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	PI
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	I
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	I
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	I
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.41C
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.67C
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	I
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	I
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	I
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.15C

209448 rows × 17 columns

In [3]:

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

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	0.00	0.29	0.00	0.0	0.00	25.090000	29.219999	0.0	68.930000	0.000
1	2010-03-01 01:00:00	0.00	0.27	0.00	0.0	0.00	24.879999	30.040001	0.0	0.000000	0.000
2	2010-03-01 01:00:00	0.00	0.28	0.00	0.0	0.00	17.410000	20.540001	0.0	72.120003	0.000
3	2010-03-01 01:00:00	0.38	0.24	1.74	0.0	0.05	15.610000	21.080000	0.0	72.970001	19.410
4	2010-03-01 01:00:00	0.79	0.00	1.32	0.0	0.00	21.430000	26.070000	0.0	0.000000	24.670
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	0.00	0.55	0.00	0.0	0.00	125.000000	219.899994	0.0	25.379999	0.000
209444	2010-08-01 00:00:00	0.00	0.27	0.00	0.0	0.00	45.709999	47.410000	0.0	0.000000	51.259
209445	2010-08-01 00:00:00	0.00	0.00	0.00	0.0	0.24	46.560001	49.040001	0.0	46.250000	0.000
209446	2010-08-01 00:00:00	0.00	0.00	0.00	0.0	0.00	46.770000	50.119999	0.0	77.709999	0.000
209447	2010-08-01 00:00:00	0.92	0.43	0.71	0.0	0.25	76.330002	88.190002	0.0	52.259998	47.150

209448 rows × 17 columns

In [4]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209448 entries, 0 to 209447
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209448 non-null  object 
 1   BEN      209448 non-null  float64
 2   CO       209448 non-null  float64
 3   EBE      209448 non-null  float64
 4   MXY      209448 non-null  float64
 5   NMHC     209448 non-null  float64
 6   NO_2     209448 non-null  float64
 7   NOx      209448 non-null  float64
 8   OXY      209448 non-null  float64
 9   O_3      209448 non-null  float64
 10  PM10     209448 non-null  float64
 11  PM25     209448 non-null  float64
 12  PXY      209448 non-null  float64
 13  SO_2     209448 non-null  float64
 14  TCH      209448 non-null  float64
 15  TOL      209448 non-null  float64
 16  station  209448 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 27.2+ 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]:

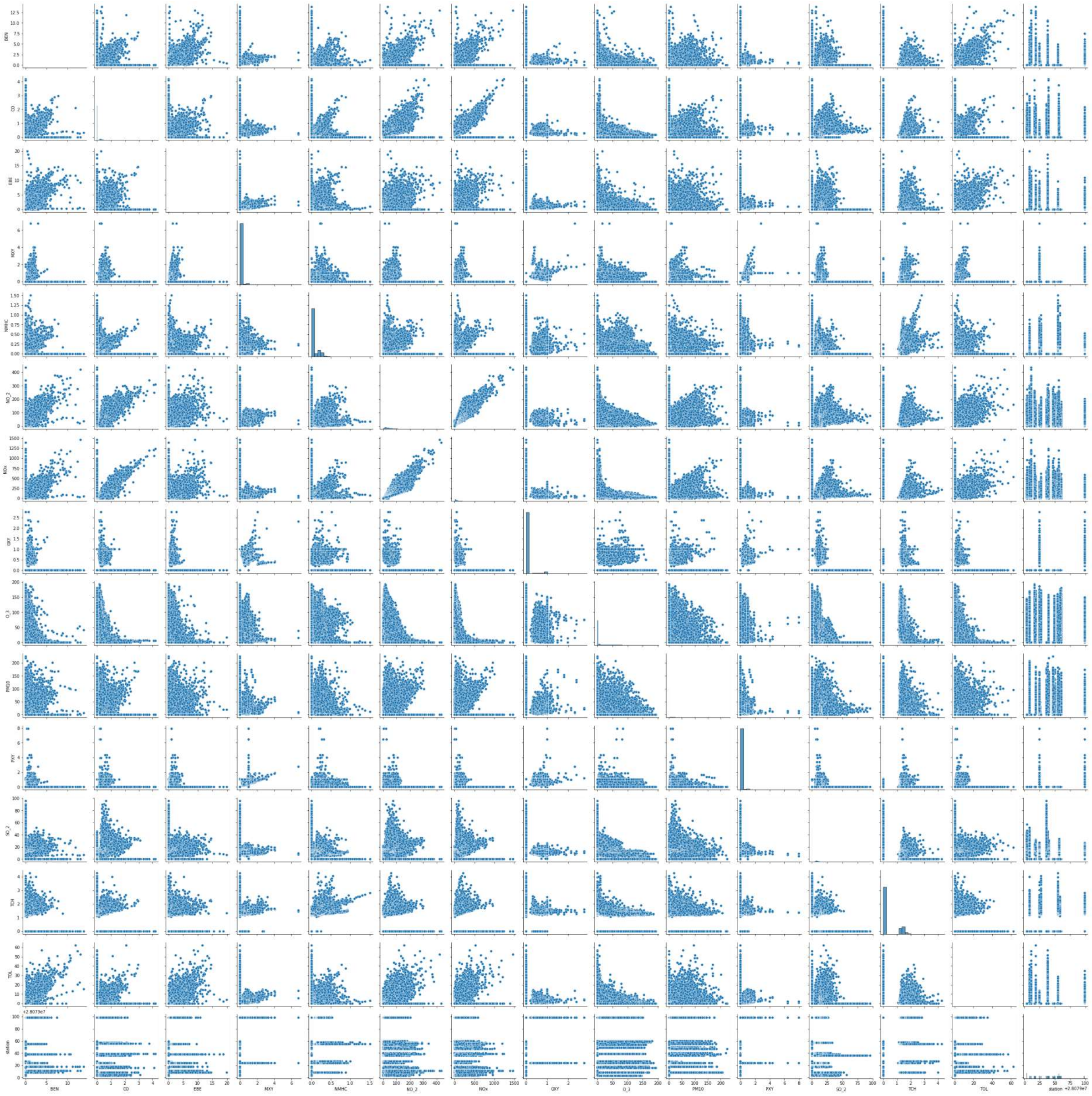
```
df2 = df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [7]:

```
sns.pairplot(df2)
```

Out[7]:

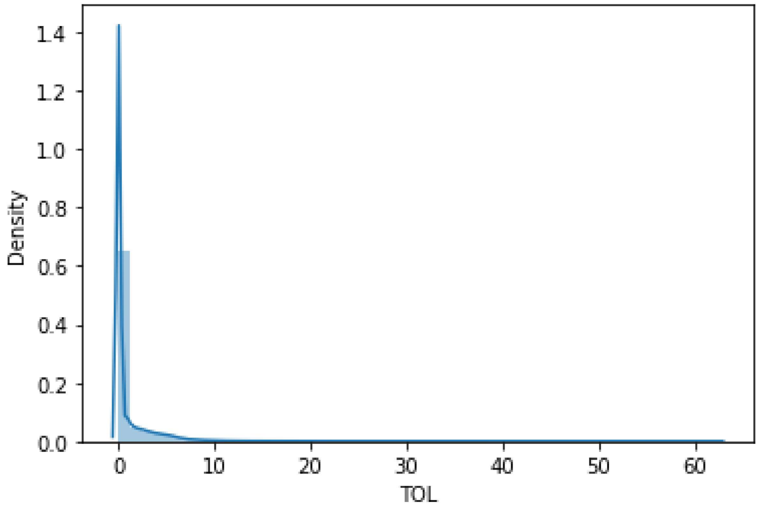
```
<seaborn.axisgrid.PairGrid at 0x270c83a9100>
```



```
In [40]: sns.distplot(df2['TOL'])
```

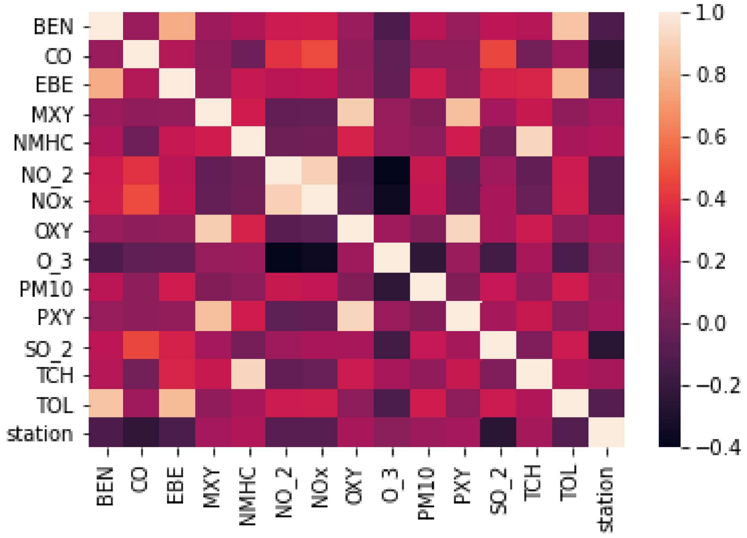
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

```
Out[40]: <AxesSubplot:xlabel='TOL', ylabel='Density'>
```



```
In [41]: sns.heatmap(df2.corr())
```

```
Out[41]: <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_)
```

```
-0.024924509858269328
```

In [14]:

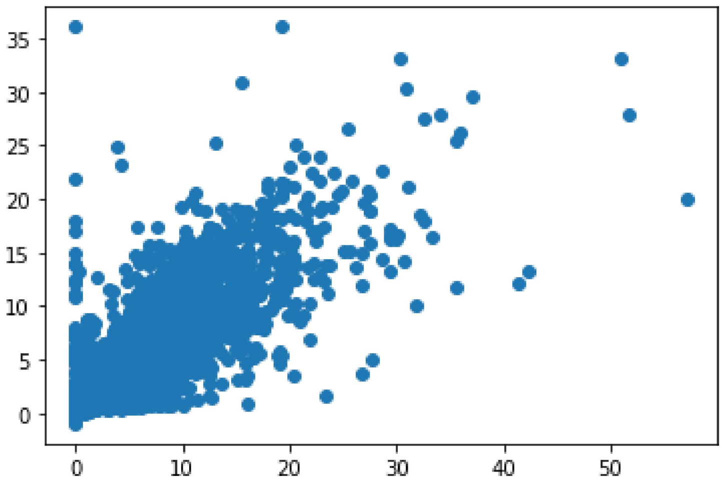
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
BEN	2.209906
CO	-0.516643
EBE	1.208764
MXY	0.409416
NMHC	1.209840
NO_2	0.000958
NOx	0.000925
OXY	-0.712777
O_3	-0.000041
PM10	0.006297
SO_2	0.016920
PXY	0.082471
TCH	-0.389755

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x2708b703bb0>
```



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

```
0.8014916447995456
```

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

Out[17]:
```
0.8055086859693753
```

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]:
```
0.8055075396355831
```

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

Out[20]:
```
0.8014917172850946
```

Lasso Regression

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

Out[21]:
```
0.09156196175090969
```

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

Out[22]:
```
0.08616910571985181
```

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_)
```

```
[ 3.33376457e-01 -0.00000000e+00  6.06608592e-01  0.00000000e+00
  0.00000000e+00  2.97636043e-03  4.78625298e-03  0.00000000e+00
  3.46717919e-04  1.97311364e-02  4.16498121e-02  0.00000000e+00
  0.00000000e+00]
```

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

```
-0.20453954044758638
```

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

```
0.5003170986998571
```

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

```
0.49927097941131615
```

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]:
```
(209448, 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([28079003, 28079004, 28079008, 28079011, 28079016, 28079017,
       28079018, 28079024, 28079026, 28079027, 28079036, 28079038,
       28079039, 28079040, 28079047, 28079048, 28079049, 28079050,
       28079054, 28079055, 28079056, 28079057, 28079058, 28079059,
       28079060, 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))
```

```
0.04261955916288693
```

In [39]:

```
print(logs.score(x_train,y_train))
```

```
0.04148336095707748
```

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.8014916447995456 and x_test and y_test score is 0.8014916447995456.

In [ ]: