Final Assessment 1

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\madrid_2001.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.0
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.0
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.7
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.8
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.8
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.3

217872 rows × 16 columns

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [4]:
```
df=data.fillna(value=0)
df
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	0.00	0.37	0.00	0.00	0.00	58.400002	87.150002	0.00	34.529999	105.0
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	0.00	0.28	0.00	0.00	0.00	50.660000	61.380001	0.00	46.310001	100.0
3	2001-08-01 01:00:00	0.00	0.47	0.00	0.00	0.00	69.790001	73.449997	0.00	40.650002	69.7
4	2001-08-01 01:00:00	0.00	0.39	0.00	0.00	0.00	22.830000	24.799999	0.00	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	0.00	0.00	0.00	73.000000	264.399994	0.00	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	0.00	0.13	71.080002	129.300003	0.00	13.460000	26.8
217869	2001-04-01 00:00:00	0.49	1.09	0.00	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	0.00	80.019997	197.000000	2.58	5.840000	37.8
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.3

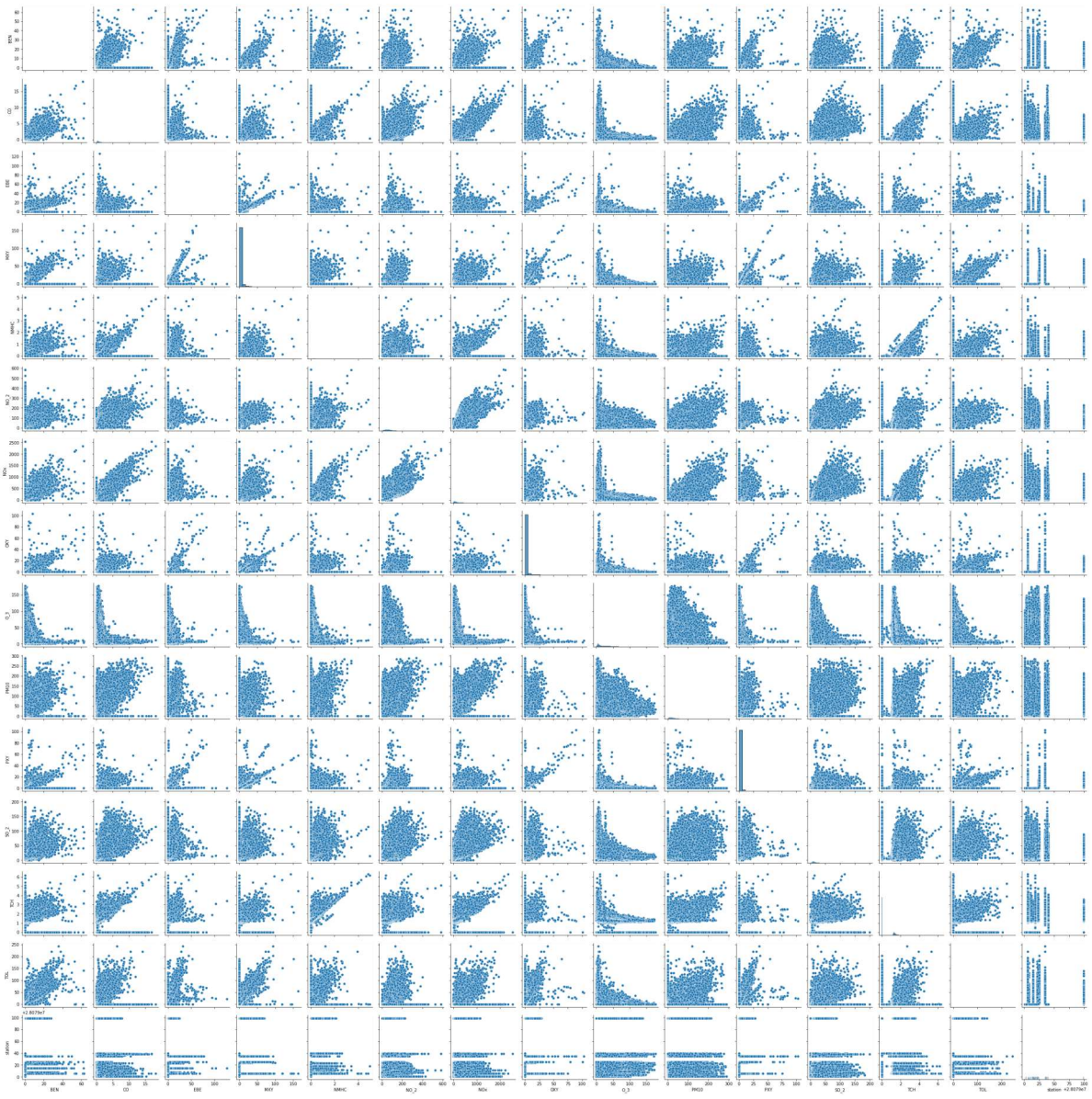
217872 rows × 16 columns

In [5]:
```
df.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]:
```
sns.pairplot(df)
```

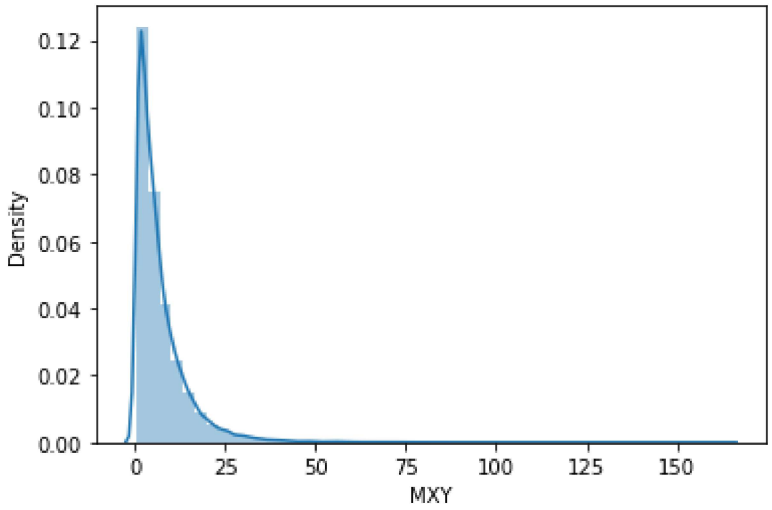
Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x1828d4b2d60>
```



In [7]:
```
sns.distplot(data["MXY"])
```

```
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[7]:
```
<AxesSubplot:xlabel='MXY', ylabel='Density'>
```



MODEL BUILDING

Linear Regression

In [8]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [52]:
```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
y=df1[['MXY']]
```

In [53]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [54]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

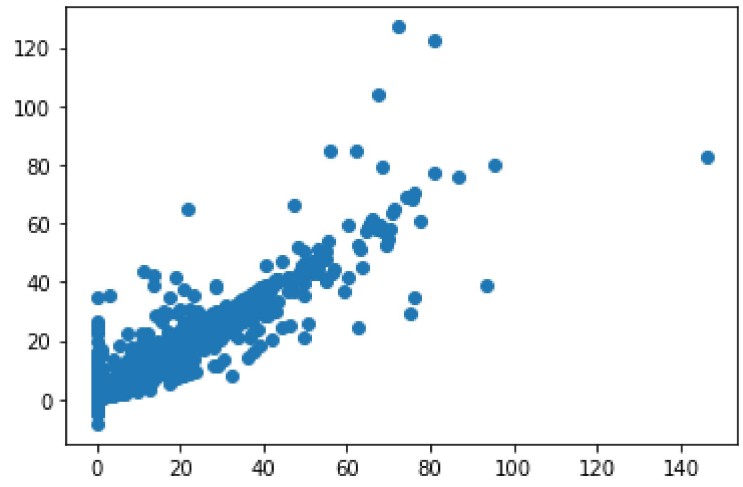
Out[54]:
```
LinearRegression()
```

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

```
[-205913.2496604]
```

In [56]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[56]:
```
<matplotlib.collections.PathCollection at 0x182b8cf1e20>
```



In [57]:
```
print(lr.score(x_test,y_test))
```

```
0.9268523589137899
```

Ridge Regression

In [16]:
```
from sklearn.linear_model import Ridge
```

In [17]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[17]:
```
Ridge(alpha=10)
```

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

Out[18]:
```
0.999999999825052
```

Lasso Regression

In [19]:
```
from sklearn.linear_model import Lasso
```

In [20]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[20]:
```
Lasso(alpha=10)
```

In [21]:
```
la.score(x_test,y_test)
```

Out[21]:
```
0.6741464245414455
```

Elastic Regression

In [22]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]:
```
ElasticNet()
```

In [23]:
```
print(en.coef_)
```

```
[ 0.00000000e+00 -0.00000000e+00  0.00000000e+00  9.07471959e-01
  0.00000000e+00 -0.00000000e+00  1.37424452e-04  0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  2.23985202e-02  1.20407602e-03]
```

In [24]:
```
print(en.predict(x_test))
```

```
[1.49871934e-02 5.33998002e+00 2.13385821e-02 ... 5.76391709e-02
 2.05472263e-01 2.14516691e+01]
```

In [25]:
```
print(en.score(x_test,y_test))
```

```
0.9961086670322732
```

Logistic Regression

In [26]:
```
from sklearn.linear_model import LogisticRegression
```

In [27]:
```
feature_matrix=df1.iloc[:,0:15]
target_vector=df1.iloc[:,-1]
```

In [28]:
```
feature_matrix.shape
```

Out[28]:
```
(217872, 15)
```

In [29]:
```
target_vector.shape
```

Out[29]:
```
(217872,)
```

In [30]:
```
from sklearn.preprocessing import StandardScaler
```

In [31]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [32]:
```
logr=LogisticRegression()
logr.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[32]:
```
LogisticRegression()
```

In [33]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]]
```

In [34]:
```
prediction=logr.predict(observation)
print(observation)
```

```
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]]
```

In [35]:
```
logr.classes_
```

Out[35]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [36]:
```
logr.score(fs,target_vector)
```

Out[36]:
```
0.9029889109201733
```

Random Forest

In [43]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [45]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.70)
```

In [46]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
<ipython-input-46-6e96556a06e6>:2: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_sa
mples,), for example using ravel().
  rfc.fit(x_train,y_train)
```

Out[46]: RandomForestClassifier()

In [47]:
```
parameters={'max_depth':[1,2,3,4,5],
           'min_samples_leaf':[5,10,15,20,25],
           'n_estimators':[10,20,30,40,50]}
```

In [48]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='acc
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:593: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:593: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:593: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:593: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for exam
ple using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

In [49]:
```
grid_search.best_score_
```

Out[49]: 0.4711372188677336

In [50]:
```
rfc_best=grid_search.best_estimator_
```

In [51]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

```
0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 33, 0, 0, 0, 0, 0,
0]'),
 Text(3777.230769230769, 1268.4, 'MXY <= 1.575\ngini = 0.787\nsamples = 70
14\nvalue = [0, 0, 0, 2602, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 246
4, 1092, 2526, 0, 0, 0, 0, 2563]'),
 Text(3433.8461538461534, 906.0, 'NO_2 <= 28.195\ngini = 0.496\nsamples =
1387\nvalue = [0, 0, 0, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 151
3, 277, 311, 0, 0, 0, 0, 102]'),
 Text(3262.1538461538457, 543.5999999999999, 'EBE <= 0.07\ngini = 0.345\ns
amples = 892\nvalue = [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0,
0, 1134, 170, 72, 0, 0, 0, 0, 41]'),
 Text(3176.307692307692, 181.19999999999982, 'gini = 0.013\nsamples = 397
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 622, 0,
4, 0, 0, 0, 0, 0]'),
 Text(3347.9999999999995, 181.19999999999982, 'gini = 0.53\nsamples = 495
\nvalue = [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 512, 170,
68, 0, 0, 0, 0, 41]'),
 Text(3605.5384615384614, 543.5999999999999, 'CO <= 0.465\ngini = 0.658\ns
amples = 495\nvalue = [0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0,
0, 379, 107, 239, 0, 0, 0, 0, 61]'),
```

Results

```
The best model is Ridge Regression 0.999999999825052
```