

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	PM
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2600
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5800
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1900
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5300
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7600
...
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.8300
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9200
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0300
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.3600

215688 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: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      24717 non-null   object 
 1   BEN        24717 non-null   float64
 2   CO         24717 non-null   float64
 3   EBE        24717 non-null   float64
 4   MXY        24717 non-null   float64
 5   NMHC       24717 non-null   float64
 6   NO_2       24717 non-null   float64
 7   NOx        24717 non-null   float64
 8   OXY        24717 non-null   float64
 9   O_3         24717 non-null   float64
 10  PM10       24717 non-null   float64
 11  PM25       24717 non-null   float64
 12  PXY        24717 non-null   float64
 13  SO_2       24717 non-null   float64
 14  TCH         24717 non-null   float64
 15  TOL         24717 non-null   float64
 16  station    24717 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

```
In [6]: data=df[['PXY', 'NOx', 'OXY']]  
data
```

Out[6]:

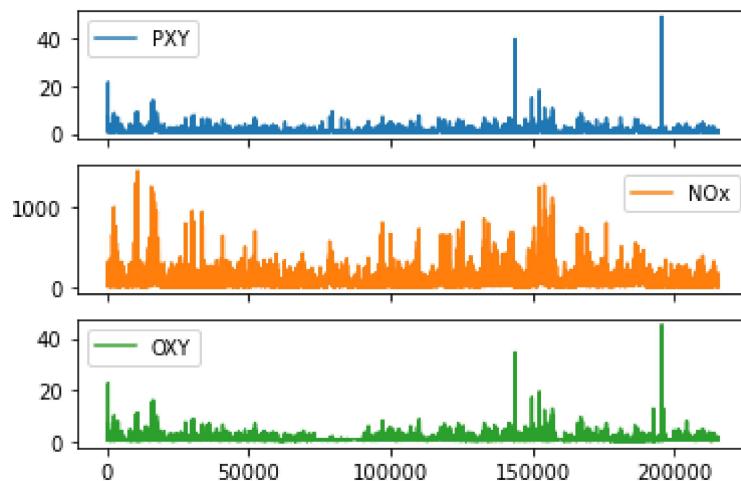
	PXY	NOx	OXY
3	1.30	81.360001	1.57
20	0.84	19.240000	1.00
24	1.07	43.919998	1.28
28	1.04	48.869999	1.21
45	0.88	19.299999	1.00
...
215659	0.84	29.490000	0.86
215663	1.03	69.870003	1.26
215667	0.92	82.629997	1.13
215683	0.74	24.510000	1.00
215687	0.83	64.480003	1.06

24717 rows × 3 columns

Line chart

```
In [7]: data.plot.line(subplots=True)
```

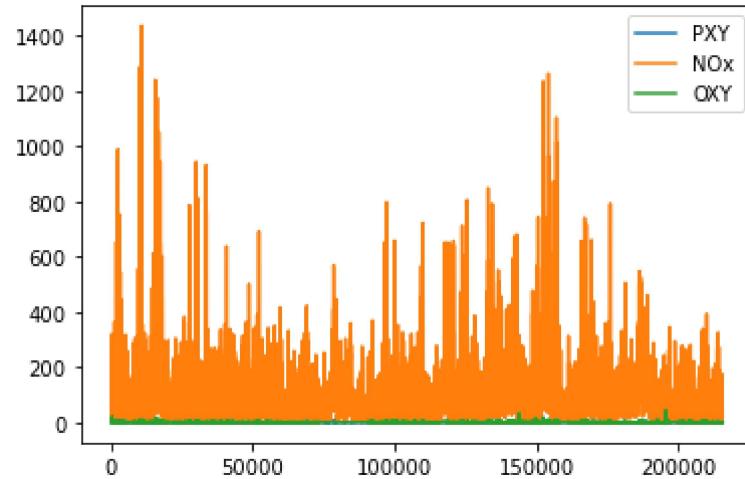
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

```
Out[8]: <AxesSubplot:>
```

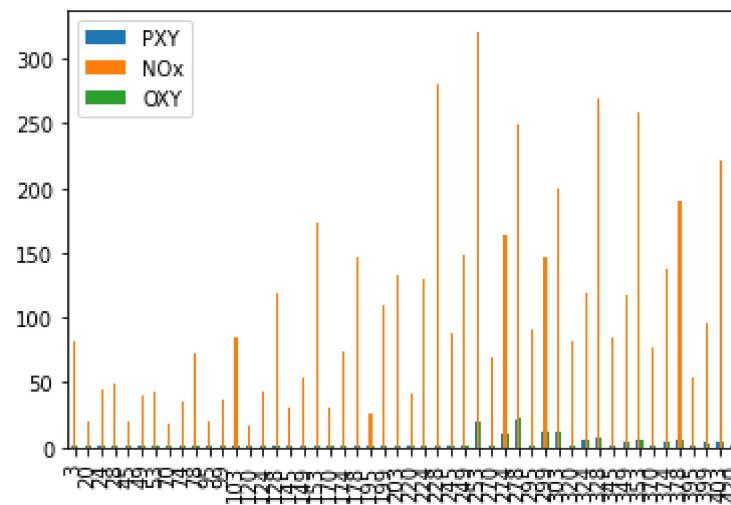


Bar chart

```
In [9]: b=data[0:50]
```

```
In [10]: b.plot.bar()
```

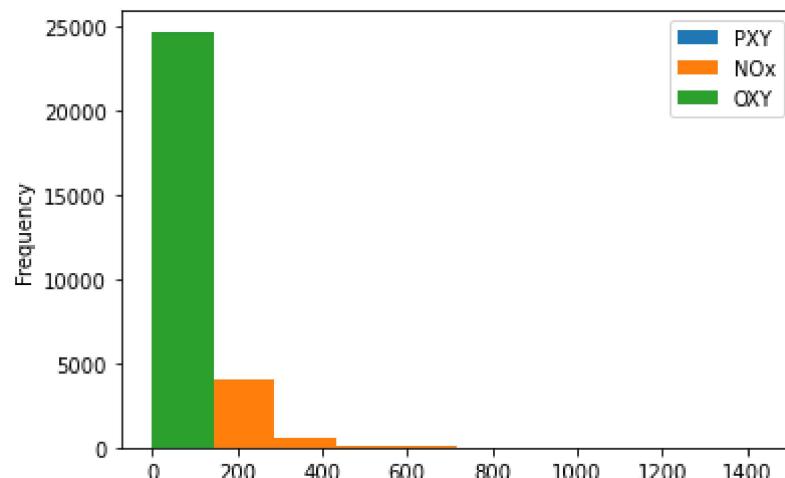
```
Out[10]: <AxesSubplot:>
```



Histogram

```
In [11]: data.plot.hist()
```

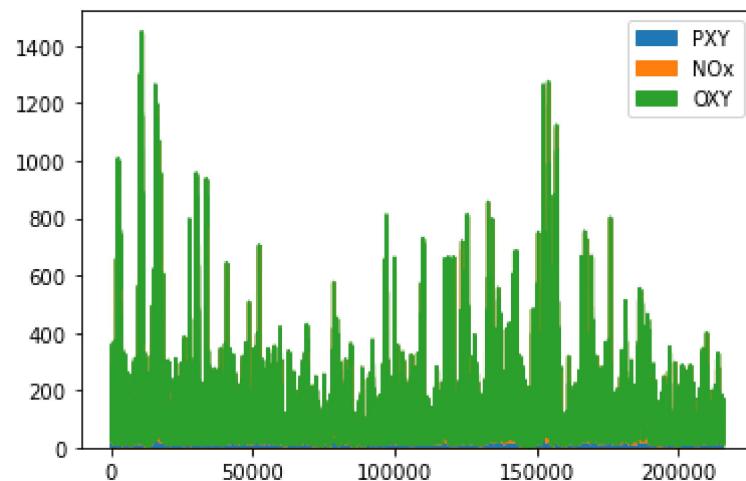
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

```
In [12]: data.plot.area()
```

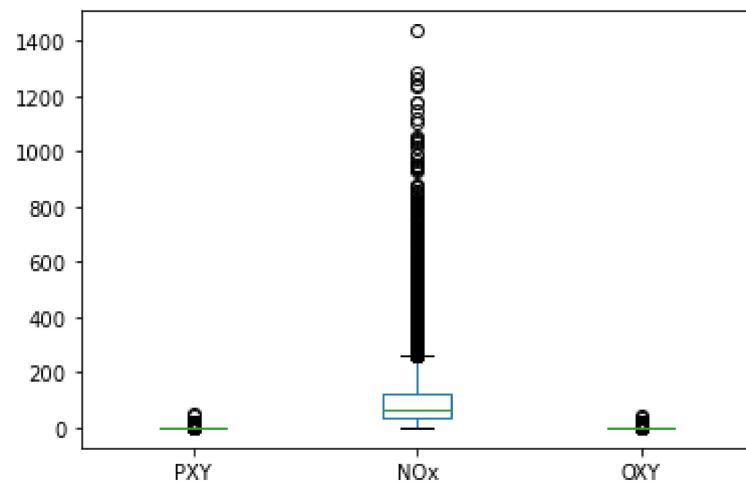
```
Out[12]: <AxesSubplot:>
```



Box chart

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

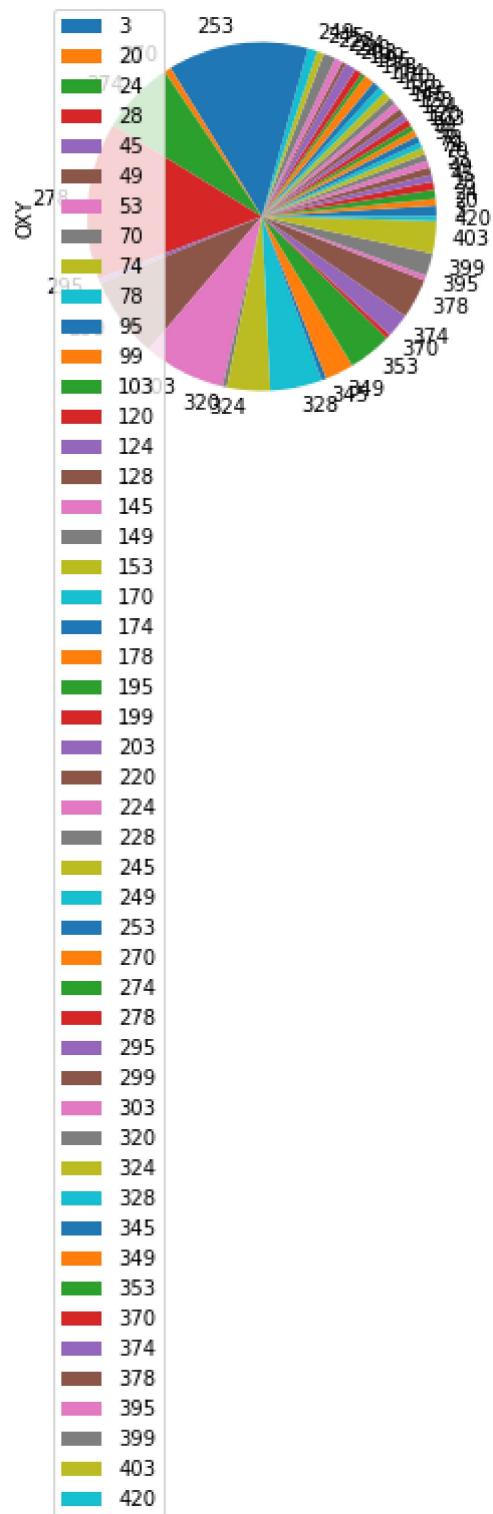
Out[13]: <AxesSubplot:>



Pie chart

```
In [14]: b.plot.pie(y='OXY' )
```

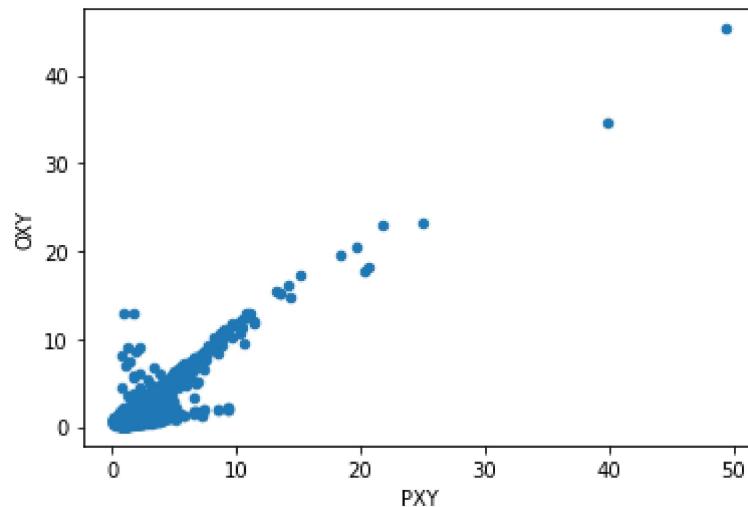
```
Out[14]: <AxesSubplot:ylabel='OXY'>
```



Scatter chart

```
In [15]: data.plot.scatter(x='PXY' ,y='OXY')
```

```
Out[15]: <AxesSubplot:xlabel='PXY', ylabel='OXY'>
```



```
In [16]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        24717 non-null   object 
 1   BEN         24717 non-null   float64
 2   CO          24717 non-null   float64
 3   EBE         24717 non-null   float64
 4   MXY         24717 non-null   float64
 5   NMHC        24717 non-null   float64
 6   NO_2         24717 non-null   float64
 7   NOx         24717 non-null   float64
 8   OXY         24717 non-null   float64
 9   O_3          24717 non-null   float64
 10  PM10        24717 non-null   float64
 11  PM25        24717 non-null   float64
 12  PXY         24717 non-null   float64
 13  SO_2         24717 non-null   float64
 14  TCH          24717 non-null   float64
 15  TOL          24717 non-null   float64
 16  station      24717 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

In [17]: `df.describe()`

Out[17]:

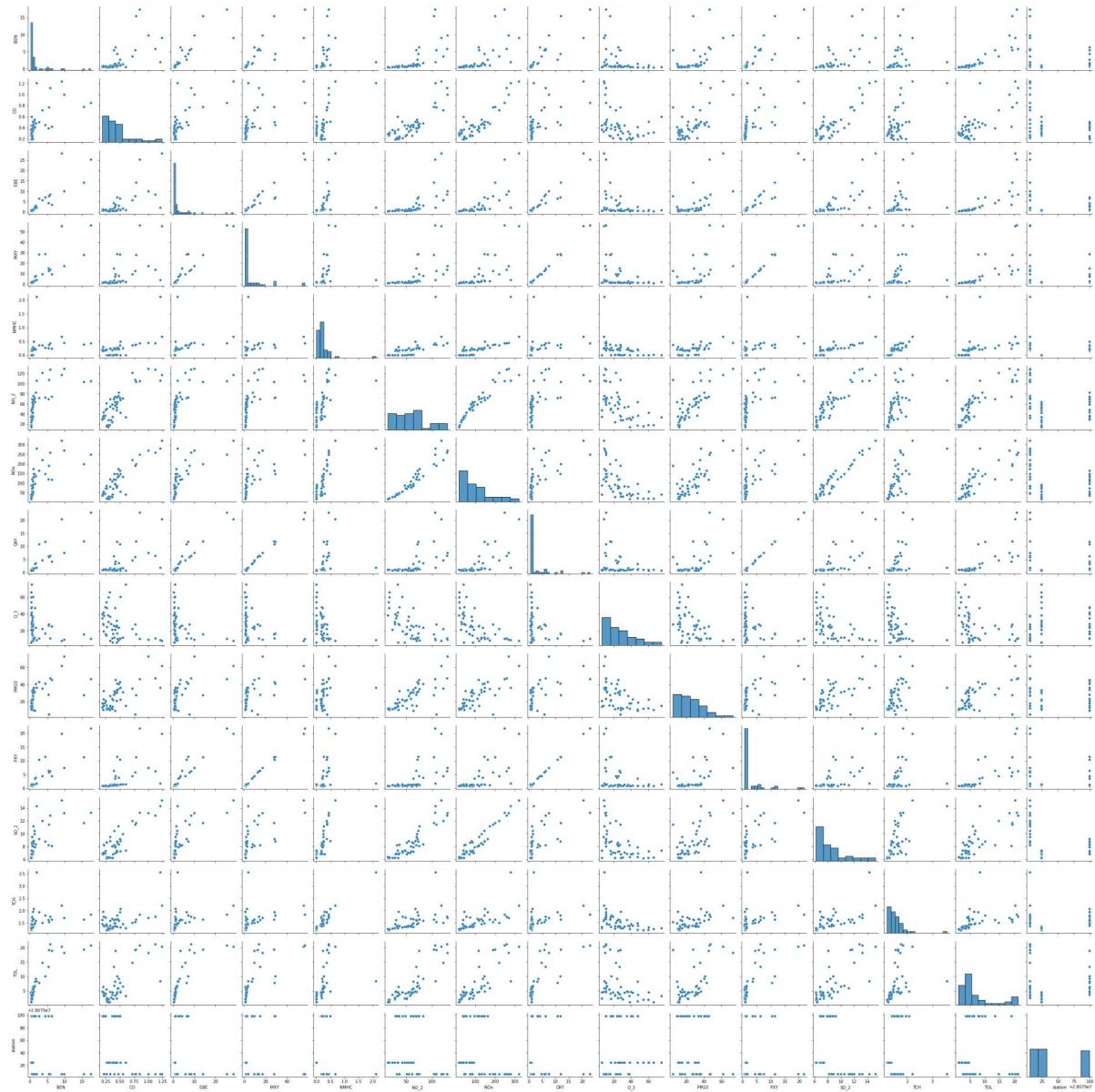
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000
mean	1.010583	0.448056	1.262430	2.244469	0.219582	55.563929	
std	1.007345	0.291706	1.074768	2.242214	0.141661	38.911677	
min	0.170000	0.060000	0.250000	0.240000	0.000000	0.600000	
25%	0.460000	0.270000	0.720000	0.990000	0.140000	26.510000	
50%	0.670000	0.370000	1.000000	1.490000	0.190000	47.930000	
75%	1.180000	0.570000	1.430000	2.820000	0.260000	76.269997	1
max	22.379999	5.570000	47.669998	56.500000	2.580000	477.399994	14

In [18]: `df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3', 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]`

EDA AND VISUALIZATION

```
In [19]: sns.pairplot(df1[0:50])
```

```
Out[19]: <seaborn.axisgrid.PairGrid at 0x2183dee99d0>
```

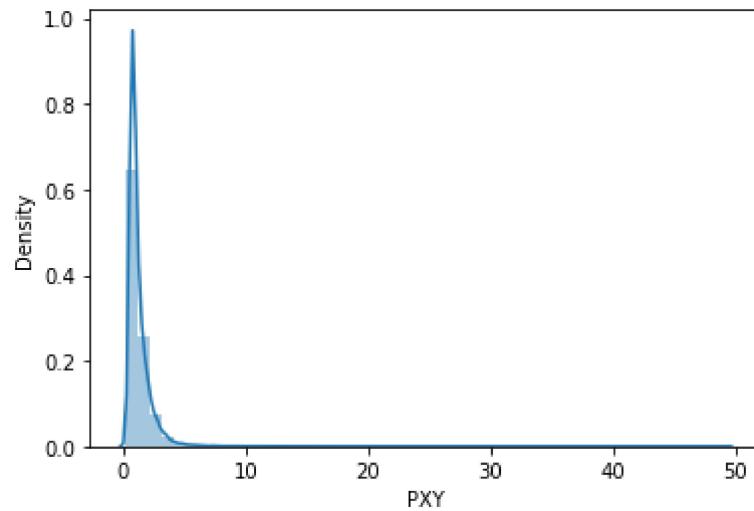


In [20]: `sns.distplot(df1['PXY'])`

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).

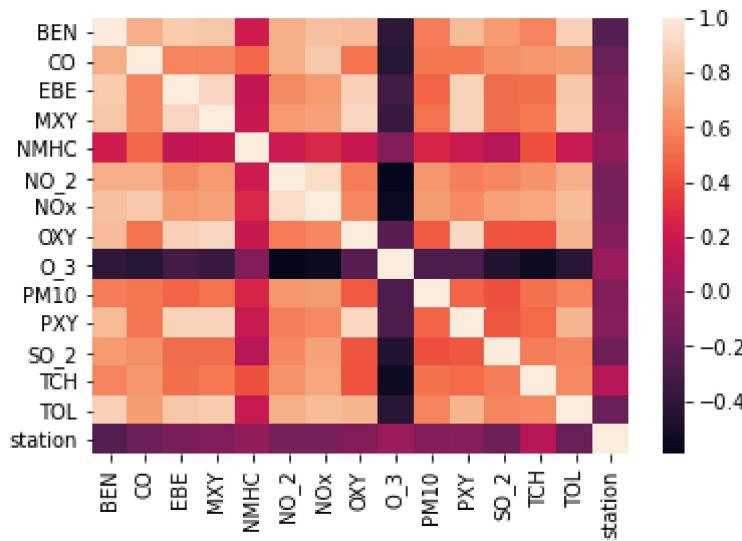
```
warnings.warn(msg, FutureWarning)
```

Out[20]: <AxesSubplot:xlabel='PXY', ylabel='Density'>



In [21]: `sns.heatmap(df1.corr())`

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BUILDING

```
In [22]: x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
           'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

```
In [23]: 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 [24]: from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]: LinearRegression()

```
In [25]: lr.intercept_
```

Out[25]: 28078901.21705944

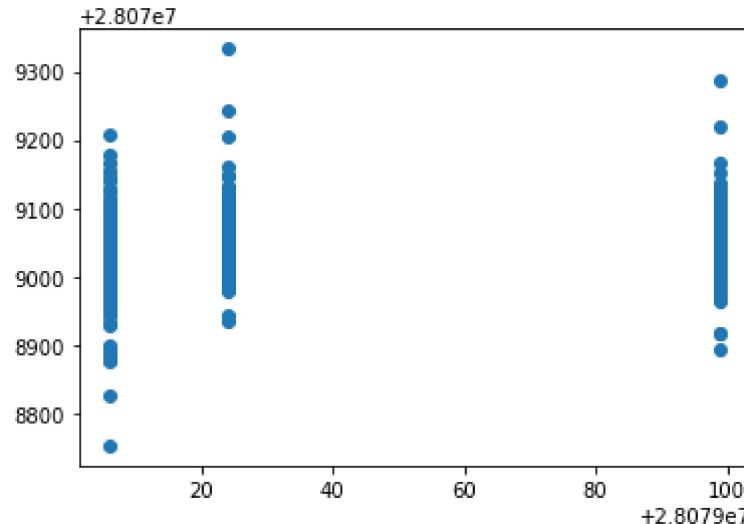
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-37.569492
CO	-27.640183
EBE	7.444821
MXY	0.104949
NMHC	-18.836149
NO_2	-0.189433
NOx	0.210363
OXY	12.034927
O_3	0.022955
PM10	-0.056456
PXY	1.639321
SO_2	-0.369752
TCH	119.045865
TOL	-1.016695

```
In [27]: prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

```
Out[27]: <matplotlib.collections.PathCollection at 0x2184ffffb610>
```



ACCURACY

```
In [28]: lr.score(x_test,y_test)
```

```
Out[28]: 0.2791163528241405
```

```
In [29]: lr.score(x_train,y_train)
```

```
Out[29]: 0.2904829269822444
```

Ridge and Lasso

```
In [30]: from sklearn.linear_model import Ridge,Lasso
```

```
In [31]: rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
Out[31]: Ridge(alpha=10)
```

Accuracy(Ridge)

```
In [32]: rr.score(x_test,y_test)
```

```
Out[32]: 0.2789736363242975
```

```
In [33]: rr.score(x_train,y_train)
```

```
Out[33]: 0.2901722163574638
```

```
In [34]: la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
Out[34]: Lasso(alpha=10)
```

```
In [35]: la.score(x_test,y_test)
```

```
Out[35]: 0.03411461118355863
```

Accuracy(Lasso)

```
In [36]: la.score(x_train,y_train)
```

```
Out[36]: 0.03832089827644303
```

Accuracy(Elastic Net)

```
In [37]: from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

```
Out[37]: ElasticNet()
```

```
In [38]: en.coef_
```

```
Out[38]: array([-7.09949154, -0.63183419,  0.29499383,  2.21487181,  0.,
       -0.23307397,  0.12859889,  1.00478444, -0.14991085,  0.07736015,
       1.84017177, -0.83222241,  1.50100091, -1.89102312])
```

```
In [39]: en.intercept_
```

```
Out[39]: 28079064.81809502
```

```
In [40]: prediction=en.predict(x_test)
```

```
In [41]: en.score(x_test,y_test)
```

```
Out[41]: 0.10463204276063354
```

Evaluation Metrics

```
In [42]: 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)))
```

```
35.84045092335339  
1465.2505318850974  
38.27859103840027
```

Logistic Regression

```
In [43]: from sklearn.linear_model import LogisticRegression
```

```
In [44]: feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_P10', 'PXY', 'SO_2', 'TCH', 'TOL']]  
target_vector=df['station']
```

```
In [45]: feature_matrix.shape
```

```
Out[45]: (24717, 14)
```

```
In [46]: target_vector.shape
```

```
Out[46]: (24717,)
```

```
In [47]: from sklearn.preprocessing import StandardScaler
```

```
In [48]: fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [49]: logr=LogisticRegression(max_iter=10000)  
logr.fit(fs,target_vector)
```

```
Out[49]: LogisticRegression(max_iter=10000)
```

```
In [50]: observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

```
In [51]: prediction=logr.predict(observation)
```

```
print(prediction)
```

```
[28079099]
```

```
In [52]: logr.classes_
```

```
Out[52]: array([28079006, 28079024, 28079099], dtype=int64)
```

```
In [53]: logr.score(fs,target_vector)
```

```
Out[53]: 0.8951733624630821
```

```
In [54]: logr.predict_proba(observation)[0][0]
```

```
Out[54]: 5.447205522232353e-13
```

```
In [55]: logr.predict_proba(observation)
```

```
Out[55]: array([[5.44720552e-13, 8.28692830e-44, 1.00000000e+00]])
```

Random Forest

```
In [56]: from sklearn.ensemble import RandomForestClassifier
```

```
In [57]: rfc=RandomForestClassifier()  
rfc.fit(x_train,y_train)
```

```
Out[57]: RandomForestClassifier()
```

```
In [58]: parameters={'max_depth':[1,2,3,4,5],  
                  'min_samples_leaf':[5,10,15,20,25],  
                  'n_estimators':[10,20,30,40,50]  
}
```

```
In [59]: 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[59]: 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 [60]: grid_search.best_score_
```

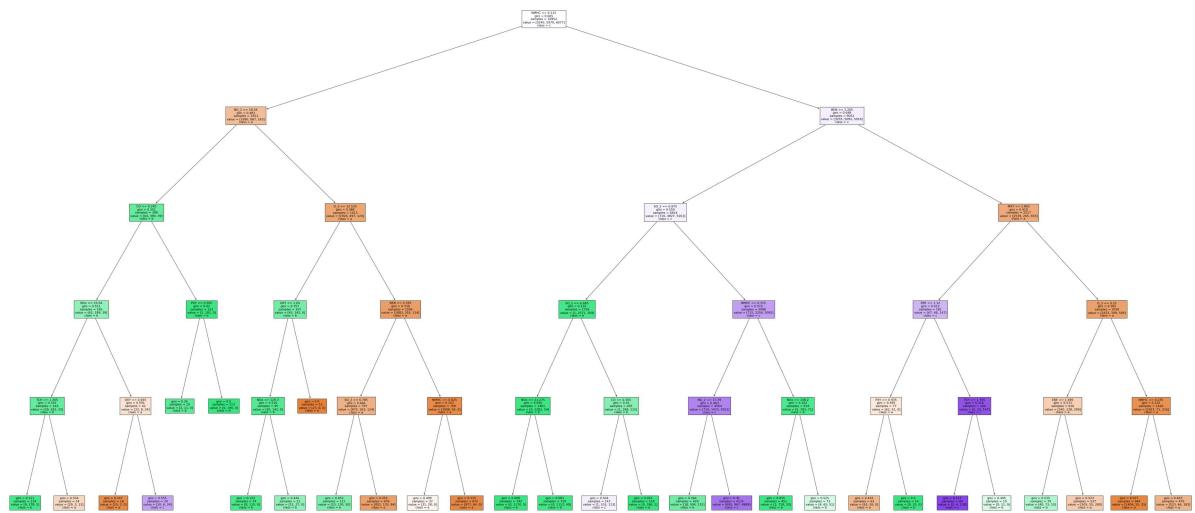
```
Out[60]: 0.8947460115206034
```

```
In [61]: rfc_best=grid_search.best_estimator_
```

```
In [62]: 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[62]: [Text(2025.3333333333335, 1993.2, 'NMHC <= 0.115\ngini = 0.665\nsamples = 109\nvalue = [5245, 5979, 6077]\nclass = c'),  
Text(909.3333333333334, 1630.8000000000002, 'NO_2 <= 18.34\ngini = 0.483\nsamples = 1921\nvalue = [1990, 887, 163]\nclass = a'),  
Text(537.3333333333334, 1268.4, 'CO <= 0.245\ngini = 0.351\nsamples = 308\nvalue = [64, 390, 39]\nclass = b'),  
Text(330.6666666666667, 906.0, 'NOx <= 19.04\ngini = 0.511\nsamples = 185\nvalue = [62, 189, 39]\nclass = b'),  
Text(165.3333333333334, 543.5999999999999, 'TCH <= 1.305\ngini = 0.332\nsamples = 143\nvalue = [29, 181, 15]\nclass = b'),  
Text(82.6666666666667, 181.1999999999982, 'gini = 0.111\nsamples = 119\nvalue = [9, 178, 2]\nclass = b'),  
Text(248.0, 181.1999999999982, 'gini = 0.554\nsamples = 24\nvalue = [20, 3, 13]\nclass = a'),  
Text(496.0, 543.5999999999999, 'OXY <= 0.655\ngini = 0.591\nsamples = 42\nvalue = [33, 8, 24]\nclass = a'),  
Text(413.3333333333337, 181.1999999999982, 'gini = 0.147\nsamples = 18\nvalue = [23, 2, 0]\nclass = a'),  
Text(578.6666666666667, 181.1999999999982, 'gini = 0.555\nsamples = 24\nvalue = [10, 6, 24]\nclass = c'),  
Text(744.0, 906.0, 'PXY <= 0.505\ngini = 0.02\nsamples = 123\nvalue = [2, 201, 0]\nclass = b'),  
Text(661.333333333334, 543.5999999999999, 'gini = 0.26\nsamples = 10\nvalue = [2, 11, 0]\nclass = b'),  
Text(826.6666666666667, 543.5999999999999, 'gini = 0.0\nsamples = 113\nvalue = [0, 190, 0]\nclass = b'),  
Text(1281.333333333335, 1268.4, 'O_3 <= 12.535\ngini = 0.388\nsamples = 1613\nvalue = [1926, 497, 124]\nclass = a'),  
Text(1074.6666666666667, 906.0, 'OXY <= 1.04\ngini = 0.357\nsamples = 107\nvalue = [43, 142, 0]\nclass = b'),  
Text(992.0, 543.5999999999999, 'NOx <= 129.3\ngini = 0.216\nsamples = 95\nvalue = [20, 142, 0]\nclass = b'),  
Text(909.333333333334, 181.1999999999982, 'gini = 0.122\nsamples = 74\nvalue = [8, 115, 0]\nclass = b'),  
Text(1074.6666666666667, 181.1999999999982, 'gini = 0.426\nsamples = 21\nvalue = [12, 27, 0]\nclass = b'),  
Text(1157.333333333335, 543.5999999999999, 'gini = 0.0\nsamples = 12\nvalue = [23, 0, 0]\nclass = a'),  
Text(1488.0, 906.0, 'BEN <= 0.785\ngini = 0.339\nsamples = 1506\nvalue = [1883, 355, 124]\nclass = a'),  
Text(1322.6666666666667, 543.5999999999999, 'SO_2 <= 6.785\ngini = 0.466\nsamples = 797\nvalue = [875, 263, 124]\nclass = a'),  
Text(1240.0, 181.1999999999982, 'gini = 0.453\nsamples = 121\nvalue = [23, 130, 30]\nclass = b'),  
Text(1405.333333333335, 181.1999999999982, 'gini = 0.354\nsamples = 676\nvalue = [852, 133, 94]\nclass = a'),  
Text(1653.333333333335, 543.5999999999999, 'NMHC <= 0.025\ngini = 0.153\nsamples = 709\nvalue = [1008, 92, 0]\nclass = a'),  
Text(1570.6666666666667, 181.1999999999982, 'gini = 0.499\nsamples = 37\nvalue = [31, 28, 0]\nclass = a'),  
Text(1736.0, 181.1999999999982, 'gini = 0.115\nsamples = 672\nvalue = [977, 64, 0]\nclass = a'),  
Text(3141.333333333335, 1630.8000000000002, 'BEN <= 1.265\ngini = 0.648\nsamples = 9031\nvalue = [3255, 5092, 5914]\nclass = c'),  
Text(2480.0, 1268.4, 'SO_2 <= 6.875\ngini = 0.559\nsamples = 6814\nvalue = [716, 4827, 5261]\nclass = c'),  
Text(2149.333333333335, 906.0, 'SO_2 <= 6.685\ngini = 0.116\nsamples = 1726
```

```
\nvalue = [1, 2571, 169]\nclass = b'),\n    Text(1984.0, 543.5999999999999, 'NOx <= 21.275\ngini = 0.045\nsamples = 1461\nvalue = [0, 2282, 54]\nclass = b'),\n    Text(1901.333333333335, 181.19999999999982, 'gini = 0.008\nsamples = 742\nvalue = [0, 1170, 5]\nclass = b'),\n    Text(2066.666666666667, 181.19999999999982, 'gini = 0.081\nsamples = 719\nvalue = [0, 1112, 49]\nclass = b'),\n    Text(2314.666666666667, 543.5999999999999, 'CO <= 0.305\ngini = 0.41\nsamples = 265\nvalue = [1, 289, 115]\nclass = b'),\n    Text(2232.0, 181.19999999999982, 'gini = 0.504\nsamples = 147\nvalue = [1, 103, 113]\nclass = c'),\n    Text(2397.333333333335, 181.19999999999982, 'gini = 0.021\nsamples = 118\nvalue = [0, 186, 2]\nclass = b'),\n    Text(2810.666666666667, 906.0, 'NMHC <= 0.315\ngini = 0.515\nsamples = 5088\nvalue = [715, 2256, 5092]\nclass = c'),\n    Text(2645.333333333335, 543.5999999999999, 'NO_2 <= 17.39\ngini = 0.463\nsamples = 4564\nvalue = [710, 1473, 5021]\nclass = c'),\n    Text(2562.666666666667, 181.19999999999982, 'gini = 0.364\nsamples = 429\nvalue = [18, 506, 132]\nclass = b'),\n    Text(2728.0, 181.19999999999982, 'gini = 0.41\nsamples = 4135\nvalue = [692, 967, 4889]\nclass = c'),\n    Text(2976.0, 543.5999999999999, 'NOx <= 148.2\ngini = 0.162\nsamples = 524\nvalue = [5, 783, 71]\nclass = b'),\n    Text(2893.333333333335, 181.19999999999982, 'gini = 0.055\nsamples = 451\nvalue = [1, 718, 20]\nclass = b'),\n    Text(3058.666666666667, 181.19999999999982, 'gini = 0.525\nsamples = 73\nvalue = [4, 65, 51]\nclass = b'),\n    Text(3802.666666666667, 1268.4, 'MXY <= 1.865\ngini = 0.419\nsamples = 2217\nvalue = [2539, 265, 653]\nclass = a'),\n    Text(3472.0, 906.0, 'EBE <= 1.12\ngini = 0.612\nsamples = 181\nvalue = [67, 66, 147]\nclass = c'),\n    Text(3306.666666666667, 543.5999999999999, 'PXY <= 0.935\ngini = 0.495\nclasses = 77\nvalue = [62, 51, 0]\nclass = a'),\n    Text(3224.0, 181.19999999999982, 'gini = 0.416\nclasses = 63\nvalue = [62, 26, 0]\nclass = a'),\n    Text(3389.333333333335, 181.19999999999982, 'gini = 0.0\nclasses = 14\nvalue = [0, 25, 0]\nclass = b'),\n    Text(3637.333333333335, 543.5999999999999, 'TCH <= 1.795\ngini = 0.216\nclasses = 104\nvalue = [5, 15, 147]\nclass = c'),\n    Text(3554.666666666667, 181.19999999999982, 'gini = 0.117\nclasses = 94\nvalue = [5, 4, 138]\nclass = c'),\n    Text(3720.0, 181.19999999999982, 'gini = 0.495\nclasses = 10\nvalue = [0, 11, 9]\nclass = b'),\n    Text(4133.333333333334, 906.0, 'O_3 <= 9.25\ngini = 0.365\nclasses = 2036\nvalue = [2472, 199, 506]\nclass = a'),\n    Text(3968.0, 543.5999999999999, 'EBE <= 1.495\ngini = 0.571\nclasses = 616\nvalue = [545, 128, 290]\nclass = a'),\n    Text(3885.333333333335, 181.19999999999982, 'gini = 0.535\nclasses = 79\nvalue = [40, 73, 10]\nclass = b'),\n    Text(4050.666666666667, 181.19999999999982, 'gini = 0.523\nclasses = 537\nvalue = [505, 55, 280]\nclass = a'),\n    Text(4298.666666666667, 543.5999999999999, 'NMHC <= 0.235\ngini = 0.232\nclasses = 1420\nvalue = [1927, 71, 216]\nclass = a'),\n    Text(4216.0, 181.19999999999982, 'gini = 0.077\nclasses = 944\nvalue = [1404, 25, 33]\nclass = a'),\n    Text(4381.333333333334, 181.19999999999982, 'gini = 0.453\nclasses = 476\nvalue = [523, 46, 183]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.2904829269822444

Ridge Regression:0.2901722163574638

Lasso Regression:0.03832089827644303

ElasticNet Regression:0.10463204276063354

Logistic Regression:0.8951733624630821

Random Forest:0.8947460115206034

Logistic Regression is suitable for this dataset