In [174]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
```

In [392]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2009
a
```

Out[392]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2009-10-01 01:00:00 | NaN | 0.27 | NaN | NaN | NaN | 39.889999 | 48.150002 | NaN | 50.680000 | 18.260000 | NaN |
| **1** | 2009-10-01 01:00:00 | NaN | 0.22 | NaN | NaN | NaN | 21.230000 | 24.260000 | NaN | 55.880001 | 10.580000 | NaN |
| **2** | 2009-10-01 01:00:00 | NaN | 0.18 | NaN | NaN | NaN | 31.230000 | 34.880001 | NaN | 49.060001 | 25.190001 | NaN |
| **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 | 26.530001 | 6.82 |
| **4** | 2009-10-01 01:00:00 | NaN | 0.41 | NaN | NaN | 0.12 | 61.349998 | 76.260002 | NaN | 38.090000 | 23.760000 | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 | 10.830000 | 7.15 |
| **215684** | 2009-06-01 00:00:00 | NaN | 0.31 | NaN | NaN | NaN | 76.110001 | 101.099998 | NaN | 41.220001 | 9.920000 | NaN |
| **215685** | 2009-06-01 00:00:00 | 0.13 | NaN | 0.86 | NaN | 0.23 | 81.050003 | 99.849998 | NaN | 24.830000 | 12.460000 | 6.77 |
| **215686** | 2009-06-01 00:00:00 | 0.21 | NaN | 2.96 | NaN | 0.10 | 72.419998 | 82.959999 | NaN | NaN | 13.030000 | NaN |
| **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 | 15.360000 | 11.61 |

215688 rows × 17 columns

In [393]:
```
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 [394]:
```
b=a.fillna(value=67)
b
```

Out[394]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2009-10-01 01:00:00 | 67.00 | 0.27 | 67.00 | 67.00 | 67.00 | 39.889999 | 48.150002 | 67.00 | 50.680000 | 18.260000 | 6 |
| **1** | 2009-10-01 01:00:00 | 67.00 | 0.22 | 67.00 | 67.00 | 67.00 | 21.230000 | 24.260000 | 67.00 | 55.880001 | 10.580000 | 6 |
| **2** | 2009-10-01 01:00:00 | 67.00 | 0.18 | 67.00 | 67.00 | 67.00 | 31.230000 | 34.880001 | 67.00 | 49.060001 | 25.190001 | 6 |
| **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 | 26.530001 | |
| **4** | 2009-10-01 01:00:00 | 67.00 | 0.41 | 67.00 | 67.00 | 0.12 | 61.349998 | 76.260002 | 67.00 | 38.090000 | 23.760000 | 6 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **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 | 10.830000 | |
| **215684** | 2009-06-01 00:00:00 | 67.00 | 0.31 | 67.00 | 67.00 | 67.00 | 76.110001 | 101.099998 | 67.00 | 41.220001 | 9.920000 | 6 |
| **215685** | 2009-06-01 00:00:00 | 0.13 | 67.00 | 0.86 | 67.00 | 0.23 | 81.050003 | 99.849998 | 67.00 | 24.830000 | 12.460000 | |
| **215686** | 2009-06-01 00:00:00 | 0.21 | 67.00 | 2.96 | 67.00 | 0.10 | 72.419998 | 82.959999 | 67.00 | 67.000000 | 13.030000 | 6 |
| **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 | 15.360000 | 1 |

215688 rows × 17 columns

In [395]:
```
b.columns
```

Out[395]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [396]:
```
c=b.head(10)
c
```

Out[396]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 | PX |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2009-10-01 01:00:00 | 67.00 | 0.27 | 67.00 | 67.00 | 67.00 | 39.889999 | 48.150002 | 67.00 | 50.680000 | 18.260000 | 67.00 | 67 |
| **1** | 2009-10-01 01:00:00 | 67.00 | 0.22 | 67.00 | 67.00 | 67.00 | 21.230000 | 24.260000 | 67.00 | 55.880001 | 10.580000 | 67.00 | 67 |
| **2** | 2009-10-01 01:00:00 | 67.00 | 0.18 | 67.00 | 67.00 | 67.00 | 31.230000 | 34.880001 | 67.00 | 49.060001 | 25.190001 | 67.00 | 67 |
| **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 | 26.530001 | 6.82 | 1 |
| **4** | 2009-10-01 01:00:00 | 67.00 | 0.41 | 67.00 | 67.00 | 0.12 | 61.349998 | 76.260002 | 67.00 | 38.090000 | 23.760000 | 67.00 | 67 |
| **5** | 2009-10-01 01:00:00 | 67.00 | 0.29 | 67.00 | 67.00 | 67.00 | 43.200001 | 50.080002 | 67.00 | 35.840000 | 21.870001 | 67.00 | 67 |
| **6** | 2009-10-01 01:00:00 | 67.00 | 0.20 | 67.00 | 67.00 | 67.00 | 35.430000 | 38.520000 | 67.00 | 33.549999 | 17.350000 | 67.00 | 67 |
| **7** | 2009-10-01 01:00:00 | 67.00 | 0.15 | 67.00 | 67.00 | 67.00 | 27.309999 | 33.150002 | 67.00 | 53.549999 | 16.520000 | 11.99 | 67 |
| **8** | 2009-10-01 01:00:00 | 67.00 | 0.21 | 67.00 | 67.00 | 0.39 | 33.889999 | 40.799999 | 67.00 | 58.549999 | 16.650000 | 67.00 | 67 |
| **9** | 2009-10-01 01:00:00 | 67.00 | 0.32 | 67.00 | 67.00 | 67.00 | 46.349998 | 60.540001 | 67.00 | 45.340000 | 15.160000 | 67.00 | 67 |

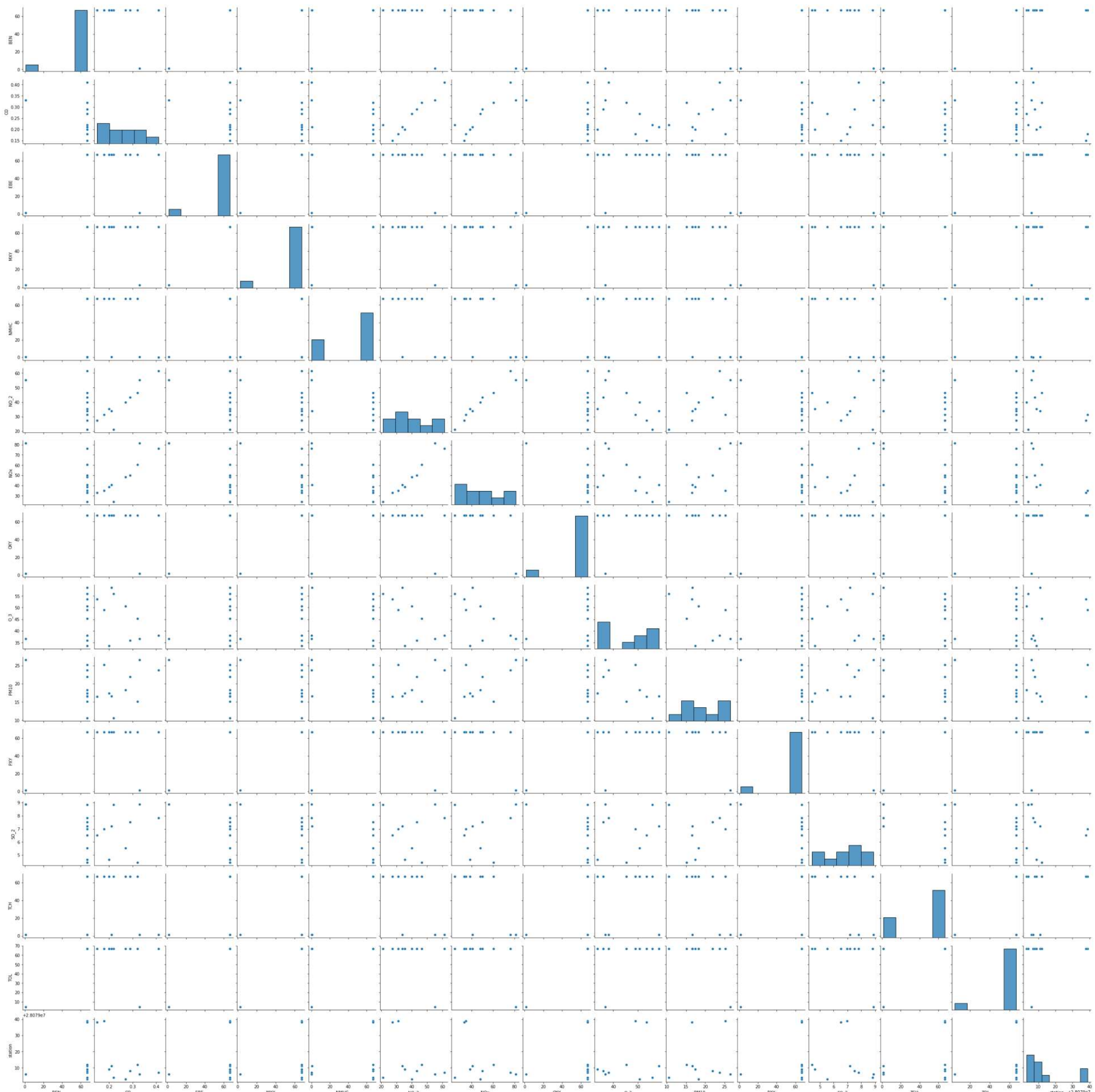
In [397]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[397]:

| | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PXY | SO_2 | TCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 67.00 | 0.27 | 67.00 | 67.00 | 67.00 | 39.889999 | 48.150002 | 67.00 | 50.680000 | 18.260000 | 67.0 | 5.55 | 67.00 |
| **1** | 67.00 | 0.22 | 67.00 | 67.00 | 67.00 | 21.230000 | 24.260000 | 67.00 | 55.880001 | 10.580000 | 67.0 | 8.84 | 67.00 |
| **2** | 67.00 | 0.18 | 67.00 | 67.00 | 67.00 | 31.230000 | 34.880001 | 67.00 | 49.060001 | 25.190001 | 67.0 | 6.98 | 67.00 |
| **3** | 0.95 | 0.33 | 1.43 | 2.68 | 0.25 | 55.180000 | 81.360001 | 1.57 | 36.669998 | 26.530001 | 1.3 | 8.88 | 1.38 |
| **4** | 67.00 | 0.41 | 67.00 | 67.00 | 0.12 | 61.349998 | 76.260002 | 67.00 | 38.090000 | 23.760000 | 67.0 | 7.82 | 1.41 |
| **5** | 67.00 | 0.29 | 67.00 | 67.00 | 67.00 | 43.200001 | 50.080002 | 67.00 | 35.840000 | 21.870001 | 67.0 | 7.51 | 67.00 |
| **6** | 67.00 | 0.20 | 67.00 | 67.00 | 67.00 | 35.430000 | 38.520000 | 67.00 | 33.549999 | 17.350000 | 67.0 | 4.65 | 67.00 |
| **7** | 67.00 | 0.15 | 67.00 | 67.00 | 67.00 | 27.309999 | 33.150002 | 67.00 | 53.549999 | 16.520000 | 67.0 | 6.52 | 67.00 |
| **8** | 67.00 | 0.21 | 67.00 | 67.00 | 0.39 | 33.889999 | 40.799999 | 67.00 | 58.549999 | 16.650000 | 67.0 | 7.20 | 1.39 |
| **9** | 67.00 | 0.32 | 67.00 | 67.00 | 67.00 | 46.349998 | 60.540001 | 67.00 | 45.340000 | 15.160000 | 67.0 | 4.43 | 67.00 |

In [398]:
```
sns.pairplot(d)
```

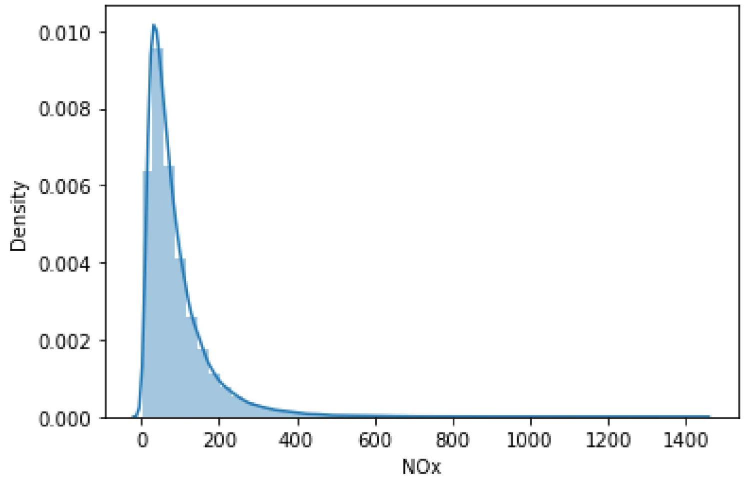
Out[398]: <seaborn.axisgrid.PairGrid at 0x1b6d04385b0>



In [399]:
```
sns.distplot(a['NOx'])
```

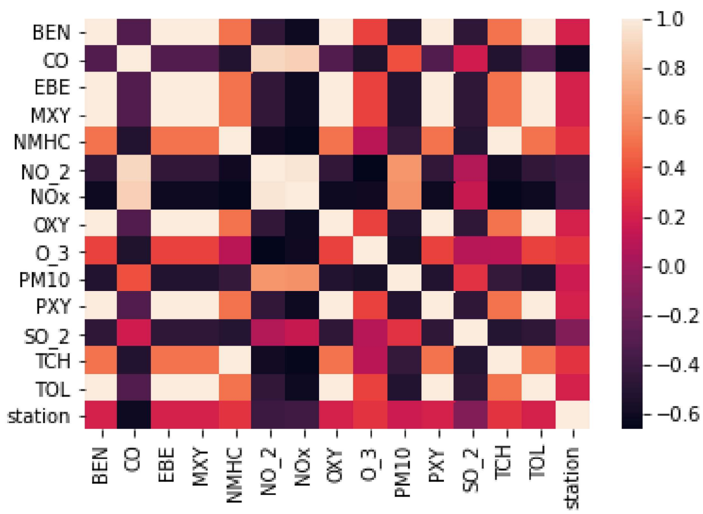
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[399]: <AxesSubplot:xlabel='NOx', ylabel='Density'>




In [400]:
```
sns.heatmap(d.corr())
```

Out[400]: <AxesSubplot:>




In [401]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [402]:
```
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 [403]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[403]: LinearRegression()

In [404]:
```
print(lr.intercept_)
```

1.1308422356275756

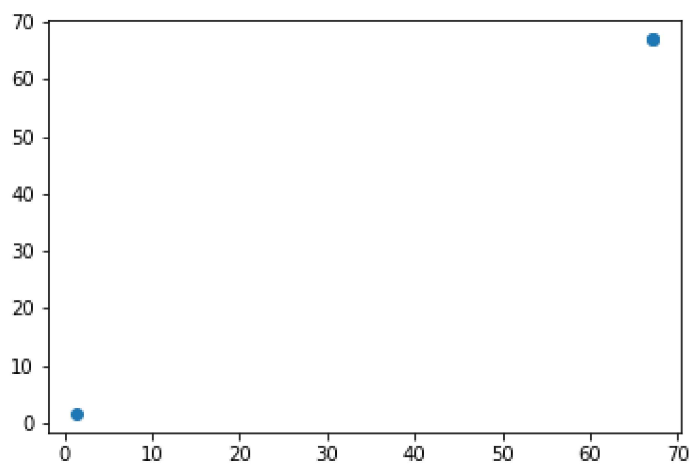
In [405]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[405]:

| | Co-efficient |
|---|---|
| **BEN** | 6.090324e-04 |
| **CO** | 1.094106e-13 |
| **EBE** | 6.046064e-04 |
| **MXY** | 5.930804e-04 |
| **NMHC** | 9.807117e-01 |
| **NO_2** | -4.873921e-15 |
| **NOx** | 3.422083e-15 |
| **OXY** | 6.033155e-04 |

In [406]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[406]: <matplotlib.collections.PathCollection at 0x1b6e6638070>




In [407]:
```
print(lr.score(x_test,y_test))
```

0.9999717377219167

In [408]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [409]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[409]: Ridge(alpha=10)

In [410]:
```
rr.score(x_test,y_test)
```

Out[410]: 0.9997050007891867

In [411]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[411]: Lasso(alpha=10)

In [412]:
```
la.score(x_test,y_test)
```

Out[412]: 0.9997847425288982

In [413]:
```
a1=b.head(7000)
a1
```

Out[413]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2009-10-01 01:00:00 | 67.00 | 0.27 | 67.00 | 67.00 | 67.00 | 39.889999 | 48.150002 | 67.00 | 50.680000 | 18.260000 | 67.0( |
| **1** | 2009-10-01 01:00:00 | 67.00 | 0.22 | 67.00 | 67.00 | 67.00 | 21.230000 | 24.260000 | 67.00 | 55.880001 | 10.580000 | 67.0( |
| **2** | 2009-10-01 01:00:00 | 67.00 | 0.18 | 67.00 | 67.00 | 67.00 | 31.230000 | 34.880001 | 67.00 | 49.060001 | 25.190001 | 67.0( |
| **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 | 26.530001 | 6.82 |
| **4** | 2009-10-01 01:00:00 | 67.00 | 0.41 | 67.00 | 67.00 | 0.12 | 61.349998 | 76.260002 | 67.00 | 38.090000 | 23.760000 | 67.0( |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. |
| **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 | 11.110000 | 2.89 |
| **6996** | 2009-10-12 16:00:00 | 67.00 | 0.23 | 67.00 | 67.00 | 67.00 | 33.090000 | 54.380001 | 67.00 | 57.480000 | 16.969999 | 67.0( |
| **6997** | 2009-10-12 16:00:00 | 0.13 | 67.00 | 0.31 | 67.00 | 0.19 | 27.670000 | 36.860001 | 67.00 | 56.240002 | 19.820000 | 9.27 |
| **6998** | 2009-10-12 16:00:00 | 0.20 | 67.00 | 1.00 | 67.00 | 0.13 | 16.459999 | 30.200001 | 67.00 | 67.000000 | 30.650000 | 67.0( |
| **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 | 11.070000 | 4.18 |

7000 rows × 17 columns

In [414]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [415]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [416]:
```
h=StandardScaler().fit_transform(f)
```

In [417]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[417]:
```
LogisticRegression(max_iter=10000)
```

In [418]:
```
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 [419]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [420]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079021]
```

In [421]:
```
logr.classes_
```

Out[421]:
```
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 [422]:
```
logr.predict_proba(i)[0][0]
```

Out[422]:
```
1.212065964534134e-142
```

In [423]:
```
logr.predict_proba(i)[0][1]
```

Out[423]:
```
2.903111278711799e-27
```

In [424]:
```
logr.score(h_test,g_test)
```

Out[424]:
```
0.5447619047619048
```

In [425]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[425]:
```
ElasticNet()
```

In [426]:
```
print(en.coef_)
```

```
[ 8.91912983e-06 -0.00000000e+00  0.00000000e+00  2.16593709e-03
  9.78783085e-01 -0.00000000e+00 -1.54893464e-03  8.04798286e-05]
```

In [427]:
```
print(en.intercept_)
```

```
1.3088825964199913
```

In [428]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999461782661847
```

In [429]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[429]:
```
RandomForestClassifier()
```

In [430]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [431]:
```
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)
```

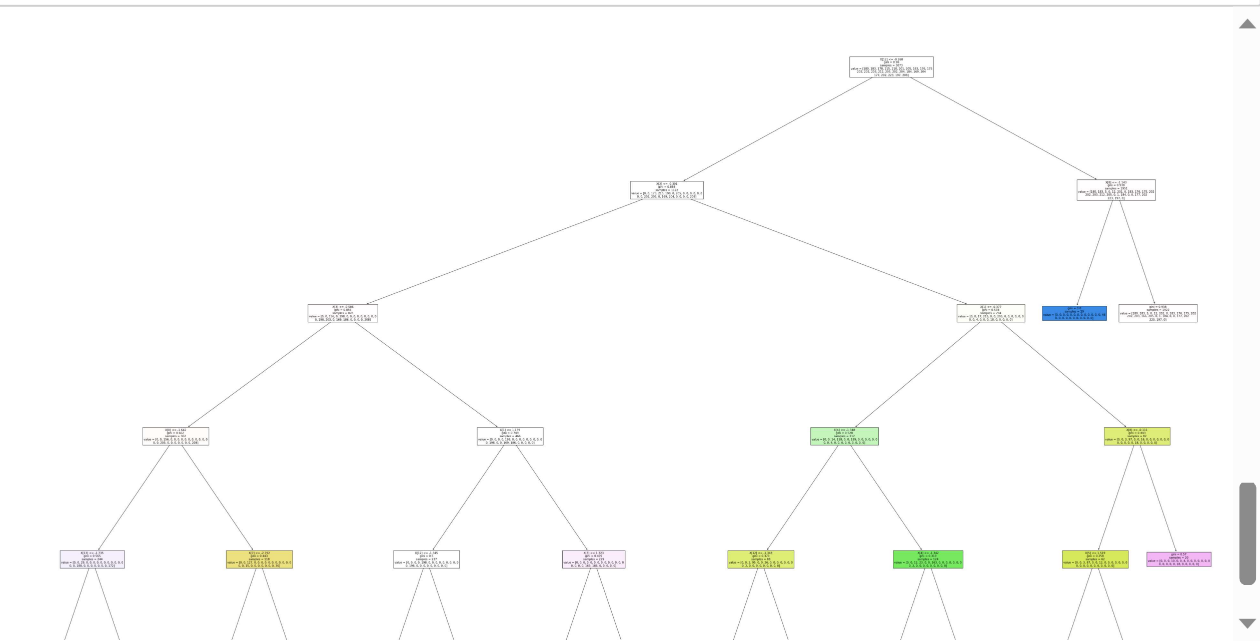
Out[431]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [432]:
```
grid_search.best_score_
```

Out[432]:
```
0.5820408163265306
```

In [433]:
```
rfc_best=grid_search.best_estimator_
```

In [434]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[20],filled=True)
```



# Conclusion: from this data set i observed that the LINEAR REGRESSION has the highest accuracy of 0.999705000789186

In [ ]:
```
0.999705000789186
```