In [478]:
```
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 [479]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_201
a
```

Out[479]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	28079004
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	28079011
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	28079056
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	28079057
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	28079058
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	28079059
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	28079060

209928 rows × 14 columns

In [480]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209928 entries, 0 to 209927
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209928 non-null  object 
 1   BEN      51393 non-null   float64
 2   CO       87127 non-null   float64
 3   EBE      51350 non-null   float64
 4   NMHC     43517 non-null   float64
 5   NO       208954 non-null  float64
 6   NO_2     208973 non-null  float64
 7   O_3      122049 non-null  float64
 8   PM10     103743 non-null  float64
 9   PM25     51079 non-null   float64
 10  SO_2     87131 non-null   float64
 11  TCH      43519 non-null   float64
 12  TOL      51175 non-null   float64
 13  station  209928 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [481]:
```
b=a.fillna(value=55)
b
```

Out[481]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	55.0	1.0	55.0	55.00	154.0	84.0	55.0	55.0	55.0	6.0	55.00	55.0	28079004
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
2	2011-11-01 01:00:00	2.9	55.0	3.8	55.00	96.0	99.0	55.0	55.0	55.0	55.0	55.00	7.2	28079011
3	2011-11-01 01:00:00	55.0	0.6	55.0	55.00	60.0	83.0	2.0	55.0	55.0	55.0	55.00	55.0	28079016
4	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	44.0	62.0	3.0	55.0	55.0	3.0	55.00	55.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	55.0	0.2	55.0	55.00	5.0	19.0	44.0	55.0	55.0	55.0	55.00	55.0	28079056
209924	2011-09-01 00:00:00	55.0	0.1	55.0	55.00	6.0	29.0	55.0	11.0	55.0	7.0	55.00	55.0	28079057
209925	2011-09-01 00:00:00	55.0	55.0	55.0	0.23	1.0	21.0	28.0	55.0	55.0	55.0	1.44	55.0	28079058
209926	2011-09-01 00:00:00	55.0	55.0	55.0	55.00	3.0	15.0	48.0	55.0	55.0	55.0	55.00	55.0	28079059
209927	2011-09-01 00:00:00	55.0	55.0	55.0	55.00	4.0	33.0	38.0	13.0	55.0	55.0	55.00	55.0	28079060

209928 rows × 14 columns

In [482]:
```
b.columns
```

Out[482]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [483]:
```
c=b.head(20)
c
```

Out[483]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	55.0	1.0	55.0	55.00	154.0	84.0	55.0	55.0	55.0	6.0	55.00	55.0	28079004
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
2	2011-11-01 01:00:00	2.9	55.0	3.8	55.00	96.0	99.0	55.0	55.0	55.0	55.0	55.00	7.2	28079011
3	2011-11-01 01:00:00	55.0	0.6	55.0	55.00	60.0	83.0	2.0	55.0	55.0	55.0	55.00	55.0	28079016
4	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	44.0	62.0	3.0	55.0	55.0	3.0	55.00	55.0	28079017
5	2011-11-01 01:00:00	0.5	0.8	0.3	55.00	102.0	75.0	2.0	35.0	55.0	5.0	55.00	4.3	28079018
6	2011-11-01 01:00:00	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.7	28079024
7	2011-11-01 01:00:00	55.0	55.0	55.0	0.36	83.0	78.0	6.0	55.0	55.0	55.0	1.80	55.0	28079027
8	2011-11-01 01:00:00	55.0	0.7	55.0	55.00	80.0	91.0	5.0	55.0	55.0	8.0	55.00	55.0	28079035
9	2011-11-01 01:00:00	55.0	0.6	55.0	55.00	63.0	71.0	55.0	33.0	55.0	6.0	55.00	55.0	28079036
10	2011-11-01 01:00:00	0.3	55.0	1.4	55.00	77.0	81.0	55.0	41.0	23.0	5.0	55.00	6.2	28079038
11	2011-11-01 01:00:00	55.0	0.6	55.0	55.00	57.0	82.0	3.0	55.0	55.0	55.0	55.00	55.0	28079039
12	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	9.0	61.0	55.0	25.0	55.0	1.0	55.00	55.0	28079040
13	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	58.0	79.0	55.0	33.0	22.0	55.0	55.00	55.0	28079047
14	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	60.0	85.0	55.0	34.0	20.0	55.0	55.00	55.0	28079048
15	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	57.0	65.0	1.0	55.0	55.0	55.0	55.00	55.0	28079049
16	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	85.0	90.0	55.0	37.0	12.0	55.0	55.00	55.0	28079050
17	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	17.0	51.0	11.0	55.0	55.0	55.0	55.00	55.0	28079054
18	2011-11-01 01:00:00	2.3	55.0	1.9	0.27	110.0	87.0	55.0	39.0	55.0	55.0	1.80	9.5	28079055
19	2011-11-01 01:00:00	55.0	0.8	55.0	55.00	133.0	92.0	7.0	55.0	55.0	55.0	55.00	55.0	28079056

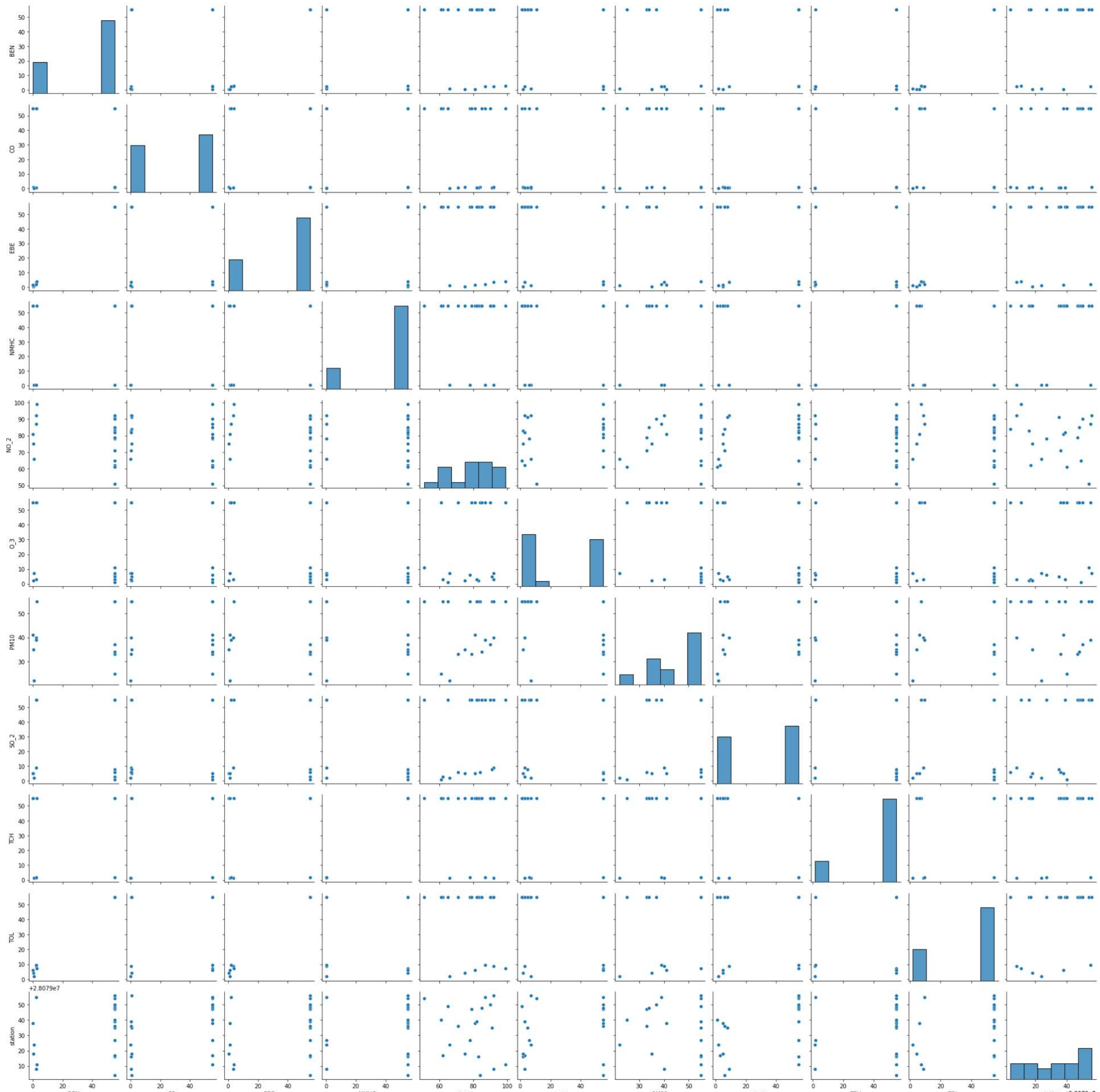
In [520]:
```
d=c[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[520]:

	BEN	CO	EBE	NMHC	NO_2	O_3	PM10	SO_2	TCH	TOL	station
0	55.0	1.0	55.0	55.00	84.0	55.0	55.0	6.0	55.00	55.0	28079004
1	2.5	0.4	3.5	0.26	92.0	3.0	40.0	9.0	1.54	8.7	28079008
2	2.9	55.0	3.8	55.00	99.0	55.0	55.0	55.0	55.00	7.2	28079011
3	55.0	0.6	55.0	55.00	83.0	2.0	55.0	55.0	55.00	55.0	28079016
4	55.0	55.0	55.0	55.00	62.0	3.0	55.0	3.0	55.00	55.0	28079017
5	0.5	0.8	0.3	55.00	75.0	2.0	35.0	5.0	55.00	4.3	28079018
6	0.7	0.3	1.1	0.16	66.0	7.0	22.0	2.0	1.36	1.7	28079024
7	55.0	55.0	55.0	0.36	78.0	6.0	55.0	55.0	1.80	55.0	28079027
8	55.0	0.7	55.0	55.00	91.0	5.0	55.0	8.0	55.00	55.0	28079035
9	55.0	0.6	55.0	55.00	71.0	55.0	33.0	6.0	55.00	55.0	28079036
10	0.3	55.0	1.4	55.00	81.0	55.0	41.0	5.0	55.00	6.2	28079038
11	55.0	0.6	55.0	55.00	82.0	3.0	55.0	55.0	55.00	55.0	28079039
12	55.0	55.0	55.0	55.00	61.0	55.0	25.0	1.0	55.00	55.0	28079040
13	55.0	55.0	55.0	55.00	79.0	55.0	33.0	55.0	55.00	55.0	28079047
14	55.0	55.0	55.0	55.00	85.0	55.0	34.0	55.0	55.00	55.0	28079048
15	55.0	55.0	55.0	55.00	65.0	1.0	55.0	55.0	55.00	55.0	28079049
16	55.0	55.0	55.0	55.00	90.0	55.0	37.0	55.0	55.00	55.0	28079050
17	55.0	55.0	55.0	55.00	51.0	11.0	55.0	55.0	55.00	55.0	28079054
18	2.3	55.0	1.9	0.27	87.0	55.0	39.0	55.0	1.80	9.5	28079055
19	55.0	0.8	55.0	55.00	92.0	7.0	55.0	55.0	55.00	55.0	28079056

In [521]:
```
sns.pairplot(d)
```

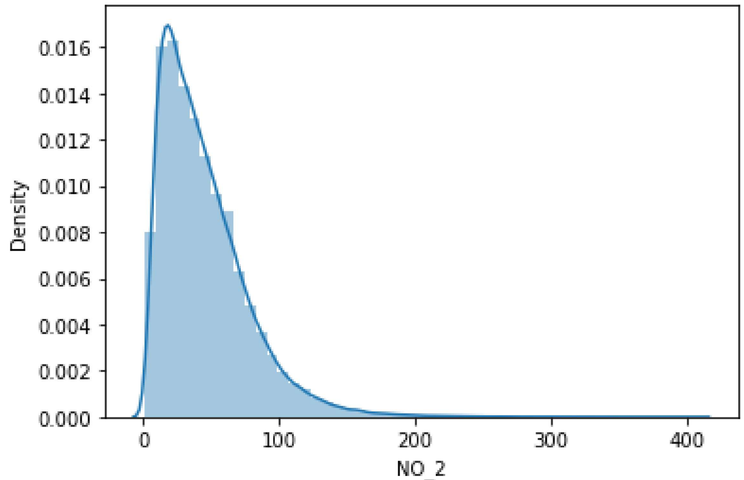
Out[521]:
```
<seaborn.axisgrid.PairGrid at 0x1b700200f40>
```



In [523]:
```
sns.distplot(a['NO_2'])
```

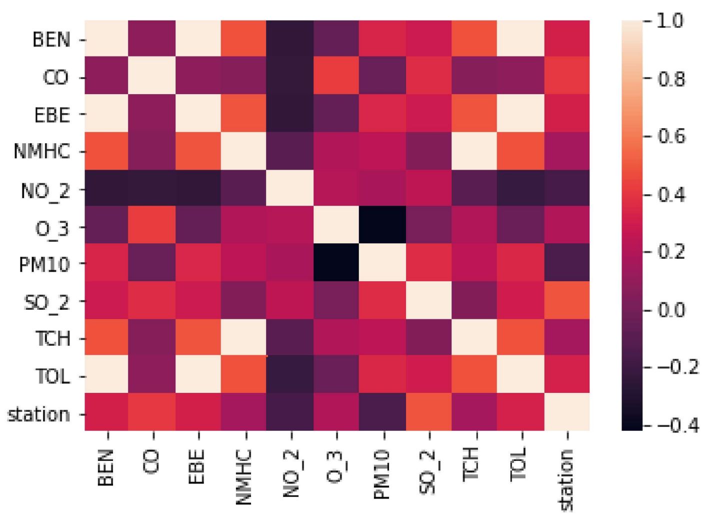
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[523]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



In [524]:
```
sns.heatmap(d.corr())
```

Out[524]: <AxesSubplot:>



In [525]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
y=d['TCH']
```

In [526]:
```
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 [527]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[527]: LinearRegression()

In [528]:
```
print(lr.intercept_)
```

```
1.368401196214812
```

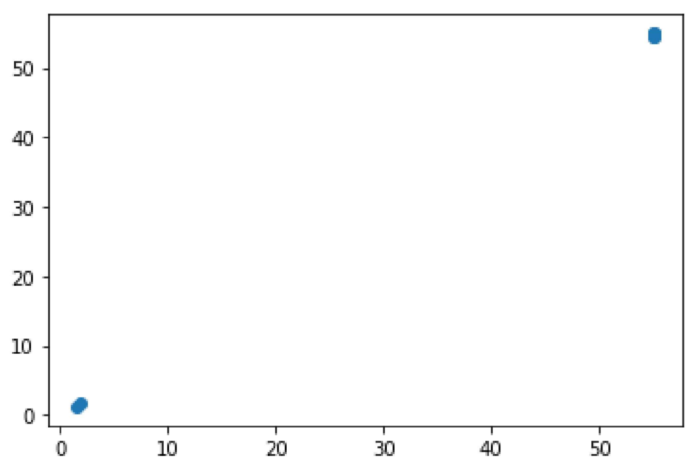
In [529]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[529]:

	Co-efficient
BEN	4.146134e-01
CO	-3.242159e-16
EBE	-4.130974e-01
NMHC	9.736040e-01
NO_2	-1.995580e-15

In [530]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[530]: <matplotlib.collections.PathCollection at 0x1b70ac4f3d0>



In [531]:
```
print(lr.score(x_test,y_test))
```

```
0.9998419489187631
```

In [532]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [533]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[533]: Ridge(alpha=10)

In [534]:
```
rr.score(x_test,y_test)
```

Out[534]: 0.9999752243404992

In [535]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[535]: Lasso(alpha=10)

In [536]:
```
la.score(x_test,y_test)
```
Out[536]: 0.9990877132711634

In [537]:
```
a1=b.head(7000)
a1
```
Out[537]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	55.0	1.0	55.0	55.00	154.0	84.0	55.0	55.0	55.0	6.0	55.00	55.0	28079004
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
2	2011-11-01 01:00:00	2.9	55.0	3.8	55.00	96.0	99.0	55.0	55.0	55.0	55.0	55.00	7.2	28079011
3	2011-11-01 01:00:00	55.0	0.6	55.0	55.00	60.0	83.0	2.0	55.0	55.0	55.0	55.00	55.0	28079016
4	2011-11-01 01:00:00	55.0	55.0	55.0	55.00	44.0	62.0	3.0	55.0	55.0	3.0	55.00	55.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6995	2011-11-13 04:00:00	55.0	0.2	55.0	55.00	1.0	17.0	50.0	55.0	55.0	55.0	55.00	55.0	28079039
6996	2011-11-13 04:00:00	55.0	55.0	55.0	55.00	1.0	6.0	55.0	11.0	55.0	1.0	55.00	55.0	28079040
6997	2011-11-13 04:00:00	55.0	55.0	55.0	55.00	1.0	14.0	55.0	12.0	8.0	55.0	55.00	55.0	28079047
6998	2011-11-13 04:00:00	55.0	55.0	55.0	55.00	2.0	16.0	55.0	8.0	4.0	55.0	55.00	55.0	28079048
6999	2011-11-13 04:00:00	55.0	55.0	55.0	55.00	1.0	8.0	57.0	55.0	55.0	55.0	55.00	55.0	28079049

7000 rows × 14 columns

In [541]:
```
e=a1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [542]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [543]:
```
h=StandardScaler().fit_transform(f)
```

In [544]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```
Out[544]: LogisticRegression(max_iter=10000)

In [545]:
```
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 [549]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58]]
```

In [550]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079059]
```

In [551]:
```
logr.classes_
```

Out[551]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [552]:
```
logr.predict_proba(i)[0][0]
```

Out[552]:
```
0.0
```

In [553]:
```
logr.predict_proba(i)[0][1]
```

Out[553]:
```
0.0
```

In [554]:
```
logr.score(h_test,g_test)
```

Out[554]:
```
0.9742857142857143
```

In [555]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[555]:
```
ElasticNet()
```

In [556]:
```
print(en.coef_)
```

```
[0.00213706 0.         0.         0.97049735 0.        ]
```

In [557]:
```
print(en.intercept_)
```

```
1.493859286726753
```

In [558]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999730240612734
```

In [559]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[559]:
```
RandomForestClassifier()
```

In [560]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [561]:
```
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)
```

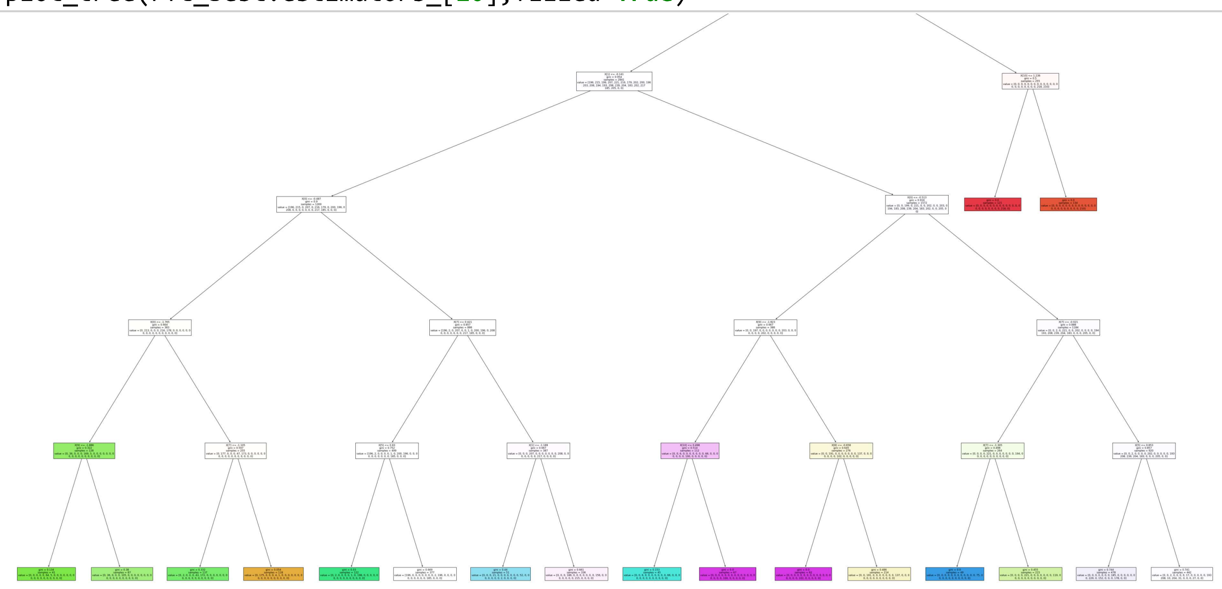
Out[561]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [562]:
```
grid_search.best_score_
```

Out[562]:
```
0.9987755102040816
```

In [563]:
```
rfc_best=grid_search.best_estimator_
```

In [564]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[20],filled=True)
```



Conclusion: from this data set i observed that the ridge has the highest accuracy of 0.9999752243404992

In [ ]:
```

```