In [124]:
```
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
from sklearn.model_selection import train_test_split
```

In [125]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2004
a
```

Out[125]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.8
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	
...	...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.8
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.0
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001	
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.9

245496 rows × 17 columns

In [126]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      65158 non-null   float64
 2   CO       226043 non-null  float64
 3   EBE      56781 non-null   float64
 4   MXY      39867 non-null   float64
 5   NMHC     107630 non-null  float64
 6   NO_2     243280 non-null  float64
 7   NOx      243283 non-null  float64
 8   OXY      39882 non-null   float64
 9   O_3      233811 non-null  float64
 10  PM10     234655 non-null  float64
 11  PM25     58145 non-null   float64
 12  PXY      39891 non-null   float64
 13  SO_2     243402 non-null  float64
 14  TCH      107650 non-null  float64
 15  TOL      64914 non-null   float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ MB
```

In [133]:
```
b=a.fillna(value=102)
b
```

Out[133]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	F
0	2004-08-01 01:00:00	102.00	0.66	102.00	102.00	102.00	89.550003	118.900002	102.00	40.020000	39.99
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.95
2	2004-08-01 01:00:00	102.00	1.02	102.00	102.00	102.00	93.389999	138.600006	102.00	20.860001	49.48
3	2004-08-01 01:00:00	102.00	0.53	102.00	102.00	102.00	87.290001	105.000000	102.00	36.730000	31.07
4	2004-08-01 01:00:00	102.00	0.17	102.00	102.00	102.00	34.910000	35.349998	102.00	86.269997	54.08
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.90
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	102.00	97.139999	146.899994	2.34	7.740000	37.68
245493	2004-06-01 00:00:00	102.00	102.00	102.00	102.00	0.13	102.699997	132.600006	102.00	17.809999	22.84
245494	2004-06-01 00:00:00	102.00	102.00	102.00	102.00	0.09	82.599998	102.599998	102.00	102.000000	45.63
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.38

245496 rows × 17 columns

In [134]:
```
b.columns
```

Out[134]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [135]:
```
c=b.head(10)
c
```

Out[135]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2004-08-01 01:00:00	102.00	0.66	102.00	102.00	102.00	89.550003	118.900002	102.00	40.020000	39.990002	2
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	10
2	2004-08-01 01:00:00	102.00	1.02	102.00	102.00	102.00	93.389999	138.600006	102.00	20.860001	49.480000	10
3	2004-08-01 01:00:00	102.00	0.53	102.00	102.00	102.00	87.290001	105.000000	102.00	36.730000	31.070000	10
4	2004-08-01 01:00:00	102.00	0.17	102.00	102.00	102.00	34.910000	35.349998	102.00	86.269997	54.080002	10
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110001	3
6	2004-08-01 01:00:00	102.00	0.43	102.00	102.00	0.17	54.270000	64.279999	102.00	66.589996	54.270000	10
7	2004-08-01 01:00:00	1.41	0.47	2.35	102.00	0.02	71.730003	87.519997	102.00	53.270000	45.180000	10
8	2004-08-01 01:00:00	102.00	1.28	102.00	102.00	102.00	147.699997	202.500000	102.00	10.280000	52.430000	10
9	2004-08-01 01:00:00	102.00	0.43	102.00	102.00	0.27	54.290001	68.099998	102.00	66.709999	54.700001	10

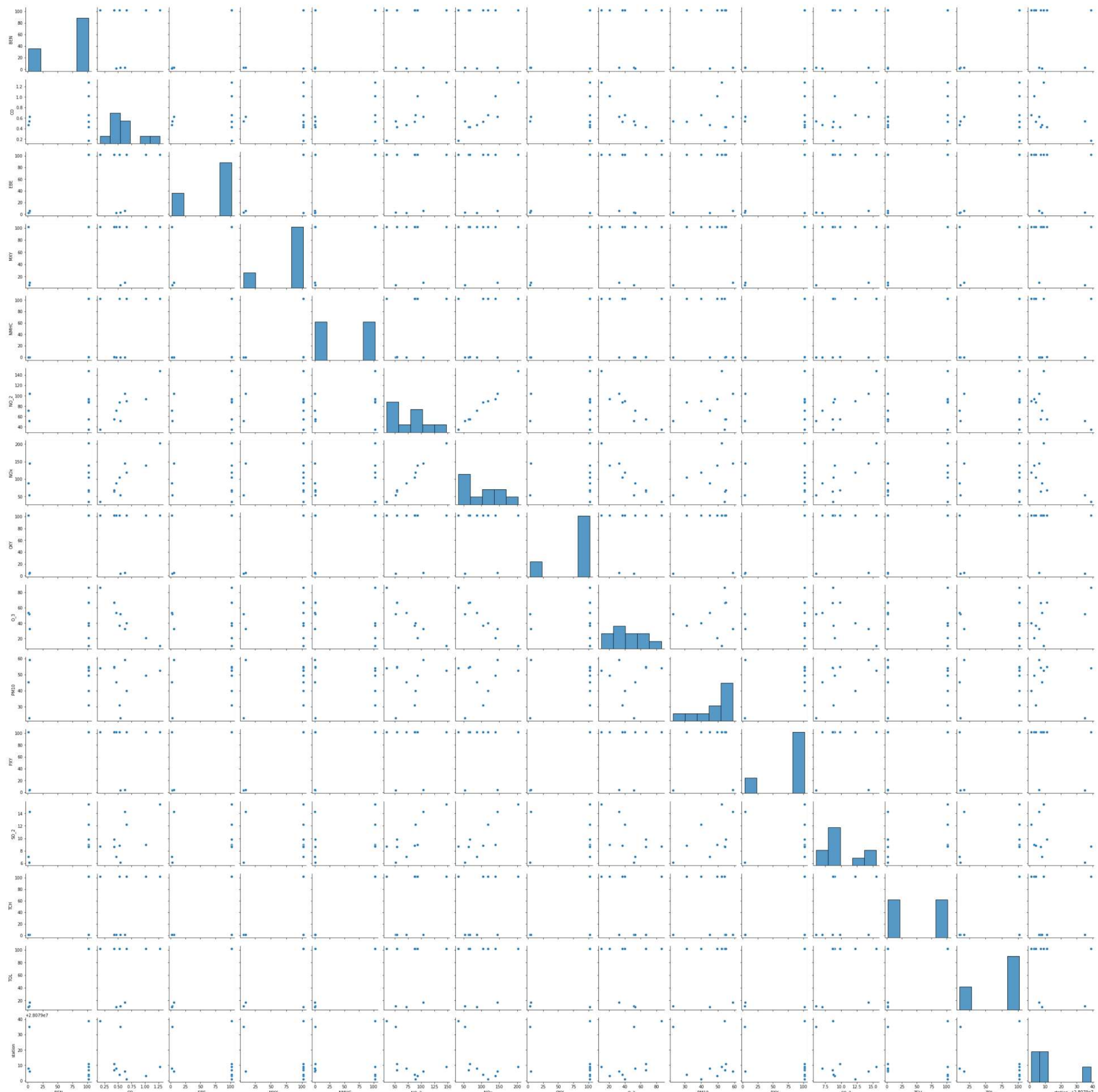
In [136]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[136]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_
0	102.00	0.66	102.00	102.00	102.00	89.550003	118.900002	102.00	40.020000	39.990002	102.00	12.2
1	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	3.38	6.1
2	102.00	1.02	102.00	102.00	102.00	93.389999	138.600006	102.00	20.860001	49.480000	102.00	8.9
3	102.00	0.53	102.00	102.00	102.00	87.290001	105.000000	102.00	36.730000	31.070000	102.00	8.8
4	102.00	0.17	102.00	102.00	102.00	34.910000	35.349998	102.00	86.269997	54.080002	102.00	8.7
5	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110001	4.16	14.2
6	102.00	0.43	102.00	102.00	0.17	54.270000	64.279999	102.00	66.589996	54.270000	102.00	8.6
7	1.41	0.47	2.35	102.00	0.02	71.730003	87.519997	102.00	53.270000	45.180000	102.00	7.0
8	102.00	1.28	102.00	102.00	102.00	147.699997	202.500000	102.00	10.280000	52.430000	102.00	15.4
9	102.00	0.43	102.00	102.00	0.27	54.290001	68.099998	102.00	66.709999	54.700001	102.00	9.8

In [137]:
```
sns.pairplot(d)
```

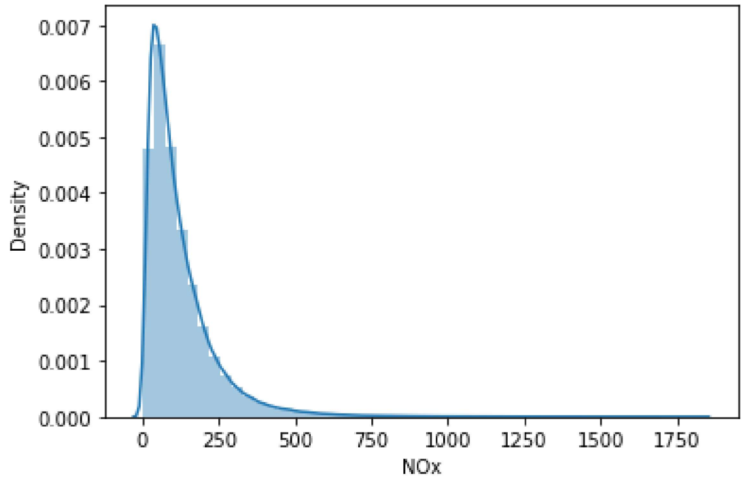
Out[137]: <seaborn.axisgrid.PairGrid at 0x1b6568facd0>



In [138]:
```
sns.distplot(a['NOx'])
```

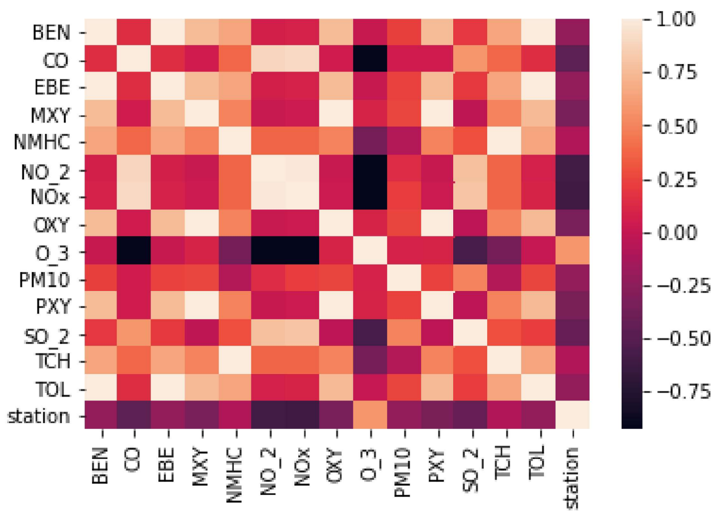
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[138]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [139]:
```
sns.heatmap(d.corr())
```

Out[139]: <AxesSubplot:>



In [140]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [141]:
```
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)
```

In [142]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[142]: LinearRegression()

In [143]:
```
print(lr.intercept_)
```

1.3901233022162884

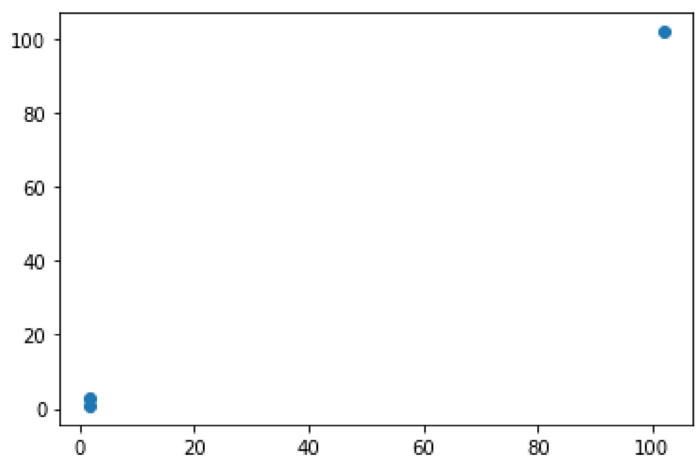
In [144]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[144]:

	Co-efficient
BEN	-2.247606e-03
CO	2.445718e+00
EBE	-2.226602e-03
MXY	-1.720846e-15
NMHC	9.883873e-01
NO_2	3.875085e-02
NOx	-4.239689e-02
OXY	0.000000e+00

In [145]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[145]: <matplotlib.collections.PathCollection at 0x1b66dd24be0>



In [146]:
```
print(lr.score(x_test,y_test))
```

0.9997659171680979

In [147]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [148]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[148]: Ridge(alpha=10)

In [149]:
```
rr.score(x_test,y_test)
```

Out[149]: 0.9999918783284977

In [150]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[150]:
```
Lasso(alpha=10)
```

In [151]:
```
la.score(x_test,y_test)
```
Out[151]:
```
0.9999852321555398
```

In [152]:
```
a1=b.head(7000)
a1
```
Out[152]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2004-08-01 01:00:00	102.00	0.66	102.00	102.00	102.00	89.550003	118.900002	102.00	40.020000	39.990002
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001
2	2004-08-01 01:00:00	102.00	1.02	102.00	102.00	102.00	93.389999	138.600006	102.00	20.860001	49.480000
3	2004-08-01 01:00:00	102.00	0.53	102.00	102.00	102.00	87.290001	105.000000	102.00	36.730000	31.070000
4	2004-08-01 01:00:00	102.00	0.17	102.00	102.00	102.00	34.910000	35.349998	102.00	86.269997	54.080002
...	...	...	...	...	...	...	...	...	...	...	...
6995	2004-08-11 11:00:00	102.00	0.35	102.00	102.00	102.00	38.959999	60.660000	102.00	28.830000	30.510000
6996	2004-08-11 11:00:00	102.00	0.44	102.00	102.00	102.00	48.400002	99.690002	102.00	24.700001	38.259998
6997	2004-08-11 11:00:00	0.20	0.20	102.00	102.00	102.00	32.580002	50.669998	102.00	6.940000	19.370001
6998	2004-08-11 11:00:00	102.00	0.38	102.00	102.00	0.10	54.660000	83.279999	102.00	23.760000	36.930000
6999	2004-08-11 11:00:00	0.66	0.20	1.03	1.88	0.02	16.709999	22.690001	1.05	50.040001	24.410000

7000 rows × 17 columns

In [153]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [154]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [155]:
```
h=StandardScaler().fit_transform(f)
```

In [156]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[156]:
```
LogisticRegression(max_iter=10000)
```

In [157]:
```
from sklearn.model_selection import train_test_split
h_train,h_test,g_train,g_test=train_test_split(h,g,test_size=0.3)
```

In [158]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [159]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079004]
```

In [160]:
```
logr.classes_
```

Out[160]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079026, 28079027, 28079035, 28079036,
       28079038, 28079039, 28079040, 28079099], dtype=int64)
```

In [161]:
```
logr.predict_proba(i)[0][0]
```

Out[161]:
```
1.1422813089665742e-98
```

In [162]:
```
logr.predict_proba(i)[0][1]
```

Out[162]:
```
3.829242017369329e-300
```

In [163]:
```
logr.score(h_test,g_test)
```

Out[163]:
```
0.5804761904761905
```

In [164]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[164]:
```
ElasticNet()
```

In [165]:
```
print(en.coef_)
```

```
[-2.18878465e-03  0.00000000e+00 -0.00000000e+00  0.00000000e+00
  9.86667759e-01  0.00000000e+00  6.09039771e-05  0.00000000e+00]
```

In [166]:
```
print(en.intercept_)
```

```
1.5579420704456908
```

In [167]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999950770974989
```

In [168]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[168]:
```
RandomForestClassifier()
```

In [169]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [170]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(h_train,g_train)
```

Out[170]:
```
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 [171]:
```
grid_search.best_score_
```

Out[171]:
```
0.6204081632653061
```

In [172]:
```
rfc_best=grid_search.best_estimator_
```

In [173]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
 Text(3529.6744186046512, 226.5, 'gini = 0.286\nsamples = 66\nvalue = [0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 18, 86, 0, 0, 0, 0, 0, 0]'),
 Text(4048.7441860465115, 1132.5, 'X[8] <= 0.116\ngini = 0.92\nsamples = 1208\nvalu
e = [149, 75, 102, 6, 0, 0, 184, 0, 178, 194, 0, 203\n170, 3, 144, 152, 9, 0, 0, 1
4, 0, 0, 0, 40\n164, 149, 18, 0]'),
 Text(3841.1162790697676, 679.5, 'X[9] <= -0.46\ngini = 0.913\nsamples = 683\nvalue
= [138, 56, 49, 0, 0, 0, 121, 0, 127, 118, 0, 93\n67, 3, 76, 81, 9, 0, 0, 0, 0, 0,
0, 19, 69\n60, 6, 0]'),
 Text(3737.3023255813955, 226.5, 'gini = 0.898\nsamples = 219\nvalue = [22, 10, 18,
0, 0, 0, 57, 0, 42, 41, 0, 42, 15\n0, 18, 46, 0, 0, 0, 0, 0, 0, 0, 0, 32, 16, 2\n
0]'),
 Text(3944.9302325581393, 226.5, 'gini = 0.911\nsamples = 464\nvalue = [116, 46, 3
1, 0, 0, 0, 64, 0, 85, 77, 0, 51, 52\n3, 58, 35, 9, 0, 0, 0, 0, 0, 0, 19, 37, 44\n
4, 0]'),
 Text(4256.372093023256, 679.5, 'X[1] <= -0.305\ngini = 0.911\nsamples = 525\nvalue
= [11, 19, 53, 6, 0, 0, 63, 0, 51, 76, 0, 110\n103, 0, 68, 71, 0, 0, 0, 14, 0, 0,
0, 21, 95\n89, 12, 0]'),
 Text(4152.558139534884, 226.5, 'gini = 0.724\nsamples = 124\nvalue = [0, 0, 2, 0,
0, 0, 0, 0, 18, 0, 0, 83, 0, 0\n0, 25, 0, 0, 0, 0, 0, 0, 0, 0, 12, 60, 5, 0]'),
```

Conclusion: from this data set i observed that the ELASTICNET has the highest accuracy of 0.9999950770974989

In [ ]: