In [119]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,precision_score,recall_score,f1_score
import warnings
warnings.filterwarnings('ignore')
```

In [64]:
```
voice=pd.read_csv('D:\\Dowloads\\voice.csv')
voice.head(10)
```

Out[64]:

IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx
0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000
0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632
0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512
0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119
0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274
0.141634	1.932562	8.308895	0.963181	0.738307	...	0.132786	0.110132	0.017112	0.253968	0.298222	0.007812	2.726562	2.718750	0.125160
0.112819	1.530643	5.987498	0.967573	0.762638	...	0.150762	0.105945	0.026230	0.266667	0.479620	0.007812	5.312500	5.304688	0.123992
0.121430	1.397156	4.766611	0.959255	0.719858	...	0.160514	0.093052	0.017758	0.144144	0.301339	0.007812	0.539062	0.531250	0.283937
0.120381	1.099746	4.070284	0.970723	0.770992	...	0.142239	0.096729	0.017957	0.250000	0.336476	0.007812	2.164062	2.156250	0.148272
0.126377	1.190368	4.787310	0.975246	0.804505	...	0.134329	0.105881	0.019300	0.262295	0.340365	0.015625	4.695312	4.679688	0.089920

Checking for null values and feature properties

In [65]:
```
voice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   meanfreq  3168 non-null   float64
 1   sd        3168 non-null   float64
 2   median    3168 non-null   float64
 3   Q25       3168 non-null   float64
 4   Q75       3168 non-null   float64
 5   IQR       3168 non-null   float64
 6   skew      3168 non-null   float64
 7   kurt      3168 non-null   float64
 8   sp.ent    3168 non-null   float64
 9   sfm       3168 non-null   float64
 10  mode      3168 non-null   float64
 11  centroid  3168 non-null   float64
 12  meanfun   3168 non-null   float64
 13  minfun    3168 non-null   float64
 14  maxfun    3168 non-null   float64
 15  meandom   3168 non-null   float64
 16  mindom    3168 non-null   float64
 17  maxdom    3168 non-null   float64
 18  dfrange   3168 non-null   float64
 19  modindx   3168 non-null   float64
 20  label     3168 non-null   object 
dtypes: float64(20), object(1)
memory usage: 519.9+ KB
```

In [66]:
```
voice.describe()
```

Out[66]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	n
count	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.00
mean	0.180907	0.057126	0.185621	0.140456	0.224765	0.084309	3.140168	36.568461	0.895127	0.408216	0.16
std	0.029918	0.016652	0.036360	0.048680	0.023639	0.042783	4.240529	134.928661	0.044980	0.177521	0.07
min	0.039363	0.018363	0.010975	0.000229	0.042946	0.014558	0.141735	2.068455	0.738651	0.036876	0.00
25%	0.163662	0.041954	0.169593	0.111087	0.208747	0.042560	1.649569	5.669547	0.861811	0.258041	0.11
50%	0.184838	0.059155	0.190032	0.140286	0.225684	0.094280	2.197101	8.318463	0.901767	0.396335	0.18
75%	0.199146	0.067020	0.210618	0.175939	0.243660	0.114175	2.931694	13.648905	0.928713	0.533676	0.22
max	0.251124	0.115273	0.261224	0.247347	0.273469	0.252225	34.725453	1309.612887	0.981997	0.842936	0.28

Encoding Data

In [67]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
voice['label'] = le.fit_transform(voice['label'])
```

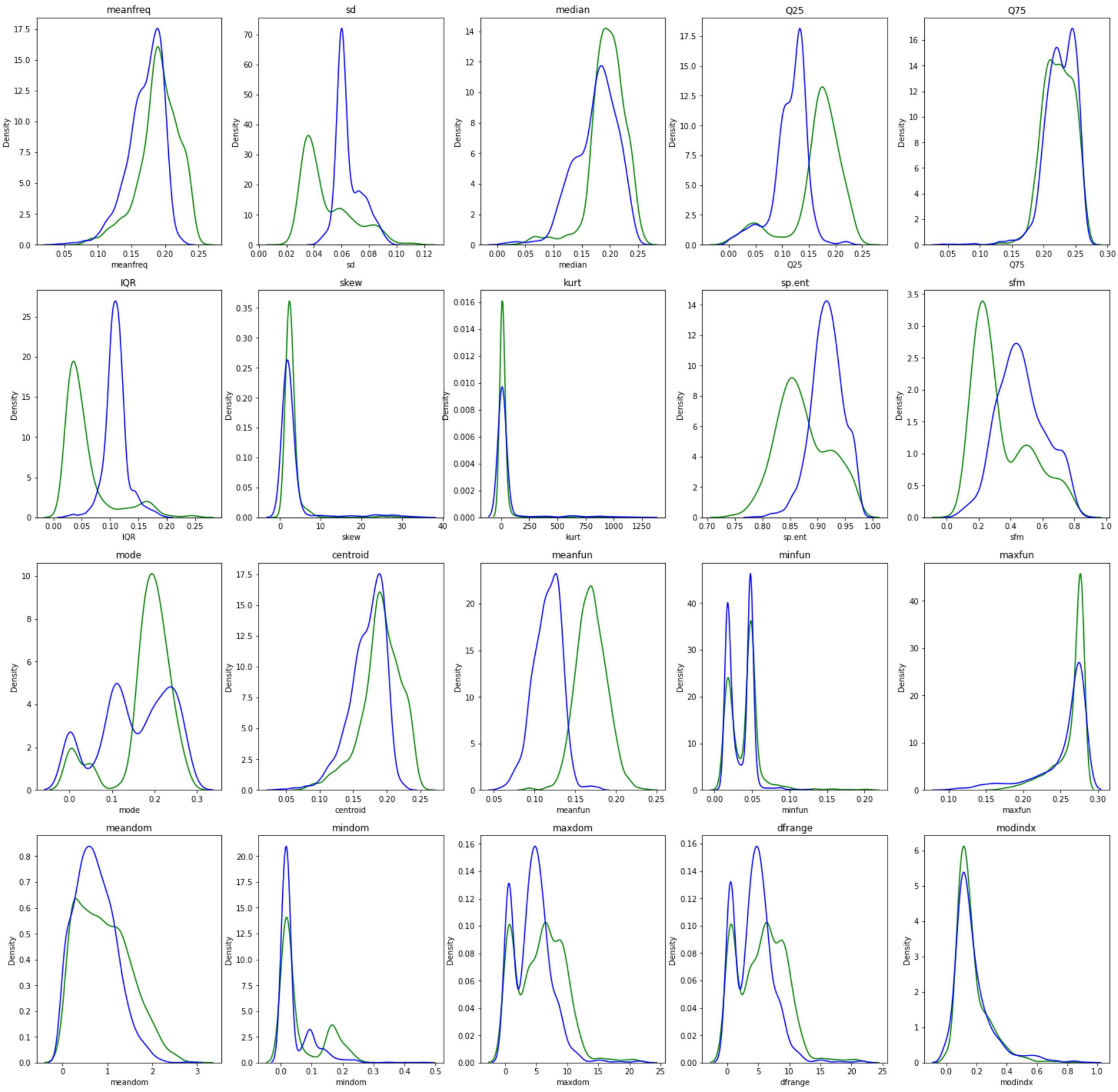
In [68]:
```
voice
```

Out[68]:

IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx
0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000
0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632
0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512
0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119
0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.151859	1.762129	6.630383	0.962934	0.763182	...	0.131884	0.182790	0.083770	0.262295	0.832899	0.007812	4.210938	4.203125	0.161929
0.162193	0.693730	2.503954	0.960716	0.709570	...	0.116221	0.188980	0.034409	0.275862	0.909856	0.039062	3.679688	3.640625	0.277897
0.190936	1.876502	6.604509	0.946854	0.654196	...	0.142056	0.209918	0.039506	0.275862	0.494271	0.007812	2.937500	2.929688	0.194759
0.176435	1.591065	5.388298	0.950436	0.675470	...	0.143659	0.172375	0.034483	0.250000	0.791360	0.007812	3.593750	3.585938	0.311002
0.180756	1.705029	5.769115	0.938829	0.601529	...	0.165509	0.185607	0.062257	0.271186	0.227022	0.007812	0.554688	0.546875	0.350000

In [69]:
```
data = voice.to_numpy()
```

In [70]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.subplots(4,5,figsize=(25,25))
for i in range(1,21):
    plt.subplot(4,5,i)
    plt.title(voice.columns[i-1])
    sns.kdeplot(voice.loc[voice['label'] == 0, voice.columns[i-1]], color= 'green', label='F')
    sns.kdeplot(voice.loc[voice['label'] == 1, voice.columns[i-1]], color= 'blue', label='M')
```



In [71]:
```
x = voice.iloc[:,:-1]
y = voice.iloc[:,-1]
```

In [72]:
```
y
```

Out[72]:
```
0       1
1       1
2       1
3       1
4       1
       ..
3163    0
3164    0
3165    0
3166    0
3167    0
Name: label, Length: 3168, dtype: int32
```

In [73]:
```
x
```

Out[73]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	min
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	0.000000	0.059781	0.084279	0.015
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	0.000000	0.066009	0.107937	0.015
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	0.000000	0.077316	0.098706	0.015
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	0.083878	0.151228	0.088965	0.017
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	0.104261	0.135120	0.106398	0.016
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3163	0.131884	0.084734	0.153707	0.049285	0.201144	0.151859	1.762129	6.630383	0.962934	0.763182	0.200836	0.131884	0.182790	0.083
3164	0.116221	0.089221	0.076758	0.042718	0.204911	0.162193	0.693730	2.503954	0.960716	0.709570	0.013683	0.116221	0.188980	0.034
3165	0.142056	0.095798	0.183731	0.033424	0.224360	0.190936	1.876502	6.604509	0.946854	0.654196	0.008006	0.142056	0.209918	0.039
3166	0.143659	0.090628	0.184976	0.043508	0.219943	0.176435	1.591065	5.388298	0.950436	0.675470	0.212202	0.143659	0.172375	0.034
3167	0.165509	0.092884	0.183044	0.070072	0.250827	0.180756	1.705029	5.769115	0.938829	0.601529	0.267702	0.165509	0.185607	0.062

3168 rows × 20 columns

Normalization

In [74]:
```
from sklearn.preprocessing import MinMaxScaler
mm_scaler = MinMaxScaler()
X = mm_scaler.fit_transform(x)
```

In [75]:
```
features=voice.columns
```

In [76]:
```
features
```

Out[76]:
```
Index(['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew', 'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun', 'maxfun',
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx', 'label'],
      dtype='object')
```

Splitting data

In [94]:
```
from sklearn import model_selection
X_train,X_test,Y_train,Y_test = model_selection.train_test_split(X,y,test_size = 0.25,random_state = 0)
```

Algorithms used for prediction

(i) Decision Tree

In [95]:
```
d_t = DecisionTreeClassifier()

d_t.fit(X_train,Y_train)

Y_pred_dt = d_t.predict(X_test)
```

In [101]:
```
Y_pred_dt
```

Out[101]:
```
array([0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0])
```

In [122]:
```
print("Confusion matrix is:")
print(confusion_matrix(Y_test,Y_pred_dt))
print("\nAccuracy score is:")
print(accuracy_score(Y_test,Y_pred_dt))
print("\nError rate is:")
print(1- (accuracy_score(Y_test,Y_pred_dt)))
print("\nPrecision score is:")
print(precision_score(Y_test,Y_pred_dt))
print("\nRecall score is:")
print(recall_score(Y_test,Y_pred_dt))
print("\nF1 score is:")
print(f1_score(Y_test,Y_pred_dt))
print()
print("\nClassification report is: ")
print(classification_report(Y_test,Y_pred_dt, digits = 5))
```

```
Confusion matrix is:
[[357  16]
 [ 15 404]]

Accuracy score is:
0.9608585858585859

Error rate is:
0.039141414141414144

Precision score is:
0.9619047619047619

Recall score is:
0.964200477326969

F1 score is:
0.9630512514898689


Classification report is:
              precision    recall  f1-score   support

           0    0.95968   0.95710   0.95839       373
           1    0.96190   0.96420   0.96305       419

    accuracy                        0.96086       792
   macro avg    0.96079   0.96065   0.96072       792
weighted avg    0.96086   0.96086   0.96086       792
```

(ii) Logistic Regression

In [98]:
```
l_r = LogisticRegression()

l_r.fit(X_train,Y_train)

Y_pred_lr = l_r.predict(X_test)
```

In [102]:
```
Y_pred_lr
```

Out[102]:
```
array([0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0])
```

In [123]:
```
print("Confusion matrix is:")
print(confusion_matrix(Y_test,Y_pred_lr))
print("\nAccuracy score is:")
print(accuracy_score(Y_test,Y_pred_lr))
print("\nError rate is:")
print(1- (accuracy_score(Y_test,Y_pred_lr)))
print("\nPrecision score is:")
print(precision_score(Y_test,Y_pred_lr))
print("\nRecall score is:")
print(recall_score(Y_test,Y_pred_lr))
print("\nF1 score is:")
print(f1_score(Y_test,Y_pred_lr))
print()
print("\nClassification report is: ")
print(classification_report(Y_test,Y_pred_lr, digits = 5))
```

```
Confusion matrix is:
[[356  17]
 [  6 413]]

Accuracy score is:
0.9709595959595959

Error rate is:
0.02904040404040409

Precision score is:
0.9604651162790697

Recall score is:
0.9856801909307876

F1 score is:
0.9729093050647821


Classification report is:
              precision    recall  f1-score   support

           0    0.98343   0.95442   0.96871       373
           1    0.96047   0.98568   0.97291       419

    accuracy                        0.97096       792
   macro avg    0.97195   0.97005   0.97081       792
weighted avg    0.97128   0.97096   0.97093       792
```

(iii) Ada-boost

In [104]:
```
a_b = AdaBoostClassifier()

a_b.fit(X_train,Y_train)

Y_pred_ab = a_b.predict(X_test)
```

In [105]:
```
Y_pred_ab
```

Out[105]:
```
array([0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0])
```

In [124]:
```
print("Confusion matrix is:")
print(confusion_matrix(Y_test,Y_pred_ab))
print("\nAccuracy score is:")
print(accuracy_score(Y_test,Y_pred_ab))
print("\nError rate is:")
print(1- (accuracy_score(Y_test,Y_pred_ab)))
print("\nPrecision score is:")
print(precision_score(Y_test,Y_pred_ab))
print("\nRecall score is:")
print(recall_score(Y_test,Y_pred_ab))
print("\nF1 score is:")
print(f1_score(Y_test,Y_pred_ab))
print()
print("\nClassification report is: ")
print(classification_report(Y_test,Y_pred_ab, digits = 5))
```

```
Confusion matrix is:
[[358  15]
 [  7 412]]

Accuracy score is:
0.9722222222222222

Error rate is:
0.02777777777777779

Precision score is:
0.9648711943793911

Recall score is:
0.9832935560859188

F1 score is:
0.9739952718676123


Classification report is:
              precision    recall  f1-score   support

           0    0.98082   0.95979   0.97019       373
           1    0.96487   0.98329   0.97400       419

    accuracy                        0.97222       792
   macro avg    0.97285   0.97154   0.97209       792
weighted avg    0.97238   0.97222   0.97220       792
```

(iv) Random Forest Classifier

In [108]:
```
r_f = RandomForestClassifier()

r_f.fit(X_train,Y_train)

Y_pred_rf = r_f.predict(X_test)
```

In [109]:
```
Y_pred_rf
```

Out[109]:
```
array([0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0])
```

In [125]:
```
print("Confusion matrix is:")
print(confusion_matrix(Y_test,Y_pred_rf))
print("\nAccuracy score is:")
print(accuracy_score(Y_test,Y_pred_rf))
print("\nError rate is:")
print(1- (accuracy_score(Y_test,Y_pred_rf)))
print("\nPrecision score is:")
print(precision_score(Y_test,Y_pred_rf))
print("\nRecall score is:")
print(recall_score(Y_test,Y_pred_rf))
print("\nF1 score is:")
print(f1_score(Y_test,Y_pred_rf))
print()
print("\nClassification report is: ")
print(classification_report(Y_test,Y_pred_rf, digits = 5))
```

```
Confusion matrix is:
[[368   5]
 [  6 413]]

Accuracy score is:
0.9861111111111112

Error rate is:
0.01388888888888884

Precision score is:
0.9880382775119617

Recall score is:
0.9856801909307876

F1 score is:
0.986857825567503


Classification report is:
              precision    recall  f1-score   support

           0    0.98396   0.98660   0.98527       373
           1    0.98804   0.98568   0.98686       419

    accuracy                        0.98611       792
   macro avg    0.98600   0.98614   0.98607       792
weighted avg    0.98612   0.98611   0.98611       792
```

In [ ]: