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_2005.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	1
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	1
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	3
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	2
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	3
...	...	...	...	...	...	...	...	...	...	
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	4
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	6
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	4
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	4

237000 rows × 17 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

In [4]:

```
df1 =df.fillna(value=0)
df1
```

Loading [MathJax]/extensions/Safe.js

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
0	2005-11-01 01:00:00	0.00	0.77	0.00	0.00	0.00	57.130001	128.699997	0.00	1
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	1
2	2005-11-01 01:00:00	0.00	0.40	0.00	0.00	0.00	46.119999	53.000000	0.00	3
3	2005-11-01 01:00:00	0.00	0.42	0.00	0.00	0.00	37.220001	52.009998	0.00	2
4	2005-11-01 01:00:00	0.00	0.57	0.00	0.00	0.00	32.160000	36.680000	0.00	3
...	...	...	...	...	...	...	...	...	...	
236995	2006-01-01 00:00:00	1.08	0.36	1.01	0.00	0.11	21.990000	23.610001	0.00	4
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	6
236997	2006-01-01 00:00:00	0.19	0.00	0.26	0.00	0.08	26.730000	30.809999	0.00	4
236998	2006-01-01 00:00:00	0.14	0.00	1.00	0.00	0.06	13.770000	17.770000	0.00	
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	4

237000 rows × 17 columns

In [5]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      237000 non-null  float64
 2   CO       237000 non-null  float64
 3   EBE      237000 non-null  float64
 4   MXY      237000 non-null  float64
 5   NMHC     237000 non-null  float64
 6   NO_2     237000 non-null  float64
 7   NOx      237000 non-null  float64
 8   OXY      237000 non-null  float64
 9   O_3      237000 non-null  float64
 10  PM10     237000 non-null  float64
 11  PM25     237000 non-null  float64
 12  PXY      237000 non-null  float64
 13  SO_2     237000 non-null  float64
 14  TCH      237000 non-null  float64
 15  TOL      237000 non-null  float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ 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	0.77	0.00	0.00	0.00	57.130001	128.699997	0.00	14.720000
1	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000
2	0.00	0.40	0.00	0.00	0.00	46.119999	53.000000	0.00	30.469999
3	0.00	0.42	0.00	0.00	0.00	37.220001	52.009998	0.00	21.379999
4	0.00	0.57	0.00	0.00	0.00	32.160000	36.680000	0.00	33.410000
...	...	...	...	...	...	...	...	...	...
236995	1.08	0.36	1.01	0.00	0.11	21.990000	23.610001	0.00	43.349998
236996	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999
236997	0.19	0.00	0.26	0.00	0.08	26.730000	30.809999	0.00	43.840000
236998	0.14	0.00	1.00	0.00	0.06	13.770000	17.770000	0.00	0.000000
236999	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998

237000 rows × 16 columns

In [8]:

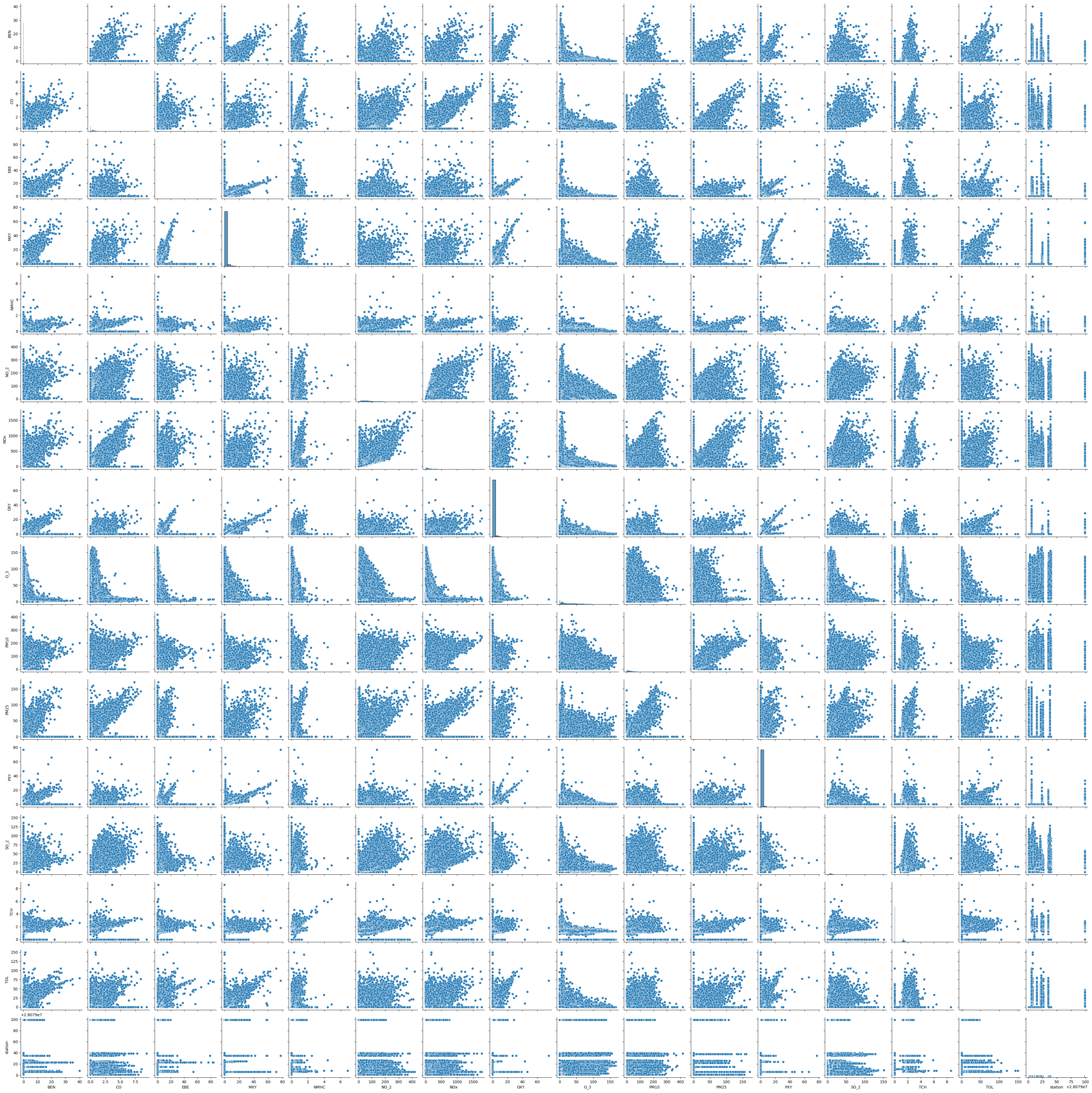
```
sns.pairplot(df2)
```

```
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 0x225dfebe190>
```

Loading [MathJax]/extensions/Safe.js

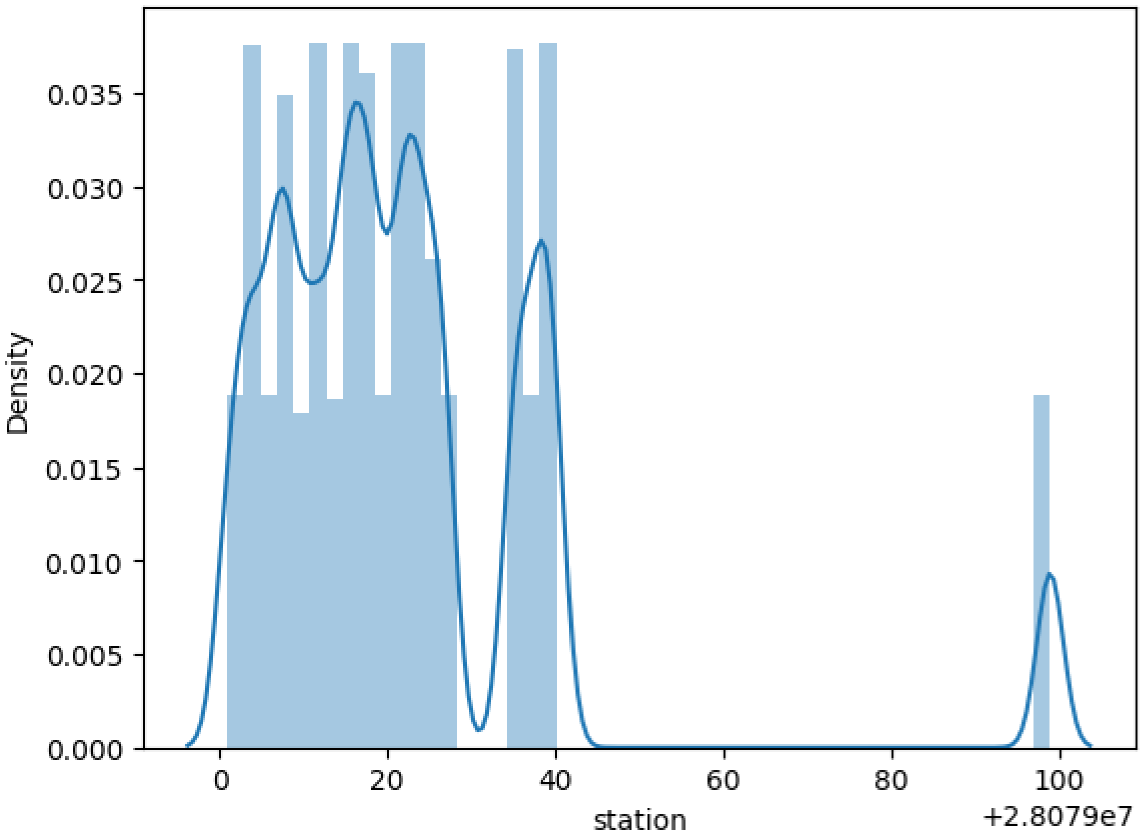


In [9]:
```python
sns.distplot(df2['station'])
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11432\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'>
```

Loading [MathJax]/extensions/Safe.js



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	-0.614996
CO	-0.793559
EBE	-0.716690
MXY	0.635453
NMHC	-10.555197
NO_2	0.012519
NOx	-0.019829
OXY	2.515576
O_3	0.002425
PM10	0.029608
PM25	0.180771
PXY	0.817951
SO_2	-0.166525
TCH	3.028844
TOL	0.006713

In [15]:

```
print(lr.intercept_)
```

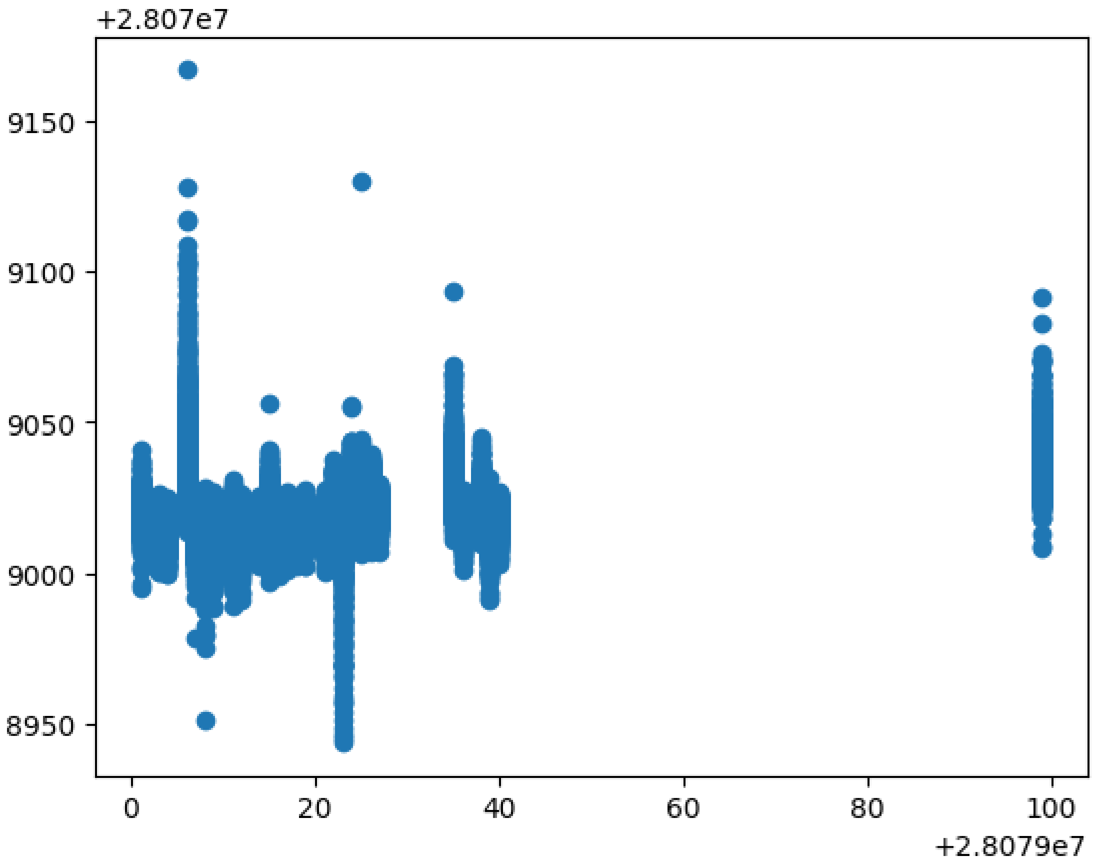
```
28079022.019519985
```

In [16]:

```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x225a145f7d0>
```



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

```
0.1107380832864725
```

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

```
0.1129082238421164
```

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]:
```
0.11074301929427832
```

Lasso

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

Out[22]:
```
▼    Lasso
Lasso(alpha=10)
```

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

Out[23]:
```
0.0498447911648362
```

elasticnet

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

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

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

```
[-0.00000000e+00 -0.00000000e+00 -0.00000000e+00  1.08064297e+00
  0.00000000e+00  1.55085453e-02 -2.79874177e-02  6.35639765e-01
 -6.97274698e-04  2.50663188e-02  2.18669845e-01  4.97420601e-01
 -1.73477848e-01  1.73667307e-01 -0.00000000e+00]
```

In [27]:
```
print(en.intercept_)
```

```
28079023.207832236
```

In [28]:
```
print(en.predict(x_test))
```

```
[28079021.8853725  28079022.23215573 28079020.45646902 ...
 28079021.99524672 28079023.04303021 28079022.53404317]
```

In [29]:
```
print(en.score(x_test,y_test))
```

```
0.10320154078626664
```

logistic

In [30]:
```
feature_matrix =df2.iloc[:,0:15]
target_vector=df2.iloc[:,-1]
```

Loading [MathJax]/extensions/Safe.js

In [31]:
```
feature_matrix=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df2['station']
```

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

Out[32]:
```
(237000, 15)
```

In [33]:
```
target_vector.shape
```

Out[33]:
```
(237000,)
```

In [34]:
```
from sklearn.preprocessing import StandardScaler
```

In [35]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [36]:
```
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[36]:
```
▼ LogisticRegression

LogisticRegression()
```

In [37]:
```
observation=[[1,2,3,4,5,6,7,8,9,11,12,13,14,15,16]]
```

In [38]:
```
prediction =logr.predict(observation)
print(prediction)
```

```
[28079024]
```

In [39]:
```
logr.classes_
```

Out[39]:
```
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 [40]:
```
logr.score(fs,target_vector)
```

Out[40]:
```
0.5786244725738396
```

Loading [MathJax]/extensions/Safe.js

In [41]:

```
logr.predict_proba(observation)[0][0]
```

Out[41]:

```
2.22237529650582e-149
```

In [42]:

```
logr.predict_proba(observation)[0][1]
```

Out[42]:

```
8.184342529276037e-203
```

Random forest

In [43]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [44]:

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

In [45]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.70)
```

In [46]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[46]:

```
▼ RandomForestClassifier
RandomForestClassifier()
```

In [47]:

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

In [48]:

```
from sklearn.model_selection import GridSearchCV
```

In [49]:

```
grid_search =GridSearchCV(estimator =rfc,param_grid=parameters,cv=2,scoring=
grid_search.fit(x_train,y_train)
```

Out[49]:

```
▶ GridSearchCV
▶ estimator: RandomForestClassifier
    ▶ RandomForestClassifier
```

In [50]:

```
grid_search.best_score_
```

Out[50]:

```
0.5341631504922644
```

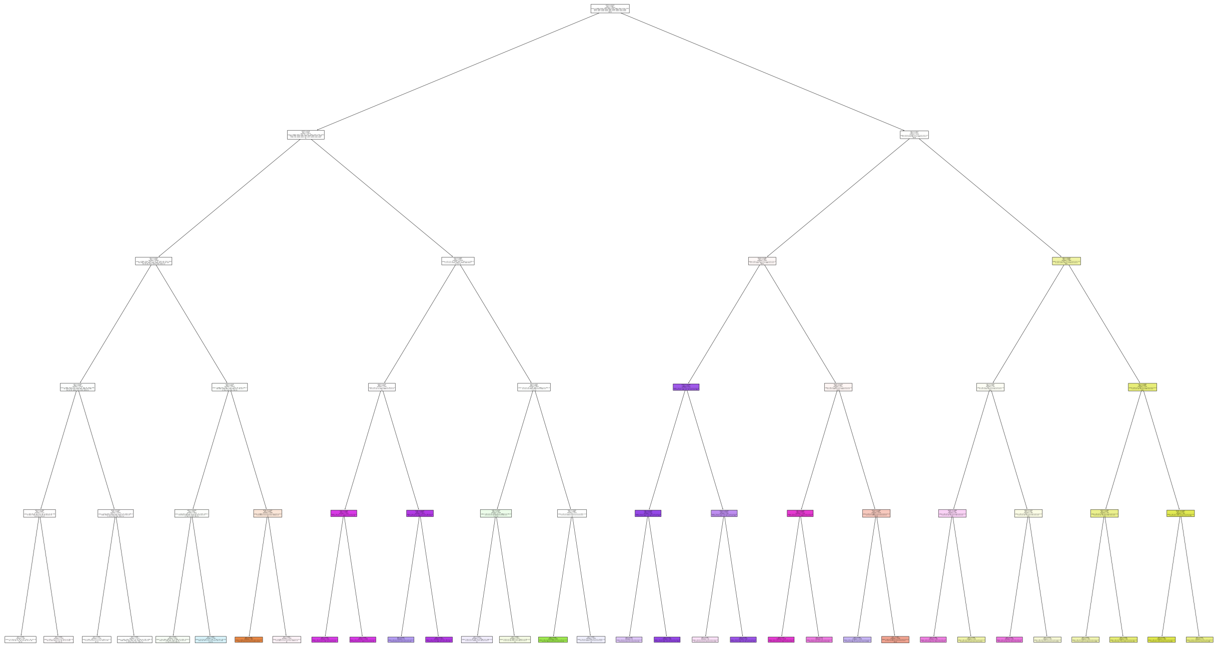
In [51]:

```
rfc_best=grid_search.best_estimator_
```

In [52]:

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

```
Out[52]: [Text(0.5, 0.9166666666666666, 'x[11] <= 0.115\ngini = 0.964\nsamples = 449
         24\nvalue = [2684, 2545, 2592, 2609, 2210, 2660, 2502, 2744, 2702\n2619, 25
         85, 2558, 2445, 2661, 2646, 2699, 2572, 2564\n2672, 1007, 2638, 2638, 2665,
         2571, 2639, 2526, 2620\n2527]'),
          Text(0.25, 0.75, 'x[13] <= 0.15\ngini = 0.958\nsamples = 38771\nvalue = [2
         684, 2545, 2592, 59, 2210, 2660, 2502, 2744, 2702\n2619, 2585, 2558, 2445,
         2661, 2646, 2699, 2572, 2564\n1499, 110, 2638, 2638, 157, 2571, 2639, 2526,
         2620\n3]'),
          Text(0.125, 0.5833333333333334, 'x[1] <= 0.545\ngini = 0.939\nsamples = 26
         562\nvalue = [2684, 2545, 2592, 27, 42, 25, 2502, 69, 2702, 2619\n46, 2558,
         2445, 2661, 2646, 2699, 2572, 49, 10, 110\n48, 29, 69, 2571, 2639, 2526, 26
         20, 0]'),
          Text(0.0625, 0.4166666666666667, 'x[1] <= 0.165\ngini = 0.932\nsamples = 1
         4140\nvalue = [804, 1573, 1547, 25, 30, 20, 165, 25, 1526, 1461\n39, 2043,
         1623, 513, 1679, 981, 1596, 29, 9, 96\n48, 29, 64, 1806, 1324, 1544, 1840,
         0]'),
          Text(0.03125, 0.25, 'x[6] <= 11.785\ngini = 0.893\nsamples = 1777\nvalue =
         [85, 66, 60, 22, 19, 9, 48, 3, 223, 13, 35, 535\n27, 66, 1, 144, 254, 2, 9,
         43, 48, 29, 62, 35\n214, 552, 219, 0]'),
          Text(0.015625, 0.08333333333333333, 'gini = 0.935\nsamples = 458\nvalue =
         [7, 64, 23, 22, 19, 9, 42, 3, 80, 7, 35, 44, 15\n12, 1, 35, 26, 2, 9, 40,
         6, 8, 62, 31, 2, 36\n80, 0]'),
          Text(0.046875, 0.08333333333333333, 'gini = 0.849\nsamples = 1319\nvalue =
         [78, 2, 37, 0, 0, 0, 6, 0, 143, 6, 0, 491, 12\n54, 0, 109, 228, 0, 0, 3, 4
         2, 21, 0, 4, 212\n516, 139, 0]'),
          Text(0.09375, 0.25, 'x[12] <= 5.115\ngini = 0.929\nsamples = 12363\nvalue
         = [719, 1507, 1487, 3, 11, 11, 117, 22, 1303, 1448, 4\n1508, 1596, 447, 167
         8, 837, 1342, 27, 0, 53, 0, 0\n2, 1771, 1110, 992, 1621, 0]'),
          Text(0.078125, 0.08333333333333333, 'gini = 0.785\nsamples = 1136\nvalue =
         [516, 156, 0, 0, 0, 1, 8, 0, 516, 0, 0, 0, 1\n5, 31, 0, 14, 15, 0, 0, 0, 0,
         0, 323, 0, 175\n36, 0]'),
          Text(0.109375, 0.08333333333333333, 'gini = 0.926\nsamples = 11227\nvalue
         = [203, 1351, 1487, 3, 11, 10, 109, 22, 787, 1448, 4\n1508, 1595, 442, 164
         7, 837, 1328, 12, 0, 53, 0, 0\n2, 1448, 1110, 817, 1585, 0]'),
          Text(0.1875, 0.4166666666666667, 'x[10] <= 0.45\ngini = 0.927\nsamples = 1
         2422\nvalue = [1880, 972, 1045, 2, 12, 5, 2337, 44, 1176, 1158, 7\n515, 82
         2, 2148, 967, 1718, 976, 20, 1, 14, 0, 0\n5, 765, 1315, 982, 780, 0]'),
          Text(0.15625, 0.25, 'x[8] <= 40.045\ngini = 0.91\nsamples = 9833\nvalue =
         [16, 972, 1045, 0, 12, 5, 2337, 44, 1176, 1158, 2\n515, 822, 2148, 967, 171
         8, 24, 20, 0, 14, 0, 0\n5, 765, 11, 982, 780, 0]'),
          Text(0.140625, 0.08333333333333333, 'gini = 0.915\nsamples = 8297\nvalue =
         [12, 928, 1005, 0, 12, 3, 1896, 37, 1014, 1113, 2\n453, 772, 1326, 937, 134
         1, 22, 20, 0, 14, 0, 0\n5, 730, 10, 843, 668, 0]'),
          Text(0.171875, 0.08333333333333333, 'gini = 0.808\nsamples = 1536\nvalue =
         [4, 44, 40, 0, 0, 2, 441, 7, 162, 45, 0, 62, 50\n822, 30, 377, 2, 0, 0, 0,
         0, 0, 0, 35, 1, 139\n112, 0]'),
          Text(0.21875, 0.25, 'x[12] <= 7.435\ngini = 0.643\nsamples = 2589\nvalue =
         [1864, 0, 0, 2, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0\n0, 0, 952, 0, 1, 0, 0, 0, 0,
         0, 1304, 0, 0\n0]'),
          Text(0.203125, 0.08333333333333333, 'gini = 0.109\nsamples = 487\nvalue =
         [716, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0\n0, 0, 21, 0, 0, 0, 0, 0, 0, 0,
         20, 0, 0, 0]'),
          Text(0.234375, 0.08333333333333333, 'gini = 0.662\nsamples = 2102\nvalue =
         [1148, 0, 0, 2, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0\n0, 0, 931, 0, 1, 0, 0, 0, 0,
         0, 1284, 0, 0\n0]'),
Loading [MathJax]/extensions/Safe.js 75, 0.5833333333333334, 'x[1] <= 0.005\ngini = 0.874\nsamples = 12
209\nvalue = [0, 0, 0, 32, 2168, 2635, 0, 2675, 0, 0, 2539, 0\n0, 0, 0, 0,
0, 2515, 1489, 0, 2590, 2609, 88, 0\n0, 0, 0, 3]'),
 Text(0.3125, 0.4166666666666667, 'x[10] <= 0.44\ngini = 0.507\nsamples = 3
330\nvalue = [0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 18, 0, 0, 0\n0, 0, 0, 0, 11, 0,
2590, 2609, 0, 0, 0, 0, 0\n0]'),
 Text(0.28125, 0.25, 'x[2] <= 0.145\ngini = 0.008\nsamples = 1651\nvalue =
[0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 5, 2609, 0, 0,
0, 0, 0, 0]'),
 Text(0.265625, 0.08333333333333333, 'gini = 0.022\nsamples = 513\nvalue =
[0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 3, 797, 0, 0,
0, 0, 0, 0]'),
 Text(0.296875, 0.08333333333333333, 'gini = 0.002\nsamples = 1138\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 2, 1812, 0, 0,
0, 0, 0, 0]'),
 Text(0.34375, 0.25, 'x[12] <= 6.325\ngini = 0.022\nsamples = 1679\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0\n0, 0, 0, 0, 11, 0, 2585, 0, 0,
0, 0, 0, 0, 0]'),
 Text(0.328125, 0.08333333333333333, 'gini = 0.444\nsamples = 8\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 8, 0, 4, 0, 0, 0, 0,
0, 0, 0]'),
 Text(0.359375, 0.08333333333333333, 'gini = 0.016\nsamples = 1671\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0\n0, 0, 0, 0, 3, 0, 2581, 0, 0,
0, 0, 0, 0, 0]'),
 Text(0.4375, 0.4166666666666667, 'x[0] <= 0.05\ngini = 0.831\nsamples = 88
79\nvalue = [0, 0, 0, 32, 2168, 2635, 0, 2669, 0, 0, 2521, 0\n0, 0, 0,
0, 2515, 1478, 0, 0, 0, 88, 0, 0\n0, 0, 3]'),
 Text(0.40625, 0.25, 'x[4] <= 0.175\ngini = 0.703\nsamples = 4358\nvalue =
[0, 0, 0, 32, 2168, 110, 0, 2669, 0, 0, 354, 0\n0, 0, 0, 0, 0, 14, 1478, 0,
0, 0, 84, 0, 0\n0, 0, 0]'),
 Text(0.390625, 0.08333333333333333, 'gini = 0.673\nsamples = 1786\nvalue =
[0, 0, 0, 20, 215, 73, 0, 1049, 0, 0, 266, 0, 0\n0, 0, 0, 0, 4, 1206, 0, 0,
0, 28, 0, 0, 0, 0\n0]'),
 Text(0.421875, 0.08333333333333333, 'gini = 0.602\nsamples = 2572\nvalue =
[0, 0, 0, 12, 1953, 37, 0, 1620, 0, 0, 88, 0, 0\n0, 0, 0, 0, 10, 272, 0, 0,
0, 56, 0, 0, 0, 0\n0]'),
 Text(0.46875, 0.25, 'x[0] <= 0.225\ngini = 0.666\nsamples = 4521\nvalue =
[0, 0, 0, 0, 0, 2525, 0, 0, 0, 0, 2167, 0, 0\n0, 0, 0, 0, 2501, 0, 0, 0, 0,
4, 0, 0, 0, 0\n3]'),
 Text(0.453125, 0.08333333333333333, 'gini = 0.185\nsamples = 1318\nvalue =
[0, 0, 0, 0, 0, 1885, 0, 0, 0, 0, 51, 0, 0, 0\n0, 0, 0, 156, 0, 0, 0, 0, 4,
0, 0, 0, 0, 0]'),
 Text(0.484375, 0.08333333333333333, 'gini = 0.601\nsamples = 3203\nvalue =
[0, 0, 0, 0, 0, 640, 0, 0, 0, 0, 2116, 0, 0\n0, 0, 0, 0, 2345, 0, 0, 0, 0,
0, 0, 0, 0, 0\n3]'),
 Text(0.75, 0.75, 'x[0] <= 1.815\ngini = 0.771\nsamples = 6153\nvalue = [0,
0, 0, 2550, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1173, 897, 0, 0, 250
8, 0, 0, 0, 0\n2524]'),
 Text(0.625, 0.5833333333333334, 'x[13] <= 0.465\ngini = 0.766\nsamples = 4
287\nvalue = [0, 0, 0, 873, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1092,
748, 0, 0, 1898, 0, 0, 0, 0\n2097]'),
 Text(0.5625, 0.4166666666666667, 'x[14] <= 4.245\ngini = 0.27\nsamples = 5
70\nvalue = [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 748,
0, 0, 72, 0, 0, 0, 0, 58]'),
 Text(0.53125, 0.25, 'x[1] <= 0.125\ngini = 0.143\nsamples = 401\nvalue =
[0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 587, 0, 0, 28, 0,
0, 0, 0, 16]'),
 Text(0.515625, 0.08333333333333333, 'gini = 0.562\nsamples = 6\nvalue =
[0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 6, 0, 0, 1, 0, 0,
0, 0, 0]'),
 Text(0.546875, 0.08333333333333333, 'gini = 0.131\nsamples = 395\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 581, 0, 0, 27, 0,
0, 0, 0, 16]'),
 Text(0.59375, 0.25, 'x[12] <= 9.155\ngini = 0.514\nsamples = 169\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 161, 0, 0, 44, 0,
0, 0, 0, 42]'),
 Text(0.578125, 0.08333333333333333, 'gini = 0.497\nsamples = 42\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 34, 0,
0, 0, 0, 29]'),
 Text(0.609375, 0.08333333333333333, 'gini = 0.226\nsamples = 127\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 161, 0, 0, 10, 0,
0, 0, 0, 13]'),
 Text(0.6875, 0.4166666666666667, 'x[10] <= 0.375\ngini = 0.722\nsamples =
3717\nvalue = [0, 0, 0, 869, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 109
2, 0, 0, 0, 1826, 0, 0, 0, 0\n2039]'),
 Text(0.65625, 0.25, 'x[1] <= 0.905\ngini = 0.013\nsamples = 1185\nvalue =
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 6, 0, 0, 0, 1826, 0,
0, 0, 0, 5]'),
 Text(0.640625, 0.08333333333333333, 'gini = 0.01\nsamples = 1179\nvalue =
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3, 0, 0, 0, 1817, 0,
0, 0, 0, 5]'),
 Text(0.671875, 0.08333333333333333, 'gini = 0.375\nsamples = 6\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3, 0, 0, 0, 9, 0, 0,
0, 0, 0]'),
 Text(0.71875, 0.25, 'x[11] <= 1.005\ngini = 0.618\nsamples = 2532\nvalue =
[0, 0, 0, 868, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1086, 0, 0, 0, 0,
0, 0, 0, 0\n2034]'),
 Text(0.703125, 0.08333333333333333, 'gini = 0.583\nsamples = 1104\nvalue =
[0, 0, 0, 267, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 962, 0, 0, 0, 0,
0, 0, 0, 0, 503]'),
 Text(0.734375, 0.08333333333333333, 'gini = 0.465\nsamples = 1428\nvalue =
[0, 0, 0, 601, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 124, 0, 0, 0, 0,
0, 0, 0, 0\n1531]'),
 Text(0.875, 0.5833333333333334, 'x[8] <= 8.945\ngini = 0.608\nsamples = 18
66\nvalue = [0, 0, 0, 1677, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 81, 1
49, 0, 0, 610, 0, 0, 0, 0\n427]'),
 Text(0.8125, 0.4166666666666667, 'x[1] <= 1.035\ngini = 0.742\nsamples = 7
45\nvalue = [0, 0, 0, 395, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 66, 11
6, 0, 0, 351, 0, 0, 0, 0\n262]'),
 Text(0.78125, 0.25, 'x[10] <= 1.695\ngini = 0.727\nsamples = 263\nvalue =
[0, 0, 0, 105, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 41, 63, 0, 0, 178,
0, 0, 0, 0\n41]'),
 Text(0.765625, 0.08333333333333333, 'gini = 0.386\nsamples = 145\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 63, 0, 0, 178, 0,
0, 0, 0, 0]'),
 Text(0.796875, 0.08333333333333333, 'gini = 0.589\nsamples = 118\nvalue =
[0, 0, 0, 105, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 41, 0, 0, 0, 0, 0,
0, 0, 0, 41]'),
 Text(0.84375, 0.25, 'x[10] <= 6.92\ngini = 0.714\nsamples = 482\nvalue =
[0, 0, 0, 290, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 25, 53, 0, 0, 173,
0, 0, 0, 0\n221]'),
 Text(0.828125, 0.08333333333333333, 'gini = 0.359\nsamples = 145\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 53, 0, 0, 173, 0,
0, 0, 0, 0]'),
 Text(0.859375, 0.08333333333333333, 'gini = 0.535\nsamples = 337\nvalue =
[0, 0, 0, 290, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 25, 0, 0, 0, 0, 0,
0, 0, 0, 221]'),
 Text(0.9375, 0.4166666666666667, 'x[1] <= 0.985\ngini = 0.435\nsamples = 1
121\nvalue = [0, 0, 0, 1282, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 15,
33, 0, 0, 259, 0, 0, 0, 0\n165]'),
 Text(0.90625, 0.25, 'x[0] <= 2.385\ngini = 0.511\nsamples = 761\nvalue =
[0, 0, 0, 783, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 14, 33, 0, 0, 244,
0, 0, 0, 0\n111]'),
 Text(0.890625, 0.08333333333333333, 'gini = 0.604\nsamples = 340\nvalue =
[0, 0, 0, 303, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5, 26, 0, 0, 143,
0, 0, 0, 0, 68]'),
 Text(0.921875, 0.08333333333333333, 'gini = 0.408\nsamples = 421\nvalue =
[0, 0, 0, 480, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 9, 7, 0, 0, 101,
0, 0, 0, 0, 43]'),
 Text(0.96875, 0.25, 'x[13] <= 1.645\ngini = 0.221\nsamples = 360\nvalue =
[0, 0, 0, 499, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1, 0, 0, 0, 15, 0,
0, 0, 0, 54]'),
 Text(0.953125, 0.08333333333333333, 'gini = 0.159\nsamples = 289\nvalue =
[0, 0, 0, 419, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1, 0, 0, 0, 12, 0,
0, 0, 0, 26]'),
 Text(0.984375, 0.08333333333333333, 'gini = 0.416\nsamples = 71\nvalue =
[0, 0, 0, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 3, 0,
0, 0, 0, 28]')]
```



concusion

The bestfit model is logistic Regression with score of 0.5786244725738396