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 [302]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2007
a
```

Out[302]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	156.199997	97
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	80.809998	N
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	53.099998	N
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	105.300003	N
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	106.500000	15
...	...	...	...	...	...	...	...	...	...	...	...	
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	6.760000	5
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000	5.700000	N
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001	13.010000	6
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN	6.610000	N
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	10.260000	7

225120 rows × 17 columns

In [303]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225120 entries, 0 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     225120 non-null  object 
 1   BEN      68885 non-null   float64
 2   CO       206748 non-null  float64
 3   EBE      68883 non-null   float64
 4   MXY      26061 non-null   float64
 5   NMHC     86883 non-null   float64
 6   NO_2     223985 non-null  float64
 7   NOx      223972 non-null  float64
 8   OXY      26062 non-null   float64
 9   O_3      211850 non-null  float64
 10  PM10     222588 non-null  float64
 11  PM25     68870 non-null   float64
 12  PXY      26062 non-null   float64
 13  SO_2     224372 non-null  float64
 14  TCH      87026 non-null   float64
 15  TOL      68845 non-null   float64
 16  station  225120 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.2+ MB
```

In [304]:
```
b=a.fillna(value=200)
b
```

Out[304]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2007-12-01 01:00:00	200.00	2.86	200.00	200.00	200.00	282.200012	1054.000000	200.00	4.030000	156.
1	2007-12-01 01:00:00	200.00	1.82	200.00	200.00	200.00	86.419998	354.600006	200.00	3.260000	80.
2	2007-12-01 01:00:00	200.00	1.47	200.00	200.00	200.00	94.639999	319.000000	200.00	5.310000	53.
3	2007-12-01 01:00:00	200.00	1.64	200.00	200.00	200.00	127.900002	476.700012	200.00	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.
...	...	...	...	...	...	...	...	...	...	...	
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	6.
225116	2007-03-01 00:00:00	200.00	0.16	200.00	200.00	200.00	46.820000	51.480000	200.00	22.150000	5.
225117	2007-03-01 00:00:00	0.24	200.00	0.20	200.00	0.09	51.259998	66.809998	200.00	18.540001	13.
225118	2007-03-01 00:00:00	0.11	200.00	1.00	200.00	0.05	24.240000	36.930000	200.00	200.000000	6.
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	10.

225120 rows × 17 columns

In [305]:
```
b.columns
```

Out[305]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [306]:
```
c=b.head(10)
c
```

Out[306]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2007-12-01 01:00:00	200.00	2.86	200.00	200.00	200.00	282.200012	1054.000000	200.00	4.030000	156.199997
1	2007-12-01 01:00:00	200.00	1.82	200.00	200.00	200.00	86.419998	354.600006	200.00	3.260000	80.809998
2	2007-12-01 01:00:00	200.00	1.47	200.00	200.00	200.00	94.639999	319.000000	200.00	5.310000	53.099998
3	2007-12-01 01:00:00	200.00	1.64	200.00	200.00	200.00	127.900002	476.700012	200.00	4.500000	105.300003
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	106.500000
5	2007-12-01 01:00:00	200.00	1.35	200.00	200.00	0.56	115.300003	319.600006	200.00	9.880000	57.500000
6	2007-12-01 01:00:00	5.54	1.87	4.65	200.00	0.75	165.100006	520.000000	200.00	4.780000	75.989998
7	2007-12-01 01:00:00	200.00	1.57	200.00	200.00	200.00	97.830002	369.000000	200.00	4.870000	59.590000
8	2007-12-01 01:00:00	200.00	0.70	200.00	200.00	200.00	107.699997	188.500000	200.00	4.560000	43.340000
9	2007-12-01 01:00:00	200.00	1.48	200.00	200.00	0.69	152.500000	485.200012	200.00	8.230000	80.830002

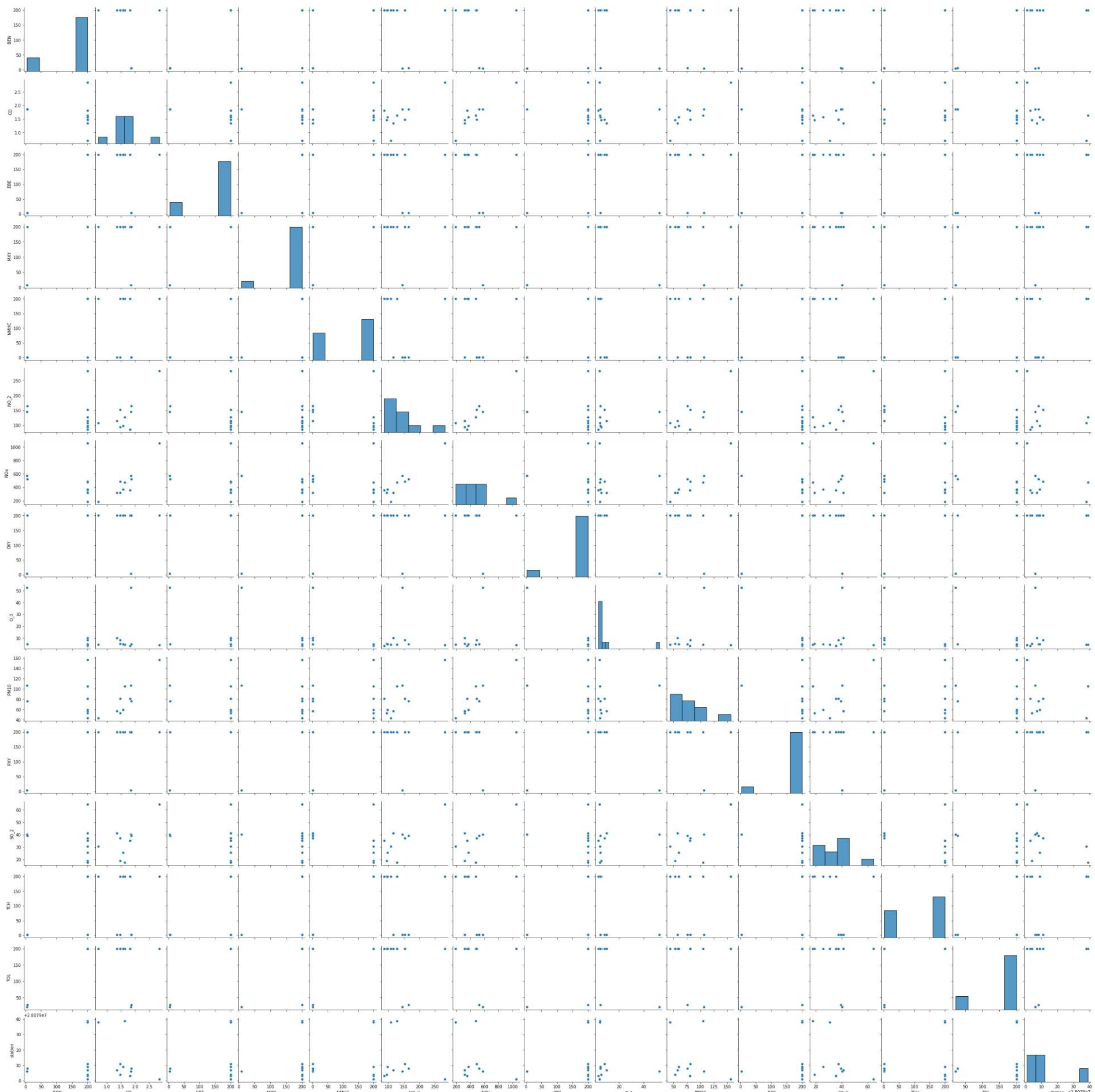
In [307]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[307]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	
0	200.00	2.86	200.00	200.00	200.00	282.200012	1054.000000	200.00	4.030000	156.199997	200.00	64
1	200.00	1.82	200.00	200.00	200.00	86.419998	354.600006	200.00	3.260000	80.809998	200.00	35
2	200.00	1.47	200.00	200.00	200.00	94.639999	319.000000	200.00	5.310000	53.099998	200.00	19
3	200.00	1.64	200.00	200.00	200.00	127.900002	476.700012	200.00	4.500000	105.300003	200.00	17
4	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.500000	3.56	40
5	200.00	1.35	200.00	200.00	0.56	115.300003	319.600006	200.00	9.880000	57.500000	200.00	41
6	5.54	1.87	4.65	200.00	0.75	165.100006	520.000000	200.00	4.780000	75.989998	200.00	39
7	200.00	1.57	200.00	200.00	200.00	97.830002	369.000000	200.00	4.870000	59.590000	200.00	25
8	200.00	0.70	200.00	200.00	200.00	107.699997	188.500000	200.00	4.560000	43.340000	200.00	30
9	200.00	1.48	200.00	200.00	0.69	152.500000	485.200012	200.00	8.230000	80.830002	200.00	37

In [308]:
```
sns.pairplot(d)
```

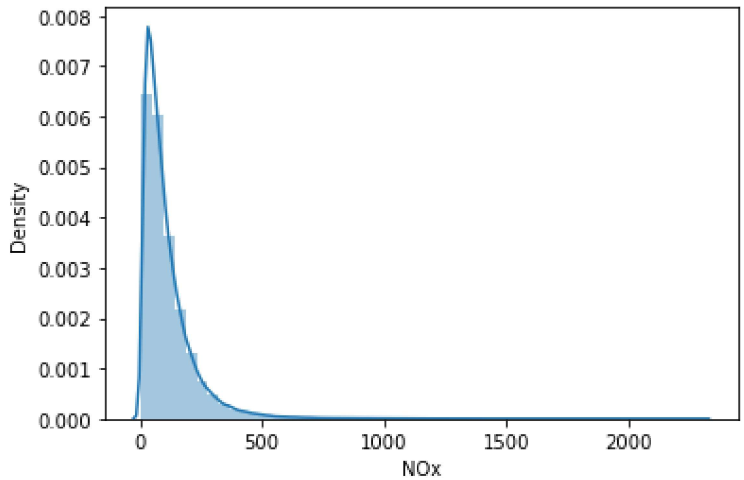
Out[308]: <seaborn.axisgrid.PairGrid at 0x1b69f2da970>



In [309]:
```
sns.distplot(a['NOx'])
```

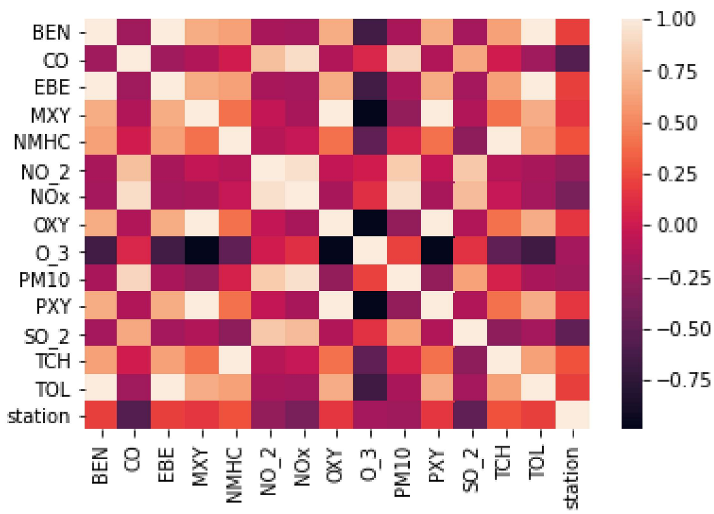
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[309]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [310]:
```
sns.heatmap(d.corr())
```

Out[310]: <AxesSubplot:>



In [311]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [312]:
```
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 [313]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[313]: LinearRegression()

In [314]:
```
print(lr.intercept_)
```

1.3719012248623699

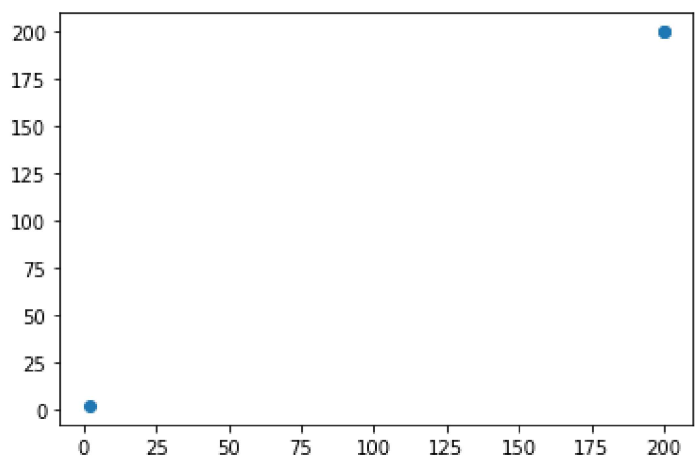
In [315]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[315]:

	Co-efficient
BEN	1.525500e-04
CO	2.187635e-13
EBE	1.530362e-04
MXY	7.916465e-05
NMHC	9.926747e-01
NO_2	1.115168e-15
NOx	-7.853641e-16
OXY	8.101575e-05

In [316]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[316]: <matplotlib.collections.PathCollection at 0x1b6b3ad9e50>



In [317]:
```
print(lr.score(x_test,y_test))
```

0.9999974008505289

In [318]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [319]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[319]: Ridge(alpha=10)

In [320]:
```
rr.score(x_test,y_test)
```

Out[320]: 0.999996315282576

In [321]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[321]: Lasso(alpha=10)

In [322]:
```
la.score(x_test,y_test)
```

Out[322]: 0.9999953067553261

In [323]:
```
a1=b.head(7000)
a1
```

Out[323]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2007-12-01 01:00:00	200.00	2.86	200.00	200.00	200.00	282.200012	1054.000000	200.00	4.030000	156.1999
1	2007-12-01 01:00:00	200.00	1.82	200.00	200.00	200.00	86.419998	354.600006	200.00	3.260000	80.8099
2	2007-12-01 01:00:00	200.00	1.47	200.00	200.00	200.00	94.639999	319.000000	200.00	5.310000	53.0999
3	2007-12-01 01:00:00	200.00	1.64	200.00	200.00	200.00	127.900002	476.700012	200.00	4.500000	105.3000
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	106.5000
...	...	...	...	...	...	...	...	...	...	...	
6995	2007-12-12 06:00:00	200.00	0.63	200.00	200.00	200.00	43.520000	99.480003	200.00	3.070000	23.3099
6996	2007-12-12 06:00:00	200.00	0.52	200.00	200.00	200.00	37.279999	73.059998	200.00	1.000000	21.1100
6997	2007-12-12 06:00:00	200.00	0.29	200.00	200.00	200.00	55.619999	118.900002	200.00	200.000000	24.7300
6998	2007-12-12 06:00:00	0.62	0.34	0.49	0.68	0.21	59.540001	101.300003	0.33	17.809999	12.1500
6999	2007-12-12 06:00:00	200.00	0.26	200.00	200.00	0.29	39.490002	43.330002	200.00	20.870001	8.6600

7000 rows × 17 columns

In [324]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [325]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [326]:
```
h=StandardScaler().fit_transform(f)
```

In [327]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[327]:
```
LogisticRegression(max_iter=10000)
```

In [328]:
```
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 [329]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [330]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079038]
```

In [331]:
```
logr.classes_
```

Out[331]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [332]:
```
logr.predict_proba(i)[0][0]
```

Out[332]:
```
9.139676819864243e-63
```

In [333]:
```
logr.predict_proba(i)[0][1]
```

Out[333]:
```
3.385841902574991e-91
```

In [334]:
```
logr.score(h_test,g_test)
```

Out[334]:
```
0.4928571428571429
```

In [335]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.p
y:530: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations. Duality gap: 8.595062614081863, tolerance: 6.715258888678174
  model = cd_fast.enet_coordinate_descent(
```

Out[335]:
```
ElasticNet()
```

In [336]:
```
print(en.coef_)
```

```
[ 0.00000000e+00 -0.00000000e+00  1.04181894e-05  3.57881209e-01
  9.92749489e-01 -0.00000000e+00 -0.00000000e+00 -3.49208720e-01]
```

In [337]:
```
print(en.intercept_)
```

```
-0.2870956300752425
```

In [338]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999977009210478
```

In [339]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[339]:
```
RandomForestClassifier()
```

In [340]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [341]:
```
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[341]:
```
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 [342]:
```
grid_search.best_score_
```

Out[342]:
```
0.5379591836734694
```

In [343]:
```
rfc_best=grid_search.best_estimator_
```

In [344]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
 Text(1997.0526315789473, 226.5, 'gini = 0.161\nsamples = 79\nvalue = [0, 0, 0, 5,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 106]'),
 Text(2232.0, 226.5, 'gini = 0.0\nsamples = 26\nvalue = [0, 0, 0, 37, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2466.9473684210525, 1132.5, 'X[8] <= -0.354\ngini = 0.315\nsamples = 35\nvalu
e = [0, 0, 0, 41, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0]'),
 Text(2349.4736842105262, 679.5, 'gini = 0.0\nsamples = 8\nvalue = [0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10]'),
 Text(2584.4210526315787, 679.5, 'gini = 0.0\nsamples = 27\nvalue = [0, 0, 0, 41,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3641.684210526316, 2038.5, 'X[0] <= -0.377\ngini = 0.956\nsamples = 2705\nval
ue = [210, 204, 201, 0, 181, 182, 171, 200, 168, 173\n189, 160, 159, 183, 180, 192,
182, 0, 195, 203\n186, 198, 208, 210, 185, 0]'),
 Text(3524.2105263157896, 1585.5, 'X[0] <= -1.529\ngini = 0.798\nsamples = 579\nval
ue = [0, 0, 0, 0, 0, 147, 0, 0, 0, 0, 189, 0, 0, 0\n0, 182, 0, 0, 203, 186, 0,
0, 0, 0, 0]'),
 Text(3054.315789473684, 1132.5, 'X[1] <= 1.539\ngini = 0.76\nsamples = 407\nvalue
= [0, 0, 0, 0, 0, 106, 0, 0, 0, 0, 26, 0, 0, 0\n0, 161, 0, 0, 184, 165, 0, 0, 0,
0, 0]'),
```

Conclusion: from this data set i observed that the ELASTICNET has the highest accuracy of 0.999997700921047

In [ ]: