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

In [193]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_2006.csv")
df
```

Out[193]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	P
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570000	40.259998	N
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000	NaN	2
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419998	NaN	N
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260000	NaN	N
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180000	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120003	NaN	N
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469999	15.640000	(
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680000	35.000000	N
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360001	NaN	N
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490002	27.920000	1

230568 rows × 17 columns

In [194]:
```
df1 = df.fillna(0)
df1
```

Out[194]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	P
0	2006-02-01 01:00:00	0.00	1.84	0.00	0.00	0.00	155.100006	490.100006	0.00	4.880000	97.570000	40.259998	0
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000	0.000000	2
2	2006-02-01 01:00:00	0.00	1.25	0.00	0.00	0.00	66.800003	192.000000	0.00	4.430000	34.419998	0.000000	0
3	2006-02-01 01:00:00	0.00	1.68	0.00	0.00	0.00	103.000000	407.799988	0.00	4.830000	28.260000	0.000000	0
4	2006-02-01 01:00:00	0.00	1.31	0.00	0.00	0.00	105.400002	269.200012	0.00	6.990000	54.180000	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	0.00	0.20	112.500000	218.000000	0.00	24.389999	93.120003	0.000000	0
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469999	15.640000	0
230565	2006-05-01 00:00:00	0.96	0.00	0.69	0.00	0.19	135.100006	179.199997	0.00	11.460000	64.680000	35.000000	0
230566	2006-05-01 00:00:00	0.50	0.00	0.67	0.00	0.10	82.599998	105.599998	0.00	0.000000	94.360001	0.000000	0
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490002	27.920000	1

230568 rows × 17 columns

In [195]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      73979 non-null   float64
 2   CO       211665 non-null  float64
 3   EBE      73948 non-null   float64
 4   MXY      33422 non-null   float64
 5   NMHC     90829 non-null   float64
 6   NO_2     228855 non-null  float64
 7   NOx      228855 non-null  float64
 8   OXY      33472 non-null   float64
 9   O_3      216511 non-null  float64
 10  PM10     227469 non-null  float64
 11  PM25     61758 non-null   float64
 12  PXY      33447 non-null   float64
 13  SO_2     229125 non-null  float64
 14  TCH      90887 non-null   float64
 15  TOL      73840 non-null   float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

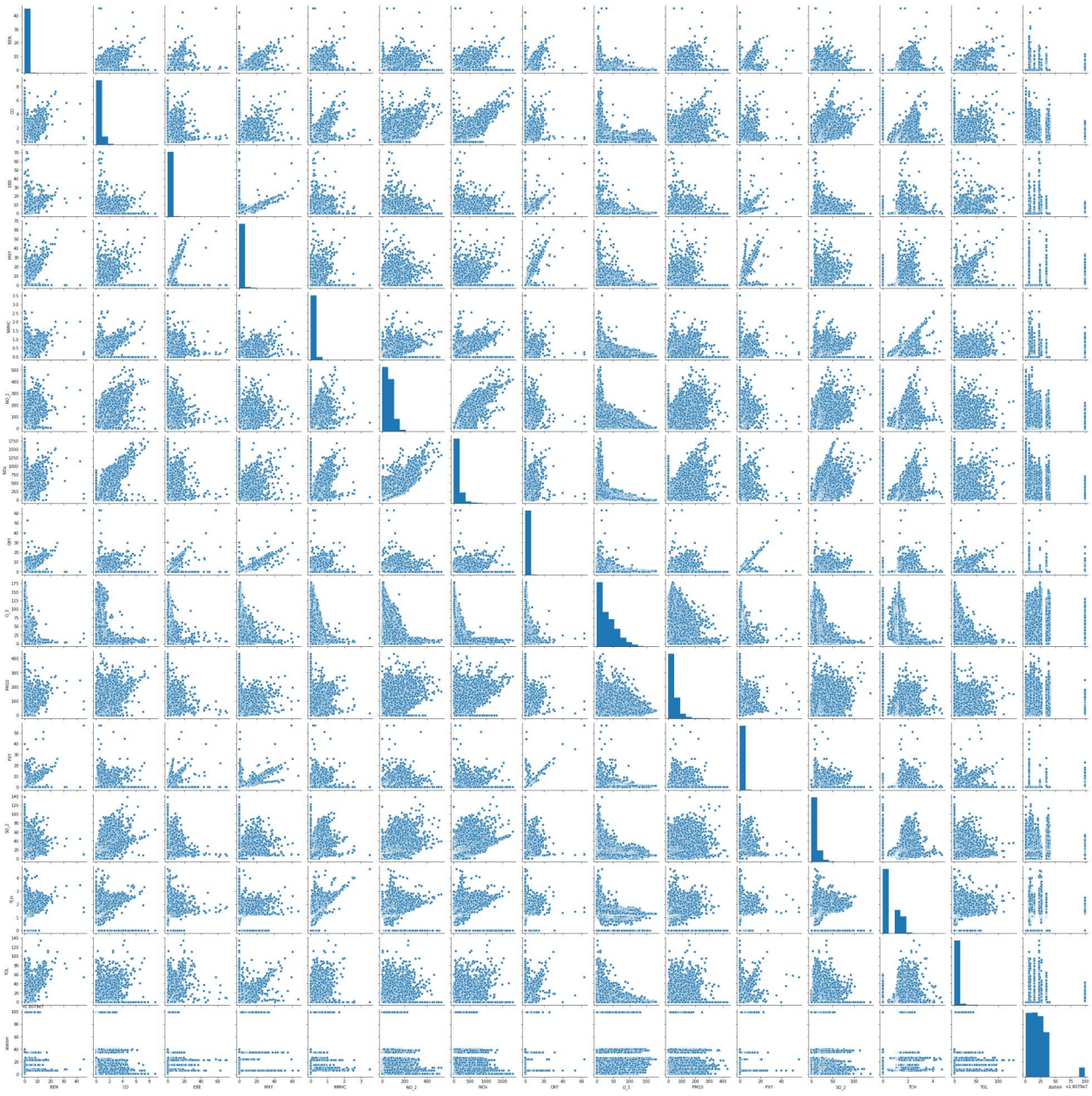
In [196]:
```
df.columns
```

Out[196]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [197]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

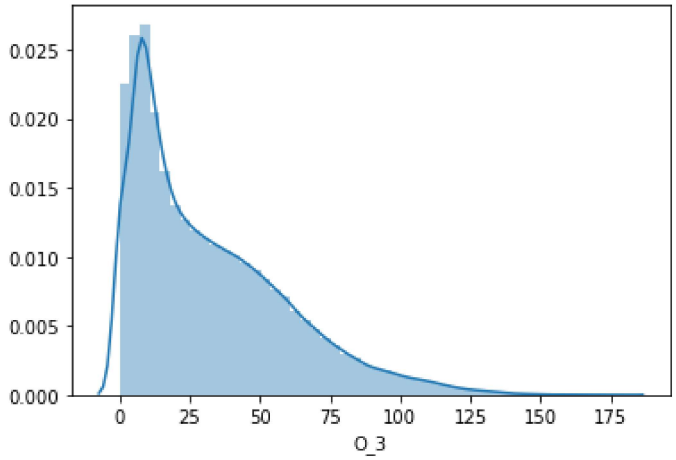
In [198]:
```
sns.pairplot(df2)
```

Out[198]:
```
<seaborn.axisgrid.PairGrid at 0x233ce4133a0>
```



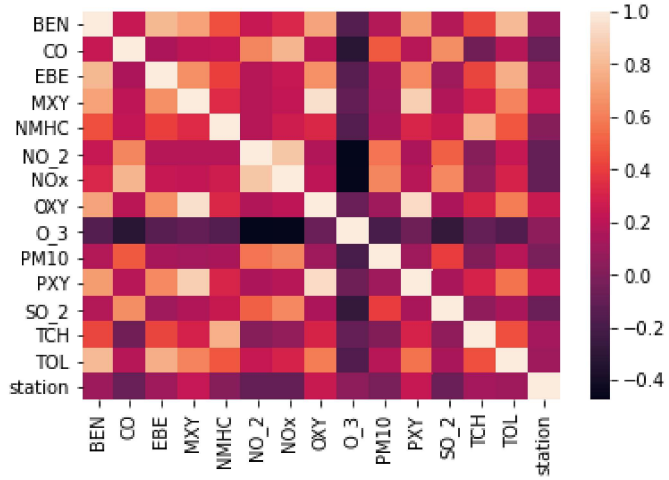
In [199]:
```
sns.distplot(df2['O_3'])
```

Out[199]: <matplotlib.axes._subplots.AxesSubplot at 0x23482a36250>



In [200]:
```
sns.heatmap(df2.corr())
```

Out[200]: <matplotlib.axes._subplots.AxesSubplot at 0x2348a86b820>



Linear Regression

In [201]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [202]:
```
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 [203]:
```
from sklearn.linear_model import LinearRegression

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

Out[203]: LinearRegression()

In [204]:
```
print(lr.intercept_)
```

```
48.70948843965615
```

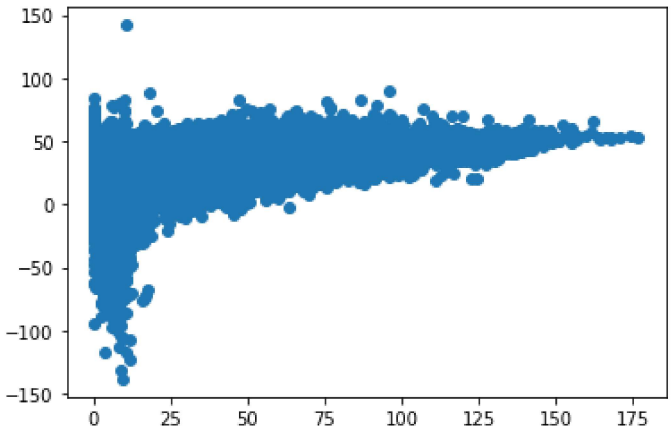
In [205]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[205]:

	Co-effecient
BEN	1.298143
CO	5.717882
EBE	-1.215999
MXY	-1.334329
NMHC	0.350453
NO_2	-0.206404
NOx	-0.103074
OXY	2.470965
PM10	0.164328
PXY	1.715296
SO_2	0.008183
TCH	-2.220611
TOL	-0.100015

In [206]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[206]:
```
<matplotlib.collections.PathCollection at 0x233e346b8b0>
```



In [207]:
```
print(lr.score(x_test,y_test))
```

```
0.2732734219484818
```

In [208]:
```
lr.score(x_train,y_train)
```

Out[208]:
```
0.2737504047834287
```

Ridge Lasso

In [209]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [210]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[210]: 0.27327253795377404

In [211]:
```
predict2=(rr.predict(x_test))
```

In [212]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[212]: Lasso(alpha=10)

In [213]:
```
la.score(x_test,y_test)
```

Out[213]: 0.26023695708625205

Elastic Net regression

In [214]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[214]: ElasticNet()

In [215]:
```
print(en.coef_)
```

```
[ 0.          0.         -0.          0.07162855 -0.         -0.21482009
 -0.08444485  0.16272789  0.16251771  0.34493999  0.07897562 -0.60081056
 -0.20381564]
```

In [216]:
```
print(en.intercept_)
```

```
48.80915507565707
```

In [217]:
```
print(en.score(x_test,y_test))
```

```
0.2642392461732971
```

In [218]:
```
print(en.score(x_train,y_train))
```

```
0.2660186900621544
```

Logistic Regression

In [219]:
```
from sklearn.linear_model import LogisticRegression
```

In [220]:
```
feature_matrix=df2.iloc[:,0:5]
target_vector=df2.iloc[:,-1]
```

In [221]:
```
from sklearn.preprocessing import StandardScaler
```

In [222]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [223]:
```
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[223]:
```
LogisticRegression()
```

In [224]:
```
df2.shape
```

Out[224]:
```
(230568, 15)
```

In [225]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [226]:
```
print(predication)
```

```
[28079099]
```

In [227]:
```
logr.classes_
```

Out[227]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 28079035, 28079036, 28079038,
       28079039, 28079040, 28079099], dtype=int64)
```

In [228]:
```
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 [229]:
```
print(logr.score(x_test,y_test))
```

```
0.0670367639617759
```

In [230]:
```
print(logr.score(x_train,y_train))
```

```
0.0665501837084952
```

Conclusion ¶

Linear Regression is bestfit model

The Score x_test,y_test is 0.2732734219484818 and x_train,y_train score is 0.2737504047834287

In [ ]:

In [ ]: