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	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28

210096 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	s
0	2015-10-01 01:00:00	0.0	0.8	0.0	0.00	90.0	82.0	0.0	0.0	0.0	10.0	0.00	0.0	280
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	280
2	2015-10-01 01:00:00	3.1	0.0	1.8	0.00	29.0	97.0	0.0	0.0	0.0	0.0	0.00	7.1	280
3	2015-10-01 01:00:00	0.0	0.6	0.0	0.00	30.0	103.0	2.0	0.0	0.0	0.0	0.00	0.0	280
4	2015-10-01 01:00:00	0.0	0.0	0.0	0.00	95.0	96.0	2.0	0.0	0.0	9.0	0.00	0.0	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210091	2015-08-01 00:00:00	0.0	0.2	0.0	0.00	11.0	33.0	53.0	0.0	0.0	0.0	0.00	0.0	280
210092	2015-08-01 00:00:00	0.0	0.2	0.0	0.00	1.0	5.0	0.0	26.0	0.0	10.0	0.00	0.0	280
210093	2015-08-01 00:00:00	0.0	0.0	0.0	0.00	1.0	7.0	74.0	0.0	0.0	0.0	0.00	0.0	280
210094	2015-08-01 00:00:00	0.0	0.0	0.0	0.00	3.0	7.0	65.0	0.0	0.0	0.0	0.00	0.0	280
210095	2015-08-01 00:00:00	0.0	0.0	0.0	0.00	1.0	9.0	54.0	29.0	0.0	0.0	0.00	0.0	280

210096 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210096 entries, 0 to 210095
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210096 non-null  object 
 1   BEN      210096 non-null  float64
 2   CO       210096 non-null  float64
 3   EBE      210096 non-null  float64
 4   NMHC     210096 non-null  float64
 5   NO       210096 non-null  float64
 6   NO_2     210096 non-null  float64
 7   O_3      210096 non-null  float64
 8   PM10     210096 non-null  float64
 9   PM25     210096 non-null  float64
 10  SO_2     210096 non-null  float64
 11  TCH      210096 non-null  float64
 12  TOL      210096 non-null  float64
 13  station  210096 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

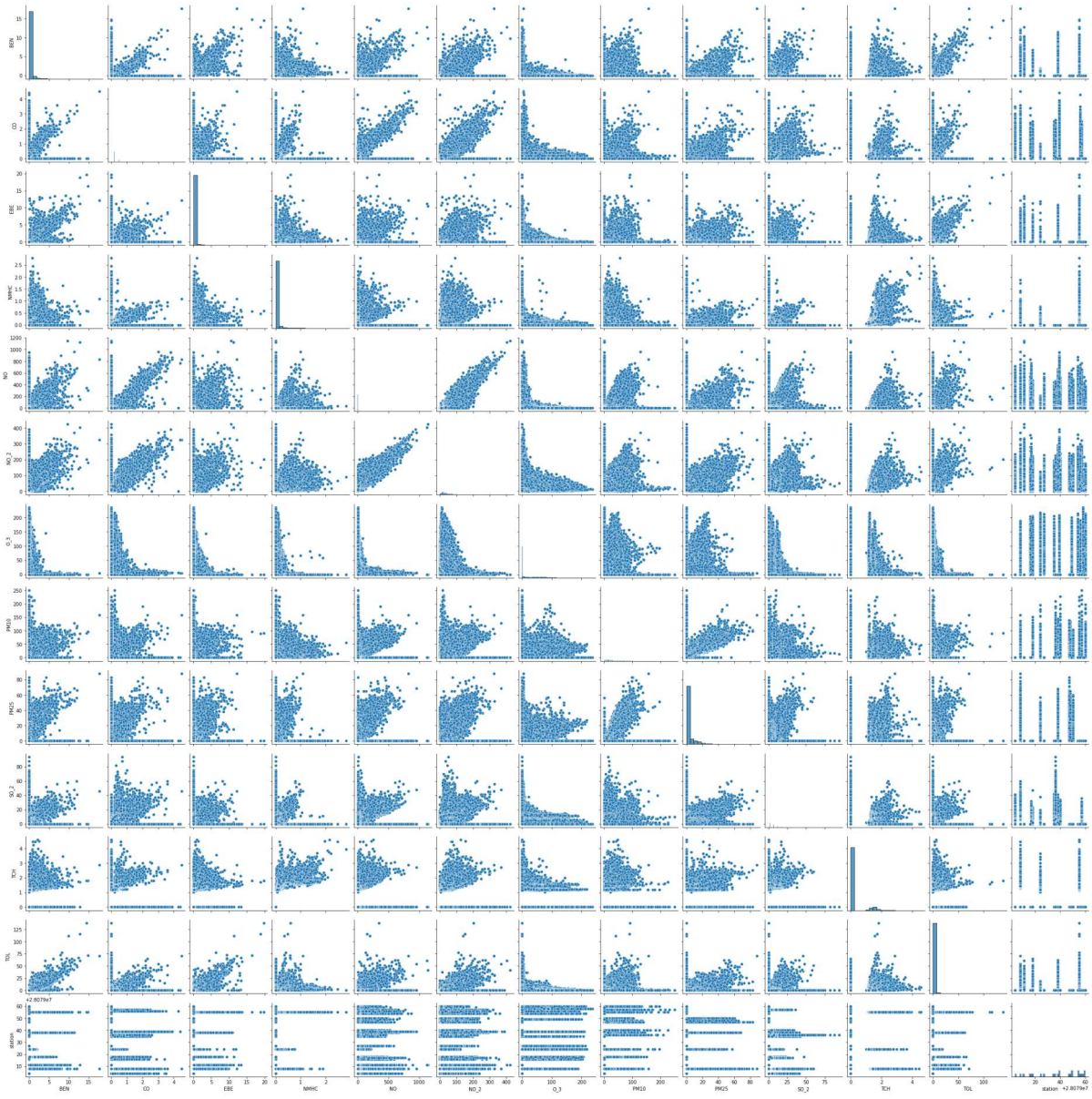
In [5]:
```
df1.columns
```

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)
```

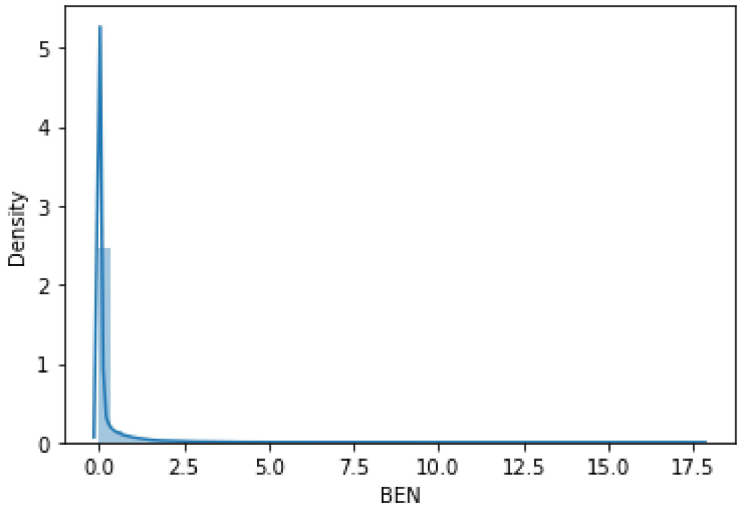
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1d42906a130>
```



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

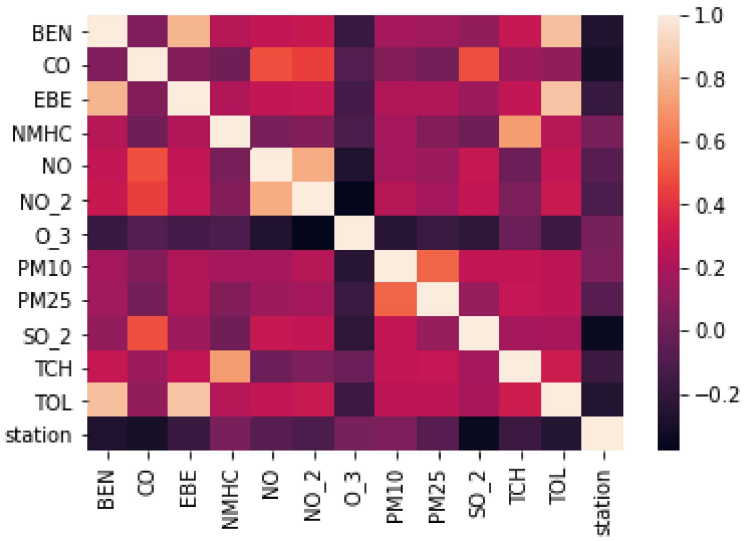
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function 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 [40]:
```
x = df2[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2',
       'PM10', 'SO_2', 'TCH','TOL']]
y = df2['O_3']
```

In [41]:
```
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 [42]:
```
from sklearn.linear_model import LinearRegression

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

Out[42]:
```
LinearRegression()
```

In [43]:
```
print(lr.intercept_)
```

```
49.11218642212168
```

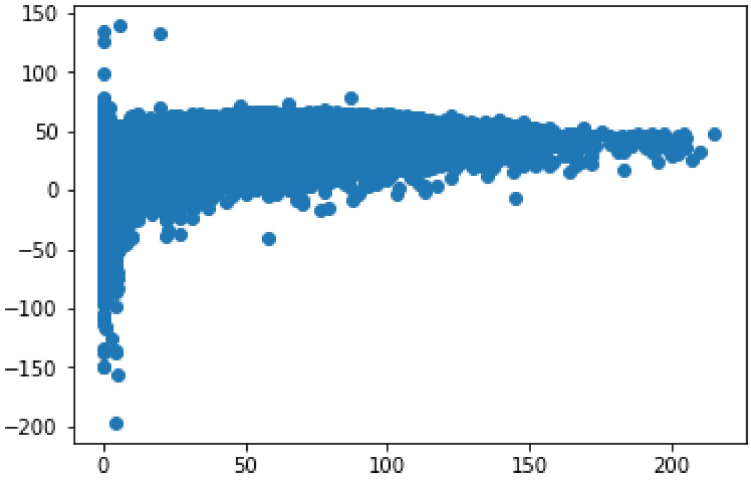
In [44]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[44]:

	Co-efficient
BEN	-9.186901
CO	20.988326
EBE	8.604433
NMHC	-55.814341
NO_2	-0.394775
PM10	-0.332471
SO_2	-1.439942
TCH	16.047494
TOL	0.112111

In [45]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[45]: <matplotlib.collections.PathCollection at 0x1d435557970>



In [46]:
```
print(lr.score(x_test,y_test))
```

0.22729184038810668

In [47]:
```
lr.score(x_train,y_train)
```

Out[47]: 0.22355902200074063

Ridge and Lasso

In [48]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [49]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_train,y_train)
```

Out[49]: 0.22355761559400067

In [50]:
```
rr.score(x_test,y_test)
```

Out[50]: 0.2272926233962146

Lasso Regression

In [51]:
```
ls = Lasso(alpha=10)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
```

Out[51]: 0.17059071420820138

In [52]:
```
ls.score(x_test,y_test)
```

Out[52]:
```
0.17339334181090804
```

ElacticNET regression

In [53]:
```
from sklearn.linear_model import ElasticNet
es = ElasticNet()
es.fit(x_train,y_train)
```

Out[53]:
```
ElasticNet()
```

In [54]:
```
print(es.coef_)
```

```
[-0.68490241  1.04859657  0.         -0.         -0.36845855 -0.32903367
 -0.64602985  1.07044187 -0.        ]
```

In [55]:
```
print(es.intercept_)
```

```
49.31283268498254
```

In [56]:
```
print(es.score(x_test,y_test))
```

```
0.18134271131610658
```

In [57]:
```
print(es.score(x_train,y_train))
```

```
0.17826750967872906
```

LogisticRegression

In [58]:
```
from sklearn.linear_model import LogisticRegression
```

In [59]:
```
feature_matrix = df2.iloc[:,0:15]
target_vector = df2.iloc[:,-1]
```

In [60]:
```
feature_matrix.shape
```

Out[60]:
```
(210096, 13)
```

In [61]:
```
from sklearn.preprocessing import StandardScaler
```

In [62]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [63]:
```
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[63]:
```
LogisticRegression()
```

In [64]:
```
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))
```

```
0.04112392708118485
```

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

```
0.04194006813221185
```

Conclusion

linear regression is bestfit model

linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.22355902200074063 and x_test and y_test score is 0.22729184038810668.

In [ ]: