In [1]:
```
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 [2]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2002.cs
a
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.3200
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.6600
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.6700
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.9900
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.2600
...	...	...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.2100
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999	NaN	15.6400
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.6200
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN	5.52	24.2199
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.9100

217296 rows × 16 columns

In [3]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217296 non-null  object 
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

In [7]:
```
b=a.fillna(value=66)
b
```

Out[7]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	
0	2002-04-01 01:00:00	66.00	1.39	66.00	66.00	66.00	145.100006	352.100006	66.00	6.54	41.990002	66.00	21.3
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.6
2	2002-04-01 01:00:00	66.00	0.80	66.00	66.00	66.00	103.699997	134.000000	66.00	13.01	28.440001	66.00	13.6
3	2002-04-01 01:00:00	66.00	1.61	66.00	66.00	66.00	97.599998	268.000000	66.00	5.12	42.180000	66.00	16.9
4	2002-04-01 01:00:00	66.00	1.90	66.00	66.00	66.00	92.089996	237.199997	66.00	7.28	76.330002	66.00	15.2
...	...	...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	66.00	66.00	66.00	81.080002	265.700012	66.00	7.21	36.750000	66.00	13.2
217292	2002-11-01 00:00:00	3.67	1.73	2.89	66.00	0.38	113.900002	373.100006	66.00	5.66	63.389999	66.00	15.6
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.6
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	66.00	149.800003	202.199997	1.00	5.75	66.000000	5.52	24.2
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.9

217296 rows × 16 columns

In [8]:
```
b.columns
```

Out[8]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
c=b.head(10)
c
```

Out[6]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	S
0	2002-04-01 01:00:00	87.00	1.39	87.00	87.00	87.00	145.100006	352.100006	87.00	6.540000	41.990002	87.00	21.32
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000	20.980000	2.53	11.66
2	2002-04-01 01:00:00	87.00	0.80	87.00	87.00	87.00	103.699997	134.000000	87.00	13.010000	28.440001	87.00	13.67
3	2002-04-01 01:00:00	87.00	1.61	87.00	87.00	87.00	97.599998	268.000000	87.00	5.120000	42.180000	87.00	16.99
4	2002-04-01 01:00:00	87.00	1.90	87.00	87.00	87.00	92.089996	237.199997	87.00	7.280000	76.330002	87.00	15.26
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.370000	27.450001	2.98	14.78
6	2002-04-01 01:00:00	87.00	0.78	87.00	87.00	0.09	101.000000	119.300003	87.00	20.549999	23.950001	87.00	13.63
7	2002-04-01 01:00:00	87.00	1.06	87.00	87.00	87.00	127.300003	204.100006	87.00	3.150000	49.639999	87.00	21.38
8	2002-04-01 01:00:00	87.00	1.21	87.00	87.00	87.00	106.300003	126.599998	87.00	22.389999	32.090000	87.00	17.01
9	2002-04-01 01:00:00	87.00	0.61	87.00	87.00	0.14	95.540001	110.699997	87.00	27.770000	24.610001	87.00	19.44

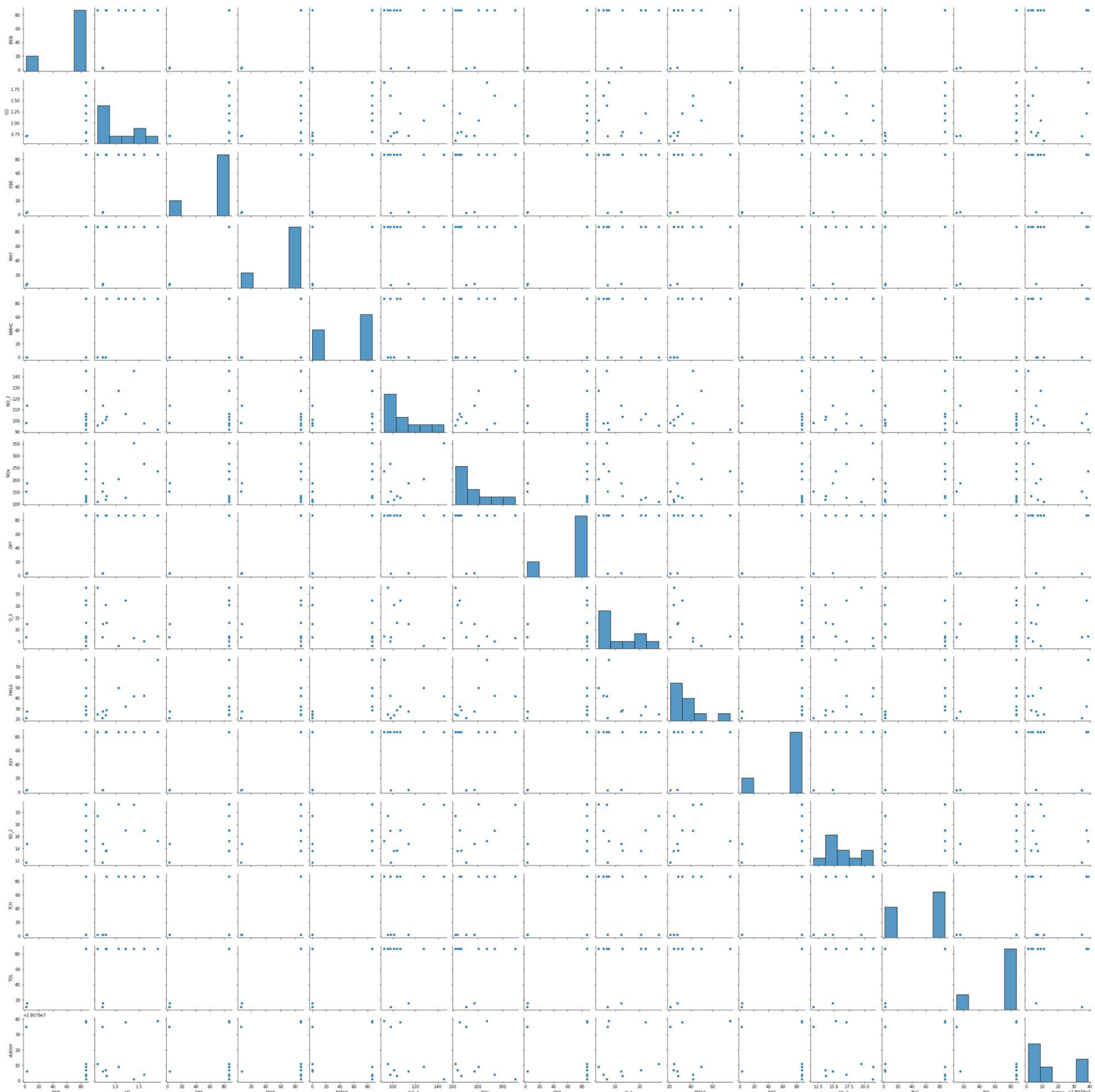
In [14]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[14]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TC
0	87.00	1.39	87.00	87.00	87.00	145.100006	352.100006	87.00	6.540000	41.990002	87.00	21.320000	87.(
1	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000	20.980000	2.53	11.660000	1.8
2	87.00	0.80	87.00	87.00	87.00	103.699997	134.000000	87.00	13.010000	28.440001	87.00	13.670000	87.(
3	87.00	1.61	87.00	87.00	87.00	97.599998	268.000000	87.00	5.120000	42.180000	87.00	16.990000	87.(
4	87.00	1.90	87.00	87.00	87.00	92.089996	237.199997	87.00	7.280000	76.330002	87.00	15.260000	87.(
5	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.370000	27.450001	2.98	14.780000	1.8
6	87.00	0.78	87.00	87.00	0.09	101.000000	119.300003	87.00	20.549999	23.950001	87.00	13.630000	1.7
7	87.00	1.06	87.00	87.00	87.00	127.300003	204.100006	87.00	3.150000	49.639999	87.00	21.389999	87.(
8	87.00	1.21	87.00	87.00	87.00	106.300003	126.599998	87.00	22.389999	32.090000	87.00	17.010000	87.(
9	87.00	0.61	87.00	87.00	0.14	95.540001	110.699997	87.00	27.770000	24.610001	87.00	19.440001	1.8

In [15]:
```
sns.pairplot(d)
```

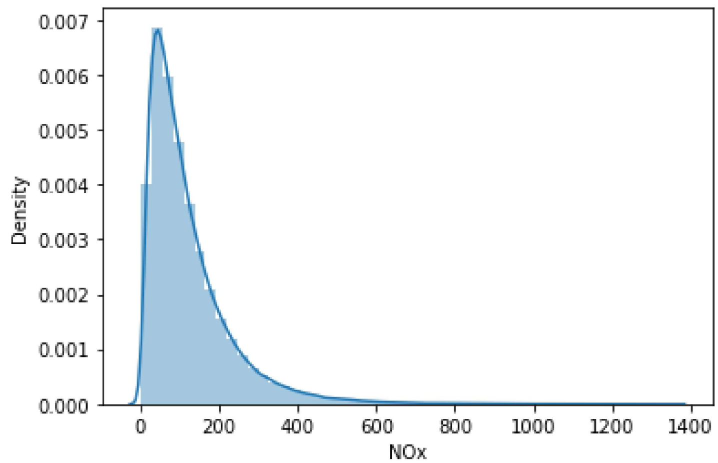
Out[15]: <seaborn.axisgrid.PairGrid at 0x1b61246d400>



In [16]:
```
sns.distplot(a['NOx'])
```

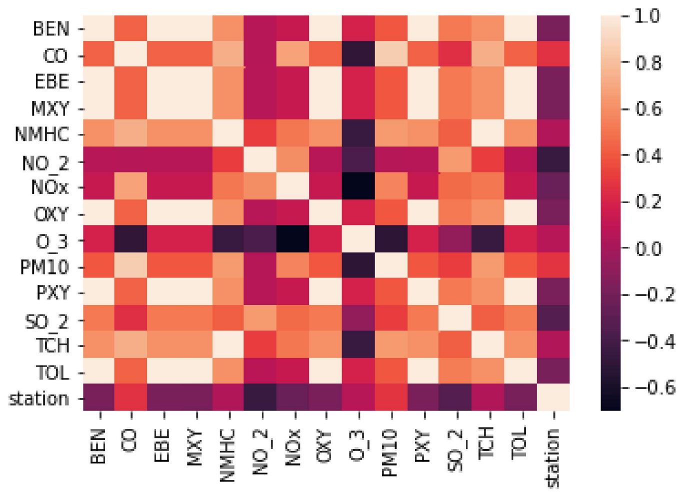
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `
histplot` (an axes-level function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

Out[16]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [17]:
```
sns.heatmap(d.corr())
```

Out[17]: <AxesSubplot:>



In [18]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [19]:
```
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 [20]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[20]: LinearRegression()

In [21]:
```
print(lr.intercept_)
```

```
1.4393744380194065
```

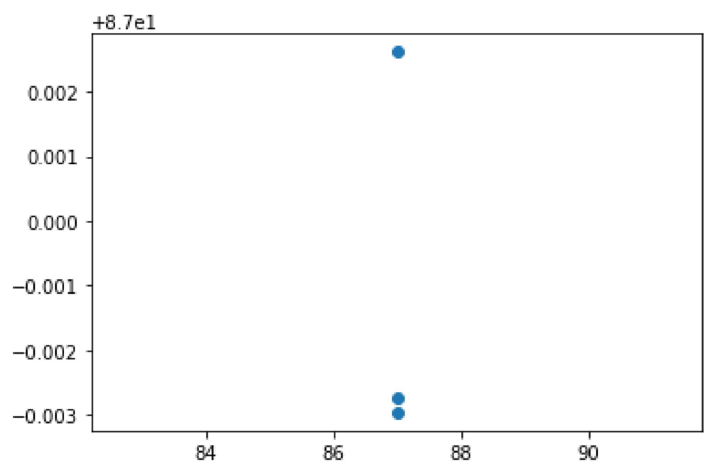
In [22]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[22]:

	Co-efficient
BEN	0.022490
CO	-0.006324
EBE	-0.039470
MXY	0.065601
NMHC	0.980551
NO_2	-0.000039
NOx	0.000036
OXY	-0.045668

In [23]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[23]: <matplotlib.collections.PathCollection at 0x1b62f55a4f0>



In [24]:
```
print(lr.score(x_test,y_test))
```

```
0.0
```

In [25]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [26]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[26]: Ridge(alpha=10)

In [27]:
```
rr.score(x_test,y_test)
```

Out[27]: 0.0

In [28]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[28]: Lasso(alpha=10)

In [29]:
```
la.score(x_test,y_test)
```

Out[29]: 0.0

In [30]:
```
a1=b.head(7000)
a1
```

Out[30]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
0	2002-04-01 01:00:00	66.00	1.39	66.00	66.00	66.00	145.100006	352.100006	66.00	6.540000	41.990002	66.00
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000	20.980000	2.53
2	2002-04-01 01:00:00	66.00	0.80	66.00	66.00	66.00	103.699997	134.000000	66.00	13.010000	28.440001	66.00
3	2002-04-01 01:00:00	66.00	1.61	66.00	66.00	66.00	97.599998	268.000000	66.00	5.120000	42.180000	66.00
4	2002-04-01 01:00:00	66.00	1.90	66.00	66.00	66.00	92.089996	237.199997	66.00	7.280000	76.330002	66.00
...	...	...	...	...	...	...	...	...	...	...	...	...
6995	2002-04-12 16:00:00	2.58	0.79	66.00	66.00	66.00	81.639999	146.500000	66.00	25.570000	28.570000	66.00
6996	2002-04-12 16:00:00	1.73	0.71	1.31	66.00	0.13	59.470001	108.500000	66.00	32.320000	20.209999	66.00
6997	2002-04-12 16:00:00	0.81	0.57	0.66	1.27	0.12	31.200001	36.630001	0.69	45.160000	8.810000	0.51
6998	2002-04-12 16:00:00	1.62	0.58	1.62	3.69	66.00	64.000000	150.100006	1.13	26.110001	66.000000	1.36
6999	2002-04-12 16:00:00	2.19	0.95	2.21	6.75	0.15	65.169998	117.099998	2.96	33.279999	22.540001	2.61

7000 rows × 16 columns

In [31]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [32]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [33]:
```
h=StandardScaler().fit_transform(f)
```

In [34]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```
Out[34]:
```
LogisticRegression(max_iter=10000)
```

In [35]:
```
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 [36]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [37]:
```
prediction=logr.predict(i)
print(prediction)
```
```
[28079038]
```

In [38]:
```
logr.classes_
```
Out[38]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079009,
       28079011, 28079012, 28079014, 28079015, 28079016, 28079017,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079035, 28079036, 28079038, 28079039, 28079040,
       28079099], dtype=int64)
```

In [39]:
```
logr.predict_proba(i)[0][0]
```
Out[39]:
```
6.805698515856327e-40
```

In [40]:
```
logr.predict_proba(i)[0][1]
```
Out[40]:
```
1.2906478801938545e-06
```

In [41]:
```
logr.score(h_test,g_test)
```
Out[41]:
```
0.5819047619047619
```

In [47]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[47]:
```
ElasticNet()
```

In [48]:
```
print(en.coef_)
```
```
[2.42956152e-06 0.00000000e+00 5.16727183e-05 0.00000000e+00
 9.79815938e-01 0.00000000e+00 2.06109883e-04 0.00000000e+00]
```

In [49]:
```
print(en.intercept_)
```
```
1.6830868582140894
```

In [50]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```
```
0.0
```

In [51]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[51]:
```
RandomForestClassifier()
```

In [52]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [53]:
```
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[53]:
```
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')
```

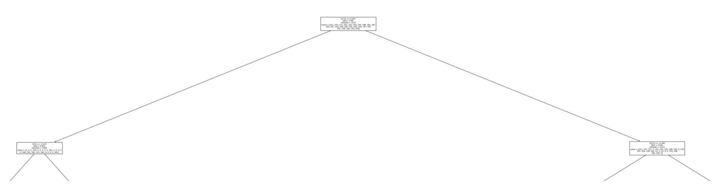
In [54]:
```
grid_search.best_score_
```

Out[54]:
```
0.556734693877551
```

In [55]:
```
rfc_best=grid_search.best_estimator_
```

In [56]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
 Text(4159.636363636364, 1132.5, 'X[8] <= -1.559\ngini = 0.919\nsamples = 1406\nvalue =
[207, 197, 163, 1, 0, 195, 0, 188, 182, 0, 194\n45, 0, 128, 181, 7, 0, 0, 0, 0, 191, 19
6, 124\n8, 0]'),
 Text(4058.181818181818, 679.5, 'gini = 0.0\nsamples = 15\nvalue = [0, 0, 0, 0, 0, 30,
0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(4261.090909090909, 679.5, 'X[8] <= 0.614\ngini = 0.919\nsamples = 1391\nvalue =
[207, 197, 163, 1, 0, 165, 0, 188, 182, 0, 194\n45, 0, 128, 181, 7, 0, 0, 0, 0, 191, 19
6, 124\n8, 0]'),
 Text(4159.636363636364, 226.5, 'gini = 0.916\nsamples = 1015\nvalue = [167, 135, 138,
1, 0, 140, 0, 186, 104, 0, 94, 43\n0, 93, 136, 7, 0, 0, 0, 0, 163, 116, 59, 6, 0]'),
 Text(4362.545454545454, 226.5, 'gini = 0.892\nsamples = 376\nvalue = [40, 62, 25, 0,
0, 25, 0, 2, 78, 0, 100, 2, 0\n35, 45, 0, 0, 0, 0, 0, 28, 80, 65, 2, 0]')]
```



Conclusion: from this data set i observed that the linear rwegression has the highest accuracy of 1.4393744380194065

In [ ]: