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_2011.csv")
data1
```

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	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	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	28
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	28
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	28
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	28
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	28
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	28

209928 rows × 14 columns

In [3]:
```
data1.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      51393 non-null   float64
 2   CO       87127 non-null   float64
 3   EBE      51350 non-null   float64
 4   NMHC     43517 non-null   float64
 5   NO       208954 non-null  float64
 6   NO_2     208973 non-null  float64
 7   O_3      122049 non-null  float64
 8   PM10     103743 non-null  float64
 9   PM25     51079 non-null   float64
 10  SO_2     87131 non-null   float64
 11  TCH      43519 non-null   float64
 12  TOL      51175 non-null   float64
 13  station  209928 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ 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	s
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	280
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	280
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	280
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	280
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	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49995	2011-06-26 20:00:00	0.0	0.2	0.0	0.00	4.0	18.0	105.0	0.0	0.0	0.0	0.00	0.0	280
49996	2011-06-26 20:00:00	0.0	0.0	0.0	0.00	1.0	12.0	80.0	0.0	0.0	6.0	0.00	0.0	280
49997	2011-06-26 20:00:00	0.2	0.4	0.3	0.00	3.0	13.0	139.0	36.0	0.0	5.0	0.00	0.2	280
49998	2011-06-26 20:00:00	0.7	0.2	2.0	0.11	4.0	8.0	136.0	0.0	20.0	6.0	1.23	0.5	280
49999	2011-06-26 20:00:00	0.0	0.0	0.0	0.16	3.0	12.0	139.0	0.0	0.0	0.0	1.30	0.0	280

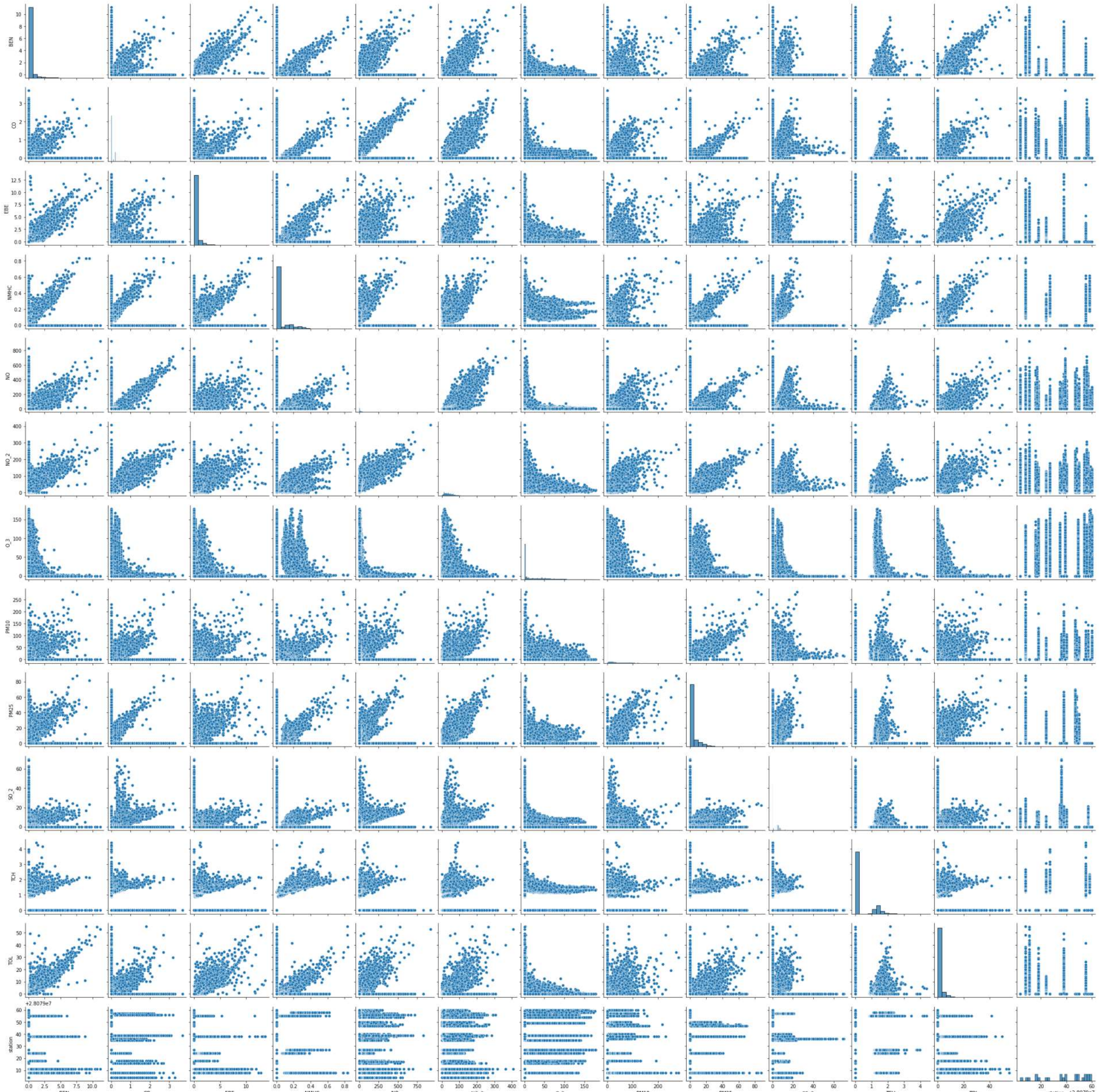
50000 rows × 14 columns

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)
```

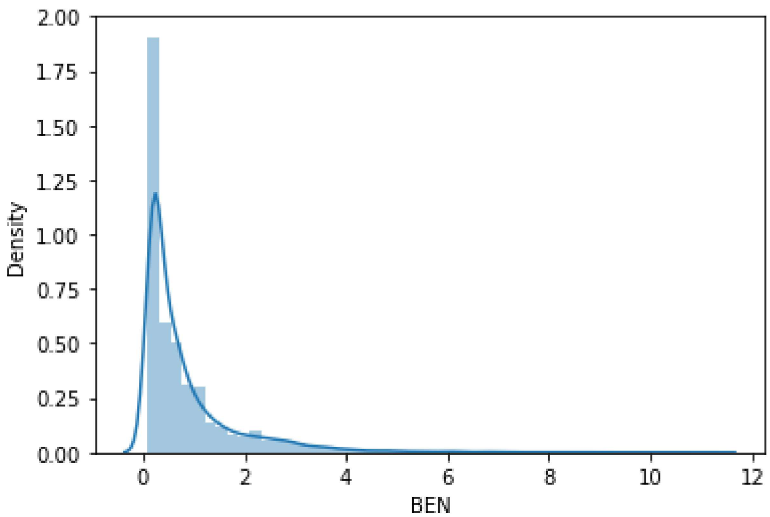
Out[7]: <seaborn.axisgrid.PairGrid at 0x16052c4d970>



In [19]:
```
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[19]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



MODEL BUILDING

1.Linear Regression

In [12]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25', '
```

In [72]:
```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'NOx', 'OXY','PM10', 'PXY', 'SO_2', '
y=df1[['station']]
```

In [73]:
```
#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 [74]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

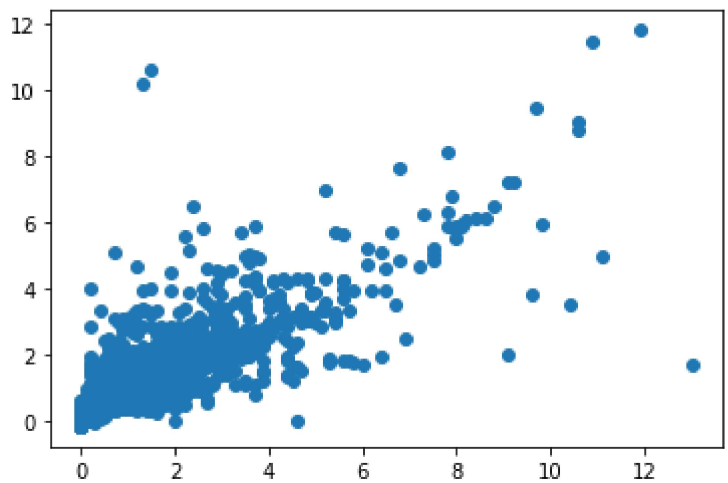
Out[74]: LinearRegression()

In [75]:
```
print(lr.intercept_)
```

93724.06026745134

In [76]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[76]: <matplotlib.collections.PathCollection at 0x1606a0ec880>



In [77]:
```
print(lr.score(x_test,y_test))
```

0.7583322733284314

2.Ridge Regression

In [78]:
```
from sklearn.linear_model import Ridge
```

In [32]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[32]: Ridge(alpha=10)

In [33]:
```
rr.score(x_test,y_test)
```

Out[33]: 0.8279703367815147

3.Lasso Regression

In [34]:
```
from sklearn.linear_model import Lasso
```

In [35]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[35]:
```
Lasso(alpha=10)
```

In [36]:
```
la.score(x_test,y_test)
```

Out[36]:
```
-5.247345148462479e-05
```

4.ElasticNet Regression

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[37]:
```
ElasticNet()
```

In [38]:
```
print(en.coef_)
```

```
[-0.          0.          0.          0.00232461 -0.          0.
 -0.          0.          0.09902706 -0.00470513]
```

In [39]:
```
print(en.predict(x_test))
```

```
[ 0.44389419  0.20324242 -0.02453722 ...  0.72496348 -0.04075361
 -0.06870489]
```

In [40]:
```
print(en.score(x_test,y_test))
```

```
0.6355758199159065
```

5.Logistic Regression

In [41]:
```
from sklearn.linear_model import LogisticRegression
```

In [51]:
```
feature_matrix = df1.iloc[:,0:11]
target_vector = df1.iloc[:,-1]
```

In [52]:
```
feature_matrix.shape
```

Out[52]:
```
(50000, 11)
```

In [53]:
```
target_vector.shape
```

Out[53]:
```
(50000,)
```

In [54]:
```
from sklearn.preprocessing import StandardScaler
```

In [55]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [56]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: 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://scikit-
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-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[56]:
```
LogisticRegression()
```

In [57]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11]]
```

In [58]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079059]
```

In [59]:
```
logr.classes_
```

Out[59]:
```
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 [60]:
```
logr.score(fs,target_vector)
```

Out[60]:
```
0.9882
```

6.Random Forest

In [92]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25','
x=df1[['CO','NMHC', 'NO_2', 'O_3', 'PM10','SO_2', 'TCH', 'TOL']]
y=df1['station']
```

In [93]:
```
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 [94]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[94]:
```
RandomForestClassifier()
```

In [95]:
```
parameters = {'max_depth':[1,2,3,4,5],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]}
```

In [96]:
```
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)
```

Out[96]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [97]:
```
grid_search.best_score_
```

Out[97]:
```
0.7426857142857143
```

In [98]:
```
rfc_best = grid_search.best_estimator_
```

In [99]:
```
from sklearn.tree import plot_tree

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

```
 Text(97.04347826086956, 181.19999999999982, 'gini = 0.647\nsamples = 203\nva
lue = [2, 0, 28, 0, 2, 0, 2, 0, 1, 2, 0, 1, 0, 40\n177, 1, 46, 5, 0, 0, 0, 6,
6, 0]'),
 Text(291.1304347826087, 181.19999999999982, 'gini = 0.666\nsamples = 2589\nv
alue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1473\n1289, 0, 1389, 0, 3, 0,
0, 0, 0, 0]'),
 Text(388.17391304347825, 543.5999999999999, 'gini = 0.0\nsamples = 48\nvalue
= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 64, 0, 0, 0, 0,
0]'),
 Text(776.3478260869565, 906.0, 'PM10 <= 1.0\ngini = 0.751\nsamples = 3627\nv
alue = [1488, 0, 0, 0, 2, 0, 0, 0, 0, 1419, 7, 0, 1456\n0, 0, 0, 0, 0, 0,
1467, 0, 0, 0]'),
 Text(582.2608695652174, 543.5999999999999, 'CO <= 0.25\ngini = 0.004\nsample
s = 905\nvalue = [1488, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0, 0, 0, 0,
0, 0, 0, 0, 0]'),
 Text(485.2173913043478, 181.19999999999982, 'gini = 0.014\nsamples = 164\nva
lue = [277, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(679.304347826087, 181.19999999999982, 'gini = 0.002\nsamples = 741\nval
```

Results

1.Linear regression : 0.7583322733284314

2.Ridge regression : 0.8279703367815147

3.Lasso regression : -5.247345148462479e-05

4.Elasticnet regression : 0.6355758199159065

5.Logistic regresssion : 0.9882

6.Random forest regression : 0.7426857142857143

Hence Logistic regression gives high accuracy for the madrid_2011 model.