Vijay(P1) 03.08.2023

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

In [8]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2001.csv")
df
```

Out[8]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	T
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.340000	NaN	N
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.110000	1.24	10
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.850000	NaN	N
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.460000	NaN	N
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.800000	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.880001	NaN	39.910000	NaN	28
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.809999	NaN	13.450000	1.32	16
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14.700000	1.40	1
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39.919998	NaN	20
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95	27.340000	1.41	22

217872 rows × 16 columns

In [10]:
```
df=df.dropna()
```

In [11]:
```
df.columns

```

Out[11]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [12]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     29669 non-null  object 
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
 14  TOL      29669 non-null  float64
 15  station  29669 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 3.8+ MB
```

In [13]:
```
1 data=df[['BEN', 'CO', 'station']]
2 data
3 
```

Out[13]:

	BEN	CO	station
1	1.50	0.34	28079035
5	2.11	0.63	28079006
21	0.80	0.43	28079024
23	1.29	0.34	28079099
25	0.87	0.06	28079035
...	...	...	...
217829	11.76	4.48	28079006
217847	9.79	2.65	28079099
217849	5.86	1.22	28079035
217853	14.47	1.83	28079006
217871	8.09	1.62	28079099

29669 rows × 3 columns

In [15]:
```
df=df.head(10000)
df
```

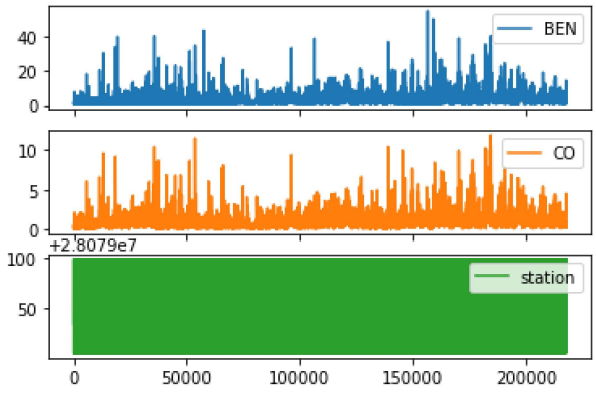
Out[15]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL	
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11	1.24	10.82	28
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.94	0.05	66.260002	118.099998	3.15	33.500000	122.699997	2.29	6.36	1.23	13.28	28
21	2001-08-01 01:00:00	0.80	0.43	0.71	1.20	0.10	27.190001	29.700001	0.76	56.990002	114.300003	0.49	10.84	1.42	3.43	28
23	2001-08-01 01:00:00	1.29	0.34	1.41	3.09	0.07	40.750000	51.570000	1.70	51.580002	102.199997	1.28	7.97	1.30	7.83	28
25	2001-08-01 02:00:00	0.87	0.06	0.88	2.41	0.01	29.709999	31.440001	1.20	56.520000	56.290001	1.02	6.90	1.17	6.49	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
62591	2001-07-16 00:00:00	2.44	0.58	1.94	4.07	0.14	70.599998	95.660004	1.83	27.500000	24.820000	1.69	10.45	1.40	11.16	28
62593	2001-07-16 01:00:00	1.08	0.17	1.00	2.77	0.04	49.529999	49.419998	1.30	37.360001	10.670000	1.13	0.63	1.20	6.49	28
62597	2001-07-16 01:00:00	2.21	0.56	2.82	6.89	0.04	57.580002	96.769997	3.60	29.420000	18.900000	2.66	12.22	1.22	13.30	28
62613	2001-07-16 01:00:00	0.57	0.73	0.44	0.86	0.14	55.130001	56.240002	0.46	26.530001	29.830000	0.44	10.16	1.35	2.13	28
62615	2001-07-16 01:00:00	2.04	0.54	1.58	3.47	0.11	63.080002	88.099998	1.48	27.389999	16.340000	1.42	10.21	1.30	8.98	28

10000 rows × 16 columns

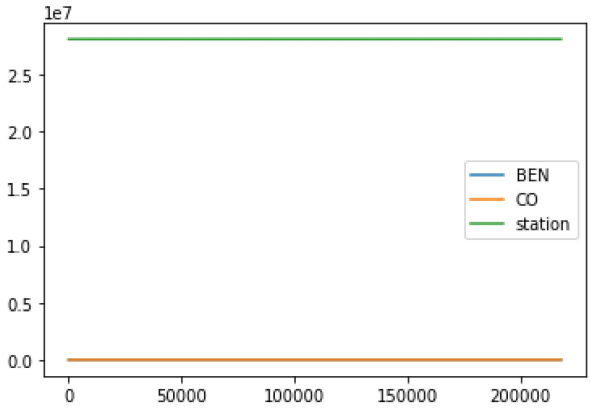
In [16]:
```
data.plot.line(subplots=True)
```

Out[16]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [17]:
```
1 data.plot.line()
2
```

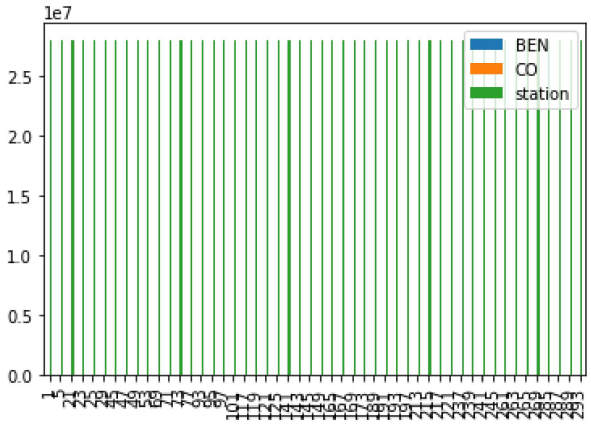
Out[17]: <AxesSubplot:>



In [18]:
```
1 b=data[0:50]
2
```

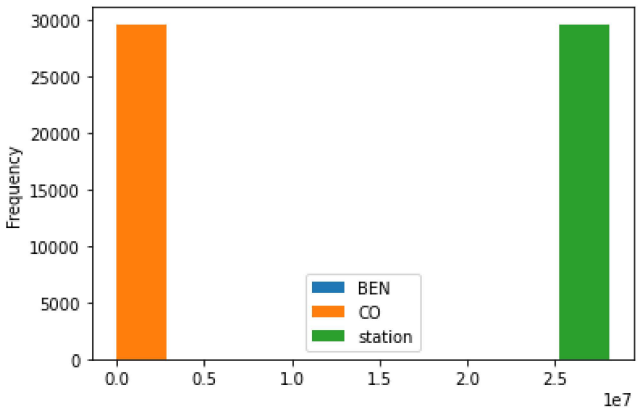
In [19]:
```
1 b.plot.bar()
```

Out[19]: <AxesSubplot:>



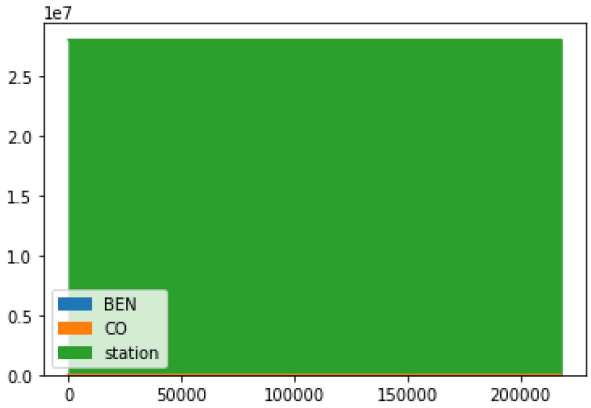
In [20]:
```
1 data.plot.hist()
2
```

Out[20]: <AxesSubplot:ylabel='Frequency'>



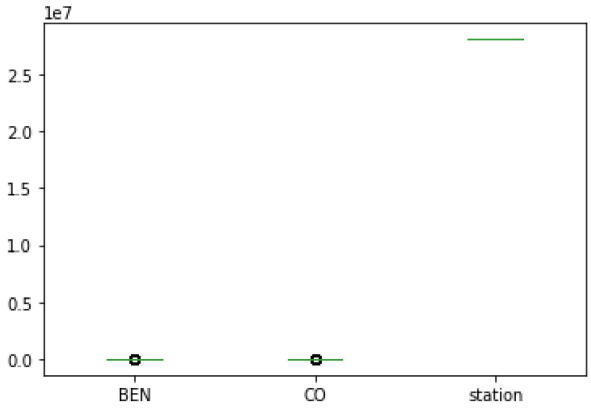
In [21]:
```
1 data.plot.area()
```

Out[21]: <AxesSubplot:>



In [22]:
```
1 data.plot.box()
```

Out[22]: <AxesSubplot:>



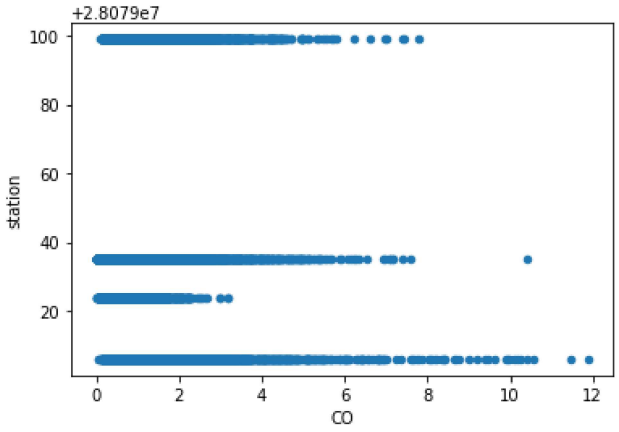
In [23]:
```
1 b.plot.pie(y='station' )
```

Out[23]: <AxesSubplot:ylabel='station'>

In [24]:
```
data.plot.scatter(x='CO' ,y='station')

```

Out[24]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 62615
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     10000 non-null  object 
 1   BEN      10000 non-null  float64
 2   CO       10000 non-null  float64
 3   EBE      10000 non-null  float64
 4   MXY      10000 non-null  float64
 5   NMHC     10000 non-null  float64
 6   NO_2     10000 non-null  float64
 7   NOx      10000 non-null  float64
 8   OXY      10000 non-null  float64
 9   O_3      10000 non-null  float64
 10  PM10     10000 non-null  float64
 11  PXY      10000 non-null  float64
 12  SO_2     10000 non-null  float64
 13  TCH      10000 non-null  float64
 14  TOL      10000 non-null  float64
 15  station  10000 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 1.3+ MB
```

In [26]:
```
df.describe()

```

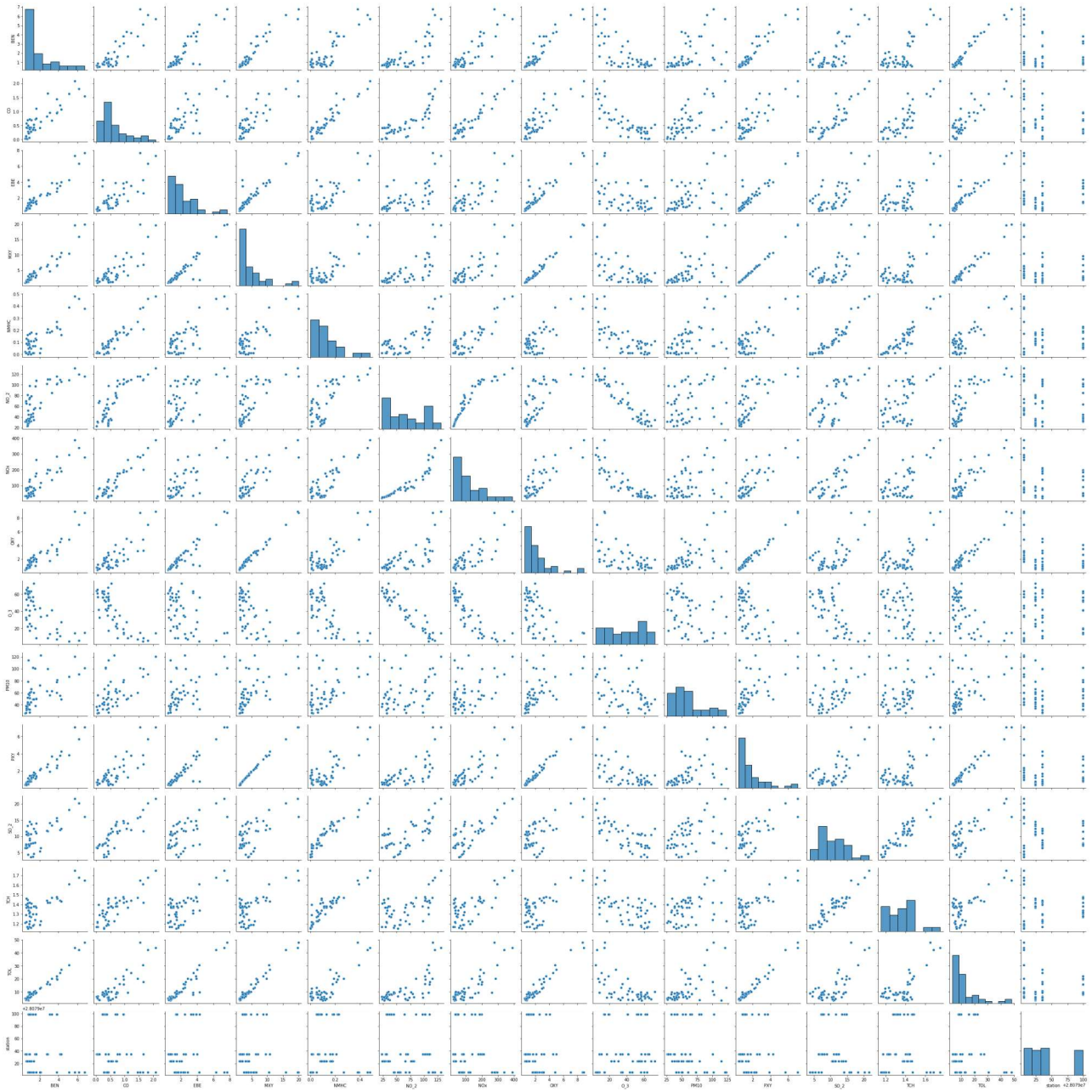
Out[26]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.
mean	2.464378	0.794057	2.725233	6.627726	0.168879	58.368090	123.900787	3.074923	35.
std	2.574364	0.707991	2.657131	7.040931	0.177680	31.978840	115.455642	3.060231	26.
min	0.180000	0.000000	0.160000	0.210000	0.000000	1.180000	1.280000	0.230000	0.
25%	0.910000	0.400000	1.080000	2.300000	0.070000	34.500000	47.307501	1.160000	13.
50%	1.770000	0.600000	2.040000	4.770000	0.120000	55.914999	95.099998	2.280000	30.
75%	3.210000	0.970000	3.530000	8.780000	0.200000	78.342497	167.500000	3.960000	50.
max	43.330002	11.460000	56.009998	150.600006	2.880000	247.600006	1661.000000	63.950001	173.

In [27]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [28]:
```
sns.pairplot(df1[0:50])
```

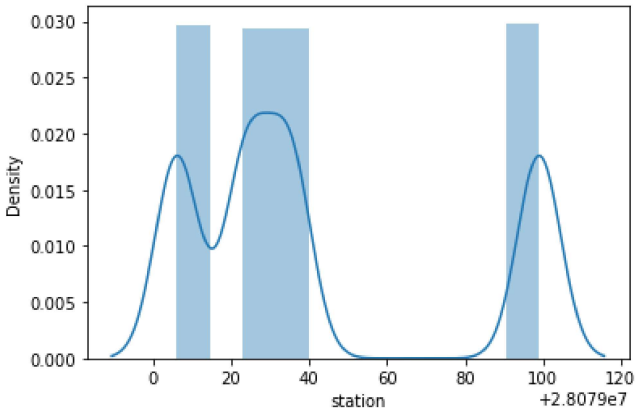
Out[28]: <seaborn.axisgrid.PairGrid at 0x21042d08430>



In [29]:
```
1 sns.distplot(df1['station'])
2
```

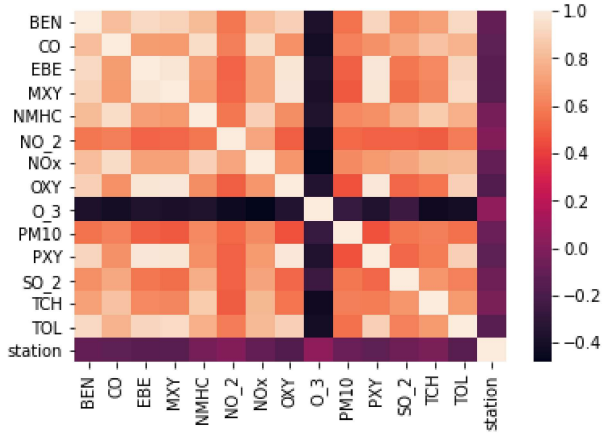
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[29]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [30]:
```
1 sns.heatmap(df1.corr())
```

Out[30]: <AxesSubplot:>



In [31]:
```
1 x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
2  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
4
```

In [32]:
```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
3
```

In [33]:
```
1 from sklearn.linear_model import LinearRegression
2 lr=LinearRegression()
3 lr.fit(x_train,y_train)
4
```

Out[33]: LinearRegression()

In [34]:
```
1 lr.intercept_
```

Out[34]: 28079036.530436143

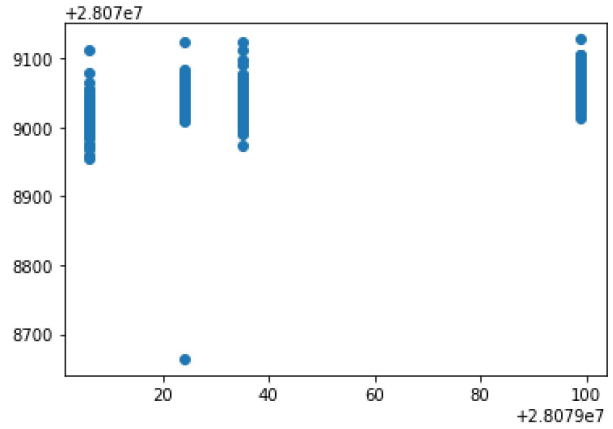
In [35]:
```
1 coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
2 coeff
```

Out[35]:

	Co-efficient
BEN	1.815061
CO	-46.024491
EBE	-2.181664
MXY	4.274769
NMHC	160.983046
NO_2	0.195182
NOx	0.014646
OXY	-34.005474
O_3	0.087986
PM10	-0.136488
PXY	31.039931
SO_2	-0.155539
TCH	9.170819
TOL	-0.642036

In [36]:
```
1 prediction =lr.predict(x_test)
2 plt.scatter(y_test,prediction)
```

Out[36]: <matplotlib.collections.PathCollection at 0x21054828670>



In [37]:
```
1 lr.score(x_test,y_test)
2
```

Out[37]: 0.2613987663911469

In [38]:
```
1 lr.score(x_train,y_train)
2
```

Out[38]: 0.2591058509180959

In [39]:
```
1 from sklearn.linear_model import Ridge,Lasso
2
```

In [40]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[40]:
```
Ridge(alpha=10)
```

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

```

Out[41]:
```
0.25146802955198544
```

In [43]:
```
rr.score(x_train,y_train)

```

Out[43]:
```
0.2515937700467924
```

In [44]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[44]:
```
Lasso(alpha=10)
```

In [45]:
```
la.score(x_train,y_train)

```

Out[45]:
```
0.0392936344978263
```

In [46]:
```
la.score(x_test,y_test)

```

Out[46]:
```
0.03326296568142484
```

In [47]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[47]:
```
ElasticNet()
```

In [48]:
```
en.coef_

```

Out[48]:
```
array([ 1.93988337, -0.        ,  0.75401455,  0.8502301 ,  0.        ,
        0.22747743, -0.04487149, -5.09923744,  0.0376579 , -0.05899578,
        1.79467969,  0.03897251,  0.29946019, -0.40427752])
```

In [49]:
```
en.intercept_

```

Out[49]:
```
28079038.430305947
```

In [50]:
```
prediction=en.predict(x_test)

```

In [51]:
```
en.score(x_test,y_test)
```

Out[51]:
```
0.07569731203877605
```

In [52]:
```
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)))

```

```
28.453854562215508
1137.6301535097366
33.72877337689197
```

In [53]:
```
from sklearn.linear_model import LogisticRegression


```

In [54]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']

```

In [55]:
```
feature_matrix.shape


```

Out[55]: (10000, 14)

In [56]:
```
target_vector.shape


```

Out[56]: (10000,)

In [57]:
```
from sklearn.preprocessing import StandardScaler
```

In [58]:
```
fs=StandardScaler().fit_transform(feature_matrix)


```

In [59]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[59]: LogisticRegression(max_iter=10000)

In [60]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]


```

In [61]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079035]
```

In [62]:
```
logr.classes_


```

Out[62]: array([28079006, 28079024, 28079035, 28079099], dtype=int64)

In [63]:
```
logr.score(fs,target_vector)


```

Out[63]: 0.9169

In [64]:
```
logr.predict_proba(observation)[0][0]


```

Out[64]: 5.304850011563579e-54

In [65]:
```
logr.predict_proba(observation)


```

Out[65]: array([[5.30485001e-54, 1.57746850e-80, 1.00000000e+00, 1.61083523e-37]])

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [67]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[67]: RandomForestClassifier()

In [68]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [69]:
```
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[69]:
```
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 [70]:
```
grid_search.best_score_

```

Out[70]:
```
0.7889999999999999
```

In [71]:
```
rfc_best=grid_search.best_estimator_
```

In [73]:
```
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','c','d'],filled=True)
```

Out[73]:
```
[Text(1980.3529411764707, 1993.2, 'NOx <= 34.32\ngini = 0.75\nsamples = 4419\nvalue = [1782, 1710, 1
749, 1759]\nclass = a'),
 Text(897.1764705882354, 1630.8000000000002, 'TCH <= 1.235\ngini = 0.31\nsamples = 766\nvalue = [36,
987, 111, 65]\nclass = b'),
 Text(437.6470588235294, 1268.4, 'NMHC <= 0.005\ngini = 0.485\nsamples = 107\nvalue = [36, 0, 110, 1
7]\nclass = c'),
 Text(175.05882352941177, 906.0, 'OXY <= 1.115\ngini = 0.488\nsamples = 32\nvalue = [33, 0, 24, 0]\n
class = a'),
 Text(87.52941176470588, 543.5999999999999, 'gini = 0.305\nsamples = 11\nvalue = [3, 0, 13, 0]\nclas
s = c'),
 Text(262.5882352941177, 543.5999999999999, 'PXY <= 1.35\ngini = 0.393\nsamples = 21\nvalue = [30,
0, 11, 0]\nclass = a'),
 Text(175.05882352941177, 181.19999999999982, 'gini = 0.0\nsamples = 11\nvalue = [20, 0, 0, 0]\nclas
s = a'),
 Text(350.11764705882354, 181.19999999999982, 'gini = 0.499\nsamples = 10\nvalue = [10, 0, 11, 0]\nc
lass = c'),
 Text(700.2352941176471, 906.0, 'TCH <= 1.225\ngini = 0.315\nsamples = 75\nvalue = [3, 0, 86, 17]\nc
lass = c'),
 Text(612.7058823529412, 543.5999999999999, 'OXY <= 1.025\ngini = 0.181\nsamples = 65\nvalue = [3,
```

Conclusion

Linear Regression=0.2591058509180959

Ridge Regression=0.2515937700467924

Lasso Regression=0.0392936344978263

ElasticNet Regression=0.07569731203877605

Logistic Regression=0.9169

Random Forest=0.7889999999999999

Logistic Regression Is Suitable for this Dataset