

# Customer Churn Analysis and Prediction

## Project Overview

- Customer churn refers to the phenomenon where customers stop doing business with a company or stop using its services. In simple terms, it's when customers leave or "churn away" from a business, often switching to a competitor or just stopping their use of the service altogether. It's an important metric for companies because retaining existing customers is typically more cost-effective than acquiring new ones, so high churn rates can negatively impact a company's revenue and growth.
- The data used in this project is a dataset obtained from Kaggle(a telco dataset). The aim of this project seeks to analyse the churn rate of customers of this organisation based on certain features. Amongst such features include; Whether or not the customer is a senior citizen. Whether the customer has a partner, dependents. The tenure of use of the customer etc.

In [1]:

```
1 # Importing the necessary libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 %matplotlib inline
```

```
In [2]: 1 # Reading the data
```

```
2 data = pd.read_csv('CustomerChurn.csv')  
3 data
```

Out[2]:

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

7043 rows × 21 columns



```
In [3]: 1 # Checking the shape(dimension) of the data
```

```
2 data.shape
```

Out[3]: (7043, 21)

```
In [4]: 1 # The columns in the data set are listed below;  
2 for column in data.columns:  
3     print (column)
```

```
customerID  
gender  
SeniorCitizen  
Partner  
Dependents  
tenure  
PhoneService  
MultipleLines  
InternetService  
OnlineSecurity  
OnlineBackup  
DeviceProtection  
TechSupport  
StreamingTV  
StreamingMovies  
Contract  
PaperlessBilling  
PaymentMethod  
MonthlyCharges  
TotalCharges  
Churn
```

```
In [5]: 1 # Getting basic information about the data
2 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 [6]: 1 data.head(2)
2 pd.options.display.max_columns
```

Out[6]: 20

```
In [7]: 1 # checking the data types of all columns  
2 data.dtypes
```

```
Out[7]: 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
```

- Senior Citizen is not considered numerical because it is more or less of a categorical(boolean) feature
- 75% of the customers have tenure less than 55 months

```
In [8]: 1 # checking for customers who churned  
2 data['Churn'].value_counts()
```

```
Out[8]: No      5174  
Yes     1869  
Name: Churn, dtype: int64
```

```
In [9]: 1 (data['Churn'].value_counts() / len(data))*100
```

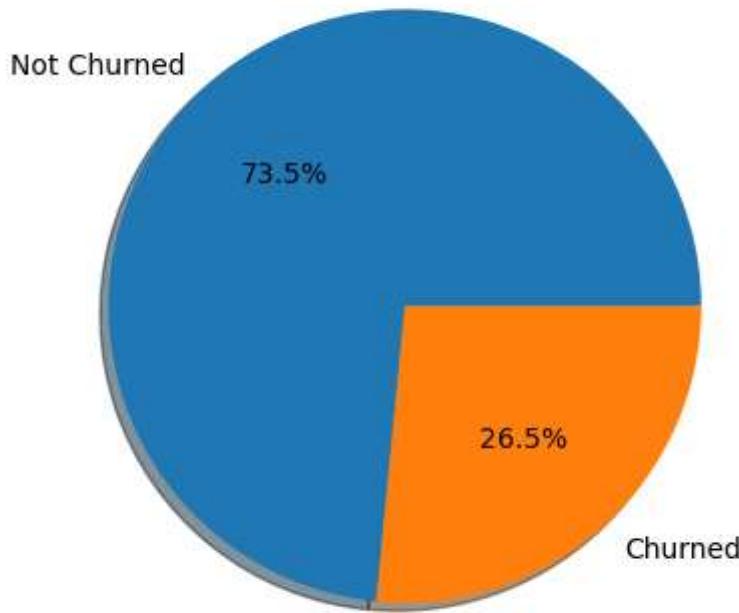
```
Out[9]: No      73.463013  
Yes     26.536987  
Name: Churn, dtype: float64
```

- Customer churn rate is 26.54% which is slightly above a quarter

```
In [10]: 1 # Visualising number of customers who have churned
2 plt.title('Percentage of Churned Customers')
3 plt.pie(x= data['Churn'].value_counts(),
4         autopct= '%1.1f%%', labels=['Not Churned', 'Churned'], shadow= True)
```

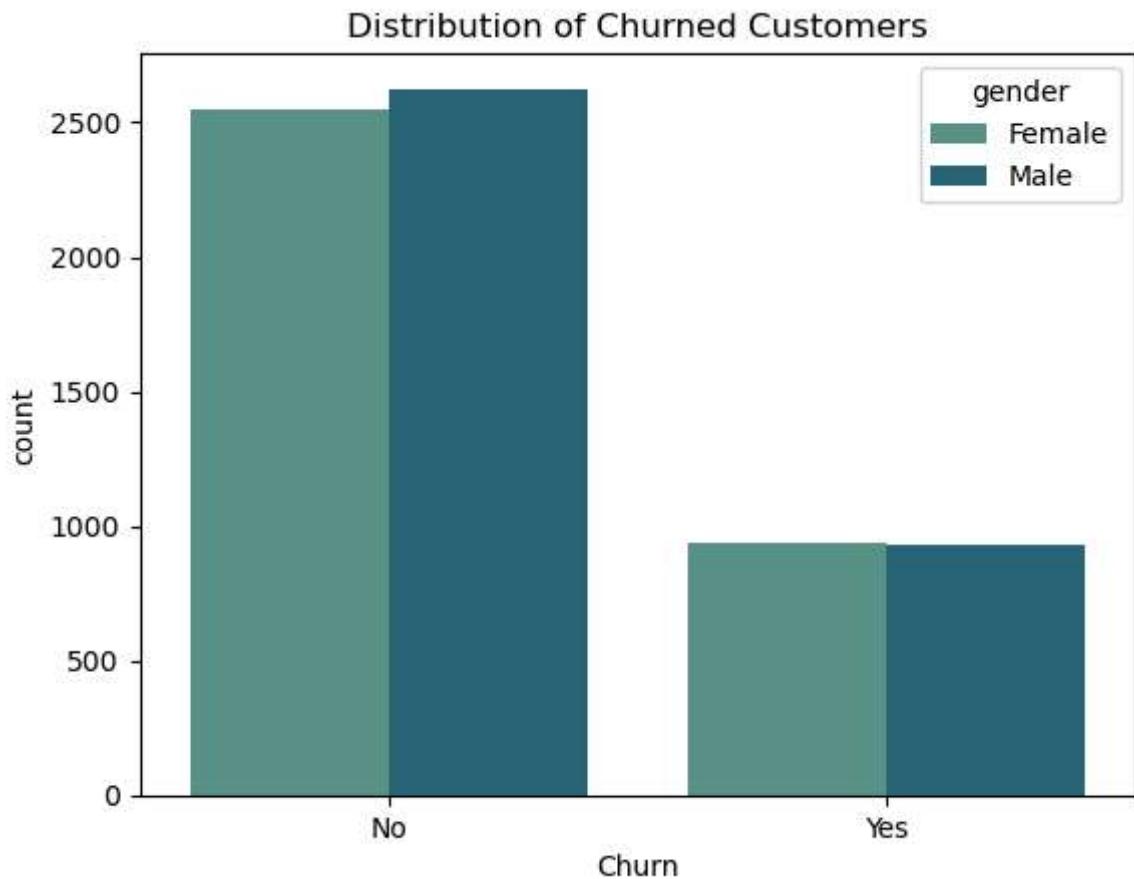
```
Out[10]: ([<matplotlib.patches.Wedge at 0x21e52f2be20>,
<matplotlib.patches.Wedge at 0x21e535dea30>],
[Text(-0.7393678277834757, 0.8144539368428056, 'Not Churned'),
Text(0.7393677515287918, -0.8144540060674139, 'Churned')],
[Text(-0.4032915424273503, 0.44424760191425755, '73.5%'),
Text(0.4032915008338864, -0.4442476396731348, '26.5%')])
```

Percentage of Churned Customers



```
In [11]: 1 plt.title('Distribution of Churned Customers')
2 sns.countplot(x= data['Churn'], hue= data['gender'], palette='crest')
```

```
Out[11]: <AxesSubplot:title={'center':'Distribution of Churned Customers'}, xlabel='Churn', ylabel='count'>
```



- There is no significant difference between males and females who have churned

In [12]: 1 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
```

# Data Cleaning

In [13]:

```
1 df = data.copy()  
2 df
```

Out[13]:

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

7043 rows × 21 columns



```
In [14]: 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 [15]: 1 df['TotalCharges'].value_counts()
```

```
Out[15]: 11
20.2      11
19.75     9
20.05     8
19.9      8
..
6849.4    1
692.35    1
130.15    1
3211.9    1
6844.5    1
Name: TotalCharges, Length: 6531, dtype: int64
```

```
In [16]: 1 # Converting the 'Total Charges' column into an appropriate data type
2 df['TotalCharges'] = df['TotalCharges'].astype('int', errors='ignore')
3 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 [17]: 1 # Missing Value treatment
2 df.dropna(how = 'any', inplace=True)
```

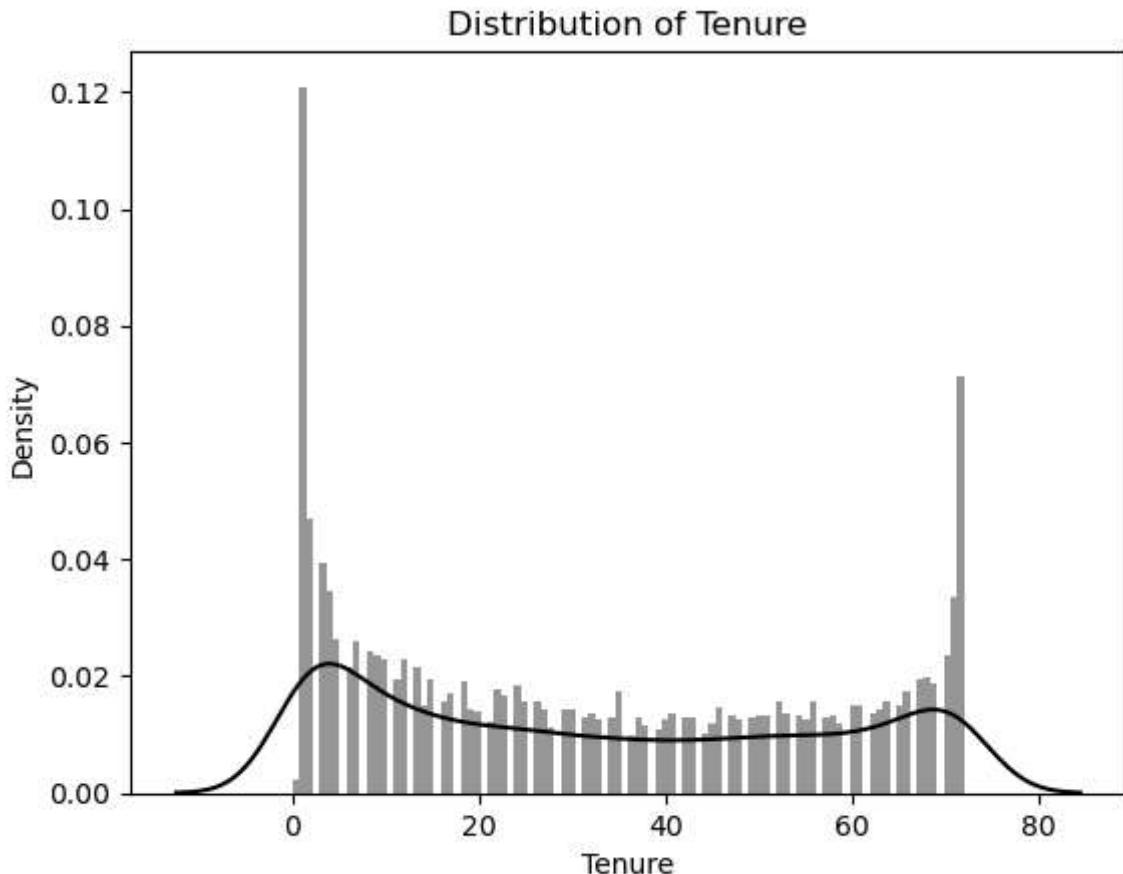
```
In [18]: 1 df.isnull().sum()
```

```
Out[18]: 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 [19]:

```
1 # Visualising the distribution of tenure
2 import warnings
3 warnings.filterwarnings(action= 'ignore')
4 plt.title('Distribution of Tenure')
5 plt.xlabel('Tenure')
6 sns.distplot(x= df['tenure'], color= 'black', bins= 100)
```

Out[19]: <AxesSubplot:title={'center':'Distribution of Tenure'}, xlabel='Tenure', ylabel='Density'>



# Data Analysis

In [20]: 1 df.head()

Out[20]:

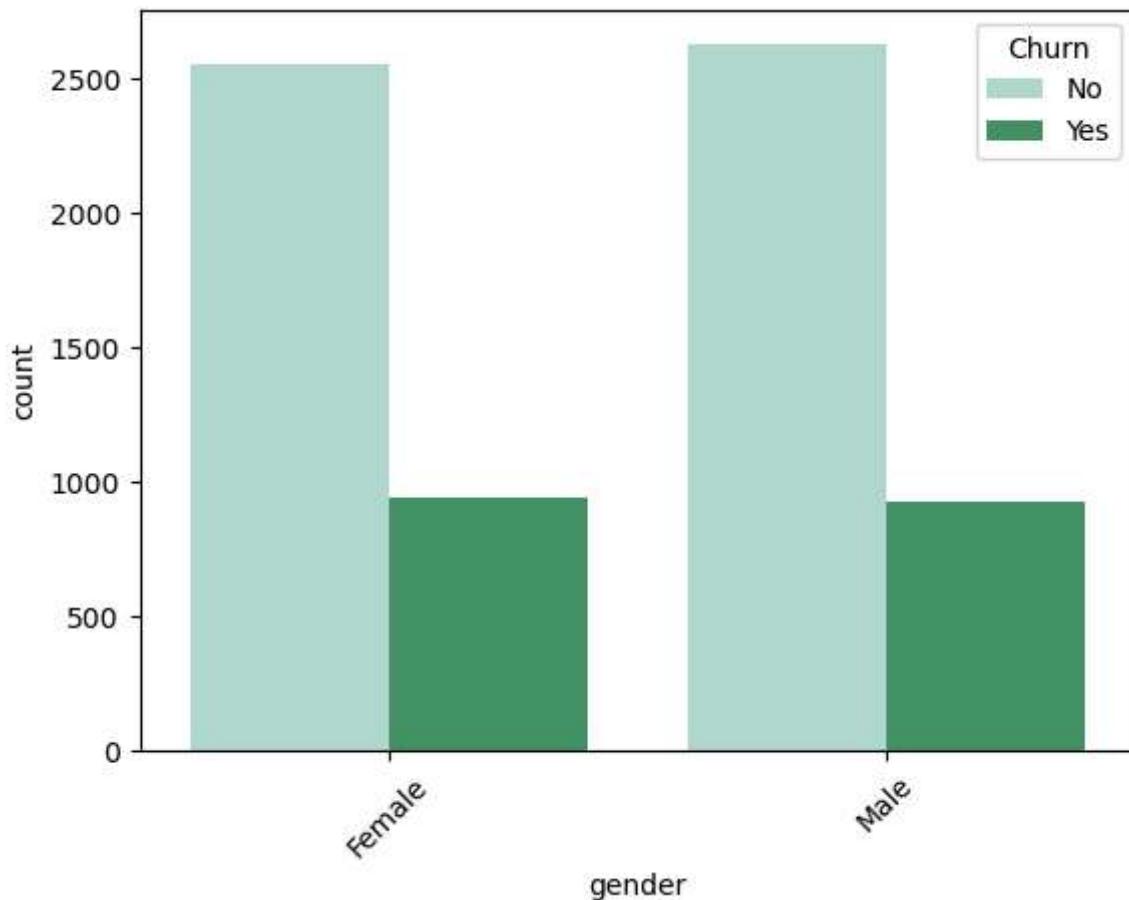
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

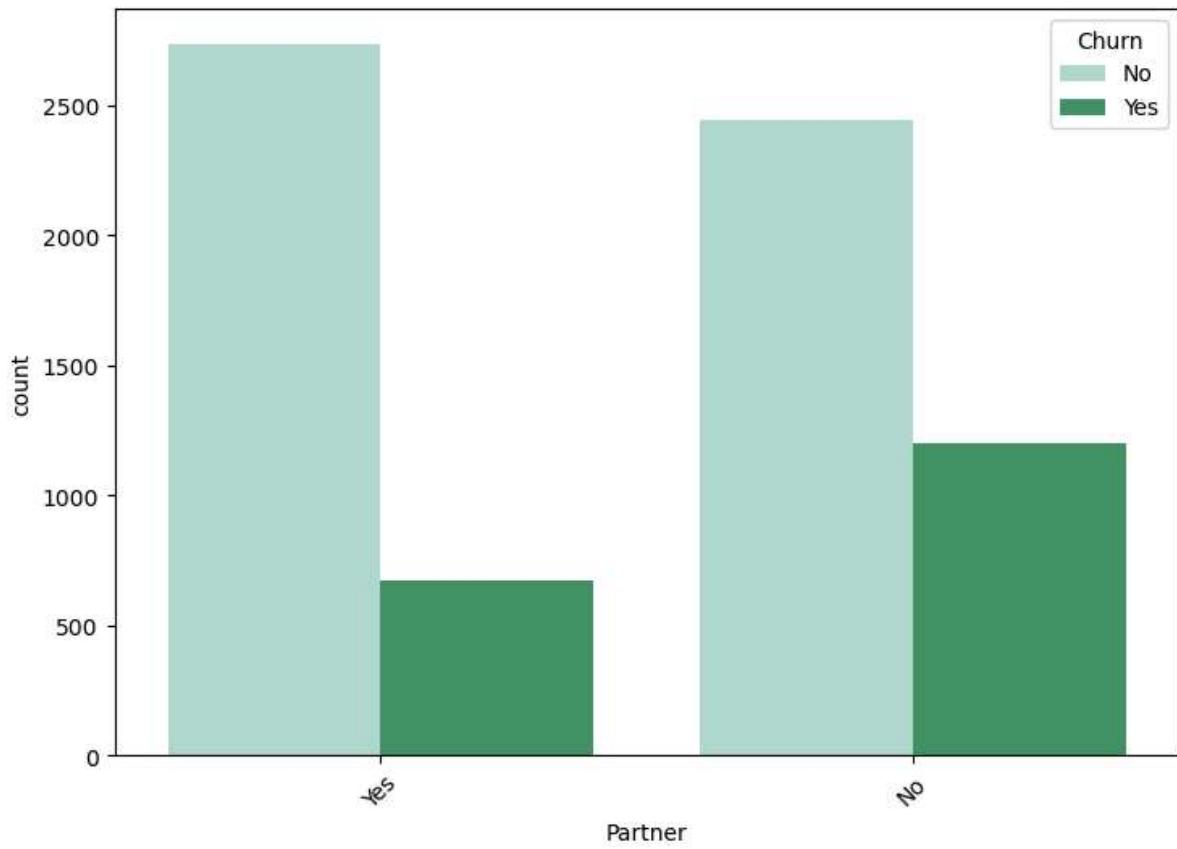
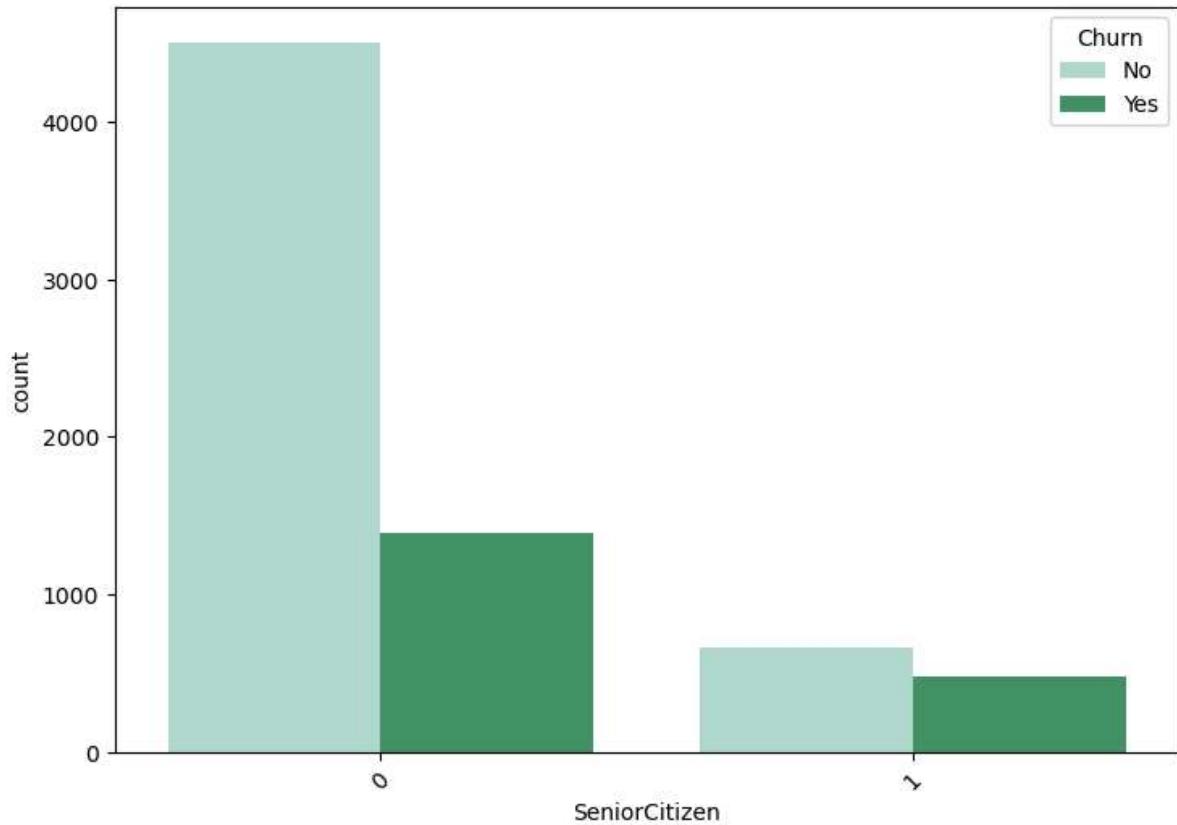
5 rows × 21 columns

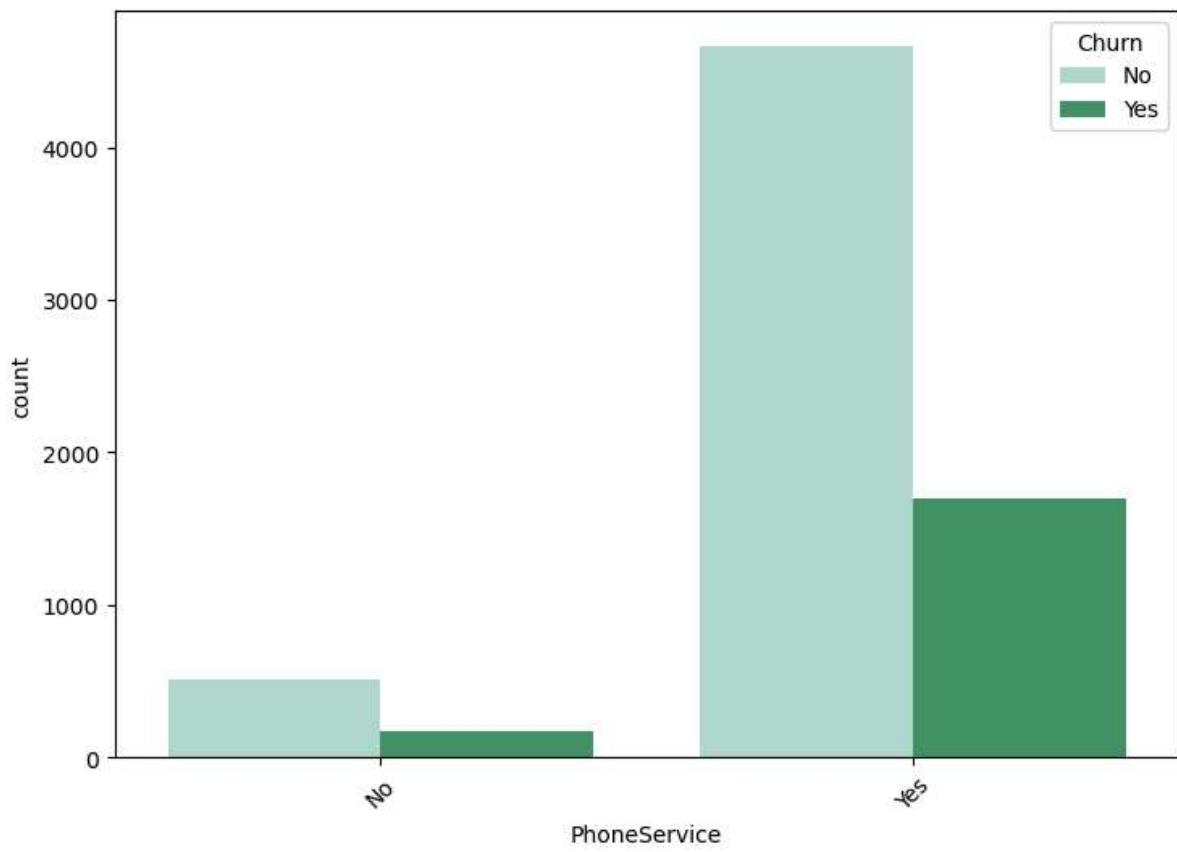
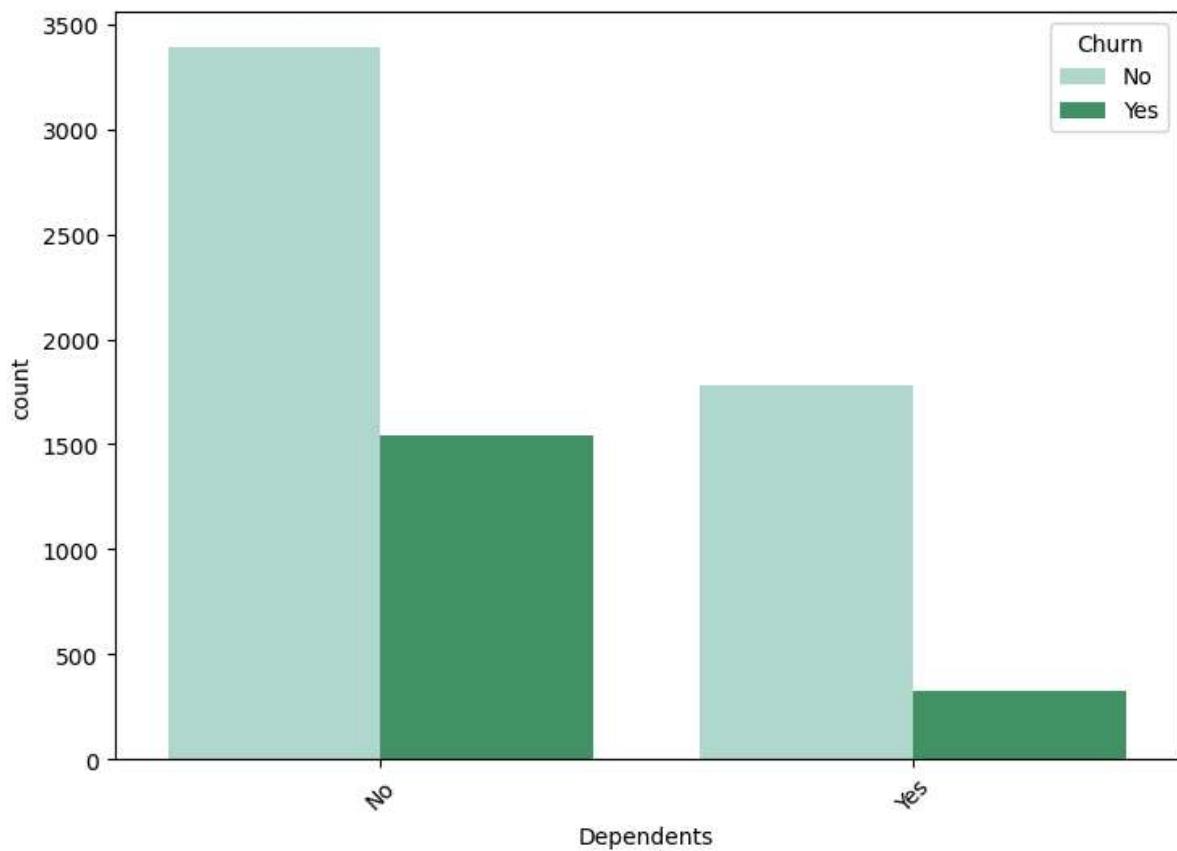
## Univariate Analysis

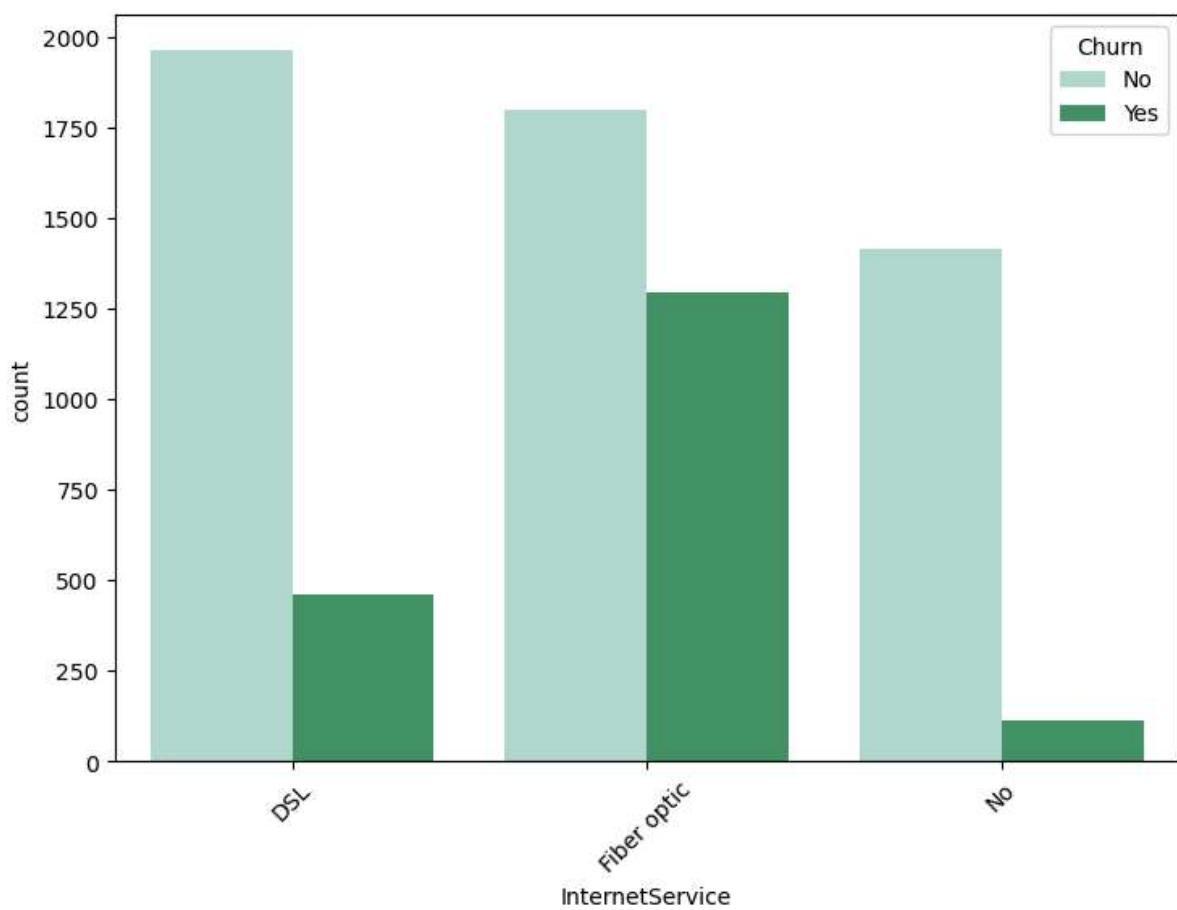
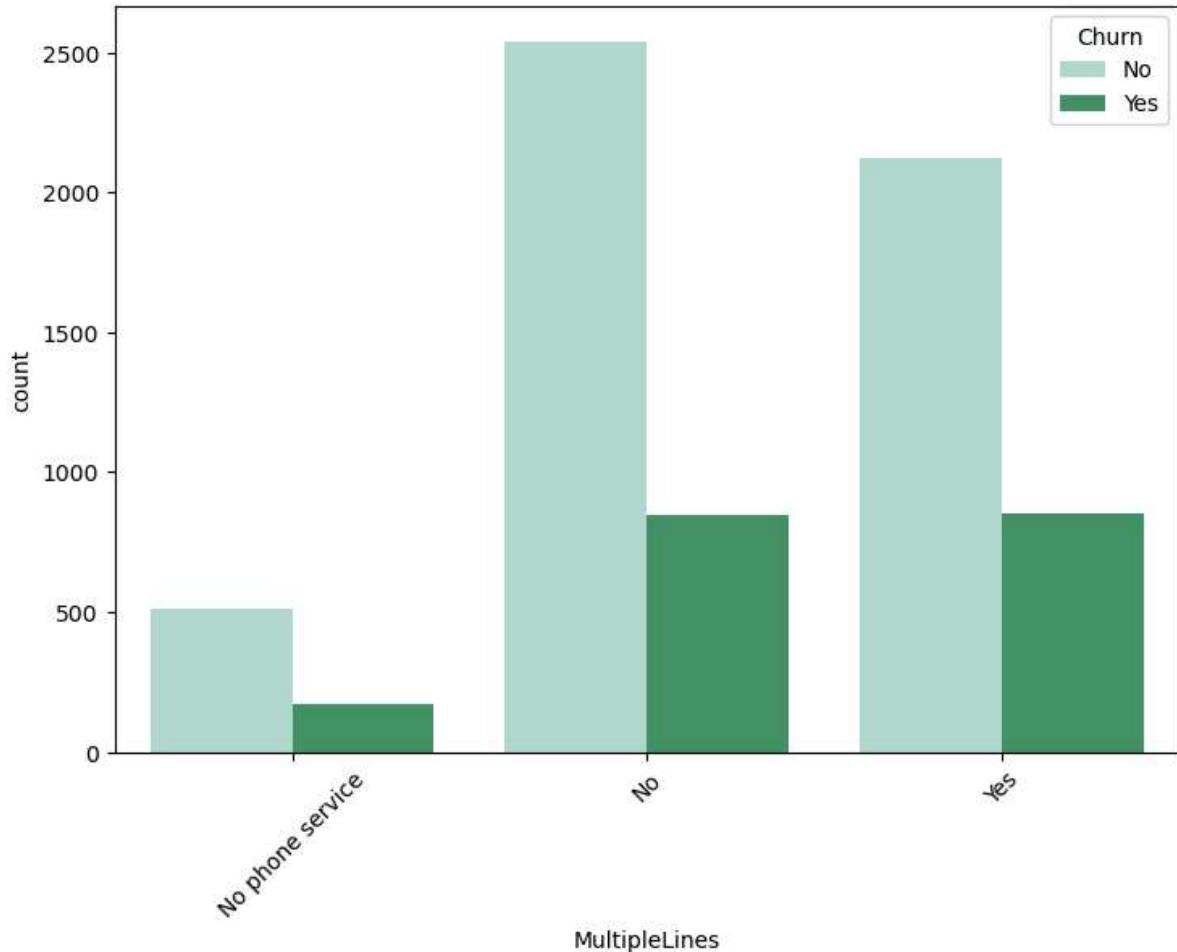
In [21]:

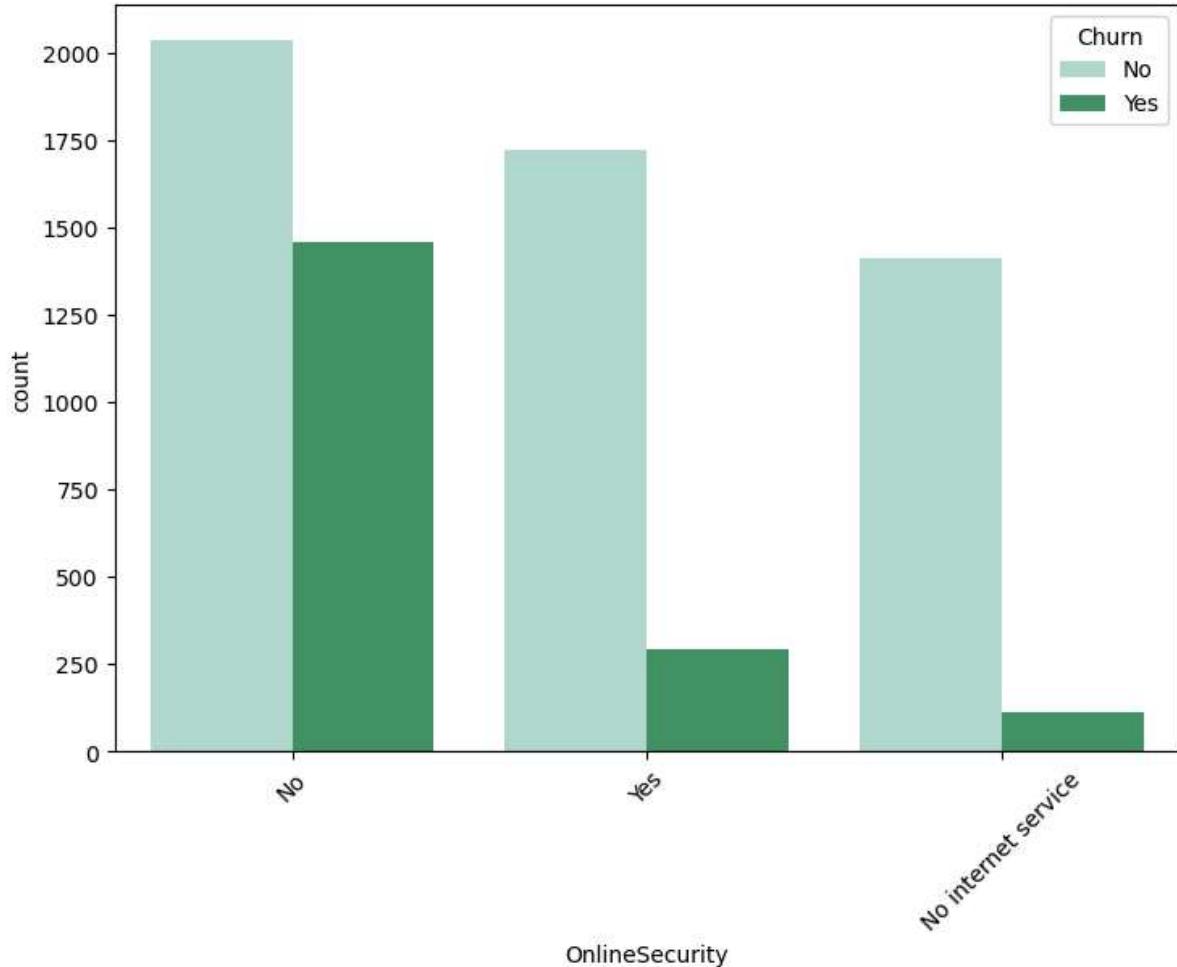
```
1 # Performing Univariate analysis on various featuers of the data in relati
2 for i, predictor in enumerate(df.drop(columns= ['MonthlyCharges', 'TotalCh
3     plt.figure(figsize=(8.5, 6))
4     plt.figure(i)
5     plt.xticks(rotation= 45)
6     sns.countplot(x= predictor, data= df, hue= 'Churn', palette='BuGn')
```

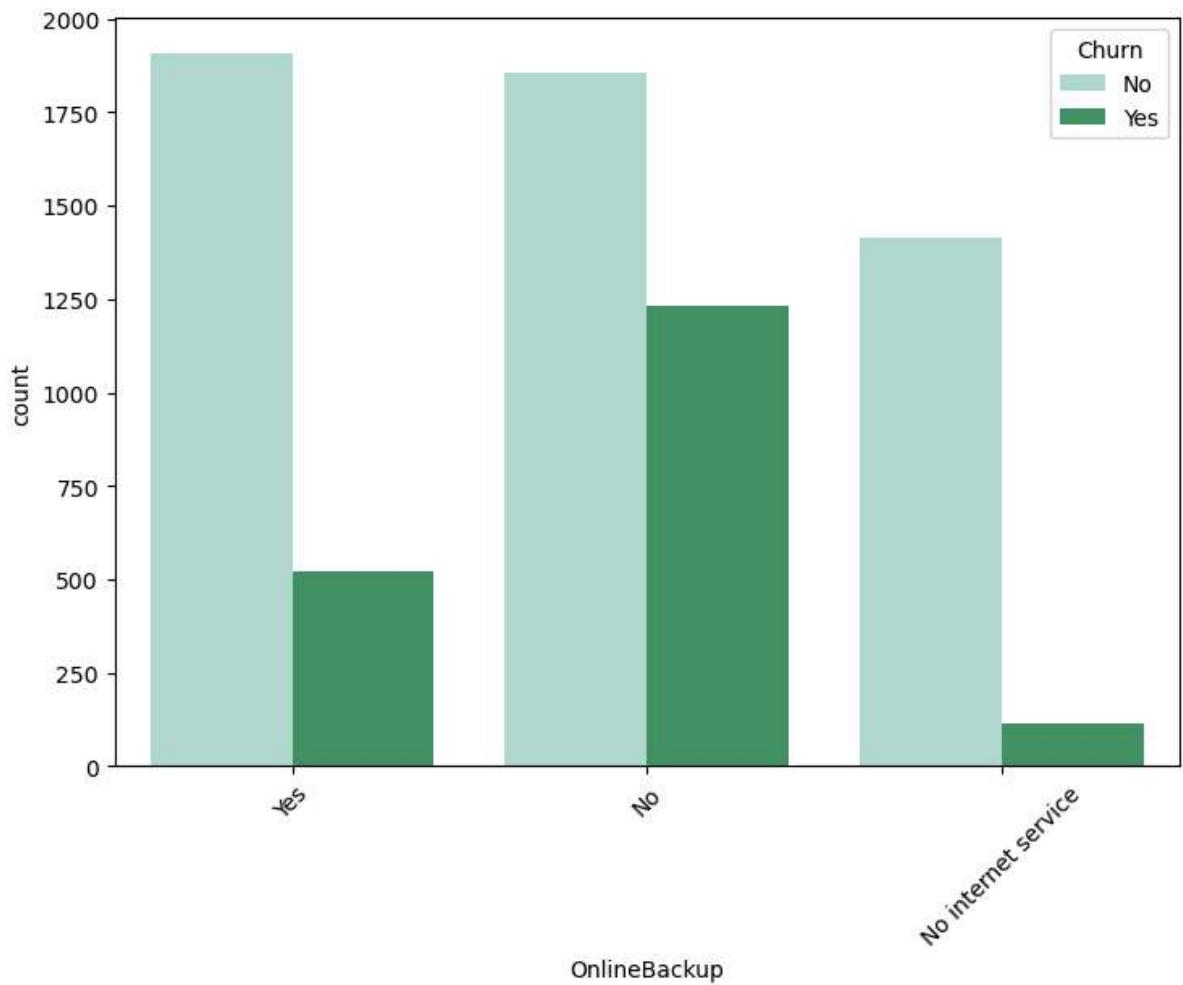


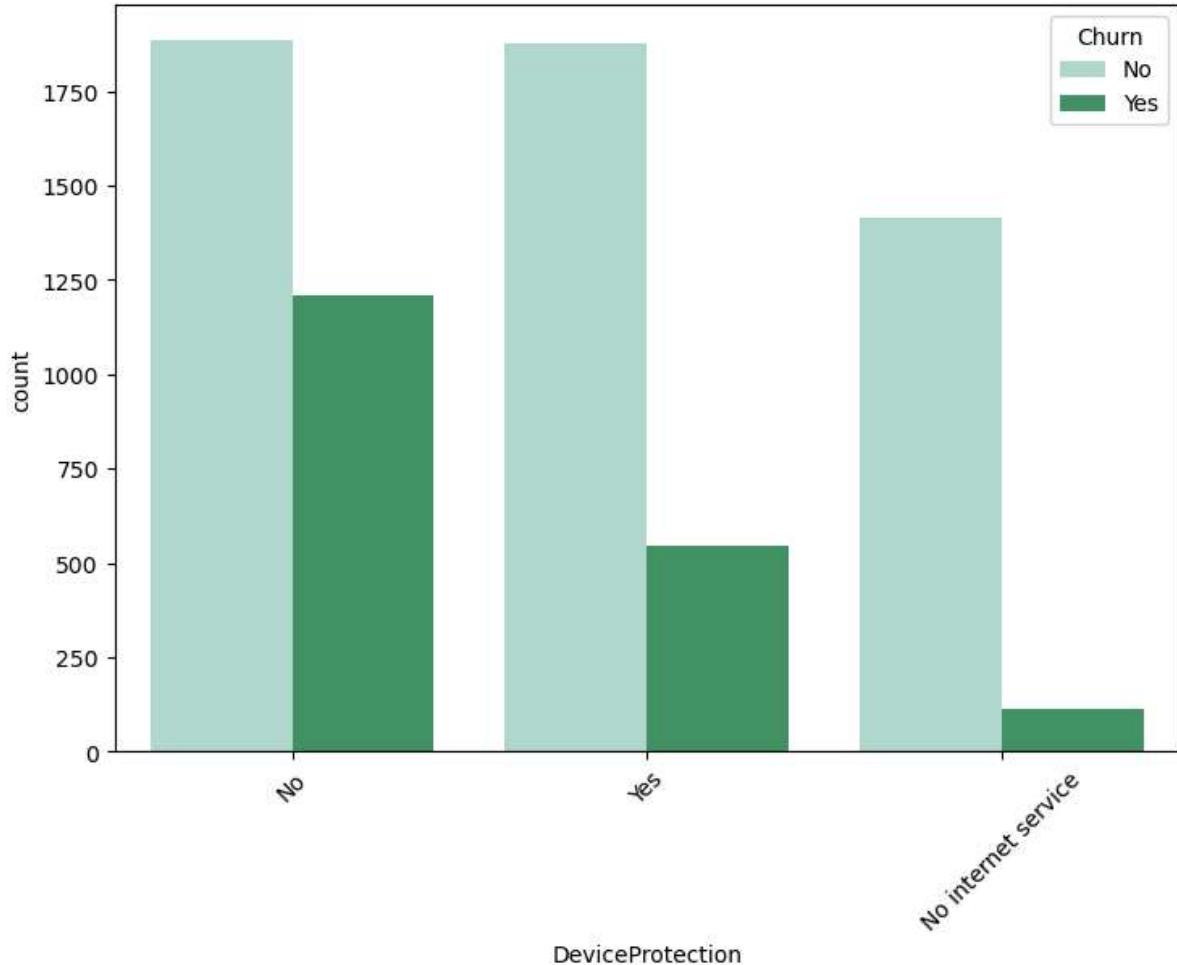


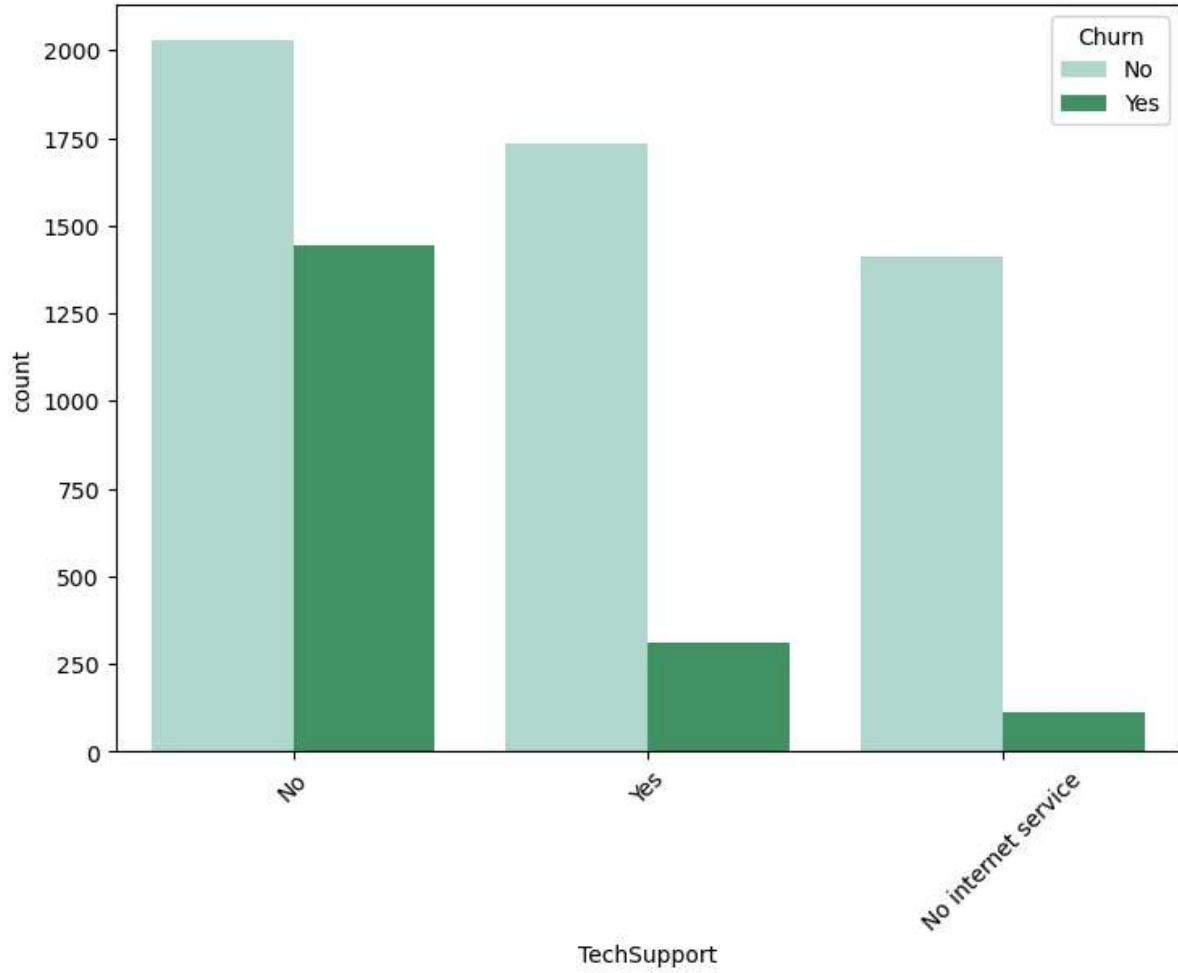


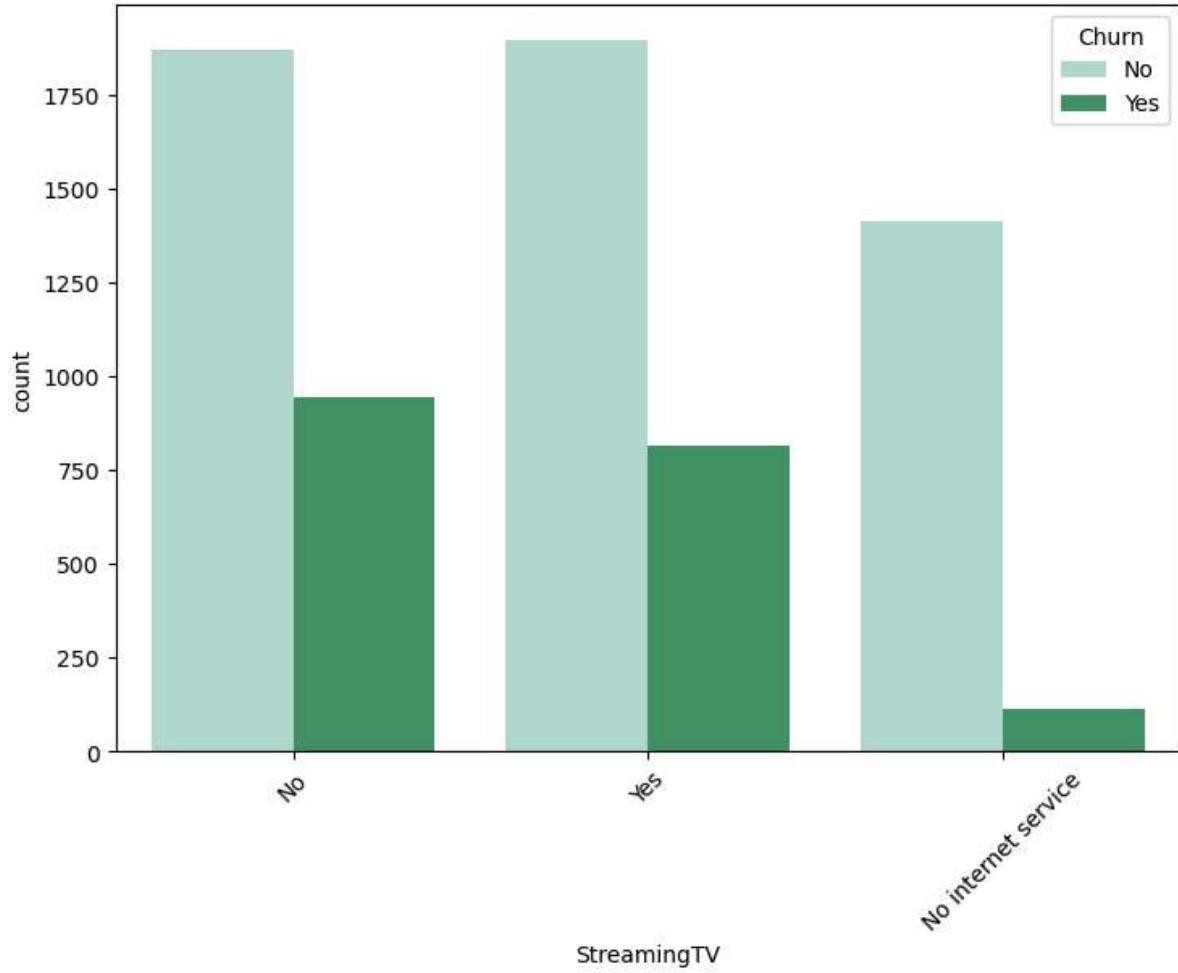


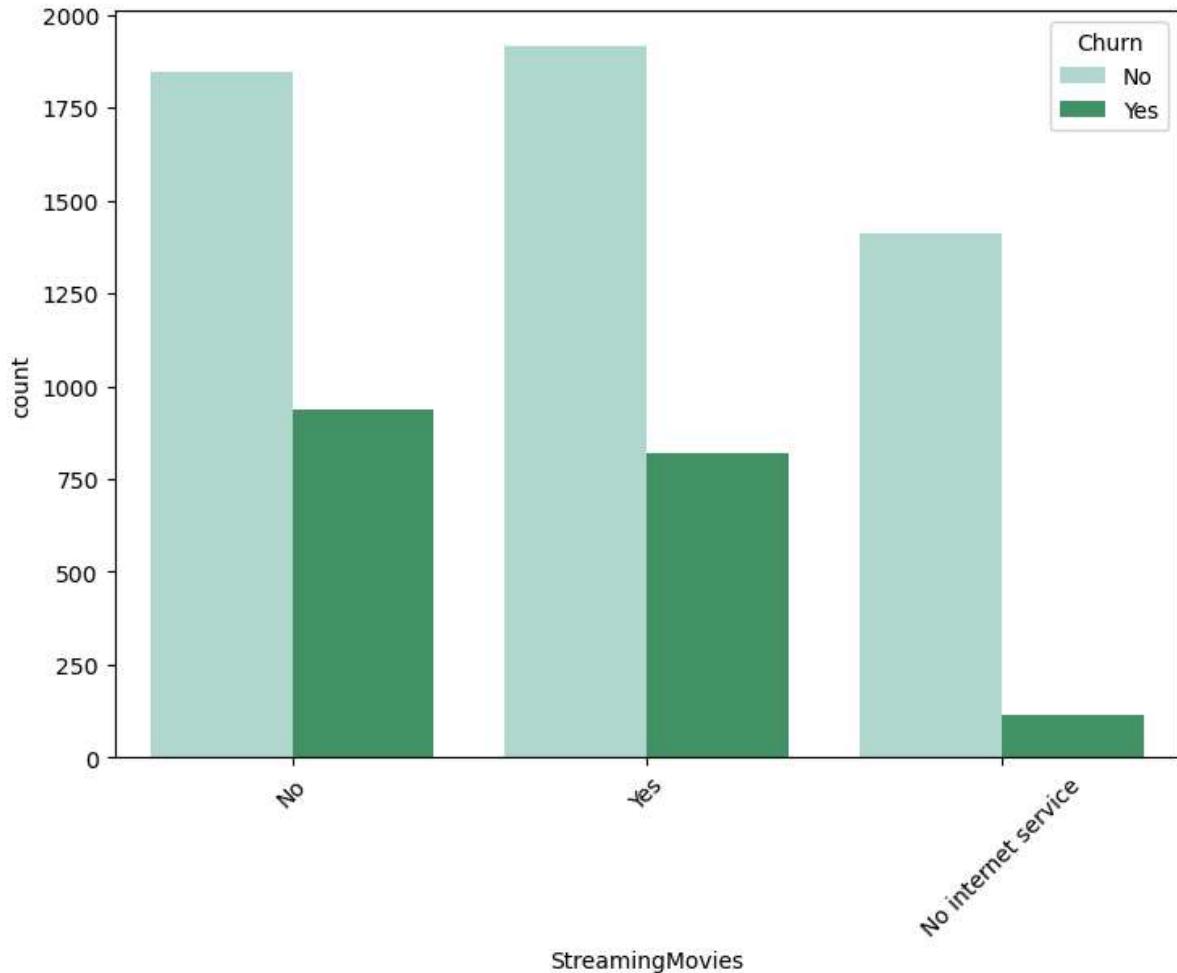


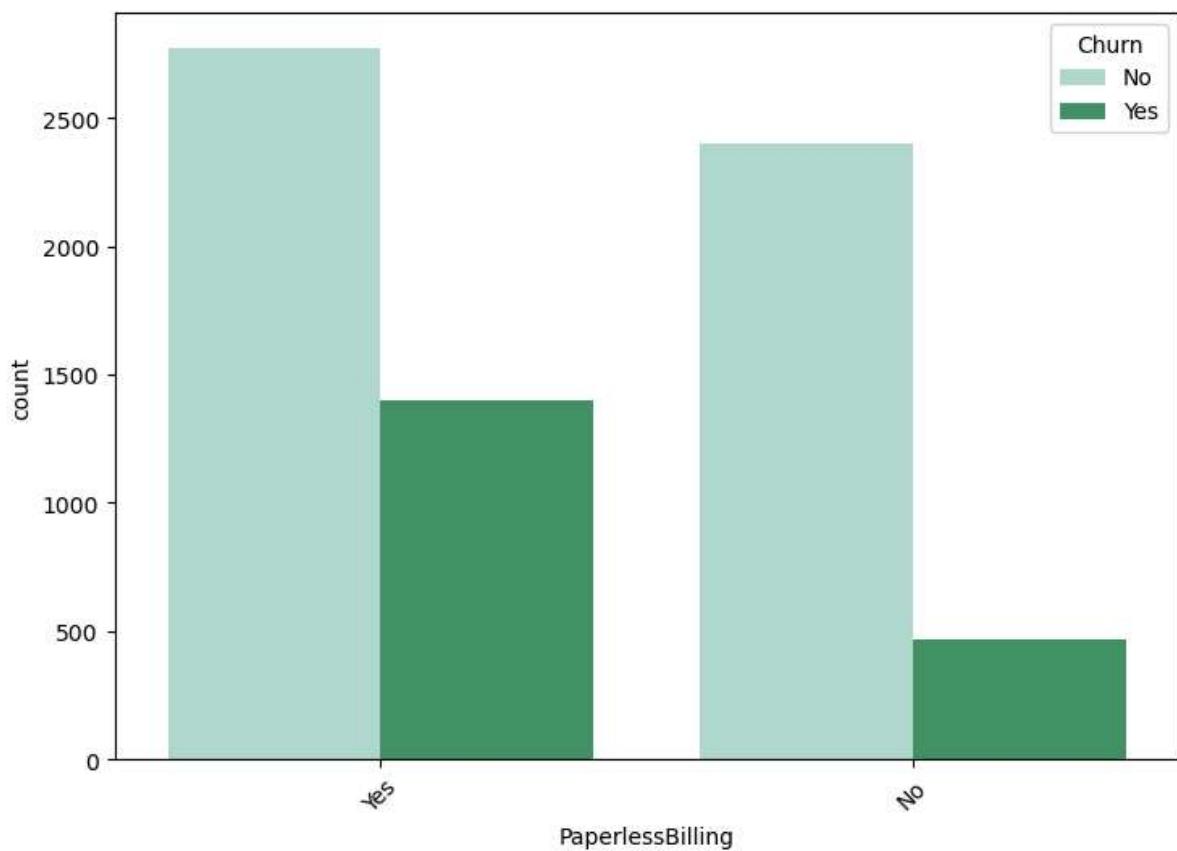
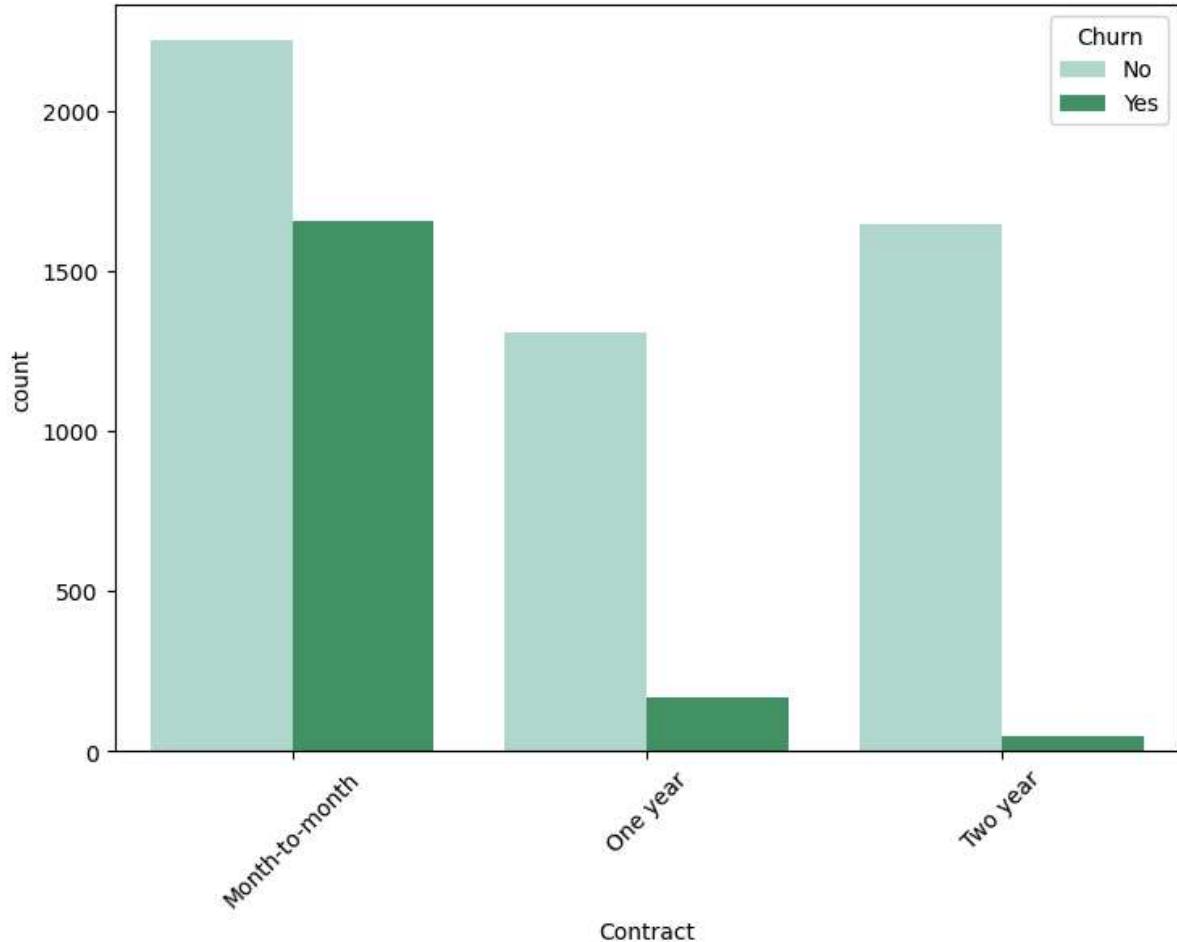


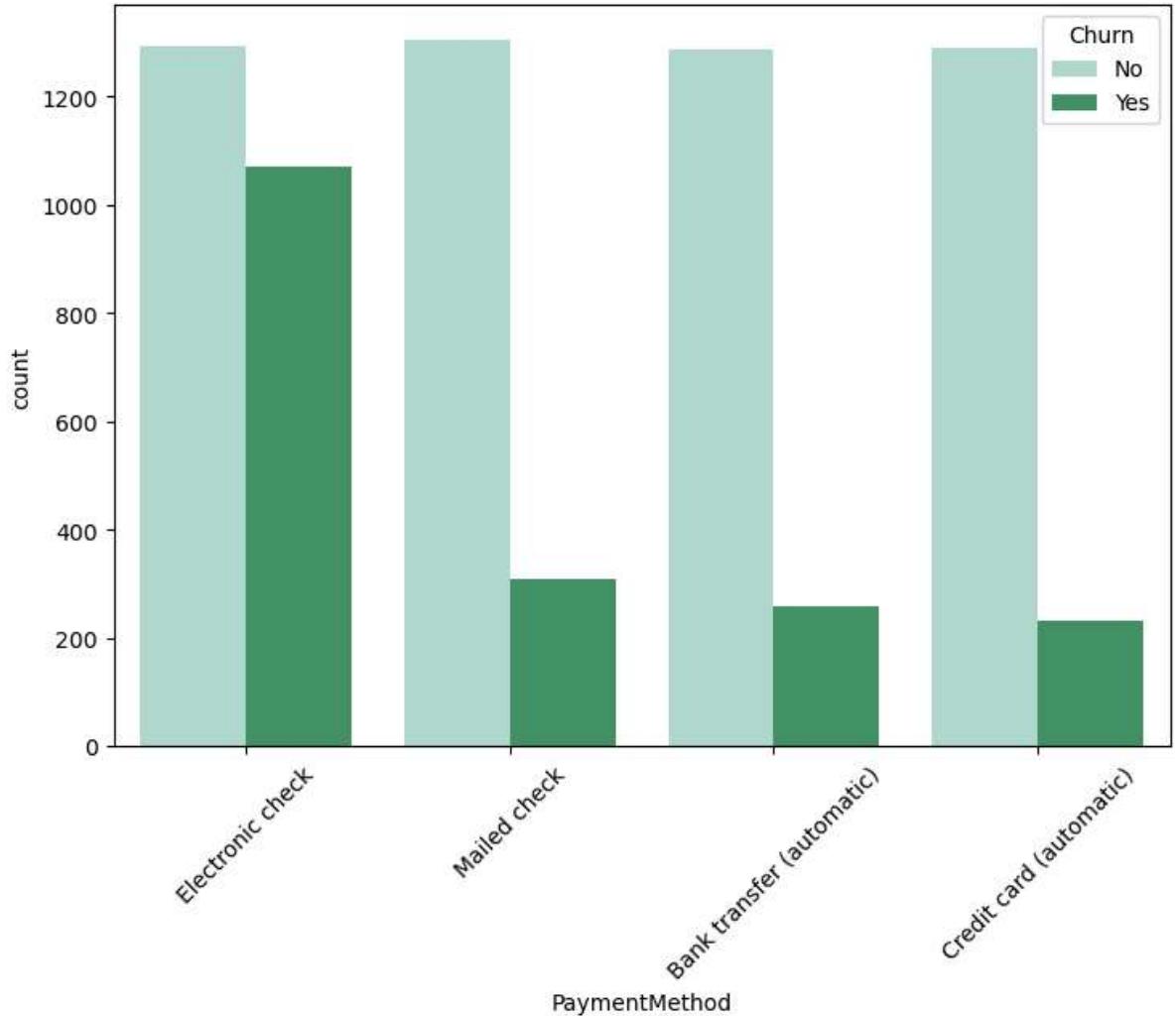




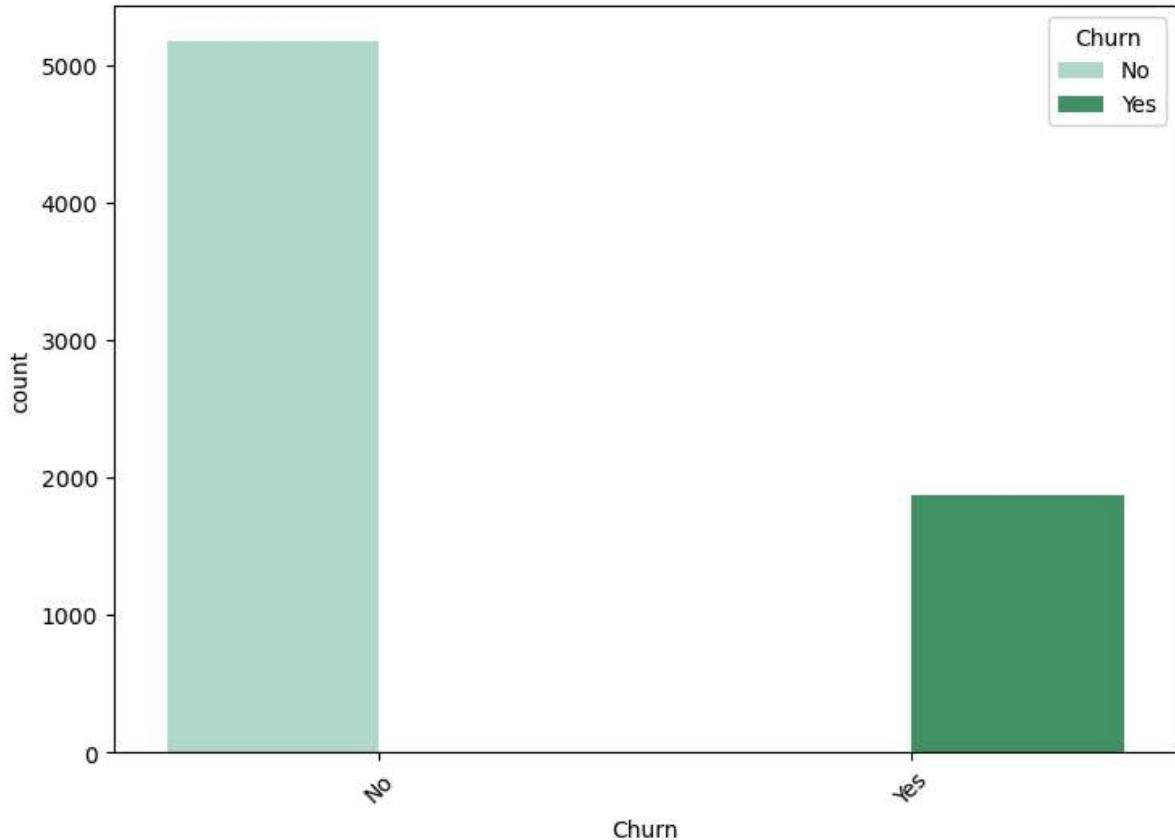








<Figure size 850x600 with 0 Axes>



```
In [22]: 1 df.columns
```

```
Out[22]: 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 [23]: 1 df['PaymentMethod'].value_counts()
```

```
Out[23]: Electronic check      2365  
Mailed check          1612  
Bank transfer (automatic) 1544  
Credit card (automatic) 1522  
Name: PaymentMethod, dtype: int64
```

```
In [24]: 1 pd.crosstab(df['PaymentMethod'], df['Churn'])
```

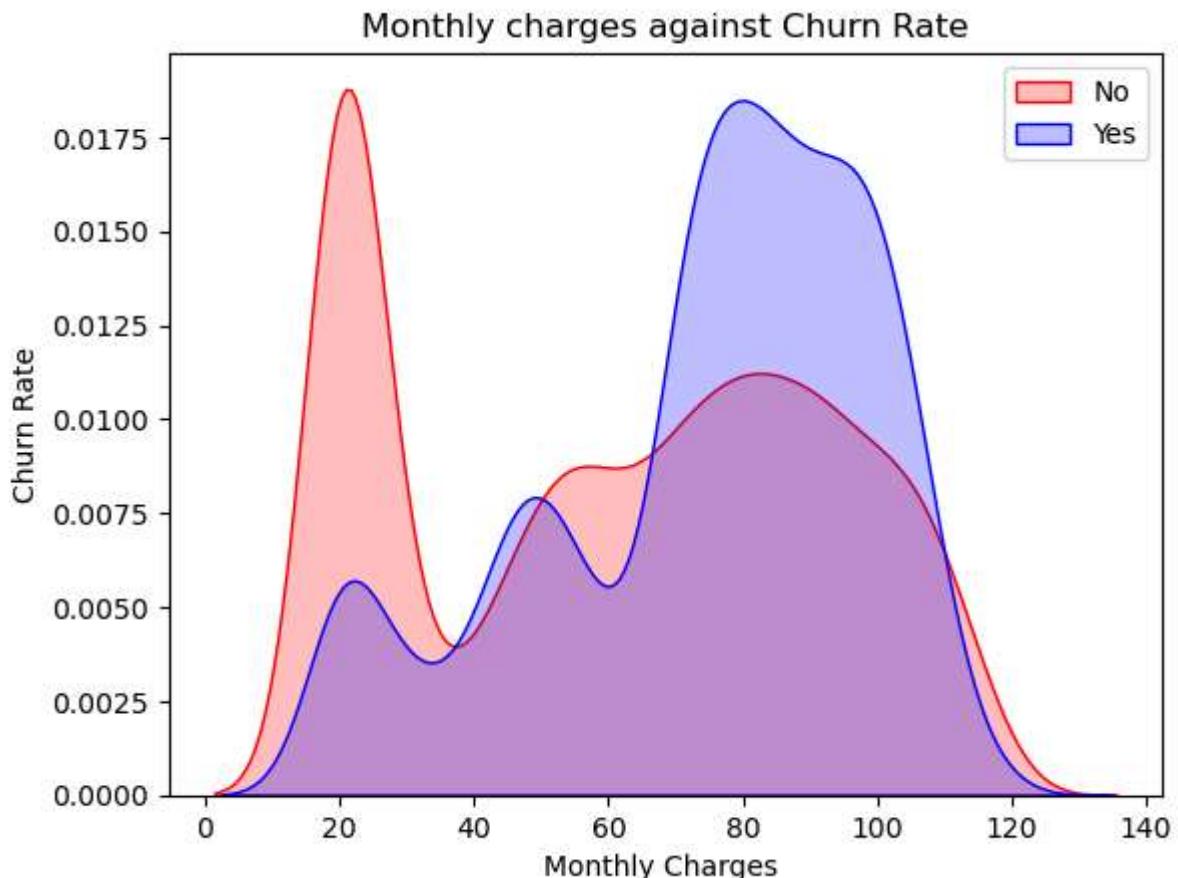
```
Out[24]:
```

	Churn	No	Yes
PaymentMethod			
Bank transfer (automatic)	1286	258	
Credit card (automatic)	1290	232	
Electronic check	1294	1071	
Mailed check	1304	308	

```
In [25]: 1 # Encoding all categorical values  
2 # df_dummies = pd.get_dummies(df.drop(columns=['TotalCharges', 'MonthlyCha  
3 # df_dummies
```

```
In [27]: 1 # checking relationship between Monthly charges and Churn rate  
2 plt.title('Monthly charges against Churn Rate')  
3 plt.ylabel('Churn Rate')  
4 plt.xlabel('Monthly Charges ')  
5 Churn = sns.kdeplot(df.MonthlyCharges[(df['Churn'] == 'No')], color='Red',  
6 Churn = sns.kdeplot(df.MonthlyCharges[(df['Churn'] == 'Yes')],  
7 ax= Churn, color='blue', shade= True)  
8 plt.legend(['No', 'Yes'])
```

Out[27]: <matplotlib.legend.Legend at 0x21e56ed5f40>



- Customers tend to churn at higher monthly charges compared to lower monthly charges

```
In [28]: 1 # Converting data into their appropriate datatypes  
2 df['TotalCharges'] = df['TotalCharges'].astype('int', errors='ignore')
```

```
In [29]: 1 df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors= 'coerce')
```

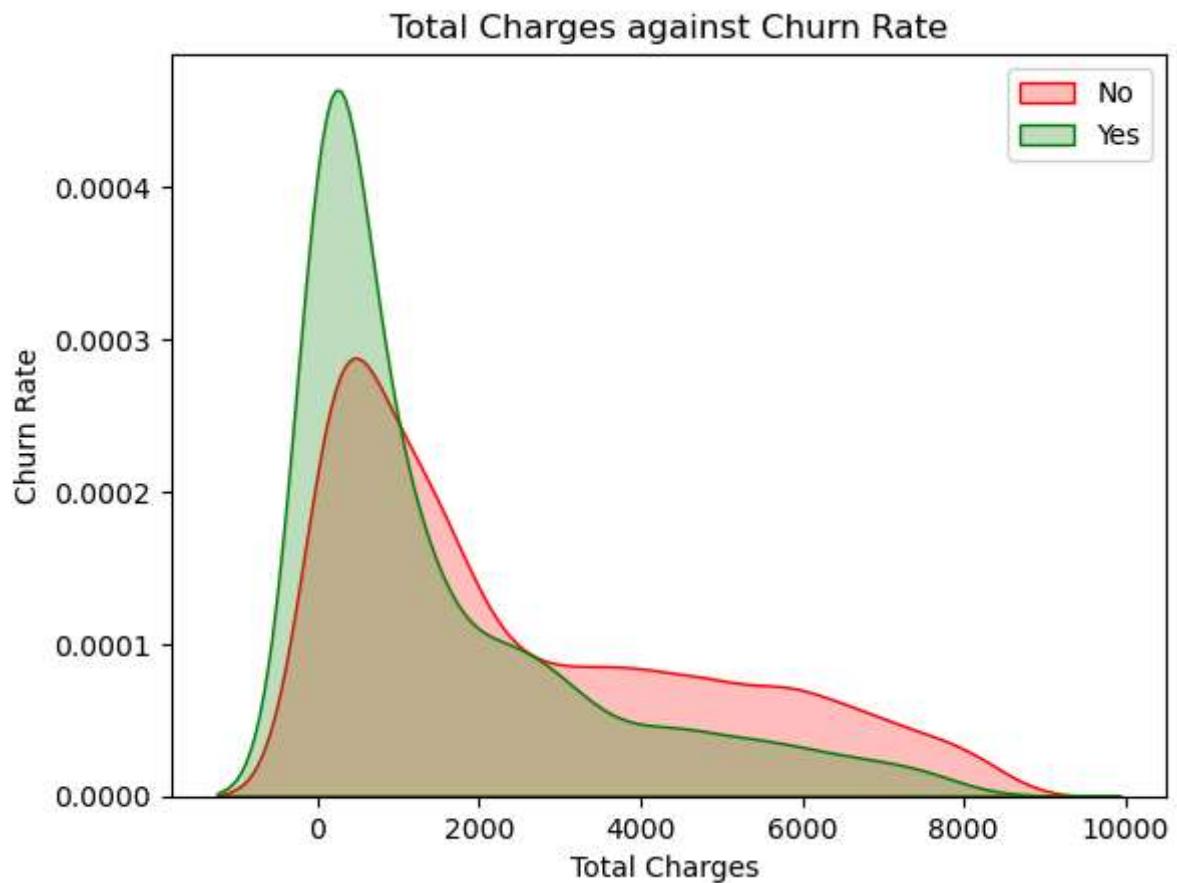
```
In [30]: 1 df.dtypes
```

```
Out[30]: 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 [31]:

```
1 # Checking total charges against churn rate
2 plt.title('Total Charges against Churn Rate')
3 plt.ylabel('Churn Rate')
4 plt.xlabel('Total Charges ')
5 Churn = sns.kdeplot(df.TotalCharges[(df['Churn'] == 'No')], color='Red', s
6 Churn = sns.kdeplot(df.TotalCharges[(df['Churn'] == 'Yes')]),
7                         ax= Churn, color='green', shade= True)
8 plt.legend(['No', 'Yes'])
```

Out[31]: <matplotlib.legend.Legend at 0x21e56f5ba30>



```
In [35]: 1 df.head()
```

```
Out[35]:
```

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

5 rows × 21 columns



```
In [36]: 1 from sklearn.preprocessing import LabelEncoder  
2 encoder = LabelEncoder()
```

```
In [37]: 1 df['Churn_encoded'] = encoder.fit_transform(df['Churn'])
```

```
In [38]: 1 df.head()
```

```
Out[38]:
```

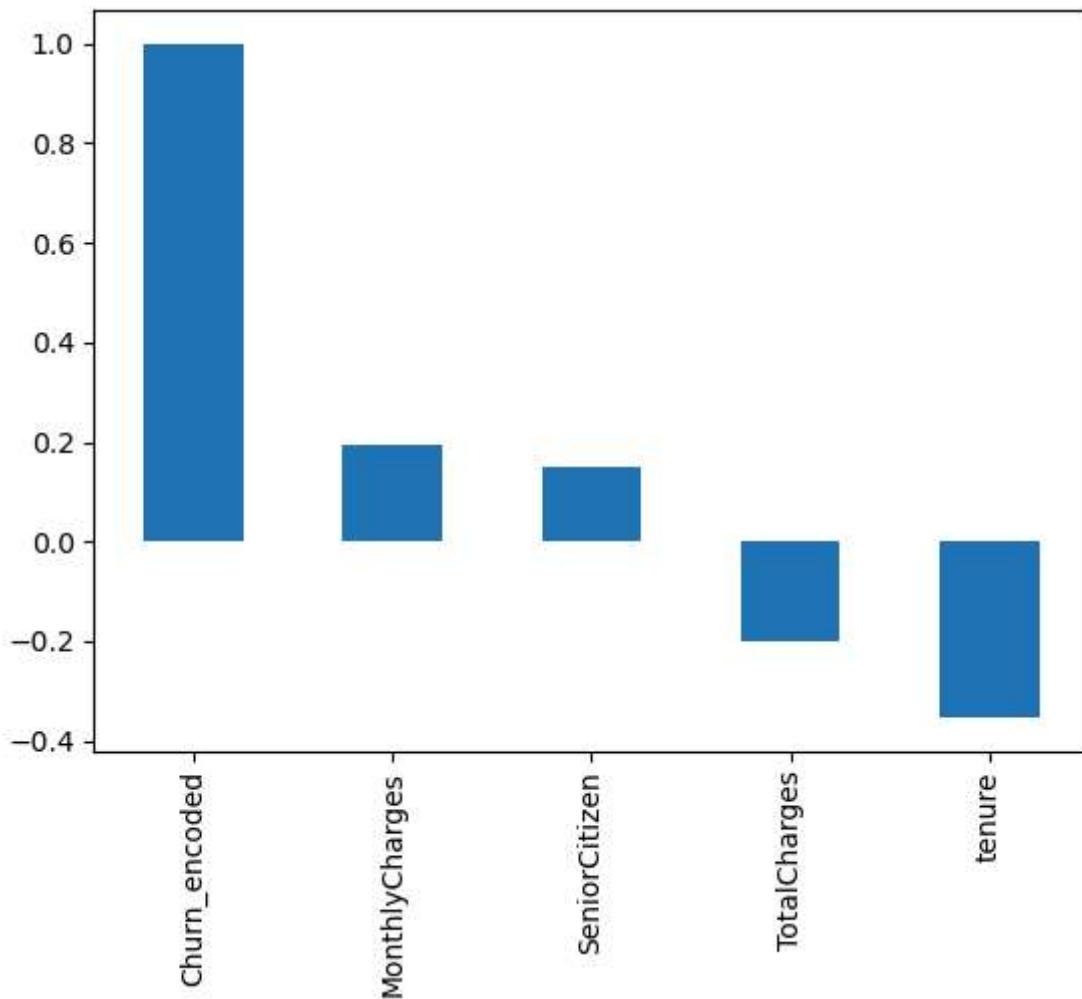
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

5 rows × 22 columns



```
In [39]: 1 # Checking the correlation of predictors with churn  
2 df.corr()['Churn_encoded'].sort_values(ascending= False).plot(kind= 'bar')
```

Out[39]: <AxesSubplot:>



```
In [42]: 1 df_dummies = pd.get_dummies(df.drop(columns= ['customerID', 'tenure', 'Mont  
2 'TotalCharges', 'Churn_encoded'])  
3 df_dummies.head()
```

Out[42]:

	SeniorCitizen	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Depend
0	0		1	0	0	1	1
1	0		0	1	1	0	1
2	0		0	1	1	0	1
3	0		0	1	1	0	1
4	0		1	0	1	0	1

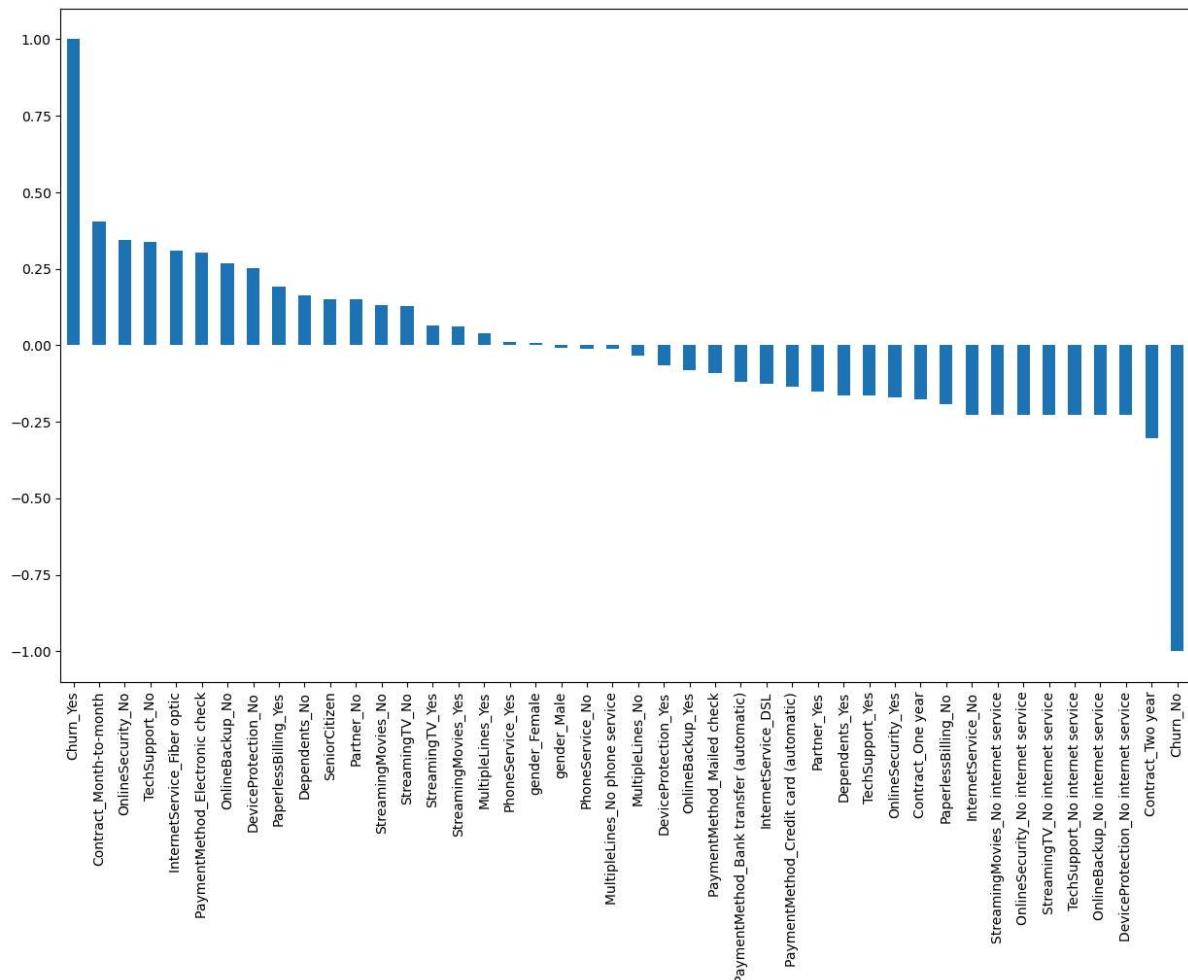
5 rows × 44 columns



In [43]:

```
1 # Checking the correlation of predictors with churn
2 plt.figure(figsize=(15, 9))
3 df_dummies.corr()['Churn_Yes'].sort_values(ascending=False).plot(kind= '
```

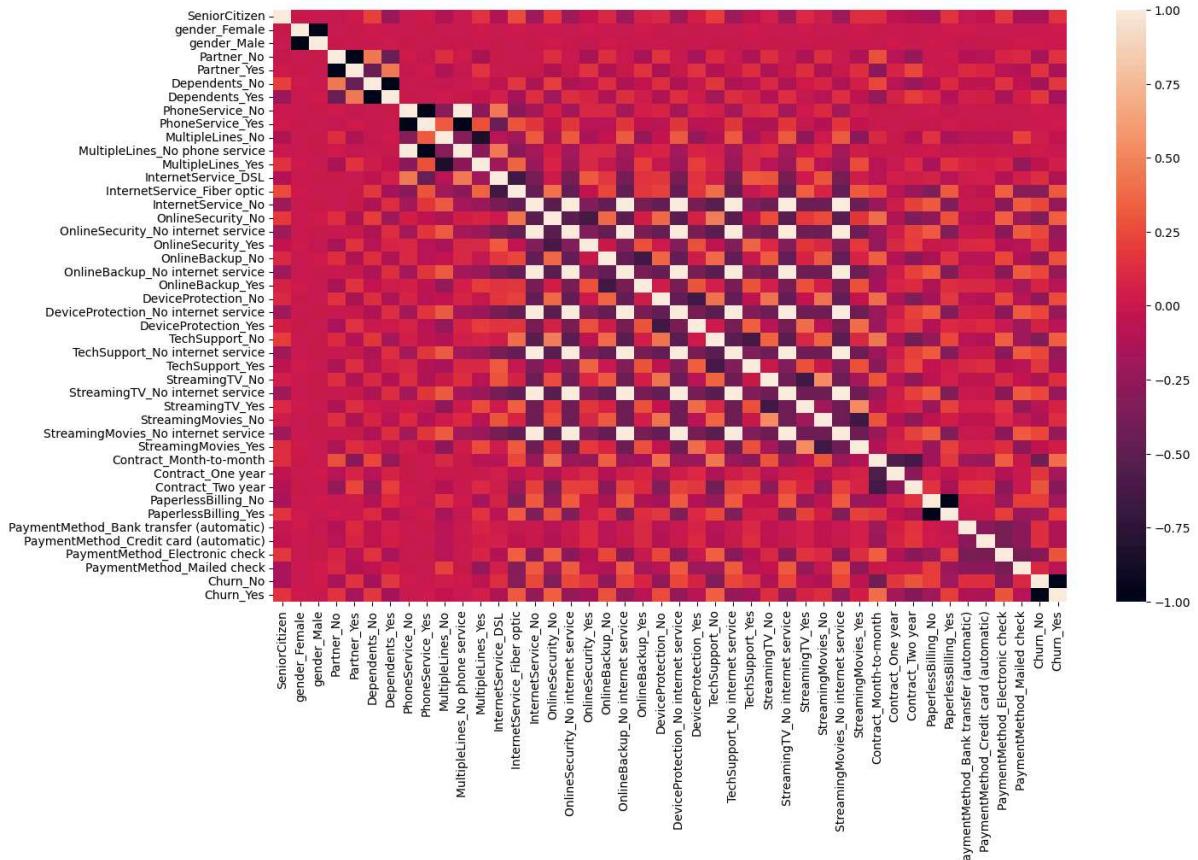
Out[43]: <AxesSubplot:>



- In instances of month-to-month Contract, No Online Security, No tech Support and others, Customers tend to churn.
- In instances of long term Contract, Device protection, Tech Support and others, Customers tend not to churn.
- Factors like gender, Multi-lines, and Phone service have little or no impact on customer churning

```
In [44]: 1 # Visualising correlation between various variables
          2 plt.figure(figsize=(15,9))
          3 sns.heatmap(df_dummies.corr())
```

Out[44]: <AxesSubplot:>



## Univariate Analysis

```
In [45]: 1 df_target0 = df.loc[df['Churn_encoded'] == 0]
          2 df_target1 = df.loc[df['Churn_encoded'] == 1]
```

```
In [46]: 1 len(df_target0), len(df_target1)
```

Out[46]: (5174, 1869)

```
In [47]: 1 # Creating a function to plot the relationships between various categorical  
2 def uniplot(df, col, title, hue= None ):  
3     sns.set_style('whitegrid')  
4     sns.set_context('talk')  
5     plt.rcParams['axes.labelsize'] = 20  
6     plt.rcParams['axes.titlesize'] = 22  
7     plt.rcParams['axes.titlepad'] = 30  
8  
9     temp = pd.Series(data =hue)  
10    fig, ax = plt.subplots()  
11    width = len(df[col].unique()) + 2 + 4*len(temp.unique())  
12    fig.set_size_inches(width, 6)  
13    plt.xticks(rotation= 45)  
14    plt.yscale('linear')  
15    plt.title(title)  
16    ax = sns.countplot(data= df, x= col, order= df[col].value_counts().index)  
17  
18    plt.show()
```

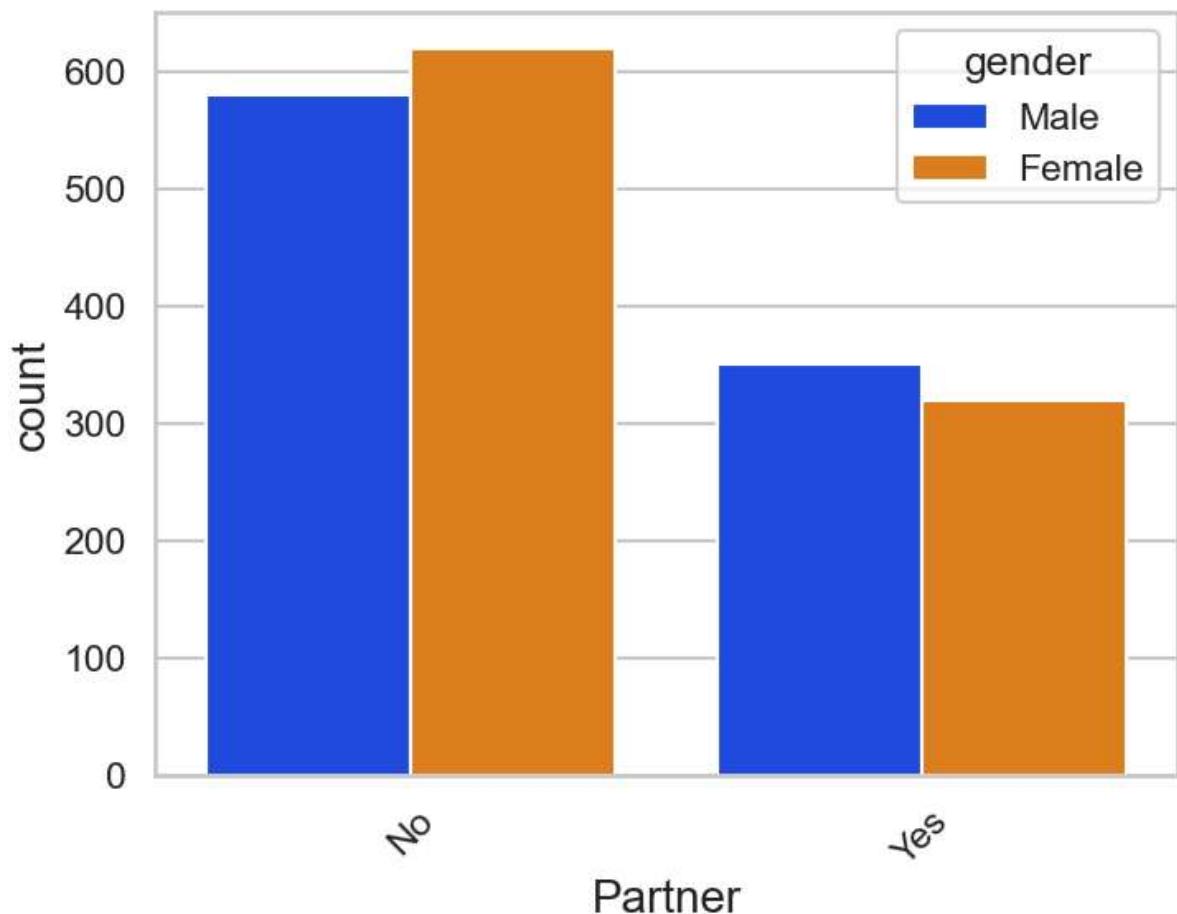
```
In [48]: 1 df.nunique()
```

```
Out[48]: customerID      7043  
gender            2  
SeniorCitizen     2  
Partner           2  
Dependents        2  
tenure            73  
PhoneService       2  
MultipleLines      3  
InternetService    3  
OnlineSecurity     3  
OnlineBackup        3  
DeviceProtection   3  
TechSupport         3  
StreamingTV        3  
StreamingMovies    3  
Contract           3  
PaperlessBilling   2  
PaymentMethod      4  
MonthlyCharges     1585  
TotalCharges       6530  
Churn              2  
Churn_encoded      2  
dtype: int64
```

In [49]:

```
1 # Distribution of gender for Churned customers  
2 uniplot(df_target1, col= 'Partner', title= 'Distribution of gender for chu
```

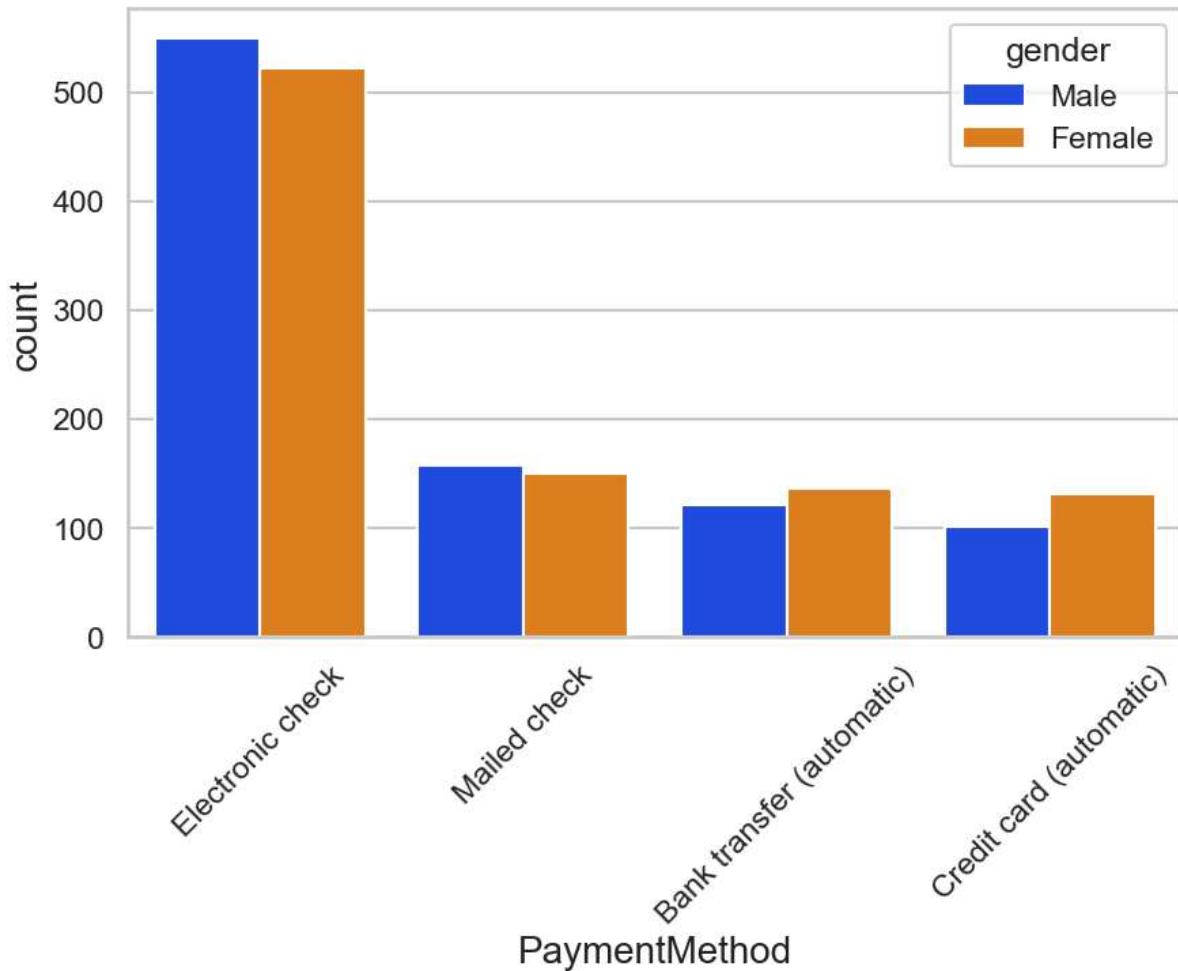
## Distribution of gender for churned customers



In [50]:

```
1 # Distribution of payment method for Churned customers
2 uniplot(df_target1, col= 'PaymentMethod', title= 'Distribution of payment')
```

Distribution of payment method for Churned Customers



- Customers who use Electronic check form the highest churners however, the difference in terms of gender is not much
- Credit card users are the least churners with females forming the highest portion

```
In [69]: 1 for i, column in enumerate(df.columns):  
2     print(df[column].value_counts())
```

```
7590-VHVEG      1
3791-LGQCY      1
6008-NAIXK      1
5956-YHHRX      1
5365-LLFYV      1
...
9796-MVYXX      1
2637-FKFSY      1
1552-AAGRX      1
4304-TSPVK      1
3186-AJIEK      1
Name: customerID, Length: 7043, dtype: int64
Male      3555
Female     3488
Name: gender, dtype: int64
0      5901
1      1142
Name: SeniorCitizen, dtype: int64
No      3641
Yes     3402
Name: Partner, dtype: int64
No      4933
Yes     2110
Name: Dependents, dtype: int64
1      613
72     362
2      238
3      200
4      176
...
28      57
39      56
44      51
36      50
0       11
Name: tenure, Length: 73, dtype: int64
Yes     6361
No      682
Name: PhoneService, dtype: int64
No      4072
Yes     2971
Name: MultipleLines, dtype: int64
Fiber optic    3096
DSL          2421
No           1526
Name: InternetService, dtype: int64
No            3498
Yes           2019
No internet service  1526
Name: OnlineSecurity, dtype: int64
No            3088
Yes           2429
No internet service  1526
Name: OnlineBackup, dtype: int64
No            3095
Yes           2422
No internet service  1526
```



```
In [67]: 1 df['MultipleLines'].replace({'No phone service': 'No'}, inplace=True)
```

```
In [68]: 1 df['MultipleLines'].value_counts()
```

```
Out[68]: No      4072  
Yes     2971  
Name: MultipleLines, dtype: int64
```

## Prediction of Customer Churn

```
In [73]: 1 df_1 = df.drop('Churn_encoded', axis=1)  
2 df_1.head()
```

```
Out[73]:   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  MultipleLines  
0      7590-VHVEG  Female           0       Yes        No         1        No          No  
1      5575-GNVDE  Male            0       No        No        34        Yes          No  
2      3668-QPYBK  Male            0       No        No         2        Yes          No  
3      7795-CFOCW  Male            0       No        No        45        No          No  
4      9237-HQITU  Female           0       No        No         2        Yes          No
```

5 rows × 21 columns



```
In [74]: 1 df_1.head()
```

```
Out[74]:   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  MultipleLines  
0      7590-VHVEG  Female           0       Yes        No         1        No          No  
1      5575-GNVDE  Male            0       No        No        34        Yes          No  
2      3668-QPYBK  Male            0       No        No         2        Yes          No  
3      7795-CFOCW  Male            0       No        No        45        No          No  
4      9237-HQITU  Female           0       No        No         2        Yes          No
```

5 rows × 21 columns



```
In [75]: 1 df.columns
```

```
Out[75]: Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',  
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',  
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',  
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',  
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn',  
       'Churn_encoded'],  
      dtype='object')
```

```
In [77]: 1 # Separating dataframe into two dataframes to enable proper encoding  
2 df_pred_1 = df_1[['customerID', 'tenure', 'MonthlyCharges', 'TotalCharges'  
3 df_pred_1.head()
```

```
Out[77]:   customerID  tenure  MonthlyCharges  TotalCharges  
0  7590-VHVEG        1        29.85        29.85  
1  5575-GNVDE       34        56.95     1889.50  
2  3668-QPYBK        2        53.85        108.15  
3  7795-CFOCW       45        42.30     1840.75  
4  9237-HQITU        2        70.70        151.65
```

```
In [78]: 1 df_pred_3 = df_1.drop(columns= ['customerID', 'tenure', 'MonthlyCharges',  
2 df_pred_3.head()
```

```
Out[78]:   gender  SeniorCitizen  Partner  Dependents  PhoneService  MultipleLines  InternetService  OnlineSecurity  StreamingTV  StreamingMovies  Contract  PaperlessBilling  PaymentMethod  
0  Female            0       Yes        No          No          No          No          No          DSL  
1  Male              0       No        No          No         Yes          No          No          DSL  
2  Male              0       No        No          No          Yes          No          No          DSL  
3  Male              0       No        No          No          No          No          No          DSL  
4  Female            0       No        No          No          Yes          No  Fiber optic
```

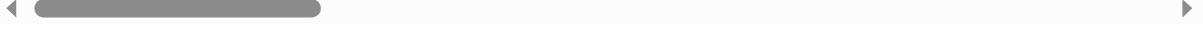
In [79]:

```
1 # # Encoding Categorical variables
2 df_pred_2 = pd.get_dummies(df_pred_3, drop_first=True)
3 df_pred_2
```

Out[79]:

	SeniorCitizen	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_
0	0	0	1	0	0	0
1	0	1	0	0	0	1
2	0	1	0	0	0	1
3	0	1	0	0	0	0
4	0	0	0	0	0	1
...	...	...	...	...	...	...
7038	0	1	1	1	1	1
7039	0	0	1	1	1	1
7040	0	0	1	1	1	0
7041	1	1	1	0	0	1
7042	0	1	0	0	0	1

7043 rows × 27 columns



```
In [83]: 1 # Merging the cumerical features and the encoded categorical features
2 df_pred = pd.concat([df_pred_1, df_pred_2], axis= 'columns')
3 df_pred
```

Out[83]:

	customerID	tenure	MonthlyCharges	TotalCharges	SeniorCitizen	gender_Male	Partner_Yes	Churn_Yes
0	7590-VHVEG	1	29.85	29.85	0	0	0	0
1	5575-GNVDE	34	56.95	1889.50	0	1	0	0
2	3668-QPYBK	2	53.85	108.15	0	1	0	0
3	7795-CFOCW	45	42.30	1840.75	0	1	0	0
4	9237-HQITU	2	70.70	151.65	0	0	0	0
...	...	...	...	...	...	...	...	...
7038	6840-RESVB	24	84.80	1990.50	0	1	0	0
7039	2234-XADUH	72	103.20	7362.90	0	0	0	0
7040	4801-JZAZL	11	29.60	346.45	0	0	0	0
7041	8361-LTMKD	4	74.40	306.60	1	1	0	0
7042	3186-AJIEK	66	105.65	6844.50	0	1	0	0

7043 rows × 31 columns



```
In [106]: 1 # Model training
2 from xgboost import XGBClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5 from sklearn.model_selection import GridSearchCV
6 from sklearn import metrics
```

```
In [95]: 1 x = df_pred.drop(['Churn_Yes', 'customerID'], axis= 1)
```

In [96]: 1 print(x)

	tenure	MonthlyCharges	TotalCharges	SeniorCitizen	gender_Male	\
0	1	29.85	29.85	0	0	
1	34	56.95	1889.50	0	1	
2	2	53.85	108.15	0	1	
3	45	42.30	1840.75	0	1	
4	2	70.70	151.65	0	0	
...	...	...	...	...	...	...
7038	24	84.80	1990.50	0	1	
7039	72	103.20	7362.90	0	0	
7040	11	29.60	346.45	0	0	
7041	4	74.40	306.60	1	1	
7042	66	105.65	6844.50	0	1	
	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_Yes		\
0	1	0	0	0	0	
1	0	0	1	0	0	
2	0	0	1	0	0	
3	0	0	0	0	0	
4	0	0	1	0	0	
...	...	...	...	...	...	...
7038	1	1	1	1	1	
7039	1	1	1	1	1	
7040	1	1	0	0	0	
7041	1	0	1	1	1	
7042	0	0	1	0	0	
	InternetService_Fiber optic	...	StreamingTV_No internet service			\
0		0	...	0	0	
1		0	...	0	0	
2		0	...	0	0	
3		0	...	0	0	
4		1	...	0	0	
...		...	...	...	...	...
7038		0	...	0	0	
7039		1	...	0	0	
7040		0	...	0	0	
7041		1	...	0	0	
7042		1	...	0	0	
	StreamingTV_Yes	StreamingMovies_No internet service				\
0	0		0	0	0	
1	0		0	0	0	
2	0		0	0	0	
3	0		0	0	0	
4	0		0	0	0	
...	...		...	...	...	...
7038	1		0	0	0	
7039	1		0	0	0	
7040	0		0	0	0	
7041	0		0	0	0	
7042	1		0	0	0	
	StreamingMovies_Yes	Contract_One year	Contract_Two year			\
0	0	0	0	0	0	
1	0	1	0	0	0	
2	0	0	0	0	0	
3	0	1	0	0	0	

```

4          0          0          0
...
7038      ...      ...
7039      1          1          0
7040      1          1          0
7041      0          0          0
7042      0          0          0
7042      1          0          1

    PaperlessBilling_Yes PaymentMethod_Credit card (automatic) \
0                  1                      0
1                  0                      0
2                  1                      0
3                  0                      0
4                  1                      0
...
7038      ...      ...
7039      1                      1
7040      1                      0
7041      1                      0
7042      1                      0

    PaymentMethod_Electronic check PaymentMethod_Mailed check
0                  1                      0
1                  0                      1
2                  0                      1
3                  0                      0
4                  1                      0
...
7038      ...      ...
7039      0                      0
7040      0                      1
7041      1                      0
7042      0                      1

[7043 rows x 29 columns]

```

In [97]: 1 y = df\_pred['Churn\_Yes']

In [98]: 1 # Splitting data into training data and testing data  
2 x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size= 0.2, r

In [111]: 1 # Hyperparameter tuning  
2 model = XGBClassifier()  
3 parameters = {'n\_estimators':[100,200, 300],  
4 'max\_depth':[10,50, 75],  
5 'learning\_rate': [0.005, 0.05, 0.1]  
6  
7 }

In [112]: 1 tune = GridSearchCV(model, parameters)

```
In [113]: 1 tune.get_params
```

```
Out[113]: <bound method BaseEstimator.get_params of GridSearchCV(estimator=XGBClassifier(base_score=None, booster=None,  
 callbacks=None, colsample_bylevel=None,  
 colsample_bynode=None,  
 colsample_bytree=None, device=None,  
 early_stopping_rounds=None,  
 enable_categorical=False, eval_metric=None  
 ne,  
 feature_types=None, gamma=None,  
 grow_policy=None, importance_type=None,  
 interaction_constraints=None,  
 learning_rate=None, max_b...  
 max_cat_threshold=None,  
 max_cat_to_onehot=None,  
 max_delta_step=None, max_depth=None,  
 max_leaves=None, min_child_weight=None,  
 missing=nan, monotone_constraints=None,  
 multi_strategy=None, n_estimators=None,  
 n_jobs=None, num_parallel_tree=None,  
 random_state=None, ...),  
 param_grid={'learning_rate': [0.005, 0.05, 0.1],  
 'max_depth': [10, 50, 75],  
 'n_estimators': [100, 200, 300]})>
```

```
In [114]: 1 tune.fit(x_train, y_train)
```

```
Out[114]: GridSearchCV(estimator=XGBClassifier(base_score=None, booster=None,  
 callbacks=None, colsample_bylevel=None,  
 colsample_bynode=None,  
 colsample_bytree=None, device=None,  
 early_stopping_rounds=None,  
 enable_categorical=False, eval_metric=None  
 ne,  
 feature_types=None, gamma=None,  
 grow_policy=None, importance_type=None,  
 interaction_constraints=None,  
 learning_rate=None, max_b...  
 max_cat_threshold=None,  
 max_cat_to_onehot=None,  
 max_delta_step=None, max_depth=None,  
 max_leaves=None, min_child_weight=None,  
 missing=nan, monotone_constraints=None,  
 multi_strategy=None, n_estimators=None,  
 n_jobs=None, num_parallel_tree=None,  
 random_state=None, ...),  
 param_grid={'learning_rate': [0.005, 0.05, 0.1],  
 'max_depth': [10, 50, 75],  
 'n_estimators': [100, 200, 300]})
```

```
In [115]: 1 tune.best_params_
```

```
Out[115]: {'learning_rate': 0.005, 'max_depth': 10, 'n_estimators': 300}
```

```
In [116]: 1 tune.best_score_
```

```
Out[116]: 0.7909130166855529
```

```
In [128]: 1 test_pred = tune.predict(x_test)
2 Accuracy = accuracy_score(y_test, test_pred)
3 print('Testing data Accuracy is: ', round(Accuracy*100, 2), '%')
```

```
Testing data Accuracy is: 79.06 %
```

- The model is 79% percent accurate in predicting customer churning rate

```
In [129]: 1 train_pred = tune.predict(x_train)
2 accuracy = accuracy_score(y_train, train_pred)
3 print('Training data Accuracy is: ', round(accuracy*100, 2), '%')
```

```
Training data Accuracy is: 86.88 %
```

## Conclusion

- Key factors that affect churn rate are:
  - Customer care quality
  - Cost of Service
  - Network quality
- It is advised that:
  - Electronic payment method is improved/reviewed since it forms the highest churners.
  - Technical support system should be enhanced.
  - Attractive packages should be offered to short-term subscription customers to convert them to long-term. This will maintain them and reduce their churning.
  - Service packages should be offered to customers at highly competitive prices as possible.