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** | 2005-11-01 01:00:00 | NaN | 0.77 | NaN | NaN | NaN | 57.130001 | 128.699997 | NaN | 14.720000 | 14.91 |
| **1** | 2005-11-01 01:00:00 | 1.52 | 0.65 | 1.49 | 4.57 | 0.25 | 86.559998 | 181.699997 | 1.27 | 11.680000 | 30.93 |
| **2** | 2005-11-01 01:00:00 | NaN | 0.40 | NaN | NaN | NaN | 46.119999 | 53.000000 | NaN | 30.469999 | 14.60 |
| **3** | 2005-11-01 01:00:00 | NaN | 0.42 | NaN | NaN | NaN | 37.220001 | 52.009998 | NaN | 21.379999 | 15.16 |
| **4** | 2005-11-01 01:00:00 | NaN | 0.57 | NaN | NaN | NaN | 32.160000 | 36.680000 | NaN | 33.410000 | 5.00 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **236995** | 2006-01-01 00:00:00 | 1.08 | 0.36 | 1.01 | NaN | 0.11 | 21.990000 | 23.610001 | NaN | 43.349998 | 5.00 |
| **236996** | 2006-01-01 00:00:00 | 0.39 | 0.54 | 1.00 | 1.00 | 0.11 | 2.200000 | 4.220000 | 1.00 | 69.639999 | 4.95 |
| **236997** | 2006-01-01 00:00:00 | 0.19 | NaN | 0.26 | NaN | 0.08 | 26.730000 | 30.809999 | NaN | 43.840000 | 4.31 |
| **236998** | 2006-01-01 00:00:00 | 0.14 | NaN | 1.00 | NaN | 0.06 | 13.770000 | 17.770000 | NaN | NaN | 5.00 |
| **236999** | 2006-01-01 00:00:00 | 0.50 | 0.40 | 0.73 | 1.84 | 0.13 | 20.940001 | 26.950001 | 1.49 | 48.259998 | 5.67 |

237000 rows × 17 columns

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

Out[3]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2005-11-01 01:00:00 | 0.00 | 0.77 | 0.00 | 0.00 | 0.00 | 57.130001 | 128.699997 | 0.00 | 14.720000 | 14.91 |
| **1** | 2005-11-01 01:00:00 | 1.52 | 0.65 | 1.49 | 4.57 | 0.25 | 86.559998 | 181.699997 | 1.27 | 11.680000 | 30.93 |
| **2** | 2005-11-01 01:00:00 | 0.00 | 0.40 | 0.00 | 0.00 | 0.00 | 46.119999 | 53.000000 | 0.00 | 30.469999 | 14.60 |
| **3** | 2005-11-01 01:00:00 | 0.00 | 0.42 | 0.00 | 0.00 | 0.00 | 37.220001 | 52.009998 | 0.00 | 21.379999 | 15.16 |
| **4** | 2005-11-01 01:00:00 | 0.00 | 0.57 | 0.00 | 0.00 | 0.00 | 32.160000 | 36.680000 | 0.00 | 33.410000 | 5.00 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **236995** | 2006-01-01 00:00:00 | 1.08 | 0.36 | 1.01 | 0.00 | 0.11 | 21.990000 | 23.610001 | 0.00 | 43.349998 | 5.00 |
| **236996** | 2006-01-01 00:00:00 | 0.39 | 0.54 | 1.00 | 1.00 | 0.11 | 2.200000 | 4.220000 | 1.00 | 69.639999 | 4.95 |
| **236997** | 2006-01-01 00:00:00 | 0.19 | 0.00 | 0.26 | 0.00 | 0.08 | 26.730000 | 30.809999 | 0.00 | 43.840000 | 4.31 |
| **236998** | 2006-01-01 00:00:00 | 0.14 | 0.00 | 1.00 | 0.00 | 0.06 | 13.770000 | 17.770000 | 0.00 | 0.000000 | 5.00 |
| **236999** | 2006-01-01 00:00:00 | 0.50 | 0.40 | 0.73 | 1.84 | 0.13 | 20.940001 | 26.950001 | 1.49 | 48.259998 | 5.67 |

237000 rows × 17 columns

In [4]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      237000 non-null  float64
 2   CO       237000 non-null  float64
 3   EBE      237000 non-null  float64
 4   MXY      237000 non-null  float64
 5   NMHC     237000 non-null  float64
 6   NO_2     237000 non-null  float64
 7   NOx      237000 non-null  float64
 8   OXY      237000 non-null  float64
 9   O_3      237000 non-null  float64
 10  PM10     237000 non-null  float64
 11  PM25     237000 non-null  float64
 12  PXY      237000 non-null  float64
 13  SO_2     237000 non-null  float64
 14  TCH      237000 non-null  float64
 15  TOL      237000 non-null  float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

In [5]:

```
df1.columns
```

Out[5]:

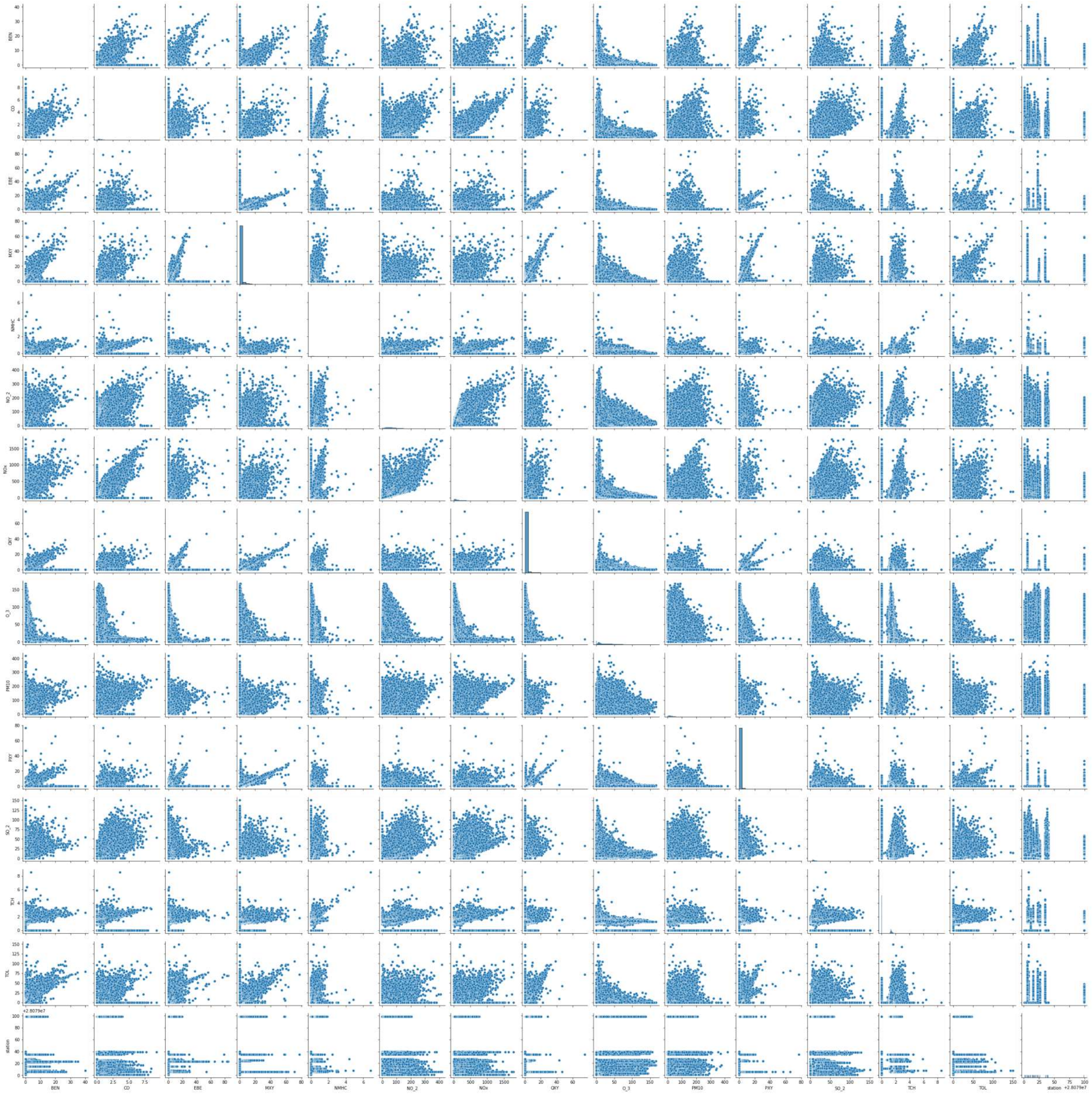
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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)
```

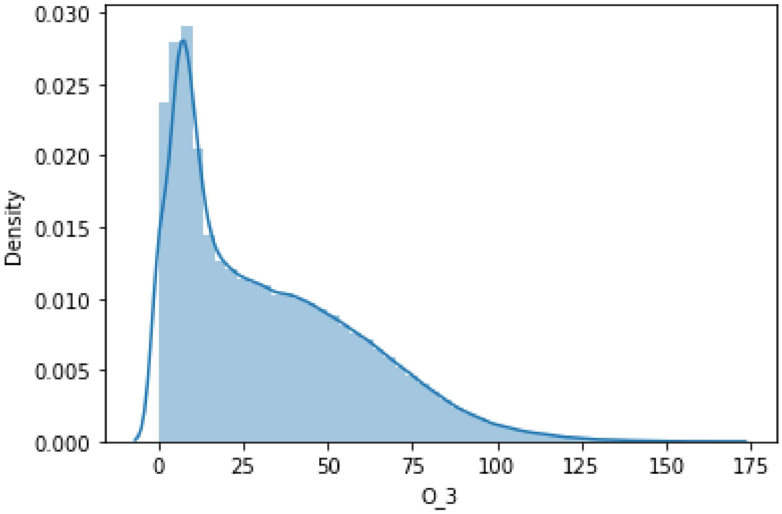
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x2e7e0b380d0>
```



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

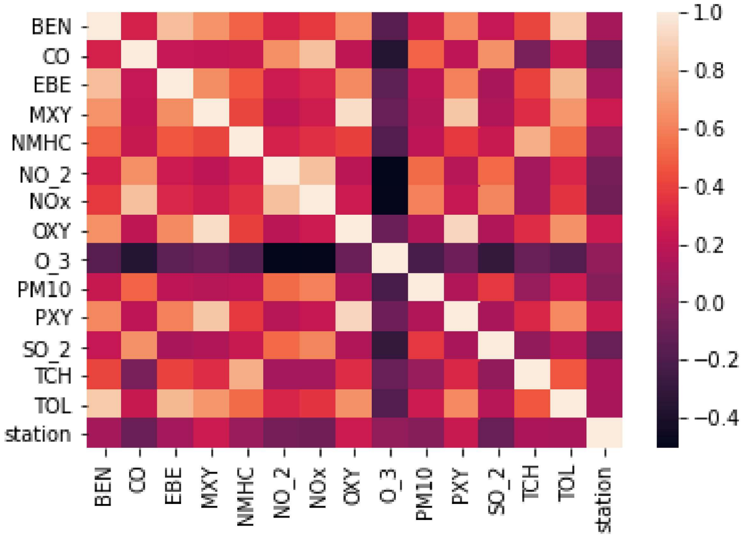
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function 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', 'PXY', 'SO_2', '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_)
```

```
-0.04022736209963362
```

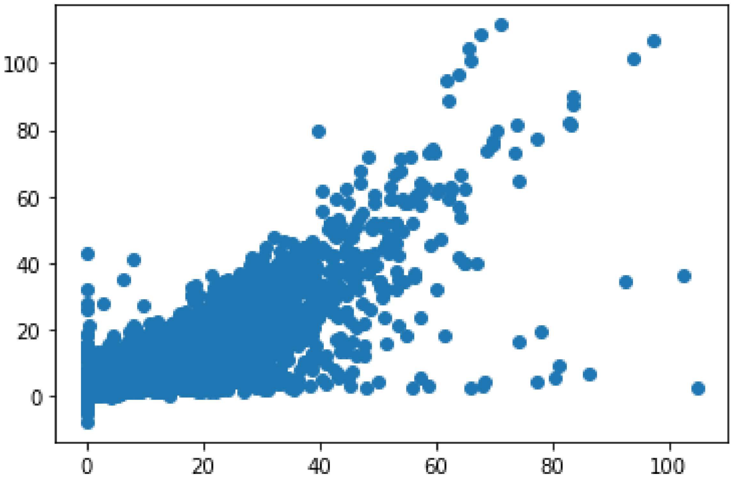
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

| | Co-efficient |
|---|---|
| **BEN** | 2.641049 |
| **CO** | -1.002311 |
| **EBE** | 0.650115 |
| **MXY** | 0.323694 |
| **NMHC** | 1.836819 |
| **NO_2** | 0.006230 |
| **NOx** | 0.002884 |
| **OXY** | -0.185963 |
| **O_3** | -0.003495 |
| **PM10** | 0.007370 |
| **PXY** | 0.191369 |
| **SO_2** | -0.012116 |
| **TCH** | 0.492996 |

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x2e781ce8370>
```



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

```
0.8048731432319678
```

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

Out[17]:
```
0.7968664876003704
```

# 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]:
```
0.7968664190572431
```

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

Out[20]:
```
0.8048703887718056
```

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

Out[21]:
```
0.13107172888751673
```

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

Out[22]:
```
0.13175784355212394
```

# 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_)

         [ 1.26268503 -0.          0.94807289  0.52849311  0.          0.00492603
           0.00350889  0.         -0.00365406  0.00623922  0.         -0.01485704
           0.11017266]
```

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

         0.031529986854826664
```

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

         0.7534455943486096
```

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

         0.7444758547422466
```

# 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]: (237000, 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,
       28079017, 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))
```

```
0.03713080168776371
```

In [39]:

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

```
0.03674502712477396
```

# Conclusion

Ridge regression is bestfit model

Ridge regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.8048731432319678 and x_test and y_test score is 0.7968664876003704.

In [ ]: