In [164]:
```
import pandas as pd
```

In [165]:
```
data=pd.read_csv("/home/placement/Desktop/csv/TelecomCustomerChurn.csv")
```

In [166]:
```
data['TotalCharges']=pd.to_numeric(data['TotalCharges'], errors='coerce')
```

In [167]:
```
data.describe()
```

Out[167]:

	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 [168]:
```
data.head(5)
```

Out[168]:

	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 [169]:
```
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 [170]:
```
data.isna().sum()
```

Out[170]:
```
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 [171]:
```
list(data)
```

Out[171]:
```
['customerID',
 'gender',
 'SeniorCitizen',
 'Partner',
 'Dependents',
 'tenure',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'MonthlyCharges',
 'TotalCharges',
 'Churn']
```

In [172]:
```
data1=data.fillna(data.median())
```

```
/tmp/ipykernel_5249/3060338577.py:1: FutureWarning: The default value of numeric_only in DataFrame.median i
s 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.
  data1=data.fillna(data.median())
```

In [173]:
```
data.shape
```

Out[173]:
```
(7043, 21)
```

In [174]:
```
data.isna().sum()
```

Out[174]:
```
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 [175]:
```
data2=data1.drop(['customerID','SeniorCitizen','PhoneService','OnlineBackup','Partner','Dependents','OnlineS
data2
```

Out[175]:

	gender	tenure	MultipleLines	InternetService	TechSupport	Contract	MonthlyCharges	TotalCharges	Churn
0	Female	1	No phone service	DSL	No	Month-to-month	29.85	29.85	No
1	Male	34	No	DSL	No	One year	56.95	1889.50	No
2	Male	2	No	DSL	No	Month-to-month	53.85	108.15	Yes
3	Male	45	No phone service	DSL	Yes	One year	42.30	1840.75	No
4	Female	2	No	Fiber optic	No	Month-to-month	70.70	151.65	Yes
...	...	...	...	...	...	...	...	...	...
7038	Male	24	Yes	DSL	Yes	One year	84.80	1990.50	No
7039	Female	72	Yes	Fiber optic	No	One year	103.20	7362.90	No
7040	Female	11	No phone service	DSL	No	Month-to-month	29.60	346.45	No
7041	Male	4	Yes	Fiber optic	No	Month-to-month	74.40	306.60	Yes
7042	Male	66	No	Fiber optic	Yes	Two year	105.65	6844.50	No

7043 rows × 9 columns

In [176]:
```
data2.shape
```

Out[176]: (7043, 9)

In [178]:
```
data2['Churn']=data2['Churn'].map({'Yes':1,'No':0})
data2
```

Out[178]:

	gender	tenure	MultipleLines	InternetService	TechSupport	Contract	MonthlyCharges	TotalCharges	Churn
0	Female	1	No phone service	DSL	No	Month-to-month	29.85	29.85	0
1	Male	34	No	DSL	No	One year	56.95	1889.50	0
2	Male	2	No	DSL	No	Month-to-month	53.85	108.15	1
3	Male	45	No phone service	DSL	Yes	One year	42.30	1840.75	0
4	Female	2	No	Fiber optic	No	Month-to-month	70.70	151.65	1
...	...	...	...	...	...	...	...	...	...
7038	Male	24	Yes	DSL	Yes	One year	84.80	1990.50	0
7039	Female	72	Yes	Fiber optic	No	One year	103.20	7362.90	0
7040	Female	11	No phone service	DSL	No	Month-to-month	29.60	346.45	0
7041	Male	4	Yes	Fiber optic	No	Month-to-month	74.40	306.60	1
7042	Male	66	No	Fiber optic	Yes	Two year	105.65	6844.50	0

7043 rows × 9 columns

In [179]:
```
data3=pd.get_dummies(data2)
```

In [180]:
```
data3
```

Out[180]:

ternetService_DSL	InternetService_Fiber optic	InternetService_No	TechSupport_No	TechSupport_No internet service	TechSupport_Yes	Contract_Month-to-month	Contract_One year	Co
1	0	0	1	0	0	1	0	
1	0	0	1	0	0	0	1	
1	0	0	1	0	0	1	0	
1	0	0	0	0	1	0	1	
0	1	0	1	0	0	1	0	
...	...	...	...	...	...	...	...	
1	0	0	0	0	1	0	1	
0	1	0	1	0	0	0	1	
1	0	0	1	0	0	1	0	
0	1	0	1	0	0	1	0	
0	1	0	0	0	1	0	0	

In [181]:
```
data3.shape
```

Out[181]: (7043, 18)

In [182]:
```
y=data3['Churn']
x=data3.drop('Churn',axis=1)
```

In [183]:
```
y
```

Out[183]:
```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7043, dtype: int64
```

In [184]:
```
x
```

Out[184]:

	tenure	MonthlyCharges	TotalCharges	gender_Female	gender_Male	MultipleLines_No	MultipleLines_No phone service	MultipleLines_Yes	InternetService_
0	1	29.85	29.85	1	0	0	1	0	
1	34	56.95	1889.50	0	1	1	0	0	
2	2	53.85	108.15	0	1	1	0	0	
3	45	42.30	1840.75	0	1	0	1	0	
4	2	70.70	151.65	1	0	1	0	0	
...	...	...	...	...	...	...	...	...	
7038	24	84.80	1990.50	0	1	0	0	1	
7039	72	103.20	7362.90	1	0	0	0	1	
7040	11	29.60	346.45	1	0	0	1	0	
7041	4	74.40	306.60	0	1	0	0	1	
7042	66	105.65	6844.50	0	1	1	0	0	

7043 rows × 17 columns

In [185]:
```
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 [186]:
```
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: ConvergenceWa
rning: 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/pre
processing.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.or
g/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[186]:
```
▾ LogisticRegression

LogisticRegression()
```

In [188]:
```
y_pred=classifier.predict(x_test)
```

In [189]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[189]:
```
array([[1519,  178],
       [ 273,  355]])
```

In [190]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[190]:
```
0.8060215053763441
```

In [ ]:
```

```