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_2015.csv")
data1
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	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	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	280
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	280
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	280
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	280
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	280
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	280
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	280

210096 rows × 14 columns

In [3]:
```
data1.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      51039 non-null   float64
 2   CO       86827 non-null   float64
 3   EBE      50962 non-null   float64
 4   NMHC     25756 non-null   float64
 5   NO       208805 non-null  float64
 6   NO_2     208805 non-null  float64
 7   O_3      121574 non-null  float64
 8   PM10     102745 non-null  float64
 9   PM25     48798 non-null   float64
 10  SO_2     86898 non-null   float64
 11  TCH      25756 non-null   float64
 12  TOL      50626 non-null   float64
 13  station  210096 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	sta
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	2807!
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	2807!
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	2807
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	2807!
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	2807!
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49995	2015-08-26 20:00:00	0.0	0.2	0.0	0.00	1.0	22.0	107.0	0.0	0.0	0.0	0.00	0.0	2807!
49996	2015-08-26 20:00:00	0.0	0.0	0.0	0.00	4.0	23.0	100.0	0.0	0.0	4.0	0.00	0.0	2807!
49997	2015-08-26 20:00:00	0.2	0.2	0.1	0.00	1.0	24.0	96.0	20.0	0.0	2.0	0.00	0.7	2807!
49998	2015-08-26 20:00:00	0.1	0.2	0.1	0.06	1.0	3.0	118.0	20.0	8.0	2.0	1.20	0.1	2807!
49999	2015-08-26 20:00:00	0.0	0.0	0.0	0.00	2.0	14.0	105.0	0.0	0.0	0.0	0.00	0.0	2807!

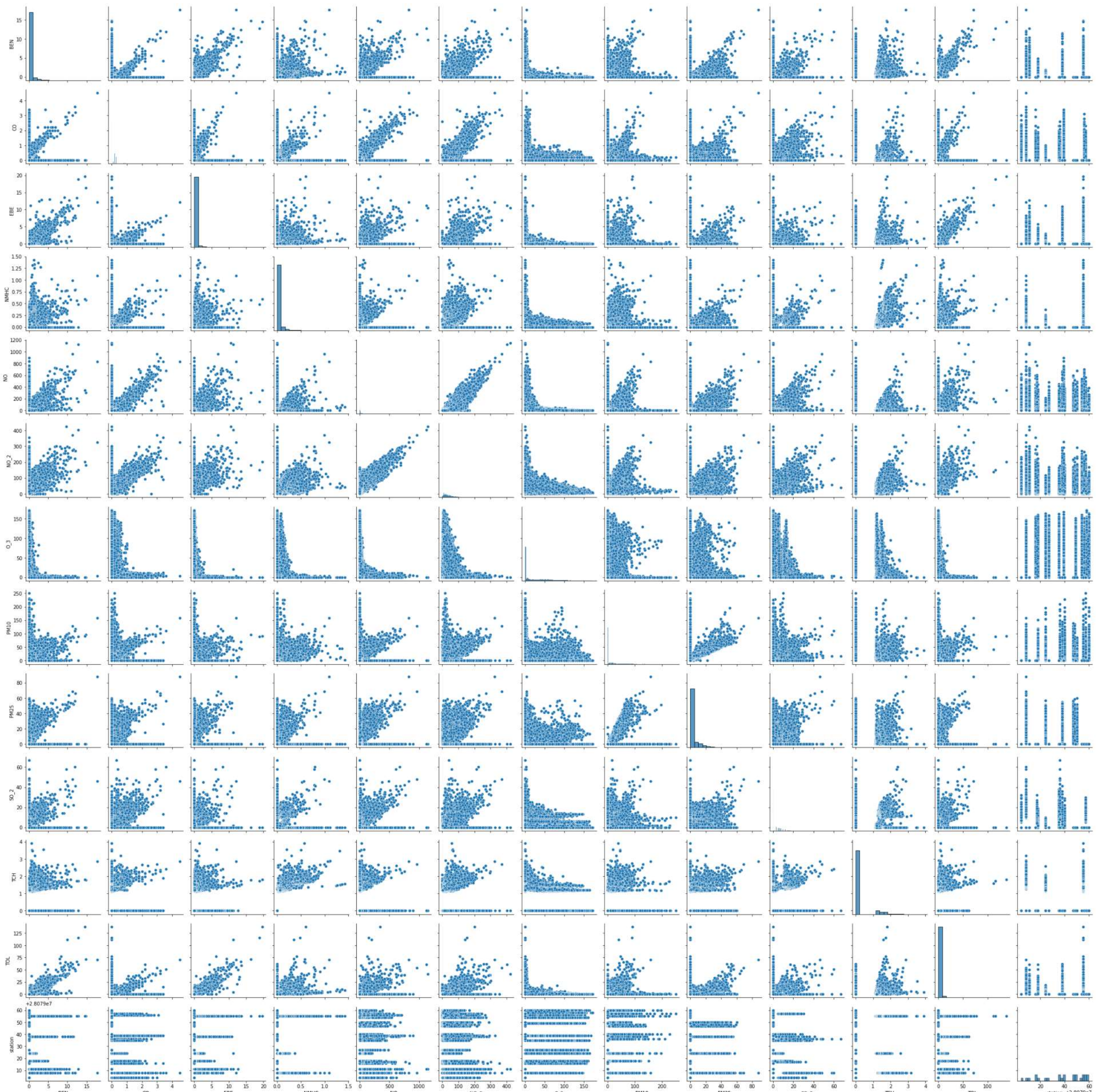
50000 rows × 14 columns

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)
```

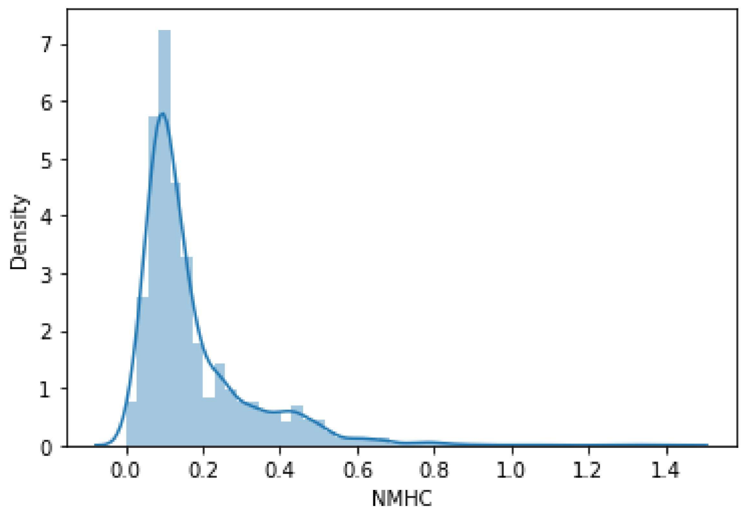
Out[7]: <seaborn.axisgrid.PairGrid at 0x2360c2d7c70>



In [8]:
```
sns.distplot(data['NMHC'])
```

```
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='NMHC', ylabel='Density'>
```



MODEL BUILDING

1.Linear Regression

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

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

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)
```

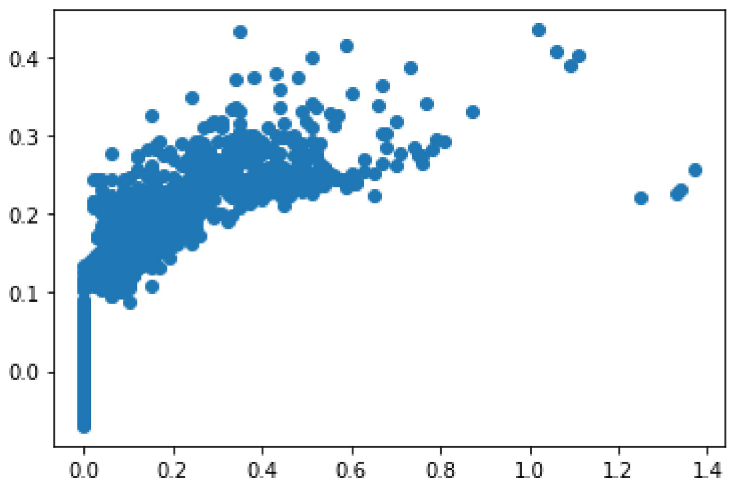
Out[12]: LinearRegression()

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

```
[-14236.08851565]
```

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

Out[14]: <matplotlib.collections.PathCollection at 0x23622de4be0>



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

```
0.7026892511305912
```

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]:
```
0.7026632140767994
```

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]:
```
-7.728655678551632e-06
```

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.  0. -0.  0.  0.  0.  0.  0. -0.]
```

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

```
[0.02121829 0.02121829 0.02121829 ... 0.02121829 0.02121829 0.02121829]
```

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

```
-7.728655678551632e-06
```

5.Logistic Regression

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

In [27]:
```
feature_matrix = df1.iloc[:,0:11]
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 [32]:
```
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[32]: LogisticRegression()

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

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

```
[28079008]
```

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

Out[35]:
```
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 [36]:
```
logr.score(fs,target_vector)
```

Out[36]: 0.7093

6.Random Forest

In [37]:
```
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 [38]:
```
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 [39]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[39]:
```
RandomForestClassifier()
```

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

In [41]:
```
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[41]:
```
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 [42]:
```
grid_search.best_score_
```

Out[42]:
```
0.6930000000000001
```

In [43]:
```
rfc_best = grid_search.best_estimator_
```

In [45]:
```
from sklearn.tree import plot_tree

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

Out[45]:
```
[Text(2155.909090909091, 1993.2, 'SO_2 <= 0.5\ngini = 0.958\nsamples = 22041
\nvalue = [1426, 1462, 1485, 1443, 1471, 1471, 1438, 1469, 1458\n1447, 1458,
1409, 1471, 1512, 1447, 1527, 1526, 1391\n1420, 1445, 1452, 1423, 1479, 147
0]'),
 Text(1192.090909090909, 1630.8000000000002, 'CO <= 0.05\ngini = 0.929\nsampl
es = 12926\nvalue = [5, 16, 1485, 1443, 7, 4, 1, 1469, 59, 15, 3\n1409, 7, 15
12, 1447, 1527, 1526, 1391, 1420, 1445, 2\n1423, 1479, 1470]'),
 Text(760.9090909090909, 1268.4, 'O_3 <= 0.5\ngini = 0.91\nsamples = 10202\nv
alue = [4, 14, 1485, 10, 7, 0, 1, 1469, 2, 15, 3, 28\n7, 1512, 1447, 1527, 15
26, 1391, 1420, 8, 1, 1423\n1479, 1470]'),
 Text(405.8181818181818, 906.0, 'PM10 <= 0.5\ngini = 0.807\nsamples = 4715\nv
alue = [4, 14, 1485, 6, 7, 0, 1, 3, 2, 15, 3, 25, 7\n1512, 1447, 1, 1526, 3,
1420, 3, 1, 18, 9, 12]'),
 Text(202.9090909090909, 543.5999999999999, 'TOL <= 0.2\ngini = 0.201\nsample
s = 1055\nvalue = [4, 14, 1485, 6, 7, 0, 0, 3, 2, 11, 0, 25, 0\n15, 21, 1, 3
0, 3, 4, 3, 0, 18, 9, 1]'),
 Text(101.45454545454545, 181.19999999999982, 'gini = 0.896\nsamples = 132\nv
alue = [4, 12, 42, 6, 7, 0, 0, 3, 2, 11, 0, 25, 0\n15, 21, 1, 30, 3, 1, 3, 0,
18, 9, 1]'),
```

Results

1.Linear regression : 0.7026892511305912

2.Ridge regression : 0.7026632140767994

3.Lasso regression : -7.728655678551632e-06

4.Elasticnet regression : -7.728655678551632e-06

5.Logistic regresssion : 0.7093

6.Random forest regression : 0.6930000000000001

Hence Logistic regression gives high accuracy for the madrid_2013 model.