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_2014.csv")
data1
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
0	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	280
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	280
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	280
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	280
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210019	2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN	280
210020	2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN	280
210021	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN	280
210022	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN	280
210023	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN	280

210024 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210024 entries, 0 to 210023
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210024 non-null  object 
 1   BEN      46703 non-null   float64
 2   CO       87023 non-null   float64
 3   EBE      46722 non-null   float64
 4   NMHC     25021 non-null   float64
 5   NO       209154 non-null  float64
 6   NO_2     209154 non-null  float64
 7   O_3      121681 non-null  float64
 8   PM10     104311 non-null  float64
 9   PM25     51954 non-null   float64
 10  SO_2     87141 non-null   float64
 11  TCH      25021 non-null   float64
 12  TOL      46570 non-null   float64
 13  station  210024 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	stati
0	2014-06-01 01:00:00	0.0	0.2	0.0	0.00	3.0	10.0	0.0	0.0	0.0	3.0	0.00	0.0	280790
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	280790
2	2014-06-01 01:00:00	0.3	0.0	0.1	0.00	2.0	6.0	0.0	0.0	0.0	0.0	0.00	1.1	280790
3	2014-06-01 01:00:00	0.0	0.2	0.0	0.00	1.0	6.0	79.0	0.0	0.0	0.0	0.00	0.0	280790
4	2014-06-01 01:00:00	0.0	0.0	0.0	0.00	1.0	6.0	75.0	0.0	0.0	4.0	0.00	0.0	280790
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49995	2014-04-27 20:00:00	0.0	0.2	0.0	0.00	2.0	15.0	80.0	0.0	0.0	0.0	0.00	0.0	280790
49996	2014-04-27 20:00:00	0.0	0.0	0.0	0.00	3.0	15.0	72.0	0.0	0.0	4.0	0.00	0.0	280790
49997	2014-04-27 20:00:00	0.1	0.5	0.1	0.00	1.0	12.0	99.0	14.0	0.0	1.0	0.00	0.4	280790
49998	2014-04-27 20:00:00	0.1	0.2	0.1	0.25	1.0	1.0	92.0	22.0	7.0	3.0	1.30	0.1	280790
49999	2014-04-27 20:00:00	0.0	0.0	0.0	0.00	1.0	7.0	89.0	0.0	0.0	0.0	0.00	0.0	280790

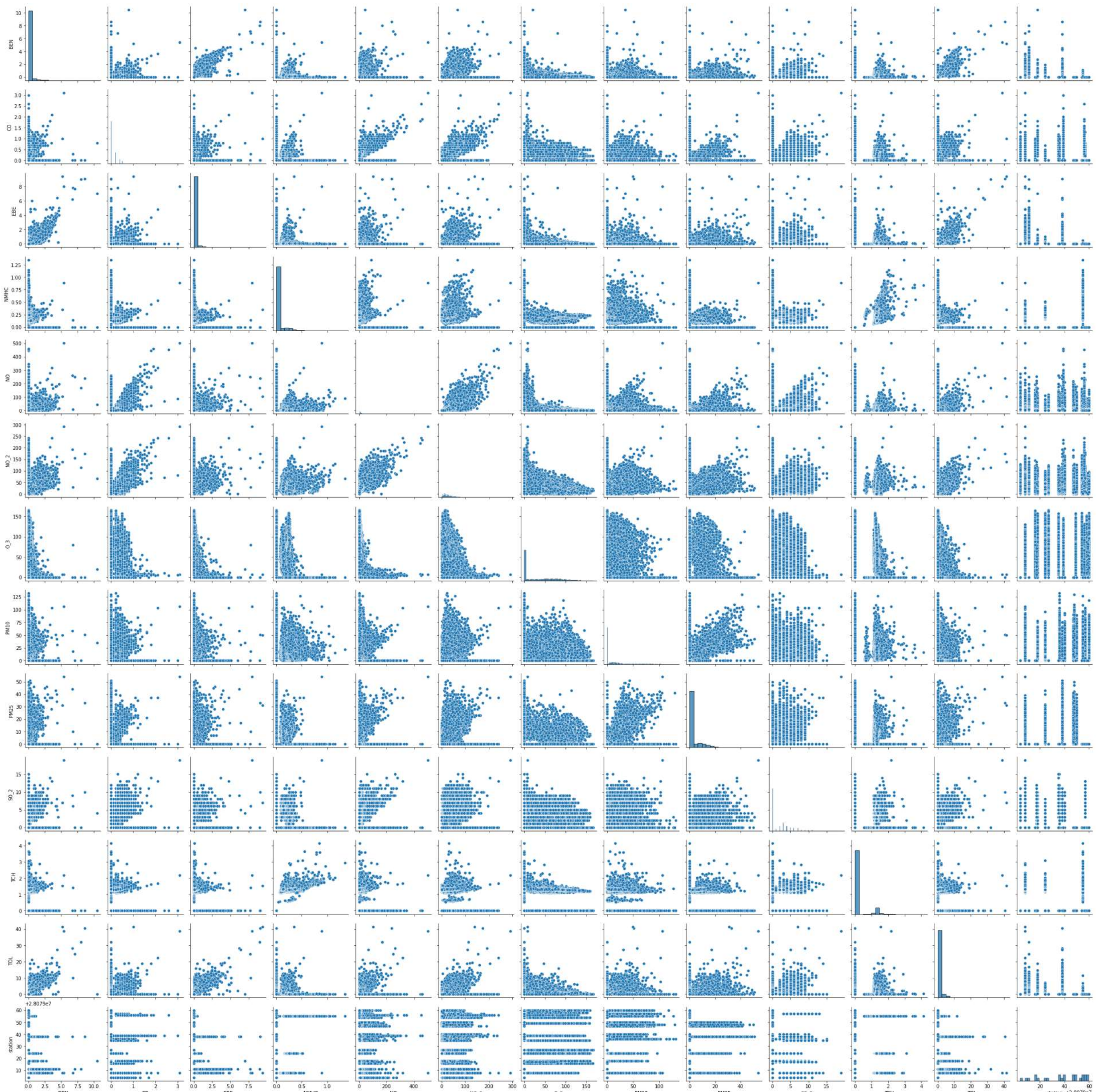
50000 rows × 14 columns

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)
```

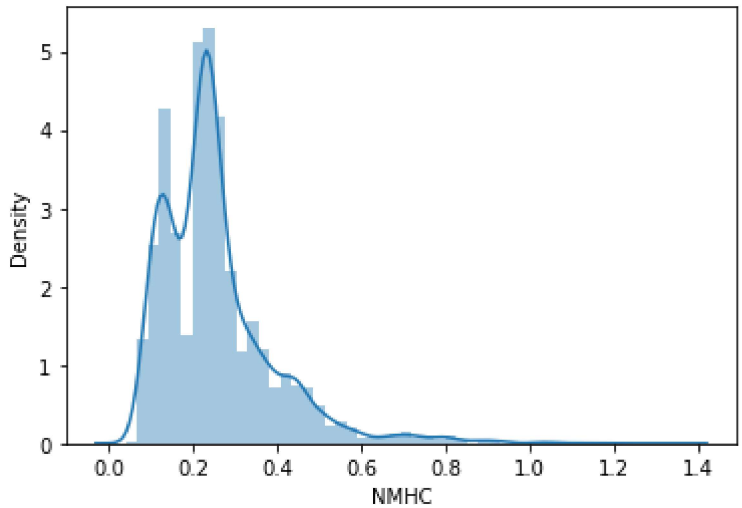
Out[7]: <seaborn.axisgrid.PairGrid at 0x1ae2f3e2220>



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]:
```
, 'EBE',  'NO', 'NO_2', 'O_3', 'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station']]
```

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)
```

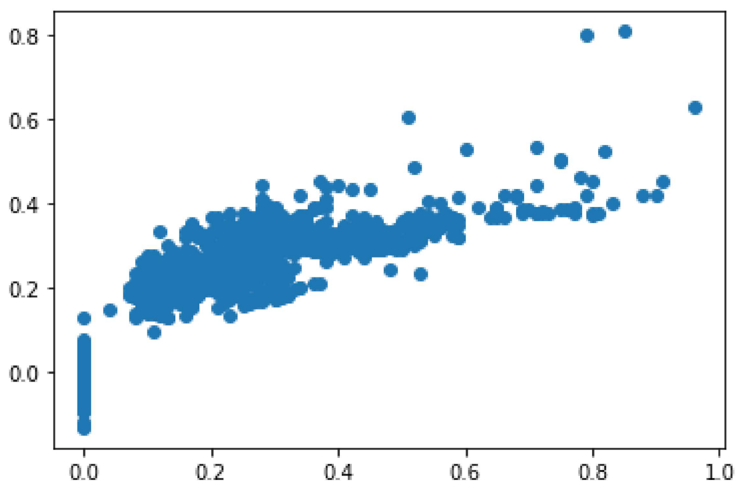
Out[12]:
```
LinearRegression()
```

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

```
[-10490.71843754]
```

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

Out[14]:
```
<matplotlib.collections.PathCollection at 0x1ae45c7f370>
```



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

```
0.8493682843504402
```

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]:
```
0.8494346624250667
```

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]:
```
-0.00013712985531388888
```

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))
```

```
[0.03157543 0.03157543 0.03157543 ... 0.03157543 0.03157543 0.03157543]
```

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

```
-0.00013712985531388888
```

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, 13)

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 [35]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13]]
```

In [36]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079059]
```

In [37]:
```
logr.classes_
```

Out[37]:
```
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 [38]:
```
logr.score(fs,target_vector)
```

Out[38]: 0.97354

6.Random Forest

In [39]:
```
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 [40]:
```
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 [41]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[41]:
```
RandomForestClassifier()
```

In [42]:
```
parameters = {'max_depth':[1,2,3,4,5],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]}
```

In [43]:
```
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[43]:
```
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 [44]:
```
grid_search.best_score_
```

Out[44]:
```
0.7221142857142857
```

In [45]:
```
rfc_best = grid_search.best_estimator_
```

In [47]:
```python
from sklearn.tree import plot_tree

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

```
20, 0, 0, 0, 1431, 24, 0, 0, 1434, 1449, 0\n1432, 0, 0, 0, 0, 0, 0, 0, 1416,
0, 0, 0]'),
 Text(1562.3999999999999, 543.5999999999999, 'CO <= 0.05\ngini = 0.78\nsample

s = 1556\nvalue = [0, 7, 0, 0, 0, 642, 11, 0, 0, 521, 314, 0, 262\n0, 0, 0,
0, 0, 0, 0, 662, 0, 0, 0]'),
 Text(1450.8, 181.19999999999982, 'gini = 0.505\nsamples = 377\nvalue = [0,
0, 0, 0, 0, 4, 0, 0, 0, 0, 314, 0, 262, 0\n0, 0, 0, 0, 0, 0, 1, 0, 0, 0]'),
 Text(1674.0, 181.19999999999982, 'gini = 0.67\nsamples = 1179\nvalue = [0,
7, 0, 0, 0, 638, 11, 0, 0, 521, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 661, 0, 0,
0]'),
 Text(2008.8, 543.5999999999999, 'TOL <= 0.25\ngini = 0.795\nsamples = 3040\n
value = [0, 13, 0, 0, 0, 789, 13, 0, 0, 933, 1135, 0\n1190, 0, 0, 0, 0, 0, 0,
0, 754, 0, 0, 0]'),
 Text(1897.1999999999998, 181.19999999999982, 'gini = 0.674\nsamples = 1869\n
value = [0, 0, 0, 0, 0, 22, 7, 0, 0, 933, 55, 0, 1190\n0, 0, 0, 0, 0, 0, 7
54, 0, 0, 0]'),
 Text(2120.4, 181.19999999999982, 'gini = 0.496\nsamples = 1171\nvalue = [0,
13, 0, 0, 0, 767, 6, 0, 0, 0, 1080, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2901.6, 1630.8000000000002, 'O_3 <= 2.0\ngini = 0.667\nsamples = 2759\n
value = [0, 1400, 0, 0, 0, 0, 1419, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1458, 0
```

Results

```
1.Linear regression :0.8493682843504402

2.Ridge regression : 0.8494346624250667

3.Lasso regression : -0.00013712985531388888

4.Elasticnet regression : -0.00013712985531388888

5.Logistic regresssion : 0.97354

6.Random forest regression : 0.7221142857142857

Hence Logistic regression gives high accuracy for the madrid_2013 model.
```