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_2018.csv")
data1
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TC
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	Na
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.4
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	Na
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	Na
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	Na
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	Na
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	Na
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	Na
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	Na

69096 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69096 entries, 0 to 69095
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     69096 non-null  object 
 1   BEN      16950 non-null  float64
 2   CH4      8440 non-null   float64
 3   CO       28598 non-null  float64
 4   EBE      16949 non-null  float64
 5   NMHC     8440 non-null   float64
 6   NO       68826 non-null  float64
 7   NO_2     68826 non-null  float64
 8   NOx      68826 non-null  float64
 9   O_3      40049 non-null  float64
 10  PM10     36911 non-null  float64
 11  PM25     18912 non-null  float64
 12  SO_2     28586 non-null  float64
 13  TCH      8440 non-null   float64
 14  TOL      16950 non-null  float64
 15  station  69096 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 8.4+ MB
```

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

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

Out[5]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH
0	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	29.0	31.0	0.0	0.0	0.0	2.0	0.0(
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.4'
2	2018-03-01 01:00:00	0.4	0.00	0.0	0.2	0.00	4.0	41.0	47.0	0.0	0.0	0.0	0.0	0.0(
3	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	35.0	37.0	54.0	0.0	0.0	0.0	0.0(
4	2018-03-01 01:00:00	0.0	0.00	0.0	0.0	0.00	1.0	27.0	29.0	49.0	0.0	0.0	3.0	0.0(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
49995	2018-02-26 21:00:00	0.0	0.00	0.6	0.0	0.00	18.0	109.0	137.0	11.0	0.0	0.0	0.0	0.0(
49996	2018-02-26 21:00:00	0.0	0.00	0.0	0.0	0.00	171.0	160.0	422.0	1.0	0.0	0.0	12.0	0.0(
49997	2018-02-26 21:00:00	0.8	0.00	0.8	0.7	0.00	19.0	109.0	138.0	10.0	25.0	0.0	7.0	0.0(
49998	2018-02-26 21:00:00	0.7	1.10	0.3	0.4	0.09	1.0	81.0	82.0	20.0	20.0	11.0	4.0	1.1(
49999	2018-02-26 21:00:00	0.0	0.00	0.0	0.0	0.00	30.0	113.0	160.0	8.0	0.0	0.0	0.0	0.0(

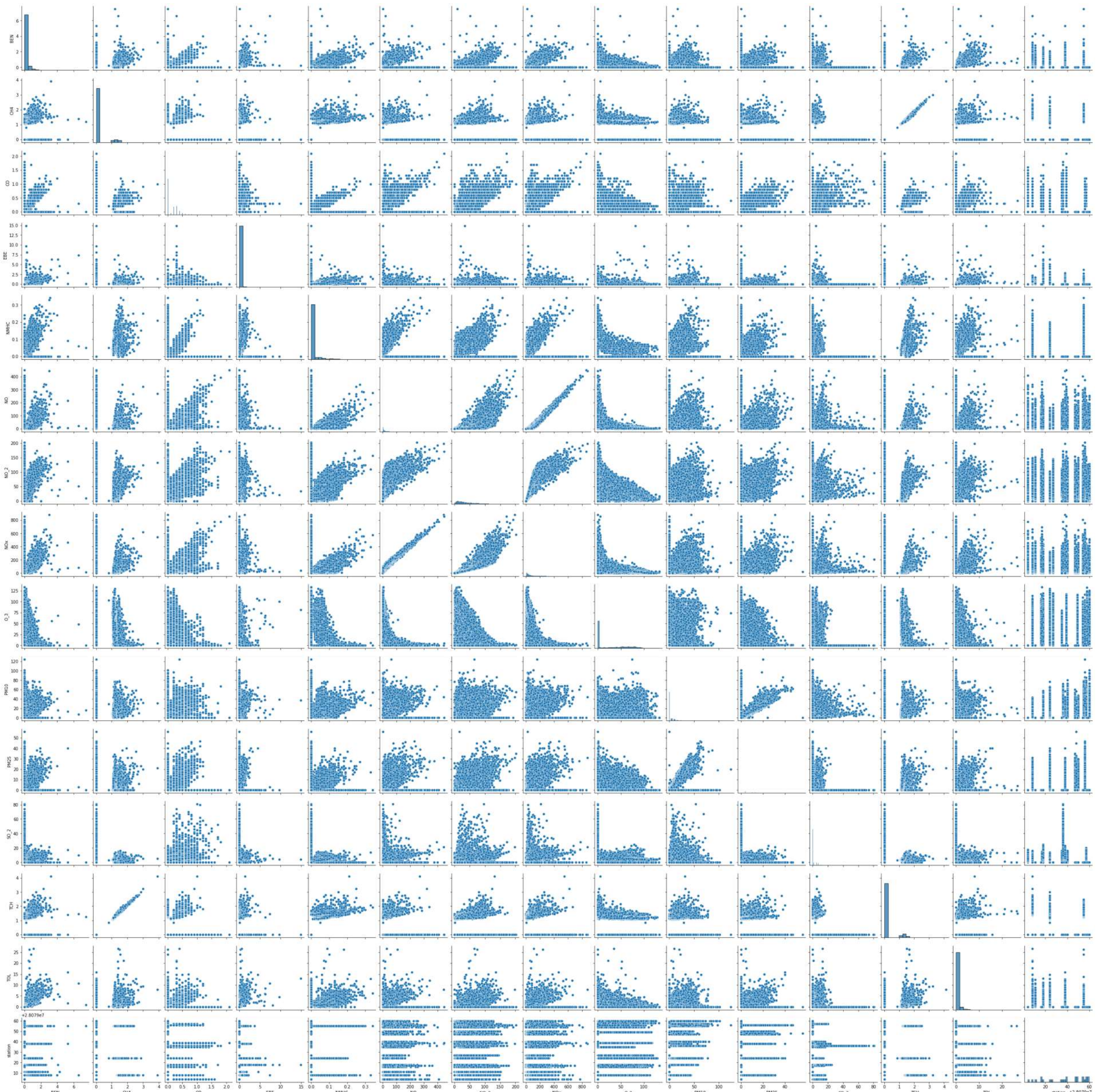
50000 rows × 16 columns

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

Out[6]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

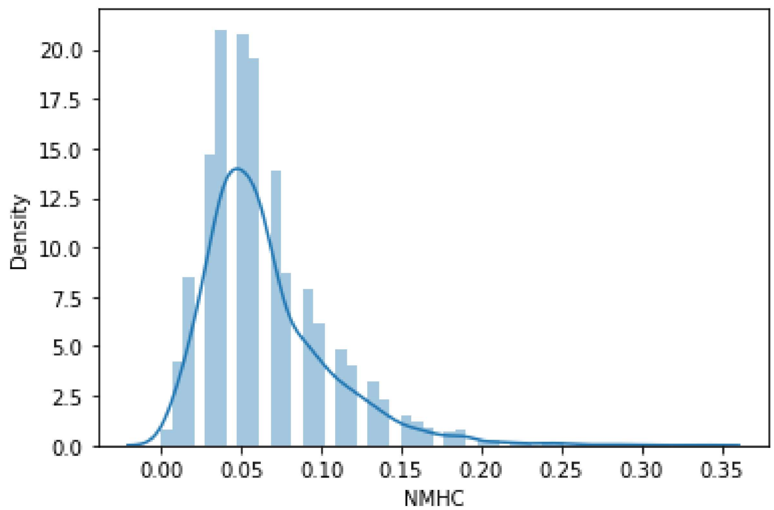
Out[7]: <seaborn.axisgrid.PairGrid at 0x21c0bfdf310>



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)
```

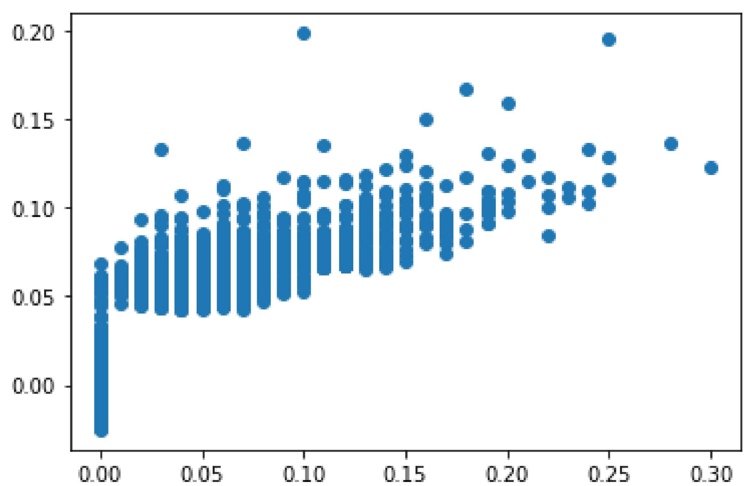
Out[12]:
```
LinearRegression()
```

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

```
[-5546.12509531]
```

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

Out[14]:
```
<matplotlib.collections.PathCollection at 0x21c28772c40>
```



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

```
0.7850736800457832
```

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]:
```
0.7851046395472827
```

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]:
```
-2.5541386561300783e-08
```

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))
```
```
[0.00810543 0.00810543 0.00810543 ... 0.00810543 0.00810543 0.00810543]
```

In [25]:
```
print(en.score(x_test,y_test))
```
```
-2.5541386561300783e-08
```

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]:
```
0.73
```

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]:
```
0.7129714285714286
```

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

In [44]:
```
from sklearn.tree import plot_tree

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

```
0, 0, 0, 0, 0, 0, 0, 172, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3766.5, 1268.4, 'SO_2 <= 0.5\ngini = 0.857\nsamples = 6334\nvalue = [0,
1439, 0, 1461, 0, 1366, 1463, 0, 1411, 0, 0\n1501, 0, 0, 0, 0, 0, 0, 0, 1391,
0, 0, 0, 0]'),
 Text(3627.0, 906.0, 'gini = 0.668\nsamples = 2758\nvalue = [0, 9, 0, 1461,
0, 1, 0, 0, 4, 0, 0, 1501, 0\n0, 0, 0, 0, 0, 0, 1391, 0, 0, 0, 0]'),
 Text(3906.0, 906.0, 'TOL <= 0.05\ngini = 0.75\nsamples = 3576\nvalue = [0, 1
430, 0, 0, 0, 1365, 1463, 0, 1407, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(3627.0, 543.5999999999999, 'O_3 <= 69.5\ngini = 0.06\nsamples = 930\nva
lue = [0, 7, 0, 0, 0, 33, 5, 0, 1407, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 0]'),
 Text(3487.5, 181.19999999999982, 'gini = 0.044\nsamples = 910\nvalue = [0,
7, 0, 0, 0, 24, 1, 0, 1390, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3766.5, 181.19999999999982, 'gini = 0.571\nsamples = 20\nvalue = [0, 0,
0, 0, 0, 9, 4, 0, 17, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(4185.0, 543.5999999999999, 'TOL <= 0.25\ngini = 0.666\nsamples = 2646\n
value = [0, 1423, 0, 0, 0, 1332, 1458, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0]'),
 Text(4045.5, 181.19999999999982, 'gini = 0.311\nsamples = 513\nvalue = [0,
```

Results

1.Linear regression : 0.7850736800457832

2.Ridge regression :0.7851046395472827

3.Lasso regression : -2.5541386561300783e-08

4.Elasticnet regression : -2.5541386561300783e-08

5.Logistic regresssion : 0.73

6.Random forest regression : 0.7129714285714286

Hence Ridge regression gives high accuracy for the madrid_2013 model.