


```
In [121]: import pandas as pd    #Import libraries
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv("/home/placement/Downloads/TelecomCustomerChurn.csv") #reading datafile
print(data)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
0	7590-VHVEG	Female	0	Yes	No	1	
1	5575-GNVDE	Male	0	No	No	34	
2	3668-QPYBK	Male	0	No	No	2	
3	7795-CF0CW	Male	0	No	No	45	
4	9237-HQITU	Female	0	No	No	2	
...	
7038	6840-RESVB	Male	0	Yes	Yes	24	
7039	2234-XADUH	Female	0	Yes	Yes	72	
7040	4801-JZAZL	Female	0	Yes	Yes	11	
7041	8361-LTMKD	Male	1	Yes	No	4	
7042	3186-AJIEK	Male	0	No	No	66	

	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	\
0	No	No phone service	DSL	No	...	
1	Yes	No	DSL	Yes	...	
2	Yes	No	DSL	Yes	...	
3	No	No phone service	DSL	Yes	...	
4	Yes	No	Fiber optic	No	...	
...	
7038	Yes	Yes	DSL	Yes	...	
7039	Yes	Yes	Fiber optic	No	...	
7040	No	No phone service	DSL	Yes	...	
7041	Yes	Yes	Fiber optic	No	...	
7042	Yes	No	Fiber optic	Yes	...	

	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	\
0	No	No	No	No	Month-to-month	
1	Yes	No	No	No	One year	
2	No	No	No	No	Month-to-month	
3	Yes	Yes	No	No	One year	
4	No	No	No	No	Month-to-month	
...	
7038	Yes	Yes	Yes	Yes	One year	
7039	Yes	No	Yes	Yes	One year	

7040	No	No	No	No	Month-to-month
7041	No	No	No	No	Month-to-month
7042	Yes	Yes	Yes	Yes	Two year

	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	\
0	Yes	Electronic check	29.85	29.85	
1	No	Mailed check	56.95	1889.5	
2	Yes	Mailed check	53.85	108.15	
3	No	Bank transfer (automatic)	42.30	1840.75	
4	Yes	Electronic check	70.70	151.65	
...	
7038	Yes	Mailed check	84.80	1990.5	
7039	Yes	Credit card (automatic)	103.20	7362.9	
7040	Yes	Electronic check	29.60	346.45	
7041	Yes	Mailed check	74.40	306.6	
7042	Yes	Bank transfer (automatic)	105.65	6844.5	

	Churn
0	No
1	No
2	Yes
3	No
4	Yes
...	...
7038	No
7039	No
7040	No
7041	Yes
7042	No

[7043 rows x 21 columns]

Exploratory data analysis

In [122]: `data.head()`

Out[122]:

	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 [123]: `data.columns`

Out[123]: Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
dtype='object')

In [124]: 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 [125]: data['TotalCharges']=pd.to_numeric(data['TotalCharges'],errors='coerce') *#convert the object type into float*

In [126]: 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 [127]: data.shape

Out[127]: (7043, 21)

```
In [128]: data.isna().sum()
```

```
Out[128]: 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 [129]: data.describe()
```

```
Out[129]:
```

	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 [130]: data1=data.fillna(data.median())
```



```
In [131]: data1.isna().sum()
```

```
Out[131]: 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 [132]: data2=data1.drop(columns='customerID')
```

In [133]: data2

Out[133]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	
...	
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes	
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No	
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	No	
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No	

7043 rows × 20 columns



In [134]: data2['Churn']=data2['Churn'].map({'Yes':1, 'No':0})

In [135]: data2

Out[135]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtect
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	
...
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes	
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No	
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	No	
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No	

7043 rows × 20 columns



```
In [136]: y=data2['Churn']
          x=data2.drop(columns='Churn')
```

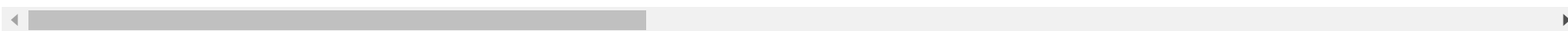
```
In [137]: x=pd.get_dummies(x)
```

In [138]: x

Out[138]:

	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	
1	0	34	56.95	1889.50	0	1	1	0	1	
2	0	2	53.85	108.15	0	1	1	0	1	
3	0	45	42.30	1840.75	0	1	1	0	1	
4	0	2	70.70	151.65	1	0	1	0	1	
...
7038	0	24	84.80	1990.50	0	1	0	1	0	
7039	0	72	103.20	7362.90	1	0	0	1	0	
7040	0	11	29.60	346.45	1	0	0	1	0	
7041	1	4	74.40	306.60	0	1	0	1	1	
7042	0	66	105.65	6844.50	0	1	1	0	1	

7043 rows × 45 columns




splitting data into training and testing sets

```
In [227]: 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 [228]: 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[228]:
```



```
└─ GridSearchCV
  └─ estimator: RandomForestClassifier
    └─ RandomForestClassifier
```

```
In [229]: RFC_cls.best_params_
```

```
Out[229]: {'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 125}
```

```
In [239]: y_pred_rfc=RFC_cls .predict(x_test) #multiply x_test with classifier
```

```
In [240]: y_pred_rfc
```

```
Out[240]: array([1, 0, 0, ..., 1, 0, 0])
```

```
In [241]: from sklearn.metrics import confusion_matrix #confusion matrix
confusion_matrix(y_test,y_pred_rfc)
```

```
Out[241]: array([[1547, 150],
                 [ 303, 325]])
```

```
In [242]: from sklearn.metrics import accuracy_score #accuracy of test data and predicted data
accuracy_score(y_test,y_pred_rfc)
```

```
Out[242]: 0.8051612903225807
```

Logistic Regression

```
In [235]: from sklearn.linear_model import LogisticRegression #making model using 64%trainingg data
classifier=LogisticRegression()
classifier.fit(x_train,y_train)
```

```
Out[235]: ▼ LogisticRegression
LogisticRegression()
```

```
In [243]: y_pred_log=classifier.predict(x_test) #multiply x_test with classifier
```

```
In [244]: y_pred_log
```

```
Out[244]: array([1, 0, 0, ..., 1, 0, 0])
```

```
In [245]: from sklearn.metrics import confusion_matrix #confusion matrix
confusion_matrix(y_test,y_pred_log)
```

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

```
In [246]: from sklearn.metrics import accuracy_score #accuracy of test data and predicted data
accuracy_score(y_test,y_pred_log)
```

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

```
In [ ]:
```