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\madrid_200
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.34
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.1	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.85
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.46
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.80
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2001-08-02 18:00:00	NaN	0.09	NaN	NaN	0.09	27.670000	33.189999	NaN	93.559998	30.309999	NaN	3.33
996	2001-08-02 18:00:00	NaN	0.41	NaN	NaN	NaN	45.639999	62.180000	NaN	86.820000	44.279999	NaN	6.77
997	2001-08-02 18:00:00	1.28	0.35	1.68	NaN	0.10	51.560001	84.430000	NaN	56.520000	50.509998	NaN	3.15
998	2001-08-02 18:00:00	NaN	0.11	NaN	NaN	NaN	33.270000	42.939999	NaN	93.910004	25.760000	NaN	5.18
999	2001-08-02 18:00:00	NaN	0.05	NaN	NaN	0.04	11.520000	12.950000	NaN	83.019997	35.660000	NaN	7.19

1000 rows × 16 columns

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

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2001-08-01 01:00:00	0.00	0.37	0.00	0.0	0.00	58.400002	87.150002	0.00	34.529999	105.000000	0.00	6.34
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.1	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11
2	2001-08-01 01:00:00	0.00	0.28	0.00	0.0	0.00	50.660000	61.380001	0.00	46.310001	100.099998	0.00	7.85
3	2001-08-01 01:00:00	0.00	0.47	0.00	0.0	0.00	69.790001	73.449997	0.00	40.650002	69.779999	0.00	6.46
4	2001-08-01 01:00:00	0.00	0.39	0.00	0.0	0.00	22.830000	24.799999	0.00	66.309998	75.180000	0.00	8.80
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2001-08-02 18:00:00	0.00	0.09	0.00	0.0	0.09	27.670000	33.189999	0.00	93.559998	30.309999	0.00	3.33
996	2001-08-02 18:00:00	0.00	0.41	0.00	0.0	0.00	45.639999	62.180000	0.00	86.820000	44.279999	0.00	6.77
997	2001-08-02 18:00:00	1.28	0.35	1.68	0.0	0.10	51.560001	84.430000	0.00	56.520000	50.509998	0.00	3.15
998	2001-08-02 18:00:00	0.00	0.11	0.00	0.0	0.00	33.270000	42.939999	0.00	93.910004	25.760000	0.00	5.18
999	2001-08-02 18:00:00	0.00	0.05	0.00	0.0	0.04	11.520000	12.950000	0.00	83.019997	35.660000	0.00	7.19

1000 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1000 non-null   object 
 1   BEN      1000 non-null   float64
 2   CO       1000 non-null   float64
 3   EBE      1000 non-null   float64
 4   MXY      1000 non-null   float64
 5   NMHC     1000 non-null   float64
 6   NO_2     1000 non-null   float64
 7   NOx      1000 non-null   float64
 8   OXY      1000 non-null   float64
 9   O_3      1000 non-null   float64
 10  PM10     1000 non-null   float64
 11  PXY      1000 non-null   float64
 12  SO_2     1000 non-null   float64
 13  TCH      1000 non-null   float64
 14  TOL      1000 non-null   float64
 15  station  1000 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 125.1+ KB
```

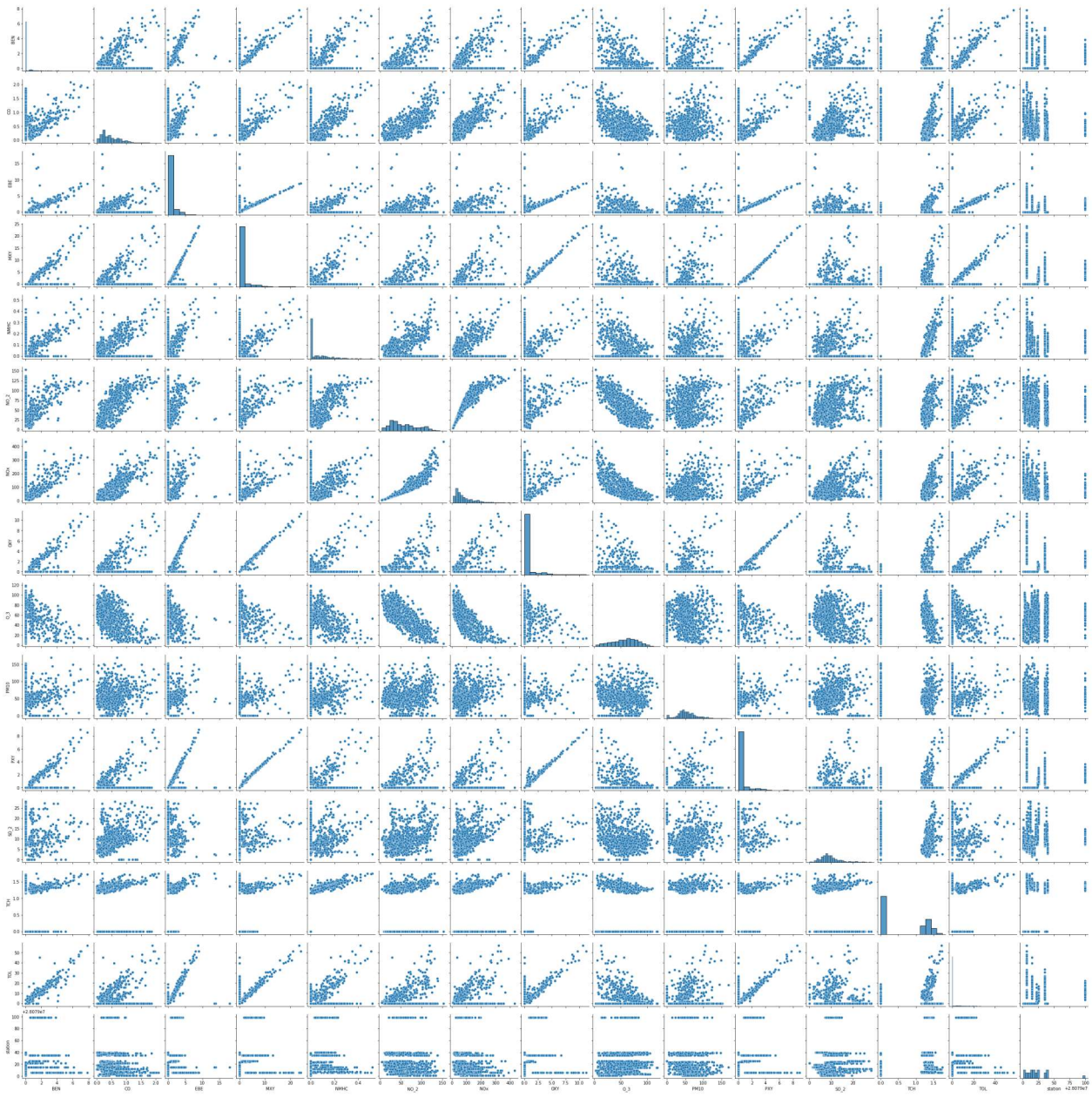
In [5]:
```
df1.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
df2 = df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

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

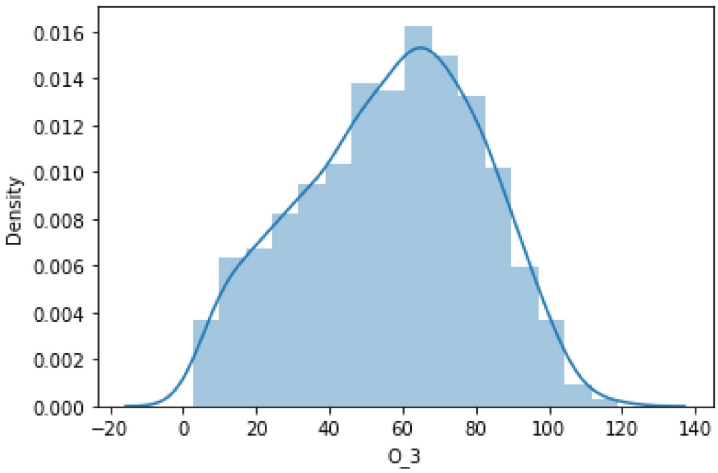
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1d84a1662e0>
```



In [8]:
```
sns.distplot(df2['O_3'])
```

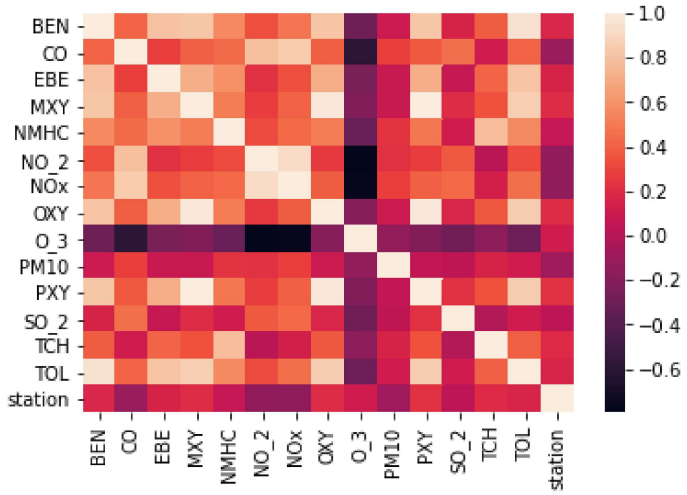
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='O_3', ylabel='Density'>



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

Out[9]: <AxesSubplot:>



Linear Regression

In [40]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PXY', '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_)
```

```
89.8621080888955
```

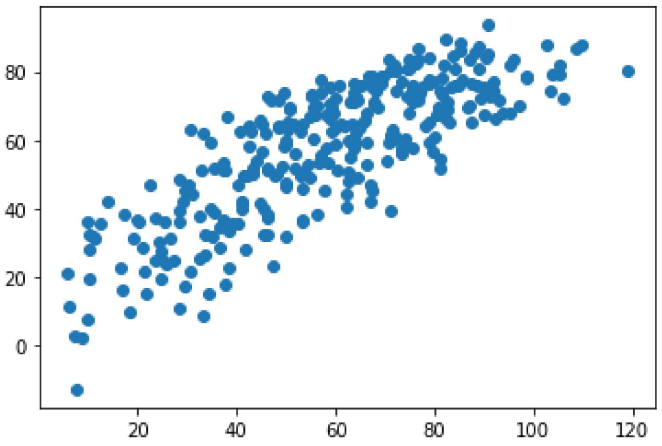
In [44]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[44]:

	Co-efficient
BEN	4.376421
CO	9.436895
EBE	-1.873199
MXY	3.097010
NMHC	39.968550
NO_2	-0.466473
NOx	-0.148728
OXY	3.996522
PM10	0.048836
PXY	-10.462474
SO_2	0.178025
TCH	-8.736975
TOL	-0.571089

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

Out[45]:
```
<matplotlib.collections.PathCollection at 0x1d859737d60>
```



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

```
0.6595392343596381
```

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

Out[47]:
```
0.7067374753517931
```

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]:
```
0.3363703840853174
```

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

Out[20]:
```
0.39128151985862714
```

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

Out[21]:
```
0.17842918731839452
```

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

Out[22]:
```
0.20430210357658973
```

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.          0.23435945 -0.         -0.03982419
  0.05451099  0.         -0.00769116  0.         -0.         -0.15602306]
```

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

```
7.99673787041862
```

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

```
0.2560465501355159
```

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

```
0.21833619156229045
```

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]:
```
(1000, 15)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [33]:
```
logs = LogisticRegression()
logs.fit(fs,target_vector)
```

Out[33]:
```
LogisticRegression()
```

In [34]:
```
observation = [[1.4,1.5,1.6,2.7,2.3,3.3,2.3,4.1,2.3,4.2,1.2,2.1,4.3,6,2.2]]
prediction = logs.predict(observation)
```

In [35]:
```
print(prediction)
```

```
[28079035]
```

In [36]:
```
logs.classes_
```

Out[36]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079009,
       28079011, 28079012, 28079014, 28079015, 28079016, 28079018,
       28079019, 28079021, 28079022, 28079023, 28079024, 28079025,
       28079035, 28079036, 28079038, 28079039, 28079040, 28079099],
      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,test_size=0
```

In [38]:
```
print(logs.score(x_test,y_test))
```

```
0.04666666666666667
```

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

```
0.04
```

Conclusion

Linear regression is bestfit model

Linear regression is best fit model for dataset madrid_2001. The score of x_train,y_train is 0.6595392343596381 and x_test and y_test score is 0.7067374753517931.

In [ ]: