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	PM
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2600
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5800
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1900
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5300
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7600
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.8300
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9200
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0300
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.3600

215688 rows × 17 columns

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

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2009-10-01 01:00:00	0.00	0.27	0.00	0.00	0.00	39.889999	48.150002	0.00	50.680000	18.2600
1	2009-10-01 01:00:00	0.00	0.22	0.00	0.00	0.00	21.230000	24.260000	0.00	55.880001	10.5800
2	2009-10-01 01:00:00	0.00	0.18	0.00	0.00	0.00	31.230000	34.880001	0.00	49.060001	25.1900
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5300
4	2009-10-01 01:00:00	0.00	0.41	0.00	0.00	0.12	61.349998	76.260002	0.00	38.090000	23.7600
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.8300
215684	2009-06-01 00:00:00	0.00	0.31	0.00	0.00	0.00	76.110001	101.099998	0.00	41.220001	9.9200
215685	2009-06-01 00:00:00	0.13	0.00	0.86	0.00	0.23	81.050003	99.849998	0.00	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	0.00	2.96	0.00	0.10	72.419998	82.959999	0.00	0.000000	13.0300
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.3600

215688 rows × 17 columns

In [4]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215688 entries, 0 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     215688 non-null  object 
 1   BEN      215688 non-null  float64
 2   CO       215688 non-null  float64
 3   EBE      215688 non-null  float64
 4   MXY      215688 non-null  float64
 5   NMHC     215688 non-null  float64
 6   NO_2     215688 non-null  float64
 7   NOx      215688 non-null  float64
 8   OXY      215688 non-null  float64
 9   O_3      215688 non-null  float64
 10  PM10     215688 non-null  float64
 11  PM25     215688 non-null  float64
 12  PXY      215688 non-null  float64
 13  SO_2     215688 non-null  float64
 14  TCH      215688 non-null  float64
 15  TOL      215688 non-null  float64
 16  station  215688 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 28.0+ MB
```

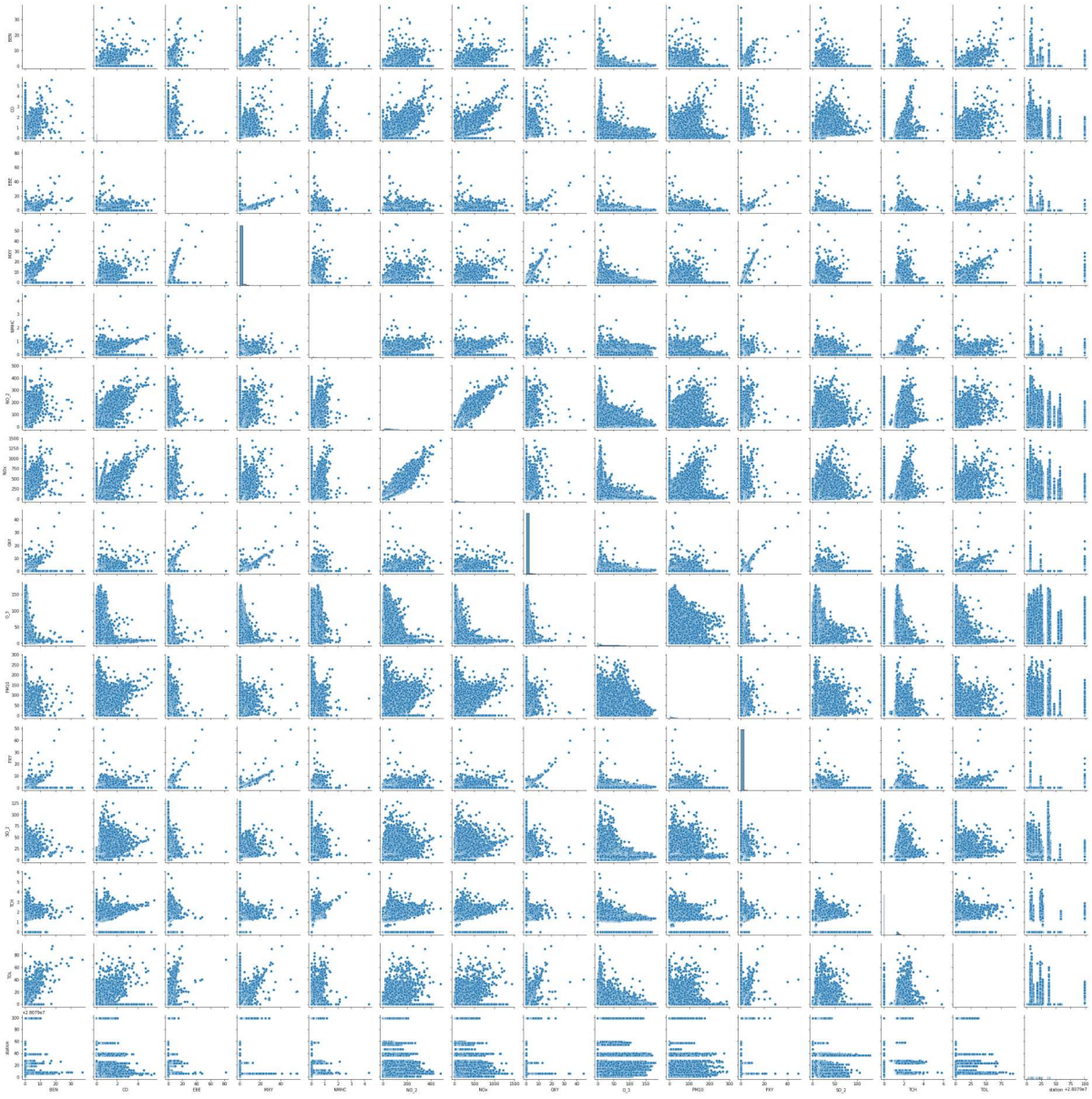
In [5]:
```
df1.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', '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)
```

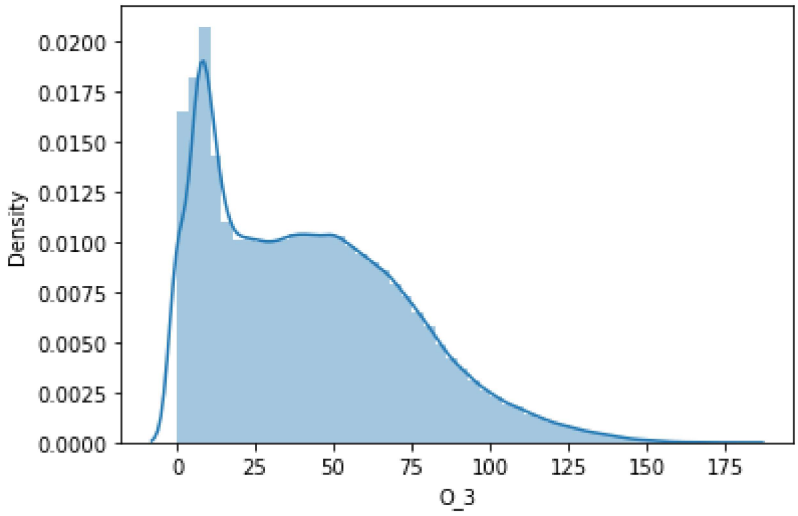
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x275b7798100>
```



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

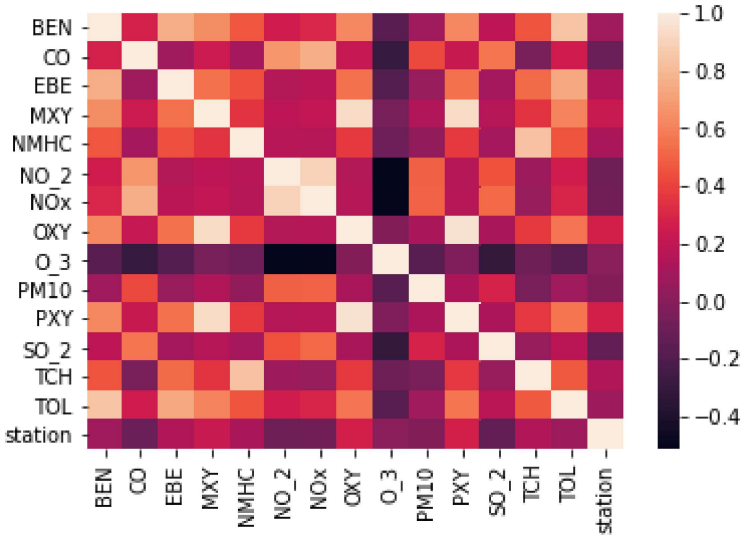
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[9]: <AxesSubplot:>



Linear Regression

In [10]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY','O_3',
       'PM10', 'SO_2','PXY', 'TCH']]
y = df2['TOL']
```

In [11]:
```
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 [12]:
```
from sklearn.linear_model import LinearRegression

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

Out[12]:
```
LinearRegression()
```

In [13]:
```
print(lr.intercept_)
```

```
-0.3723706182969704
```

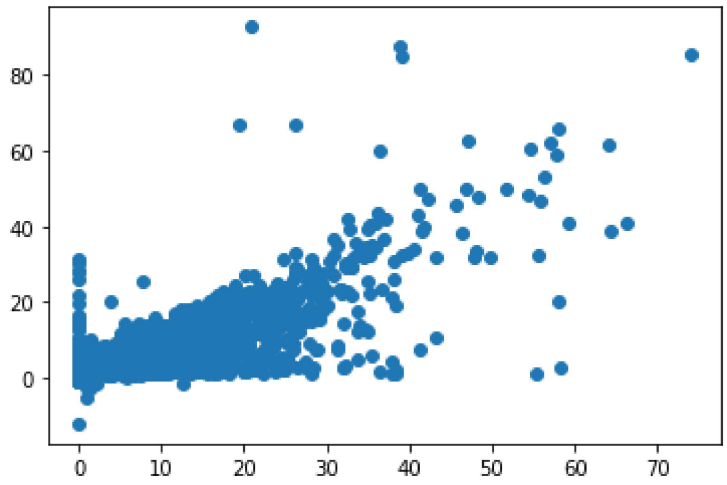
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
BEN	2.917280
CO	0.857621
EBE	0.787063
MXY	1.048616
NMHC	-0.353015
NO_2	0.001645
NOx	0.000447
OXY	-0.877963
O_3	0.001787
PM10	-0.003253
SO_2	-0.008250
PXY	-0.833015
TCH	0.449490

In [15]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x275d6532250>
```



In [16]:
```
print(lr.score(x_test,y_test))
```

```
0.7545625644354046
```

In [17]:
```
lr.score(x_train,y_train)
```

Out[17]:
```
0.7574119082039459
```

Ridge and Lasso

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_train,y_train)
```

Out[19]:
```
0.7574118226459362
```

In [20]:
```
rr.score(x_test,y_test)
```

Out[20]:
```
0.7545652002689648
```

Lasso Regression

In [21]:
```
ls = Lasso(alpha=10)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
```

Out[21]:
```
0.08784371784240086
```

In [22]:
```
ls.score(x_test,y_test)
```

Out[22]:
```
0.08962566786821169
```

ElacticNET regression

In [23]:
```
from sklearn.linear_model import ElasticNet
es = ElasticNet()
es.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(es.coef_)
```

```
[ 0.61970185  0.          0.72367586  0.45856478  0.         -0.
  0.0083787   0.          0.         -0.00869396  0.          0.
  0.05251583]
```

In [25]:
```
print(es.intercept_)
```

```
0.004884391233474172
```

In [26]:
```
print(es.score(x_test,y_test))
```

```
0.5704697651280848
```

In [27]:
```
print(es.score(x_train,y_train))
```

```
0.5597444154674283
```

LogisticRegression

In [28]:
```
from sklearn.linear_model import LogisticRegression
```

In [29]:
```
feature_matrix = df2.iloc[:,0:15]
target_vector = df2.iloc[:,-1]
```

In [30]:
```
feature_matrix.shape
```

Out[30]:
```
(215688, 15)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [33]:
```
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[33]:
```
LogisticRegression()
```

In [34]:
```
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, 28079016, 28079017,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079054, 28079057, 28079058, 28079059,
       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))
```

```
0.04104656374117174
```

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

```
0.0404289281432763
```

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.7574119082039459 and x_test and y_test score is 0.7545625644354046.

In [ ]: