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	TCI
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0	NaI
1	2017-06-01 01:00:00	0.6	NaN	0.3	0.4	0.08	3.0	39.0	NaN	71.0	22.0	9.0	7.0	1.4
2	2017-06-01 01:00:00	0.2	NaN	NaN	0.1	NaN	1.0	14.0	NaN	NaN	NaN	NaN	NaN	NaI
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN	NaI
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0	NaI
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN	NaI
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0	NaI
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN	NaI
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN	NaI
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN	NaI

210120 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	TCH
0	2017-06-01 01:00:00	0.0	0.0	0.3	0.0	0.00	4.0	38.0	0.0	0.0	0.0	0.0	5.0	0.0
1	2017-06-01 01:00:00	0.6	0.0	0.3	0.4	0.08	3.0	39.0	0.0	71.0	22.0	9.0	7.0	1.4
2	2017-06-01 01:00:00	0.2	0.0	0.0	0.1	0.00	1.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0
3	2017-06-01 01:00:00	0.0	0.0	0.2	0.0	0.00	1.0	9.0	0.0	91.0	0.0	0.0	0.0	0.0
4	2017-06-01 01:00:00	0.0	0.0	0.0	0.0	0.00	1.0	19.0	0.0	69.0	0.0	0.0	2.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210115	2017-08-01 00:00:00	0.0	0.0	0.2	0.0	0.00	1.0	27.0	0.0	65.0	0.0	0.0	0.0	0.0
210116	2017-08-01 00:00:00	0.0	0.0	0.2	0.0	0.00	1.0	14.0	0.0	0.0	73.0	0.0	7.0	0.0
210117	2017-08-01 00:00:00	0.0	0.0	0.0	0.0	0.00	1.0	4.0	0.0	83.0	0.0	0.0	0.0	0.0
210118	2017-08-01 00:00:00	0.0	0.0	0.0	0.0	0.00	1.0	11.0	0.0	78.0	0.0	0.0	0.0	0.0
210119	2017-08-01 00:00:00	0.0	0.0	0.0	0.0	0.00	1.0	14.0	0.0	77.0	60.0	0.0	0.0	0.0

210120 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      210120 non-null  float64
 2   CH4      210120 non-null  float64
 3   CO       210120 non-null  float64
 4   EBE      210120 non-null  float64
 5   NMHC     210120 non-null  float64
 6   NO       210120 non-null  float64
 7   NO_2     210120 non-null  float64
 8   NOx      210120 non-null  float64
 9   O_3      210120 non-null  float64
 10  PM10     210120 non-null  float64
 11  PM25     210120 non-null  float64
 12  SO_2     210120 non-null  float64
 13  TCH      210120 non-null  float64
 14  TOL      210120 non-null  float64
 15  station  210120 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 25.6+ MB
```

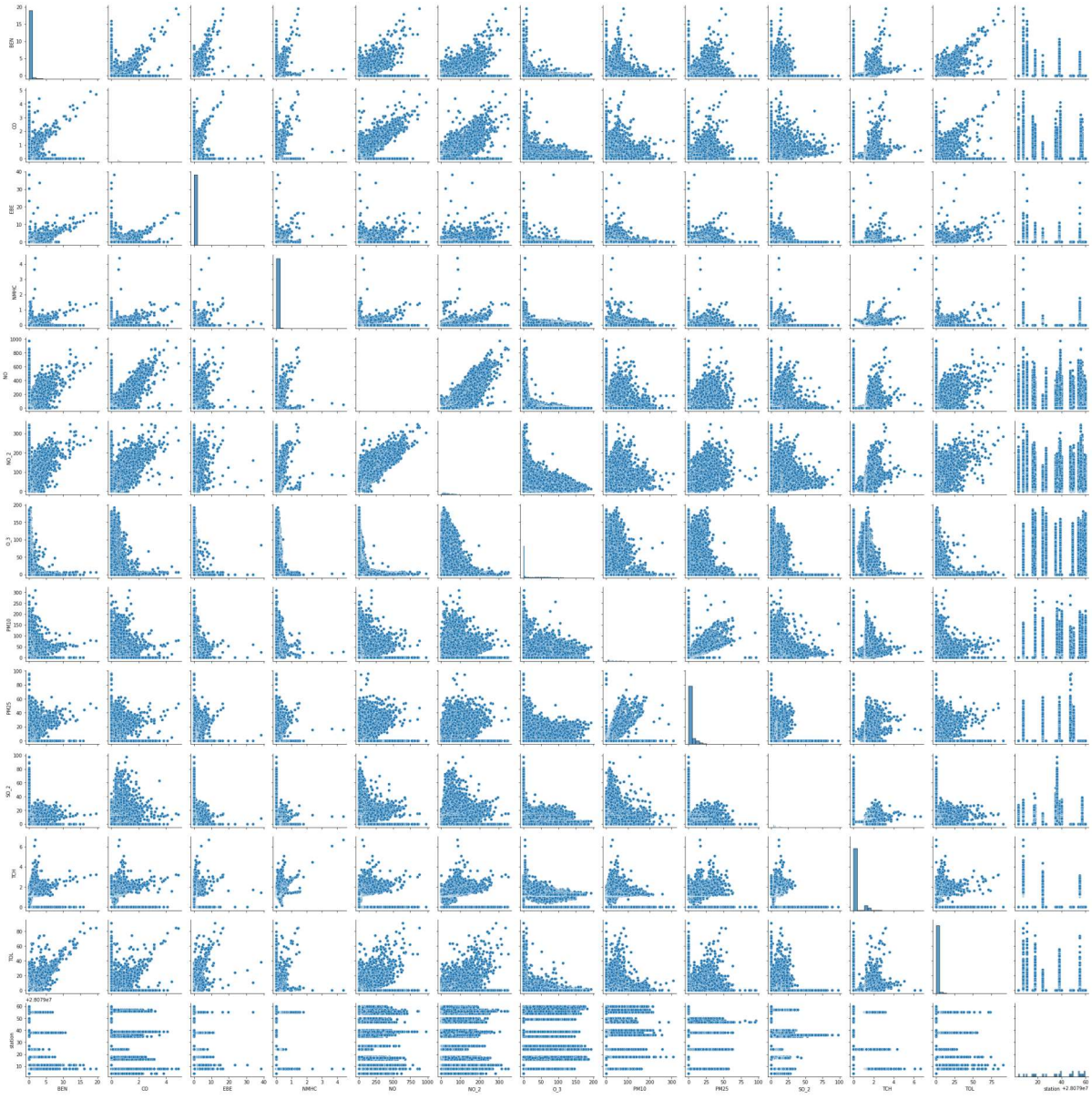
In [5]:
```
df1.columns
```

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)
```

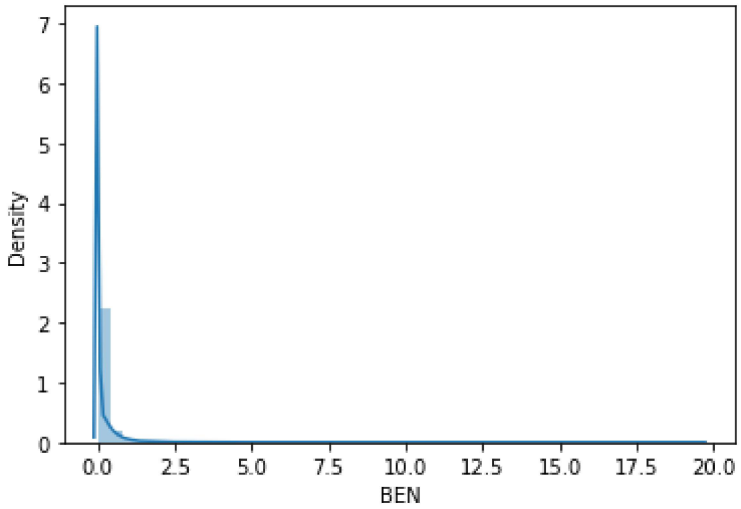
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x2b48eea72e0>
```



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

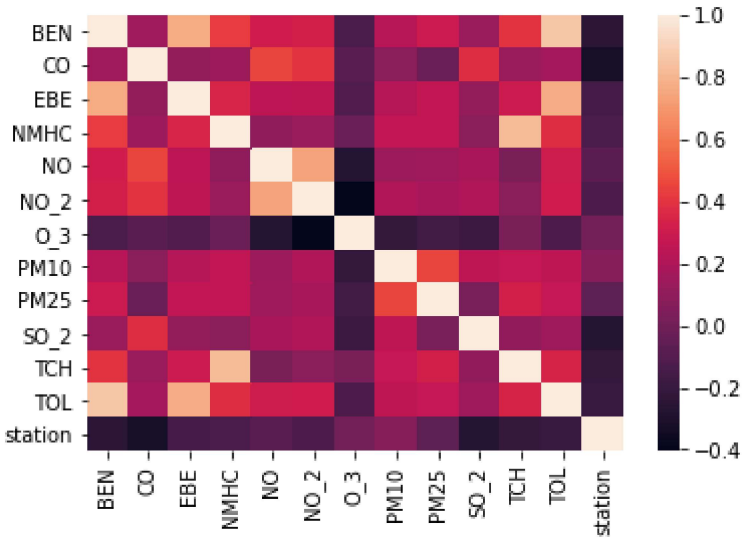
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

```
-0.17167456541078308
```

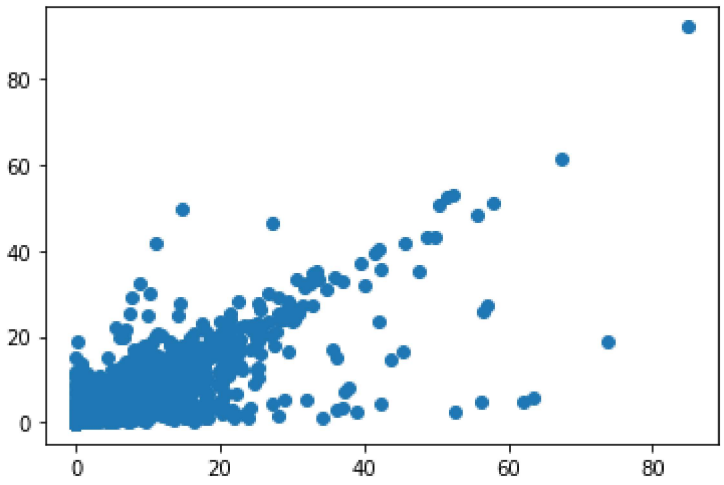
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
BEN	3.314568
CO	0.214024
EBE	1.524219
NMHC	-0.015600
NO_2	0.001396
O_3	0.001089
PM10	0.006430
SO_2	0.001581
TCH	-0.080214

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x2b49f63f100>
```



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

```
0.7560978519681353
```

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

Out[17]:
```
0.7939677103807374
```

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]:
```
0.7939676383020309
```

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

Out[20]:
```
0.7561124822395594
```

Lasso Regression

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

Out[21]:
```
0.0807494840814229
```

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

Out[22]:
```
0.07715945876763819
```

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_)
```

```
[0.38645152 0.         0.01237049 0.         0.01799895 0.00185033
 0.02577234 0.         0.         ]
```

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

```
-0.48968966210180154
```

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

```
0.24179440926172902
```

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

```
0.25152688776380705
```

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]:
```
(210120, 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)
```

```
[28079059]
```

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))
```

```
0.04200774160797005
```

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

```
0.04139131380707623
```

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.7939677103807374 and x_test and y_test score is 0.7560978519681353.

In [ ]: