In [2]:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
data1=pd.read_csv("/home/placement/Downloads/TelecomCustomerChurn.csv")
#reading the values
```

In [5]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [6]:
```
data1['TotalCharges']=pd.to_numeric(data1['TotalCharges'],errors='coerce')
# change the dtype from object to integer
```

In [7]:
```
data1.isna().sum()
```

Out[7]:
```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

In [105]:
```
data1=data1.fillna(data1.median())Downloads/
data1
```

Out[105]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DevicePro
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	

7043 rows × 21 columns

In [5]:
```
y=data1['Churn']
x=data1.drop(['customerID','Churn'],axis=1)
```

In [6]:

```
x=pd.get_dummies(x)
x.isna().sum()
```

Out[6]:

```
SeniorCitizen          0
tenure                 0
MonthlyCharges         0
gender_Female          0
gender_Male            0
                      ..
TotalCharges_997.75    0
TotalCharges_998.1     0
TotalCharges_999.45    0
TotalCharges_999.8     0
TotalCharges_999.9     0
Length: 6575, dtype: int64
```

In [7]:
```
x
```

Out[7]:

	SeniorCitizen	tenure	MonthlyCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneServ
0	0	1	29.85	1	0	0	1	1	0	
1	0	34	56.95	0	1	1	0	1	0	
2	0	2	53.85	0	1	1	0	1	0	
3	0	45	42.30	0	1	1	0	1	0	
4	0	2	70.70	1	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	
7038	0	24	84.80	0	1	0	1	0	1	
7039	0	72	103.20	1	0	0	1	0	1	
7040	0	11	29.60	1	0	0	1	0	1	
7041	1	4	74.40	0	1	0	1	1	0	
7042	0	66	105.65	0	1	1	0	1	0	

7043 rows × 6575 columns

In [ ]:

In [110]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [111]:

```
from sklearn.model_selection import GridSearchCV #GridSearchCV is for parameter tuning
from sklearn.ensemble import RandomForestClassifier
cls=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200] #number of decision trees in the forest, default = 100
criterion=['gini','entropy'] #criteria for choosing nodes default = 'gini'
max_depth=[3,5,10] #maximum number of nodes in a tree default = None (it will go till all possible nodes)
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth} #this will undergo 8*2
RFC_cls = GridSearchCV(cls, parameters)
RFC_cls.fit(x_train,y_train)
```

Out[111]:

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [112]:

```
RFC_cls.best_params_
```

Out[112]:

```
{'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 150}
```

In [113]:

```
cls=RandomForestClassifier(n_estimators=200,criterion='entropy',max_depth=10)
```

In [114]:

```
cls.fit(x_train,y_train)
```

Out[114]:

```
RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=200)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [116]:

```
p=cls.predict(x_test)
```

In [117]:

```
p
```

Out[117]:

```
array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [118]: from sklearn.metrics import confusion_matrix
          confusion_matrix(y_test,p)
```

```
Out[118]: array([[1548,  149],
                 [ 299,  329]])
```

```
In [119]: from sklearn.metrics import accuracy_score
          accuracy_score(y_test,p)
```

```
Out[119]: 0.8073118279569892
```

```
In [120]: from sklearn.linear_model import LogisticRegression
          clas=LogisticRegression()
          clas.fit(x_train,y_train)
```

```
Out[120]: LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [121]: y_pred=clas.predict(x_test)
```

```
In [124]: y_pred
```

```
Out[124]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [125]: from sklearn.metrics import confusion_matrix
          confusion_matrix(y_test,y_pred)
```

```
Out[125]: array([[1526,  171],
                 [ 266,  362]])
```

```
In [123]: from sklearn.metrics import accuracy_score
          accuracy_score(y_test,y_pred)
```

```
Out[123]: 0.8120430107526881
```

In [ ]:

In [ ]:

In [ ]: