Student: Bimal Kandel

In this project, students will develop a predictive model for estimating the customer lifetime value

(CLV) of an e-commerce business. They will start by acquiring and preprocessing data using Python and SQL, extracting relevant customer and transactional data from the company's database. With the help of Pandas, students will perform advanced data analysis and manipulation to derive features for CLV prediction, such as customer demographics, purchase frequency, and monetary value.

They will then utilize statistical analysis techniques to gain insights into customer behavior and calculate probabilistic metrics related to customer retention and churn. The ultimate goal is to build a robust predictive model that can accurately forecast the CLV for individual customers, thereby enabling the business to optimize marketing strategies, customer acquisition efforts, and overall revenue generation.

1. The preprocessed dataset containing relevant customer and transactional data
2. Exploratory data analysis (EDA) report highlighting key insights and patterns in the data
3. Trained predictive model for estimating customer churn
4. Evaluation metrics and performance analysis of the churn prediction model
5. Interactive dashboard showcasing churn metrics and insights derived from the predictive model, enabling stakeholders to make data-driven decisions and optimize business strategies for maximizing customer retention.

```
pip install missingno
```

```
Collecting missingno
  Downloading missingno-0.5.2-py3-none-any.whl.metadata (639 bytes)
Requirement already satisfied: numpy in
/opt/anaconda3/lib/python3.11/site-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in
/opt/anaconda3/lib/python3.11/site-packages (from missingno) (3.8.0)
Requirement already satisfied: scipy in
/opt/anaconda3/lib/python3.11/site-packages (from missingno) (1.11.4)
Requirement already satisfied: seaborn in
/opt/anaconda3/lib/python3.11/site-packages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
```

```
>missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (23.1)
Requirement already satisfied: pillow>=6.2.0 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (2.8.2)
Requirement already satisfied: pandas>=0.25 in
/opt/anaconda3/lib/python3.11/site-packages (from seaborn->missingno)
(2.1.4)
Requirement already satisfied: pytz>=2020.1 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas>=0.25-
>seaborn->missingno) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas>=0.25-
>seaborn->missingno) (2023.3)
Requirement already satisfied: six>=1.5 in
/opt/anaconda3/lib/python3.11/site-packages (from python-
dateutil>=2.7->matplotlib->missingno) (1.16.0)
Downloading missingno-0.5.2-py3-none-any.whl (8.7 kB)
Installing collected packages: missingno
Successfully installed missingno-0.5.2
Note: you may need to restart the kernel to use updated packages.
```

```python
# importing necessary libraries

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')
```

1. Data Acquisition and Preprocessing

```python
# Extracting data in CSV file
df = pd.read_csv("//users/iambimalk/Downloads/WA_Fn-UseC_-Telco-
Customer-Churn.csv")
```

```
data

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

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

     DeviceProtection TechSupport StreamingTV StreamingMovies
Contract  \
0                  No          No          No              No  Month-
to-month
1                 Yes          No          No              No
One year
2                  No          No          No              No  Month-
to-month
3                 Yes         Yes          No              No
One year
```

```
4                    No          No          No              No  Month-
to-month
...                 ...         ...         ...             ...
...
7038                Yes         Yes         Yes             Yes
One year
7039                Yes          No         Yes             Yes
One year
7040                 No          No          No              No  Month-
to-month
7041                 No          No          No              No  Month-
to-month
7042                Yes         Yes         Yes             Yes
Two year

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

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
...    ...
7038    No
7039    No
```

```
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]

df.head()

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

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
Yes
2                No             DSL            Yes  ...
No
3  No phone service             DSL            Yes  ...
Yes
4                No     Fiber optic             No  ...
No

  TechSupport StreamingTV StreamingMovies        Contract
PaperlessBilling  \
0          No          No              No  Month-to-month
Yes
1          No          No              No        One year
No
2          No          No              No  Month-to-month
Yes
3         Yes          No              No        One year
No
4          No          No              No  Month-to-month
Yes

              PaymentMethod MonthlyCharges  TotalCharges Churn
0          Electronic check          29.85         29.85    No
1              Mailed check          56.95        1889.5    No
```

```
2                Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4            Electronic check          70.70        151.65   Yes

[5 rows x 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

```
df.shape

(7043, 21)

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
```

```
df.columns.values
```

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

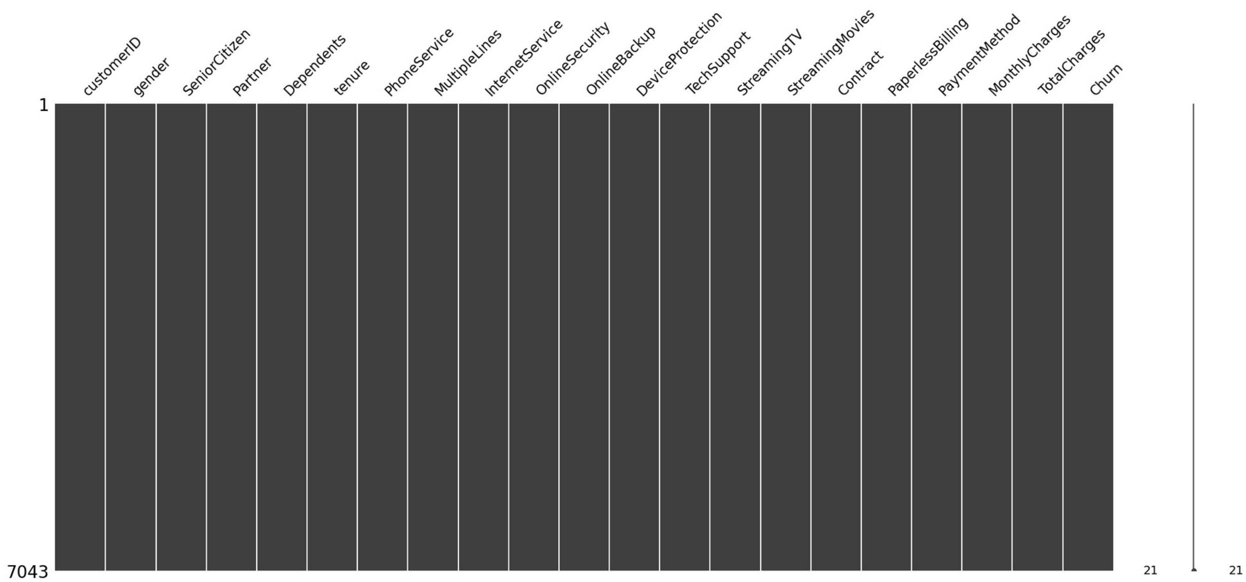
```
df.dtypes
```

```
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 variable we would like to manipulate and explore is Churn

Visualize missing values

```
msno.matrix(df);
```



From above visualisation in fact there is no missing data.

Data Manipulation

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

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

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No
2                No             DSL            Yes          Yes
3  No phone service             DSL            Yes           No
4                No     Fiber optic             No           No

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

```
month

  PaperlessBilling              PaymentMethod  MonthlyCharges
TotalCharges  \
0              Yes           Electronic check           29.85
29.85
1               No               Mailed check           56.95
1889.5
2              Yes               Mailed check           53.85
108.15
3               No  Bank transfer (automatic)           42.30
1840.75
4              Yes           Electronic check           70.70
151.65

  Churn
0    No
1    No
2   Yes
3    No
4   Yes
```

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

```
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.

```
df[np.isnan(df['TotalCharges'])]
```

```
      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
488   Female              0     Yes        Yes       0           No
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
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

         MultipleLines InternetService       OnlineSecurity  \
488   No phone service             DSL                  Yes
753                 No              No  No internet service
936                 No             DSL                  Yes
1082               Yes              No  No internet service
1340  No phone service             DSL                  Yes
3331                No              No  No internet service
3826               Yes              No  No internet service
4380                No              No  No internet service
5218                No              No  No internet service
6670               Yes             DSL                   No
6754               Yes             DSL                  Yes

             OnlineBackup     DeviceProtection          TechSupport  \
488                    No                  Yes                  Yes
753   No internet service  No internet service  No internet service
936                   Yes                  Yes                   No
1082  No internet service  No internet service  No internet service
1340                  Yes                  Yes                  Yes
3331  No internet service  No internet service  No internet service
3826  No internet service  No internet service  No internet service
4380  No internet service  No internet service  No internet service
5218  No internet service  No internet service  No internet service
6670                  Yes                  Yes                  Yes
6754                  Yes                   No                  Yes

              StreamingTV      StreamingMovies  Contract
PaperlessBilling  \
488                   Yes                   No  Two year
Yes
753   No internet service  No internet service  Two year
No
936                   Yes                  Yes  Two year
No
1082  No internet service  No internet service  Two year
No
1340                  Yes                   No  Two year
```

```
No
3331  No internet service  No internet service  Two year
No
3826  No internet service  No internet service  Two year
No
4380  No internet service  No internet service  Two year
No
5218  No internet service  No internet service  One year
Yes
6670                  Yes                   No  Two year
No
6754                   No                   No  Two year
Yes

                  PaymentMethod  MonthlyCharges  TotalCharges Churn
488   Bank transfer (automatic)           52.55           NaN    No
753                Mailed check           20.25           NaN    No
936                Mailed check           80.85           NaN    No
1082               Mailed check           25.75           NaN    No
1340    Credit card (automatic)           56.05           NaN    No
3331               Mailed check           19.85           NaN    No
3826               Mailed check           25.35           NaN    No
4380               Mailed check           20.00           NaN    No
5218               Mailed check           19.70           NaN    No
6670               Mailed check           73.35           NaN    No
6754  Bank transfer (automatic)           61.90           NaN    No
```

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.

```
df[df['tenure'] == 0].index

Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754],
dtype='int64')
```

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.

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

Index([], dtype='int64')

df.fillna(df["TotalCharges"].mean())

      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0     Female              0     Yes         No       1           No
1       Male              0      No         No      34          Yes
2       Male              0      No         No       2          Yes
3       Male              0      No         No      45           No
```

```
4     Female              0      No         No       2          Yes
...      ...            ...     ...        ...     ...          ...
7038    Male              0     Yes        Yes      24          Yes
7039  Female              0     Yes        Yes      72          Yes
7040  Female              0     Yes        Yes      11           No
7041    Male              1     Yes         No       4          Yes
7042    Male              0      No         No      66          Yes

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

     DeviceProtection TechSupport StreamingTV StreamingMovies
Contract  \
0                  No          No          No              No  Month-
to-month
1                 Yes          No          No              No
One year
2                  No          No          No              No  Month-
to-month
3                 Yes         Yes          No              No
One year
4                  No          No          No              No  Month-
to-month
...               ...         ...         ...             ...
...
7038              Yes         Yes         Yes             Yes
One year
7039              Yes          No         Yes             Yes
One year
7040               No          No          No              No  Month-
to-month
7041               No          No          No              No  Month-
to-month
7042              Yes         Yes         Yes             Yes
Two year

     PaperlessBilling              PaymentMethod  MonthlyCharges  \
0                 Yes           Electronic check           29.85
1                  No               Mailed check           56.95
2                 Yes               Mailed check           53.85
```

```
3                    No  Bank transfer (automatic)           42.30
4                   Yes           Electronic check           70.70
...                 ...                        ...             ...
7038                Yes               Mailed check           84.80
7039                Yes    Credit card (automatic)          103.20
7040                Yes           Electronic check           29.60
7041                Yes               Mailed check           74.40
7042                Yes  Bank transfer (automatic)          105.65

      TotalCharges Churn
0            29.85    No
1          1889.50    No
2           108.15   Yes
3          1840.75    No
4           151.65   Yes
...            ...   ...
7038       1990.50    No
7039       7362.90    No
7040        346.45    No
7041        306.60   Yes
7042       6844.50    No

[7032 rows x 20 columns]
```

```
df.isnull().sum()
```

```
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
```

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

```
   gender SeniorCitizen Partner Dependents  tenure PhoneService  \
0  Female            No     Yes         No       1           No
1    Male            No      No         No      34          Yes
2    Male            No      No         No       2          Yes
3    Male            No      No         No      45           No
4  Female            No      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No
2                No             DSL            Yes          Yes
3  No phone service             DSL            Yes           No
4                No     Fiber optic             No           No

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

  PaperlessBilling              PaymentMethod  MonthlyCharges
TotalCharges  \
0              Yes           Electronic check           29.85
29.85
1               No               Mailed check           56.95
1889.50
2              Yes               Mailed check           53.85
108.15
3               No  Bank transfer (automatic)           42.30
1840.75
4              Yes           Electronic check           70.70
151.65

  Churn
0    No
1    No
2   Yes
3    No
4   Yes
```

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

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

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

```
            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
```

Data Visualisation

```
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="Gender"),
              1, 1)
fig.add_trace(go.Pie(labels=c_labels,
values=df['Churn'].value_counts(), name="Churn"),
              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=False),
                 dict(text='Churn', x=0.84, y=0.5, font_size=20,
showarrow=False)])

fig.show()
```

A customer churn rate of 26.6% was observed. The customer base comprises 49.5% females and 50.5% males."

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

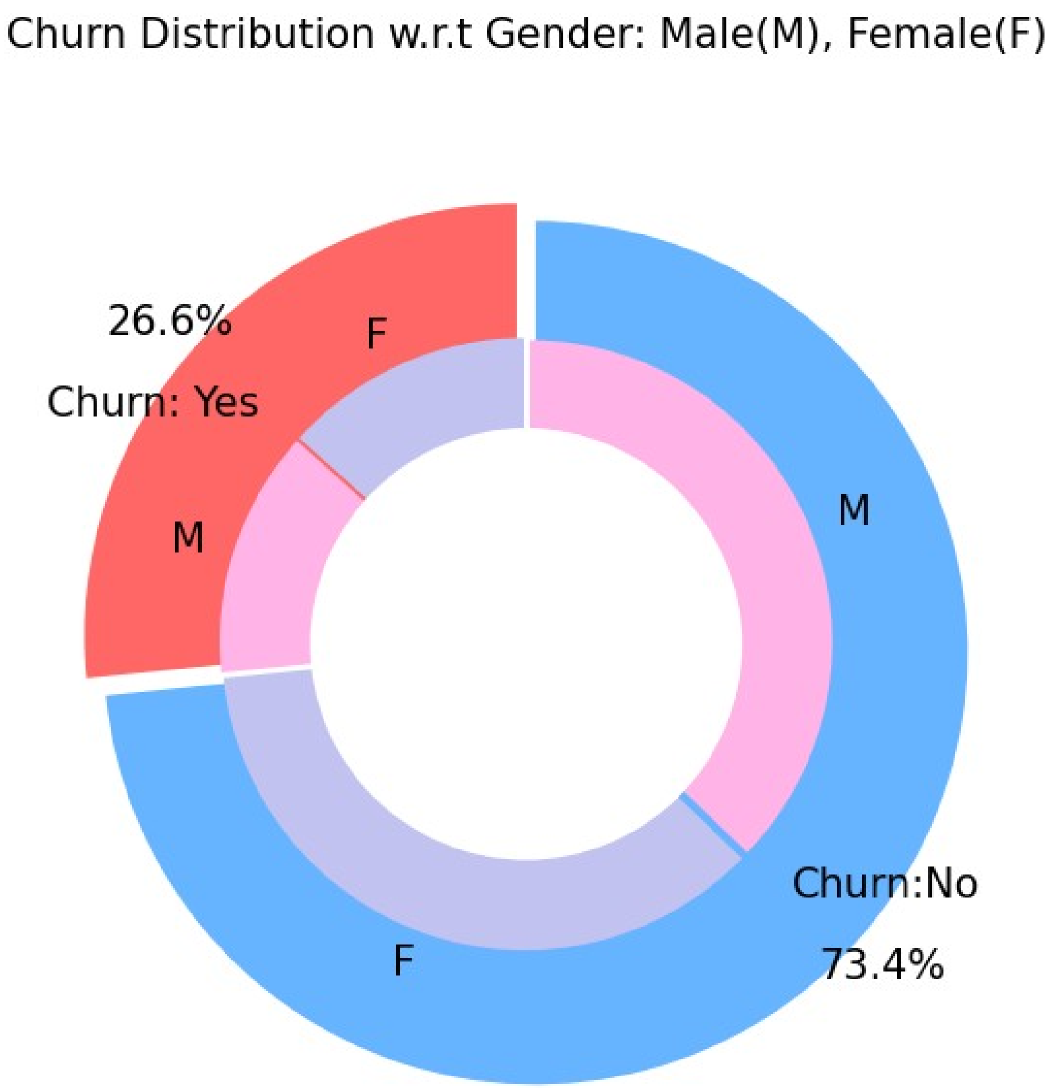
```
gender
Female    2544
Male      2619
Name: Churn, dtype: int64
```

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

```
gender
Female    939
Male      930
Name: Churn, dtype: int64
```

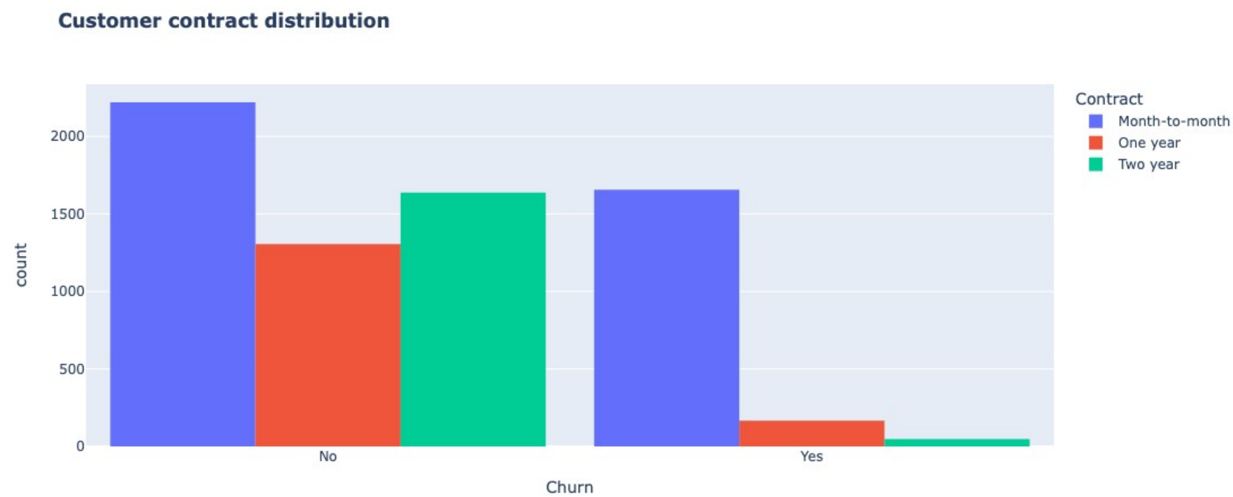
```
plt.figure(figsize=(6, 6))
labels =["Churn: Yes","Churn:No"]
values = [1869,5163]
labels_gender = ["F","M","F","M"]
sizes_gender = [939,930 , 2544,2619]
colors = ['#ff6666', '#66b3ff']
colors_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']
explode = (0.3,0.3)
explode_gender = (0.1,0.1,0.1,0.1)
textprops = {"fontsize":15}
#Plot
plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08,
labeldistance=0.8,colors=colors, startangle=90,frame=True,
explode=explode,radius=10, textprops =textprops, counterclock =
True, )
plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startan
gle=90, explode=explode_gender,radius=7, textprops =textprops,
counterclock = True, )
#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, y=1.1)

# show plot

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



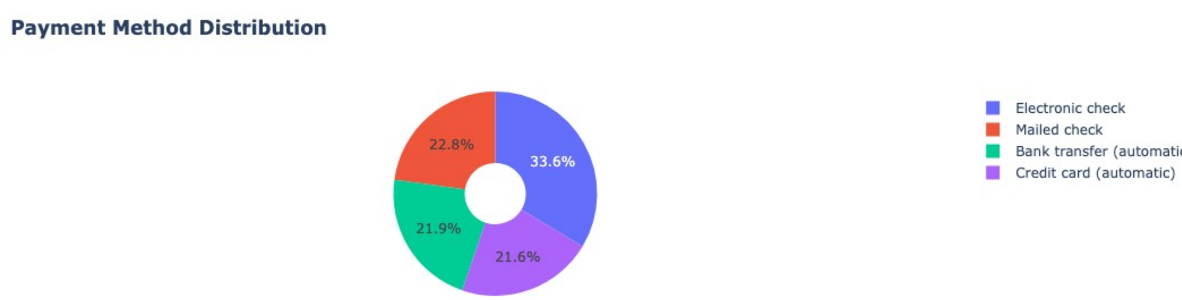
The proportion of customers switching service providers was roughly the same for both genders, indicating no significant gender-based difference in customer churn

```
fig = px.histogram(df, x="Churn", color="Contract", barmode="group",
title="<b>Customer contract distribution<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



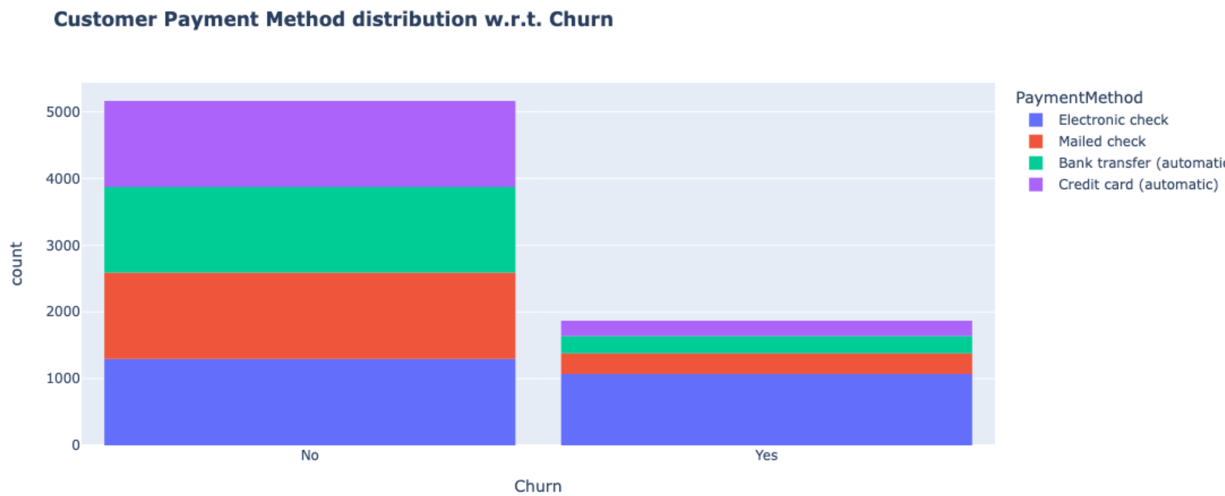
Customers with Month-to-Month contracts exhibited the highest churn rate at approximately 75%, significantly higher than those with One-Year contracts (13%) and Two-Year contracts (3%)

```
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()
```

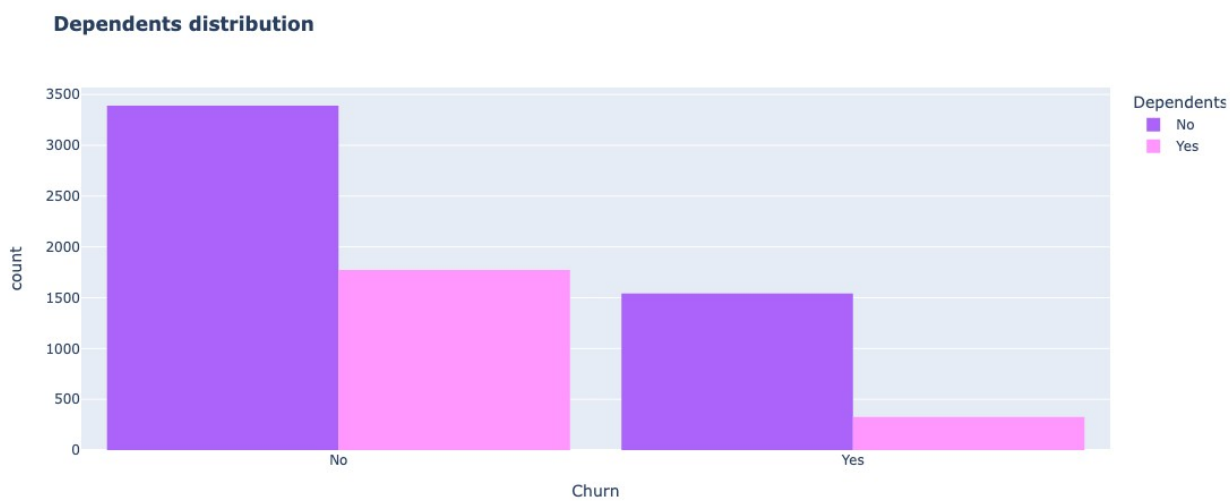



```
fig = px.histogram(df, x="Churn", color="PaymentMethod",
title="<b>Customer Payment Method distribution w.r.t. Churn</b>")
```

```
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



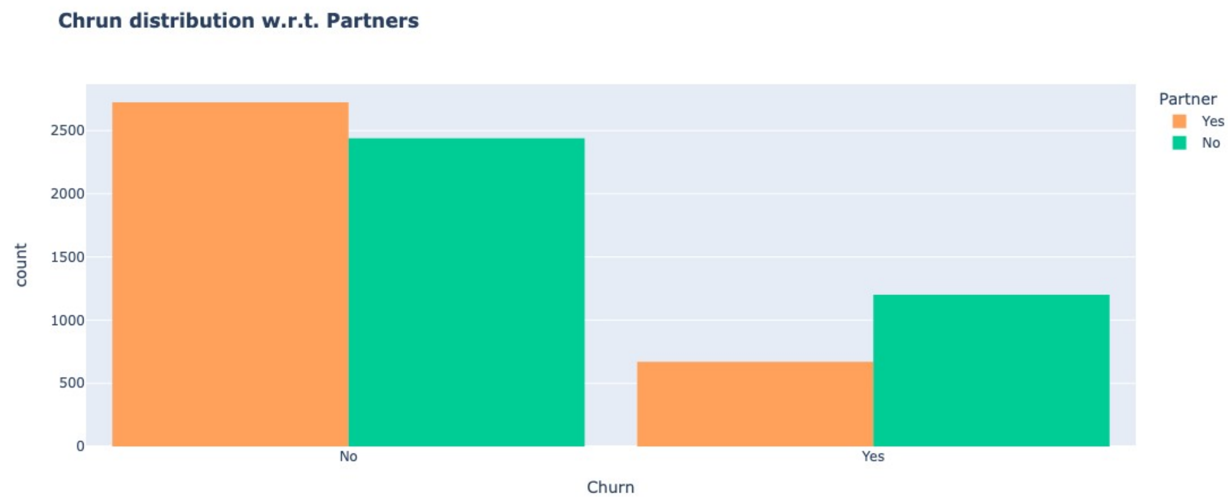

```
color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
fig = px.histogram(df, x="Churn", color="Dependents", barmode="group",
title="<b>Dependents distribution</b>", color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



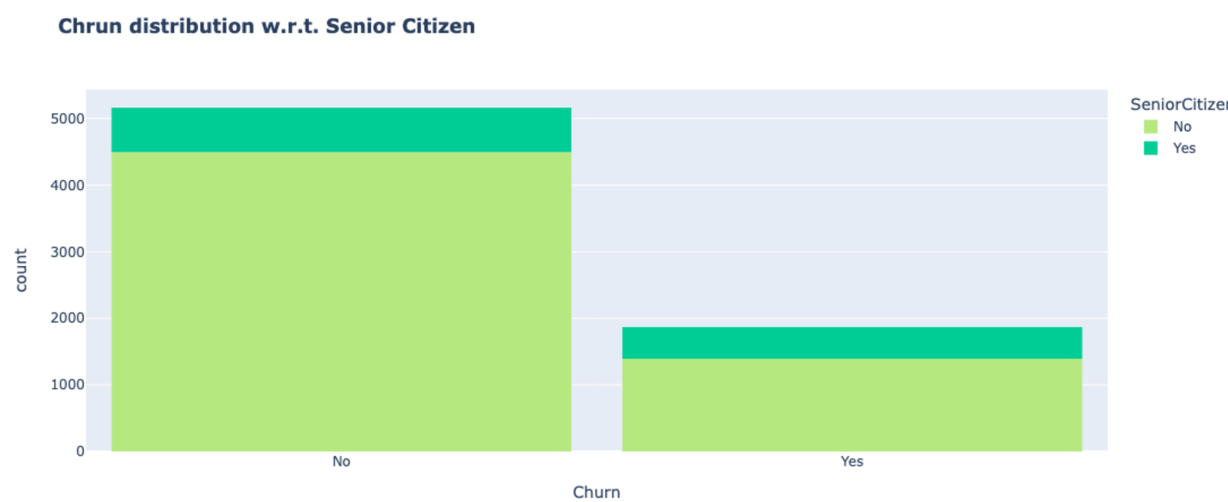
Analysis indicates that single customers or those without children demonstrate a higher propensity for churn compared to customers with dependents

```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="Partner", barmode="group",
```

```
title="<b>Chrun distribution w.r.t. Partners</b>",
color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

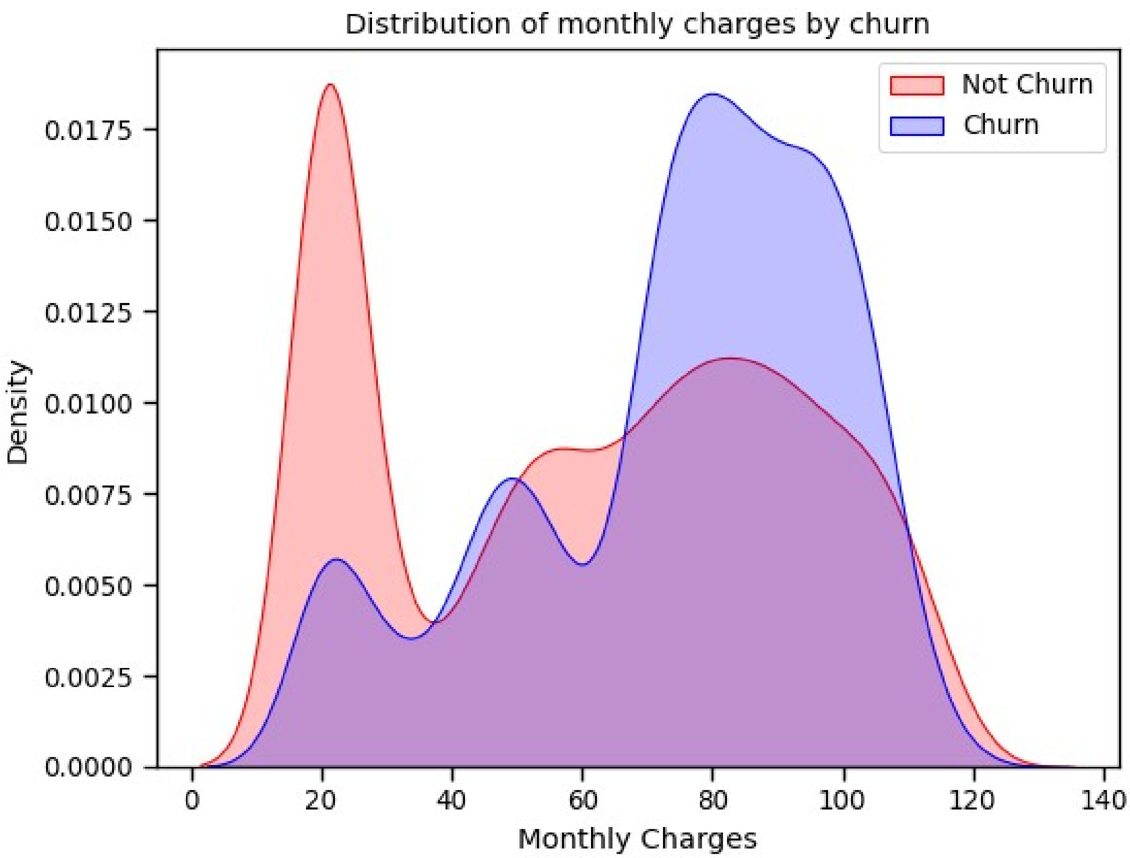



```
color_map = {"Yes": '#00CC96', "No": '#B6E880'}
fig = px.histogram(df, x="Churn", color="SeniorCitizen",
title="<b>Chrun distribution w.r.t. Senior Citizen</b>",
color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



The proportion of senior citizens in the customer base is relatively small

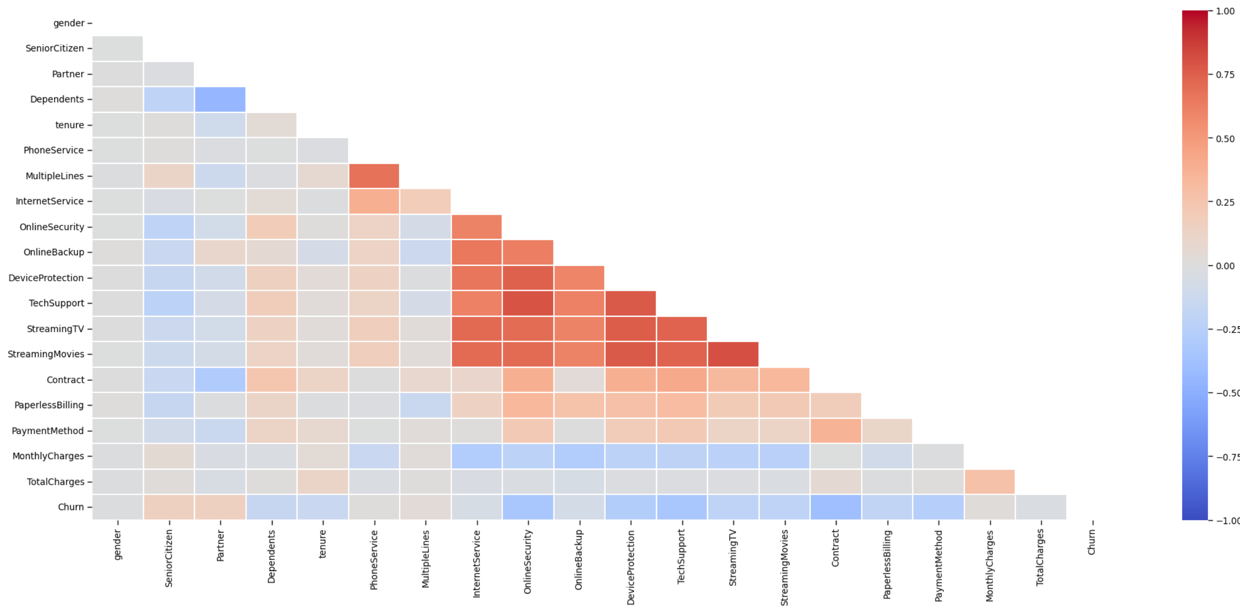
```
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'No') ],
                color="Red", shade = True);
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],
                ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper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



A discernible correlation emerges between the magnitude of monthly service charges and customer churn propensity, wherein individuals incurring higher monthly expenses demonstrate a greater likelihood of terminating their service contracts

```
plt.figure(figsize=(25, 10))
corr = df.apply(lambda x: pd.factorize(x)[0]).corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set the annotation format to display two decimal places
sns.heatmap(corr, mask=mask, xticklabels=corr.columns,
yticklabels=corr.columns,
            annot=True, fmt=".2f", linewidths=.2, cmap='coolwarm',
```

```
vmin=-1, vmax=1)

plt.show()
```



Churn Prediction Model

Splitting the data into train and test sets

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

df = df.apply(lambda x: object_to_int(x))
df.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService \
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

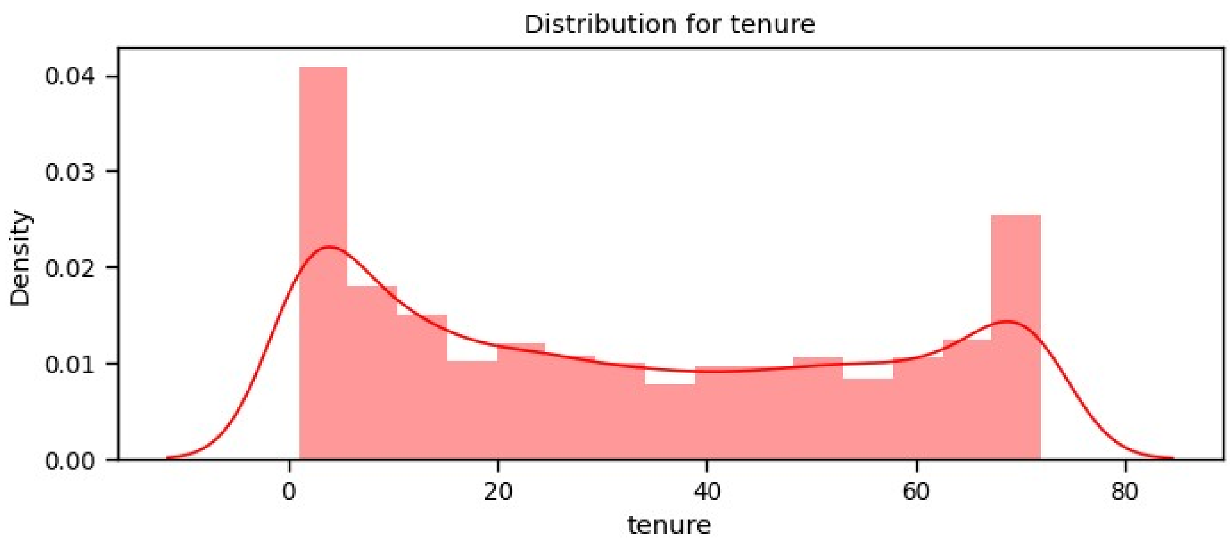
	MultipleLines	InternetService	OnlineSecurity	OnlineBackup \
0	1	0	0	2
1	0	0	2	0
2	0	0	2	2
3	1	0	2	0
4	0	1	0	0

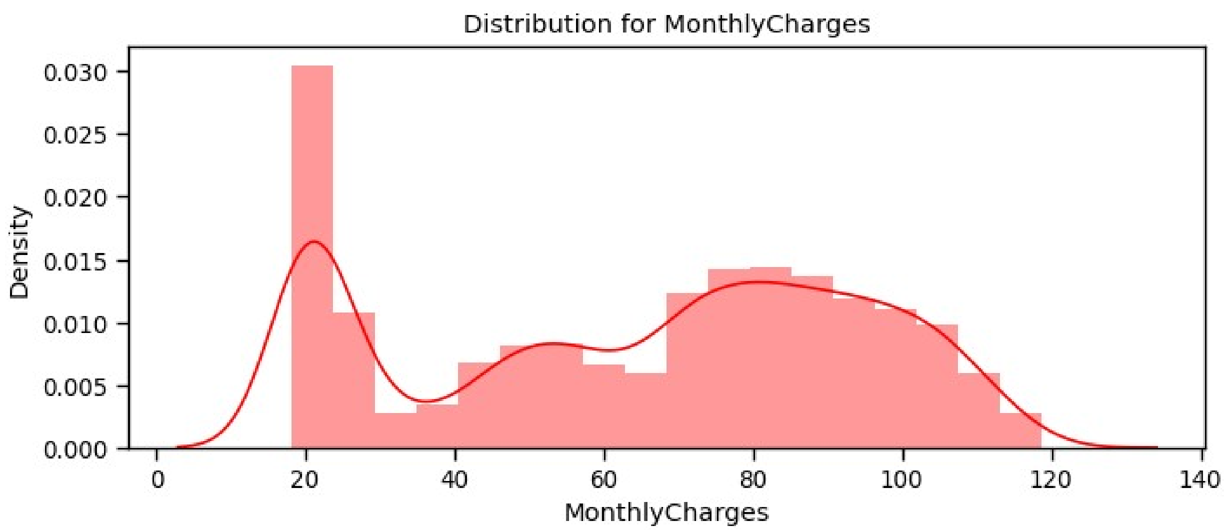
```
   DeviceProtection  TechSupport  StreamingTV  StreamingMovies
Contract  \
0                 0            0            0                0
0
1                 2            0            0                0
1
2                 0            0            0                0
0
3                 2            2            0                0
1
4                 0            0            0                0
0

   PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges
Churn
0                 1              2           29.85         29.85
0
1                 0              3           56.95       1889.50
0
2                 1              3           53.85        108.15
1
3                 0              0           42.30       1840.75
0
4                 1              2           70.70        151.65
1
```

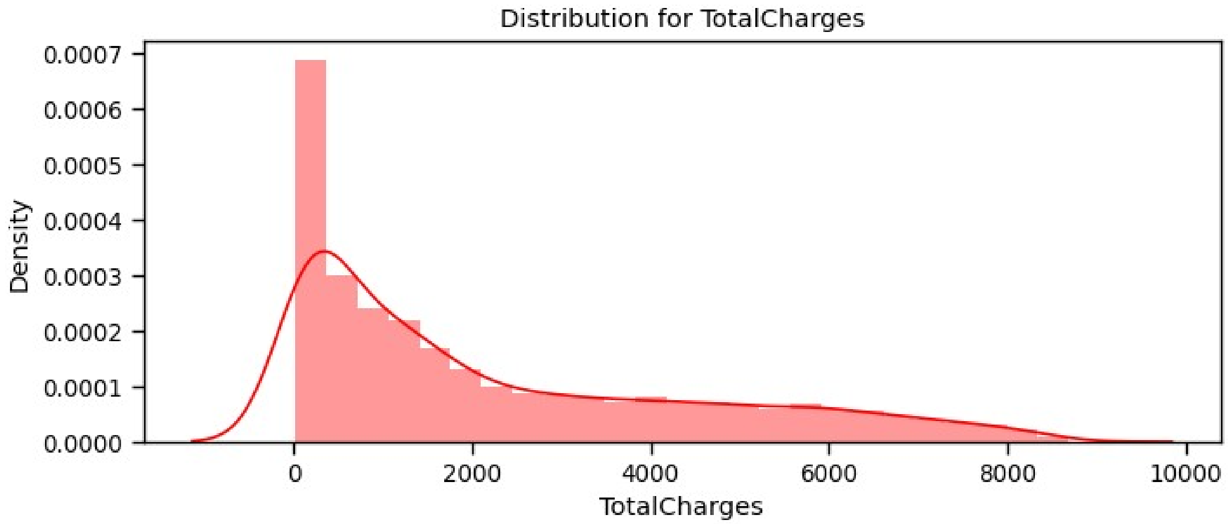
```
plt.figure(figsize=(14,7))
df.corr()['Churn'].sort_values(ascending = False)
```

```
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>

X = df.drop(columns = ['Churn'])
y = df['Churn'].values

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =
0.30, random_state = 40, stratify=y)

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

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

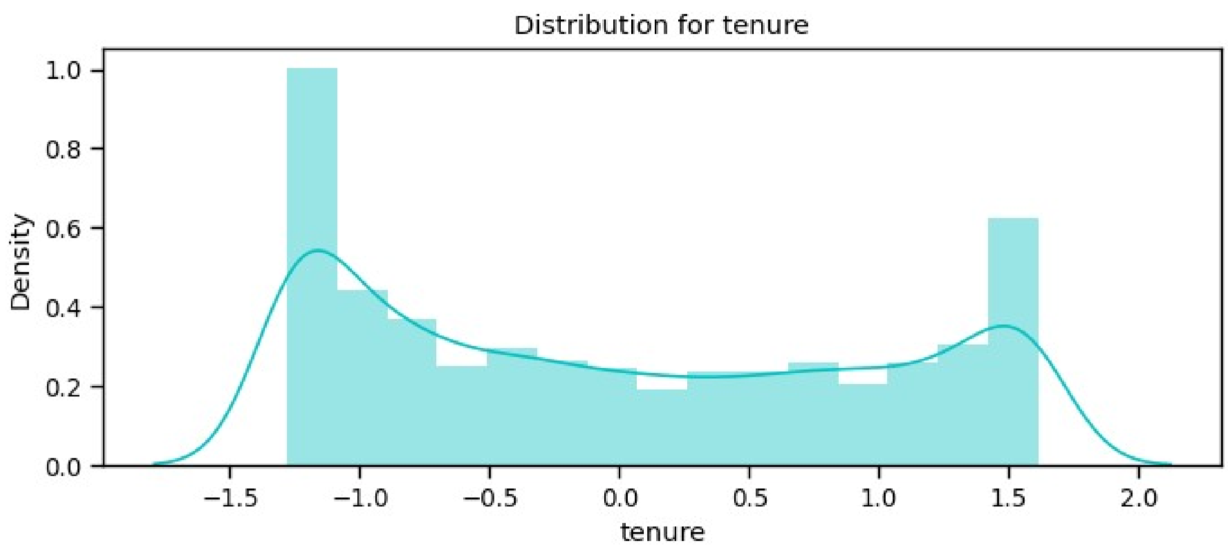


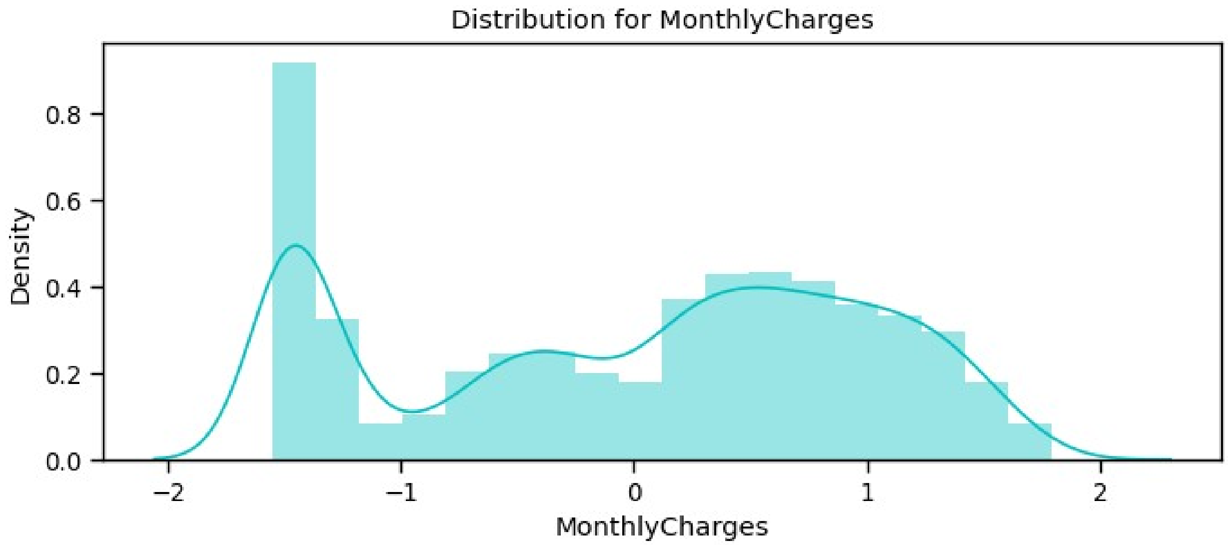


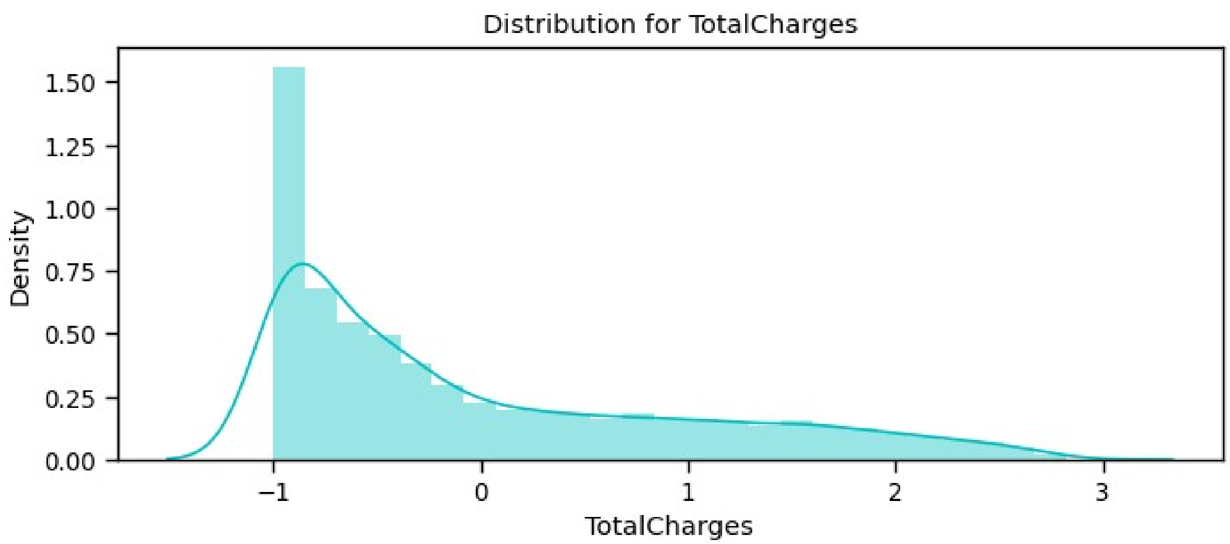


Given the inherent variability in the numerical features, characterized by their distinct value ranges, a standardization technique, specifically the StandardScaler, will be employed to transform these features. This process aims to normalize the data by converting it into a common scale, thereby mitigating the potential influence of features with larger scales on the model's performance and ensuring a more equitable contribution of each feature to the overall analysis

```
df_std =
pd.DataFrame(StandardScaler().fit_transform(df[num_cols].astype('float
64')),
                       columns=num_cols)
for feat in numerical_cols: distplot(feat, df_std, color='c')
```







Machine learning model

KNN

```
knn_model = KNeighborsClassifier(n_neighbors = 11)
knn_model.fit(X_train,y_train)
predicted_y = knn_model.predict(X_test)
accuracy_knn = knn_model.score(X_test,y_test)
print("KNN accuracy:",accuracy_knn)

KNN accuracy: 0.7763033175355451

print(classification_report(y_test, predicted_y))

              precision    recall  f1-score   support
```

```
           0       0.83      0.87      0.85      1549
           1       0.59      0.52      0.55       561

    accuracy                           0.78      2110
   macro avg       0.71      0.69      0.70      2110
weighted avg       0.77      0.78      0.77      2110
```

1. Precision:

Class 0: 0.83 This means that when the model predicts a customer will not churn (class 0), it is correct 83% of the time. Class 1: 0.59 When the model predicts a customer will churn (class 1), it is correct 59% of the time.

1. Recall (Sensitivity):

Class 0: 0.87 The model correctly identifies 87% of the customers who actually did not churn. Class 1: 0.52 The model correctly identifies 52% of the customers who actually did churn.

1. F1-score:

Class 0: 0.85 This is the harmonic mean of precision and recall for class 0, providing a balanced measure of the model's performance. Class 1: 0.55 Similarly, this is the F1-score for class 1, indicating a balance between precision and recall for this class.

1. Support:

Class 0: 1549 This represents the number of instances in the dataset belonging to class 0 (customers who did not churn). Class 1: 561 This represents the number of instances in the dataset belonging to class 1 (customers who churned).

1. Accuracy: 0.78

This is the overall accuracy of the model, indicating that it correctly predicts the churn status for 78% of the customers in the dataset.

1. Macro Average: 0.71 for precision, 0.69 for recall, and 0.70 for F1-score.

This represents the unweighted average of the metric for each class.

1. Weighted Average: 0.77 for precision, 0.78 for recall, and 0.77 for F1-score.

This represents the average of the metric for each class, weighted by the support (number of instances) of each class. This gives more weight to the class with more instances. Interpretation:

Class Imbalance: The support values (1549 vs. 561) indicate a class imbalance, with class 0 (no churn) being more frequent. Good Performance for Class 0: The model performs well in predicting customers who will not churn, with high precision, recall, and F1-score for class 0. Room for Improvement for Class 1: The model struggles to accurately predict customers who will churn, as evidenced by lower precision, recall, and F1-score for class 1. This suggests a need for improvement in identifying and predicting churn in this specific customer segment. Overall:

The model exhibits decent overall accuracy, but there's room for improvement in predicting customer churn (class 1). Addressing this imbalance and improving the model's performance on the minority class (class 1) is crucial for better churn prediction and customer retention strategies.

```
svc_model = SVC(random_state = 1)
svc_model.fit(X_train,y_train)
predict_y = svc_model.predict(X_test)
accuracy_svc = svc_model.score(X_test,y_test)
print("SVM accuracy is :",accuracy_svc)
```

```
SVM accuracy is : 0.8075829383886256
```

```
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
```

```
lr_pred= lr_model.predict(X_test)
report = classification_report(y_test,lr_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.86      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      0.81      2110
```

Customer churn poses a significant threat to a firm's profitability, necessitating the implementation of strategic countermeasures. A fundamental approach to mitigating churn lies in a profound understanding of the customer base. This involves proactively identifying at-risk customers and implementing targeted strategies to enhance their satisfaction. Prioritizing exceptional customer service is paramount in this endeavor. Furthermore, cultivating customer loyalty through personalized experiences and tailored service offerings can effectively reduce churn rates. To proactively address future churn, many firms proactively engage in post-churn surveys to gain valuable insights into the reasons for customer departure, enabling them to implement preventative measures.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Define true labels and predicted labels (replace with your actual
data)
y_true = [0] * 1549 + [1] * 561  # True labels based on support (class
```

```
0: 1549, class 1: 561)
y_pred = [0] * 1379 + [1] * 170 + [0] * 236 + [1] * 325  # Predictions
to match precision and recall

# Generate the confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0,
1])
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```