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	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	2
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	2
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	2
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	2
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	2
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	2
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	2
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	2
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	2

209880 rows × 14 columns

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

Out[3]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2013-11-01 01:00:00	0.0	0.6	0.0	0.0	135.0	74.0	0.0	0.0	0.0	7.0	0.0	0.0	28
1	2013-11-01 01:00:00	1.5	0.5	1.3	0.0	71.0	83.0	2.0	23.0	16.0	12.0	0.0	8.3	28
2	2013-11-01 01:00:00	3.9	0.0	2.8	0.0	49.0	70.0	0.0	0.0	0.0	0.0	0.0	9.0	28
3	2013-11-01 01:00:00	0.0	0.5	0.0	0.0	82.0	87.0	3.0	0.0	0.0	0.0	0.0	0.0	28
4	2013-11-01 01:00:00	0.0	0.0	0.0	0.0	242.0	111.0	2.0	0.0	0.0	12.0	0.0	0.0	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	0.0	0.4	0.0	0.0	8.0	39.0	52.0	0.0	0.0	0.0	0.0	0.0	28
209876	2013-03-01 00:00:00	0.0	0.4	0.0	0.0	1.0	11.0	0.0	6.0	0.0	2.0	0.0	0.0	28
209877	2013-03-01 00:00:00	0.0	0.0	0.0	0.0	2.0	4.0	75.0	0.0	0.0	0.0	0.0	0.0	28
209878	2013-03-01 00:00:00	0.0	0.0	0.0	0.0	2.0	11.0	52.0	0.0	0.0	0.0	0.0	0.0	28
209879	2013-03-01 00:00:00	0.0	0.0	0.0	0.0	1.0	10.0	75.0	3.0	0.0	0.0	0.0	0.0	28

209880 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

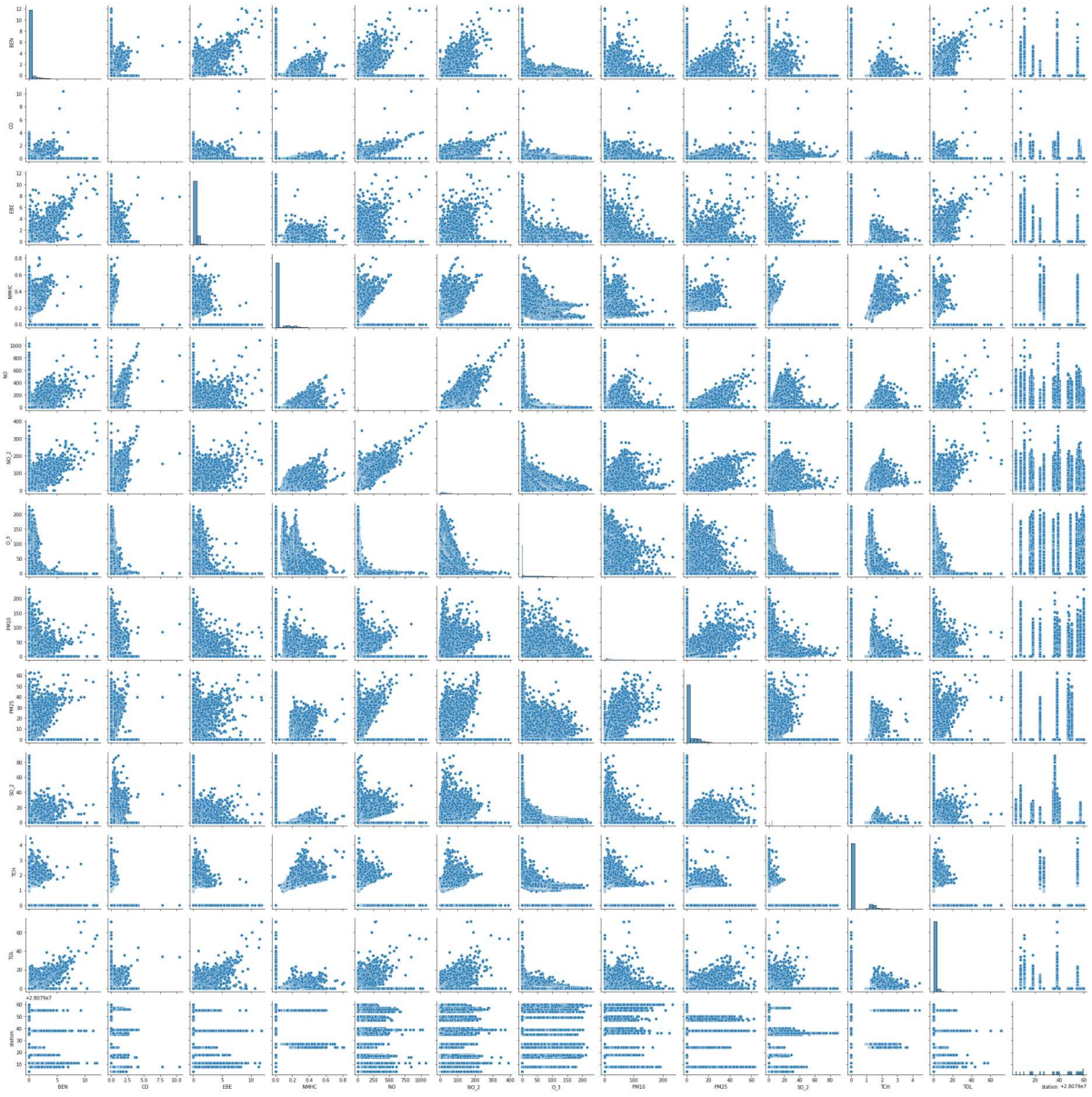
In [5]:
```
df1.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       '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)
```

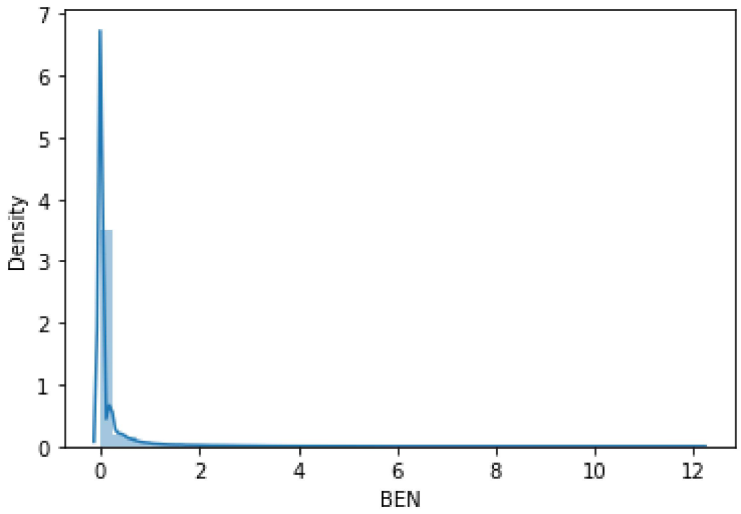
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x2d7a1c07070>
```



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

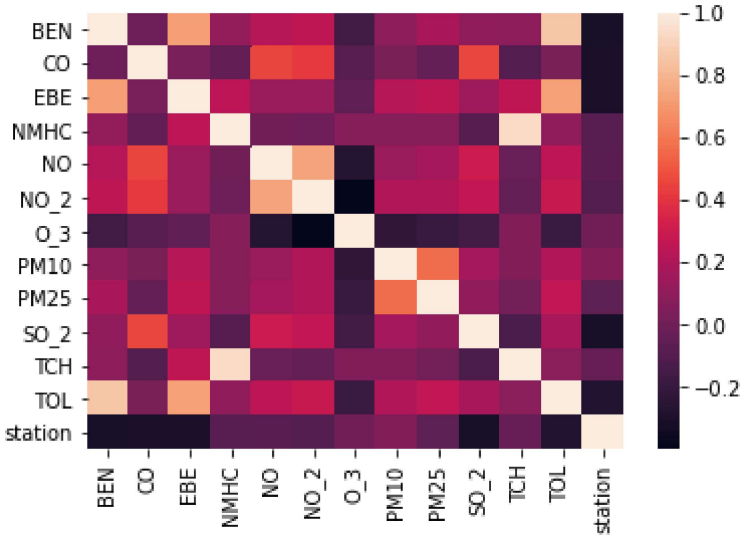
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

```
-0.16904769261130836
```

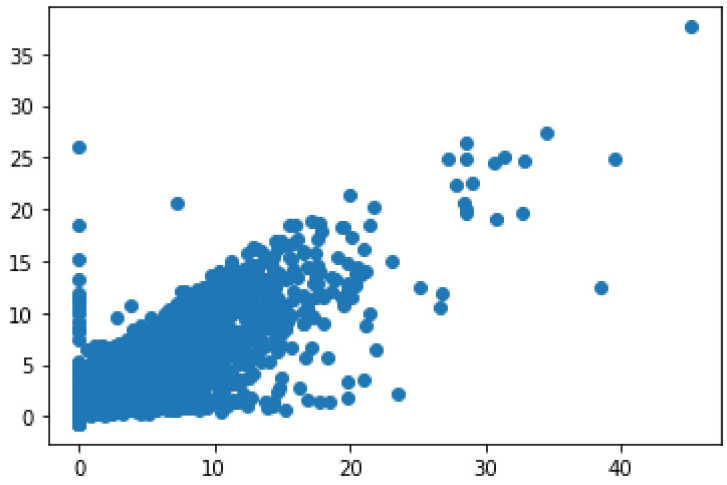
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
BEN	2.447570
CO	-0.096580
EBE	0.836385
NMHC	-1.126954
NO_2	0.003408
O_3	-0.000675
PM10	0.009174
SO_2	0.028376
TCH	0.041091

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x2d7b22325b0>
```



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

```
0.7859219097340893
```

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

Out[17]:
```
0.7820255282984251
```

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]:
```
0.782022808455145
```

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

Out[20]:
```
0.7859268373681724
```

Lasso Regression

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

Out[21]:
```
0.04460170689965948
```

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

Out[22]:
```
0.043789860640038425
```

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_)
```

```
[ 0.31259584 -0.          0.00611654  0.          0.01365985 -0.00255466
  0.0158514   0.00288115  0.        ]
```

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

```
0.014090238653764575
```

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

```
0.2379360000070897
```

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

```
0.23918651700229066
```

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]:
```
(209880, 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://scikit-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-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  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]]
prediction = logs.predict(observation)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-34-4487412f9698> in <module>
      1 observation = [[1.4,1.5,1.6,2.7,2.3,3.3,2.3,4.1,2.3,4.2,1.2]]
----> 2 prediction = logs.predict(observation)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in predict(self, X)
    307             Predicted class label per sample.
    308         """
--> 309         scores = self.decision_function(X)
    310         if len(scores.shape) == 1:
    311             indices = (scores > 0).astype(int)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in decision_function(self, X)
    286         n_features = self.coef_.shape[1]
    287         if X.shape[1] != n_features:
--> 288             raise ValueError("X has %d features per sample; expecting %d"
    289                              % (X.shape[1], n_features))
    290

ValueError: X has 11 features per sample; expecting 13
```

In [ ]:
```
print(prediction)
```

In [ ]:
```
logs.classes_
```

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

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

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.8038041251750647 and x_test and y_test score is 0.8096303763473245.

In [ ]: