

```
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("madrid_2007.csv")
```

```
In [3]: df.head()
```

Out[3]:

	date	BEN	CO	EBE	MXV	NMHC	NO_2	NOx	OXY	O_3	PM1
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	156.19999
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	80.80999
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	53.09999
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	105.30000
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.50000

```
In [4]: df=df.dropna()
```

```
In [5]: df.columns
```

Out[5]: Index(['date', 'BEN', 'CO', 'EBE', 'MXV', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3', 'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'], dtype='object')

In [6]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25443 entries, 4 to 225119
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  -
0   date        25443 non-null  object
1   BEN         25443 non-null  float64
2   CO          25443 non-null  float64
3   EBE         25443 non-null  float64
4   MXY         25443 non-null  float64
5   NMHC        25443 non-null  float64
6   NO_2        25443 non-null  float64
7   NOx         25443 non-null  float64
8   OXY         25443 non-null  float64
9   O_3         25443 non-null  float64
10  PM10        25443 non-null  float64
11  PM25        25443 non-null  float64
12  PXY         25443 non-null  float64
13  SO_2        25443 non-null  float64
14  TCH         25443 non-null  float64
15  TOL         25443 non-null  float64
16  station     25443 non-null  int64
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

In [7]: data=df[['CO', 'station']]
data

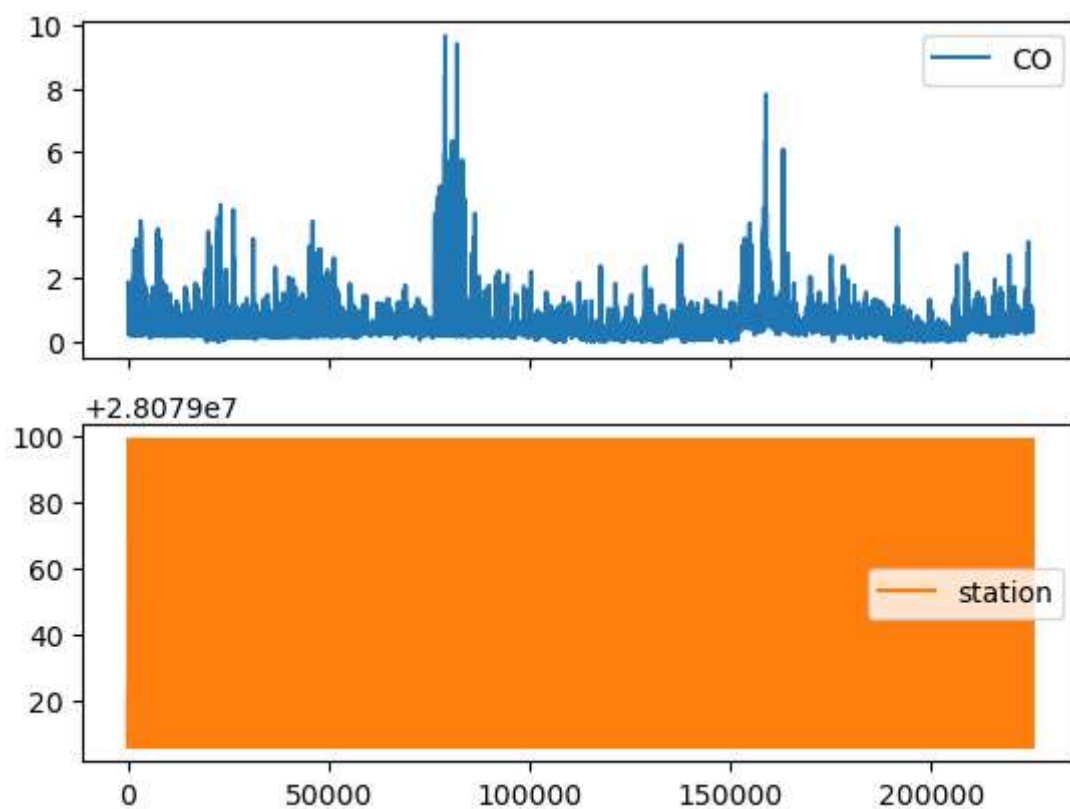
Out[7]:

	CO	station
4	1.86	28079006
21	0.31	28079024
25	1.42	28079099
30	1.89	28079006
47	0.30	28079024
...
225073	0.47	28079006
225094	0.45	28079099
225098	0.41	28079006
225115	0.45	28079024
225119	0.40	28079099

25443 rows × 2 columns

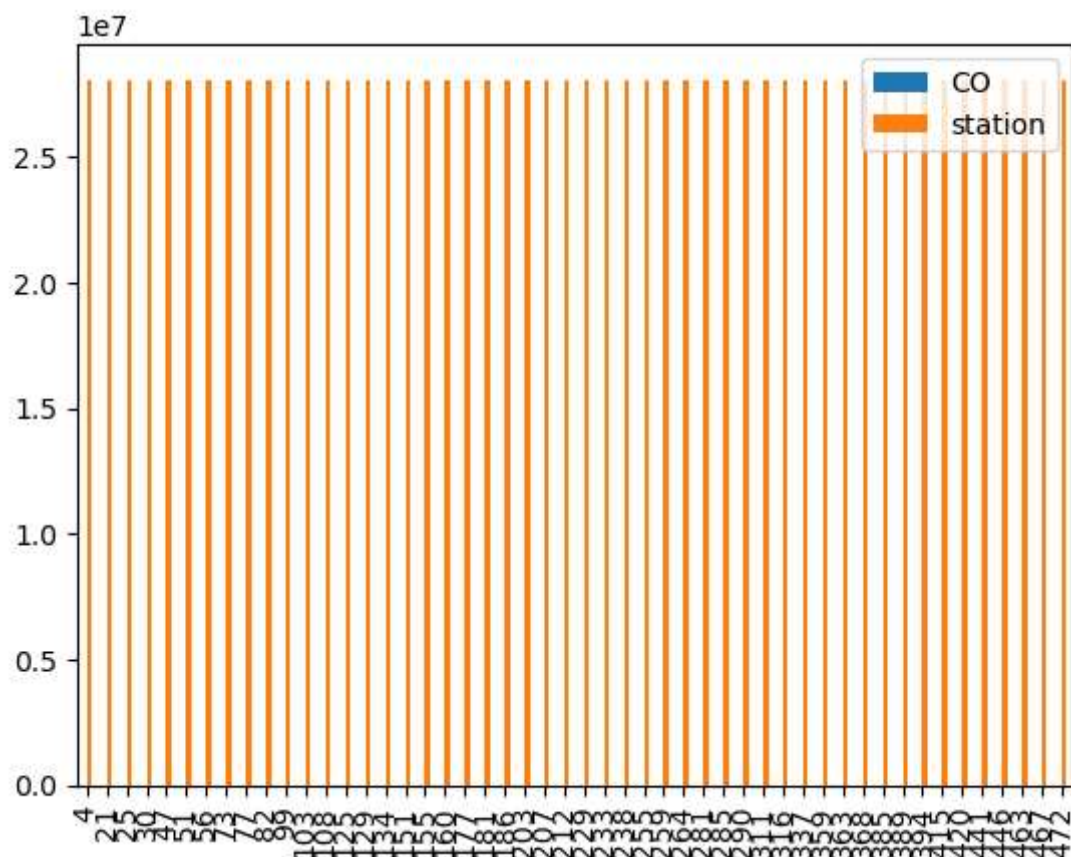
```
In [8]: data.plot.line(subplots=True)
```

```
Out[8]: array([<Axes: >, <Axes: >], dtype=object)
```



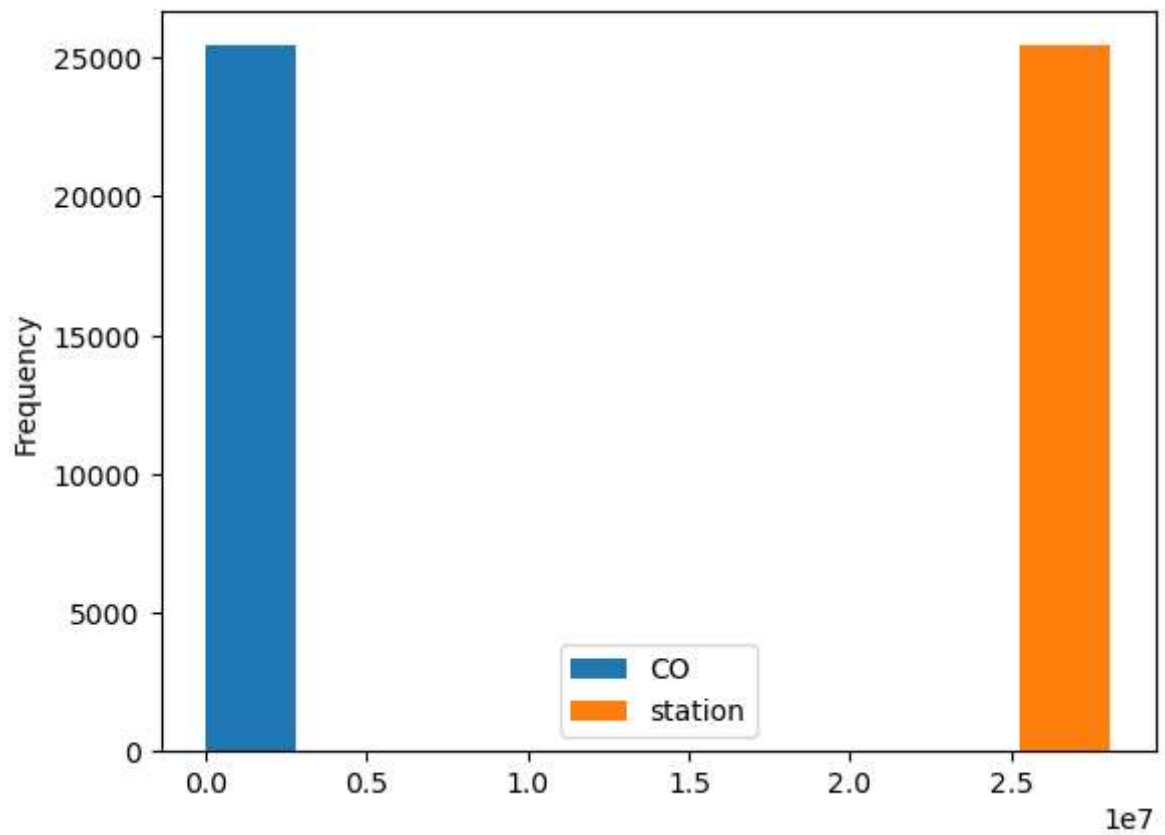
```
In [9]: b=data[0:50]  
b.plot.bar()
```

Out[9]: <Axes: >



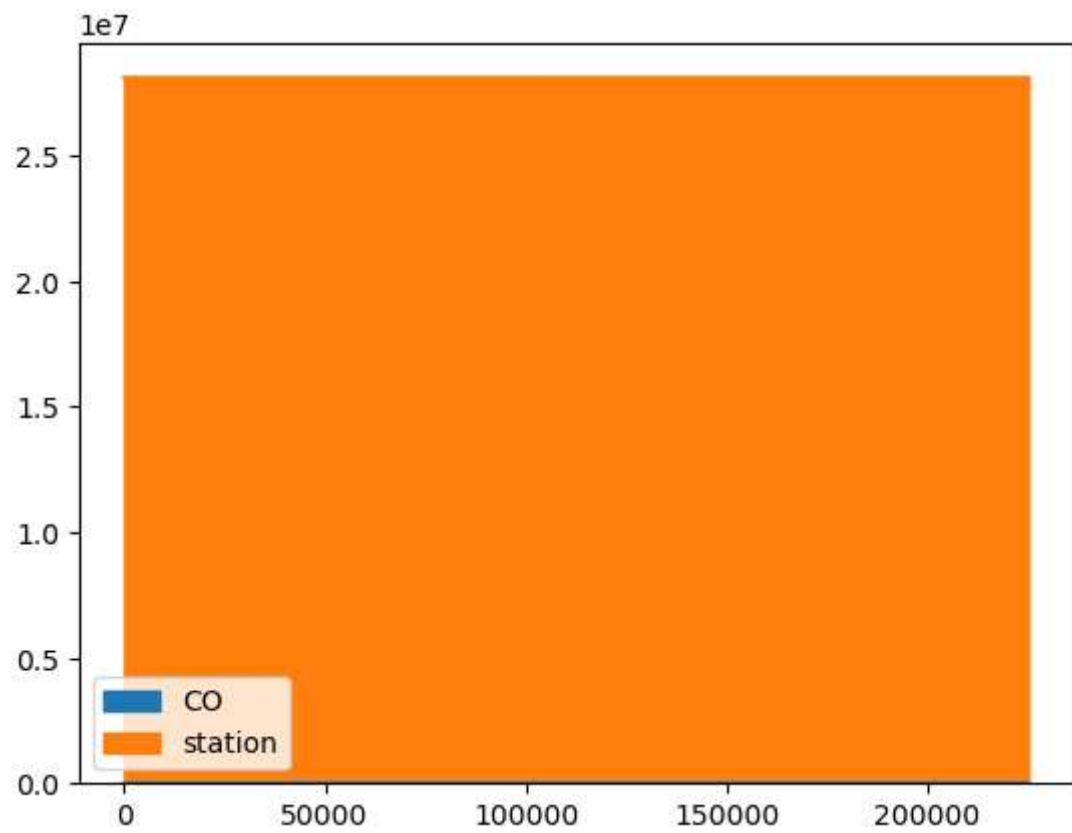
```
In [10]: data.plot.hist()
```

```
Out[10]: <Axes: ylabel='Frequency'>
```



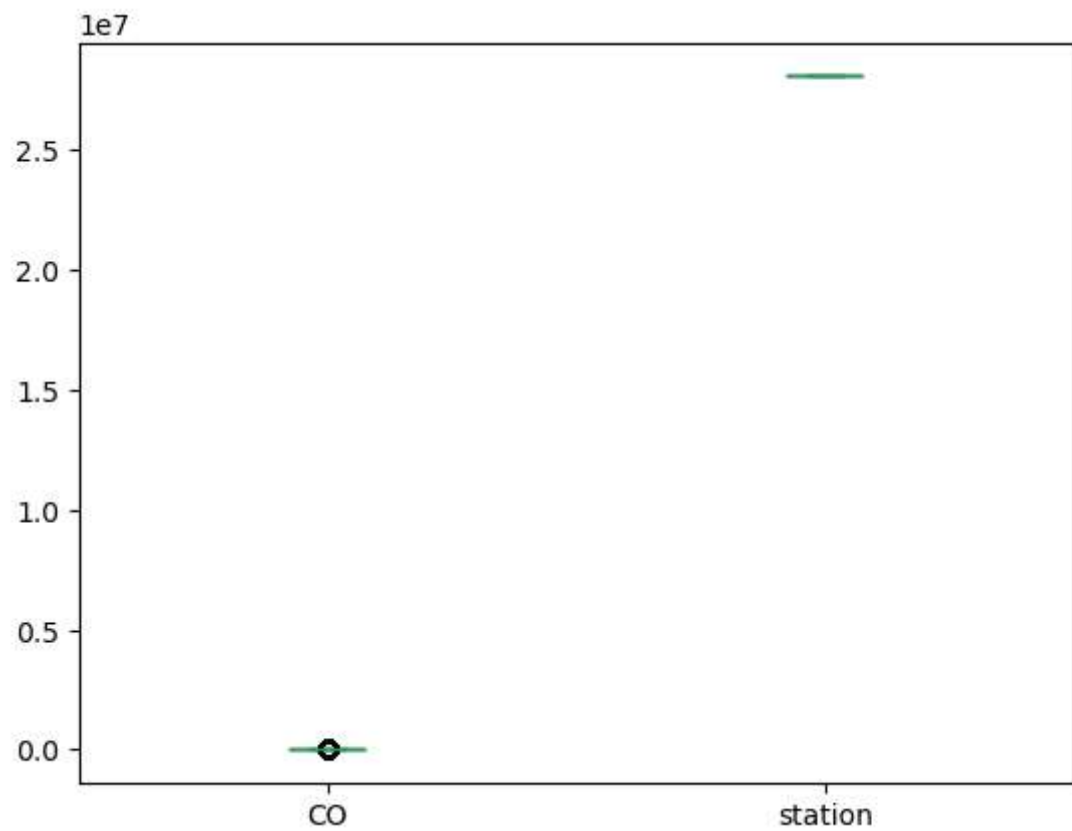
```
In [11]: data.plot.area()
```

```
Out[11]: <Axes: >
```



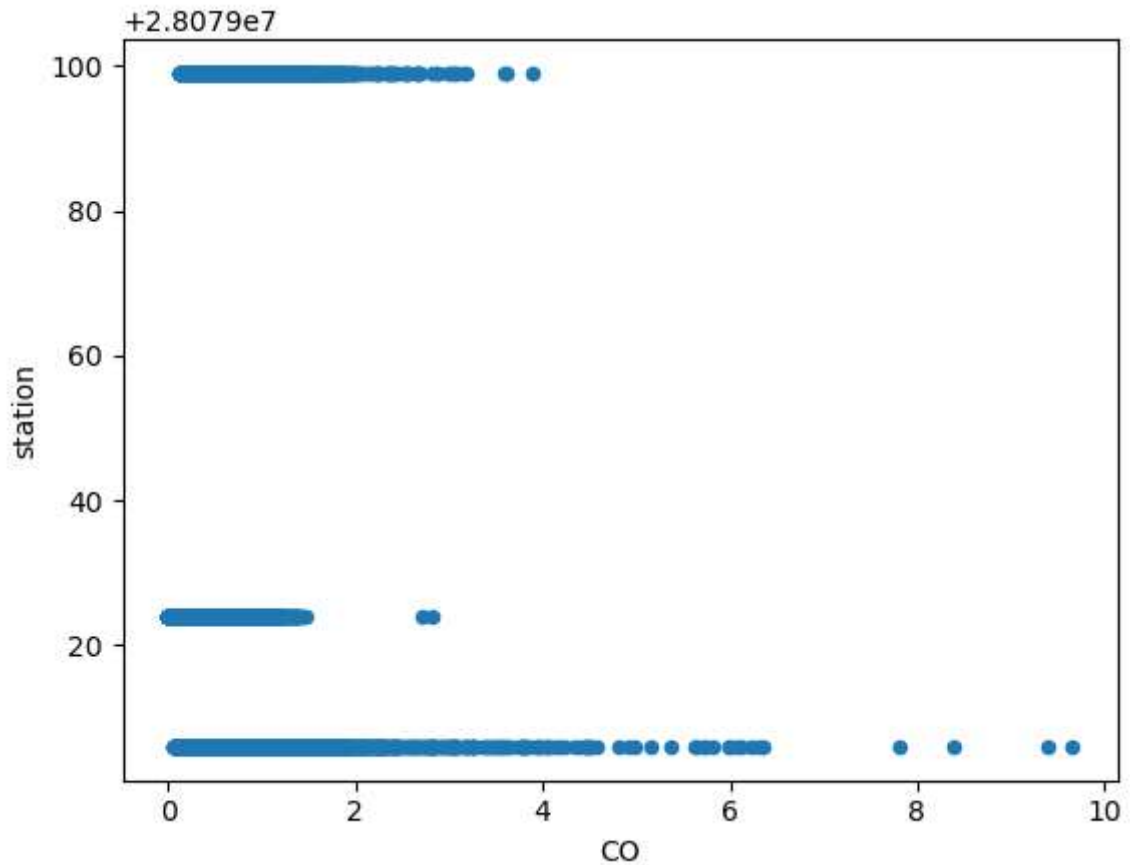
```
In [12]: data.plot.box()
```

```
Out[12]: <Axes: >
```



```
In [13]: data.plot.scatter(x='CO',y='station')
```

```
Out[13]: <Axes: xlabel='CO', ylabel='station'>
```



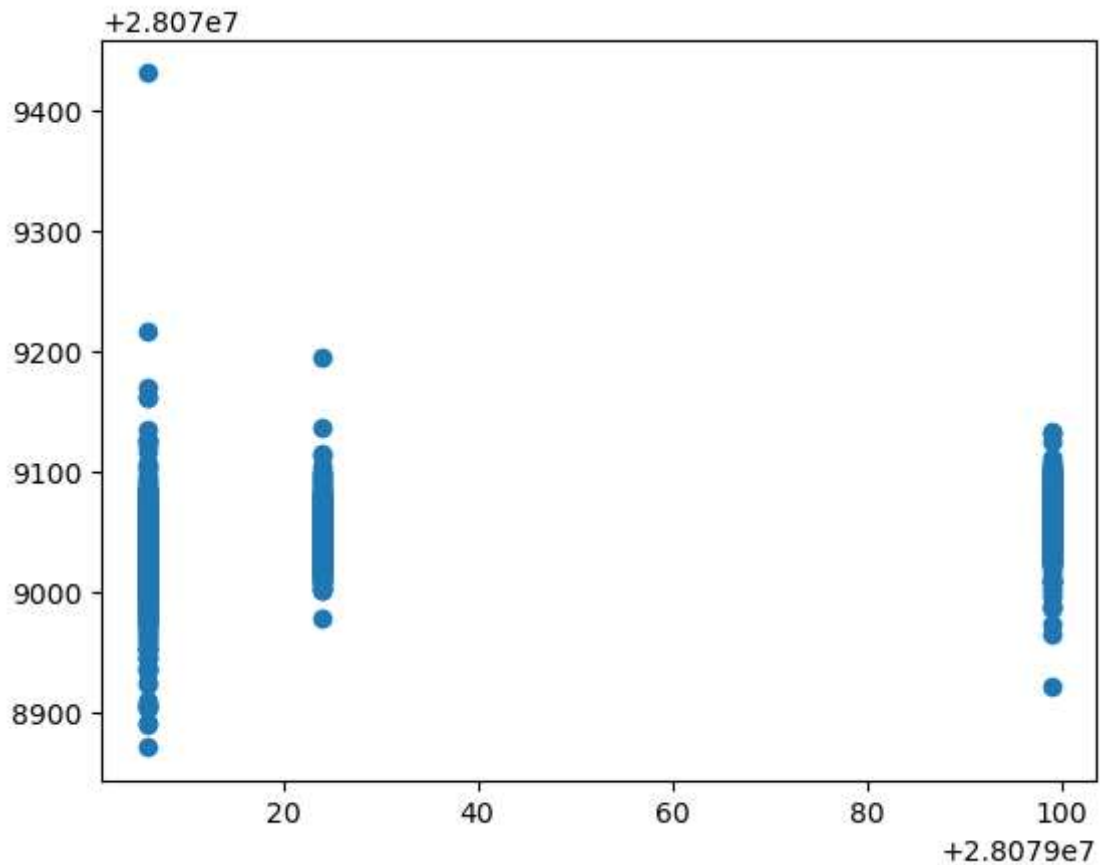
```
In [14]: x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',  
              'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]  
y=df['station']
```

```
In [15]: 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 [16]: from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
lr.intercept_
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x27955286110>



```
In [17]: print(lr.score(x_test,y_test))
print(lr.score(x_train,y_train))
```

0.13612242113139272
0.16831326728048046

Ridge and Lasso

```
In [18]: from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
print(rr.score(x_test,y_test))
print(rr.score(x_train,y_train))
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
0.13616589275057867
0.1682643542232688
```

```
Out[18]:
```

▾ Lasso
 Lasso(alpha=10)

```
In [19]: la.score(x_test,y_test)
```

```
Out[19]: 0.01265890622426824
```

ElasticNet

```
In [20]: from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

```
Out[20]:
```

▾ ElasticNet
 ElasticNet()

```
In [21]: en.coef_
```

```
Out[21]: array([-8.25636166,  0.          ,  0.          ,  0.06763943, -0.          ,
                0.04267699, -0.04992269,  0.68817514, -0.0531482 ,  0.1565664 ,
                0.69923193, -0.          ,  0.          ,  1.04751697])
```

```
In [22]: en.intercept_
```

```
Out[22]: 28079046.260731436
```

```
In [23]: prediction=en.predict(x_test)
```

```
In [24]: en.score(x_test,y_test)
```

```
Out[24]: 0.06602527642607048
```

Evaluation Metrics

```
In [25]: 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)))
```

```
36.606726880655906
1534.7742224731157
39.17619458897349
```

Logistics Regression

```
In [26]: from sklearn.linear_model import LogisticRegression
```

```
In [27]: feature_matrix=df[['BEN', 'CO', 'EBE', 'MXV', 'NMHC', 'NO_2', 'NOx', 'OXY', 'C
'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

```
In [28]: from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(feature_matrix)
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
logr.score(fs,target_vector)
```

```
Out[28]: 0.8146838030106512
```

```
In [29]: observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
logr.predict_proba(observation)
```

```
Out[29]: array([[1.08275398e-19, 1.80383815e-19, 1.00000000e+00]])
```

Random Forest

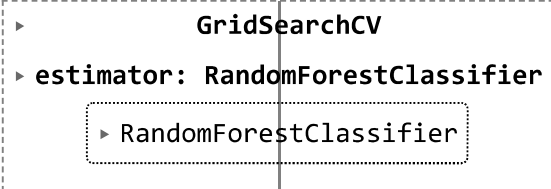
```
In [30]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
Out[30]: ▼ RandomForestClassifier
RandomForestClassifier()
```

```
In [31]: parameters={'max_depth':[1,2,3,4,5],  
                    'min_samples_leaf':[5,10,15,20,25],  
                    'n_estimators':[10,20,30,40,50]  
                    }
```

```
In [32]: 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[32]:
```



```
  ▶ GridSearchCV  
  ▶ estimator: RandomForestClassifier  
      ▶ RandomForestClassifier
```

```
In [33]: rfc_best=grid_search.best_estimator_  
         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[33]: [Text(0.5145833333333333, 0.9166666666666666, 'CO <= 0.315\ngini = 0.666\nsam
ples = 11323\nvalue = [6022, 5721, 6067]\nnclass = c'),
Text(0.2666666666666666, 0.75, 'TCH <= 1.385\ngini = 0.564\nsamples = 4038
\nvalue = [994, 3762, 1632]\nnclass = b'),
Text(0.1333333333333333, 0.5833333333333334, 'CO <= 0.135\ngini = 0.634\nsa
mples = 2740\nvalue = [916, 2021, 1399]\nnclass = b'),
Text(0.06666666666666667, 0.4166666666666667, 'NO_2 <= 27.26\ngini = 0.185\n
samples = 648\nvalue = [84, 923, 20]\nnclass = b'),
Text(0.0333333333333333, 0.25, 'OXY <= 0.355\ngini = 0.107\nsamples = 543\n
value = [28, 813, 20]\nnclass = b'),
Text(0.016666666666666666, 0.0833333333333333, 'gini = 0.0\nsamples = 10\nv
alue = [18, 0, 0]\nnclass = a'),
Text(0.05, 0.0833333333333333, 'gini = 0.069\nsamples = 533\nvalue = [10, 8
13, 20]\nnclass = b'),
Text(0.1, 0.25, 'NMHC <= 0.175\ngini = 0.447\nsamples = 105\nvalue = [56, 11
0, 0]\nnclass = b'),
Text(0.0833333333333333, 0.0833333333333333, 'gini = 0.222\nsamples = 36\n
value = [55, 8, 0]\nnclass = a'),
Text(0.11666666666666667, 0.0833333333333333, 'gini = 0.019\nsamples = 69\n
value = [1, 102, 0]\nnclass = b'),
Text(0.2, 0.4166666666666667, 'TOL <= 1.005\ngini = 0.653\nsamples = 2092\nv
alue = [832, 1098, 1379]\nnclass = c'),
Text(0.16666666666666666, 0.25, 'NOx <= 24.36\ngini = 0.435\nsamples = 452\n
value = [148, 516, 54]\nnclass = b'),
Text(0.15, 0.0833333333333333, 'gini = 0.175\nsamples = 339\nvalue = [21, 4
80, 29]\nnclass = b'),
Text(0.1833333333333333, 0.0833333333333333, 'gini = 0.489\nsamples = 113
\nvalue = [127, 36, 25]\nnclass = a'),
Text(0.2333333333333334, 0.25, 'OXY <= 0.485\ngini = 0.618\nsamples = 1640
\nvalue = [684, 582, 1325]\nnclass = c'),
Text(0.21666666666666667, 0.0833333333333333, 'gini = 0.409\nsamples = 262
\nvalue = [301, 67, 36]\nnclass = a'),
Text(0.25, 0.0833333333333333, 'gini = 0.566\nsamples = 1378\nvalue = [383,
515, 1289]\nnclass = c'),
Text(0.4, 0.5833333333333334, 'NOx <= 33.9\ngini = 0.266\nsamples = 1298\nva
lue = [78, 1741, 233]\nnclass = b'),
Text(0.3333333333333333, 0.4166666666666667, 'EBE <= 1.455\ngini = 0.022\nsa
mples = 332\nvalue = [5, 528, 1]\nnclass = b'),
Text(0.3, 0.25, 'BEN <= 0.225\ngini = 0.004\nsamples = 321\nvalue = [0, 512,
1]\nnclass = b'),
Text(0.2833333333333333, 0.0833333333333333, 'gini = 0.03\nsamples = 44\nva
lue = [0, 65, 1]\nnclass = b'),
Text(0.31666666666666665, 0.0833333333333333, 'gini = 0.0\nsamples = 277\nv
alue = [0, 447, 0]\nnclass = b'),
Text(0.36666666666666664, 0.25, 'NMHC <= 0.415\ngini = 0.363\nsamples = 11\n
value = [5, 16, 0]\nnclass = b'),
Text(0.35, 0.0833333333333333, 'gini = 0.473\nsamples = 6\nvalue = [5, 8,
0]\nnclass = b'),
Text(0.3833333333333333, 0.0833333333333333, 'gini = 0.0\nsamples = 5\nval
ue = [0, 8, 0]\nnclass = b'),
Text(0.4666666666666667, 0.4166666666666667, 'SO_2 <= 7.765\ngini = 0.336\ns
amples = 966\nvalue = [73, 1213, 232]\nnclass = b'),
Text(0.4333333333333333, 0.25, 'NMHC <= 0.285\ngini = 0.491\nsamples = 460
\nvalue = [56, 484, 196]\nnclass = b'),
Text(0.4166666666666667, 0.0833333333333333, 'gini = 0.503\nsamples = 198\n
value = [14, 102, 188]\nnclass = c'),
Text(0.45, 0.0833333333333333, 'gini = 0.208\nsamples = 262\nvalue = [42, 3

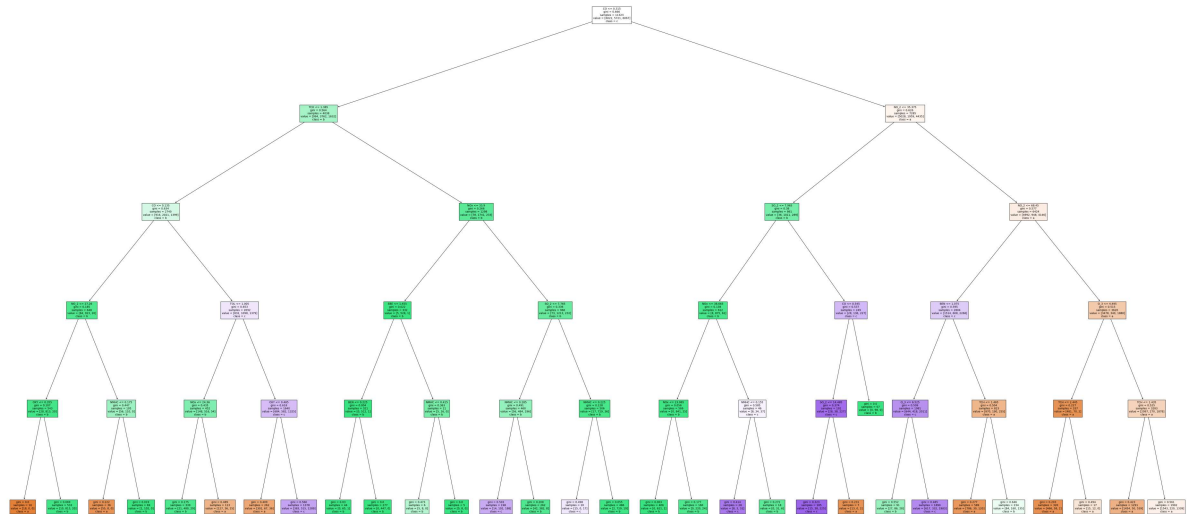
```

```

82, 8]\n\nclass = b'),
  Text(0.5, 0.25, 'NMHC <= 0.225\n\ngini = 0.128\n\nsamples = 506\n\nvalue = [17, 72
9, 36]\n\nclass = b'),
  Text(0.48333333333333334, 0.08333333333333333, 'gini = 0.498\n\nsamples = 20\n\nvalue = [15, 0, 17]\n\nclass = c'),
  Text(0.51666666666666667, 0.08333333333333333, 'gini = 0.055\n\nsamples = 486\n\nvalue = [2, 729, 19]\n\nclass = b'),
  Text(0.7625, 0.75, 'NO_2 <= 35.375\n\ngini = 0.626\n\nsamples = 7285\n\nvalue = [5
028, 1959, 4435]\n\nclass = a'),
  Text(0.6583333333333333, 0.58333333333333334, 'SO_2 <= 7.965\n\ngini = 0.38\n\nsam
ples = 861\n\nvalue = [36, 1011, 289]\n\nclass = b'),
  Text(0.6, 0.41666666666666667, 'NOx <= 38.665\n\ngini = 0.138\n\nsamples = 612\n\nv
alue = [8, 875, 62]\n\nclass = b'),
  Text(0.56666666666666667, 0.25, 'NOx <= 23.065\n\ngini = 0.056\n\nsamples = 566\n\nv
alue = [0, 841, 25]\n\nclass = b'),
  Text(0.55, 0.08333333333333333, 'gini = 0.003\n\nsamples = 406\n\nvalue = [0, 62
1, 1]\n\nclass = b'),
  Text(0.58333333333333334, 0.08333333333333333, 'gini = 0.177\n\nsamples = 160\n\nv
alue = [0, 220, 24]\n\nclass = b'),
  Text(0.6333333333333333, 0.25, 'NMHC <= 0.155\n\ngini = 0.585\n\nsamples = 46\n\nv
alue = [8, 34, 37]\n\nclass = c'),
  Text(0.61666666666666667, 0.08333333333333333, 'gini = 0.414\n\nsamples = 28\n\nv
alue = [8, 3, 31]\n\nclass = c'),
  Text(0.65, 0.08333333333333333, 'gini = 0.272\n\nsamples = 18\n\nvalue = [0, 31,
6]\n\nclass = b'),
  Text(0.71666666666666667, 0.41666666666666667, 'CO <= 0.545\n\ngini = 0.537\n\nsam
ples = 249\n\nvalue = [28, 136, 227]\n\nclass = c'),
  Text(0.7, 0.25, 'SO_2 <= 14.485\n\ngini = 0.374\n\nsamples = 192\n\nvalue = [28, 3
8, 227]\n\nclass = c'),
  Text(0.6833333333333333, 0.08333333333333333, 'gini = 0.323\n\nsamples = 185\n\nv
alue = [15, 38, 225]\n\nclass = c'),
  Text(0.71666666666666667, 0.08333333333333333, 'gini = 0.231\n\nsamples = 7\n\nva
lue = [13, 0, 2]\n\nclass = a'),
  Text(0.7333333333333333, 0.25, 'gini = 0.0\n\nsamples = 57\n\nvalue = [0, 98, 0]
\n\nclass = b'),
  Text(0.86666666666666667, 0.58333333333333334, 'NO_2 <= 68.45\n\ngini = 0.577\n\ns
amples = 6424\n\nvalue = [4992, 948, 4146]\n\nclass = a'),
  Text(0.8, 0.41666666666666667, 'BEN <= 1.075\n\ngini = 0.595\n\nsamples = 2804\n\nv
alue = [1514, 608, 2266]\n\nclass = c'),
  Text(0.76666666666666667, 0.25, 'O_3 <= 6.525\n\ngini = 0.509\n\nsamples = 1981\n\nv
alue = [644, 418, 2011]\n\nclass = c'),
  Text(0.75, 0.08333333333333333, 'gini = 0.552\n\nsamples = 91\n\nvalue = [27, 8
6, 28]\n\nclass = b'),
  Text(0.7833333333333333, 0.08333333333333333, 'gini = 0.485\n\nsamples = 1890
\n\nvalue = [617, 332, 1983]\n\nclass = c'),
  Text(0.83333333333333334, 0.25, 'TCH <= 1.465\n\ngini = 0.504\n\nsamples = 823\n\nv
alue = [870, 190, 255]\n\nclass = a'),
  Text(0.81666666666666667, 0.08333333333333333, 'gini = 0.277\n\nsamples = 589\n\nv
alue = [786, 30, 120]\n\nclass = a'),
  Text(0.85, 0.08333333333333333, 'gini = 0.646\n\nsamples = 234\n\nvalue = [84, 1
60, 135]\n\nclass = b'),
  Text(0.9333333333333333, 0.41666666666666667, 'O_3 <= 4.495\n\ngini = 0.515\n\nsa
mples = 3620\n\nvalue = [3478, 340, 1880]\n\nclass = a'),
  Text(0.9, 0.25, 'TCH <= 2.445\n\ngini = 0.227\n\nsamples = 337\n\nvalue = [481, 7
0, 2]\n\nclass = a'),
  Text(0.8833333333333333, 0.08333333333333333, 'gini = 0.203\n\nsamples = 320\n\nv
alue = [466, 58, 2]\n\nclass = a'),

```

```
Text(0.9166666666666666, 0.08333333333333333, 'gini = 0.494\nsamples = 17\nvalue = [15, 12, 0]\nclass = a'),
Text(0.9666666666666667, 0.25, 'TCH <= 1.435\ngini = 0.525\nsamples = 3283\nvalue = [2997, 270, 1878]\nclass = a'),
Text(0.95, 0.08333333333333333, 'gini = 0.423\nsamples = 1291\nvalue = [145, 4, 539]\nclass = a'),
Text(0.9833333333333333, 0.08333333333333333, 'gini = 0.561\nsamples = 1992\nvalue = [1543, 220, 1339]\nclass = a')]
```



Conclusion

```
In [34]: print("Linear Regression:",lr.score(x_test,y_test))
print("Ridge Regression:",rr.score(x_test,y_test))
print("Lasso Regression",la.score(x_test,y_test))
print("ElasticNet Regression:",en.score(x_test,y_test))
print("Logistic Regression:",logr.score(fs,target_vector))
print("Random Forest:",grid_search.best_score_)
```

```
Linear Regression: 0.13612242113139272
Ridge Regression: 0.13616589275057867
Lasso Regression 0.01265890622426824
ElasticNet Regression: 0.06602527642607048
Logistic Regression: 0.8146838030106512
Random Forest: 0.8248175182481752
```

Logistic Is Better!!!