In [262]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [263]:
```
df = pd.read_csv("C:/Users/YASH/Downloads/churn.csv")
```

In [264]:
```
df.head()
```

Out[264]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtectior
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No

5 rows × 21 columns

In [265]:
```
df.shape
```

Out[265]:
```
(7043, 21)
```

In [266]:
```
1 df.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 [267]:
```
1 df.columns
```

Out[267]:
```
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 [268]:
```
1 df.describe()
```

Out[268]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [269]:
```
1 df.isnull().sum()
```

Out[269]:
```
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 [270]:
```
1 df.duplicated()
```

Out[270]:
```
0       False
1       False
2       False
3       False
4       False
        ...
7038    False
7039    False
7040    False
7041    False
7042    False
Length: 7043, dtype: bool
```

- Here we are

In [271]:
```
1 df.dtypes
```

Out[271]:
```
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
```

As we can see from the above data that the TOTAL CHARGES Column is into object datatype which should be actually numeric.

Hence performed pandas numeric to float convertion

In [272]:
```
1 df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

- Now we can see that it has been converted to float datatype

In [273]:
```
1 df.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 [274]:
```
1 df['SeniorCitizen'].value_counts()
```

Out[274]:
```
0    5901
1    1142
Name: SeniorCitizen, dtype: int64
```

- Here we will drop customerID column cause its of no use.

In [275]:
```
df.drop(columns = 'customerID',inplace = True,axis = 1)
```

In [276]:
```
df.info()
```

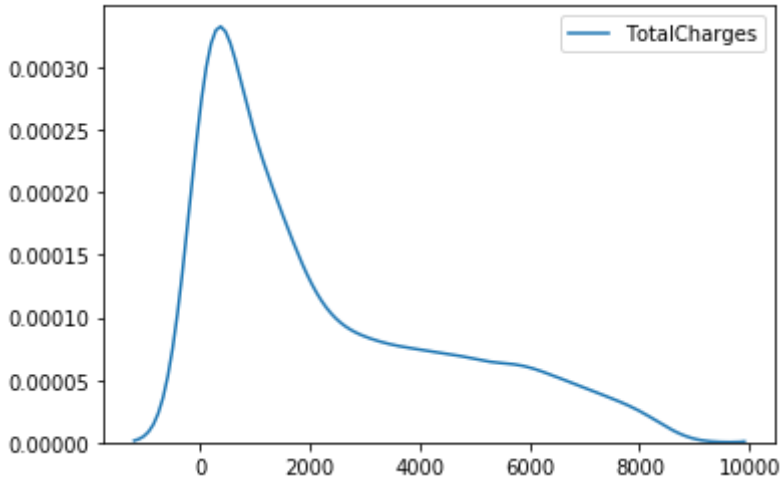
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7043 non-null   object 
 1   SeniorCitizen     7043 non-null   int64  
 2   Partner           7043 non-null   object 
 3   Dependents        7043 non-null   object 
 4   tenure            7043 non-null   int64  
 5   PhoneService      7043 non-null   object 
 6   MultipleLines     7043 non-null   object 
 7   InternetService   7043 non-null   object 
 8   OnlineSecurity    7043 non-null   object 
 9   OnlineBackup      7043 non-null   object 
 10  DeviceProtection  7043 non-null   object 
 11  TechSupport       7043 non-null   object 
 12  StreamingTV       7043 non-null   object 
 13  StreamingMovies   7043 non-null   object 
 14  Contract          7043 non-null   object 
 15  PaperlessBilling  7043 non-null   object 
 16  PaymentMethod     7043 non-null   object 
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7043 non-null   object 
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

In [192]:
```
df['TotalCharges'].isnull().sum()
df['TotalCharges'].describe()
```

Out[192]:
```
count    7032.000000
mean     2283.300441
std      2266.771362
min        18.800000
25%       401.450000
50%      1397.475000
75%      3794.737500
max      8684.800000
Name: TotalCharges, dtype: float64
```

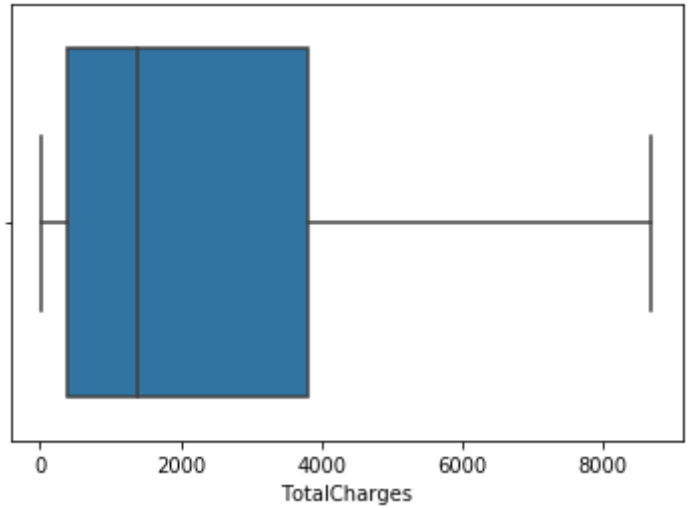
In [277]:
```
sns.kdeplot(df['TotalCharges'])
### The representation shows that the values are positively skewed and hence can apply median()
```

Out[277]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x214dc6d0808>
```



In [278]:
```
1 sns.boxplot(df['TotalCharges'])
```

Out[278]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x214dc809cc8>
```



In [279]:
```
1 df['TotalCharges'].describe()
```

Out[279]:
```
count    7032.000000
mean     2283.300441
std      2266.771362
min        18.800000
25%       401.450000
50%      1397.475000
75%      3794.737500
max      8684.800000
Name: TotalCharges, dtype: float64
```

In [280]:
```
df['TotalCharges'].fillna(1397.475,inplace = True)
```

In [281]:
```
df['TotalCharges']
```

Out[281]:
```
0         29.85
1       1889.50
2        108.15
3       1840.75
4        151.65
          ...
7038    1990.50
7039    7362.90
7040     346.45
7041     306.60
7042    6844.50
Name: TotalCharges, Length: 7043, dtype: float64
```

In [282]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7043 non-null   object 
 1   SeniorCitizen     7043 non-null   int64  
 2   Partner           7043 non-null   object 
 3   Dependents        7043 non-null   object 
 4   tenure            7043 non-null   int64  
 5   PhoneService      7043 non-null   object 
 6   MultipleLines     7043 non-null   object 
 7   InternetService   7043 non-null   object 
 8   OnlineSecurity    7043 non-null   object 
 9   OnlineBackup      7043 non-null   object 
 10  DeviceProtection  7043 non-null   object 
 11  TechSupport       7043 non-null   object 
 12  StreamingTV       7043 non-null   object 
 13  StreamingMovies   7043 non-null   object 
 14  Contract          7043 non-null   object 
 15  PaperlessBilling  7043 non-null   object 
 16  PaymentMethod     7043 non-null   object 
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   float64
 19  Churn             7043 non-null   object 
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

- From the above we now understood that their are only 4 numerical columns.
- And 16 are categorical columns

- We found a target variable in the dataset called "CHURN"
- as its discrete(nominal) variable we can consider this as a CLASSIFICATION TASK

In [283]:
```
1 df['SeniorCitizen'].value_counts()   #OHE
```

Out[283]:
```
0    5901
1    1142
Name: SeniorCitizen, dtype: int64
```

In [284]:
```
1 df['Partner'].value_counts()   #OHE
```

Out[284]:
```
No     3641
Yes    3402
Name: Partner, dtype: int64
```

In [285]:
```
1 df['Dependents'].value_counts()   #OHE
```

Out[285]:
```
No     4933
Yes    2110
Name: Dependents, dtype: int64
```

In [286]:
```
1 df['PhoneService'].value_counts() #OHE
```

Out[286]:
```
Yes    6361
No      682
Name: PhoneService, dtype: int64
```

In [287]:
```
1 df['MultipleLines'].value_counts() #OHE
```

Out[287]:
```
No                  3390
Yes                 2971
No phone service     682
Name: MultipleLines, dtype: int64
```

In [288]:
```
1 df['InternetService'].value_counts()   #OHE
```

Out[288]:
```
Fiber optic    3096
DSL            2421
No             1526
Name: InternetService, dtype: int64
```

In [289]:
```
1 df['OnlineSecurity'].unique()  #OHE
```

Out[289]:
```
array(['No', 'Yes', 'No internet service'], dtype=object)
```

In [290]:
```
1 df['OnlineBackup'].unique()   #OHE
```

Out[290]:
```
array(['Yes', 'No', 'No internet service'], dtype=object)
```

In [291]:
```
1 df['DeviceProtection'].unique()  #OHE
```

Out[291]:
```
array(['No', 'Yes', 'No internet service'], dtype=object)
```

In [292]:
```
1 df['TechSupport'].unique()   # OHE
```

Out[292]:
```
array(['No', 'Yes', 'No internet service'], dtype=object)
```

In [293]:
```
1 df['StreamingTV'].unique()   #OHE
```

Out[293]:
```
array(['No', 'Yes', 'No internet service'], dtype=object)
```

In [294]:
```
1 df['StreamingMovies'].unique()   #OHE
```

Out[294]:
```
array(['No', 'Yes', 'No internet service'], dtype=object)
```

In [295]:
```
1 df['PaymentMethod'].unique()    #OHE
```

Out[295]:
```
array(['Electronic check', 'Mailed check', 'Bank transfer (automatic)',
       'Credit card (automatic)'], dtype=object)
```

In [296]:
```
1 df['PaperlessBilling'].unique()    # one hot encoding
```

Out[296]:
```
array(['Yes', 'No'], dtype=object)
```

In [297]:
```
df['Contract'].unique()    #LE
```

Out[297]:
```
array(['Month-to-month', 'One year', 'Two year'], dtype=object)
```

a. Identify the Target Variable and Splitting the Data into Train and Test

In [298]:
```
# Identifying the inputs (X) and output (y)

y = df['Churn']
X = df[['gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges']]
```

In [299]:
```
# split into train and test

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y, train_size = 0.75, random_state = 100)

```

In [300]:
```
1 X_train.head()
```

Out[300]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtecti
6630	Male	1	No	No	16	Yes	No	Fiber optic	No	No	
7013	Female	0	No	No	40	Yes	Yes	Fiber optic	No	Yes	Y
2224	Male	1	Yes	No	17	Yes	No	Fiber optic	No	Yes	
6580	Female	0	Yes	Yes	49	Yes	No	DSL	No	Yes	Y
1501	Male	0	No	No	13	Yes	No	No	No internet service	No internet service	No inter serv

In [301]:
```
1 print(X_train.shape, y_train.shape)
2
3 print(X_test.shape, y_test.shape)
```

```
(5282, 19) (5282,)
(1761, 19) (1761,)
```

b. Separating Categorical and Numerical Columns:

In [302]:
```
1 X_train.dtypes
```

Out[302]:
```
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
dtype: object
```

In [303]:
```
X_train_cat = X_train.select_dtypes(include=['object'])
X_train_cat.head()
```

Out[303]:

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Stre
6630	Male	No	No	Yes	No	Fiber optic	No	No	No	No	
7013	Female	No	No	Yes	Yes	Fiber optic	No	Yes	Yes	No	
2224	Male	Yes	No	Yes	No	Fiber optic	No	Yes	No	No	
6580	Female	Yes	Yes	Yes	No	DSL	No	Yes	Yes	No	
1501	Male	No	No	Yes	No	No	No internet service	No internet service	No internet service	No internet service	N

In [304]:
```
X_train_num = X_train[['tenure','MonthlyCharges','TotalCharges']]
X_train_num.head()
```

Out[304]:

	tenure	MonthlyCharges	TotalCharges
6630	16	78.75	1218.25
7013	40	93.40	3756.40
2224	17	76.45	1233.40
6580	49	78.00	3824.20
1501	13	19.95	243.65

c. Applying OneHotEncoding on Categorical Columns

In [305]:
```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(drop = 'first',sparse = False)

X_train_cat_ohe = pd.DataFrame(encoder.fit_transform(X_train_cat),
                               columns = encoder.get_feature_names(X_train_cat.columns),
                               index = X_train_cat.index)
```

In [306]:
```
X_train_cat_ohe.head()
```

Out[306]:

	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes	InternetService_Fiber optic	InternetService_No
6630	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
7013	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0
2224	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0
6580	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
1501	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

5 rows × 26 columns

In [307]:
```
X_train_cat_ohe.drop(columns = ['Contract_One year','Contract_Two year'],inplace= True,axis = 1)
```

In [308]:
```
1 X_train_cat_ohe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5282 entries, 6630 to 5640
Data columns (total 24 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   gender_Male                             5282 non-null   float64
 1   Partner_Yes                             5282 non-null   float64
 2   Dependents_Yes                          5282 non-null   float64
 3   PhoneService_Yes                        5282 non-null   float64
 4   MultipleLines_No phone service          5282 non-null   float64
 5   MultipleLines_Yes                       5282 non-null   float64
 6   InternetService_Fiber optic             5282 non-null   float64
 7   InternetService_No                      5282 non-null   float64
 8   OnlineSecurity_No internet service      5282 non-null   float64
 9   OnlineSecurity_Yes                      5282 non-null   float64
 10  OnlineBackup_No internet service        5282 non-null   float64
 11  OnlineBackup_Yes                        5282 non-null   float64
 12  DeviceProtection_No internet service    5282 non-null   float64
 13  DeviceProtection_Yes                    5282 non-null   float64
 14  TechSupport_No internet service         5282 non-null   float64
 15  TechSupport_Yes                         5282 non-null   float64
 16  StreamingTV_No internet service         5282 non-null   float64
 17  StreamingTV_Yes                         5282 non-null   float64
 18  StreamingMovies_No internet service     5282 non-null   float64
 19  StreamingMovies_Yes                     5282 non-null   float64
 20  PaperlessBilling_Yes                    5282 non-null   float64
 21  PaymentMethod_Credit card (automatic)   5282 non-null   float64
 22  PaymentMethod_Electronic check          5282 non-null   float64
 23  PaymentMethod_Mailed check              5282 non-null   float64
dtypes: float64(24)
memory usage: 1.0 MB
```

Encoding Ordinal columns

- As we can see from the data only contract column has some order and hence its ordinal column hence lets perform Label Encoding.

In [309]:
```
X_train_cat_le = pd.DataFrame(index = X_train_cat.index)
X_train_cat_le.head()
```

Out[309]:

6630
7013
2224
6580
1501

In [310]:
```
X_train_cat.Contract.unique()
```

Out[310]:
```
array(['Month-to-month', 'One year', 'Two year'], dtype=object)
```

In [311]:
```
cut_encoder = {'Month-to-month' : 1, 'One year' : 2,'Two year' : 3}
X_train_cat_le['Contract'] = X_train_cat['Contract'].apply(lambda x : cut_encoder[x])
X_train_cat_le.head()
```

Out[311]:

	Contract
6630	1
7013	1
2224	1
6580	2
1501	3

- Scaling the Numerical Features

In [312]:
```
1 X_train_num.head()
```

Out[312]:

	tenure	MonthlyCharges	TotalCharges
6630	16	78.75	1218.25
7013	40	93.40	3756.40
2224	17	76.45	1233.40
6580	49	78.00	3824.20
1501	13	19.95	243.65

In [313]:
```
1 #Scaling the Numerical Features
2
3 from sklearn.preprocessing import StandardScaler
4 scaler = StandardScaler()
5
6 X_train_num_rescaled = pd.DataFrame(scaler.fit_transform(X_train_num),
7                                      columns = X_train_num.columns,
8                                      index = X_train_num.index)
```

In [314]:
```
1 X_train_num_rescaled.head()
```

Out[314]:

	tenure	MonthlyCharges	TotalCharges
6630	-0.669552	0.471732	-0.471643
7013	0.305416	0.957819	0.649818
2224	-0.628928	0.395418	-0.464949
6580	0.671028	0.446847	0.679775
1501	-0.791422	-1.479251	-0.902262

In [315]:
```
1 X_train_transformed = pd.concat([X_train_num_rescaled,X_train_cat_ohe,X_train_cat_le,X_train['SeniorCitizen']],axis
```

In [316]:
```
1 X_train_transformed.head()
```

Out[316]:

	tenure	MonthlyCharges	TotalCharges	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes
6630	-0.669552	0.471732	-0.471643	1.0	0.0	0.0	1.0	0.0	0.0
7013	0.305416	0.957819	0.649818	0.0	0.0	0.0	1.0	0.0	1.0
2224	-0.628928	0.395418	-0.464949	1.0	1.0	0.0	1.0	0.0	0.0
6580	0.671028	0.446847	0.679775	0.0	1.0	1.0	1.0	0.0	0.0
1501	-0.791422	-1.479251	-0.902262	1.0	0.0	0.0	1.0	0.0	0.0

5 rows × 29 columns

In [317]:
```
1 X_train_transformed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5282 entries, 6630 to 5640
Data columns (total 29 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   tenure                                  5282 non-null   float64
 1   MonthlyCharges                          5282 non-null   float64
 2   TotalCharges                            5282 non-null   float64
 3   gender_Male                             5282 non-null   float64
 4   Partner_Yes                             5282 non-null   float64
 5   Dependents_Yes                          5282 non-null   float64
 6   PhoneService_Yes                        5282 non-null   float64
 7   MultipleLines_No phone service          5282 non-null   float64
 8   MultipleLines_Yes                       5282 non-null   float64
 9   InternetService_Fiber optic             5282 non-null   float64
 10  InternetService_No                      5282 non-null   float64
 11  OnlineSecurity_No internet service      5282 non-null   float64
 12  OnlineSecurity_Yes                      5282 non-null   float64
 13  OnlineBackup_No internet service        5282 non-null   float64
 14  OnlineBackup_Yes                        5282 non-null   float64
 15  DeviceProtection_No internet service    5282 non-null   float64
 16  DeviceProtection_Yes                    5282 non-null   float64
 17  TechSupport_No internet service         5282 non-null   float64
 18  TechSupport_Yes                         5282 non-null   float64
 19  StreamingTV_No internet service         5282 non-null   float64
 20  StreamingTV_Yes                         5282 non-null   float64
 21  StreamingMovies_No internet service     5282 non-null   float64
 22  StreamingMovies_Yes                     5282 non-null   float64
 23  PaperlessBilling_Yes                    5282 non-null   float64
 24  PaymentMethod_Credit card (automatic)   5282 non-null   float64
 25  PaymentMethod_Electronic check          5282 non-null   float64
 26  PaymentMethod_Mailed check              5282 non-null   float64
 27  Contract                                5282 non-null   int64  
 28  SeniorCitizen                           5282 non-null   int64  
dtypes: float64(27), int64(2)
memory usage: 1.2 MB
```

- Now we can see that all the values of X_train have been converted to numerical values we got total 29 columns.

Now preparing the test data

In [318]:
```
1 X_test.head()
```

Out[318]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtecti
4880	Male	0	Yes	No	50	Yes	No	No	No internet service	No internet service	No inter serv
1541	Male	0	No	No	72	Yes	No	No	No internet service	No internet service	No inter serv
1289	Male	0	No	No	63	Yes	Yes	DSL	Yes	Yes	Y
5745	Female	0	Yes	Yes	61	Yes	Yes	No	No internet service	No internet service	No inter serv
4873	Female	0	No	No	7	Yes	No	No	No internet service	No internet service	No inter serv

In [319]:
```
1 X_test.dtypes
```

Out[319]:
```
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
dtype: object
```

In [320]:
```
X_test_cat = X_test.select_dtypes(include=['object'])
X_test_cat.head()
```

Out[320]:

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Stre
4880	Male	Yes	No	Yes	No	No	No internet service	No internet service	No internet service	No internet service	N
1541	Male	No	No	Yes	No	No	No internet service	No internet service	No internet service	No internet service	N
1289	Male	No	No	Yes	Yes	DSL	Yes	Yes	Yes	Yes	
5745	Female	Yes	Yes	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	N
4873	Female	No	No	Yes	No	No	No internet service	No internet service	No internet service	No internet service	N

In [321]:
```
X_test_num = X_test[['tenure','MonthlyCharges','TotalCharges']]
X_test_num.head()
```

Out[321]:

	tenure	MonthlyCharges	TotalCharges
4880	50	20.55	1067.65
1541	72	19.85	1434.10
1289	63	68.80	4111.35
5745	61	24.20	1445.20
4873	7	19.30	144.95

Applying one Hot Encoding to the test data

In [322]:
```
X_test_cat_ohe = pd.DataFrame(encoder.transform(X_test_cat),
                              columns = encoder.get_feature_names(X_test_cat.columns),
                              index = X_test_cat.index)
X_test_cat_ohe.head()
```

Out[322]:

	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes	InternetService_Fiber optic	InternetService_No
4880	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0
1541	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
1289	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
5745	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0
4873	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

5 rows × 26 columns

In [323]:
```
X_test_cat_ohe.drop(columns = ['Contract_One year','Contract_Two year'],axis = 1,inplace = True)
```

In [324]:
```
X_test_cat_ohe.head()
```

Out[324]:

	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes	InternetService_Fiber optic	InternetService_No
4880	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0
1541	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
1289	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
5745	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0
4873	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

5 rows × 24 columns

Standardizing the Numerical values in the test data

In [325]:
```
X_test_num_rescaled = pd.DataFrame(scaler.transform(X_test_num),
                                   columns = X_test_num.columns,
                                   index = X_test_num.index)
```

In [326]:
```
X_test_num_rescaled.head()
```

Out[326]:

	tenure	MonthlyCharges	TotalCharges
4880	0.711652	-1.459343	-0.538184
1541	1.605372	-1.482569	-0.376271
1289	1.239759	0.141591	0.806650
5745	1.158512	-1.338236	-0.371367
4873	-1.035164	-1.500818	-0.945872

- Label Encoding on Contract columns for test data

In [327]:
```
X_test_cat_le = pd.DataFrame(index = X_test_cat.index)
X_test_cat_le.head()
```

Out[327]:

4880
1541
1289
5745
4873

In [328]:
```
cut_encoder = {'Month-to-month' : 1, 'One year' : 2,'Two year' : 3}
X_test_cat_le['Contract'] = X_test_cat['Contract'].apply(lambda x : cut_encoder[x])
X_test_cat_le.head()
```

Out[328]:

	Contract
4880	3
1541	3
1289	2
5745	3
4873	1

In [329]:
```
X_test_transformed = pd.concat([X_test_num_rescaled,X_test_cat_ohe,X_test_cat_le,X_test['SeniorCitizen']],axis = 1)
```

In [330]:
```
X_test_transformed.columns
```

Out[330]:
```
Index(['tenure', 'MonthlyCharges', 'TotalCharges', 'gender_Male',
       'Partner_Yes', 'Dependents_Yes', 'PhoneService_Yes',
       'MultipleLines_No phone service', 'MultipleLines_Yes',
       'InternetService_Fiber optic', 'InternetService_No',
       'OnlineSecurity_No internet service', 'OnlineSecurity_Yes',
       'OnlineBackup_No internet service', 'OnlineBackup_Yes',
       'DeviceProtection_No internet service', 'DeviceProtection_Yes',
       'TechSupport_No internet service', 'TechSupport_Yes',
       'StreamingTV_No internet service', 'StreamingTV_Yes',
       'StreamingMovies_No internet service', 'StreamingMovies_Yes',
       'PaperlessBilling_Yes', 'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check',
       'Contract', 'SeniorCitizen'],
      dtype='object')
```

In [331]:
```
1 X_test_transformed.head()
```

Out[331]:

	tenure	MonthlyCharges	TotalCharges	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_No phone service	MultipleLines_Yes
4880	0.711652	-1.459343	-0.538184	1.0	1.0	0.0	1.0	0.0	0.0
1541	1.605372	-1.482569	-0.376271	1.0	0.0	0.0	1.0	0.0	0.0
1289	1.239759	0.141591	0.806650	1.0	0.0	0.0	1.0	0.0	1.0
5745	1.158512	-1.338236	-0.371367	0.0	1.0	1.0	1.0	0.0	1.0
4873	-1.035164	-1.500818	-0.945872	0.0	0.0	0.0	1.0	0.0	0.0

5 rows × 29 columns

LOGISTIC REGRESSION

In [332]:
```
1 from sklearn.linear_model import LogisticRegression
2 classifier = LogisticRegression()
3 classifier.fit(X_train_transformed, y_train)
```

Out[332]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [333]:
```
1 y_test_pred = classifier.predict(X_test_transformed)
```

In [336]:
```
temp_df = pd.DataFrame({'Actual' : y_test,'Predicted' :y_test_pred })
temp_df.head()
```

Out[336]:

	Actual	Predicted
4880	No	No
1541	No	No
1289	No	No
5745	No	No
4873	No	No

In [338]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_test_pred)
```

Out[338]: 0.7842135150482681

- Accuracy Rate = 78% when tried with Logistic Regression

KNN CLASSIFIER

In [344]:
```
from sklearn.neighbors import KNeighborsClassifier
neighbour = KNeighborsClassifier()
neighbour.fit(X_train_transformed, y_train)
```

Out[344]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [345]:
```
y_test_pred = neighbour.predict(X_test_transformed)
```

In [346]:
```
temp_df = pd.DataFrame({'Actual' : y_test,'Predicted' :y_test_pred })
temp_df.head()
```

Out[346]:

	Actual	Predicted
4880	No	No
1541	No	No
1289	No	No
5745	No	No
4873	No	No

In [347]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_test_pred)
```

Out[347]: 0.7558205565019875

- Accuracy rate is 75% with KNN Classifier

DECISION TREE CLASSIFIER

In [348]:
```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree.fit(X_train_transformed,y_train)
```

Out[348]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [349]:
```
y_test_pred = tree.predict(X_test_transformed)
```

In [350]:
```
temp_df = pd.DataFrame({'Actual' : y_test,'Predicted' :y_test_pred })
temp_df.head()
```

Out[350]:

	Actual	Predicted
4880	No	No
1541	No	No
1289	No	No
5745	No	No
4873	No	No

In [351]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_test_pred)
```

Out[351]: 0.6990346394094265

- Accuracy Rate with Decision Tree - 69%

Ensemble Random Forest

In [353]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train_transformed,y_train)
```

Out[353]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [355]:
```
y_test_pred = rfc.predict(X_test_transformed)
```

In [356]:
```
temp_df = pd.DataFrame({'Actual' : y_test,'Predicted' :y_test_pred })
temp_df.head()
```

Out[356]:

	Actual	Predicted
4880	No	No
1541	No	No
1289	No	No
5745	No	No
4873	No	No

In [357]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_test_pred)
```

Out[357]:
```
0.7825099375354913
```

- Random Forest Accuracy Rate - 78%

GRADIENT BOOSTING CLASSIFIER

In [359]:
```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier()
gbc.fit(X_train_transformed,y_train)
```

Out[359]:
```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [362]:
```
y_test_pred = gbc.predict(X_test_transformed)
```

In [363]:
```
temp_df = pd.DataFrame({'Actual' : y_test,'Predicted' :y_test_pred })
temp_df.head()
```

Out[363]:

	Actual	Predicted
4880	No	No
1541	No	No
1289	No	No
5745	No	No
4873	No	No

In [364]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_test_pred)
```

Out[364]:
```
0.7830777967064169
```

- Accuracy Rate with Gradient Boosting is 78%

NAIVE BAYES CLASSIFIER

In [365]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train_transformed,y_train)
```

Out[365]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [366]:
```
y_test_pred = gnb.predict(X_test_transformed)
```

In [367]:
```
temp_df = pd.DataFrame({'Actual' : y_test,'Predicted' :y_test_pred })
temp_df.head()
```

Out[367]:

	Actual	Predicted
4880	No	No
1541	No	No
1289	No	No
5745	No	No
4873	No	No

In [368]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_test_pred)
```

Out[368]:
```
0.639977285633163
```

- Accuracy Rate with Naive Bayes - 63%

```
Logistic Regression : 0.7842135150482681
KNN CLASSIFIER : 0.7558205565019875
```

```
Decision Tree : 0.6990346394094265
Random Forest : 0.7825099375354913
Gradient Boosting : 0.7830777967064169
Naive Bayes : 0.639977285633163
```

- Logistic/random/Gradient are giving the highest accuracy

In [ ]:

```
```