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

In [2]:
```
df = pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002
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	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002
...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
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
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
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
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

217296 rows × 16 columns

In [3]:
```
df1 = df.fillna(0)
df1
```

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
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
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	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
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
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
...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

217296 rows × 16 columns

In [4]:
```
df1.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      217296 non-null  float64
 2   CO       217296 non-null  float64
 3   EBE      217296 non-null  float64
 4   MXY      217296 non-null  float64
 5   NMHC     217296 non-null  float64
 6   NO_2     217296 non-null  float64
 7   NOx      217296 non-null  float64
 8   OXY      217296 non-null  float64
 9   O_3      217296 non-null  float64
 10  PM10     217296 non-null  float64
 11  PXY      217296 non-null  float64
 12  SO_2     217296 non-null  float64
 13  TCH      217296 non-null  float64
 14  TOL      217296 non-null  float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

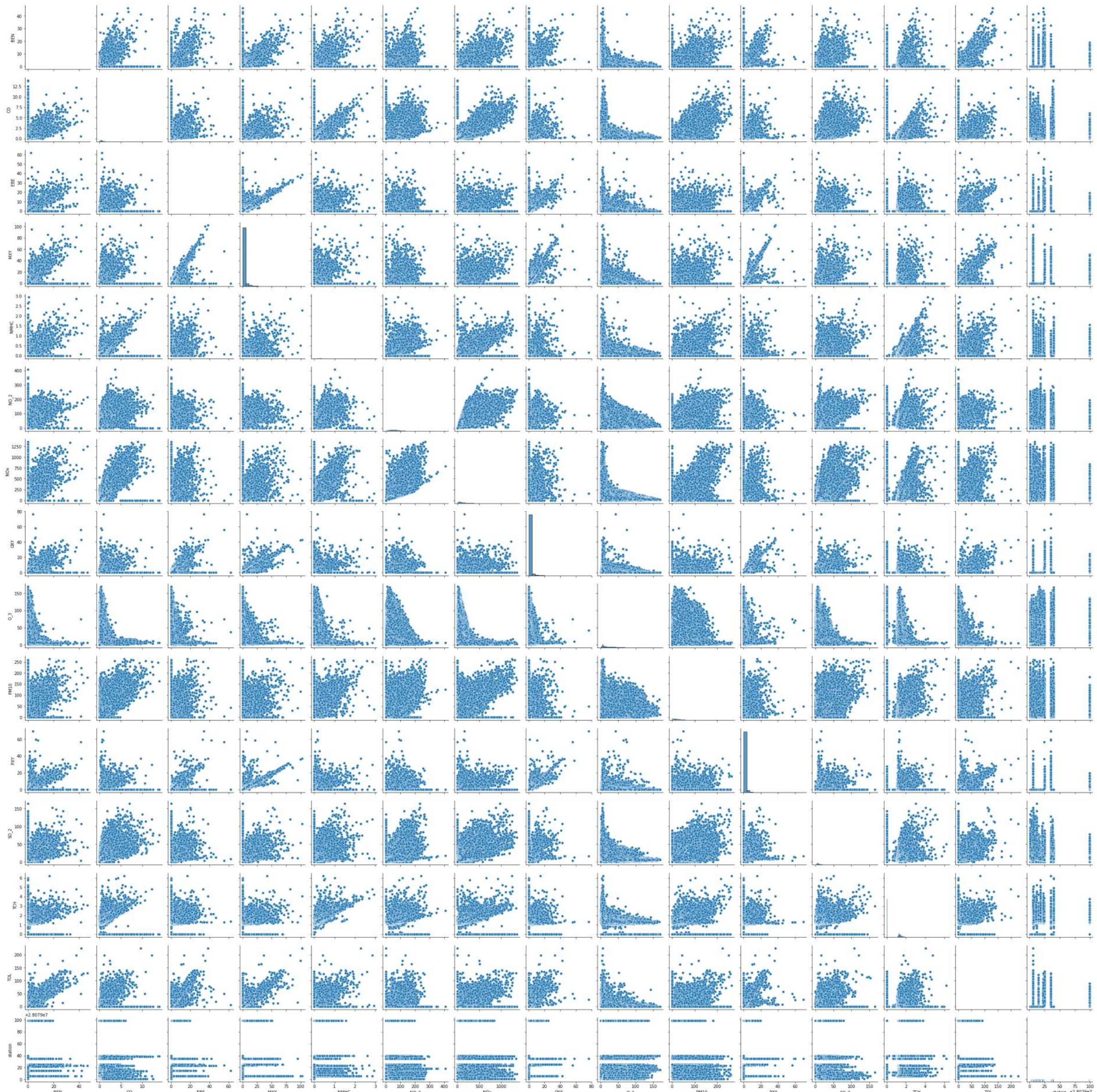
In [5]:
```
df1.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
df2 = df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [7]:
```
sns.pairplot(df2)
```

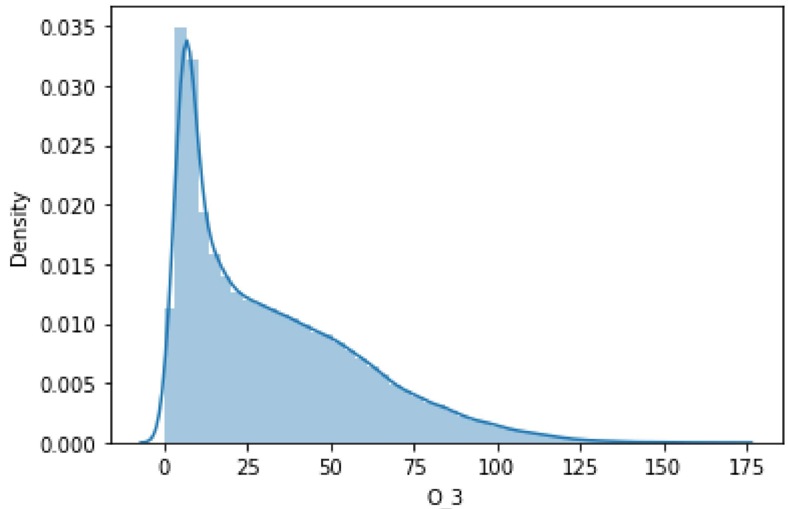
Out[7]: <seaborn.axisgrid.PairGrid at 0x1a2e6aecbb0>



In [8]:
```
sns.distplot(df2['O_3'])
```

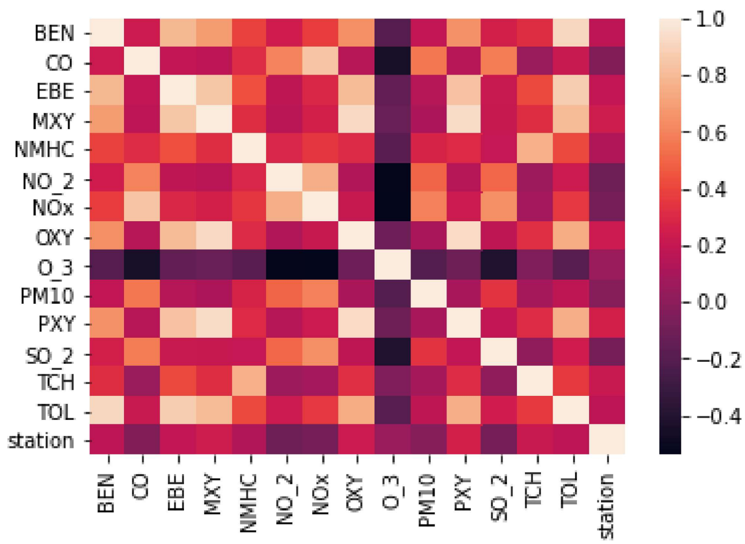
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='O_3', ylabel='Density'>



In [9]:
```
sns.heatmap(df2.corr())
```

Out[9]: <AxesSubplot:>



Linear Regression

In [40]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY','O_3',
       'PM10', 'PXY', 'SO_2', 'TCH']]
y = df2['TOL']
```

In [41]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

In [42]:
```
from sklearn.linear_model import LinearRegression

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

Out[42]:
```
LinearRegression()
```

In [43]:
```
print(lr.intercept_)
```

```
0.15445106913505802
```

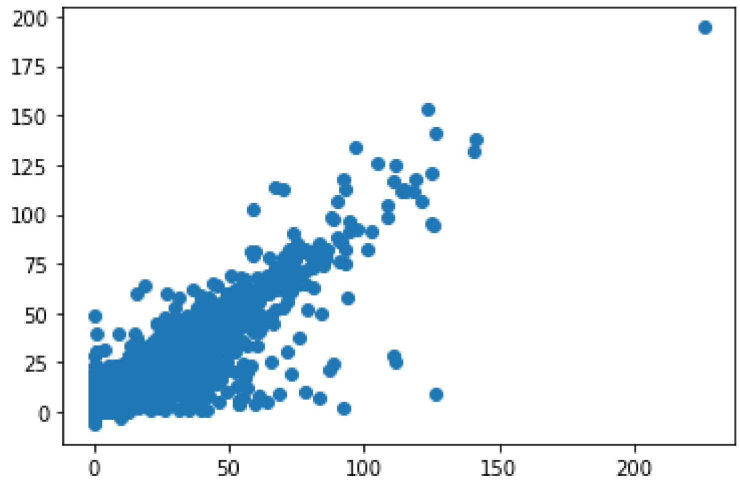
In [44]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[44]:

	Co-efficient
BEN	2.658109
CO	-0.429622
EBE	1.189738
MXY	0.601485
NMHC	0.601419
NO_2	0.000689
NOx	0.002673
OXY	-0.142393
O_3	-0.002631
PM10	0.004690
PXY	-0.396247
SO_2	-0.010880
TCH	0.084608

In [45]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[45]: <matplotlib.collections.PathCollection at 0x1a293b3c520>



In [46]:
```
print(lr.score(x_test,y_test))
```

0.9139589808706485

In [47]:
```
lr.score(x_train,y_train)
```

Out[47]: 0.9132416711077419

Ridge and Lasso

In [48]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [49]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_train,y_train)
```

Out[49]: 0.9132416667241228

In [50]:
```
rr.score(x_test,y_test)
```

Out[50]: 0.9139588309183644

Lasso

In [51]:
```
ls = Lasso(alpha=10)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
```

Out[51]: 0.5649864405066657

In [52]:
```
ls.score(x_test,y_test)
```

Out[52]:
```
0.5664973165235041
```

ElacticNET regression

In [53]:
```
from sklearn.linear_model import ElasticNet
es = ElasticNet()
es.fit(x_train,y_train)
```

Out[53]:
```
ElasticNet()
```

In [54]:
```
print(es.coef_)
```

```
[ 1.97785934e+00 -0.00000000e+00  9.06914606e-01  6.45558025e-01
  0.00000000e+00 -0.00000000e+00  4.66685232e-03  0.00000000e+00
 -7.89027482e-04  1.19724155e-03  0.00000000e+00 -8.81212202e-03
  0.00000000e+00]
```

In [55]:
```
print(es.intercept_)
```

```
0.0891927372950092
```

In [56]:
```
print(es.score(x_test,y_test))
```

```
0.8954605772559113
```

In [57]:
```
print(es.score(x_train,y_train))
```

```
0.8949096910833548
```

LogisticRegression

In [58]:
```
from sklearn.linear_model import LogisticRegression
```

In [59]:
```
feature_matrix = df2.iloc[:,0:15]
target_vector = df2.iloc[:,-1]
```

In [60]:
```
feature_matrix.shape
```

Out[60]:
```
(217296, 15)
```

In [61]:
```
from sklearn.preprocessing import StandardScaler
```

In [62]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [63]:
```
logs = LogisticRegression()
logs.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: 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[63]:
```
LogisticRegression()
```

In [64]:
```
observation = [[1.4,1.5,1.6,2.7,2.3,3.3,2.3,4.1,2.3,4.2,1.2,2.1,4.3,6,2.2]]
prediction = logs.predict(observation)
```

In [35]:
```
print(prediction)
```

```
[28079099]
```

In [36]:
```
logs.classes_
```

Out[36]:
```
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 [37]:
```
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 [38]:
```
print(logs.score(x_test,y_test))
```

```
0.03959256929850128
```

In [39]:
```
print(logs.score(x_train,y_train))
```

```
0.04062271953296035
```

Conclusion

Ridge regression is bestfit model

Ridge regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.9139589808706485 and x_test and y_test score is 0.9132416711077419

In [ ]:

```

```