CUSTOMER CHURN PREDICTION | TELECOM DATASET

Content

- Logistic Regression

1. Introduction

What is Customer Churn?

Customer churn is defined as when customers or subscribers discontinue doing business with a firm or service.

Customers in the telecom industry can choose from a variety of service providers and actively switch from one to the next. The telecommunications business has an annual churn rate of 15-25 percent in this highly competitive market.

Individualized customer retention is tough because most firms have a large number of customers and can't afford to devote much time to each of them. The costs would be too great, outweighing the additional revenue. However, if a corporation could forecast which customers are likely to leave ahead of time, it could focus customer retention efforts only on these "high risk" clients. The ultimate goal is to expand its coverage area and retrieve more customers loyalty. The core to succeed in this market lies in the customer itself.

Customer churn is a critical metric because it is much less expensive to retain existing customers than it is to acquire new customers.

To reduce customer churn, telecom companies need to predict which customers are at high risk of churn.

To detect early signs of potential churn, one must first develop a holistic view of the customers and their interactions across numerous channels, including store/branch visits, product purchase histories, customer service calls, Web-based transactions, and social media interactions, to mention a few.

As a result, by addressing churn, these businesses may not only preserve their market position, but also grow and thrive. More customers they have in their network, the lower the cost of initiation and the larger the profit. As a result, the company's key focus for success is reducing client attrition and implementing effective retention strategy.

Objectives

I will explore the data and try to answer some questions like:

- What's the % of Churn Customers and customers that keep in with the active services?
- Is there any patterns in Churn Customers based on the gender?
- Is there any patterns/preference in Churn Customers based on the type of service provided?
- What's the most profitable service types?
- Which features and services are most profitable?
- Many more questions that will arise during the analysis

---

2. Loading libraries and data

In [115…

```
import sys
!{sys.executable} -m pip install missingno
import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: missingno in /opt/anaconda3/lib/python3.12/site-
packages (0.5.2)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-pack
ages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.12/sit
e-packages (from missingno) (3.9.2)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.12/site-pack
ages (from missingno) (1.13.1)
Requirement already satisfied: seaborn in /opt/anaconda3/lib/python3.12/site-pa
ckages (from missingno) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python3.1
2/site-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.12/si
te-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib->missingno) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python3.1
2/site-packages (from matplotlib->missingno) (24.1)
Requirement already satisfied: pillow>=8 in /opt/anaconda3/lib/python3.12/site-
packages (from matplotlib->missingno) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python3.1
2/site-packages (from matplotlib->missingno) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from matplotlib->missingno) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in /opt/anaconda3/lib/python3.12/sit
e-packages (from seaborn->missingno) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/lib/python3.12/si
te-packages (from pandas>=1.2->seaborn->missingno) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/anaconda3/lib/python3.12/
site-packages (from pandas>=1.2->seaborn->missingno) (2023.3)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/site-p
ackages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
```

In [116…

```python
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f
```

In [117…

```python
df = pd.read_csv('/Users/avijit/Desktop/APR_assignment01/WA_Fn-UseC_-Telco-Cus
```

---

3. Undertanding the data

Each row represents a customer, each column contains customer's attributes described on the column Metadata.

In [61]:
```
df.head()
```

Out[61]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneServ
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-CFOCW	Male	0	No	No	45	
4	9237-HQITU	Female	0	No	No	2	

5 rows × 21 columns

The data set includes information about:

- **Customers who left within the last month** – the column is called Churn
- **Services that each customer has signed up for** – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- **Customer account information** - how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- **Demographic info about customers** – gender, age range, and if they have partners and dependents

In [62]:
```
df.shape
```

Out[62]:
```
(7043, 21)
```

In [63]:
```
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 [64]:
```
df.columns.values
```

Out[64]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [65]:
```
df.dtypes
```

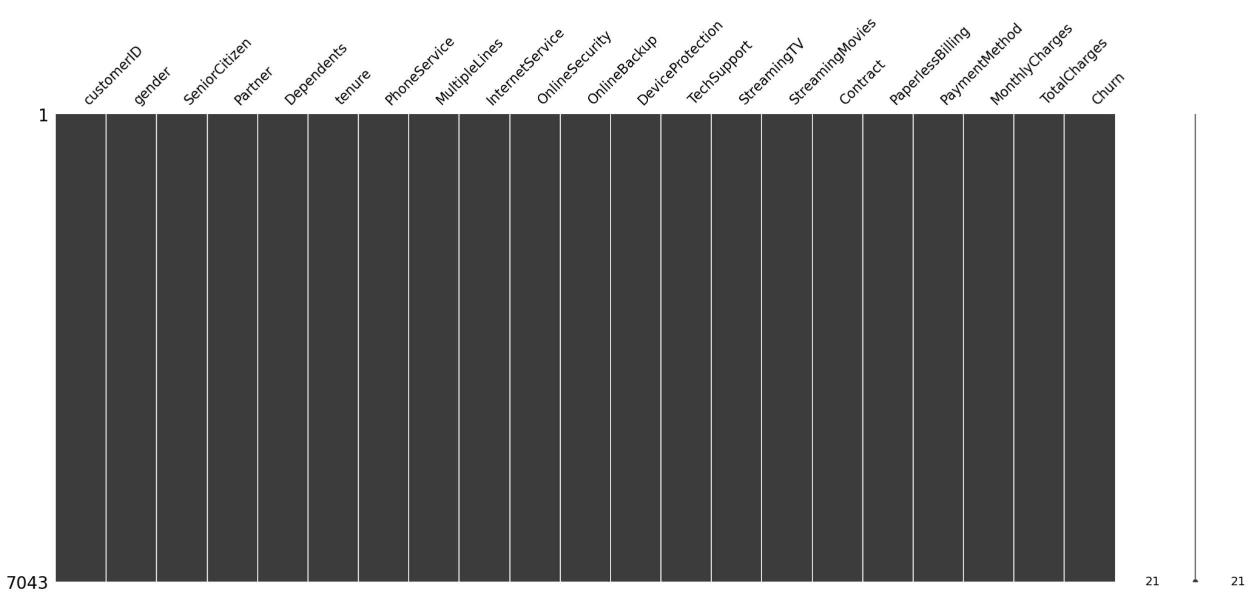
Out[65]:
```
customerID          object
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges        object
Churn               object
dtype: object
```

- The target the we will use to guide the exploration is **Churn**

---

4. Visualize missing values

In [66]:
```
# Visualize missing values as a matrix
msno.matrix(df);
```



> Using this matrix we can very quickly find the pattern of missingness in the dataset.

- From the above visualisation we can observe that it has no peculiar pattern that stands out. In fact there is no missing data.

---

5. Data Manipulation

In [67]:
```
df = df.drop(['customerID'], axis = 1)
df.head()
```

Out[67]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleL
0	Female	0	Yes	No	1	No	No p se
1	Male	0	No	No	34	Yes	
2	Male	0	No	No	2	Yes	
3	Male	0	No	No	45	No	No p se
4	Female	0	No	No	2	Yes	

- On deep analysis, we can find some indirect missingness in our data (which can be in form of blankspaces). Let's see that!

In [68]:
```
df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()
```

Out[68]:
```
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
```

- Here we see that the TotalCharges has 11 missing values. Let's check this data.

In [69]:
```
df[np.isnan(df['TotalCharges'])]
```

Out[69]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multi
488	Female	0	Yes	Yes	0	No	N
753	Male	0	No	Yes	0	Yes	
936	Female	0	Yes	Yes	0	Yes	
1082	Male	0	Yes	Yes	0	Yes	
1340	Female	0	Yes	Yes	0	No	N
3331	Male	0	Yes	Yes	0	Yes	
3826	Male	0	Yes	Yes	0	Yes	
4380	Female	0	Yes	Yes	0	Yes	
5218	Male	0	Yes	Yes	0	Yes	
6670	Female	0	Yes	Yes	0	Yes	
6754	Male	0	No	Yes	0	Yes	

- It can also be noted that the Tenure column is 0 for these entries even though the MonthlyCharges column is not empty.

Let's see if there are any other 0 values in the tenure column.

In [70]:
```python
df[df['tenure'] == 0].index
```

Out[70]:
```
Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754], dtyp
e='int64')
```

- There are no additional missing values in the Tenure column.

Let's delete the rows with missing values in Tenure columns since there are only 11 rows and deleting them will not affect the data.

In [71]:
```python
df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)
df[df['tenure'] == 0].index
```

Out[71]:
```
Index([], dtype='int64')
```

> To solve the problem of missing values in TotalCharges column, I decided to fill it with the mean of TotalCharges values.

In [113…
```
df.fillna(df["TotalCharges"].mean())
```

Out[113…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multi
0	0	0	1	0	1	0	
1	1	0	0	0	34	1	
2	1	0	0	0	2	1	
3	1	0	0	0	45	0	
4	0	0	0	0	2	1	
...	...	...	...	...	...	...	
7038	1	0	1	1	24	1	
7039	0	0	1	1	72	1	
7040	0	0	1	1	11	0	
7041	1	1	1	0	4	1	
7042	1	0	0	0	66	1	

7032 rows × 20 columns

In [73]:
```
df.isnull().sum()
```

Out[73]:
```
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 [74]:
```
df["SeniorCitizen"]= df["SeniorCitizen"].map({0: "No", 1: "Yes"})
df.head()
```

Out[74]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleL
0	Female	No	Yes	No	1	No	No p se
1	Male	No	No	No	34	Yes	
2	Male	No	No	No	2	Yes	
3	Male	No	No	No	45	No	No p se
4	Female	No	No	No	2	Yes	

In [75]:
```
df["InternetService"].describe(include=['object', 'bool'])
```

Out[75]:
```
count              7032
unique                3
top         Fiber optic
freq               3096
Name: InternetService, dtype: object
```

In [76]:
```
numerical_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
df[numerical_cols].describe()
```

Out[76]:

	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000
mean	32.421786	64.798208	2283.300441
std	24.545260	30.085974	2266.771362
min	1.000000	18.250000	18.800000
25%	9.000000	35.587500	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.862500	3794.737500
max	72.000000	118.750000	8684.800000

---

6. Data Visualization

In [77]:
```
g_labels = ['Male', 'Female']
```

```
c_labels = ['No', 'Yes']
# Create subplots: use 'domain' type for Pie subplot
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain
fig.add_trace(go.Pie(labels=g_labels, values=df['gender'].value_counts(), name
              1, 1)
fig.add_trace(go.Pie(labels=c_labels, values=df['Churn'].value_counts(), name=
              1, 2)

# Use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent+name", textfont_size=16)

fig.update_layout(
    title_text="Gender and Churn Distributions",
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='Gender', x=0.16, y=0.5, font_size=20, showarrow=Fa
                 dict(text='Churn', x=0.84, y=0.5, font_size=20, showarrow=Fal
fig.show()
```

- 26.6 % of customers switched to another firm.
- Customers are 49.5 % female and 50.5 % male.

In [78]:
```
df["Churn"][df["Churn"]=="No"].groupby(by=df["gender"]).count()
```

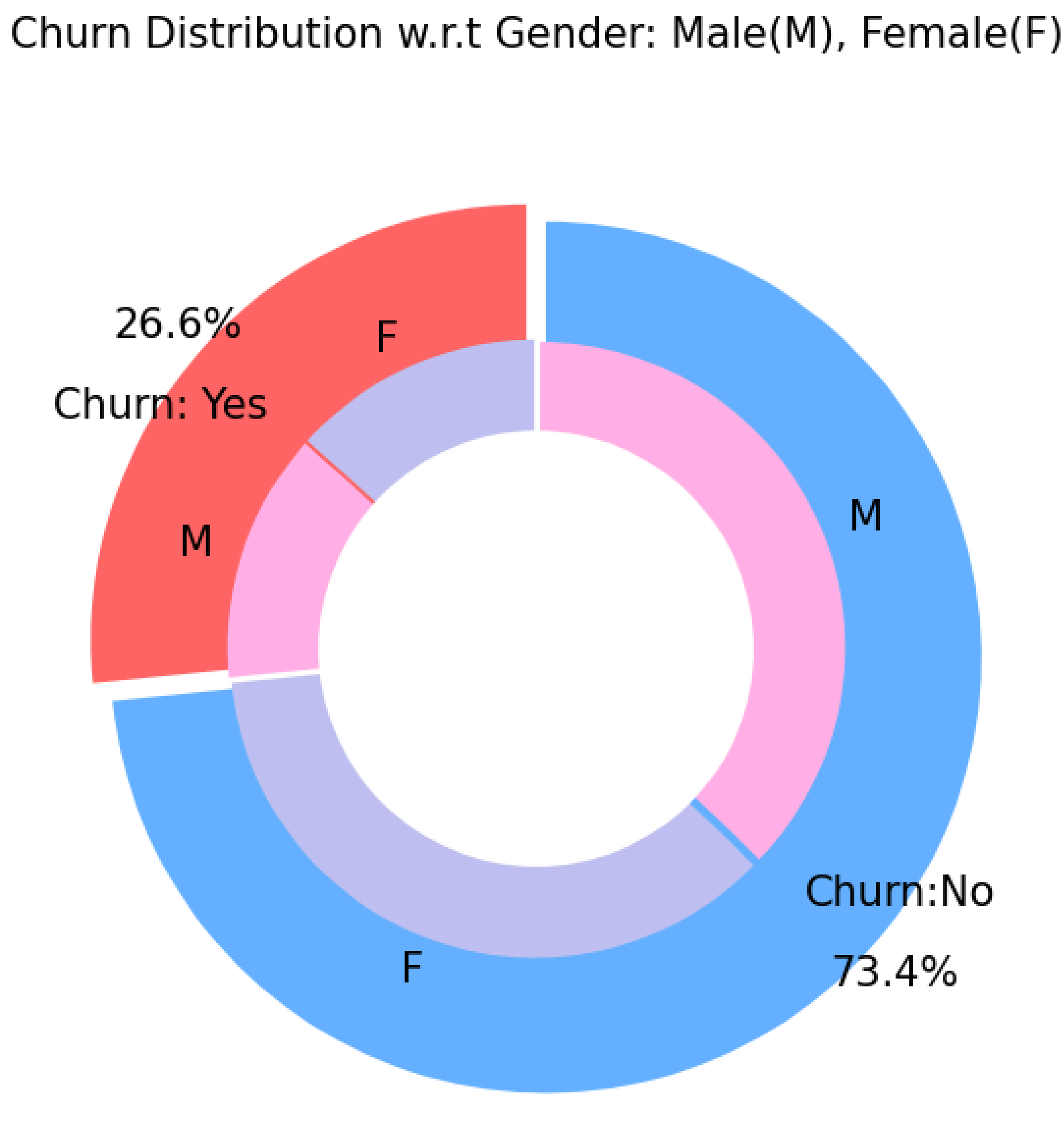
Out[78]:
```
gender
Female    2544
Male      2619
Name: Churn, dtype: int64
```

In [79]:
```
df["Churn"][df["Churn"]=="Yes"].groupby(by=df["gender"]).count()
```

Out[79]:
```
gender
Female    939
Male      930
Name: Churn, dtype: int64
```

In [80]:
```
plt.figure(figsize=(6, 6))
labels =["Churn: Yes","Churn:No"]
values = [1869,5163]
labels_gender = ["F","M","F","M"]
sizes_gender = [939,930 , 2544,2619]
colors = ['#ff6666', '#66b3ff']
colors_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']
explode = (0.3,0.3)
explode_gender = (0.1,0.1,0.1,0.1)
textprops = {"fontsize":15}
#Plot
plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08, labeldistanc
plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startangle=90,
#Draw circle
centre_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)
fig = plt.gcf()
```

```
fig.gca().add_artist(centre_circle)

plt.title('Churn Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15,

# show plot

plt.axis('equal')
plt.tight_layout()
plt.show()
```



- There is negligible difference in customer percentage/ count who chnaged the service provider. Both genders behaved in similar fashion when it comes to migrating to another service provider/firm.

In [81]:
```
fig = px.histogram(df, x="Churn", color="Contract", barmode="group", title="<b
fig.update_layout(width=700, height=500, bargap=0.1)
```

```
fig.show()
```

- About 75% of customer with Month-to-Month Contract opted to move out as compared to 13% of customrs with One Year Contract and 3% with Two Year Contract

In [82]:
```
labels = df['PaymentMethod'].unique()
values = df['PaymentMethod'].value_counts()

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.update_layout(title_text="<b>Payment Method Distribution</b>")
fig.show()
```

In [83]:
```
fig = px.histogram(df, x="Churn", color="PaymentMethod", title="<b>Customer Pa
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

- Major customers who moved out were having Electronic Check as Payment Method.
- Customers who opted for Credit-Card automatic transfer or Bank Automatic Transfer and Mailed Check as Payment Method were less likely to move out.

In [84]:
```
df["InternetService"].unique()
```

Out[84]:
```
array(['DSL', 'Fiber optic', 'No'], dtype=object)
```

In [85]:
```
df[df["gender"]=="Male"][["InternetService", "Churn"]].value_counts()
```

Out[85]:
```
InternetService  Churn
DSL              No       992
Fiber optic      No       910
No               No       717
Fiber optic      Yes      633
DSL              Yes      240
No               Yes       57
Name: count, dtype: int64
```

In [86]:
```
df[df["gender"]=="Female"][["InternetService", "Churn"]].value_counts()
```

```
Out[86]: InternetService  Churn
         DSL              No       965
         Fiber optic      No       889
         No               No       690
         Fiber optic      Yes      664
         DSL              Yes      219
         No               Yes       56
         Name: count, dtype: int64
```

```
In [87]: fig = go.Figure()

         fig.add_trace(go.Bar(
           x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
                ["Female", "Male", "Female", "Male"]],
           y = [965, 992, 219, 240],
           name = 'DSL',
         ))

         fig.add_trace(go.Bar(
           x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
                ["Female", "Male", "Female", "Male"]],
           y = [889, 910, 664, 633],
           name = 'Fiber optic',
         ))

         fig.add_trace(go.Bar(
           x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
                ["Female", "Male", "Female", "Male"]],
           y = [690, 717, 56, 57],
           name = 'No Internet',
         ))

         fig.update_layout(title_text="<b>Churn Distribution w.r.t. Internet Service ar

         fig.show()
```

- A lot of customers choose the Fiber optic service and it's also evident that the customers who use Fiber optic have high churn rate, this might suggest a dissatisfaction with this type of internet service.
- Customers having DSL service are majority in number and have less churn rate compared to Fibre optic service.

```
In [88]: color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
         fig = px.histogram(df, x="Churn", color="Dependents", barmode="group", title="
         fig.update_layout(width=700, height=500, bargap=0.1)
         fig.show()
```

- Customers without dependents are more likely to churn

In [89]:
```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="Partner", barmode="group", title="<b>
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

- Customers that doesn't have partners are more likely to churn

In [90]:
```
color_map = {"Yes": '#00CC96', "No": '#B6E880'}
fig = px.histogram(df, x="Churn", color="SeniorCitizen", title="<b>Chrun distr
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

- It can be observed that the fraction of senior citizen is very less.
- Most of the senior citizens churn.

In [91]:
```
color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
fig = px.histogram(df, x="Churn", color="OnlineSecurity", barmode="group", tit
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

- Most customers churn in the absence of online security,

In [92]:
```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="PaperlessBilling",  title="<b>Chrun c
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

- Customers with Paperless Billing are most likely to churn.

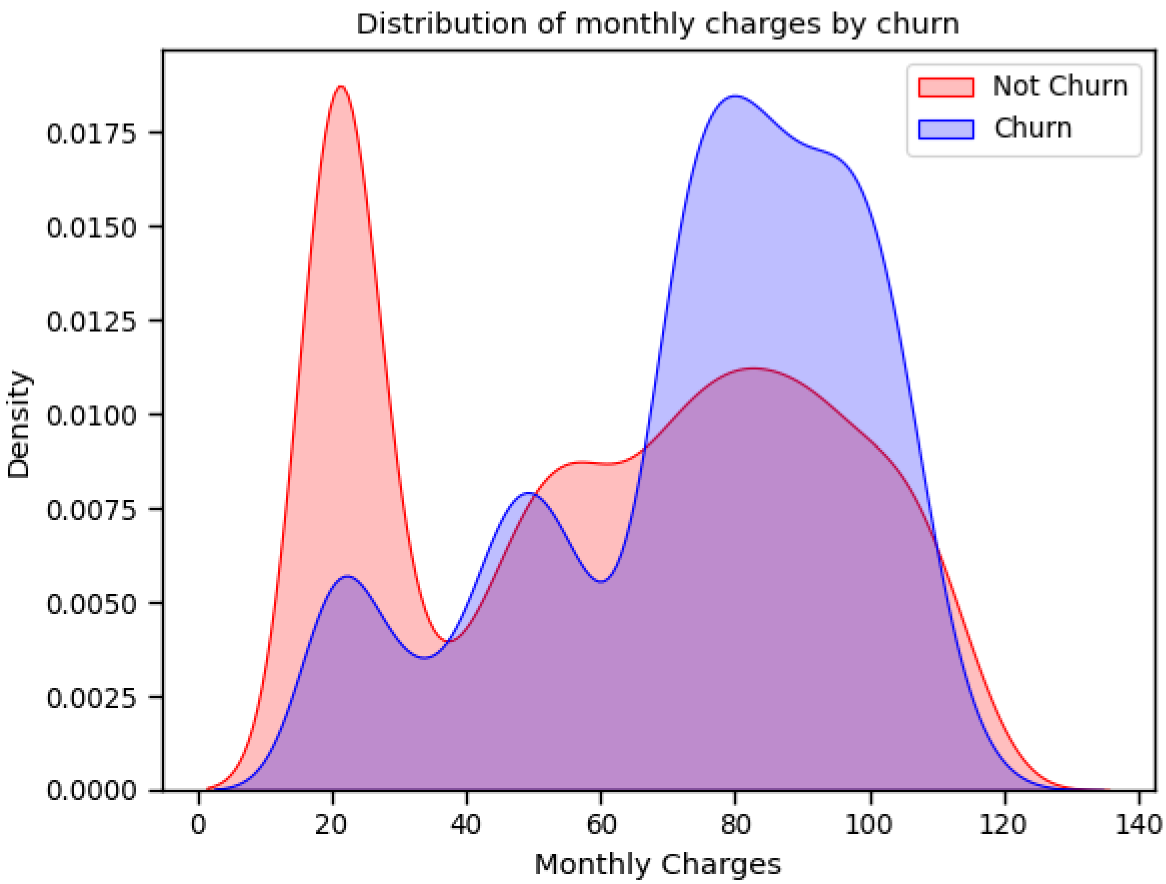
In [93]:
```
fig = px.histogram(df, x="Churn", color="TechSupport",barmode="group",  title=
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

- Customers with no TechSupport are most likely to migrate to another service provider.

In [94]:
```
color_map = {"Yes": '#00CC96', "No": '#B6E880'}
fig = px.histogram(df, x="Churn", color="PhoneService", title="<b>Chrun distri
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

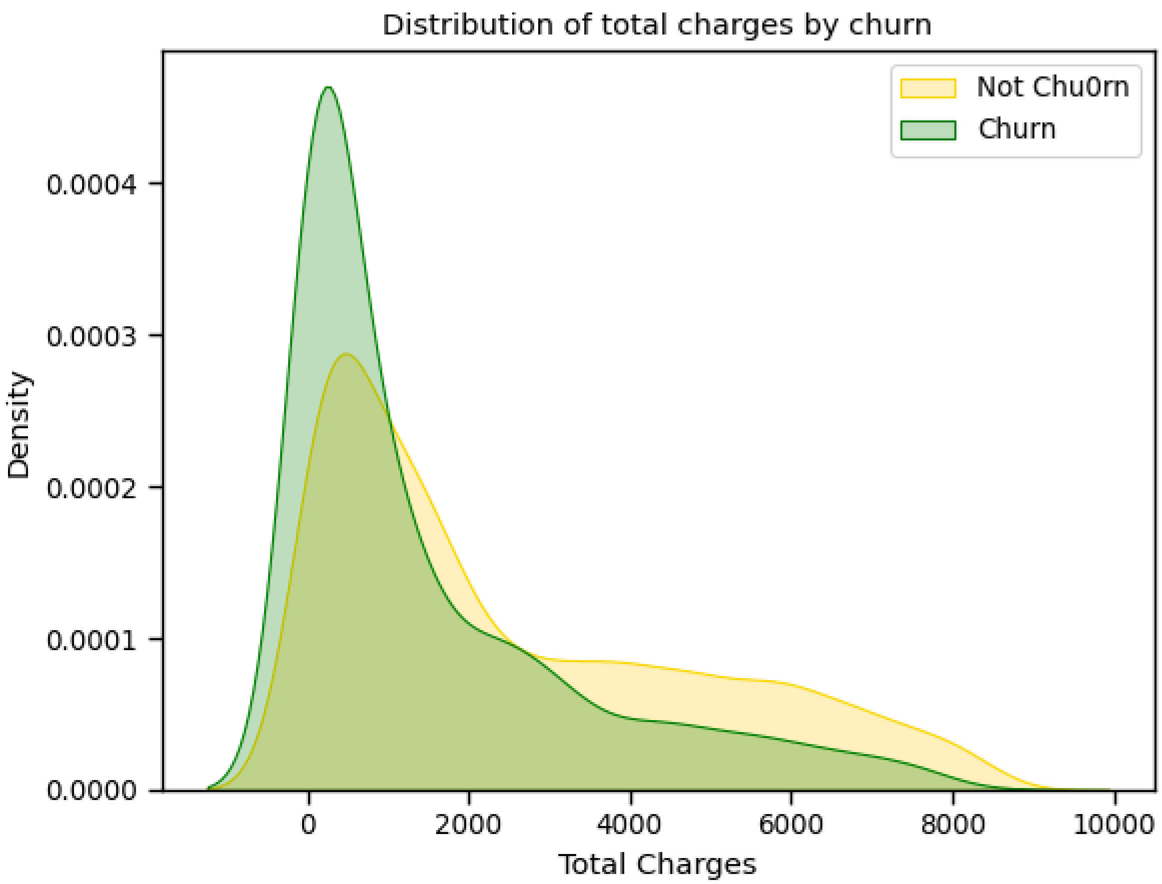
- Very small fraction of customers don't have a phone service and out of that, 1/3rd Customers are more likely to churn.

In [95]:
```
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'No') ],
                color="Red", shade = True);
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],
                ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Monthly Charges');
ax.set_title('Distribution of monthly charges by churn');
```



- Customers with higher Monthly Charges are also more likely to churn

In [96]:
```
ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'No') ],
                color="Gold", shade = True);
ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'Yes') ],
                ax =ax, color="Green", shade= True);
ax.legend(["Not Chu0rn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Total Charges');
ax.set_title('Distribution of total charges by churn');
```



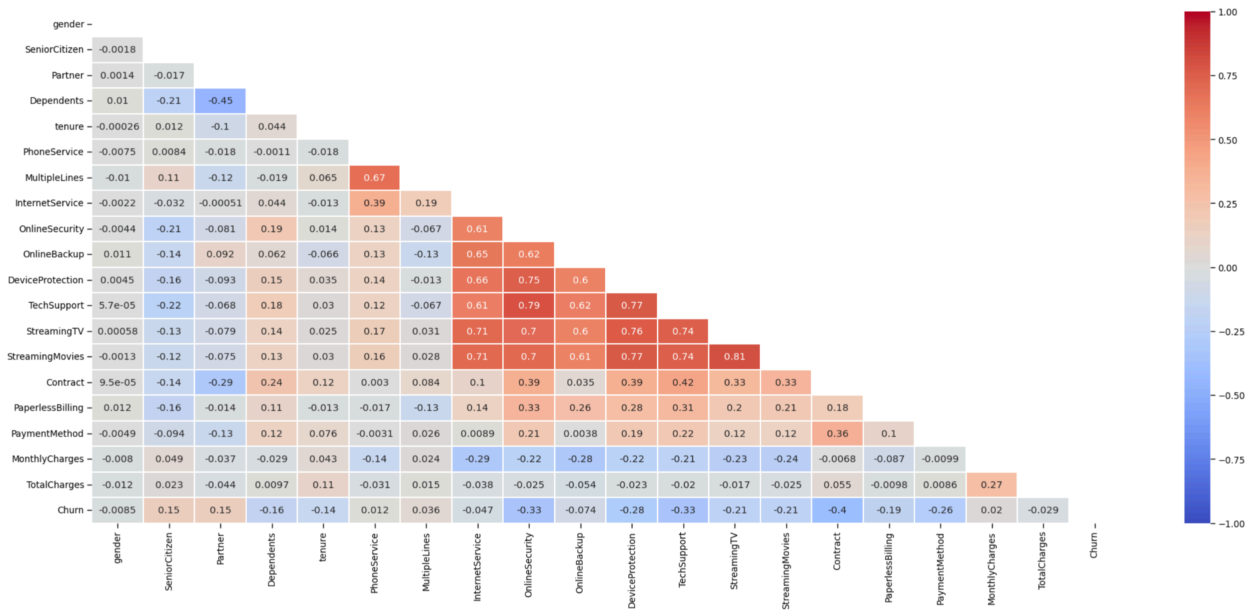
In [97]:

```
fig = px.box(df, x='Churn', y = 'tenure')

# Update yaxis properties
fig.update_yaxes(title_text='Tenure (Months)', row=1, col=1)
# Update xaxis properties
fig.update_xaxes(title_text='Churn', row=1, col=1)

# Update size and title
fig.update_layout(autosize=True, width=750, height=600,
    title_font=dict(size=25, family='Courier'),
    title='<b>Tenure vs Churn</b>',
)

fig.show()
```

- New customers are more likely to churn

In [98]:

```
plt.figure(figsize=(25, 10))

corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.c
```



7. Data Preprocessing

Splitting the data into train and test sets

In [99]:
```
def object_to_int(dataframe_series):
    if dataframe_series.dtype=='object':
        dataframe_series = LabelEncoder().fit_transform(dataframe_series)
    return dataframe_series
```

In [100…
```
df = df.apply(lambda x: object_to_int(x))
df.head()
```

Out[100…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleL
0	0	0	1	0	1	0	
1	1	0	0	0	34	1	
2	1	0	0	0	2	1	
3	1	0	0	0	45	0	
4	0	0	0	0	2	1	

In [101…
```
plt.figure(figsize=(14,7))
df.corr()['Churn'].sort_values(ascending = False)
```

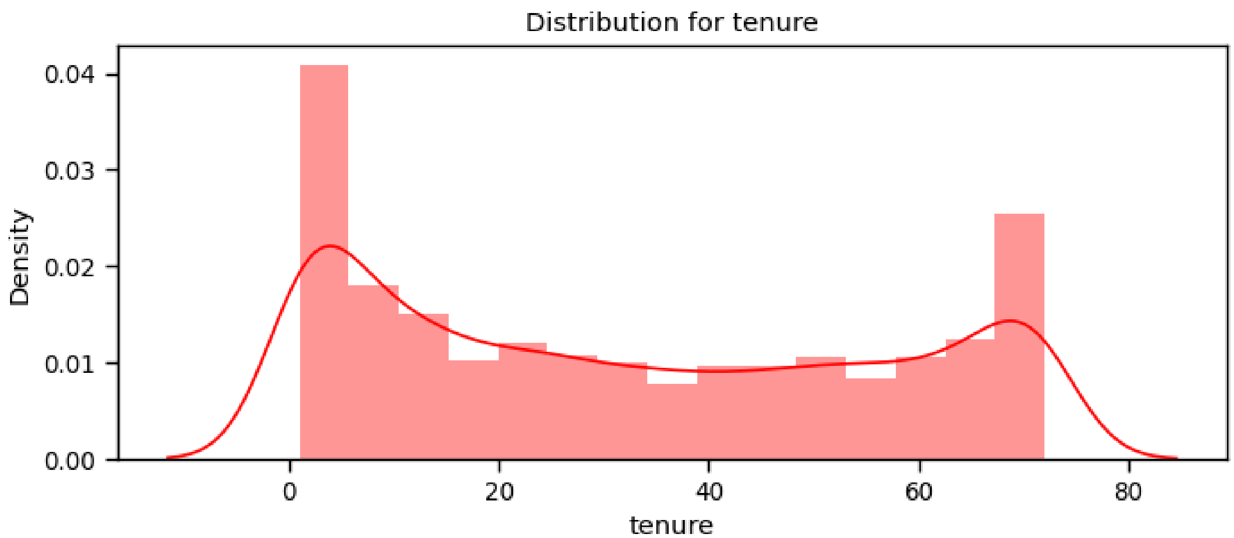
```
Out[101… Churn               1.000000
         MonthlyCharges      0.192858
         PaperlessBilling    0.191454
         SeniorCitizen       0.150541
         PaymentMethod       0.107852
         MultipleLines       0.038043
         PhoneService        0.011691
         gender             -0.008545
         StreamingTV        -0.036303
         StreamingMovies    -0.038802
         InternetService    -0.047097
         Partner            -0.149982
         Dependents         -0.163128
         DeviceProtection   -0.177883
         OnlineBackup       -0.195290
         TotalCharges       -0.199484
         TechSupport        -0.282232
         OnlineSecurity     -0.289050
         tenure             -0.354049
         Contract           -0.396150
         Name: Churn, dtype: float64
        <Figure size 1400x700 with 0 Axes>
```

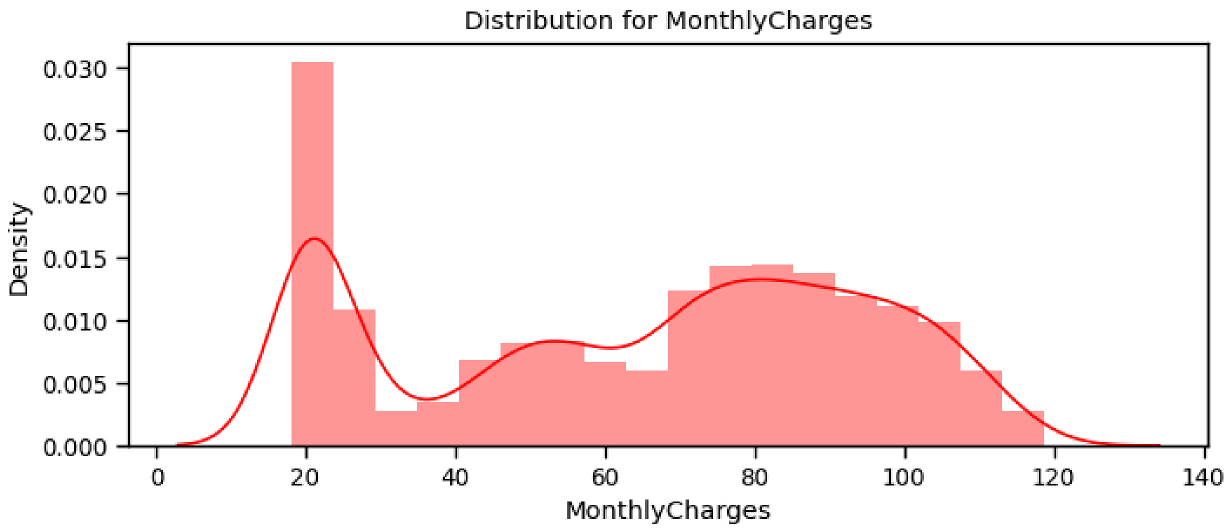
In [102…
```
X = df.drop(columns = ['Churn'])
y = df['Churn'].values
```

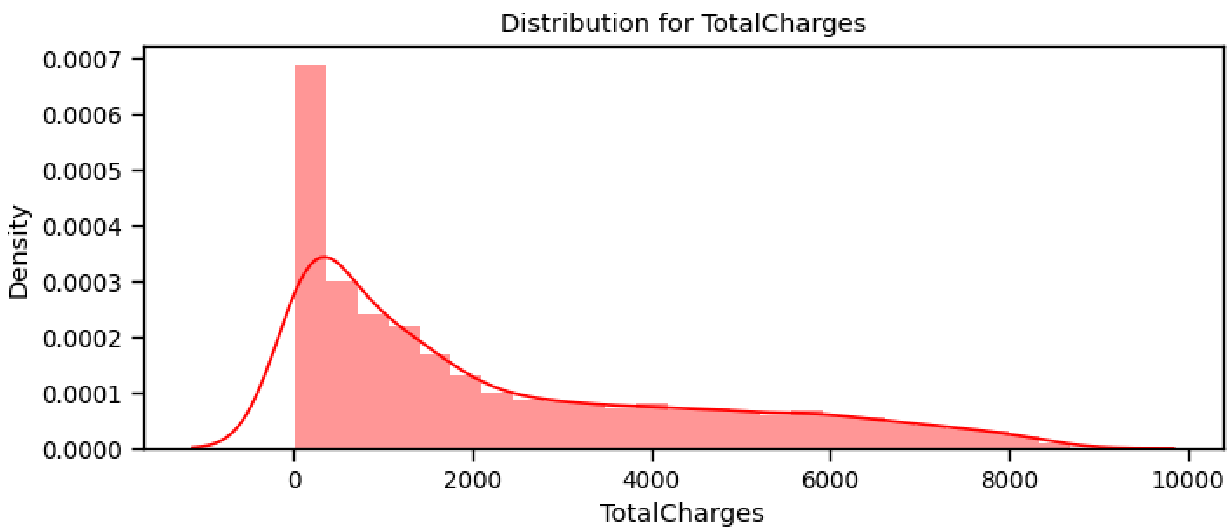
In [103…
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30, ranc
```

In [104…
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

In [105…
```
num_cols = ["tenure", 'MonthlyCharges', 'TotalCharges']
for feat in num_cols: distplot(feat, df)
```





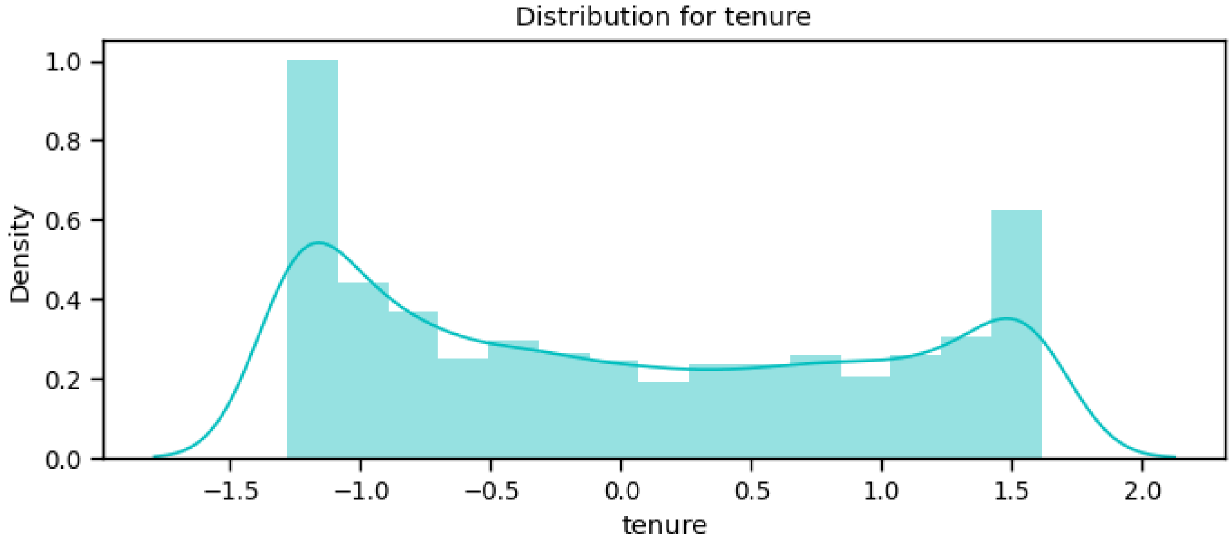


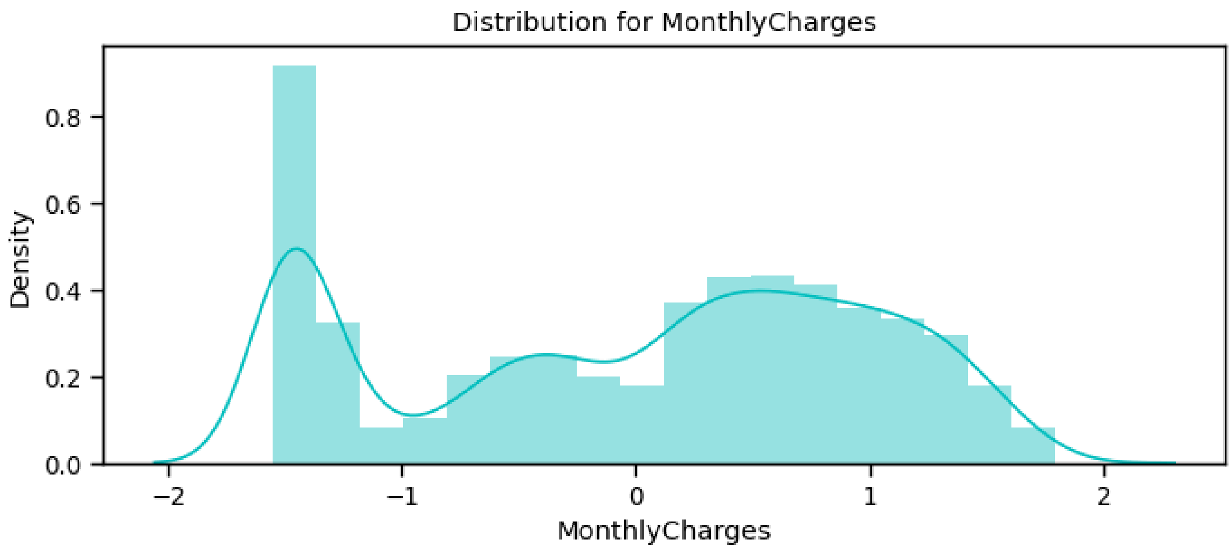
Since the numerical features are distributed over different value ranges, I will use standard scalar to scale them down to the same range.

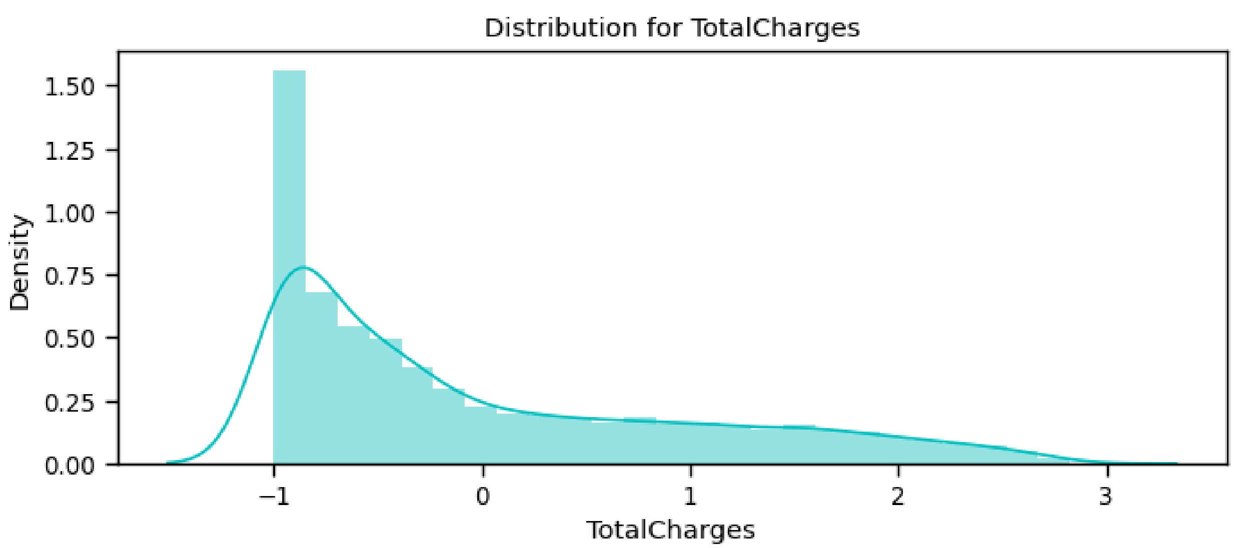
Standardizing numeric attributes

In [106…

```
df_std = pd.DataFrame(StandardScaler().fit_transform(df[num_cols].astype('floa
                       columns=num_cols)
for feat in numerical_cols: distplot(feat, df_std, color='c')
```







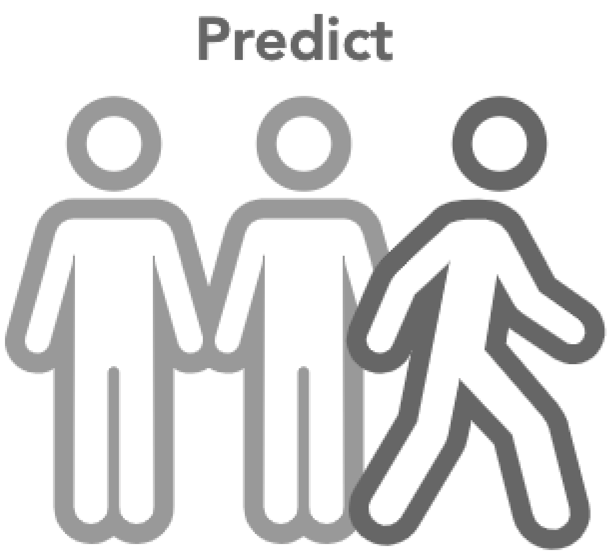
In [107…

```
# Divide the columns into 3 categories, one ofor standardisation, one for labe
```

```
cat_cols_ohe =['PaymentMethod', 'Contract', 'InternetService'] # those that ne
cat_cols_le = list(set(X_train.columns)- set(num_cols) - set(cat_cols_ohe)) #t
```

In [108…
```
scaler= StandardScaler()

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])
```

8. Machine Learning Model Evaluations and Predictions



Logistic Regression

In [109…
```
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy is :",accuracy_lr)
```

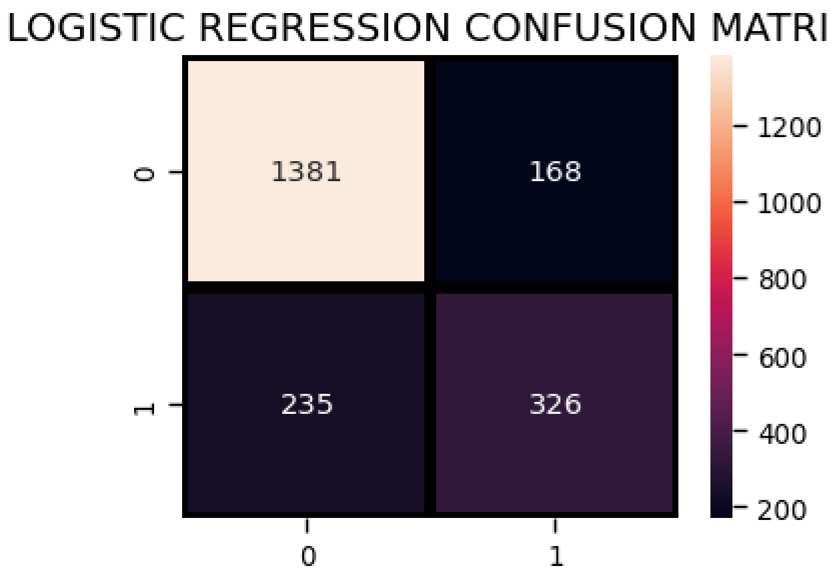
```
Logistic Regression accuracy is : 0.8090047393364929
```

In [110…
```
lr_pred= lr_model.predict(X_test)
report = classification_report(y_test,lr_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1549
           1       0.66      0.58      0.62       561

    accuracy                           0.81      2110
   macro avg       0.76      0.74      0.75      2110
weighted avg       0.80      0.81      0.80      2110
```

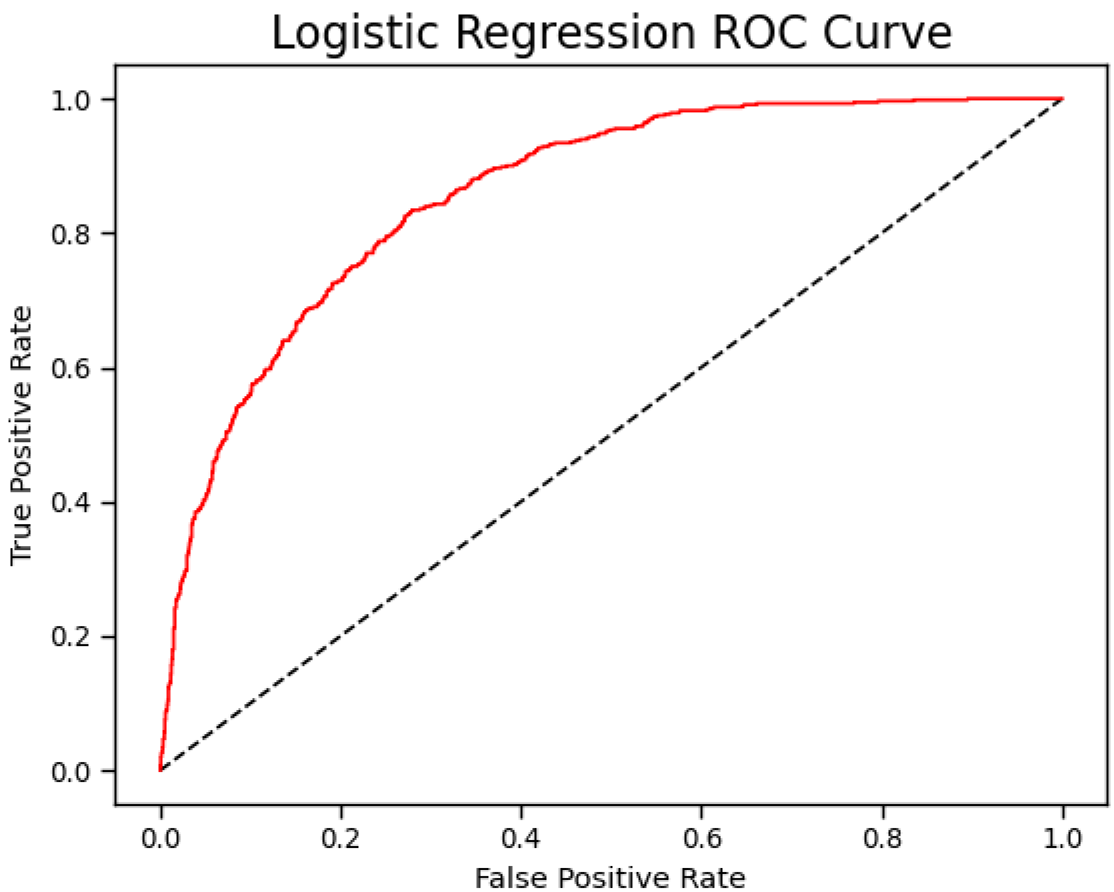
In [111…

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, lr_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("LOGISTIC REGRESSION CONFUSION MATRIX",fontsize=14)
plt.show()
```



In [112…

```
y_pred_prob = lr_model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```



9. Output and Conclusion

From the confusion matrix we can see that: There are a total of 1549 actual non-churn values (1381 predicted as non-churn + 168 predicted as churn).

The algorithm predicts 1381 of them correctly as non-churn. The algorithm incorrectly predicts 168 as churn.

There are a total of 561 actual churn values (235 predicted as non-churn + 326 predicted as churn). The algorithm predicts 326 of them correctly as churn. The algorithm incorrectly predicts 235 as non-churn.

This breakdown reflects the prediction performance visualized in the confusion matrix.

Customer churn is definitely bad to a firm ’s profitability. Various strategies can be implemented to eliminate customer churn. The best way to avoid customer churn is for a company to truly know its customers. This includes identifying customers who are at risk of churning and working to improve their satisfaction. Improving customer service is, of course, at the top of the priority for tackling this issue. Building customer loyalty through relevant experiences and specialized service is another strategy to reduce customer churn. Some firms survey customers who have already churned to understand their reasons for leaving in order to adopt a proactive approach to avoiding future customer churn.

THANK YOU