

```
In [216]: import pandas as pd
```

```
In [217]: data=pd.read_csv("/home/placement/Desktop/csv/TelecomCustomerChurn.csv")
```

```
In [218]: data['TotalCharges']=pd.to_numeric(data['TotalCharges'], errors='coerce')
```

```
In [219]: data.describe()
```

Out[219]:

	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 [220]: data.head()
```

```
Out[220]:
```

	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 [221]: 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 [222]: list(data)
```

```
Out[222]: ['customerID',  
          'gender',  
          'SeniorCitizen',  
          'Partner',  
          'Dependents',  
          'tenure',  
          'PhoneService',  
          'MultipleLines',  
          'InternetService',  
          'OnlineSecurity',  
          'OnlineBackup',  
          'DeviceProtection',  
          'TechSupport',  
          'StreamingTV',  
          'StreamingMovies',  
          'Contract',  
          'PaperlessBilling',  
          'PaymentMethod',  
          'MonthlyCharges',  
          'TotalCharges',  
          'Churn']
```

```
In [223]: data=data.fillna(data.median())
```

```
/tmp/ipykernel_15970/495656529.py:1: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.  
  data=data.fillna(data.median())
```

```
In [224]: data.shape
```

```
Out[224]: (7043, 21)
```

In [225]: data.dtypes

```
Out[225]: 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 [226]: #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 [227]: databackup=data.copy()

```
In [228]: x=data.drop(['customerID','Churn'],axis=1)
y=data['Churn']
```

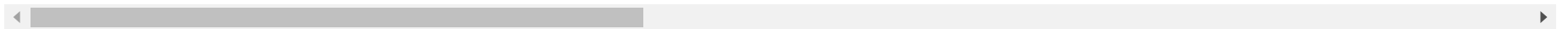
```
In [229]: x=pd.get_dummies(x)
```

```
In [230]: x.head()
```

Out[230]:

	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 [231]: x_train.isna().sum()
```

```
Out[231]: SeniorCitizen      0
          tenure            0
          MonthlyCharges    0
          TotalCharges      0
          gender_Female     0
          gender_Male       0
          Partner_No        0
          Partner_Yes       0
          Dependents_No     0
          Dependents_Yes    0
          PhoneService_No   0
          PhoneService_Yes  0
          MultipleLines_No  0
          MultipleLines_No phone service 0
          MultipleLines_Yes 0
          InternetService_DSL 0
          InternetService_Fiber optic    0
          InternetService_No 0
          OnlineSecurity_No  0
          OnlineSecurity_No internet service 0
          OnlineSecurity_Yes 0
          OnlineBackup_No    0
          OnlineBackup_No internet service 0
          OnlineBackup_Yes   0
          DeviceProtection_No 0
          DeviceProtection_No internet service 0
          DeviceProtection_Yes 0
          TechSupport_No      0
          TechSupport_No internet service 0
          TechSupport_Yes     0
          StreamingTV_No      0
          StreamingTV_No internet service 0
          StreamingTV_Yes     0
          StreamingMovies_No  0
          StreamingMovies_No internet service 0
          StreamingMovies_Yes 0
          Contract_Month-to-month 0
          Contract_One year   0
          Contract_Two year   0
```

```
PaperlessBilling_No      0
PaperlessBilling_Yes     0
PaymentMethod_Bank transfer (automatic)  0
PaymentMethod_Credit card (automatic)    0
PaymentMethod_Electronic check           0
PaymentMethod_Mailed check              0
dtype: int64
```

```
In [232]: data['TotalCharges']=data['TotalCharges'].fillna(data['TotalCharges'].median())
```

```
In [233]: data.isna().sum()
```

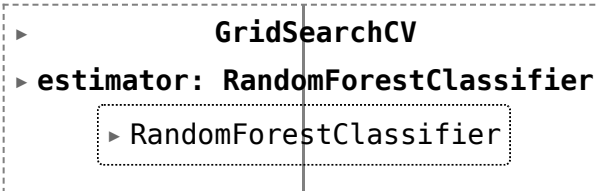
```
Out[233]: customerID      0
gender      0
SeniorCitizen  0
Partner      0
Dependents    0
tenure      0
PhoneService  0
MultipleLines  0
InternetService  0
OnlineSecurity  0
OnlineBackup  0
DeviceProtection  0
TechSupport    0
StreamingTV    0
StreamingMovies  0
Contract      0
PaperlessBilling  0
PaymentMethod  0
MonthlyCharges  0
TotalCharges    0
Churn          0
dtype: int64
```

```
In [234]: 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 [235]: 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[235]:
```



```

  ▸ GridSearchCV
  ▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier

```

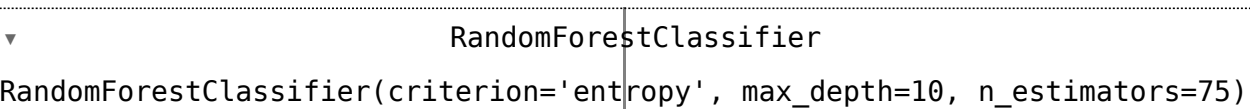
```
In [236]: RFC_cls.best_params_
```

```
Out[236]: {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 100}
```

```
In [237]: cls=RandomForestClassifier(n_estimators=75,criterion='entropy',max_depth=10)
```

```
In [238]: cls.fit(x_train,y_train)
```

```
Out[238]:
```



```

  ▾ RandomForestClassifier
  RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=75)

```

```
In [240]: rfy_pred=cls.predict(x_test)
rfy_pred
```

```
Out[240]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [241]: from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,rfy_pred)
```

```
Out[241]: array([[1550,  147],  
                [ 307,  321]])
```

```
In [242]: from sklearn.metrics import accuracy_score  
accuracy_score(y_test,rfy_pred)
```

```
Out[242]: 0.8047311827956989
```

```
In [243]: 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[243]: ▾ LogisticRegression  
LogisticRegression()
```

```
In [244]: y_pred=classifier.predict(x_test)  
y_pred
```

```
Out[244]: array(['Yes', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

```
In [245]: from sklearn.metrics import confusion_matrix  
          confusion_matrix(y_test,y_pred)
```

```
Out[245]: array([[1526,  171],  
                [ 266,  362]])
```

```
In [246]: from sklearn.metrics import accuracy_score  
          accuracy_score(y_test,y_pred)
```

```
Out[246]: 0.8120430107526881
```

```
In [ ]:
```