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

In [269]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_20
df
```

Out[269]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.889
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.040
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.270
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.850
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.020
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.540
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.910
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690

226392 rows × 17 columns

In [270]:

```
df1 = df.fillna(0)
df1
```

Out[270]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2008-06-01 01:00:00	0.00	0.47	0.00	0.00	0.00	83.089996	120.699997	0.00	16.990000	16.889
1	2008-06-01 01:00:00	0.00	0.59	0.00	0.00	0.00	94.820000	130.399994	0.00	17.469999	19.040
2	2008-06-01 01:00:00	0.00	0.55	0.00	0.00	0.00	75.919998	104.599998	0.00	13.470000	20.270
3	2008-06-01 01:00:00	0.00	0.36	0.00	0.00	0.00	61.029999	66.559998	0.00	23.110001	10.850
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450
226388	2008-11-01 00:00:00	0.00	0.30	0.00	0.00	0.00	41.880001	48.500000	0.00	35.830002	15.020
226389	2008-11-01 00:00:00	0.25	0.00	0.56	0.00	0.11	83.610001	102.199997	0.00	14.130000	17.540
226390	2008-11-01 00:00:00	0.54	0.00	2.70	0.00	0.18	70.639999	81.860001	0.00	0.000000	11.910
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690

226392 rows × 17 columns

In [271]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 226392 entries, 0 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     226392 non-null  object 
 1   BEN      67047 non-null   float64
 2   CO       208109 non-null  float64
 3   EBE      67044 non-null   float64
 4   MXY      25867 non-null   float64
 5   NMHC     85079 non-null   float64
 6   NO_2     225315 non-null  float64
 7   NOx      225311 non-null  float64
 8   OXY      25878 non-null   float64
 9   O_3      215716 non-null  float64
 10  PM10     220179 non-null  float64
 11  PM25     67833 non-null   float64
 12  PXY      25877 non-null   float64
 13  SO_2     225405 non-null  float64
 14  TCH      85107 non-null   float64
 15  TOL      66940 non-null   float64
 16  station  226392 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.4+ MB
```

In [272]:

```
df.columns
```

Out[272]:

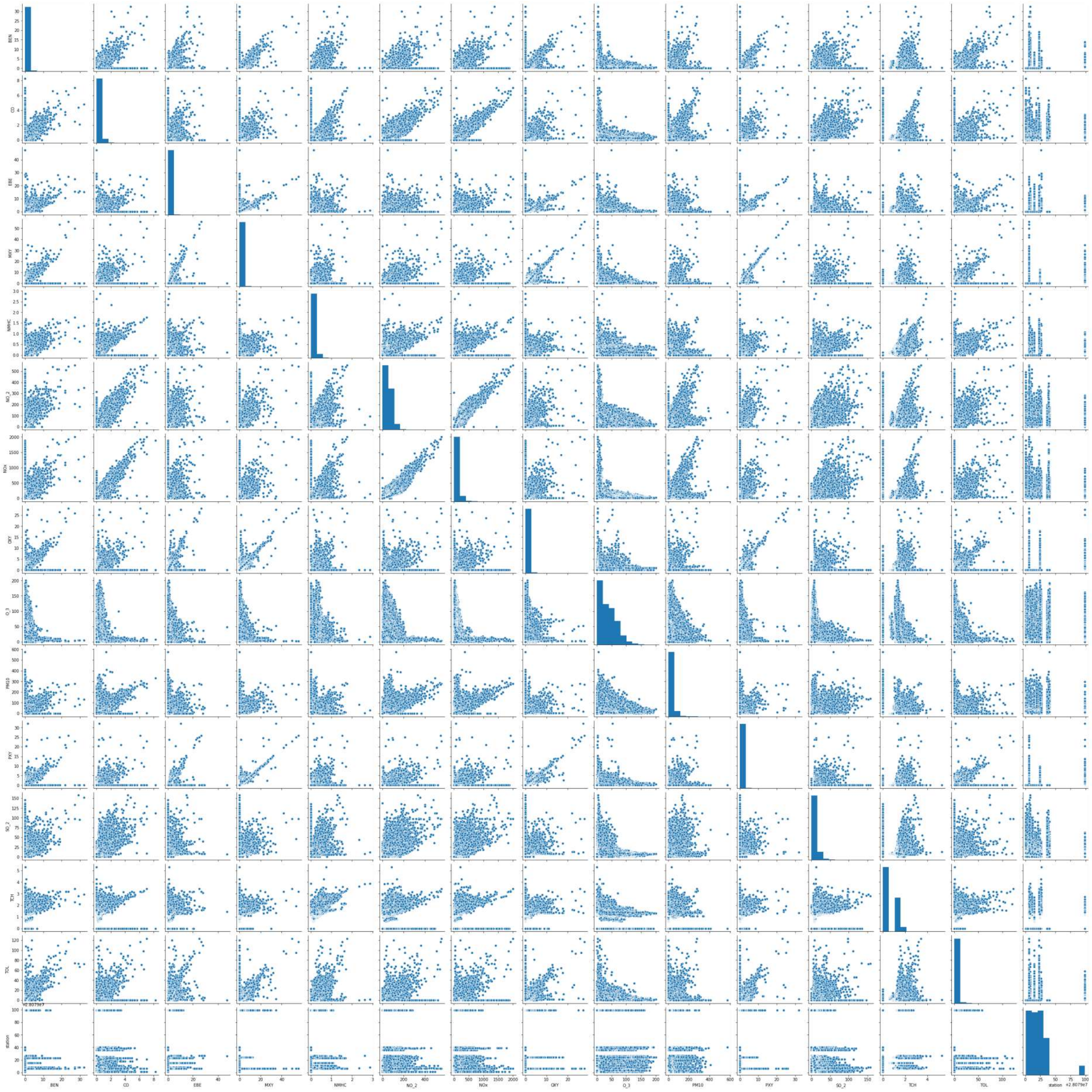
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [273]:

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

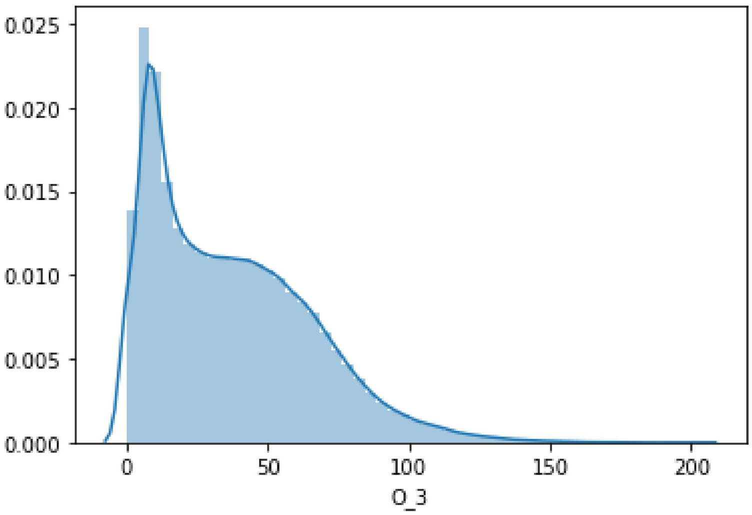
In [274]:
```
sns.pairplot(df2)
```

Out[274]:
```
<seaborn.axisgrid.PairGrid at 0x233e3762460>
```



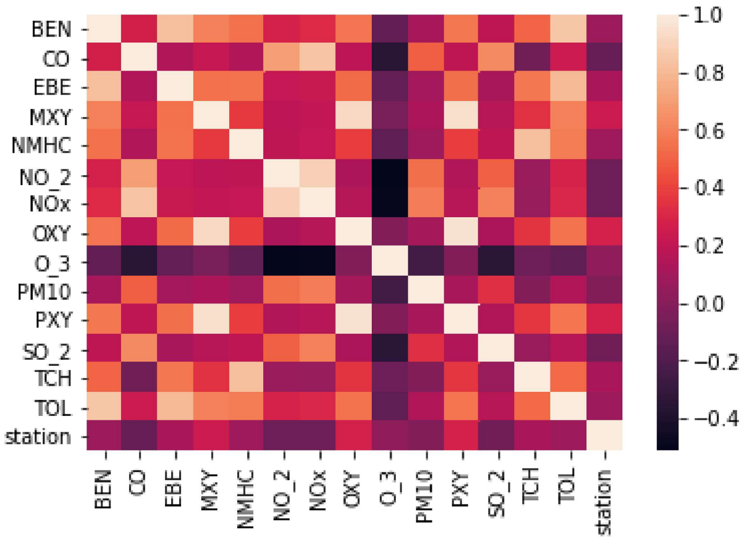
In [275]:
```
sns.distplot(df2['O_3'])
```

Out[275]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2350b2fb4f0>
```



In [276]:
```
sns.heatmap(df2.corr())
```

Out[276]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2350dd62910>
```



Linear Regression

In [277]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [278]:
```
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 [279]:

```
from sklearn.linear_model import LinearRegression

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

Out[279]:

```
LinearRegression()
```

In [280]:

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

```
59.08351350855042
```

In [281]:

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

Out[281]:

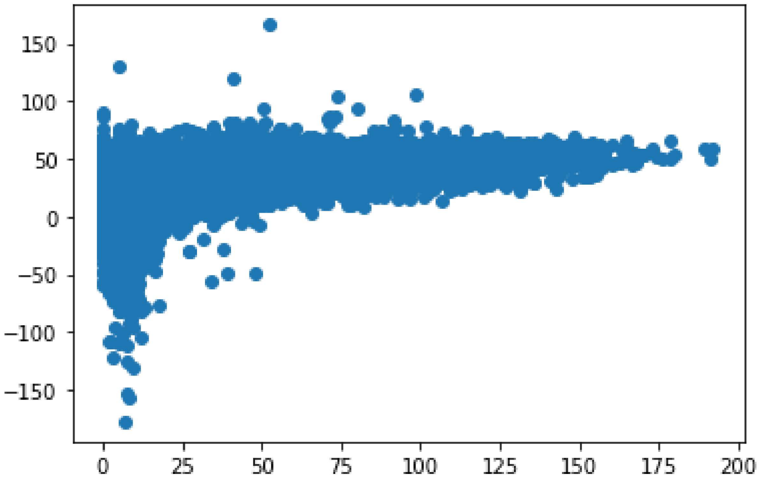
	Co-effecient
BEN	3.804848
CO	16.892690
EBE	-1.916293
MXY	-2.446715
NMHC	-2.624988
NO_2	-0.282340
NOx	-0.099939
OXY	3.144582
PM10	0.097670
PXY	4.304703
SO_2	-0.512151
TCH	-1.584020
TOL	-0.091192

In [282]:

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

Out[282]:

```
<matplotlib.collections.PathCollection at 0x23511da4dc0>
```



In [283]:

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

```
0.30472025600643393
```

In [284]:

```
lr.score(x_train,y_train)
```

Out[284]:

```
0.2973802470196698
```

Ridge Lasso

In [285]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [286]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[286]:

```
0.30472134365526515
```

In [287]:

```
predict2=(rr.predict(x_test))
```

In [288]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[288]:

```
Lasso(alpha=10)
```

In [289]:

```
la.score(x_test,y_test)
```

Out[289]:

```
0.2802924270178049
```

Elastic Net regression

In [290]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[290]:

```
ElasticNet()
```

In [291]:

```
print(en.coef_)
```

```
[ 0.0162072   0.          -0.           0.59858908 -0.          -0.32385527
 -0.04199357  0.40519487  0.09301885  0.14153153 -0.35292605 -0.59160507
  0.03162004]
```

In [292]:

```
print(en.intercept_)
```

```
60.34758566862013
```

In [293]:

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

```
0.2885227128727964
```

In [294]:

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

```
0.2806035137518861
```

Logistic Regression

In [295]:

```
from sklearn.linear_model import LogisticRegression
```

In [296]:

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

In [297]:

```
from sklearn.preprocessing import StandardScaler
```

In [298]:

```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [299]:
```
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[299]:
```
LogisticRegression()
```

In [300]:
```
df2.shape
```

Out[300]:
```
(226392, 15)
```

In [301]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [302]:
```
print(predication)
```

```
[28079099]
```

In [303]:
```
logr.classes_
```

Out[303]:
```
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 [304]:
```
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 [305]:
```
print(logr.score(x_test,y_test))
```

```
0.10528873052798964
```

In [306]:
```
print(logr.score(x_train,y_train))
```

```
0.10626348801696177
```

Conclusion

Linear Regression is bestfit model

The Score x_test,y_test is 0.30472025600643393 and x_train,y_train score is 0.2973802470196698

```
In [ ]:
```

```
In [ ]:
```