In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import re
from sklearn.datasets import load_digits
```

In [2]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\ma
```

Out[2]:

	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
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
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN

210720 rows × 14 columns

In [3]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210720 entries, 0 to 210719
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210720 non-null  object 
 1   BEN      51511 non-null   float64
 2   CO       87097 non-null   float64
 3   EBE      51482 non-null   float64
 4   NMHC     30736 non-null   float64
 5   NO       209871 non-null  float64
 6   NO_2     209872 non-null  float64
 7   O_3      122339 non-null  float64
 8   PM10     104838 non-null  float64
 9   PM25     52164 non-null   float64
 10  SO_2     87333 non-null   float64
 11  TCH      30736 non-null   float64
 12  TOL      51373 non-null   float64
 13  station  210720 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.5+ MB
```

In [4]:
```
b=a.fillna(value=188)
```

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH
0	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	7.0	18.0	188.0	188.0	188.0	2.0	188.00
1	2012-09-01 01:00:00	0.3	0.3	0.7	188.00	3.0	18.0	55.0	10.0	9.0	1.0	188.00
2	2012-09-01 01:00:00	0.4	188.0	0.7	188.00	2.0	10.0	188.0	188.0	188.0	188.0	188.00
3	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	1.0	6.0	50.0	188.0	188.0	188.0	188.00
4	2012-09-01 01:00:00	188.0	188.0	188.0	188.00	1.0	13.0	54.0	188.0	188.0	3.0	188.00
...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	188.0	0.6	188.0	188.00	37.0	84.0	14.0	188.0	188.0	188.0	188.00
210716	2012-03-01 00:00:00	188.0	0.4	188.0	188.00	5.0	76.0	188.0	17.0	188.0	7.0	188.00
210717	2012-03-01 00:00:00	188.0	188.0	188.0	0.34	3.0	41.0	24.0	188.0	188.0	188.0	1.34
210718	2012-03-01 00:00:00	188.0	188.0	188.0	188.00	2.0	44.0	36.0	188.0	188.0	188.0	188.00
210719	2012-03-01 00:00:00	188.0	188.0	188.0	188.00	2.0	56.0	40.0	18.0	188.0	188.0	188.00

210720 rows × 14 columns

In [5]:

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25
',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
c=b.head(11)
```

Out[6]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	7.0	18.0	188.0	188.0	188.0	2.0	188.00	188.
1	2012-09-01 01:00:00	0.3	0.3	0.7	188.00	3.0	18.0	55.0	10.0	9.0	1.0	188.00	2.4
2	2012-09-01 01:00:00	0.4	188.0	0.7	188.00	2.0	10.0	188.0	188.0	188.0	188.0	188.00	1.5
3	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	1.0	6.0	50.0	188.0	188.0	188.0	188.00	188.
4	2012-09-01 01:00:00	188.0	188.0	188.0	188.00	1.0	13.0	54.0	188.0	188.0	3.0	188.00	188.
5	2012-09-01 01:00:00	0.2	0.2	1.0	188.00	1.0	9.0	57.0	14.0	188.0	1.0	188.00	0.2
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6
7	2012-09-01 01:00:00	188.0	188.0	188.0	0.11	1.0	2.0	65.0	188.0	188.0	188.0	1.18	188.
8	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	6.0	14.0	57.0	188.0	188.0	2.0	188.00	188.
9	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	1.0	7.0	188.0	13.0	188.0	1.0	188.00	188.
10	2012-09-01 01:00:00	0.2	188.0	0.7	188.00	3.0	13.0	188.0	12.0	6.0	1.0	188.00	0.8

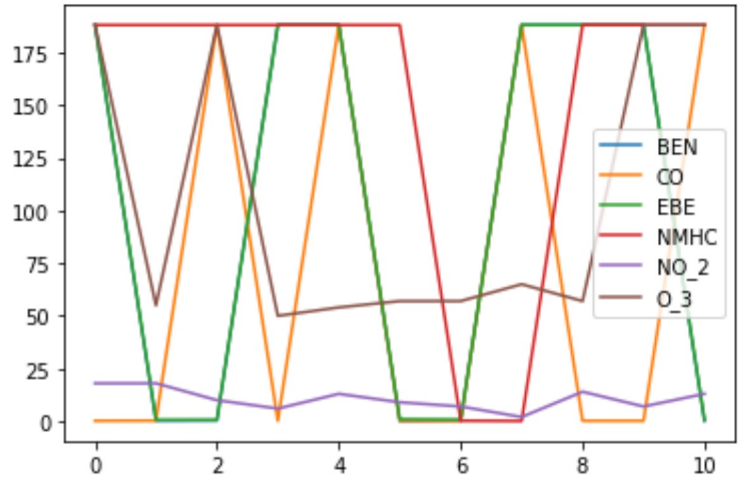
In [7]:
```
d=c[['BEN','CO','EBE','NMHC','NO_2','O_3']]
```

Out[7]:

	BEN	CO	EBE	NMHC	NO_2	O_3
0	188.0	0.2	188.0	188.00	18.0	188.0
1	0.3	0.3	0.7	188.00	18.0	55.0
2	0.4	188.0	0.7	188.00	10.0	188.0
3	188.0	0.2	188.0	188.00	6.0	50.0
4	188.0	188.0	188.0	188.00	13.0	54.0
5	0.2	0.2	1.0	188.00	9.0	57.0
6	0.4	0.2	0.8	0.24	7.0	57.0
7	188.0	188.0	188.0	0.11	2.0	65.0
8	188.0	0.2	188.0	188.00	14.0	57.0
9	188.0	0.2	188.0	188.00	7.0	188.0
10	0.2	188.0	0.7	188.00	13.0	188.0

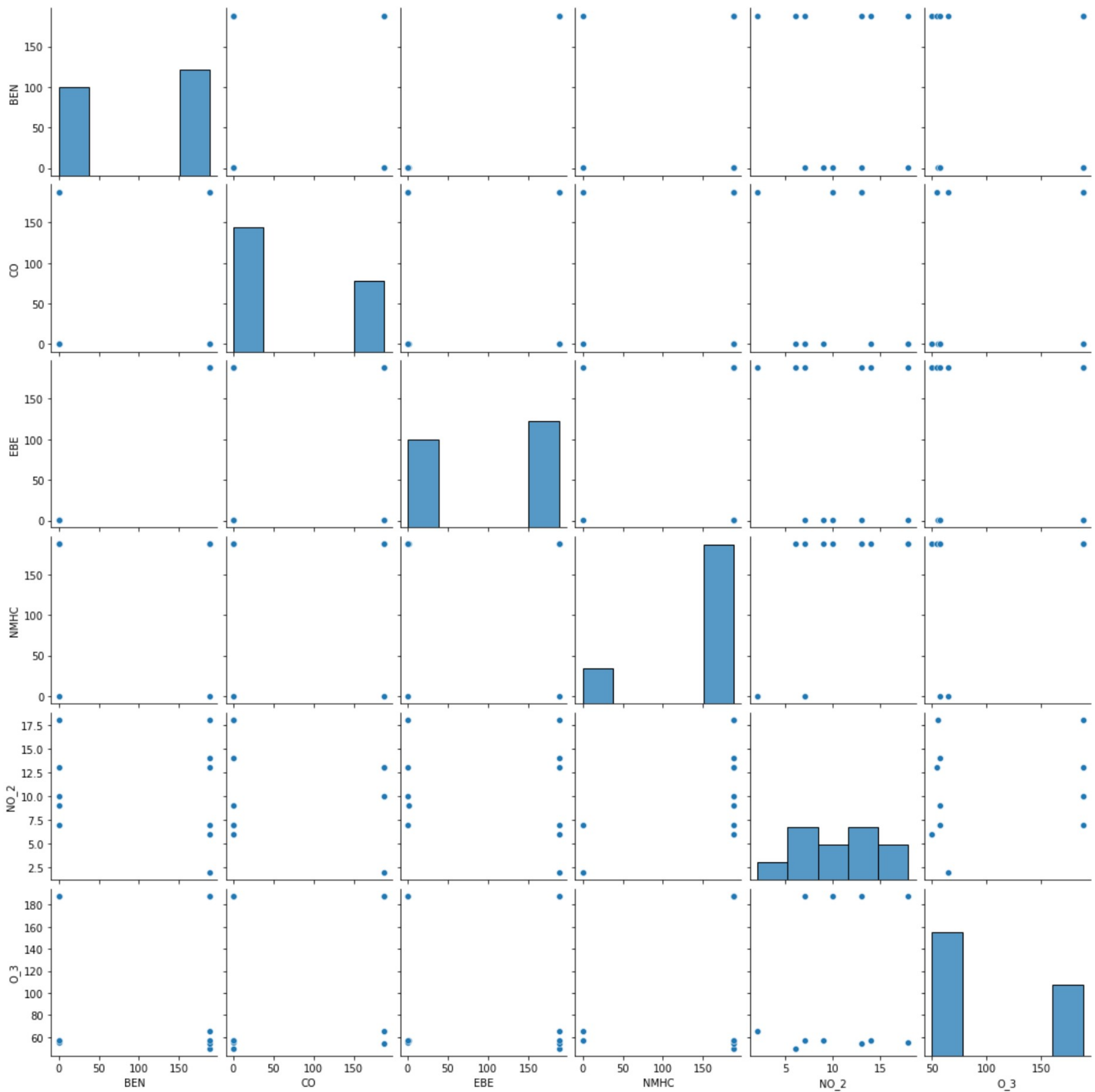
In [8]:
```
d.plot.line()
```

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x1b8df750610>



In [10]:
```python
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
```

In [11]:
```python
from sklearn.model_selection import train_test_split
```

In [12]:
```python
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

Out[12]: LinearRegression()

In [13]:

2.2431834167946363e-11

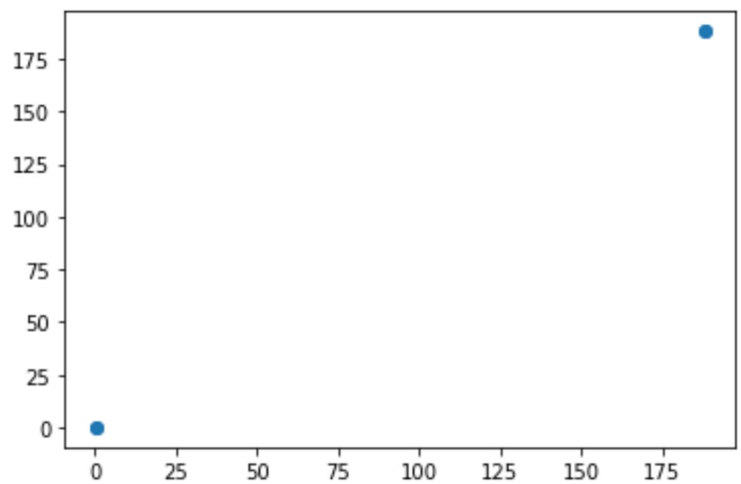
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	5.588715e-11
CO	1.000000e+00
EBE	-5.600700e-11
NMHC	4.406879e-16
NO_2	-2.105775e-15

In [15]:
```
prediction=lr.predict(x_test)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1b8e27d0310>



In [16]:

1.0

In [17]:

In [18]:
```
rr=Ridge(alpha=10)
```

Out[18]: Ridge(alpha=10)

In [19]:

Out[19]: 0.9999995643541453

In [20]:
```
la=Lasso(alpha=10)
```

Out[20]: Lasso(alpha=10)

In [21]:

Out[21]: 0.9999977145129909

In [22]:
```
a1=b.head(5000)
```

Out[22]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	T
0	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	7.0	18.0	188.0	188.0	188.0	2.0	188.00	18
1	2012-09-01 01:00:00	0.3	0.3	0.7	188.00	3.0	18.0	55.0	10.0	9.0	1.0	188.00	
2	2012-09-01 01:00:00	0.4	188.0	0.7	188.00	2.0	10.0	188.0	188.0	188.0	188.0	188.00	
3	2012-09-01 01:00:00	188.0	0.2	188.0	188.00	1.0	6.0	50.0	188.0	188.0	188.0	188.00	18
4	2012-09-01 01:00:00	188.0	188.0	188.0	188.00	1.0	13.0	54.0	188.0	188.0	3.0	188.00	18
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	2012-09-09 17:00:00	188.0	0.2	188.0	188.00	2.0	8.0	96.0	188.0	188.0	188.0	188.00	18
4996	2012-09-09 17:00:00	188.0	188.0	188.0	188.00	2.0	5.0	99.0	188.0	188.0	3.0	188.00	18
4997	2012-09-09 17:00:00	0.2	0.2	1.0	188.00	1.0	5.0	93.0	27.0	188.0	1.0	188.00	
4998	2012-09-09 17:00:00	0.5	0.2	1.1	0.22	1.0	3.0	97.0	27.0	12.0	3.0	1.32	
4999	2012-09-09 17:00:00	188.0	188.0	188.0	0.11	1.0	4.0	110.0	188.0	188.0	188.0	1.18	18

5000 rows × 14 columns

In [23]:
```
e=a1[['BEN', 'CO', 'EBE','NMHC', 'NO_2','O_3',
```

In [24]:
```
f=e.iloc[:,0:14]
```

In [25]:
```
```

In [26]:
```
logr=LogisticRegression(max_iter=10000)
```

Out[26]: LogisticRegression(max_iter=10000)

In [27]:
```
from sklearn.model_selection import train_test_split
```

In [28]:
```
```

In [29]:
```
prediction=logr.predict(i)
```

```
[28079050]
```

In [30]:

Out[30]:
```
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 [31]:

Out[31]:
```
0.0
```

In [32]:

Out[32]:
```
0.0
```

In [33]:

Out[33]:
```
0.9573333333333334
```

In [34]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_d
escent.py:530: ConvergenceWarning: Objective did not converge. You might want
to increase the number of iterations. Duality gap: 5.37854093374016, toleranc
e: 5.037333428315741
  model = cd_fast.enet_coordinate_descent(
```

Out[34]:
```
ElasticNet()
```

In [35]:

```
[ 2.18053242e-01  9.99868788e-01 -2.18377624e-01 -1.22495536e-05
  0.00000000e+00]
```

In [36]:

```
0.07726383756725852
```

In [37]:
```
prediction=en.predict(x_test)
```

```
0.9999996066476035
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

Out[38]:
```
RandomForestClassifier()
```

In [39]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [40]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
```

Out[40]:
```
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')
```

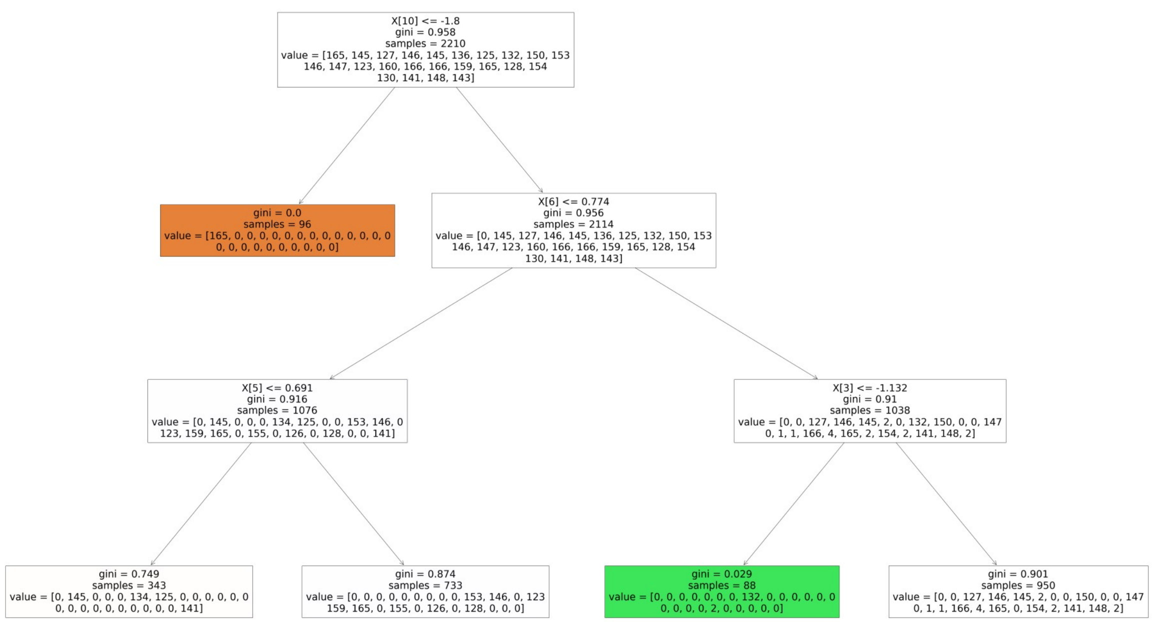
In [41]:

Out[41]:
```
0.9982857142857142
```

In [42]:

In [43]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
```

Out[43]:
```
[Text(1674.0, 2378.25, 'X[10] <= -1.8\ngini = 0.958\nsamples = 2210\nvalue =
[165, 145, 127, 146, 145, 136, 125, 132, 150, 153\n146, 147, 123, 160, 166, 1
66, 159, 165, 128, 154\n130, 141, 148, 143]'),
 Text(1116.0, 1698.75, 'gini = 0.0\nsamples = 96\nvalue = [165, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2232.0, 1698.75, 'X[6] <= 0.774\ngini = 0.956\nsamples = 2114\nvalue =
[0, 145, 127, 146, 145, 136, 125, 132, 150, 153\n146, 147, 123, 160, 166, 16
6, 159, 165, 128, 154\n130, 141, 148, 143]'),
 Text(1116.0, 1019.25, 'X[5] <= 0.691\ngini = 0.916\nsamples = 1076\nvalue =
[0, 145, 0, 0, 0, 134, 125, 0, 0, 153, 146, 0\n123, 159, 165, 0, 155, 0, 126,
0, 128, 0, 0, 141]'),
 Text(558.0, 339.75, 'gini = 0.749\nsamples = 343\nvalue = [0, 145, 0, 0, 0,
134, 125, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 141]'),
 Text(1674.0, 339.75, 'gini = 0.874\nsamples = 733\nvalue = [0, 0, 0, 0, 0,
0, 0, 0, 0, 153, 146, 0, 123\n159, 165, 0, 155, 0, 126, 0, 128, 0, 0, 0]'),
 Text(3348.0, 1019.25, 'X[3] <= -1.132\ngini = 0.91\nsamples = 1038\nvalue =
[0, 0, 127, 146, 145, 2, 0, 132, 150, 0, 0, 147\n0, 1, 1, 166, 4, 165, 2, 15
4, 2, 141, 148, 2]'),
 Text(2790.0, 339.75, 'gini = 0.029\nsamples = 88\nvalue = [0, 0, 0, 0, 0, 0,
0, 132, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2, 0, 0, 0, 0, 0]'),
 Text(3906.0, 339.75, 'gini = 0.901\nsamples = 950\nvalue = [0, 0, 127, 146,
145, 2, 0, 0, 150, 0, 0, 147\n0, 1, 1, 166, 4, 165, 0, 154, 2, 141, 148,
2]')]
```



From this observation I had observe that the ELASTICNET is a highest accuracy of 0.9999996066476035

In [ ]: