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 2009

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2009.csv")
a
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.

215688 rows × 17 columns

In [3]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215688 entries, 0 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     215688 non-null  object 
 1   BEN      60082 non-null   float64
 2   CO       190801 non-null  float64
 3   EBE      60081 non-null   float64
 4   MXY      24846 non-null   float64
 5   NMHC     74748 non-null   float64
 6   NO_2     214562 non-null  float64
 7   NOx      214565 non-null  float64
 8   OXY      24854 non-null   float64
 9   O_3      204482 non-null  float64
 10  PM10     196331 non-null  float64
 11  PM25     55822 non-null   float64
 12  PXY      24854 non-null   float64
 13  SO_2     212671 non-null  float64
 14  TCH      75213 non-null   float64
 15  TOL      59920 non-null   float64
 16  station  215688 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 28.0+ MB
```

In [4]:

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2009-10-01 01:00:00	87.00	0.27	87.00	87.00	87.00	39.889999	48.150002	87.00	50.680000
1	2009-10-01 01:00:00	87.00	0.22	87.00	87.00	87.00	21.230000	24.260000	87.00	55.880001
2	2009-10-01 01:00:00	87.00	0.18	87.00	87.00	87.00	31.230000	34.880001	87.00	49.060001
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998
4	2009-10-01 01:00:00	87.00	0.41	87.00	87.00	0.12	61.349998	76.260002	87.00	38.090000
...	...	...	...	...	...	...	...	...	...	...
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998
215684	2009-06-01 00:00:00	87.00	0.31	87.00	87.00	87.00	76.110001	101.099998	87.00	41.220001
215685	2009-06-01 00:00:00	0.13	87.00	0.86	87.00	0.23	81.050003	99.849998	87.00	24.830000
215686	2009-06-01 00:00:00	0.21	87.00	2.96	87.00	0.10	72.419998	82.959999	87.00	87.000000
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998

215688 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	PM
0	2009-10-01 01:00:00	87.00	0.27	87.00	87.00	87.00	39.889999	48.150002	87.00	50.680000	18.260(
1	2009-10-01 01:00:00	87.00	0.22	87.00	87.00	87.00	21.230000	24.260000	87.00	55.880001	10.580(
2	2009-10-01 01:00:00	87.00	0.18	87.00	87.00	87.00	31.230000	34.880001	87.00	49.060001	25.190(
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.530(
4	2009-10-01 01:00:00	87.00	0.41	87.00	87.00	0.12	61.349998	76.260002	87.00	38.090000	23.760(
5	2009-10-01 01:00:00	87.00	0.29	87.00	87.00	87.00	43.200001	50.080002	87.00	35.840000	21.870(
6	2009-10-01 01:00:00	87.00	0.20	87.00	87.00	87.00	35.430000	38.520000	87.00	33.549999	17.350(
7	2009-10-01 01:00:00	87.00	0.15	87.00	87.00	87.00	27.309999	33.150002	87.00	53.549999	16.520(
8	2009-10-01 01:00:00	87.00	0.21	87.00	87.00	0.39	33.889999	40.799999	87.00	58.549999	16.650(
9	2009-10-01 01:00:00	87.00	0.32	87.00	87.00	87.00	46.349998	60.540001	87.00	45.340000	15.160(

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	PXY
0	87.00	0.27	87.00	87.00	87.00	39.889999	48.150002	87.00	50.680000	18.260000	87.0
1	87.00	0.22	87.00	87.00	87.00	21.230000	24.260000	87.00	55.880001	10.580000	87.0
2	87.00	0.18	87.00	87.00	87.00	31.230000	34.880001	87.00	49.060001	25.190001	87.0
3	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.530001	1.3
4	87.00	0.41	87.00	87.00	0.12	61.349998	76.260002	87.00	38.090000	23.760000	87.0
5	87.00	0.29	87.00	87.00	87.00	43.200001	50.080002	87.00	35.840000	21.870001	87.0
6	87.00	0.20	87.00	87.00	87.00	35.430000	38.520000	87.00	33.549999	17.350000	87.0
7	87.00	0.15	87.00	87.00	87.00	27.309999	33.150002	87.00	53.549999	16.520000	87.0
8	87.00	0.21	87.00	87.00	0.39	33.889999	40.799999	87.00	58.549999	16.650000	87.0
9	87.00	0.32	87.00	87.00	87.00	46.349998	60.540001	87.00	45.340000	15.160000	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_)
```

```
0.6181014730252841
```

In [12]:

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

Out[12]:

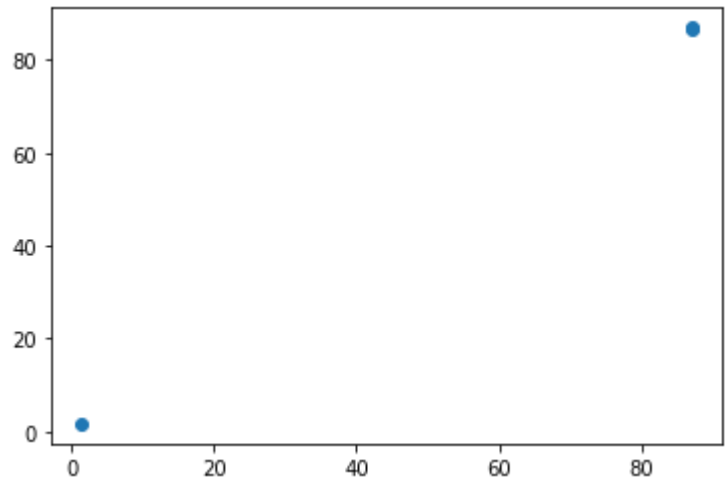
	Co-efficient
BEN	1.398881e-13
CO	-3.428767e+00
EBE	-6.342149e-15
MXY	-4.440892e-16
NMHC	9.878962e-01
NO_2	2.564760e-02
NOx	6.348843e-03
OXY	0.000000e+00

In [13]:

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

Out[13]:

```
<matplotlib.collections.PathCollection at 0x2a95bf6d640>
```



In [14]:

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

```
0.9999338807553346
```

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]:

```
0.9999954967522613
```

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]:

```
0.9999534400930205
```

In [20]:

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

Out[20]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2009-10-01 01:00:00	87.00	0.27	87.00	87.00	87.00	39.889999	48.150002	87.00	50.680000	18
1	2009-10-01 01:00:00	87.00	0.22	87.00	87.00	87.00	21.230000	24.260000	87.00	55.880001	10
2	2009-10-01 01:00:00	87.00	0.18	87.00	87.00	87.00	31.230000	34.880001	87.00	49.060001	25
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26
4	2009-10-01 01:00:00	87.00	0.41	87.00	87.00	0.12	61.349998	76.260002	87.00	38.090000	23
...	...	...	...	...	...	...	...	...	...	...	
6995	2009-10-12 16:00:00	0.42	0.74	0.43	1.08	0.49	11.680000	15.810000	0.67	84.389999	11
6996	2009-10-12 16:00:00	87.00	0.23	87.00	87.00	87.00	33.090000	54.380001	87.00	57.480000	16
6997	2009-10-12 16:00:00	0.13	87.00	0.31	87.00	0.19	27.670000	36.860001	87.00	56.240002	19
6998	2009-10-12 16:00:00	0.20	87.00	1.00	87.00	0.13	16.459999	30.200001	87.00	87.000000	30
6999	2009-10-12 16:00:00	0.23	0.25	0.63	1.08	0.18	22.760000	32.700001	0.67	64.739998	11

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	0.27	87.00	87.00	87.00	39.889999	48.150002	87.00	50.680000	18.260000	8
1	87.00	0.22	87.00	87.00	87.00	21.230000	24.260000	87.00	55.880001	10.580000	8
2	87.00	0.18	87.00	87.00	87.00	31.230000	34.880001	87.00	49.060001	25.190001	8
3	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.530001	
4	87.00	0.41	87.00	87.00	0.12	61.349998	76.260002	87.00	38.090000	23.760000	8
...	...	...	...	...	...	...	...	...	...	...	
6995	0.42	0.74	0.43	1.08	0.49	11.680000	15.810000	0.67	84.389999	11.110000	
6996	87.00	0.23	87.00	87.00	87.00	33.090000	54.380001	87.00	57.480000	16.969999	8
6997	0.13	87.00	0.31	87.00	0.19	27.670000	36.860001	87.00	56.240002	19.820000	8
6998	0.20	87.00	1.00	87.00	0.13	16.459999	30.200001	87.00	87.000000	30.650000	8
6999	0.23	0.25	0.63	1.08	0.18	22.760000	32.700001	0.67	64.739998	11.070000	

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)
```

```
[28079021]
```

In [28]:

```
logr.classes_
```

Out[28]:

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

In [29]:

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

Out[29]:

```
5.400803276887222e-153
```

In [30]:

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

Out[30]:

```
5.714501500703068e-41
```

In [31]:

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

Out[31]:

```
0.5633333333333334
```

In [32]:

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

Out[32]:

```
ElasticNet()
```

In [33]:

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

```
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 9.86230309e-01 0.00000000e+00 5.33110402e-04 0.00000000e+00]
```

In [34]:

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

```
1.1573688835558116
```

In [35]:

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

```
0.9999988176366412
```

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]:

```
0.5775510204081633
```

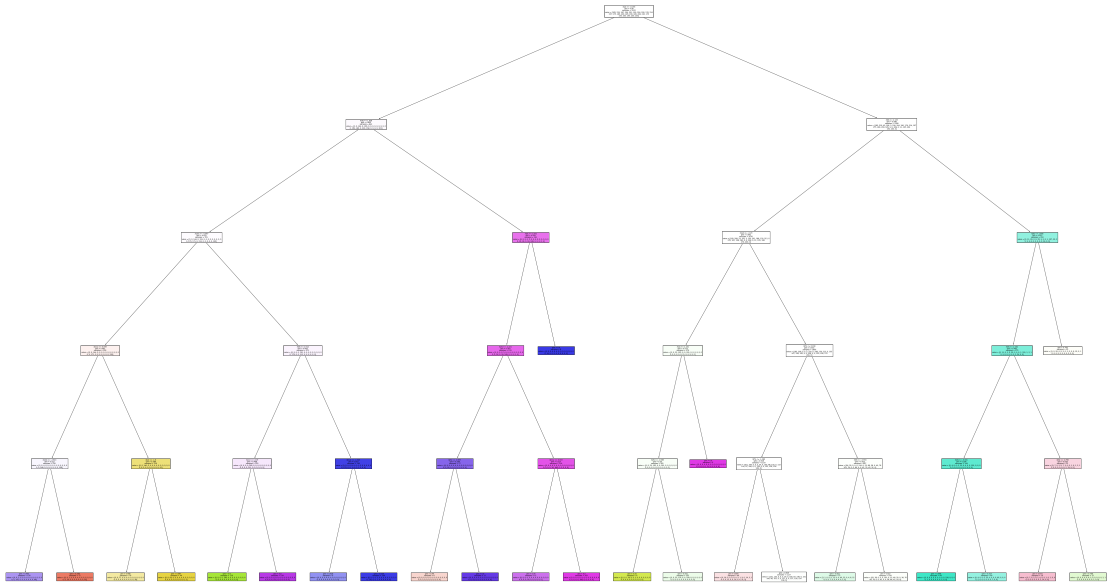
In [40]:

```
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: Ridge score=0.9999960729058915.It has the highest accuracy.

Madrid 2010

In [42]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2010.csv")
a
```

Out[42]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47

209448 rows × 17 columns

In [43]:

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

Out[43]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	Na
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	Na
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	Na
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.4100
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.6700
...	...	...	...	...	...	...	...	...	...	...	
995	2010-03-02 18:00:00	0.51	0.20	0.91	1.27	0.39	20.330000	22.940001	1.42	86.410004	14.2800
996	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	0.13	28.370001	40.669998	NaN	73.480003	Na
997	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	44.029999	50.509998	NaN	NaN	22.0499
998	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	31.770000	37.040001	NaN	85.040001	Na
999	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	32.500000	39.279999	NaN	NaN	16.3500

1000 rows × 17 columns

In [44]:

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

Out[44]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
11	2010-03-01 01:00:00	0.78	0.18	0.84	0.73	0.28	10.420000	11.900000	1.00	90.309998	18.3700
23	2010-03-01 01:00:00	0.70	0.23	1.00	0.73	0.18	17.820000	22.290001	1.00	70.550003	23.6399
35	2010-03-01 02:00:00	0.58	0.17	0.84	0.73	0.28	3.500000	4.950000	1.00	68.849998	5.6000
47	2010-03-01 02:00:00	0.33	0.21	0.84	0.73	0.17	10.810000	14.900000	1.00	74.750000	7.8900
59	2010-03-01 03:00:00	0.38	0.16	0.64	1.00	0.26	2.750000	4.200000	1.00	93.629997	5.1300
...	...	...	...	...	...	...	...	...	...	...	
947	2010-03-02 16:00:00	1.44	0.20	1.86	1.78	0.46	17.090000	20.230000	1.88	91.129997	14.3600
959	2010-03-02 16:00:00	0.64	0.31	1.21	1.78	0.23	28.459999	38.939999	1.88	80.120003	10.7400
971	2010-03-02 17:00:00	0.68	0.18	1.30	1.57	0.43	17.090000	19.990000	1.89	72.089996	13.1000
983	2010-03-02 17:00:00	0.65	0.31	1.39	1.57	0.21	33.500000	45.240002	1.89	73.099998	14.5000
995	2010-03-02 18:00:00	0.51	0.20	0.91	1.27	0.39	20.330000	22.940001	1.42	86.410004	14.2800

83 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	PXY
11	0.78	0.18	0.84	0.73	0.28	10.420000	11.900000	1.00	90.309998	18.370001	0.81
23	0.70	0.23	1.00	0.73	0.18	17.820000	22.290001	1.00	70.550003	23.639999	0.81
35	0.58	0.17	0.84	0.73	0.28	3.500000	4.950000	1.00	68.849998	5.600000	0.81
47	0.33	0.21	0.84	0.73	0.17	10.810000	14.900000	1.00	74.750000	7.890000	0.81
59	0.38	0.16	0.64	1.00	0.26	2.750000	4.200000	1.00	93.629997	5.130000	0.79
...	...	...	...	...	...	...	...	...	...	...	...
947	1.44	0.20	1.86	1.78	0.46	17.090000	20.230000	1.88	91.129997	14.360000	1.84
959	0.64	0.31	1.21	1.78	0.23	28.459999	38.939999	1.88	80.120003	10.740000	1.84
971	0.68	0.18	1.30	1.57	0.43	17.090000	19.990000	1.89	72.089996	13.100000	1.26
983	0.65	0.31	1.39	1.57	0.21	33.500000	45.240002	1.89	73.099998	14.500000	1.26
995	0.51	0.20	0.91	1.27	0.39	20.330000	22.940001	1.42	86.410004	14.280000	0.98

83 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_)
```

```
28079056.293549296
```

In [51]:

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

Out[51]:

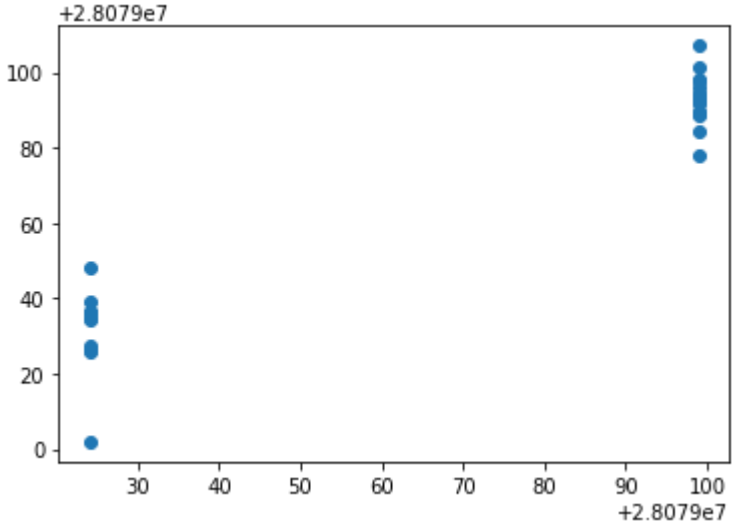
	Co-efficient
BEN	-52.703785
CO	334.102227
EBE	36.961952
MXY	-7.639622
NMHC	-224.077957
NO_2	-0.909075
NOx	0.088937
OXY	18.311069
O_3	-0.177018
PM10	-0.050966

In [52]:

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

Out[52]:

```
<matplotlib.collections.PathCollection at 0x2a95dcdaeb0>
```



In [53]:

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

```
0.9083143136566486
```

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]:

```
0.29214401316710437
```

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]:

```
0.26758908780304247
```

In [58]:

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

Out[58]:

```
ElasticNet()
```

In [59]:

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

```
[-0.94459074  0.          4.63890773  3.05589677 -2.07547586 -4.83341341
  3.61175272  0.51042608 -0.63410204 -1.68189221]
```

In [60]:

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

```
28079113.433931526
```

In [61]:

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

```
0.2465375704662155
```

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)
```

```
[28079024]
```

In [66]:

```
logr.classes_
```

Out[66]:

```
array([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]:

```
0.56
```

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]:

```
1.0
```

In [73]:

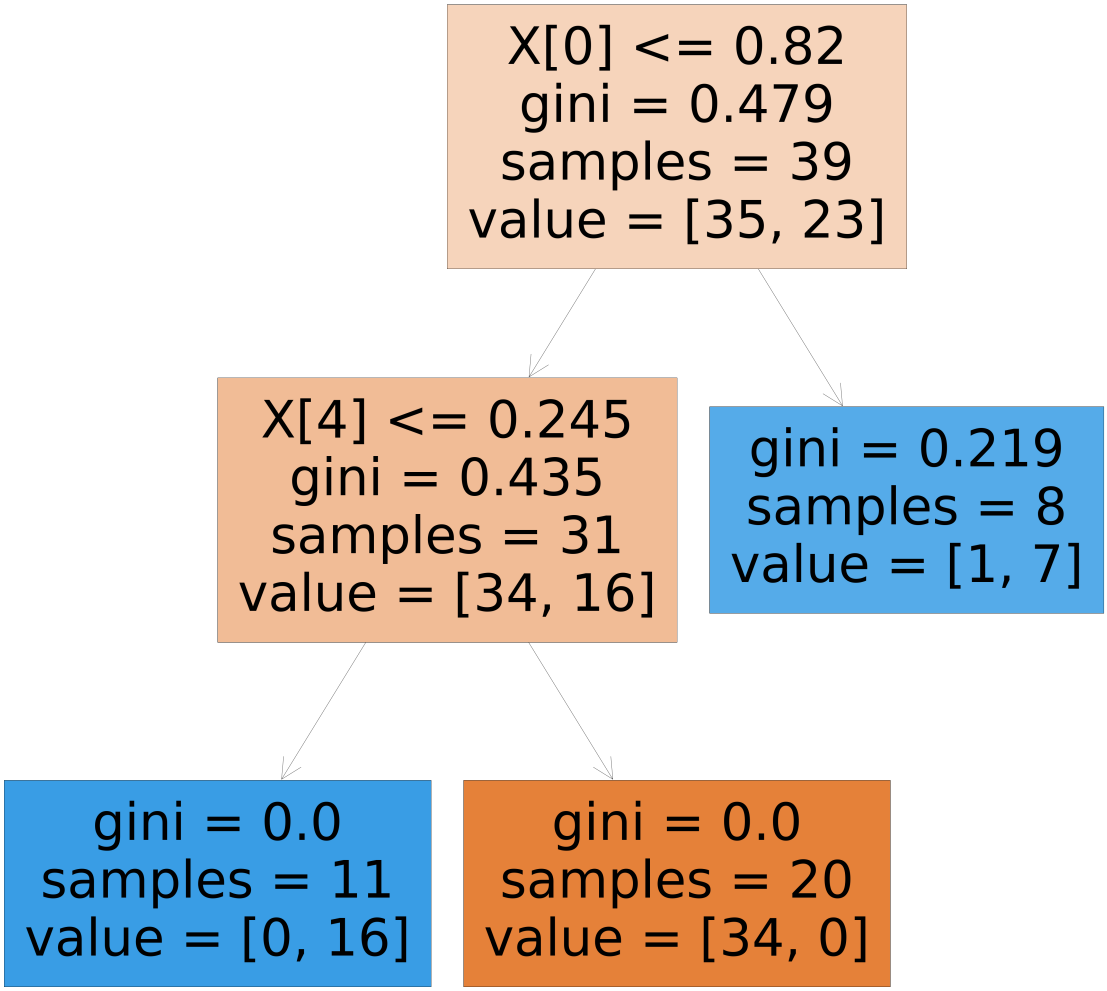
```
rfc_best=grid_search.best_estimator_
```

In [74]:

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

Out[74]:

```
[Text(2678.3999999999996, 3624.0, 'X[0] <= 0.82\ngini = 0.479\nsamples = 3
9\nvalue = [35, 23]'),
 Text(1785.6, 2174.4, 'X[4] <= 0.245\ngini = 0.435\nsamples = 31\nvalue =
[34, 16]'),
 Text(892.8, 724.7999999999997, 'gini = 0.0\nsamples = 11\nvalue = [0, 1
6]'),
 Text(2678.3999999999996, 724.7999999999997, 'gini = 0.0\nsamples = 20\nva
lue = [34, 0]'),
 Text(3571.2, 2174.4, 'gini = 0.219\nsamples = 8\nvalue = [1, 7]')]
```



Conclusion: RandomForest score=1.0. This has the highest accuracy.

Madrid 2011

In [75]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2011.csv")
a
```

Out[75]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOl
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN

209928 rows × 14 columns

In [76]:

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

Out[76]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	2011-11-04 12:00:00	NaN	0.4	NaN	NaN	23.0	52.0	24.0	NaN	NaN	NaN	NaN	NaN
1996	2011-11-04 12:00:00	NaN	NaN	NaN	NaN	15.0	43.0	20.0	NaN	NaN	3.0	NaN	NaN
1997	2011-11-04 12:00:00	0.2	0.5	0.2	NaN	21.0	47.0	18.0	13.0	NaN	2.0	NaN	1.2
1998	2011-11-04 12:00:00	0.7	0.3	2.0	0.16	10.0	35.0	26.0	10.0	7.0	2.0	1.34	3.1
1999	2011-11-04 12:00:00	NaN	NaN	NaN	0.17	7.0	31.0	43.0	NaN	NaN	NaN	1.28	NaN

2000 rows × 14 columns

In [77]:

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

Out[77]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28
6	2011-11-01 01:00:00	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.7	28
25	2011-11-01 02:00:00	1.8	0.3	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	1.71	7.4	28
30	2011-11-01 02:00:00	1.0	0.4	1.3	0.18	31.0	67.0	5.0	25.0	18.0	3.0	1.40	2.9	28
49	2011-11-01 03:00:00	1.3	0.2	2.4	0.22	29.0	72.0	3.0	33.0	20.0	8.0	1.75	6.2	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1950	2011-11-04 10:00:00	1.0	0.4	2.2	0.22	31.0	61.0	10.0	16.0	13.0	3.0	1.78	2.9	28
1969	2011-11-04 11:00:00	1.9	0.3	2.5	0.15	38.0	57.0	16.0	21.0	11.0	8.0	1.31	6.6	28
1974	2011-11-04 11:00:00	0.7	0.3	2.5	0.20	15.0	43.0	19.0	10.0	7.0	2.0	1.76	3.3	28
1993	2011-11-04 12:00:00	1.0	0.3	1.6	0.14	30.0	55.0	24.0	18.0	10.0	7.0	1.40	4.9	28
1998	2011-11-04 12:00:00	0.7	0.3	2.0	0.16	10.0	35.0	26.0	10.0	7.0	2.0	1.34	3.1	28

165 rows × 14 columns

In [78]:

```
b.columns
```

Out[78]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'P
M25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [79]:

```
b=b[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
b
```

Out[79]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
6	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.7	28079024
25	1.8	0.3	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	1.71	7.4	28079008
30	1.0	0.4	1.3	0.18	31.0	67.0	5.0	25.0	18.0	3.0	1.40	2.9	28079024
49	1.3	0.2	2.4	0.22	29.0	72.0	3.0	33.0	20.0	8.0	1.75	6.2	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1950	1.0	0.4	2.2	0.22	31.0	61.0	10.0	16.0	13.0	3.0	1.78	2.9	28079024
1969	1.9	0.3	2.5	0.15	38.0	57.0	16.0	21.0	11.0	8.0	1.31	6.6	28079008
1974	0.7	0.3	2.5	0.20	15.0	43.0	19.0	10.0	7.0	2.0	1.76	3.3	28079024
1993	1.0	0.3	1.6	0.14	30.0	55.0	24.0	18.0	10.0	7.0	1.40	4.9	28079008
1998	0.7	0.3	2.0	0.16	10.0	35.0	26.0	10.0	7.0	2.0	1.34	3.1	28079024

165 rows × 13 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_)
```

```
28079004.011541203
```

In [84]:

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

Out[84]:

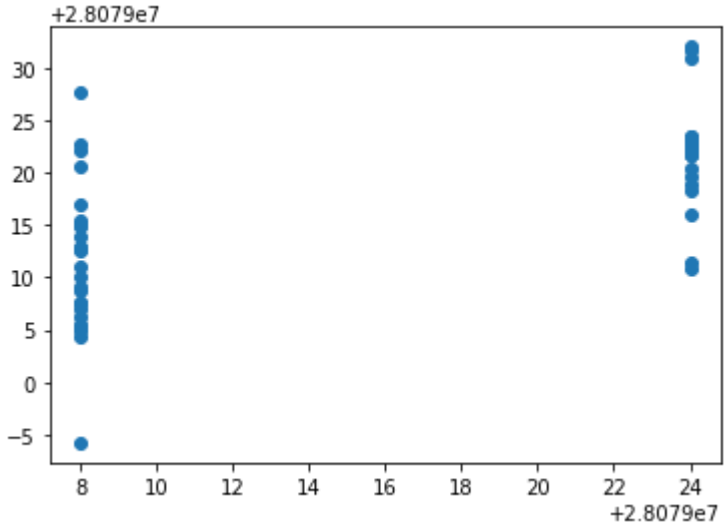
	Co-efficient
BEN	-8.293828
CO	97.736483
EBE	2.596222
NMHC	7.584429
NO	-0.576532

In [85]:

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

Out[85]:

```
<matplotlib.collections.PathCollection at 0x2a95e29cee0>
```



In [86]:

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

```
0.2983438088128415
```

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]:

```
0.10949990447518754
```

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]:

```
0.1342859246108833
```

In [91]:

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

Out[91]:

```
ElasticNet()
```

In [92]:

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

```
[ 0.          0.          0.24960629  0.         -0.24491266]
```

In [93]:

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

```
28079019.37398652
```

In [94]:

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

```
0.11137093683449806
```

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)
```

```
[28079008]
```

In [99]:

```
logr.classes_
```

Out[99]:

```
array([28079008, 28079024], dtype=int64)
```

In [100]:

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

Out[100]:

```
1.0
```

In [101]:

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

Out[101]:

```
0.5
```

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]:

```
0.8867211131276467
```

In [106]:

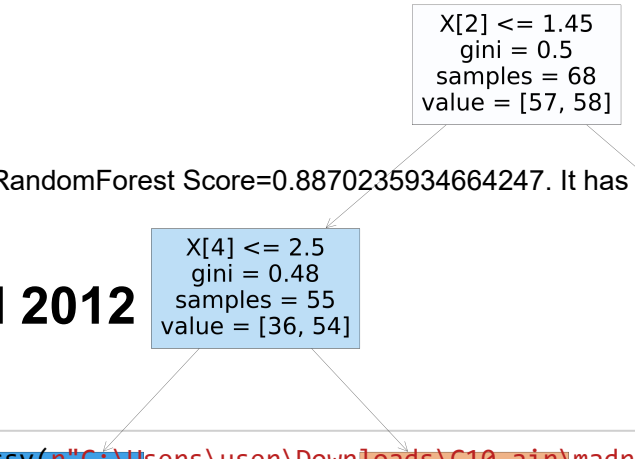
```
rfc_best=grid_search.best_estimator_
```

In [107]:

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

Out[107]:

```
[Text(2637.818181818182, 3913.92, 'X[2] <= 1.45\ngini = 0.5\nsamples = 68
\nvalue = [57, 58]'),
 Text(1623.2727272727273, 3044.1600000000003, 'X[4] <= 2.5\ngini = 0.48\ns
amples = 55\nvalue = [36, 54]'),
 Text(811.6363636363636, 2174.4, 'X[0] <= 0.45\ngini = 0.048\nsamples = 25
\nvalue = [1, 40]'),
 Text(405.8181818181818, 1304.6400000000003, 'gini = 0.165\nsamples = 7\nv
alue = [1, 10]'),
 Text(1217.4545454545455, 1304.6400000000003, 'gini = 0.0\nsamples = 18\nv
alue = [0, 30]'),
 Text(2434.909090909091, 2174.4, 'X[3] <= 0.185\ngini = 0.408\nsamples = 3
0\nvalue = [35, 14]'),
 Text(2029.090909090909, 1304.6400000000003, 'X[4] <= 7.5\ngini = 0.272\ns
amples = 24\nvalue = [31, 6]'),
 Text(1623.2727272727273, 434.8800000000001, 'gini = 0.496\nsamples = 9\nv
alue = [6, 5]'),
 Text(2434.909090909091, 434.8800000000001, 'gini = 0.074\nsamples = 15\nv
alue = [25, 1]'),
 Text(2840.7272727272725, 1304.6400000000003, 'gini = 0.444\nsamples = 6\n
value = [4, 8]'),
 Text(3652.3636363636365, 3044.1600000000003, 'X[1] <= 0.25\ngini = 0.269
\nsamples = 13\nvalue = [21, 4]'),
 Text(3246.5454545454545, 2174.4, 'gini = 0.0\nsamples = 5\nvalue = [8,
0]'),
 Text(4058.181818181818, 2174.4, 'gini = 0.36\nsamples = 8\nvalue = [13,
4]')]
```



Conclusion: RandomForest Score=0.8870235934664247. It has the highest accuracy.

Madrid 2012

In [108]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\madrid_2012.csv")
a
```

Out[108]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	2.0	77.0	18.0	NaN	NaN	2.0	NaN	NaN
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN

210720 rows × 14 columns

In [109]:

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

Out[109]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	2
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	2
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	2
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	2
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1995	2012-09-04 12:00:00	NaN	0.4	NaN	NaN	3.0	10.0	59.0	NaN	NaN	NaN	NaN	NaN	2
1996	2012-09-04 12:00:00	NaN	NaN	NaN	NaN	4.0	15.0	61.0	NaN	NaN	3.0	NaN	NaN	2
1997	2012-09-04 12:00:00	0.1	0.3	1.0	NaN	7.0	19.0	57.0	20.0	NaN	1.0	NaN	0.7	2
1998	2012-09-04 12:00:00	0.5	0.2	1.2	0.24	3.0	12.0	63.0	13.0	8.0	3.0	1.35	1.2	2
1999	2012-09-04 12:00:00	NaN	NaN	NaN	0.11	1.0	6.0	78.0	NaN	NaN	NaN	1.18	NaN	2

2000 rows × 14 columns

In [110]:

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

Out[110]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6	28(
30	2012-09-01 02:00:00	0.4	0.2	0.7	0.24	1.0	5.0	55.0	5.0	5.0	2.0	1.33	0.5	28(
54	2012-09-01 03:00:00	0.4	0.2	0.7	0.24	1.0	4.0	56.0	6.0	4.0	2.0	1.33	0.5	28(
78	2012-09-01 04:00:00	0.3	0.2	0.7	0.25	1.0	5.0	54.0	6.0	5.0	2.0	1.34	0.4	28(
102	2012-09-01 05:00:00	0.4	0.2	0.7	0.24	1.0	3.0	53.0	8.0	5.0	2.0	1.33	0.5	28(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1902	2012-09-04 08:00:00	0.4	0.2	0.8	0.23	1.0	13.0	51.0	7.0	8.0	2.0	1.34	0.4	28(
1926	2012-09-04 09:00:00	0.6	0.2	0.9	0.24	2.0	21.0	46.0	12.0	8.0	2.0	1.35	0.6	28(
1950	2012-09-04 10:00:00	0.6	0.2	1.0	0.24	4.0	19.0	47.0	13.0	9.0	2.0	1.35	0.9	28(
1974	2012-09-04 11:00:00	0.5	0.2	1.2	0.25	4.0	15.0	54.0	15.0	9.0	2.0	1.36	1.2	28(
1998	2012-09-04 12:00:00	0.5	0.2	1.2	0.24	3.0	12.0	63.0	13.0	8.0	3.0	1.35	1.2	28(

84 rows × 14 columns

In [111]:

```
b.columns
```

Out[111]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'P
M25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [112]:

```
c=b[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
d=b['station']
```

In [117]:

```
x=c.iloc[:,0:11]
y=c.iloc[:,-1]
```

In [118]:

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

In [119]:

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

Out[119]:

```
LinearRegression()
```

In [120]:

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

```
0.698411509777724
```

In [121]:

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

Out[121]:

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

In [122]:

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

Out[122]:

```
0.44872555626777366
```

In [123]:

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

Out[123]:

```
Lasso()
```

In [124]:

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

Out[124]:

```
-0.150303495066209
```

In [125]:

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

Out[125]:

```
ElasticNet()
```

In [126]:

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

```
-0.14983006866203818
```

In [139]:

```
x1=c.drop('NO',axis=1)
y1=c['NO']
```

In [170]:

```
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.4,random_state=40)
```

In [171]:

```
f=StandardScaler().fit_transform(x1)
```

In [172]:

```
logr=LogisticRegression(max_iter=10000)
logr.fit(x1,y1)
```

Out[172]:

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

In [173]:

```
g=[[10,20,30,40,50,60,70,80,90,100,110]]
```

In [174]:

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

```
[3.]
```

In [175]:

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

Out[175]:

```
1.1971510187584438e-121
```

In [177]:

```
logr.score(x1_test,y1_test)
```

Out[177]:

```
0.9117647058823529
```

In [178]:

```
rfc=RandomForestClassifier()
rfc.fit(x1_train,y1_train)
```

Out[178]:

```
RandomForestClassifier()
```

In [179]:

```
parameters={'max_depth':[1,2,3,4,5,6],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
            }
```

In [180]:

```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x1_train,y1_train)
```

Out[180]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [181]:

```
grid_search.best_score_
```

Out[181]:

```
0.78
```

In [182]:

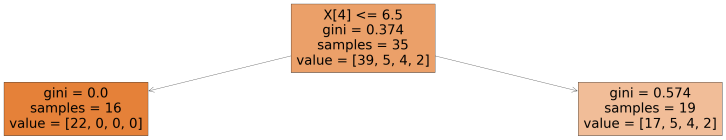
```
rfc_best=grid_search.best_estimator_
```

In [183]:

```
plt.figure(figsize=(80,10))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[183]:

```
[Text(2232.0, 407.70000000000005, 'X[4] <= 6.5\ngini = 0.374\nsamples = 35
\nvalue = [39, 5, 4, 2]'),
 Text(1116.0, 135.89999999999998, 'gini = 0.0\nsamples = 16\nvalue = [22,
0, 0, 0]'),
 Text(3348.0, 135.89999999999998, 'gini = 0.574\nsamples = 19\nvalue = [1
7, 5, 4, 2]')]
```



Conclusion: Logistic Score=0.9117647058823529. It has the highest accuracy.

In [ ]: