

```
In [160]: import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

```
In [161]: data=pd.read_csv("/home/placement/Downloads/TelecomCustomerChurn.csv")
```

```
In [162]: data['TotalCharges']=pd.to_numeric(data['TotalCharges'],errors='coerce')
```

```
In [163]: data.describe()
```

Out[163]:

	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 [164]: 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 [165]: list(data)
```

```
Out[165]: ['customerID',  
           'gender',  
           'SeniorCitizen',  
           'Partner',  
           'Dependents',  
           'tenure',  
           'PhoneService',  
           'MultipleLines',  
           'InternetService',  
           'OnlineSecurity',  
           'OnlineBackup',  
           'DeviceProtection',  
           'TechSupport',  
           'StreamingTV',  
           'StreamingMovies',  
           'Contract',  
           'PaperlessBilling',  
           'PaymentMethod',  
           'MonthlyCharges',  
           'TotalCharges',  
           'Churn']
```

```
In [166]: data.isna().sum()
```

```
Out[166]: 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 [167]: data.dtypes
```

```
Out[167]: 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 [168]: data.backup=data.copy()
```

```
In [169]: x=data.drop(['customerID','Churn'],axis=1)
y=data['Churn']
```

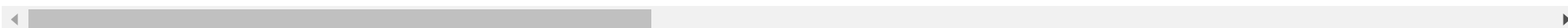
```
In [170]: x=pd.get_dummies(x)
```

```
In [171]: x.head()
```

```
Out[171]:
```

	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 × 11 columns



```
In [172]: x['TotalCharges']=x['TotalCharges'].fillna(x['TotalCharges'].median())
```

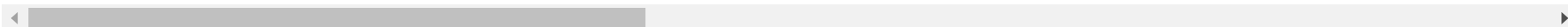
```
In [173]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

```
In [174]: x_test.head(5)
```

```
Out[174]:
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Y
185	0	1	24.80	24.80	1	0	0	1	1	
2715	0	41	25.25	996.45	0	1	1	0	1	
3825	0	52	19.35	1031.70	1	0	0	1	0	
1807	0	1	76.35	76.35	1	0	1	0	1	
132	0	67	50.55	3260.10	0	1	1	0	1	

5 rows × 45 columns



```
In [175]: y_test.head(5)
```

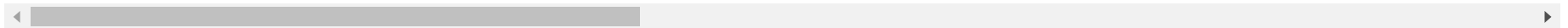
```
Out[175]: 185      Yes
2715     No
3825     No
1807     Yes
132      No
Name: Churn, dtype: object
```

```
In [176]: x_train.head(5)
```

```
Out[176]:
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Y
298	0	40	74.55	3015.75	0	1	0	1	0	
3318	0	10	29.50	255.25	0	1	1	0	1	
5586	0	27	19.15	501.35	1	0	1	0	1	
6654	0	7	86.50	582.50	1	0	0	1	1	
5362	0	65	24.75	1715.10	0	1	0	1	0	

5 rows × 45 columns



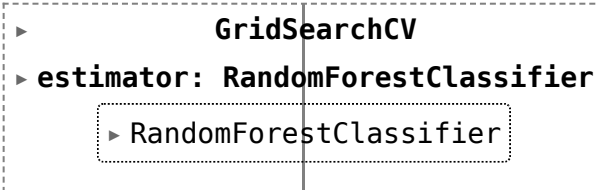
```
In [177]: y_train.head(5)
```

```
Out[177]: 298      No
3318     Yes
5586     No
6654     Yes
5362     No
Name: Churn, dtype: object
```



```
In [178]: 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[178]:



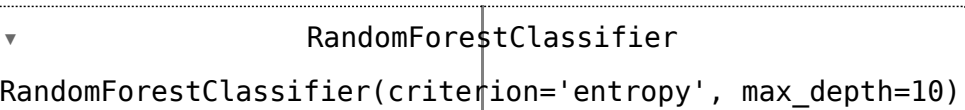
```
In [179]: RFC_cls.best_params_
```

Out[179]: {'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 100}

```
In [185]: cls=RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=10)
```

```
In [186]: cls.fit(x_train,y_train)
```

Out[186]:



```
In [187]: rfy_pred=cls.predict(x_test)
```

```
In [188]: rfy_pred
```

```
Out[188]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [191]: from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,rfy_pred)
```

```
Out[191]: array([[1546, 151],  
                [ 297, 331]])
```

```
In [192]: from sklearn.metrics import accuracy_score  
accuracy_score(y_test,rfy_pred)
```

```
Out[192]: 0.8073118279569892
```

```
In [193]: from sklearn.linear_model import LogisticRegression  
classifier=LogisticRegression()  
classifier.fit(x_train,y_train)
```

```
Out[193]: 

▼ LogisticRegression



LogisticRegression()


```

```
In [194]: y_pred=classifier.predict(x_test)  
y_pred
```

```
Out[194]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [195]: from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,y_pred)
```

```
Out[195]: array([[1526, 171],  
                [ 266, 362]])
```

```
In [196]: from sklearn.metrics import accuracy_score  
accuracy_score(y_test,y_pred)
```

```
Out[196]: 0.8120430107526881
```

```
In [ ]:
```