Vijay(P4) 3/08/2023

In [75]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [77]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2004.csv")
df
```

Out[77]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TC
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.860001	NaN	12.20	Na
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	NaN	3.38	6.12	1.:
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	NaN	NaN	8.99	Na
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	NaN	NaN	8.82	Na
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	NaN	NaN	8.71	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.860000	0.52	6.62	1.:
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	NaN	2.35	6.92	Na
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.040000	NaN	7.82	1.:
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001	NaN	NaN	5.53	1.:
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.959999	2.29	8.68	1.:

245496 rows × 17 columns

In [78]:
```
df=df.dropna()
```

In [79]:
```
df.columns

```

Out[79]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [80]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [204]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3 
```

Out[204]:

	BEN	CO	station
5	3.24	0.63	28079006
22	0.55	0.36	28079024
26	1.80	0.46	28079099
32	1.94	0.67	28079006
49	0.29	0.30	28079024
...	...	...	...
11379	1.05	0.25	28079099
11385	1.94	0.48	28079006
11403	0.44	0.04	28079024
11407	0.99	0.22	28079099
11413	1.59	0.47	28079006

1000 rows × 3 columns

In [205]:
```
df=df.head(1000)
df
```

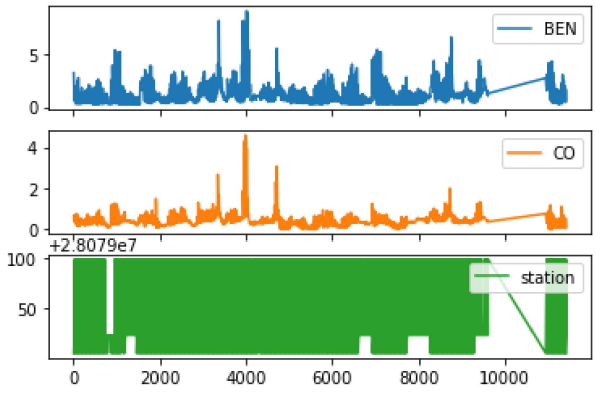
Out[205]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110001	38.049999	4.16	14.28	1.54
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.549999	22.780001	0.39	7.29	1.55
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.630001	29.459999	2.21	8.54	1.55
32	2004-08-01 02:00:00	1.94	0.67	3.14	4.91	0.06	113.500000	165.800003	2.56	26.980000	86.930000	45.639999	2.14	13.47	1.67
49	2004-08-01 02:00:00	0.29	0.30	0.47	0.76	0.07	33.919998	34.840000	0.46	75.570000	48.959999	25.959999	0.34	6.38	1.47
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
11379	2004-08-18 01:00:00	1.05	0.25	1.62	3.37	0.12	43.799999	55.410000	1.91	28.820000	17.940001	12.210000	1.75	5.10	1.38
11385	2004-08-18 02:00:00	1.94	0.48	3.05	5.87	0.07	80.139999	139.500000	2.87	11.790000	33.369999	14.270000	2.33	9.01	1.34
11403	2004-08-18 02:00:00	0.44	0.04	0.50	0.82	0.00	17.770000	18.340000	0.80	51.450001	12.860000	7.380000	0.36	3.74	1.28
11407	2004-08-18 02:00:00	0.99	0.22	1.51	3.12	0.11	41.320000	51.279999	1.87	27.320000	14.060000	9.590000	1.78	5.02	1.45
11413	2004-08-18 03:00:00	1.59	0.47	2.59	4.63	0.08	79.209999	156.199997	2.22	11.460000	62.110001	18.040001	1.80	9.75	1.36

1000 rows × 17 columns

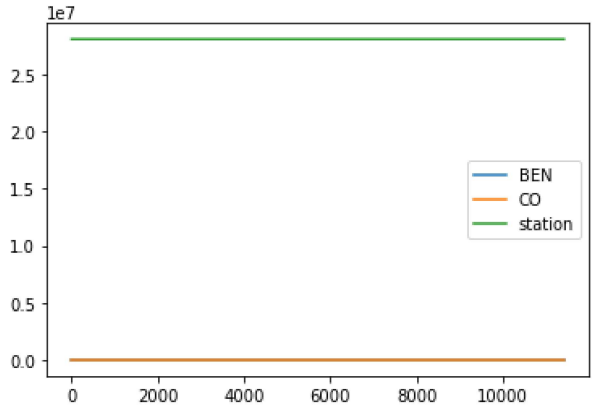
In [206]:
```
data.plot.line(subplots=True)
```

Out[206]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [207]:
```
1 data.plot.line()
2
```

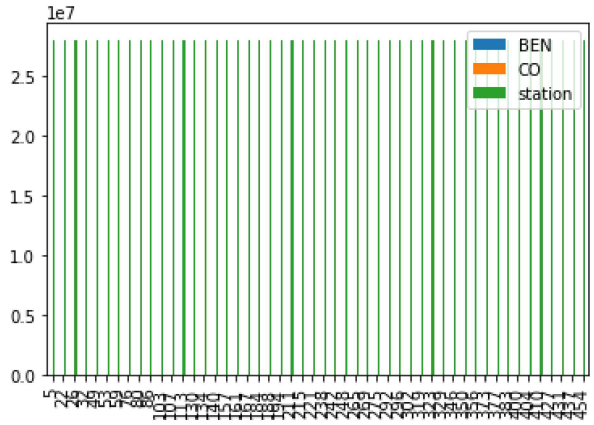
Out[207]: <AxesSubplot:>



In [208]:
```
1 b=data[0:50]
2
```

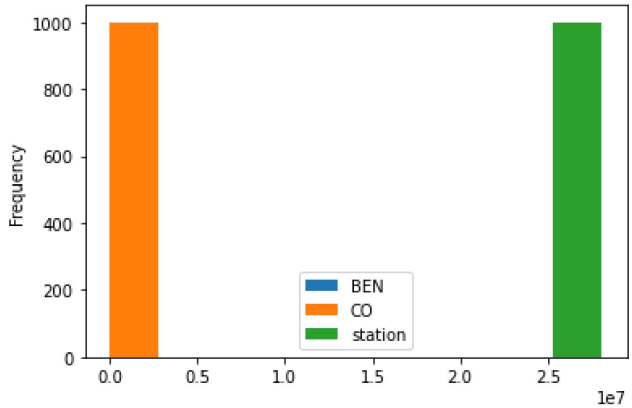
In [209]:
```
1 b.plot.bar()
```

Out[209]: <AxesSubplot:>



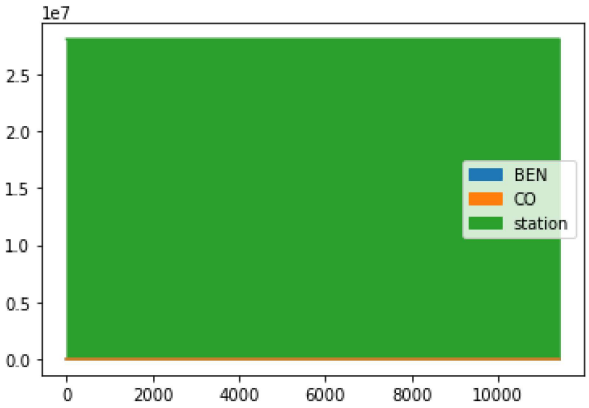
In [210]:
```
1 data.plot.hist()
2
```

Out[210]: <AxesSubplot:ylabel='Frequency'>



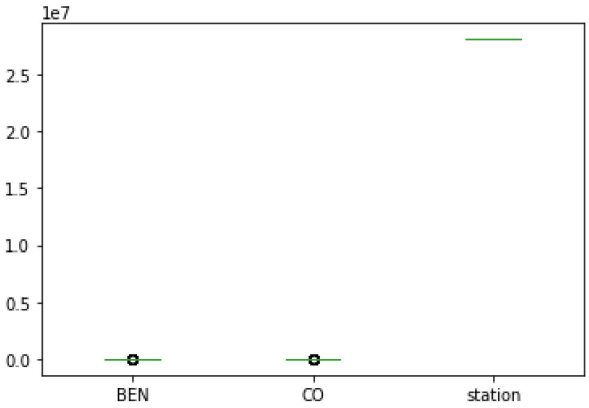
In [211]:
```
1 data.plot.area()
```

Out[211]: <AxesSubplot:>



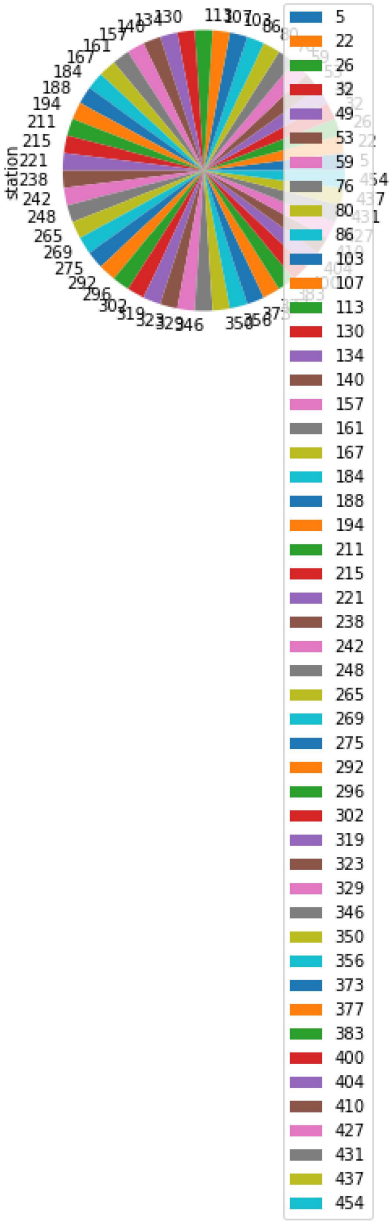
In [212]:
```
1 data.plot.box()
```

Out[212]: <AxesSubplot:>



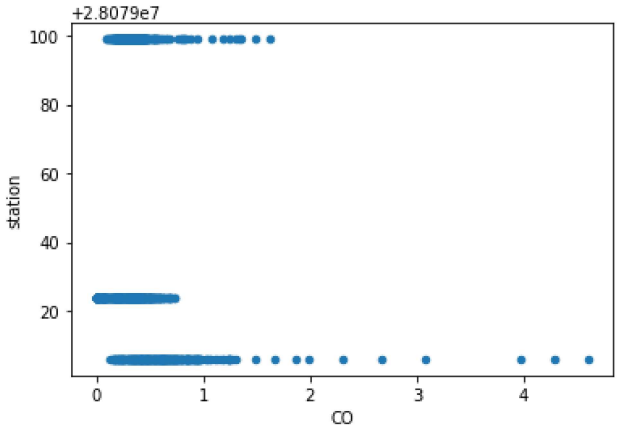
In [214]:
```
1 b.plot.pie(y='station' )
```

