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_2007.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49
...	...	...	...	...	...	...	...	...	...
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37

225120 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225120 entries, 0 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     225120 non-null  object 
 1   BEN      68885 non-null   float64
 2   CO       206748 non-null  float64
 3   EBE      68883 non-null   float64
 4   MXY      26061 non-null   float64
 5   NMHC     86883 non-null   float64
 6   NO_2     223985 non-null  float64
 7   NOx      223972 non-null  float64
 8   OXY      26062 non-null   float64
 9   O_3      211850 non-null  float64
 10  PM10     222588 non-null  float64
 11  PM25     68870 non-null   float64
 12  PXY      26062 non-null   float64
 13  SO_2     224372 non-null  float64
 14  TCH      87026 non-null   float64
 15  TOL      68845 non-null   float64
 16  station  225120 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.2+ MB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
0	2007-12-01 01:00:00	0.00	2.86	0.00	0.00	0.00	282.200012	1054.000000	0.00
1	2007-12-01 01:00:00	0.00	1.82	0.00	0.00	0.00	86.419998	354.600006	0.00
2	2007-12-01 01:00:00	0.00	1.47	0.00	0.00	0.00	94.639999	319.000000	0.00
3	2007-12-01 01:00:00	0.00	1.64	0.00	0.00	0.00	127.900002	476.700012	0.00
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49
...	...	...	...	...	...	...	...	...	...
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00
225116	2007-03-01 00:00:00	0.00	0.16	0.00	0.00	0.00	46.820000	51.480000	0.00
225117	2007-03-01 00:00:00	0.24	0.00	0.20	0.00	0.09	51.259998	66.809998	0.00
225118	2007-03-01 00:00:00	0.11	0.00	1.00	0.00	0.05	24.240000	36.930000	0.00
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37

225120 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225120 entries, 0 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     225120 non-null  object 
 1   BEN      225120 non-null  float64
 2   CO       225120 non-null  float64
 3   EBE      225120 non-null  float64
 4   MXY      225120 non-null  float64
 5   NMHC     225120 non-null  float64
 6   NO_2     225120 non-null  float64
 7   NOx      225120 non-null  float64
 8   OXY      225120 non-null  float64
 9   O_3      225120 non-null  float64
 10  PM10     225120 non-null  float64
 11  PM25     225120 non-null  float64
 12  PXY      225120 non-null  float64
 13  SO_2     225120 non-null  float64
 14  TCH      225120 non-null  float64
 15  TOL      225120 non-null  float64
 16  station  225120 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.2+ 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]:

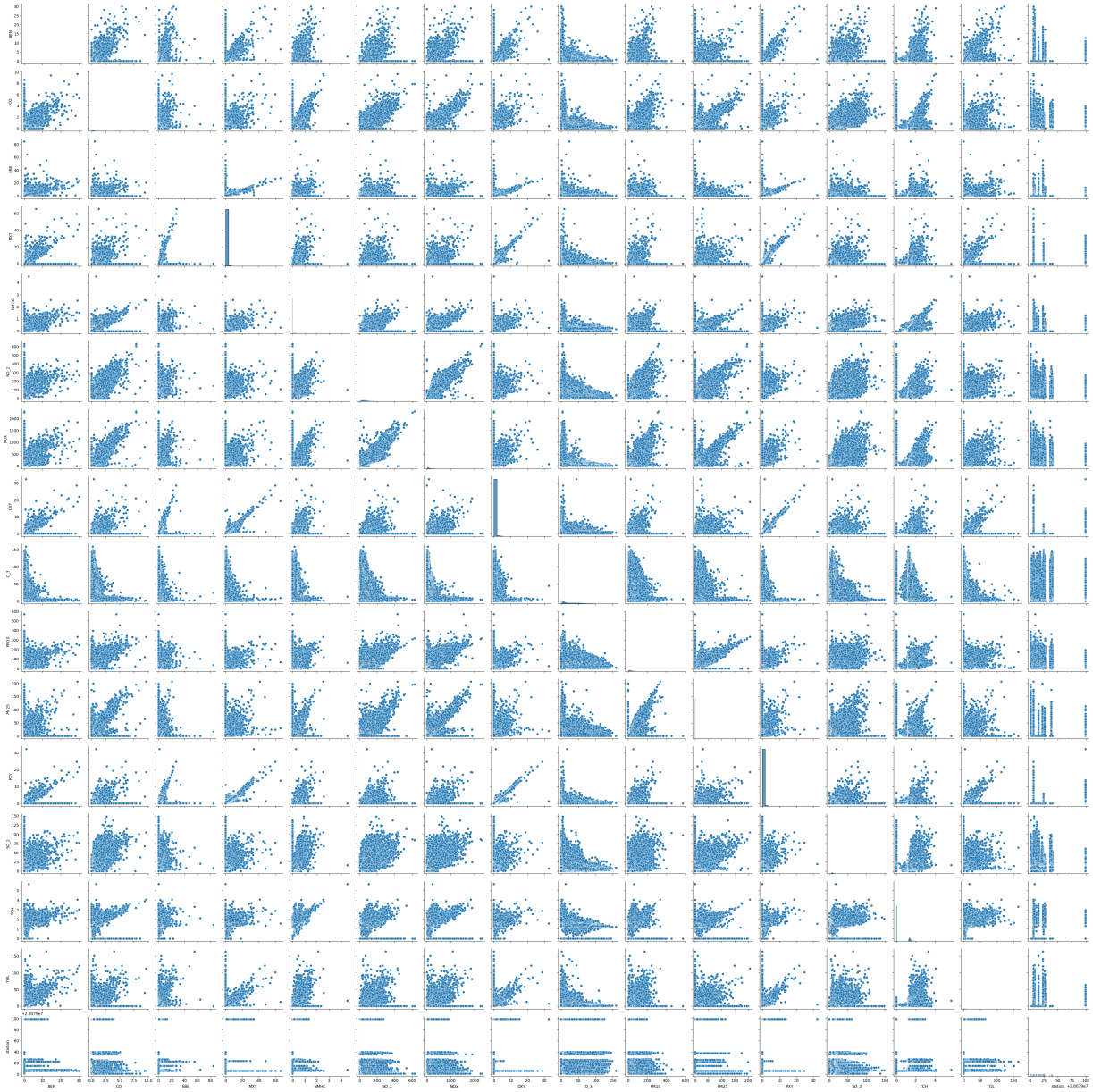
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_
0	0.00	2.86	0.00	0.00	0.00	282.200012	1054.000000	0.00	4.03000
1	0.00	1.82	0.00	0.00	0.00	86.419998	354.600006	0.00	3.26000
2	0.00	1.47	0.00	0.00	0.00	94.639999	319.000000	0.00	5.31000
3	0.00	1.64	0.00	0.00	0.00	127.900002	476.700012	0.00	4.50000
4	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.68999
...	...	...	...	...	...	...	...	...	.
225115	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.20999
225116	0.00	0.16	0.00	0.00	0.00	46.820000	51.480000	0.00	22.15000
225117	0.24	0.00	0.20	0.00	0.09	51.259998	66.809998	0.00	18.54000
225118	0.11	0.00	1.00	0.00	0.05	24.240000	36.930000	0.00	0.00000
225119	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.15000

225120 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)
```

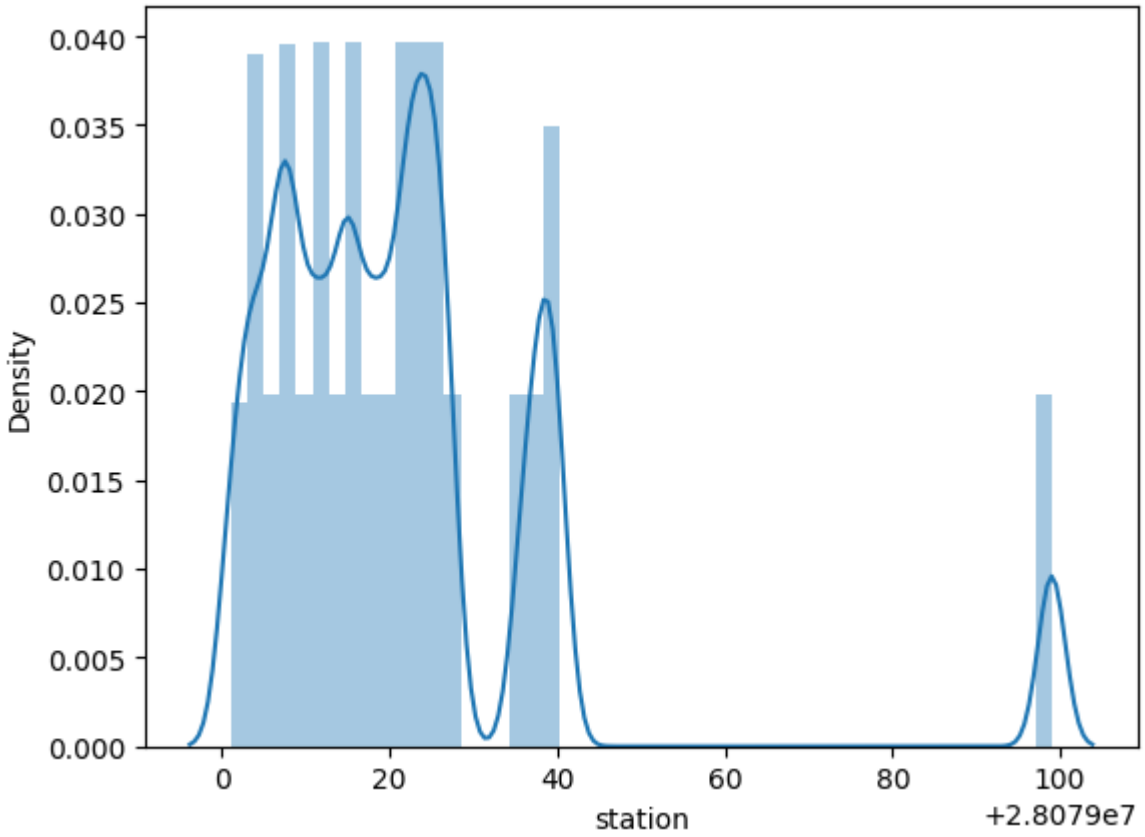
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x29aa7fbce90>
```



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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_20236\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]:

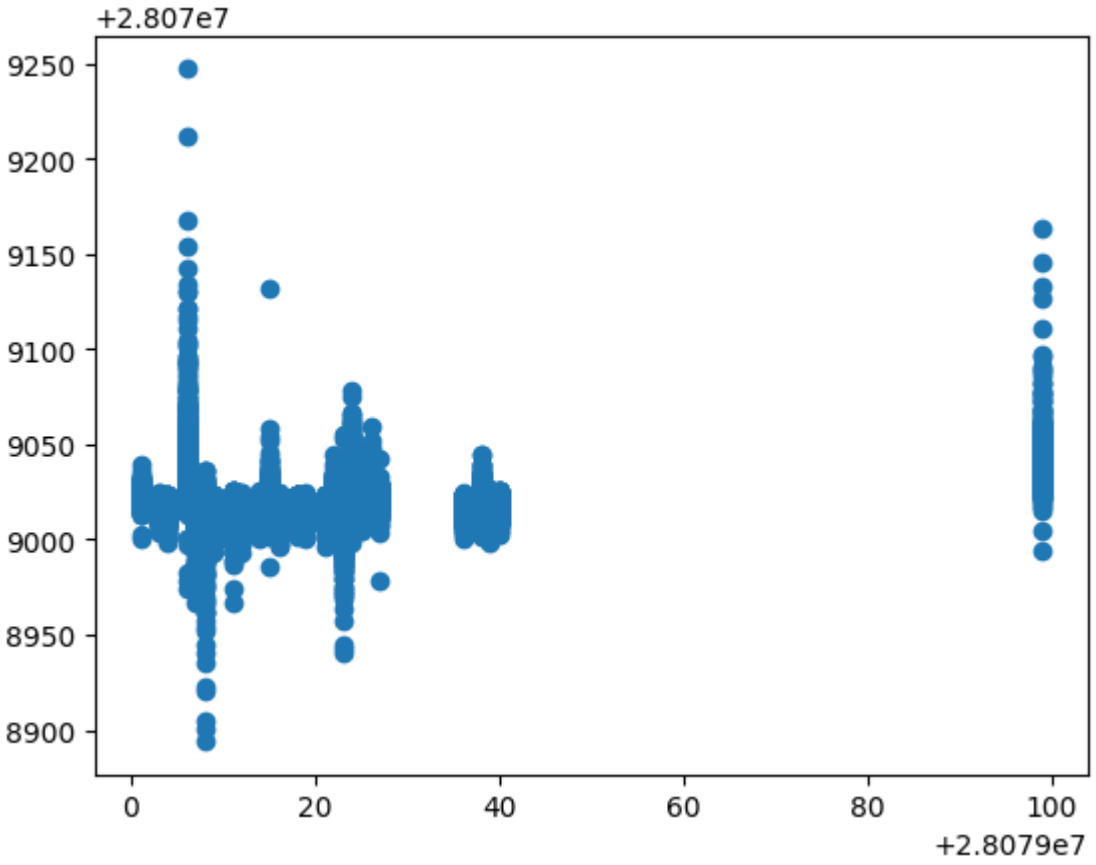
	Co-efficient
BEN	-4.901166
CO	-3.332806
EBE	1.320768
MXY	-4.076326
NMHC	-18.241084
NO_2	-0.038461
NOx	-0.000335
OXY	7.389035
O_3	-0.022905
PM10	0.025985
PM25	0.231014
PXY	13.800943
SO_2	-0.117629
TCH	3.086789
TOL	0.317025

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

```
28079023.788329333
```

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

Out[16]:
```
<matplotlib.collections.PathCollection at 0x29ace65e990>
```



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

```
0.1429631374738325
```

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

```
0.14756150749971852
```

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]:
```
0.14294134583097806
```

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]:
```
0.05464033474941632
```

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.          -0.           0.           1.53407132 -0.          -0.04534596
 -0.01527569  1.05895994 -0.02195549  0.01720276  0.32492437  0.86063822
 -0.15735869  0.03856496  0.03672839]
```

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

```
28079025.29709729
```

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

```
[28079024.34385792 28079021.33979359 28079019.91992053 ...
 28079019.69538361 28079018.8352405  28079011.60003699]
```

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

```
0.09907894813506457
```

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]: (225120, 15)

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

Out[32]: (225120,)

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, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

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

Out[39]: 0.46988717128642504

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

Out[40]:
```
1.745952639955472e-161
```

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

Out[41]:
```
6.936763848931293e-219
```

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]:
```
0.4933072731580195
```

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

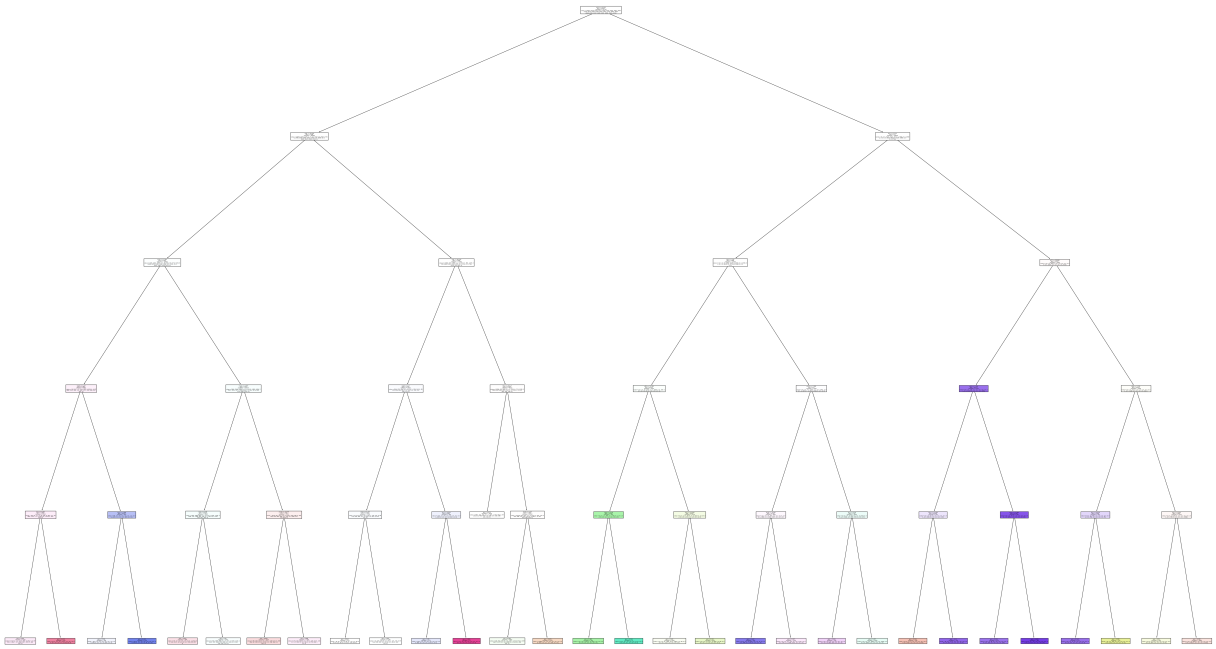
In [51]:
```
py.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[51]: [Text(0.49375, 0.9166666666666666, 'x[13] <= 0.305\ngini = 0.961\nsamples = 42794\nvalue = [2585, 2548, 2549, 2677, 2629, 2670, 2618, 2681, 2664\n2705, 2575, 2585, 2558, 2583, 2553, 2733, 2644, 2625\n2648, 2556, 2497, 2628, 2615, 2089, 2595, 2726]'),
 Text(0.25416666666666665, 0.75, 'x[5] <= 59.255\ngini = 0.938\nsamples = 26288\nvalue = [2585, 2548, 2549, 3, 10, 46, 2618, 25, 2664, 2705\n6, 2585, 2558, 2583, 2553, 2733, 5, 9, 2648, 32, 1\n2628, 2615, 2089, 2595, 0]'),
 Text(0.13333333333333333, 0.5833333333333334, 'x[12] <= 5.065\ngini = 0.933\nsamples = 14805\nvalue = [625, 1601, 1391, 2, 9, 45, 691, 18, 1773, 1430, 6\n2198, 1878, 1281, 1190, 1258, 5, 9, 1274, 31, 1\n1879, 1385, 1359, 1970, 0]'),
 Text(0.06666666666666667, 0.4166666666666667, 'x[10] <= 0.34\ngini = 0.886\nsamples = 3051\nvalue = [61, 55, 1, 2, 2, 45, 281, 9, 283, 4, 4, 87\n666, 4, 92, 450, 1, 6, 360, 21, 1, 979, 268, 497\n623, 0]'),
 Text(0.03333333333333333, 0.25, 'x[1] <= 0.465\ngini = 0.857\nsamples = 2594\nvalue = [16, 55, 1, 2, 2, 45, 281, 9, 283, 4, 3, 87\n666, 4, 92, 7, 1, 5, 360, 19, 1, 979, 5, 497\n623, 0]'),
 Text(0.016666666666666666, 0.08333333333333333, 'gini = 0.845\nsamples = 2293\nvalue = [16, 51, 1, 2, 2, 45, 265, 9, 282, 4, 3, 87\n657, 4, 86, 7, 1, 5, 360, 19, 1, 972, 5, 165\n530, 0]'),
 Text(0.05, 0.08333333333333333, 'gini = 0.455\nsamples = 301\nvalue = [0, 4, 0, 0, 0, 0, 16, 0, 1, 0, 0, 0, 9, 0\n6, 0, 0, 0, 0, 0, 0, 7, 0, 332, 93, 0]'),
 Text(0.1, 0.25, 'x[1] <= 0.285\ngini = 0.531\nsamples = 457\nvalue = [45, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0\n0, 443, 0, 1, 0, 2, 0, 0, 263, 0, 0, 0]'),
 Text(0.08333333333333333, 0.08333333333333333, 'gini = 0.534\nsamples = 309\nvalue = [15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0\n0, 246, 0, 0, 0, 2, 0, 0, 229, 0, 0, 0]'),
 Text(0.11666666666666667, 0.08333333333333333, 'gini = 0.405\nsamples = 148\nvalue = [30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 197, 0, 1, 0, 0, 0, 0, 34, 0, 0, 0]'),
 Text(0.2, 0.4166666666666667, 'x[12] <= 9.655\ngini = 0.93\nsamples = 11754\nvalue = [564, 1546, 1390, 0, 7, 0, 410, 9, 1490, 1426, 2\n2111, 1212, 1277, 1098, 808, 4, 3, 914, 10, 0, 900\n1117, 862, 1347, 0]'),
 Text(0.16666666666666666, 0.25, 'x[1] <= 0.145\ngini = 0.924\nsamples = 8043\nvalue = [405, 1089, 859, 0, 7, 0, 272, 9, 1031, 1136, 2\n1706, 747, 1168, 815, 540, 4, 3, 759, 10, 0, 294\n642, 847, 305, 0]'),
 Text(0.15, 0.08333333333333333, 'gini = 0.837\nsamples = 986\nvalue = [72, 5, 0, 0, 7, 0, 0, 4, 6, 22, 1, 236, 22\n3, 245, 45, 0, 0, 145, 10, 0, 0, 194, 451, 83\n0]'),
 Text(0.18333333333333332, 0.08333333333333333, 'gini = 0.92\nsamples = 7057\nvalue = [333, 1084, 859, 0, 0, 0, 272, 5, 1025, 1114, 1\n1470, 725, 1165, 570, 495, 4, 3, 614, 0, 0, 294\n448, 396, 222, 0]'),
 Text(0.23333333333333334, 0.25, 'x[12] <= 13.975\ngini = 0.91\nsamples = 3711\nvalue = [159, 457, 531, 0, 0, 0, 138, 0, 459, 290, 0, 405\n465, 109, 283, 268, 0, 0, 155, 0, 0, 606, 475\n15, 1042, 0]'),
 Text(0.21666666666666667, 0.08333333333333333, 'gini = 0.894\nsamples = 2524\nvalue = [85, 295, 323, 0, 0, 0, 94, 0, 239, 271, 0, 290\n288, 92, 183, 211, 0, 0, 90, 0, 0, 236, 289, 14\n961, 0]'),
 Text(0.25, 0.08333333333333333, 'gini = 0.898\nsamples = 1187\nvalue = [74, 162, 208, 0, 0, 0, 44, 0, 220, 19, 0, 115\n177, 17, 100, 57, 0, 0, 65, 0, 0, 370, 186, 1\n81, 0]'),
 Text(0.375, 0.5833333333333334, 'x[6] <= 99.755\ngini = 0.928\nsamples = 11483\nvalue = [1960, 947, 1158, 1, 1, 1, 1927, 7, 891, 1275, 0\n387, 680, 1475, 0, 0, 1374, 1, 0, 749\n1230, 730, 625, 0]'),

```
 Text(0.3333333333333333, 0.4166666666666667, 'x[10] <= 0.5\ngini = 0.931\n
samples = 2150\nvalue = [154, 273, 324, 0, 0, 0, 176, 1, 178, 147, 0, 89\n1
76, 238, 382, 265, 0, 0, 150, 0, 0, 215, 237\n239, 177, 0]'),
 Text(0.3, 0.25, 'x[12] <= 5.715\ngini = 0.914\nsamples = 1760\nvalue = [2,
273, 324, 0, 0, 0, 176, 1, 178, 147, 0, 89\n176, 238, 382, 2, 0, 0, 150, 0,
0, 215, 8, 239\n177, 0]'),
 Text(0.2833333333333333, 0.08333333333333333, 'gini = 0.844\nsamples = 375
\nvalue = [0, 8, 4, 0, 0, 0, 82, 0, 7, 0, 0, 2, 48, 0\n57, 0, 0, 0, 60, 0,
0, 133, 0, 134, 59, 0]'),
 Text(0.31666666666666665, 0.08333333333333333, 'gini = 0.904\nsamples = 13
85\nvalue = [2, 265, 320, 0, 0, 0, 94, 1, 171, 147, 0, 87\n128, 238, 325,
2, 0, 0, 90, 0, 0, 82, 8, 105\n118, 0]'),
 Text(0.36666666666666664, 0.25, 'x[1] <= 0.635\ngini = 0.651\nsamples = 39
0\nvalue = [152, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 263, 0, 0, 0, 0,
0, 0, 229, 0, 0, 0]'),
 Text(0.35, 0.08333333333333333, 'gini = 0.65\nsamples = 373\nvalue = [151,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 263, 0, 0, 0, 0, 0, 0, 207, 0, 0,
0]'),
 Text(0.38333333333333336, 0.08333333333333333, 'gini = 0.083\nsamples = 17
\nvalue = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 22, 0, 0, 0]'),
 Text(0.4166666666666667, 0.4166666666666667, 'x[6] <= 145.15\ngini = 0.924
\nsamples = 9333\nvalue = [1806, 674, 834, 1, 1, 1, 1751, 6, 713, 1128, 0\n
298, 504, 1064, 981, 1210, 0, 0, 1224, 1, 0, 534\n993, 491, 448, 0]'),
 Text(0.4, 0.25, 'gini = 0.929\nsamples = 3239\nvalue = [497, 322, 300, 0,
0, 1, 494, 2, 290, 385, 0, 94\n206, 351, 406, 478, 0, 0, 339, 0, 0, 192, 38
9\n201, 163, 0]'),
 Text(0.43333333333333335, 0.25, 'x[10] <= 1.09\ngini = 0.919\nsamples = 60
94\nvalue = [1309, 352, 534, 1, 1, 0, 1257, 4, 423, 743, 0\n204, 298, 713,
575, 732, 0, 0, 885, 1, 0, 342\n604, 290, 285, 0]'),
 Text(0.4166666666666667, 0.08333333333333333, 'gini = 0.903\nsamples = 449
7\nvalue = [20, 352, 534, 0, 1, 0, 1257, 4, 423, 743, 0, 204\n298, 713, 57
5, 2, 0, 0, 885, 0, 0, 342, 63, 290\n285, 0]'),
 Text(0.45, 0.08333333333333333, 'gini = 0.621\nsamples = 1597\nvalue = [12
89, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 730, 0, 0, 0, 1, 0, 0, 541,
0, 0, 0]'),
 Text(0.7333333333333333, 0.75, 'x[3] <= 0.075\ngini = 0.9\nsamples = 16506
\nvalue = [0, 0, 0, 2674, 2619, 2624, 0, 2656, 0, 0, 2569\n0, 0, 0, 0, 0, 2
639, 2616, 0, 2524, 2496, 0, 0\n0, 0, 2726]'),
 Text(0.6, 0.5833333333333334, 'x[0] <= 0.05\ngini = 0.858\nsamples = 11506
\nvalue = [0, 0, 0, 33, 2619, 2624, 0, 2656, 0, 0, 2569, 0\n0, 0, 0, 0, 263
9, 25, 0, 2524, 2496, 0, 0, 0\n0, 0]'),
 Text(0.5333333333333333, 0.4166666666666667, 'x[9] <= 12.085\ngini = 0.543
\nsamples = 3515\nvalue = [0, 0, 0, 29, 2619, 21, 0, 2656, 0, 0, 168, 0\n0,
0, 0, 0, 6, 24, 0, 3, 0, 0, 0, 0, 0, 0]'),
 Text(0.5, 0.25, 'x[10] <= 0.48\ngini = 0.479\nsamples = 797\nvalue = [0,
0, 0, 2, 440, 4, 0, 799, 0, 0, 14, 0, 0\n0, 0, 0, 4, 0, 0, 1, 0, 0, 0, 0,
0, 0]'),
 Text(0.48333333333333334, 0.08333333333333333, 'gini = 0.465\nsamples = 78
6\nvalue = [0, 0, 0, 0, 440, 4, 0, 799, 0, 0, 0, 0, 0, 0\n0, 0, 4, 0, 0, 0,
0, 0, 0, 0, 0, 0]'),
 Text(0.5166666666666667, 0.08333333333333333, 'gini = 0.304\nsamples = 11
\nvalue = [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 14, 0, 0, 0\n0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0]'),
 Text(0.5666666666666667, 0.25, 'x[9] <= 42.095\ngini = 0.547\nsamples = 27
18\nvalue = [0, 0, 0, 27, 2179, 17, 0, 1857, 0, 0, 154, 0\n0, 0, 0, 0, 2, 2
```

```
4, 0, 2, 0, 0, 0, 0, 0, 0]'),
 Text(0.55, 0.08333333333333333, 'gini = 0.542\nsamples = 1890\nvalue = [0,
0, 0, 21, 1464, 16, 0, 1369, 0, 0, 68, 0, 0\n0, 0, 0, 1, 23, 0, 2, 0, 0, 0,
0, 0, 0]'),
 Text(0.5833333333333334, 0.08333333333333333, 'gini = 0.551\nsamples = 828
\nvalue = [0, 0, 0, 6, 715, 1, 0, 488, 0, 0, 86, 0, 0\n0, 0, 0, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0]'),
 Text(0.6666666666666666, 0.4166666666666667, 'x[14] <= 2.365\ngini = 0.8\n
samples = 7991\nvalue = [0, 0, 0, 4, 0, 2603, 0, 0, 0, 0, 2401, 0, 0\n0, 0,
0, 2633, 1, 0, 2521, 2496, 0, 0, 0, 0, 0]'),
 Text(0.6333333333333333, 0.25, 'x[12] <= 6.365\ngini = 0.722\nsamples = 35
67\nvalue = [0, 0, 0, 0, 0, 746, 0, 0, 0, 0, 51, 0, 0, 0\n0, 0, 1089, 1, 0,
1839, 1971, 0, 0, 0, 0, 0]'),
 Text(0.6166666666666667, 0.08333333333333333, 'gini = 0.447\nsamples = 815
\nvalue = [0, 0, 0, 0, 0, 169, 0, 0, 0, 0, 12, 0, 0, 0\n0, 0, 949, 0, 0, 19
1, 0, 0, 0, 0, 0, 0]'),
 Text(0.65, 0.08333333333333333, 'gini = 0.637\nsamples = 2752\nvalue = [0,
0, 0, 0, 0, 577, 0, 0, 0, 0, 39, 0, 0, 0\n0, 0, 140, 1, 0, 1648, 1971, 0,
0, 0, 0, 0]'),
 Text(0.7, 0.25, 'x[1] <= 0.06\ngini = 0.75\nsamples = 4424\nvalue = [0, 0,
0, 4, 0, 1857, 0, 0, 0, 0, 2350, 0, 0\n0, 0, 0, 1544, 0, 0, 682, 525, 0, 0,
0, 0, 0]'),
 Text(0.6833333333333333, 0.08333333333333333, 'gini = 0.494\nsamples = 784
\nvalue = [0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 0, 0, 0, 682, 52
5, 0, 0, 0, 0, 0]'),
 Text(0.7166666666666667, 0.08333333333333333, 'gini = 0.657\nsamples = 364
0\nvalue = [0, 0, 0, 4, 0, 1855, 0, 0, 0, 0, 2349, 0, 0\n0, 0, 0, 1544, 0,
0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.8666666666666667, 0.5833333333333334, 'x[5] <= 29.965\ngini = 0.667
\nsamples = 5000\nvalue = [0, 0, 0, 2641, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0,
0, 0, 2591, 0, 0, 0, 0, 0, 0, 0, 2726]'),
 Text(0.8, 0.4166666666666667, 'x[4] <= 0.155\ngini = 0.377\nsamples = 1204
\nvalue = [0, 0, 0, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 1443, 0, 0,
0, 0, 0, 0, 0, 305]'),
 Text(0.7666666666666667, 0.25, 'x[11] <= 0.995\ngini = 0.628\nsamples = 30
3\nvalue = [0, 0, 0, 83, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 209, 0, 0,
0, 0, 0, 0, 0, 171]'),
 Text(0.75, 0.08333333333333333, 'gini = 0.594\nsamples = 184\nvalue = [0,
0, 0, 71, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 48, 0, 0, 0, 0, 0, 0, 0, 1
44]'),
 Text(0.7833333333333333, 0.08333333333333333, 'gini = 0.33\nsamples = 119
\nvalue = [0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 161, 0, 0,
0, 0, 0, 0, 0, 27]'),
 Text(0.8333333333333334, 0.25, 'x[11] <= 0.995\ngini = 0.227\nsamples = 90
1\nvalue = [0, 0, 0, 45, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 1234, 0, 0,
0, 0, 0, 0, 0, 134]'),
 Text(0.8166666666666667, 0.08333333333333333, 'gini = 0.384\nsamples = 434
\nvalue = [0, 0, 0, 38, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 519, 0, 0,
0, 0, 0, 0, 0, 125]'),
 Text(0.85, 0.08333333333333333, 'gini = 0.043\nsamples = 467\nvalue = [0,
0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 715, 0, 0, 0, 0, 0, 0, 0,
9]'),
 Text(0.9333333333333333, 0.4166666666666667, 'x[8] <= 5.125\ngini = 0.635
\nsamples = 3796\nvalue = [0, 0, 0, 2513, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0,
0, 0, 1148, 0, 0, 0, 0, 0, 0, 0, 2421]'),
 Text(0.9, 0.25, 'x[5] <= 84.09\ngini = 0.541\nsamples = 442\nvalue = [0,
```

```
0, 0, 278, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 359, 0, 0, 0, 0, 0, 0, 0,
35]'),
 Text(0.8833333333333333, 0.08333333333333333, 'gini = 0.345\nsamples = 227
\nvalue = [0, 0, 0, 73, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 277, 0, 0,
0, 0, 0, 0, 0, 4]'),
 Text(0.9166666666666666, 0.08333333333333333, 'gini = 0.508\nsamples = 215
\nvalue = [0, 0, 0, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 82, 0, 0,
0, 0, 0, 0, 0, 31]'),
 Text(0.9666666666666667, 0.25, 'x[2] <= 0.635\ngini = 0.614\nsamples = 335
4\nvalue = [0, 0, 0, 2235, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 789, 0,
0, 0, 0, 0, 0, 0, 2386]'),
 Text(0.95, 0.08333333333333333, 'gini = 0.56\nsamples = 466\nvalue = [0,
0, 0, 372, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 304, 0, 0, 0, 0, 0, 0, 0,
53]'),
 Text(0.9833333333333333, 0.08333333333333333, 'gini = 0.582\nsamples = 288
8\nvalue = [0, 0, 0, 1863, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 485, 0,
0, 0, 0, 0, 0, 0, 2333]')]
```



concusion

The bestfit model is Random Forest with score of 0.4933072731580195

In [ ]: