Importing Libraries

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

Importing Datasets

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs(Dataset)\madrid_2013.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	28079008
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28079011
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28079056
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28079057
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28079058
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28079059
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28079060

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.fillna(1)
```

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'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [6]:
```
data=df[['CO', 'station']]
data
```

Out[6]:

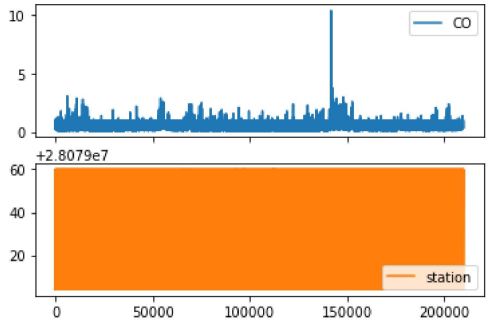
	CO	station
0	0.6	28079004
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

Line chart

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

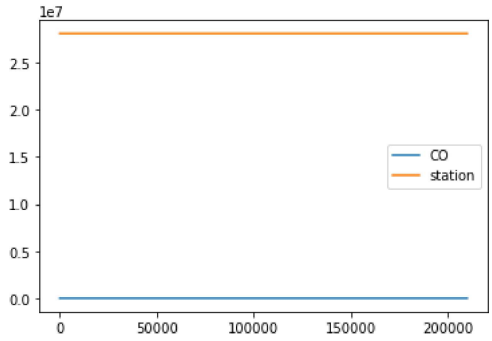
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

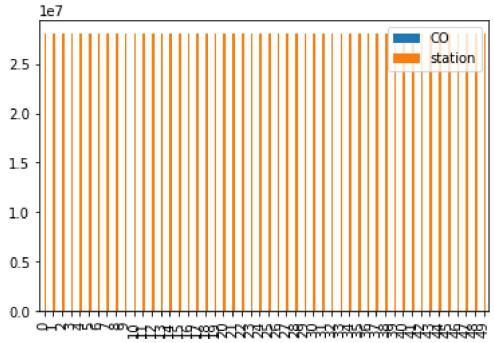


Bar chart

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

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

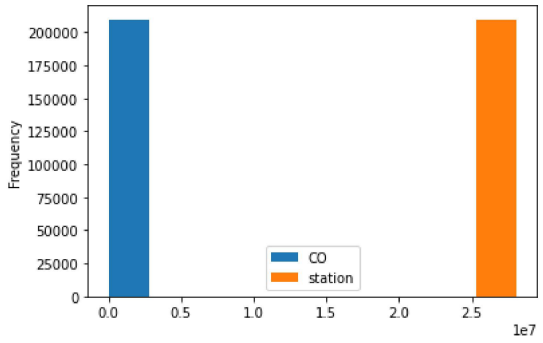
Out[10]: <AxesSubplot:>



Histogram

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

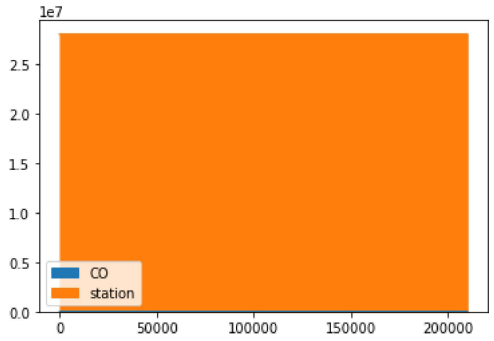
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

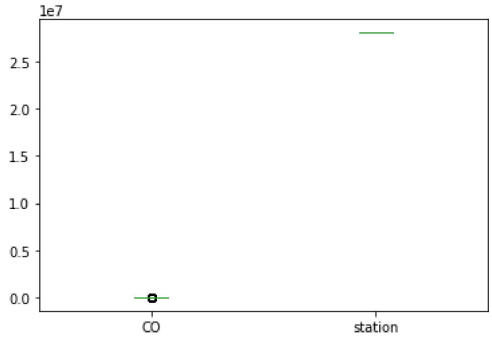
Out[12]: <AxesSubplot:>



Box chart

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

Out[13]: <AxesSubplot:>



Pie chart

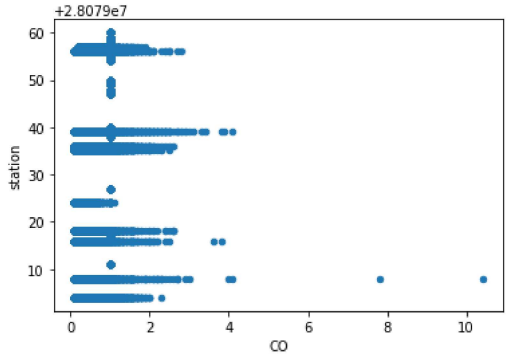
In [14]:
```
b.plot.pie(y='station' )
```

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

Scatter chart

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

Out[15]: <AxesSubplot:xlabel='CO', ylabel='station'>



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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [17]:
```
df.describe()
```

Out[17]:

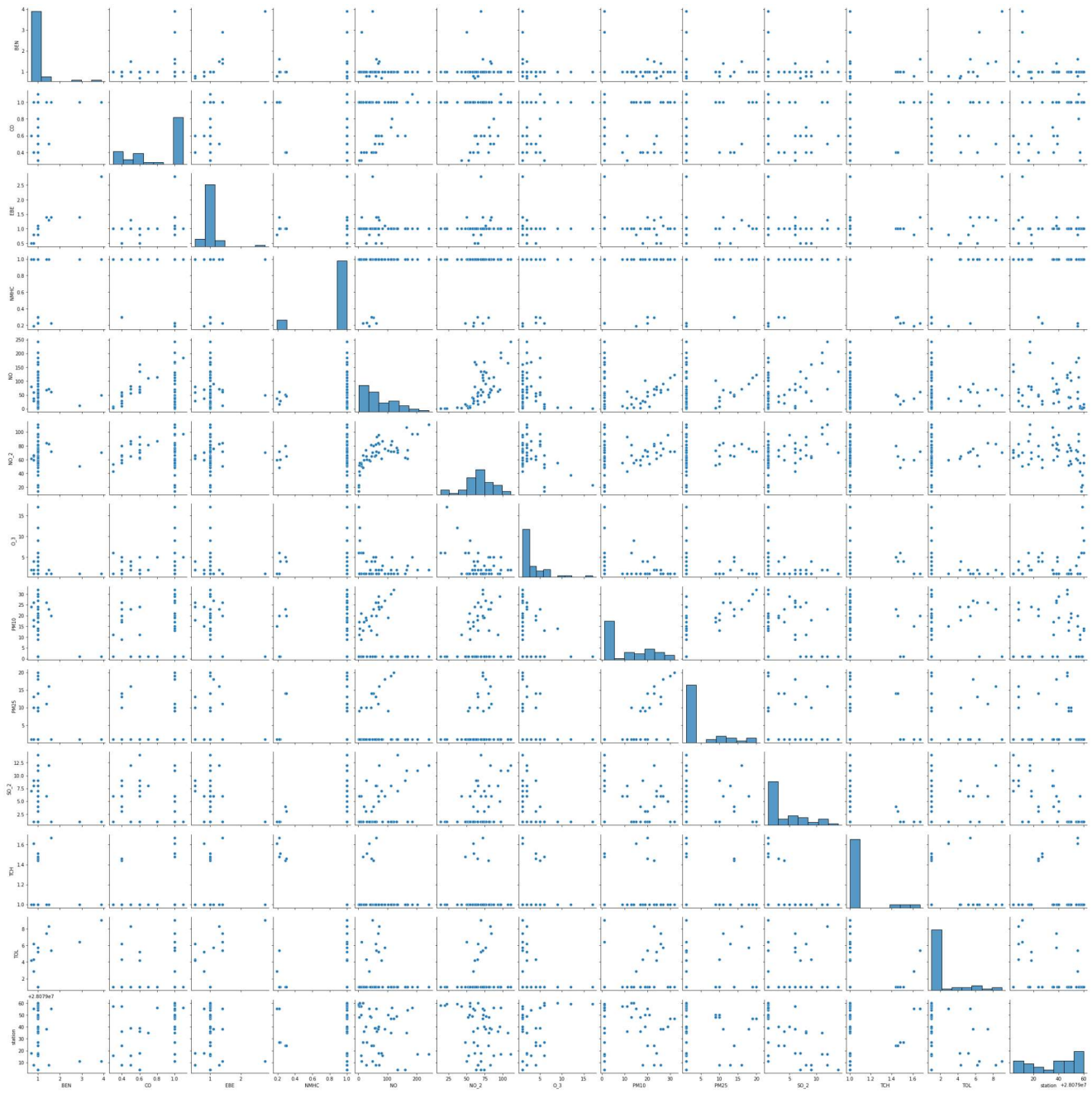
	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.586402	29.461235	9.636635	3.213098	2.417243
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.866588	35.362880	13.492716	5.044685	3.093256
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.000000	1.000000	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.000000	8.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.000000	54.000000	14.000000	1.000000	3.000000
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.000000	226.000000	232.000000	63.000000	89.000000

In [18]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [19]:
```
sns.pairplot(df1[0:50])
```

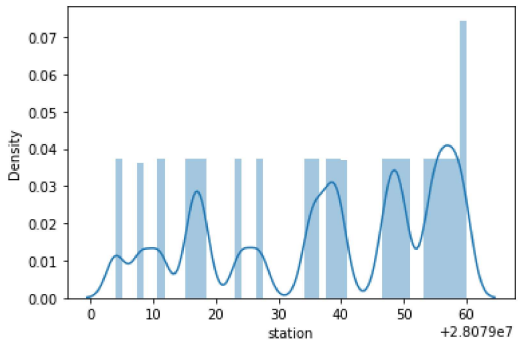
Out[19]: <seaborn.axisgrid.PairGrid at 0x25e5f3517c0>



In [20]:
```
sns.distplot(df1['station'])
```

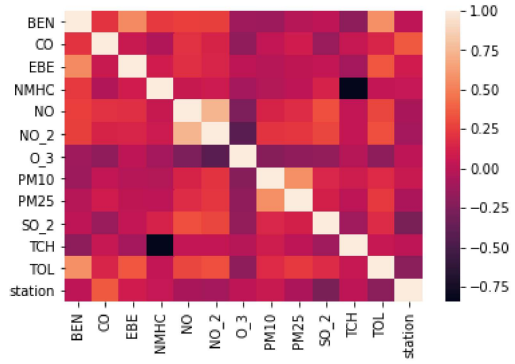
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function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

Out[20]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [21]:
```
sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [23]:
```
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)
```

Linear Regression

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]: LinearRegression()

In [25]:
```
lr.intercept_
```

Out[25]: 28078976.553258862

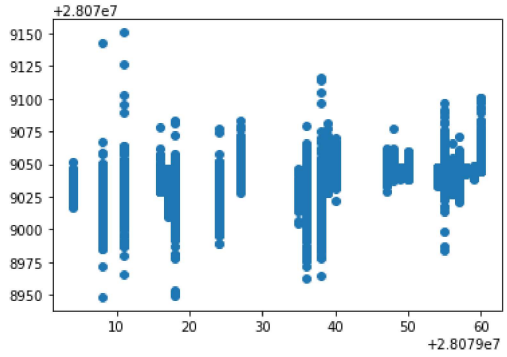
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	1.657878
CO	18.952351
EBE	10.543432
NMHC	17.978444
NO	-0.009421
NO_2	-0.039553
O_3	0.009344
PM10	0.274927
PM25	-0.364441
SO_2	-0.922414
TCH	25.194917
TOL	-3.431898

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[27]: <matplotlib.collections.PathCollection at 0x25e668e2310>



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```

Out[28]: 0.30689539488758666

In [29]:
```
lr.score(x_train,y_train)
```

Out[29]: 0.30733570662165033

Ridge and Lasso

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```

Out[32]: 0.3068862714596511

In [33]:
```
rr.score(x_train,y_train)
```

Out[33]: 0.3073328908760592

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]:
```
Lasso(alpha=10)
```

In [35]:
```
la.score(x_test,y_test)
```

Out[35]:
```
0.044718571166753374
```

Accuracy(Lasso)

In [36]:
```
la.score(x_train,y_train)
```

Out[36]:
```
0.04404802899149829
```

Accuracy(Elastic Net)

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[37]:
```
ElasticNet()
```

In [38]:
```
en.coef_
```

Out[38]:
```
array([ 0.32169598,  2.65239587,  0.445145  ,  0.        ,  0.03539034,
       -0.05587248, -0.0187539 ,  0.23181921, -0.34642439, -1.3416689 ,
       -0.        , -1.5387983 ])
```

In [39]:
```
en.intercept_
```

Out[39]:
```
28079041.16717371
```

In [40]:
```
prediction=en.predict(x_test)
```

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

Out[41]:
```
0.1658773605891095
```

Evaluation Metrics

In [42]:
```
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)))
```

```
13.81031956518184
258.6514122925606
16.082643199815156
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [45]:
```
feature_matrix.shape
```

Out[45]:
```
(209880, 10)
```

In [46]:
```
target_vector.shape
```

Out[46]:
```
(209880,)
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [52]:
```
logr.classes_
```

Out[52]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.6612921669525443
```

In [54]:
```
logr.predict_proba(observation)[0][0]
```

Out[54]:
```
9.49253547859177e-217
```

In [55]:
```
logr.predict_proba(observation)
```

Out[55]:
```
array([[9.49253548e-217, 6.03969072e-001, 1.69773000e-169,
        1.44179094e-134, 1.71060740e-074, 3.96021369e-001,
        9.55808997e-006, 5.22717178e-089, 5.48319507e-081,
        1.32436170e-079, 1.07294134e-076, 3.50636612e-129,
        1.69529056e-079, 3.82520459e-158, 4.22872970e-161,
        3.57928159e-187, 2.10845766e-164, 8.33937392e-188,
        1.12752042e-082, 7.42692411e-129, 7.66872499e-080,
        6.30044443e-191, 4.32093567e-191, 3.26054498e-071]])
```

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]:
```
RandomForestClassifier()
```

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [59]:
```
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[59]:
```
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 [60]:
```
grid_search.best_score_
```

Out[60]:
```
0.7485569985569985
```

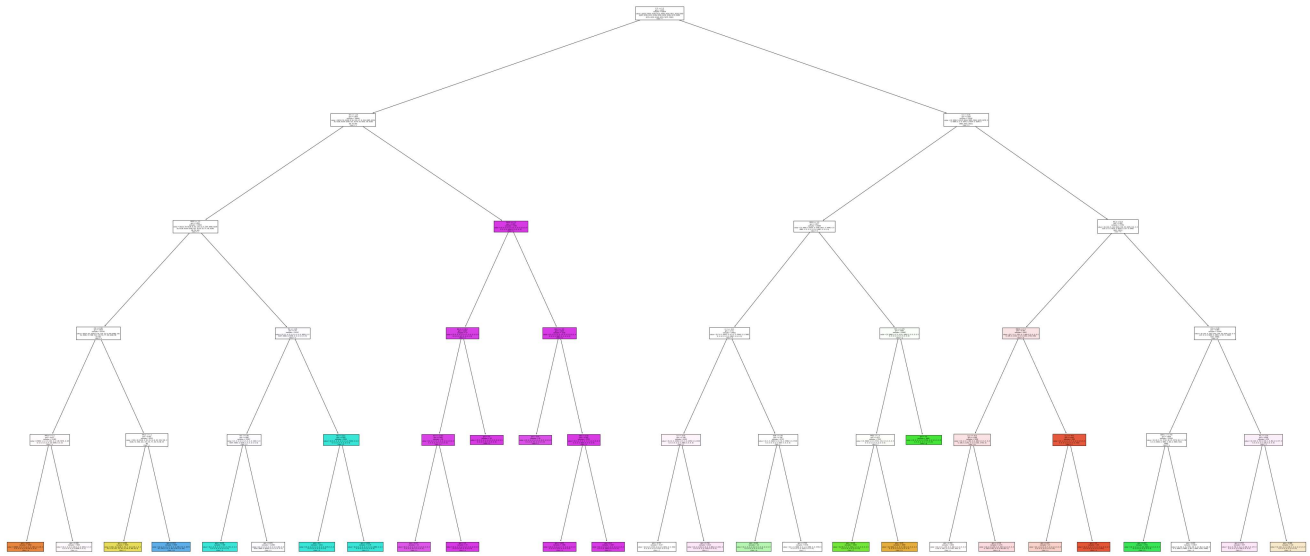
In [61]:
```
rfc_best=grid_search.best_estimator_
```

In [62]:
```
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','e','f','g','h','i','j','k','l','m','n','o
```

Out[62]: [Text(2242.3333333333335, 1993.2, 'O_3 <= 1.5\ngini = 0.958\nsamples = 92874\nvalue = [6210, 5900, 6108, 6123, 6094, 6019, 5971, 6234, 6282\n6085, 6236, 6110, 6106, 6259, 6258, 6196, 6179, 6089\n6031, 6242, 6166, 6053, 5975, 5990]\nclass = i'),
 Text(1198.6666666666667, 1630.8000000000002, 'TCH <= 1.25\ngini = 0.904\nsamples = 39694\nvalue = [6210, 54, 6108, 8, 50, 124, 27, 9, 204, 6085, 6236\n30, 6106, 6259, 6258, 321, 6179, 55, 6031, 58, 6166\n88, 54, 69]\nclass = n'),
 Text(661.3333333333334, 1268.4, 'PM25 <= 1.5\ngini = 0.894\nsamples = 35902\nvalue = [6210, 54, 6108, 8, 50, 124, 15, 6, 204, 6085, 6236\n30, 6106, 6259, 6258, 321, 6179, 55, 77, 58, 6166\n88, 54, 69]\nclass = n'),
 Text(330.6666666666667, 906.0, 'CO <= 0.95\ngini = 0.825\nsamples = 20745\nvalue = [6210, 43, 6108, 8, 50, 124, 10, 6, 204, 6085, 181\n30, 6106, 72, 598, 321, 143, 55, 77, 58, 6166, 88\n54, 69]\nclass = a'),
 Text(165.33333333333334, 543.5999999999999, 'PM10 <= 1.5\ngini = 0.677\nsamples = 11492\nvalue = [5909, 1, 0, 4, 0, 100, 0, 0, 139, 5876, 0, 28\n0, 0, 0, 0, 0, 0, 0, 20, 6096, 0, 0, 0]\nclass = u'),
 Text(82.66666666666667, 181.19999999999982, 'gini = 0.074\nsamples = 3892\nvalue = [5909, 0, 0, 4, 0, 0, 0, 0, 139, 4, 0, 28, 0\n0, 0, 0, 0, 0, 0, 20, 40, 0, 0, 0]\nclass = a'),
 Text(248.0, 181.19999999999982, 'gini = 0.508\nsamples = 7600\nvalue = [0, 1, 0, 0, 0, 100, 0, 0, 0, 5872, 0, 0, 0\n0, 0, 0, 0, 0, 0, 6056, 0, 0, 0]\nclass = u'),
 Text(496.0, 543.5999999999999, 'PM10 <= 1.5\ngini = 0.651\nsamples = 9253\nvalue = [301, 42, 6108, 4, 50, 24, 10, 6, 65, 209, 181, 2\n6106, 72, 598, 321, 143, 55, 77, 38, 70, 88, 54\n69]\nclass = c'),
 Text(413.33333333333337, 181.19999999999982, 'gini = 0.298\nsamples = 4659\nvalue = [301, 42, 6108, 4, 50, 7, 10, 6, 65, 1, 9, 2\n33, 5, 19, 321, 70, 55, 14, 38, 5, 88, 54, 4]\nclass = c'),
 Text(578.6666666666667, 181.19999999999982, 'gini = 0.315\nsamples = 4594\nvalue = [0, 0, 0, 0, 0, 17, 0, 0, 0, 208, 172, 0, 6073\n67, 579, 0, 73, 0, 63, 0, 65, 0, 0, 65]\nclass = m'),
 Text(992.0, 906.0, 'TOL <= 1.05\ngini = 0.75\nsamples = 15157\nvalue = [0, 11, 0, 0, 0, 0, 5, 0, 0, 0, 6055, 0, 0\n6187, 5660, 0, 6036, 0, 0, 0, 0, 0, 0, 0]\nclass = n'),
 Text(826.6666666666667, 543.5999999999999, 'TOL <= 0.95\ngini = 0.698\nsamples = 11957\nvalue = [0, 3, 0, 0, 0, 0, 5, 0, 0, 0, 1002, 0, 0\n6187, 5660, 0, 6036, 0, 0, 0, 0, 0, 0, 0]\nclass = n'),
 Text(744.0, 181.19999999999982, 'gini = 0.016\nsamples = 469\nvalue = [0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 741, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(909.3333333333334, 181.19999999999982, 'gini = 0.676\nsamples = 11488\nvalue = [0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 261, 0, 0\n6187, 5660, 0, 6036, 0, 0, 0, 0, 0, 0, 0]\nclass = n'),
 Text(1157.3333333333335, 543.5999999999999, 'NO <= 23.5\ngini = 0.003\nsamples = 3200\nvalue = [0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 5053, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(1074.6666666666667, 181.19999999999982, 'gini = 0.0\nsamples = 2021\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3173, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(1240.0, 181.19999999999982, 'gini = 0.008\nsamples = 1179\nvalue = [0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 1880, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(1736.0, 1268.4, 'PM10 <= 2.5\ngini = 0.005\nsamples = 3792\nvalue = [0, 0, 0, 0, 0, 0, 12, 3, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5954, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1570.6666666666667, 906.0, 'NO_2 <= 20.5\ngini = 0.064\nsamples = 57\nvalue = [0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 87, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1488.0, 543.5999999999999, 'TCH <= 1.415\ngini = 0.095\nsamples = 39\nvalue = [0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 57, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1405.3333333333335, 181.19999999999982, 'gini = 0.227\nsamples = 15\nvalue = [0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 20, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1570.6666666666667, 181.19999999999982, 'gini = 0.0\nsamples = 24\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 37, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1653.3333333333335, 543.5999999999999, 'gini = 0.0\nsamples = 18\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 30, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1901.3333333333335, 906.0, 'NO_2 <= 3.5\ngini = 0.004\nsamples = 3735\nvalue = [0, 0, 0, 0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5867, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1818.6666666666667, 543.5999999999999, 'gini = 0.19\nsamples = 30\nvalue = [0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 42, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1984.0, 543.5999999999999, 'EBE <= 0.95\ngini = 0.002\nsamples = 3705\nvalue = [0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5825, 0, 0, 0, 0, 0]\nclass = s'),
 Text(1901.3333333333335, 181.19999999999982, 'gini = 0.005\nsamples = 1768\nvalue = [0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2772, 0, 0, 0, 0, 0]\nclass = s'),
 Text(2066.666666666667, 181.19999999999982, 'gini = 0.0\nsamples = 1937\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3053, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3286.0, 1630.8000000000002, 'CO <= 0.95\ngini = 0.929\nsamples = 53180\nvalue = [0, 5846, 0, 6115, 6044, 5895, 5944, 6225, 6078, 0\n0, 6080, 0, 0, 0, 5875, 0, 6034, 0, 6184, 0\n5965, 5921, 5921]\nclass = h'),
 Text(2769.3333333333335, 1268.4, 'PM10 <= 1.5\ngini = 0.857\nsamples = 25889\nvalue = [0, 5682, 0, 6005, 0, 5746, 5911, 0, 5946, 0, 0\n5862, 0, 0, 0, 0, 0, 0, 0, 5867, 0, 0, 0, 0]\nclass = d'),
 Text(2480.0, 906.0, 'O_3 <= 38.5\ngini = 0.751\nsamples = 14902\nvalue = [0, 14, 0, 6005, 0, 25, 21, 0, 5946, 0, 0, 5862\n0, 0, 0, 0, 0, 0, 0, 5867, 0, 0, 0, 0]\nclass = d'),
 Text(2314.666666666667, 543.5999999999999, 'CO <= 0.45\ngini = 0.749\nsamples = 6150\nvalue = [0, 12, 0, 2336, 0, 15, 3, 0, 2448, 0, 0, 2133\n0, 0, 0, 0, 0, 0, 0, 2860, 0, 0, 0, 0]\nclass = t'),
 Text(2232.0, 181.19999999999982, 'gini = 0.751\nsamples = 4137\nvalue = [0, 10, 0, 1723, 0, 6, 3, 0, 1642, 0, 0, 1502\n0, 0, 0, 0, 0, 0, 1757, 0, 0, 0, 0]\nclass = t'),
 Text(2397.3333333333335, 181.19999999999982, 'gini = 0.736\nsamples = 2013\nvalue = [0, 2, 0, 613, 0, 9, 0, 0, 806, 0, 0, 631, 0\n0, 0, 0, 0, 0, 0, 1103, 0, 0, 0, 0]\nclass = t'),
 Text(2645.3333333333335, 543.5999999999999, 'BEN <= 0.95\ngini = 0.749\nsamples = 8752\nvalue = [0, 2, 0, 3669, 0, 10, 18, 0, 3498, 0, 0, 3729\n0, 0, 0, 0, 0, 0, 0, 3007, 0, 0, 0, 0]\nclass = l'),
 Text(2562.666666666667, 181.19999999999982, 'gini = 0.503\nsamples = 16\nvalue = [0, 1, 0, 0, 0, 10, 17, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(2728.0, 181.19999999999982, 'gini = 0.748\nsamples = 8736\nvalue = [0, 1, 0, 3669, 0, 0, 1, 0, 3498, 0, 0, 3729, 0\n0, 0, 0, 0, 0, 3007, 0, 0, 0, 0]\nclass = l'),
 Text(3058.666666666667, 906.0, 'TCH <= 1.075\ngini = 0.667\nsamples = 10987\nvalue = [0, 5668, 0, 0, 0, 5721, 5890, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(2976.0, 543.5999999999999, 'PM25 <= 1.5\ngini = 0.513\nsamples = 7322\nvalue = [0, 5668, 0, 0, 0, 5721, 151, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(2893.3333333333335, 181.19999999999982, 'gini = 0.018\nsamples = 3663\nvalue = [0, 50, 0, 0, 0, 5721, 3, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(3058.666666666667, 181.19999999999982, 'gini = 0.05\nsamples = 3659\nvalue = [0, 5618, 0, 0, 0, 0, 148, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(3141.3333333333335, 543.5999999999999, 'gini = 0.0\nsamples = 3665\nvalue = [0, 0, 0, 0, 0, 0, 5739, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(3802.666666666667, 1268.4, 'NO_2 <= 6.5\ngini = 0.864\nsamples = 27291\nvalue = [0, 164, 0, 110, 6044, 149, 33, 6225, 132, 0, 0\n218, 0, 0, 0, 5875, 0, 6034, 0, 317, 0, 5965\n5921, 5921]\nclass = h'),

```
 Text(3472.0, 906.0, 'PM10 <= 1.5\ngini = 0.797\nsamples = 3507\nvalue = [0, 1, 0, 1, 302, 0, 7, 649, 0, 0, 0, 3, 0, 0\n0, 185, 0, 1142, 0, 0, 0, 1001, 1730, 435]\nclass = w'),
 Text(3306.666666666667, 543.5999999999999, 'O_3 <= 78.5\ngini = 0.768\nsamples = 3209\nvalue = [0, 0, 0, 1, 302, 0, 0, 649, 0, 0, 0, 3, 0, 0\n0, 185, 0, 1142, 0, 0, 0, 1001, 1730, 3]\nclass = w'),
 Text(3224.0, 181.19999999999982, 'gini = 0.764\nsamples = 1936\nvalue = [0, 0, 0, 1, 219, 0, 0, 292, 0, 0, 0, 3, 0, 0\n0, 85, 0, 591, 0, 0, 0, 911, 903, 2]\nclass = v'),
 Text(3389.3333333333335, 181.19999999999982, 'gini = 0.718\nsamples = 1273\nvalue = [0, 0, 0, 0, 83, 0, 0, 357, 0, 0, 0, 0, 0, 0\n0, 100, 0, 551, 0, 0, 0, 90, 827, 1]\nclass = w'),
 Text(3637.3333333333335, 543.5999999999999, 'O_3 <= 44.5\ngini = 0.036\nsamples = 298\nvalue = [0, 1, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 432]\nclass = x'),
 Text(3554.666666666667, 181.19999999999982, 'gini = 0.537\nsamples = 15\nvalue = [0, 1, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 10]\nclass = x'),
 Text(3720.0, 181.19999999999982, 'gini = 0.0\nsamples = 283\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 422]\nclass = x'),
 Text(4133.333333333334, 906.0, 'CO <= 1.05\ngini = 0.864\nsamples = 23784\nvalue = [0, 163, 0, 109, 5742, 149, 26, 5576, 132, 0, 0\n215, 0, 0, 0, 5690, 0, 4892, 0, 317, 0, 4964\n4191, 5486]\nclass = e'),
 Text(3968.0, 543.5999999999999, 'NMHC <= 0.745\ngini = 0.858\nsamples = 23308\nvalue = [0, 40, 0, 31, 5742, 49, 25, 5576, 46, 0, 0, 38\n0, 0, 0, 5690, 0, 4892, 0, 89, 0, 4964, 4191\n5486]\nclass = e'),
 Text(3885.3333333333335, 181.19999999999982, 'gini = 0.007\nsamples = 3491\nvalue = [0, 0, 0, 0, 0, 0, 20, 5564, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(4050.666666666667, 181.19999999999982, 'gini = 0.835\nsamples = 19817\nvalue = [0, 40, 0, 31, 5742, 49, 5, 12, 46, 0, 0, 38, 0\n0, 0, 5690, 0, 4892, 0, 89, 0, 4964, 4191, 5486]\nclass = e'),
 Text(4298.666666666667, 543.5999999999999, 'TOL <= 1.95\ngini = 0.806\nsamples = 476\nvalue = [0, 123, 0, 78, 0, 100, 1, 0, 86, 0, 0, 177, 0\n0, 0, 0, 0, 0, 0, 228, 0, 0, 0, 0]\nclass = t'),
 Text(4216.0, 181.19999999999982, 'gini = 0.715\nsamples = 357\nvalue = [0, 2, 0, 78, 0, 11, 1, 0, 86, 0, 0, 177, 0\n0, 0, 0, 0, 0, 0, 228, 0, 0, 0, 0]\nclass = t'),
 Text(4381.333333333334, 181.19999999999982, 'gini = 0.488\nsamples = 119\nvalue = [0, 121, 0, 0, 0, 89, 0, 0, 0, 0, 0, 0, 0, 0
\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b')]
```

Conclusion

Accuracy

Linear Regression:0.30733570662165033

Ridge Regression:0.3073328908760592

Lasso Regression:0.04404802899149829

ElasticNet Regression:0.1658773605891095

Logistic Regression:0.6612921669525443

Random Forest:0.7485569985569985

Random Forest is suitable for this dataset