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 [345]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2008
a
```

Out[345]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM2!
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.889999	10.4(
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.040001	NaI
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.270000	NaI
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.850000	NaI
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000	21.9(
...	...	...	...	...	...	...	...	...	...	...	...	..
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450000	5.1!
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.020000	NaI
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.540001	13.9
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.910000	NaI
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690000	11.4:

226392 rows × 17 columns

In [346]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 226392 entries, 0 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     226392 non-null  object 
 1   BEN      67047 non-null   float64
 2   CO       208109 non-null  float64
 3   EBE      67044 non-null   float64
 4   MXY      25867 non-null   float64
 5   NMHC     85079 non-null   float64
 6   NO_2     225315 non-null  float64
 7   NOx      225311 non-null  float64
 8   OXY      25878 non-null   float64
 9   O_3      215716 non-null  float64
 10  PM10     220179 non-null  float64
 11  PM25     67833 non-null   float64
 12  PXY      25877 non-null   float64
 13  SO_2     225405 non-null  float64
 14  TCH      85107 non-null   float64
 15  TOL      66940 non-null   float64
 16  station  226392 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.4+ MB
```

In [347]:
```
b=a.fillna(value=67)
b
```

Out[347]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	F
0	2008-06-01 01:00:00	145.00	0.47	145.00	145.00	145.00	83.089996	120.699997	145.00	16.990000	16.88
1	2008-06-01 01:00:00	145.00	0.59	145.00	145.00	145.00	94.820000	130.399994	145.00	17.469999	19.04
2	2008-06-01 01:00:00	145.00	0.55	145.00	145.00	145.00	75.919998	104.599998	145.00	13.470000	20.27
3	2008-06-01 01:00:00	145.00	0.36	145.00	145.00	145.00	61.029999	66.559998	145.00	23.110001	10.85
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.16
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.45
226388	2008-11-01 00:00:00	145.00	0.30	145.00	145.00	145.00	41.880001	48.500000	145.00	35.830002	15.02
226389	2008-11-01 00:00:00	0.25	145.00	0.56	145.00	0.11	83.610001	102.199997	145.00	14.130000	17.54
226390	2008-11-01 00:00:00	0.54	145.00	2.70	145.00	0.18	70.639999	81.860001	145.00	145.000000	11.91
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.69

226392 rows × 17 columns

In [348]:
```
b.columns
```

Out[348]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [349]:
```
c=b.head(10)
c
```

Out[349]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
0	2008-06-01 01:00:00	145.00	0.47	145.00	145.00	145.00	83.089996	120.699997	145.00	16.990000	16.889999	1
1	2008-06-01 01:00:00	145.00	0.59	145.00	145.00	145.00	94.820000	130.399994	145.00	17.469999	19.040001	14
2	2008-06-01 01:00:00	145.00	0.55	145.00	145.00	145.00	75.919998	104.599998	145.00	13.470000	20.270000	14
3	2008-06-01 01:00:00	145.00	0.36	145.00	145.00	145.00	61.029999	66.559998	145.00	23.110001	10.850000	14
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000	2
5	2008-06-01 01:00:00	145.00	0.47	145.00	145.00	0.22	67.820000	101.099998	145.00	20.610001	23.389999	14
6	2008-06-01 01:00:00	0.17	0.40	0.44	145.00	0.15	72.639999	91.220001	145.00	17.040001	19.940001	14
7	2008-06-01 01:00:00	145.00	0.51	145.00	145.00	145.00	80.440002	141.500000	145.00	10.310000	37.259998	14
8	2008-06-01 01:00:00	145.00	0.36	145.00	145.00	145.00	68.150002	85.639999	145.00	23.580000	15.060000	
9	2008-06-01 01:00:00	145.00	0.18	145.00	145.00	0.16	58.330002	64.769997	145.00	35.060001	7.400000	14

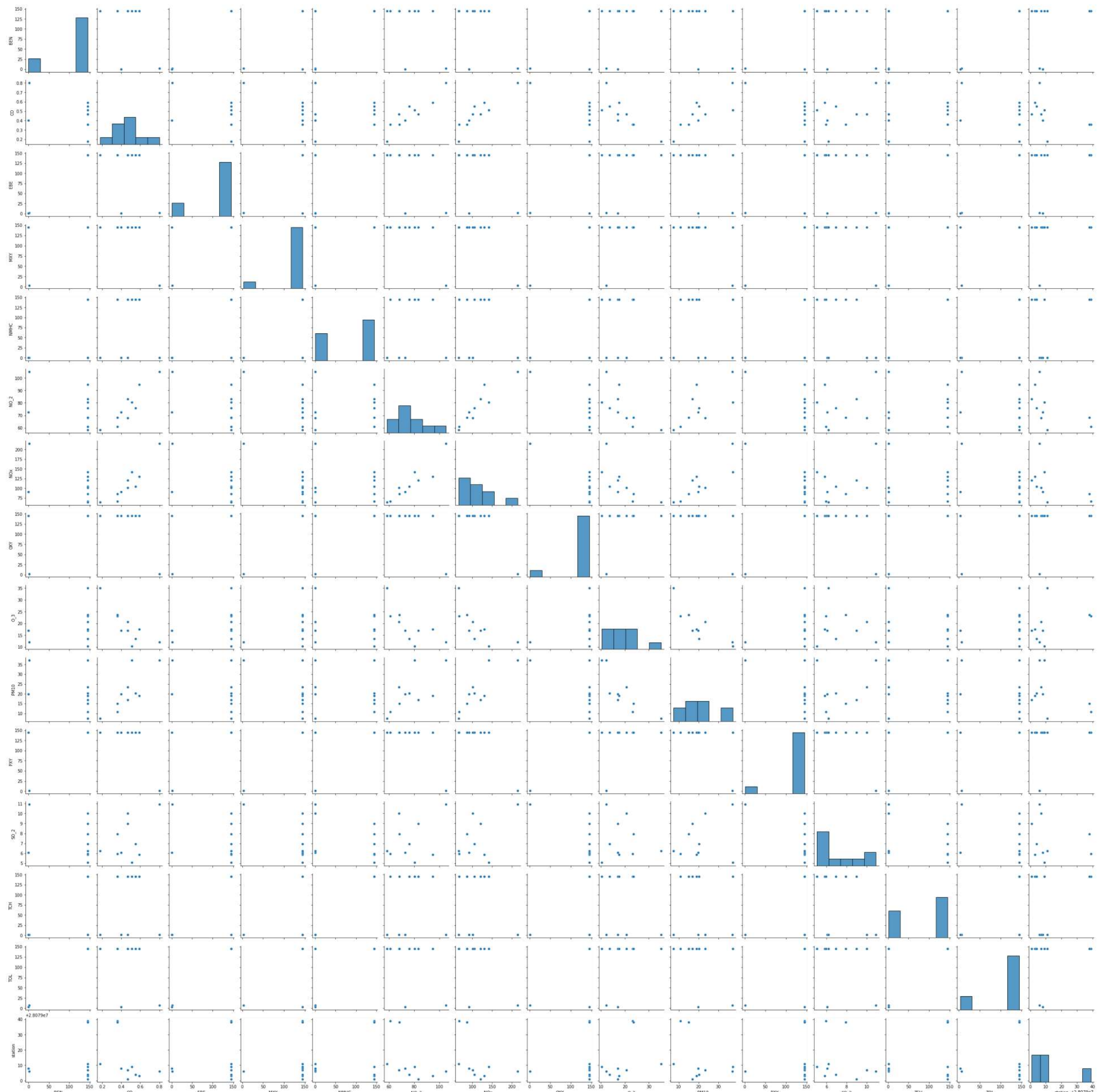
In [350]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[350]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_
0	145.00	0.47	145.00	145.00	145.00	83.089996	120.699997	145.00	16.990000	16.889999	145.00	8.9
1	145.00	0.59	145.00	145.00	145.00	94.820000	130.399994	145.00	17.469999	19.040001	145.00	5.8
2	145.00	0.55	145.00	145.00	145.00	75.919998	104.599998	145.00	13.470000	20.270000	145.00	6.9
3	145.00	0.36	145.00	145.00	145.00	61.029999	66.559998	145.00	23.110001	10.850000	145.00	5.9
4	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000	1.43	10.9
5	145.00	0.47	145.00	145.00	0.22	67.820000	101.099998	145.00	20.610001	23.389999	145.00	10.0
6	0.17	0.40	0.44	145.00	0.15	72.639999	91.220001	145.00	17.040001	19.940001	145.00	6.0
7	145.00	0.51	145.00	145.00	145.00	80.440002	141.500000	145.00	10.310000	37.259998	145.00	5.0
8	145.00	0.36	145.00	145.00	145.00	68.150002	85.639999	145.00	23.580000	15.060000	145.00	7.9
9	145.00	0.18	145.00	145.00	0.16	58.330002	64.769997	145.00	35.060001	7.400000	145.00	6.2

In [351]:
```
sns.pairplot(d)
```

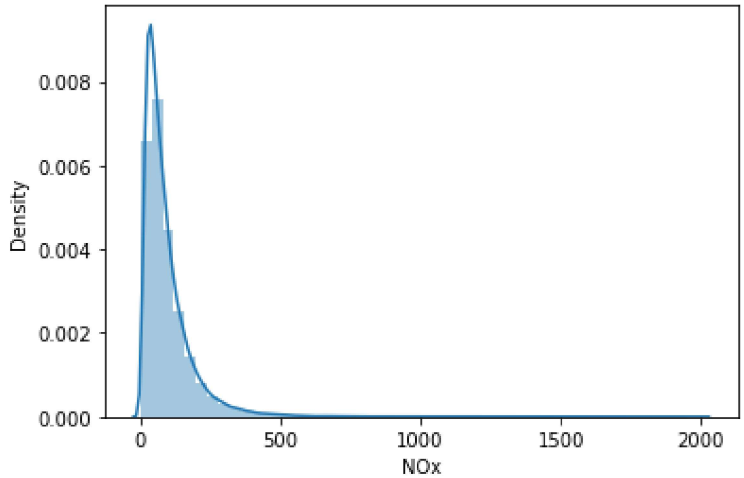
Out[351]: <seaborn.axisgrid.PairGrid at 0x1b6b1786d30>



In [352]:
```
sns.distplot(a['NOx'])
```

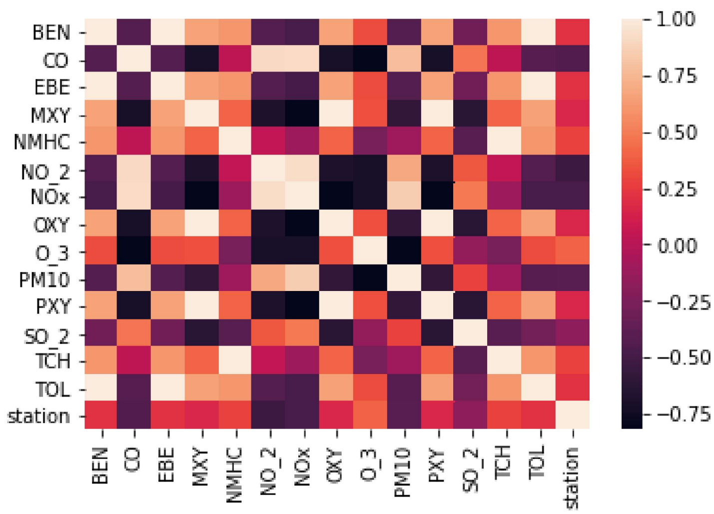
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[352]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [353]:
```
sns.heatmap(d.corr())
```

Out[353]: <AxesSubplot:>



In [354]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [355]:
```
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 [356]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[356]: LinearRegression()

In [357]:
```
print(lr.intercept_)
```

```
1.2344948896697616
```

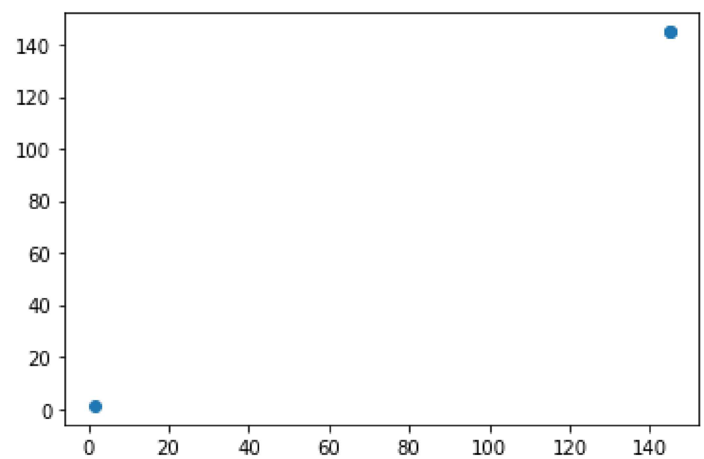
In [358]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[358]:

	Co-efficient
BEN	-4.838476e-04
CO	6.625942e-14
EBE	-4.832031e-04
MXY	-1.479348e-04
NMHC	9.927506e-01
NO_2	-1.152870e-14
NOx	3.615275e-15
OXY	-1.493934e-04

In [359]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[359]: <matplotlib.collections.PathCollection at 0x1b6c94ca910>



In [360]:
```
print(lr.score(x_test,y_test))
```

```
0.9999997345787882
```

In [361]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [362]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[362]: Ridge(alpha=10)

In [363]:
```
rr.score(x_test,y_test)
```

Out[363]: 0.9999988523657496

In [364]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[364]:
```
Lasso(alpha=10)
```

In [365]:
```
la.score(x_test,y_test)
```

Out[365]:
```
0.9999947449730094
```

In [366]:
```
a1=b.head(7000)
a1
```

Out[366]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2008-06-01 01:00:00	145.00	0.47	145.0	145.00	145.00	83.089996	120.699997	145.00	16.990000	16.889999
1	2008-06-01 01:00:00	145.00	0.59	145.0	145.00	145.00	94.820000	130.399994	145.00	17.469999	19.040001
2	2008-06-01 01:00:00	145.00	0.55	145.0	145.00	145.00	75.919998	104.599998	145.00	13.470000	20.270000
3	2008-06-01 01:00:00	145.00	0.36	145.0	145.00	145.00	61.029999	66.559998	145.00	23.110001	10.850000
4	2008-06-01 01:00:00	1.68	0.80	1.7	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000
...	...	...	...	...	...	...	...	...	...	...	...
6995	2008-06-12 06:00:00	145.00	0.32	145.0	145.00	145.00	65.290001	86.440002	145.00	18.590000	15.790000
6996	2008-06-12 06:00:00	145.00	0.12	145.0	145.00	145.00	27.959999	31.129999	145.00	59.799999	18.430000
6997	2008-06-12 06:00:00	145.00	0.14	145.0	145.00	0.13	15.480000	18.360001	145.00	73.620003	9.890000
6998	2008-06-12 06:00:00	145.00	0.29	145.0	145.00	145.00	15.630000	18.490000	145.00	78.970001	8.470000
6999	2008-06-12 06:00:00	145.00	0.16	145.0	145.00	145.00	11.860000	14.410000	145.00	88.370003	11.270000

7000 rows × 17 columns

In [367]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [368]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [369]:
```
h=StandardScaler().fit_transform(f)
```

In [370]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[370]:
```
LogisticRegression(max_iter=10000)
```

In [371]:
```
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 [372]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [373]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079039]
```

In [374]:
```
logr.classes_
```

Out[374]:
```
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 [375]:
```
logr.predict_proba(i)[0][0]
```

Out[375]:
```
1.434851782584846e-58
```

In [376]:
```
logr.predict_proba(i)[0][1]
```

Out[376]:
```
2.0338132081220725e-57
```

In [377]:
```
logr.score(h_test,g_test)
```

Out[377]:
```
0.5480952380952381
```

In [378]:
```
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: 5.907676344145945, tolerance: 3.5369965392932508
  model = cd_fast.enet_coordinate_descent(
```

Out[378]:
```
ElasticNet()
```

In [379]:
```
print(en.coef_)
```

```
[-5.65388513e-04  0.00000000e+00 -1.73285441e-04  1.21588122e-01
  9.92482759e-01 -0.00000000e+00 -0.00000000e+00 -1.20484102e-01]
```

In [380]:
```
print(en.intercept_)
```

```
1.0341488827156695
```

In [381]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999999550715535
```

In [382]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[382]:
```
RandomForestClassifier()
```

In [383]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [384]:
```
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[384]:
```
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')
```

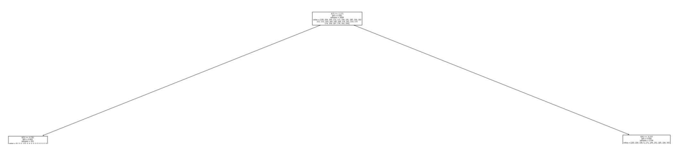
In [385]:
```
grid_search.best_score_
```

Out[385]:
```
0.6612244897959183
```

In [386]:
```
rfc_best=grid_search.best_estimator_
```

In [391]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[20],filled=True)
```

```
= [192, 14, 87, 0, 0, 0, 6, 0, 10, 109, 0, 24, 8\n124, 167, 5, 0, 0, 119, 0, 0, 19
1, 124, 29, 79\n0]'),
 Text(3826.2857142857147, 226.5, 'gini = 0.885\nsamples = 351\nvalue = [120, 10, 3
6, 0, 0, 0, 6, 0, 6, 43, 0, 16, 0\n57, 54, 5, 0, 0, 72, 0, 0, 66, 35, 8, 20, 0]'),
 Text(4008.4897959183677, 226.5, 'gini = 0.89\nsamples = 463\nvalue = [72, 4, 51,
0, 0, 0, 0, 0, 4, 66, 0, 8, 8, 67\n113, 0, 0, 0, 47, 0, 0, 125, 89, 21, 59, 0]'),
 Text(4281.795918367347, 679.5, 'X[5] <= -0.717\ngini = 0.844\nsamples = 253\nvalue
= [3, 1, 96, 0, 4, 0, 0, 0, 0, 40, 0, 0, 0, 31\n28, 0, 0, 0, 26, 0, 0, 8, 63, 11, 8
1, 0]'),
 Text(4190.693877551021, 226.5, 'gini = 0.699\nsamples = 94\nvalue = [0, 0, 38, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 4\n1, 0, 0, 0, 5, 0, 0, 2, 27, 5, 66, 0]'),
 Text(4372.897959183674, 226.5, 'gini = 0.858\nsamples = 159\nvalue = [3, 1, 58, 0,
4, 0, 0, 0, 0, 40, 0, 0, 0, 27\n27, 0, 0, 0, 21, 0, 0, 6, 36, 6, 15, 0]')]
```



Conclusion: from this data set i observed that the ELASTICNET has the highest accuracy of 0.9999999550715535

In [ ]: