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

In [41]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_2002.csv")
df
```

Out[41]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.320000
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.660000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.670000
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.990000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.260000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.210000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999	NaN	15.640000
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.620000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN	5.52	24.219999
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.910000

217296 rows × 16 columns

In [42]:
```
df1 = df.fillna(0)
df1
```

Out[42]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2002-04-01 01:00:00	0.00	1.39	0.00	0.00	0.00	145.100006	352.100006	0.00	6.54	41.990002	0.00	21.320000
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.660000
2	2002-04-01 01:00:00	0.00	0.80	0.00	0.00	0.00	103.699997	134.000000	0.00	13.01	28.440001	0.00	13.670000
3	2002-04-01 01:00:00	0.00	1.61	0.00	0.00	0.00	97.599998	268.000000	0.00	5.12	42.180000	0.00	16.990000
4	2002-04-01 01:00:00	0.00	1.90	0.00	0.00	0.00	92.089996	237.199997	0.00	7.28	76.330002	0.00	15.260000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	0.00	0.00	0.00	81.080002	265.700012	0.00	7.21	36.750000	0.00	13.210000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	0.00	0.38	113.900002	373.100006	0.00	5.66	63.389999	0.00	15.640000
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.620000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	0.00	149.800003	202.199997	1.00	5.75	0.000000	5.52	24.219999
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.910000

217296 rows × 16 columns

In [43]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217296 non-null  object 
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

In [44]:

```
df.columns
```

Out[44]:

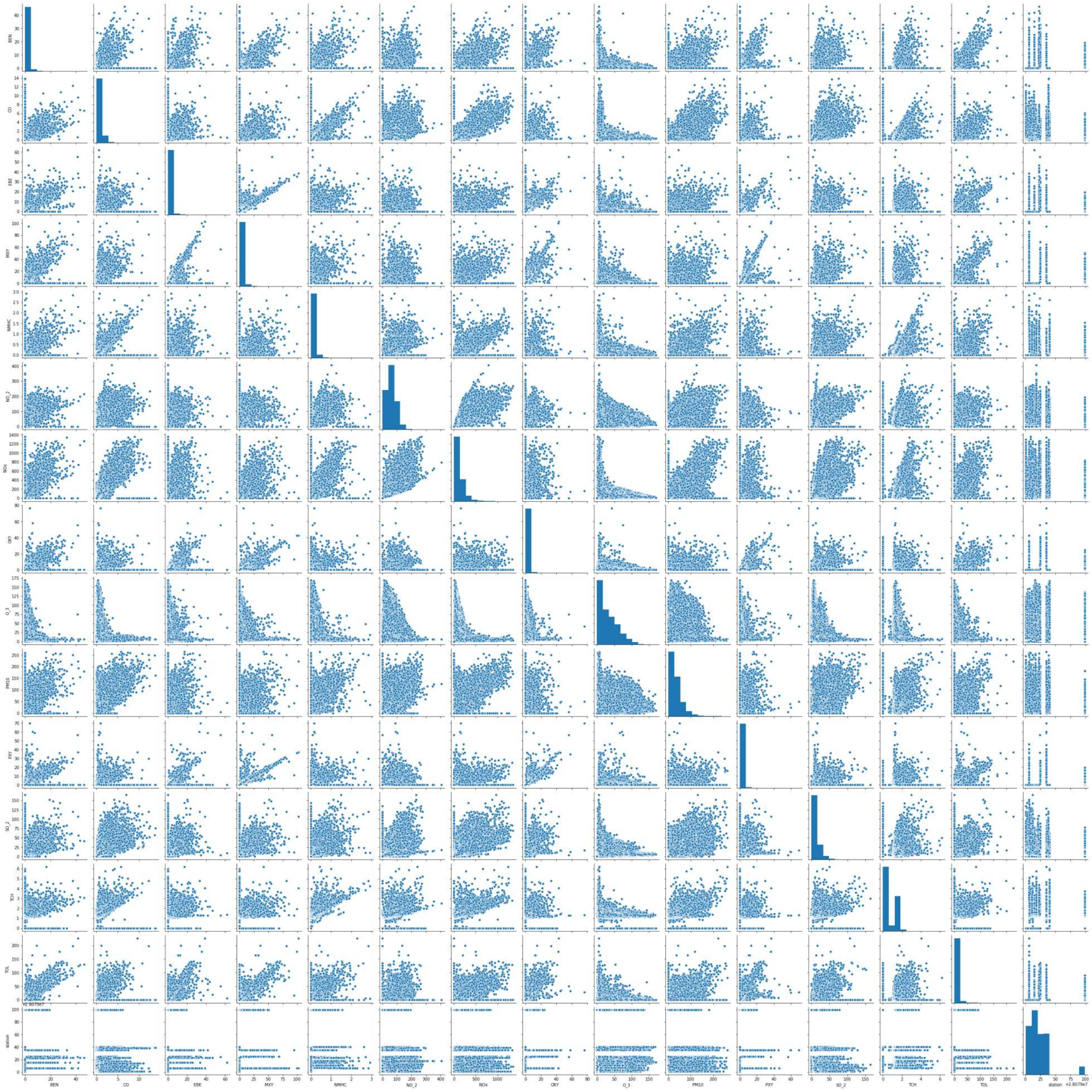
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [45]:

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

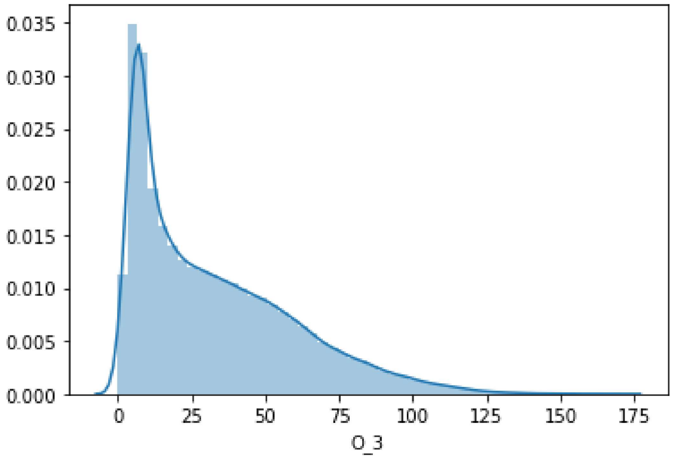
In [46]:
```
sns.pairplot(df2)
```

Out[46]:
```
<seaborn.axisgrid.PairGrid at 0x23316d258e0>
```



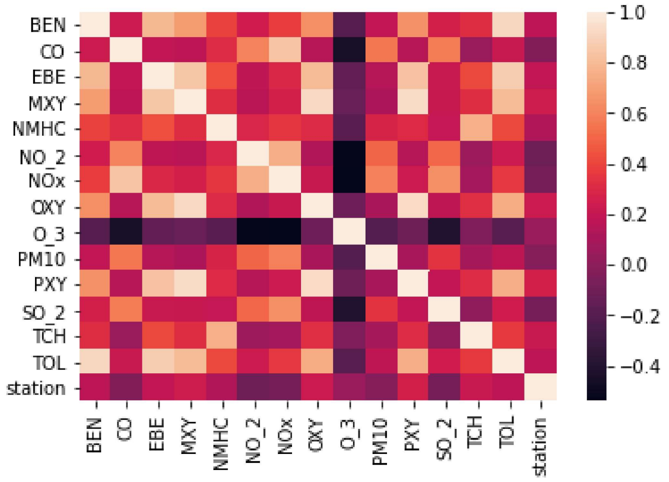
In [47]:
```
sns.distplot(df2['O_3'])
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2333c5d7880>
```



In [48]:
```
sns.heatmap(df2.corr())
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23356ce0280>
```



Linear Regression

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

In [50]:
```
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 [51]:
```
from sklearn.linear_model import LinearRegression

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

Out[51]:
```
LinearRegression()
```

In [52]:
```
print(lr.intercept_)
```

```
57.469790661936074
```

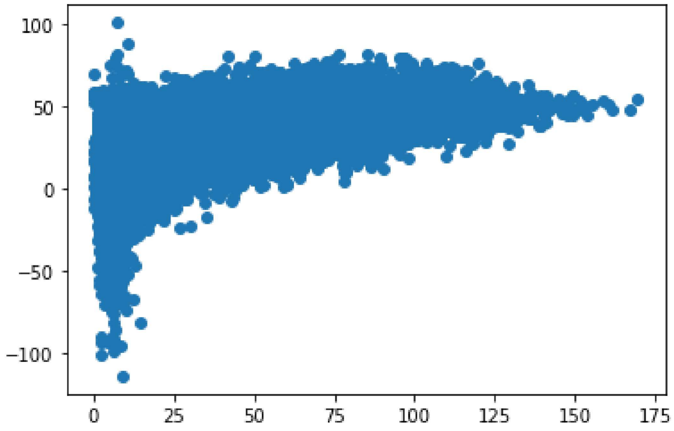
In [53]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[53]:

	Co-effecient
BEN	0.522743
CO	-2.175383
EBE	0.113789
MXY	0.070354
NMHC	14.353484
NO_2	-0.273541
NOx	-0.079958
OXY	-0.333165
PM10	0.211886
PXY	0.910906
SO_2	-0.216371
TCH	-2.408368
TOL	-0.243377

In [54]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[54]:
```
<matplotlib.collections.PathCollection at 0x2331998d100>
```



In [55]:
```
print(lr.score(x_test,y_test))
```

```
0.3635724046526313
```

In [56]:
```
lr.score(x_train,y_train)
```

Out[56]:
```
0.36284560662269716
```

Ridge Lasso

In [57]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [58]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[58]:
```
0.36357168178345 09
```

In [59]:
```
predict2=(rr.predict(x_test))
```

In [60]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[60]:
```
Lasso(alpha=10)
```

In [61]:
```
la.score(x_test,y_test)
```

Out[61]:
```
0.357904457532028 04
```

Elastic Net regression

In [62]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[62]:
```
ElasticNet()
```

In [63]:
```
print(en.coef_)
```

```
[ 0.         -0.          0.          0.05185668  0.         -0.26186518
 -0.08971544  0.          0.20528086  0.         -0.21073604 -0.
 -0.00487909]
```

In [64]:
```
print(en.intercept_)
```

```
56.02023772231269
```

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

```
0.36070101767965235
```

In [66]:
```
print(en.score(x_train,y_train))
```

```
0.3598254732927437
```

Logistic Regression

In [67]:
```
from sklearn.linear_model import LogisticRegression
```

In [68]:
```
feature_matrix=df2.iloc[:,0:5]
target_vector=df2.iloc[:,-1]
```

In [69]:
```
from sklearn.preprocessing import StandardScaler
```

In [70]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [71]:
```
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[71]:
```
LogisticRegression()
```

In [72]:
```
df2.shape
```

Out[72]:
```
(217296, 15)
```

In [73]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [74]:
```
print(predication)
```

```
[28079099]
```

In [75]:
```
logr.classes_
```

Out[75]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [76]:
```
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 [77]:
```
print(logr.score(x_test,y_test))
```

```
0.08981576646366718
```

In [78]:
```
print(logr.score(x_train,y_train))
```

```
0.0886744199806715
```

Conclusion

Linear Regression is bestfit model

Linear Regression is bestfit model for dataset madrid_2001. The Score x_test,y_test is 0.3635724046526313 and x_train,y_train score is 0.36284560662269716

In [ ]:

In [ ]: