000
209444	2010-08-01 00:00:00	0.00	0.27	0.00	0.0	0.00	45.709999	47.410000	0.0	0.000000	51.259
209445	2010-08-01 00:00:00	0.00	0.00	0.00	0.0	0.24	46.560001	49.040001	0.0	46.250000	0.000
209446	2010-08-01 00:00:00	0.00	0.00	0.00	0.0	0.00	46.770000	50.119999	0.0	77.709999	0.000
209447	2010-08-01 00:00:00	0.92	0.43	0.71	0.0	0.25	76.330002	88.190002	0.0	52.259998	47.150

209448 rows × 17 columns

In [47]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209448 entries, 0 to 209447
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209448 non-null  object 
 1   BEN      60268 non-null   float64
 2   CO       94982 non-null   float64
 3   EBE      60253 non-null   float64
 4   MXY      6750 non-null    float64
 5   NMHC     51727 non-null   float64
 6   NO_2     208219 non-null  float64
 7   NOx      208210 non-null  float64
 8   OXY      6750 non-null    float64
 9   O_3      126684 non-null  float64
 10  PM10     106186 non-null  float64
 11  PM25     55514 non-null   float64
 12  PXY      6740 non-null    float64
 13  SO_2     93184 non-null   float64
 14  TCH      51730 non-null   float64
 15  TOL      60171 non-null   float64
 16  station  209448 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 27.2+ MB
```

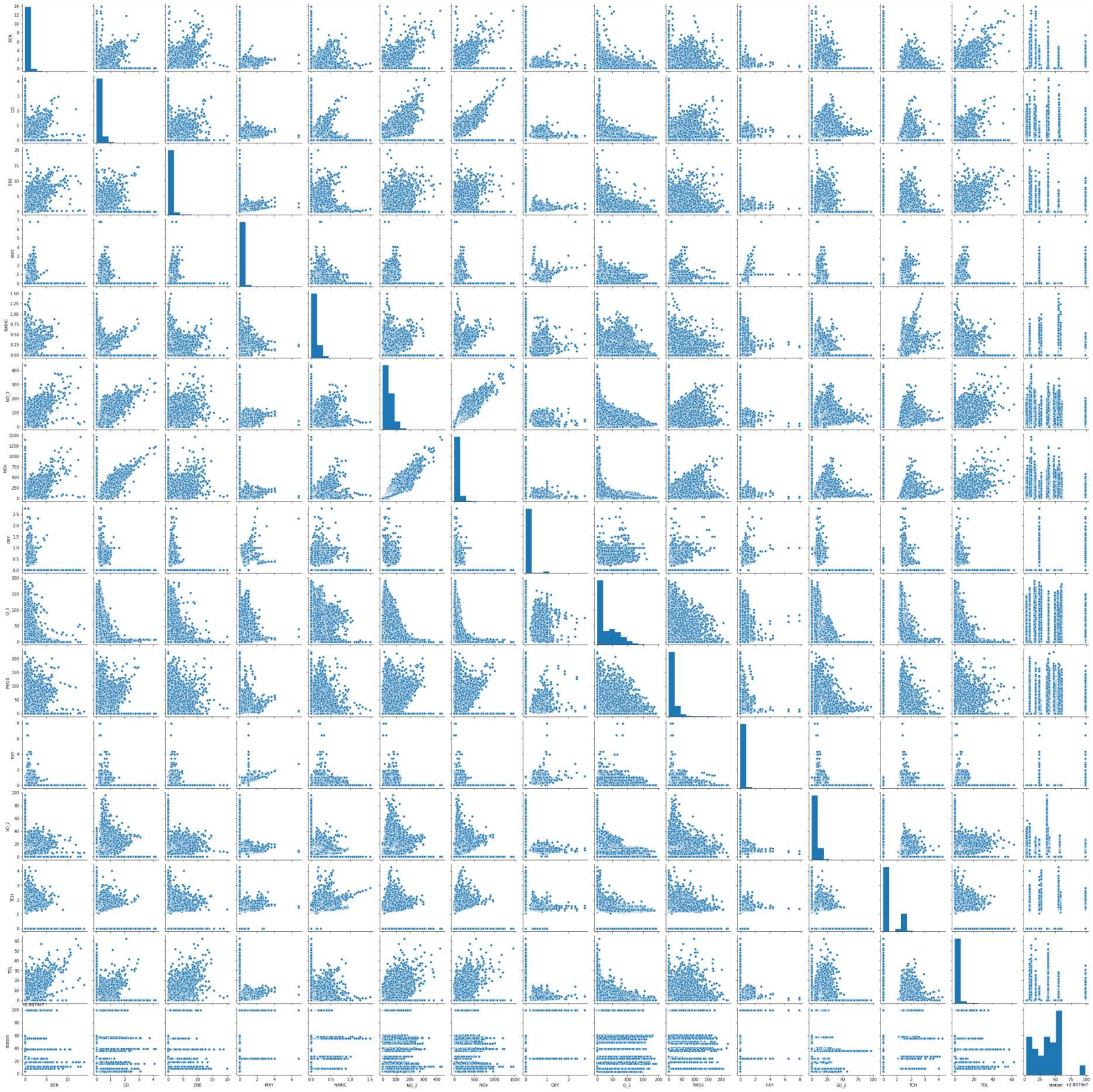
In [48]:
```
df.columns
```

Out[48]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [49]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

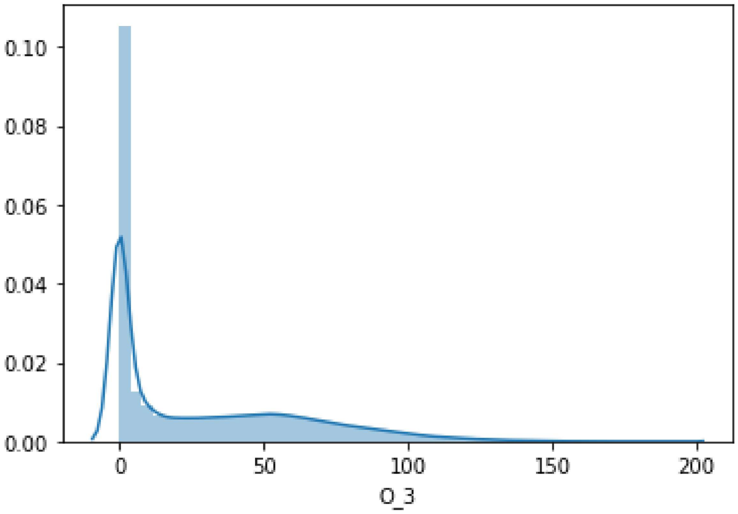
In [50]:
```
sns.pairplot(df2)
```

Out[50]:
```
<seaborn.axisgrid.PairGrid at 0x1ac0db62820>
```



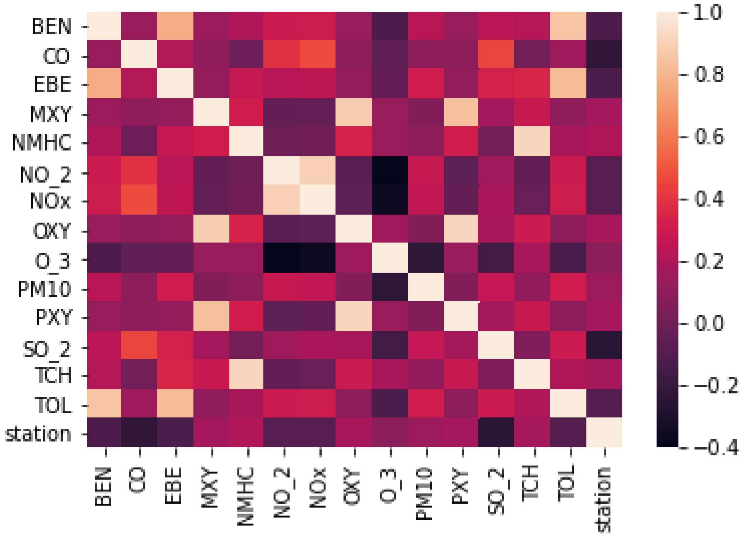
In [51]:
```
sns.distplot(df2['O_3'])
```

Out[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac5f63f1f0>
```



In [52]:
```
sns.heatmap(df2.corr())
```

Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac5ff29670>
```



Linear Regression

In [53]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [54]:
```
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 [55]:

```
from sklearn.linear_model import LinearRegression

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

Out[55]:

```
LinearRegression()
```

In [56]:

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

```
47.43549509386216
```

In [57]:

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

Out[57]:

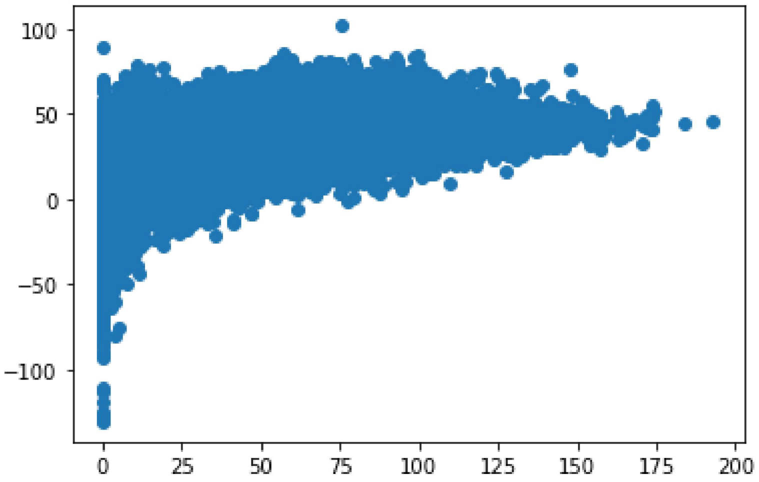
	Co-effecient
BEN	-3.270166
CO	27.242984
EBE	3.883891
MXY	-3.514638
NMHC	-48.684393
NO_2	-0.404697
NOx	-0.016305
OXY	30.458939
PM10	-0.245842
PXY	-4.226543
SO_2	-1.182267
TCH	14.136102
TOL	0.015777

In [58]:

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

Out[58]:

```
<matplotlib.collections.PathCollection at 0x1ac61185c70>
```



```
In [59]: print(lr.score(x_test,y_test))

         0.258645747289049
```

```
In [60]: lr.score(x_train,y_train)

Out[60]: 0.2571856671711411
```

Ridge Lasso

```
In [61]: from sklearn.linear_model import Ridge,Lasso
```

```
In [62]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)
         rr.score(x_test,y_test)

Out[62]: 0.25865307425094985
```

```
In [63]: predict2=(rr.predict(x_test))
```

```
In [64]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[64]: Lasso(alpha=10)
```

```
In [65]: la.score(x_test,y_test)

Out[65]: 0.18152281962192107
```

Elastic Net regression

```
In [66]: from sklearn.linear_model import ElasticNet
         en=ElasticNet()
         en.fit(x_train,y_train)

Out[66]: ElasticNet()
```

```
In [67]: print(en.coef_)

         [ 0.          0.8909809   1.49547789  0.19498996  0.          -0.43545558
           0.02069334  0.38943572 -0.25780997  0.14675393 -0.52754491  3.376013
           0.        ]
```

```
In [68]: print(en.intercept_)

         49.839760307248845
```

```
In [69]: print(en.score(x_test,y_test))

         0.21074452488157325
```

In [70]:
```
print(en.score(x_train,y_train))
```

```
0.20880071012353074
```

Logistic Regression

In [71]:
```
from sklearn.linear_model import LogisticRegression
```

In [72]:
```
feature_matrix=df2.iloc[:,0:5]
target_vector=df2.iloc[:,-1]
```

In [73]:
```
from sklearn.preprocessing import StandardScaler
```

In [74]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [75]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
762: 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[75]:
```
LogisticRegression()
```

In [76]:
```
df2.shape
```

Out[76]:
```
(209448, 15)
```

In [77]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [78]:
```
print(predication)
```

```
[28079099]
```

In [79]:
```
logr.classes_
```

Out[79]:
```
array([28079003, 28079004, 28079008, 28079011, 28079016, 28079017,
       28079018, 28079024, 28079026, 28079027, 28079036, 28079038,
       28079039, 28079040, 28079047, 28079048, 28079049, 28079050,
       28079054, 28079055, 28079056, 28079057, 28079058, 28079059,
       28079060, 28079099], dtype=int64)
```

In [80]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(feature_matrix,target_vector,t
```

In [81]:

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

```
0.06501153815548659
```

In [82]:

```
print(logr.score(x_train,y_train))
```

```
0.06485782297613445
```

Conclusion ¶

Ridge Regression is bestfit model

The Score x_test,y_test is 0.0.258645747289049

In [ ]:

```
```

In [ ]:

```
```