20104016

DEENA

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("madrid_2018.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN

69096 rows × 16 columns

Data Cleaning and Data Preprocessing

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

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

Out[4]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4562 entries, 1 to 69078
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     4562 non-null   object 
 1   BEN      4562 non-null   float64
 2   CH4      4562 non-null   float64
 3   CO       4562 non-null   float64
 4   EBE      4562 non-null   float64
 5   NMHC     4562 non-null   float64
 6   NO       4562 non-null   float64
 7   NO_2     4562 non-null   float64
 8   NOx      4562 non-null   float64
 9   O_3      4562 non-null   float64
 10  PM10     4562 non-null   float64
 11  PM25     4562 non-null   float64
 12  SO_2     4562 non-null   float64
 13  TCH      4562 non-null   float64
 14  TOL      4562 non-null   float64
 15  station  4562 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 605.9+ KB
```

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

Out[6]:

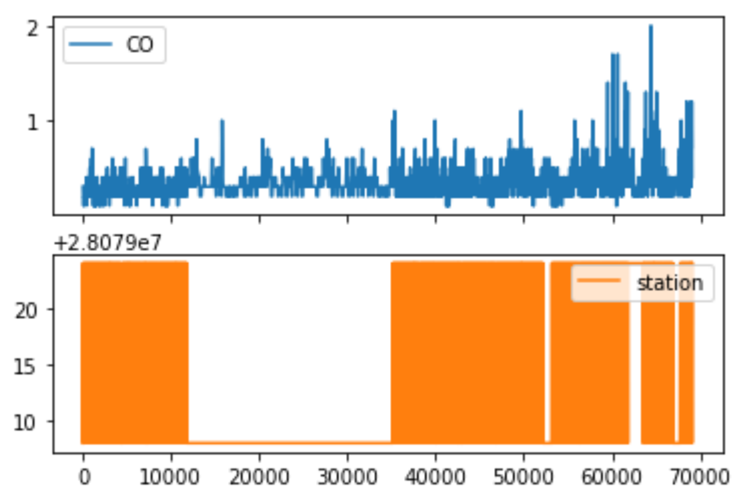
	CO	station
1	0.3	28079008
6	0.2	28079024
25	0.2	28079008
30	0.2	28079024
49	0.2	28079008
...	...	...
69030	0.7	28079024
69049	1.2	28079008
69054	0.6	28079024
69073	1.0	28079008
69078	0.4	28079024

4562 rows × 2 columns

Line chart

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

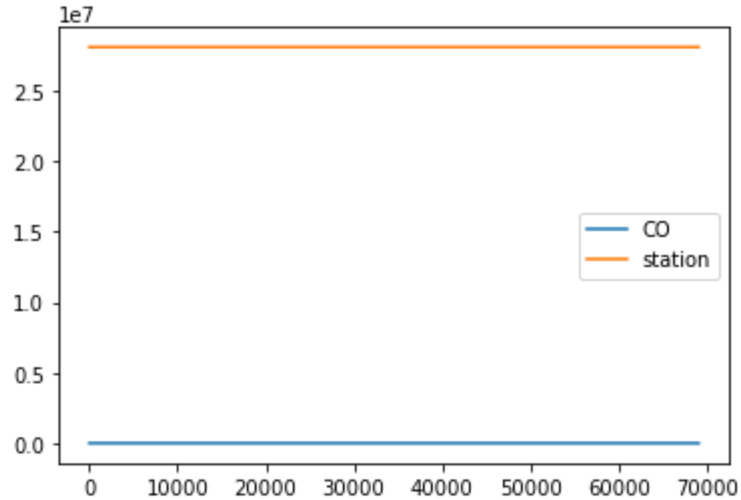
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

Out[8]: <AxesSubplot:>

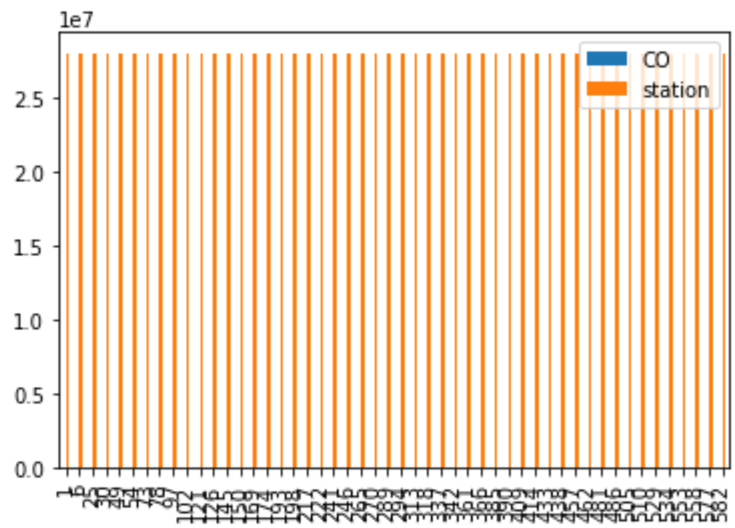


Bar chart

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

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

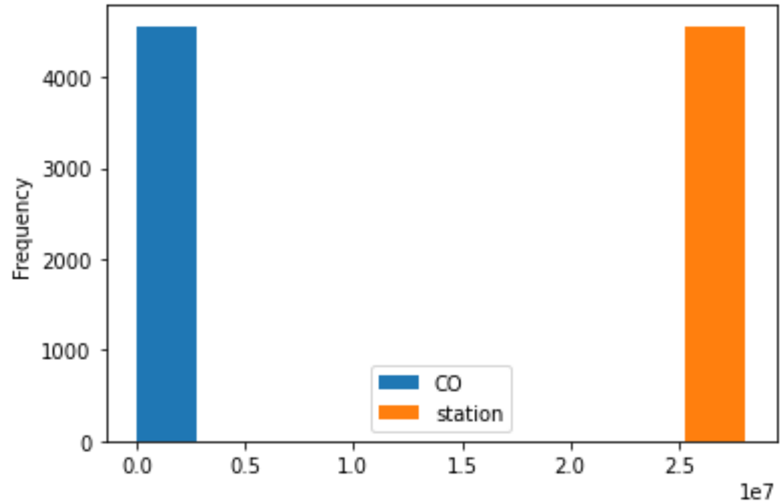
Out[10]: <AxesSubplot:>



Histogram

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

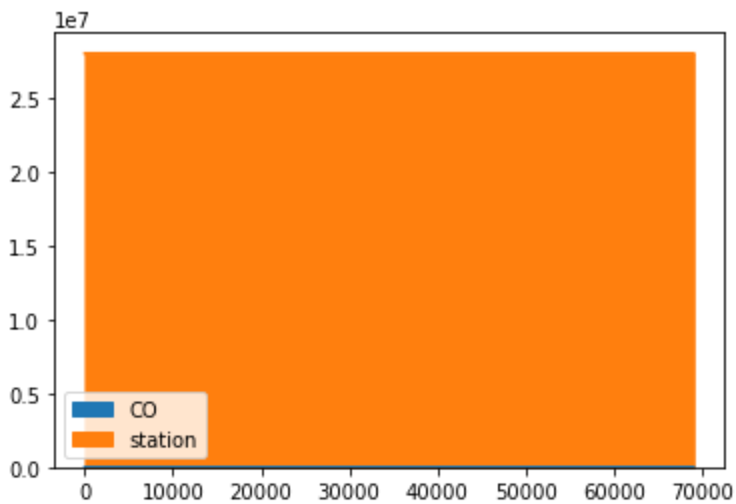
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

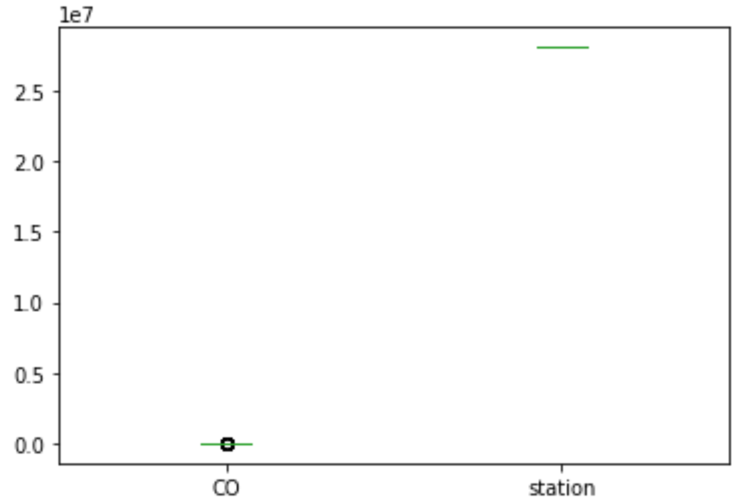
Out[12]: <AxesSubplot:>



Box chart

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

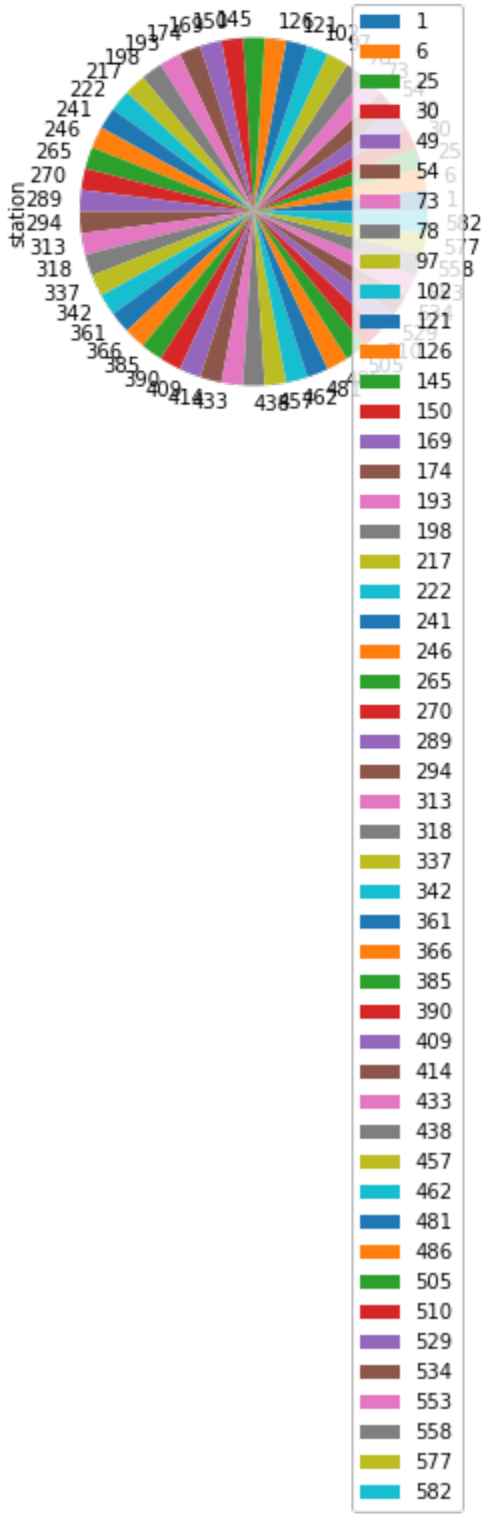
Out[13]: <AxesSubplot:>



Pie chart

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

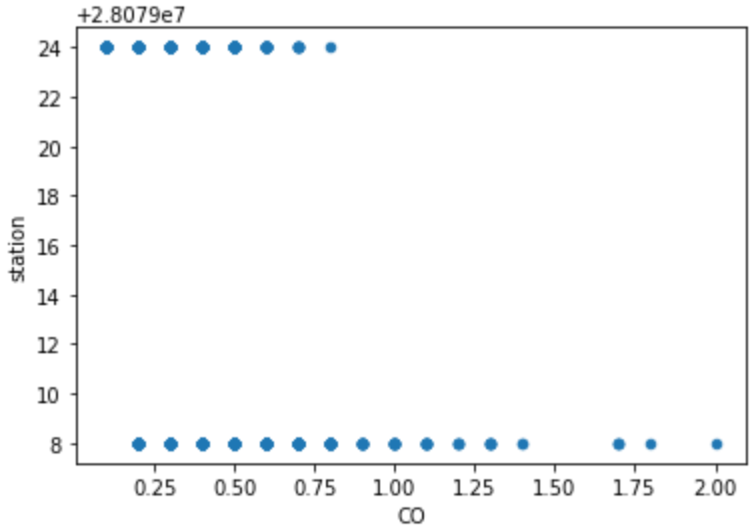
Out[14]: <AxesSubplot:ylabel='station'>



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'>
Int64Index: 4562 entries, 1 to 69078
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     4562 non-null   object 
 1   BEN      4562 non-null   float64
 2   CH4      4562 non-null   float64
 3   CO       4562 non-null   float64
 4   EBE      4562 non-null   float64
 5   NMHC     4562 non-null   float64
 6   NO       4562 non-null   float64
 7   NO_2     4562 non-null   float64
 8   NOx      4562 non-null   float64
 9   O_3      4562 non-null   float64
 10  PM10     4562 non-null   float64
 11  PM25     4562 non-null   float64
 12  SO_2     4562 non-null   float64
 13  TCH      4562 non-null   float64
 14  TOL      4562 non-null   float64
```

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

Out[17]:

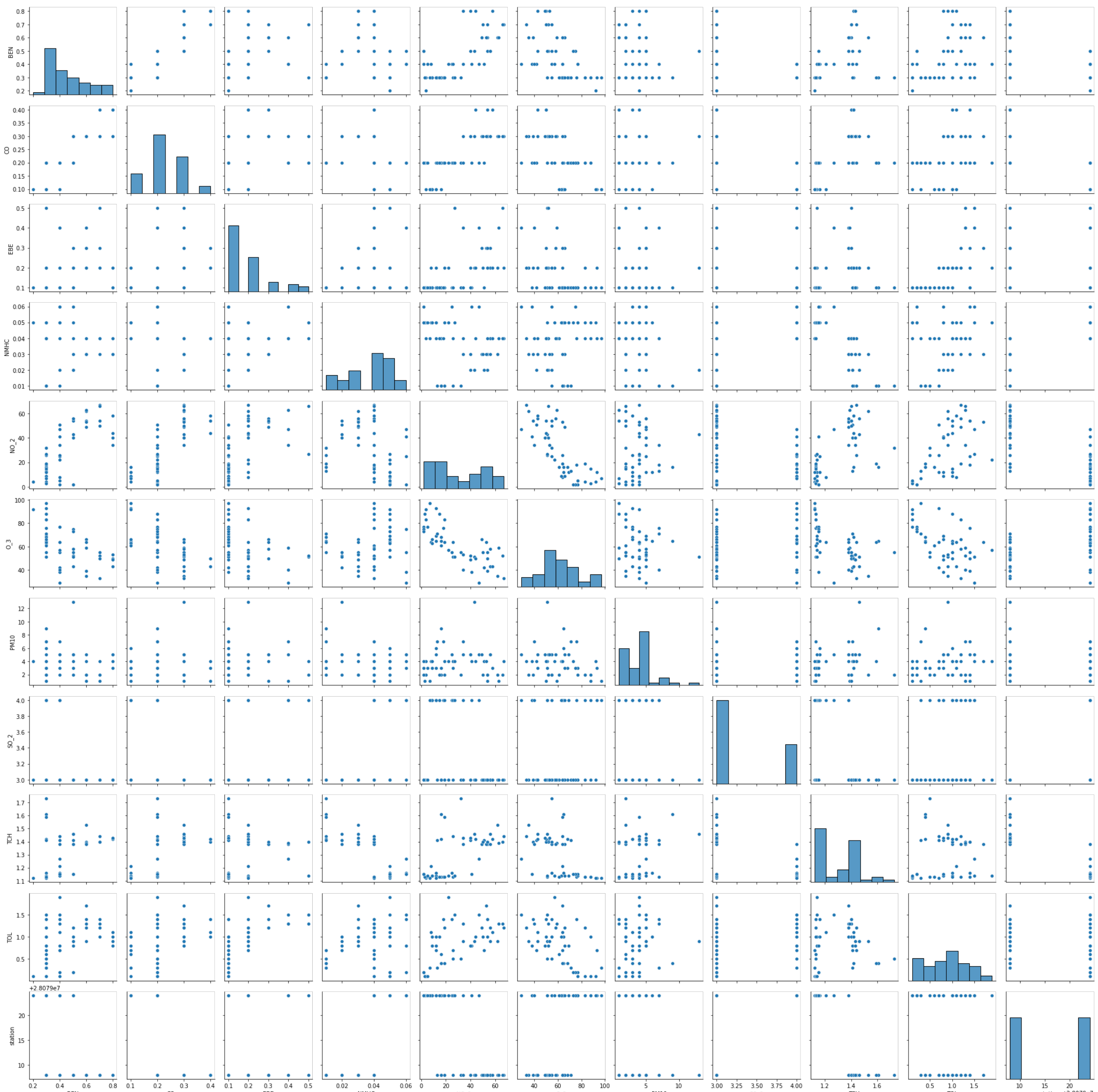
	BEN	CH4	CO	EBE	NMHC	NO	NO_
count	4562.00000	4562.000000	4562.000000	4562.000000	4562.000000	4562.000000	4562.00000
mean	0.69349	1.329163	0.330579	0.286782	0.056773	21.742218	44.15212
std	0.46832	0.214399	0.161489	0.354442	0.037711	35.539531	30.23401
min	0.10000	0.020000	0.100000	0.100000	0.000000	1.000000	1.00000
25%	0.40000	1.120000	0.200000	0.100000	0.030000	1.000000	20.00000
50%	0.60000	1.390000	0.300000	0.200000	0.050000	9.000000	41.00000
75%	0.90000	1.420000	0.400000	0.300000	0.070000	27.000000	64.00000
max	6.60000	3.920000	2.000000	7.400000	0.490000	431.000000	184.00000

In [18]:
```
df1=df[['BEN', 'CO', 'EBE','NMHC', 'NO_2', 'O_3',
        'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

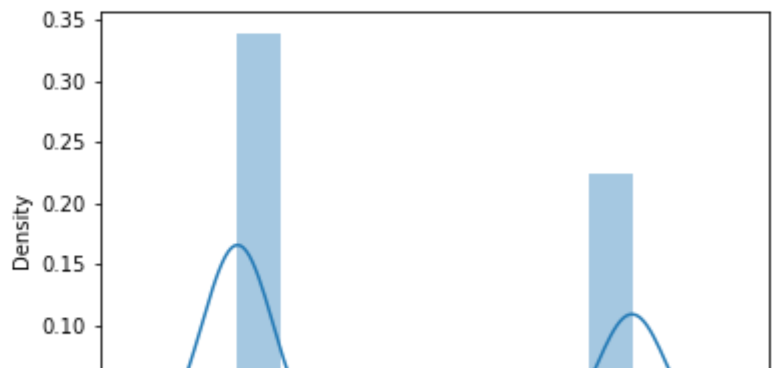
Out[19]: <seaborn.axisgrid.PairGrid at 0x26c9391ae50>



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

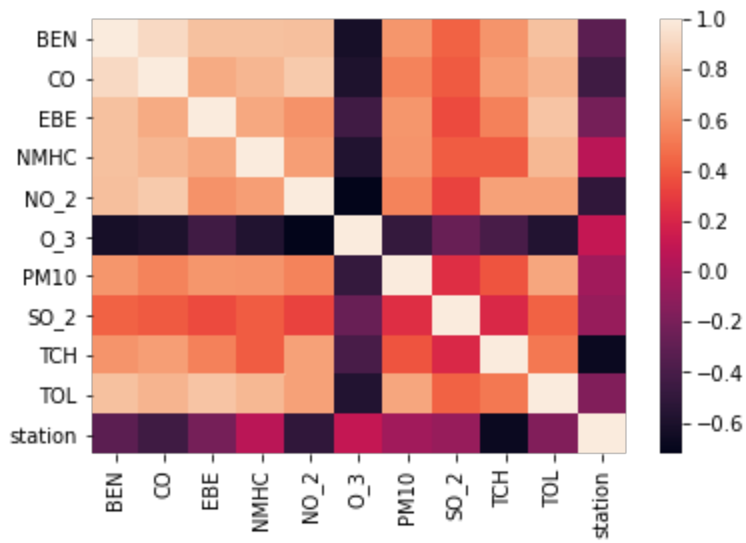
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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_2', 'O_3',
       'PM10', '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]: 28079043.1001105

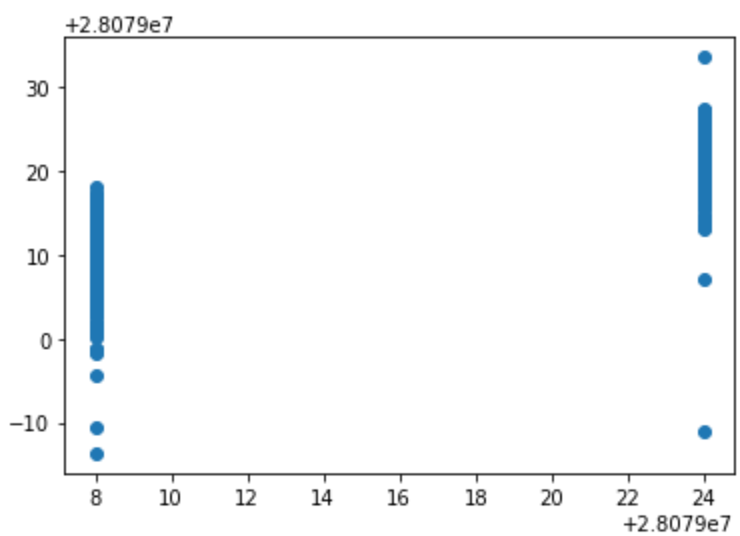
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-0.060927
CO	-19.295989
EBE	0.219468
NMHC	150.456442
NO_2	-0.152606
O_3	-0.083677
PM10	0.106848
SO_2	-0.034261
TCH	-15.489608
TOL	-0.174559

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

Out[27]: <matplotlib.collections.PathCollection at 0x26c9a544a30>



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```
Out[28]: 0.8110631136261367

In [29]:
```
lr.score(x_train,y_train)
```
Out[29]: 0.8017336125303979

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]: 0.686599407328742

In [33]:
```
rr.score(x_train,y_train)
```
Out[33]: 0.6780003514302112

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

In [35]:
```
la.score(x_train,y_train)
```
Out[35]: 0.4107108090429348

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```
Out[36]: 0.40771049345016597

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.        , -0.        ,  0.        ,  0.        , -0.26583293,
       -0.14189405,  0.23586415,  0.08329442, -0.04766456,  0.12784576])
```

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

Out[39]:
```
28079028.900839474
```

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

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

Out[41]:
```
0.46059452684249924
```

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)))
```

```
4.9453085772802
33.46978436780551
5.785307629487433
```

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]:
```
(4562, 10)
```

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

Out[46]:
```
(4562,)
```

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([28079008, 28079024], dtype=int64)
```

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

Out[53]:
```
0.9888206926786497
```

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

Out[54]:
```
1.0
```

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

Out[55]:
```
array([[1.00000000e+00, 1.42669593e-19]])
```

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="ac
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]:
```
0.9934222296505195
```

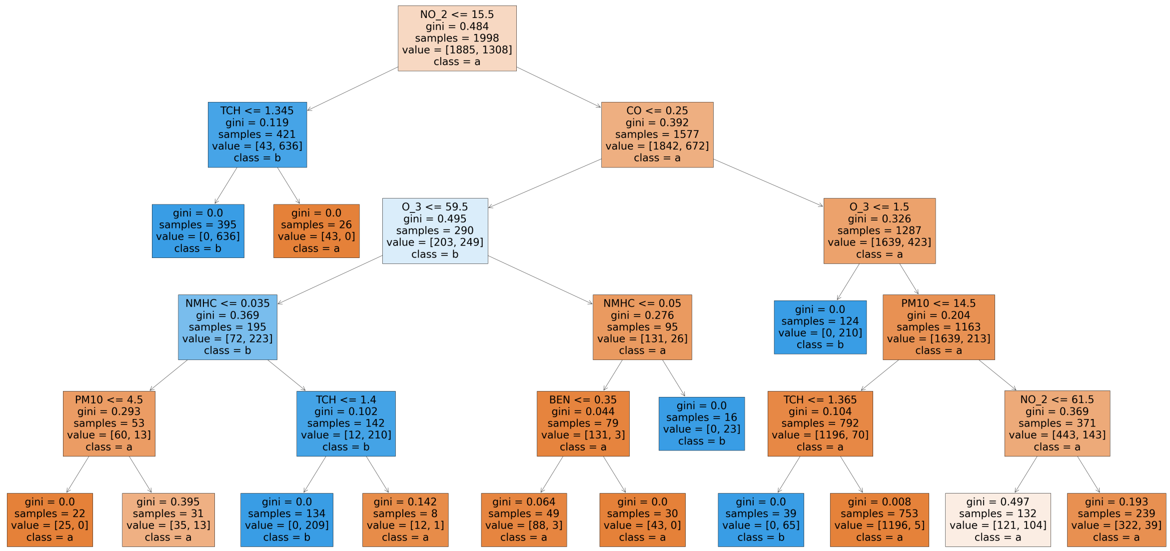
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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'
```

Out[62]:
```
[Text(1757.6999999999998, 1993.2, 'NO_2 <= 15.5\ngini = 0.484\nsamples = 199
8\nvalue = [1885, 1308]\nclass = a'),
 Text(1004.4, 1630.8000000000002, 'TCH <= 1.345\ngini = 0.119\nsamples = 421\
nvalue = [43, 636]\nclass = b'),
 Text(781.1999999999999, 1268.4, 'gini = 0.0\nsamples = 395\nvalue = [0, 63
6]\nclass = b'),
 Text(1227.6, 1268.4, 'gini = 0.0\nsamples = 26\nvalue = [43, 0]\nclass = a
'),
 Text(2511.0, 1630.8000000000002, 'CO <= 0.25\ngini = 0.392\nsamples = 1577\n
value = [1842, 672]\nclass = a'),
 Text(1674.0, 1268.4, 'O_3 <= 59.5\ngini = 0.495\nsamples = 290\nvalue = [20
3, 249]\nclass = b'),
 Text(892.8, 906.0, 'NMHC <= 0.035\ngini = 0.369\nsamples = 195\nvalue = [72,
223]\nclass = b'),
 Text(446.4, 543.5999999999999, 'PM10 <= 4.5\ngini = 0.293\nsamples = 53\nval
ue = [60, 13]\nclass = a'),
 Text(223.2, 181.19999999999982, 'gini = 0.0\nsamples = 22\nvalue = [25, 0]\n
class = a'),
 Text(669.5999999999999, 181.19999999999982, 'gini = 0.395\nsamples = 31\nval
ue = [35, 13]\nclass = a'),
 Text(1339.1999999999998, 543.5999999999999, 'TCH <= 1.4\ngini = 0.102\nsampl
es = 142\nvalue = [12, 210]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.0\nsamples = 134\nvalue = [0, 20
9]\nclass = b'),
 Text(1562.3999999999999, 181.19999999999982, 'gini = 0.142\nsamples = 8\nval
ue = [12, 1]\nclass = a'),
 Text(2455.2, 906.0, 'NMHC <= 0.05\ngini = 0.276\nsamples = 95\nvalue = [131,
26]\nclass = a'),
 Text(2232.0, 543.5999999999999, 'BEN <= 0.35\ngini = 0.044\nsamples = 79\nva
lue = [131, 3]\nclass = a'),
 Text(2008.8, 181.19999999999982, 'gini = 0.064\nsamples = 49\nvalue = [88,
3]\nclass = a'),
 Text(2455.2, 181.19999999999982, 'gini = 0.0\nsamples = 30\nvalue = [43, 0]\
nclass = a'),
 Text(2678.3999999999996, 543.5999999999999, 'gini = 0.0\nsamples = 16\nvalue
= [0, 23]\nclass = b'),
 Text(3348.0, 1268.4, 'O_3 <= 1.5\ngini = 0.326\nsamples = 1287\nvalue = [163
9, 423]\nclass = a'),
 Text(3124.7999999999997, 906.0, 'gini = 0.0\nsamples = 124\nvalue = [0, 21
0]\nclass = b'),
 Text(3571.2, 906.0, 'PM10 <= 14.5\ngini = 0.204\nsamples = 1163\nvalue = [16
39, 213]\nclass = a'),
 Text(3124.7999999999997, 543.5999999999999, 'TCH <= 1.365\ngini = 0.104\nsam
ples = 792\nvalue = [1196, 70]\nclass = a'),
 Text(2901.6, 181.19999999999982, 'gini = 0.0\nsamples = 39\nvalue = [0, 65]\
nclass = b'),
 Text(3348.0, 181.19999999999982, 'gini = 0.008\nsamples = 753\nvalue = [119
6, 5]\nclass = a'),
 Text(4017.6, 543.5999999999999, 'NO_2 <= 61.5\ngini = 0.369\nsamples = 371\n
value = [443, 143]\nclass = a'),
 Text(3794.3999999999996, 181.19999999999982, 'gini = 0.497\nsamples = 132\nv
```

```
alue = [121, 104]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.193\nsamples = 239\nvalue = [322,
39]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression :0.8017336125303979

Ridge Regression :0.4107108090429348

Lasso Regression :0.40771049345016597

ElasticNet Regression : 0.460594526842499 24

Logistic Regression : 0.9888206926786497

Random Forest :0.9934222296505195

Random Forest is suitable for this dataset