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
from sklearn.model_selection import train_test_split
```

Madrid 2001

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2001.csv")
a
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	10
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	10
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	10
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	6
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	7
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	4
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	2
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	4
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	3
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	3

217872 rows × 16 columns

In [3]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [4]:

```
b=a.fillna(value=87)
b
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2001-08-01 01:00:00	87.00	0.37	87.00	87.00	87.00	58.400002	87.150002	87.00	34.529999
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000
2	2001-08-01 01:00:00	87.00	0.28	87.00	87.00	87.00	50.660000	61.380001	87.00	46.310001
3	2001-08-01 01:00:00	87.00	0.47	87.00	87.00	87.00	69.790001	73.449997	87.00	40.650002
4	2001-08-01 01:00:00	87.00	0.39	87.00	87.00	87.00	22.830000	24.799999	87.00	66.309998
...	...	...	...	...	...	...	...	...	...	...
217867	2001-04-01 00:00:00	10.45	1.81	87.00	87.00	87.00	73.000000	264.399994	87.00	5.200000
217868	2001-04-01 00:00:00	5.20	0.69	4.56	87.00	0.13	71.080002	129.300003	87.00	13.460000
217869	2001-04-01 00:00:00	0.49	1.09	87.00	1.00	0.19	76.279999	128.399994	0.35	5.020000
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	87.00	80.019997	197.000000	2.58	5.840000
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000

217872 rows × 16 columns

In [5]:

```
b.columns
```

Out[5]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:

```
c=b.head(10)
c
```

Out[6]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	I
0	2001-08-01 01:00:00	87.00	0.37	87.00	87.00	87.00	58.400002	87.150002	87.00	34.529999	105.0
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	87.00	0.28	87.00	87.00	87.00	50.660000	61.380001	87.00	46.310001	100.0
3	2001-08-01 01:00:00	87.00	0.47	87.00	87.00	87.00	69.790001	73.449997	87.00	40.650002	69.7
4	2001-08-01 01:00:00	87.00	0.39	87.00	87.00	87.00	22.830000	24.799999	87.00	66.309998	75.1
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.94	0.05	66.260002	118.099998	3.15	33.500000	122.6
6	2001-08-01 01:00:00	87.00	0.28	87.00	87.00	87.00	35.799999	39.590000	87.00	68.250000	124.9
7	2001-08-01 01:00:00	87.00	0.67	87.00	87.00	87.00	74.830002	112.000000	87.00	26.410000	113.0
8	2001-08-01 01:00:00	87.00	0.41	87.00	87.00	87.00	33.209999	37.299999	87.00	62.299999	125.3
9	2001-08-01 01:00:00	87.00	0.17	87.00	87.00	0.13	24.129999	36.970001	87.00	46.200001	95.5

In [7]:

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

Out[7]:

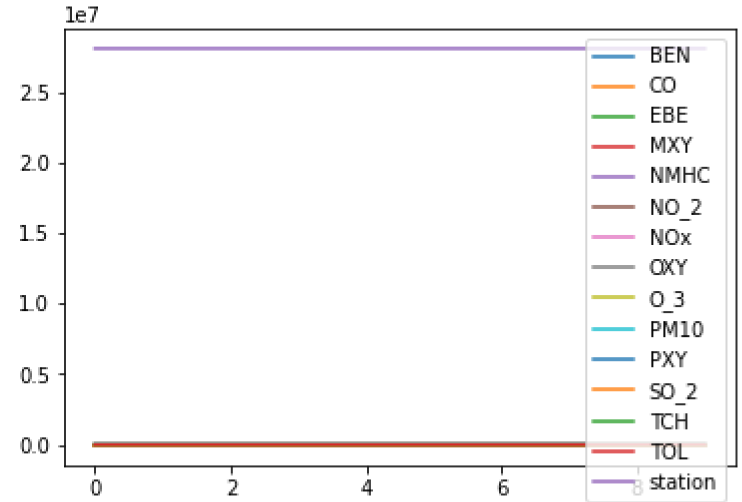
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PX
0	87.00	0.37	87.00	87.00	87.00	58.400002	87.150002	87.00	34.529999	105.000000	87.
1	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.
2	87.00	0.28	87.00	87.00	87.00	50.660000	61.380001	87.00	46.310001	100.099998	87.
3	87.00	0.47	87.00	87.00	87.00	69.790001	73.449997	87.00	40.650002	69.779999	87.
4	87.00	0.39	87.00	87.00	87.00	22.830000	24.799999	87.00	66.309998	75.180000	87.
5	2.11	0.63	2.48	5.94	0.05	66.260002	118.099998	3.15	33.500000	122.699997	2.
6	87.00	0.28	87.00	87.00	87.00	35.799999	39.590000	87.00	68.250000	124.900002	87.
7	87.00	0.67	87.00	87.00	87.00	74.830002	112.000000	87.00	26.410000	113.000000	87.
8	87.00	0.41	87.00	87.00	87.00	33.209999	37.299999	87.00	62.299999	125.300003	87.
9	87.00	0.17	87.00	87.00	0.13	24.129999	36.970001	87.00	46.200001	95.589996	87.

In [8]:

```
d.plot.line()
```

Out[8]:

<AxesSubplot:>

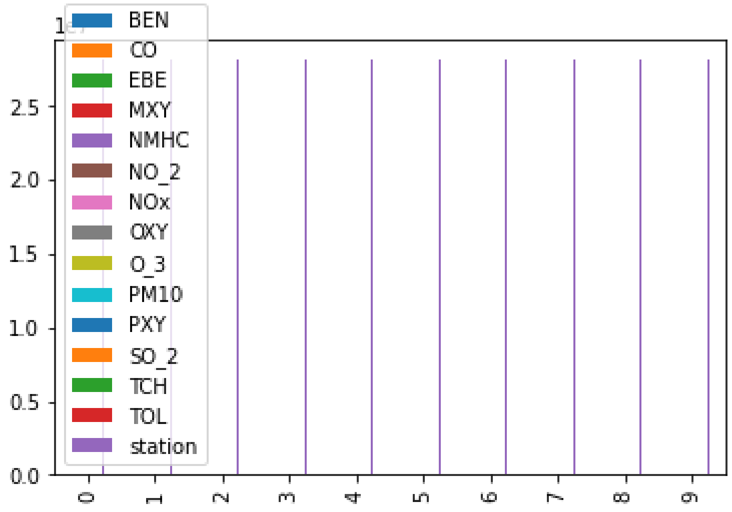


In [9]:

```
d.plot.bar()
```

Out[9]:

<AxesSubplot:>

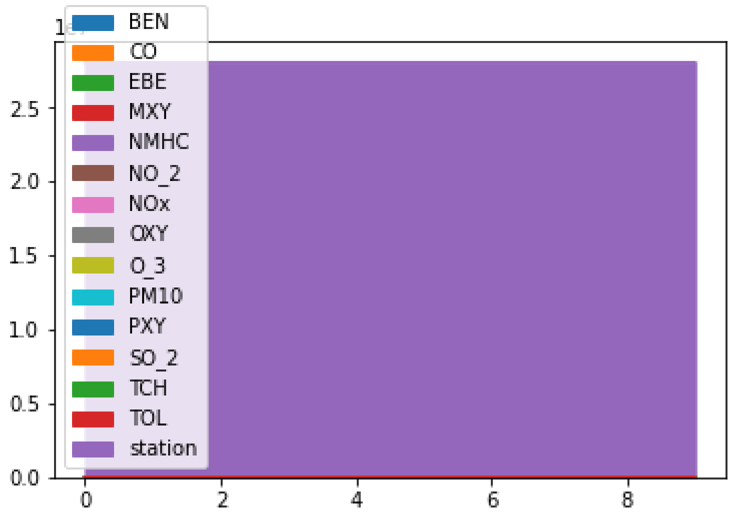


In [10]:

```
d.plot.area()
```

Out[10]:

<AxesSubplot:>

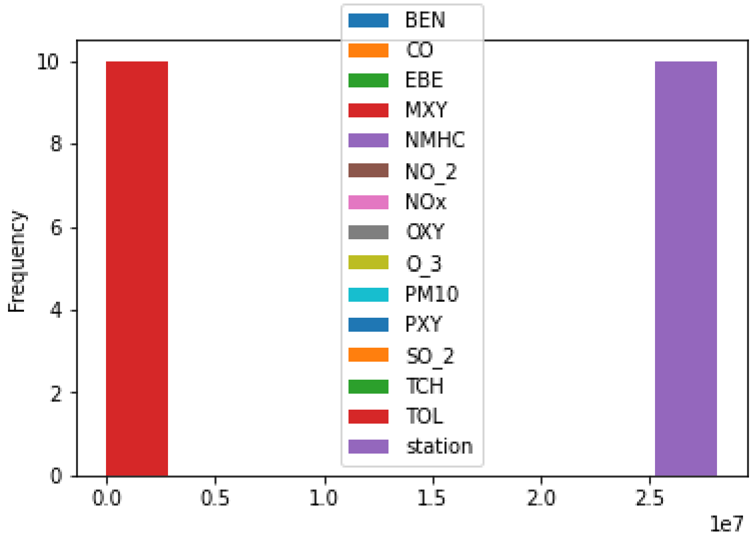


In [11]:

```
d.plot.hist()
```

Out[11]:

```
<AxesSubplot:ylabel='Frequency'>
```

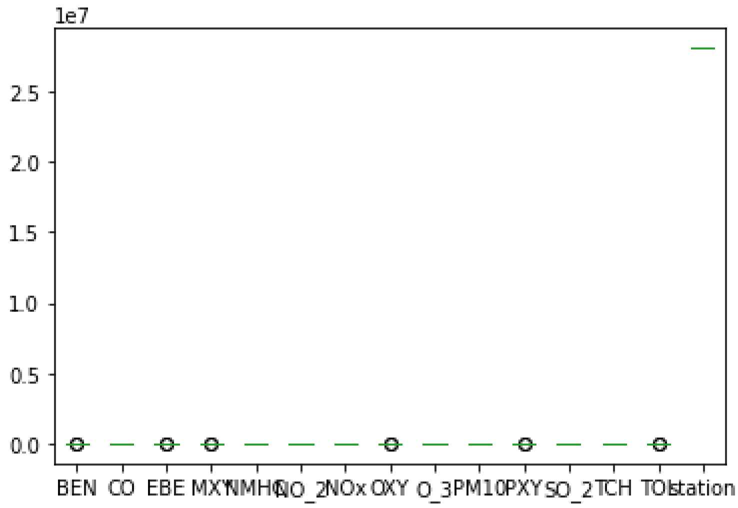


In [12]:

```
d.plot.box()
```

Out[12]:

```
<AxesSubplot:>
```

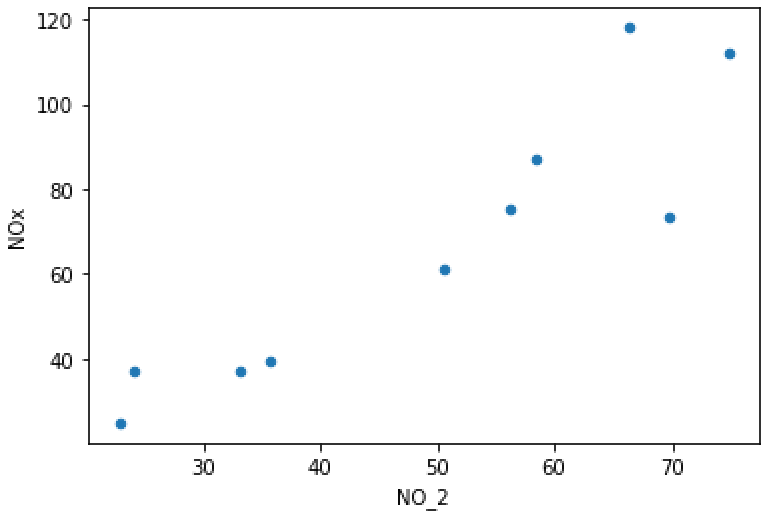


In [13]:

```
d.plot.scatter(x='NO_2',y='NOx')
```

Out[13]:

```
<AxesSubplot:xlabel='NO_2', ylabel='NOx'>
```

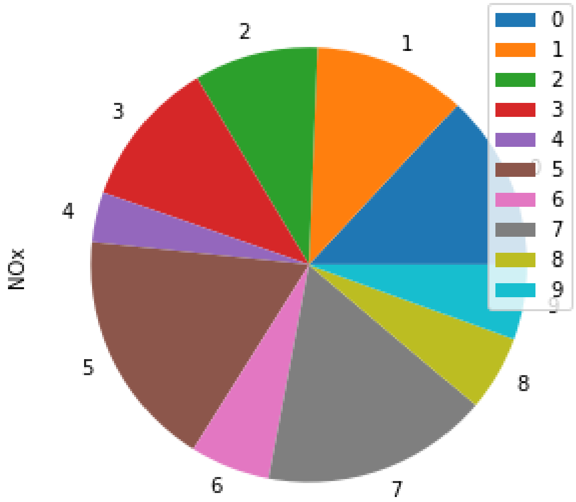


In [14]:

```
d.plot.pie(y='NOx',figsize=(5,5))
```

Out[14]:

```
<AxesSubplot:ylabel='NOx'>
```

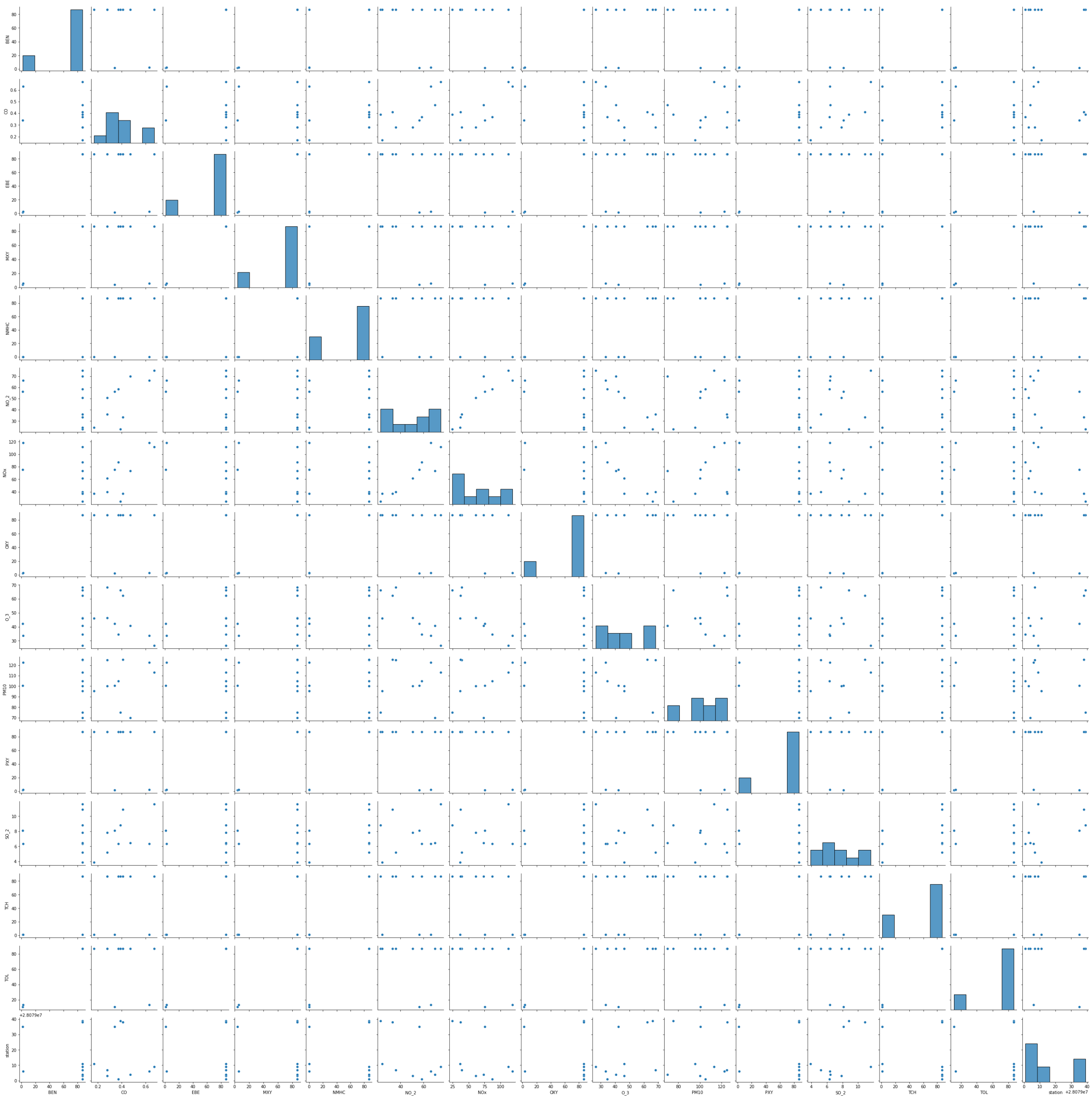


In [15]:

```
sns.pairplot(d)
```

Out[15]:

<seaborn.axisgrid.PairGrid at 0x1cbe59fb0d0>



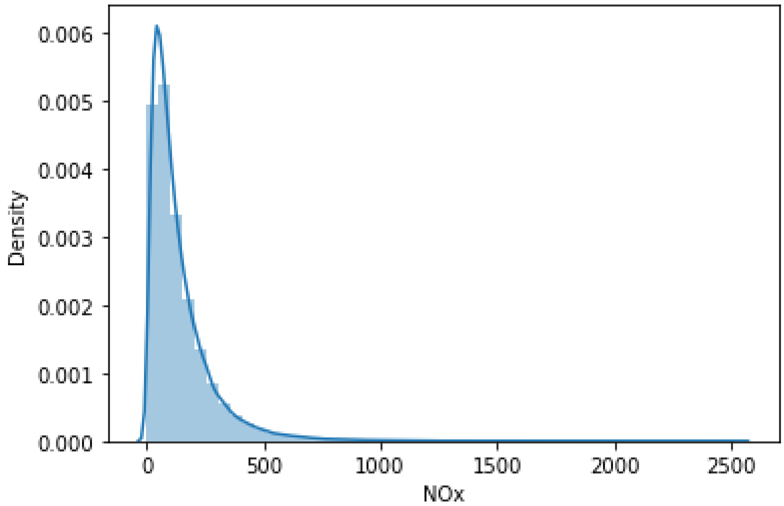
In [16]:

```
sns.distplot(a['NOx'])
```

```
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[16]:

<AxesSubplot:xlabel='NOx', ylabel='Density'>

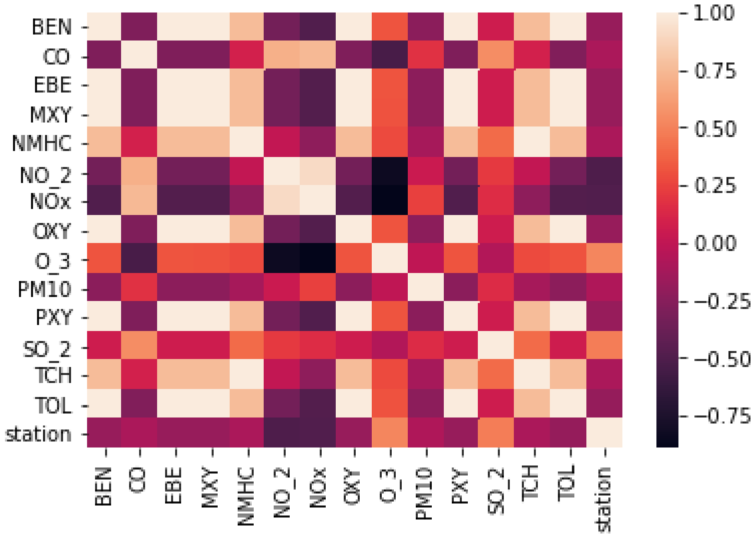


In [17]:

```
sns.heatmap(d.corr())
```

Out[17]:

<AxesSubplot:>



In [18]:

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

In [19]:

```
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 [20]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[20]:

```
LinearRegression()
```

In [21]:

```
print(lr.intercept_)
```

```
1.311960462608127
```

In [22]:

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

Out[22]:

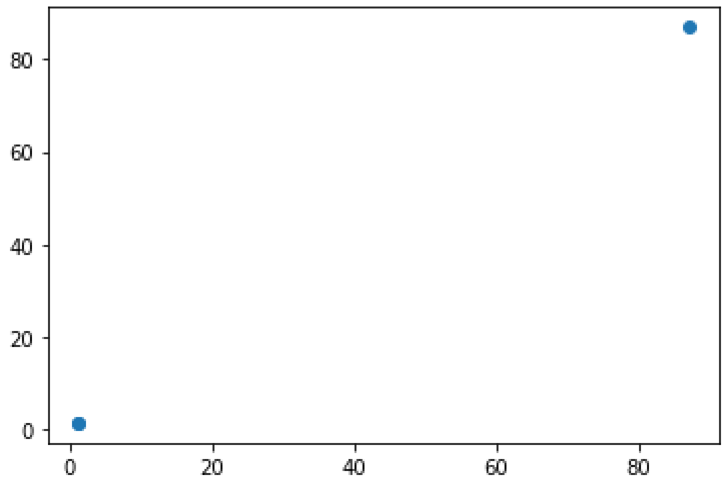
	Co-efficient
BEN	0.000000e+00
CO	-6.211505e-14
EBE	0.000000e+00
MXY	0.000000e+00
NMHC	9.849200e-01
NO_2	7.333005e-16
NOx	-9.633179e-17
OXY	0.000000e+00

In [23]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[23]:

```
&lt;matplotlib.collections.PathCollection at 0x1cbf3a3d130&gt;
```



In [24]:

```
print(lr.score(x_test,y_test))
```

```
0.9999924406308133
```

In [25]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [26]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[26]:

```
Ridge(alpha=10)
```

In [27]:

```
rr.score(x_test,y_test)
```

Out[27]:

```
0.9999685317021586
```

In [28]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[28]:

```
Lasso(alpha=10)
```

In [29]:

```
la.score(x_test,y_test)
```

Out[29]:

```
0.9996339281315851
```

In [30]:

```
a1=b.head(7000)
a1
```

Out[30]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2001-08-01 01:00:00	87.00	0.37	87.000000	87.0	87.00	58.400002	87.150002	87.00	34.529999
1	2001-08-01 01:00:00	1.50	0.34	1.490000	4.1	0.07	56.250000	75.169998	2.11	42.160000
2	2001-08-01 01:00:00	87.00	0.28	87.000000	87.0	87.00	50.660000	61.380001	87.00	46.310001
3	2001-08-01 01:00:00	87.00	0.47	87.000000	87.0	87.00	69.790001	73.449997	87.00	40.650002
4	2001-08-01 01:00:00	87.00	0.39	87.000000	87.0	87.00	22.830000	24.799999	87.00	66.309998
...	...	...	...	...	...	...	...	...	...	...
6995	2001-08-13 04:00:00	87.00	0.00	87.000000	87.0	0.08	18.580000	18.590000	87.00	56.660000
6996	2001-08-13 04:00:00	87.00	0.09	87.000000	87.0	87.00	29.580000	32.770000	87.00	52.709999
6997	2001-08-13 04:00:00	1.38	0.17	30.530001	87.0	0.25	54.880001	68.870003	87.00	23.240000
6998	2001-08-13 04:00:00	87.00	0.01	87.000000	87.0	87.00	19.580000	20.990000	87.00	51.270000
6999	2001-08-13 04:00:00	87.00	0.00	87.000000	87.0	0.05	17.200001	18.219999	87.00	38.090000

7000 rows × 16 columns

In [31]:

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

Out[31]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1
0	87.00	0.37	87.000000	87.0	87.00	58.400002	87.150002	87.00	34.529999	105.00000
1	1.50	0.34	1.490000	4.1	0.07	56.250000	75.169998	2.11	42.160000	100.59999
2	87.00	0.28	87.000000	87.0	87.00	50.660000	61.380001	87.00	46.310001	100.09999
3	87.00	0.47	87.000000	87.0	87.00	69.790001	73.449997	87.00	40.650002	69.77999
4	87.00	0.39	87.000000	87.0	87.00	22.830000	24.799999	87.00	66.309998	75.18000
...	...	...	...	...	...	...	...	...	...	.
6995	87.00	0.00	87.000000	87.0	0.08	18.580000	18.590000	87.00	56.660000	22.82000
6996	87.00	0.09	87.000000	87.0	87.00	29.580000	32.770000	87.00	52.709999	38.47000
6997	1.38	0.17	30.530001	87.0	0.25	54.880001	68.870003	87.00	23.240000	18.18000
6998	87.00	0.01	87.000000	87.0	87.00	19.580000	20.990000	87.00	51.270000	33.83000
6999	87.00	0.00	87.000000	87.0	0.05	17.200001	18.219999	87.00	38.090000	43.45999

7000 rows × 15 columns

In [32]:

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

In [33]:

```
h=StandardScaler().fit_transform(f)
```

In [34]:

```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[34]:

```
LogisticRegression(max_iter=10000)
```

In [49]:

```
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 [50]:

```
i=[[10,20,30,40,50,60,11,22,33,44,55,54,21,78]]
```

In [51]:

```
prediction=logr.predict(i)
print(prediction)
```

```
[28079021]
```

In [52]:

```
logr.classes_
```

Out[52]:

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

In [53]:

```
logr.predict_proba(i)[0][0]
```

Out[53]:

```
2.528560047135413e-268
```

In [54]:

```
logr.predict_proba(i)[0][1]
```

Out[54]:

```
4.8254923316380694e-139
```

In [55]:

```
logr.score(h_test,g_test)
```

Out[55]:

```
0.6261904761904762
```

In [40]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[40]:

```
ElasticNet()
```

In [41]:

```
print(en.coef_)
```

```
[0.         0.         0.         0.         0.98384654 0.
 0.         0.        ]
```

In [42]:

```
print(en.intercept_)
```

```
1.3920293645489608
```

In [43]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999809149884327
```

In [56]:

```
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[56]:

```
RandomForestClassifier()
```

In [57]:

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

In [58]:

```
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[58]:

```
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 [59]:

```
grid_search.best_score_
```

Out[59]:

```
0.633061224489796
```

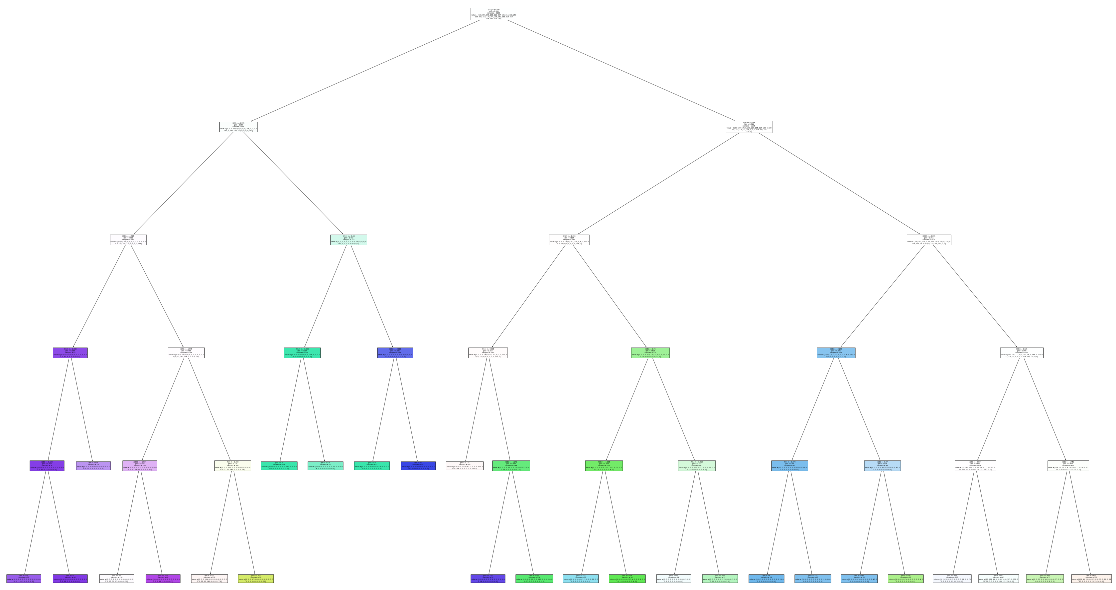
In [60]:

```
rfc_best=grid_search.best_estimator_
```

In [78]:

```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```



Conclusion: Linear score=0.9999924406308133, Ridge score=0.9999685317021586, Lasso Score=0.9996339281315851, Elastic Score=0.9999809149884327, Logistic Score=0.6261904761904762, RandomForest score=0.633061224489796.

Hence, Linear Regression model has the highest accuracy.

Madrid 2002

In [79]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2002.csv")
a
```

Out[79]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.99(
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.98(
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.44(
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.18(
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.33(
...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.75(
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.38(
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.64(
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	I
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.24(

217296 rows × 16 columns

In [80]:

```
a=a.head(1000)
a
```

Out[80]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.540000	41.99
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000	20.98
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.010000	28.44
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.120000	42.18
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.280000	76.33
...	...	...	...	...	...	...	...	...	...	...	
995	2002-04-02 16:00:00	2.19	0.36	NaN	NaN	NaN	58.709999	93.349998	NaN	70.830002	19.85
996	2002-04-02 16:00:00	0.87	0.30	1.00	NaN	0.09	32.580002	45.150002	NaN	77.910004	14.43
997	2002-04-02 16:00:00	0.46	0.50	0.27	0.41	0.10	11.550000	13.290000	0.49	82.260002	19.80
998	2002-04-02 16:00:00	1.11	0.44	0.96	2.00	NaN	96.790001	209.600006	0.66	34.540001	
999	2002-04-02 16:00:00	1.98	0.45	2.12	6.27	0.11	49.419998	75.730003	2.86	76.000000	25.16

1000 rows × 16 columns

In [81]:

```
b=a.dropna()
b
```

Out[81]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.200000	0.15	98.150002	153.399994	2.67	6.850000
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.650000	0.11	113.699997	187.000000	3.53	12.370000
22	2002-04-01 01:00:00	2.02	0.80	1.57	3.660000	0.15	93.860001	101.300003	1.77	6.990000
24	2002-04-01 01:00:00	3.02	1.04	2.43	5.380000	0.21	103.699997	195.399994	2.15	14.040000
26	2002-04-01 02:00:00	2.02	0.53	2.24	5.970000	0.12	91.599998	136.199997	2.55	6.760000
...	...	...	...	...	...	...	...	...	...	...
974	2002-04-02 15:00:00	2.09	0.57	2.17	5.880000	0.15	51.470001	83.099998	2.59	71.410004
976	2002-04-02 16:00:00	1.96	0.40	2.09	5.440000	0.07	57.000000	83.410004	2.35	67.790001
980	2002-04-02 16:00:00	5.06	0.63	6.80	17.209999	0.09	69.510002	141.600006	7.95	67.720001
997	2002-04-02 16:00:00	0.46	0.50	0.27	0.410000	0.10	11.550000	13.290000	0.49	82.260002
999	2002-04-02 16:00:00	1.98	0.45	2.12	6.270000	0.11	49.419998	75.730003	2.86	76.000000

160 rows × 16 columns

In [82]:

```
b.columns
```

Out[82]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [83]:

```
b=b[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
b
```

Out[83]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
1	1.93	0.71	2.33	6.200000	0.15	98.150002	153.399994	2.67	6.850000	20.980000
5	3.19	0.72	3.23	7.650000	0.11	113.699997	187.000000	3.53	12.370000	27.450001
22	2.02	0.80	1.57	3.660000	0.15	93.860001	101.300003	1.77	6.990000	33.000000
24	3.02	1.04	2.43	5.380000	0.21	103.699997	195.399994	2.15	14.040000	37.310001
26	2.02	0.53	2.24	5.970000	0.12	91.599998	136.199997	2.55	6.760000	19.980000
...	...	...	...	...	...	...	...	...	...	...
974	2.09	0.57	2.17	5.880000	0.15	51.470001	83.099998	2.59	71.410004	31.059999
976	1.96	0.40	2.09	5.440000	0.07	57.000000	83.410004	2.35	67.790001	25.690001
980	5.06	0.63	6.80	17.209999	0.09	69.510002	141.600006	7.95	67.720001	21.950001
997	0.46	0.50	0.27	0.410000	0.10	11.550000	13.290000	0.49	82.260002	19.809999
999	1.98	0.45	2.12	6.270000	0.11	49.419998	75.730003	2.86	76.000000	25.160000

160 rows × 15 columns

In [102]:

```
x=b.iloc[:,0:10]
y=b.iloc[:,-1]
```

In [103]:

```
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 [104]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[104]:

```
LinearRegression()
```

In [105]:

```
print(lr.intercept_)
```

```
28079025.314919464
```

In [106]:

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

Out[106]:

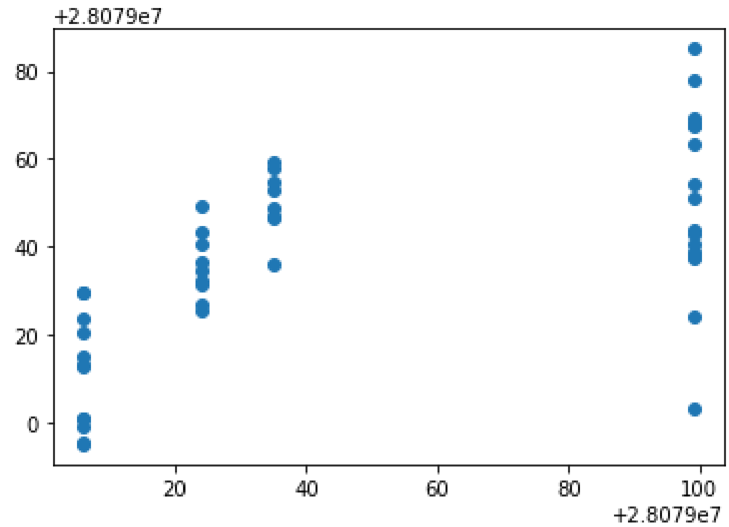
	Co-efficient
BEN	10.322864
CO	-96.321162
EBE	-44.748442
MXY	26.519033
NMHC	346.911033
NO_2	0.283597
NOx	-0.083552
OXY	-30.689805
O_3	0.271343
PM10	0.570190

In [107]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[107]:

```
<matplotlib.collections.PathCollection at 0x1cb80829b20>
```



In [108]:

```
print(lr.score(x_test,y_test))
```

```
0.26497089777435334
```

In [109]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[109]:

```
Ridge(alpha=10)
```

In [110]:

```
rr.score(x_test,y_test)
```

Out[110]:

```
0.09013363420469
```

In [111]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[111]:

```
Lasso(alpha=10)
```

In [112]:

```
la.score(x_test,y_test)
```

Out[112]:

```
-0.016235272401284417
```

In [113]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[113]:

```
ElasticNet()
```

In [114]:

```
print(en.coef_)
```

```
[-0.         -0.50619145 -3.18185688  0.          0.          1.06276317
 -0.28769662 -2.82256335  0.71368423  0.53976801]
```

In [115]:

```
print(en.intercept_)
```

```
28078969.129685692
```

In [116]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.05127349309556084
```

In [117]:

```
f=StandardScaler().fit_transform(x)
```

In [118]:

```
logr=LogisticRegression()
logr.fit(f,y)
```

Out[118]:

```
LogisticRegression()
```

In [120]:

```
g=[[10,20,30,40,50,60,70,80,90,10]]
```

In [121]:

```
prediction=logr.predict(g)
print(prediction)
```

```
[28079006]
```

In [122]:

```
logr.classes_
```

Out[122]:

```
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

In [123]:

```
logr.predict_proba(g)[0][0]
```

Out[123]:

```
1.0
```

In [131]:

```
logr.score(x_test,y_test)
```

Out[131]:

```
0.25
```

In [124]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[124]:

```
RandomForestClassifier()
```

In [125]:

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

In [126]:

```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[126]:

```
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 [127]:

```
grid_search.best_score_
```

Out[127]:

```
0.6875
```

In [128]:

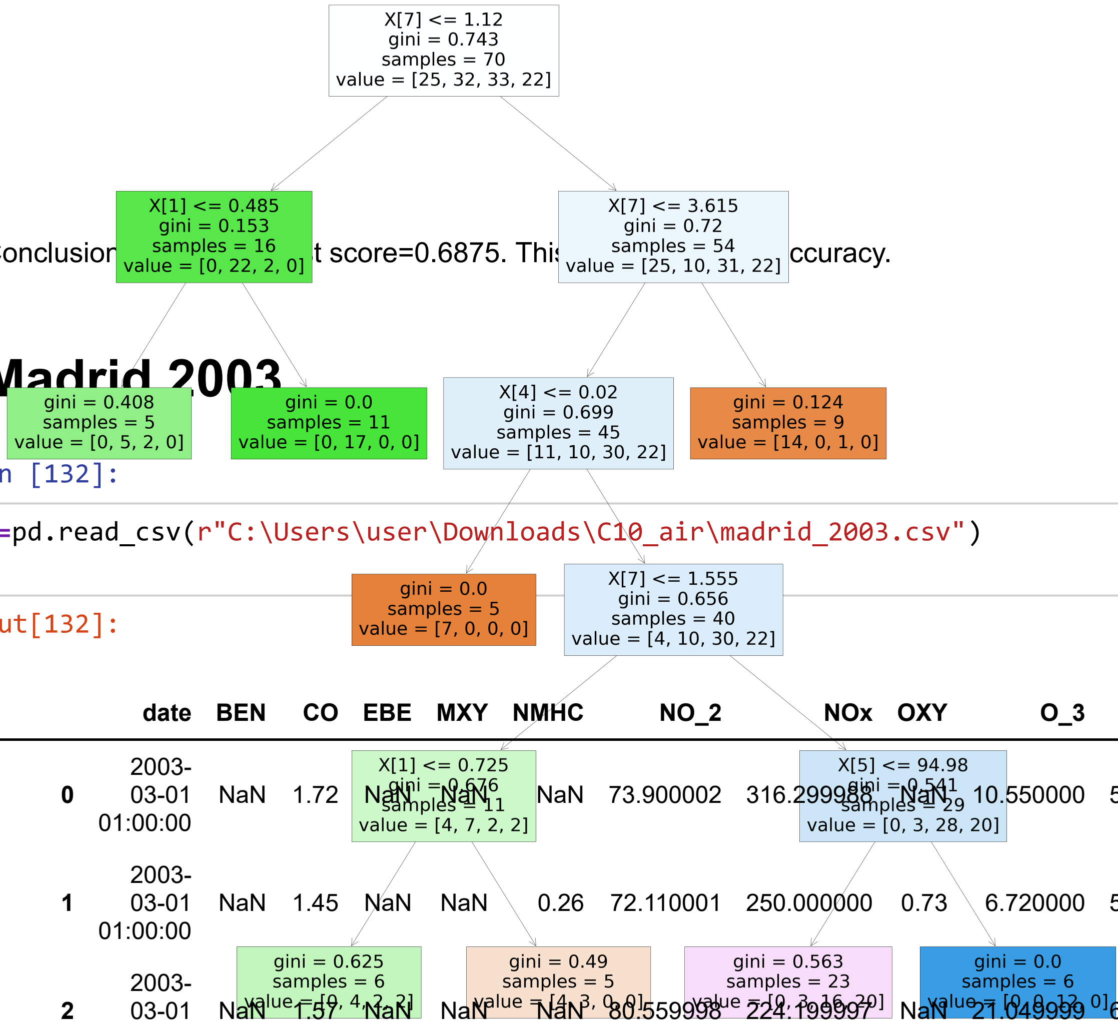
```
rfc_best=grid_search.best_estimator_
```

In [130]:

```
plt.figure(figsize=(80,80))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[130]:

```
[Text(1785.6, 3986.4, 'X[7] <= 1.12\ngini = 0.743\nsamples = 70\nvalue =
[25, 32, 33, 22]'),
 Text(892.8, 3261.6000000000004, 'X[1] <= 0.485\ngini = 0.153\nsamples = 1
6\nvalue = [0, 22, 2, 0]'),
 Text(446.4, 2536.8, 'gini = 0.408\nsamples = 5\nvalue = [0, 5, 2, 0]'),
 Text(1339.1999999999998, 2536.8, 'gini = 0.0\nsamples = 11\nvalue = [0, 1
7, 0, 0]'),
 Text(2678.3999999999996, 3261.6000000000004, 'X[7] <= 3.615\ngini = 0.72
\nsamples = 54\nvalue = [25, 10, 31, 22]'),
 Text(2232.0, 2536.8, 'X[4] <= 0.02\ngini = 0.699\nsamples = 45\nvalue =
[11, 10, 30, 22]'),
 Text(1785.6, 1812.0, 'gini = 0.0\nsamples = 5\nvalue = [7, 0, 0, 0]'),
 Text(2678.3999999999996, 1812.0, 'X[7] <= 1.555\ngini = 0.656\nsamples =
40\nvalue = [4, 10, 30, 22]'),
 Text(1785.6, 1087.1999999999998, 'X[1] <= 0.725\ngini = 0.676\nsamples =
11\nvalue = [4, 7, 2, 2]'),
 Text(1339.1999999999998, 362.39999999999964, 'gini = 0.625\nsamples = 6\n
value = [0, 4, 2, 2]'),
 Text(2232.0, 362.39999999999964, 'gini = 0.49\nsamples = 5\nvalue = [4,
3, 0, 0]'),
 Text(3571.2, 1087.1999999999998, 'X[5] <= 94.98\ngini = 0.541\nsamples =
29\nvalue = [0, 3, 28, 20]'),
 Text(3124.7999999999997, 362.39999999999964, 'gini = 0.563\nsamples = 23
\nvalue = [0, 3, 16, 20]'),
 Text(4017.6, 362.39999999999964, 'gini = 0.0\nsamples = 6\nvalue = [0, 0,
12, 0]'),
 Text(3124.7999999999997, 2536.8, 'gini = 0.124\nsamples = 9\nvalue = [14,
0, 1, 0]')]
```



Conclusion ... t score=0.6875. Thi... ccuracy.

Madrid 2003

In [132]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2003.csv")
a
```

Out[132]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.
...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.

243984 rows × 16 columns

In [133]:

```
a=a.head(2000)
a
```

Out[133]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.20
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.38
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.24
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.83
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.77
...	...	...	...	...	...	...	...	...	...	...	
1995	2003-03-04 00:00:00	NaN	1.70	NaN	NaN	0.27	76.070000	291.399994	NaN	6.970000	19.67
1996	2003-03-04 00:00:00	NaN	1.49	NaN	NaN	NaN	64.769997	204.600006	NaN	2.380000	32.41
1997	2003-03-04 00:00:00	NaN	1.71	NaN	NaN	NaN	51.099998	130.600006	NaN	7.810000	22.07
1998	2003-03-04 00:00:00	NaN	1.54	NaN	NaN	0.36	73.330002	287.600006	NaN	4.030000	39.18
1999	2003-03-04 00:00:00	NaN	0.86	NaN	NaN	NaN	58.360001	109.400002	NaN	8.860000	19.70

2000 rows × 16 columns

In [135]:

```
b=a.dropna()
b
```

Out[135]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.95	95.1500
23	2003-03-01 01:00:00	3.46	1.27	3.43	7.08	0.18	54.250000	173.300003	3.37	6.54	53.0099
27	2003-03-01 01:00:00	6.39	1.79	5.75	10.88	0.33	75.459999	281.100006	3.68	6.69	63.8400
33	2003-03-01 02:00:00	7.42	1.47	10.63	24.73	0.35	83.309998	277.200012	11.00	9.90	58.8800
51	2003-03-01 02:00:00	3.62	1.29	3.20	7.08	0.19	42.209999	166.300003	3.41	6.38	47.5999
...	...	...	...	...	...	...	...	...	...	...	
1959	2003-03-03 22:00:00	3.38	1.25	3.17	6.34	0.23	60.830002	172.500000	2.66	7.19	29.9900
1965	2003-03-03 23:00:00	5.59	1.19	7.04	17.40	0.28	62.180000	200.800003	8.45	10.09	19.2999
1983	2003-03-03 23:00:00	1.10	0.64	0.84	1.75	0.07	51.970001	83.900002	1.05	5.91	35.0099
1987	2003-03-03 23:00:00	3.76	1.20	3.54	7.02	0.23	60.580002	170.100006	2.86	6.71	26.6200
1993	2003-03-04 00:00:00	4.54	1.69	6.07	15.06	0.35	71.220001	314.700012	7.48	10.07	46.3499

212 rows × 16 columns

In [136]:

```
b.columns
```

Out[136]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [137]:

```
b=b[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
b
```

Out[137]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
5	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.95	95.150002	7.94
23	3.46	1.27	3.43	7.08	0.18	54.250000	173.300003	3.37	6.54	53.009998	2.62
27	6.39	1.79	5.75	10.88	0.33	75.459999	281.100006	3.68	6.69	63.840000	4.24
33	7.42	1.47	10.63	24.73	0.35	83.309998	277.200012	11.00	9.90	58.880001	8.93
51	3.62	1.29	3.20	7.08	0.19	42.209999	166.300003	3.41	6.38	47.599998	2.70
...	...	...	...	...	...	...	...	...	...	...	...
1959	3.38	1.25	3.17	6.34	0.23	60.830002	172.500000	2.66	7.19	29.990000	2.99
1965	5.59	1.19	7.04	17.40	0.28	62.180000	200.800003	8.45	10.09	19.299999	6.90
1983	1.10	0.64	0.84	1.75	0.07	51.970001	83.900002	1.05	5.91	35.009998	0.72
1987	3.76	1.20	3.54	7.02	0.23	60.580002	170.100006	2.86	6.71	26.620001	3.28
1993	4.54	1.69	6.07	15.06	0.35	71.220001	314.700012	7.48	10.07	46.349998	5.99

212 rows × 15 columns

In [138]:

```
x=b.iloc[:,0:5]
y=b.iloc[:,-1]
```

In [139]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [140]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[140]:

```
LinearRegression()
```

In [141]:

```
print(lr.intercept_)
```

```
28079100.77042889
```

In [142]:

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

Out[142]:

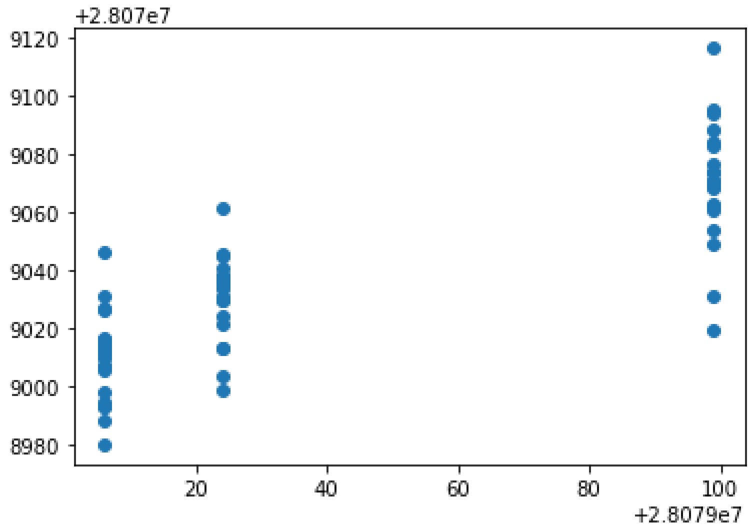
	Co-efficient
BEN	-4.354966
CO	-170.372248
EBE	16.835167
MXY	-11.874334
NMHC	860.450085

In [143]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[143]:

<matplotlib.collections.PathCollection at 0x1cb80a0d790>



In [144]:

```
print(lr.score(x_test,y_test))
```

0.6422489114361061

In [145]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[145]:

Ridge(alpha=10)

In [146]:

```
rr.score(x_test,y_test)
```

Out[146]:

```
0.1877820357689176
```

In [147]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[147]:

```
Lasso(alpha=10)
```

In [148]:

```
la.score(x_test,y_test)
```

Out[148]:

```
0.04866758105027491
```

In [149]:

```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[149]:

```
ElasticNet()
```

In [150]:

```
print(en.coef_)
```

```
[-1.06400298  1.31833139  3.55029825 -4.02609241  1.16258014]
```

In [151]:

```
print(en.intercept_)
```

```
28079055.404426787
```

In [152]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.07228820797072222
```

In [153]:

```
f=StandardScaler().fit_transform(x)
```

In [154]:

```
logr=LogisticRegression()
logr.fit(f,y)
```

Out[154]:

```
LogisticRegression()
```

In [155]:

```
g=[[10,20,30,40,50]]
```

In [156]:

```
prediction=logr.predict(g)
print(prediction)
```

```
[28079099]
```

In [157]:

```
logr.classes_
```

Out[157]:

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

In [158]:

```
logr.predict_proba(g)[0][0]
```

Out[158]:

```
1.3162606173431188e-26
```

In [159]:

```
logr.score(x_test,y_test)
```

Out[159]:

```
0.34375
```

In [160]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[160]:

```
RandomForestClassifier()
```

In [161]:

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

In [162]:

```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[162]:

```
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 [163]:

```
grid_search.best_score_
```

Out[163]:

```
0.8040540540540541
```

In [164]:

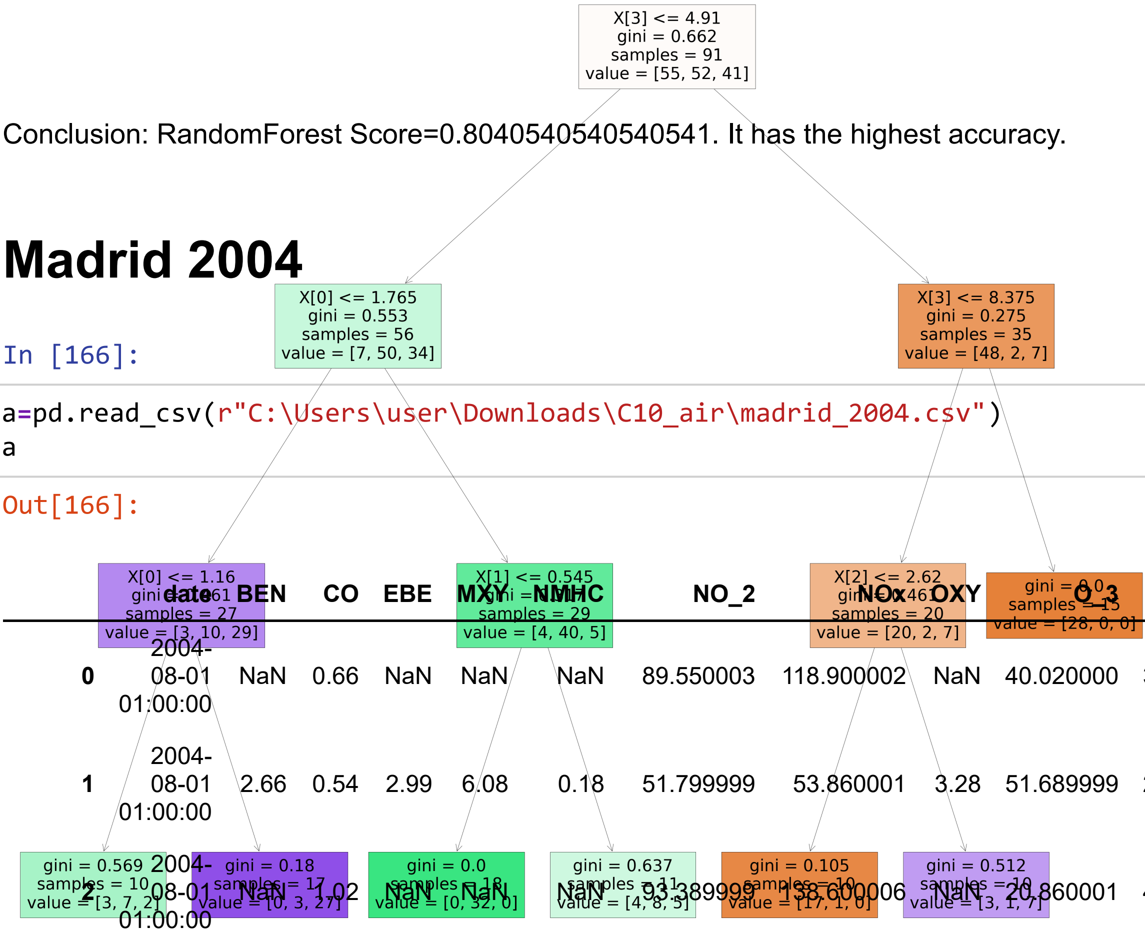
```
rfc_best=grid_search.best_estimator_
```

In [165]:

```
plt.figure(figsize=(80,80))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[165]:

```
[Text(2575.3846153846152, 3805.2000000000003, 'X[3] <= 4.91\ngini = 0.662
\nsamples = 91\nvalue = [55, 52, 41]'),
 Text(1373.5384615384614, 2718.0, 'X[0] <= 1.765\ngini = 0.553\nsamples =
56\nvalue = [7, 50, 34]'),
 Text(686.7692307692307, 1630.8000000000002, 'X[0] <= 1.16\ngini = 0.461\n
samples = 27\nvalue = [3, 10, 29]'),
 Text(343.38461538461536, 543.5999999999999, 'gini = 0.569\nsamples = 10\n
value = [3, 7, 2]'),
 Text(1030.1538461538462, 543.5999999999999, 'gini = 0.18\nsamples = 17\nv
alue = [0, 3, 27]'),
 Text(2060.3076923076924, 1630.8000000000002, 'X[1] <= 0.545\ngini = 0.317
\nsamples = 29\nvalue = [4, 40, 5]'),
 Text(1716.9230769230767, 543.5999999999999, 'gini = 0.0\nsamples = 18\nva
lue = [0, 32, 0]'),
 Text(2403.6923076923076, 543.5999999999999, 'gini = 0.637\nsamples = 11\n
value = [4, 8, 5]'),
 Text(3777.230769230769, 2718.0, 'X[3] <= 8.375\ngini = 0.275\nsamples = 3
5\nvalue = [48, 2, 7]'),
 Text(3433.8461538461534, 1630.8000000000002, 'X[2] <= 2.62\ngini = 0.461
\nsamples = 20\nvalue = [20, 2, 7]'),
 Text(3090.461538461538, 543.5999999999999, 'gini = 0.105\nsamples = 10\nv
alue = [17, 1, 0]'),
 Text(3777.230769230769, 543.5999999999999, 'gini = 0.512\nsamples = 10\nv
alue = [3, 1, 7]'),
 Text(4120.615384615385, 1630.8000000000002, 'gini = 0.0\nsamples = 15\nva
lue = [28, 0, 0]')]
```



Conclusion: RandomForest Score=0.8040540540540541. It has the highest accuracy.

Madrid 2004

In [166]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2004.csv")
a
```

Out[166]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39
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
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54
...	...	...	...	...	...	...	...	...	...	...	
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
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
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45
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

245496 rows × 17 columns

In [167]:

```
a=a.head(2000)
a
```

Out[167]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	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	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.48
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.07
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.08
...	...	...	...	...	...	...	...	...	...	...	
1995	2004-08-04 01:00:00	NaN	0.24	NaN	NaN	NaN	33.070000	36.939999	NaN	38.599998	4.19
1996	2004-08-04 01:00:00	NaN	0.02	NaN	NaN	NaN	18.049999	19.959999	NaN	60.380001	1.94
1997	2004-08-04 01:00:00	0.93	0.24	0.94	1.50	0.01	29.910000	40.490002	1.25	50.660000	3.90
1998	2004-08-04 01:00:00	NaN	0.26	NaN	NaN	0.09	29.660000	35.599998	NaN	49.730000	5.32
1999	2004-08-04 01:00:00	0.23	0.33	0.90	NaN	0.00	47.259998	60.220001	NaN	33.980000	9.52

2000 rows × 17 columns

In [168]:

```
b=a.dropna()
b
```

Out[168]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
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	59.1
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.5
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.6
32	2004-08-01 02:00:00	1.94	0.67	3.14	4.91	0.06	113.500000	165.800003	2.56	26.980000	86.9
49	2004-08-01 02:00:00	0.29	0.30	0.47	0.76	0.07	33.919998	34.840000	0.46	75.570000	48.9
...	...	...	...	...	...	...	...	...	...	...	
1963	2004-08-03 23:00:00	1.23	0.39	1.34	3.08	0.11	52.779999	68.750000	2.79	32.990002	30.2
1969	2004-08-04 00:00:00	1.02	0.36	1.00	1.18	0.02	43.529999	66.279999	1.55	42.520000	25.5
1987	2004-08-04 00:00:00	0.58	0.08	0.51	0.64	0.00	10.330000	10.820000	0.68	46.009998	16.4
1991	2004-08-04 00:00:00	1.32	0.27	1.31	1.90	0.08	35.349998	46.880001	1.43	39.380001	32.4
1997	2004-08-04 01:00:00	0.93	0.24	0.94	1.50	0.01	29.910000	40.490002	1.25	50.660000	3.9

195 rows × 17 columns

In [169]:

```
b.columns
```

Out[169]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [170]:

```
b=b[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
b
```

Out[170]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
5	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110001	38.049999	4.16
22	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.549999	22.780001	0.39
26	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.630001	29.459999	2.21
32	1.94	0.67	3.14	4.91	0.06	113.500000	165.800003	2.56	26.980000	86.930000	45.639999	2.14
49	0.29	0.30	0.47	0.76	0.07	33.919998	34.840000	0.46	75.570000	48.959999	25.959999	0.34
...	...	...	...	...	...	...	...	...	...	...	...	...
1963	1.23	0.39	1.34	3.08	0.11	52.779999	68.750000	2.79	32.990002	30.200001	19.670000	2.73
1969	1.02	0.36	1.00	1.18	0.02	43.529999	66.279999	1.55	42.520000	25.570000	12.660000	1.72
1987	0.58	0.08	0.51	0.64	0.00	10.330000	10.820000	0.68	46.009998	16.410000	6.300000	0.27

In [171]:

```
x=b.iloc[:,0:6]
y=b.iloc[:,-1]
```

In [172]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [173]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[173]:

```
LinearRegression()
```

In [174]:

```
print(lr.score(x_test,y_test))
```

```
0.18735247964299628
```

In [175]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[175]:

```
Ridge(alpha=10)
```

In [176]:

```
rr.score(x_test,y_test)
```

Out[176]:

```
0.11385632125962974
```

In [177]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[177]:

```
Lasso(alpha=10)
```

In [178]:

```
la.score(x_test,y_test)
```

Out[178]:

```
0.018612493009876552
```

In [179]:

```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[179]:

```
ElasticNet()
```

In [180]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.034590122360959485
```

In [181]:

```
f=StandardScaler().fit_transform(x)
```

In [182]:

```
logr=LogisticRegression()
logr.fit(f,y)
```

Out[182]:

```
LogisticRegression()
```

In [183]:

```
g=[[10,20,30,40,50,60]]
```

In [184]:

```
prediction=logr.predict(g)
print(prediction)
```

```
[28079006]
```

In [185]:

```
logr.predict_proba(g)[0][0]
```

Out[185]:

```
1.0
```

In [186]:

```
logr.score(x_test,y_test)
```

Out[186]:

```
0.2542372881355932
```

In [187]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[187]:

```
RandomForestClassifier()
```

In [188]:

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

In [189]:

```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[189]:

```
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 [190]:

```
grid_search.best_score_
```

Out[190]:

```
0.9044117647058824
```

In [191]:

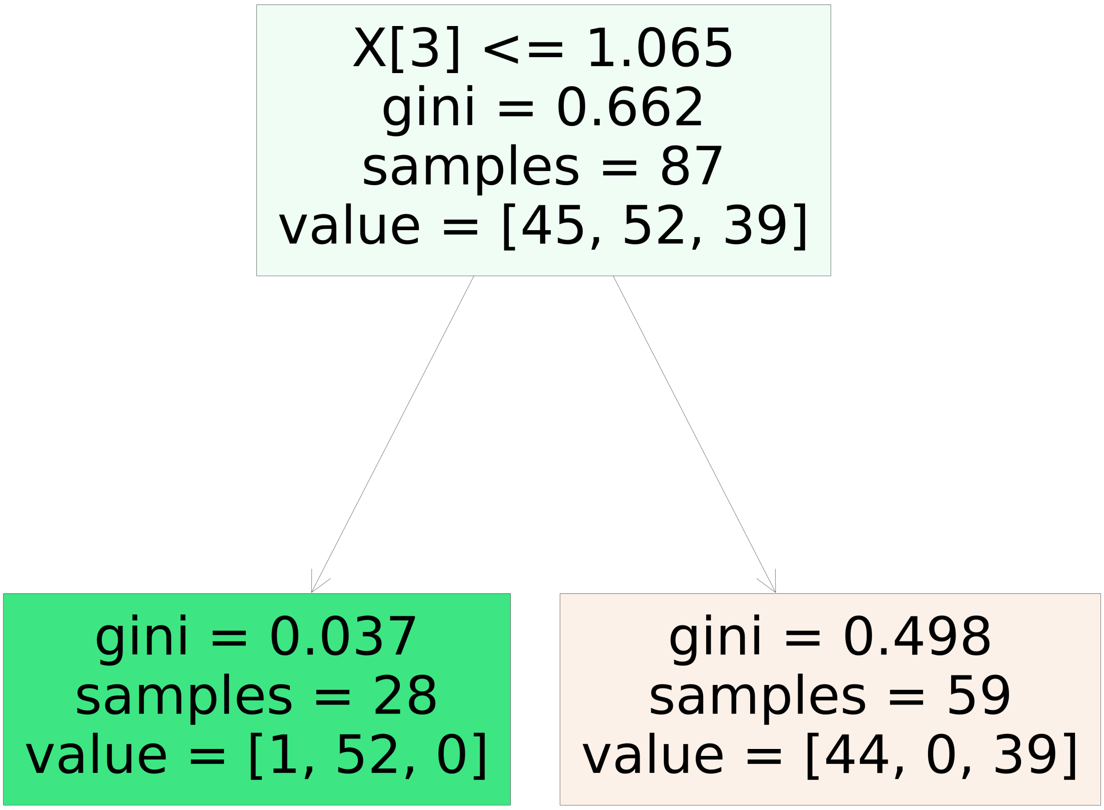
```
rfc_best=grid_search.best_estimator_
```

In [192]:

```
plt.figure(figsize=(80,80))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[192]:

```
[Text(2232.0, 3261.6000000000004, 'X[3] <= 1.065\ngini = 0.662\nsamples =
87\nvalue = [45, 52, 39]'),
 Text(1116.0, 1087.1999999999998, 'gini = 0.037\nsamples = 28\nvalue = [1,
52, 0]'),
 Text(3348.0, 1087.1999999999998, 'gini = 0.498\nsamples = 59\nvalue = [4
4, 0, 39]')]
```



Conclusion: RandomForest Score=0.9044117647058824. It has the highest accuracy.