In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as py
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
df =pd.read_csv(r"D:\datasets\madrid_2006.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN
...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01

230568 rows × 17 columns

In [3]:
```
df.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 [4]:
```
df1 =df.fillna(value=0)
df1
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
0	2006-02-01 01:00:00	0.00	1.84	0.00	0.00	0.00	155.100006	490.100006	0.00
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04
2	2006-02-01 01:00:00	0.00	1.25	0.00	0.00	0.00	66.800003	192.000000	0.00
3	2006-02-01 01:00:00	0.00	1.68	0.00	0.00	0.00	103.000000	407.799988	0.00
4	2006-02-01 01:00:00	0.00	1.31	0.00	0.00	0.00	105.400002	269.200012	0.00
...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	0.00	0.20	112.500000	218.000000	0.00
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61
230565	2006-05-01 00:00:00	0.96	0.00	0.69	0.00	0.19	135.100006	179.199997	0.00
230566	2006-05-01 00:00:00	0.50	0.00	0.67	0.00	0.10	82.599998	105.599998	0.00
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01

230568 rows × 17 columns

In [5]:
```
df1.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      230568 non-null  float64
 2   CO       230568 non-null  float64
 3   EBE      230568 non-null  float64
 4   MXY      230568 non-null  float64
 5   NMHC     230568 non-null  float64
 6   NO_2     230568 non-null  float64
 7   NOx      230568 non-null  float64
 8   OXY      230568 non-null  float64
 9   O_3      230568 non-null  float64
 10  PM10     230568 non-null  float64
 11  PM25     230568 non-null  float64
 12  PXY      230568 non-null  float64
 13  SO_2     230568 non-null  float64
 14  TCH      230568 non-null  float64
 15  TOL      230568 non-null  float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

In [6]:
```
df1.columns
```

Out[6]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [7]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
df2
```

Out[7]:

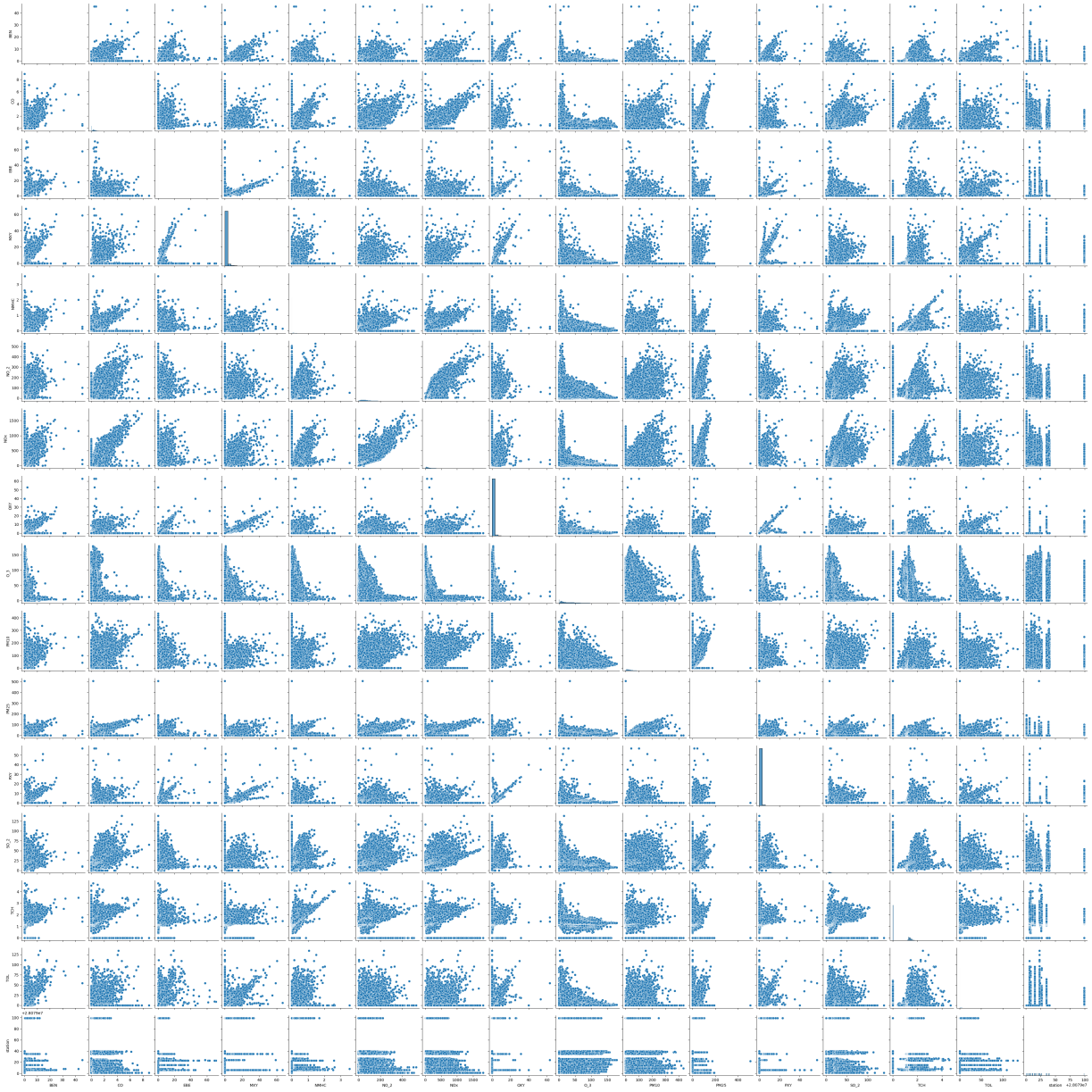
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	0.00	1.84	0.00	0.00	0.00	155.100006	490.100006	0.00	4.880000
1	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000
2	0.00	1.25	0.00	0.00	0.00	66.800003	192.000000	0.00	4.430000
3	0.00	1.68	0.00	0.00	0.00	103.000000	407.799988	0.00	4.830000
4	0.00	1.31	0.00	0.00	0.00	105.400002	269.200012	0.00	6.990000
...	...	...	...	...	...	...	...	...	...
230563	5.88	0.83	6.23	0.00	0.20	112.500000	218.000000	0.00	24.389999
230564	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000
230565	0.96	0.00	0.69	0.00	0.19	135.100006	179.199997	0.00	11.460000
230566	0.50	0.00	0.67	0.00	0.10	82.599998	105.599998	0.00	0.000000
230567	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000

230568 rows × 16 columns

In [8]:
```
sns.pairplot(df2)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\seabor
n\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

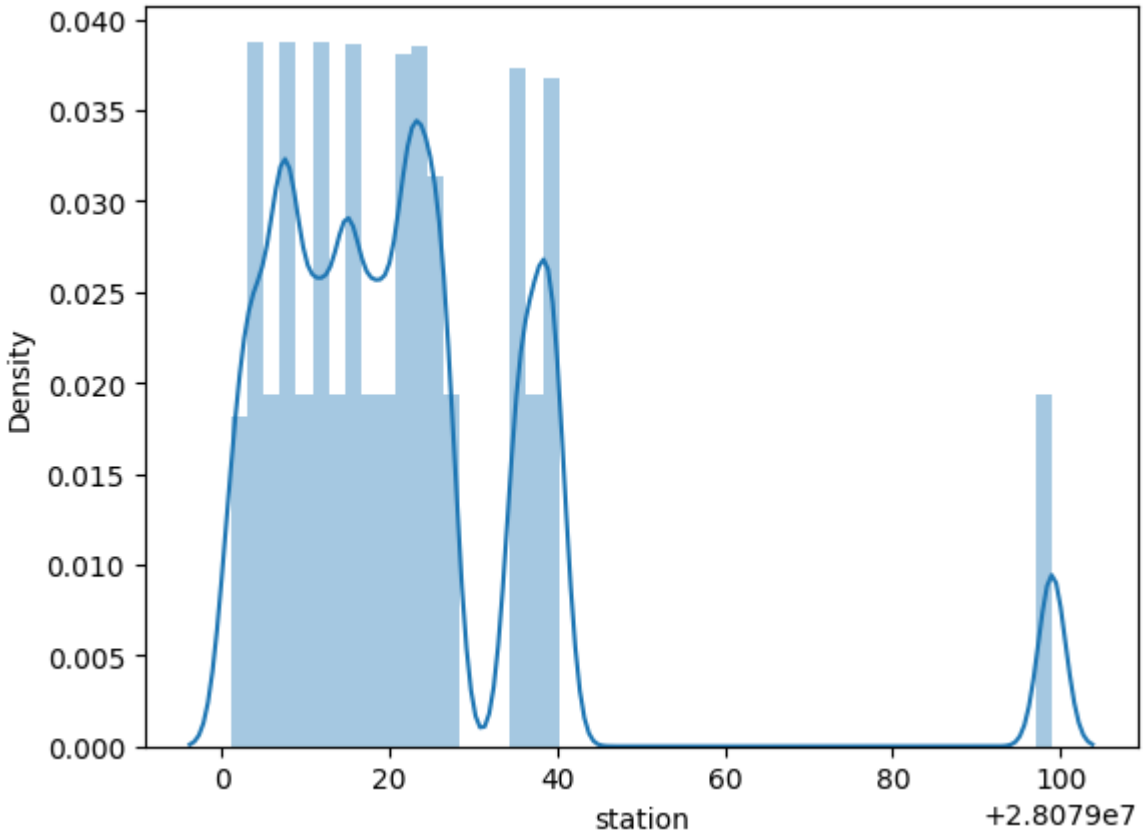
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x29dc50d35d0>
```



In [9]:
```
sns.distplot(df2['station'])
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17572\1070072814.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2['station'])
```

Out[9]: <Axes: xlabel='station', ylabel='Density'>



In [10]:
```
x=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df2['station']
```

In [11]:
```
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)
```

linear

In [12]:
```
from sklearn.linear_model import LinearRegression
```

In [13]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
▼ LinearRegression

LinearRegression()
```

In [14]:
```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[14]:

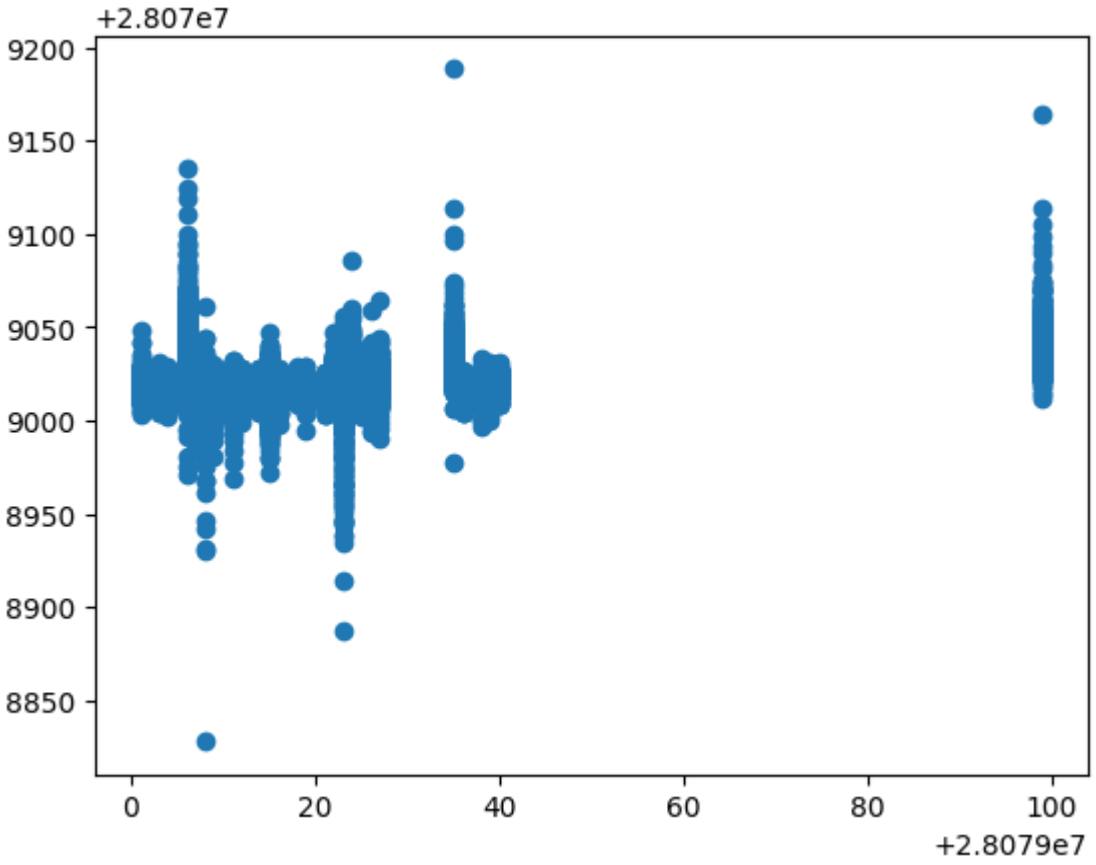
	Co-efficient
BEN	-4.997351
CO	1.253111
EBE	-1.356873
MXY	0.669524
NMHC	-22.946795
NO_2	-0.036151
NOx	-0.018197
OXY	5.290444
O_3	-0.012774
PM10	0.041814
PM25	0.155528
PXY	1.097395
SO_2	-0.112969
TCH	5.255443
TOL	0.454390

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

```
28079022.85623591
```

In [16]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x29d931f8310>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
0.1252452709108267
```

In [18]:
```
print(lr.score(x_train,y_train))
```

```
0.12736871834403318
```

Ridge

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

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

Out[20]:
```
▼     Ridge

Ridge(alpha=10)
```

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

Out[21]:
```
0.12525382469213653
```

Lasso

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]:
```
0.038482712151716725
```

elasticnet

In [24]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

In [25]:
```
print(en.coef_)
```

```
[-0.20447227 -0.         -0.09253026  1.48205817 -0.         -0.02747718
 -0.02742433  0.78734004 -0.01183667  0.03811669  0.19700218  0.49465604
 -0.07639807  0.1498616   0.        ]
```

In [26]:
```
print(en.intercept_)
```

```
28079024.489736784
```

In [27]:
```
print(en.predict(x_test))
```

```
[28079025.27457917 28079015.95792437 28079020.63722851 ...
 28079029.18656831 28079023.20719109 28079008.76715033]
```

In [28]:
```
print(en.score(x_test,y_test))
```

```
0.0943705170682394
```

logistic

In [29]:
```
feature_matrix =df2.iloc[:,0:15]
target_vector=df2.iloc[:,-1]
```

In [30]:
```
feature_matrix=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df2['station']
```

In [31]:
```
feature_matrix.shape
```

Out[31]: (230568, 15)

In [32]:
```
target_vector.shape
```

Out[32]: (230568,)

In [33]:
```
from sklearn.preprocessing import StandardScaler
```

In [34]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [35]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklear
n\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

Out[35]:
```
▼ LogisticRegression

LogisticRegression()
```

In [36]:
```
observation=[[1,2,3,4,5,6,7,8,9,11,12,13,14,15,16]]
```

In [37]:
```
prediction =logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [38]:
```
logr.classes_
```

Out[38]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 28079035, 28079036, 28079038,
       28079039, 28079040, 28079099], dtype=int64)
```

In [39]:
```
logr.score(fs,target_vector)
```

Out[39]: 0.5840489573574824

In [40]:
```
logr.predict_proba(observation)[0][0]
```

Out[40]:
```
3.250139032944616e-119
```

In [41]:
```
logr.predict_proba(observation)[0][1]
```

Out[41]:
```
6.615527487769511e-173
```

Random forest

In [42]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [43]:
```
x=df2.drop('station',axis=1)
y=df2['station']
```

In [44]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.70)
```

In [45]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[45]:
```
▼ RandomForestClassifier

RandomForestClassifier()
```

In [46]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf' :[6,7,8,9,10],
            'n_estimators':[11,12,13,14,15]}
```

In [47]:
```
from sklearn.model_selection import GridSearchCV
```

In [48]:
```
grid_search =GridSearchCV(estimator =rfc,param_grid=parameters,cv=2,scoring=
grid_search.fit(x_train,y_train)
```

Out[48]:
```
▶ GridSearchCV
▶ estimator: RandomForestClassifier
    ▶ RandomForestClassifier
```

In [49]:
```
grid_search.best_score_
```

Out[49]:
```
0.5189243891860633
```

In [50]:
```
rfc_best=grid_search.best_estimator_
```

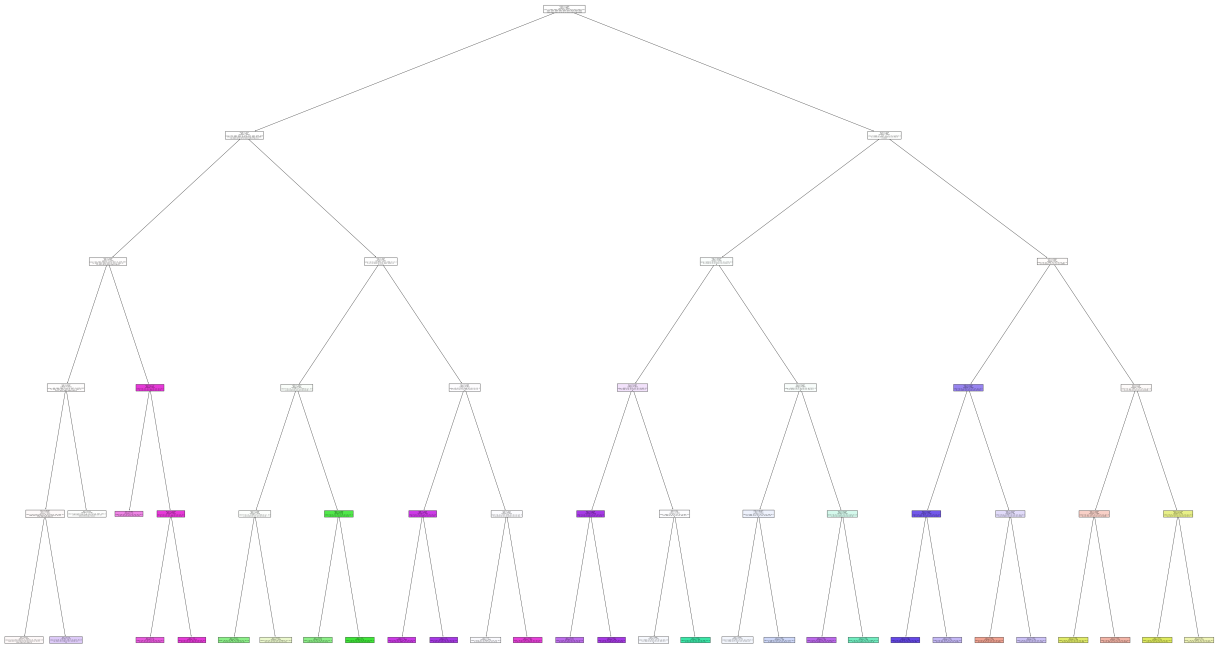
In [51]:
```
py.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[51]: [Text(0.46120689655172414, 0.9166666666666666, 'x[10] <= 0.595\ngini = 0.963\nsamples = 43864\nvalue = [2386, 2496, 2589, 2509, 2633, 2615, 2647, 2690, 2578\n2557, 2733, 2690, 2563, 2628, 2555, 2676, 2664, 2638\n1584, 2562, 2692, 2479, 2672, 2642, 2288, 2796, 2608]'),
 Text(0.19827586206896552, 0.75, 'x[13] <= 0.22\ngini = 0.95\nsamples = 32227\nvalue = [60, 2496, 2589, 34, 2633, 2615, 2647, 2690, 2578\n2557, 14, 2690, 2563, 2628, 2555, 62, 2664, 60, 1584\n38, 2692, 2479, 2672, 2083, 2288, 2796, 11]'),
 Text(0.08620689655172414, 0.5833333333333334, 'x[0] <= 0.205\ngini = 0.932\nsamples = 22965\nvalue = [60, 2496, 2589, 14, 43, 44, 2647, 31, 2578, 2557\n10, 2690, 2563, 2628, 2555, 62, 36, 42, 1584, 12\n10, 1041, 2672, 2083, 2288, 2796, 0]'),
 Text(0.05172413793103448, 0.4166666666666667, 'x[1] <= 0.415\ngini = 0.929\nsamples = 22330\nvalue = [60, 2496, 2589, 11, 43, 42, 2647, 31, 2578, 2557\n10, 2690, 2563, 2628, 2555, 62, 34, 42, 1584, 12\n10, 47, 2672, 2083, 2288, 2796, 0]'),
 Text(0.034482758620689655, 0.25, 'x[5] <= 78.695\ngini = 0.914\nsamples = 9604\nvalue = [23, 1223, 1316, 11, 32, 32, 415, 15, 776, 1255, 10\n1396, 641, 1408, 529, 35, 29, 42, 1381, 12, 10, 34\n1717, 505, 162, 2061, 0]'),
 Text(0.017241379310344827, 0.08333333333333333, 'gini = 0.912\nsamples = 9032\nvalue = [23, 1194, 1284, 11, 32, 28, 247, 14, 766, 1227, 10\n1360, 627, 1317, 500, 35, 29, 42, 1021, 12, 10, 33\n1676, 469, 161, 2060, 0]'),
 Text(0.05172413793103448, 0.08333333333333333, 'gini = 0.776\nsamples = 572\nvalue = [0, 29, 32, 0, 0, 4, 168, 1, 10, 28, 0, 36, 14\n91, 29, 0, 0, 0, 360, 0, 0, 1, 41, 36, 1, 1\n0]'),
 Text(0.06896551724137931, 0.25, 'gini = 0.919\nsamples = 12726\nvalue = [37, 1273, 1273, 0, 11, 10, 2232, 16, 1802, 1302, 0\n1294, 1922, 1220, 2026, 27, 5, 0, 203, 0, 0, 13\n955, 1578, 2126, 735, 0]'),
 Text(0.1206896551724138, 0.4166666666666667, 'x[12] <= 7.17\ngini = 0.014\nsamples = 635\nvalue = [0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 2, 0, 0, 0, 0, 994, 0, 0, 0, 0, 0]'),
 Text(0.10344827586206896, 0.25, 'gini = 0.484\nsamples = 10\nvalue = [0, 0, 0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 1, 0, 0, 0, 0, 11, 0, 0, 0, 0, 0]'),
 Text(0.13793103448275862, 0.25, 'x[11] <= 0.405\ngini = 0.004\nsamples = 625\nvalue = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 1, 0, 0, 0, 0, 983, 0, 0, 0, 0, 0]'),
 Text(0.1206896551724138, 0.08333333333333333, 'gini = 0.272\nsamples = 10\nvalue = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 1, 0, 0, 0, 0, 11, 0, 0, 0, 0, 0]'),
 Text(0.15517241379310345, 0.08333333333333333, 'gini = 0.0\nsamples = 615\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 972, 0, 0, 0, 0, 0]'),
 Text(0.3103448275862069, 0.5833333333333334, 'x[2] <= 0.05\ngini = 0.83\nsamples = 9262\nvalue = [0, 0, 0, 20, 2590, 2571, 0, 2659, 0, 0, 4, 0\n0, 0, 0, 0, 2628, 18, 0, 26, 2682, 1438, 0, 0\n0, 0, 11]'),
 Text(0.2413793103448276, 0.4166666666666667, 'x[5] <= 132.9\ngini = 0.509\nsamples = 3311\nvalue = [0, 0, 0, 2, 2590, 25, 0, 2659, 0, 0, 1, 0, 0\n0, 0, 0, 16, 0, 0, 1, 1, 4, 0, 0, 0, 0, 0]'),
 Text(0.20689655172413793, 0.25, 'x[4] <= 0.185\ngini = 0.509\nsamples = 3278\nvalue = [0, 0, 0, 2, 2586, 25, 0, 2616, 0, 0, 1, 0, 0\n0, 0, 0, 16, 0, 0, 1, 1, 4, 0, 0, 0, 0, 0]'),
 Text(0.1896551724137931, 0.08333333333333333, 'gini = 0.422\nsamples = 843\nvalue = [0, 0, 0, 0, 369, 6, 0, 970, 0, 0, 1, 0, 0, 0\n0, 0, 15, 0, 0, 1, 1, 2, 0, 0, 0, 0, 0]'),
 Text(0.2758620689655172, 0.25, 'gini = 0.495\nsamples = 24

35\nvalue = [0, 0, 0, 2, 2217, 19, 0, 1646, 0, 0, 0, 0, 0\n0, 0, 0, 1, 0,
0, 0, 0, 2, 0, 0, 0, 0, 0]'),
 Text(0.27586206896551724, 0.25, 'x[8] <= 8.105\ngini = 0.156\nsamples = 33
\nvalue = [0, 0, 0, 0, 4, 0, 0, 43, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0]'),
 Text(0.25862068965517243, 0.08333333333333333, 'gini = 0.408\nsamples = 10
\nvalue = [0, 0, 0, 0, 4, 0, 0, 10, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0]'),
 Text(0.29310344827586204, 0.08333333333333333, 'gini = 0.0\nsamples = 23\n
value = [0, 0, 0, 0, 0, 0, 0, 33, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0]'),
 Text(0.3793103448275862, 0.4166666666666667, 'x[1] <= 0.065\ngini = 0.742
\nsamples = 5951\nvalue = [0, 0, 0, 18, 0, 2546, 0, 0, 0, 0, 3, 0, 0, 0\n0,
0, 2612, 18, 0, 25, 2681, 1434, 0, 0, 0, 0\n11]'),
 Text(0.3448275862068966, 0.25, 'x[8] <= 3.56\ngini = 0.018\nsamples = 1702
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 25, 268
1, 0, 0, 0, 0, 0, 0]'),
 Text(0.3275862068965517, 0.08333333333333333, 'gini = 0.0\nsamples = 1689
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 268
1, 0, 0, 0, 0, 0, 0]'),
 Text(0.3620689655172414, 0.08333333333333333, 'gini = 0.0\nsamples = 13\nv
alue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 25, 0, 0,
0, 0, 0, 0, 0]'),
 Text(0.41379310344827586, 0.25, 'x[7] <= 0.21\ngini = 0.652\nsamples = 424
9\nvalue = [0, 0, 0, 18, 0, 2546, 0, 0, 0, 0, 3, 0, 0, 0\n0, 0, 2612, 18,
0, 0, 0, 1434, 0, 0, 0, 0, 11]'),
 Text(0.39655172413793105, 0.08333333333333333, 'gini = 0.5\nsamples = 3284
\nvalue = [0, 0, 0, 0, 0, 2546, 0, 0, 0, 0, 3, 0, 0, 0\n0, 0, 2612, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.43103448275862066, 0.08333333333333333, 'gini = 0.062\nsamples = 96
5\nvalue = [0, 0, 0, 18, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 18, 0, 0,
0, 1434, 0, 0, 0, 0, 11]'),
 Text(0.7241379310344828, 0.75, 'x[11] <= 0.09\ngini = 0.864\nsamples = 116
37\nvalue = [2326, 0, 0, 2475, 0, 0, 0, 0, 0, 0, 2719, 0, 0\n0, 0, 2614, 0,
2578, 0, 2524, 0, 0, 0, 559, 0\n0, 2597]'),
 Text(0.5862068965517241, 0.5833333333333334, 'x[9] <= 29.235\ngini = 0.779
\nsamples = 6898\nvalue = [2326, 0, 0, 122, 0, 0, 0, 0, 0, 0, 2719, 0, 0\n
0, 0, 2614, 0, 53, 0, 2524, 0, 0, 0, 559, 0, 0\n8]'),
 Text(0.5172413793103449, 0.4166666666666667, 'x[1] <= 0.005\ngini = 0.772
\nsamples = 3058\nvalue = [925, 0, 0, 45, 0, 0, 0, 0, 0, 0, 962, 0, 0\n0,
0, 996, 0, 26, 0, 1562, 0, 0, 0, 305, 0, 0\n5]'),
 Text(0.4827586206896552, 0.25, 'x[6] <= 16.64\ngini = 0.009\nsamples = 100
8\nvalue = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 6, 0, 0, 0, 1562,
0, 0, 0, 0, 0, 0, 0]'),
 Text(0.46551724137931033, 0.08333333333333333, 'gini = 0.355\nsamples = 17
\nvalue = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 4, 0, 0, 0, 18, 0,
0, 0, 0, 0, 0, 0]'),
 Text(0.5, 0.08333333333333333, 'gini = 0.003\nsamples = 991\nvalue = [0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 2, 0, 0, 0, 1544, 0, 0, 0, 0, 0,
0, 0]'),
 Text(0.5517241379310345, 0.25, 'x[0] <= 0.1\ngini = 0.731\nsamples = 2050
\nvalue = [924, 0, 0, 45, 0, 0, 0, 0, 0, 0, 962, 0, 0\n0, 0, 990, 0, 26, 0,
0, 0, 0, 0, 305, 0, 0\n5]'),
 Text(0.5344827586206896, 0.08333333333333333, 'gini = 0.686\nsamples = 158
2\nvalue = [924, 0, 0, 42, 0, 0, 0, 0, 0, 0, 222, 0, 0\n0, 0, 990, 0, 26,
0, 0, 0, 0, 0, 305, 0, 0\n0]'),

```
 Text(0.5689655172413793, 0.08333333333333333, 'gini = 0.021\nsamples = 468
\nvalue = [0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 740, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 5]'),
 Text(0.6551724137931034, 0.4166666666666667, 'x[2] <= 0.055\ngini = 0.767
\nsamples = 3840\nvalue = [1401, 0, 0, 77, 0, 0, 0, 0, 0, 0, 1757, 0, 0\n0,
0, 1618, 0, 27, 0, 962, 0, 0, 0, 254, 0, 0\n3]'),
 Text(0.6206896551724138, 0.25, 'x[9] <= 91.515\ngini = 0.68\nsamples = 249
0\nvalue = [1401, 0, 0, 76, 0, 0, 0, 0, 0, 0, 544, 0, 0\n0, 0, 1618, 0, 27,
0, 11, 0, 0, 0, 254, 0, 0\n1]'),
 Text(0.603448275862069, 0.08333333333333333, 'gini = 0.678\nsamples = 2141
\nvalue = [1243, 0, 0, 56, 0, 0, 0, 0, 0, 0, 474, 0, 0\n0, 0, 1359, 0, 27,
0, 11, 0, 0, 0, 199, 0, 0\n0]'),
 Text(0.6379310344827587, 0.08333333333333333, 'gini = 0.683\nsamples = 349
\nvalue = [158, 0, 0, 20, 0, 0, 0, 0, 0, 0, 70, 0, 0, 0\n0, 259, 0, 0, 0,
0, 0, 0, 0, 55, 0, 0, 1]'),
 Text(0.6896551724137931, 0.25, 'x[2] <= 0.555\ngini = 0.494\nsamples = 135
0\nvalue = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1213, 0, 0, 0\n0, 0, 0, 0, 0, 95
1, 0, 0, 0, 0, 0, 0, 2]'),
 Text(0.6724137931034483, 0.08333333333333333, 'gini = 0.311\nsamples = 522
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 158, 0, 0, 0\n0, 0, 0, 0, 0, 661,
0, 0, 0, 0, 0, 0, 0]'),
 Text(0.7068965517241379, 0.08333333333333333, 'gini = 0.341\nsamples = 828
\nvalue = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1055, 0, 0, 0\n0, 0, 0, 0, 0, 290,
0, 0, 0, 0, 0, 0, 2]'),
 Text(0.8620689655172413, 0.5833333333333334, 'x[7] <= 1.005\ngini = 0.666
\nsamples = 4739\nvalue = [0, 0, 0, 2353, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0,
0, 0, 2525, 0, 0, 0, 0, 0, 0, 0, 0, 2589]'),
 Text(0.7931034482758621, 0.4166666666666667, 'x[3] <= 1.005\ngini = 0.473
\nsamples = 1993\nvalue = [0, 0, 0, 393, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0,
0, 0, 2167, 0, 0, 0, 0, 0, 0, 0, 0, 574]'),
 Text(0.7586206896551724, 0.25, 'x[5] <= 34.54\ngini = 0.251\nsamples = 120
0\nvalue = [0, 0, 0, 94, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 1607, 0, 0,
0, 0, 0, 0, 0, 0, 169]'),
 Text(0.7413793103448276, 0.08333333333333333, 'gini = 0.136\nsamples = 989
\nvalue = [0, 0, 0, 24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 1444, 0, 0,
0, 0, 0, 0, 0, 0, 89]'),
 Text(0.7758620689655172, 0.08333333333333333, 'gini = 0.613\nsamples = 211
\nvalue = [0, 0, 0, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 163, 0, 0,
0, 0, 0, 0, 0, 0, 80]'),
 Text(0.8275862068965517, 0.25, 'x[0] <= 0.435\ngini = 0.645\nsamples = 793
\nvalue = [0, 0, 0, 299, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 560, 0, 0,
0, 0, 0, 0, 0, 0, 405]'),
 Text(0.8103448275862069, 0.08333333333333333, 'gini = 0.482\nsamples = 226
\nvalue = [0, 0, 0, 26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 90, 0, 0, 0,
0, 0, 0, 0, 0, 233]'),
 Text(0.8448275862068966, 0.08333333333333333, 'gini = 0.612\nsamples = 567
\nvalue = [0, 0, 0, 273, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 470, 0, 0,
0, 0, 0, 0, 0, 0, 172]'),
 Text(0.9310344827586207, 0.4166666666666667, 'x[3] <= 6.92\ngini = 0.572\n
samples = 2746\nvalue = [0, 0, 0, 1960, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 358, 0, 0, 0, 0, 0, 0, 0, 0, 2015]'),
 Text(0.896551724137931, 0.25, 'x[4] <= 0.095\ngini = 0.572\nsamples = 1999
\nvalue = [0, 0, 0, 1127, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 331, 0, 0,
0, 0, 0, 0, 0, 0, 1698]'),
 Text(0.8793103448275862, 0.08333333333333333, 'gini = 0.338\nsamples = 298
\nvalue = [0, 0, 0, 372, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 25, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 69]'),
 Text(0.9137931034482759, 0.08333333333333333, 'gini = 0.542\nsamples = 170
1\nvalue = [0, 0, 0, 755, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 306, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1629]'),
 Text(0.9655172413793104, 0.25, 'x[13] <= 1.515\ngini = 0.426\nsamples = 74
7\nvalue = [0, 0, 0, 833, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 27, 0, 0,
0, 0, 0, 0, 0, 0, 0, 317]'),
 Text(0.9482758620689655, 0.08333333333333333, 'gini = 0.27\nsamples = 334
\nvalue = [0, 0, 0, 437, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 8, 0, 0, 0,
0, 0, 0, 0, 0, 0, 74]'),
 Text(0.9827586206896551, 0.08333333333333333, 'gini = 0.501\nsamples = 413
\nvalue = [0, 0, 0, 396, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 19, 0, 0,
0, 0, 0, 0, 0, 0, 0, 243]')]
```



concusion

The bestfit model is logistic Regression with score of 0.5840489573574824

In [ ]: