mk 3/08/2023

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 [4]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_yea
a
```

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	2
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	2
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	2
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	2
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	2
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	2
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	2
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	2
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	2

209880 rows × 14 columns

In [5]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      50462 non-null   float64
 2   CO       87018 non-null   float64
 3   EBE      50463 non-null   float64
 4   NMHC     25935 non-null   float64
 5   NO       209108 non-null  float64
 6   NO_2     209108 non-null  float64
 7   O_3      121858 non-null  float64
 8   PM10     104339 non-null  float64
 9   PM25     51980 non-null   float64
 10  SO_2     86970 non-null   float64
 11  TCH      25935 non-null   float64
 12  TOL      50317 non-null   float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [6]:
```
b=a.fillna(value=104)
b
```

Out[6]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	T
0	2013-11-01 01:00:00	104.0	0.6	104.0	104.0	135.0	74.0	104.0	104.0	104.0	7.0	104.0	10
1	2013-11-01 01:00:00	1.5	0.5	1.3	104.0	71.0	83.0	2.0	23.0	16.0	12.0	104.0	
2	2013-11-01 01:00:00	3.9	104.0	2.8	104.0	49.0	70.0	104.0	104.0	104.0	104.0	104.0	
3	2013-11-01 01:00:00	104.0	0.5	104.0	104.0	82.0	87.0	3.0	104.0	104.0	104.0	104.0	10
4	2013-11-01 01:00:00	104.0	104.0	104.0	104.0	242.0	111.0	2.0	104.0	104.0	12.0	104.0	10
...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	104.0	0.4	104.0	104.0	8.0	39.0	52.0	104.0	104.0	104.0	104.0	10
209876	2013-03-01 00:00:00	104.0	0.4	104.0	104.0	1.0	11.0	104.0	6.0	104.0	2.0	104.0	10
209877	2013-03-01 00:00:00	104.0	104.0	104.0	104.0	2.0	4.0	75.0	104.0	104.0	104.0	104.0	10
209878	2013-03-01 00:00:00	104.0	104.0	104.0	104.0	2.0	11.0	52.0	104.0	104.0	104.0	104.0	10
209879	2013-03-01 00:00:00	104.0	104.0	104.0	104.0	1.0	10.0	75.0	3.0	104.0	104.0	104.0	10

209880 rows × 14 columns

In [7]:
```
b.columns
```

Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [8]:

```
c=b.head(10000)
c
```

Out[8]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2013-11-01 01:00:00	104.0	0.6	104.0	104.0	135.0	74.0	104.0	104.0	104.0	7.0	104.0	104.(
1	2013-11-01 01:00:00	1.5	0.5	1.3	104.0	71.0	83.0	2.0	23.0	16.0	12.0	104.0	8.:
2	2013-11-01 01:00:00	3.9	104.0	2.8	104.0	49.0	70.0	104.0	104.0	104.0	104.0	104.0	9.(
3	2013-11-01 01:00:00	104.0	0.5	104.0	104.0	82.0	87.0	3.0	104.0	104.0	104.0	104.0	104.(
4	2013-11-01 01:00:00	104.0	104.0	104.0	104.0	242.0	111.0	2.0	104.0	104.0	12.0	104.0	104.(
...	...	...	...	...	...	...	...	...	...	...	...	...	..
9995	2013-11-18 09:00:00	104.0	0.7	104.0	104.0	93.0	57.0	4.0	104.0	104.0	104.0	104.0	104.(
9996	2013-11-18 09:00:00	104.0	104.0	104.0	104.0	138.0	69.0	104.0	23.0	104.0	6.0	104.0	104.(
9997	2013-11-18 09:00:00	104.0	104.0	104.0	104.0	168.0	64.0	104.0	22.0	15.0	104.0	104.0	104.(
9998	2013-11-18 09:00:00	104.0	104.0	104.0	104.0	110.0	89.0	104.0	22.0	16.0	104.0	104.0	104.(
9999	2013-11-18 09:00:00	104.0	104.0	104.0	104.0	53.0	42.0	2.0	104.0	104.0	104.0	104.0	104.(

10000 rows × 14 columns

In [9]:

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

Out[9]:

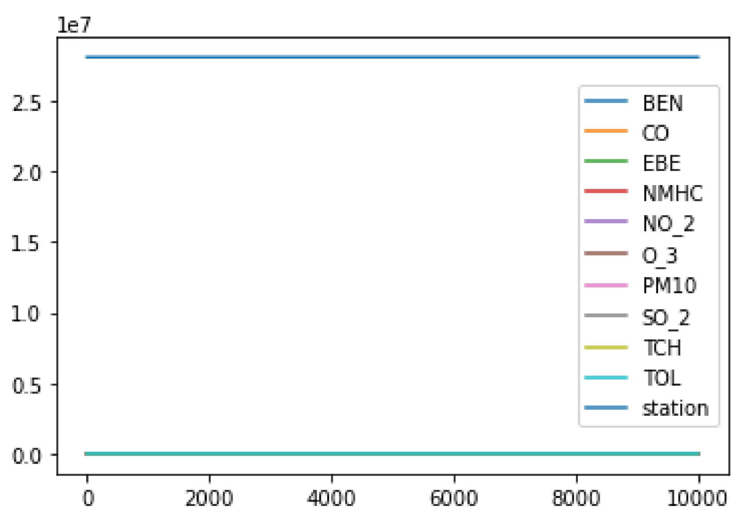
	BEN	CO	EBE	NMHC	NO_2	O_3	PM10	SO_2	TCH	TOL	station
0	104.0	0.6	104.0	104.0	74.0	104.0	104.0	7.0	104.0	104.0	28079004
1	1.5	0.5	1.3	104.0	83.0	2.0	23.0	12.0	104.0	8.3	28079008
2	3.9	104.0	2.8	104.0	70.0	104.0	104.0	104.0	104.0	9.0	28079011
3	104.0	0.5	104.0	104.0	87.0	3.0	104.0	104.0	104.0	104.0	28079016
4	104.0	104.0	104.0	104.0	111.0	2.0	104.0	12.0	104.0	104.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...
9995	104.0	0.7	104.0	104.0	57.0	4.0	104.0	104.0	104.0	104.0	28079039
9996	104.0	104.0	104.0	104.0	69.0	104.0	23.0	6.0	104.0	104.0	28079040
9997	104.0	104.0	104.0	104.0	64.0	104.0	22.0	104.0	104.0	104.0	28079047
9998	104.0	104.0	104.0	104.0	89.0	104.0	22.0	104.0	104.0	104.0	28079048
9999	104.0	104.0	104.0	104.0	42.0	2.0	104.0	104.0	104.0	104.0	28079049

10000 rows × 11 columns

In [10]:

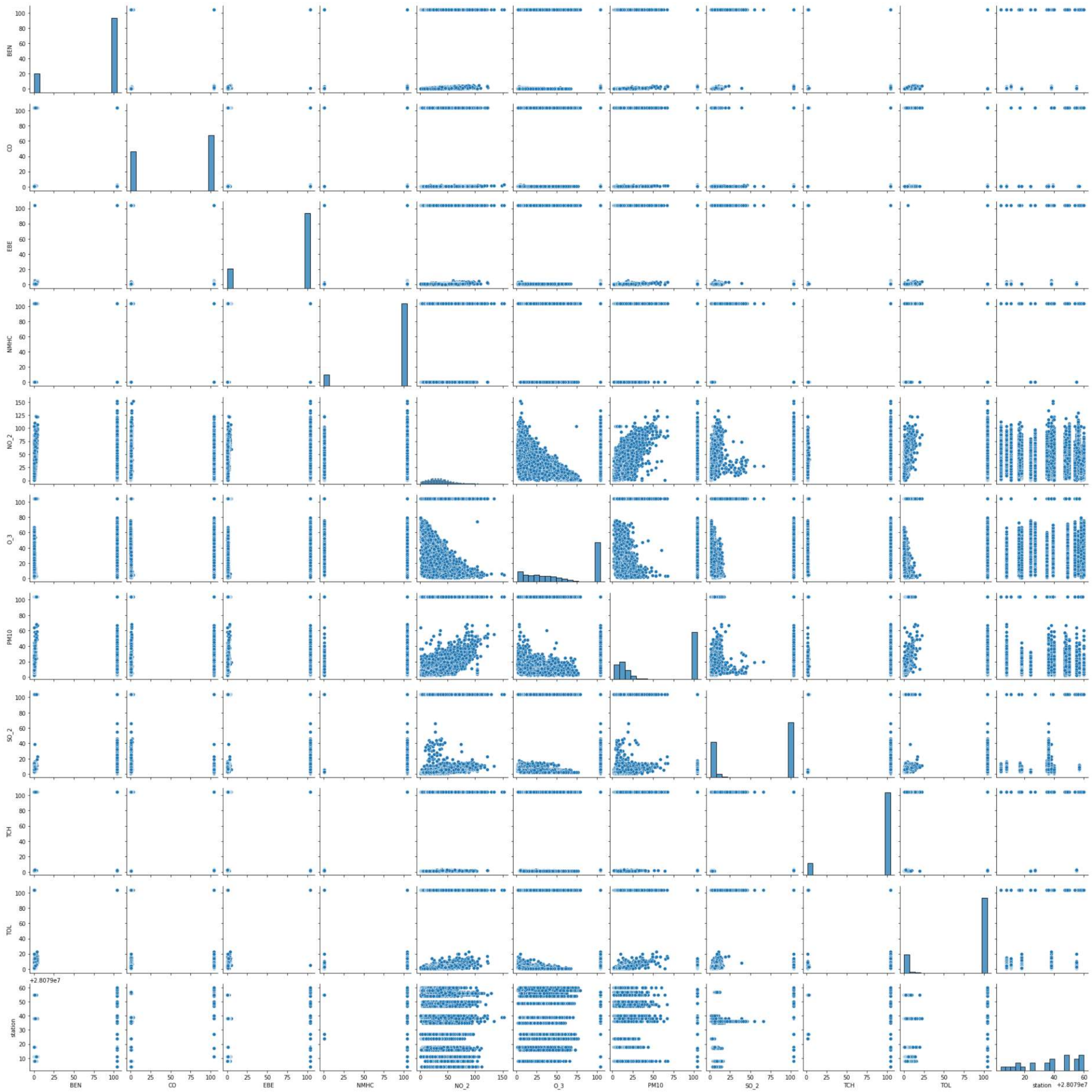
```
d.plot.line()
```

Out[10]: <AxesSubplot:>



In [11]:
```
sns.pairplot(d)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x1f70ad5bd90>



In [12]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
y=d['TCH']
```

In [13]:
```
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 [14]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

```

Out[14]: LinearRegression()

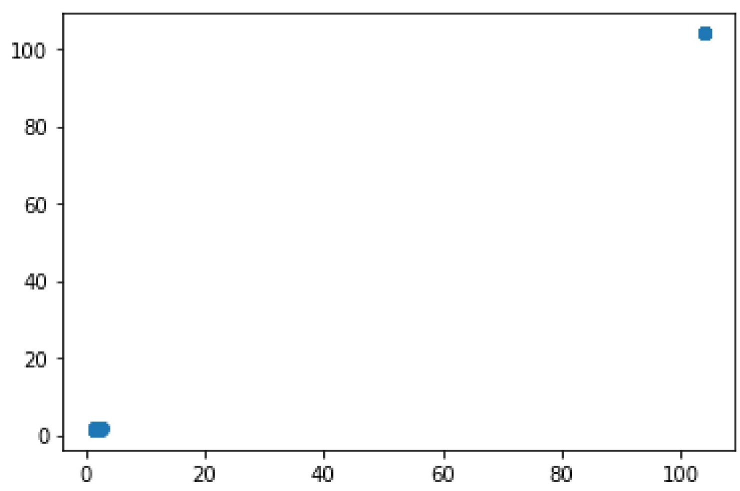
In [47]:
```
lr.fit(x_train,y_train)
```
Out[47]: LinearRegression()

In [15]:
```
print(lr.intercept_)
```
1.2479844840821386

In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```
Out[16]:

	Co-efficient
BEN	-0.000969
CO	0.000221
EBE	0.000703
NMHC	0.987915
NO_2	0.000508

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```
Out[17]: <matplotlib.collections.PathCollection at 0x1f714444850>



In [18]:
```
print(lr.score(x_test,y_test))
```
0.9999953135331272

In [48]:
```
print(lr.score(x_train,y_train))
```
0.9999954606932426

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]: Ridge(alpha=10)

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]: 0.9999953138694069

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]: Lasso(alpha=10)

In [23]:
```
la.score(x_test,y_test)
```

Out[23]: 0.9999189899159493

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

Out[25]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOI
0	2013-11-01 01:00:00	104.0	0.6	104.0	104.0	135.0	74.0	104.0	104.0	104.0	7.0	104.0	104.(
1	2013-11-01 01:00:00	1.5	0.5	1.3	104.0	71.0	83.0	2.0	23.0	16.0	12.0	104.0	8.:
2	2013-11-01 01:00:00	3.9	104.0	2.8	104.0	49.0	70.0	104.0	104.0	104.0	104.0	104.0	9.(
3	2013-11-01 01:00:00	104.0	0.5	104.0	104.0	82.0	87.0	3.0	104.0	104.0	104.0	104.0	104.(
4	2013-11-01 01:00:00	104.0	104.0	104.0	104.0	242.0	111.0	2.0	104.0	104.0	12.0	104.0	104.(
...	...	...	...	...	...	...	...	...	...	...	...	...	..
6995	2013-11-13 04:00:00	104.0	0.2	104.0	104.0	1.0	8.0	40.0	104.0	104.0	104.0	104.0	104.(
6996	2013-11-13 04:00:00	104.0	104.0	104.0	104.0	1.0	5.0	104.0	3.0	104.0	1.0	104.0	104.(
6997	2013-11-13 04:00:00	104.0	104.0	104.0	104.0	1.0	6.0	104.0	3.0	2.0	104.0	104.0	104.(
6998	2013-11-13 04:00:00	104.0	104.0	104.0	104.0	1.0	9.0	104.0	5.0	1.0	104.0	104.0	104.(
6999	2013-11-13 04:00:00	104.0	104.0	104.0	104.0	1.0	9.0	43.0	104.0	104.0	104.0	104.0	104.(

7000 rows × 14 columns

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

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

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

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]]
```

In [32]:
```
prediction=logr.predict(i)
print(prediction)
```
```
[28079050]
```

In [33]:
```
logr.classes_
```
Out[33]:
```
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 [34]:
```
logr.predict_proba(i)[0][0]
```
Out[34]: 0.0

In [35]:
```
logr.predict_proba(i)[0][1]
```
Out[35]: 0.0

In [36]:
```
logr.score(h_test,g_test)
```
Out[36]: 0.9576190476190476

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[37]: ElasticNet()

In [38]:
```
print(en.coef_)
```
```
[-0.          0.         -0.          0.98701057  0.        ]
```

In [39]:
```
print(en.intercept_)
```
```
1.3400191911879489
```

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

```
0.9999944213935371
```

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=
grid_search.fit(h_train,g_train)
```

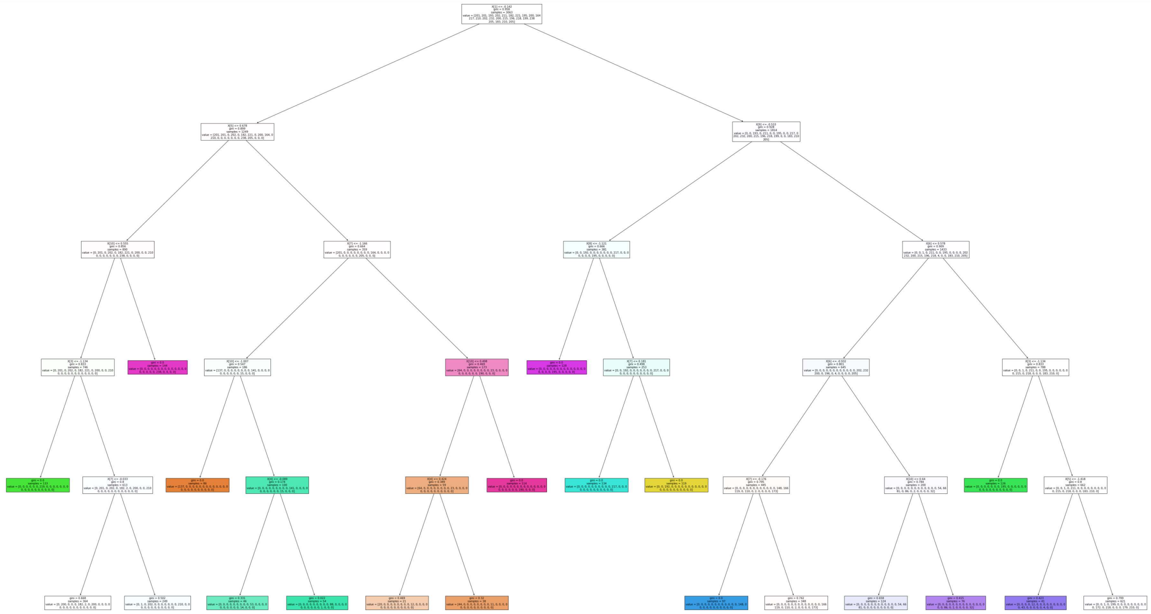
Out[43]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [44]:
```
grid_search.best_score_
```

Out[44]:
```
0.9985714285714286
```

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

In [46]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```



Conclusion

Linear Regression=0.9999954606932426

Ridge Regression=0.9991898899159493

Lasso Regression=0.9991898899159493

ElasticNet Regression=0.9999944213935371

Logistic Regression=0.9576190476190476

Random Forest=0.9985714285714286

In [ ]:
```
Linear Regression is suitable in this dataset
```