Customer_churn_Project.ipynb
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Gemini

```
[ ] #import the required libraries
    import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.ticker as mtick
    import matplotlib.pyplot as plt
    import missingno as msno
    import plotly.express as px

    # pip install plotly.subplots
    import plotly.subplots as sp

    #  !pip install plotly
    import plotly.graph_objects as go

    # pip list | grep plotly
    import warnings
    warnings.filterwarnings('ignore')
```

Mohd Sameer Hussain Hashmi

www.linkedin.com/in/mohdsameer28

https://github.com/SameerHussain128?tab=repositories

```
[ ] df = pd.read_csv('Customer-Churn.csv')
```

```
[ ] df.head()
```

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

5 rows × 21 columns

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

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	Onl
0	Female	0	Yes	No	1	No	No phone service	DSL	
1	Male	0	No	No	34	Yes	No	DSL	
2	Male	0	No	No	2	Yes	No	DSL	
3	Male	0	No	No	45	No	No phone service	DSL	
4	Female	0	No	No	2	Yes	No	Fiber optic	

```
[ ] # Display the information
    df.info()
```

```
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtyp
---  ------            --------------  ----
 0   gender            7043 non-null   obje
 1   SeniorCitizen     7043 non-null   int6
 2   Partner           7043 non-null   obje
 3   Dependents        7043 non-null   obje
 4   tenure            7043 non-null   int6
 5   PhoneService      7043 non-null   obje
 6   MultipleLines     7043 non-null   obje
 7   InternetService   7043 non-null   obje
 8   OnlineSecurity    7043 non-null   obje
```

Customer_churn_Project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share S

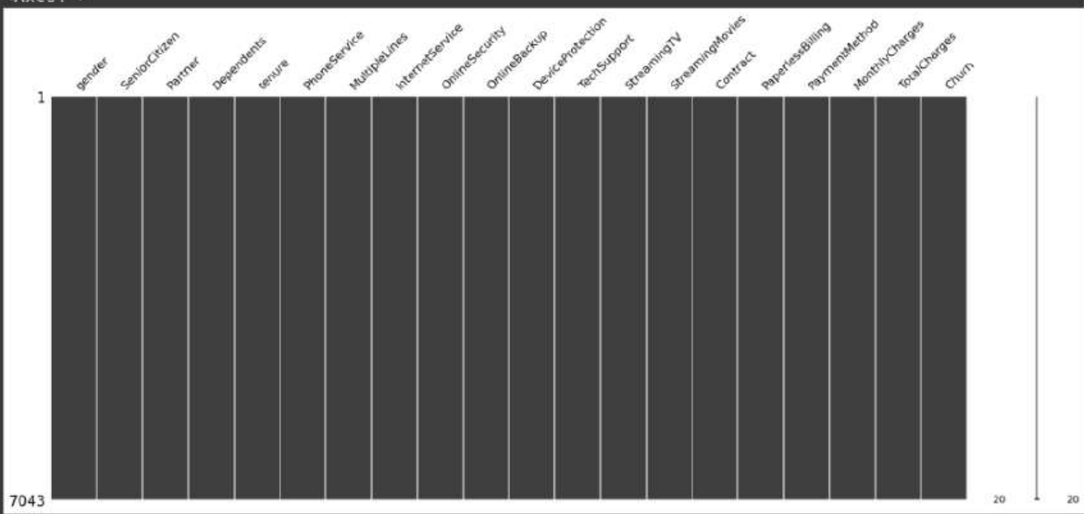
+ Code + Text Connect Gemini

```
[ ] # Display the information
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtyp
---  ------            --------------  ----
 0   gender            7043 non-null   obje
 1   SeniorCitizen     7043 non-null   int6
 2   Partner           7043 non-null   obje
 3   Dependents        7043 non-null   obje
 4   tenure            7043 non-null   int6
 5   PhoneService      7043 non-null   obje
 6   MultipleLines     7043 non-null   obje
 7   InternetService   7043 non-null   obje
 8   OnlineSecurity    7043 non-null   obje
 9   OnlineBackup      7043 non-null   obje
 10  DeviceProtection  7043 non-null   obje
 11  TechSupport       7043 non-null   obje
 12  StreamingTV       7043 non-null   obje
 13  StreamingMovies   7043 non-null   obje
 14  Contract          7043 non-null   obje
 15  PaperlessBilling  7043 non-null   obje
 16  PaymentMethod     7043 non-null   obje
 17  MonthlyCharges    7043 non-null   floa
 18  TotalCharges      7043 non-null   obje
 19  Churn             7043 non-null   obje
```

```
[ ] # Visualize missing values as a matrix
    msno.matrix(df)
```

<Axes: >




```
[ ] df.isnull().sum()
```

No missing values dataset is clear

0

Customer_churn_Project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Gemini

```
df.describe()
```

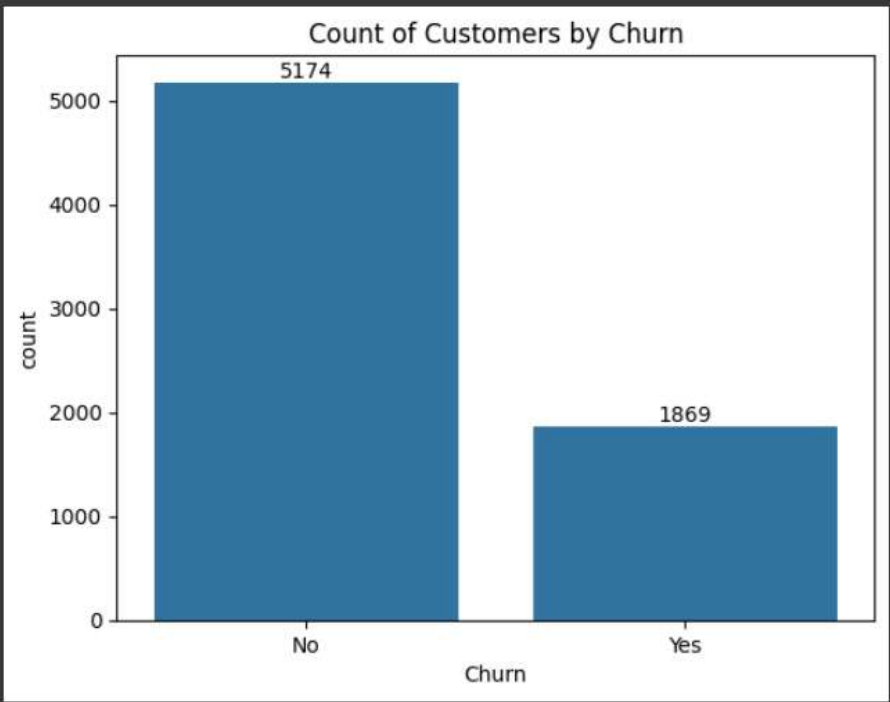
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

```
# Converting object values to numeric in SeniorCitizen column

def conv(value):
    if value == 1:
        return "yes"
    else:
        return "no"

df['SeniorCitizen'] = df["SeniorCitizen"].apply(conv)
```

Data Visualization

Exploratory Data Analysis

```
ax = sns.countplot(x = 'Churn', data = df)

ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Churn")
plt.show()
```

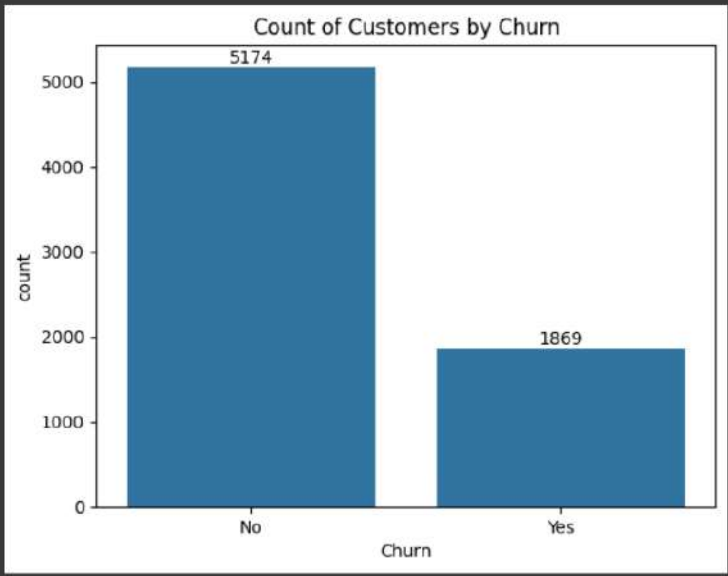


- As from the above visualization we can see that less no.of customers churn

Customer_churn_Project.ipynb

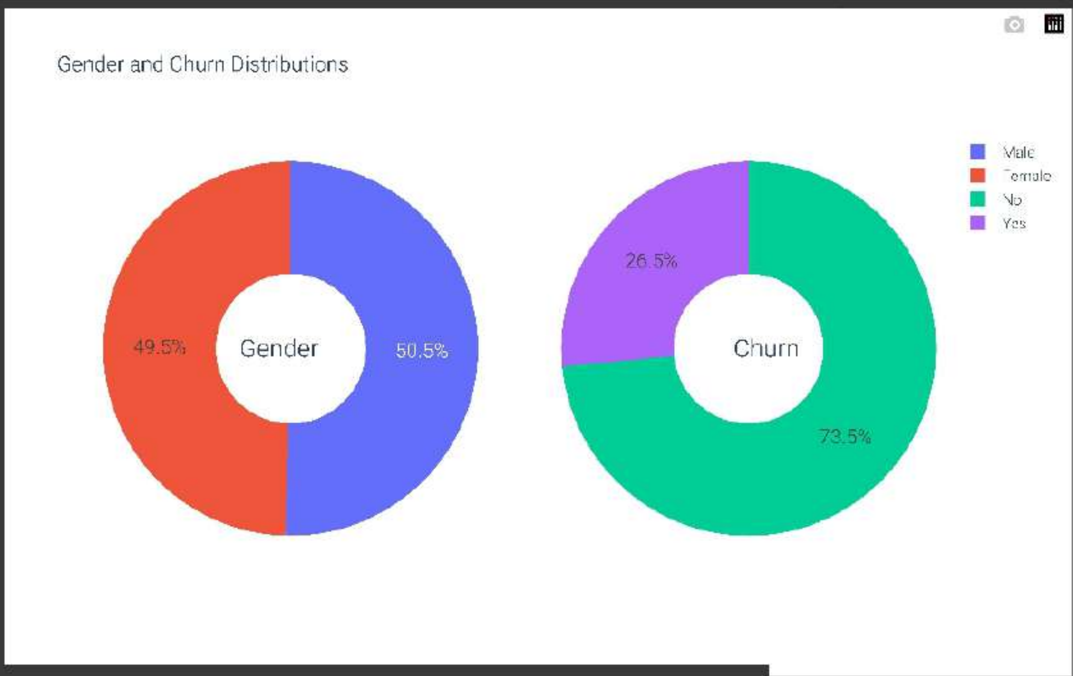
File Edit View Insert Runtime Tools Help All changes saved

```
ax = sns.countplot(x = 'Churn', data = df)

ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Churn")
plt.show()
```



- As from the above visualization we can see that less no.of customers churn

```
Fig_labels = ['Male', 'Female']
c_labels = ['No', 'Yes']
# Create subplots: use 'domain' type for Pie subplot
fig = sp.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()
```




```
[ ] import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd

    # Assuming df is your DataFrame
    # If you don't have the DataFrame, you'll need to create or load it
```

Customer_churn_Project.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Gemini

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import pandas as pd

# Assuming df is your DataFrame
# If you don't have the DataFrame, you'll need to create or load it

# Create subplots: use 'xy' type for Bar subplots
fig = make_subplots(rows=1, cols=2, subplot_titles=("Gender Distribution", "Churn Distribution"))

# Gender Distribution
gender_counts = df['gender'].value_counts()
fig.add_trace(
    go.Bar(x=gender_counts.index, y=gender_counts.values, name="Gender"),
    row=1, col=1
)

# Churn Distribution
churn_counts = df['Churn'].value_counts()
fig.add_trace(
    go.Bar(x=churn_counts.index, y=churn_counts.values, name="Churn"),
    row=1, col=2
)

# Update layout
fig.update_layout(
    title_text="Gender and Churn Distributions",
    showlegend=False,
    height=500,
    width=800
)

# Update y-axes
fig.update_yaxes(title_text="Count", row=1, col=1)
fig.update_yaxes(title_text="Count", row=1, col=2)

fig.show()
```

Insights :

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

```
result = df.groupby(['gender', 'Churn'])['Churn'].count().unstack(fill_value=0)
print(result)
```

```
Churn     No   Yes
gender
Female  2549   939
Male    2625   930
```

```
# Create a bar plot
ax = result.plot(kind='bar', figsize=(10, 6), width=0.8)
```

Customer_churn_Project.ipynb

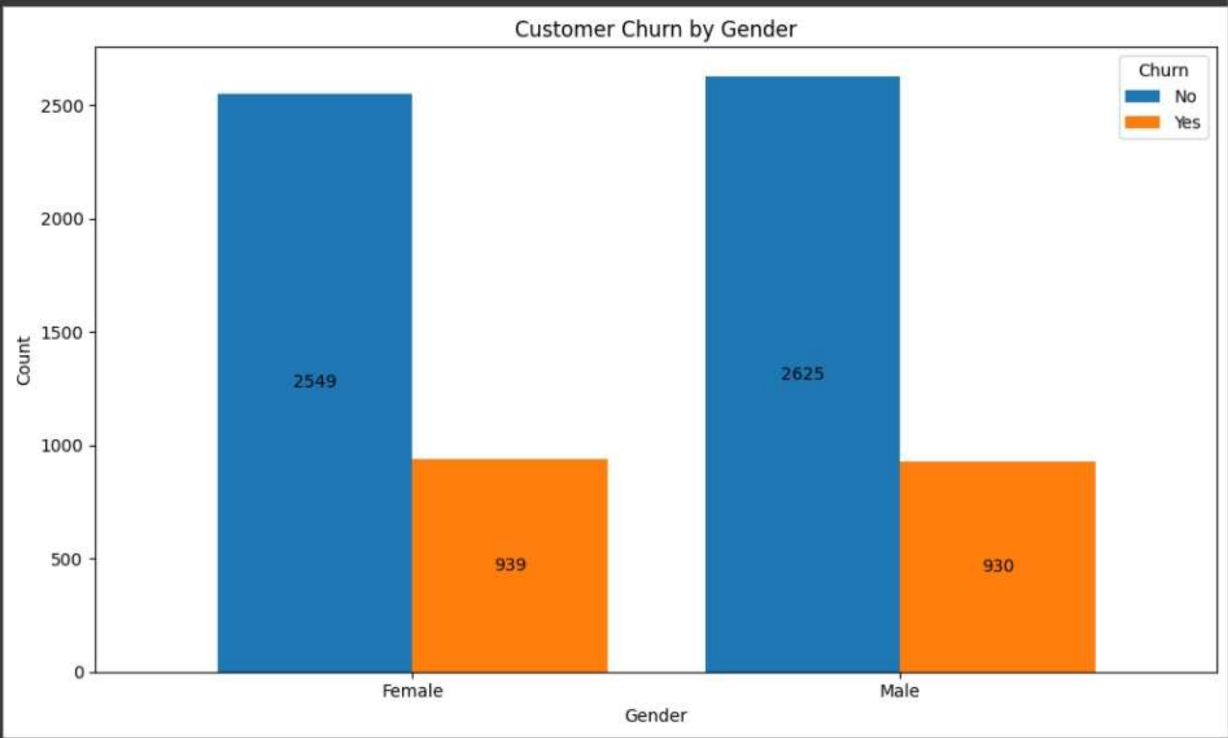
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Gemini

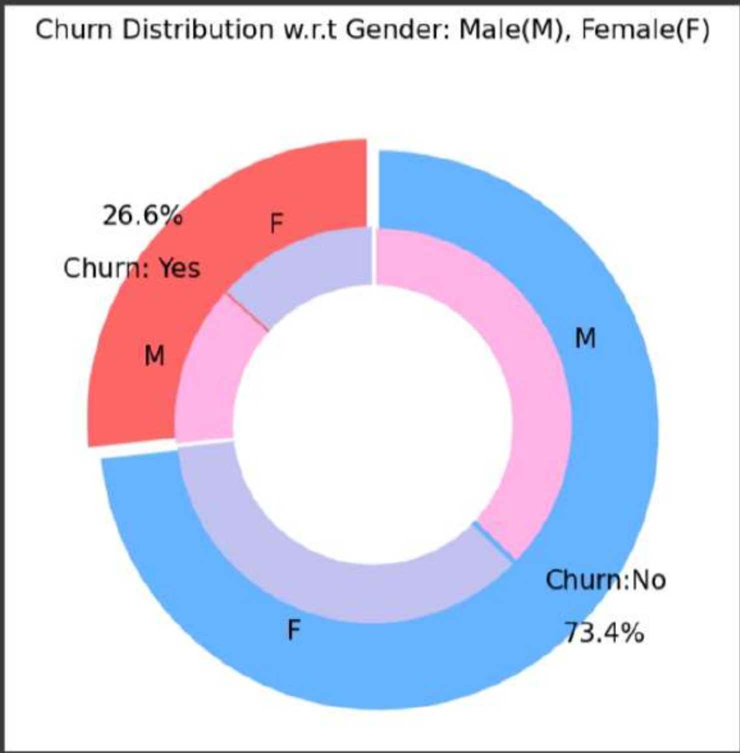
```
Churn     No   Yes
gender
Female  2549  939
Male    2625  930
```

```
[ ] # Create a bar plot
    ax = result.plot(kind='bar', figsize=(10, 6), width=0.8)

    # Customize the plot
    plt.title('Customer Churn by Gender')
    plt.xlabel('Gender')
    plt.ylabel('Count')
    plt.legend(title='Churn')

    # Rotate x-axis labels to horizontal and adjust their position
    plt.xticks(rotation=0, ha='center')

    # Add value labels on the bars
    for container in ax.containers:
        ax.bar_label(container, label_type='center')

    # Adjust layout and display the plot
    plt.tight_layout()
    plt.show()
```




```
[ ] 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, st
    plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startangle=90, explode=explode_gender,
    #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.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()
```

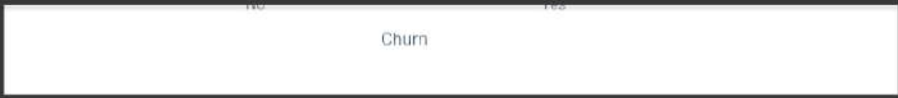


There is negligible difference in customer percentage/ count who chnaged the service provider. Both genders behaved in similar fashion when it comes to migrating to another service provider/firm.

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

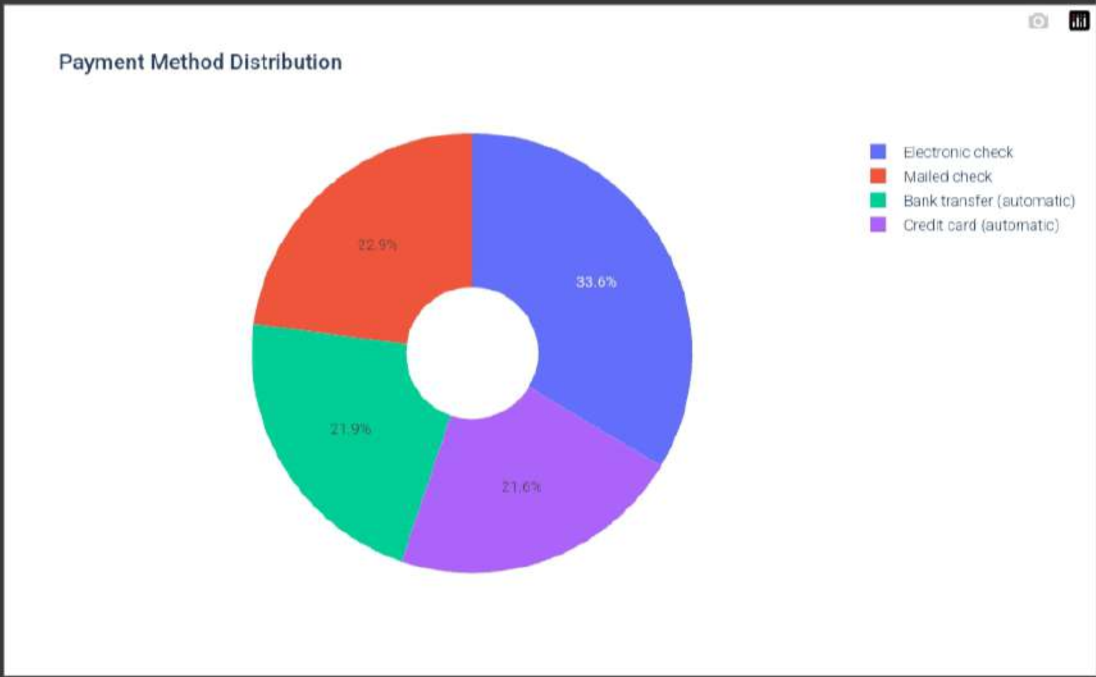
About 75% of customer with Month-to-Month Contract opted to move out as compared to 13% of customrs with One Year Contract and 3% with Two Year Contract

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



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

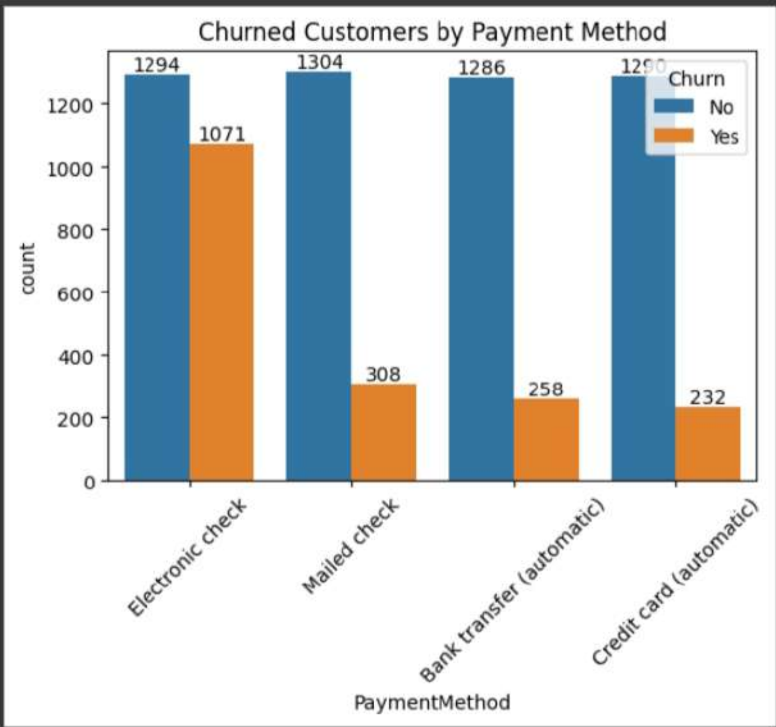
```
[ ] labels = df['PaymentMethod'].unique()
    values = df['PaymentMethod'].value_counts()

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


```
[ ] plt.figure(figsize = (6,4))
    ax = sns.countplot(x = "PaymentMethod", data = df, hue = "Churn")
    ax.bar_label(ax.containers[0])
    ax.bar_label(ax.containers[1])
    plt.title("Churned Customers by Payment Method")
    plt.xticks(rotation = 45)
    plt.show()
```

```
[ ] plt.figure(figsize = (6,4))
    ax = sns.countplot(x = "PaymentMethod", data = df, hue = "Churn")
    ax.bar_label(ax.containers[0])
    ax.bar_label(ax.containers[1])
    plt.title("Churned Customers by Payment Method")
    plt.xticks(rotation = 45)
    plt.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.

```
[ ] df["InternetService"].unique()
```

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

```
[ ] # Filter data for male customers and get the value counts
    male_data = df[df["gender"] == "Male"][["InternetService", "Churn"]].value_counts().unstack(fill_valu

    # Create a figure and axis
    fig, ax = plt.subplots(figsize=(12, 6))

    # Create a grouped bar plot
    male_data.plot(kind='bar', ax=ax)

    # Customize the plot
    plt.title('Internet Service and Churn Distribution for Male Customers')
    plt.xlabel('Internet Service')
    plt.ylabel('Count')
    plt.legend(title='Churn')

    # Rotate x-axis labels for better readability
    plt.xticks(rotation=0)

    # Add value labels on the bars
    for container in ax.containers:
        ax.bar_label(container)

    # Adjust layout
    plt.tight_layout()

    # Display the plot
    plt.show()

    # Optional: Display the data table
    print(male_data)
```

Customer_churn_Project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

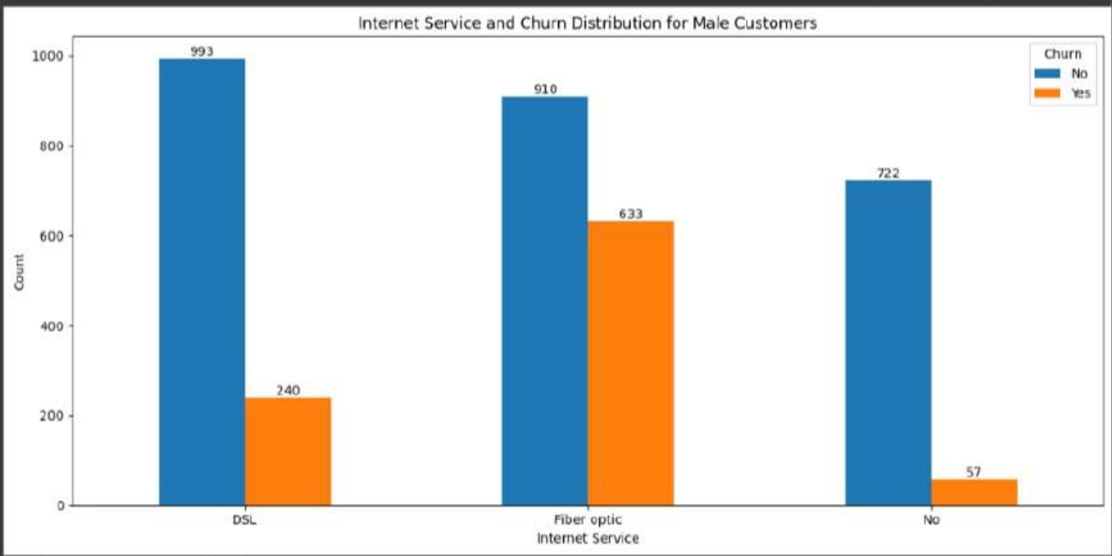
Comment Share

+ Code + Text Connect Gemini

```
[ ] df["InternetService"].unique()
```

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

```
[ ] # Filter data for male customers and get the value counts
male_data = df[df["gender"] == "Male"][["InternetService", "Churn"]].value_counts().unstack(fill_valu

# Create a figure and axis
fig, ax = plt.subplots(figsize=(12, 6))

# Create a grouped bar plot
male_data.plot(kind='bar', ax=ax)

# Customize the plot
plt.title('Internet Service and Churn Distribution for Male Customers')
plt.xlabel('Internet Service')
plt.ylabel('Count')
plt.legend(title='Churn')

# Rotate x-axis labels for better readability
plt.xticks(rotation=0)

# Add value labels on the bars
for container in ax.containers:
    ax.bar_label(container)

# Adjust layout
plt.tight_layout()

# Display the plot
plt.show()

# Optional: Display the data table
print(male_data)
```

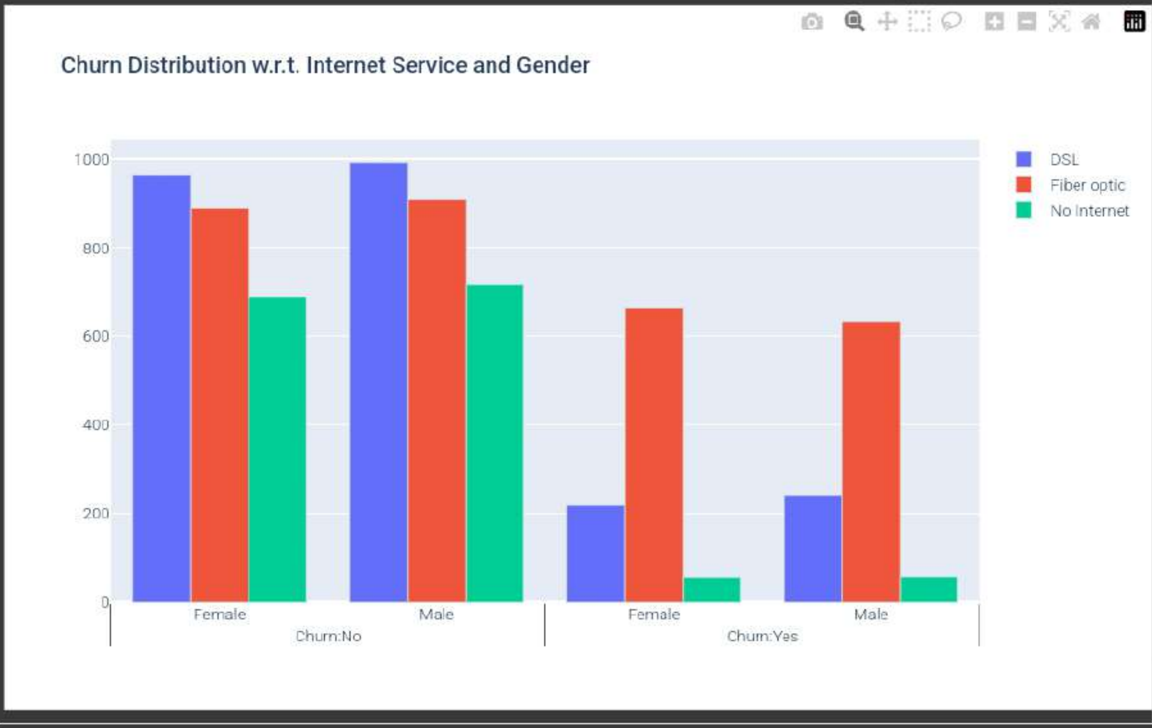



```
Churn            No  Yes
InternetService
DSL             993  240
Fiber optic     910  633
No              722   57
```

```
[ ] 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',
```

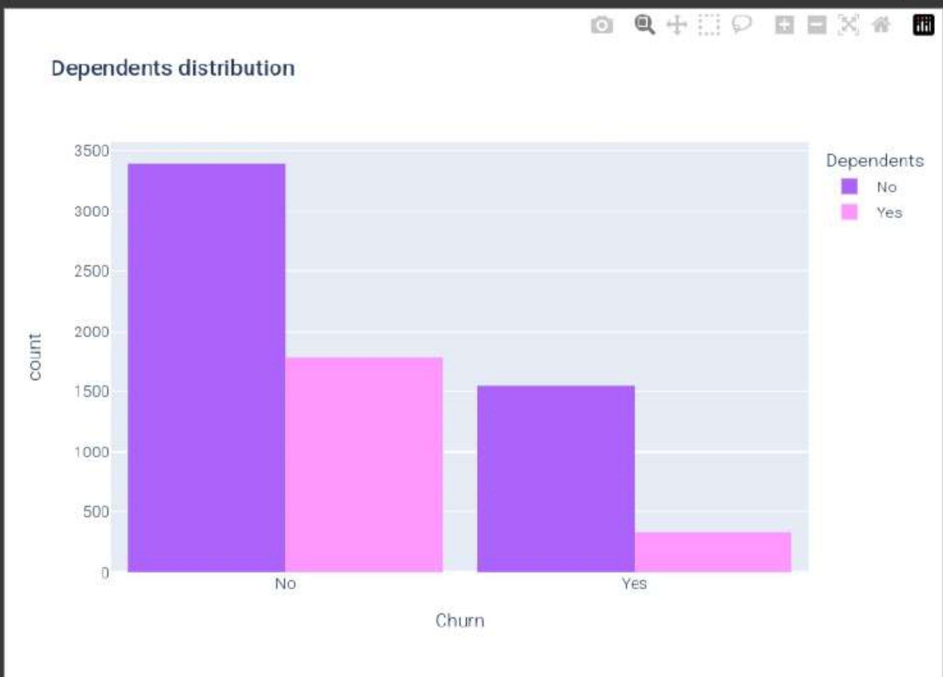
Customer_churn_Project.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text Connect Gemini

```
    name = 'No Internet',
))

fig.update_layout(title_text="<b>Churn Distribution w.r.t. Internet Service and Gender</b>")

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.

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



Customers without dependents are more likely to churn

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

CO Customer_churn_Project.ipynb

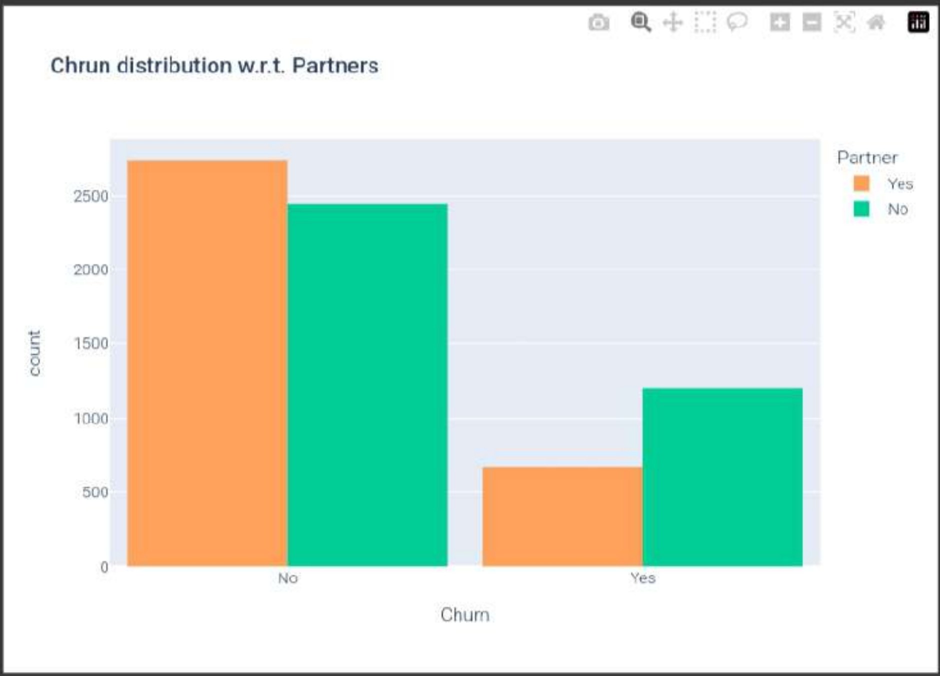
File Edit View Insert Runtime Tools Help All changes saved

Comment Share S

+ Code + Text Connect Gemini

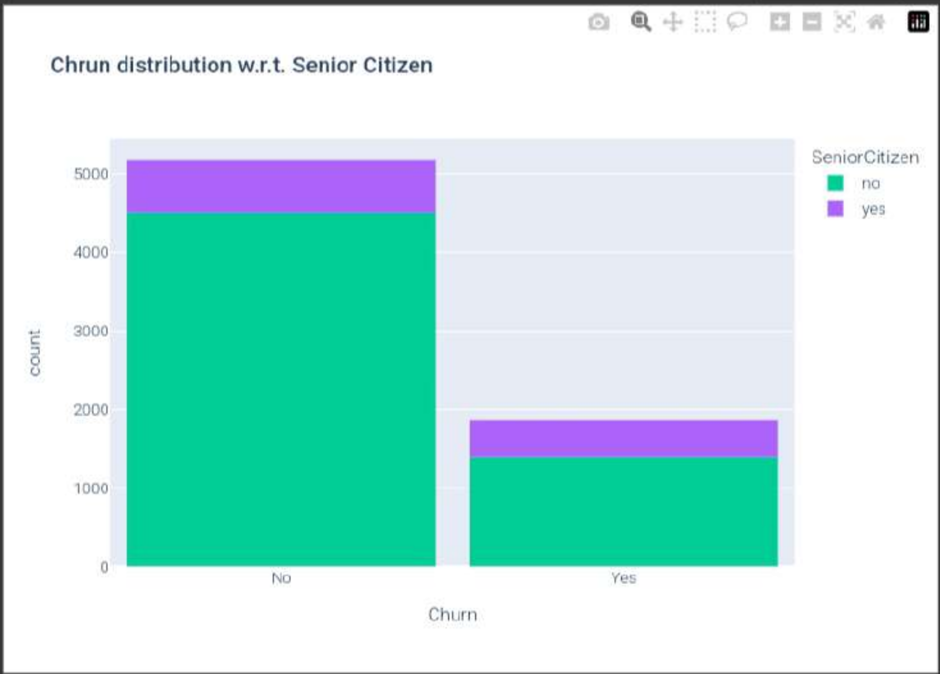
Customers without dependents are more likely to churn

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



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

```
[ ] color_map = {"Yes": '#00CC96', "No": '#B6E880'}
    fig = px.histogram(df, x="Churn", color="SeniorCitizen", title="<b>Chrun distribution w.r.t. Senior C
    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.
- Less num of senior citizen churn

```
[ ] color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
    fig = px.histogram(df, x="Churn", color="OnlineSecurity", barmode="group", title="<b>Churn w.r.t Onli
    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.
- Less num of senior citizen churn

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

- Most customers churn in the absence of online security

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

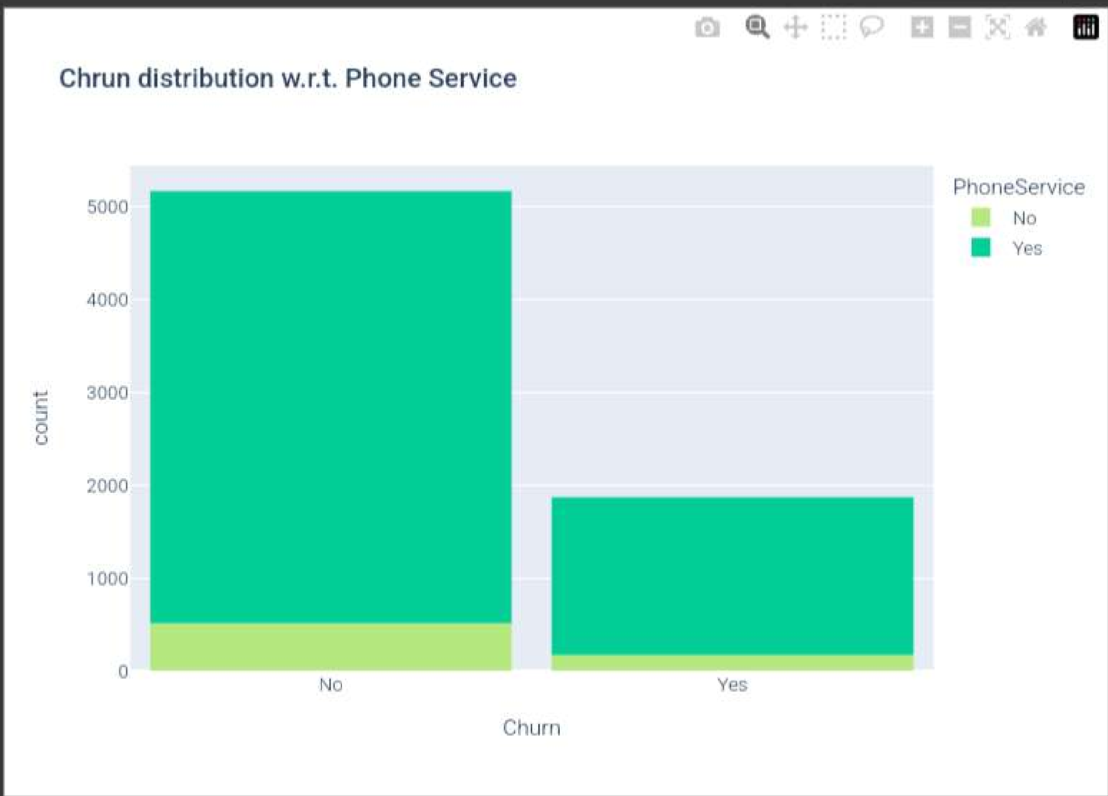
- Customers with Paperless Billing are most likely to churn.

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

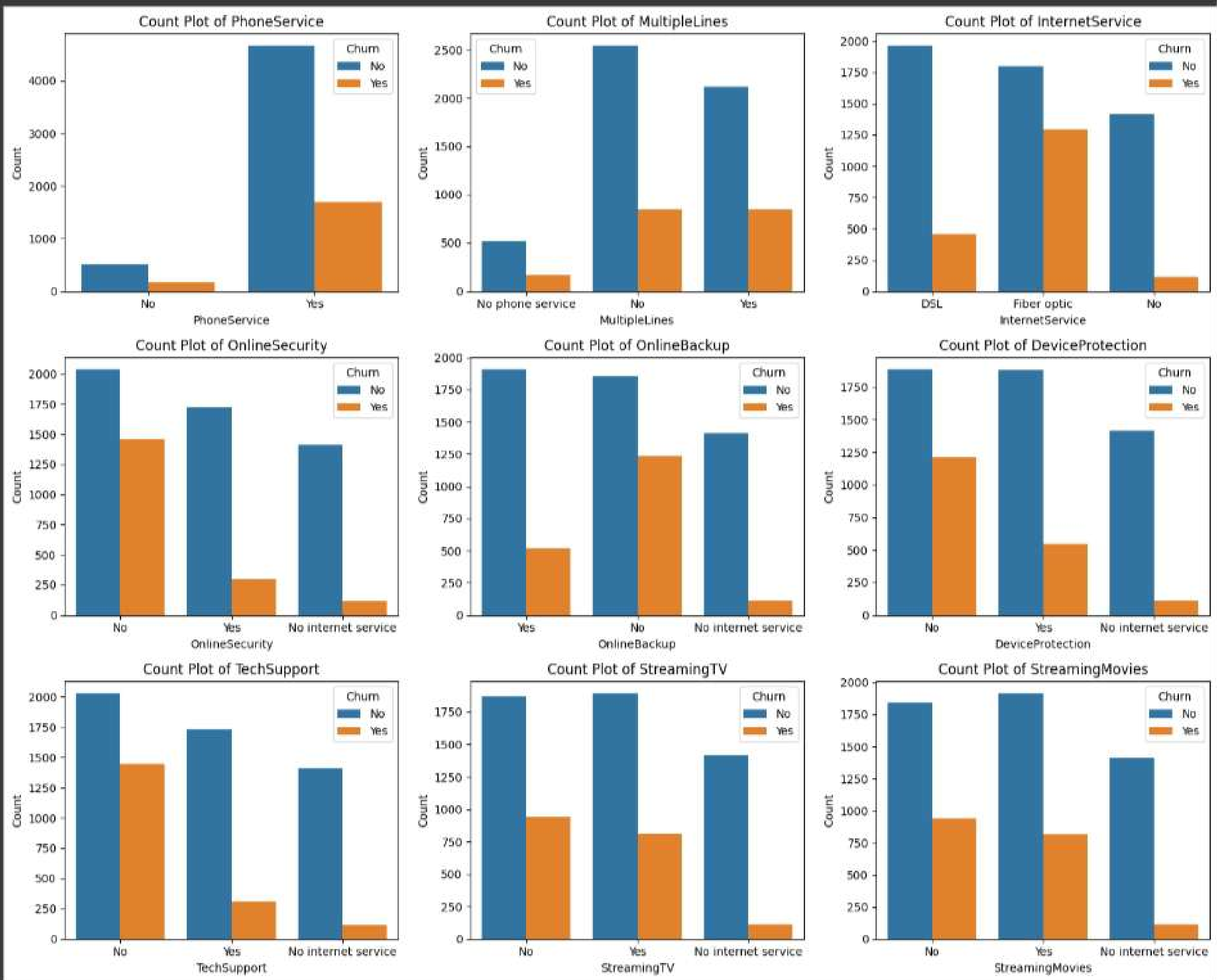
- Customers with no TechSupport are most likely to migrate to another service provider.
- Customers with no TechSupport are Churn.

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



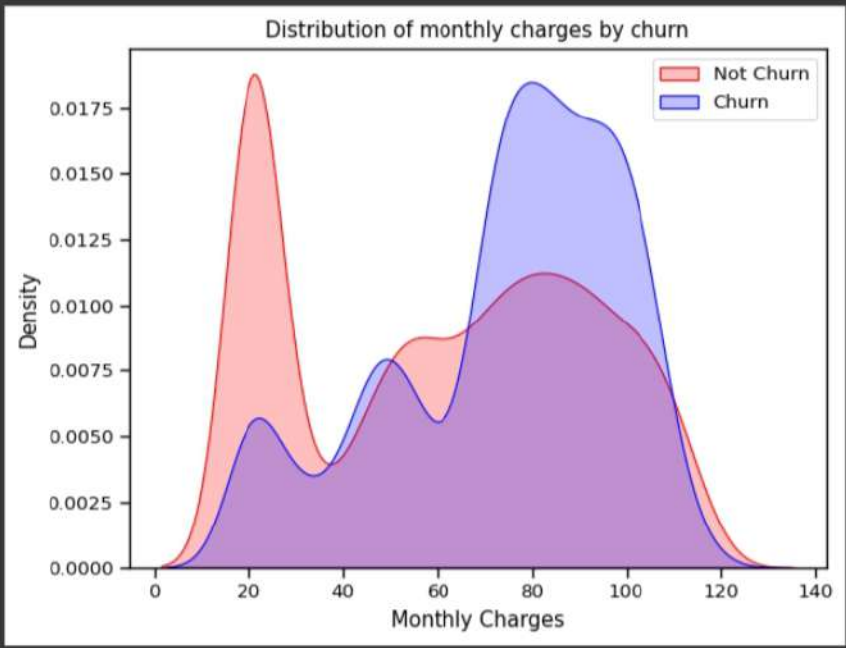
- Customers with Phone service are most likely to Churn
- Need to look Phone service service and take neccessary action for it

```python
columns = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
           'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

# Number of columns for the subplot grid (you can change this)
n_cols = 3
n_rows = (len(columns) + n_cols - 1) // n_cols  # Calculate number of rows needed

# Create subplots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4))  # Adjust figsize as needed

# Flatten the axes array for easy iteration (handles both 1D and 2D arrays)
axes = axes.flatten()

# Iterate over columns and plot count plots
for i, col in enumerate(columns):
    sns.countplot(x=col, data=df, ax=axes[i], hue = df["Churn"])
    axes[i].set_title(f'Count Plot of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')

# Remove empty subplots (if any)
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



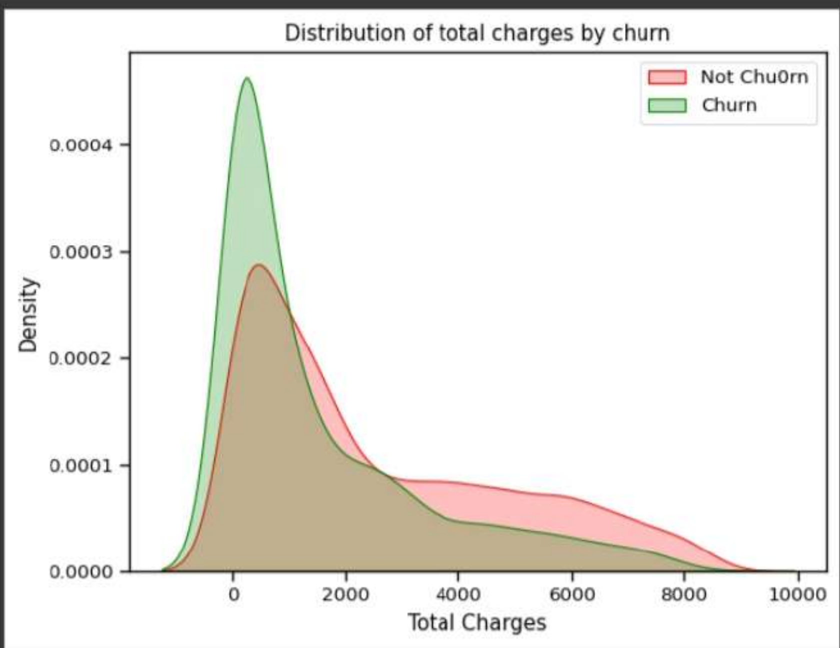
- Very small fraction of customers don't have a phone service and out of that, 1/3rd Customers are more likely to churn

```python
[ ] 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

```python
[ ] ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'No') ],
                    color="red", 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');
```

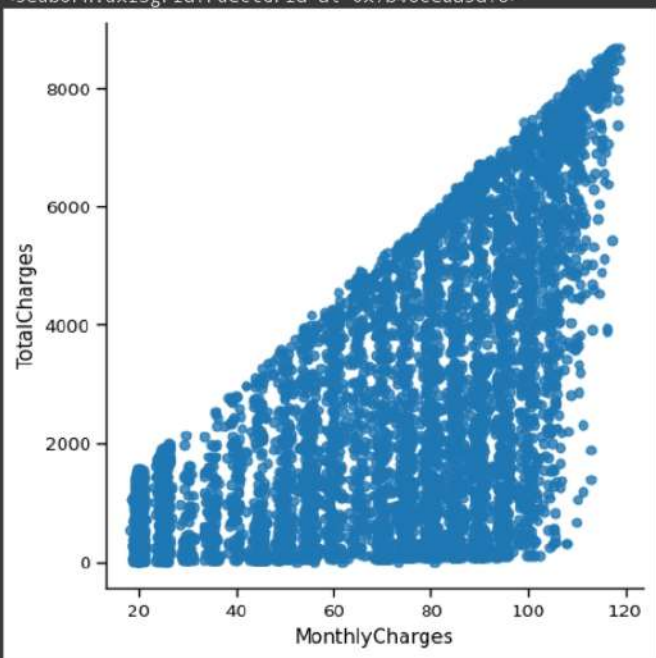



```python
[ ] fig = px.box(df, x='Churn', y = 'tenure')

    # 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'),
```

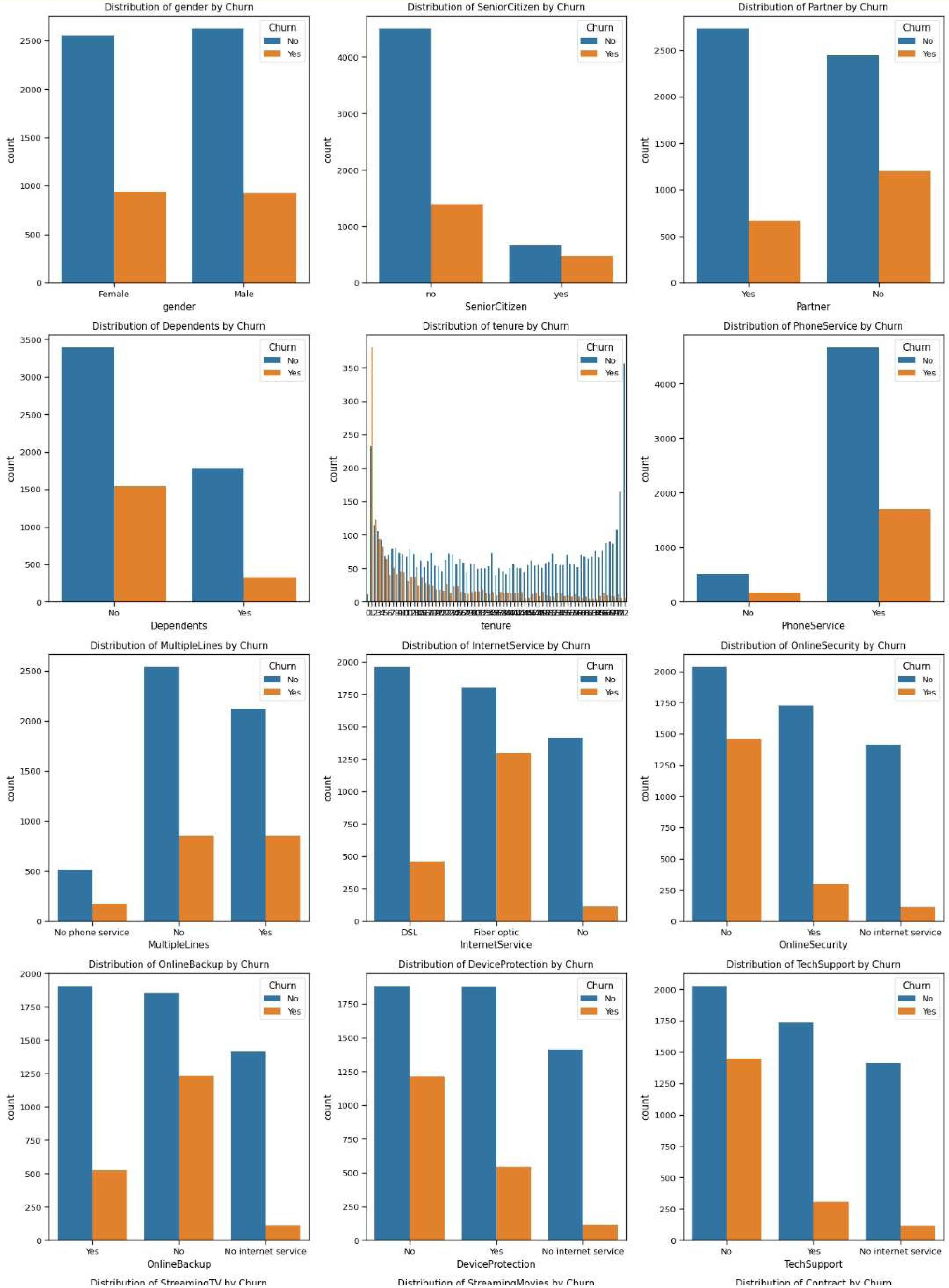
```
fig = px.box(df, x='Churn', y = 'tenure')

# 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()
```


```
# Relationship between Monthly Charges and Total Charges
sns.lmplot(data=df, x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

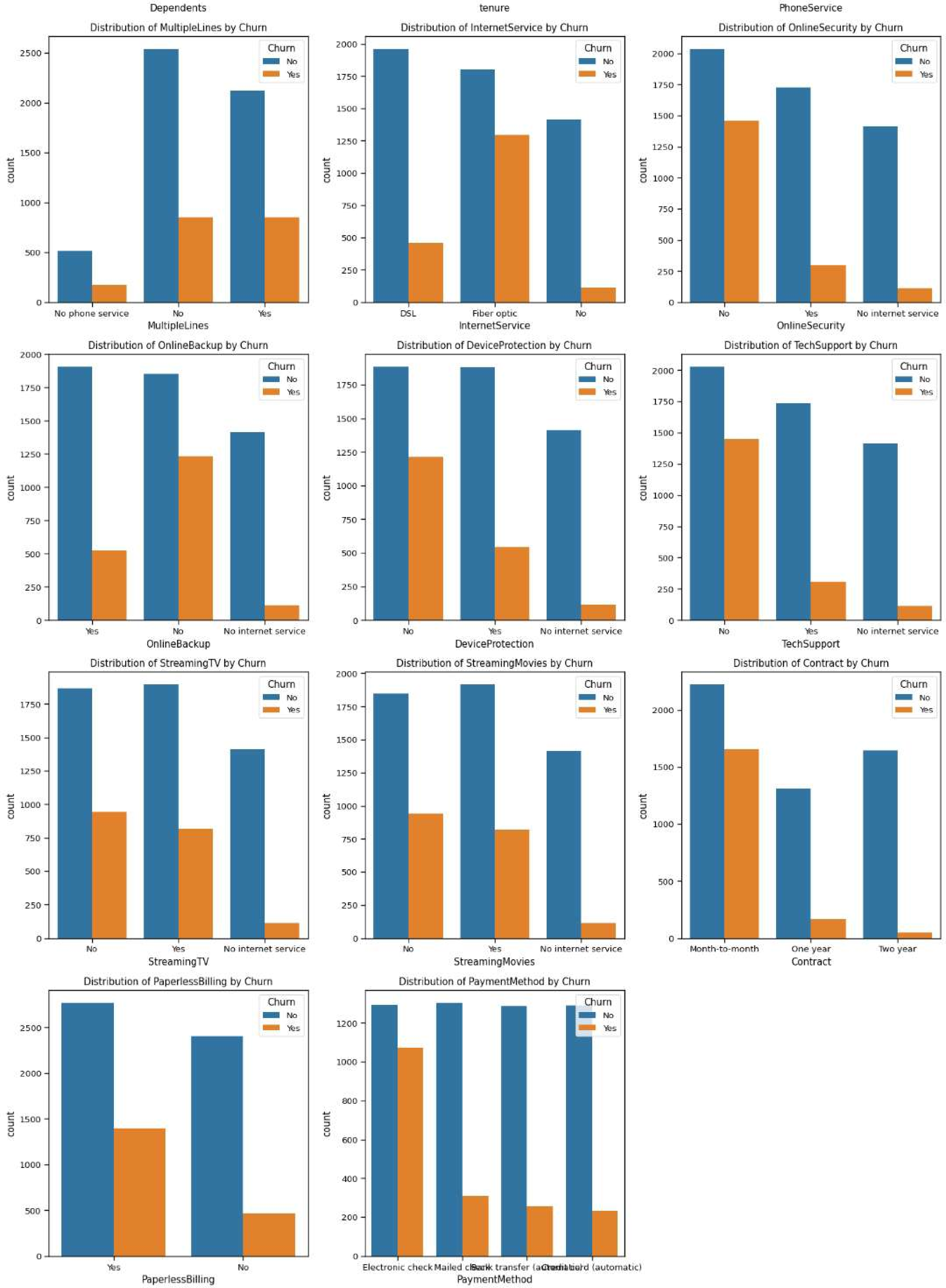
```
<seaborn.axisgrid.FacetGrid at 0x7b40ecaa3df0>
```



- Total Charges increase as Monthly Charges increase - as expected.



Distribution of gender by Churn
Churn
No
Yes
count
Female
Male
gender
Distribution of SeniorCitizen by Churn
no
yes
SeniorCitizen
Distribution of Partner by Churn
Yes
No
Partner
Distribution of Dependents by Churn
No
Yes
Dependents
Distribution of tenure by Churn
tenure
Distribution of PhoneService by Churn
No
Yes
PhoneService
Distribution of MultipleLines by Churn
No phone service
No
Yes
MultipleLines
Distribution of InternetService by Churn
DSL
Fiber optic
No
InternetService
Distribution of OnlineSecurity by Churn
No
Yes
No internet service
OnlineSecurity
Distribution of OnlineBackup by Churn
Yes
No
No internet service
OnlineBackup
Distribution of DeviceProtection by Churn
No
Yes
No internet service
DeviceProtection
Distribution of TechSupport by Churn
No
Yes
No internet service
TechSupport
Distribution of StreamingTV by Churn
Distribution of StreamingMovies by Churn
Distribution of Contract by Churn


Dependents
tenure
PhoneService
Distribution of MultipleLines by Churn
Churn
No
Yes
count
No phone service
No
Yes
MultipleLines
Distribution of InternetService by Churn
DSL
Fiber optic
No
InternetService
Distribution of OnlineSecurity by Churn
No
Yes
No internet service
OnlineSecurity
Distribution of OnlineBackup by Churn
Yes
No
No internet service
OnlineBackup
Distribution of DeviceProtection by Churn
No
Yes
No internet service
DeviceProtection
Distribution of TechSupport by Churn
No
Yes
No internet service
TechSupport
Distribution of StreamingTV by Churn
No
Yes
No internet service
StreamingTV
Distribution of StreamingMovies by Churn
No
Yes
No internet service
StreamingMovies
Distribution of Contract by Churn
Month-to-month
One year
Two year
Contract
Distribution of PaperlessBilling by Churn
Yes
No
PaperlessBilling
Distribution of PaymentMethod by Churn
Electronic check
PaymentMethod

Customer_churn_Project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Gemini

Training and Testing - dividing dataset into train test split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
#FEATURE SCALING

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Modelling - Implementing Algorithm

1. LogisticRegression

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
#Predicting the result
y_pred = logreg.predict(x_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[924  93]
 [195 197]]
```

```
#This is to get the Models Accuracy
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test, y_pred)
print(ac)
```

```
0.7955997161107168
```

```
#This is to get the Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
             precision    recall  f1-score

           0       0.83      0.91      0.87
           1       0.68      0.50      0.58

    accuracy                           0.80
   macro avg       0.75      0.71      0.72
 eighted avg       0.78      0.80      0.79
```

```
bias = logreg.score(x_train, y_train)
bias
```

```
0.8074192403265885
```

```
variance = logreg.score(x_test, y_test)
variance
```

```
0.7955997161107168
```

Customer_churn_Project.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share

+ Code + Text Connect Gemini

```
0.7955997161107168
```

2. Decision Tree Classifier - without Resampling

```
from sklearn.tree import DecisionTreeClassifier
model_dt=DecisionTreeClassifier(criterion = "gini",random_state = 0,max_depth=150, min_samples_leaf=1
model_dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=150, min_samples_leaf=100, random_state=0)
```

```
y_pred=model_dt.predict(x_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
from sklearn import metrics
model_dt.score(x_test,y_test)
```

```
0.7934705464868701
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, labels=[0,1]))
```

	precision	recall	f1-score
0	0.81	0.93	0.87
1	0.70	0.45	0.55
accuracy			0.79
macro avg	0.76	0.69	0.71
weighted avg	0.78	0.79	0.78

The Dataset is Imbalance lets implement SMOTE technique to balance the dataset

Decision Tree Classifier with Resampling

```
from imblearn.combine import SMOTEENN

# Initialize the SMOTEENN object
sm = SMOTEENN()

# Use fit_resample instead of fit_sample
X_resampled, y_resampled = sm.fit_resample(x, y)
```

```
xr_train,xr_test,yr_train,yr_test = train_test_split(X_resampled, y_resampled,test_size=0.2)
```

```
model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 0,max_depth=200, min_samples_
```

```
model_dt_smote.fit(xr_train,yr_train)
model_dt_smote
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=200, min_samples_leaf=80, random_state=0)
```

```
yr_predict = model_dt_smote.predict(xr_test)
yr_predict
```

```
array([0, 0, 1, ..., 1, 1, 1])
```

```
from sklearn.metrics import confusion_matrix
```

The Dataset is Imbalance lets implement SMOTE technique to balance the dataset

Decision Tree Classifier with Resampling

```
from imblearn.combine import SMOTEENN

# Initialize the SMOTEENN object
sm = SMOTEENN()

# Use fit_resample instead of fit_sample
X_resampled, y_resampled = sm.fit_resample(x, y)
```

```
xr_train,xr_test,yr_train,yr_test = train_test_split(X_resampled, y_resampled,test_size=0.2)
```

```
model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 0,max_depth=200, min_samples_
```

```
model_dt_smote.fit(xr_train,yr_train)
model_dt_smote
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=200, min_samples_leaf=80, random_state=0)
```

```
yr_predict = model_dt_smote.predict(xr_test)
yr_predict
```

```
array([0, 0, 1, ..., 1, 1, 1])
```

```
from sklearn.metrics import confusion_matrix
print(metrics.confusion_matrix(yr_test, yr_predict))
```

```
[[497  42]
 [ 56 577]]
```

```
model_score_r = model_dt_smote.score(xr_test, yr_test)
model_score_r
```

```
0.9163822525597269
```

```
from sklearn.metrics import classification_report
print(classification_report(yr_test, yr_predict, labels=[0,1]))
```

	precision	recall	f1-score	support
0	0.90	0.92	0.91	539
1	0.93	0.91	0.92	633
acy			0.92	1172
avg	0.92	0.92	0.92	1172
avg	0.92	0.92	0.92	1172

Customer_churn_Project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
RandomForestClassifier
RandomForestClassifier()
```

```python
# Predicting the Test set results
y_pred_rf = model_rf.predict(x_test)
y_pred_rf
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```python
from sklearn.metrics import confusion_matrix
print(metrics.confusion_matrix(y_test, y_pred_rf))
```

```
[[914 103]
 [204 188]]
```

```python
# Accuracy check
model_rf.score(x_test,y_test)
```

```
0.7821149751596878
```

Random Forest Classifier - with Resampling

```python
from imblearn.combine import SMOTEENN
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score

# Initialize the SMOTEENN object
sm = SMOTEENN()

# Resample the data using SMOTEENN
X_resampled, y_resampled = sm.fit_resample(x, y)

# Split the resampled data into training and testing sets
xr_train, xr_test, yr_train, yr_test = train_test_split(X_resampled, y_resampled, test_size=0.2, rand

# Initialize the Random Forest Classifier
model_rf_smote = RandomForestClassifier(criterion="gini", random_state=0, n_estimators=100, max_depth

# Train the model on the resampled training data
model_rf_smote.fit(xr_train, yr_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=200, min_samples_leaf=80, random_state=0)
```

```python
# Make predictions on the test data
y_pred_rf_smote = model_rf_smote.predict(xr_test)
y_pred_rf_smote
```

```
array([0, 0, 1, ..., 0, 1, 1])
```

```python
# Print classification report
print(classification_report(yr_test, y_pred_rf_smote))

# Calculate and print the ROC-AUC score
y_pred_prob_rf_smote = model_rf_smote.predict_proba(xr_test)[:, 1]
roc_auc = roc_auc_score(yr_test, y_pred_prob_rf_smote)
print(f'ROC-AUC Score: {roc_auc}')
```

```
              precision    recall  f1-score

           0       0.94      0.89      0.92
           1       0.91      0.95      0.93

    accuracy                           0.92
   macro avg       0.93      0.92      0.92
weighted avg       0.92      0.92      0.92

ROC-AUC Score: 0.9794310761409466
```

Customer_churn_Project.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:55 PM
Comment Share S
+ Code + Text Connect Gemini

```
ROC-AUC Score: 0.9791227354979486
```

LogisticRegression - with Resampling

```
[ ] from imblearn.combine import SMOTEENN
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, roc_auc_score

    # Initialize the SMOTEENN object
    sm = SMOTEENN()

    # Resample the data using SMOTEENN
    X_resampled, y_resampled = sm.fit_resample(x, y)

    # Split the resampled data into training and testing sets
    xl_train, xl_test, yl_train, yl_test = train_test_split(X_resampled, y_resampled, test_size=0.2, rand

    # Initialize the Logistic Regression model
    model_lr_smote = LogisticRegression(random_state=0, max_iter=2000)

    # Train the model on the resampled training data
    model_lr_smote.fit(xl_train, yl_train)

    # Make predictions on the test data
    y_pred_lr_smote = model_lr_smote.predict(xl_test)

    # Print classification report
    print(classification_report(yl_test, y_pred_lr_smote))

    # Calculate and print the ROC-AUC score
    y_pred_prob_lr_smote = model_lr_smote.predict_proba(xl_test)[:, 1]
    roc_auc = roc_auc_score(yl_test, y_pred_prob_lr_smote)
    print(f'ROC-AUC Score: {roc_auc}')
```

```
              precision    recall  f1-score

           0       0.95      0.95      0.95
           1       0.96      0.96      0.96

    accuracy                           0.96
   macro avg       0.96      0.96      0.96
weighted avg       0.96      0.96      0.96

ROC-AUC Score: 0.9921840538989094
```

Logistic regression, Decision tree classifier and Random forest classifier all algorithms are working well but **Logistic regression working much better with 96% accuracy**

Explore code / https://github.com/SameerHussain128?tab=repositories

www.linkedin.com/in/mohdsameer28