Vijay(P16) 03/08/2023

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2016.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28079008
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0	28079011
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN	28079056
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN	28079057
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN	28079058
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN	28079059
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN	28079060

209496 rows × 14 columns

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16932 non-null  object 
 1   BEN      16932 non-null  float64
 2   CO       16932 non-null  float64
 3   EBE      16932 non-null  float64
 4   NMHC     16932 non-null  float64
 5   NO       16932 non-null  float64
 6   NO_2     16932 non-null  float64
 7   O_3      16932 non-null  float64
 8   PM10     16932 non-null  float64
 9   PM25     16932 non-null  float64
 10  SO_2     16932 non-null  float64
 11  TCH      16932 non-null  float64
 12  TOL      16932 non-null  float64
 13  station  16932 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

In [6]:
```
data=df[['BEN', 'TOL', 'TCH']]
data
```

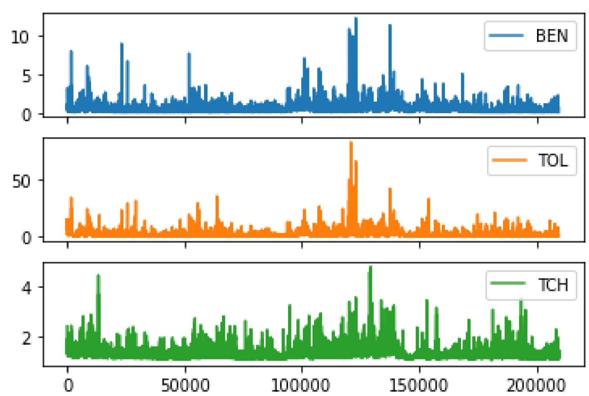
Out[6]:

	BEN	TOL	TCH
1	3.1	14.4	2.44
6	0.7	5.0	1.35
25	2.7	15.0	2.30
30	0.7	5.0	1.35
49	1.7	10.7	1.95
...	...	...	...
209430	0.1	0.2	1.15
209449	0.6	1.9	1.48
209454	0.1	0.3	1.15
209473	0.6	1.9	1.50
209478	0.1	0.2	1.15

16932 rows × 3 columns

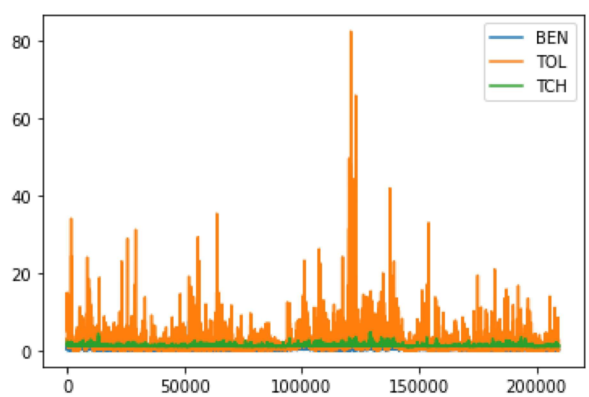
In [7]:
```
data.plot.line(subplots=True)
```

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



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

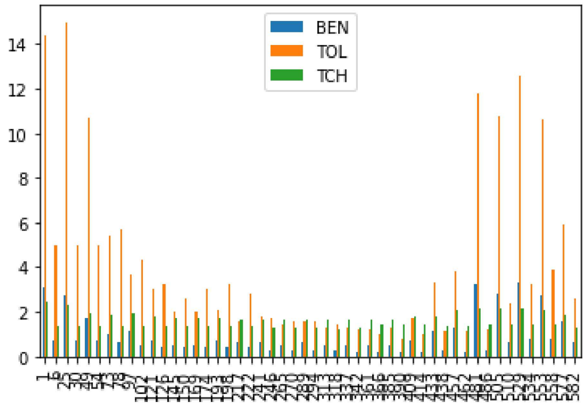
Out[8]:
```
<AxesSubplot:>
```



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

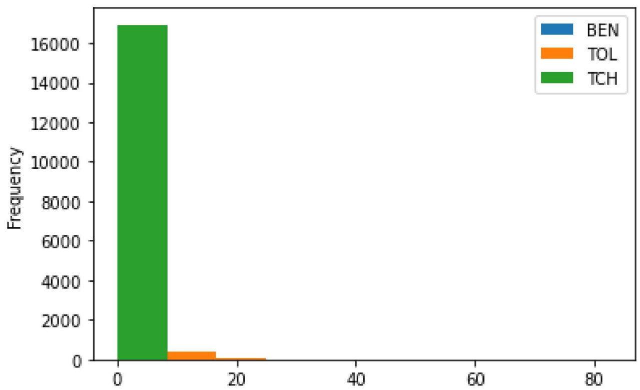
In [10]:
```
1 b.plot.bar()
```

Out[10]: <AxesSubplot:>



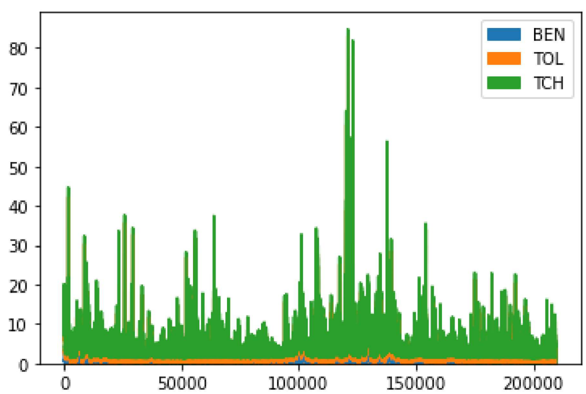
In [11]:
```
1 data.plot.hist()
```

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



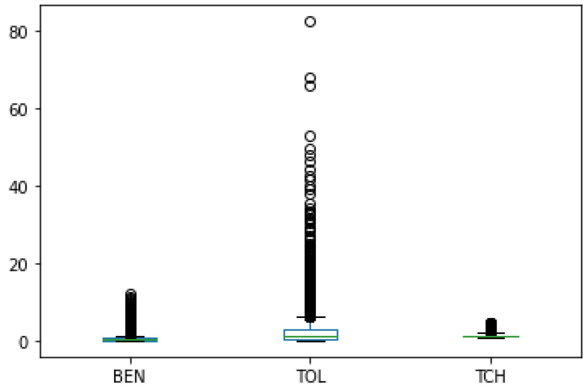
In [12]:
```
1 data.plot.area()
```

Out[12]: <AxesSubplot:>



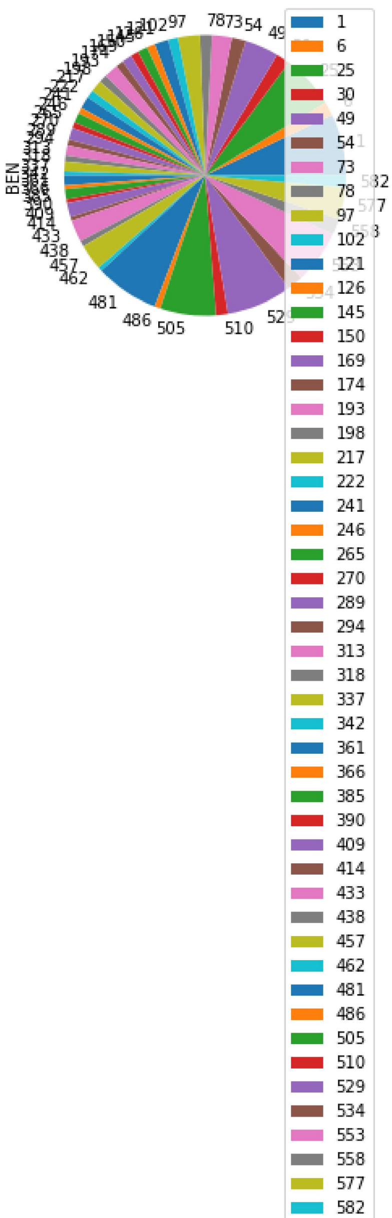
In [13]:
```
data.plot.box()
```

Out[13]: <AxesSubplot:>



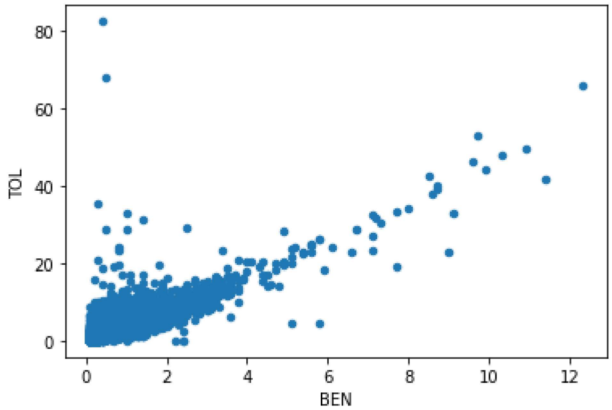
In [14]:
```
b.plot.pie(y='BEN' )
```

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



In [15]:
```
data.plot.scatter(x='BEN' ,y='TOL')
```

Out[15]:
```
<AxesSubplot:xlabel='BEN', ylabel='TOL'>
```



In [16]:
```
df.describe()
```

Out[16]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	
count	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.(
mean	0.537970	0.349941	0.298955	0.099913	20.815734	39.373376	48.118474	19.248110	10.
std	0.599479	0.203807	0.450204	0.079850	40.986063	31.170307	32.560277	18.509093	8.(
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	1.000000	1.000000	0.(
25%	0.200000	0.200000	0.100000	0.050000	1.000000	14.000000	21.000000	9.000000	5.(
50%	0.400000	0.300000	0.200000	0.090000	7.000000	34.000000	46.000000	15.000000	8.(
75%	0.700000	0.400000	0.300000	0.120000	23.000000	58.000000	69.000000	24.000000	14.(
max	12.300000	4.500000	13.500000	2.210000	829.000000	319.000000	181.000000	367.000000	215.(

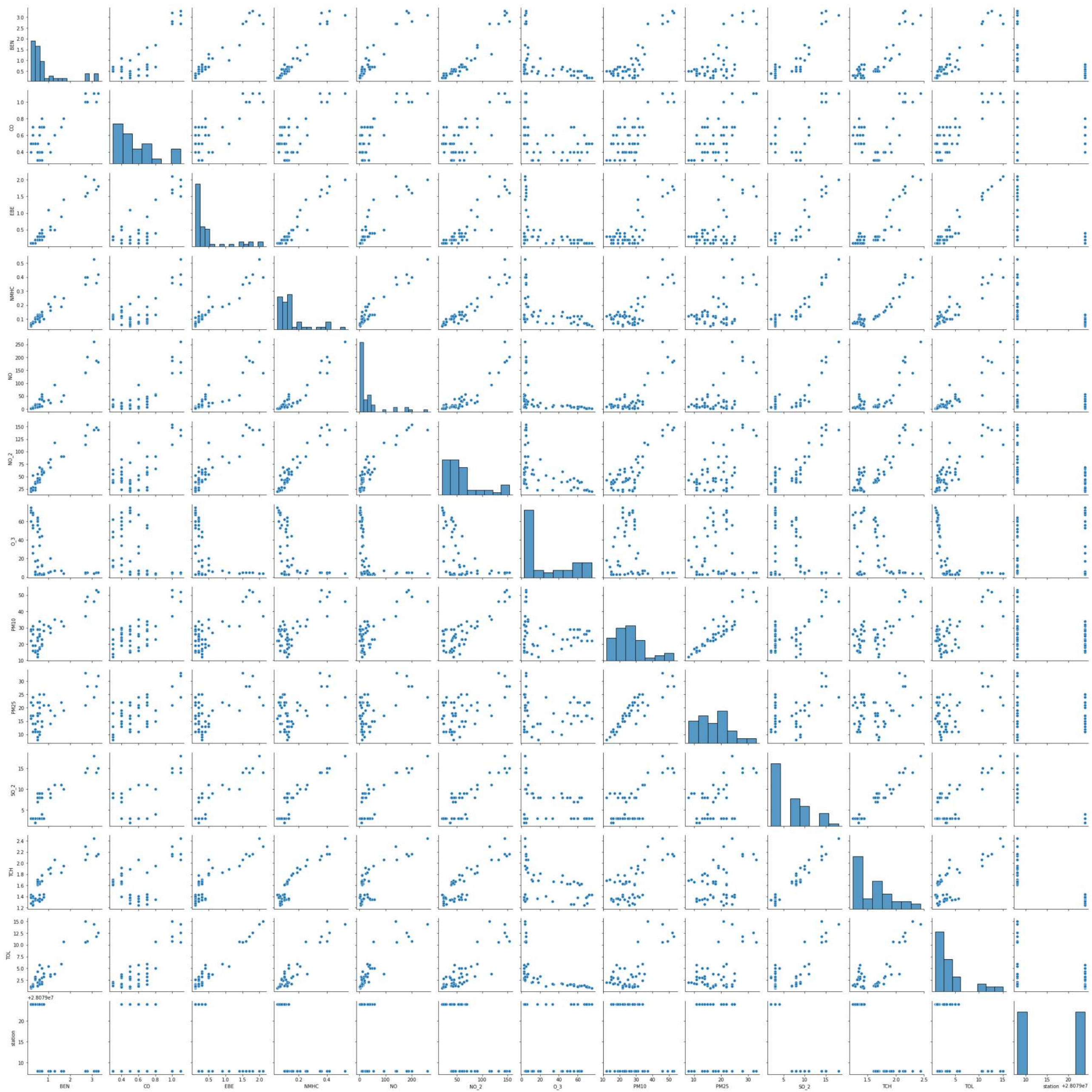
In [17]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
'SO_2', 'TCH', 'TOL', 'station']]
```

In [18]:

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

Out[18]:

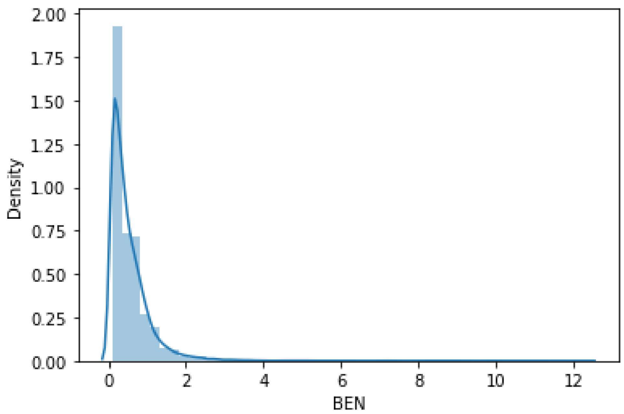
```
<seaborn.axisgrid.PairGrid at 0x2d2f4a702e0>
```



In [19]:
```
sns.distplot(df1['BEN'])
```

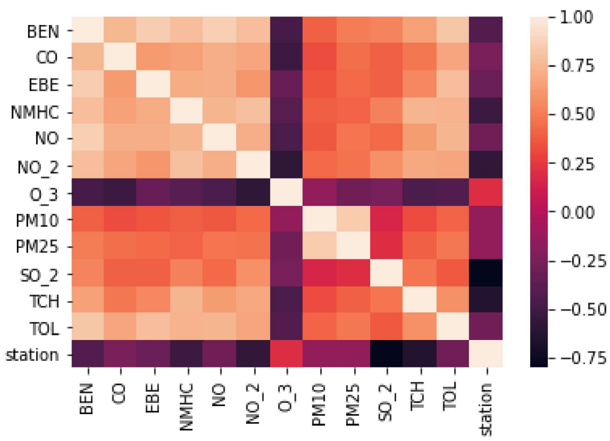
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[19]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



In [20]:
```
sns.heatmap(df1.corr())
```

Out[20]: <AxesSubplot:>



In [21]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [22]:
```
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 [23]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[23]: LinearRegression()

In [24]:
```
lr.intercept_
```

Out[24]: 28079042.32381048

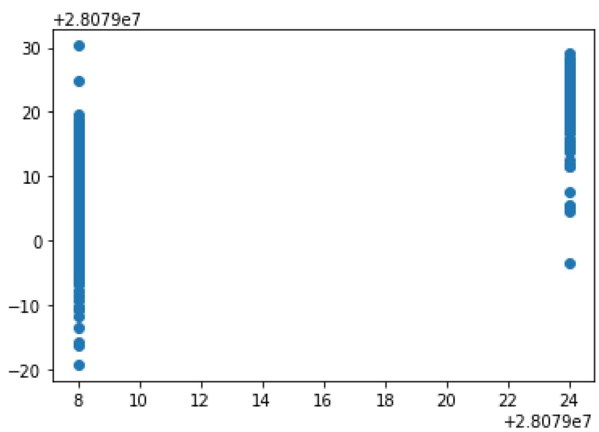
In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

	Co-efficient
BEN	-1.843533
CO	4.911084
EBE	0.560646
NMHC	0.870575
NO	0.068100
NO_2	-0.062596
O_3	-0.023603
PM10	-0.012790
PM25	0.089795
SO_2	-0.811247
TCH	-14.337630
TOL	0.184437

In [26]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[26]: <matplotlib.collections.PathCollection at 0x2d280b8a310>



In [27]:
```
lr.score(x_test,y_test)
```

Out[27]: 0.8220137982748653

In [28]:
```
lr.score(x_train,y_train)
```

Out[28]: 0.8302178271427866

In [29]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [30]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[30]: Ridge(alpha=10)

In [31]:
```
rr.score(x_test,y_test)
```

Out[31]: 0.8221409241828994

In [32]:
```
rr.score(x_train,y_train)
```

Out[32]: 0.8301311859727917

In [33]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[33]:
```
Lasso(alpha=10)
```

In [34]:
```
la.score(x_test,y_test)
```
Out[34]:
```
0.6482056484857444
```

In [35]:
```
la.score(x_train,y_train)
```
Out[35]:
```
0.6463260941245638
```

In [36]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[36]:
```
ElasticNet()
```

In [37]:
```
en.coef_
```
Out[37]:
```
array([-0.        ,  0.        , -0.        , -0.        ,  0.04717921,
       -0.10627211, -0.02193956,  0.00250391,  0.04354719, -0.86986901,
       -0.01448539,  0.        ])
```

In [38]:
```
en.intercept_
```
Out[38]:
```
28079026.30199501
```

In [39]:
```
prediction=en.predict(x_test)
```

In [40]:
```
en.score(x_test,y_test)
```
Out[40]:
```
0.7058559284487442
```

In [41]:
```
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)))
```
```
3.3493497255052933
18.82514763140473
4.338795642964154
```

In [42]:
```
from sklearn.linear_model import LogisticRegression
```

In [43]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [44]:
```
feature_matrix.shape
```
Out[44]:
```
(16932, 10)
```

In [45]:
```
target_vector.shape
```
Out[45]:
```
(16932,)
```

In [46]:
```
from sklearn.preprocessing import StandardScaler
```

In [47]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [48]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```
Out[48]:
```
LogisticRegression(max_iter=10000)
```

In [49]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [50]:
```
prediction=logr.predict(observation)
print(prediction)
```
```
[28079008]
```

In [51]:
```
logr.classes_
```
Out[51]:
```
array([28079008, 28079024], dtype=int64)
```

In [52]:
```
logr.score(fs,target_vector)
```
Out[52]:
```
0.9923812898653437
```

In [53]:
```
logr.predict_proba(observation)[0][0]
```
Out[53]:
```
1.0
```

In [54]:
```
logr.predict_proba(observation)
```
Out[54]:
```
array([[1.0000000e+00, 1.6336121e-46]])
```

In [55]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [56]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[56]:
```
RandomForestClassifier()
```

In [57]:
```
parameters={'max_depth':[1,2,3,4,5],
'min_samples_leaf':[5,10,15,20,25],
'n_estimators':[10,20,30,40,50]
}
```

In [58]:
```
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[58]:
```
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 [59]:
```
grid_search.best_score_
```
Out[59]:
```
0.9937563280458994
```

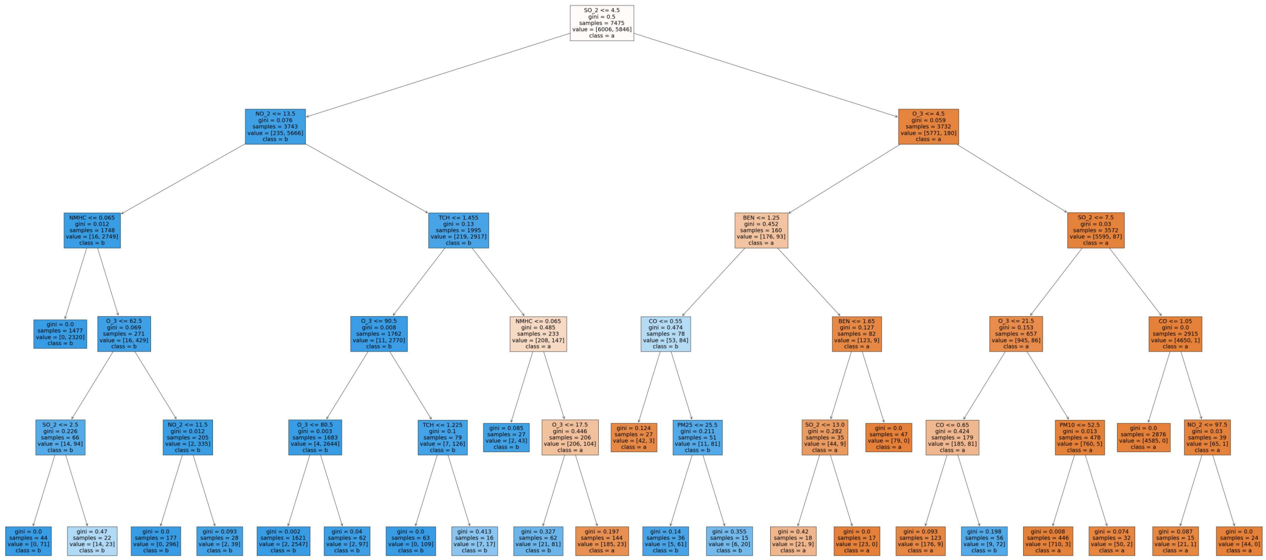
In [60]:
```
rfc_best=grid_search.best_estimator_
```

In [61]:

```
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[61]:
```
[Text(2120.4, 1993.2, 'SO_2 <= 4.5\ngini = 0.5\nsamples = 7475\nvalue = [6006, 5846]\nclass = a'),
 Text(976.5, 1630.8000000000002, 'NO_2 <= 13.5\ngini = 0.076\nsamples = 3743\nvalue = [235, 5666]\nclass = b'),
 Text(334.79999999999995, 1268.4, 'NMHC <= 0.065\ngini = 0.012\nsamples = 1748\nvalue = [16, 2749]\nclass = b'),
 Text(223.2, 906.0, 'gini = 0.0\nsamples = 1477\nvalue = [0, 2320]\nclass = b'),
 Text(446.4, 906.0, 'O_3 <= 62.5\ngini = 0.069\nsamples = 271\nvalue = [16, 429]\nclass = b'),
 Text(223.2, 543.5999999999999, 'SO_2 <= 2.5\ngini = 0.226\nsamples = 66\nvalue = [14, 94]\nclass = b'),
 Text(111.6, 181.19999999999982, 'gini = 0.0\nsamples = 44\nvalue = [0, 71]\nclass = b'),
 Text(334.79999999999995, 181.19999999999982, 'gini = 0.47\nsamples = 22\nvalue = [14, 23]\nclass = b'),
 Text(669.5999999999999, 543.5999999999999, 'NO_2 <= 11.5\ngini = 0.012\nsamples = 205\nvalue = [2, 335]\nclass = b'),
 Text(558.0, 181.19999999999982, 'gini = 0.0\nsamples = 177\nvalue = [0, 296]\nclass = b'),
 Text(781.1999999999999, 181.19999999999982, 'gini = 0.093\nsamples = 28\nvalue = [2, 39]\nclass = b'),
 Text(1618.1999999999998, 1268.4, 'TCH <= 1.455\ngini = 0.13\nsamples = 1995\nvalue = [219, 2917]\nclass = b'),
 Text(1339.1999999999998, 906.0, 'O_3 <= 90.5\ngini = 0.008\nsamples = 1762\nvalue = [11, 2770]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'O_3 <= 80.5\ngini = 0.003\nsamples = 1683\nvalue = [4, 2644]\nclass = b'),
 Text(1004.4, 181.19999999999982, 'gini = 0.002\nsamples = 1621\nvalue = [2, 2547]\nclass = b'),
 Text(1227.6, 181.19999999999982, 'gini = 0.04\nsamples = 62\nvalue = [2, 97]\nclass = b'),
 Text(1562.3999999999999, 543.5999999999999, 'TCH <= 1.225\ngini = 0.1\nsamples = 79\nvalue = [7, 126]\nclass = b'),
 Text(1450.8, 181.19999999999982, 'gini = 0.0\nsamples = 63\nvalue = [0, 109]\nclass = b'),
 Text(1674.0, 181.19999999999982, 'gini = 0.413\nsamples = 16\nvalue = [7, 17]\nclass = b'),
 Text(1897.1999999999998, 906.0, 'NMHC <= 0.065\ngini = 0.485\nsamples = 233\nvalue = [208, 147]\nclass = a'),
 Text(1785.6, 543.5999999999999, 'gini = 0.085\nsamples = 27\nvalue = [2, 43]\nclass = b'),
 Text(2008.8, 543.5999999999999, 'O_3 <= 17.5\ngini = 0.446\nsamples = 206\nvalue = [206, 104]\nclass = a'),
 Text(1897.1999999999998, 181.19999999999982, 'gini = 0.327\nsamples = 62\nvalue = [21, 81]\nclass = b'),
 Text(2120.4, 181.19999999999982, 'gini = 0.197\nsamples = 144\nvalue = [185, 23]\nclass = a'),
 Text(3264.2999999999997, 1630.8000000000002, 'O_3 <= 4.5\ngini = 0.059\nsamples = 3732\nvalue = [5771, 180]\nclass = a'),
 Text(2678.3999999999996, 1268.4, 'BEN <= 1.25\ngini = 0.452\nsamples = 160\nvalue = [176, 93]\nclass = a'),
 Text(2343.6, 906.0, 'CO <= 0.55\ngini = 0.474\nsamples = 78\nvalue = [53, 84]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'gini = 0.124\nsamples = 27\nvalue = [42, 3]\nclass = a'),
 Text(2455.2, 543.5999999999999, 'PM25 <= 25.5\ngini = 0.211\nsamples = 51\nvalue = [11, 81]\nclass = b'),
 Text(2343.6, 181.19999999999982, 'gini = 0.14\nsamples = 36\nvalue = [5, 61]\nclass = b'),
 Text(2566.7999999999997, 181.19999999999982, 'gini = 0.355\nsamples = 15\nvalue = [6, 20]\nclass = b'),
 Text(3013.2, 906.0, 'BEN <= 1.65\ngini = 0.127\nsamples = 82\nvalue = [123, 9]\nclass = a'),
 Text(2901.6, 543.5999999999999, 'SO_2 <= 13.0\ngini = 0.282\nsamples = 35\nvalue = [44, 9]\nclass = a'),
 Text(2790.0, 181.19999999999982, 'gini = 0.42\nsamples = 18\nvalue = [21, 9]\nclass = a'),
 Text(3013.2, 181.19999999999982, 'gini = 0.0\nsamples = 17\nvalue = [23, 0]\nclass = a'),
 Text(3124.7999999999997, 543.5999999999999, 'gini = 0.0\nsamples = 47\nvalue = [79, 0]\nclass = a'),
 Text(3850.2, 1268.4, 'SO_2 <= 7.5\ngini = 0.03\nsamples = 3572\nvalue = [5595, 87]\nclass = a'),
 Text(3571.2, 906.0, 'O_3 <= 21.5\ngini = 0.153\nsamples = 657\nvalue = [945, 86]\nclass = a'),
 Text(3348.0, 543.5999999999999, 'CO <= 0.65\ngini = 0.424\nsamples = 179\nvalue = [185, 81]\nclass = a'),
 Text(3236.3999999999996, 181.19999999999982, 'gini = 0.093\nsamples = 123\nvalue = [176, 9]\nclass = a'),
 Text(3459.6, 181.19999999999982, 'gini = 0.198\nsamples = 56\nvalue = [9, 72]\nclass = b'),
 Text(3794.3999999999996, 543.5999999999999, 'PM10 <= 52.5\ngini = 0.013\nsamples = 478\nvalue = [760, 5]\nclass = a'),
 Text(3682.7999999999997, 181.19999999999982, 'gini = 0.008\nsamples = 446\nvalue = [710, 3]\nclass = a'),
 Text(3906.0, 181.19999999999982, 'gini = 0.074\nsamples = 32\nvalue = [50, 2]\nclass = a'),
 Text(4129.2, 906.0, 'CO <= 1.05\ngini = 0.0\nsamples = 2915\nvalue = [4650, 1]\nclass = a'),
 Text(4017.6, 543.5999999999999, 'gini = 0.0\nsamples = 2876\nvalue = [4585, 0]\nclass = a'),
 Text(4240.8, 543.5999999999999, 'NO_2 <= 97.5\ngini = 0.03\nsamples = 39\nvalue = [65, 1]\nclass = a'),
```

```
 Text(4129.2, 181.19999999999982, 'gini = 0.087\nsamples = 15\nvalue = [21, 1]\nclass = a'),
 Text(4352.4, 181.19999999999982, 'gini = 0.0\nsamples = 24\nvalue = [44, 0]\nclass = a')]
```



Conclusion

Linear Regression=0.8302178271427866

Ridge Regression=0.8301311859727917

Lasso Regression=0.6463260941245638

ElasticNet Regression=0.7058559284487442

Logistic Regression=0.9923812898653437

Random Forest=0.9937563280458994

Random Forest is suitable for this dataset

In [ ]:
```
1
```