Out[214]: <AxesSubplot:ylabel='station'>



In [215]:
```
1 data.plot.scatter(x='CO' ,y='station')
2
```

Out[215]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [216]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 5 to 11413
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     1000 non-null   object
 1   BEN      1000 non-null   float64
 2   CO       1000 non-null   float64
 3   EBE      1000 non-null   float64
 4   MXY      1000 non-null   float64
 5   NMHC     1000 non-null   float64
 6   NO_2     1000 non-null   float64
 7   NOx      1000 non-null   float64
 8   OXY      1000 non-null   float64
 9   O_3      1000 non-null   float64
 10  PM10     1000 non-null   float64
 11  PM25     1000 non-null   float64
 12  PXY      1000 non-null   float64
 13  SO_2     1000 non-null   float64
 14  TCH      1000 non-null   float64
 15  TOL      1000 non-null   float64
 16  station  1000 non-null   int64
dtypes: float64(15), int64(1), object(1)
memory usage: 140.6+ KB
```

In [217]:
```
1 df.describe()
2
```

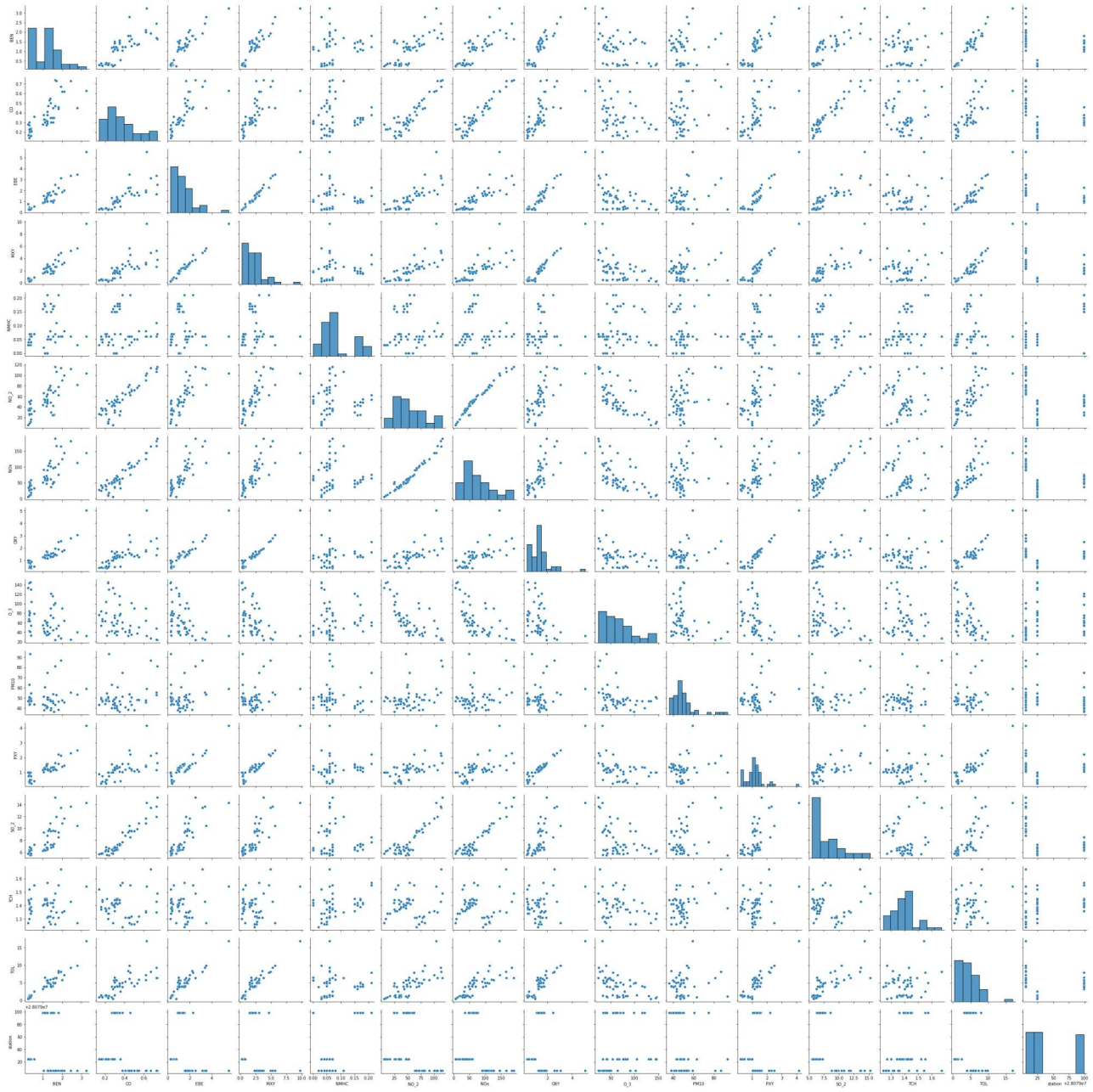
Out[217]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
count	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	100
mean	1.35431	0.432410	1.602970	2.920440	0.098980	47.350710	76.490840	1.883740	51.380820	3
std	1.11013	0.356087	1.222496	2.496344	0.117098	35.178428	82.492057	1.585253	28.560341	1
min	0.18000	0.000000	0.200000	0.230000	0.000000	2.660000	2.880000	0.340000	4.530000	
25%	0.54000	0.260000	0.887500	1.000000	0.020000	21.227500	25.034999	1.000000	29.890000	1
50%	1.16000	0.370000	1.240000	2.270000	0.080000	42.100000	58.394999	1.470000	46.615000	2
75%	1.70000	0.520000	1.990000	3.992500	0.140000	66.032497	99.715002	2.312500	67.967499	4
max	9.12000	4.600000	8.580000	16.920000	2.250000	249.300003	1001.000000	22.020000	145.699997	15

In [218]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [219]:
```
sns.pairplot(df1[0:50])
```

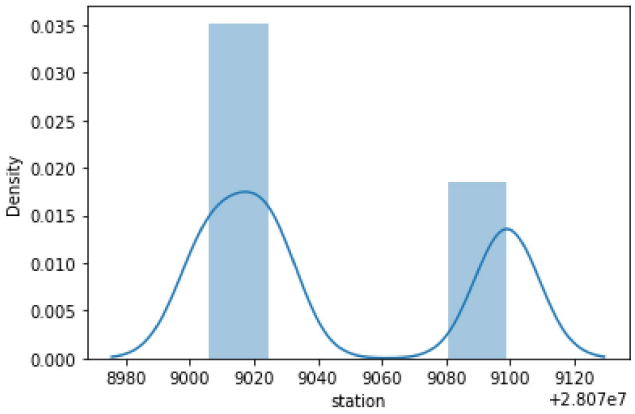
Out[219]: <seaborn.axisgrid.PairGrid at 0x210783cbb50>



In [220]:
```
1 sns.distplot(df1['station'])
2
```

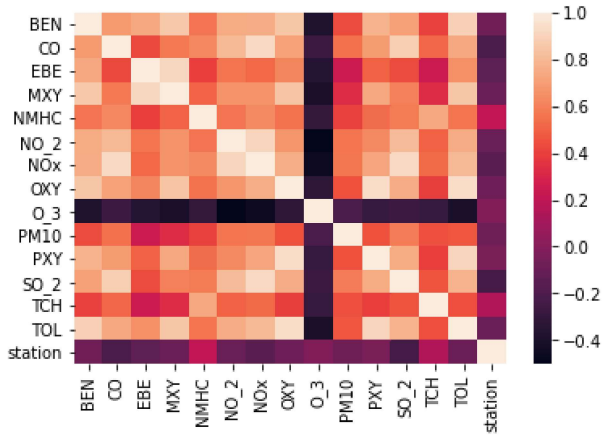
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[220]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [221]:
```
1 sns.heatmap(df1.corr())
```

Out[221]: <AxesSubplot:>



In [223]:
```
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 [224]:
```
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 [225]:
```
1 from sklearn.linear_model import LinearRegression
2 lr=LinearRegression()
3 lr.fit(x_train,y_train)
4
```

Out[225]: LinearRegression()

In [226]:
```
1 lr.intercept_
```

Out[226]: 28079004.572474387

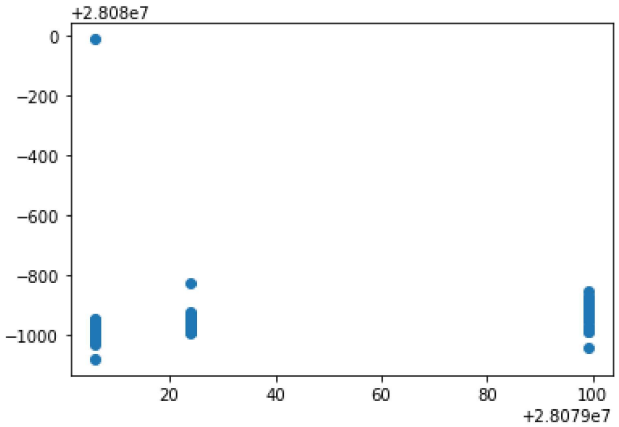
In [227]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[227]:

	Co-efficient
BEN	6.590204
CO	-81.808770
EBE	-14.591977
MXY	4.876039
NMHC	413.097377
NO_2	0.210447
NOx	-0.055137
OXY	-14.320955
O_3	0.114974
PM10	-0.055534
PXY	16.466320
SO_2	-3.901383
TCH	42.312890
TOL	-1.019305

In [228]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[228]: <matplotlib.collections.PathCollection at 0x2100767fb50>



In [229]:
```
lr.score(x_test,y_test)

```

Out[229]: -1.3341643673438681

In [230]:
```
lr.score(x_train,y_train)

```

Out[230]: 0.589093344280073

In [231]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [232]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[232]: Ridge(alpha=10)

In [233]:
```
rr.score(x_test,y_test)

```

Out[233]: 0.13760460097635652

In [234]:
```
rr.score(x_train,y_train)

```

Out[234]: 0.37694945407225344

In [235]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[235]: Lasso(alpha=10)

In [236]:
```
la.score(x_train,y_train)

```

Out[236]: 0.0509512270753808

In [237]:
```
la.score(x_test,y_test)

```

Out[237]: 0.029580305610412472

In [238]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[238]: ElasticNet()

In [239]:
```
en.coef_

```

Out[239]:
```
array([ 1.31736733, -0.4716478 , -3.34115106, -0.87339091,  1.6894006 ,
        0.13552175,  0.06646454,  0.        , -0.05142267,  0.02676611,
        4.92434463, -5.69115182,  1.74485048,  0.30232822])
```

In [240]:
```
en.intercept_

```

Out[240]: 28079066.78682329

In [241]:
```
prediction=en.predict(x_test)

```

In [242]:
```
en.score(x_test,y_test)
```

Out[242]: 0.11883640933508433

In [243]:
```
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)))

```

```
35.41511028869699
1475.9652277738023
38.418292879483886
```

In [244]:
```
from sklearn.linear_model import LogisticRegression

```

In [245]:
```
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 [246]:
```
feature_matrix.shape

```

Out[246]: (1000, 14)

In [247]:
```
target_vector.shape

```

Out[247]: (1000,)

In [248]:
```
from sklearn.preprocessing import StandardScaler
```

In [249]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [250]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[250]: LogisticRegression(max_iter=10000)

In [251]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

```

In [252]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079006]
```

In [253]:
```
logr.classes_

```

Out[253]: array([28079006, 28079024, 28079099], dtype=int64)

In [254]:
```
logr.score(fs,target_vector)

```

Out[254]: 0.953

In [255]:
```
logr.predict_proba(observation)[0][0]

```

Out[255]: 0.9999999999997267

In [256]:
```
logr.predict_proba(observation)

```

Out[256]: array([[1.00000000e+00, 1.43973542e-56, 2.73340992e-13]])

In [257]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [258]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[258]: RandomForestClassifier()

In [259]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [260]:
```
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[260]:
```
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 [261]:
```
grid_search.best_score_
```

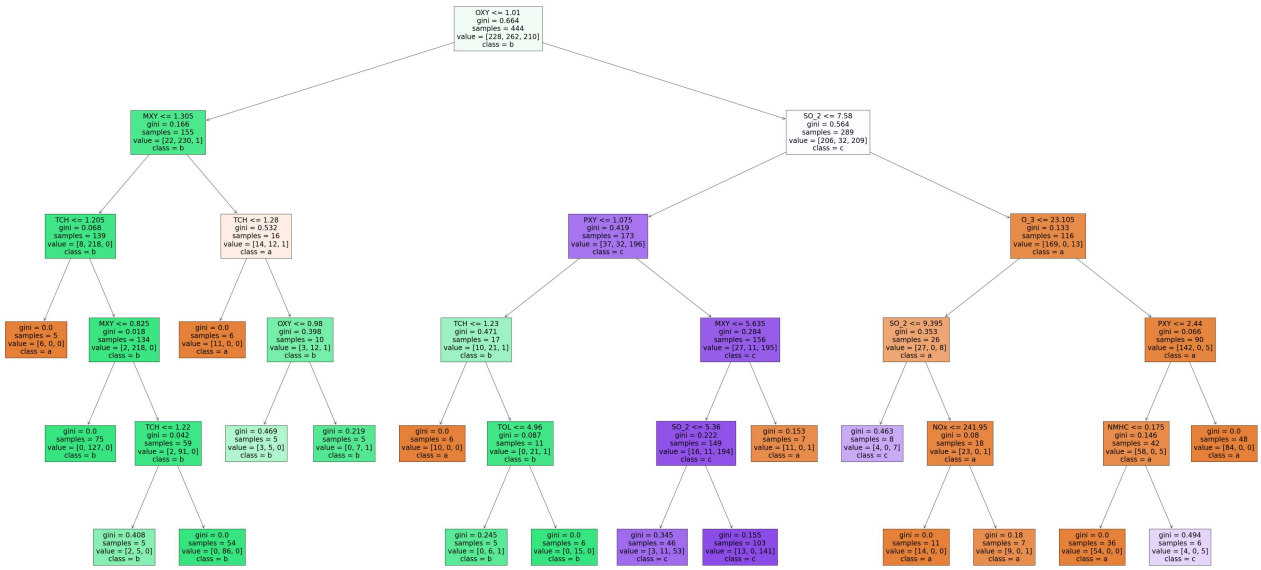
Out[261]:
```
0.9342857142857143
```

In [262]:
```
rfc_best=grid_search.best_estimator_
```

In [263]:

```
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[263]:
```
[Text(1770.2068965517242, 1993.2, 'OXY <= 1.01\ngini = 0.664\nsamples = 444\nvalue = [228, 262, 210]\nclass = b'),
 Text(615.7241379310345, 1630.8000000000002, 'MXY <= 1.305\ngini = 0.166\nsamples = 155\nvalue = [22, 230, 1]\nclass = b'),
 Text(307.86206896551727, 1268.4, 'TCH <= 1.205\ngini = 0.068\nsamples = 139\nvalue = [8, 218, 0]\nclass = b'),
 Text(153.93103448275863, 906.0, 'gini = 0.0\nsamples = 5\nvalue = [6, 0, 0]\nclass = a'),
 Text(461.79310344827593, 906.0, 'MXY <= 0.825\ngini = 0.018\nsamples = 134\nvalue = [2, 218, 0]\nclass = b'),
 Text(307.86206896551727, 543.5999999999999, 'gini = 0.0\nsamples = 75\nvalue = [0, 127, 0]\nclass = b'),
 Text(615.7241379310345, 543.5999999999999, 'TCH <= 1.22\ngini = 0.042\nsamples = 59\nvalue = [2, 91, 0]\nclass = b'),
 Text(461.79310344827593, 181.19999999999982, 'gini = 0.408\nsamples = 5\nvalue = [2, 5, 0]\nclass = b'),
 Text(769.6551724137931, 181.19999999999982, 'gini = 0.0\nsamples = 54\nvalue = [0, 86, 0]\nclass = b'),
 Text(923.5862068965519, 1268.4, 'TCH <= 1.28\ngini = 0.532\nsamples = 16\nvalue = [14, 12, 1]\nclass = a'),
 Text(769.6551724137931, 906.0, 'gini = 0.0\nsamples = 6\nvalue = [11, 0, 0]\nclass = a'),
 Text(1077.5172413793105, 906.0, 'OXY <= 0.98\ngini = 0.398\nsamples = 10\nvalue = [3, 12, 1]\nclass = b'),
 Text(923.5862068965519, 543.5999999999999, 'gini = 0.469\nsamples = 5\nvalue = [3, 5, 0]\nclass = b'),
 Text(1231.448275862069, 543.5999999999999, 'gini = 0.219\nsamples = 5\nvalue = [0, 7, 1]\nclass = b'),
 Text(2924.689655172414, 1630.8000000000002, 'SO_2 <= 7.58\ngini = 0.564\nsamples = 289\nvalue = [206, 32, 209]\nclass = c'),
 Text(2155.034482758621, 1268.4, 'PXY <= 1.075\ngini = 0.419\nsamples = 173\nvalue = [37, 32, 196]\nclass = c'),
 Text(1693.2413793103449, 906.0, 'TCH <= 1.23\ngini = 0.471\nsamples = 17\nvalue = [10, 21, 1]\nclass = b'),
 Text(1539.3103448275863, 543.5999999999999, 'gini = 0.0\nsamples = 6\nvalue = [10, 0, 0]\nclass = a'),
 Text(1847.1724137931037, 543.5999999999999, 'TOL <= 4.96\ngini = 0.087\nsamples = 11\nvalue = [0, 21, 1]\nclass = b'),
 Text(1693.2413793103449, 181.19999999999982, 'gini = 0.245\nsamples = 5\nvalue = [0, 6, 1]\nclass = b'),
 Text(2001.1034482758623, 181.19999999999982, 'gini = 0.0\nsamples = 6\nvalue = [0, 15, 0]\nclass = b'),
 Text(2616.8275862068967, 906.0, 'MXY <= 5.635\ngini = 0.284\nsamples = 156\nvalue = [27, 11, 195]\nclass = c'),
 Text(2462.896551724138, 543.5999999999999, 'SO_2 <= 5.36\ngini = 0.222\nsamples = 149\nvalue = [16, 11, 194]\nclass = c'),
 Text(2308.9655172413795, 181.19999999999982, 'gini = 0.345\nsamples = 46\nvalue = [3, 11, 53]\nclass = c'),
 Text(2616.8275862068967, 181.19999999999982, 'gini = 0.155\nsamples = 103\nvalue = [13, 0, 141]\nclass = c'),
 Text(2770.7586206896553, 543.5999999999999, 'gini = 0.153\nsamples = 7\nvalue = [11, 0, 1]\nclass = a'),
 Text(3694.3448275862074, 1268.4, 'O_3 <= 23.105\ngini = 0.133\nsamples = 116\nvalue = [169, 0, 13]\nclass = a'),
 Text(3232.551724137931, 906.0, 'SO_2 <= 9.395\ngini = 0.353\nsamples = 26\nvalue = [27, 0, 8]\nclass = a'),
 Text(3078.6206896551726, 543.5999999999999, 'gini = 0.463\nsamples = 8\nvalue = [4, 0, 7]\nclass = c'),
 Text(3386.4827586206898, 543.5999999999999, 'NOx <= 241.95\ngini = 0.08\nsamples = 18\nvalue = [23, 0, 1]\nclass = a'),
 Text(3232.551724137931, 181.19999999999982, 'gini = 0.0\nsamples = 11\nvalue = [14, 0, 0]\nclass = a'),
 Text(3540.4137931034484, 181.19999999999982, 'gini = 0.18\nsamples = 7\nvalue = [9, 0, 1]\nclass = a'),
 Text(4156.137931034483, 906.0, 'PXY <= 2.44\ngini = 0.066\nsamples = 90\nvalue = [142, 0, 5]\nclass = a'),
 Text(4002.2068965517246, 543.5999999999999, 'NMHC <= 0.175\ngini = 0.146\nsamples = 42\nvalue = [58, 0, 5]\nclass = a'),
 Text(3848.275862068966, 181.19999999999982, 'gini = 0.0\nsamples = 36\nvalue = [54, 0, 0]\nclass = a'),
 Text(4156.137931034483, 181.19999999999982, 'gini = 0.494\nsamples = 6\nvalue = [4, 0, 5]\nclass = c'),
 Text(4310.068965517242, 543.5999999999999, 'gini = 0.0\nsamples = 48\nvalue = [84, 0, 0]\nclass = a')]
```



Conclusion

Linear Regression=0.589093344280073

Ridge Regression=0.37694945407225344

Lasso Regression=0.0509512270753808

ElasticNet Regression=0.11883640933508433

Logistic Regression=0.953

Random Forest=0.9342857142857143

Logistic Regression Is Suitable for this Dataset