Vijay(P12) 3/08/2023

In [337]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [338]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_201
df
```

Out[338]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	28079008
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28079011
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28079056
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28079057
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28079058
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28079059
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28079060

210720 rows × 14 columns

In [339]:
```
df=df.dropna()
```

In [340]:
```
df.columns

```

Out[340]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [341]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10916 entries, 6 to 210702
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     10916 non-null  object 
 1   BEN      10916 non-null  float64
 2   CO       10916 non-null  float64
 3   EBE      10916 non-null  float64
 4   NMHC     10916 non-null  float64
 5   NO       10916 non-null  float64
 6   NO_2     10916 non-null  float64
 7   O_3      10916 non-null  float64
 8   PM10     10916 non-null  float64
 9   PM25     10916 non-null  float64
 10  SO_2     10916 non-null  float64
 11  TCH      10916 non-null  float64
 12  TOL      10916 non-null  float64
 13  station  10916 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.2+ MB
```

In [342]:
```
1 data=df[['BEN', 'CO','station','PM25']]
2 data
3
```

Out[342]:

	BEN	CO	station	PM25
6	0.4	0.2	28079024	7.0
30	0.4	0.2	28079024	5.0
54	0.4	0.2	28079024	4.0
78	0.3	0.2	28079024	5.0
102	0.4	0.2	28079024	5.0
...	...	...	...	...
210654	0.6	0.3	28079024	21.0
210673	2.0	0.4	28079008	25.0
210678	0.7	0.3	28079024	18.0
210697	1.5	0.4	28079008	21.0
210702	0.6	0.3	28079024	16.0

10916 rows × 4 columns

In [343]:
```
df=df.head(1500)
df
```

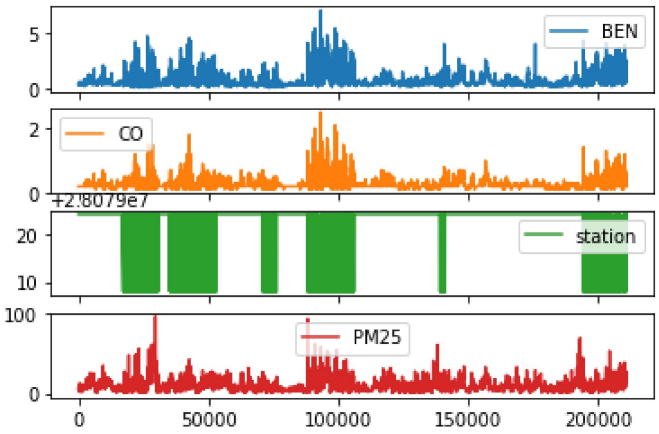
Out[343]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6	28079024
30	2012-09-01 02:00:00	0.4	0.2	0.7	0.24	1.0	5.0	55.0	5.0	5.0	2.0	1.33	0.5	28079024
54	2012-09-01 03:00:00	0.4	0.2	0.7	0.24	1.0	4.0	56.0	6.0	4.0	2.0	1.33	0.5	28079024
78	2012-09-01 04:00:00	0.3	0.2	0.7	0.25	1.0	5.0	54.0	6.0	5.0	2.0	1.34	0.4	28079024
102	2012-09-01 05:00:00	0.4	0.2	0.7	0.24	1.0	3.0	53.0	8.0	5.0	2.0	1.33	0.5	28079024
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
27822	2012-08-19 08:00:00	0.6	0.2	1.6	0.23	1.0	12.0	46.0	80.0	30.0	3.0	1.34	1.0	28079024
27841	2012-08-19 09:00:00	0.2	0.3	0.5	0.16	6.0	26.0	38.0	60.0	30.0	2.0	1.45	1.2	28079008
27846	2012-08-19 09:00:00	0.6	0.2	1.8	0.23	1.0	13.0	43.0	82.0	27.0	3.0	1.33	1.2	28079024
27865	2012-08-19 10:00:00	0.2	0.3	0.4	0.16	6.0	21.0	40.0	52.0	25.0	2.0	1.31	1.5	28079008
27870	2012-08-19 10:00:00	0.7	0.2	1.7	0.23	2.0	13.0	45.0	72.0	19.0	3.0	1.34	1.0	28079024

1500 rows × 14 columns

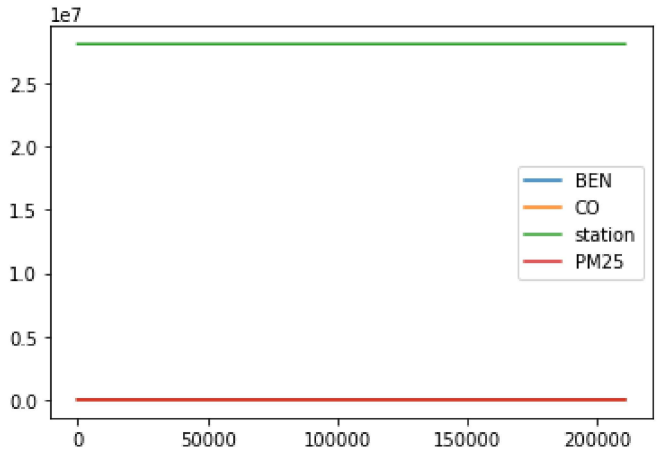
In [344]:
```
data.plot.line(subplots=True)
```

Out[344]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
      dtype=object)
```



In [345]:
```
1 data.plot.line()
2
```

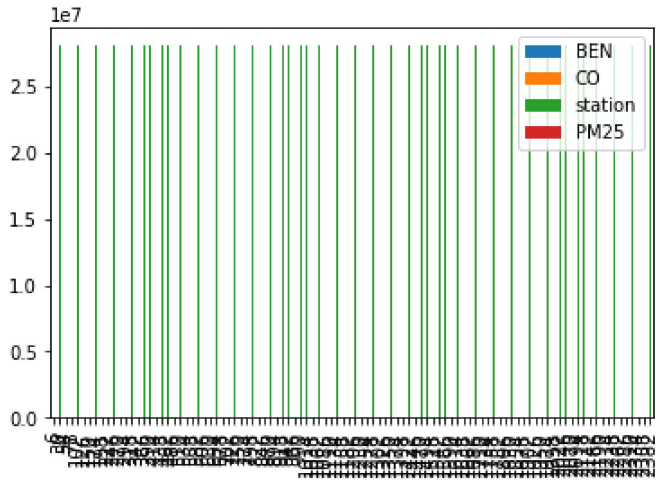
Out[345]: <AxesSubplot:>



In [346]:
```
1 b=data[0:100]
2
```

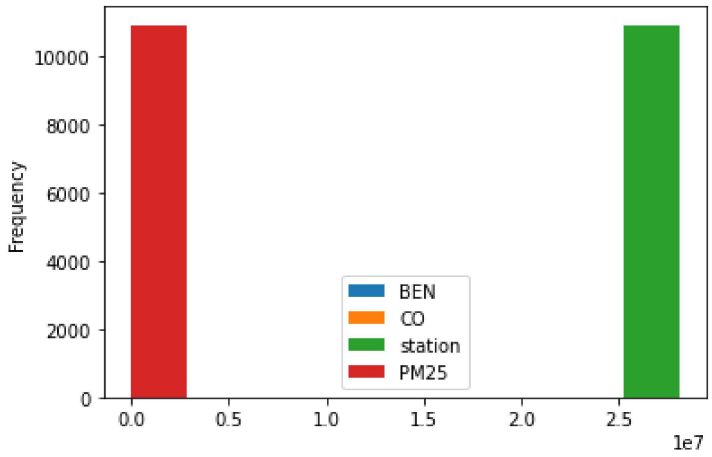
In [347]:
```
1 b.plot.bar()
```

Out[347]: <AxesSubplot:>



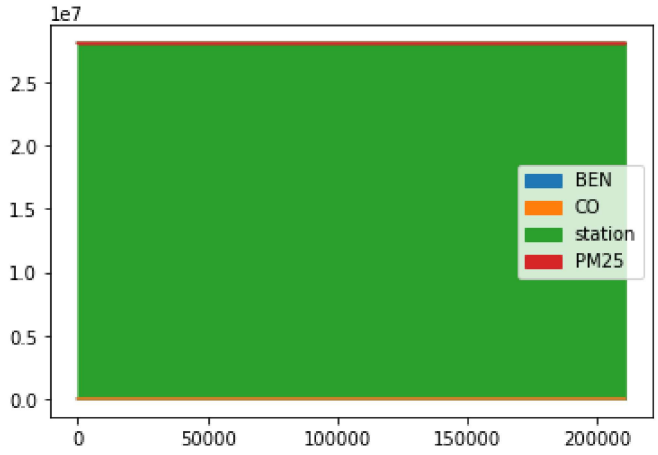
In [348]:
```
1 data.plot.hist()
2
```

Out[348]: <AxesSubplot:ylabel='Frequency'>



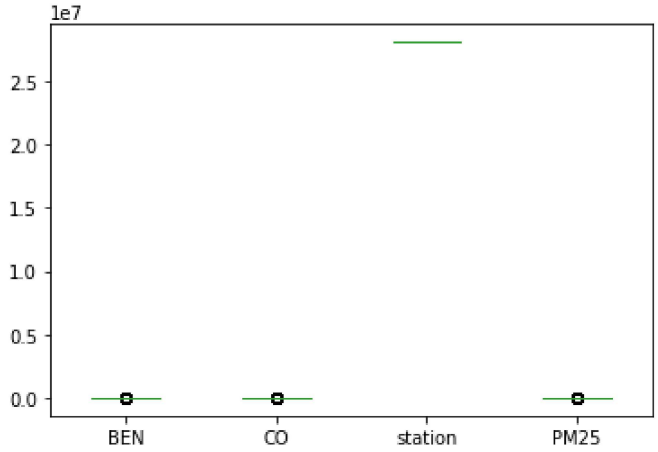
In [349]:
```
1 data.plot.area()
```

Out[349]: <AxesSubplot:>



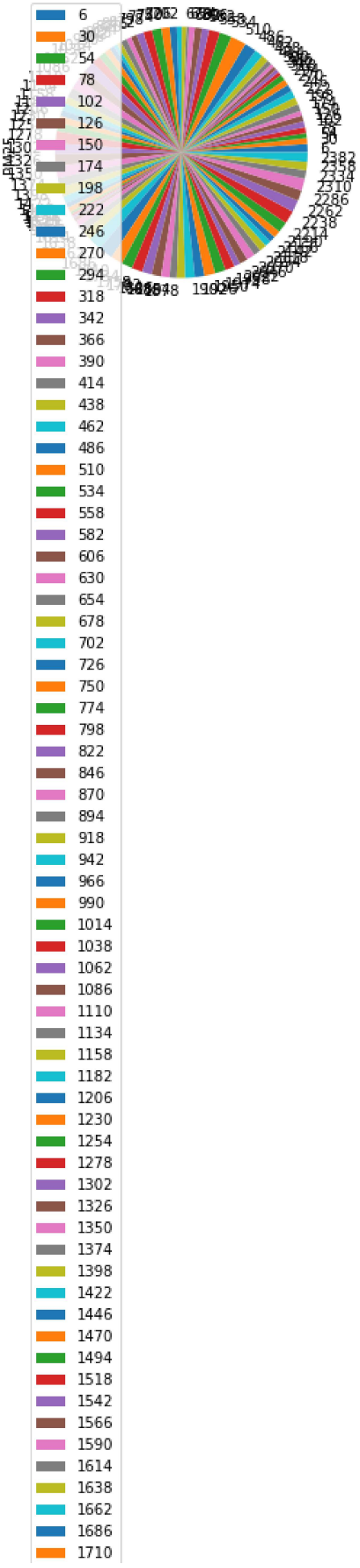
In [350]:
```
1 data.plot.box()
```

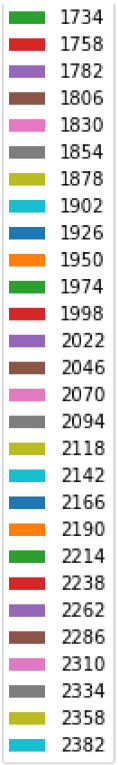
Out[350]: <AxesSubplot:>



In [351]:
```
1 b.plot.pie(y='PM25' )
```

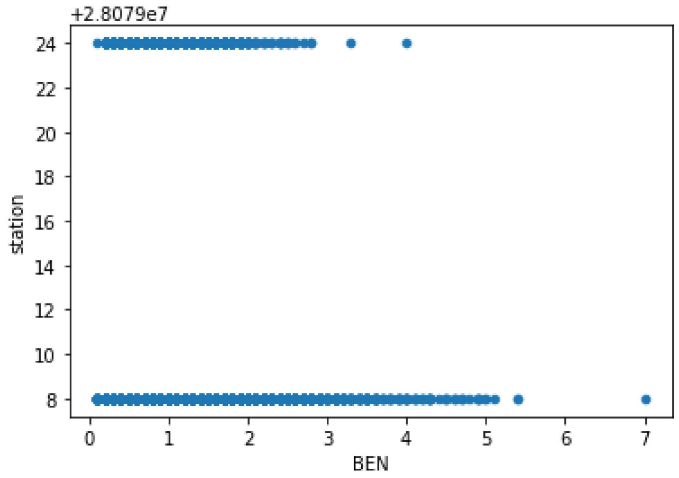
Out[351]:
```
<AxesSubplot:ylabel='PM25'>
```


6
30
54
78
102
126
150
174
198
222
246
270
294
318
342
366
390
414
438
462
486
510
534
558
582
606
630
654
678
702
726
750
774
798
822
846
870
894
918
942
966
990
1014
1038
1062
1086
1110
1134
1158
1182
1206
1230
1254
1278
1302
1326
1350
1374
1398
1422
1446
1470
1494
1518
1542
1566
1590
1614
1638
1662
1686
1710



In [352]:
```
1 data.plot.scatter(x='BEN' ,y='station')
2
```

Out[352]: <AxesSubplot:xlabel='BEN', ylabel='station'>



In [353]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1500 entries, 6 to 27870
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1500 non-null   object 
 1   BEN      1500 non-null   float64
 2   CO       1500 non-null   float64
 3   EBE      1500 non-null   float64
 4   NMHC     1500 non-null   float64
 5   NO       1500 non-null   float64
 6   NO_2     1500 non-null   float64
 7   O_3      1500 non-null   float64
 8   PM10     1500 non-null   float64
 9   PM25     1500 non-null   float64
 10  SO_2     1500 non-null   float64
 11  TCH      1500 non-null   float64
 12  TOL      1500 non-null   float64
 13  station  1500 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 175.8+ KB
```

In [354]:
```
1 df.describe()
2 
```

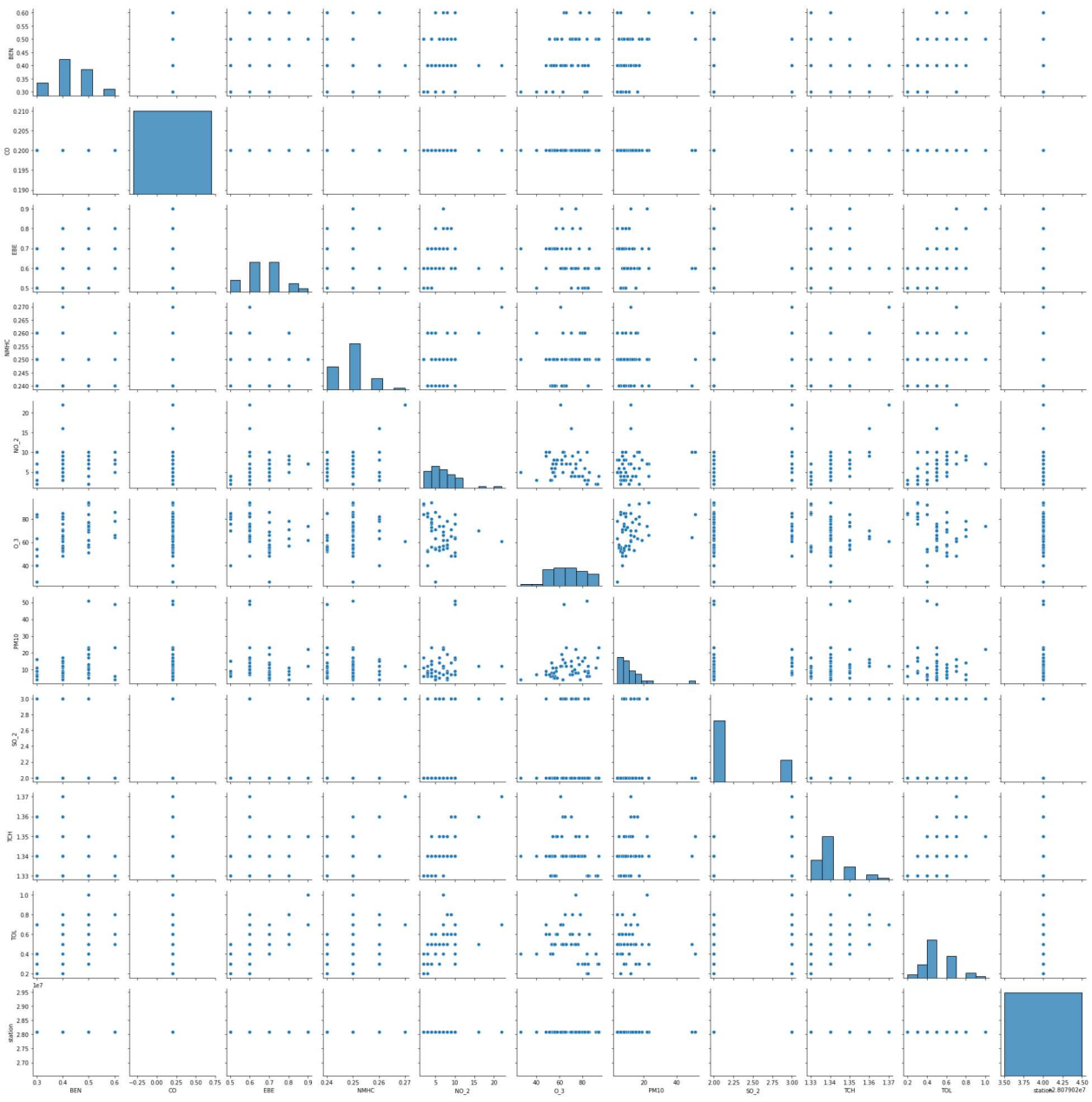
Out[354]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000
mean	0.657067	0.243067	1.350800	0.239393	7.233333	25.780000	56.140667	31.410000
std	0.487752	0.111566	0.722612	0.051479	17.286086	24.727594	27.533270	28.376117
min	0.100000	0.100000	0.400000	0.130000	1.000000	1.000000	2.000000	1.000000
25%	0.400000	0.200000	0.900000	0.210000	1.000000	7.000000	37.000000	13.000000
50%	0.500000	0.200000	1.200000	0.230000	1.000000	19.000000	56.000000	22.500000
75%	0.700000	0.300000	1.600000	0.260000	7.000000	37.000000	76.000000	37.000000
max	4.700000	1.500000	9.300000	0.580000	227.000000	225.000000	146.000000	267.000000

In [355]:
```
1 df1=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2',  'O_3',
2  'PM10', 'SO_2', 'TCH', 'TOL','station']]
```

In [356]:
```
1 sns.pairplot(df1[0:50])
```

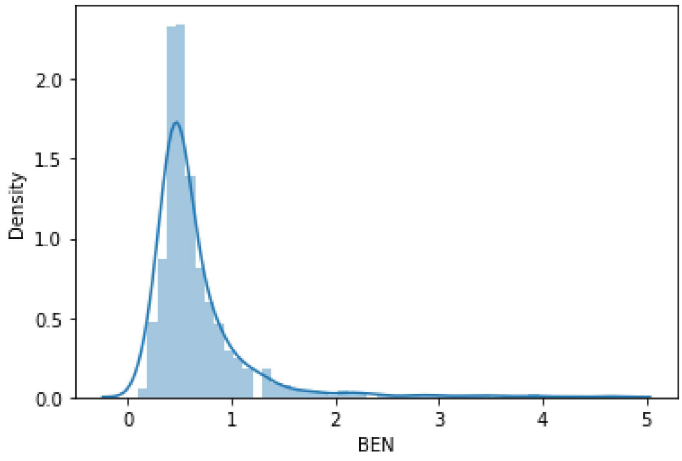
Out[356]: <seaborn.axisgrid.PairGrid at 0x27d6d688e50>



In [357]:
```
1 sns.distplot(df1['BEN'])
2
```

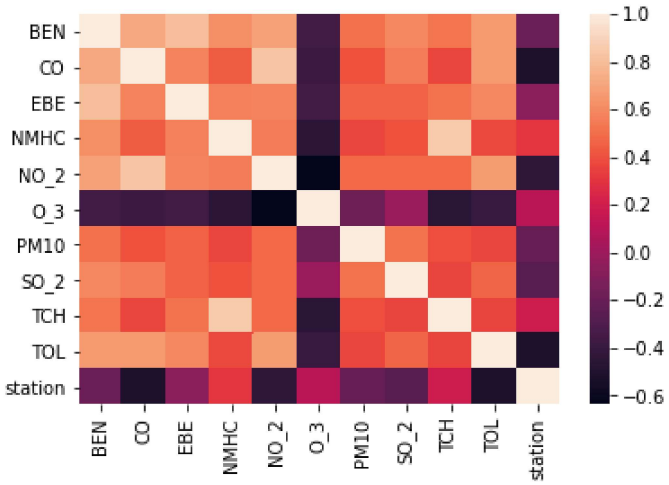
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function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

Out[357]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



In [358]:
```
1 sns.heatmap(df1.corr())
```

Out[358]: <AxesSubplot:>



In [359]:
```
1 x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2  'PM10', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
4
```

In [360]:
```
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 [361]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

```

Out[361]:
```
LinearRegression()
```

In [362]:
```
lr.intercept_
```

Out[362]:
```
28079017.623623546
```

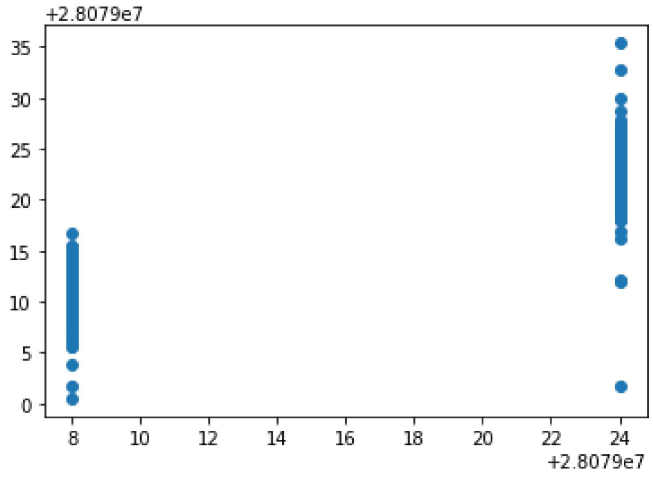
In [363]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[363]:

	Co-efficient
BEN	0.567602
CO	-20.191971
EBE	1.789918
NMHC	129.590220
NO_2	-0.115752
O_3	-0.008981
PM10	-0.012853
SO_2	-0.528702
TCH	-14.532320
TOL	-1.006126

In [364]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[364]:
```
<matplotlib.collections.PathCollection at 0x27d77830850>
```



In [365]:
```
lr.score(x_test,y_test)

```

Out[365]:
```
0.8150266581010389
```

In [366]:
```
lr.score(x_train,y_train)

```

Out[366]: 0.7828180874509963

In [367]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [368]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[368]: Ridge(alpha=10)

In [369]:
```
rr.score(x_test,y_test)

```

Out[369]: 0.598087691334901

In [370]:
```
rr.score(x_train,y_train)

```

Out[370]: 0.5903305288132328

In [371]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[371]: Lasso(alpha=10)

In [372]:
```
la.score(x_train,y_train)

```

Out[372]: 0.2229387550086066

In [373]:
```
la.score(x_test,y_test)

```

Out[373]: 0.20382996900983497

In [374]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[374]: ElasticNet()

In [375]:
```
en.coef_

```

Out[375]:
```
array([ 0.05177873, -0.        ,  0.87501109,  0.        , -0.12782712,
       -0.07033409,  0.01319515,  0.        ,  0.        , -0.85215596])
```

In [376]:
```
en.intercept_

```

Out[376]: 28079027.11581178

In [377]:
```
prediction=en.predict(x_test)

```

In [378]:
```
en.score(x_test,y_test)
```

Out[378]: 0.3512897622154286

In [379]:
```
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)))

```

```
4.666075804101096
34.576146754667846
5.880148531684199
```

In [380]:
```
from sklearn.linear_model import LogisticRegression

```

In [381]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
 'PM10',  'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']

```

In [382]:
```
feature_matrix.shape

```

Out[382]: (1500, 10)

In [383]:
```
target_vector.shape

```

Out[383]: (1500,)

In [384]:
```
from sklearn.preprocessing import StandardScaler
```

In [385]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [386]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[386]: LogisticRegression(max_iter=10000)

In [387]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]

```

In [388]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079008]
```

In [389]:
```
logr.classes_

```

Out[389]:
```
array([28079008, 28079024], dtype=int64)
```

In [390]:
```
logr.score(fs,target_vector)

```

Out[390]:
```
0.9826666666666667
```

In [391]:
```
logr.predict_proba(observation)[0][0]

```

Out[391]:
```
0.9999999606943034
```

In [392]:
```
logr.predict_proba(observation)

```

Out[392]:
```
array([[9.99999961e-01, 3.93056966e-08]])
```

In [393]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [394]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[394]:
```
RandomForestClassifier()
```

In [395]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [396]:
```
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[396]:
```
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')
```

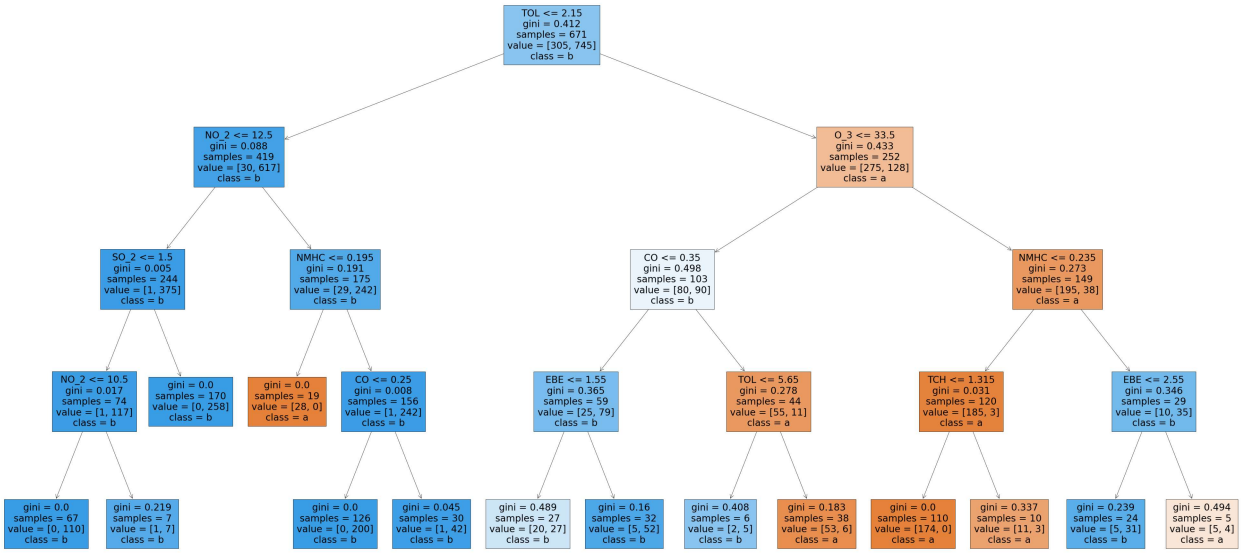
In [397]:
```
grid_search.best_score_
```

Out[397]:
```
0.9866666666666667
```

In [398]:
```
rfc_best=grid_search.best_estimator_
```

In [399]:
```
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']
```

Out[399]:
```
[Text(1974.4615384615383, 1956.96, 'TOL <= 2.15\ngini = 0.412\nsamples = 671\nvalue = [305, 745]\nclass = b'),
 Text(858.4615384615383, 1522.0800000000002, 'NO_2 <= 12.5\ngini = 0.088\nsamples = 419\nvalue = [30, 617]\nclass = b'),
 Text(515.0769230769231, 1087.2, 'SO_2 <= 1.5\ngini = 0.005\nsamples = 244\nvalue = [1, 375]\nclass = b'),
 Text(343.38461538461536, 652.3200000000002, 'NO_2 <= 10.5\ngini = 0.017\nsamples = 74\nvalue = [1, 117]\nclass = b'),
 Text(171.69230769230768, 217.44000000000005, 'gini = 0.0\nsamples = 67\nvalue = [0, 110]\nclass = b'),
 Text(515.0769230769231, 217.44000000000005, 'gini = 0.219\nsamples = 7\nvalue = [1, 7]\nclass = b'),
 Text(686.7692307692307, 652.3200000000002, 'gini = 0.0\nsamples = 170\nvalue = [0, 258]\nclass = b'),
 Text(1201.8461538461538, 1087.2, 'NMHC <= 0.195\ngini = 0.191\nsamples = 175\nvalue = [29, 242]\nclass = b'),
 Text(1030.1538461538462, 652.3200000000002, 'gini = 0.0\nsamples = 19\nvalue = [28, 0]\nclass = a'),
 Text(1373.5384615384614, 652.3200000000002, 'CO <= 0.25\ngini = 0.008\nsamples = 156\nvalue = [1, 242]\nclass = b'),
 Text(1201.8461538461538, 217.44000000000005, 'gini = 0.0\nsamples = 126\nvalue = [0, 200]\nclass = b'),
 Text(1545.230769230769, 217.44000000000005, 'gini = 0.045\nsamples = 30\nvalue = [1, 42]\nclass = b'),
 Text(3090.461538461538, 1522.0800000000002, 'O_3 <= 33.5\ngini = 0.433\nsamples = 252\nvalue = [275, 128]\nclass = a'),
 Text(2403.6923076923076, 1087.2, 'CO <= 0.35\ngini = 0.498\nsamples = 103\nvalue = [80, 90]\nclass = b'),
 Text(2060.3076923076924, 652.3200000000002, 'EBE <= 1.55\ngini = 0.365\nsamples = 59\nvalue = [25, 79]\nclass = b'),
 Text(1888.6153846153845, 217.44000000000005, 'gini = 0.489\nsamples = 27\nvalue = [20, 27]\nclass = b'),
 Text(2232.0, 217.44000000000005, 'gini = 0.16\nsamples = 32\nvalue = [5, 52]\nclass = b'),
 Text(2747.076923076923, 652.3200000000002, 'TOL <= 5.65\ngini = 0.278\nsamples = 44\nvalue = [55, 11]\nclass = a'),
 Text(2575.3846153846152, 217.44000000000005, 'gini = 0.408\nsamples = 6\nvalue = [2, 5]\nclass = b'),
 Text(2918.7692307692305, 217.44000000000005, 'gini = 0.183\nsamples = 38\nvalue = [53, 6]\nclass = a'),
 Text(3777.230769230769, 1087.2, 'NMHC <= 0.235\ngini = 0.273\nsamples = 149\nvalue = [195, 38]\nclass = a'),
 Text(3433.8461538461534, 652.3200000000002, 'TCH <= 1.315\ngini = 0.031\nsamples = 120\nvalue = [185, 3]\nclass = a'),
 Text(3262.1538461538457, 217.44000000000005, 'gini = 0.0\nsamples = 110\nvalue = [174, 0]\nclass = a'),
 Text(3605.5384615384614, 217.44000000000005, 'gini = 0.337\nsamples = 10\nvalue = [11, 3]\nclass = a'),
 Text(4120.615384615385, 652.3200000000002, 'EBE <= 2.55\ngini = 0.346\nsamples = 29\nvalue = [10, 35]\nclass = b'),
 Text(3948.9230769230767, 217.44000000000005, 'gini = 0.239\nsamples = 24\nvalue = [5, 31]\nclass = b'),
 Text(4292.307692307692, 217.44000000000005, 'gini = 0.494\nsamples = 5\nvalue = [5, 4]\nclass = a')]
```



Conclusion

Linear Regression=0.7828180874509963

Ridge Regression=0.5903305288132328

Lasso Regression=0.2229387550086066

ElasticNet Regression=0.3512897622154286

Logistic Regression=0.9826666666666667

Random Forest=0.9866666666666667

Random Forest is Suitable for this Dataset