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 [175]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2005
a
```

Out[175]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PX
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.91	10.65	Na
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.93	NaN	1.5
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.60	NaN	Na
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.16	NaN	Na
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.00	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998	5.00	NaN	Na
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95	1.49	1.0
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.31	2.93	Na
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.00	NaN	Na
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67	2.11	1.0

237000 rows × 17 columns

In [176]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

In [177]:
```python
b=a.fillna(value=102)
b
```

Out[177]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2005-11-01 01:00:00	102.00	0.77	102.00	102.00	102.00	57.130001	128.699997	102.00	14.720000	14.91
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.93
2	2005-11-01 01:00:00	102.00	0.40	102.00	102.00	102.00	46.119999	53.000000	102.00	30.469999	14.60
3	2005-11-01 01:00:00	102.00	0.42	102.00	102.00	102.00	37.220001	52.009998	102.00	21.379999	15.16
4	2005-11-01 01:00:00	102.00	0.57	102.00	102.00	102.00	32.160000	36.680000	102.00	33.410000	5.00
...	...	...	...	...	...	...	...	...	...	...	...
236995	2006-01-01 00:00:00	1.08	0.36	1.01	102.00	0.11	21.990000	23.610001	102.00	43.349998	5.00
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95
236997	2006-01-01 00:00:00	0.19	102.00	0.26	102.00	0.08	26.730000	30.809999	102.00	43.840000	4.31
236998	2006-01-01 00:00:00	0.14	102.00	1.00	102.00	0.06	13.770000	17.770000	102.00	102.000000	5.00
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67

237000 rows × 17 columns

In [178]:
```python
b.columns
```

Out[178]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [179]:
```
c=b.head(10)
c
```

Out[179]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM
0	2005-11-01 01:00:00	102.00	0.77	102.00	102.00	102.00	57.130001	128.699997	102.00	14.720000	14.910000	10
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.930000	102
2	2005-11-01 01:00:00	102.00	0.40	102.00	102.00	102.00	46.119999	53.000000	102.00	30.469999	14.600000	102
3	2005-11-01 01:00:00	102.00	0.42	102.00	102.00	102.00	37.220001	52.009998	102.00	21.379999	15.160000	102
4	2005-11-01 01:00:00	102.00	0.57	102.00	102.00	102.00	32.160000	36.680000	102.00	33.410000	5.000000	102
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.070000	17
6	2005-11-01 01:00:00	102.00	0.55	102.00	102.00	0.27	50.279999	77.209999	102.00	19.120001	18.209999	102
7	2005-11-01 01:00:00	0.20	0.38	1.00	102.00	0.27	51.759998	72.989998	102.00	14.810000	16.430000	102
8	2005-11-01 01:00:00	102.00	0.70	102.00	102.00	102.00	39.040001	43.860001	102.00	25.379999	16.139999	102
9	2005-11-01 01:00:00	102.00	0.56	102.00	102.00	102.00	41.820000	51.869999	102.00	24.290001	7.130000	7

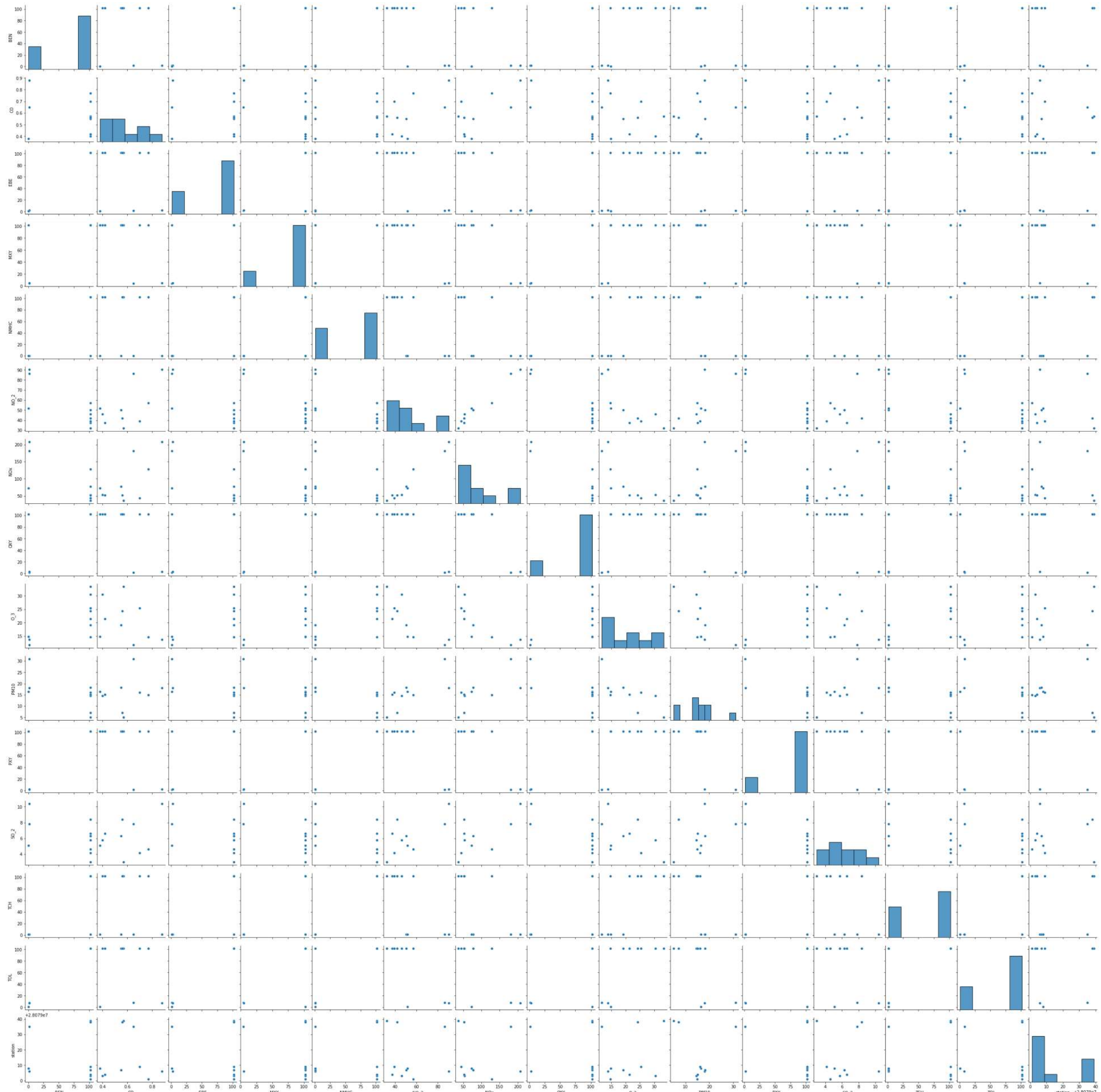
In [180]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[180]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	102.00	0.77	102.00	102.00	102.00	57.130001	128.699997	102.00	14.720000	14.910000	102.00	4.62
1	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.930000	1.59	7.80
2	102.00	0.40	102.00	102.00	102.00	46.119999	53.000000	102.00	30.469999	14.600000	102.00	5.76
3	102.00	0.42	102.00	102.00	102.00	37.220001	52.009998	102.00	21.379999	15.160000	102.00	6.60
4	102.00	0.57	102.00	102.00	102.00	32.160000	36.680000	102.00	33.410000	5.000000	102.00	3.00
5	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.070000	2.44	10.39
6	102.00	0.55	102.00	102.00	0.27	50.279999	77.209999	102.00	19.120001	18.209999	102.00	6.28
7	0.20	0.38	1.00	102.00	0.27	51.759998	72.989998	102.00	14.810000	16.430000	102.00	5.11
8	102.00	0.70	102.00	102.00	102.00	39.040001	43.860001	102.00	25.379999	16.139999	102.00	4.18
9	102.00	0.56	102.00	102.00	102.00	41.820000	51.869999	102.00	24.290001	7.130000	102.00	8.37

In [181]:
```
sns.pairplot(d)
```

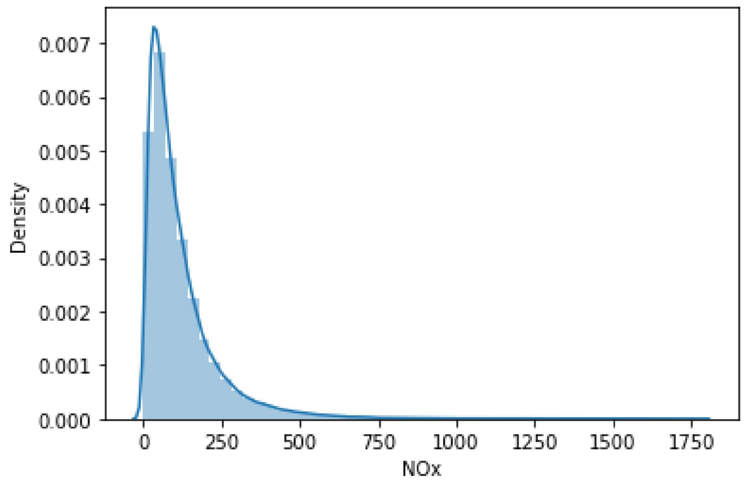
Out[181]: <seaborn.axisgrid.PairGrid at 0x1b65712f700>



In [182]:
```
sns.distplot(a['NOx'])
```

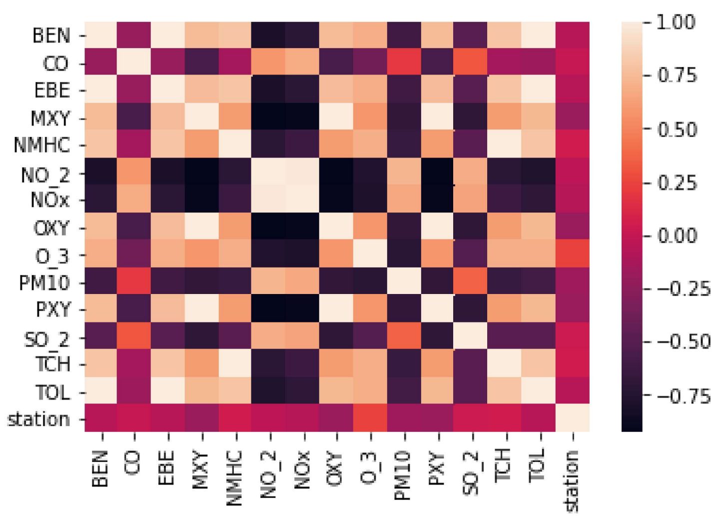
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[182]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [183]:
```
sns.heatmap(d.corr())
```

Out[183]: <AxesSubplot:>



In [184]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [185]:
```
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 [186]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[186]: LinearRegression()

In [187]:
```
print(lr.intercept_)
```

```
-0.25120449480472473
```

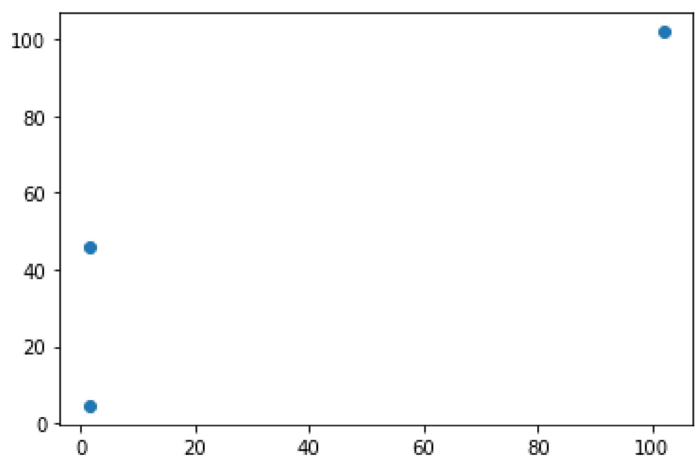
In [188]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[188]:

	Co-efficient
BEN	3.362127e-01
CO	-4.538211e-14
EBE	7.287275e-02
MXY	2.387457e-01
NMHC	5.488915e-01
NO_2	-1.439444e-16
NOx	8.936519e-16
OXY	-1.942599e-01

In [189]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[189]: <matplotlib.collections.PathCollection at 0x1b603997220>



In [190]:
```
print(lr.score(x_test,y_test))
```

```
0.7011963992356471
```

In [191]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [192]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[192]: Ridge(alpha=10)

In [193]:
```
rr.score(x_test,y_test)
```

Out[193]: -0.09671547468017438

In [194]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[194]: Lasso(alpha=10)

In [195]:
```
la.score(x_test,y_test)
```
Out[195]: -0.4922086413609599

In [196]:
```
a1=b.head(7000)
a1
```
Out[196]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2005-11-01 01:00:00	102.00	0.77	102.00	102.00	102.00	57.130001	128.699997	102.00	14.720000	14.9100
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.9300
2	2005-11-01 01:00:00	102.00	0.40	102.00	102.00	102.00	46.119999	53.000000	102.00	30.469999	14.6000
3	2005-11-01 01:00:00	102.00	0.42	102.00	102.00	102.00	37.220001	52.009998	102.00	21.379999	15.1600
4	2005-11-01 01:00:00	102.00	0.57	102.00	102.00	102.00	32.160000	36.680000	102.00	33.410000	5.0000
...	...	...	...	...	...	...	...	...	...	...	
6995	2005-11-11 21:00:00	1.11	0.56	1.85	4.41	0.25	73.570000	100.599998	1.33	11.450000	29.1299
6996	2005-11-11 21:00:00	0.49	102.00	0.25	102.00	0.14	119.800003	254.500000	102.00	2.060000	49.2900
6997	2005-11-11 21:00:00	0.25	102.00	0.51	102.00	0.10	73.500000	104.300003	102.00	102.000000	22.5800
6998	2005-11-11 21:00:00	1.59	0.83	2.06	8.59	0.26	87.279999	118.400002	3.23	7.390000	45.3100
6999	2005-11-11 22:00:00	102.00	0.78	102.00	102.00	102.00	53.900002	166.000000	102.00	11.820000	32.6199

7000 rows × 17 columns

In [197]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [198]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [199]:
```
h=StandardScaler().fit_transform(f)
```

In [200]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[200]:
```
LogisticRegression(max_iter=10000)
```

In [201]:
```
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 [202]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [203]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079039]
```

In [204]:
```
logr.classes_
```

Out[204]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079026, 28079027, 28079035, 28079036, 28079038,
       28079039, 28079040, 28079099], dtype=int64)
```

In [205]:
```
logr.predict_proba(i)[0][0]
```

Out[205]:
```
1.0904259422981613e-265
```

In [206]:
```
logr.predict_proba(i)[0][1]
```

Out[206]:
```
7.244658692284546e-196
```

In [207]:
```
logr.score(h_test,g_test)
```

Out[207]:
```
0.5061904761904762
```

In [208]:
```
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: 2.349501093043866, tolerance: 1.44677193401142
  model = cd_fast.enet_coordinate_descent(
```

Out[208]:
```
ElasticNet()
```

In [209]:
```
print(en.coef_)
```

```
[ 0.77875749 -0.          0.18350609  0.02114982  0.01912012 -0.
 -0.00120644  0.        ]
```

In [210]:
```
print(en.intercept_)
```

```
-0.18985834564632853
```

In [211]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
-0.4443370815034544
```

In [212]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[212]:
```
RandomForestClassifier()
```

In [213]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [214]:
```
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[214]:
```
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')
```

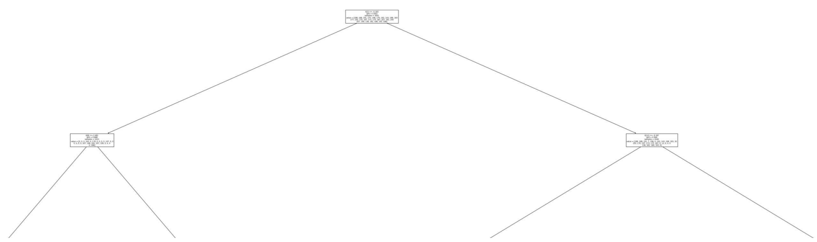
In [215]:
```
grid_search.best_score_
```

Out[215]:
```
0.5565306122448979
```

In [216]:
```
rfc_best=grid_search.best_estimator_
```

In [217]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
0, 0, 0, 8, 0, 12, 99, 0, 15, 62\n47, 25, 8, 1, 0, 0, 0, 0, 0, 14, 3, 35, 37\n0]'),
 Text(3978.782608695652, 226.5, 'gini = 0.693\nsamples = 34\nvalue = [0, 3, 0, 0,
0, 0, 0, 0, 0, 6, 0, 4, 2, 2\n29, 0, 0, 0, 0, 0, 0, 0, 0, 0, 14, 0, 0]'),
 Text(4269.913043478261, 679.5, 'X[8] <= -0.603\ngini = 0.895\nsamples = 243\nvalue
= [16, 32, 12, 0, 0, 0, 88, 0, 40, 11, 30, 4, 4\n3, 5, 39, 31, 0, 0, 0, 0, 0, 23, 2
7, 22, 4\n0]'),
 Text(4172.869565217391, 226.5, 'gini = 0.892\nsamples = 119\nvalue = [5, 25, 12,
0, 0, 0, 0, 0, 22, 9, 0, 4, 4, 2\n2, 30, 31, 0, 0, 0, 0, 0, 11, 18, 8, 4, 0]'),
 Text(4366.95652173913, 226.5, 'gini = 0.768\nsamples = 124\nvalue = [11, 7, 0, 0,
0, 0, 88, 0, 18, 2, 30, 0, 0, 1\n3, 9, 0, 0, 0, 0, 0, 0, 12, 9, 14, 0, 0]')]
```



Conclusion: from this data set i observed that the LINEAR REGRESSION has the highest accuracy of 0.701196399235647

In [ ]: