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 [435]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_201
a
```

Out[435]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	NaN	
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	NaN	
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	NaN	
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410000	7.87
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670000	22.03
...	...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	NaN	
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259998	
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	NaN	
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	NaN	
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150002	26.86

209448 rows × 17 columns

In [436]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209448 entries, 0 to 209447
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209448 non-null  object 
 1   BEN      60268 non-null   float64
 2   CO       94982 non-null   float64
 3   EBE      60253 non-null   float64
 4   MXY      6750 non-null    float64
 5   NMHC     51727 non-null   float64
 6   NO_2     208219 non-null  float64
 7   NOx      208210 non-null  float64
 8   OXY      6750 non-null    float64
 9   O_3      126684 non-null  float64
 10  PM10     106186 non-null  float64
 11  PM25     55514 non-null   float64
 12  PXY      6740 non-null    float64
 13  SO_2     93184 non-null   float64
 14  TCH      51730 non-null   float64
 15  TOL      60171 non-null   float64
 16  station  209448 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 27.2+ MB
```

In [437]:
```
b=a.fillna(value=67)
b
```

Out[437]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2010-03-01 01:00:00	67.00	0.29	67.00	67.0	67.00	25.090000	29.219999	67.0	68.930000	67.000000	67
1	2010-03-01 01:00:00	67.00	0.27	67.00	67.0	67.00	24.879999	30.040001	67.0	67.000000	67.000000	67
2	2010-03-01 01:00:00	67.00	0.28	67.00	67.0	67.00	17.410000	20.540001	67.0	72.120003	67.000000	67
3	2010-03-01 01:00:00	0.38	0.24	1.74	67.0	0.05	15.610000	21.080000	67.0	72.970001	19.410000	7
4	2010-03-01 01:00:00	0.79	67.00	1.32	67.0	67.00	21.430000	26.070000	67.0	67.000000	24.670000	22
...	...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	67.00	0.55	67.00	67.0	67.00	125.000000	219.899994	67.0	25.379999	67.000000	67
209444	2010-08-01 00:00:00	67.00	0.27	67.00	67.0	67.00	45.709999	47.410000	67.0	67.000000	51.259998	67
209445	2010-08-01 00:00:00	67.00	67.00	67.00	67.0	0.24	46.560001	49.040001	67.0	46.250000	67.000000	67
209446	2010-08-01 00:00:00	67.00	67.00	67.00	67.0	67.00	46.770000	50.119999	67.0	77.709999	67.000000	67
209447	2010-08-01 00:00:00	0.92	0.43	0.71	67.0	0.25	76.330002	88.190002	67.0	52.259998	47.150002	26

209448 rows × 17 columns

In [438]:
```
b.columns
```

Out[438]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [439]:
```
c=b.head(10)
c
```

Out[439]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25
0	2010-03-01 01:00:00	67.00	0.29	67.00	67.0	67.00	25.090000	29.219999	67.0	68.930000	67.000000	67.000000
1	2010-03-01 01:00:00	67.00	0.27	67.00	67.0	67.00	24.879999	30.040001	67.0	67.000000	67.000000	67.000000
2	2010-03-01 01:00:00	67.00	0.28	67.00	67.0	67.00	17.410000	20.540001	67.0	72.120003	67.000000	67.000000
3	2010-03-01 01:00:00	0.38	0.24	1.74	67.0	0.05	15.610000	21.080000	67.0	72.970001	19.410000	7.870000
4	2010-03-01 01:00:00	0.79	67.00	1.32	67.0	67.00	21.430000	26.070000	67.0	67.000000	24.670000	22.030001
5	2010-03-01 01:00:00	0.56	67.00	0.58	67.0	67.00	21.370001	25.870001	67.0	67.000000	67.000000	67.000000
6	2010-03-01 01:00:00	67.00	67.00	67.00	67.0	67.00	16.660000	25.230000	67.0	67.000000	39.799999	67.000000
7	2010-03-01 01:00:00	67.00	0.23	67.00	67.0	67.00	17.799999	21.639999	67.0	55.880001	67.000000	67.000000
8	2010-03-01 01:00:00	67.00	67.00	67.00	67.0	67.00	12.050000	14.870000	67.0	57.369999	67.000000	67.000000
9	2010-03-01 01:00:00	1.48	0.18	0.51	67.0	67.00	16.780001	21.680000	67.0	78.660004	21.969999	67.000000

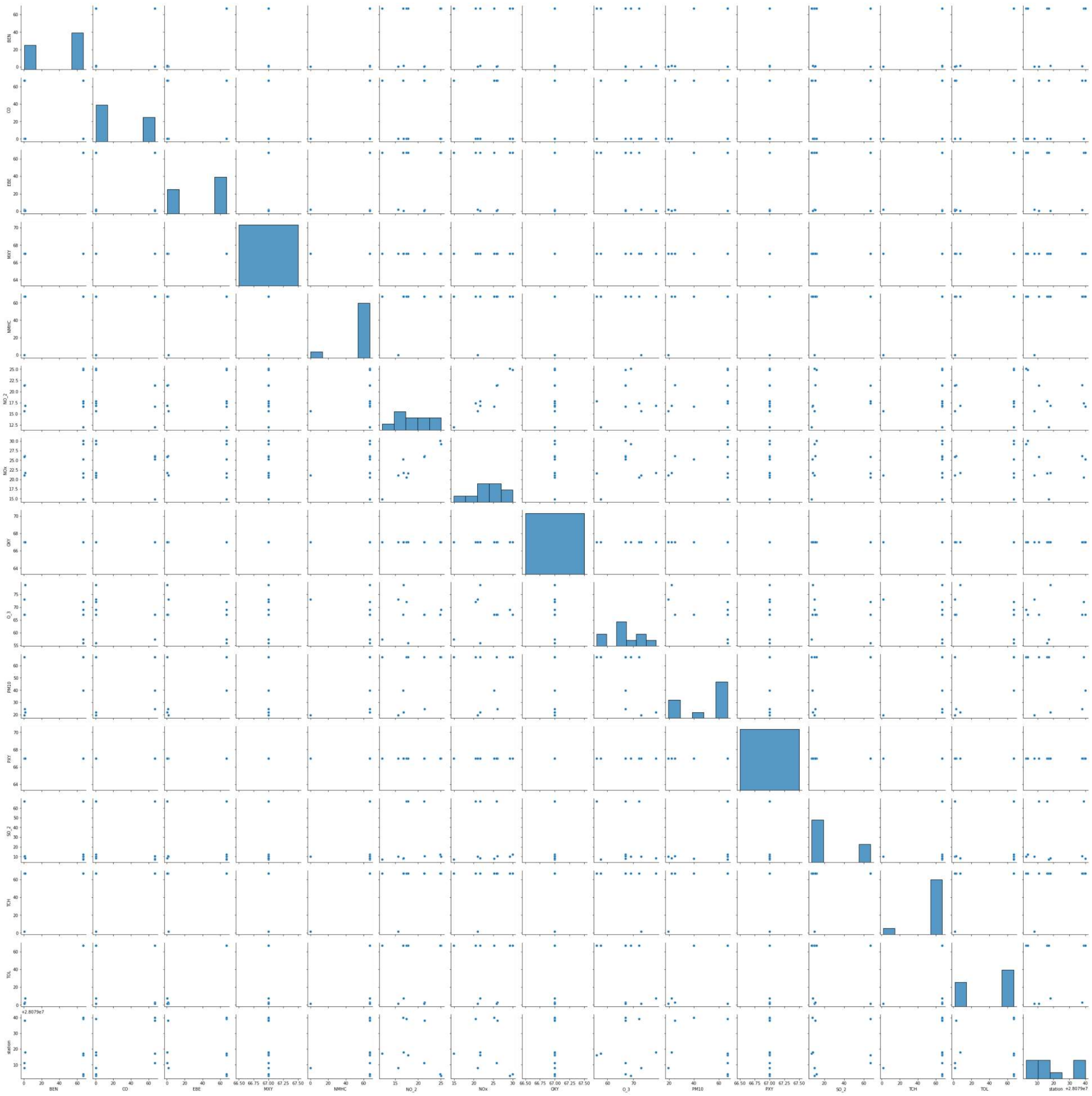
In [440]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[440]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH
0	67.00	0.29	67.00	67.0	67.00	25.090000	29.219999	67.0	68.930000	67.000000	67.0	10.15	67.00
1	67.00	0.27	67.00	67.0	67.00	24.879999	30.040001	67.0	67.000000	67.000000	67.0	12.24	67.00
2	67.00	0.28	67.00	67.0	67.00	17.410000	20.540001	67.0	72.120003	67.000000	67.0	67.00	67.00
3	0.38	0.24	1.74	67.0	0.05	15.610000	21.080000	67.0	72.970001	19.410000	67.0	10.06	1.52
4	0.79	67.00	1.32	67.0	67.00	21.430000	26.070000	67.0	67.000000	24.670000	67.0	10.68	67.00
5	0.56	67.00	0.58	67.0	67.00	21.370001	25.870001	67.0	67.000000	67.000000	67.0	67.00	67.00
6	67.00	67.00	67.00	67.0	67.00	16.660000	25.230000	67.0	67.000000	39.799999	67.0	7.80	67.00
7	67.00	0.23	67.00	67.0	67.00	17.799999	21.639999	67.0	55.880001	67.000000	67.0	67.00	67.00
8	67.00	67.00	67.00	67.0	67.00	12.050000	14.870000	67.0	57.369999	67.000000	67.0	7.06	67.00
9	1.48	0.18	0.51	67.0	67.00	16.780001	21.680000	67.0	78.660004	21.969999	67.0	8.28	67.00

In [441]:
```
sns.pairplot(d)
```

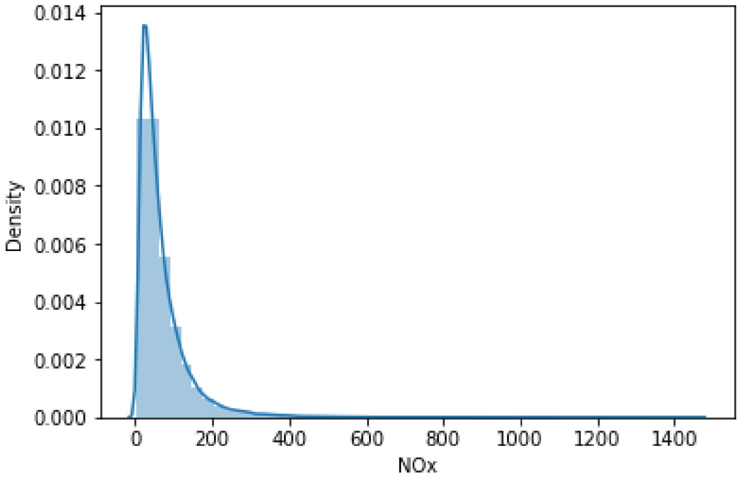
Out[441]:
```
<seaborn.axisgrid.PairGrid at 0x1b6e49fa070>
```



In [442]:
```
sns.distplot(a['NOx'])
```

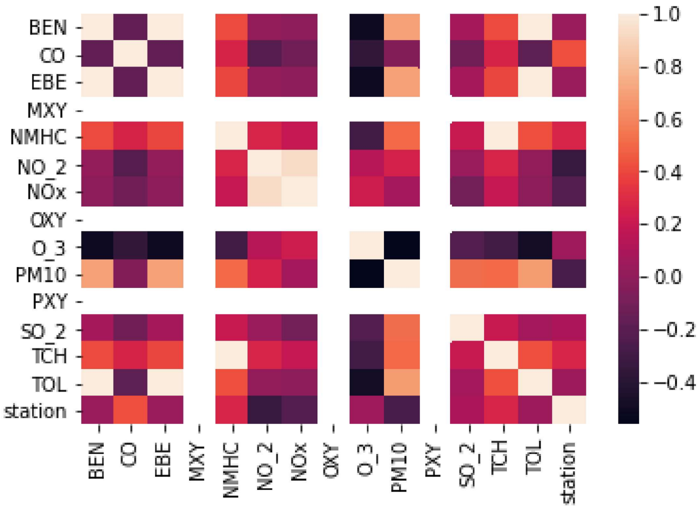
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[442]:
```
<AxesSubplot:xlabel='NOx', ylabel='Density'>
```



In [443]:
```
sns.heatmap(d.corr())
```

Out[443]:
```
<AxesSubplot:>
```



In [444]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [445]:
```
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 [446]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[446]:
```
LinearRegression()
```

In [447]:
```
print(lr.intercept_)
```

```
1.4710978143876048
```

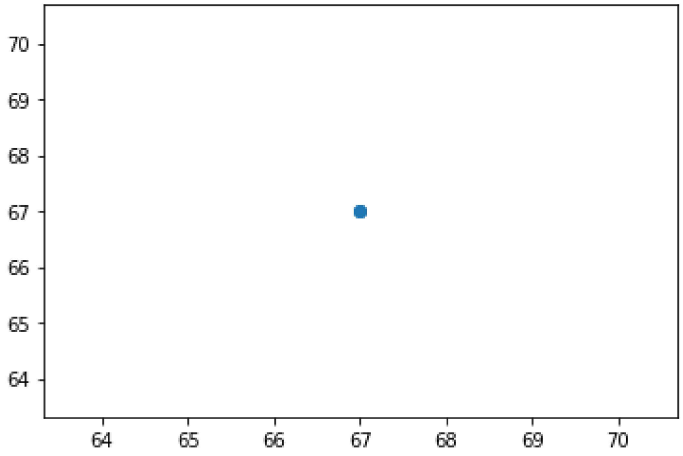
In [448]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[448]:

	Co-efficient
BEN	-2.945889e-14
CO	-1.879010e-16
EBE	2.932495e-14
MXY	0.000000e+00
NMHC	9.780433e-01
NO_2	2.978164e-16
NOx	-5.110453e-16
OXY	0.000000e+00

In [449]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[449]:
```
<matplotlib.collections.PathCollection at 0x1b6fb84c3d0>
```



In [450]:
```
print(lr.score(x_test,y_test))
```

```
0.0
```

In [451]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [452]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[452]:
```
Ridge(alpha=10)
```

In [453]:
```
rr.score(x_test,y_test)
```

Out[453]:
```
0.0
```

In [454]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[454]:
```
Lasso(alpha=10)
```

In [455]:
```
la.score(x_test,y_test)
```

Out[455]:
```
0.0
```

In [456]:
```
a1=b.head(7000)
a1
```

Out[456]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2010-03-01 01:00:00	67.00	0.29	67.00	67.00	67.00	25.090000	29.219999	67.00	68.930000	67.000000	67.0
1	2010-03-01 01:00:00	67.00	0.27	67.00	67.00	67.00	24.879999	30.040001	67.00	67.000000	67.000000	67.0
2	2010-03-01 01:00:00	67.00	0.28	67.00	67.00	67.00	17.410000	20.540001	67.00	72.120003	67.000000	67.0
3	2010-03-01 01:00:00	0.38	0.24	1.74	67.00	0.05	15.610000	21.080000	67.00	72.970001	19.410000	7.87
4	2010-03-01 01:00:00	0.79	67.00	1.32	67.00	67.00	21.430000	26.070000	67.00	67.000000	24.670000	22.03
...	...	...	...	...	...	...	...	...	...	...	...	
6995	2010-03-13 06:00:00	0.69	0.26	0.47	0.53	0.23	40.490002	42.220001	0.84	22.170000	15.860000	13.44
6996	2010-03-13 06:00:00	67.00	67.00	67.00	67.00	0.09	52.590000	66.339996	67.00	23.850000	67.000000	67.0
6997	2010-03-13 06:00:00	67.00	67.00	67.00	67.00	67.00	41.950001	44.310001	67.00	67.000000	20.950001	15.58
6998	2010-03-13 06:00:00	67.00	67.00	67.00	67.00	67.00	27.459999	30.540001	67.00	47.369999	67.000000	67.0
6999	2010-03-13 06:00:00	67.00	67.00	67.00	67.00	67.00	36.830002	42.049999	67.00	67.000000	15.720000	12.73

7000 rows × 17 columns

In [457]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [458]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

```
In [459]: h=StandardScaler().fit_transform(f)
```

```
In [460]: logr=LogisticRegression(max_iter=10000)
          logr.fit(h,g)

Out[460]: LogisticRegression(max_iter=10000)
```

```
In [461]: 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 [462]: i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

```
In [463]: prediction=logr.predict(i)
          print(prediction)

          [28079056]
```

```
In [464]: logr.classes_

Out[464]: array([28079003, 28079004, 28079008, 28079011, 28079016, 28079017,
                 28079018, 28079024, 28079027, 28079036, 28079038, 28079039,
                 28079040, 28079047, 28079049, 28079050, 28079054, 28079055,
                 28079056, 28079057, 28079058, 28079059, 28079060, 28079099],
                dtype=int64)
```

```
In [465]: logr.predict_proba(i)[0][0]

Out[465]: 3.18903840427058e-234
```

```
In [466]: logr.predict_proba(i)[0][1]

Out[466]: 1.7322678526505758e-54
```

```
In [467]: logr.score(h_test,g_test)

Out[467]: 0.810952380952381
```

```
In [468]: from sklearn.linear_model import ElasticNet
          en=ElasticNet()
          en.fit(x_train,y_train)

Out[468]: ElasticNet()
```

```
In [469]: print(en.coef_)

          [0.         0.         0.         0.         0.97624298 0.
           0.         0.        ]
```

```
In [470]: print(en.intercept_)

          1.5745015045348723
```

In [471]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.0
```

In [472]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[472]:
```
RandomForestClassifier()
```

In [473]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [474]:
```
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)
```

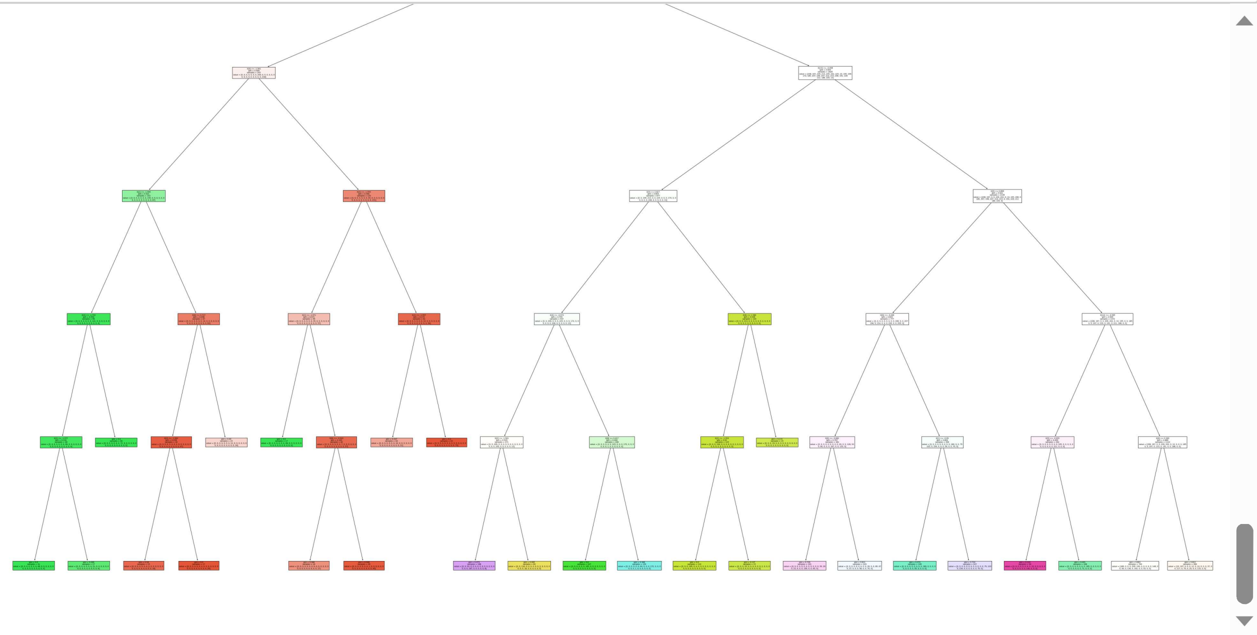
Out[474]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [475]:
```
grid_search.best_score_
```

Out[475]:
```
0.8222448979591837
```

In [476]:
```
rfc_best=grid_search.best_estimator_
```

In [477]:
```
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 RANDON FORESR has the highest accuracy of 0.8222448979591837

In [ ]: