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	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490

230568 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	2006-02-01 01:00:00	0.00	1.84	0.00	0.00	0.00	155.100006	490.100006	0.00	4.880000	97.570
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820
2	2006-02-01 01:00:00	0.00	1.25	0.00	0.00	0.00	66.800003	192.000000	0.00	4.430000	34.419
3	2006-02-01 01:00:00	0.00	1.68	0.00	0.00	0.00	103.000000	407.799988	0.00	4.830000	28.260
4	2006-02-01 01:00:00	0.00	1.31	0.00	0.00	0.00	105.400002	269.200012	0.00	6.990000	54.180
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	0.00	0.20	112.500000	218.000000	0.00	24.389999	93.120
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469
230565	2006-05-01 00:00:00	0.96	0.00	0.69	0.00	0.19	135.100006	179.199997	0.00	11.460000	64.680
230566	2006-05-01 00:00:00	0.50	0.00	0.67	0.00	0.10	82.599998	105.599998	0.00	0.000000	94.360
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490

230568 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      230568 non-null  float64
 2   CO       230568 non-null  float64
 3   EBE      230568 non-null  float64
 4   MXY      230568 non-null  float64
 5   NMHC     230568 non-null  float64
 6   NO_2     230568 non-null  float64
 7   NOx      230568 non-null  float64
 8   OXY      230568 non-null  float64
 9   O_3      230568 non-null  float64
 10  PM10     230568 non-null  float64
 11  PM25     230568 non-null  float64
 12  PXY      230568 non-null  float64
 13  SO_2     230568 non-null  float64
 14  TCH      230568 non-null  float64
 15  TOL      230568 non-null  float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

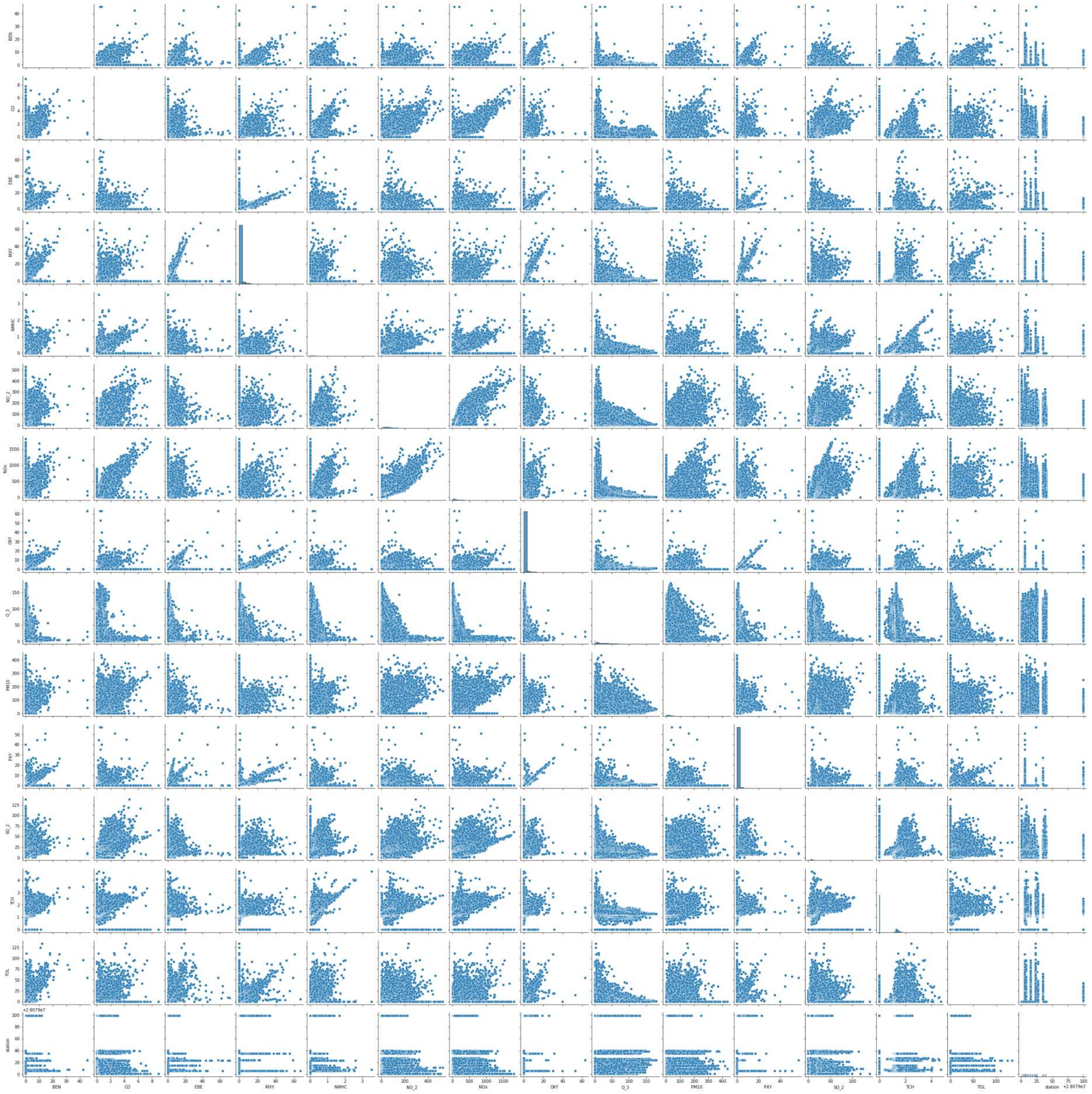
In [5]:
```
df1.columns
```

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)
```

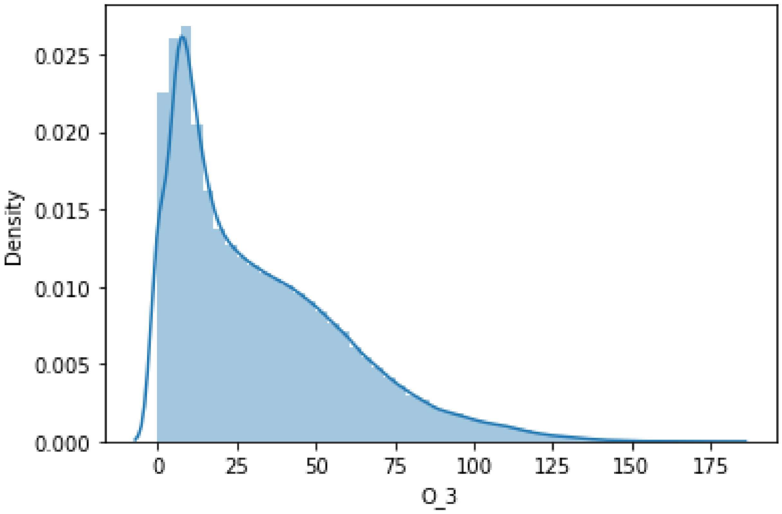
```
Out[7]: <seaborn.axisgrid.PairGrid at 0x22b3b3c83a0>
```



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

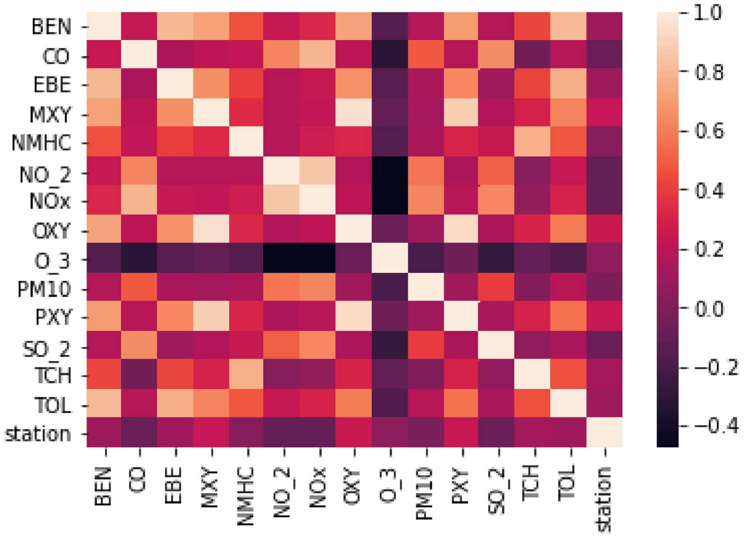
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

```
0.03472314719500336
```

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

Out[14]:

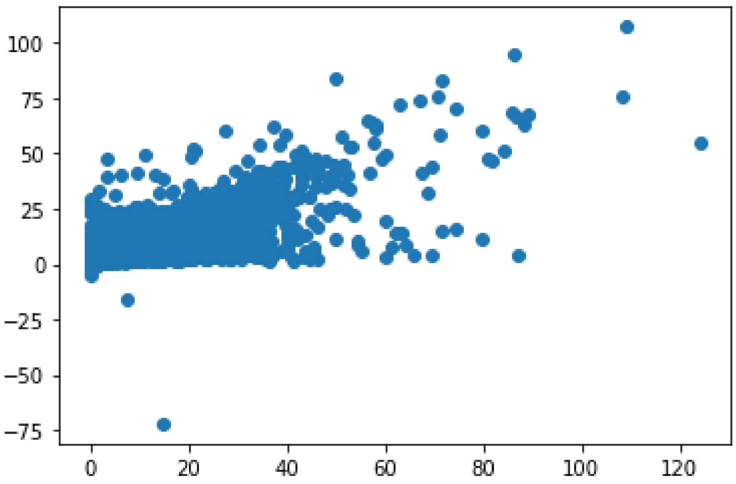
	Co-efficient
BEN	2.682264
CO	-0.846981
EBE	1.227905
MXY	0.748094
NMHC	3.348286
NO_2	0.000413
NOx	0.003728
OXY	-1.399115
O_3	-0.000885
PM10	0.005943
SO_2	-0.016347
PXY	-0.146965
TCH	0.231525

In [15]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:

```
<matplotlib.collections.PathCollection at 0x22b54174400>
```



In [16]:

```
print(lr.score(x_test,y_test))
```

```
0.72890938141895
```

In [17]:

```
lr.score(x_train,y_train)
```

Out[17]:

```
0.7124827238944019
```

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]:

```
0.7124823727700607
```

In [20]:

```
rr.score(x_test,y_test)
```

Out[20]:

```
0.7289161191203073
```

Lasso Regression

In [21]:

```
ls = Lasso(alpha=10)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
```

Out[21]:

```
0.0919837781879258
```

```
In [ ]:
```

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

```
Out[22]: 0.09511663618765664
```

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_)
```

```
[ 0.78660316 -0.          1.14448493  0.47985545  0.         -0.00307366
  0.00681863  0.         -0.00334459  0.00412656 -0.02488555  0.
  0.16037875]
```

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

```
0.29099540070667507
```

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

```
0.6350423415829536
```

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

```
0.6272076132852207
```

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]: (230568, 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)
```

```
[28079035]
```

In [36]:

```
logs.classes_
```

Out[36]:

```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 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))
```

```
0.037038643362102615
```

In [39]:

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

```
0.038402200784401194
```

Conclusion

Linear regression is bestfit model

Linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.7223613684892253 and x_test and y_test score is 0.7154606125539744.

In [ ]:

```
```