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 [218]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2006
a
```

Out[218]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570000	40.2
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000	
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419998	
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260000	
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180000	
...	...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120003	
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469999	15.6
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680000	35.0
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360001	
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490002	27.9

230568 rows × 17 columns

In [219]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      73979 non-null   float64
 2   CO       211665 non-null  float64
 3   EBE      73948 non-null   float64
 4   MXY      33422 non-null   float64
 5   NMHC     90829 non-null   float64
 6   NO_2     228855 non-null  float64
 7   NOx      228855 non-null  float64
 8   OXY      33472 non-null   float64
 9   O_3      216511 non-null  float64
 10  PM10     227469 non-null  float64
 11  PM25     61758 non-null   float64
 12  PXY      33447 non-null   float64
 13  SO_2     229125 non-null  float64
 14  TCH      90887 non-null   float64
 15  TOL      73840 non-null   float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

In [261]:

```
b=a.fillna(value=60)
b
```

Out[261]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2006-02-01 01:00:00	60.00	1.84	60.00	60.00	60.00	155.100006	490.100006	60.00	4.880000	97.570000
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000
2	2006-02-01 01:00:00	60.00	1.25	60.00	60.00	60.00	66.800003	192.000000	60.00	4.430000	34.419998
3	2006-02-01 01:00:00	60.00	1.68	60.00	60.00	60.00	103.000000	407.799988	60.00	4.830000	28.260000
4	2006-02-01 01:00:00	60.00	1.31	60.00	60.00	60.00	105.400002	269.200012	60.00	6.990000	54.180000
...	...	...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	60.00	0.20	112.500000	218.000000	60.00	24.389999	93.120003
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469999
230565	2006-05-01 00:00:00	0.96	60.00	0.69	60.00	0.19	135.100006	179.199997	60.00	11.460000	64.680000
230566	2006-05-01 00:00:00	0.50	60.00	0.67	60.00	0.10	82.599998	105.599998	60.00	60.000000	94.360001
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490002

230568 rows × 17 columns

In [262]:

```
b.columns
```

Out[262]:

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

In [263]:
```
c=b.head(10)
c
```

Out[263]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM
0	2006-02-01 01:00:00	60.00	1.84	60.00	60.000000	60.00	155.100006	490.100006	60.00	4.88	97.570000	40.2599
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.360000	0.32	94.339996	229.699997	3.04	7.10	25.820000	60.0000
2	2006-02-01 01:00:00	60.00	1.25	60.00	60.000000	60.00	66.800003	192.000000	60.00	4.43	34.419998	60.0000
3	2006-02-01 01:00:00	60.00	1.68	60.00	60.000000	60.00	103.000000	407.799988	60.00	4.83	28.260000	60.0000
4	2006-02-01 01:00:00	60.00	1.31	60.00	60.000000	60.00	105.400002	269.200012	60.00	6.99	54.180000	60.0000
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99	89.190002	43.1500
6	2006-02-01 01:00:00	60.00	1.28	60.00	60.000000	0.57	94.320000	294.000000	60.00	6.77	55.130001	60.0000
7	2006-02-01 01:00:00	0.27	1.51	0.28	60.000000	0.46	144.699997	385.299988	60.00	5.30	80.150002	60.0000
8	2006-02-01 01:00:00	60.00	2.65	60.00	60.000000	60.00	197.100006	673.099976	60.00	2.64	142.500000	60.0000
9	2006-02-01 01:00:00	60.00	1.30	60.00	60.000000	60.00	130.899994	282.000000	60.00	5.14	49.029999	24.0100

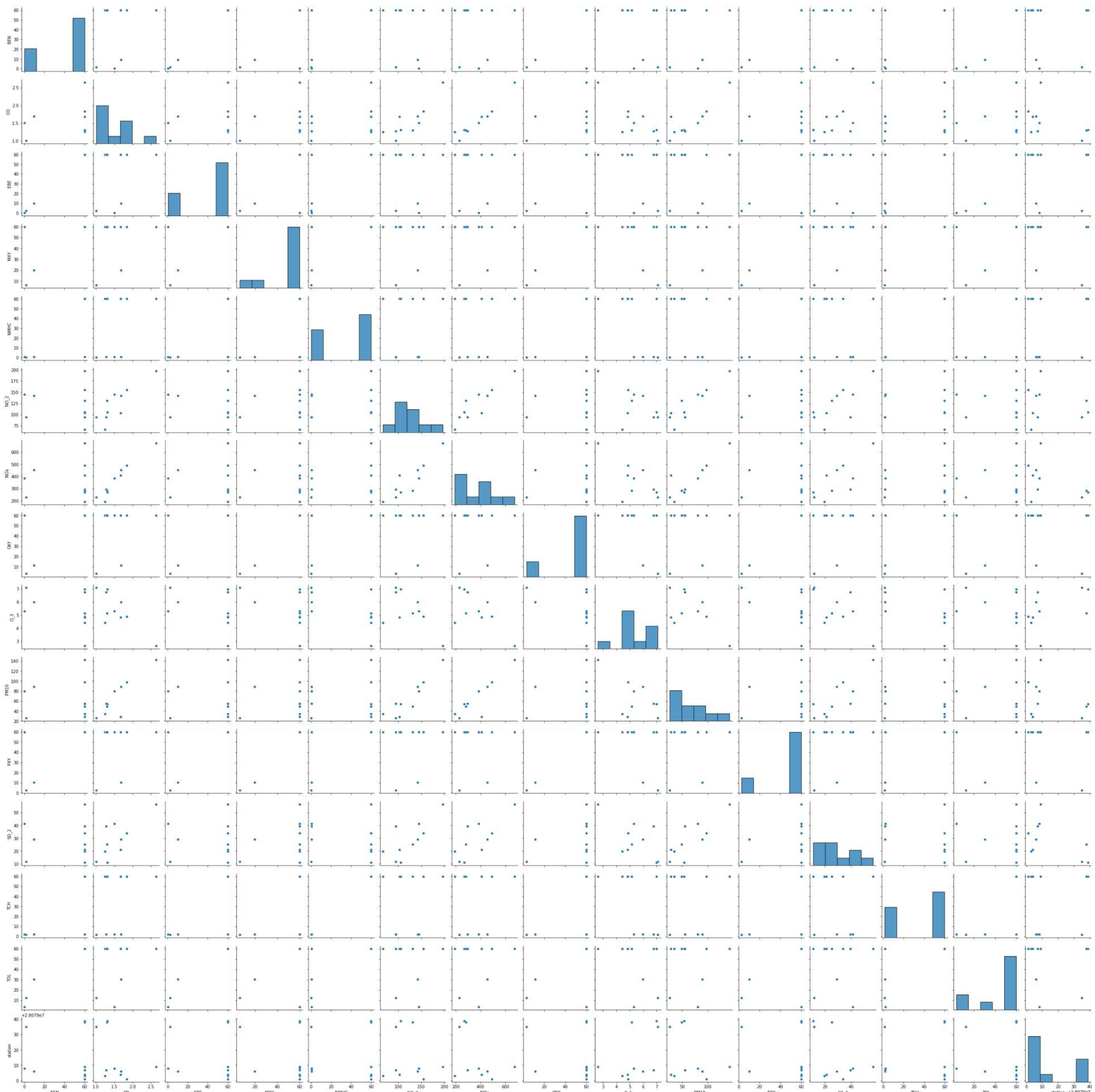
In [264]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[264]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	60.00	1.84	60.00	60.000000	60.00	155.100006	490.100006	60.00	4.88	97.570000	60.00	33.779999
1	1.68	1.01	2.38	6.360000	0.32	94.339996	229.699997	3.04	7.10	25.820000	2.48	11.890000
2	60.00	1.25	60.00	60.000000	60.00	66.800003	192.000000	60.00	4.43	34.419998	60.00	19.719999
3	60.00	1.68	60.00	60.000000	60.00	103.000000	407.799988	60.00	4.83	28.260000	60.00	21.129999
4	60.00	1.31	60.00	60.000000	60.00	105.400002	269.200012	60.00	6.99	54.180000	60.00	11.050000
5	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99	89.190002	10.11	28.990000
6	60.00	1.28	60.00	60.000000	0.57	94.320000	294.000000	60.00	6.77	55.130001	60.00	39.299999
7	0.27	1.51	0.28	60.000000	0.46	144.699997	385.299988	60.00	5.30	80.150002	60.00	41.400002
8	60.00	2.65	60.00	60.000000	60.00	197.100006	673.099976	60.00	2.64	142.500000	60.00	56.509998
9	60.00	1.30	60.00	60.000000	60.00	130.899994	282.000000	60.00	5.14	49.029999	60.00	25.129999

In [265]:
```
sns.pairplot(d)
```

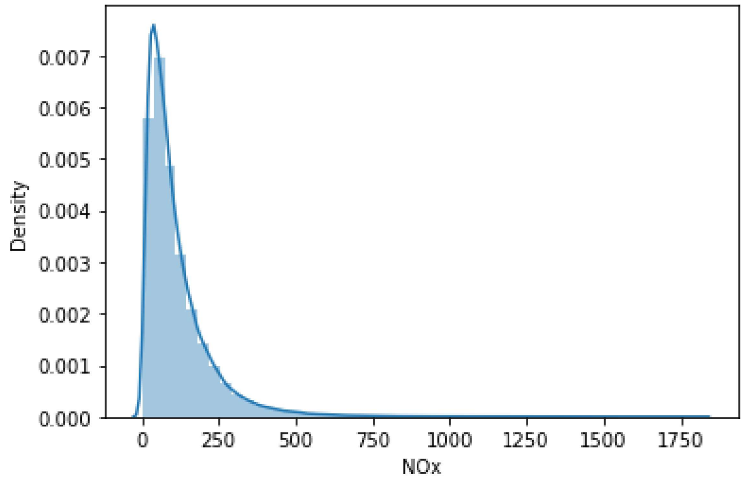
Out[265]: <seaborn.axisgrid.PairGrid at 0x1b68b7cd3d0>



In [266]:
```
sns.distplot(a['NOx'])
```

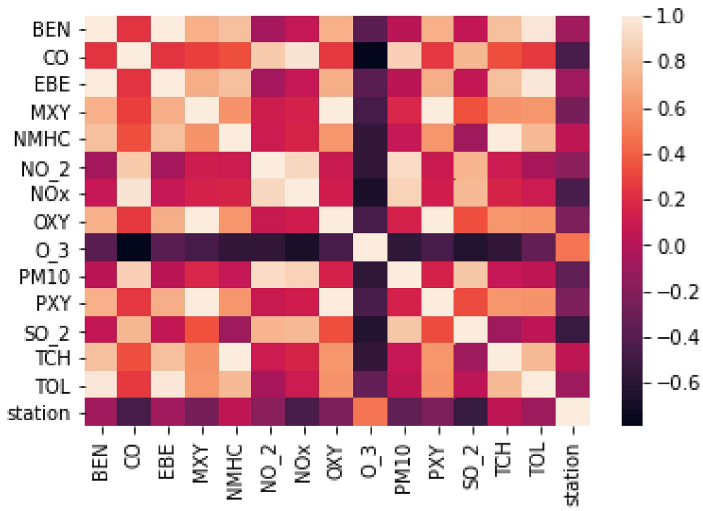
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[266]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [267]:
```
sns.heatmap(d.corr())
```

Out[267]: <AxesSubplot:>



In [268]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [269]:
```
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 [270]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[270]: LinearRegression()

In [271]:
```
print(lr.intercept_)
```

```
1.7841336927158977
```

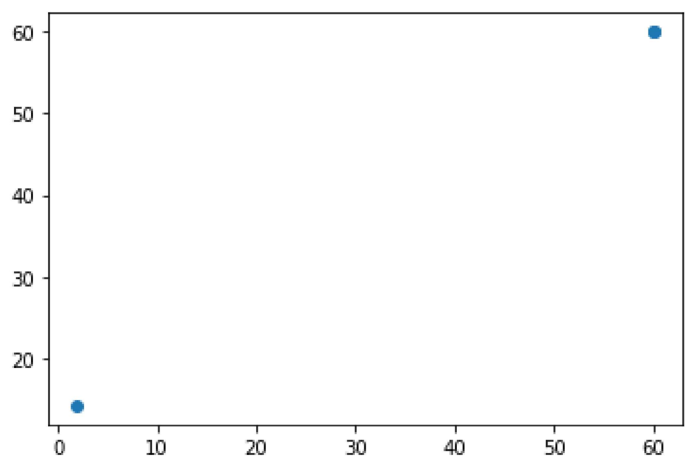
In [272]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[272]:

	Co-efficient
BEN	1.013620e-01
CO	-4.007668e-14
EBE	1.006963e-01
MXY	-2.545772e-01
NMHC	7.699373e-01
NO_2	-2.870174e-16
NOx	1.494060e-16
OXY	2.528460e-01

In [273]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[273]: <matplotlib.collections.PathCollection at 0x1b69f2a6340>



In [274]:
```
print(lr.score(x_test,y_test))
```

```
0.93204609328714
```

In [275]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [276]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[276]: Ridge(alpha=10)

In [277]:
```
rr.score(x_test,y_test)
```

Out[277]: 0.8329333580798355

In [278]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[278]:
```
Lasso(alpha=10)
```

In [279]:
```
la.score(x_test,y_test)
```
Out[279]:
```
0.9997172818812317
```

In [280]:
```
a1=b.head(7000)
a1
```
Out[280]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM2
0	2006-02-01 01:00:00	60.00	1.84	60.00	60.00	60.00	155.100006	490.100006	60.00	4.88	97.570000	40.25999
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.10	25.820000	60.00000
2	2006-02-01 01:00:00	60.00	1.25	60.00	60.00	60.00	66.800003	192.000000	60.00	4.43	34.419998	60.00000
3	2006-02-01 01:00:00	60.00	1.68	60.00	60.00	60.00	103.000000	407.799988	60.00	4.83	28.260000	60.00000
4	2006-02-01 01:00:00	60.00	1.31	60.00	60.00	60.00	105.400002	269.200012	60.00	6.99	54.180000	60.00000
...	...	...	...	...	...	...	...	...	...	...	...	.
6995	2006-02-12 06:00:00	1.54	0.44	2.80	7.86	0.19	61.410000	84.349998	2.85	8.77	12.230000	60.00000
6996	2006-02-12 06:00:00	60.00	0.46	60.00	60.00	60.00	53.340000	75.160004	60.00	7.88	10.200000	60.00000
6997	2006-02-12 06:00:00	60.00	1.06	60.00	60.00	60.00	73.279999	231.899994	60.00	4.38	22.160000	60.00000
6998	2006-02-12 06:00:00	60.00	0.57	60.00	60.00	60.00	47.400002	52.240002	60.00	15.17	28.490000	60.00000
6999	2006-02-12 06:00:00	2.12	0.66	2.39	4.76	0.11	74.879997	163.600006	2.69	8.16	27.770000	19.55999

7000 rows × 17 columns

In [281]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [282]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [283]:
```
h=StandardScaler().fit_transform(f)
```

In [284]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[284]:
```
LogisticRegression(max_iter=10000)
```

In [285]:
```
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 [286]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58,58,29,78]]
```

In [287]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079039]
```

In [288]:
```
logr.classes_
```

Out[288]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079026, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [289]:
```
logr.predict_proba(i)[0][0]
```

Out[289]:
```
1.2401677708906995e-76
```

In [290]:
```
logr.predict_proba(i)[0][1]
```

Out[290]:
```
1.7276678823793815e-128
```

In [291]:
```
logr.score(h_test,g_test)
```

Out[291]:
```
0.56
```

In [292]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[292]:
```
ElasticNet()
```

In [293]:
```
print(en.coef_)
```

```
[ 3.63595980e-02  0.00000000e+00  7.60390821e-02  0.00000000e+00
  8.60453177e-01  0.00000000e+00 -6.89881428e-04  1.69675997e-02]
```

In [294]:
```
print(en.intercept_)
```

```
0.900645067964291
```

In [295]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9777319877517467
```

In [296]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[296]:
```
RandomForestClassifier()
```

In [297]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [298]:
```
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[298]:
```
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 [299]:
```
grid_search.best_score_
```

Out[299]:
```
0.5604081632653062
```

In [300]:
```
rfc_best=grid_search.best_estimator_
```

In [301]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
9, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2824.1632653061224, 679.5, 'X[1] <= -0.295\ngini = 0.47\nsamples = 132\nvalue
= [0, 0, 0, 0, 135, 0, 0, 82, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(2733.061224489796, 226.5, 'gini = 0.498\nsamples = 93\nvalue = [0, 0, 0, 0, 6
8, 0, 0, 78, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2915.265306122449, 226.5, 'gini = 0.106\nsamples = 39\nvalue = [0, 0, 0, 0, 6
7, 0, 0, 4, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3735.183673469388, 1585.5, 'X[8] <= -0.644\ngini = 0.933\nsamples = 1797\nval
ue = [194, 173, 205, 0, 0, 0, 216, 0, 206, 168, 0, 186\n186, 175, 201, 174, 0, 0,
0, 0, 0, 201, 181, 168\n196, 0]'),
 Text(3370.775510204082, 1132.5, 'X[1] <= -0.322\ngini = 0.533\nsamples = 142\nvalu
e = [0, 0, 0, 0, 0, 0, 109, 0, 113, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 7, 0, 1, 0,
0]'),
 Text(3188.571428571429, 679.5, 'X[1] <= -0.341\ngini = 0.148\nsamples = 33\nvalue
= [0, 0, 0, 0, 0, 0, 2, 0, 47, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0]'),
 Text(3097.469387755102, 226.5, 'gini = 0.0\nsamples = 14\nvalue = [0, 0, 0, 0, 0,
0, 0, 0, 21, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
```

Conclusion: from this data set i observed that the LASSO has the highest accuracy of 0.9997172818812317

In [ ]: