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	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	2
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	2
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	2
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	2
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	2
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	2
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	2

209928 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	2011-11-01 01:00:00	0.0	1.0	0.0	0.00	154.0	84.0	0.0	0.0	0.0	6.0	0.00	0.0	28
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28
2	2011-11-01 01:00:00	2.9	0.0	3.8	0.00	96.0	99.0	0.0	0.0	0.0	0.0	0.00	7.2	28
3	2011-11-01 01:00:00	0.0	0.6	0.0	0.00	60.0	83.0	2.0	0.0	0.0	0.0	0.00	0.0	28
4	2011-11-01 01:00:00	0.0	0.0	0.0	0.00	44.0	62.0	3.0	0.0	0.0	3.0	0.00	0.0	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	0.0	0.2	0.0	0.00	5.0	19.0	44.0	0.0	0.0	0.0	0.00	0.0	28
209924	2011-09-01 00:00:00	0.0	0.1	0.0	0.00	6.0	29.0	0.0	11.0	0.0	7.0	0.00	0.0	28
209925	2011-09-01 00:00:00	0.0	0.0	0.0	0.23	1.0	21.0	28.0	0.0	0.0	0.0	1.44	0.0	28
209926	2011-09-01 00:00:00	0.0	0.0	0.0	0.00	3.0	15.0	48.0	0.0	0.0	0.0	0.00	0.0	28
209927	2011-09-01 00:00:00	0.0	0.0	0.0	0.00	4.0	33.0	38.0	13.0	0.0	0.0	0.00	0.0	28

209928 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209928 entries, 0 to 209927
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209928 non-null  object 
 1   BEN      209928 non-null  float64
 2   CO       209928 non-null  float64
 3   EBE      209928 non-null  float64
 4   NMHC     209928 non-null  float64
 5   NO       209928 non-null  float64
 6   NO_2     209928 non-null  float64
 7   O_3      209928 non-null  float64
 8   PM10     209928 non-null  float64
 9   PM25     209928 non-null  float64
 10  SO_2     209928 non-null  float64
 11  TCH      209928 non-null  float64
 12  TOL      209928 non-null  float64
 13  station  209928 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

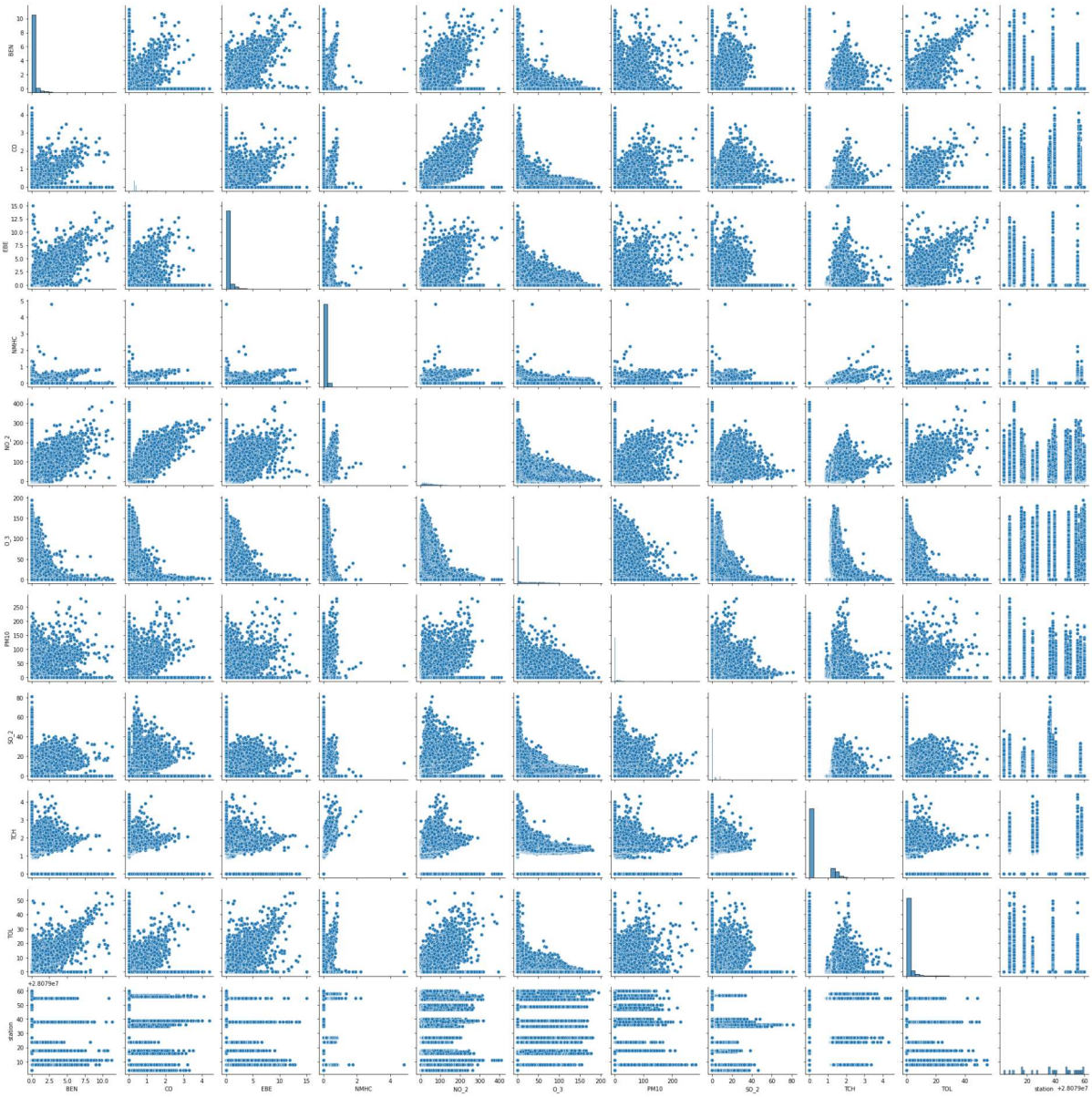
In [5]:
```
df1.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [7]:
```
df2 = df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [8]:
```
sns.pairplot(df2)
```

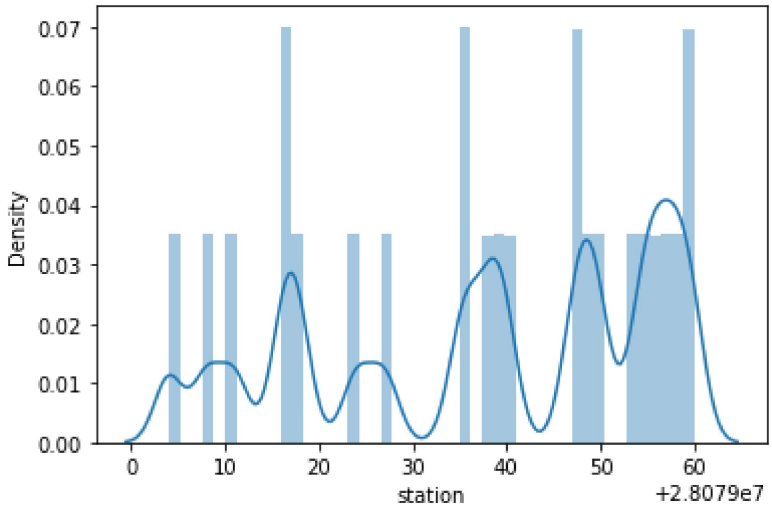
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2343f9fc0d0>
```



In [11]:
```
sns.distplot(df2['station'])
```

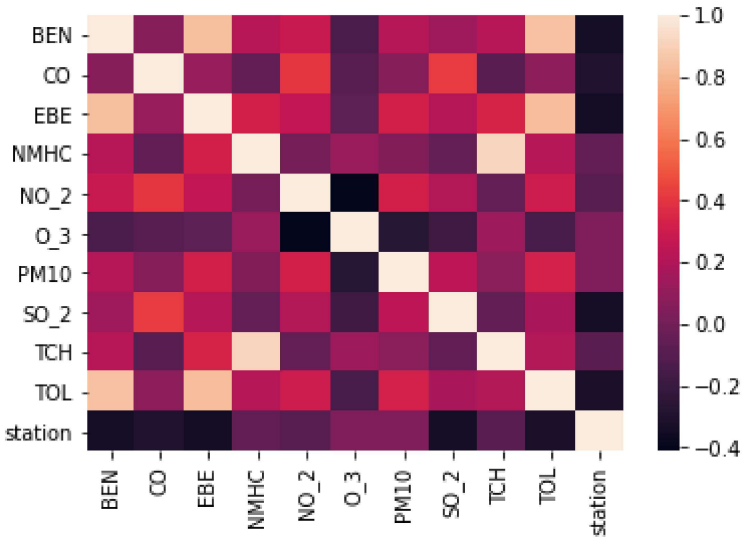
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[11]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [10]:
```
sns.heatmap(df2.corr())
```

Out[10]: <AxesSubplot:>



Linear Regression

In [12]:
```
x = df2[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH']]
y = df2['TOL']
```

In [13]:
```
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 [14]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
0.03276585532911702
```

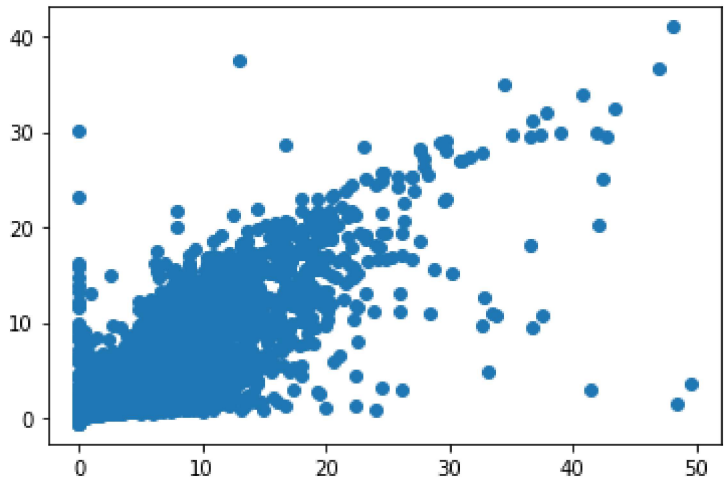
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
BEN	2.030339
CO	-0.191419
EBE	1.462198
NMHC	1.777961
NO_2	0.001853
O_3	-0.001643
PM10	0.010513
SO_2	0.002611
TCH	-0.430407

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x2344a045d90>
```



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.773521701312003
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]:
```
0.7848390703863859
```

Ridge and Lasso

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

Out[21]:
```
0.7848370696834108
```

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

Out[22]:
```
0.7735588174527517
```

Lasso Regression

In [23]:
```
ls = Lasso(alpha=10)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
```

Out[23]:
```
0.08845469383755222
```

In [24]:
```
ls.score(x_test,y_test)
```

Out[24]:
```
0.08867849552144558
```

ElacticNET regression

In [25]:
```
from sklearn.linear_model import ElasticNet
es = ElasticNet()
es.fit(x_train,y_train)
```

Out[25]:
```
ElasticNet()
```

In [26]:
```
print(es.coef_)
```

```
[ 4.82007060e-01 -0.00000000e+00  5.76536821e-01  0.00000000e+00
  1.20001330e-02 -1.95242620e-04  2.55589026e-02  4.62469419e-03
  0.00000000e+00]
```

In [27]:
```
print(es.intercept_)
```

```
-0.2038597760522718
```

In [28]:
```
print(es.score(x_test,y_test))
```

```
0.48002892644823913
```

In [29]:
```
print(es.score(x_train,y_train))
```

```
0.4814298885597946
```

LogisticRegression

In [30]:
```
from sklearn.linear_model import LogisticRegression
```

In [31]:
```
feature_matrix = df2.iloc[:,0:15]
target_vector = df2.iloc[:,-1]
```

In [32]:
```
feature_matrix.shape
```

Out[32]:
```
(209928, 11)
```

In [33]:
```
from sklearn.preprocessing import StandardScaler
```

In [34]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [35]:
```
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[35]:
```
LogisticRegression()
```

In [37]:
```
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)
```

In [38]:
```
print(prediction)
```

```
[28079059]
```

In [39]:
```
logs.classes_
```

Out[39]:
```
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 [40]:
```
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 [41]:
```
print(logs.score(x_test,y_test))
```

```
0.0407278616681751
```

In [43]:
```
print(logs.score(x_train,y_train))
```

```
0.04150419533307474
```

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.7848390703863859 and x_test and y_test score is 0.773521701312003.

In [ ]: