In [45]:
```
import pandas as pd
import numpy as np
```

In [46]:
```
data=pd.read_csv("/home/placement/Downloads/TelecomCustomerChurn.csv")
```

In [47]:
```
data.head()
```

Out[47]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtect
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	...	

5 rows × 21 columns

In [61]:
```
data.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   float64
 20  Churn             7043 non-null   object 
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

```
In [49]: data.isna().sum()
```

```
Out[49]: 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        0
         Churn               0
         dtype: int64
```

In [50]:
```
data.dtypes
```

Out[50]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [51]:
```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'],errors='coerce')
#data['TotalCharges'] = data['TotalCharges'].replace('',np.nan).astype(float).value
```

In [52]:
```
data.dtypes
```

Out[52]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

In [53]:
```
#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
#RFC_cls = GridSearchCV(cls, parameters)
#RFC_cls.fit(X_train,y_train)
```

In [54]:
```
databackup=data.copy()
```

In [55]:
```
data['TotalCharges']=data['TotalCharges'].fillna(data['TotalCharges'].median())
```

In [56]:
```
x=data.drop(['customerID','Churn'],axis=1)
y=data['Churn']
```

In [57]:
```
x=pd.get_dummies(x)
```

In [58]:
```
x.head()
```

Out[58]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes
0	0	1	29.85	29.85	1	0	0	1	1	0
1	0	34	56.95	1889.50	0	1	1	0	1	0
2	0	2	53.85	108.15	0	1	1	0	1	0
3	0	45	42.30	1840.75	0	1	1	0	1	0
4	0	2	70.70	151.65	1	0	1	0	1	0

5 rows × 45 columns

In [63]:
```
list(data)
```

Out[63]:
```
['customerID',
 'gender',
 'SeniorCitizen',
 'Partner',
 'Dependents',
 'tenure',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'MonthlyCharges',
 'TotalCharges',
 'Churn']
```

In [59]:
```
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 [60]:
```
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[60]:
```
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 [64]:
```
RFC_cls.best_params_
```

Out[64]:
```
{'criterion': 'gini', 'max_depth': 10, 'n_estimators': 50}
```

In [66]:
```
cls=RandomForestClassifier(n_estimators=50,criterion='entropy',max_depth=10)
```

In [67]:
```
cls.fit(x_train,y_train)
```

Out[67]:
```
RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=50)
```

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 [68]:
```
rfy_pred=cls.predict(x_test)
```

In [69]:
```
rfy_pred
```

Out[69]:
```
array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

In [70]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,rfy_pred)
```

Out[70]:
```
array([[1538,  159],
       [ 300,  328]])
```

In [71]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,rfy_pred)
```

Out[71]:
```
0.8025806451612904
```

In [72]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
classifier.fit(x_train,y_train)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceW
arning: lbfgs failed to converge (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 (https://scikit-learn.org/stable/modules/pr
eprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.or
g/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[72]:
```
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 [73]:
```
y_pred=classifier.predict(x_test)
```

In [74]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[74]:
```
0.8120430107526881
```

```
In [ ]:
```