Final Assessment 1

Kaviyadevi(20106064)

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

In [2]:
```
#importing dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\madrid_2012.csv")
data1
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	280
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	280
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	280
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	280
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	280
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	280
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	280
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	280
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	280

210720 rows × 14 columns

In [3]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210720 entries, 0 to 210719
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210720 non-null  object 
 1   BEN      51511 non-null   float64
 2   CO       87097 non-null   float64
 3   EBE      51482 non-null   float64
 4   NMHC     30736 non-null   float64
 5   NO       209871 non-null  float64
 6   NO_2     209872 non-null  float64
 7   O_3      122339 non-null  float64
 8   PM10     104838 non-null  float64
 9   PM25     52164 non-null   float64
 10  SO_2     87333 non-null   float64
 11  TCH      30736 non-null   float64
 12  TOL      51373 non-null   float64
 13  station  210720 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.5+ MB
```

In [4]:
```
data=data1.head(50000)
```

In [5]:
```
#filling null values
df=data.fillna(0)
df
```

Out[5]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	stati
0	2012-09-01 01:00:00	0.0	0.2	0.0	0.00	7.0	18.0	0.0	0.0	0.0	2.0	0.00	0.0	280790
1	2012-09-01 01:00:00	0.3	0.3	0.7	0.00	3.0	18.0	55.0	10.0	9.0	1.0	0.00	2.4	280790
2	2012-09-01 01:00:00	0.4	0.0	0.7	0.00	2.0	10.0	0.0	0.0	0.0	0.0	0.00	1.5	280790
3	2012-09-01 01:00:00	0.0	0.2	0.0	0.00	1.0	6.0	50.0	0.0	0.0	0.0	0.00	0.0	280790
4	2012-09-01 01:00:00	0.0	0.0	0.0	0.00	1.0	13.0	54.0	0.0	0.0	3.0	0.00	0.0	280790
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49995	2012-03-26 20:00:00	0.0	0.3	0.0	0.00	7.0	54.0	33.0	0.0	0.0	0.0	0.00	0.0	280790
49996	2012-03-26 20:00:00	0.0	0.0	0.0	0.00	3.0	29.0	62.0	0.0	0.0	3.0	0.00	0.0	280790
49997	2012-03-26 20:00:00	0.2	0.4	1.0	0.00	4.0	36.0	62.0	29.0	0.0	2.0	0.00	0.4	280790
49998	2012-03-26 20:00:00	0.3	0.2	0.2	0.17	1.0	26.0	72.0	25.0	14.0	2.0	1.23	0.8	280790
49999	2012-03-26 20:00:00	0.0	0.0	0.0	0.17	1.0	31.0	67.0	0.0	0.0	0.0	1.22	0.0	280790

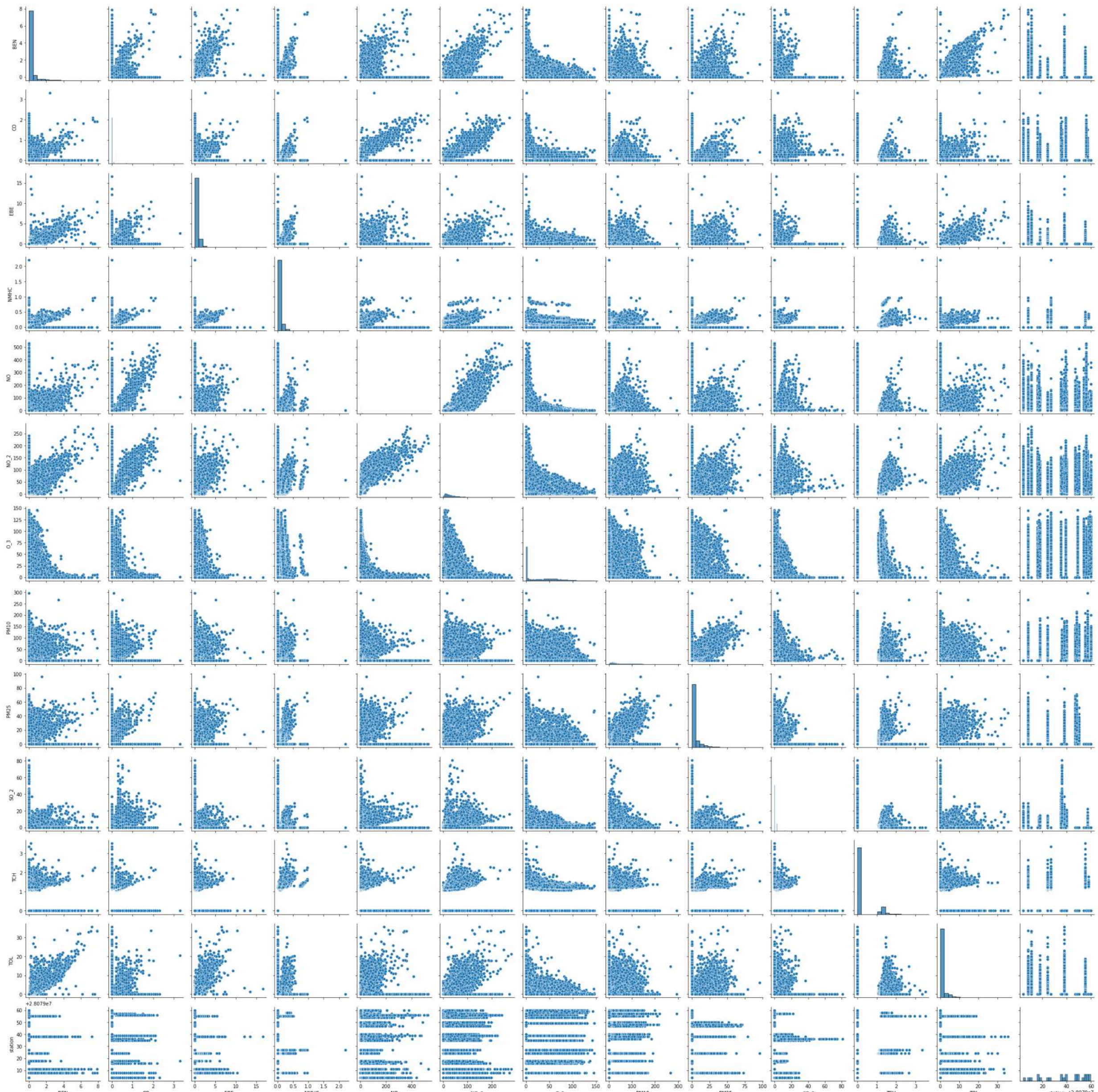
50000 rows × 14 columns

In [6]:
```
df.columns
```

Out[6]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

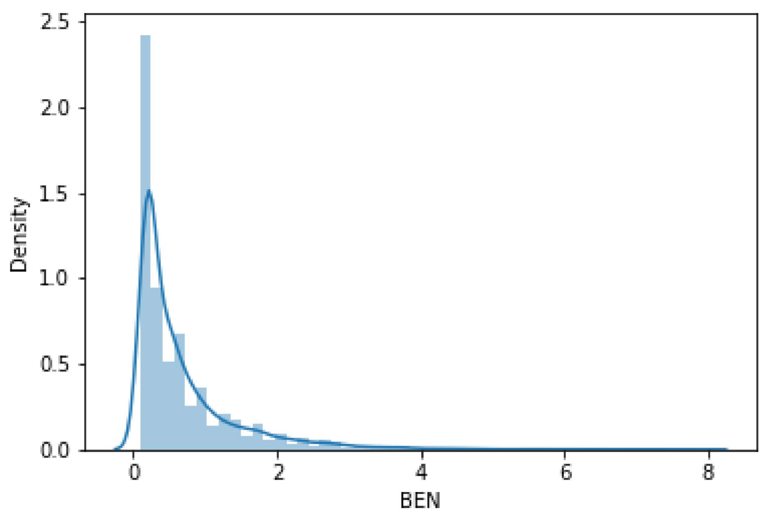
Out[7]: <seaborn.axisgrid.PairGrid at 0x1d115d93220>



In [8]:
```
sns.distplot(data["BEN"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

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



MODEL BUILDING

1.Linear Regression

In [9]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [10]:
```
x=df1[['CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
y=df1[['BEN']]
```

In [11]:
```
#split the dataset into trainning and test
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 [12]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

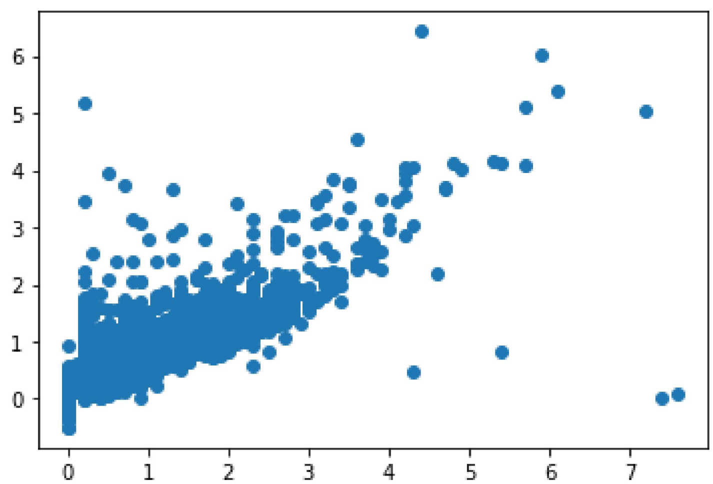
Out[12]:
```
LinearRegression()
```

In [13]:
```
print(lr.intercept_)
```

```
[57987.49167714]
```

In [14]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0x1d12cc75eb0>
```



In [15]:
```
print(lr.score(x_test,y_test))
```

```
0.7537693237620529
```

2.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]:
```
0.7534104823730857
```

3.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]:
```
-2.649297426904334e-05
```

4.ElasticNet 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.          0.          0.          0.00053264  0.00209504 -0.
  0.00123358  0.         -0.          0.          0.04399504 -0.00614486]
```

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

```
[0.05702859 0.03235718 0.10240988 ... 0.08483021 0.00349833 0.07321679]
```

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

```
0.40344157128662583
```

5.Logistic Regression

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

In [27]:
```
feature_matrix = df1.iloc[:,0:14]
target_vector = df1.iloc[:,-1]
```

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

Out[28]:
```
(50000, 11)
```

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

Out[29]:
```
(50000,)
```

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

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

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

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

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

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

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

6.Random Forest

In [ ]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
x=df1[['CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df1['station']
```

In [ ]:
```
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 [ ]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

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

In [ ]:
```
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)
```

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

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

In [ ]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

Results

```
1.Linear regression : 0.8455591055403516

2.Ridge regression : 0.845561600678548

3.Lasso regression : -0.00022865063965538113

4.Elasticnet regression : 0.6309166326143854

5.Logistic regresssion : 0.75178

6.Random forest regression : 0.7791428571428571

Hence Linear regression gives high accuracy for the madrid_2012 model.
```