

```
In [29]: import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

```
In [30]: data=pd.read_csv("/home/placenet/Downloads/TelecomCustomerChurn.csv")
```

```
In [31]: data['TotalCharges']=pd.to_numeric(data['TotalCharges'],errors='coerce')
```

```
In [32]: data.describe()
```

```
Out[32]:
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7032.000000
mean	0.162147	32.371149	64.761692	2283.300441
std	0.368612	24.559481	30.090047	2266.771362
min	0.000000	0.000000	18.250000	18.800000
25%	0.000000	9.000000	35.500000	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.850000	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

```
In [33]: 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          7032 non-null   float64
20  Churn                 7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

In [34]: data.head()

Out[34]:

	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 [35]: data.isna().sum()
```

```
Out[35]: 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 [36]: data1=data.fillna(data.median())
data1
```

```
Out[36]:
```

	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 [37]: data1.isna().sum()
```

```
Out[37]: 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 [38]: list(data1)
```

```
Out[38]: ['customerID',  
          'gender',  
          'SeniorCitizen',  
          'Partner',  
          'Dependents',  
          'tenure',  
          'PhoneService',  
          'MultipleLines',  
          'InternetService',  
          'OnlineSecurity',  
          'OnlineBackup',  
          'DeviceProtection',  
          'TechSupport',  
          'StreamingTV',  
          'StreamingMovies',  
          'Contract',  
          'PaperlessBilling',  
          'PaymentMethod',  
          'MonthlyCharges',  
          'TotalCharges',  
          'Churn']
```

```
In [39]: data1.shape
```

```
Out[39]: (7043, 21)
```

```
In [40]: data1['SeniorCitizen']=data1['SeniorCitizen'].map({0:'No',1:'Yes'})
data1
```

```
Out[40]:
```

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

7043 rows × 21 columns



```
In [41]: x=data1.drop(['customerID','Churn'],axis=1)
y=data1['Churn']
```



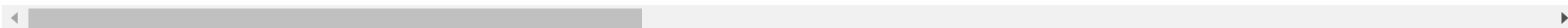
```
In [42]: x=pd.get_dummies(x)
```

```
In [43]: x.head(10)
```

```
Out[43]:
```

	tenure	MonthlyCharges	TotalCharges	gender_Female	gender_Male	SeniorCitizen_No	SeniorCitizen_Yes	Partner_No	Partner_Yes	Dependent
0	1	29.85	29.85	1	0	1	0	0	1	
1	34	56.95	1889.50	0	1	1	0	1	0	
2	2	53.85	108.15	0	1	1	0	1	0	
3	45	42.30	1840.75	0	1	1	0	1	0	
4	2	70.70	151.65	1	0	1	0	1	0	
5	8	99.65	820.50	1	0	1	0	1	0	
6	22	89.10	1949.40	0	1	1	0	1	0	
7	10	29.75	301.90	1	0	1	0	1	0	
8	28	104.80	3046.05	1	0	1	0	0	1	
9	62	56.15	3487.95	0	1	1	0	1	0	

10 rows × 46 columns



```
In [44]: #divide the data into testing & training
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 [45]: #importing Random Forest Classifier from sklearn.ensemble
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[45]: 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 [46]: RFC_cls.best_params_
```

```
Out[46]: {'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 125}
```

```
In [47]: cls=RandomForestClassifier(n_estimators=125,criterion='entropy',max_depth=10)
```

```
In [48]: cls.fit(x_train,y_train)
```

```
Out[48]: RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=125)
```

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

```
Out[49]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [50]: from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,rfy_pred)
```

```
Out[50]: array([[1556, 141],  
               [ 304, 324]])
```

```
In [51]: from sklearn.metrics import accuracy_score  
accuracy_score(y_test,rfy_pred)
```

```
Out[51]: 0.8086021505376344
```

```
In [52]: #Logistic Regression  
from sklearn.linear_model import LogisticRegression  
classifier=LogisticRegression()  
classifier.fit(x_train,y_train)#command for taining/fitting the model
```

```
Out[52]: 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 [53]: y_pred=classifier.predict(x_test)  
y_pred
```

```
Out[53]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [54]: from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,y_pred)
```

```
Out[54]: array([[1538, 159],  
               [ 279, 349]])
```

```
In [55]: from sklearn.metrics import accuracy_score  
accuracy_score(y_test,y_pred)
```

```
Out[55]: 0.8116129032258065
```

```
In [ ]:
```

