20104016

DEENA

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("madrid_2010.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.

209448 rows × 17 columns

Data Cleaning and Data Preprocessing

In [3]:
```
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: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     6666 non-null   object 
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
 14  TCH      6666 non-null   float64
 15  TOL      6666 non-null   float64
 16  station  6666 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 937.4+ KB
```

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

Out[6]:

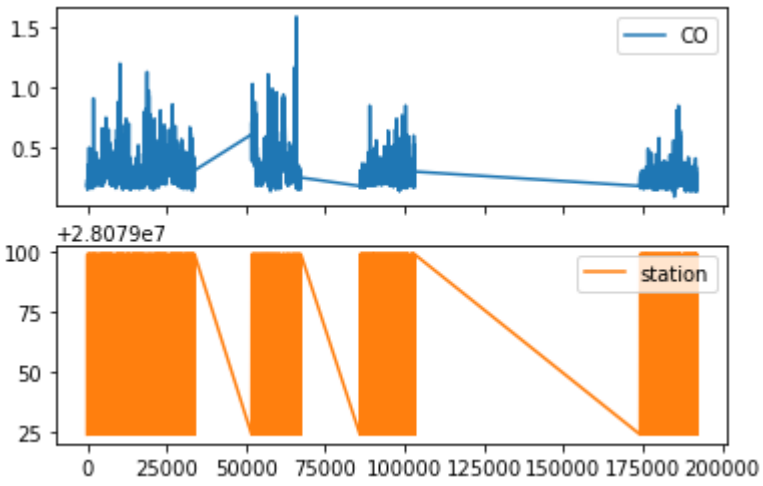
	CO	station
11	0.18	28079024
23	0.23	28079099
35	0.17	28079024
47	0.21	28079099
59	0.16	28079024
...	...	...
191879	0.26	28079099
191891	0.16	28079024
191903	0.28	28079099
191915	0.16	28079024
191927	0.25	28079099

6666 rows × 2 columns

Line chart

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

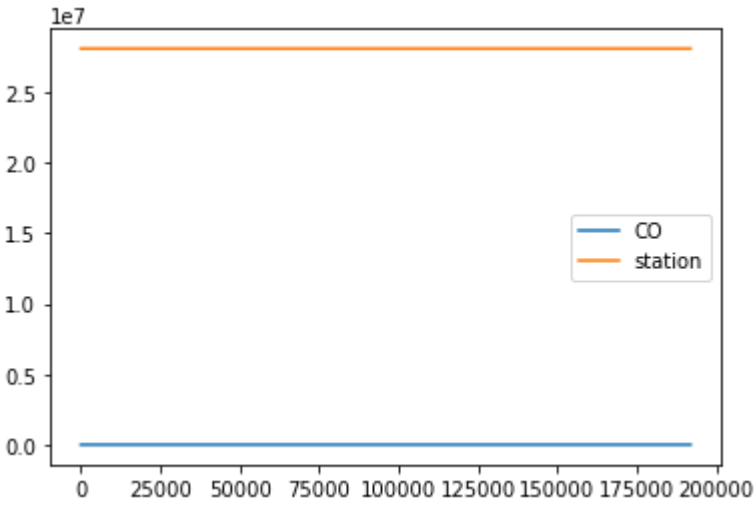
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

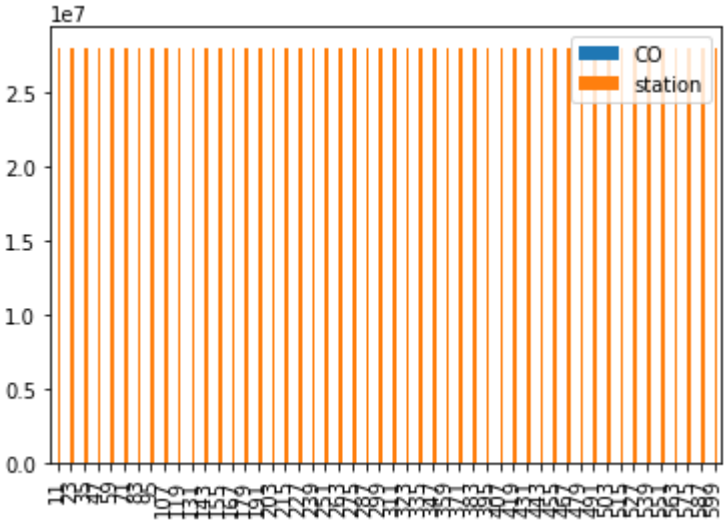


Bar chart

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

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

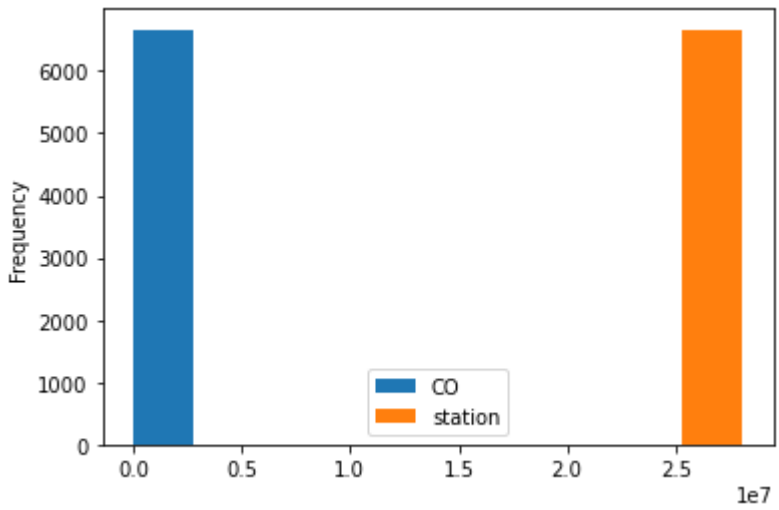
Out[10]: <AxesSubplot:>



Histogram

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

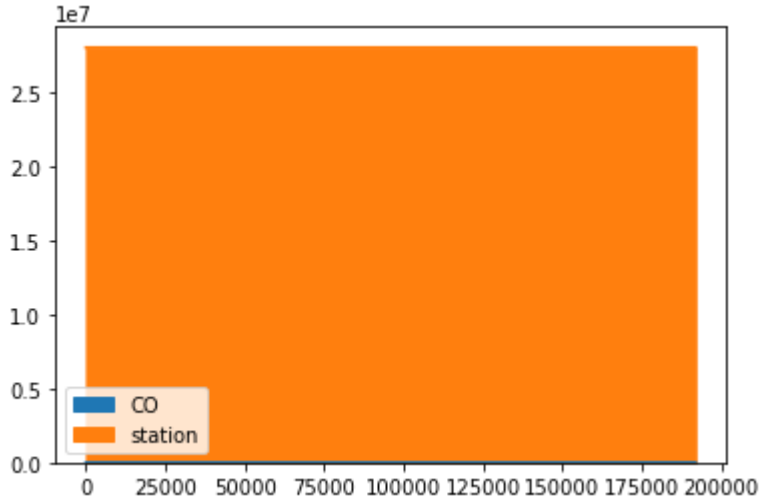
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

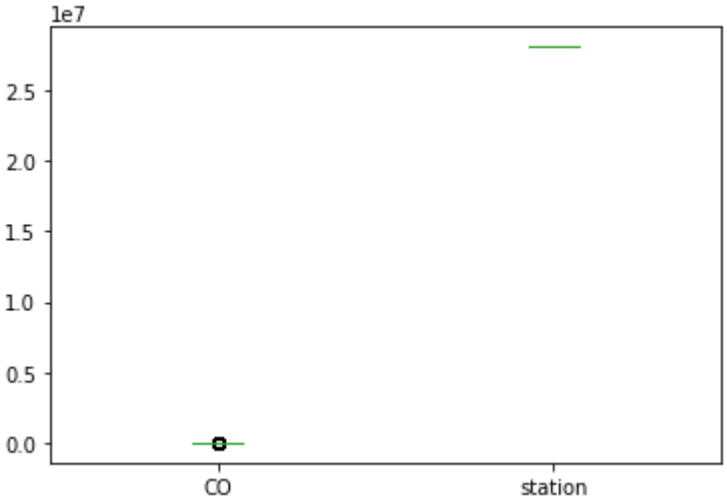
Out[12]: <AxesSubplot:>



Box chart

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

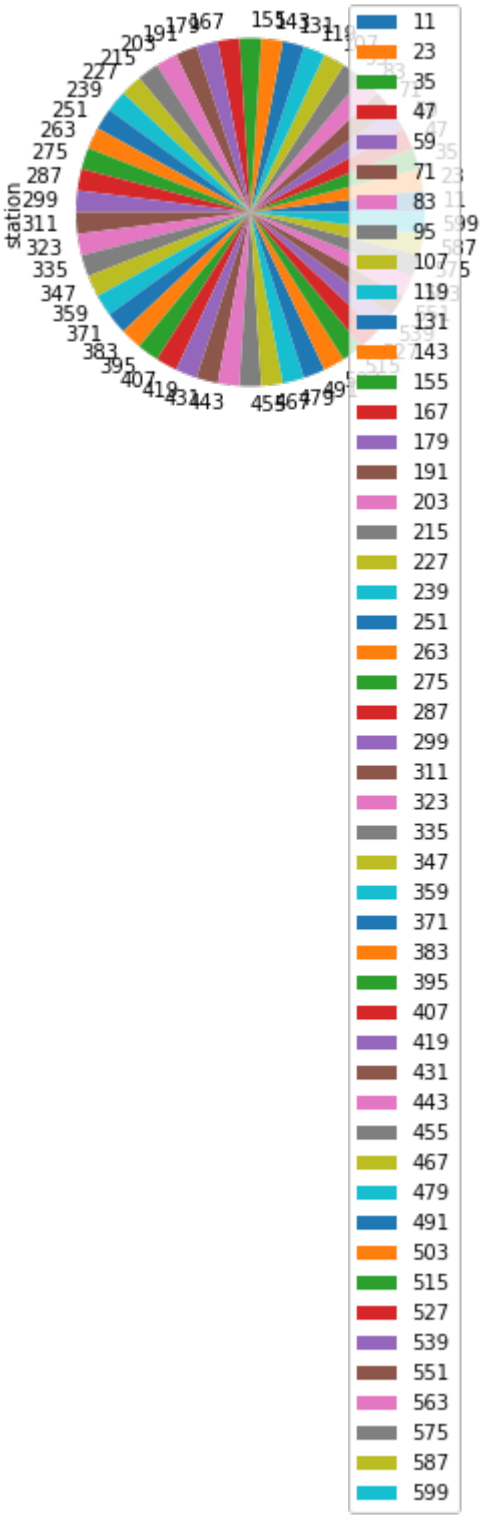
Out[13]: <AxesSubplot:>



Pie chart

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

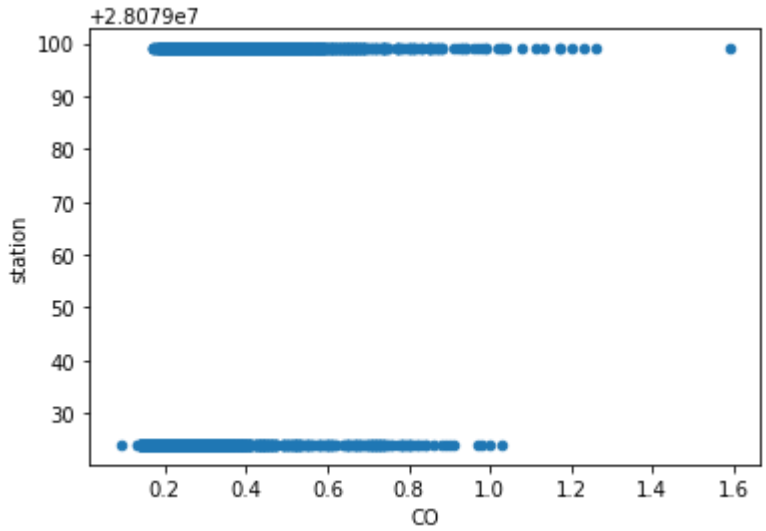
Out[14]: <AxesSubplot:ylabel='station'>



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: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     6666 non-null   object 
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
 14  TCH      6666 non-null   float64
```

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

Out[17]:

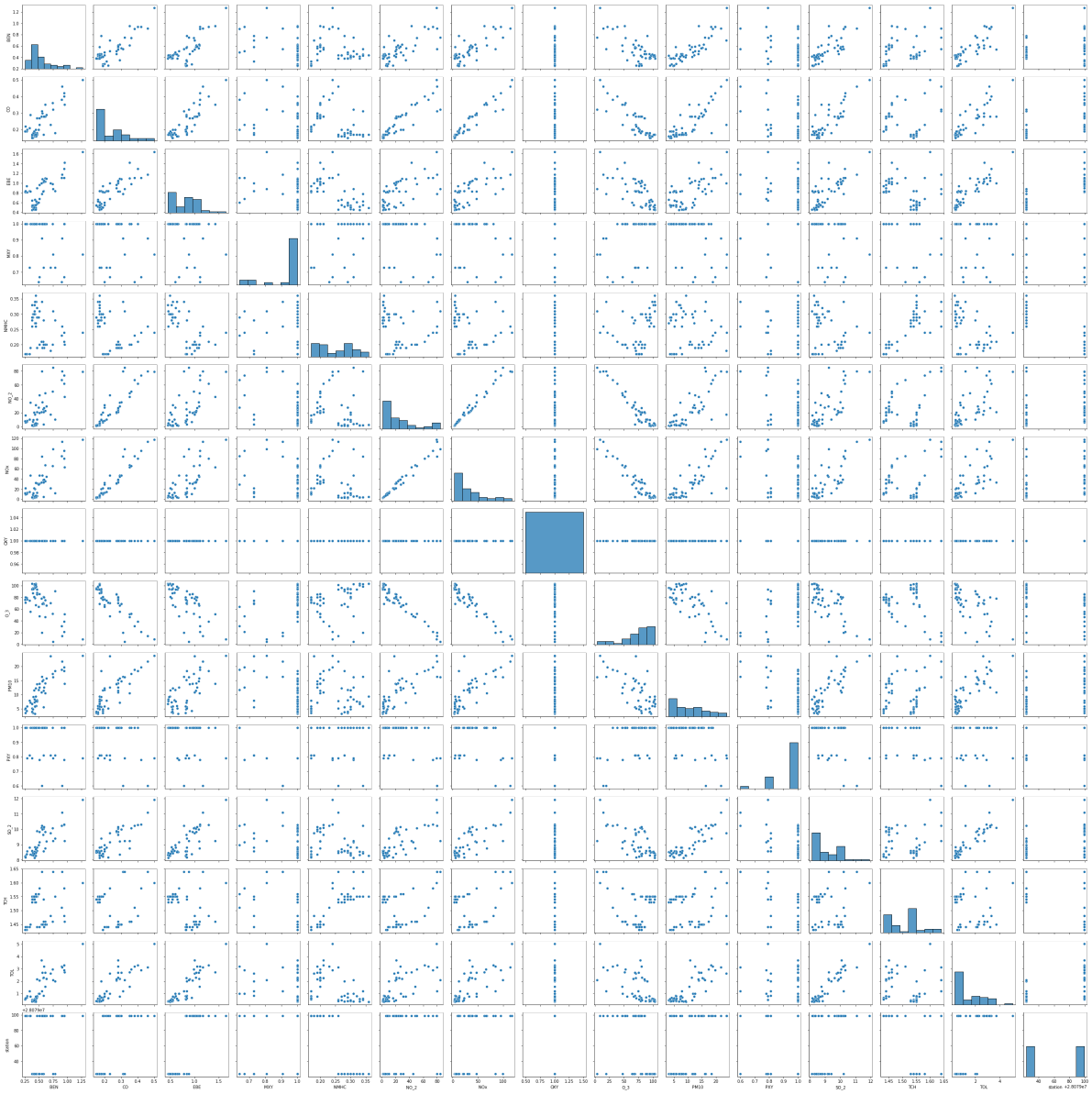
	BEN	CO	EBE	MXY	NMHC	NO_2	NO
count	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.0000
mean	0.648425	0.296280	0.840585	0.839959	0.243378	33.888744	47.5406
std	0.395346	0.133296	0.508031	0.382263	0.115730	23.465169	41.2305
min	0.170000	0.090000	0.140000	0.110000	0.000000	1.290000	2.7600
25%	0.380000	0.200000	0.470000	0.590000	0.180000	15.752500	19.4425
50%	0.540000	0.260000	0.755000	1.000000	0.220000	29.320000	36.7700
75%	0.810000	0.340000	1.000000	1.000000	0.280000	47.657500	62.1025
max	5.110000	1.590000	5.190000	6.810000	0.930000	133.399994	409.2999

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])
```

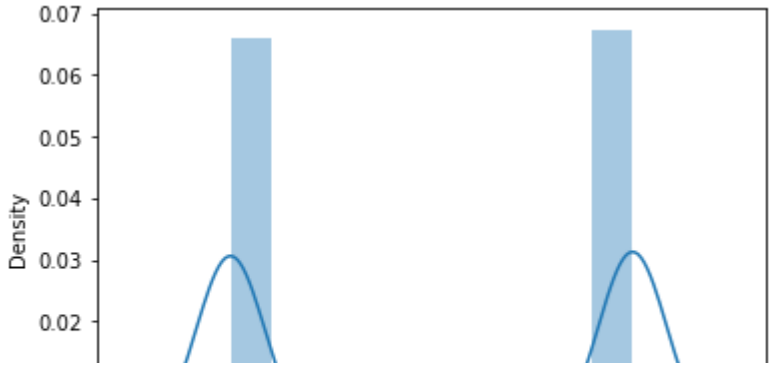
Out[19]: <seaborn.axisgrid.PairGrid at 0x218558e6670>



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

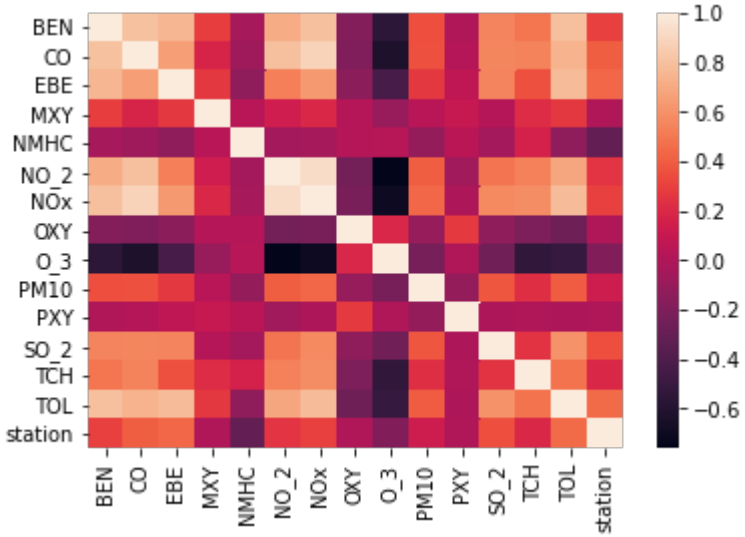
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  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 BULDING

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]:
```
28078926.87889828
```

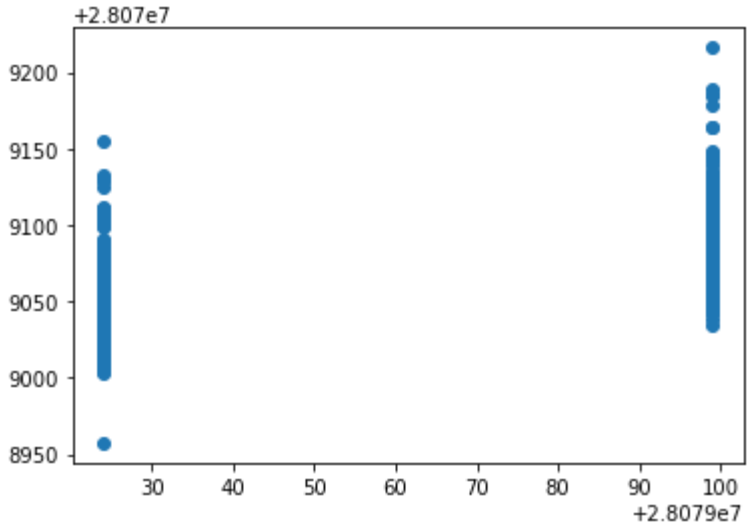
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-36.541075
CO	161.371563
EBE	17.746888
MXY	-9.031173
NMHC	-73.909469
NO_2	0.332682
NOx	-0.617959
OXY	31.328833
O_3	0.092290
PM10	-0.161082
PXY	-4.084014
SO_2	1.888577
TCH	49.705917
TOL	9.849776

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

Out[27]: <matplotlib.collections.PathCollection at 0x21863924850>



ACCURACY

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

Out[28]: 0.4246550732299478

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

Out[29]: 0.4285982294479104

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]: 0.4089393977094645

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

Out[33]: 0.41714395559857365

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

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

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

Out[35]:
```
0.18165357519751268
```

Accuracy(Lasso)

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

Out[36]:
```
0.18496029059812846
```

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([-0.        ,  0.18406252,  2.79574067, -1.09197465, -1.2565913 ,
        0.03448127, -0.12292726,  0.53708216, -0.02801595, -0.1107858 ,
        0.        ,  2.5902298 ,  0.        ,  7.1658922 ])
```

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

Out[39]:
```
28079026.13096261
```

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

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

Out[41]:
```
0.2373867327042073
```

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)))
```

```
30.735722763402386
1072.2704779480814
32.74554134455684
```

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
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df['station']
```

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

Out[45]: (6666, 14)

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

Out[46]: (6666,)

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([28079024, 28079099], dtype=int64)

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

Out[53]: 0.8660366036603661

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

Out[54]: 0.0

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

Out[55]: array([[0., 1.]])

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="ac
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]:
```
0.9309901414487785
```

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

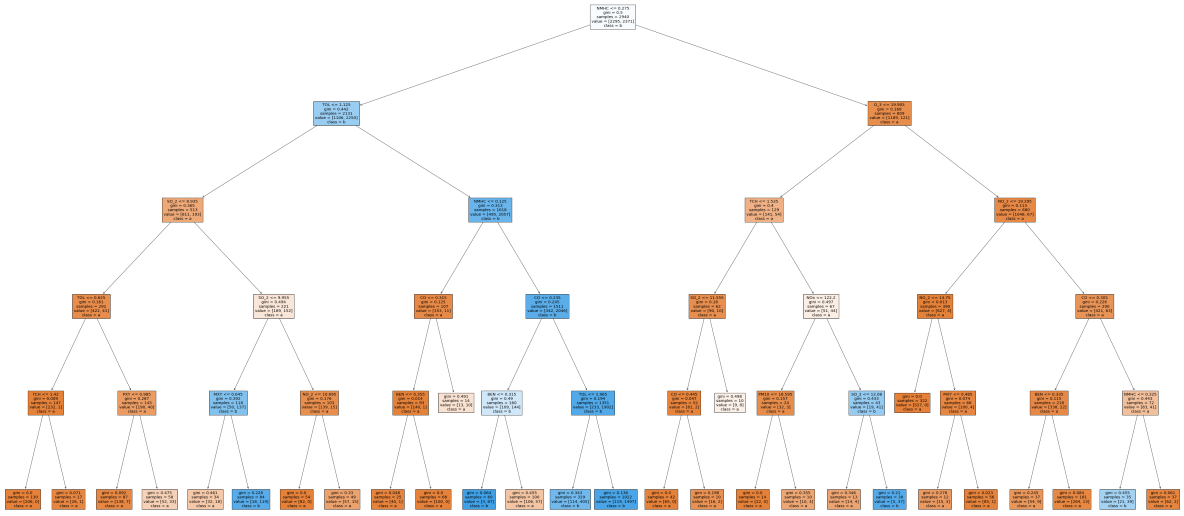
In [62]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
```

Out[62]:
```
[Text(2307.1153846153843, 1993.2, 'NMHC <= 0.275\ngini = 0.5\nsamples = 2940\
nvalue = [2295, 2371]\nclass = b'),
 Text(1266.230769230769, 1630.8000000000002, 'TOL <= 1.125\ngini = 0.442\nsam
ples = 2131\nvalue = [1106, 2250]\nclass = b'),
 Text(686.7692307692307, 1268.4, 'SO_2 <= 8.935\ngini = 0.365\nsamples = 513\
nvalue = [611, 193]\nclass = a'),
 Text(343.38461538461536, 906.0, 'TOL <= 0.625\ngini = 0.161\nsamples = 292\n
value = [422, 41]\nclass = a'),
 Text(171.69230769230768, 543.5999999999999, 'TCH <= 1.42\ngini = 0.009\nsamp
les = 147\nvalue = [232, 1]\nclass = a'),
 Text(85.84615384615384, 181.19999999999982, 'gini = 0.0\nsamples = 130\nvalu
e = [206, 0]\nclass = a'),
 Text(257.53846153846155, 181.19999999999982, 'gini = 0.071\nsamples = 17\nva
lue = [26, 1]\nclass = a'),
 Text(515.0769230769231, 543.5999999999999, 'PXY <= 0.985\ngini = 0.287\nsamp
les = 145\nvalue = [190, 40]\nclass = a'),
 Text(429.23076923076917, 181.19999999999982, 'gini = 0.092\nsamples = 87\nva
lue = [138, 7]\nclass = a'),
 Text(600.9230769230769, 181.19999999999982, 'gini = 0.475\nsamples = 58\nval
ue = [52, 33]\nclass = a'),
 Text(1030.1538461538462, 906.0, 'SO_2 <= 9.955\ngini = 0.494\nsamples = 221\
nvalue = [189, 152]\nclass = a'),
 Text(858.4615384615383, 543.5999999999999, 'MXY <= 0.645\ngini = 0.392\nsamp
les = 118\nvalue = [50, 137]\nclass = b'),
 Text(772.6153846153845, 181.19999999999982, 'gini = 0.461\nsamples = 34\nval
ue = [32, 18]\nclass = a'),
 Text(944.3076923076923, 181.19999999999982, 'gini = 0.228\nsamples = 84\nval
ue = [18, 119]\nclass = b'),
 Text(1201.8461538461538, 543.5999999999999, 'NO_2 <= 10.695\ngini = 0.176\ns
amples = 103\nvalue = [139, 15]\nclass = a'),
 Text(1116.0, 181.19999999999982, 'gini = 0.0\nsamples = 54\nvalue = [82, 0]\
nclass = a'),
 Text(1287.6923076923076, 181.19999999999982, 'gini = 0.33\nsamples = 49\nval
ue = [57, 15]\nclass = a'),
 Text(1845.6923076923076, 1268.4, 'NMHC <= 0.125\ngini = 0.313\nsamples = 161
8\nvalue = [495, 2057]\nclass = b'),
 Text(1631.0769230769229, 906.0, 'CO <= 0.315\ngini = 0.125\nsamples = 107\nv
alue = [153, 11]\nclass = a'),
 Text(1545.230769230769, 543.5999999999999, 'BEN <= 0.355\ngini = 0.014\nsamp
les = 93\nvalue = [140, 1]\nclass = a'),
 Text(1459.3846153846152, 181.19999999999982, 'gini = 0.048\nsamples = 25\nva
lue = [40, 1]\nclass = a'),
 Text(1631.0769230769229, 181.19999999999982, 'gini = 0.0\nsamples = 68\nvalu
e = [100, 0]\nclass = a'),
 Text(1716.9230769230767, 543.5999999999999, 'gini = 0.491\nsamples = 14\nval
ue = [13, 10]\nclass = a'),
 Text(2060.3076923076924, 906.0, 'CO <= 0.235\ngini = 0.245\nsamples = 1511\n
value = [342, 2046]\nclass = b'),
 Text(1888.6153846153845, 543.5999999999999, 'BEN <= 0.315\ngini = 0.49\nsamp
les = 160\nvalue = [109, 144]\nclass = b'),
 Text(1802.7692307692307, 181.19999999999982, 'gini = 0.064\nsamples = 60\nva
```

```
lue = [3, 87]\nclass = b'),
 Text(1974.4615384615383, 181.19999999999982, 'gini = 0.455\nsamples = 100\nv
alue = [106, 57]\nclass = a'),
 Text(2232.0, 543.5999999999999, 'TOL <= 1.965\ngini = 0.194\nsamples = 1351\
nvalue = [233, 1902]\nclass = b'),
 Text(2146.153846153846, 181.19999999999982, 'gini = 0.343\nsamples = 329\nva
lue = [114, 405]\nclass = b'),
 Text(2317.846153846154, 181.19999999999982, 'gini = 0.136\nsamples = 1022\nv
alue = [119, 1497]\nclass = b'),
 Text(3347.9999999999995, 1630.8000000000002, 'O_3 <= 19.905\ngini = 0.168\ns
amples = 809\nvalue = [1189, 121]\nclass = a'),
 Text(2875.846153846154, 1268.4, 'TCH <= 1.535\ngini = 0.4\nsamples = 129\nva
lue = [141, 54]\nclass = a'),
 Text(2661.230769230769, 906.0, 'SO_2 <= 11.555\ngini = 0.18\nsamples = 62\nv
alue = [90, 10]\nclass = a'),
 Text(2575.3846153846152, 543.5999999999999, 'CO <= 0.445\ngini = 0.047\nsamp
les = 52\nvalue = [81, 2]\nclass = a'),
 Text(2489.5384615384614, 181.19999999999982, 'gini = 0.0\nsamples = 42\nvalu
e = [65, 0]\nclass = a'),
 Text(2661.230769230769, 181.19999999999982, 'gini = 0.198\nsamples = 10\nval
ue = [16, 2]\nclass = a'),
 Text(2747.076923076923, 543.5999999999999, 'gini = 0.498\nsamples = 10\nvalu
e = [9, 8]\nclass = a'),
 Text(3090.461538461538, 906.0, 'NOx <= 122.2\ngini = 0.497\nsamples = 67\nva
lue = [51, 44]\nclass = a'),
 Text(2918.7692307692305, 543.5999999999999, 'PM10 <= 18.595\ngini = 0.157\ns
amples = 24\nvalue = [32, 3]\nclass = a'),
 Text(2832.9230769230767, 181.19999999999982, 'gini = 0.0\nsamples = 14\nvalu
e = [22, 0]\nclass = a'),
 Text(3004.6153846153843, 181.19999999999982, 'gini = 0.355\nsamples = 10\nva
lue = [10, 3]\nclass = a'),
 Text(3262.1538461538457, 543.5999999999999, 'SO_2 <= 12.08\ngini = 0.433\nsa
mples = 43\nvalue = [19, 41]\nclass = b'),
 Text(3176.307692307692, 181.19999999999982, 'gini = 0.346\nsamples = 13\nval
ue = [14, 4]\nclass = a'),
 Text(3347.9999999999995, 181.19999999999982, 'gini = 0.21\nsamples = 30\nval
ue = [5, 37]\nclass = b'),
 Text(3820.1538461538457, 1268.4, 'NO_2 <= 19.295\ngini = 0.113\nsamples = 68
0\nvalue = [1048, 67]\nclass = a'),
 Text(3519.6923076923076, 906.0, 'NO_2 <= 14.75\ngini = 0.013\nsamples = 390\
nvalue = [627, 4]\nclass = a'),
 Text(3433.8461538461534, 543.5999999999999, 'gini = 0.0\nsamples = 322\nvalu
e = [527, 0]\nclass = a'),
 Text(3605.5384615384614, 543.5999999999999, 'MXY <= 0.485\ngini = 0.074\nsam
ples = 68\nvalue = [100, 4]\nclass = a'),
 Text(3519.6923076923076, 181.19999999999982, 'gini = 0.278\nsamples = 12\nva
lue = [15, 3]\nclass = a'),
 Text(3691.3846153846152, 181.19999999999982, 'gini = 0.023\nsamples = 56\nva
lue = [85, 1]\nclass = a'),
 Text(4120.615384615385, 906.0, 'CO <= 0.305\ngini = 0.226\nsamples = 290\nva
lue = [421, 63]\nclass = a'),
 Text(3948.9230769230767, 543.5999999999999, 'BEN <= 0.335\ngini = 0.115\nsam
ples = 218\nvalue = [338, 22]\nclass = a'),
 Text(3863.076923076923, 181.19999999999982, 'gini = 0.245\nsamples = 37\nval
ue = [54, 9]\nclass = a'),
 Text(4034.7692307692305, 181.19999999999982, 'gini = 0.084\nsamples = 181\nv
```

```
alue = [284, 13]\nclass = a'),
 Text(4292.307692307692, 543.5999999999999, 'NMHC <= 0.325\ngini = 0.443\nsam
ples = 72\nvalue = [83, 41]\nclass = a'),
 Text(4206.461538461538, 181.19999999999982, 'gini = 0.455\nsamples = 35\nval
ue = [21, 39]\nclass = b'),
 Text(4378.153846153846, 181.19999999999982, 'gini = 0.061\nsamples = 37\nval
```



Conclusion

Accuracy

Linear Regression :0.4285982294479104

Ridge Regression :0.18165357519751268

Lasso Regression : 0.18496029059812846

ElasticNet Regression :0.2373867327042073

Logistic Regression 0.8660366036603661

Random Forest :0.9309901414487785

Random Forest is suitable for this dataset