

```
In [1]: import pandas as pd  
import matplotlib.pyplot as plt
```

```
In [2]: data=pd.read_csv("/home/placement/Downloads/TelecomCustomerChurn.csv")
```

```
In [3]: data.head()
```

Out[3]:

	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 [4]: 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   object
20  Churn                 7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
In [5]: data.isna().sum()
```

```
Out[5]: 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 [6]: data.dtypes
```

```
Out[6]: 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 [7]: data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
```

```
In [8]: data.dtypes
```

```
Out[8]: 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 [9]: databackup=data.copy()
```

```
In [10]: data['TotalCharges']=data['TotalCharges'].fillna(data['TotalCharges'].median())
```

```
In [11]: x=data.drop(['customerID','Churn'],axis=1)
y=data['Churn']
```

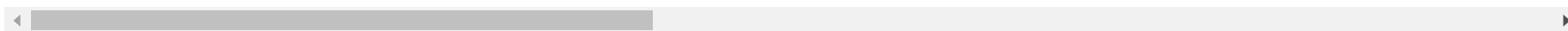
```
In [12]: x=pd.get_dummies(x)
```

```
In [13]: x.head()
```

```
Out[13]:
```

	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



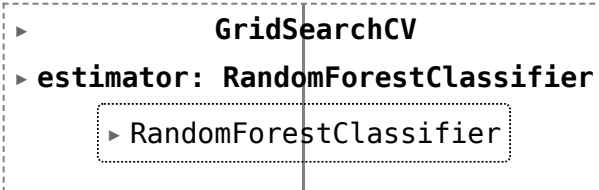
```
In [14]: list(data)
```

```
Out[14]: ['customerID',  
          'gender',  
          'SeniorCitizen',  
          'Partner',  
          'Dependents',  
          'tenure',  
          'PhoneService',  
          'MultipleLines',  
          'InternetService',  
          'OnlineSecurity',  
          'OnlineBackup',  
          'DeviceProtection',  
          'TechSupport',  
          'StreamingTV',  
          'StreamingMovies',  
          'Contract',  
          'PaperlessBilling',  
          'PaymentMethod',  
          'MonthlyCharges',  
          'TotalCharges',  
          'Churn']
```

```
In [15]: 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 [16]: from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
cls=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200] t = 100
criterion=['gini','entropy']
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[16]:
```



```

  ▸ GridSearchCV
  ▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier

```

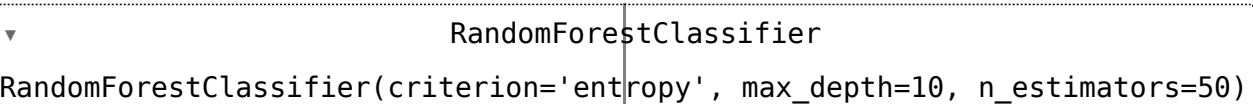
```
In [17]: RFC_cls.best_params_
```

```
Out[17]: {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 100}
```

```
In [18]: cls=RandomForestClassifier(n_estimators=50,criterion='entropy',max_depth=10)
```

```
In [19]: cls.fit(x_train,y_train)
```

```
Out[19]:
```



```

  ▾ RandomForestClassifier
  RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=50)

```

```
In [20]: rfy_pred=cls.predict(x_test)
```



```
In [21]: rfy_pred
```

```
Out[21]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [22]: from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, rfy_pred)
```

```
Out[22]: array([[1540, 157],
               [ 290, 338]])
```

```
In [23]: from sklearn.metrics import accuracy_score
accuracy_score(y_test, rfy_pred)
```

```
Out[23]: 0.807741935483871
```

```
In [24]: 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: ConvergenceWarning: 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/preprocessing.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.org/stable/modules/linear_model.html#logistic-regression)

```
n_iter_i = _check_optimize_result(
```

```
Out[24]: ▾ LogisticRegression
LogisticRegression()
```

```
In [25]: y_pred = classifier.predict(x_test)
```

```
In [26]: from sklearn.metrics import accuracy_score  
accuracy_score(y_test,y_pred)
```

```
Out[26]: 0.8120430107526881
```

```
In [ ]:
```