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_2013.csv")
data1
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	28
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28

209880 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      50462 non-null   float64
 2   CO       87018 non-null   float64
 3   EBE      50463 non-null   float64
 4   NMHC     25935 non-null   float64
 5   NO       209108 non-null  float64
 6   NO_2     209108 non-null  float64
 7   O_3      121858 non-null  float64
 8   PM10     104339 non-null  float64
 9   PM25     51980 non-null   float64
 10  SO_2     86970 non-null   float64
 11  TCH      25935 non-null   float64
 12  TOL      50317 non-null   float64
 13  station  209880 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	2013-11-01 01:00:00	0.0	0.6	0.0	0.00	135.0	74.0	0.0	0.0	0.0	7.0	0.00	0.0	2807
1	2013-11-01 01:00:00	1.5	0.5	1.3	0.00	71.0	83.0	2.0	23.0	16.0	12.0	0.00	8.3	2807
2	2013-11-01 01:00:00	3.9	0.0	2.8	0.00	49.0	70.0	0.0	0.0	0.0	0.0	0.00	9.0	2807
3	2013-11-01 01:00:00	0.0	0.5	0.0	0.00	82.0	87.0	3.0	0.0	0.0	0.0	0.00	0.0	2807
4	2013-11-01 01:00:00	0.0	0.0	0.0	0.00	242.0	111.0	2.0	0.0	0.0	12.0	0.00	0.0	2807
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49995	2013-04-26 20:00:00	0.0	0.3	0.0	0.00	5.0	20.0	75.0	0.0	0.0	0.0	0.00	0.0	2807
49996	2013-04-26 20:00:00	0.0	0.0	0.0	0.00	1.0	21.0	81.0	0.0	0.0	2.0	0.00	0.0	2807
49997	2013-04-26 20:00:00	0.1	0.3	1.0	0.00	2.0	30.0	79.0	46.0	0.0	3.0	0.00	0.5	2807
49998	2013-04-26 20:00:00	0.4	0.2	0.7	0.24	1.0	16.0	96.0	16.0	7.0	2.0	1.31	0.9	2807
49999	2013-04-26 20:00:00	0.0	0.0	0.0	0.10	1.0	7.0	97.0	0.0	0.0	0.0	1.17	0.0	2807

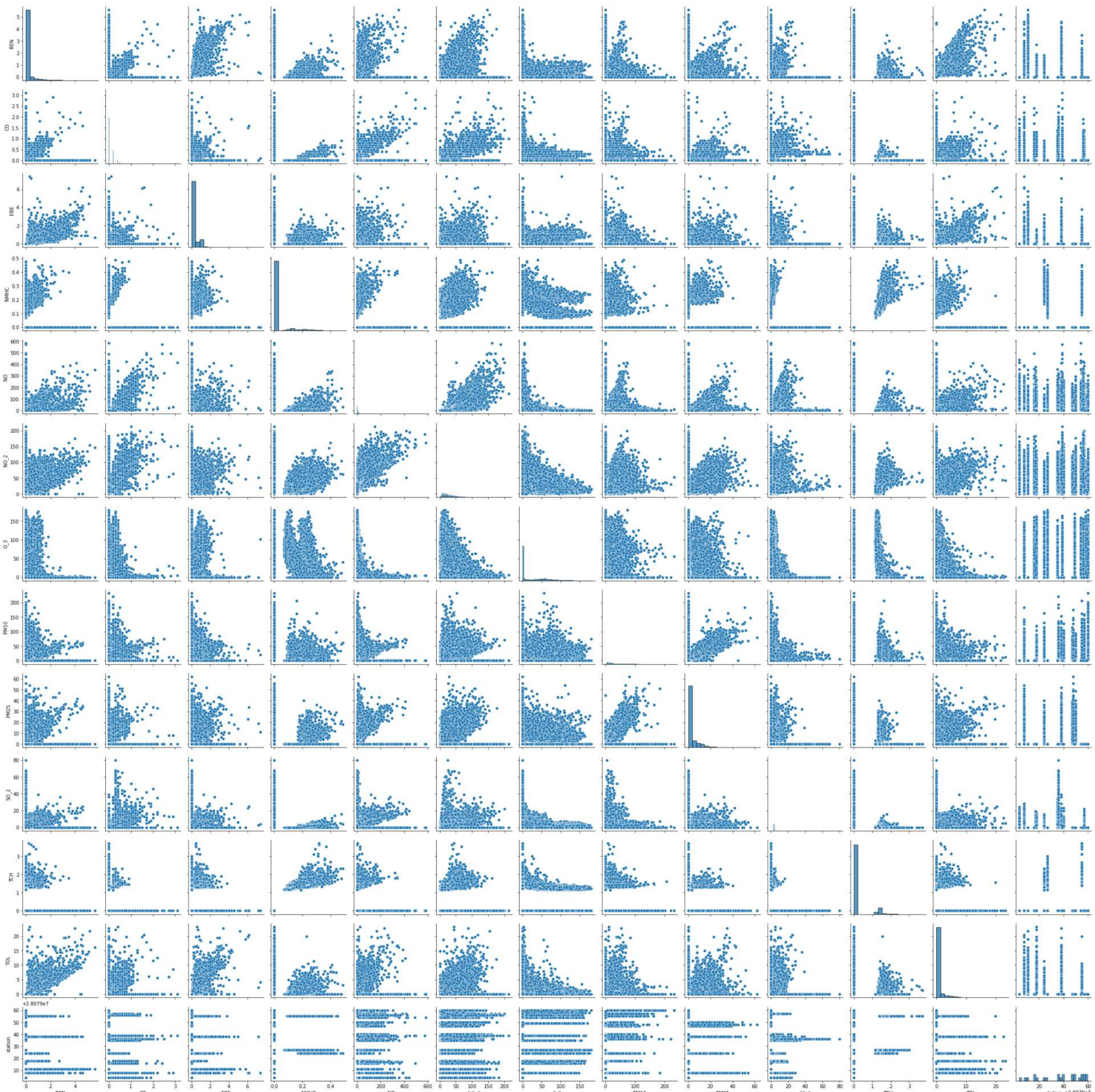
50000 rows × 14 columns

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)
```

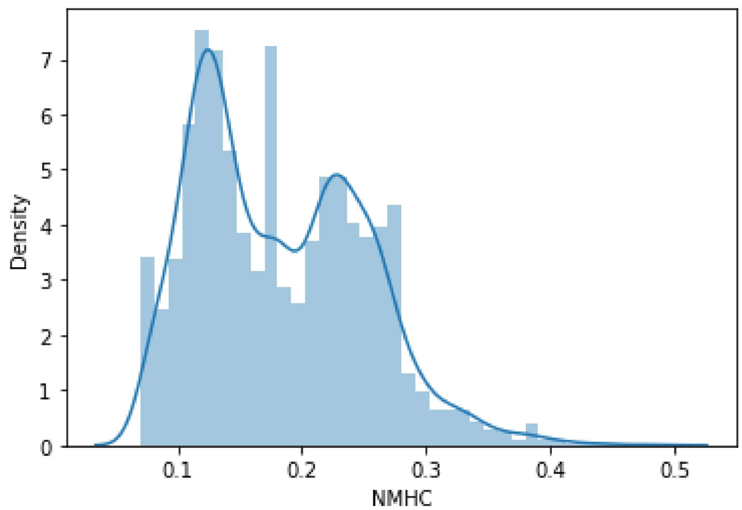
Out[7]: <seaborn.axisgrid.PairGrid at 0x1b48bca8c70>



In [9]:
```
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[9]:
```
<AxesSubplot:xlabel='NMHC', ylabel='Density'>
```



MODEL BUILDING

1.Linear Regression

In [11]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25', 'SO_2',
```

In [18]:
```
x=df1[[ 'BEN', 'CO', 'EBE',  'NO', 'NO_2', 'O_3', 'PM10', 'PM25', 'SO_2', 'TCH',
y=df1[['NMHC']]
```

In [19]:
```
#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 [20]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

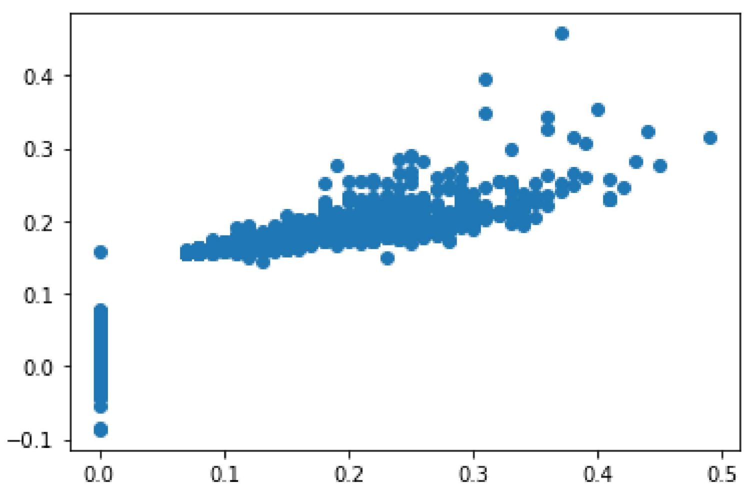
Out[20]:
```
LinearRegression()
```

In [21]:
```
print(lr.intercept_)
```

```
[2988.53432101]
```

In [22]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x1b4a3054130>
```



In [23]:
```
print(lr.score(x_test,y_test))
```

```
0.9052008873644165
```

2.Ridge Regression

In [24]:
```
from sklearn.linear_model import Ridge
```

In [25]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[25]:
```
Ridge(alpha=10)
```

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

Out[26]: 0.9052143974747214

3.Lasso Regression

In [27]:
```
from sklearn.linear_model import Lasso
```

In [28]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[28]: Lasso(alpha=10)

In [29]:
```
la.score(x_test,y_test)
```

Out[29]: -0.0003581314365290744

4.ElasticNet Regression

In [30]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[30]: ElasticNet()

In [31]:
```
print(en.coef_)
```

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

In [32]:
```
print(en.predict(x_test))
```

```
[0.02220114 0.02220114 0.02220114 ... 0.02220114 0.02220114 0.02220114]
```

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

```
-0.0003581314365290744
```

5.Logistic Regression

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

In [35]:
```
feature_matrix = df1.iloc[:,0:11]
target_vector = df1.iloc[:,-1]
```

In [36]:
```
feature_matrix.shape
```

Out[36]:
```
(50000, 11)
```

In [37]:
```
target_vector.shape
```

Out[37]:
```
(50000,)
```

In [38]:
```
from sklearn.preprocessing import StandardScaler
```

In [39]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [40]:
```
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[40]:
```
LogisticRegression()
```

In [41]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11]]
```

In [42]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [43]:
```
logr.classes_
```

Out[43]:
```
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 [44]:
```
logr.score(fs,target_vector)
```

Out[44]:
```
0.7341
```

6.Random Forest

In [45]:
```
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 [46]:
```
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 [47]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[47]:
```
RandomForestClassifier()
```

In [48]:
```
parameters = {'max_depth':[1,2,3,4,5],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]}
```

In [49]:
```
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[49]:
```
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 [50]:
```
grid_search.best_score_
```

Out[50]:
```
0.7119428571428571
```

In [51]:
```
rfc_best = grid_search.best_estimator_
```

In [52]:
```
from s klearn.tree import plot_tree

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

Out[52]:
```
[Text(2245.95, 1993.2, 'SO_2 <= 0.5\ngini = 0.958\nsamples = 22117\nvalue =
[1424, 1460, 1501, 1451, 1446, 1508, 1413, 1484, 1482\n1481, 1506, 1423, 147
1, 1433, 1461, 1423, 1461, 1492\n1505, 1497, 1451, 1431, 1404, 1392]'),
 Text(1088.1, 1630.8000000000002, 'TCH <= 0.575\ngini = 0.929\nsamples = 1295
6\nvalue = [7, 17, 1501, 1451, 4, 4, 5, 1484, 5, 1, 1, 1423\n1, 1433, 1461, 1
423, 1461, 1492, 1505, 1497, 0, 1431\n1404, 1392]'),
 Text(613.8, 1268.4, 'O_3 <= 0.5\ngini = 0.917\nsamples = 11124\nvalue = [7,
17, 1501, 1451, 4, 4, 0, 3, 5, 1, 1, 1423\n1, 1433, 1461, 1423, 1461, 1492, 2
6, 1497, 0, 1431\n1404, 1392]'),
 Text(334.79999999999995, 906.0, 'NO_2 <= 28.5\ngini = 0.757\nsamples = 3759
\nvalue = [7, 11, 1501, 3, 0, 0, 0, 2, 0, 1, 1, 4, 1\n1433, 1461, 5, 1461, 7,
26, 6, 0, 3, 2, 2]'),
 Text(223.2, 543.5999999999999, 'PM10 <= 1.0\ngini = 0.754\nsamples = 1973\nv
alue = [4, 11, 742, 1, 0, 0, 0, 2, 0, 0, 1, 4, 0\n900, 811, 4, 568, 7, 17, 4,
0, 3, 2, 2]'),
 Text(111.6, 181.19999999999982, 'gini = 0.126\nsamples = 502\nvalue = [4, 1
1, 742, 1, 0, 0, 0, 2, 0, 0, 1, 4, 0, 3\n6, 4, 0, 7, 0, 4, 0, 3, 2, 0]'),
 Text(334.79999999999995, 181.19999999999982, 'gini = 0.661\nsamples = 1471\n
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 897\n805, 0, 568, 0, 17, 0,
0, 0, 0, 2]'),
```

Results

1.Linear regression : 0.9052008873644165

2.Ridge regression : 0.9052143974747214

3.Lasso regression : -0.0003581314365290744

4.Elasticnet regression : -0.0003581314365290744

5.Logistic regresssion : 0.7341

6.Random forest regression : 0.7119428571428571

Hence Linear regression gives high accuracy for the madrid_2013 model.