In [11]:

```
import pandas as pd
import numpy as np
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')
import missingno as msno
```

I...

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

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score,
confusion_matrix, precision_score, f1_score,
accuracy_score, classification_report
```

I...

```
df=pd.read_csv(r"D:\personal end to end project\Telecom
Churn Prediction\WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

In [19]:

```
df.head()
```

Out[19]:

	customerID	gender	SeniorCitizen	Partner	Dependent
0	7590-VHVEG	Female	0	Yes	N

	customerID	gender	SeniorCitizen	Partner	Dependents
1	5575-GNVDE	Male	0	No	No
2	3668-QPYBK	Male	0	No	No
3	7795-CFOCW	Male	0	No	No
4	9237-HQITU	Female	0	No	No

5 rows × 21 columns

In [20]:
```
df.columns.values
```

Out[20]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partn
er', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'Int
ernetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtec
tion',
       'TechSupport', 'StreamingTV', 'StreamingMovie
s', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCh
arges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [21]:
```
df.dtypes
```

Out[21]:
```
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
```

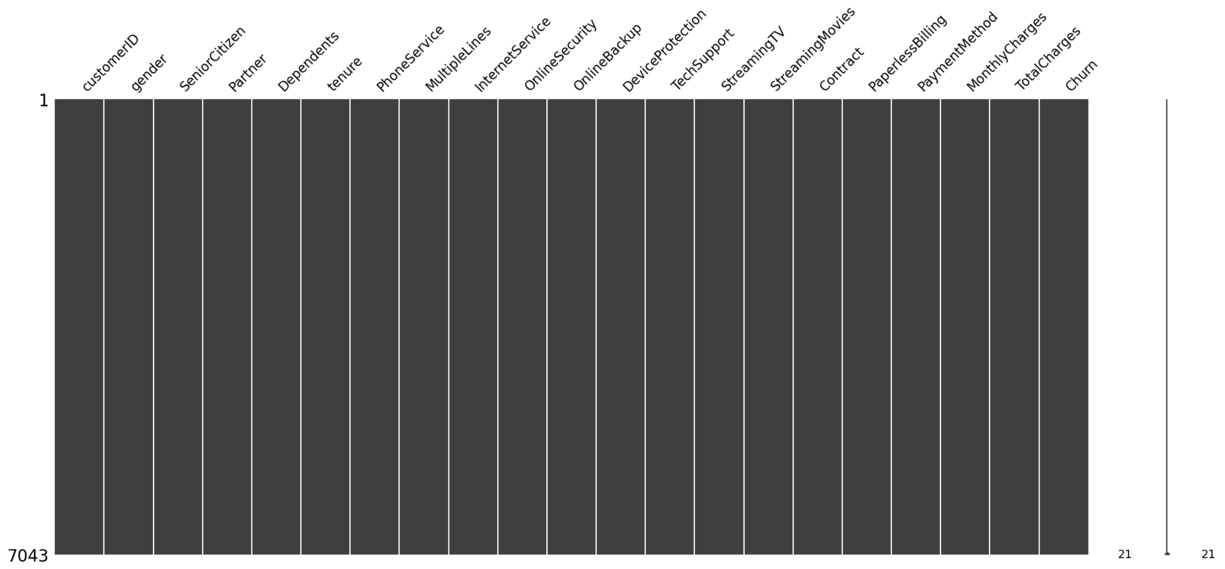
In [22]:
```
df.shape
```

Out[22]:
```
(7043, 21)
```

In [23]:
```
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 [24]:
```
msno.matrix(df);
```



In [25]:
```
df = df.drop(['customerID'], axis = 1)
df.head()
```

Out[25]:

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

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

Out[26]:
```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
```

```
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

In [27]:
```
df[np.isnan(df['TotalCharges'])]
```

Out[27]:

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

In [28]:
```
df[df['tenure'] == 0].index
```

Out[28]:
```
Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 52
18, 6670, 6754], dtype='int64')
```

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

Out[29]:
```
Index([], dtype='int64')
```

In [30]:
```
df.fillna(df["TotalCharges"].mean())
```

Out[30]:

	gender	SeniorCitizen	Partner	Dependents	tenure
0	Female	0	Yes	No	1
1	Male	0	No	No	34
2	Male	0	No	No	2
3	Male	0	No	No	45
4	Female	0	No	No	2
...	...	...	...	...	...
7038	Male	0	Yes	Yes	24
7039	Female	0	Yes	Yes	72
7040	Female	0	Yes	Yes	11
7041	Male	1	Yes	No	4
7042	Male	0	No	No	66

7032 rows × 20 columns

In [31]:
```
df.isnull().sum()
```
Out[31]:
```
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 [...
```
df["SeniorCitizen"]= df["SeniorCitizen"].map({0: "No",
1: "Yes"})
df.head()
```
Out[32]:

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

In [3...
```
df["InternetService"].describe(include=['object',
         'bool'])
```

Out[33]:
```
count             7032
unique               3
top        Fiber optic
freq              3096
Name: InternetService, dtype: object
```

In [3...
```
numerical_cols = ['tenure', 'MonthlyCharges',
'TotalCharges']
df[numerical_cols].describe()
```

Out[34]:

	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

In ...
```
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()
```

In […

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

Out[36]:

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

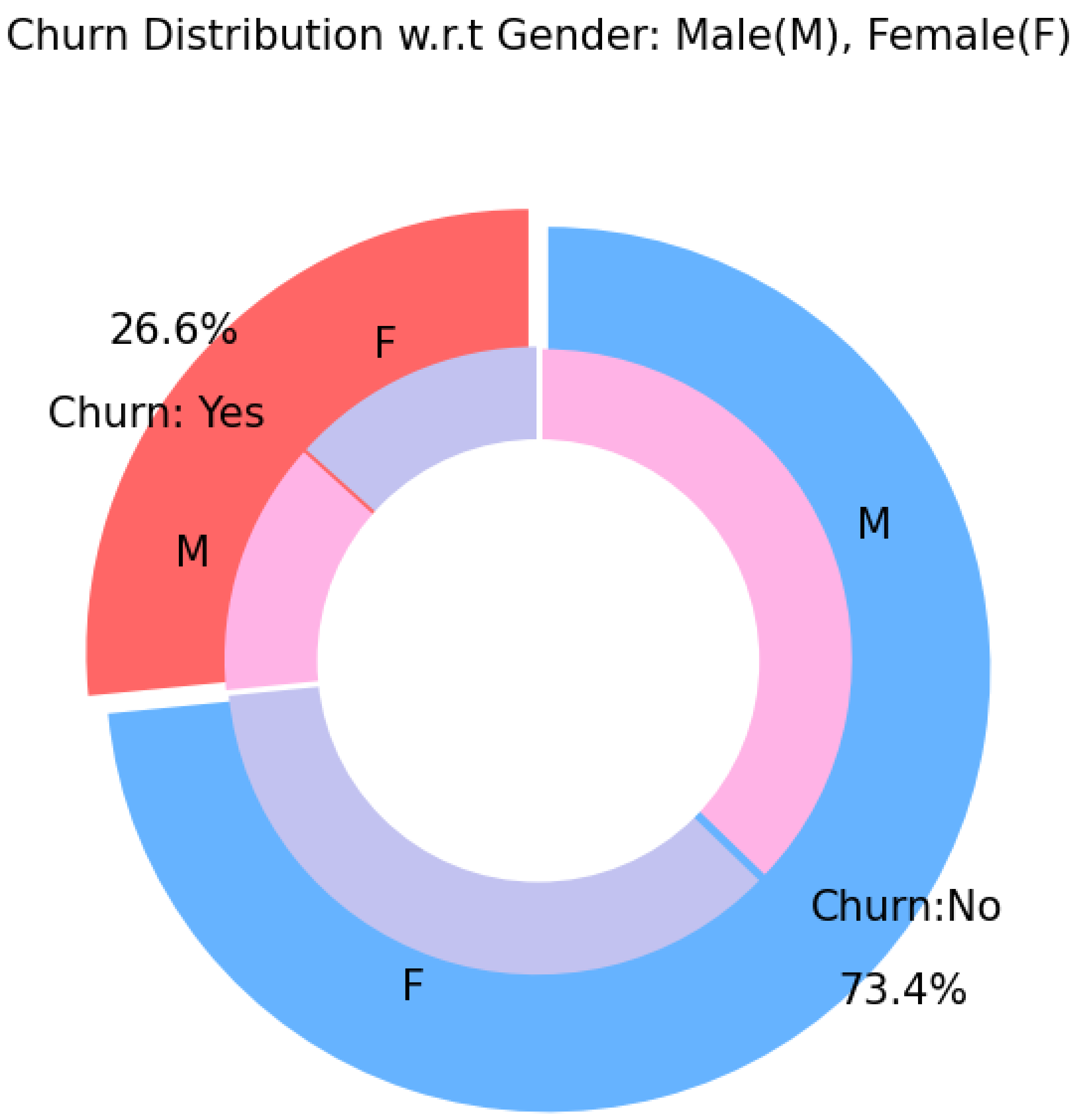
In […

```
df["Churn"]
```

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

Out[37]:
```
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_gen
    der,startangle=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()
```



```
I… 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()
```

In [...

```
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()
```

In [42]:
```
df["InternetService"].unique()
```
Out[42]:
```
array(['DSL', 'Fiber optic', 'No'], dtype=object)
```
In [...
```
df[df["gender"]=="Male"][["InternetService",
"Churn"]].value_counts()
```
Out[43]:
```
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 [...
```
df[df["gender"]=="Female"][["InternetService",
"Churn"]].value_counts()
```
Out[44]:
```
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 ...
```
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 and Gender</b>")

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()
```

```
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()
```

```
[ ]: color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
     fig = px.histogram(df, x="Churn", color="OnlineSecurity",
     barmode="group", title="<b>Churn w.r.t Online
     Security</b>", color_discrete_map=color_map)
     fig.update_layout(width=700, height=500, bargap=0.1)
     fig.show()
```

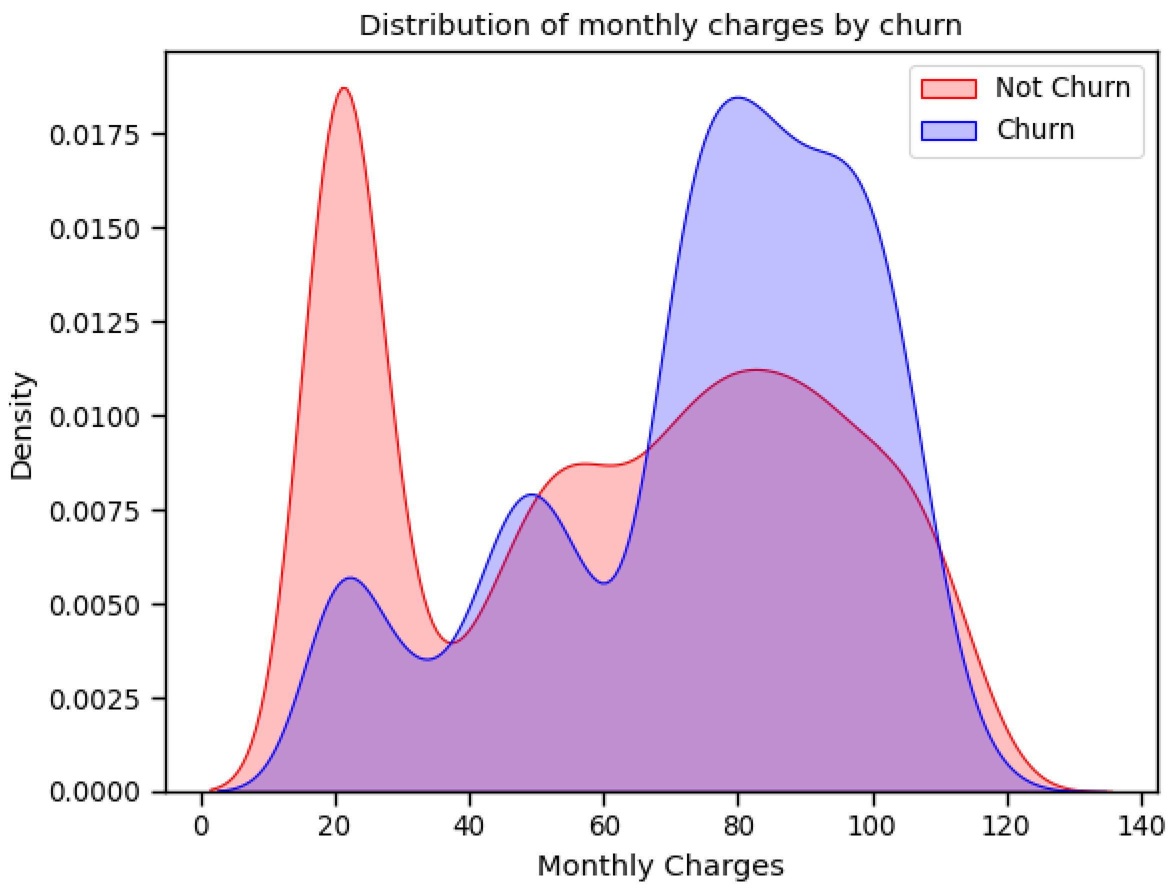
```
[... color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
     fig = px.histogram(df, x="Churn", color="PaperlessBilling",
     title="<b>Chrun distribution w.r.t. Paperless Billing</b>",
     color_discrete_map=color_map)
     fig.update_layout(width=700, height=500, bargap=0.1)
     fig.show()
```

```
[... fig = px.histogram(df, x="Churn",
     color="TechSupport",barmode="group",  title="<b>Chrun
     distribution w.r.t. TechSupport</b>")
     fig.update_layout(width=700, height=500, bargap=0.1)
     fig.show()
```

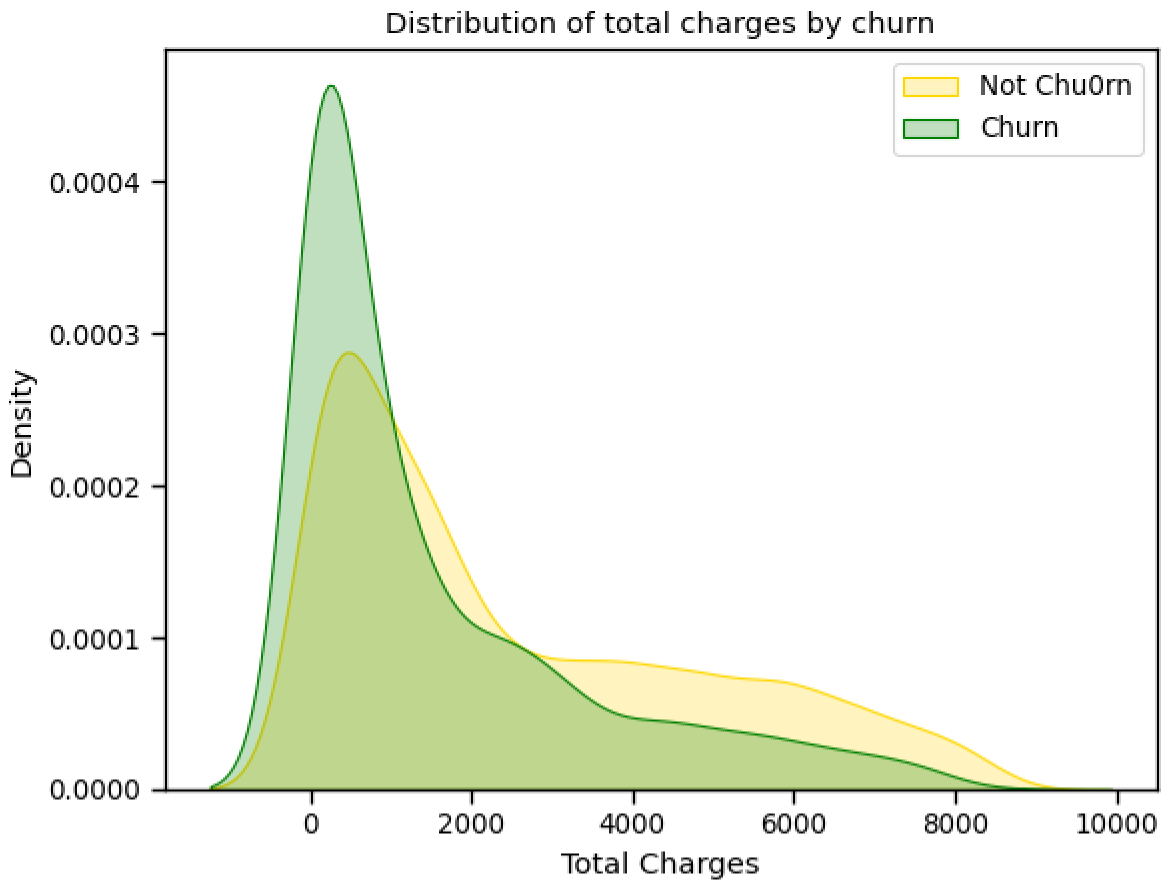
```
color_map = {"Yes": '#00CC96', "No": '#B6E880'}
fig = px.histogram(df, x="Churn", color="PhoneService",
title="<b>Chrun distribution w.r.t. Phone Service</b>",
color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

In [5…

```
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');
```

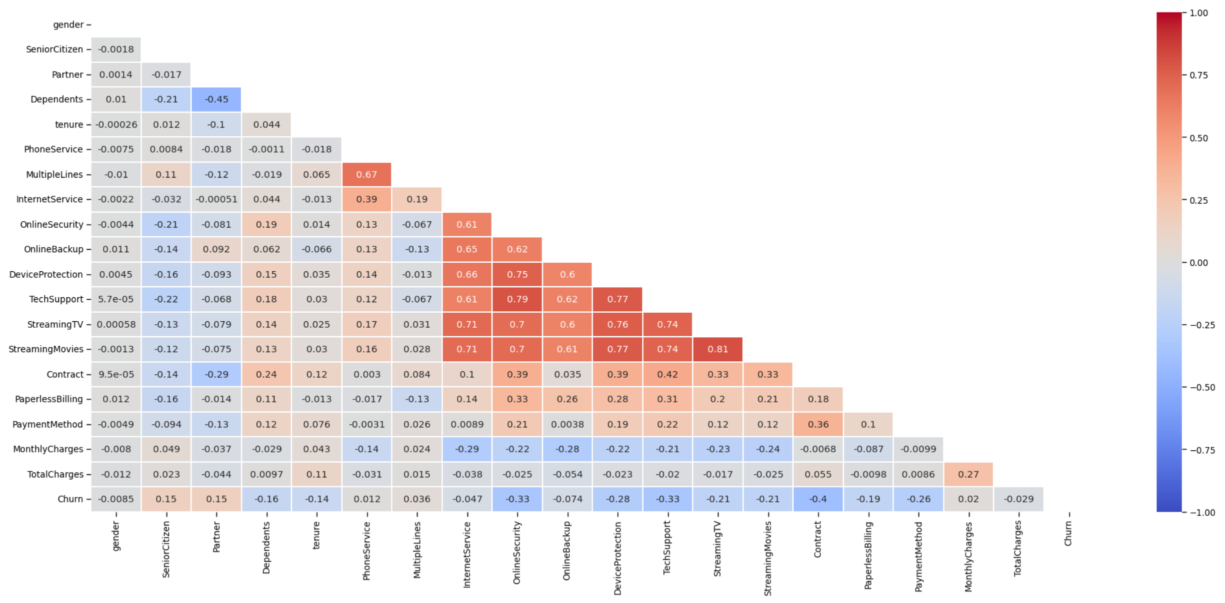


In [5...
```
ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'No')
],
                color="Gold", shade = True);
ax = sns.kdeplot(df.TotalCharges[(df["Churn"] ==
'Yes') ],
                ax =ax, color="Green", shade= True);
ax.legend(["Not Chu0rn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Total Charges');
ax.set_title('Distribution of total charges by
churn');
```



In [5…
```
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()
```

```
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.columns,
annot=True, linewidths=.2, cmap='coolwarm', vmin=-1,
vmax=1)
```



In [...

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

In [58]:

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

Out[58]:

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

In [59]:

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

Out[59]:

```
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
```

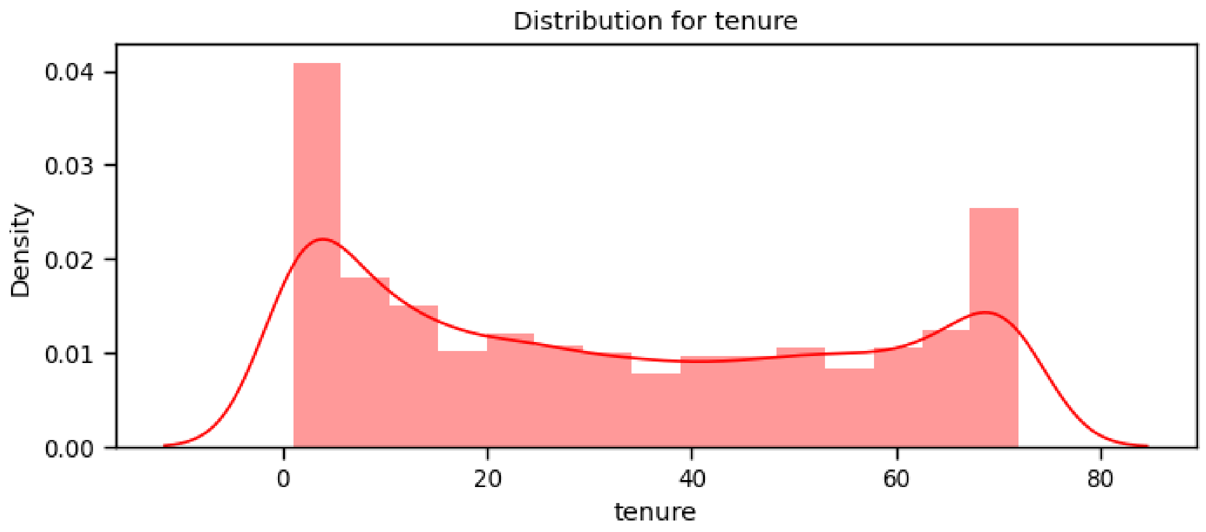
```
<Figure size 1400x700 with 0 Axes>
```

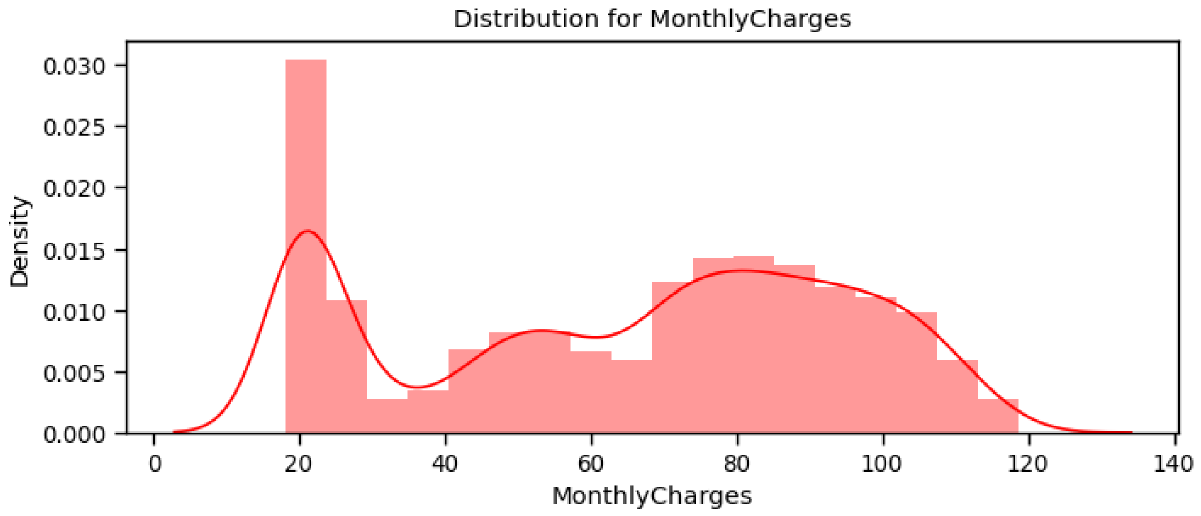
In [60]:
```
X = df.drop(columns = ['Churn'])
y = df['Churn'].values
```

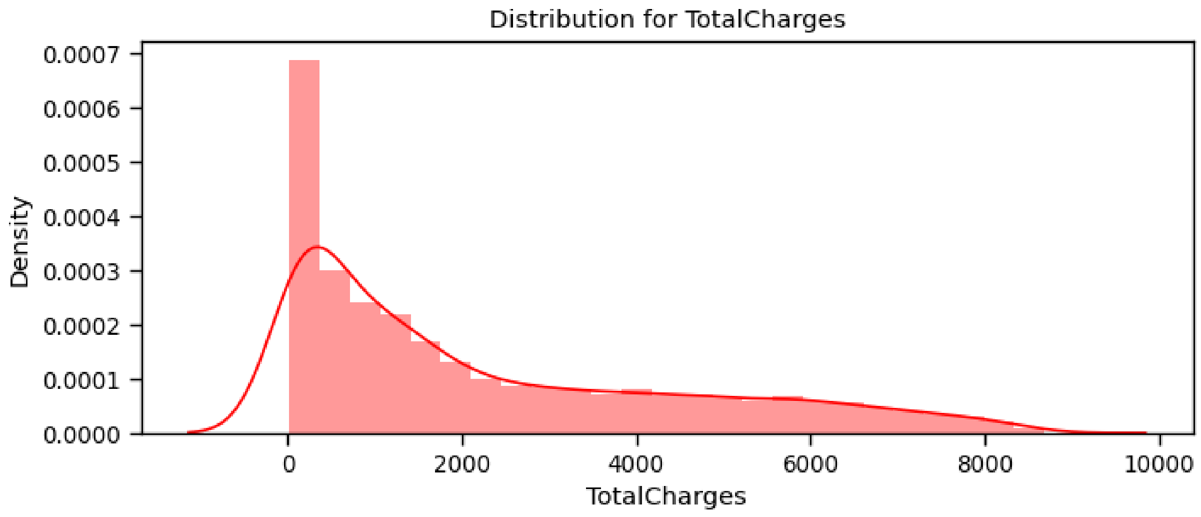
In...
```
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size = 0.30, random_state = 40,
stratify=y)
```

In [62]:
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

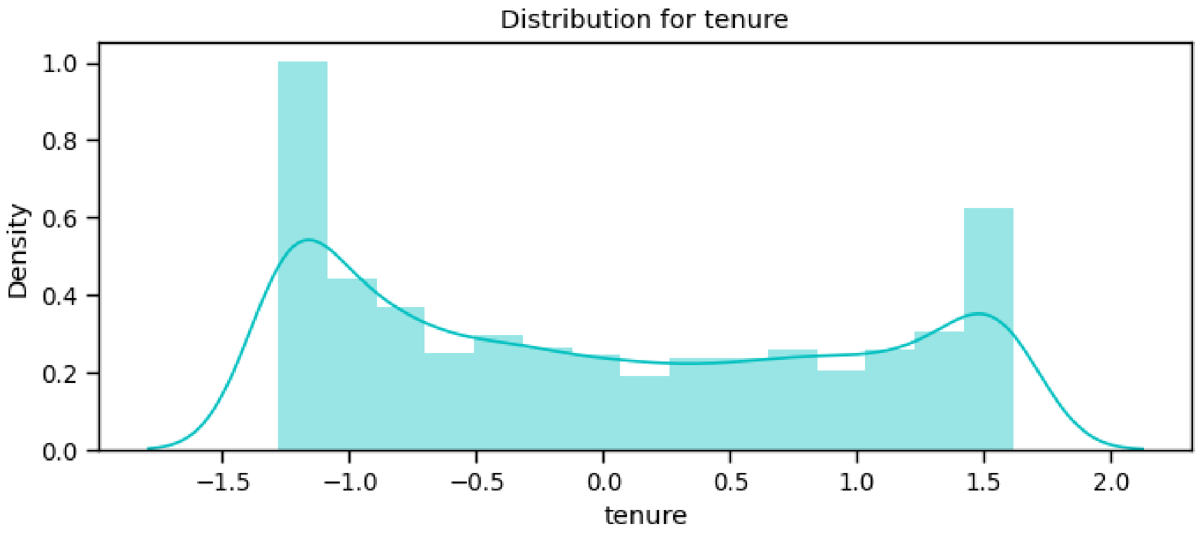
In [6...
```
num_cols = ["tenure", 'MonthlyCharges',
'TotalCharges']
for feat in num_cols: distplot(feat, df)
```

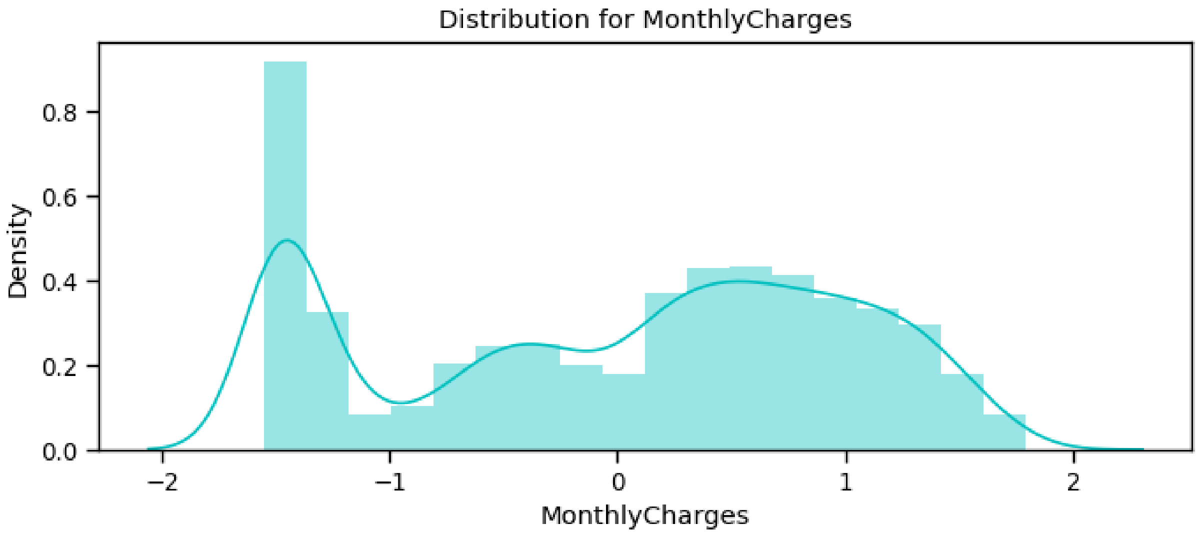


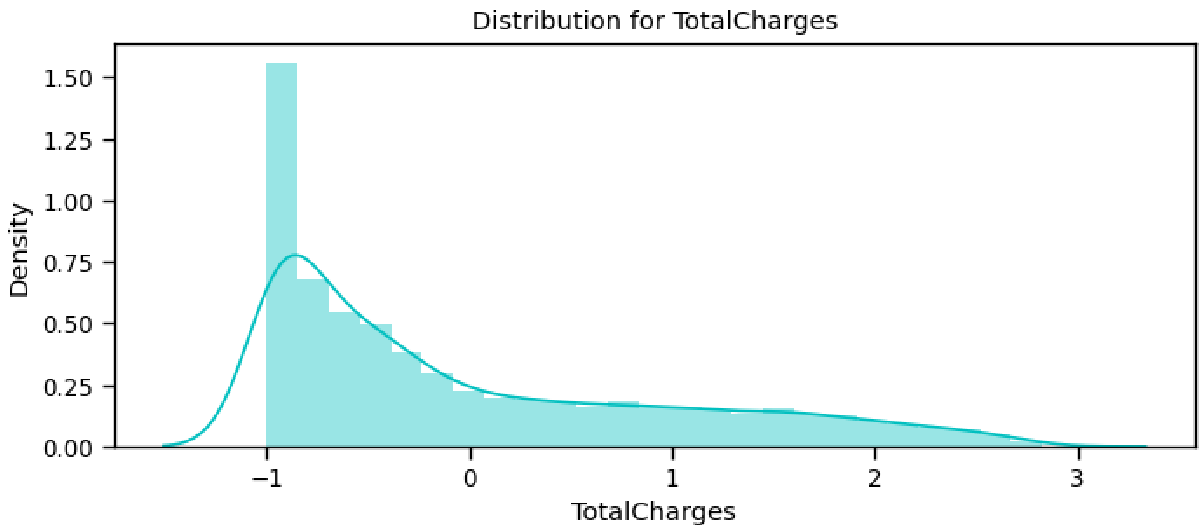




In …
```
df_std =
pd.DataFrame(StandardScaler().fit_transform(df[num_cols]
.astype('float64')),
                        columns=num_cols)
for feat in numerical_cols: distplot(feat, df_std,
color='c')
```







In…
```
cat_cols_ohe =['PaymentMethod', 'Contract',
'InternetService'] # those that need one-hot encoding
cat_cols_le = list(set(X_train.columns)- set(num_cols) -
set(cat_cols_ohe)) #those that need label encoding
```

In [6…
```
scaler= StandardScaler()

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

In [69]:
```
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
```

In [70]:
```
print(classification_report(y_test, predict_y))
```

```
              precision    recall  f1-score   support

           0       0.84      0.92      0.88      1549
           1       0.69      0.50      0.58       561

    accuracy                           0.81      2110
   macro avg       0.76      0.71      0.73      2110
weighted avg       0.80      0.81      0.80      2110
```

In [ ]: