

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"E:\154\C10_air\csvs_per_year\csvs_per_year\madrid_2007.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999
...
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000

225120 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: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      25443 non-null   object 
 1   BEN        25443 non-null   float64
 2   CO         25443 non-null   float64
 3   EBE        25443 non-null   float64
 4   MXY        25443 non-null   float64
 5   NMHC       25443 non-null   float64
 6   NO_2       25443 non-null   float64
 7   NOx        25443 non-null   float64
 8   OXY        25443 non-null   float64
 9   O_3         25443 non-null   float64
 10  PM10       25443 non-null   float64
 11  PM25       25443 non-null   float64
 12  PXY        25443 non-null   float64
 13  SO_2       25443 non-null   float64
 14  TCH         25443 non-null   float64
 15  TOL         25443 non-null   float64
 16  station    25443 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

```
In [6]: data=df[['CO' , 'station']]  
data
```

Out[6]:

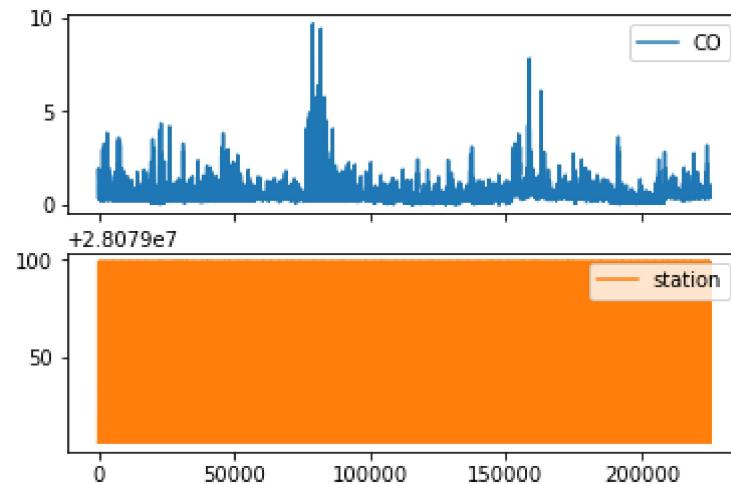
	CO	station
4	1.86	28079006
21	0.31	28079024
25	1.42	28079099
30	1.89	28079006
47	0.30	28079024
...
225073	0.47	28079006
225094	0.45	28079099
225098	0.41	28079006
225115	0.45	28079024
225119	0.40	28079099

25443 rows × 2 columns

Line chart

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

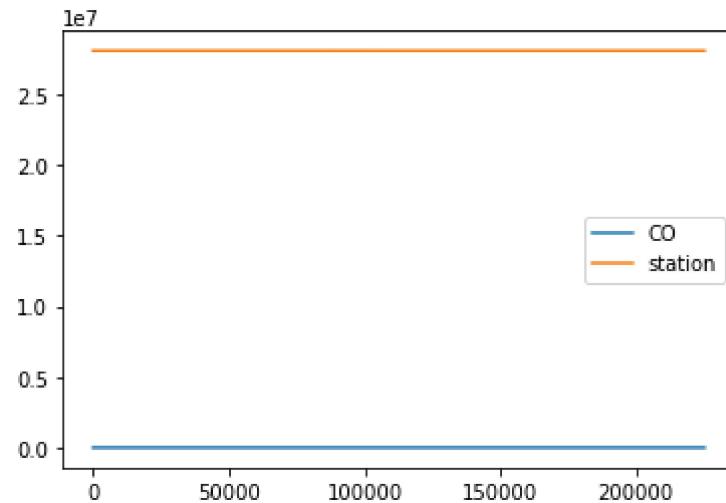
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

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

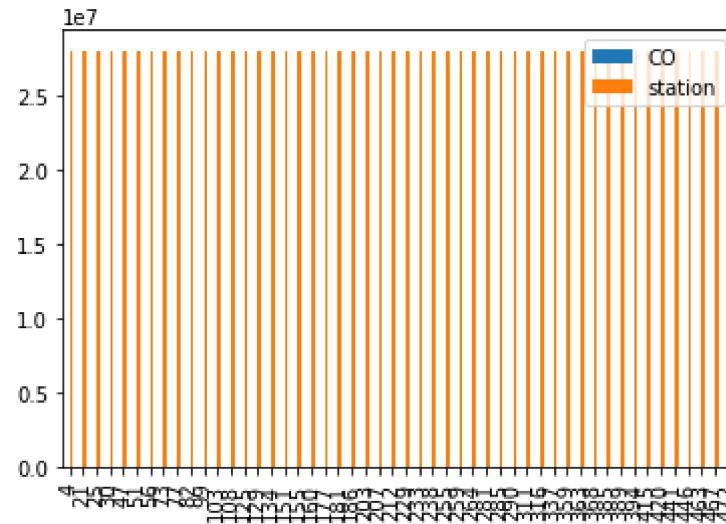


Bar chart

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

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

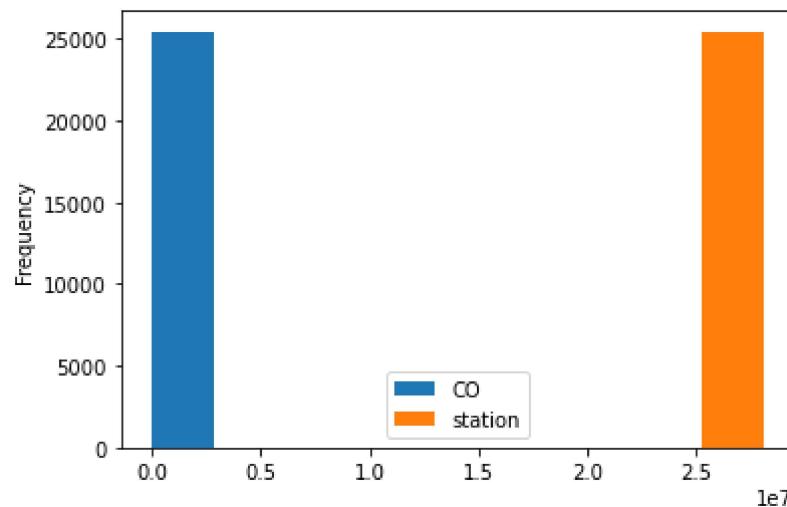
```
Out[10]: <AxesSubplot:>
```



Histogram

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

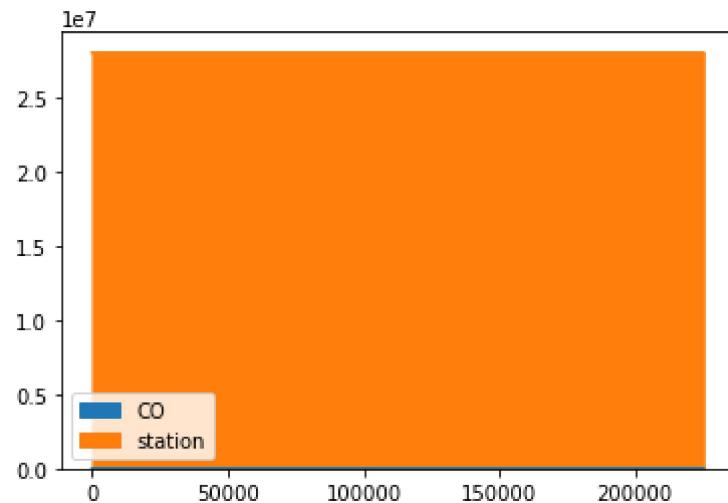
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

