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   CH4   CO  EBE  NMHC     NO   NO_2    NOx   O_3  PM10  PM25  SO_2  T

            2018-
    0       03-01  NaN   NaN  0.3  NaN   NaN    1.0   29.0   31.0   NaN   NaN   NaN   2.0  N
         01:00:00

            2018-
    1       03-01  0.5  1.39  0.3  0.2  0.02    6.0   40.0   49.0  52.0   5.0   4.0   3.0  1
         01:00:00

            2018-
    2       03-01  0.4   NaN  NaN  0.2   NaN    4.0   41.0   47.0   NaN   NaN   NaN   NaN  N
         01:00:00

            2018-
    3       03-01  NaN   NaN  0.3  NaN   NaN    1.0   35.0   37.0  54.0   NaN   NaN   NaN  N
         01:00:00

            2018-
    4       03-01  NaN   NaN  NaN  NaN   NaN    1.0   27.0   29.0  49.0   NaN   NaN   3.0  N
         01:00:00

  ...         ...  ...   ...  ...  ...   ...    ...    ...    ...   ...   ...   ...   ...

            2018-
69091       02-01  NaN   NaN  0.5  NaN   NaN   66.0   91.0  192.0   1.0  35.0  22.0   NaN  N
         00:00:00

            2018-
69092       02-01  NaN   NaN  0.7  NaN   NaN   87.0  107.0  241.0   NaN  29.0   NaN  15.0  N
         00:00:00

            2018-
69093       02-01  NaN   NaN  NaN  NaN   NaN   28.0   48.0   91.0   2.0   NaN   NaN   NaN  N
         00:00:00

            2018-
69094       02-01  NaN   NaN  NaN  NaN   NaN  141.0  103.0  320.0   2.0   NaN   NaN   NaN  N
         00:00:00

            2018-
69095       02-01  NaN   NaN  NaN  NaN   NaN   69.0   96.0  202.0   3.0  26.0   NaN   NaN  N
         00:00:00

69096 rows × 16 columns
```

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

Out[3]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TC
0	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	29.0	31.0	0.0	0.0	0.0	2.0	0.(
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.4
2	2018-03-01 01:00:00	0.4	0.00	0.0	0.2	0.00	4.0	41.0	47.0	0.0	0.0	0.0	0.0	0.(
3	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	35.0	37.0	54.0	0.0	0.0	0.0	0.(
4	2018-03-01 01:00:00	0.0	0.00	0.0	0.0	0.00	1.0	27.0	29.0	49.0	0.0	0.0	3.0	0.(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
69091	2018-02-01 00:00:00	0.0	0.00	0.5	0.0	0.00	66.0	91.0	192.0	1.0	35.0	22.0	0.0	0.(
69092	2018-02-01 00:00:00	0.0	0.00	0.7	0.0	0.00	87.0	107.0	241.0	0.0	29.0	0.0	15.0	0.(
69093	2018-02-01 00:00:00	0.0	0.00	0.0	0.0	0.00	28.0	48.0	91.0	2.0	0.0	0.0	0.0	0.(
69094	2018-02-01 00:00:00	0.0	0.00	0.0	0.0	0.00	141.0	103.0	320.0	2.0	0.0	0.0	0.0	0.(
69095	2018-02-01 00:00:00	0.0	0.00	0.0	0.0	0.00	69.0	96.0	202.0	3.0	26.0	0.0	0.0	0.(

69096 rows × 16 columns

In [4]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69096 entries, 0 to 69095
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     69096 non-null  object 
 1   BEN      69096 non-null  float64
 2   CH4      69096 non-null  float64
 3   CO       69096 non-null  float64
 4   EBE      69096 non-null  float64
 5   NMHC     69096 non-null  float64
 6   NO       69096 non-null  float64
 7   NO_2     69096 non-null  float64
 8   NOx      69096 non-null  float64
 9   O_3      69096 non-null  float64
 10  PM10     69096 non-null  float64
 11  PM25     69096 non-null  float64
 12  SO_2     69096 non-null  float64
 13  TCH      69096 non-null  float64
 14  TOL      69096 non-null  float64
 15  station  69096 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 8.4+ MB
```

In [5]:

```
df1.columns
```

Out[5]:

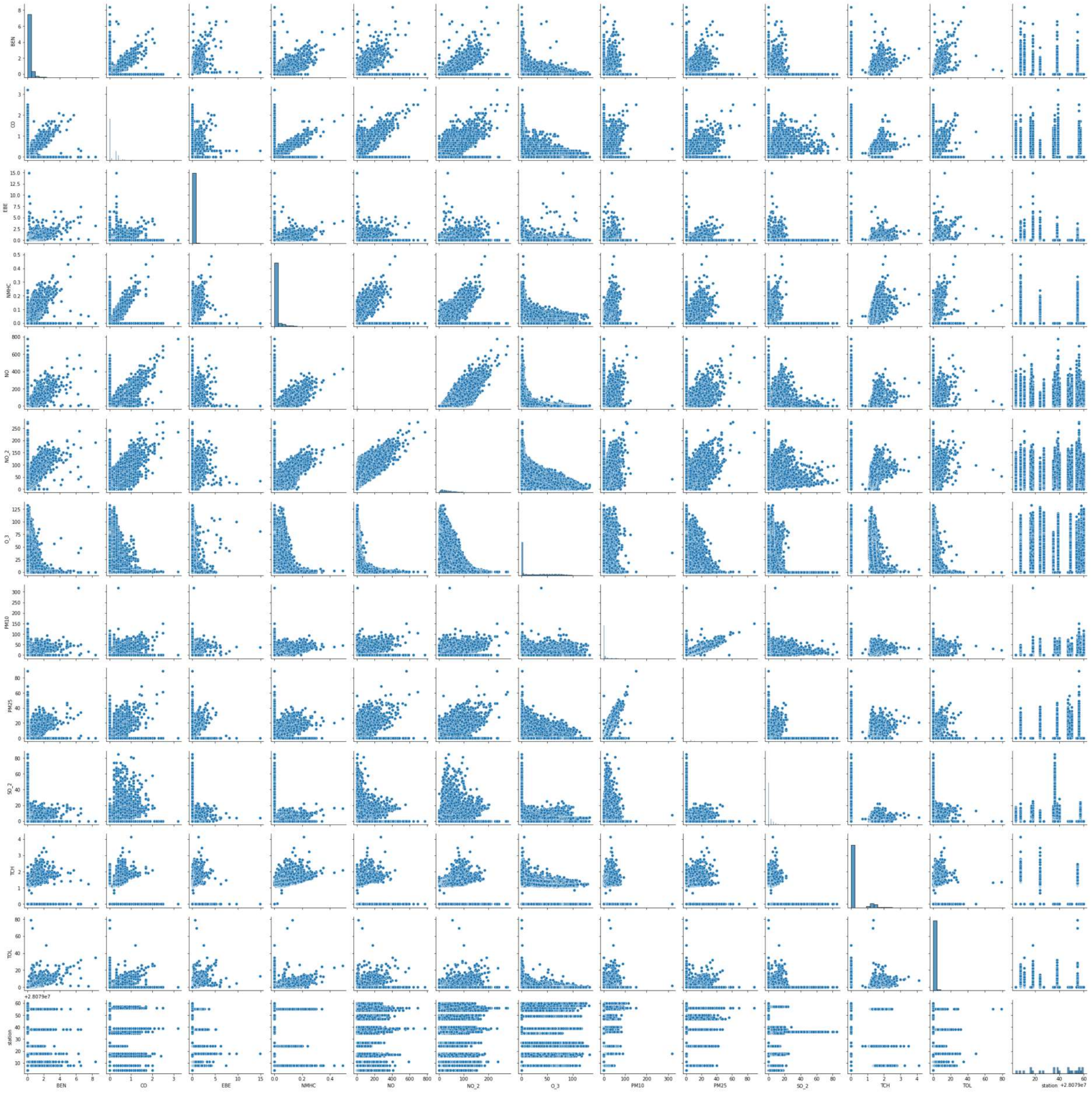
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:

```
df2 = df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

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

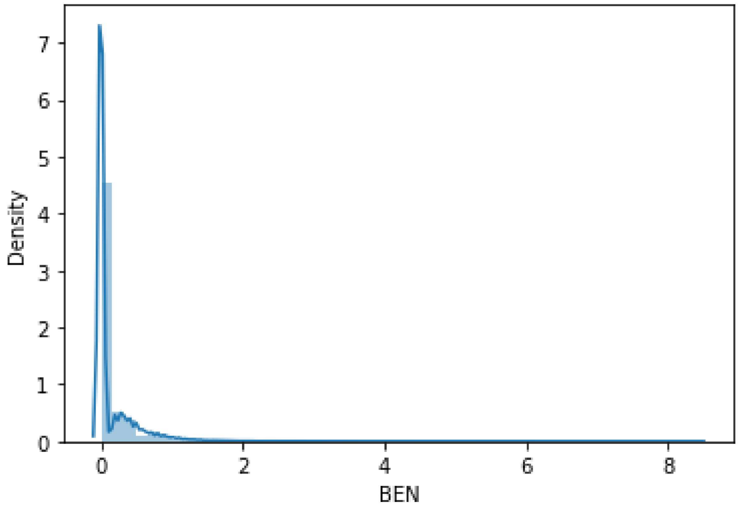
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x24d5ba1f0d0>
```



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

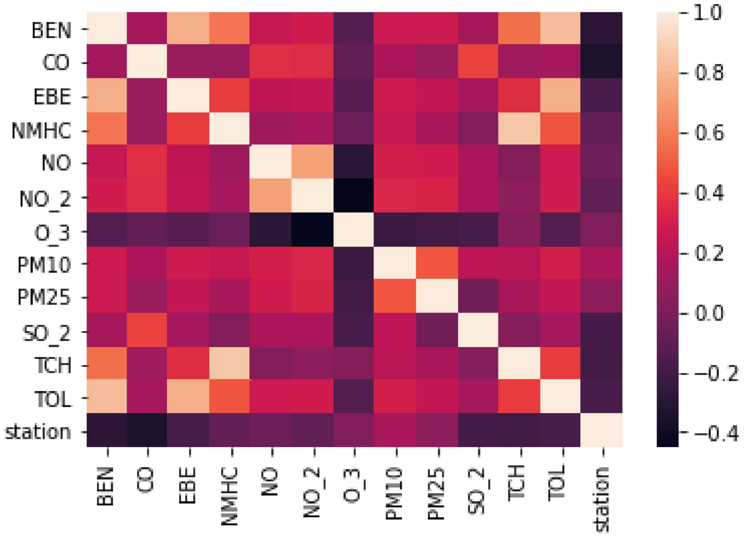
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[9]:
```
<AxesSubplot:>
```



Linear Regression

In [10]:
```
x = df2[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', '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_)
```

```
-0.14832902244219354
```

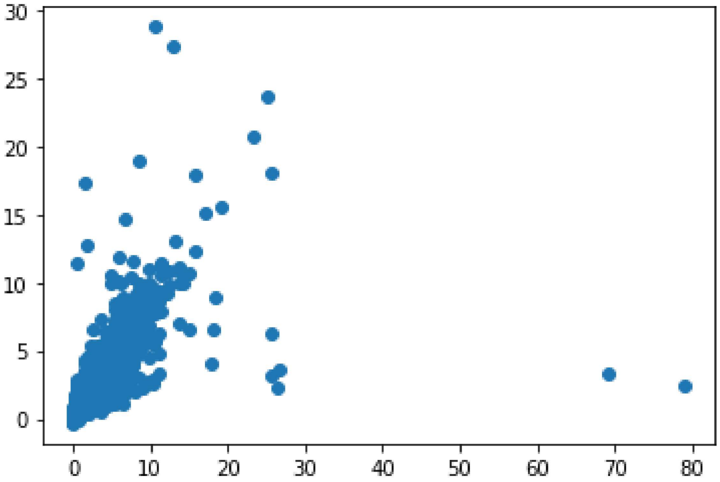
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
BEN	2.394190
CO	0.156559
EBE	1.799070
NMHC	5.568408
NO_2	0.001102
O_3	0.000709
PM10	0.005639
SO_2	-0.001743
TCH	-0.413736

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

Out[15]: <matplotlib.collections.PathCollection at 0x24d73917040>
```



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

         0.608862676725141
```

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

Out[17]: 0.7944686110318933
```

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]: 0.7935311151581462
```

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

Out[20]: 0.6059052961232164
```

Lasso Regression

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

Out[21]: 0.01929207195571936
```

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

```
Out[22]: 0.01518357981381624
```

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.         0.         0.         0.         0.01020351 0.
 0.02236037 0.         0.         ]
```

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

```
-0.1512106487369515
```

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

```
0.1141549057267518
```

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

```
0.13797303550394302
```

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]: (69096, 13)
```

```
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 = [[1.4,1.5,1.6,2.7,2.3,3.3,2.3,4.1,2.3,4.2,1.2,12,2]]
prediction = logs.predict(observation)
```

In [35]:

```
print(prediction)
```

```
[28079060]
```

In [36]:

```
logs.classes_
```

Out[36]:

```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      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))
```

```
0.040522938877900525
```

In [39]:

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

```
0.042177517729030126
```

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.7944686110318933 and x_test and y_test score is 0.608862676725141.

In [ ]: