In [59]:
```
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 [60]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_200
a
```

Out[60]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43

243984 rows × 16 columns

In [61]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243984 entries, 0 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     243984 non-null  object 
 1   BEN      69745 non-null   float64
 2   CO       225340 non-null  float64
 3   EBE      61244 non-null   float64
 4   MXY      42045 non-null   float64
 5   NMHC     111951 non-null  float64
 6   NO_2     242625 non-null  float64
 7   NOx      242629 non-null  float64
 8   OXY      42072 non-null   float64
 9   O_3      234131 non-null  float64
 10  PM10     240896 non-null  float64
 11  PXY      42063 non-null   float64
 12  SO_2     242729 non-null  float64
 13  TCH      111991 non-null  float64
 14  TOL      69439 non-null   float64
 15  station  243984 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 29.8+ MB
```

In [62]:
```
b=a.fillna(value=66)
b
```

Out[62]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2003-03-01 01:00:00	66.00	1.72	66.00	66.00	66.00	73.900002	316.299988	66.00	10.550000	55.209999
1	2003-03-01 01:00:00	66.00	1.45	66.00	66.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999
2	2003-03-01 01:00:00	66.00	1.57	66.00	66.00	66.00	80.559998	224.199997	66.00	21.049999	63.240002
3	2003-03-01 01:00:00	66.00	2.45	66.00	66.00	66.00	78.370003	450.399994	66.00	4.220000	67.839996
4	2003-03-01 01:00:00	66.00	3.26	66.00	66.00	66.00	96.250000	479.100006	66.00	8.460000	95.779999
...	...	...	...	...	...	...	...	...	...	...	...
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	66.00	10.450000	14.760000	1.00	34.610001	7.400000
243981	2003-10-01 00:00:00	66.00	66.00	66.00	66.00	0.07	34.639999	50.810001	66.00	32.160000	16.830000
243982	2003-10-01 00:00:00	66.00	66.00	66.00	66.00	0.07	32.580002	41.020000	66.00	66.000000	13.570000
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000

243984 rows × 16 columns

In [63]:
```
b.columns
```

Out[63]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [64]:

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

Out[64]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
0	2003-03-01 01:00:00	66.00	1.72	66.00	66.00	66.00	73.900002	316.299988	66.00	10.550000	55.209999	66.00
1	2003-03-01 01:00:00	66.00	1.45	66.00	66.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999	66.00
2	2003-03-01 01:00:00	66.00	1.57	66.00	66.00	66.00	80.559998	224.199997	66.00	21.049999	63.240002	66.00
3	2003-03-01 01:00:00	66.00	2.45	66.00	66.00	66.00	78.370003	450.399994	66.00	4.220000	67.839996	66.00
4	2003-03-01 01:00:00	66.00	3.26	66.00	66.00	66.00	96.250000	479.100006	66.00	8.460000	95.779999	66.00
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000	95.150002	7.94
6	2003-03-01 01:00:00	66.00	1.38	66.00	66.00	0.29	89.580002	230.000000	66.00	7.200000	54.000000	66.00
7	2003-03-01 01:00:00	66.00	1.58	66.00	66.00	0.30	93.639999	334.600006	66.00	4.190000	26.620001	66.00
8	2003-03-01 01:00:00	66.00	66.00	66.00	66.00	66.00	66.000000	66.000000	66.00	66.000000	66.000000	66.00
9	2003-03-01 01:00:00	66.00	1.92	66.00	66.00	66.00	71.839996	181.399994	66.00	5.330000	39.360001	66.00

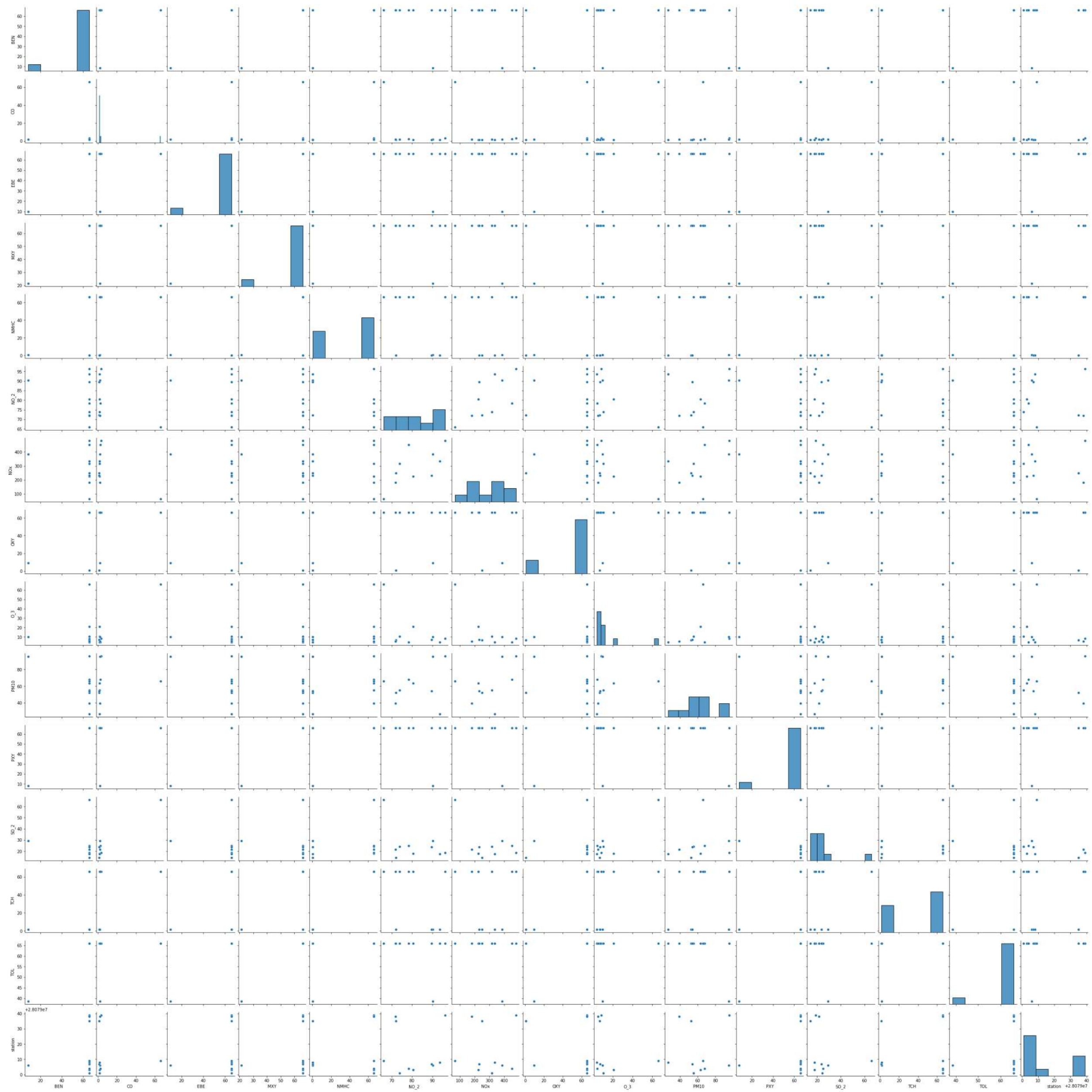
In [65]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[65]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	66.00	1.72	66.00	66.00	66.00	73.900002	316.299988	66.00	10.550000	55.209999	66.00	24.299999
1	66.00	1.45	66.00	66.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999	66.00	14.230000
2	66.00	1.57	66.00	66.00	66.00	80.559998	224.199997	66.00	21.049999	63.240002	66.00	17.879999
3	66.00	2.45	66.00	66.00	66.00	78.370003	450.399994	66.00	4.220000	67.839996	66.00	24.900000
4	66.00	3.26	66.00	66.00	66.00	96.250000	479.100006	66.00	8.460000	95.779999	66.00	18.750000
5	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000	95.150002	7.94	29.270000
6	66.00	1.38	66.00	66.00	0.29	89.580002	230.000000	66.00	7.200000	54.000000	66.00	23.709999
7	66.00	1.58	66.00	66.00	0.30	93.639999	334.600006	66.00	4.190000	26.620001	66.00	17.740000
8	66.00	66.00	66.00	66.00	66.00	66.000000	66.000000	66.00	66.000000	66.000000	66.00	66.000000
9	66.00	1.92	66.00	66.00	66.00	71.839996	181.399994	66.00	5.330000	39.360001	66.00	21.639999

In [66]:
```
sns.pairplot(d)
```

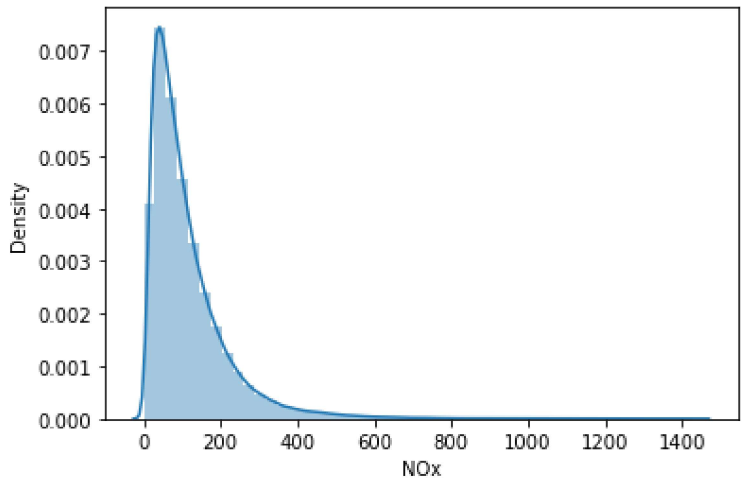
Out[66]: <seaborn.axisgrid.PairGrid at 0x1b634b6ae50>



In [67]:
```
sns.distplot(a['NOx'])
```

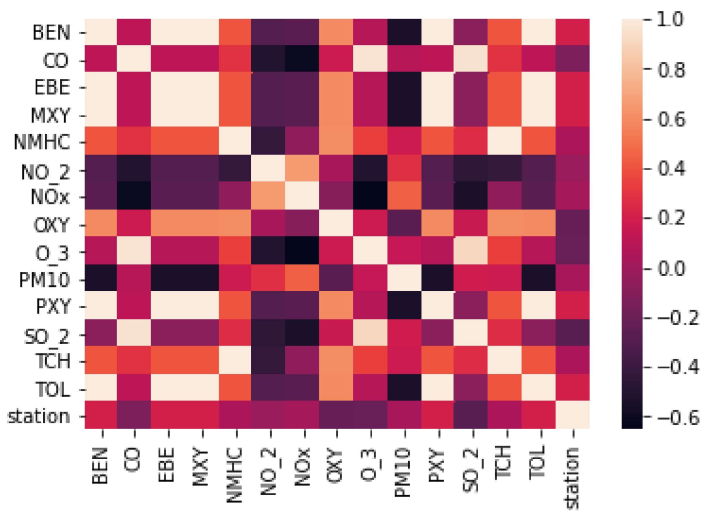
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[67]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [68]:
```
sns.heatmap(d.corr())
```

Out[68]: <AxesSubplot:>



In [69]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [70]:
```
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 [71]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[71]: LinearRegression()

In [72]:
```
print(lr.intercept_)
```

```
1.095958549205534
```

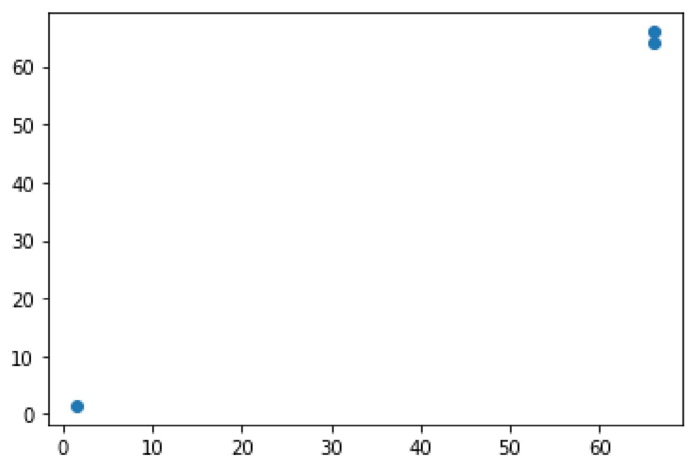
In [73]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[73]:

	Co-efficient
BEN	0.000440
CO	-0.026509
EBE	0.000429
MXY	0.000340
NMHC	0.981414
NO_2	0.000441
NOx	0.000142
OXY	0.000432

In [74]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[74]: <matplotlib.collections.PathCollection at 0x1b6454da400>



In [75]:
```
print(lr.score(x_test,y_test))
```

```
0.9989117112530806
```

In [76]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [77]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[77]: Ridge(alpha=10)

In [78]:
```
rr.score(x_test,y_test)
```

Out[78]: 0.9997914937369043

In [79]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[79]:
```
Lasso(alpha=10)
```

In [80]:
```
la.score(x_test,y_test)
```

Out[80]:
```
0.9999207923608555
```

In [81]:
```
a1=b.head(7000)
a1
```

Out[81]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	I
0	2003-03-01 01:00:00	66.00	1.72	66.00	66.00	66.00	73.900002	316.299988	66.00	10.550000	55.209999	6
1	2003-03-01 01:00:00	66.00	1.45	66.00	66.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999	6
2	2003-03-01 01:00:00	66.00	1.57	66.00	66.00	66.00	80.559998	224.199997	66.00	21.049999	63.240002	6
3	2003-03-01 01:00:00	66.00	2.45	66.00	66.00	66.00	78.370003	450.399994	66.00	4.220000	67.839996	6
4	2003-03-01 01:00:00	66.00	3.26	66.00	66.00	66.00	96.250000	479.100006	66.00	8.460000	95.779999	6
...	...	...	...	...	...	...	...	...	...	...	...	
6995	2003-03-11 10:00:00	1.53	0.88	1.50	2.96	0.17	51.119999	154.800003	1.42	8.690000	47.549999	
6996	2003-03-11 10:00:00	3.68	0.81	3.72	8.24	66.00	143.300003	408.799988	0.59	5.860000	130.100006	:
6997	2003-03-11 10:00:00	66.00	66.00	66.00	66.00	0.22	108.199997	305.000000	66.00	12.920000	115.800003	6
6998	2003-03-11 10:00:00	66.00	66.00	66.00	66.00	0.13	95.540001	292.500000	66.00	66.000000	71.199997	6
6999	2003-03-11 10:00:00	4.21	1.75	2.81	8.05	0.26	96.910004	289.000000	2.45	9.690000	84.500000	:

7000 rows × 16 columns

In [103]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [104]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [105]:
```
h=StandardScaler().fit_transform(f)
```

In [106]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[106]:
```
LogisticRegression(max_iter=10000)
```

In [107]:
```
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 [108]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [109]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079009]
```

In [110]:
```
logr.classes_
```

Out[110]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079026, 28079027, 28079035, 28079036,
       28079038, 28079039, 28079040, 28079099], dtype=int64)
```

In [111]:
```
logr.predict_proba(i)[0][0]
```

Out[111]:
```
4.2122408912106517e-07
```

In [112]:
```
logr.predict_proba(i)[0][1]
```

Out[112]:
```
0.04630224657883687
```

In [113]:
```
logr.score(h_test,g_test)
```

Out[113]:
```
0.6233333333333333
```

In [114]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[114]:
```
ElasticNet()
```

In [115]:
```
print(en.coef_)
```

```
[ 2.33058774e-05  0.00000000e+00  7.77688419e-04  0.00000000e+00
  9.80396589e-01 -0.00000000e+00  0.00000000e+00  3.03708875e-05]
```

In [116]:
```
print(en.intercept_)
```

```
1.2126198373680808
```

In [117]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999991829399287
```

In [118]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[118]:
```
RandomForestClassifier()
```

In [119]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [120]:
```
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)
```

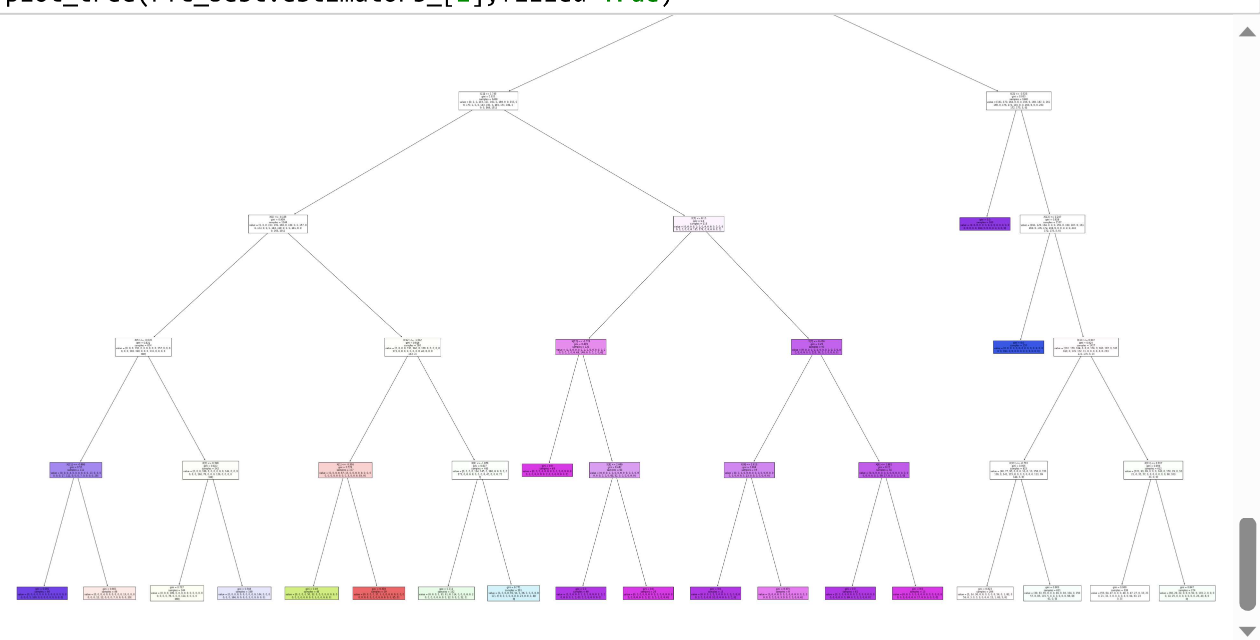
Out[120]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [121]:
```
grid_search.best_score_
```

Out[121]:
```
0.5663265306122449
```

In [122]:
```
rfc_best=grid_search.best_estimator_
```

In [123]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```



Conclusion: from this data set i observed that the ELASTICNET has the highest accuracy of 0.999999182939928

In [ ]: