

Importing Libraries

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

```
In [2]: df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs(Dataset)\ma
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570000
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.410000
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260000
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180000
...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120000
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.460000
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680000
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360000
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490000

230568 rows × 17 columns

Data Cleaning and Data Preprocessing

```
In [3]: df=df.dropna()
```

```
In [4]: df.columns
```

```
Out[4]: Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      24758 non-null   object 
 1   BEN        24758 non-null   float64
 2   CO         24758 non-null   float64
 3   EBE        24758 non-null   float64
 4   MXY        24758 non-null   float64
 5   NMHC       24758 non-null   float64
 6   NO_2       24758 non-null   float64
 7   NOx        24758 non-null   float64
 8   OXY        24758 non-null   float64
 9   O_3         24758 non-null   float64
 10  PM10       24758 non-null   float64
 11  PM25       24758 non-null   float64
 12  PXY        24758 non-null   float64
 13  SO_2       24758 non-null   float64
 14  TCH         24758 non-null   float64
 15  TOL         24758 non-null   float64
 16  station    24758 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

```
In [7]: data=df[['NMHC', 'NO_2', 'O_3']]  
data
```

Out[7]:

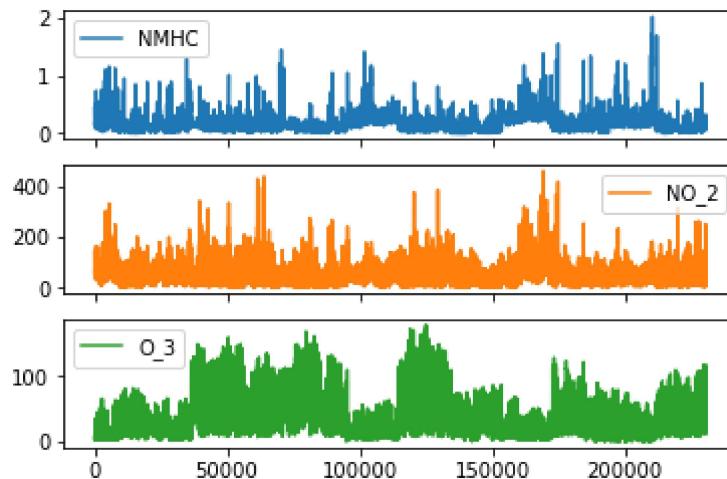
	NMHC	NO_2	O_3
5	0.44	142.199997	5.990000
22	0.17	59.910000	2.450000
25	0.40	117.699997	4.780000
31	0.25	92.059998	5.920000
48	0.16	60.189999	2.280000
...
230538	0.10	49.259998	64.599998
230541	0.33	63.220001	17.670000
230547	0.26	202.399994	11.130000
230564	0.08	51.900002	48.410000
230567	0.24	107.300003	17.730000

24758 rows × 3 columns

Line chart

```
In [8]: data.plot.line(subplots=True)
```

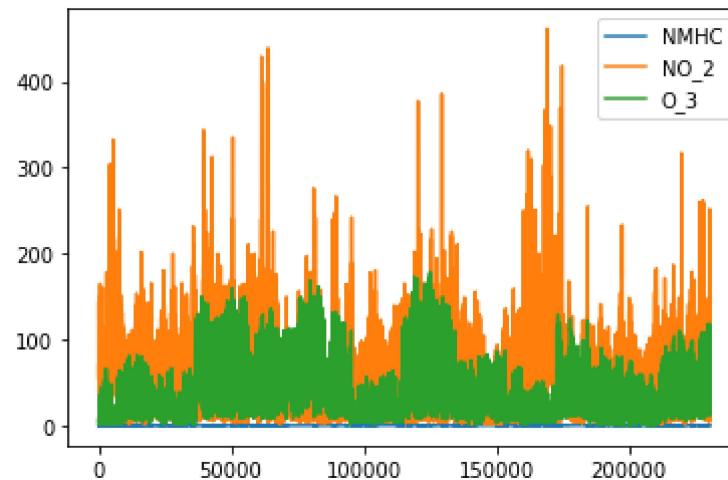
Out[8]: array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

```
In [10]: data.plot.line()
```

```
Out[10]: <AxesSubplot:>
```

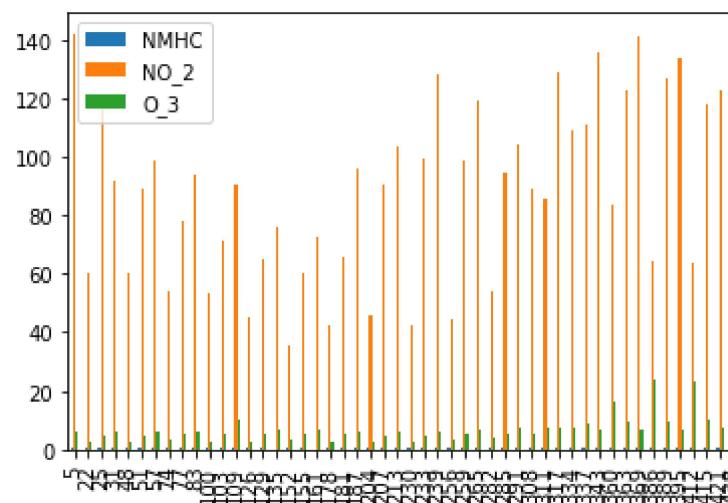


Bar chart

```
In [11]: b=data[0:50]
```

```
In [12]: b.plot.bar()
```

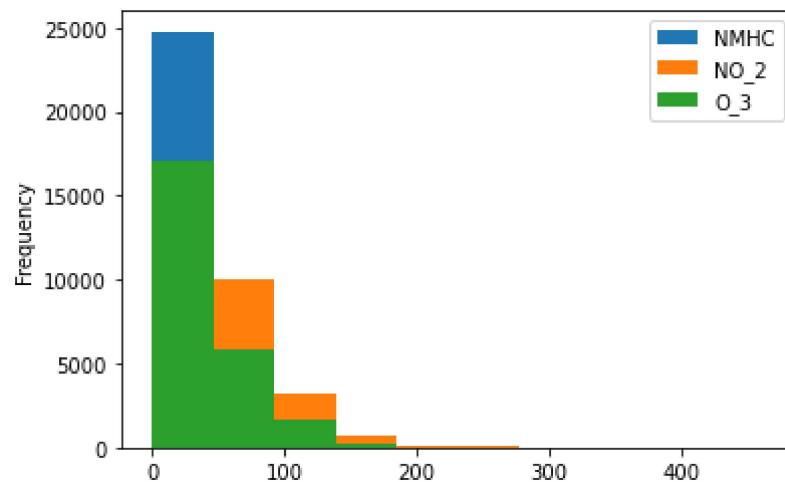
```
Out[12]: <AxesSubplot:>
```



Histogram

In [13]: `data.plot.hist()`

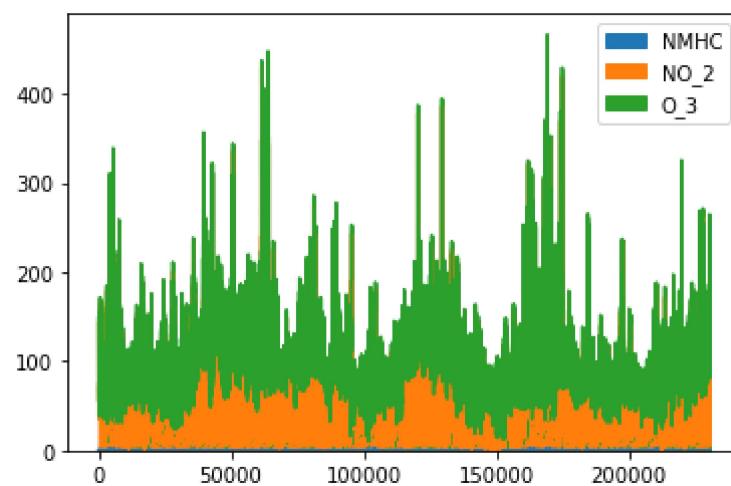
Out[13]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [14]: `data.plot.area()`

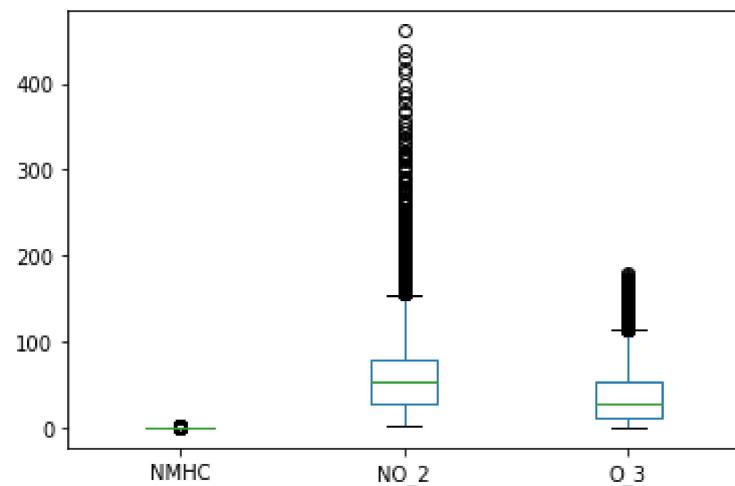
Out[14]: <AxesSubplot:>



Box chart

In [15]: `data.plot.box()`

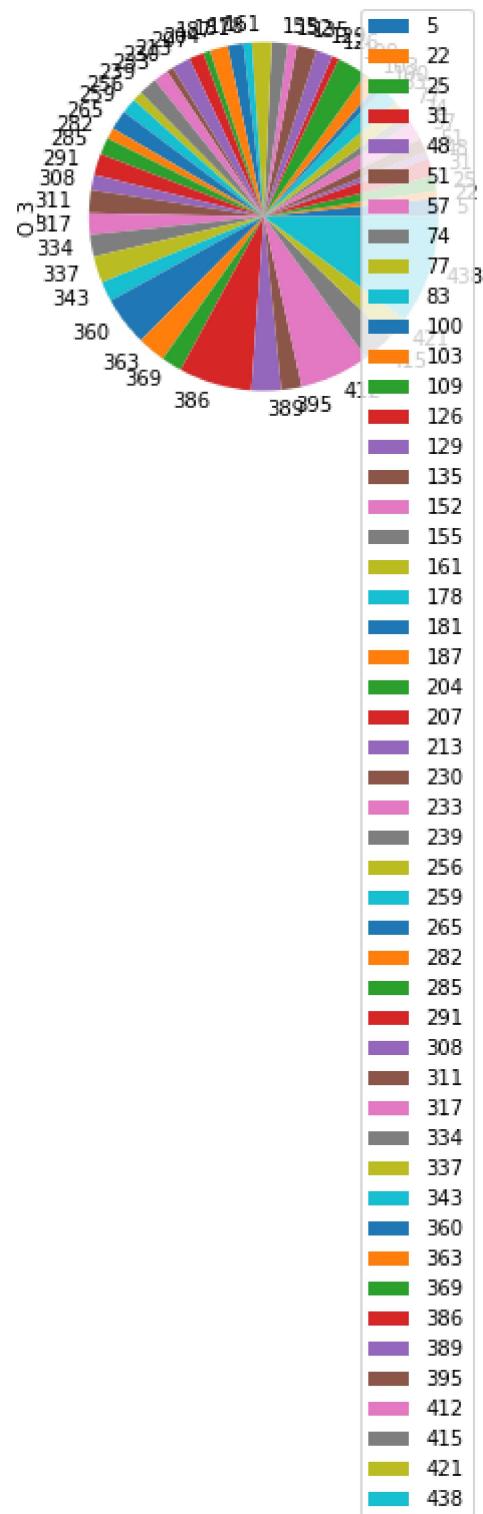
Out[15]: <AxesSubplot:>



Pie chart

```
In [17]: b.plot.pie(y='0_3' )
```

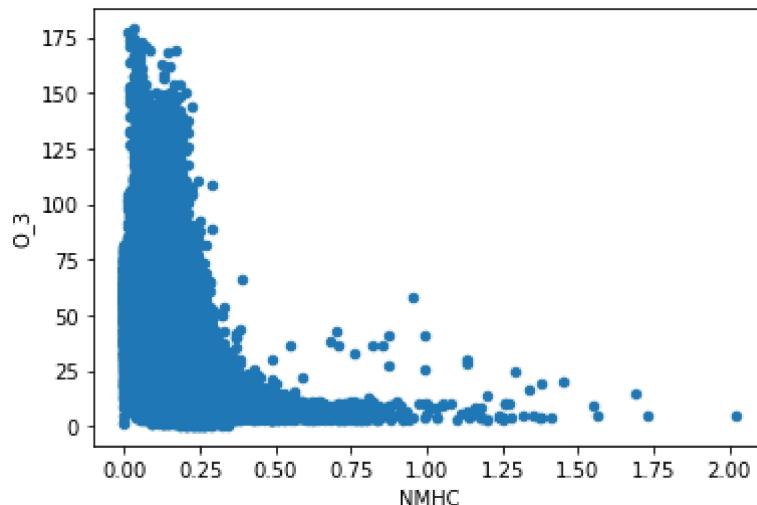
```
Out[17]: <AxesSubplot:ylabel='0_3'>
```



Scatter chart

```
In [18]: data.plot.scatter(x='NMHC' ,y='O_3')
```

```
Out[18]: <AxesSubplot:xlabel='NMHC', ylabel='O_3'>
```



```
In [19]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        24758 non-null   object 
 1   BEN          24758 non-null   float64
 2   CO           24758 non-null   float64
 3   EBE          24758 non-null   float64
 4   MXY          24758 non-null   float64
 5   NMHC         24758 non-null   float64
 6   NO_2          24758 non-null   float64
 7   NOx          24758 non-null   float64
 8   OXY          24758 non-null   float64
 9   O_3           24758 non-null   float64
 10  PM10         24758 non-null   float64
 11  PM25         24758 non-null   float64
 12  PXY          24758 non-null   float64
 13  SO_2          24758 non-null   float64
 14  TCH           24758 non-null   float64
 15  TOL           24758 non-null   float64
 16  station       24758 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

In [20]: df.describe()

Out[20]:

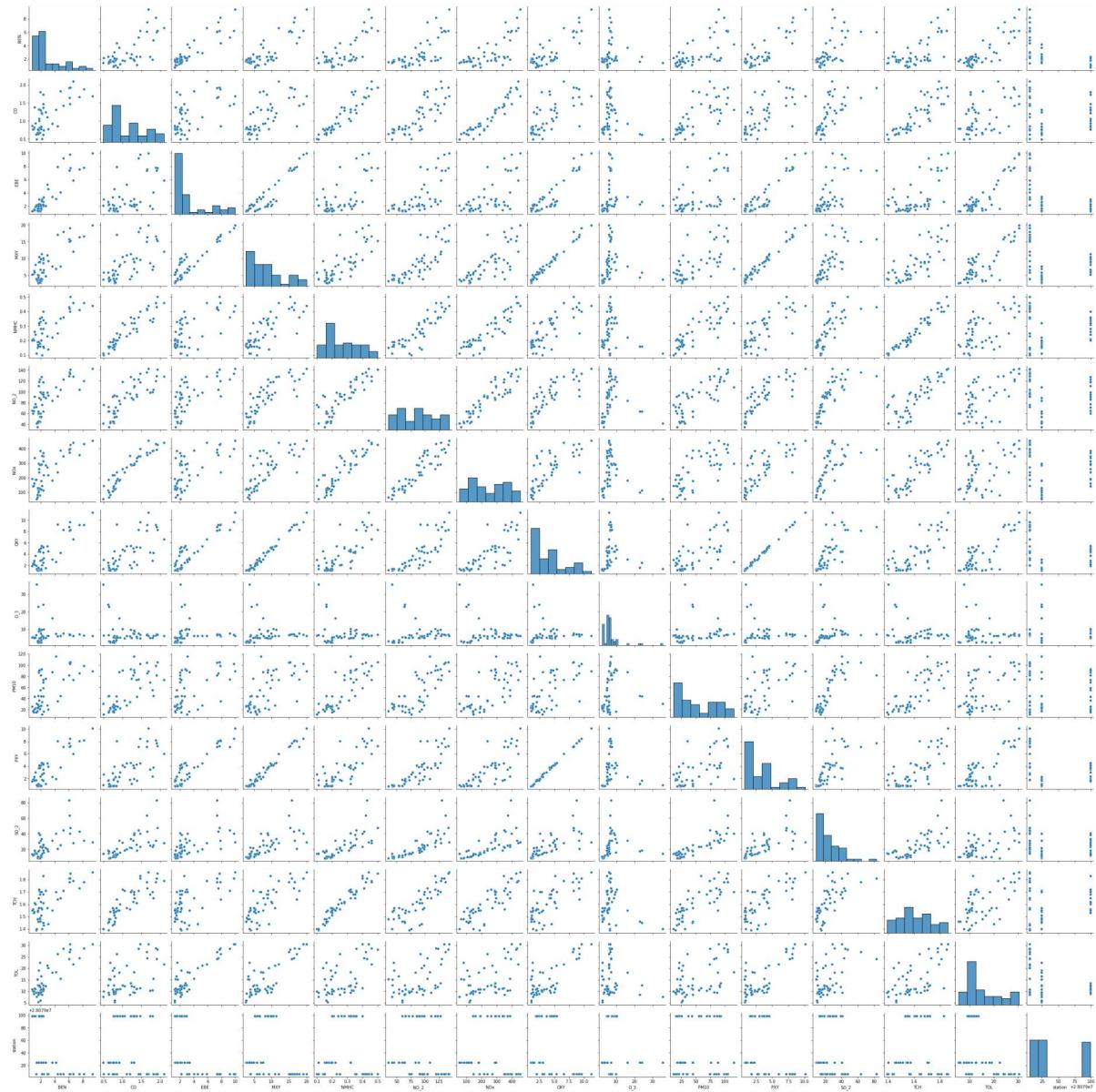
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	247
mean	1.350624	0.600713	1.824534	3.835034	0.176546	58.333481	1
std	1.541636	0.419048	1.868939	4.069036	0.126683	40.529382	1
min	0.110000	0.000000	0.170000	0.150000	0.000000	1.680000	
25%	0.450000	0.360000	0.810000	1.060000	0.100000	28.450001	
50%	0.850000	0.500000	1.130000	2.500000	0.150000	52.959999	
75%	1.680000	0.720000	2.160000	5.090000	0.220000	79.347498	1
max	45.430000	7.250000	57.799999	66.900002	2.020000	461.299988	16

In [21]: df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3', 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]

EDA AND VISUALIZATION

```
In [22]: sns.pairplot(df1[0:50])
```

```
Out[22]: <seaborn.axisgrid.PairGrid at 0x21a7763ad30>
```

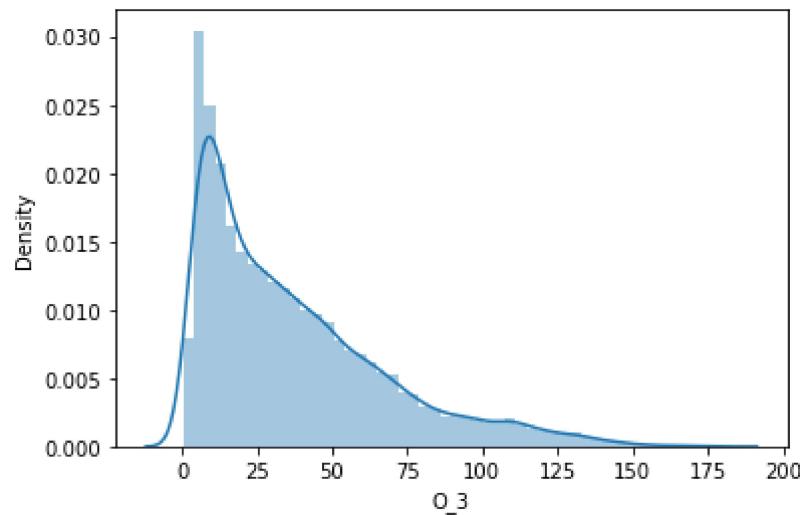


```
In [23]: sns.distplot(df1['O_3'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

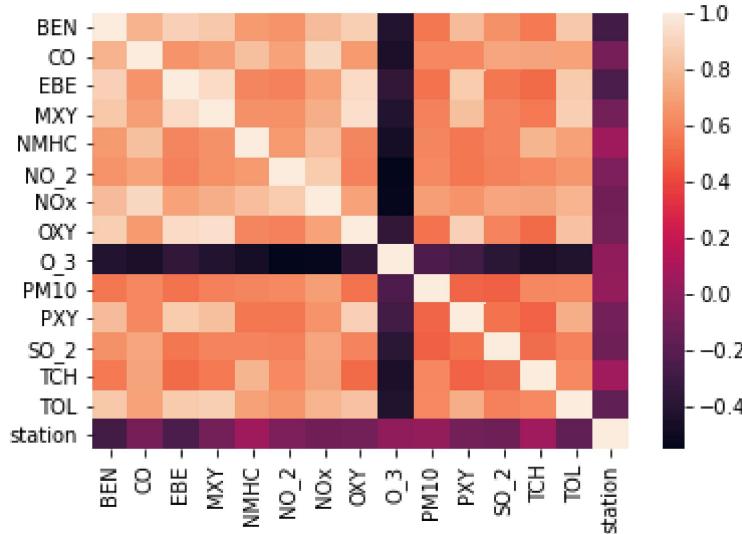
```
warnings.warn(msg, FutureWarning)
```

```
Out[23]: <AxesSubplot:xlabel='O_3', ylabel='Density'>
```



```
In [24]: sns.heatmap(df1.corr())
```

```
Out[24]: <AxesSubplot:>
```



TO TRAIN THE MODEL AND MODEL BUILDING

```
In [67]: x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
           'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

```
In [68]: 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)
```

Linear Regression

```
In [69]: from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

```
Out[69]: LinearRegression()
```

```
In [70]: lr.intercept_
```

```
Out[70]: 28079021.191809315
```

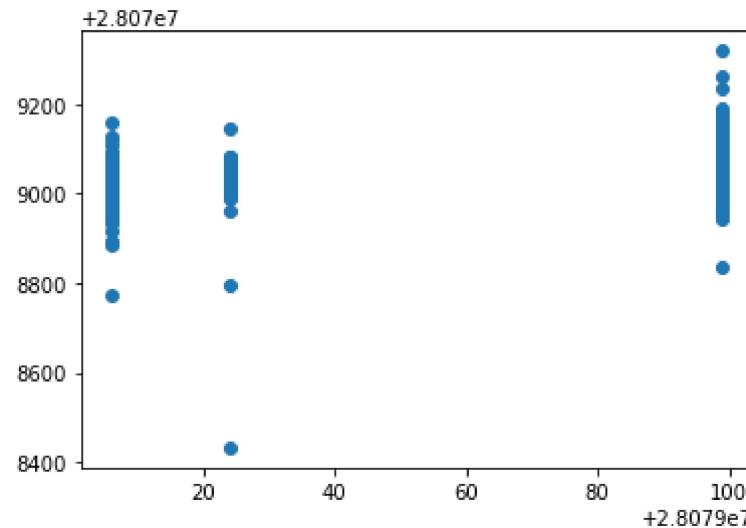
```
In [71]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

```
Out[71]:
```

	Co-efficient
BEN	-18.378990
CO	-9.956654
EBE	-23.531143
MXY	4.689670
NMHC	126.649643
NO_2	-0.016238
NOx	-0.004541
OXY	15.492118
O_3	-0.053784
PM10	0.131450
PXY	5.674572
SO_2	-0.652901
TCH	17.650518
TOL	-0.517190

```
In [72]: prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

```
Out[72]: <matplotlib.collections.PathCollection at 0x21a07f864f0>
```



ACCURACY

```
In [73]: lr.score(x_test,y_test)
```

```
Out[73]: 0.38347413896089644
```

```
In [74]: lr.score(x_train,y_train)
```

```
Out[74]: 0.3972450971871986
```

Ridge and Lasso

```
In [75]: from sklearn.linear_model import Ridge,Lasso
```

```
In [76]: rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
Out[76]: Ridge(alpha=10)
```

Accuracy(Ridge)

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

```
Out[77]: 0.3820493845006965
```

```
In [78]: rr.score(x_train,y_train)
```

```
Out[78]: 0.39661106884033615
```

```
In [79]: la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
Out[79]: Lasso(alpha=10)
```

```
In [81]: la.score(x_test,y_test)
```

```
Out[81]: 0.05647819697010681
```

Accuracy(Lasso)

```
In [80]: la.score(x_train,y_train)
```

```
Out[80]: 0.062336248080204326
```

Accuracy(Elastic Net)

```
In [82]: from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

```
Out[82]: ElasticNet()
```

```
In [83]: en.coef_
```

```
Out[83]: array([-8.55559241e+00,  0.00000000e+00, -9.05278481e+00,  3.41965661e+00,
        4.01206179e-01, -3.00597081e-03,  5.10526761e-03,  3.44982093e+00,
       -1.23661221e-01,  2.93481571e-01,  2.43724729e+00, -4.22212620e-01,
       5.28019988e-01, -9.96915988e-01])
```

```
In [84]: en.intercept_
```

```
Out[84]: 28079052.093522523
```

```
In [85]: prediction=en.predict(x_test)
```

```
In [86]: en.score(x_test,y_test)
```

```
Out[86]: 0.2328938395355542
```

Evaluation Metrics

```
In [87]: from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

32.338665036423414
1266.5452114192653
35.58855450027811

Logistic Regression

```
In [88]: from sklearn.linear_model import LogisticRegression
```

```
In [89]: feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_P10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

```
In [90]: feature_matrix.shape
```

```
Out[90]: (24758, 14)
```

```
In [91]: target_vector.shape
```

```
Out[91]: (24758,)
```

```
In [92]: from sklearn.preprocessing import StandardScaler
```

```
In [93]: fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [94]: logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

```
Out[94]: LogisticRegression(max_iter=10000)
```

```
In [95]: observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

```
In [96]: prediction=logr.predict(observation)
```

```
print(prediction)
```

```
[28079099]
```

```
In [97]: logr.classes_
```

```
Out[97]: array([28079006, 28079024, 28079099], dtype=int64)
```

```
In [98]: logr.score(fs,target_vector)
```

```
Out[98]: 0.8741416915744405
```

```
In [99]: logr.predict_proba(observation)[0][0]
```

```
Out[99]: 3.5557727473608076e-15
```

```
In [100]: logr.predict_proba(observation)
```

```
Out[100]: array([[3.55577275e-15, 7.80743173e-29, 1.00000000e+00]])
```

Random Forest

```
In [101]: from sklearn.ensemble import RandomForestClassifier
```

```
In [102]: rfc=RandomForestClassifier()  
rfc.fit(x_train,y_train)
```

```
Out[102]: RandomForestClassifier()
```

```
In [103]: parameters={'max_depth':[1,2,3,4,5],  
                     'min_samples_leaf':[5,10,15,20,25],  
                     'n_estimators':[10,20,30,40,50]  
}
```

```
In [104]: from sklearn.model_selection import GridSearchCV  
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")  
grid_search.fit(x_train,y_train)
```

```
Out[104]: 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 [105]: grid_search.best_score_
```

```
Out[105]: 0.8757068667051355
```

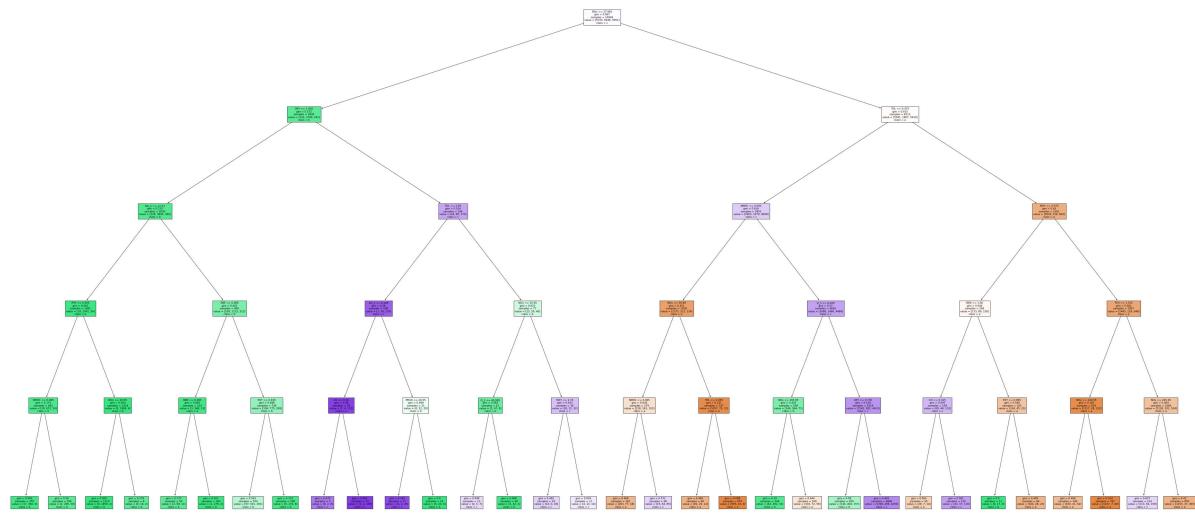
```
In [106]: rfc_best=grid_search.best_estimator_
```

```
In [107]: from sklearn.tree import plot_tree  
  
plt.figure(figsize=(80,40))  
plot_tree(rfc_best.estimators_[5], feature_names=x.columns, class_names=['a', 'b'])
```

```
Out[107]: [Text(2232.0, 1993.2, 'NOx <= 37.965\ngini = 0.667\nsamples = 10949\nvalue = [5733, 5646, 5951]\nnclass = c'),  
Text(1116.0, 1630.8000000000002, 'OXY <= 1.005\ngini = 0.272\nsamples = 2836\nvalue = [152, 3749, 541]\nnclass = b'),  
Text(558.0, 1268.4, 'NO_2 <= 17.27\ngini = 0.215\nsamples = 2650\nvalue = [1  
28, 3664, 366]\nnclass = b'),  
Text(279.0, 906.0, 'PXY <= 0.955\ngini = 0.055\nsamples = 1681\nvalue = [19,  
2541, 54]\nnclass = b'),  
Text(139.5, 543.5999999999999, 'NMHC <= 0.065\ngini = 0.172\nsamples = 463\nvalue = [19, 672, 50]\nnclass = b'),  
Text(69.75, 181.1999999999982, 'gini = 0.082\nsamples = 255\nvalue = [17, 3  
82, 0]\nnclass = b'),  
Text(209.25, 181.1999999999982, 'gini = 0.26\nsamples = 208\nvalue = [2, 29  
0, 50]\nnclass = b'),  
Text(418.5, 543.5999999999999, 'NOx <= 20.95\ngini = 0.004\nsamples = 1218\nvalue = [0, 1869, 4]\nnclass = b'),  
Text(348.75, 181.1999999999982, 'gini = 0.002\nsamples = 1210\nvalue = [0,  
1859, 2]\nnclass = b'),  
Text(488.25, 181.1999999999982, 'gini = 0.278\nsamples = 8\nvalue = [0, 10,  
2]\nnclass = b'),  
Text(837.0, 906.0, 'EBE <= 0.485\ngini = 0.425\nsamples = 969\nvalue = [109,  
1123, 312]\nnclass = b'),  
Text(697.5, 543.5999999999999, 'BEN <= 0.285\ngini = 0.085\nsamples = 233\nvalue = [3, 348, 13]\nnclass = b'),  
Text(627.75, 181.1999999999982, 'gini = 0.273\nsamples = 50\nvalue = [2, 6  
9, 11]\nnclass = b'),  
Text(767.25, 181.1999999999982, 'gini = 0.021\nsamples = 183\nvalue = [1, 2  
79, 2]\nnclass = b'),  
Text(976.5, 543.5999999999999, 'PXY <= 0.915\ngini = 0.496\nsamples = 736\nvalue = [106, 775, 299]\nnclass = b'),  
Text(906.75, 181.1999999999982, 'gini = 0.563\nsamples = 550\nvalue = [97,  
501, 291]\nnclass = b'),  
Text(1046.25, 181.1999999999982, 'gini = 0.112\nsamples = 186\nvalue = [9,  
274, 8]\nnclass = b'),  
Text(1674.0, 1268.4, 'TOL <= 2.65\ngini = 0.524\nsamples = 186\nvalue = [24,  
85, 175]\nnclass = c'),  
Text(1395.0, 906.0, 'SO_2 <= 8.345\ngini = 0.28\nsamples = 109\nvalue = [1,  
26, 135]\nnclass = c'),  
Text(1255.5, 543.5999999999999, 'CO <= 0.16\ngini = 0.08\nsamples = 78\nvalue = [1, 4, 115]\nnclass = c'),  
Text(1185.75, 181.1999999999982, 'gini = 0.375\nsamples = 5\nvalue = [0, 2,  
6]\nnclass = c'),  
Text(1325.25, 181.1999999999982, 'gini = 0.052\nsamples = 73\nvalue = [1,  
2, 109]\nnclass = c'),  
Text(1534.5, 543.5999999999999, 'PM10 <= 20.55\ngini = 0.499\nsamples = 31\nvalue = [0, 22, 20]\nnclass = b'),  
Text(1464.75, 181.1999999999982, 'gini = 0.165\nsamples = 17\nvalue = [0,  
2, 20]\nnclass = c'),  
Text(1604.25, 181.1999999999982, 'gini = 0.0\nsamples = 14\nvalue = [0, 20,  
0]\nnclass = b'),  
Text(1953.0, 906.0, 'NOx <= 32.65\ngini = 0.623\nsamples = 77\nvalue = [23,  
59, 40]\nnclass = b'),  
Text(1813.5, 543.5999999999999, 'O_3 <= 49.995\ngini = 0.362\nsamples = 41\nvalue = [5, 47, 8]\nnclass = b'),  
Text(1743.75, 181.1999999999982, 'gini = 0.648\nsamples = 13\nvalue = [4,  
5, 7]\nnclass = c'),  
Text(1883.25, 181.1999999999982, 'gini = 0.088\nsamples = 28\nvalue = [1, 4
```

```
2, 1]\nclass = b'),  
    Text(2092.5, 543.5999999999999, 'MXY <= 2.74\ngini = 0.612\nsamples = 36\nvalue = [18, 12, 32]\nclass = c'),  
    Text(2022.75, 181.1999999999982, 'gini = 0.492\nsamples = 19\nvalue = [14, 0, 18]\nclass = c'),  
    Text(2162.25, 181.1999999999982, 'gini = 0.604\nsamples = 17\nvalue = [4, 12, 14]\nclass = c'),  
    Text(3348.0, 1630.800000000002, 'TOL <= 9.475\ngini = 0.615\nsamples = 8113\nvalue = [5581, 1897, 5410]\nclass = a'),  
    Text(2790.0, 1268.4, 'NMHC <= 0.095\ngini = 0.616\nsamples = 5852\nvalue = [2963, 1679, 4608]\nclass = c'),  
    Text(2511.0, 906.0, 'NOx <= 58.89\ngini = 0.351\nsamples = 1026\nvalue = [1275, 213, 124]\nclass = a'),  
    Text(2371.5, 543.5999999999999, 'NMHC <= 0.065\ngini = 0.634\nsamples = 279\nvalue = [218, 141, 103]\nclass = a'),  
    Text(2301.75, 181.1999999999982, 'gini = 0.466\nsamples = 181\nvalue = [203, 77, 18]\nclass = a'),  
    Text(2441.25, 181.1999999999982, 'gini = 0.571\nsamples = 98\nvalue = [15, 64, 85]\nclass = c'),  
    Text(2650.5, 543.5999999999999, 'TOL <= 2.495\ngini = 0.151\nsamples = 747\nvalue = [1057, 72, 21]\nclass = a'),  
    Text(2580.75, 181.1999999999982, 'gini = 0.466\nsamples = 90\nvalue = [92, 29, 12]\nclass = a'),  
    Text(2720.25, 181.1999999999982, 'gini = 0.098\nsamples = 657\nvalue = [965, 43, 9]\nclass = a'),  
    Text(3069.0, 906.0, 'O_3 <= 6.025\ngini = 0.57\nsamples = 4826\nvalue = [1688, 1466, 4484]\nclass = c'),  
    Text(2929.5, 543.5999999999999, 'NOx <= 165.05\ngini = 0.435\nsamples = 508\nvalue = [146, 564, 71]\nclass = b'),  
    Text(2859.75, 181.1999999999982, 'gini = 0.19\nsamples = 348\nvalue = [42, 492, 15]\nclass = b'),  
    Text(2999.25, 181.1999999999982, 'gini = 0.644\nsamples = 160\nvalue = [104, 72, 56]\nclass = a'),  
    Text(3208.5, 543.5999999999999, 'OXY <= 0.735\ngini = 0.518\nsamples = 4318\nvalue = [1542, 902, 4413]\nclass = c'),  
    Text(3138.75, 181.1999999999982, 'gini = 0.58\nsamples = 450\nvalue = [144, 404, 157]\nclass = b'),  
    Text(3278.25, 181.1999999999982, 'gini = 0.463\nsamples = 3868\nvalue = [1398, 498, 4256]\nclass = c'),  
    Text(3906.0, 1268.4, 'MXY <= 5.075\ngini = 0.43\nsamples = 2261\nvalue = [2618, 218, 802]\nclass = a'),  
    Text(3627.0, 906.0, 'BEN <= 1.66\ngini = 0.644\nsamples = 264\nvalue = [173, 89, 156]\nclass = a'),  
    Text(3487.5, 543.5999999999999, 'CO <= 0.315\ngini = 0.597\nsamples = 159\nvalue = [69, 44, 133]\nclass = c'),  
    Text(3417.75, 181.1999999999982, 'gini = 0.583\nsamples = 30\nvalue = [26, 7, 14]\nclass = a'),  
    Text(3557.25, 181.1999999999982, 'gini = 0.561\nsamples = 129\nvalue = [43, 37, 119]\nclass = c'),  
    Text(3766.5, 543.5999999999999, 'PXY <= 0.995\ngini = 0.548\nsamples = 105\nvalue = [104, 45, 23]\nclass = a'),  
    Text(3696.75, 181.1999999999982, 'gini = 0.0\nsamples = 11\nvalue = [0, 17, 0]\nclass = b'),  
    Text(3836.25, 181.1999999999982, 'gini = 0.495\nsamples = 94\nvalue = [104, 28, 23]\nclass = a'),  
    Text(4185.0, 906.0, 'TCH <= 1.515\ngini = 0.382\nsamples = 1997\nvalue = [2445, 129, 646]\nclass = a'),
```

```
Text(4045.5, 543.5999999999999, 'NOx <= 133.55\ngini = 0.187\nsamples = 933\nvalue = [1317, 28, 122]\nclass = a'),
Text(3975.75, 181.1999999999982, 'gini = 0.494\nsamples = 146\nvalue = [16
0, 21, 62]\nclass = a'),
Text(4115.25, 181.1999999999982, 'gini = 0.104\nsamples = 787\nvalue = [115
7, 7, 60]\nclass = a'),
Text(4324.5, 543.5999999999999, 'NOx <= 225.35\ngini = 0.493\nsamples = 1064
value = [1128, 101, 524]\nclass = a'),
Text(4254.75, 181.1999999999982, 'gini = 0.627\nsamples = 214\nvalue = [11
3, 64, 159]\nclass = c'),
Text(4394.25, 181.1999999999982, 'gini = 0.42\nsamples = 850\nvalue = [101
5, 37, 365]\nclass = a')
```



Conclusion

Accuracy

Linear Regression:0.3972450971871986

Ridge Regression:0.39661106884033615

Lasso Regression:0.062336248080204326

ElasticNet Regression:0.2328938395355542

Logistic Regression:0.8741416915744405

Random Forest:0.8757068667051355

Random Forest is suitable for this dataset

