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_2004.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN
...	...	...	...	...	...	...	...	...	...
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60

245496 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      65158 non-null   float64
 2   CO       226043 non-null  float64
 3   EBE      56781 non-null   float64
 4   MXY      39867 non-null   float64
 5   NMHC     107630 non-null  float64
 6   NO_2     243280 non-null  float64
 7   NOx      243283 non-null  float64
 8   OXY      39882 non-null   float64
 9   O_3      233811 non-null  float64
 10  PM10     234655 non-null  float64
 11  PM25     58145 non-null   float64
 12  PXY      39891 non-null   float64
 13  SO_2     243402 non-null  float64
 14  TCH      107650 non-null  float64
 15  TOL      64914 non-null   float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ MB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
0	2004-08-01 01:00:00	0.00	0.66	0.00	0.00	0.00	89.550003	118.900002	0.00
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28
2	2004-08-01 01:00:00	0.00	1.02	0.00	0.00	0.00	93.389999	138.600006	0.00
3	2004-08-01 01:00:00	0.00	0.53	0.00	0.00	0.00	87.290001	105.000000	0.00
4	2004-08-01 01:00:00	0.00	0.17	0.00	0.00	0.00	34.910000	35.349998	0.00
...	...	...	...	...	...	...	...	...	...
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	0.00	97.139999	146.899994	2.34
245493	2004-06-01 00:00:00	0.00	0.00	0.00	0.00	0.13	102.699997	132.600006	0.00
245494	2004-06-01 00:00:00	0.00	0.00	0.00	0.00	0.09	82.599998	102.599998	0.00
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60

245496 rows × 17 columns

In [5]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      245496 non-null  float64
 2   CO       245496 non-null  float64
 3   EBE      245496 non-null  float64
 4   MXY      245496 non-null  float64
 5   NMHC     245496 non-null  float64
 6   NO_2     245496 non-null  float64
 7   NOx      245496 non-null  float64
 8   OXY      245496 non-null  float64
 9   O_3      245496 non-null  float64
 10  PM10     245496 non-null  float64
 11  PM25     245496 non-null  float64
 12  PXY      245496 non-null  float64
 13  SO_2     245496 non-null  float64
 14  TCH      245496 non-null  float64
 15  TOL      245496 non-null  float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ 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 [8]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
df2
```

Out[8]:

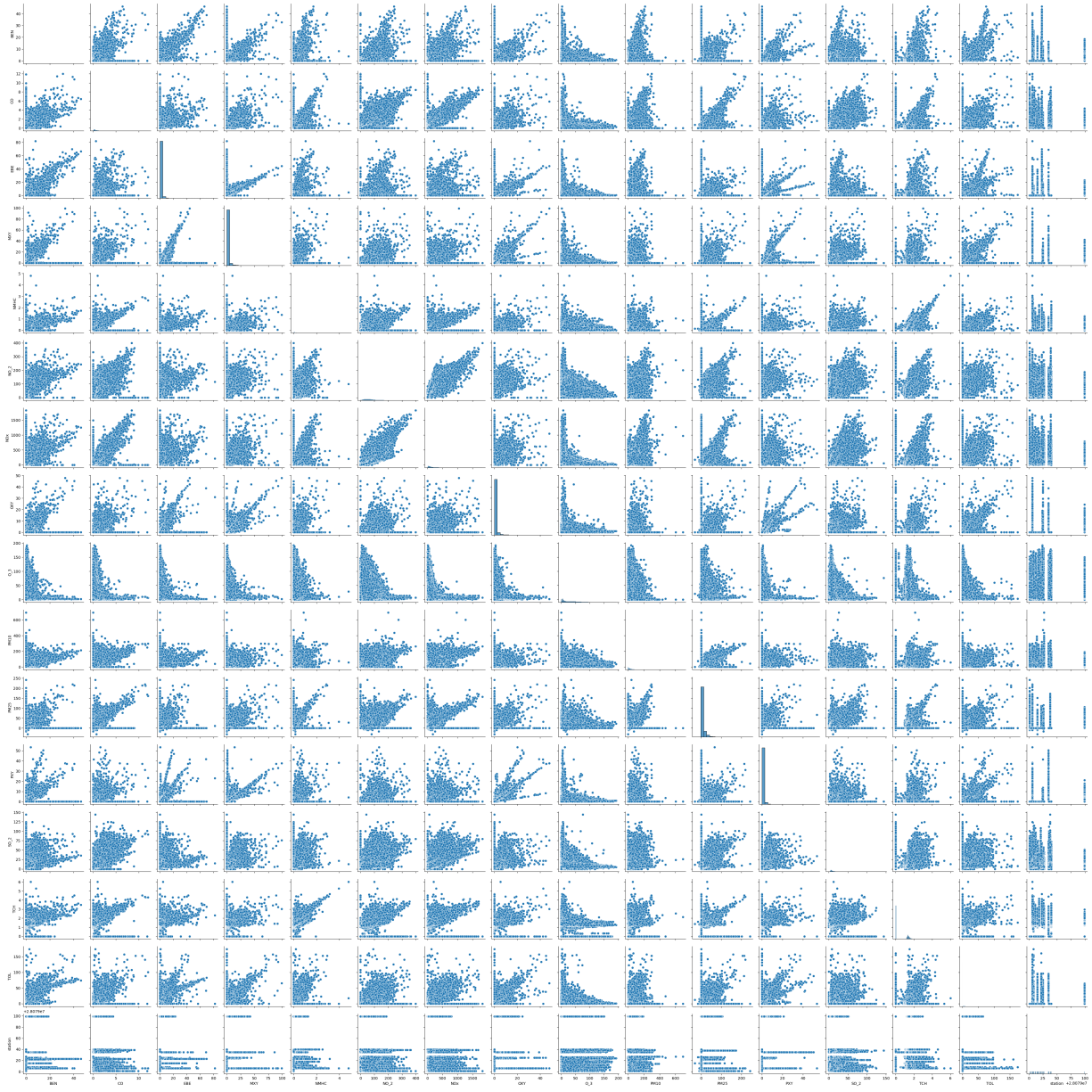
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	0.00	0.66	0.00	0.00	0.00	89.550003	118.900002	0.00	40.020000
1	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999
2	0.00	1.02	0.00	0.00	0.00	93.389999	138.600006	0.00	20.860001
3	0.00	0.53	0.00	0.00	0.00	87.290001	105.000000	0.00	36.730000
4	0.00	0.17	0.00	0.00	0.00	34.910000	35.349998	0.00	86.269997
...	...	...	...	...	...	...	...	...	...
245491	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999
245492	2.49	0.75	2.44	4.57	0.00	97.139999	146.899994	2.34	7.740000
245493	0.00	0.00	0.00	0.00	0.13	102.699997	132.600006	0.00	17.809999
245494	0.00	0.00	0.00	0.00	0.09	82.599998	102.599998	0.00	0.000000
245495	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000

245496 rows × 16 columns

In [9]:
```
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)
```

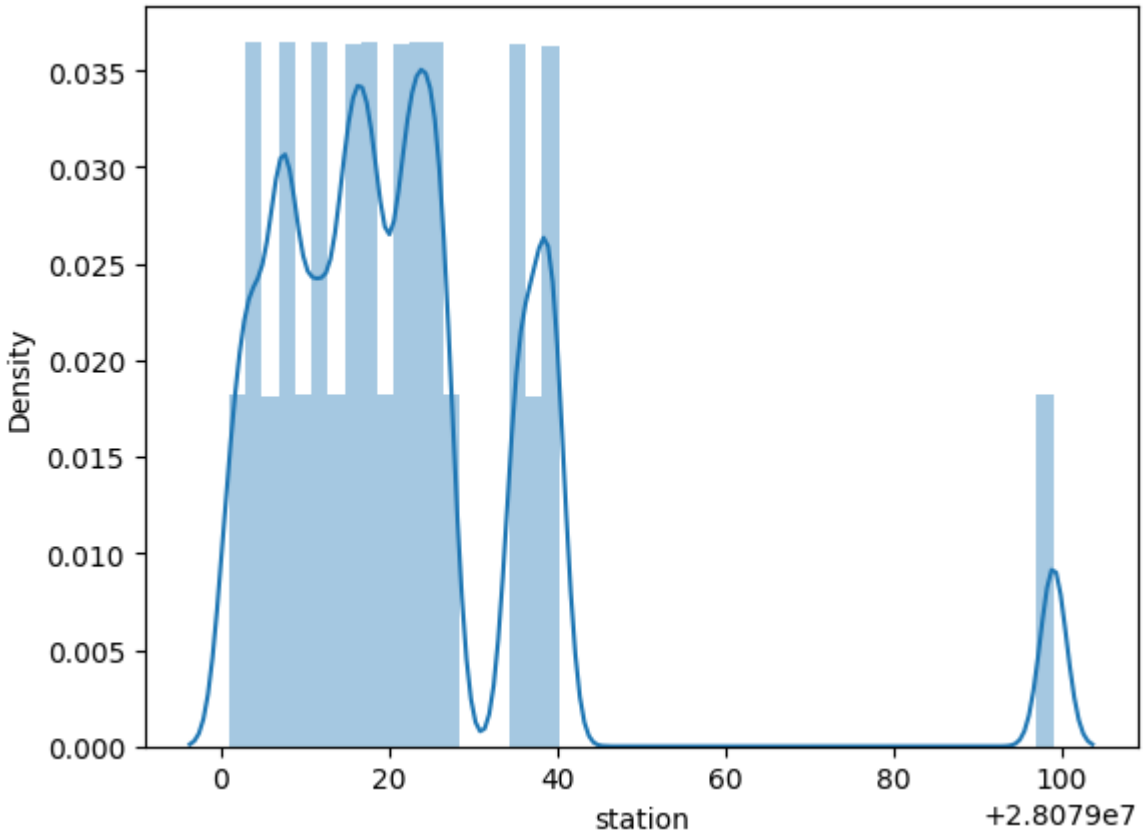
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1df27da4e90>
```



In [11]:
```
sns.distplot(df2['station'])
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_5048\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[11]: <Axes: xlabel='station', ylabel='Density'>



In [13]:
```
x=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df2['station']
```

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)
```

linear

In [21]:
```
from sklearn.linear_model import LinearRegression
```

In [22]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[22]:
```
▼ LinearRegression

LinearRegression()
```

In [23]:
```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[23]:

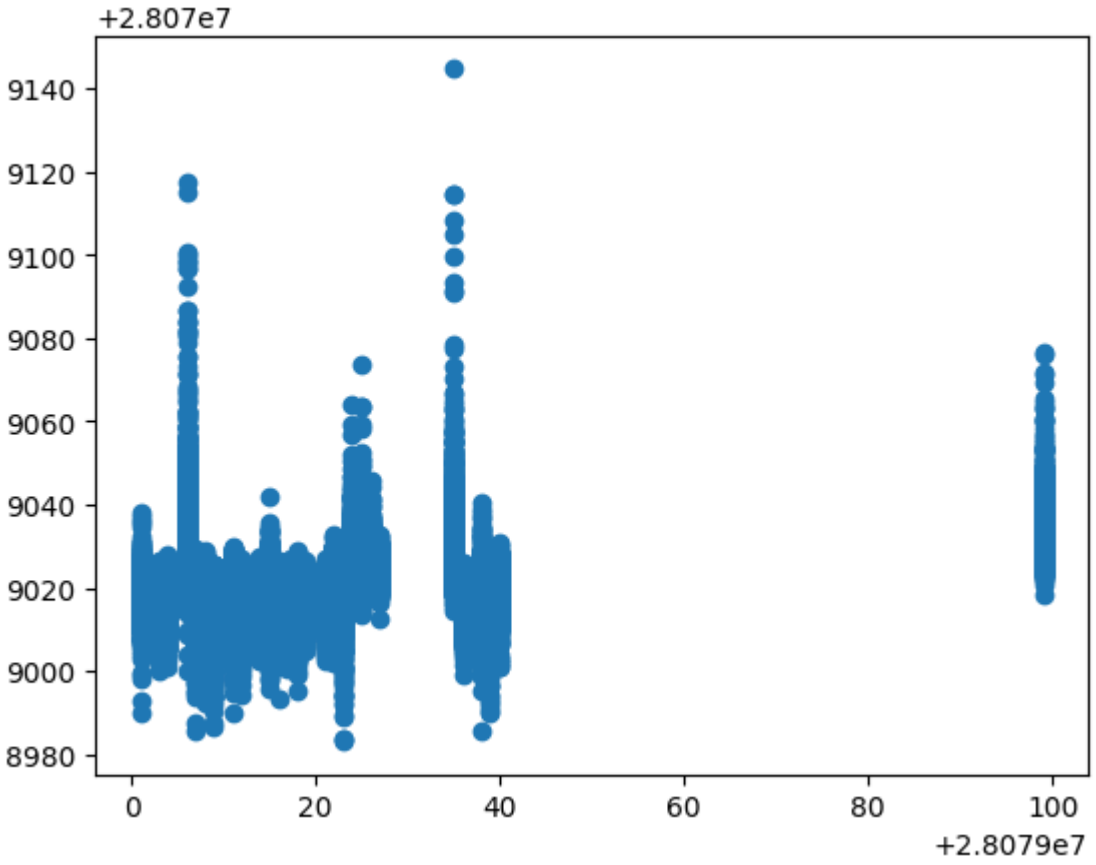
	Co-efficient
BEN	-0.262028
CO	-3.738927
EBE	0.067800
MXY	0.261178
NMHC	-6.564362
NO_2	-0.079584
NOx	0.011911
OXY	1.531053
O_3	-0.015951
PM10	0.026883
PM25	0.156959
PXY	1.699519
SO_2	-0.145245
TCH	3.137689
TOL	-0.079277

In [24]:
```
print(lr.intercept_)
```

```
28079025.382064547
```

In [25]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[25]:
```
<matplotlib.collections.PathCollection at 0x1df5ab1a1d0>
```



In [26]:
```
print(lr.score(x_test,y_test))
```

```
0.11998283318388314
```

In [27]:
```
print(lr.score(x_train,y_train))
```

```
0.11999440524556182
```

Ridge

In [28]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [29]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[29]:
```
▼     Ridge

Ridge(alpha=10)
```

In [30]:
```
rr.score(x_test,y_test)
```

Out[30]:
```
0.11998428591160937
```

Lasso

In [31]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[31]:
```
Lasso(alpha=10)
```

In [35]:
```
la.score(x_test,y_test)
```

Out[35]:
```
0.05162318484723172
```

elasticnet

In [36]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[36]:
```
ElasticNet()
```

In [37]:
```
print(en.coeff_)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[37], line 1
----> 1 print(en.coeff_)

AttributeError: 'ElasticNet' object has no attribute 'coeff_'
```

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

```
28079026.14620236
```

In [39]:
```
print(en.predict(x_test))
```

```
[28079017.37188012 28079017.57941925 28079022.99555518 ...
 28079019.24038233 28079023.85637269 28079034.1072114 ]
```

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

```
0.106728198863277
```

logistic

In [41]:
```
feature_matrix =df2.iloc[:,0:15]
target_vector=df2.iloc[:,-1]
```

In [42]:
```
feature_matrix=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df2['station']
```

In [43]:
```
feature_matrix.shape
```

Out[43]:
```
(245496, 15)
```

In [44]:
```
target_vector.shape
```

Out[44]:
```
(245496,)
```

In [45]:
```
from sklearn.preprocessing import StandardScaler
```

In [50]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [51]:
```
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[51]:
```
▼ LogisticRegression

LogisticRegression()
```

In [62]:
```
observation=[[1,2,3,4,5,6,7,8,9,11,12,13,14,15,16]]
```

In [63]:
```
prediction =logr.predict(observation)
print(prediction)
```

```
[28079024]
```

In [64]:
```
logr.classes_
```

Out[64]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079026, 28079027, 28079035, 28079036,
       28079038, 28079039, 28079040, 28079099], dtype=int64)
```

In [65]:
```
logr.score(fs,target_vector)
```

Out[65]:
```
0.5467787662528106
```

In [67]:
```
logr.predict_proba(observation)[0][0]
```

Out[67]:
```
5.605431597800388e-138
```

In [69]:
```
logr.predict_proba(observation)[0][1]
```

Out[69]:
```
1.5413918664514151e-179
```

Random forest

In [70]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [74]:
```
x=df2.drop('station',axis=1)
y=df2['station']
```

In [76]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.70)
```

In [77]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[77]:
```
▼ RandomForestClassifier

RandomForestClassifier()
```

In [78]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf' :[6,7,8,9,10],
            'n_estimators':[11,12,13,14,15]}
```

In [79]:
```
from sklearn.model_selection import GridSearchCV
```

In [80]:
```
grid_search =GridSearchCV(estimator =rfc,param_grid=parameters,cv=2,scoring=
grid_search.fit(x_train,y_train)
```

Out[80]:
```
▶ GridSearchCV
▶ estimator: RandomForestClassifier
    ▶ RandomForestClassifier
```

In [81]:
```
grid_search.best_score_
```

Out[81]:
```
0.5159542689550294
```

In [82]:
```
rfc_best=grid_search.best_estimator_
```

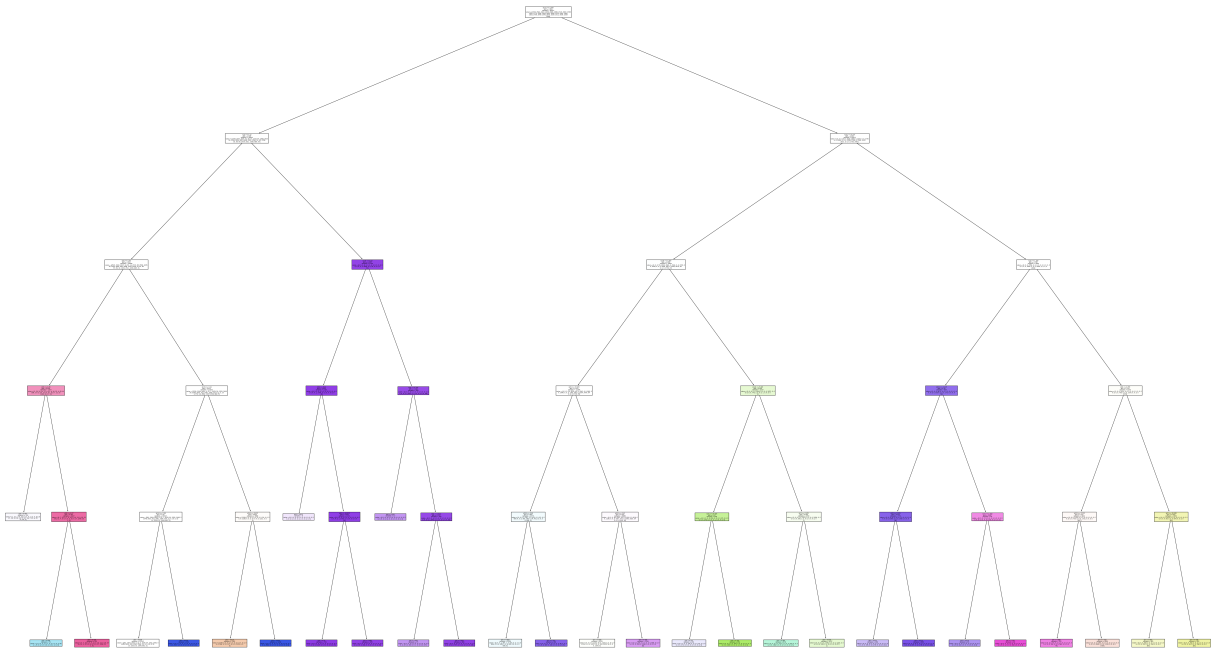
In [86]:
```
py.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[86]: [Text(0.45047169811320753, 0.9166666666666666, 'x[13] <= 0.095\ngini = 0.964\nsamples = 46514\nvalue = [2539, 2543, 2627, 2659, 2689, 2696, 2719, 2607, 2568\n2635, 2770, 2594, 2654, 2649, 2608, 2519, 2556, 2589\n2833, 2708, 2649, 2545, 2679, 2639, 2511, 2636, 2639\n2588]'),
 Text(0.2028301886792453, 0.75, 'x[3] <= 0.12\ngini = 0.939\nsamples = 26042\nvalue = [2539, 2543, 2627, 42, 59, 13, 2719, 65, 2568, 2635\n30, 2594, 2654, 600, 2608, 2519, 2556, 31, 65, 2708\n55, 69, 66, 2639, 2511, 2636, 889, 97]'),
 Text(0.10377358490566038, 0.5833333333333334, 'x[9] <= 0.44\ngini = 0.935\nsamples = 24292\nvalue = [2539, 2543, 2627, 34, 59, 13, 2719, 65, 2568, 2635\n30, 2594, 2654, 600, 2608, 2519, 2556, 31, 62, 60\n55, 69, 32, 2639, 2511, 2636, 889, 5]'),
 Text(0.03773584905660377, 0.4166666666666667, 'x[5] <= 2.26\ngini = 0.634\nsamples = 1357\nvalue = [9, 39, 24, 34, 37, 6, 41, 6, 16, 35, 4, 45, 44\n188, 27, 15, 39, 19, 57, 41, 18, 17, 22, 25\n1284, 42, 27, 0]'),
 Text(0.018867924528301886, 0.25, 'gini = 0.947\nsamples = 330\nvalue = [1, 32, 2, 34, 30, 6, 22, 6, 2, 22, 4, 26, 25\n27, 16, 11, 30, 19, 57, 41, 18, 7, 22, 8, 35\n17, 26, 0]'),
 Text(0.05660377358490566, 0.25, 'x[8] <= 4.67\ngini = 0.391\nsamples = 1027\nvalue = [8, 7, 22, 0, 7, 0, 19, 0, 14, 13, 0, 19, 19\n161, 11, 4, 9, 0, 0, 0, 0, 10, 0, 17, 1249, 25\n1, 0]'),
 Text(0.03773584905660377, 0.08333333333333333, 'gini = 0.658\nsamples = 72\nvalue = [1, 0, 4, 0, 2, 0, 7, 0, 0, 0, 0, 0, 0, 66\n1, 1, 6, 0, 0, 0, 0, 10, 0, 0, 21, 2, 0, 0]'),
 Text(0.07547169811320754, 0.08333333333333333, 'gini = 0.319\nsamples = 955\nvalue = [7, 7, 18, 0, 5, 0, 12, 0, 14, 13, 0, 19, 19\n95, 10, 3, 3, 0, 0, 0, 0, 0, 17, 1228, 23\n1, 0]'),
 Text(0.16981132075471697, 0.4166666666666667, 'x[10] <= 0.21\ngini = 0.932\nsamples = 22935\nvalue = [2530, 2504, 2603, 0, 22, 7, 2678, 59, 2552, 2600\n26, 2549, 2610, 412, 2581, 2504, 2517, 12, 5, 19\n37, 52, 10, 2614, 1227, 2594, 862, 5]'),
 Text(0.1320754716981132, 0.25, 'x[14] <= 0.69\ngini = 0.92\nsamples = 19347\nvalue = [281, 2504, 2603, 0, 22, 7, 2678, 59, 2552, 2600, 0\n2549, 2610, 412, 2581, 2504, 359, 12, 0, 19, 0, 52\n10, 2614, 15, 2594, 862, 0]'),
 Text(0.11320754716981132, 0.08333333333333333, 'gini = 0.918\nsamples = 19145\nvalue = [281, 2504, 2603, 0, 22, 5, 2678, 59, 2552, 2600, 0\n2549, 2610, 412, 2581, 2504, 22, 9, 0, 19, 0, 52\n10, 2614, 15, 2594, 862, 0]'),
 Text(0.1509433962264151, 0.08333333333333333, 'gini = 0.029\nsamples = 202\nvalue = [0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 337, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.20754716981132076, 0.25, 'x[14] <= 0.435\ngini = 0.655\nsamples = 3588\nvalue = [2249, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26, 0, 0, 0\n0, 0, 2158, 0, 5, 0, 37, 0, 0, 0, 1212, 0, 0\n5]'),
 Text(0.18867924528301888, 0.08333333333333333, 'gini = 0.514\nsamples = 2342\nvalue = [2249, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17, 0, 0, 0\n0, 0, 148, 0, 5, 0, 37, 0, 0, 0, 1212, 0, 0\n5]'),
 Text(0.22641509433962265, 0.08333333333333333, 'gini = 0.009\nsamples = 1246\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0\n0, 0, 2010, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3018867924528302, 0.5833333333333334, 'x[14] <= 6.53\ngini = 0.095\nsamples = 1750\nvalue = [0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 3, 2648, 0, 0, 34, 0, 0, 0, 0\n92]'),
 Text(0.2641509433962264, 0.4166666666666667, 'x[0] <= 0.22\ngini = 0.059\nsamples = 1182\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 1, 1796, 0, 0, 21, 0, 0, 0, 0\n34]'),
 Text(0.2641509433962264, 0.25, 'gini = 0.615\nsamples = 8\nvalue = [0, 0,

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 6, 0, 0, 2, 0, 0, 0, 0,
5]'),
 Text(0.2830188679245283, 0.25, 'x[8] <= 44.825\ngini = 0.052\nsamples = 11
74\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1, 1790,
0, 0, 19, 0, 0, 0, 0\n29]'),
 Text(0.2641509433962264, 0.08333333333333333, 'gini = 0.03\nsamples = 739
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 1149,
0, 0, 13, 0, 0, 0, 0, 5]'),
 Text(0.3018867924528302, 0.08333333333333333, 'gini = 0.089\nsamples = 435
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1, 641, 0,
0, 6, 0, 0, 0, 0, 24]'),
 Text(0.33962264150943394, 0.4166666666666667, 'x[0] <= 0.705\ngini = 0.162
\nsamples = 568\nvalue = [0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 0, 2, 852, 0, 0, 13, 0, 0, 0, 0, 58]'),
 Text(0.32075471698113206, 0.25, 'gini = 0.532\nsamples = 13\nvalue = [0,
0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 12, 0, 0, 0, 0, 0, 0,
0, 3]'),
 Text(0.3584905660377358, 0.25, 'x[6] <= 103.5\ngini = 0.152\nsamples = 555
\nvalue = [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2, 840, 0,
0, 13, 0, 0, 0, 0, 55]'),
 Text(0.33962264150943394, 0.08333333333333333, 'gini = 0.454\nsamples = 75
\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2, 83, 0,
0, 1, 0, 0, 0, 0, 37]'),
 Text(0.37735849056603776, 0.08333333333333333, 'gini = 0.083\nsamples = 48
0\nvalue = [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 757,
0, 0, 12, 0, 0, 0, 0, 18]'),
 Text(0.6981132075471698, 0.75, 'x[11] <= 0.105\ngini = 0.922\nsamples = 20
472\nvalue = [0, 0, 0, 2617, 2630, 2683, 0, 2542, 0, 0, 2740\n0, 0, 2049,
0, 0, 0, 2558, 2768, 0, 2594, 2476\n2613, 0, 0, 0, 1750, 2491]'),
 Text(0.5471698113207547, 0.5833333333333334, 'x[14] <= 0.365\ngini = 0.896
\nsamples = 14537\nvalue = [0, 0, 0, 59, 2630, 2683, 0, 2542, 0, 0, 2740, 0
\n0, 2049, 0, 0, 0, 2558, 942, 0, 2594, 2476, 65\n0, 0, 0, 1750, 27]'),
 Text(0.4716981132075472, 0.4166666666666667, 'x[6] <= 37.625\ngini = 0.879
\nsamples = 11264\nvalue = [0, 0, 0, 59, 2630, 40, 0, 2542, 0, 0, 869, 0\n
0, 2049, 0, 0, 0, 1782, 940, 0, 2594, 2476, 64\n0, 0, 0, 1750, 25]'),
 Text(0.4339622641509434, 0.25, 'x[10] <= 0.69\ngini = 0.847\nsamples = 237
9\nvalue = [0, 0, 0, 3, 387, 1, 0, 433, 0, 0, 53, 0, 0\n899, 0, 0, 0, 365,
368, 0, 34, 451, 10, 0, 0\n0, 696, 9]'),
 Text(0.41509433962264153, 0.08333333333333333, 'gini = 0.815\nsamples = 20
98\nvalue = [0, 0, 0, 1, 387, 1, 0, 433, 0, 0, 7, 0, 0\n899, 0, 0, 0, 365,
1, 0, 8, 451, 10, 0, 0, 0\n696, 0]'),
 Text(0.4528301886792453, 0.08333333333333333, 'gini = 0.321\nsamples = 281
\nvalue = [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 46, 0, 0, 0\n0, 0, 0, 0, 367, 0, 2
6, 0, 0, 0, 0, 0, 0, 9]'),
 Text(0.5094339622641509, 0.25, 'x[10] <= 0.42\ngini = 0.872\nsamples = 888
5\nvalue = [0, 0, 0, 56, 2243, 39, 0, 2109, 0, 0, 816, 0\n0, 1150, 0, 0, 0,
1417, 572, 0, 2560, 2025, 54\n0, 0, 0, 1054, 16]'),
 Text(0.49056603773584906, 0.08333333333333333, 'gini = 0.834\nsamples = 65
91\nvalue = [0, 0, 0, 8, 2243, 39, 0, 2109, 0, 0, 8, 0, 0\n1150, 0, 0, 0, 1
417, 19, 0, 320, 2025, 54, 0, 0\n0, 1054, 0]'),
 Text(0.5283018867924528, 0.08333333333333333, 'gini = 0.555\nsamples = 229
4\nvalue = [0, 0, 0, 48, 0, 0, 0, 0, 0, 0, 808, 0, 0, 0\n0, 0, 0, 0, 553,
0, 2240, 0, 0, 0, 0, 0, 0\n16]'),
 Text(0.6226415094339622, 0.4166666666666667, 'x[12] <= 5.345\ngini = 0.605
\nsamples = 3273\nvalue = [0, 0, 0, 0, 0, 2643, 0, 0, 0, 0, 1871, 0, 0\n0,
0, 0, 0, 776, 2, 0, 0, 0, 1, 0, 0, 0, 0\n2]'),
```

```
 Text(0.5849056603773585, 0.25, 'x[5] <= 39.405\ngini = 0.47\nsamples = 558
\nvalue = [0, 0, 0, 0, 0, 616, 0, 0, 0, 0, 35, 0, 0, 0\n0, 0, 0, 276, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0]'),
 Text(0.5660377358490566, 0.08333333333333333, 'gini = 0.571\nsamples = 204
\nvalue = [0, 0, 0, 0, 0, 149, 0, 0, 0, 0, 29, 0, 0, 0\n0, 0, 0, 171, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.6037735849056604, 0.08333333333333333, 'gini = 0.316\nsamples = 354
\nvalue = [0, 0, 0, 0, 0, 467, 0, 0, 0, 0, 6, 0, 0, 0\n0, 0, 0, 105, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0]'),
 Text(0.660377358490566, 0.25, 'x[1] <= 0.455\ngini = 0.595\nsamples = 2715
\nvalue = [0, 0, 0, 0, 0, 2027, 0, 0, 0, 0, 1836, 0, 0\n0, 0, 0, 0, 500, 2,
0, 0, 0, 0, 0, 0, 0, 0\n2]'),
 Text(0.6415094339622641, 0.08333333333333333, 'gini = 0.584\nsamples = 678
\nvalue = [0, 0, 0, 0, 0, 327, 0, 0, 0, 0, 616, 0, 0, 0\n0, 0, 0, 168, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1]'),
 Text(0.6792452830188679, 0.08333333333333333, 'gini = 0.576\nsamples = 203
7\nvalue = [0, 0, 0, 0, 0, 1700, 0, 0, 0, 0, 1220, 0, 0\n0, 0, 0, 0, 332,
2, 0, 0, 0, 0, 0, 0, 0, 0\n1]'),
 Text(0.8490566037735849, 0.5833333333333334, 'x[3] <= 1.185\ngini = 0.746
\nsamples = 5935\nvalue = [0, 0, 0, 2558, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0,
0, 0, 0, 1826, 0, 0, 0, 2548, 0, 0, 0, 0\n2464]'),
 Text(0.7735849056603774, 0.4166666666666667, 'x[11] <= 1.115\ngini = 0.39
\nsamples = 830\nvalue = [0, 0, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 0, 1027, 0, 0, 0, 173, 0, 0, 0, 0\n92]'),
 Text(0.7358490566037735, 0.25, 'x[9] <= 8.415\ngini = 0.332\nsamples = 774
\nvalue = [0, 0, 0, 41, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1006, 0,
0, 0, 108, 0, 0, 0, 0\n89]'),
 Text(0.7169811320754716, 0.08333333333333333, 'gini = 0.632\nsamples = 168
\nvalue = [0, 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 135, 0,
0, 0, 69, 0, 0, 0, 0, 41]'),
 Text(0.7547169811320755, 0.08333333333333333, 'gini = 0.212\nsamples = 606
\nvalue = [0, 0, 0, 26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 871, 0,
0, 0, 39, 0, 0, 0, 0, 48]'),
 Text(0.8113207547169812, 0.25, 'x[7] <= 1.005\ngini = 0.487\nsamples = 56
\nvalue = [0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 21, 0, 0,
0, 65, 0, 0, 0, 0, 3]'),
 Text(0.7924528301886793, 0.08333333333333333, 'gini = 0.516\nsamples = 16
\nvalue = [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 20, 0, 0,
0, 7, 0, 0, 0, 0, 0]'),
 Text(0.8301886792452831, 0.08333333333333333, 'gini = 0.199\nsamples = 40
\nvalue = [0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1, 0, 0,
0, 58, 0, 0, 0, 0, 3]'),
 Text(0.9245283018867925, 0.4166666666666667, 'x[0] <= 2.765\ngini = 0.719
\nsamples = 5105\nvalue = [0, 0, 0, 2510, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0,
0, 0, 0, 799, 0, 0, 0, 2375, 0, 0, 0, 0\n2372]'),
 Text(0.8867924528301887, 0.25, 'x[12] <= 5.335\ngini = 0.721\nsamples = 34
42\nvalue = [0, 0, 0, 1100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 730,
0, 0, 0, 1721, 0, 0, 0, 0\n1867]'),
 Text(0.8679245283018868, 0.08333333333333333, 'gini = 0.486\nsamples = 552
\nvalue = [0, 0, 0, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 124, 0,
0, 0, 589, 0, 0, 0, 0\n78]'),
 Text(0.9056603773584906, 0.08333333333333333, 'gini = 0.716\nsamples = 289
0\nvalue = [0, 0, 0, 1040, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 606,
0, 0, 0, 1132, 0, 0, 0, 0\n1789]'),
 Text(0.9622641509433962, 0.25, 'x[5] <= 99.085\ngini = 0.616\nsamples = 16
63\nvalue = [0, 0, 0, 1410, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 69,
```

```
0, 0, 0, 654, 0, 0, 0, 0\n505]'),
 Text(0.9433962264150944, 0.08333333333333333, 'gini = 0.661\nsamples = 861
\nvalue = [0, 0, 0, 628, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 59, 0,
0, 0, 316, 0, 0, 0, 0\n350]'),
 Text(0.9811320754716981, 0.08333333333333333, 'gini = 0.546\nsamples = 802
\nvalue = [0, 0, 0, 782, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 10, 0,
0, 0, 338, 0, 0, 0, 0\n155]')]
```



concusion

The bestfit model is logistic Regression with score of 0.5467787662528106

In [ ]: