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 2005

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2005.csv")
a
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.
...	...	...	...	...	...	...	...	...	...	...	
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998	5.
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.

237000 rows × 17 columns

In [3]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

In [4]:

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2005-11-01 01:00:00	87.00	0.77	87.00	87.00	87.00	57.130001	128.699997	87.00	14.720000
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000
2	2005-11-01 01:00:00	87.00	0.40	87.00	87.00	87.00	46.119999	53.000000	87.00	30.469999
3	2005-11-01 01:00:00	87.00	0.42	87.00	87.00	87.00	37.220001	52.009998	87.00	21.379999
4	2005-11-01 01:00:00	87.00	0.57	87.00	87.00	87.00	32.160000	36.680000	87.00	33.410000
...	...	...	...	...	...	...	...	...	...	...
236995	2006-01-01 00:00:00	1.08	0.36	1.01	87.00	0.11	21.990000	23.610001	87.00	43.349998
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999
236997	2006-01-01 00:00:00	0.19	87.00	0.26	87.00	0.08	26.730000	30.809999	87.00	43.840000
236998	2006-01-01 00:00:00	0.14	87.00	1.00	87.00	0.06	13.770000	17.770000	87.00	87.000000
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998

237000 rows × 17 columns

In [5]:

```
b.columns
```

Out[5]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PM25', '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	P
0	2005-11-01 01:00:00	87.00	0.77	87.00	87.00	87.00	57.130001	128.699997	87.00	14.720000	14.91(
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.93(
2	2005-11-01 01:00:00	87.00	0.40	87.00	87.00	87.00	46.119999	53.000000	87.00	30.469999	14.60(
3	2005-11-01 01:00:00	87.00	0.42	87.00	87.00	87.00	37.220001	52.009998	87.00	21.379999	15.16(
4	2005-11-01 01:00:00	87.00	0.57	87.00	87.00	87.00	32.160000	36.680000	87.00	33.410000	5.00(
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.07(
6	2005-11-01 01:00:00	87.00	0.55	87.00	87.00	0.27	50.279999	77.209999	87.00	19.120001	18.20(
7	2005-11-01 01:00:00	0.20	0.38	1.00	87.00	0.27	51.759998	72.989998	87.00	14.810000	16.43(
8	2005-11-01 01:00:00	87.00	0.70	87.00	87.00	87.00	39.040001	43.860001	87.00	25.379999	16.13(
9	2005-11-01 01:00:00	87.00	0.56	87.00	87.00	87.00	41.820000	51.869999	87.00	24.290001	7.13(

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	0.77	87.00	87.00	87.00	57.130001	128.699997	87.00	14.720000	14.910000	87.0
1	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.930000	1.5
2	87.00	0.40	87.00	87.00	87.00	46.119999	53.000000	87.00	30.469999	14.600000	87.0
3	87.00	0.42	87.00	87.00	87.00	37.220001	52.009998	87.00	21.379999	15.160000	87.0
4	87.00	0.57	87.00	87.00	87.00	32.160000	36.680000	87.00	33.410000	5.000000	87.0
5	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.070000	2.4
6	87.00	0.55	87.00	87.00	0.27	50.279999	77.209999	87.00	19.120001	18.209999	87.0
7	0.20	0.38	1.00	87.00	0.27	51.759998	72.989998	87.00	14.810000	16.430000	87.0
8	87.00	0.70	87.00	87.00	87.00	39.040001	43.860001	87.00	25.379999	16.139999	87.0
9	87.00	0.56	87.00	87.00	87.00	41.820000	51.869999	87.00	24.290001	7.130000	87.0

In [8]:

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

In [9]:

```
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 [10]:

```
from sklearn.linear_model import LinearRegression

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

Out[10]:

```
LinearRegression()
```

In [11]:

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

```
1.3046947428330142
```

In [12]:

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

Out[12]:

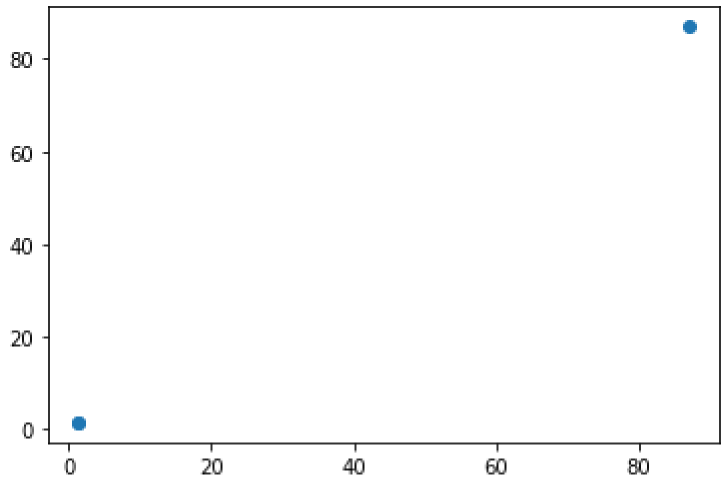
	Co-efficient
BEN	-9.883302e-04
CO	3.991521e-15
EBE	-9.792211e-04
MXY	0.000000e+00
NMHC	9.869711e-01
NO_2	-1.248144e-15
NOx	1.337900e-15
OXY	0.000000e+00

In [13]:

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

Out[13]:

```
<matplotlib.collections.PathCollection at 0x1fb71abd610>
```



In [14]:

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

```
0.9999880728948256
```

In [15]:

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

In [16]:

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

Out[16]:

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

In [17]:

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

Out[17]:

```
0.9974996485054091
```

In [18]:

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

Out[18]:

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

In [19]:

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

Out[19]:

```
0.999896379768667
```

In [20]:

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

Out[20]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2005-11-01 01:00:00	87.00	0.77	87.00	87.00	87.00	57.130001	128.699997	87.00	14.720000
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000
2	2005-11-01 01:00:00	87.00	0.40	87.00	87.00	87.00	46.119999	53.000000	87.00	30.469999
3	2005-11-01 01:00:00	87.00	0.42	87.00	87.00	87.00	37.220001	52.009998	87.00	21.379999
4	2005-11-01 01:00:00	87.00	0.57	87.00	87.00	87.00	32.160000	36.680000	87.00	33.410000
...	...	...	...	...	...	...	...	...	...	...
6995	2005-11-11 21:00:00	1.11	0.56	1.85	4.41	0.25	73.570000	100.599998	1.33	11.450000
6996	2005-11-11 21:00:00	0.49	87.00	0.25	87.00	0.14	119.800003	254.500000	87.00	2.060000
6997	2005-11-11 21:00:00	0.25	87.00	0.51	87.00	0.10	73.500000	104.300003	87.00	87.000000
6998	2005-11-11 21:00:00	1.59	0.83	2.06	8.59	0.26	87.279999	118.400002	3.23	7.390000
6999	2005-11-11 22:00:00	87.00	0.78	87.00	87.00	87.00	53.900002	166.000000	87.00	11.820000

7000 rows × 17 columns

In [21]:

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

Out[21]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	87.00	0.77	87.00	87.00	87.00	57.130001	128.699997	87.00	14.720000	14.910000
1	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.930000
2	87.00	0.40	87.00	87.00	87.00	46.119999	53.000000	87.00	30.469999	14.600000
3	87.00	0.42	87.00	87.00	87.00	37.220001	52.009998	87.00	21.379999	15.160000
4	87.00	0.57	87.00	87.00	87.00	32.160000	36.680000	87.00	33.410000	5.000000
...	...	...	...	...	...	...	...	...	...	...
6995	1.11	0.56	1.85	4.41	0.25	73.570000	100.599998	1.33	11.450000	29.129999
6996	0.49	87.00	0.25	87.00	0.14	119.800003	254.500000	87.00	2.060000	49.290001
6997	0.25	87.00	0.51	87.00	0.10	73.500000	104.300003	87.00	87.000000	22.580000
6998	1.59	0.83	2.06	8.59	0.26	87.279999	118.400002	3.23	7.390000	45.310001
6999	87.00	0.78	87.00	87.00	87.00	53.900002	166.000000	87.00	11.820000	32.619999

7000 rows × 15 columns

In [22]:

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

In [23]:

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

In [24]:

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

Out[24]:

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

In [25]:

```
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 [26]:

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

In [27]:

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

```
[28079039]
```

In [28]:

```
logr.classes_
```

Out[28]:

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

In [29]:

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

Out[29]:

```
3.753358031646902e-270
```

In [30]:

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

Out[30]:

```
2.6875000480093768e-192
```

In [31]:

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

Out[31]:

```
0.5219047619047619
```

In [32]:

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

Out[32]:

```
ElasticNet()
```

In [33]:

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

```
[-6.20369071e-05 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
  9.85375399e-01 -0.00000000e+00 -0.00000000e+00  0.00000000e+00]
```

In [34]:

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

```
1.261713491480208
```

In [35]:

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

```
0.999992864019473
```

In [36]:

```
from sklearn.ensemble import RandomForestClassifier

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

Out[36]:

```
RandomForestClassifier()
```

In [37]:

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

In [38]:

```
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[38]:

```
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 [39]:

```
grid_search.best_score_
```

Out[39]:

```
0.5553061224489796
```

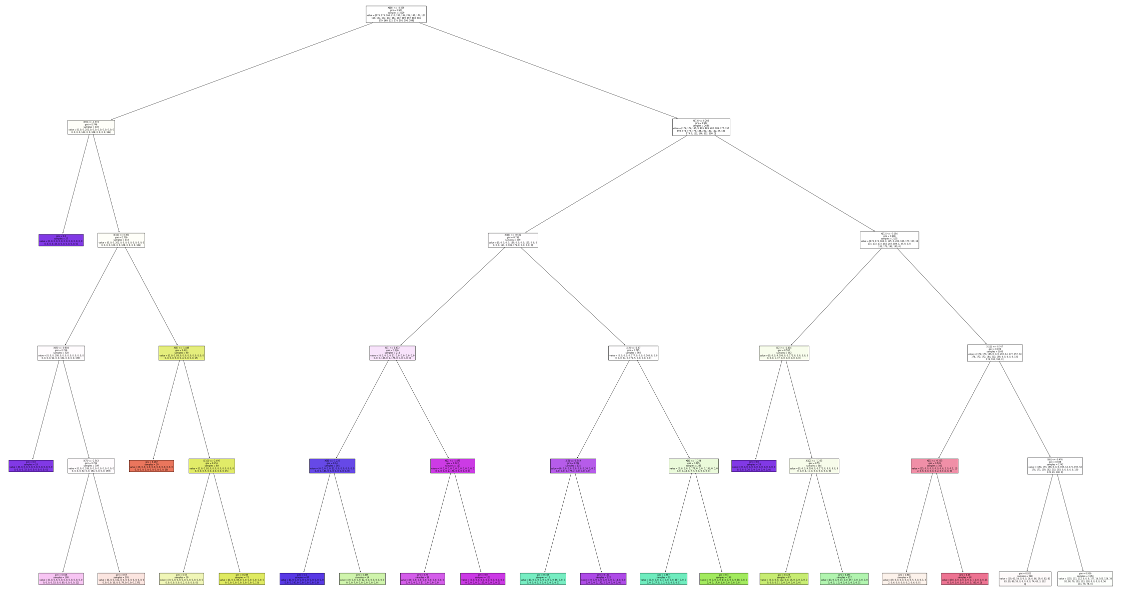
In [40]:

```
rfc_best=grid_search.best_estimator_
```

In [41]:

```
from sklearn.tree import plot_tree

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



Conclusion: Linear score=0.9999880728948256.It has the highest accuracy.

Madrid 2006

In [42]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2006.csv")
a
```

Out[42]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52

230568 rows × 17 columns

In [43]:

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

Out[43]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.88	97.57000
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.10	25.82000
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.43	34.41999
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.83	28.26000
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.99	54.18000
...	...	...	...	...	...	...	...	...	...	...	..
995	2006-02-02 15:00:00	0.20	0.92	0.13	NaN	0.32	135.000000	271.299988	NaN	12.48	98.87999
996	2006-02-02 15:00:00	NaN	0.96	NaN	NaN	NaN	134.199997	230.300003	NaN	8.23	77.29000
997	2006-02-02 15:00:00	NaN	1.21	NaN	NaN	NaN	141.699997	251.399994	NaN	14.56	146.60000
998	2006-02-02 15:00:00	NaN	1.38	NaN	NaN	0.35	83.239998	218.399994	NaN	8.91	94.30999
999	2006-02-02 15:00:00	NaN	1.22	NaN	NaN	NaN	83.820000	237.500000	NaN	8.03	91.66000

1000 rows × 17 columns

In [44]:

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

Out[44]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99	89.
22	2006-02-01 01:00:00	1.69	0.79	1.24	2.670000	0.17	59.910000	120.199997	1.11	2.45	25.
25	2006-02-01 01:00:00	2.35	1.47	2.64	9.660000	0.40	117.699997	346.399994	5.15	4.78	59.
31	2006-02-01 02:00:00	4.39	0.85	7.92	17.139999	0.25	92.059998	237.000000	9.24	5.92	35.
48	2006-02-01 02:00:00	1.93	0.79	1.24	2.740000	0.16	60.189999	125.099998	1.11	2.28	26.
...	...	...	...	...	...	...	...	...	...	...	
961	2006-02-02 13:00:00	2.31	1.25	2.36	10.110000	0.34	120.900002	317.600006	4.77	8.05	105.
967	2006-02-02 14:00:00	5.35	1.40	8.01	16.420000	0.39	116.599998	336.799988	8.53	7.00	91.
984	2006-02-02 14:00:00	3.64	0.83	4.88	9.370000	0.32	101.300003	221.199997	3.60	10.73	103.
987	2006-02-02 14:00:00	2.03	1.02	2.77	10.920000	0.30	110.400002	250.600006	5.14	10.44	86.
993	2006-02-02 15:00:00	5.61	1.43	8.89	16.879999	0.42	124.400002	332.299988	8.43	7.88	96.

114 rows × 17 columns

In [45]:

```
b.columns
```

Out[45]:

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

In [46]:

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

Out[46]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	I
5	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99	89.190002	1
22	1.69	0.79	1.24	2.670000	0.17	59.910000	120.199997	1.11	2.45	25.570000	(
25	2.35	1.47	2.64	9.660000	0.40	117.699997	346.399994	5.15	4.78	59.029999	.
31	4.39	0.85	7.92	17.139999	0.25	92.059998	237.000000	9.24	5.92	35.139999	:
48	1.93	0.79	1.24	2.740000	0.16	60.189999	125.099998	1.11	2.28	26.719999	(
...	...	...	...	...	...	...	...	...	...	...	
961	2.31	1.25	2.36	10.110000	0.34	120.900002	317.600006	4.77	8.05	105.599998	.
967	5.35	1.40	8.01	16.420000	0.39	116.599998	336.799988	8.53	7.00	91.269997	'
984	3.64	0.83	4.88	9.370000	0.32	101.300003	221.199997	3.60	10.73	103.199997	:
987	2.03	1.02	2.77	10.920000	0.30	110.400002	250.600006	5.14	10.44	86.180000	.
993	5.61	1.43	8.89	16.879999	0.42	124.400002	332.299988	8.43	7.88	96.820000	'

114 rows × 15 columns

In [47]:

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

In [48]:

```
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 [49]:

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

Out[49]:

```
LinearRegression()
```

In [50]:

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

```
28079031.625850987
```

In [51]:

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

Out[51]:

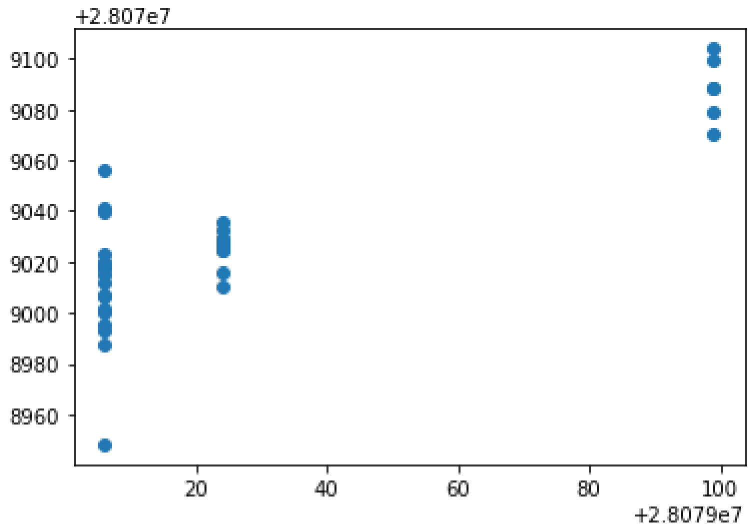
	Co-efficient
BEN	-15.068521
CO	-1.769391
EBE	-20.163669
MXY	4.181043
NMHC	121.256806
NO_2	0.066619
NOx	0.033429
OXY	11.005707
O_3	0.286746
PM10	-0.046453

In [52]:

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

Out[52]:

```
<matplotlib.collections.PathCollection at 0x1fb00045b20>
```



In [53]:

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

```
0.6946949587128924
```

In [54]:

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

Out[54]:

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

In [55]:

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

Out[55]:

```
0.6687715165120873
```

In [56]:

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

Out[56]:

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

In [57]:

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

Out[57]:

```
0.5077056017481698
```

In [58]:

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

Out[58]:

```
ElasticNet()
```

In [59]:

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

```
[-11.55763505   0.         -15.31882814   5.38582316   0.
   0.35172197   0.08588764   2.18792367   0.25010666  -0.06463277]
```

In [60]:

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

```
28079024.774824414
```

In [61]:

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

```
0.5740498211457492
```

In [62]:

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

In [63]:

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

Out[63]:

```
LogisticRegression()
```

In [64]:

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

In [65]:

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

```
[28079006]
```

In [66]:

```
logr.classes_
```

Out[66]:

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

In [67]:

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

Out[67]:

```
1.0
```

In [68]:

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

Out[68]:

```
0.4857142857142857
```

In [69]:

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

Out[69]:

```
RandomForestClassifier()
```

In [70]:

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

In [71]:

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

Out[71]:

```
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 [72]:

```
grid_search.best_score_
```

Out[72]:

```
0.8605769230769231
```

In [73]:

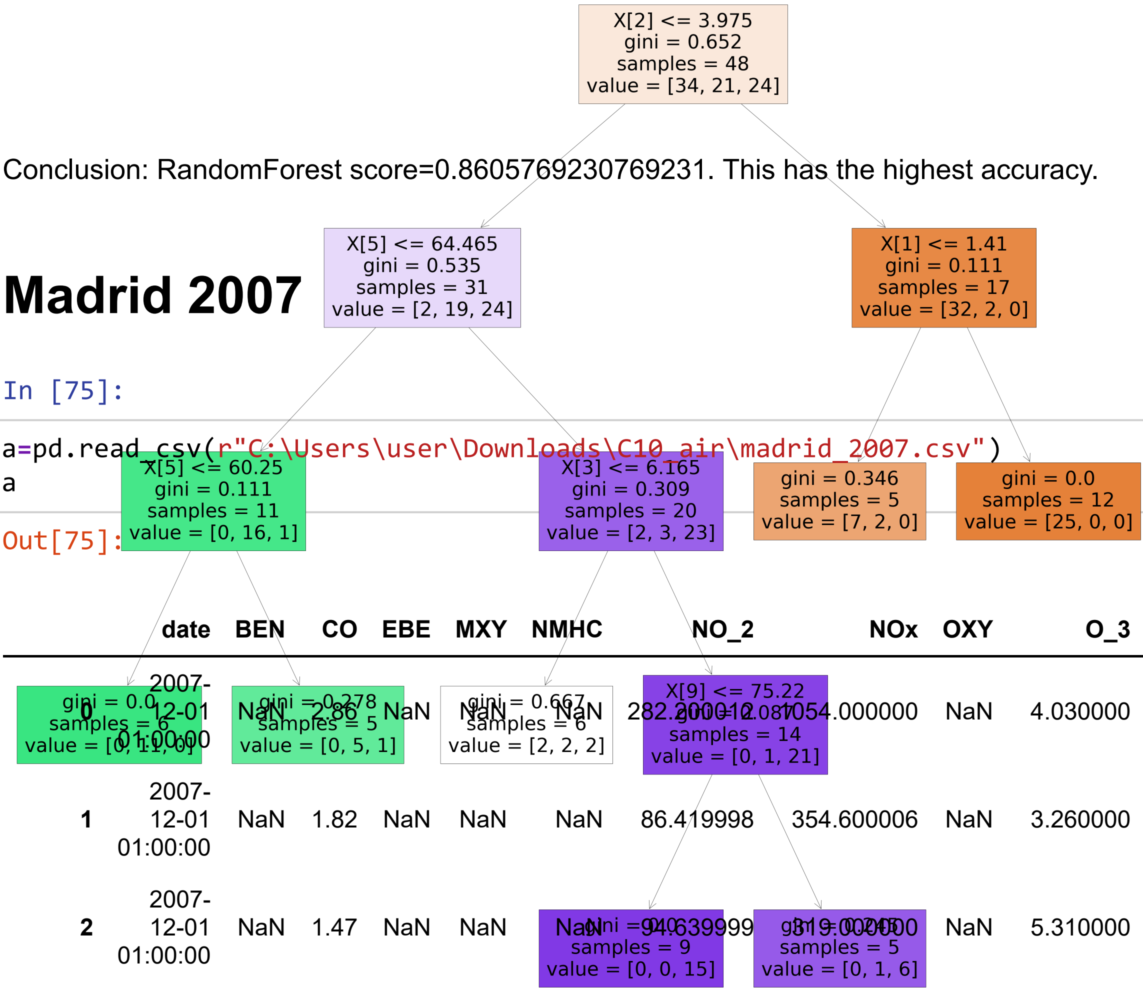
```
rfc_best=grid_search.best_estimator_
```

In [74]:

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

Out[74]:

```
[Text(2637.818181818182, 3913.92, 'X[2] <= 3.975\ngini = 0.652\nsamples = 
48\nvalue = [34, 21, 24]'),
 Text(1623.2727272727273, 3044.1600000000003, 'X[5] <= 64.465\ngini = 0.53
5\nsamples = 31\nvalue = [2, 19, 24]'),
 Text(811.6363636363636, 2174.4, 'X[5] <= 60.25\ngini = 0.111\nsamples = 1
1\nvalue = [0, 16, 1]'),
 Text(405.8181818181818, 1304.6400000000003, 'gini = 0.0\nsamples = 6\nval
ue = [0, 11, 0]'),
 Text(1217.4545454545455, 1304.6400000000003, 'gini = 0.278\nsamples = 5\n
value = [0, 5, 1]'),
 Text(2434.909090909091, 2174.4, 'X[3] <= 6.165\ngini = 0.309\nsamples = 2
0\nvalue = [2, 3, 23]'),
 Text(2029.090909090909, 1304.6400000000003, 'gini = 0.667\nsamples = 6\nv
alue = [2, 2, 2]'),
 Text(2840.7272727272725, 1304.6400000000003, 'X[9] <= 75.22\ngini = 0.087
\nsamples = 14\nvalue = [0, 1, 21]'),
 Text(2434.909090909091, 434.8800000000001, 'gini = 0.0\nsamples = 9\nvalu
e = [0, 0, 15]'),
 Text(3246.5454545454545, 434.8800000000001, 'gini = 0.245\nsamples = 5\nv
alue = [0, 1, 6]'),
 Text(3652.3636363636365, 3044.1600000000003, 'X[1] <= 1.41\ngini = 0.111
\nsamples = 17\nvalue = [32, 2, 0]'),
 Text(3246.5454545454545, 2174.4, 'gini = 0.346\nsamples = 5\nvalue = [7, 
2, 0]'),
 Text(4058.181818181818, 2174.4, 'gini = 0.0\nsamples = 12\nvalue = [25, 
0, 0]')]
```



Conclusion: RandomForest score=0.8605769230769231. This has the highest accuracy.

Madrid 2007

In [75]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2007.csv")
a
```

Out[75]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	1
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	1
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	1
...	...	...	...	...	...	...	...	...	...	...	
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999	
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000	
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001	
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN	
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000	

225120 rows × 17 columns

In [76]:

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

Out[76]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	156
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	80
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	53
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	105
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
...	...	...	...	...	...	...	...	...	...	...	
1995	2007-12-04 05:00:00	NaN	0.48	NaN	NaN	NaN	92.959999	213.199997	NaN	9.890000	38
1996	2007-12-04 05:00:00	1.04	0.80	2.32	NaN	0.53	87.180000	298.299988	NaN	NaN	21
1997	2007-12-04 05:00:00	1.26	0.27	3.62	8.30	0.40	61.240002	131.100006	2.46	1.000000	25
1998	2007-12-04 05:00:00	NaN	0.66	NaN	NaN	NaN	67.870003	243.600006	NaN	6.300000	34
1999	2007-12-04 05:00:00	1.65	NaN	0.56	NaN	0.48	102.800003	343.500000	NaN	8.420000	53

2000 rows × 17 columns

In [77]:

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

Out[77]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
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.!
21	2007-12-01 01:00:00	1.98	0.31	2.56	6.06	0.35	76.059998	208.899994	1.70	1.000000	37.'
25	2007-12-01 01:00:00	2.82	1.42	3.15	7.02	0.49	123.099998	402.399994	2.60	7.160000	70.8
30	2007-12-01 02:00:00	4.65	1.89	4.41	8.21	0.65	151.000000	622.700012	3.55	58.080002	117.(
47	2007-12-01 02:00:00	1.97	0.30	2.15	5.08	0.33	78.760002	189.800003	1.62	1.000000	34.'
...	...	...	...	...	...	...	...	...	...	...	
1954	2007-12-04 04:00:00	3.24	0.77	4.59	7.80	0.38	72.970001	278.899994	2.56	19.940001	58.!
1971	2007-12-04 04:00:00	2.46	0.29	3.74	8.57	0.43	68.750000	154.000000	2.58	1.650000	36.(
1975	2007-12-04 04:00:00	1.93	0.62	3.28	8.18	0.45	70.839996	203.300003	2.57	5.920000	39.:
1980	2007-12-04 05:00:00	2.83	0.67	3.67	6.05	0.33	70.599998	224.000000	1.94	17.370001	48.!
1997	2007-12-04 05:00:00	1.26	0.27	3.62	8.30	0.40	61.240002	131.100006	2.46	1.000000	25.'

204 rows × 17 columns

In [78]:

```
b.columns
```

Out[78]:

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

In [79]:

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

Out[79]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
4	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.500000	3
21	1.98	0.31	2.56	6.06	0.35	76.059998	208.899994	1.70	1.000000	37.799999	1
25	2.82	1.42	3.15	7.02	0.49	123.099998	402.399994	2.60	7.160000	70.809998	2
30	4.65	1.89	4.41	8.21	0.65	151.000000	622.700012	3.55	58.080002	117.099998	3
47	1.97	0.30	2.15	5.08	0.33	78.760002	189.800003	1.62	1.000000	34.740002	1
...	...	...	...	...	...	...	...	...	...	...	
1954	3.24	0.77	4.59	7.80	0.38	72.970001	278.899994	2.56	19.940001	58.950001	2
1971	2.46	0.29	3.74	8.57	0.43	68.750000	154.000000	2.58	1.650000	36.680000	2
1975	1.93	0.62	3.28	8.18	0.45	70.839996	203.300003	2.57	5.920000	39.250000	2
1980	2.83	0.67	3.67	6.05	0.33	70.599998	224.000000	1.94	17.370001	48.549999	2
1997	1.26	0.27	3.62	8.30	0.40	61.240002	131.100006	2.46	1.000000	25.139999	2

204 rows × 15 columns

In [80]:

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

In [81]:

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

In [82]:

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

Out[82]:

```
LinearRegression()
```

In [83]:

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

```
28079049.202183556
```

In [84]:

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

Out[84]:

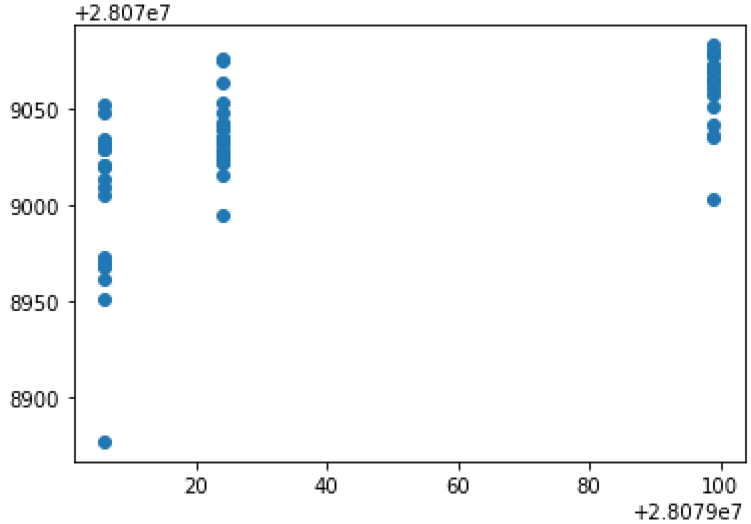
	Co-efficient
BEN	-46.668184
CO	76.731744
EBE	-11.694865
MXY	10.648704
NMHC	55.835179

In [85]:

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

Out[85]:

```
<matplotlib.collections.PathCollection at 0x1fb00628340>
```



In [86]:

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

```
0.19118050275762688
```

In [87]:

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

Out[87]:

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

In [88]:

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

Out[88]:

```
0.2729908756708399
```

In [89]:

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

Out[89]:

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

In [90]:

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

Out[90]:

```
0.07717393061665911
```

In [91]:

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

Out[91]:

```
ElasticNet()
```

In [92]:

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

```
[-10.5267769    4.41789412  -2.31393451   3.26756122   0.        ]
```

In [93]:

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

```
28079057.887038697
```

In [94]:

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

```
0.15936293659234257
```

In [95]:

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

In [96]:

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

Out[96]:

```
LogisticRegression()
```

In [97]:

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

In [98]:

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

```
[28079099]
```

In [99]:

```
logr.classes_
```

Out[99]:

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

In [100]:

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

Out[100]:

```
1.3439431794811398e-49
```

In [101]:

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

Out[101]:

```
0.3387096774193548
```

In [102]:

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

Out[102]:

```
RandomForestClassifier()
```

In [103]:

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

In [104]:

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

Out[104]:

```
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 [105]:

```
grid_search.best_score_
```

Out[105]:

```
0.8450704225352113
```

In [106]:

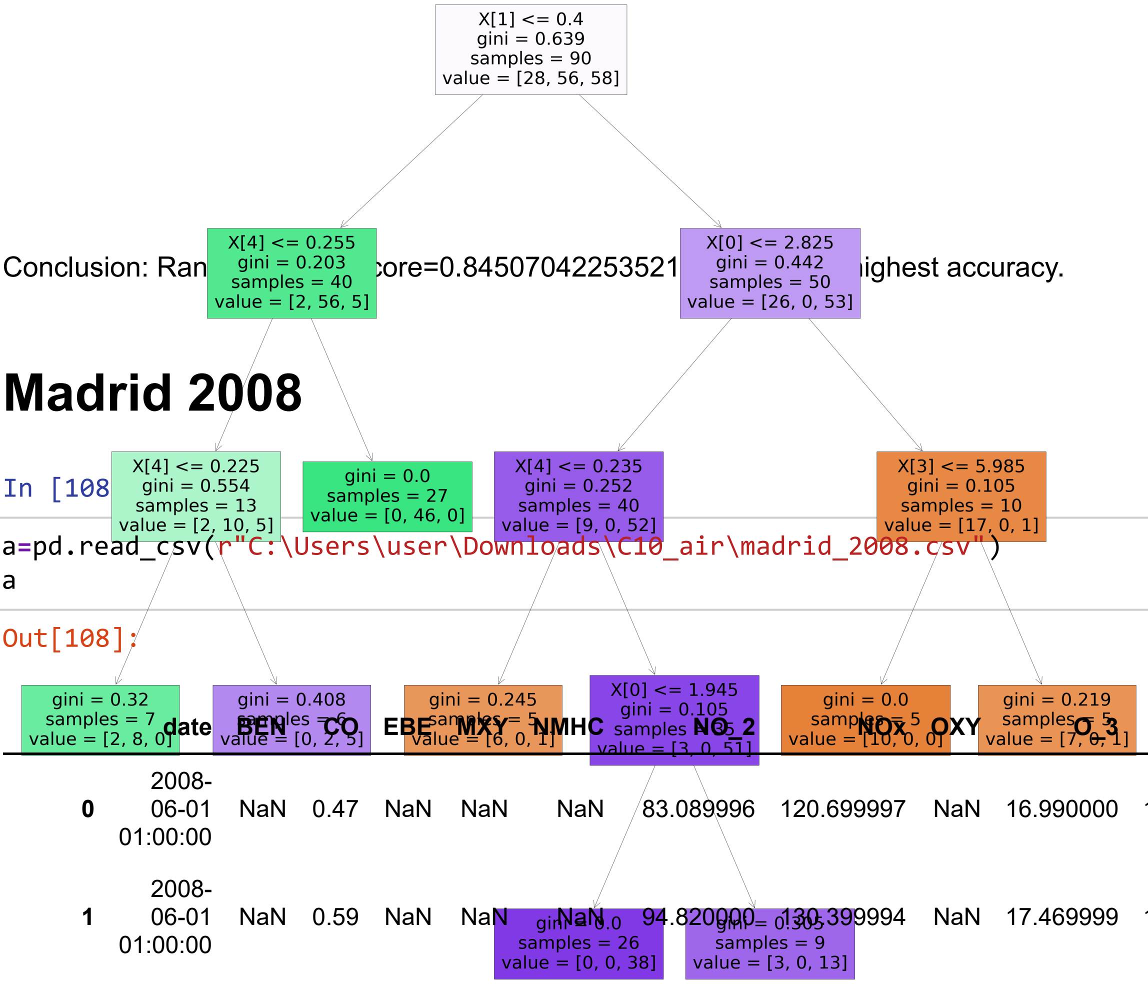
```
rfc_best=grid_search.best_estimator_
```

In [107]:

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

Out[107]:

```
[Text(2046.0, 3913.92, 'X[1] <= 0.4\ngini = 0.639\nsamples = 90\nvalue =
[28, 56, 58]'),
 Text(1116.0, 3044.1600000000003, 'X[4] <= 0.255\ngini = 0.203\nsamples =
40\nvalue = [2, 56, 5]'),
 Text(744.0, 2174.4, 'X[4] <= 0.225\ngini = 0.554\nsamples = 13\nvalue =
[2, 10, 5]'),
 Text(372.0, 1304.6400000000003, 'gini = 0.32\nsamples = 7\nvalue = [2, 8,
0]'),
 Text(1116.0, 1304.6400000000003, 'gini = 0.408\nsamples = 6\nvalue = [0,
2, 5]'),
 Text(1488.0, 2174.4, 'gini = 0.0\nsamples = 27\nvalue = [0, 46, 0]'),
 Text(2976.0, 3044.1600000000003, 'X[0] <= 2.825\ngini = 0.442\nsamples =
50\nvalue = [26, 0, 53]'),
 Text(2232.0, 2174.4, 'X[4] <= 0.235\ngini = 0.252\nsamples = 40\nvalue =
[9, 0, 52]'),
 Text(1860.0, 1304.6400000000003, 'gini = 0.245\nsamples = 5\nvalue = [6,
0, 1]'),
 Text(2604.0, 1304.6400000000003, 'X[0] <= 1.945\ngini = 0.105\nsamples =
35\nvalue = [3, 0, 51]'),
 Text(2232.0, 434.8800000000001, 'gini = 0.0\nsamples = 26\nvalue = [0, 0,
38]'),
 Text(2976.0, 434.8800000000001, 'gini = 0.305\nsamples = 9\nvalue = [3,
0, 13]'),
 Text(3720.0, 2174.4, 'X[3] <= 5.985\ngini = 0.105\nsamples = 10\nvalue =
[17, 0, 1]'),
 Text(3348.0, 1304.6400000000003, 'gini = 0.0\nsamples = 5\nvalue = [10,
0, 0]'),
 Text(4092.0, 1304.6400000000003, 'gini = 0.219\nsamples = 5\nvalue = [7,
0, 1]')]
```



Conclusion: Ran... core=0.84507042253521 ... ighest accuracy.

Madrid 2008

In [108]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2008.csv")
a
```

Out[108]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12

226392 rows × 17 columns

In [109]:

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

Out[109]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.8
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.0
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.2
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.8
4	2008-06-01 01:00:00	1.68	0.80	1.7	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.1
...	...	...	...	...	...	...	...	...	...	...	
1995	2008-06-04 05:00:00	NaN	0.11	NaN	NaN	NaN	22.450001	27.830000	NaN	49.660000	10.7
1996	2008-06-04 05:00:00	0.20	0.16	1.0	NaN	0.13	16.490000	19.920000	NaN	58.160000	7.8
1997	2008-06-04 05:00:00	0.20	0.28	1.0	1.00	0.29	9.380000	9.980000	1.00	86.279999	8.0
1998	2008-06-04 05:00:00	NaN	0.26	NaN	NaN	NaN	28.750000	42.820000	NaN	54.349998	6.1
1999	2008-06-04 05:00:00	0.19	NaN	1.0	NaN	0.11	39.560001	40.160000	NaN	53.270000	2.3

2000 rows × 17 columns

In [110]:

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

Out[110]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.1
21	2008-06-01 01:00:00	0.32	0.37	1.00	0.39	0.33	21.580000	22.180000	1.00	35.770000	7.9
25	2008-06-01 01:00:00	0.73	0.39	1.04	1.70	0.18	64.839996	86.709999	1.31	23.379999	14.7
30	2008-06-01 02:00:00	1.95	0.51	1.98	3.77	0.24	79.750000	143.399994	2.03	18.090000	31.1
47	2008-06-01 02:00:00	0.36	0.39	0.39	0.50	0.34	26.790001	27.389999	1.00	33.029999	7.6
...	...	...	...	...	...	...	...	...	...	...	
1954	2008-06-04 04:00:00	0.31	0.17	0.66	1.09	0.16	14.630000	20.770000	0.68	46.340000	15.4
1971	2008-06-04 04:00:00	0.20	0.26	1.00	1.00	0.30	9.760000	10.360000	1.00	88.480003	11.6
1975	2008-06-04 04:00:00	0.25	0.17	0.65	1.05	0.12	17.410000	22.000000	0.85	61.380001	10.5
1980	2008-06-04 05:00:00	0.25	0.17	0.59	0.83	0.16	12.900000	17.580000	0.52	70.309998	10.9
1997	2008-06-04 05:00:00	0.20	0.28	1.00	1.00	0.29	9.380000	9.980000	1.00	86.279999	8.0

230 rows × 17 columns

In [111]:

```
b.columns
```

Out[111]:

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

In [112]:

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

Out[112]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_
4	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000	21.90	1.43	10.9
21	0.32	0.37	1.00	0.39	0.33	21.580000	22.180000	1.00	35.770000	7.900000	6.14	1.00	5.3
25	0.73	0.39	1.04	1.70	0.18	64.839996	86.709999	1.31	23.379999	14.760000	9.84	1.22	6.8
30	1.95	0.51	1.98	3.77	0.24	79.750000	143.399994	2.03	18.090000	31.139999	18.41	1.81	8.9
47	0.36	0.39	0.39	0.50	0.34	26.790001	27.389999	1.00	33.029999	7.620000	6.25	0.38	5.5
...	...	...	...	...	...	...	...	...	...	...	...	...	
1954	0.31	0.17	0.66	1.09	0.16	14.630000	20.770000	0.68	46.340000	15.490000	5.71	0.53	6.2
1971	0.20	0.26	1.00	1.00	0.30	9.760000	10.360000	1.00	88.480003	11.670000	7.30	1.00	5.5
1975	0.25	0.17	0.65	1.05	0.12	17.410000	22.000000	0.85	61.380001	10.530000	6.31	0.77	6.2

In [113]:

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

In [114]:

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

In [115]:

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

Out[115]:

```
LinearRegression()
```

In [116]:

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

```
0.521094210347764
```

In [117]:

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

Out[117]:

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

In [118]:

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

Out[118]:

```
0.17005709953553705
```

In [119]:

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

Out[119]:

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

In [120]:

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

Out[120]:

```
-0.0017259380056555695
```

In [121]:

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

Out[121]:

```
ElasticNet()
```

In [122]:

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

```
0.030800475738876088
```

In [123]:

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

In [124]:

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

Out[124]:

```
LogisticRegression()
```

In [125]:

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

In [126]:

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

```
[28079006]
```

In [127]:

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

Out[127]:

```
1.0
```

In [128]:

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

Out[128]:

```
0.3188405797101449
```

In [129]:

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

Out[129]:

```
RandomForestClassifier()
```

In [130]:

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

In [131]:

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

Out[131]:

```
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 [132]:

```
grid_search.best_score_
```

Out[132]:

```
0.8511574074074073
```

In [133]:

```
rfc_best=grid_search.best_estimator_
```

In [134]:

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

Out[134]:

```
[Text(1826.1818181818182, 3805.2000000000003, 'X[5] <= 23.85\ngini = 0.656
\nsamples = 107\nvalue = [42, 53, 66]'),
 Text(811.6363636363636, 2718.0, 'X[2] <= 0.825\ngini = 0.105\nsamples = 3
7\nvalue = [3, 51, 0]'),
 Text(405.8181818181818, 1630.8000000000002, 'gini = 0.397\nsamples = 9\nv
alue = [3, 8, 0]'),
 Text(1217.4545454545455, 1630.8000000000002, 'gini = 0.0\nsamples = 28\nv
alue = [0, 43, 0]'),
 Text(2840.7272727272725, 2718.0, 'X[4] <= 0.205\ngini = 0.486\nsamples =
70\nvalue = [39, 2, 66]'),
 Text(2029.090909090909, 1630.8000000000002, 'X[4] <= 0.165\ngini = 0.163
\nsamples = 42\nvalue = [6, 0, 61]'),
 Text(1623.2727272727273, 543.5999999999999, 'gini = 0.0\nsamples = 17\nva
lue = [0, 0, 29]'),
 Text(2434.909090909091, 543.5999999999999, 'gini = 0.266\nsamples = 25\nv
alue = [6, 0, 32]'),
 Text(3652.3636363636365, 1630.8000000000002, 'X[1] <= 0.415\ngini = 0.301
\nsamples = 28\nvalue = [33, 2, 5]'),
 Text(3246.5454545454545, 543.5999999999999, 'gini = 0.653\nsamples = 5\nv
alue = [2, 2, 3]'),
 Text(4058.181818181818, 543.5999999999999, 'gini = 0.114\nsamples = 23\nv
alue = [31, 0, 2]')]
```

Conclusion: RandomForest Score=0.8511574074074073. It has the highest accuracy.