Importing Libraries

In [491]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [521]:
```
df=pd.read_csv(r"C:\Users\user\Desktop\csvs_per_year\csvs_per_year\madrid_2012.
df
```

Out[521]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28
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	28
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28

210720 rows × 14 columns

Data Cleaning and Data Preprocessing

In [522]:
```
df=df.dropna()
```

In [523]:
```
df.columns
```

Out[523]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [524]:
```
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 [525]:
```
data=df[['CO' ,'station']]
data
```

Out[525]:

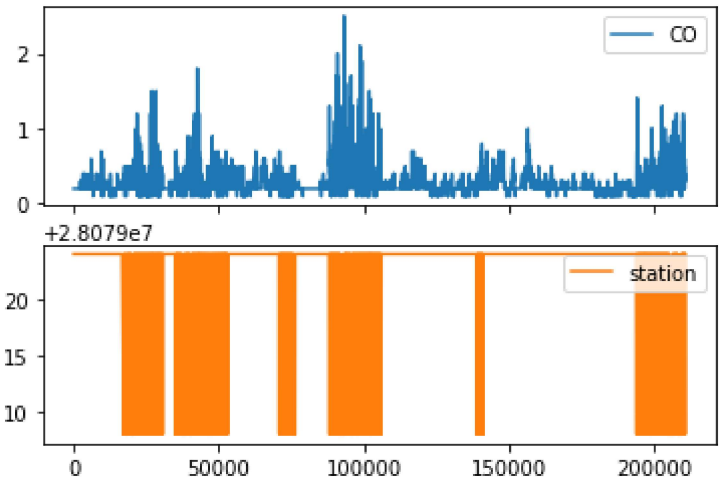
	CO	station
6	0.2	28079024
30	0.2	28079024
54	0.2	28079024
78	0.2	28079024
102	0.2	28079024
...	...	...
210654	0.3	28079024
210673	0.4	28079008
210678	0.3	28079024
210697	0.4	28079008
210702	0.3	28079024

10916 rows × 2 columns

Line chart

In [526]:
```
data.plot.line(subplots=True)
```

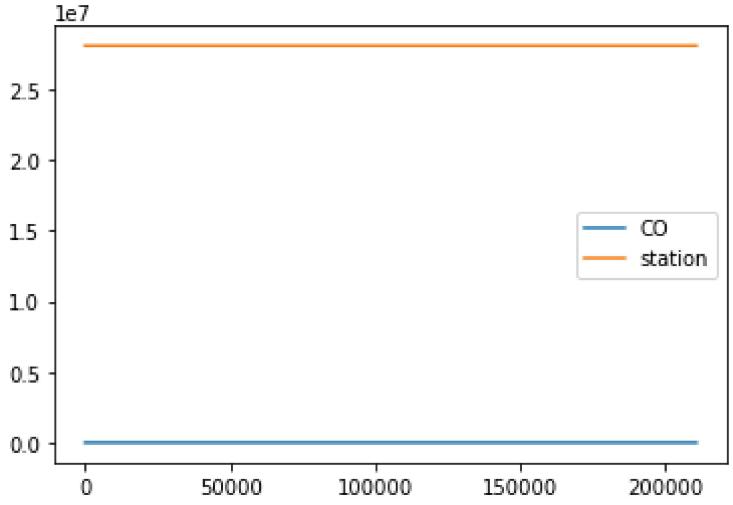
Out[526]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [527]:
```
data.plot.line()
```

Out[527]: <AxesSubplot:>

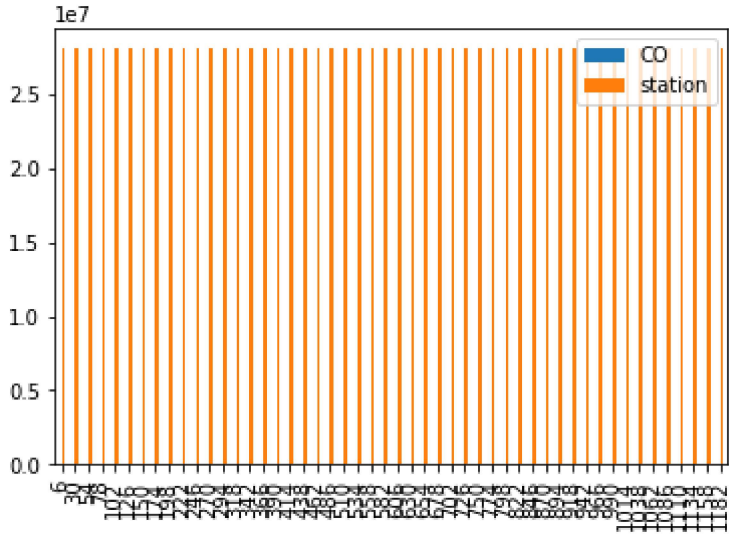


Bar chart

In [528]:
```
b=data[0:50]
```

In [529]:
```
b.plot.bar()
```

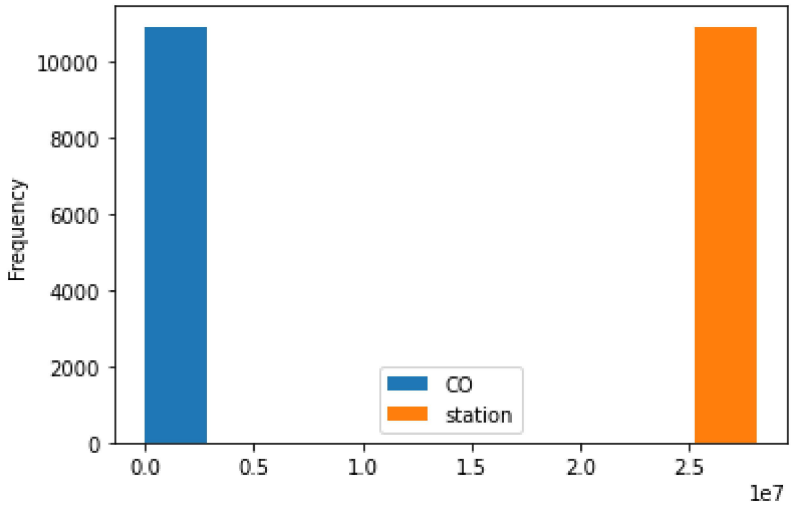
Out[529]: <AxesSubplot:>



Histogram

In [530]:
```
data.plot.hist()
```

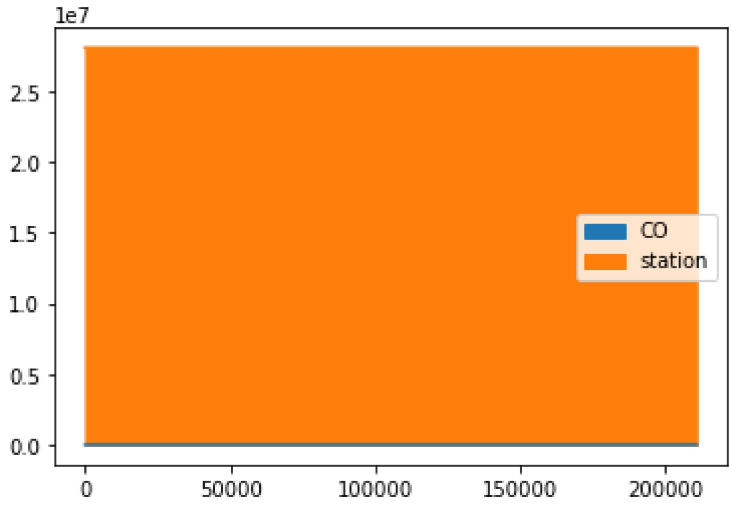
Out[530]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [531]:
```
data.plot.area()
```

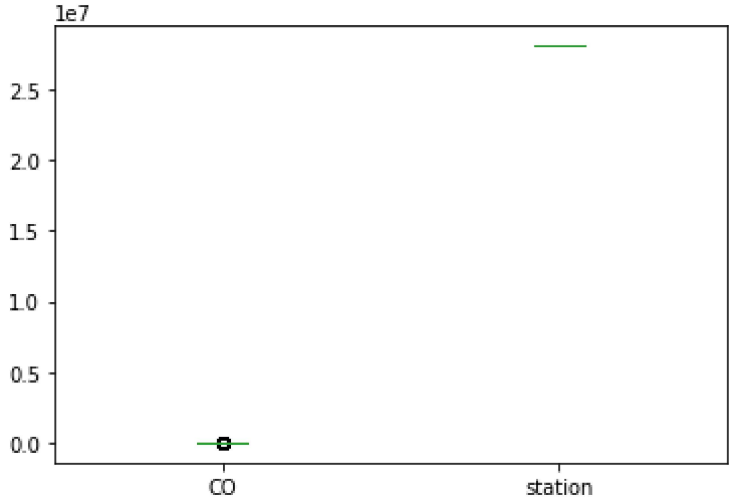
Out[531]: <AxesSubplot:>



Box chart

In [532]:
```
data.plot.box()
```

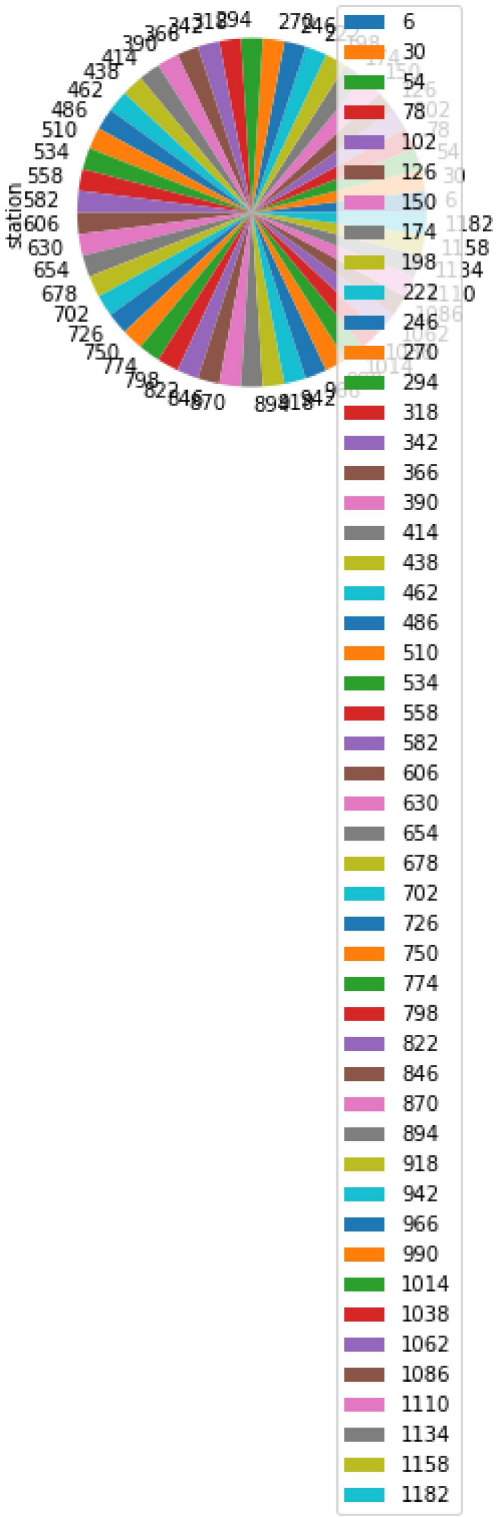
Out[532]: <AxesSubplot:>



Pie chart

In [533]:
```
b.plot.pie(y='station' )
```

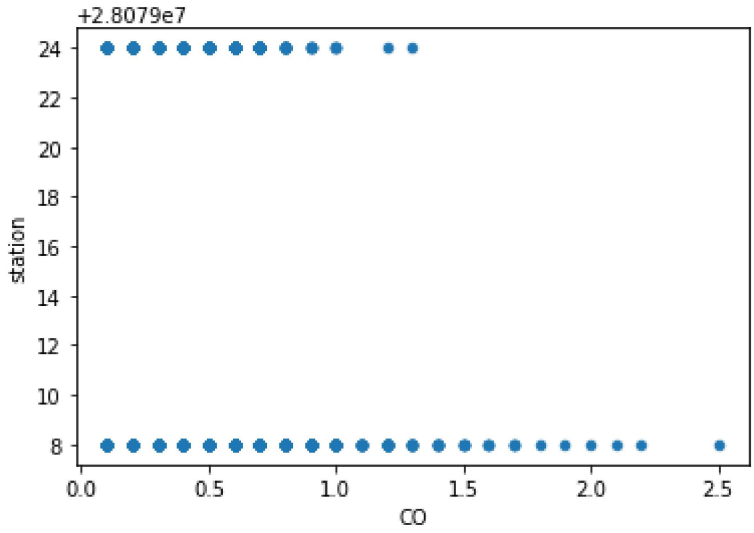
Out[533]: <AxesSubplot:ylabel='station'>



Scatter chart

In [534]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[534]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [535]:
```
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 [536]:
```
df.describe()
```

Out[536]:

	BEN	CO	EBE	NMHC	NO	NO_2	
count	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	109
mean	0.784014	0.279333	0.992213	0.215755	18.795529	31.262642	
std	0.632755	0.167922	0.804554	0.075169	40.038872	27.234732	
min	0.100000	0.100000	0.100000	0.050000	0.000000	1.000000	
25%	0.400000	0.200000	0.500000	0.160000	1.000000	9.000000	
50%	0.600000	0.200000	0.800000	0.220000	3.000000	24.000000	
75%	0.900000	0.300000	1.200000	0.250000	18.000000	47.000000	
max	7.000000	2.500000	9.700000	0.670000	525.000000	225.000000	15

In [537]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-537-9c3e63dc22cd> in <module>
----> 1 df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
      2  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem_
_(self, key)
   3028             if is_iterator(key):
   3029                 key = list(key)
-> 3030             indexer = self.loc._get_listlike_indexer(key, axis=1, rai
se_missing=True)[1]
   3031
   3032         # take() does not accept boolean indexers

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_li
stlike_indexer(self, key, axis, raise_missing)
   1264             keyarr, indexer, new_indexer = ax._reindex_non_unique(key
arr)
   1265
-> 1266         self._validate_read_indexer(keyarr, indexer, axis, raise_miss
ing=raise_missing)
   1267         return keyarr, indexer
   1268

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _valida
te_read_indexer(self, key, indexer, axis, raise_missing)
   1314             if raise_missing:
   1315                 not_found = list(set(key) - set(ax))
-> 1316                 raise KeyError(f"{not_found} not in index")
   1317
   1318             not_found = key[missing_mask]

KeyError: "['MXY', 'NOx', 'PXY', 'OXY'] not in index"
```

EDA AND VISUALIZATION

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

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

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

TO TRAIN THE MODEL AND MODEL BULDING

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

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

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

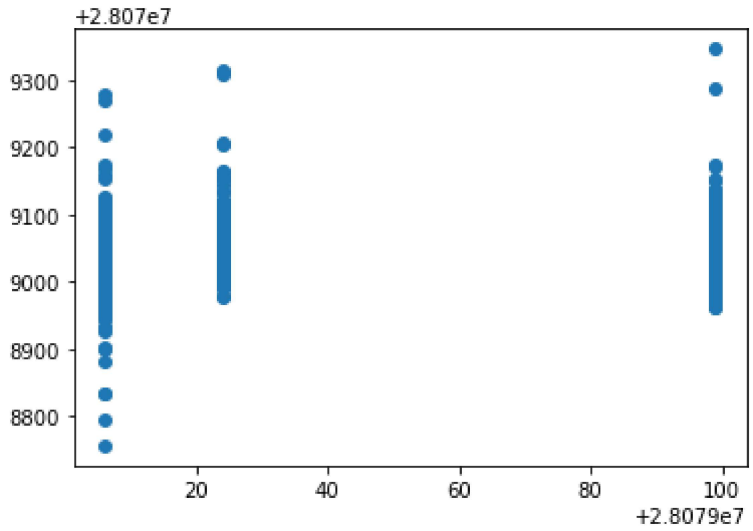
In [538]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[538]:

	Co-efficient
BEN	-37.039973
CO	-30.754126
EBE	7.531596
MXY	-1.633098
NMHC	-18.185915
NO_2	-0.183246
NOx	0.207947
OXY	14.764541
O_3	0.024889
PM10	-0.051958
PXY	2.828355
SO_2	-0.302665
TCH	124.896713
TOL	-1.148805

In [539]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[539]: <matplotlib.collections.PathCollection at 0x1cda5112df0>



ACCURACY

In [540]:
```
lr.score(x_test,y_test)
```

Out[540]: 0.26828993596392614

In [541]:
```
lr.score(x_train,y_train)
```

Out[541]: 0.2946269921837963

Ridge and Lasso

In [542]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [543]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[543]: Ridge(alpha=10)

Accuracy(Ridge)

In [544]:
```
rr.score(x_test,y_test)
```

Out[544]:
```
0.2700143791476979
```

In [545]:
```
rr.score(x_train,y_train)
```

Out[545]:
```
0.2942743131240403
```

In [546]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[546]:
```
Lasso(alpha=10)
```

In [547]:
```
la.score(x_train,y_train)
```

Out[547]:
```
0.033961539134792496
```

Accuracy(Lasso)

In [548]:
```
la.score(x_test,y_test)
```

Out[548]:
```
0.03863380766863844
```

In [549]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[549]:
```
ElasticNet()
```

In [550]:
```
en.coef_
```

Out[550]:
```
array([-7.07079106, -0.68564137,  0.42393327,  2.12561565, -0.        ,
       -0.24862351,  0.13491165,  1.24030818, -0.13651085,  0.08671355,
        1.92554874, -0.71980373,  1.47185506, -1.99024297])
```

In [551]:
```
en.intercept_
```

Out[551]:
```
28079063.075263444
```

In [552]:
```
prediction=en.predict(x_test)
```

In [553]:
```
en.score(x_test,y_test)
```

Out[553]:
```
0.11209585600575955
```

Evaluation Metrics

In [554]:
```
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)))
```

```
35.84169726560099
1460.280829285163
38.21362099154126
```

Logistic Regression

In [555]:
```
from sklearn.linear_model import LogisticRegression
```

In [556]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-556-60ce984ebae0> in <module>
----> 1 feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx',
'OXY', 'O_3',
      2  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
      3 target_vector=df[ 'station']

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem_
_(self, key)
   3028             if is_iterator(key):
   3029                 key = list(key)
-> 3030             indexer = self.loc._get_listlike_indexer(key, axis=1, rai
se_missing=True)[1]
   3031
   3032         # take() does not accept boolean indexers

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_li
stlike_indexer(self, key, axis, raise_missing)
   1264             keyarr, indexer, new_indexer = ax._reindex_non_unique(key
arr)
   1265
-> 1266         self._validate_read_indexer(keyarr, indexer, axis, raise_miss
ing=raise_missing)
   1267         return keyarr, indexer
   1268

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _valida
te_read_indexer(self, key, indexer, axis, raise_missing)
   1314             if raise_missing:
   1315                 not_found = list(set(key) - set(ax))
-> 1316                 raise KeyError(f"{not_found} not in index")
   1317
   1318             not_found = key[missing_mask]

KeyError: "['MXY', 'NOx', 'PXY', 'OXY'] not in index"
```

In [557]:
```
feature_matrix.shape
```

Out[557]: (24717, 14)

In [558]:
```
target_vector.shape
```

Out[558]: (24717,)

In [559]:
```
from sklearn.preprocessing import StandardScaler
```

In [560]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [561]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[561]:
```
LogisticRegression(max_iter=10000)
```

In [562]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [563]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [564]:
```
logr.classes_
```

Out[564]:
```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [565]:
```
logr.score(fs,target_vector)
```

Out[565]:
```
0.8951733624630821
```

In [566]:
```
logr.predict_proba(observation)[0][0]
```

Out[566]:
```
5.447205522232353e-13
```

In [567]:
```
logr.predict_proba(observation)
```

Out[567]:
```
array([[5.44720552e-13, 8.28692830e-44, 1.00000000e+00]])
```

Random Forest

In [568]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [569]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[569]:
```
RandomForestClassifier()
```

In [570]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [571]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[571]:
```
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 [572]:
```
grid_search.best_score_
```

Out[572]:
```
0.8953238457514017
```

In [573]:
```
rfc_best=grid_search.best_estimator_
```

In [574]:
```
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',
```

Out[574]:
```
[Text(2431.285714285714, 1993.2, 'EBE <= 1.005\ngini = 0.665\nsamples = 108
35\nvalue = [5223, 5948, 6130]\nclass = c'),
 Text(1275.4285714285713, 1630.8000000000002, 'TOL <= 0.895\ngini = 0.579\n
samples = 6002\nvalue = [1522, 5411, 2640]\nclass = b'),
 Text(637.7142857142857, 1268.4, 'NMHC <= 0.185\ngini = 0.182\nsamples = 16
63\nvalue = [17, 2344, 245]\nclass = b'),
 Text(318.85714285714283, 906.0, 'TCH <= 1.325\ngini = 0.343\nsamples = 749
\nvalue = [17, 921, 235]\nclass = b'),
 Text(159.42857142857142, 543.5999999999999, 'OXY <= 0.835\ngini = 0.099\ns
amples = 570\nvalue = [10, 844, 36]\nclass = b'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.544\nsamples = 52\nv
alue = [8, 47, 24]\nclass = b'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.034\nsamples = 518\n
value = [2, 797, 12]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'EBE <= 0.53\ngini = 0.431\nsam
ples = 179\nvalue = [7, 77, 199]\nclass = c'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.142\nsamples = 42\n
value = [5, 60, 0]\nclass = b'),
 Text(558.0, 181.19999999999982, 'gini = 0.161\nsamples = 137\nvalue = [2,
```

Conclusion

Accuracy

Linear Regression:0.6273759315249119

Ridge Regression:0.36239748001773564

Lasso Regression:0.6048567528892628

ElasticNet Regression:0.4525528104685921

Logistic Regression:0.9293697325027482

Random Forest:0.9611307585114501

From the above data, we can conclude that random forest regression is preferrable to other regression types