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\mad
a
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.0(
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.0
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.7
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.8(
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.8
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.3

217872 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [4]:
```
b=a.fillna(value=87)
b
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	87.00	0.37	87.00	87.00	87.00	58.400002	87.150002	87.00	34.529999	105.
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.
2	2001-08-01 01:00:00	87.00	0.28	87.00	87.00	87.00	50.660000	61.380001	87.00	46.310001	100.
3	2001-08-01 01:00:00	87.00	0.47	87.00	87.00	87.00	69.790001	73.449997	87.00	40.650002	69.
4	2001-08-01 01:00:00	87.00	0.39	87.00	87.00	87.00	22.830000	24.799999	87.00	66.309998	75.
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	87.00	87.00	87.00	73.000000	264.399994	87.00	5.200000	47.
217868	2001-04-01 00:00:00	5.20	0.69	4.56	87.00	0.13	71.080002	129.300003	87.00	13.460000	26.
217869	2001-04-01 00:00:00	0.49	1.09	87.00	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	87.00	80.019997	197.000000	2.58	5.840000	37.
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.

217872 rows × 16 columns

In [5]:
```
b.columns
```

Out[5]:
```
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	PM1(
0	2001-08-01 01:00:00	87.00	0.37	87.00	87.00	87.00	58.400002	87.150002	87.00	34.529999	105.00000(
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.59999
2	2001-08-01 01:00:00	87.00	0.28	87.00	87.00	87.00	50.660000	61.380001	87.00	46.310001	100.09999
3	2001-08-01 01:00:00	87.00	0.47	87.00	87.00	87.00	69.790001	73.449997	87.00	40.650002	69.77999
4	2001-08-01 01:00:00	87.00	0.39	87.00	87.00	87.00	22.830000	24.799999	87.00	66.309998	75.18000(
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.94	0.05	66.260002	118.099998	3.15	33.500000	122.69999
6	2001-08-01 01:00:00	87.00	0.28	87.00	87.00	87.00	35.799999	39.590000	87.00	68.250000	124.90000
7	2001-08-01 01:00:00	87.00	0.67	87.00	87.00	87.00	74.830002	112.000000	87.00	26.410000	113.00000(
8	2001-08-01 01:00:00	87.00	0.41	87.00	87.00	87.00	33.209999	37.299999	87.00	62.299999	125.30000
9	2001-08-01 01:00:00	87.00	0.17	87.00	87.00	0.13	24.129999	36.970001	87.00	46.200001	95.58999(

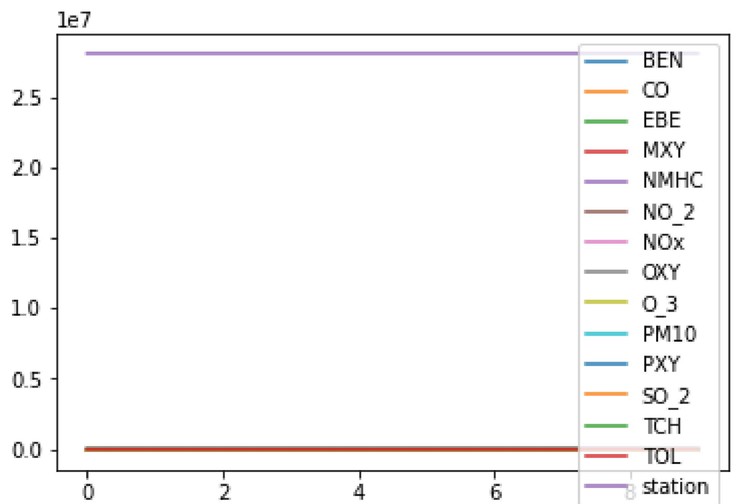
In [8]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[8]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
0	87.00	0.37	87.00	87.00	87.00	58.400002	87.150002	87.00	34.529999	105.000000	87.00
1	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73
2	87.00	0.28	87.00	87.00	87.00	50.660000	61.380001	87.00	46.310001	100.099998	87.00
3	87.00	0.47	87.00	87.00	87.00	69.790001	73.449997	87.00	40.650002	69.779999	87.00
4	87.00	0.39	87.00	87.00	87.00	22.830000	24.799999	87.00	66.309998	75.180000	87.00
5	2.11	0.63	2.48	5.94	0.05	66.260002	118.099998	3.15	33.500000	122.699997	2.29
6	87.00	0.28	87.00	87.00	87.00	35.799999	39.590000	87.00	68.250000	124.900002	87.00
7	87.00	0.67	87.00	87.00	87.00	74.830002	112.000000	87.00	26.410000	113.000000	87.00
8	87.00	0.41	87.00	87.00	87.00	33.209999	37.299999	87.00	62.299999	125.300003	87.00
9	87.00	0.17	87.00	87.00	0.13	24.129999	36.970001	87.00	46.200001	95.589996	87.00

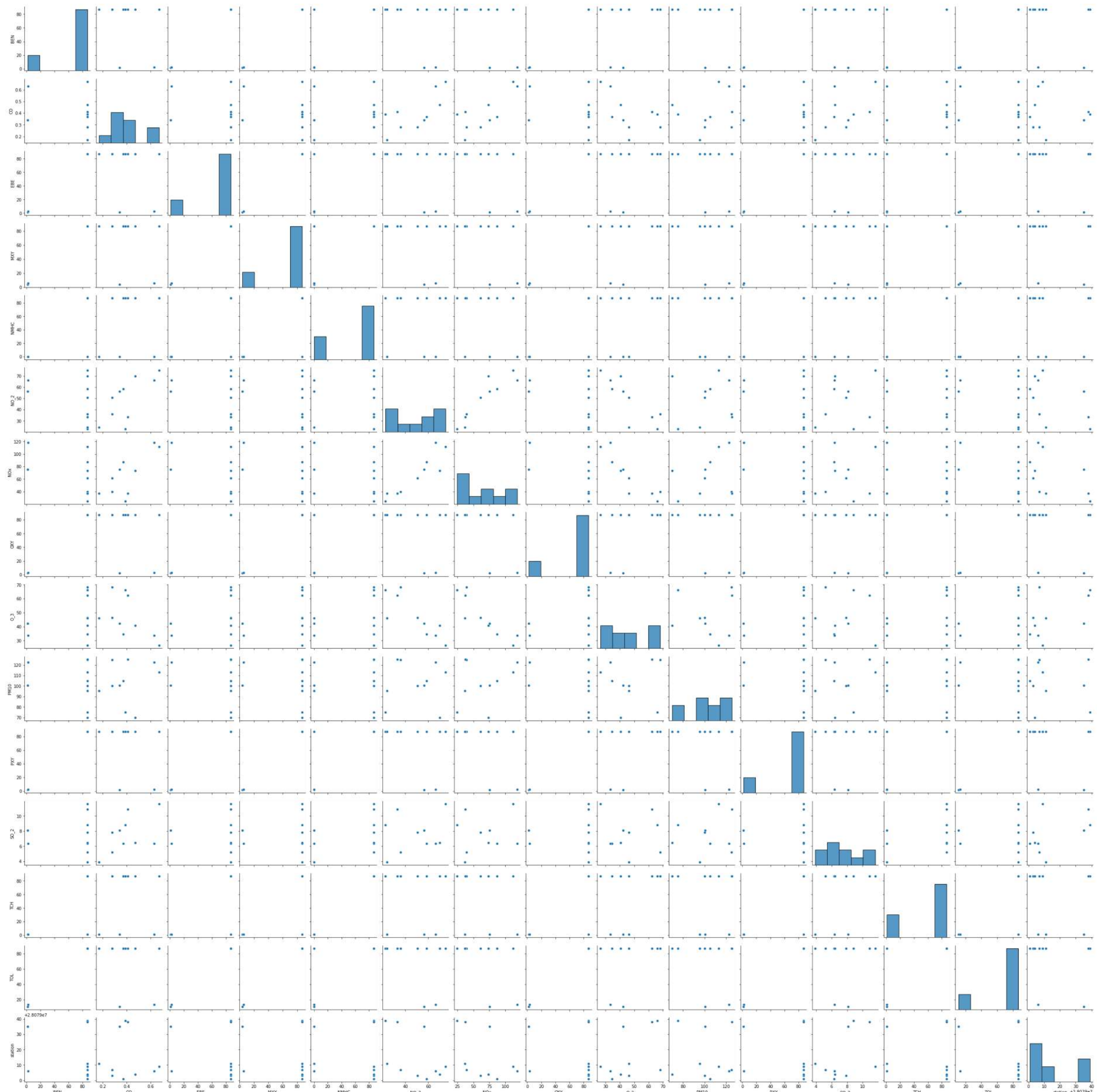
In [9]:
```
d.plot.line()
```

Out[9]: <AxesSubplot:>



In [10]:
```
sns.pairplot(d)
```

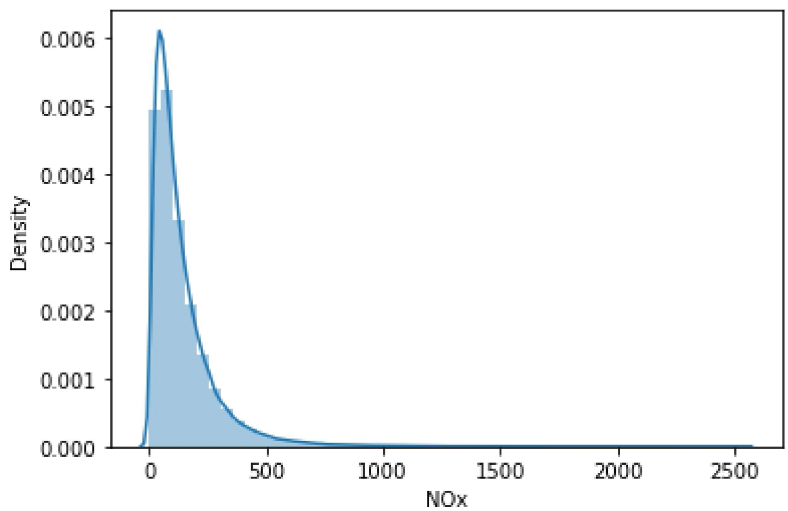
Out[10]: <seaborn.axisgrid.PairGrid at 0x1c3213d7700>



In [11]:
```
sns.distplot(a['NOx'])
```

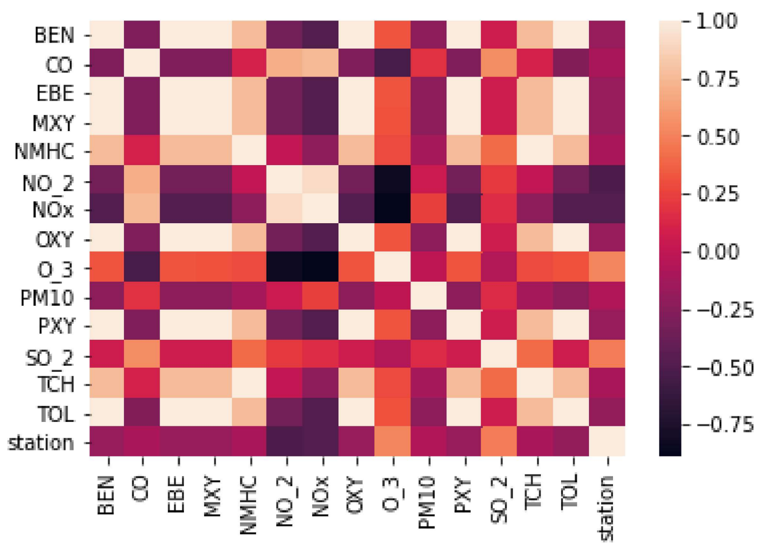
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[11]: <AxesSubplot:xlabel='NOx', ylabel='Density'>



In [12]:
```
sns.heatmap(d.corr())
```

Out[12]: <AxesSubplot:>



linear regression

In [13]:
```
x=d[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY']]
y=d['TCH']
```

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
1.1672524421982615
```

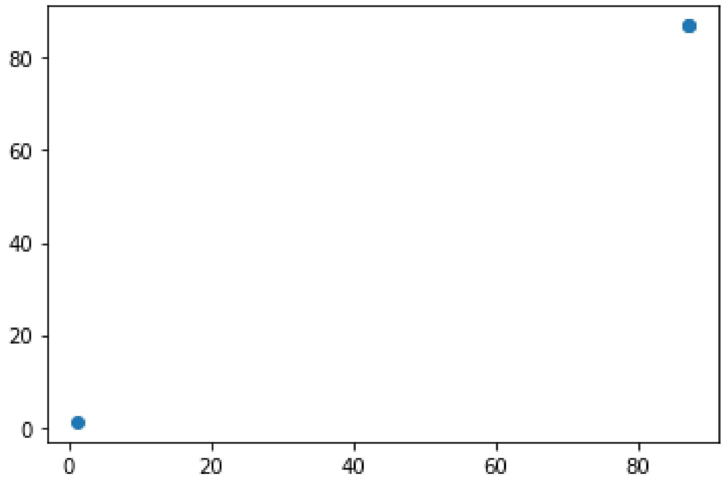
In [17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
BEN	4.197552e-04
CO	-1.288123e-14
EBE	4.198043e-04
MXY	4.069907e-04
NMHC	9.849200e-01
NO_2	-1.384461e-15
NOx	6.467450e-16
OXY	4.167604e-04

In [18]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x1c32f481a60>
```



In [19]:
```
print(lr.score(x_test,y_test))
```

```
0.9999999874528135
```

ridge

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]:
```
0.9999064008491518
```

lasso

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]:
```
Lasso(alpha=10)
```

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

Out[24]: 0.9999455601905967

In [25]:
```
a1=b.head(7000)
a1
```

Out[25]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	87.00	0.37	87.000000	87.0	87.00	58.400002	87.150002	87.00	34.529999	105
1	2001-08-01 01:00:00	1.50	0.34	1.490000	4.1	0.07	56.250000	75.169998	2.11	42.160000	100
2	2001-08-01 01:00:00	87.00	0.28	87.000000	87.0	87.00	50.660000	61.380001	87.00	46.310001	100
3	2001-08-01 01:00:00	87.00	0.47	87.000000	87.0	87.00	69.790001	73.449997	87.00	40.650002	69
4	2001-08-01 01:00:00	87.00	0.39	87.000000	87.0	87.00	22.830000	24.799999	87.00	66.309998	75
...	...	...	...	...	...	...	...	...	...	...	
6995	2001-08-13 04:00:00	87.00	0.00	87.000000	87.0	0.08	18.580000	18.590000	87.00	56.660000	22
6996	2001-08-13 04:00:00	87.00	0.09	87.000000	87.0	87.00	29.580000	32.770000	87.00	52.709999	38
6997	2001-08-13 04:00:00	1.38	0.17	30.530001	87.0	0.25	54.880001	68.870003	87.00	23.240000	18
6998	2001-08-13 04:00:00	87.00	0.01	87.000000	87.0	87.00	19.580000	20.990000	87.00	51.270000	33
6999	2001-08-13 04:00:00	87.00	0.00	87.000000	87.0	0.05	17.200001	18.219999	87.00	38.090000	43

7000 rows × 16 columns

In [26]:
```
e=a1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [27]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [28]:
```
h=StandardScaler().fit_transform(f)
```

logistic regression

In [29]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[29]:
```
LogisticRegression(max_iter=10000)
```

In [30]:
```
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 [31]:
```
i=[[10,20,30,40,50,60,11,22,33,44,55,54,21,78]]
```

In [32]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079021]
```

In [33]:
```
logr.classes_
```

Out[33]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079009,
       28079011, 28079012, 28079014, 28079015, 28079016, 28079018,
       28079019, 28079021, 28079022, 28079023, 28079024, 28079025,
       28079035, 28079036, 28079038, 28079039, 28079040, 28079099],
      dtype=int64)
```

In [34]:
```
logr.predict_proba(i)[0][0]
```

Out[34]:
```
2.528560047135413e-268
```

In [35]:
```
logr.predict_proba(i)[0][1]
```

Out[35]:
```
4.8254923316380694e-139
```

In [36]:
```
logr.score(h_test,g_test)
```

Out[36]:
```
0.6580952380952381
```

elastic net

In [37]:
```
from sklearn.linear_modparameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }train)
```

Out[37]:
```
ElasticNet()
```

In [38]:
```
print(en.coef_)
```

```
[4.03774205e-06 0.00000000e+00 6.75326452e-08 1.59177368e-03
 9.84302947e-01 0.00000000e+00 0.00000000e+00 6.12270211e-05]
```

In [39]:
```
print(en.intercept_)
```

```
1.205669535972298
```

In [40]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999996548442733
```

random forest

In [41]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[41]:
```
RandomForestClassifier()
```

In [42]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [43]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accu
grid_search.fit(h_train,g_train)
```

Out[43]:
```
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 [44]:
```
grid_search.best_score_
```

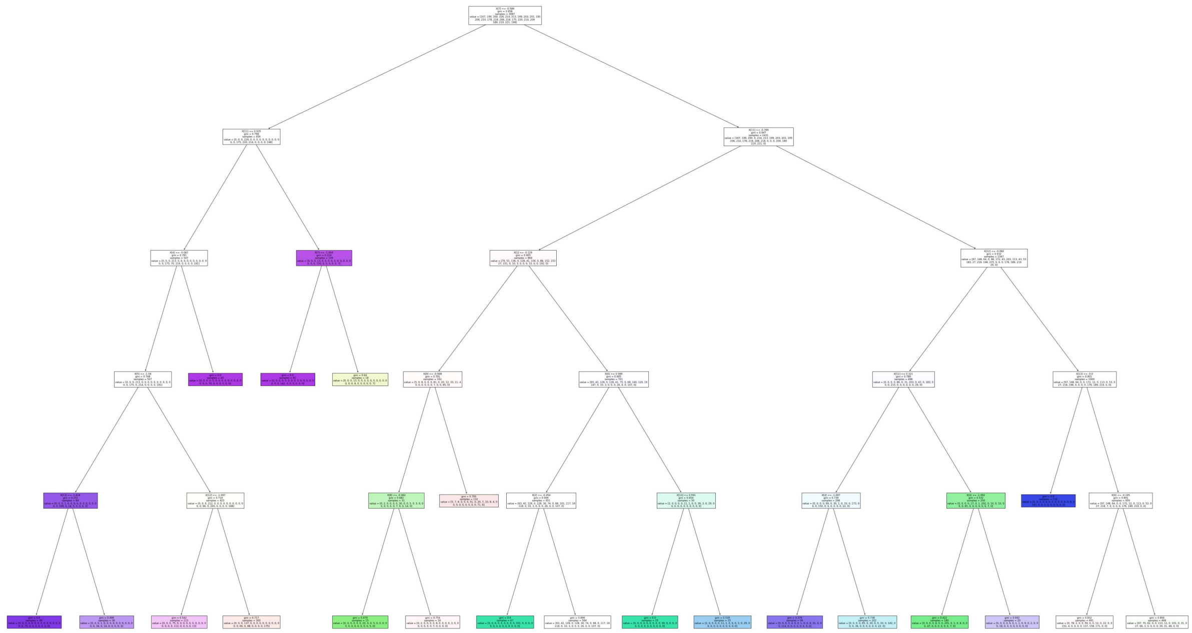
Out[44]:
```
0.6297959183673469
```

In [45]:
```
rfc_best=grid_search.best_estimator_
```

In [47]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
Out[47]: [Text(1893.818181818182, 2491.5, 'X[7] <= -0.586\ngini = 0.958\nsamples = 308
7\nvalue = [167, 199, 200, 226, 214, 213, 199, 203, 201, 195\n206, 210, 178,
218, 208, 218, 175, 220, 214, 209\n189, 219, 221, 198]'),
 Text(946.909090909091, 2038.5, 'X[11] <= 0.935\ngini = 0.798\nsamples = 656
\nvalue = [0, 0, 0, 226, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 175, 220, 214,
0, 0, 0, 0, 198]'),
 Text(676.3636363636364, 1585.5, 'X[4] <= -0.087\ngini = 0.781\nsamples = 547
\nvalue = [0, 0, 0, 213, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 175, 70, 214, 0,
0, 0, 0, 191]'),
 Text(541.0909090909091, 1132.5, 'X[5] <= -1.04\ngini = 0.748\nsamples = 507
\nvalue = [0, 0, 0, 213, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 175, 0, 214, 0,
0, 0, 0, 191]'),
 Text(270.54545454545456, 679.5, 'X[13] <= -1.818\ngini = 0.251\nsamples = 84
\nvalue = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 109, 0, 14, 0, 0,
0, 0, 3]'),
 Text(135.27272727272728, 226.5, 'gini = 0.0\nsamples = 48\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 75, 0, 0, 0, 0, 0, 0, 0]'),
 Text(405.81818181818187, 226.5, 'gini = 0.496\nsamples = 36\nvalue = [0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 34, 0, 14, 0, 0, 0, 0, 3]'),
 Text(811.6363636363637, 679.5, 'X[12] <= -1.097\ngini = 0.719\nsamples = 423
\nvalue = [0, 0, 0, 212, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 66, 0, 200, 0,
0, 0, 0, 188]'),
 Text(676.3636363636364, 226.5, 'gini = 0.542\nsamples = 123\nvalue = [0, 0,
0, 75, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 112, 0, 0, 0, 0, 13]'),
 Text(946.909090909091, 226.5, 'gini = 0.717\nsamples = 300\nvalue = [0, 0,
0, 137, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 66, 0, 88, 0, 0, 0, 0, 175]'),
 Text(811.6363636363637, 1132.5, 'gini = 0.0\nsamples = 40\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 70, 0, 0, 0, 0, 0, 0]'),
 Text(1217.4545454545455, 1585.5, 'X[7] <= -1.954\ngini = 0.214\nsamples = 10
9\nvalue = [0, 0, 0, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 150, 0, 0, 0,
0, 0, 7]'),
 Text(1082.1818181818182, 1132.5, 'gini = 0.0\nsamples = 91\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 142, 0, 0, 0, 0, 0, 0]'),
 Text(1352.7272727272727, 1132.5, 'gini = 0.64\nsamples = 18\nvalue = [0, 0,
0, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 8, 0, 0, 0, 0, 0, 7]'),
 Text(2840.727272727273, 2038.5, 'X[11] <= -0.399\ngini = 0.947\nsamples = 24
31\nvalue = [167, 199, 200, 0, 214, 213, 199, 203, 201, 195\n206, 210, 178, 2
18, 208, 218, 0, 0, 0, 209, 189\n219, 221, 0]'),
 Text(1961.4545454545455, 1585.5, 'X[1] <= -0.129\ngini = 0.905\nsamples = 86
4\nvalue = [70, 51, 136, 0, 128, 41, 156, 0, 88, 152, 153\n27, 151, 0, 10, 3,
0, 0, 0, 33, 0, 0, 192, 0]'),
 Text(1623.2727272727275, 1132.5, 'X[9] <= -0.948\ngini = 0.791\nsamples = 16
3\nvalue = [5, 9, 8, 0, 0, 0, 81, 0, 20, 12, 33, 11, 4\n0, 0, 0, 0, 0, 0, 7,
0, 0, 85, 0]'),
 Text(1488.0, 679.5, 'X[8] <= -0.062\ngini = 0.682\nsamples = 31\nvalue = [0,
2, 0, 0, 0, 0, 30, 0, 0, 5, 0, 3, 0, 0\n0, 0, 0, 0, 7, 0, 0, 14, 0]'),
 Text(1352.7272727272727, 226.5, 'gini = 0.479\nsamples = 15\nvalue = [0, 0,
0, 0, 0, 0, 22, 0, 0, 5, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 5, 0]'),
 Text(1623.2727272727275, 226.5, 'gini = 0.754\nsamples = 16\nvalue = [0, 2,
0, 0, 0, 0, 8, 0, 0, 0, 0, 3, 0, 0\n0, 0, 0, 0, 7, 0, 0, 9, 0]'),
 Text(1758.5454545454547, 679.5, 'gini = 0.795\nsamples = 132\nvalue = [5, 7,
8, 0, 0, 0, 51, 0, 20, 7, 33, 8, 4, 0\n0, 0, 0, 0, 0, 0, 0, 71, 0]'),
 Text(2299.636363636364, 1132.5, 'X[6] <= 0.589\ngini = 0.905\nsamples = 701
\nvalue = [65, 42, 128, 0, 128, 41, 75, 0, 68, 140, 120, 16\n147, 0, 10, 3,
0, 0, 0, 26, 0, 0, 107, 0]'),
 Text(2029.0909090909092, 679.5, 'X[2] <= -0.454\ngini = 0.906\nsamples = 651
\nvalue = [63, 42, 128, 0, 128, 30, 74, 0, 68, 101, 117, 16\n118, 0, 10, 3,
```

```
0, 0, 0, 26, 0, 0, 107, 0]'),
 Text(1893.818181818182, 226.5, 'gini = 0.0\nsamples = 67\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 101, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2164.3636363636365, 226.5, 'gini = 0.896\nsamples = 584\nvalue = [63, 4
2, 128, 0, 128, 30, 74, 0, 68, 0, 117, 16\n118, 0, 10, 3, 0, 0, 0, 26, 0, 0,
107, 0]'),
 Text(2570.1818181818185, 679.5, 'X[13] <= 0.591\ngini = 0.654\nsamples = 50
\nvalue = [2, 0, 0, 0, 0, 11, 1, 0, 0, 39, 3, 0, 29, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0]'),
 Text(2434.909090909091, 226.5, 'gini = 0.0\nsamples = 25\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 39, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2705.4545454545455, 226.5, 'gini = 0.539\nsamples = 25\nvalue = [2, 0,
0, 0, 0, 11, 1, 0, 0, 0, 3, 0, 29, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3720.0, 1585.5, 'X[12] <= -0.082\ngini = 0.932\nsamples = 1567\nvalue =
[97, 148, 64, 0, 86, 172, 43, 203, 113, 43, 53\n183, 27, 218, 198, 215, 0, 0,
0, 176, 189, 219\n29, 0]'),
 Text(3381.818181818182, 1132.5, 'X[11] <= 0.321\ngini = 0.789\nsamples = 498
\nvalue = [0, 0, 0, 0, 86, 0, 31, 203, 0, 43, 0, 183, 0\n0, 0, 215, 0, 0, 0,
0, 0, 0, 29, 0]'),
 Text(3111.2727272727275, 679.5, 'X[4] <= -1.097\ngini = 0.739\nsamples = 298
\nvalue = [0, 0, 0, 0, 69, 0, 30, 1, 0, 33, 0, 173, 0\n0, 0, 150, 0, 0, 0, 0,
0, 0, 22, 0]'),
 Text(2976.0, 226.5, 'gini = 0.379\nsamples = 96\nvalue = [0, 0, 0, 0, 4, 0,
0, 1, 0, 0, 0, 31, 0, 0\n0, 114, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3246.545454545455, 226.5, 'gini = 0.738\nsamples = 202\nvalue = [0, 0,
0, 0, 65, 0, 30, 0, 0, 33, 0, 142, 0\n0, 0, 36, 0, 0, 0, 0, 0, 0, 22, 0]'),
 Text(3652.3636363636365, 679.5, 'X[4] <= -1.092\ngini = 0.532\nsamples = 200
\nvalue = [0, 0, 0, 0, 17, 0, 1, 202, 0, 10, 0, 10, 0\n0, 0, 65, 0, 0, 0, 0,
0, 0, 7, 0]'),
 Text(3517.0909090909095, 226.5, 'gini = 0.422\nsamples = 180\nvalue = [0, 0,
0, 0, 8, 0, 0, 201, 0, 1, 0, 8, 0, 0\n0, 47, 0, 0, 0, 0, 0, 0, 7, 0]'),
 Text(3787.636363636364, 226.5, 'gini = 0.692\nsamples = 20\nvalue = [0, 0,
0, 0, 9, 0, 1, 1, 0, 9, 0, 2, 0, 0\n0, 18, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(4058.1818181818185, 1132.5, 'X[13] <= -0.0\ngini = 0.901\nsamples = 106
9\nvalue = [97, 148, 64, 0, 0, 172, 12, 0, 113, 0, 53, 0\n27, 218, 198, 0, 0,
0, 0, 176, 189, 219, 0, 0]'),
 Text(3922.909090909091, 679.5, 'gini = 0.0\nsamples = 110\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n191, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(4193.454545454546, 679.5, 'X[6] <= -0.105\ngini = 0.891\nsamples = 959
\nvalue = [97, 148, 64, 0, 0, 172, 12, 0, 113, 0, 53, 0\n27, 218, 7, 0, 0, 0,
0, 176, 189, 219, 0, 0]'),
 Text(4058.1818181818185, 226.5, 'gini = 0.832\nsamples = 491\nvalue = [0, 7
8, 2, 0, 0, 59, 0, 0, 12, 0, 22, 0, 0\n152, 4, 0, 0, 0, 0, 137, 158, 171, 0,
0]'),
 Text(4328.727272727273, 226.5, 'gini = 0.894\nsamples = 468\nvalue = [97, 7
0, 62, 0, 0, 113, 12, 0, 101, 0, 31, 0\n27, 66, 3, 0, 0, 0, 0, 39, 31, 48, 0,
0]')]
```



from this data set i observed that the linear rwegression has the highest accuracy of 1.1672524421982615