In [1]:

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

In [79]:

```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_2003.csv")
df
```

Out[79]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	24.2999
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	14.2300
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	17.8799
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	24.9000
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	18.7500
...	...	...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.8700
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50	8.3600
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN	5.3300
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN	6.8300
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.0600

243984 rows × 16 columns

In [80]:

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

Out[80]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO
0	2003-03-01 01:00:00	0.00	1.72	0.00	0.00	0.00	73.900002	316.299988	0.00	10.550000	55.209999	0.00	24.2999
1	2003-03-01 01:00:00	0.00	1.45	0.00	0.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999	0.00	14.2300
2	2003-03-01 01:00:00	0.00	1.57	0.00	0.00	0.00	80.559998	224.199997	0.00	21.049999	63.240002	0.00	17.8799
3	2003-03-01 01:00:00	0.00	2.45	0.00	0.00	0.00	78.370003	450.399994	0.00	4.220000	67.839996	0.00	24.9000
4	2003-03-01 01:00:00	0.00	3.26	0.00	0.00	0.00	96.250000	479.100006	0.00	8.460000	95.779999	0.00	18.7500
...	...	...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.8700
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	0.00	10.450000	14.760000	1.00	34.610001	7.400000	0.50	8.3600
243981	2003-10-01 00:00:00	0.00	0.00	0.00	0.00	0.07	34.639999	50.810001	0.00	32.160000	16.830000	0.00	5.3300
243982	2003-10-01 00:00:00	0.00	0.00	0.00	0.00	0.07	32.580002	41.020000	0.00	0.000000	13.570000	0.00	6.8300
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.0600

243984 rows × 16 columns

In [81]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243984 entries, 0 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     243984 non-null  object 
 1   BEN      69745 non-null   float64
 2   CO       225340 non-null  float64
 3   EBE      61244 non-null   float64
 4   MXY      42045 non-null   float64
 5   NMHC     111951 non-null  float64
 6   NO_2     242625 non-null  float64
 7   NOx      242629 non-null  float64
 8   OXY      42072 non-null   float64
 9   O_3      234131 non-null  float64
 10  PM10     240896 non-null  float64
 11  PXY      42063 non-null   float64
 12  SO_2     242729 non-null  float64
 13  TCH      111991 non-null  float64
 14  TOL      69439 non-null   float64
 15  station  243984 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 29.8+ MB
```

In [82]:

```
df.columns
```

Out[82]:

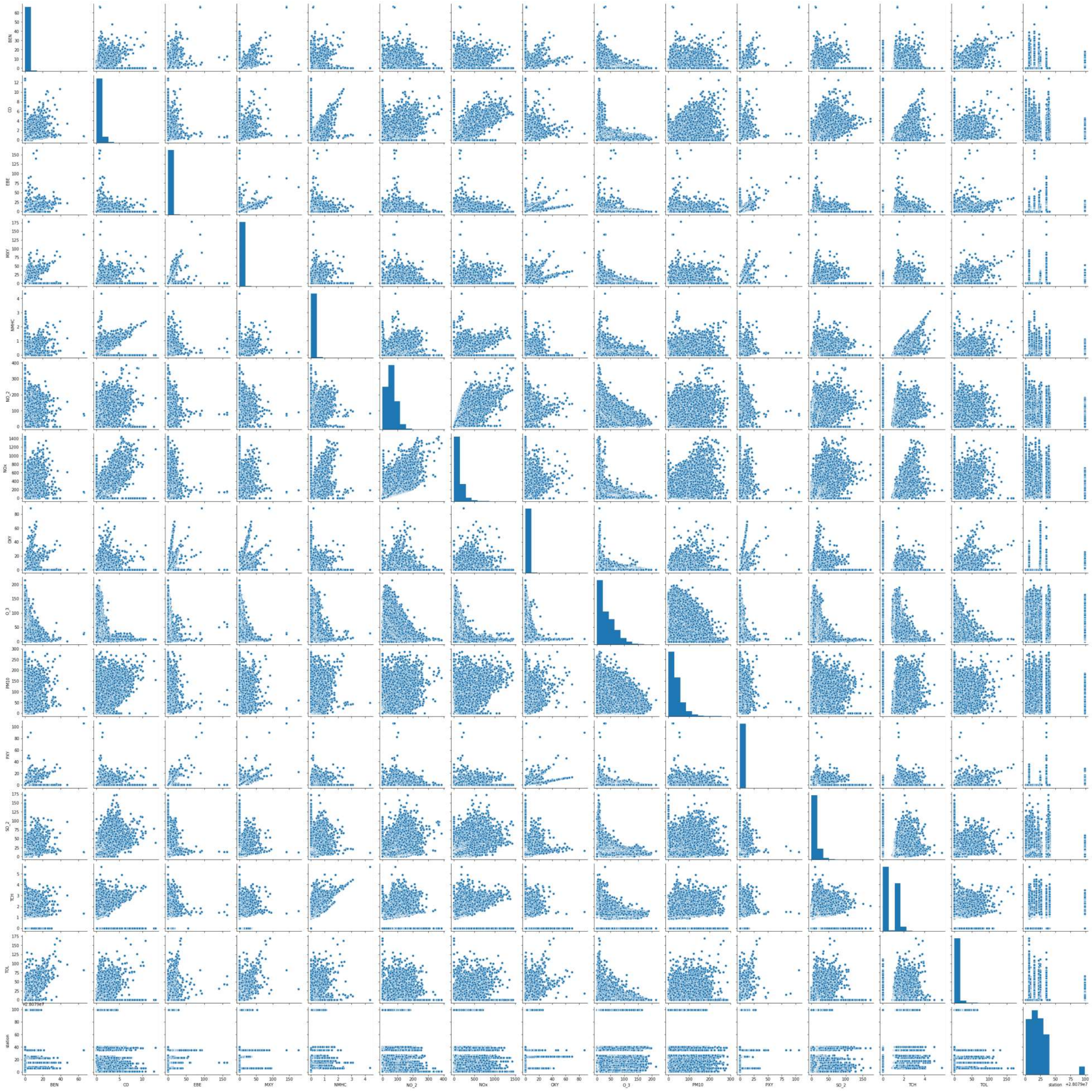
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [83]:

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

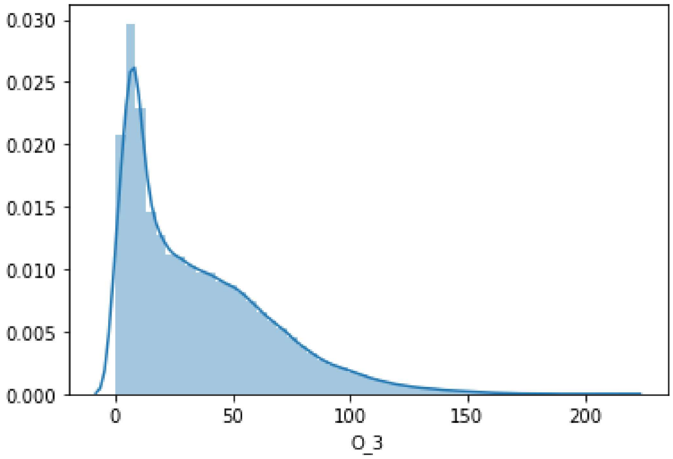
In [84]:
```
sns.pairplot(df2)
```

Out[84]:
```
<seaborn.axisgrid.PairGrid at 0x23359cc7730>
```



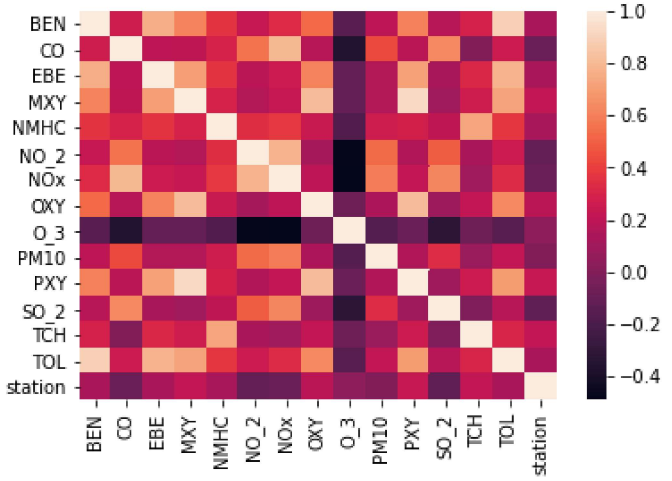
In [85]:
```
sns.distplot(df2['O_3'])
```

Out[85]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23387a9dc10>
```



In [86]:
```
sns.heatmap(df2.corr())
```

Out[86]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23387c763d0>
```



Linear Regression

In [87]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [88]:
```
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 [89]:
```
from sklearn.linear_model import LinearRegression

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

Out[89]:
```
LinearRegression()
```

```
In [90]: print(lr.intercept_)

56.85094428750709
```

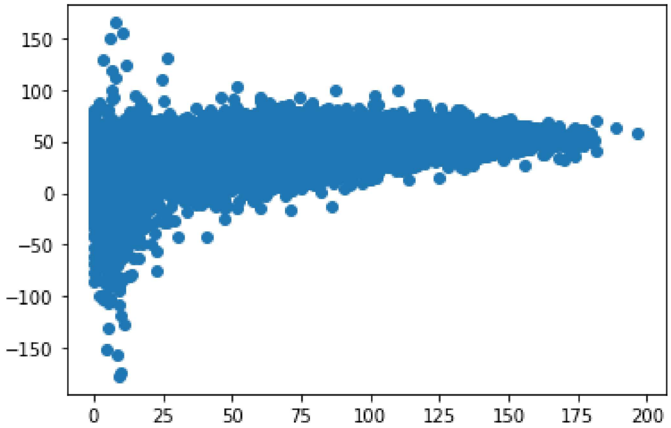
```
In [91]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
         coeff
```

Out[91]:

	Co-effecient
BEN	0.792465
CO	3.809723
EBE	0.194738
MXY	-0.505778
NMHC	12.186985
NO_2	-0.253641
NOx	-0.139582
OXY	0.152022
PM10	0.241296
PXY	1.193437
SO_2	-0.107634
TCH	-2.044950
TOL	-0.196108

```
In [92]: prediction=lr.predict(x_test)
         plt.scatter(y_test,prediction)

Out[92]: <matplotlib.collections.PathCollection at 0x2331b4d7100>
```



```
In [93]: print(lr.score(x_test,y_test))

0.3017668265853545
```

```
In [94]: lr.score(x_train,y_train)

Out[94]: 0.3022061230314904
```

Ridge Lasso

```
In [95]: from sklearn.linear_model import Ridge,Lasso
```

```
In [96]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)
         rr.score(x_test,y_test)

Out[96]: 0.301766194315902
```

```
In [97]: predict2=(rr.predict(x_test))
```

```
In [98]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[98]: Lasso(alpha=10)
```

```
In [99]: la.score(x_test,y_test)

Out[99]: 0.29628583548354337
```

Elastic Net regression

```
In [100]: from sklearn.linear_model import ElasticNet
          en=ElasticNet()
          en.fit(x_train,y_train)

Out[100]: ElasticNet()
```

```
In [101]: print(en.coef_)

          [ 0.          0.04553653  0.04562882  0.          0.         -0.26688012
           -0.11853961  0.          0.23935068  0.         -0.02741358 -0.
           -0.        ]
```

```
In [102]: print(en.intercept_)

          56.335079634826144
```

```
In [103]: print(en.score(x_test,y_test))

          0.2970261544701138
```

```
In [104]: print(en.score(x_train,y_train))

          0.2977571089260954
```

Logistic Regression

```
In [105]: from sklearn.linear_model import LogisticRegression
```

```
In [106]: feature_matrix=df2.iloc[:,0:5]
          target_vector=df2.iloc[:,-1]
```

```
In [107]: from sklearn.preprocessing import StandardScaler
```

In [108]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [109]:
```
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[109]:
```
LogisticRegression()
```

In [110]:
```
df2.shape
```

Out[110]:
```
(243984, 15)
```

In [111]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [112]:
```
print(predication)
```

```
[28079099]
```

In [113]:
```
logr.classes_
```

Out[113]:
```
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 [114]:
```
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 [115]:
```
print(logr.score(x_test,y_test))
```

```
0.054839062243838464
```

In [116]:
```
print(logr.score(x_train,y_train))
```

```
0.05379183549195494
```

Conclusion

Linear Regression is bestfit model

Linear Regression is bestfit model for dataset madrid_2001. The Score x_test,y_test is 0.3017668265853545 and x_train,y_train score is 0.3022061230314904

In [ ]:

In [ ]: