In [31]:
```
import pandas as r
d=r.read_csv("/home/placement/Downloads/TelecomCustomerChurn.csv")
d.describe()
```

Out[31]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [32]:
```
d.head()
```

Out[32]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtec
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 [33]:
```
d.isna().sum()
```

Out[33]:
```
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 [34]:
```
d.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 [35]:
```
#d1=d.drop(['customerID','gender','PaymentMethod','PaperlessBilling','Dependents','OnlineSecurity','OnlineBa
 #'TechSupport','StreamingTV','StreamingMovies','InternetService'],axis=1)
```

In [36]:
```
d['Churn']=d['Churn'].map({'Yes':1,'No':0})
```

In [37]:
```
d['PhoneService']=d['PhoneService'].map({'Yes':1,'No':0})
```

In [38]:
```
d['Partner']=d['Partner'].map({'Yes':1,'No':0})
```

In [39]:
```
d['TotalCharges']=r.to_numeric(d['TotalCharges'], errors='coerce')
```

In [ ]:
```
```

In [40]:
```
d.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   int64  
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   int64  
 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      7032 non-null   float64
 20  Churn             7043 non-null   int64  
dtypes: float64(2), int64(5), object(14)
memory usage: 1.1+ MB
```

In [41]:
```
#d=d.fillna(d.mean())
```

In [42]:
```
d
```

Out[42]:

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

7043 rows × 21 columns

In [43]:
```
d.isna().sum()
```

Out[43]:
```
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 [44]:
```
d=d.fillna(d.mean())
```

```
/tmp/ipykernel_13877/1862675393.py:1: FutureWarning: The default value of numeric_only in DataFrame.mean is
deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is d
eprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  d=d.fillna(d.mean())
```

In [45]:
```
y=d['Churn']
x=d.drop(['customerID','Churn'],axis=1)
```

In [46]:
```
x.isna().sum()
```

Out[46]:
```
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
dtype: int64
```

In [53]:
```
x
```

Out[53]:
```
      SeniorCitizen  Partner  tenure  PhoneService  MonthlyCharges  TotalCharges  gender_Female  gender_Male  Dependents_No  Dependents_Ye
   0              0        1       1             0           29.85         29.85              1            0              1
   1              0        0      34             1           56.95       1889.50              0            1              1
   2              0        0       2             1           53.85        108.15              0            1              1
   3              0        0      45             0           42.30       1840.75              0            1              1
   4              0        0       2             1           70.70        151.65              1            0              1
 ...            ...      ...     ...           ...             ...           ...            ...          ...            ...
7038              0        1      24             1           84.80       1990.50              0            1              0
7039              0        1      72             1          103.20       7362.90              1            0              0
7040              0        1      11             0           29.60        346.45              1            0              0
7041              1        1       4             1           74.40        306.60              0            1              1
7042              0        0      66             1          105.65       6844.50              0            1              1

7043 rows × 43 columns
```

In [48]:
```
#d=r.get_dummies(d)
```

In [49]:
```
x=r.get_dummies(x)
x
```

Out[49]:

	SeniorCitizen	Partner	tenure	PhoneService	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Dependents_No	Dependents_Ye
0	0	1	1	0	29.85	29.85	1	0	1	(
1	0	0	34	1	56.95	1889.50	0	1	1	(
2	0	0	2	1	53.85	108.15	0	1	1	(
3	0	0	45	0	42.30	1840.75	0	1	1	(
4	0	0	2	1	70.70	151.65	1	0	1	(
...	...	...	...	...	...	...	...	...	...	..
7038	0	1	24	1	84.80	1990.50	0	1	0	1
7039	0	1	72	1	103.20	7362.90	1	0	0	1
7040	0	1	11	0	29.60	346.45	1	0	0	1
7041	1	1	4	1	74.40	306.60	0	1	1	(
7042	0	0	66	1	105.65	6844.50	0	1	1	(

7043 rows × 43 columns

In [66]:
```
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 [51]:
```
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[51]:
```
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 [56]:
```
RFC_cls.best_params_
```

Out[56]:
```
{'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 200}
```

In [57]:
```
cls=RandomForestClassifier(n_estimators=200,criterion="entropy",max_depth=10)
```

In [58]:
```
cls.fit(x_train,y_train)
```

Out[58]:
```
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 [61]:
```
rfy_pred=cls.predict(x_test)
```

In [62]:
```
rfy_pred
```

Out[62]:
```
array([1, 0, 0, ..., 1, 0, 0])
```

In [64]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,rfy_pred)
```

Out[64]:
```
array([[1548,  149],
       [ 293,  335]])
```

In [65]:
```
from sklearn.metrics import accuracy_score
accuracy_score(rfy_pred,y_test)
```

Out[65]:
```
0.8098924731182796
```

In [68]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
classifier.fit(x_train,y_train)
```

Out[68]:
```
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 [69]:
```
y_pred=classifier.predict(x_test)
y_pred
```

Out[69]:
```
array([1, 0, 0, ..., 1, 0, 0])
```

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

Out[70]:
```
array([[1538,  159],
       [ 277,  351]])
```

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

```
Out[71]: 0.8124731182795699
```

```
In [ ]:
```