Vijay(P6) 3/08/2023

In [327]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [390]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2006.csv")
df
```

Out[390]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570000	40.259998	NaN	33.779999
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	NaN	2.48	11.890000
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419998	NaN	NaN	19.719999
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260000	NaN	NaN	21.129999
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180000	NaN	NaN	11.050000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120003	NaN	NaN	7.400000
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	29.469999	15.640000	0.50	8.840000
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680000	35.000000	NaN	12.110000
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360001	NaN	NaN	4.890000
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	52.490002	27.920000	1.70	9.170000

230568 rows × 17 columns

In [391]:
```
df=df.dropna()
```

In [392]:
```
df.columns

```

Out[392]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [393]:
```
1 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 [394]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3
```

Out[394]:

	BEN	CO	station
5	9.41	1.69	28079006
22	1.69	0.79	28079024
25	2.35	1.47	28079099
31	4.39	0.85	28079006
48	1.93	0.79	28079024
...	...	...	...
230538	0.42	0.40	28079024
230541	1.63	0.94	28079099
230547	3.99	1.06	28079006
230564	0.76	0.32	28079024
230567	1.95	0.74	28079099

24758 rows × 3 columns

In [395]:
```
df=df.head(10000)
df
```

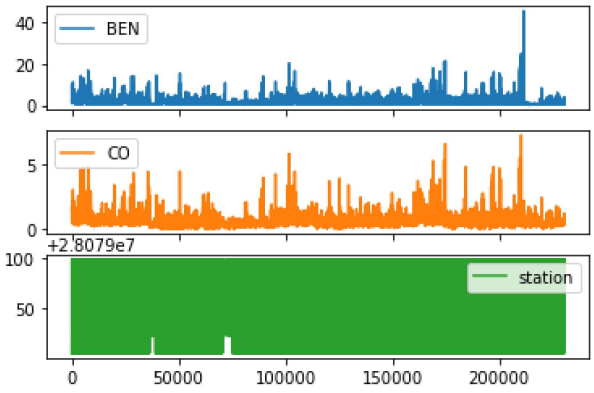
Out[395]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.990000	89.190002	43.150002	10.11	28.9
22	2006-02-01 01:00:00	1.69	0.79	1.24	2.670000	0.17	59.910000	120.199997	1.11	2.450000	25.570000	18.570000	0.79	9.9
25	2006-02-01 01:00:00	2.35	1.47	2.64	9.660000	0.40	117.699997	346.399994	5.15	4.780000	59.029999	32.169998	4.46	23.1
31	2006-02-01 02:00:00	4.39	0.85	7.92	17.139999	0.25	92.059998	237.000000	9.24	5.920000	35.139999	22.010000	8.06	20.8
48	2006-02-01 02:00:00	1.93	0.79	1.24	2.740000	0.16	60.189999	125.099998	1.11	2.280000	26.719999	18.570000	0.82	9.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
94673	2006-08-30 19:00:00	1.46	0.53	2.02	3.970000	0.08	71.010002	140.500000	1.98	30.799999	37.580002	28.010000	1.61	8.9
94689	2006-08-30 19:00:00	0.32	0.52	0.38	0.560000	0.11	9.620000	11.740000	1.00	77.419998	15.920000	2.290000	1.00	7.6
94693	2006-08-30 19:00:00	0.46	0.35	0.82	2.080000	0.14	44.590000	69.900002	1.19	57.150002	40.290001	15.350000	1.07	7.8
94699	2006-08-30 20:00:00	1.79	0.58	2.00	4.030000	0.12	74.010002	150.899994	2.00	24.090000	24.139999	18.510000	1.66	9.2
94715	2006-08-30 20:00:00	0.24	0.36	0.54	0.660000	0.10	12.260000	14.250000	1.00	70.449997	22.980000	5.850000	1.00	7.6

10000 rows × 17 columns

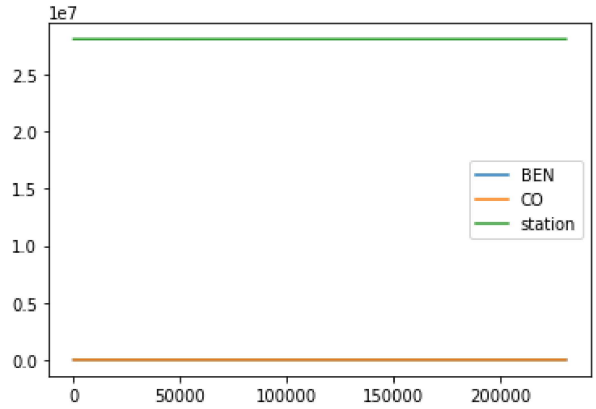
In [396]:
```
data.plot.line(subplots=True)
```

Out[396]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [397]:
```
1 data.plot.line()
2
```

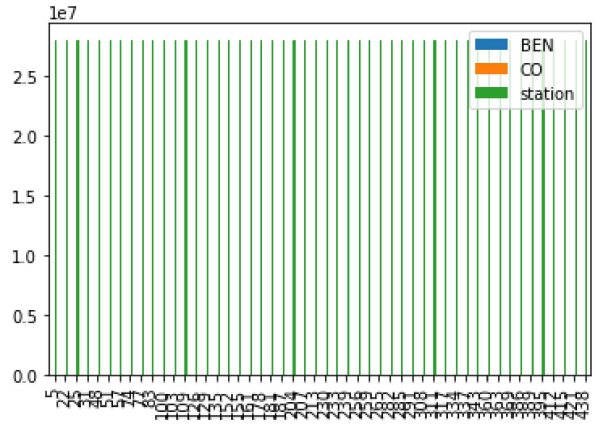
Out[397]: <AxesSubplot:>



In [398]:
```
1 b=data[0:50]
2
```

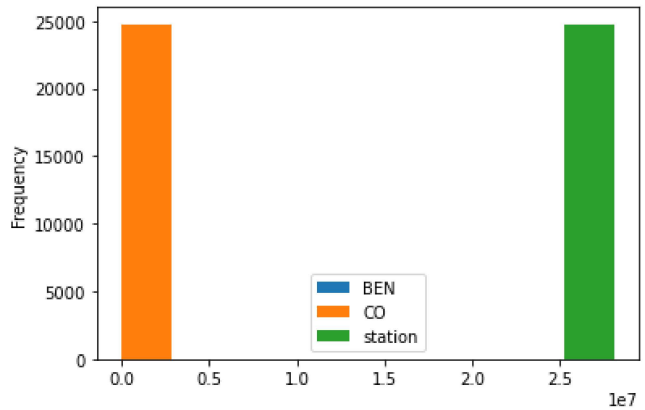
In [399]:
```
1 b.plot.bar()
```

Out[399]: <AxesSubplot:>



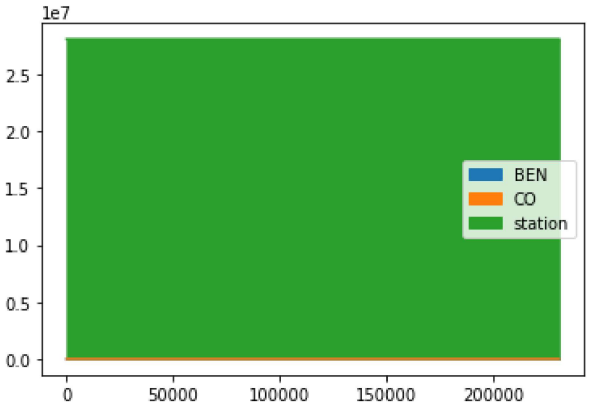
In [400]:
```
1 data.plot.hist()
2
```

Out[400]: <AxesSubplot:ylabel='Frequency'>



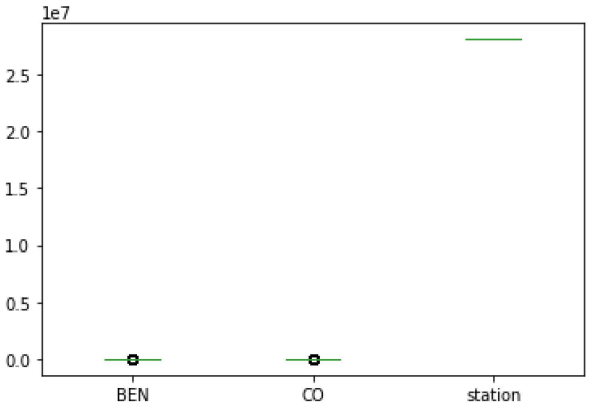
In [401]:
```
1 data.plot.area()
```

Out[401]: <AxesSubplot:>



In [402]:
```
1 data.plot.box()
```

Out[402]: <AxesSubplot:>

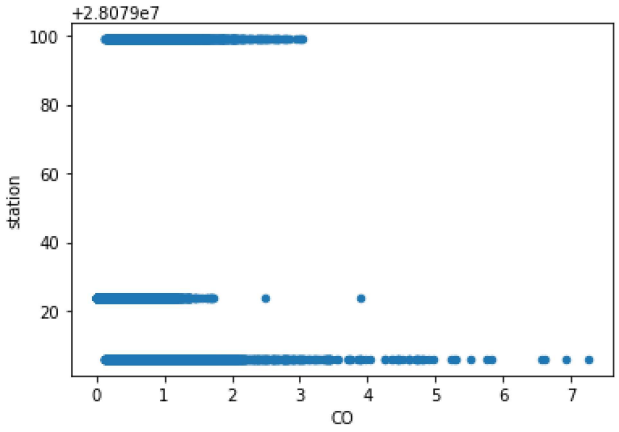


In [403]:
```
1 b.plot.pie(y='station' )
```

Out[403]: <AxesSubplot:ylabel='station'>

In [404]:
```
1 data.plot.scatter(x='CO' ,y='station')
2
```

Out[404]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [405]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 5 to 94715
Data columns (total 17 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  PM25     10000 non-null  float64
 12  PXY      10000 non-null  float64
 13  SO_2     10000 non-null  float64
 14  TCH      10000 non-null  float64
 15  TOL      10000 non-null  float64
 16  station  10000 non-null  int64
dtypes: float64(15), int64(1), object(1)
memory usage: 1.4+ MB
```

In [406]:
```
1 df.describe()
2
```

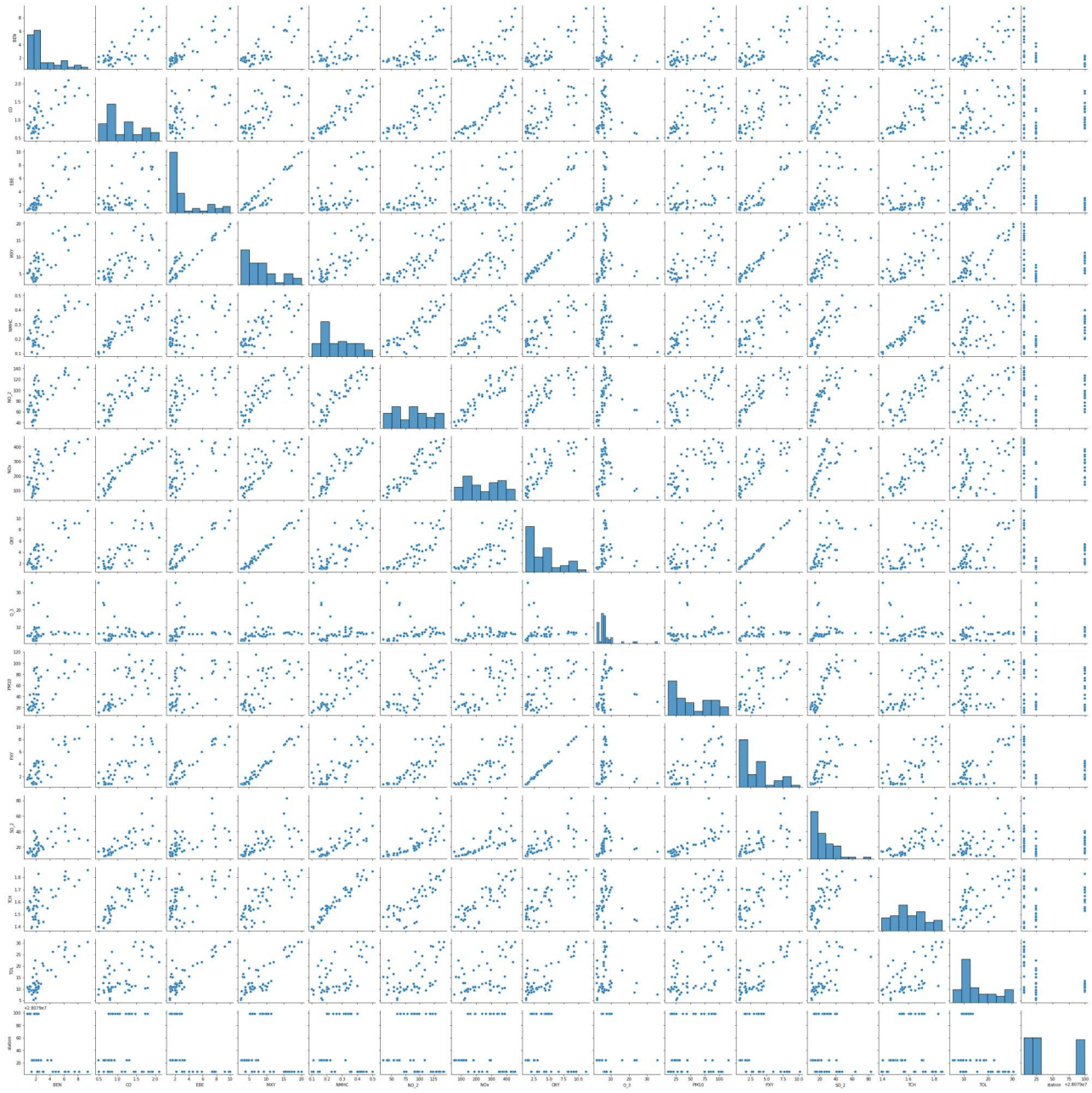
Out[406]:

	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	1.346984	0.582088	1.812246	3.825016	0.154214	58.899764	113.371383	2.010414	43.
std	1.416481	0.416500	1.825564	4.076519	0.107825	42.731971	116.752809	1.886838	33.
min	0.130000	0.000000	0.170000	0.150000	0.000000	1.790000	2.020000	0.190000	1.
25%	0.460000	0.340000	0.790000	1.000000	0.090000	25.817500	32.217501	0.960000	14.
50%	0.870000	0.480000	1.090000	2.440000	0.140000	51.744999	79.364998	1.240000	35.
75%	1.700000	0.710000	2.122500	5.060000	0.180000	82.617500	153.625004	2.440000	64.
max	16.900000	4.790000	37.450001	66.900002	1.450000	438.700012	1279.000000	25.049999	168.

In [407]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [408]:
```
sns.pairplot(df1[0:50])
```

Out[408]: <seaborn.axisgrid.PairGrid at 0x2102d4a8610>

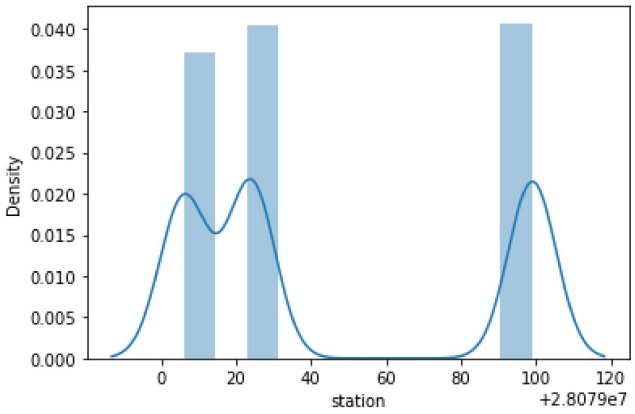


In [409]:
```
sns.distplot(df1['station'])

```

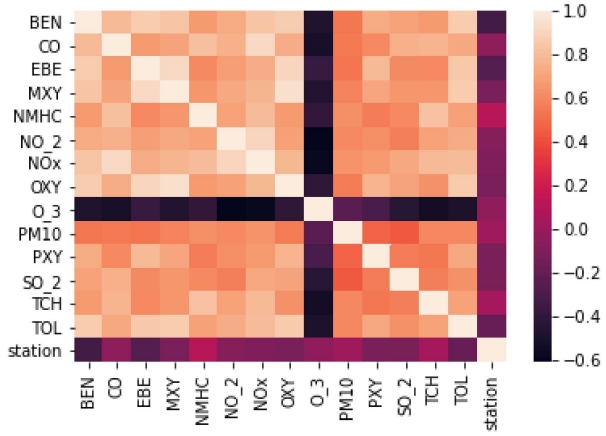
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[409]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [410]:
```
sns.heatmap(df1.corr())
```

Out[410]: <AxesSubplot:>



In [411]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']

```

In [412]:
```
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)

```

In [413]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

```

Out[413]: LinearRegression()

In [414]:
```
lr.intercept_
```

Out[414]: 28079065.61052085

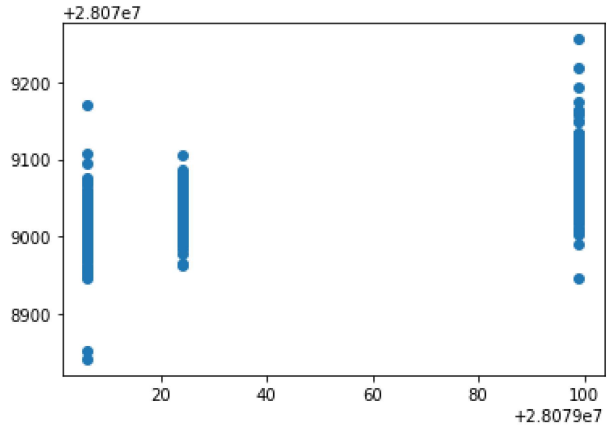
In [415]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[415]:

	Co-efficient
BEN	-25.498473
CO	1.385723
EBE	-16.554128
MXY	3.296136
NMHC	169.962839
NO_2	-0.071097
NOx	-0.000558
OXY	12.727748
O_3	-0.250329
PM10	0.133200
PXY	5.127215
SO_2	-0.405084
TCH	-8.901934
TOL	-0.589435

In [416]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[416]: <matplotlib.collections.PathCollection at 0x21086deba30>



In [417]:
```
lr.score(x_test,y_test)

```

Out[417]: 0.4617191908758437

In [418]:
```
lr.score(x_train,y_train)

```

Out[418]: 0.4441398393609025

In [419]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [420]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[420]: Ridge(alpha=10)

In [421]:
```
rr.score(x_test,y_test)

```

Out[421]: 0.45920564760149407

In [422]:
```
rr.score(x_train,y_train)

```

Out[422]: 0.43837711691775905

In [423]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[423]: Lasso(alpha=10)

In [424]:
```
la.score(x_train,y_train)

```

Out[424]: 0.12508235315862137

In [425]:
```
la.score(x_test,y_test)

```

Out[425]: 0.12174806831279561

In [426]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[426]: ElasticNet()

In [427]:
```
en.coef_

```

Out[427]:
```
array([-10.86532069,   0.22560891,  -8.0905617 ,   2.9414993 ,
         0.45379798,  -0.06958589,   0.04514174,   2.76154299,
        -0.20335166,   0.29691529,   2.28882068,  -0.16842098,
         0.21714439,  -1.3351952 ])
```

In [428]:
```
en.intercept_

```

Out[428]: 28079059.862034153

In [429]:
```
prediction=en.predict(x_test)

```

In [430]:
```
en.score(x_test,y_test)
```

Out[430]: 0.28641446201483867

In [431]:
```
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)))

```

```
30.757393950788924
1145.0167789207833
33.83809656172733
```

In [432]:
```
from sklearn.linear_model import LogisticRegression

```

In [433]:
```
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 [434]:
```
feature_matrix.shape

```

Out[434]: (10000, 14)

In [435]:
```
target_vector.shape

```

Out[435]: (10000,)

In [436]:
```
from sklearn.preprocessing import StandardScaler
```

In [437]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [438]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[438]: LogisticRegression(max_iter=10000)

In [439]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

```

In [440]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079006]
```

In [441]:
```
logr.classes_

```

Out[441]: array([28079006, 28079024, 28079099], dtype=int64)

In [442]:
```
logr.score(fs,target_vector)

```

Out[442]: 0.8916

In [443]:
```
logr.predict_proba(observation)[0][0]

```

Out[443]: 0.7145321075031248

In [444]:
```
logr.predict_proba(observation)

```

Out[444]: array([[7.14532108e-01, 1.48326693e-21, 2.85467892e-01]])

In [445]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [446]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[446]: RandomForestClassifier()

In [447]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [448]:
```
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[448]:
```
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 [449]:
```
grid_search.best_score_
```

Out[449]:
```
0.8971428571428571
```

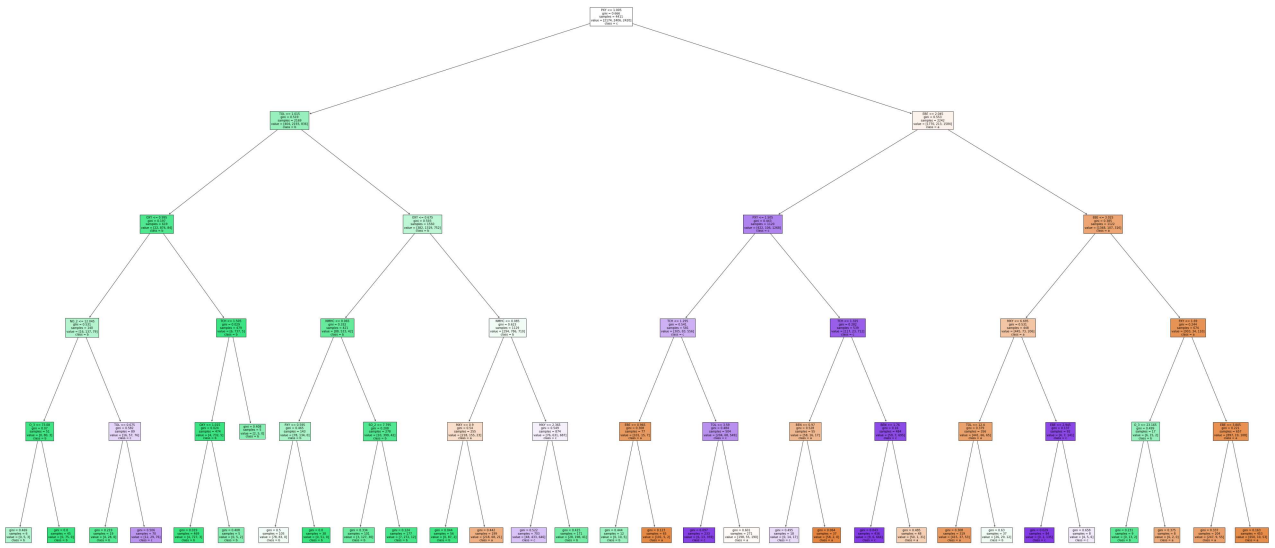
In [450]:
```
rfc_best=grid_search.best_estimator_
```

In [451]:

```
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[451]:
```
[Text(2148.3, 1993.2, 'PXY <= 1.005\ngini = 0.666\nsamples = 4411\nvalue = [2174, 2406, 2420]\nclass =
c'),
 Text(1023.0000000000001, 1630.8000000000002, 'TOL <= 1.015\ngini = 0.519\nsamples = 2169\nvalue = [40
4, 2193, 836]\nclass = b'),
 Text(558.0, 1268.4, 'OXY <= 0.995\ngini = 0.197\nsamples = 619\nvalue = [22, 874, 84]\nclass = b'),
 Text(297.6, 906.0, 'NO_2 <= 12.045\ngini = 0.531\nsamples = 140\nvalue = [16, 137, 79]\nclass = b'),
 Text(148.8, 543.5999999999999, 'O_3 <= 73.08\ngini = 0.07\nsamples = 51\nvalue = [0, 80, 3]\nclass =
b'),
 Text(74.4, 181.19999999999982, 'gini = 0.469\nsamples = 6\nvalue = [0, 5, 3]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 45\nvalue = [0, 75, 0]\nclass =
b'),
 Text(446.40000000000003, 543.5999999999999, 'TOL <= 0.675\ngini = 0.582\nsamples = 89\nvalue = [16, 5
7, 76]\nclass = c'),
 Text(372.0, 181.19999999999982, 'gini = 0.219\nsamples = 19\nvalue = [4, 28, 0]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.506\nsamples = 70\nvalue = [12, 29, 76]\nclass =
c'),
 Text(818.4000000000001, 906.0, 'TCH <= 1.505\ngini = 0.029\nsamples = 479\nvalue = [6, 737, 5]\nclass
= b'),
 Text(744.0, 543.5999999999999, 'OXY <= 1.015\ngini = 0.024\nsamples = 474\nvalue = [4, 732, 5]\nclass
= b'),
 Text(669.6, 181.19999999999982, 'gini = 0.019\nsamples = 469\nvalue = [4, 727, 3]\nclass = b'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.408\nsamples = 5\nvalue = [0, 5, 2]\nclass =
b'),
 Text(892.8000000000001, 543.5999999999999, 'gini = 0.408\nsamples = 5\nvalue = [2, 5, 0]\nclass = b'),
 Text(1488.0, 1268.4, 'OXY <= 0.675\ngini = 0.593\nsamples = 1550\nvalue = [382, 1319, 752]\nclass =
b'),
 Text(1190.4, 906.0, 'NMHC <= 0.065\ngini = 0.332\nsamples = 421\nvalue = [88, 533, 42]\nclass = b'),
 Text(1041.6000000000001, 543.5999999999999, 'PXY <= 0.595\ngini = 0.465\nsamples = 143\nvalue = [78, 1
34, 0]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.5\nsamples = 105\nvalue = [78, 83, 0]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.0\nsamples = 38\nvalue = [0, 51, 0]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'SO_2 <= 7.795\ngini = 0.208\nsamples = 278\nvalue = [10, 399, 42]\ncl
ass = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.334\nsamples = 101\nvalue = [3, 127, 30]\nclass
= b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.124\nsamples = 177\nvalue = [7, 272, 12]\nclass
= b'),
 Text(1785.6000000000001, 906.0, 'NMHC <= 0.085\ngini = 0.623\nsamples = 1129\nvalue = [294, 786, 710]
\nclass = b'),
 Text(1636.8000000000002, 543.5999999999999, 'MXY <= 0.9\ngini = 0.54\nsamples = 255\nvalue = [218, 15
5, 23]\nclass = a'),
 Text(1562.4, 181.19999999999982, 'gini = 0.044\nsamples = 56\nvalue = [0, 87, 2]\nclass = b'),
 Text(1711.2, 181.19999999999982, 'gini = 0.442\nsamples = 199\nvalue = [218, 68, 21]\nclass = a'),
 Text(1934.4, 543.5999999999999, 'MXY <= 2.365\ngini = 0.549\nsamples = 874\nvalue = [76, 631, 687]\ncl
ass = c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.522\nsamples = 703\nvalue = [48, 433, 646]\ncla
ss = c'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.415\nsamples = 171\nvalue = [28, 198, 41]\nclas
s = b'),
 Text(3273.6000000000004, 1630.8000000000002, 'EBE <= 2.045\ngini = 0.553\nsamples = 2242\nvalue = [177
0, 213, 1584]\nclass = a'),
 Text(2678.4, 1268.4, 'PXY <= 1.505\ngini = 0.443\nsamples = 1120\nvalue = [422, 106, 1268]\nclass =
c'),
 Text(2380.8, 906.0, 'TCH <= 1.295\ngini = 0.541\nsamples = 581\nvalue = [305, 83, 556]\nclass = c'),
 Text(2232.0, 543.5999999999999, 'EBE <= 0.965\ngini = 0.308\nsamples = 77\nvalue = [101, 15, 7]\nclass
= a'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.444\nsamples = 12\nvalue = [0, 10, 5]\nclass =
b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.123\nsamples = 65\nvalue = [101, 5, 2]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'TOL <= 3.59\ngini = 0.484\nsamples = 504\nvalue = [204, 6
8, 549]\nclass = c'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.097\nsamples = 233\nvalue = [6, 13, 359]\nclass
= c'),
 Text(2604.0, 181.19999999999982, 'gini = 0.601\nsamples = 271\nvalue = [198, 55, 190]\nclass = a'),
 Text(2976.0, 906.0, 'TCH <= 1.325\ngini = 0.282\nsamples = 539\nvalue = [117, 23, 712]\nclass = c'),
 Text(2827.2000000000003, 543.5999999999999, 'BEN <= 0.97\ngini = 0.528\nsamples = 55\nvalue = [58, 16,
17]\nclass = a'),
 Text(2752.8, 181.19999999999982, 'gini = 0.495\nsamples = 18\nvalue = [0, 14, 17]\nclass = c'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.064\nsamples = 37\nvalue = [58, 2, 0]\nclass =
a'),
 Text(3124.8, 543.5999999999999, 'BEN <= 1.76\ngini = 0.16\nsamples = 484\nvalue = [59, 7, 695]\nclass
```

```
= c'),
 Text(3050.4, 181.19999999999982, 'gini = 0.043\nsamples = 436\nvalue = [9, 6, 664]\nclass = c'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.485\nsamples = 48\nvalue = [50, 1, 31]\nclass =
a'),
 Text(3868.8, 1268.4, 'EBE <= 3.025\ngini = 0.385\nsamples = 1122\nvalue = [1348, 107, 316]\nclass =
a'),
 Text(3571.2000000000003, 906.0, 'MXY <= 6.695\ngini = 0.531\nsamples = 448\nvalue = [445, 73, 206]\ncl
ass = a'),
 Text(3422.4, 543.5999999999999, 'TOL <= 12.4\ngini = 0.379\nsamples = 356\nvalue = [441, 66, 65]\nclas
s = a'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.308\nsamples = 319\nvalue = [415, 37, 53]\nclas
s = a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.63\nsamples = 37\nvalue = [26, 29, 12]\nclass = b'),
 Text(3720.0000000000005, 543.5999999999999, 'EBE <= 2.945\ngini = 0.137\nsamples = 92\nvalue = [4, 7,
141]\nclass = c'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.029\nsamples = 84\nvalue = [0, 2, 135]\nclass =
c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.658\nsamples = 8\nvalue = [4, 5, 6]\nclass = c'),
 Text(4166.400000000001, 906.0, 'PXY <= 1.69\ngini = 0.244\nsamples = 674\nvalue = [903, 34, 110]\nclas
s = a'),
 Text(4017.6000000000004, 543.5999999999999, 'O_3 <= 23.165\ngini = 0.499\nsamples = 17\nvalue = [6, 1
5, 2]\nclass = b'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.231\nsamples = 9\nvalue = [0, 13, 2]\nclass =
b'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.375\nsamples = 8\nvalue = [6, 2, 0]\nclass =
a'),
 Text(4315.200000000001, 543.5999999999999, 'EBE <= 3.805\ngini = 0.221\nsamples = 657\nvalue = [897, 1
9, 108]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.337\nsamples = 204\nvalue = [247, 9, 55]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.163\nsamples = 453\nvalue = [650, 10, 53]\nclass = a')]
```



Conclusion

Linear Regression=0.4441398393609025

Ridge Regression=0.43837711691775905

Lasso Regression=0.12508235315862137

ElasticNet Regression=0.28641446201483867

Logistic Regression=0.8916

Random Forest=0.8971428571428571

Random Forest is Suitable for this Dataset