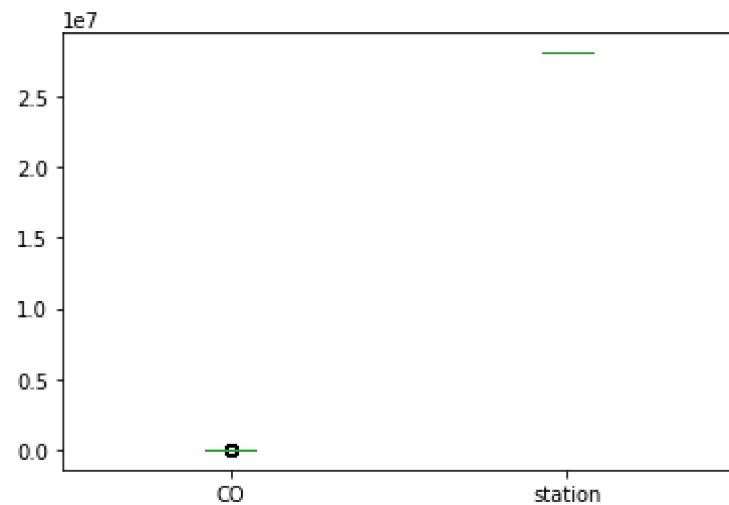
```
Out[12]: <AxesSubplot:>
```



Box chart

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

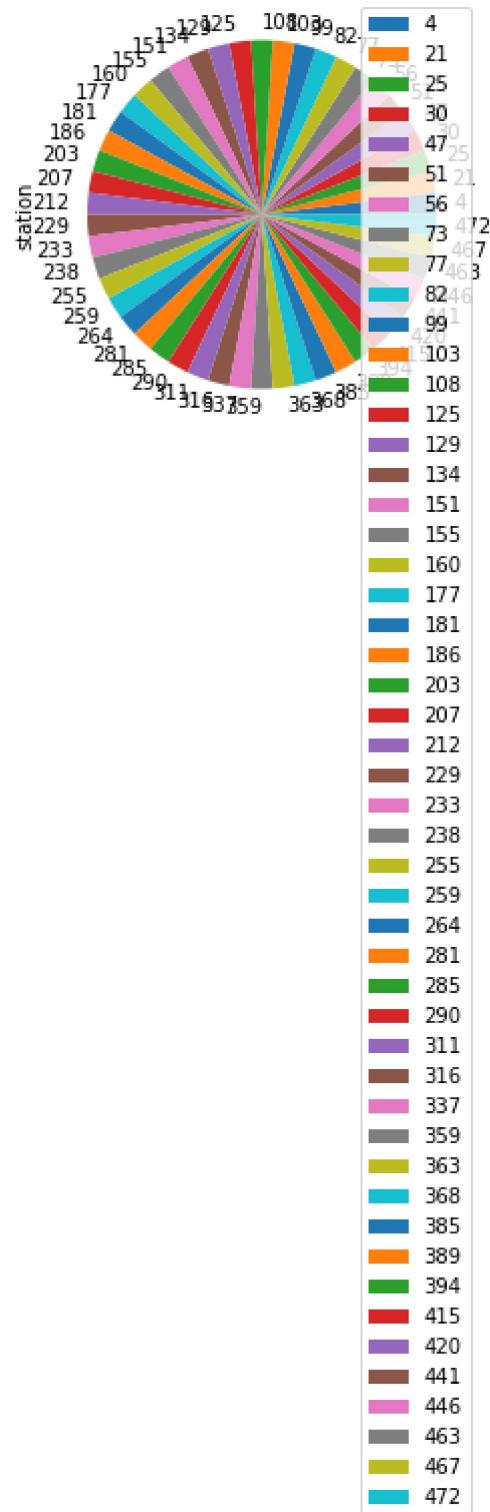
Out[13]: <AxesSubplot:>



Pie chart

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

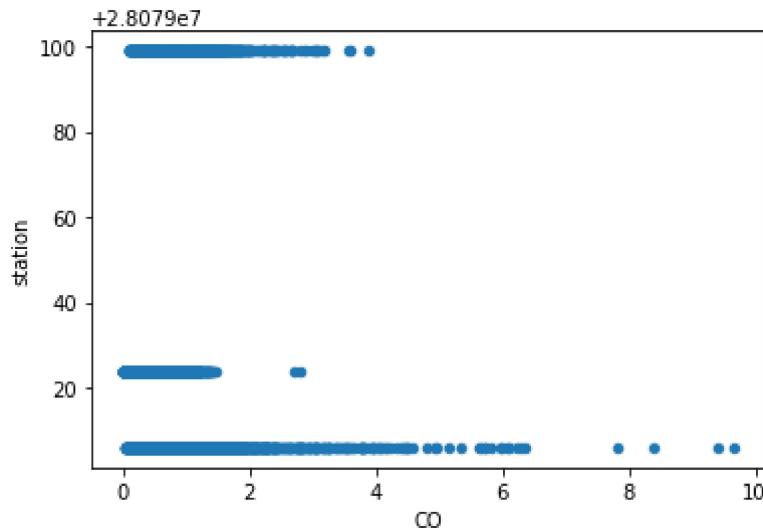
```
Out[14]: <AxesSubplot:ylabel='station'>
```



Scatter chart

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

```
Out[15]: <AxesSubplot:xlabel='CO', ylabel='station'>
```



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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      25443 non-null   object 
 1   BEN       25443 non-null   float64
 2   CO        25443 non-null   float64
 3   EBE       25443 non-null   float64
 4   MXY       25443 non-null   float64
 5   NMHC      25443 non-null   float64
 6   NO_2      25443 non-null   float64
 7   NOx       25443 non-null   float64
 8   OXY       25443 non-null   float64
 9   O_3        25443 non-null   float64
 10  PM10      25443 non-null   float64
 11  PM25      25443 non-null   float64
 12  PXY       25443 non-null   float64
 13  SO_2      25443 non-null   float64
 14  TCU       25443 non-null   float64
```

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

Out[17]:

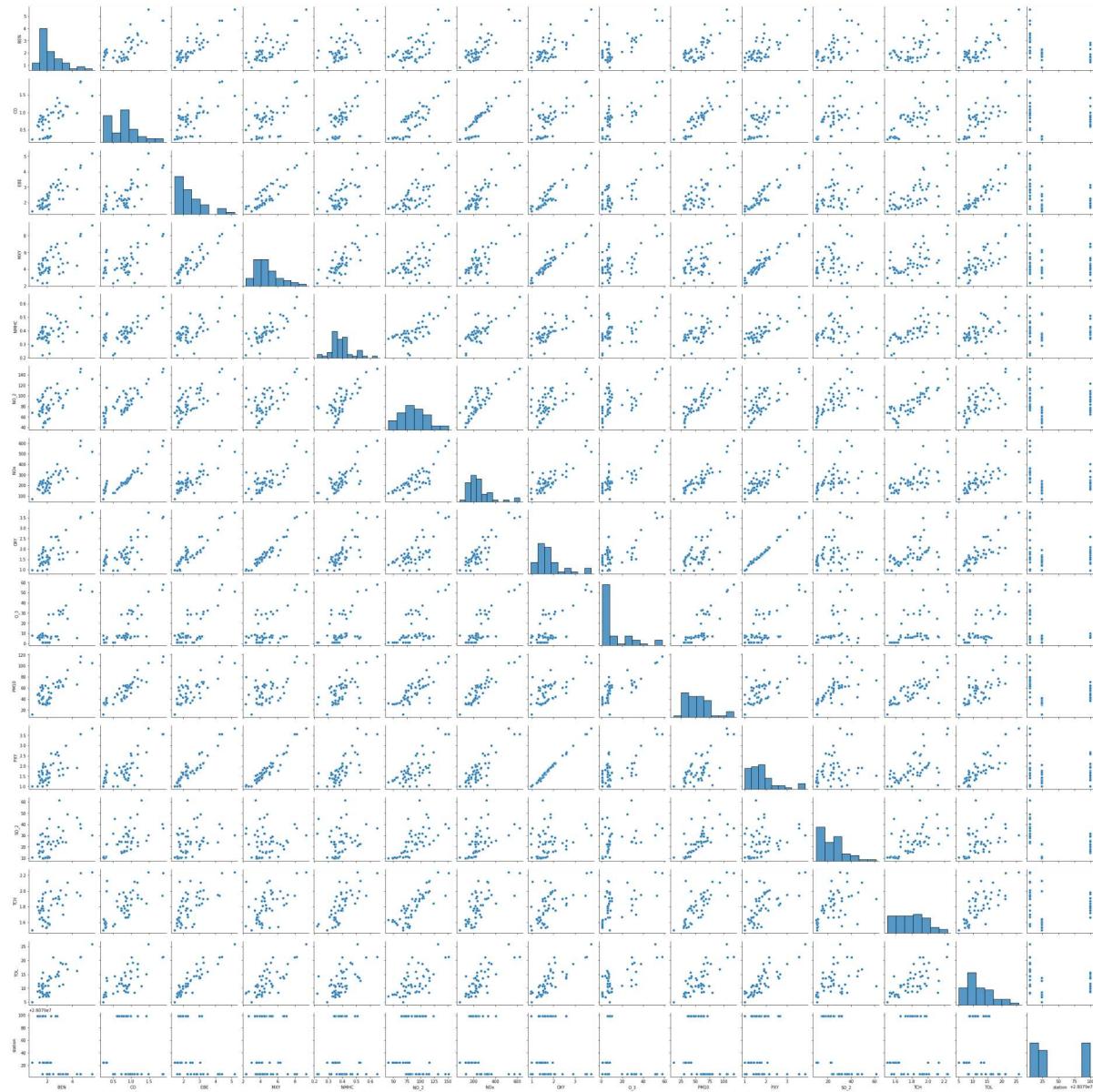
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	254
mean	1.146744	0.505120	1.394071	2.392008	0.249967	58.532683	1
std	1.278733	0.423231	1.268265	2.784302	0.142627	37.755029	1
min	0.130000	0.000000	0.120000	0.150000	0.000000	1.690000	
25%	0.450000	0.260000	0.780000	0.960000	0.160000	31.285001	
50%	0.770000	0.400000	1.000000	1.500000	0.220000	54.080002	
75%	1.390000	0.640000	1.580000	2.855000	0.300000	79.230003	1
max	30.139999	9.660000	31.680000	65.480003	2.570000	430.299988	18

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])
```

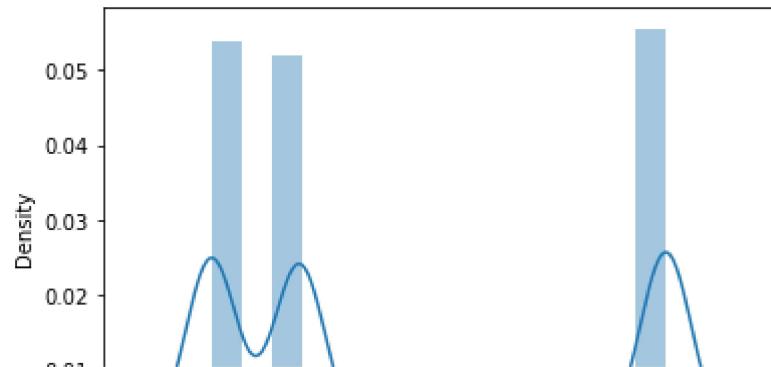
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x1f407323580>
```



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

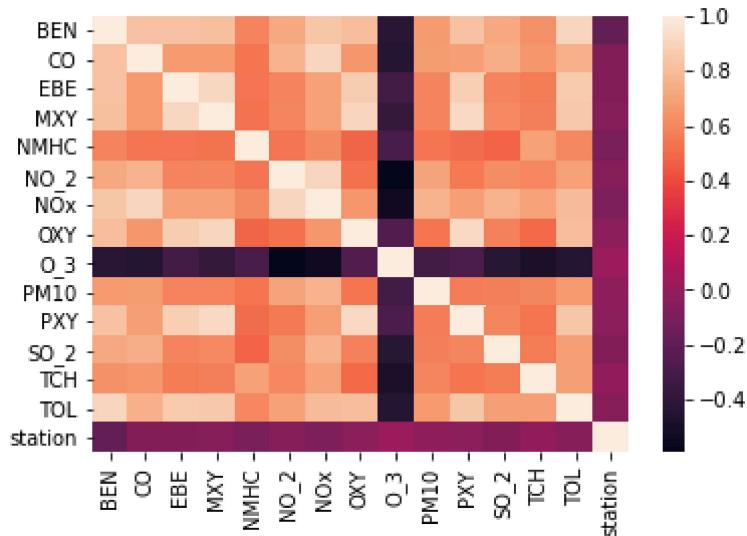
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[20]: <AxesSubplot:xlabel='station', 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]: 28079010.288099166
```

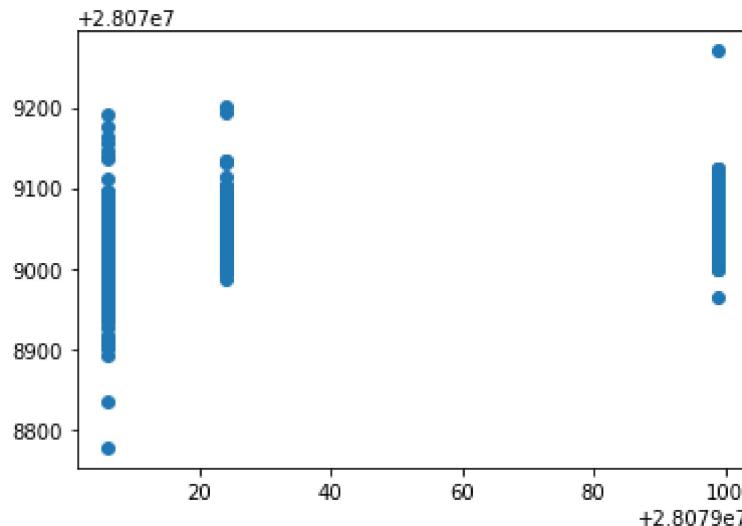
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[26]:
```

	Co-efficient
BEN	-33.138185
CO	18.778752
EBE	0.302200
MXY	-1.354572
NMHC	-43.489818
NO_2	0.121565
NOx	-0.050002
OXY	6.667793
O_3	-0.036114
PM10	0.143025
PXY	5.417814
SO_2	0.215428
TCH	25.896226
TOL	3.445289

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x1f415cabca0>
```



ACCURACY

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

```
Out[28]: 0.15767304227090928
```

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

```
Out[29]: 0.1591674479220705
```

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]: 0.1579550318661158
```

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

```
Out[33]: 0.15911093658937425
```

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

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

Accuracy(Lasso)

```
In [35]: la.score(x_train,y_train)
```

```
Out[35]: 0.013922341680993933
```

```
In [36]: la.score(x_test,y_test)
```

```
Out[36]: 0.012307155704892159
```

ElasticNet

```
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([-7.89478347,  0.          , -0.          ,  0.06040151, -0.
                  0.05123916, -0.05530601,  0.5465374 , -0.0631162 ,  0.16303496,
                  0.61876107,  0.          ,  0.          ,  1.02802182])
```

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

```
Out[39]: 28079046.637902286
```

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

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

```
Out[41]: 0.06785011580181377
```

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)))
```

36.79889504821354
1540.7890051729723
39.25288531016506

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]: (25443, 14)
```

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

```
Out[46]: (25443,)
```

```
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]: 0.8146838030106512
```

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

```
Out[54]: 1.082753977181323e-19
```

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

```
Out[55]: array([[1.08275398e-19, 1.80383815e-19, 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_
```

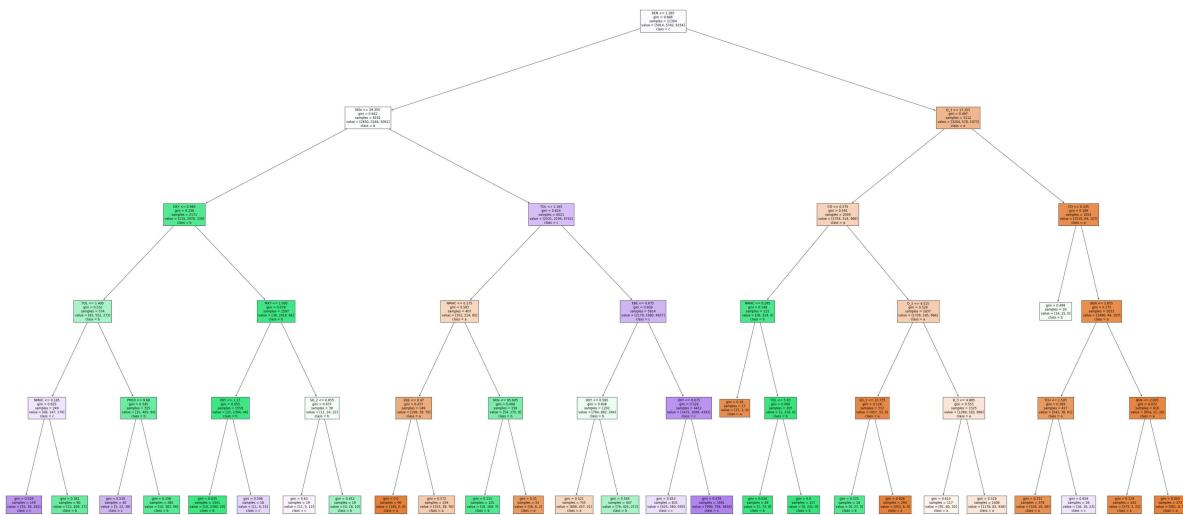
```
Out[60]: 0.8251544076361594
```

```
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(2478.807692307692, 1993.2, 'BEN <= 1.285\ngini = 0.666\nsamples = 11304
\nvalue = [5914, 5742, 6154]\nclass = c'),
Text(1373.5384615384614, 1630.8000000000002, 'NOx <= 29.355\ngini = 0.642\nsamples = 8192\nvalue = [2650, 5164, 5081]\nclass = b'),
Text(686.7692307692307, 1268.4, 'OXY <= 0.985\ngini = 0.238\nsamples = 2171
\nvalue = [119, 2970, 339]\nclass = b'),
Text(343.38461538461536, 906.0, 'TOL <= 1.495\ngini = 0.532\nsamples = 574\nvalue = [83, 552, 273]\nclass = b'),
Text(171.69230769230768, 543.5999999999999, 'NMHC <= 0.185\ngini = 0.625\nsamples = 249\nvalue = [68, 147, 179]\nclass = c'),
Text(85.84615384615384, 181.1999999999982, 'gini = 0.528\nsamples = 159\nvalue = [55, 38, 162]\nclass = c'),
Text(257.53846153846155, 181.1999999999982, 'gini = 0.361\nsamples = 90\nvalue = [13, 109, 17]\nclass = b'),
Text(515.0769230769231, 543.5999999999999, 'PM10 <= 9.68\ngini = 0.345\nsamples = 325\nvalue = [15, 405, 94]\nclass = b'),
Text(429.23076923076917, 181.1999999999982, 'gini = 0.538\nsamples = 40\nvalue = [5, 22, 38]\nclass = c'),
Text(600.9230769230769, 181.1999999999982, 'gini = 0.256\nsamples = 285\nvalue = [10, 383, 56]\nclass = b'),
Text(1030.1538461538462, 906.0, 'MXY <= 1.095\ngini = 0.078\nsamples = 1597
\nvalue = [36, 2418, 66]\nclass = b'),
Text(858.4615384615383, 543.5999999999999, 'OXY <= 1.13\ngini = 0.055\nsamples = 1559\nvalue = [25, 2394, 44]\nclass = b'),
Text(772.6153846153845, 181.1999999999982, 'gini = 0.035\nsamples = 1541\nvalue = [14, 2390, 29]\nclass = b'),
Text(944.3076923076923, 181.1999999999982, 'gini = 0.598\nsamples = 18\nvalue = [11, 4, 15]\nclass = c'),
Text(1201.8461538461538, 543.5999999999999, 'SO_2 <= 6.655\ngini = 0.637\nsamples = 38\nvalue = [11, 24, 22]\nclass = b'),
Text(1116.0, 181.1999999999982, 'gini = 0.63\nsamples = 19\nvalue = [11, 5,
12]\nclass = c'),
Text(1287.6923076923076, 181.1999999999982, 'gini = 0.452\nsamples = 19\nvalue = [0, 19, 10]\nclass = b'),
Text(2060.3076923076924, 1268.4, 'TOL <= 1.165\ngini = 0.624\nsamples = 6021
\nvalue = [2531, 2194, 4742]\nclass = c'),
Text(1716.9230769230767, 906.0, 'NMHC <= 0.175\ngini = 0.583\nsamples = 407
\nvalue = [352, 214, 85]\nclass = a'),
Text(1545.230769230769, 543.5999999999999, 'EBE <= 0.47\ngini = 0.437\nsamples = 249\nvalue = [298, 39, 76]\nclass = a'),
Text(1459.3846153846152, 181.1999999999982, 'gini = 0.0\nsamples = 90\nvalue = [145, 0, 0]\nclass = a'),
Text(1631.0769230769229, 181.1999999999982, 'gini = 0.572\nsamples = 159\nvalue = [153, 39, 76]\nclass = a'),
Text(1888.6153846153845, 543.5999999999999, 'NOx <= 95.695\ngini = 0.406\nsamples = 158\nvalue = [54, 175, 9]\nclass = b'),
Text(1802.7692307692307, 181.1999999999982, 'gini = 0.231\nsamples = 125\nvalue = [18, 169, 7]\nclass = b'),
Text(1974.4615384615383, 181.1999999999982, 'gini = 0.31\nsamples = 33\nvalue = [36, 6, 2]\nclass = a'),
Text(2403.6923076923076, 906.0, 'EBE <= 0.675\ngini = 0.609\nsamples = 5614
\nvalue = [2179, 1980, 4657]\nclass = c'),
Text(2232.0, 543.5999999999999, 'OXY <= 0.585\ngini = 0.608\nsamples = 1202
\nvalue = [764, 882, 264]\nclass = b'),
Text(2146.153846153846, 181.1999999999982, 'gini = 0.521\nsamples = 755\nvalue = [688, 457, 51]\nclass = a'),
Text(2317.846153846154, 181.1999999999982, 'gini = 0.545\nsamples = 447\nvalue = [119, 2970, 339]\nclass = b')]
```

```
lue = [76, 425, 213]\nclass = b'),  
    Text(2575.3846153846152, 543.5999999999999, 'OXY <= 0.675\ngini = 0.528\nsamples = 4412\nvalue = [1415, 1098, 4393]\nclass = c'),  
    Text(2489.5384615384614, 181.1999999999982, 'gini = 0.653\nsamples = 831\nvalue = [425, 340, 559]\nclass = c'),  
    Text(2661.230769230769, 181.1999999999982, 'gini = 0.478\nsamples = 3581\nvalue = [990, 758, 3834]\nclass = c'),  
    Text(3584.076923076923, 1630.8000000000002, 'O_3 <= 17.355\ngini = 0.497\nsamples = 3112\nvalue = [3264, 578, 1073]\nclass = a'),  
    Text(3133.3846153846152, 1268.4, 'CO <= 0.375\ngini = 0.591\nsamples = 2059\nvalue = [1754, 514, 966]\nclass = a'),  
    Text(2832.9230769230767, 906.0, 'NMHC <= 0.285\ngini = 0.148\nsamples = 222\nvalue = [28, 319, 0]\nclass = b'),  
    Text(2747.076923076923, 543.5999999999999, 'gini = 0.18\nsamples = 17\nvalue = [27, 3, 0]\nclass = a'),  
    Text(2918.7692307692305, 543.5999999999999, 'TOL <= 5.63\ngini = 0.006\nsamples = 205\nvalue = [1, 316, 0]\nclass = b'),  
    Text(2832.9230769230767, 181.1999999999982, 'gini = 0.026\nsamples = 48\nvalue = [1, 74, 0]\nclass = b'),  
    Text(3004.6153846153843, 181.1999999999982, 'gini = 0.0\nsamples = 157\nvalue = [0, 242, 0]\nclass = b'),  
    Text(3433.8461538461534, 906.0, 'O_3 <= 4.515\ngini = 0.526\nsamples = 1837\nvalue = [1726, 195, 966]\nclass = a'),  
    Text(3262.1538461538457, 543.5999999999999, 'SO_2 <= 10.735\ngini = 0.126\nsamples = 312\nvalue = [457, 33, 0]\nclass = a'),  
    Text(3176.307692307692, 181.1999999999982, 'gini = 0.225\nsamples = 18\nvalue = [4, 27, 0]\nclass = b'),  
    Text(3347.999999999995, 181.1999999999982, 'gini = 0.026\nsamples = 294\nvalue = [453, 6, 0]\nclass = a'),  
    Text(3605.5384615384614, 543.5999999999999, 'O_3 <= 4.885\ngini = 0.553\nsamples = 1525\nvalue = [1269, 162, 966]\nclass = a'),  
    Text(3519.6923076923076, 181.1999999999982, 'gini = 0.614\nsamples = 117\nvalue = [91, 80, 30]\nclass = a'),  
    Text(3691.3846153846152, 181.1999999999982, 'gini = 0.529\nsamples = 1408\nvalue = [1178, 82, 936]\nclass = a'),  
    Text(4034.7692307692305, 1268.4, 'CO <= 0.245\ngini = 0.188\nsamples = 1053\nvalue = [1510, 64, 107]\nclass = a'),  
    Text(3948.9230769230767, 906.0, 'gini = 0.499\nsamples = 20\nvalue = [14, 15, 0]\nclass = b'),  
    Text(4120.615384615385, 906.0, 'BEN <= 1.655\ngini = 0.175\nsamples = 1033\nvalue = [1496, 49, 107]\nclass = a'),  
    Text(3948.9230769230767, 543.5999999999999, 'TCH <= 1.505\ngini = 0.309\nsamples = 417\nvalue = [542, 38, 81]\nclass = a'),  
    Text(3863.076923076923, 181.1999999999982, 'gini = 0.231\nsamples = 379\nvalue = [526, 20, 58]\nclass = a'),  
    Text(4034.7692307692305, 181.1999999999982, 'gini = 0.659\nsamples = 38\nvalue = [16, 18, 23]\nclass = c'),  
    Text(4292.307692307692, 543.5999999999999, 'BEN <= 2.005\ngini = 0.072\nsamples = 616\nvalue = [954, 11, 26]\nclass = a'),  
    Text(4206.461538461538, 181.1999999999982, 'gini = 0.114\nsamples = 243\nvalue = [373, 3, 21]\nclass = a'),  
    Text(4378.153846153846, 181.1999999999982, 'gini = 0.043\nsamples = 373\nvalue = [581, 8, 5]\nclass = a')]
```



Accuracy

Linear Regression

In [63]: `lr.score(x_train,y_train)`

Out[63]: 0.1591674479220705

Ridge Regression

In [64]: `rr.score(x_train,y_train)`

Out[64]: 0.15911093658937425

Lasso Regression

In [65]: `la.score(x_test,y_test)`

Out[65]: 0.012307155704892159

ElasticNet Regression

In [66]: `en.score(x_test,y_test)`

Out[66]: 0.06785011580181377

Logistic Regression

```
In [67]: logr.score(fs,target_vector)
```

```
Out[67]: 0.8146838030106512
```

Random Forest

```
In [68]: grid_search.best_score_
```

```
Out[68]: 0.8251544076361594
```

Conclusion

Random Forest is suitable for this dataset