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 [565]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2012
a
```

Out[565]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	28079008
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28079011
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28079056
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28079057
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28079058
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28079059
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28079060

210720 rows × 14 columns

In [566]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210720 entries, 0 to 210719
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210720 non-null  object 
 1   BEN      51511 non-null   float64
 2   CO       87097 non-null   float64
 3   EBE      51482 non-null   float64
 4   NMHC     30736 non-null   float64
 5   NO       209871 non-null  float64
 6   NO_2     209872 non-null  float64
 7   O_3      122339 non-null  float64
 8   PM10     104838 non-null  float64
 9   PM25     52164 non-null   float64
 10  SO_2     87333 non-null   float64
 11  TCH      30736 non-null   float64
 12  TOL      51373 non-null   float64
 13  station  210720 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.5+ MB
```

In [567]:
```
b=a.fillna(value=55)
b
```

Out[567]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	7.0	18.0	55.0	55.0	55.0	2.0	55.00	55.0	28079004
1	2012-09-01 01:00:00	0.3	0.3	0.7	55.00	3.0	18.0	55.0	10.0	9.0	1.0	55.00	2.4	28079008
2	2012-09-01 01:00:00	0.4	55.0	0.7	55.00	2.0	10.0	55.0	55.0	55.0	55.0	55.00	1.5	28079011
3	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	1.0	6.0	50.0	55.0	55.0	55.0	55.00	55.0	28079016
4	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	1.0	13.0	54.0	55.0	55.0	3.0	55.00	55.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	55.0	0.6	55.0	55.00	37.0	84.0	14.0	55.0	55.0	55.0	55.00	55.0	28079056
210716	2012-03-01 00:00:00	55.0	0.4	55.0	55.00	5.0	76.0	55.0	17.0	55.0	7.0	55.00	55.0	28079057
210717	2012-03-01 00:00:00	55.0	55.0	55.0	0.34	3.0	41.0	24.0	55.0	55.0	55.0	1.34	55.0	28079058
210718	2012-03-01 00:00:00	55.0	55.0	55.0	55.00	2.0	44.0	36.0	55.0	55.0	55.0	55.00	55.0	28079059
210719	2012-03-01 00:00:00	55.0	55.0	55.0	55.00	2.0	56.0	40.0	18.0	55.0	55.0	55.00	55.0	28079060

210720 rows × 14 columns

In [568]:
```
b.columns
```

Out[568]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [569]:
```
c=b.head(20)
c
```

Out[569]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	7.0	18.0	55.0	55.0	55.0	2.0	55.00	55.0	28079004
1	2012-09-01 01:00:00	0.3	0.3	0.7	55.00	3.0	18.0	55.0	10.0	9.0	1.0	55.00	2.4	28079008
2	2012-09-01 01:00:00	0.4	55.0	0.7	55.00	2.0	10.0	55.0	55.0	55.0	55.0	55.00	1.5	28079011
3	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	1.0	6.0	50.0	55.0	55.0	55.0	55.00	55.0	28079016
4	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	1.0	13.0	54.0	55.0	55.0	3.0	55.00	55.0	28079017
5	2012-09-01 01:00:00	0.2	0.2	1.0	55.00	1.0	9.0	57.0	14.0	55.0	1.0	55.00	0.2	28079018
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6	28079024
7	2012-09-01 01:00:00	55.0	55.0	55.0	0.11	1.0	2.0	65.0	55.0	55.0	55.0	1.18	55.0	28079027
8	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	6.0	14.0	57.0	55.0	55.0	2.0	55.00	55.0	28079035
9	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	1.0	7.0	55.0	13.0	55.0	1.0	55.00	55.0	28079036
10	2012-09-01 01:00:00	0.2	55.0	0.7	55.00	3.0	13.0	55.0	12.0	6.0	1.0	55.00	0.8	28079038
11	2012-09-01 01:00:00	55.0	0.2	55.0	55.00	1.0	8.0	58.0	55.0	55.0	55.0	55.00	55.0	28079039
12	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	1.0	10.0	55.0	15.0	55.0	2.0	55.00	55.0	28079040
13	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	2.0	13.0	55.0	15.0	7.0	55.0	55.00	55.0	28079047
14	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	1.0	9.0	55.0	31.0	9.0	55.0	55.00	55.0	28079048
15	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	1.0	8.0	60.0	55.0	55.0	55.0	55.00	55.0	28079049
16	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	1.0	8.0	55.0	15.0	5.0	55.0	55.00	55.0	28079050
17	2012-09-01 01:00:00	55.0	55.0	55.0	55.00	4.0	9.0	55.0	55.0	55.0	55.0	55.00	55.0	28079054
18	2012-09-01 01:00:00	0.2	55.0	1.0	0.09	2.0	14.0	55.0	11.0	55.0	55.0	1.33	0.3	28079055
19	2012-09-01 01:00:00	55.0	0.3	55.0	55.00	7.0	23.0	48.0	55.0	55.0	55.0	55.00	55.0	28079056

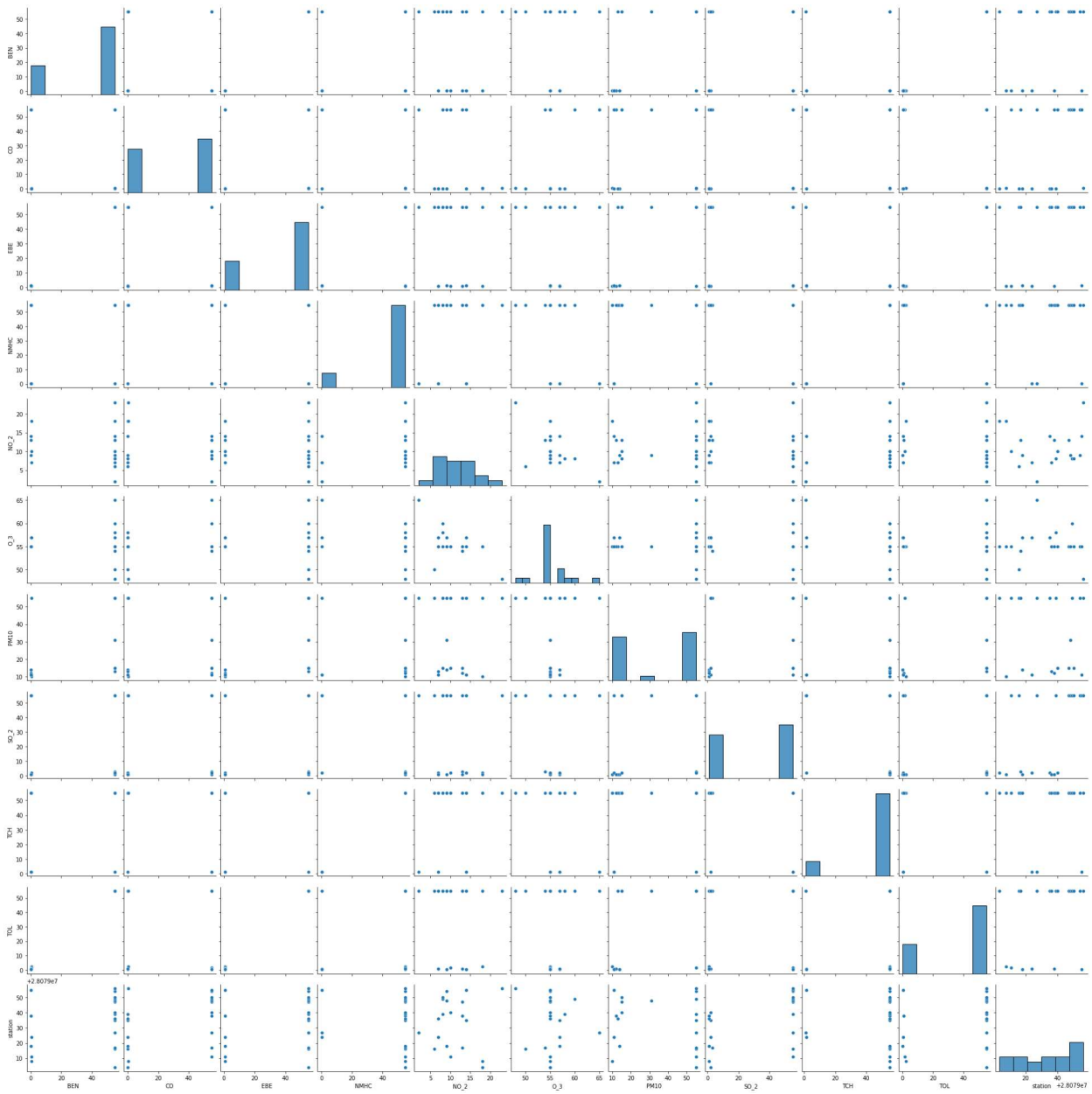
In [570]:
```
d=c[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[570]:

	BEN	CO	EBE	NMHC	NO_2	O_3	PM10	SO_2	TCH	TOL	station
0	55.0	0.2	55.0	55.00	18.0	55.0	55.0	2.0	55.00	55.0	28079004
1	0.3	0.3	0.7	55.00	18.0	55.0	10.0	1.0	55.00	2.4	28079008
2	0.4	55.0	0.7	55.00	10.0	55.0	55.0	55.0	55.00	1.5	28079011
3	55.0	0.2	55.0	55.00	6.0	50.0	55.0	55.0	55.00	55.0	28079016
4	55.0	55.0	55.0	55.00	13.0	54.0	55.0	3.0	55.00	55.0	28079017
5	0.2	0.2	1.0	55.00	9.0	57.0	14.0	1.0	55.00	0.2	28079018
6	0.4	0.2	0.8	0.24	7.0	57.0	11.0	2.0	1.33	0.6	28079024
7	55.0	55.0	55.0	0.11	2.0	65.0	55.0	55.0	1.18	55.0	28079027
8	55.0	0.2	55.0	55.00	14.0	57.0	55.0	2.0	55.00	55.0	28079035
9	55.0	0.2	55.0	55.00	7.0	55.0	13.0	1.0	55.00	55.0	28079036
10	0.2	55.0	0.7	55.00	13.0	55.0	12.0	1.0	55.00	0.8	28079038
11	55.0	0.2	55.0	55.00	8.0	58.0	55.0	55.0	55.00	55.0	28079039
12	55.0	55.0	55.0	55.00	10.0	55.0	15.0	2.0	55.00	55.0	28079040
13	55.0	55.0	55.0	55.00	13.0	55.0	15.0	55.0	55.00	55.0	28079047
14	55.0	55.0	55.0	55.00	9.0	55.0	31.0	55.0	55.00	55.0	28079048
15	55.0	55.0	55.0	55.00	8.0	60.0	55.0	55.0	55.00	55.0	28079049
16	55.0	55.0	55.0	55.00	8.0	55.0	15.0	55.0	55.00	55.0	28079050
17	55.0	55.0	55.0	55.00	9.0	55.0	55.0	55.0	55.00	55.0	28079054
18	0.2	55.0	1.0	0.09	14.0	55.0	11.0	55.0	1.33	0.3	28079055
19	55.0	0.3	55.0	55.00	23.0	48.0	55.0	55.0	55.00	55.0	28079056

In [571]:
```
sns.pairplot(d)
```

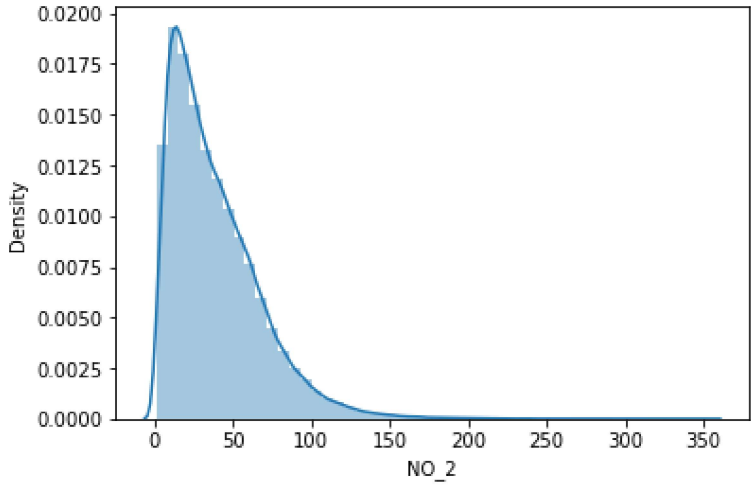
Out[571]: <seaborn.axisgrid.PairGrid at 0x1b70ad97d00>



In [572]:
```
sns.distplot(a['NO_2'])
```

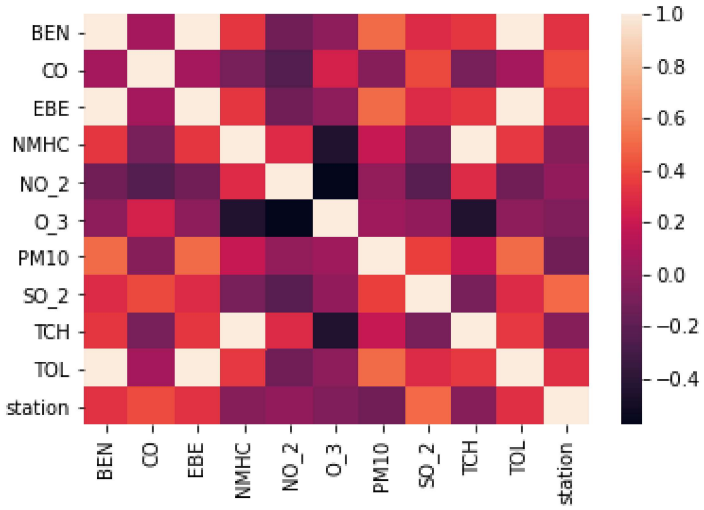
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[572]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



In [573]:
```
sns.heatmap(d.corr())
```

Out[573]: <AxesSubplot:>



In [574]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
y=d['TCH']
```

In [575]:
```
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 [576]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[576]: LinearRegression()

In [577]:
```
print(lr.intercept_)
```

```
1.0226357980782268
```

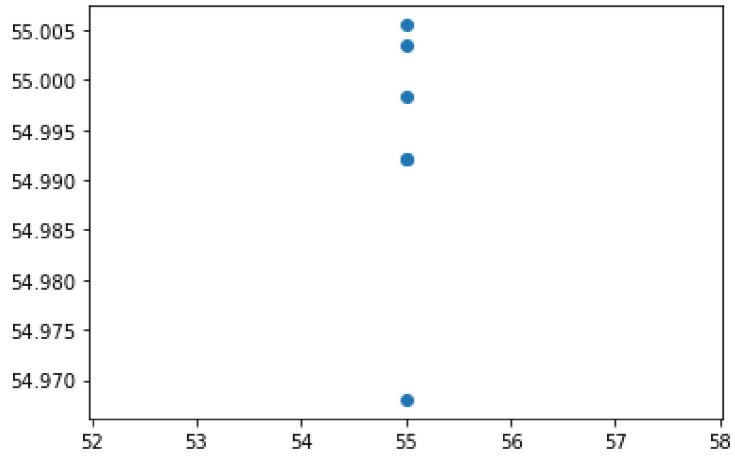
In [578]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[578]:

	Co-efficient
BEN	-0.142889
CO	0.000325
EBE	0.143981
NMHC	0.979391
NO_2	0.003125

In [579]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[579]:
```
<matplotlib.collections.PathCollection at 0x1b710f88eb0>
```



In [580]:
```
print(lr.score(x_test,y_test))
```

```
0.0
```

In [581]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [582]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[582]:
```
Ridge(alpha=10)
```

In [583]:
```
rr.score(x_test,y_test)
```

Out[583]:
```
0.0
```

In [584]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[584]:
```
Lasso(alpha=10)
```

In [585]:
```
la.score(x_test,y_test)
```

Out[585]:
```
0.0
```

In [586]:
```
a1=b.head(7000)
a1
```

Out[586]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	55.0	0.2	55.0	55.0	7.0	18.0	55.0	55.0	55.0	2.0	55.0	55.0	28079004
1	2012-09-01 01:00:00	0.3	0.3	0.7	55.0	3.0	18.0	55.0	10.0	9.0	1.0	55.0	2.4	28079008
2	2012-09-01 01:00:00	0.4	55.0	0.7	55.0	2.0	10.0	55.0	55.0	55.0	55.0	55.0	1.5	28079011
3	2012-09-01 01:00:00	55.0	0.2	55.0	55.0	1.0	6.0	50.0	55.0	55.0	55.0	55.0	55.0	28079016
4	2012-09-01 01:00:00	55.0	55.0	55.0	55.0	1.0	13.0	54.0	55.0	55.0	3.0	55.0	55.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6995	2012-09-13 04:00:00	55.0	0.1	55.0	55.0	1.0	5.0	51.0	55.0	55.0	55.0	55.0	55.0	28079039
6996	2012-09-13 04:00:00	55.0	55.0	55.0	55.0	1.0	6.0	55.0	5.0	55.0	2.0	55.0	55.0	28079040
6997	2012-09-13 04:00:00	55.0	55.0	55.0	55.0	1.0	6.0	55.0	7.0	6.0	55.0	55.0	55.0	28079047
6998	2012-09-13 04:00:00	55.0	55.0	55.0	55.0	1.0	9.0	55.0	5.0	1.0	55.0	55.0	55.0	28079048
6999	2012-09-13 04:00:00	55.0	55.0	55.0	55.0	1.0	5.0	43.0	55.0	55.0	55.0	55.0	55.0	28079049

7000 rows × 14 columns

In [587]:
```
e=a1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [588]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [589]:
```
h=StandardScaler().fit_transform(f)
```

In [590]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[590]:
```
LogisticRegression(max_iter=10000)
```

In [591]:
```
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 [592]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58]]
```

In [593]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079059]
```

In [594]:
```
logr.classes_
```

Out[594]:
```
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 [595]:
```
logr.predict_proba(i)[0][0]
```

Out[595]:
```
0.0
```

In [596]:
```
logr.predict_proba(i)[0][1]
```

Out[596]:
```
0.0
```

In [597]:
```
logr.score(h_test,g_test)
```

Out[597]:
```
0.9323809523809524
```

In [598]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[598]:
```
ElasticNet()
```

In [599]:
```
print(en.coef_)
```

```
[0.         0.         0.         0.97738616 0.        ]
```

In [600]:
```
print(en.intercept_)
```

```
1.2208088274935562
```

In [601]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.0
```

In [602]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[602]:
```
RandomForestClassifier()
```

In [603]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [604]:
```
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)
```

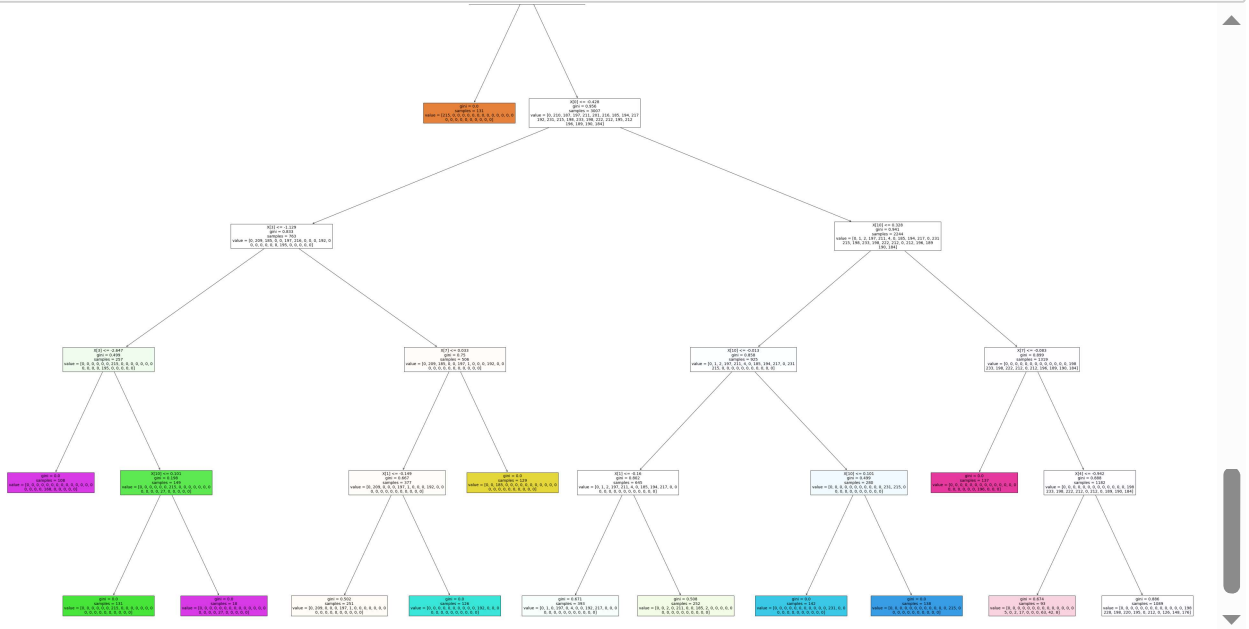
Out[604]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [605]:
```
grid_search.best_score_
```

Out[605]:
```
0.9961224489795919
```

In [606]:
```
rfc_best=grid_search.best_estimator_
```

In [607]:
```
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 0.9961224489795919

In [ ]: