```
Start coding or generate with AI.
```

CUSTOMER CHURN ANALYSIS IN A TELECOMMUNICATION COMPANY

```
#IMPORTING NECESSARY LIBRARIES
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
```

Loading the data file

```
df = pd.read_csv("/content/WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

Examine the top 5 rows of the dataset to have a glimpse of what it entails

```
df.head()
```

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

5 rows × 21 columns

Exploring the shape and column attributes of the data

```
df.shape
```

```
(7043, 21)
```

```
df.columns
```

```
Index(['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
```

only 3 columns have numerical attributes

```
df.describe()
```

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

The SeniorCitizen column is a categorical column which can be represented in a Yes(1) or No(0) pattern, therefore, its descriptive summary statistics is not considered.

The maximum tenure of customers is 72 months, the tenure column describes the number of months the customers have stayed with the company.

The average monthly charge is 64.76 USD.

Double-click (or enter) to edit

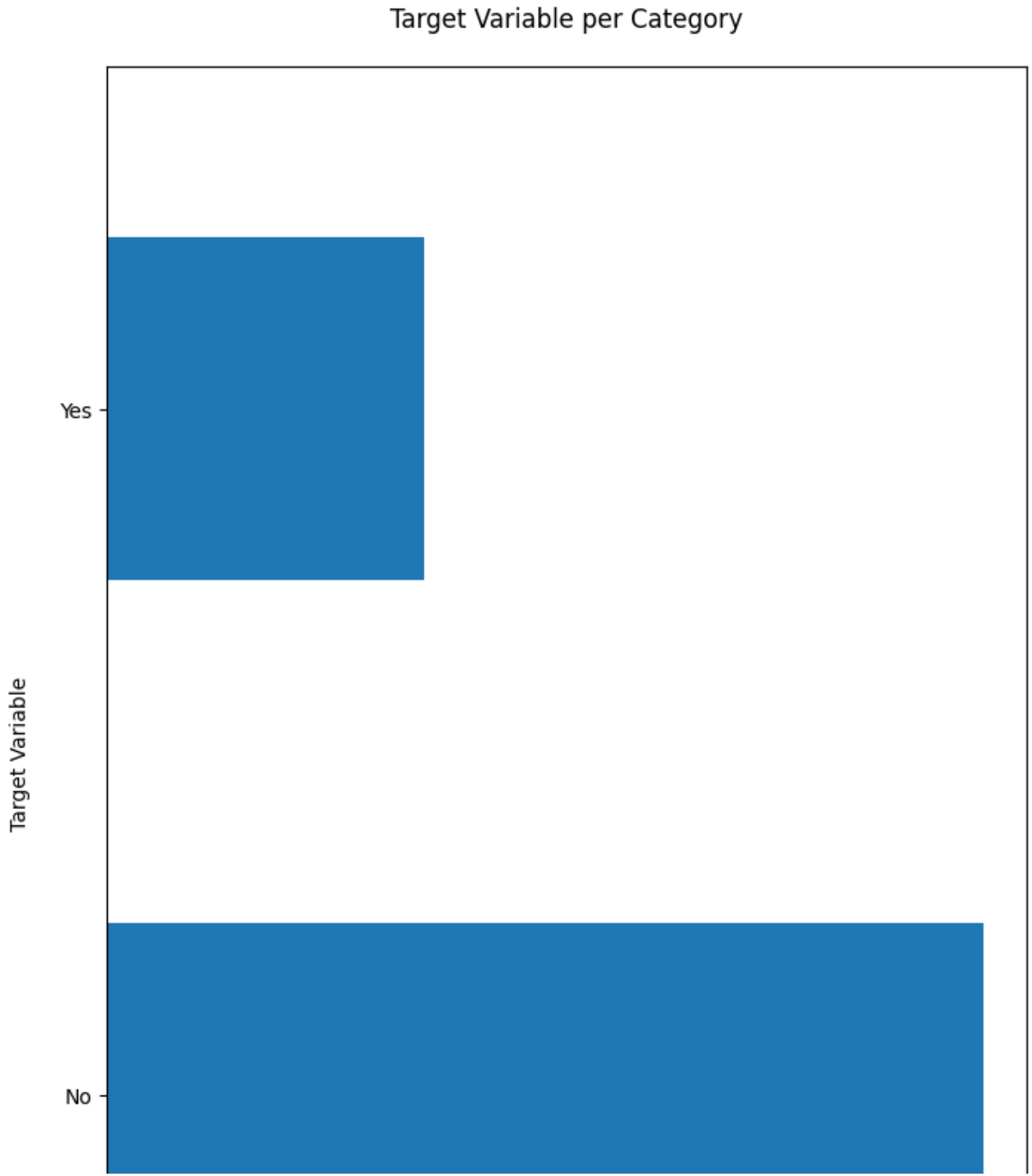
Being the target column, we should be able to analyse and viaualize it foor more insight.

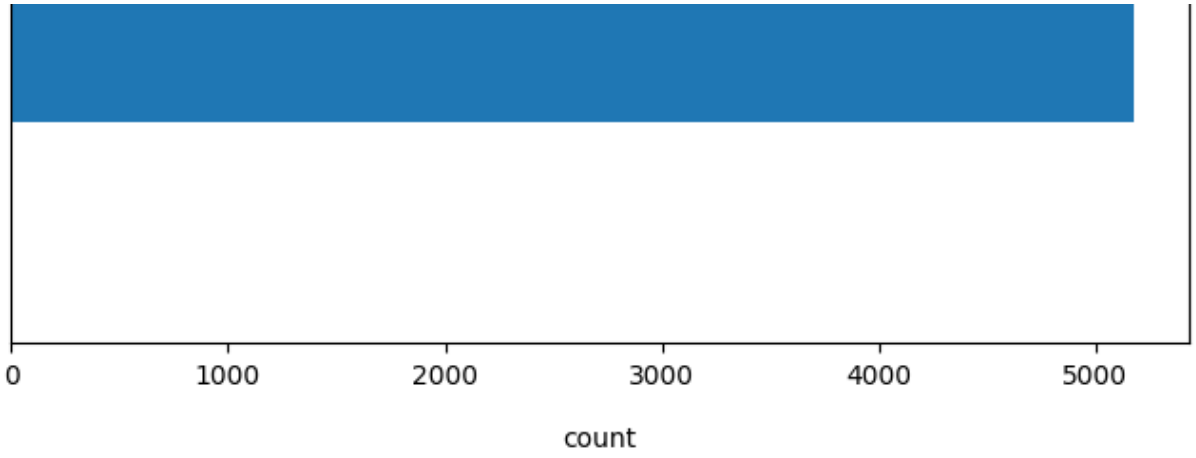
```
df["Churn"].value_counts()
```

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

```
#PLOTTING THE churn column counts for better visualization
df["Churn"].value_counts().plot(kind = "barh", figsize = (8, 12) )
plt.xlabel('count', labelpad=14)
plt.ylabel("Target Variable", labelpad= 14)
plt.title("Target Variable per Category", y = 1.02)
```

```
Text(0.5, 1.02, 'Target Variable per Category')
```



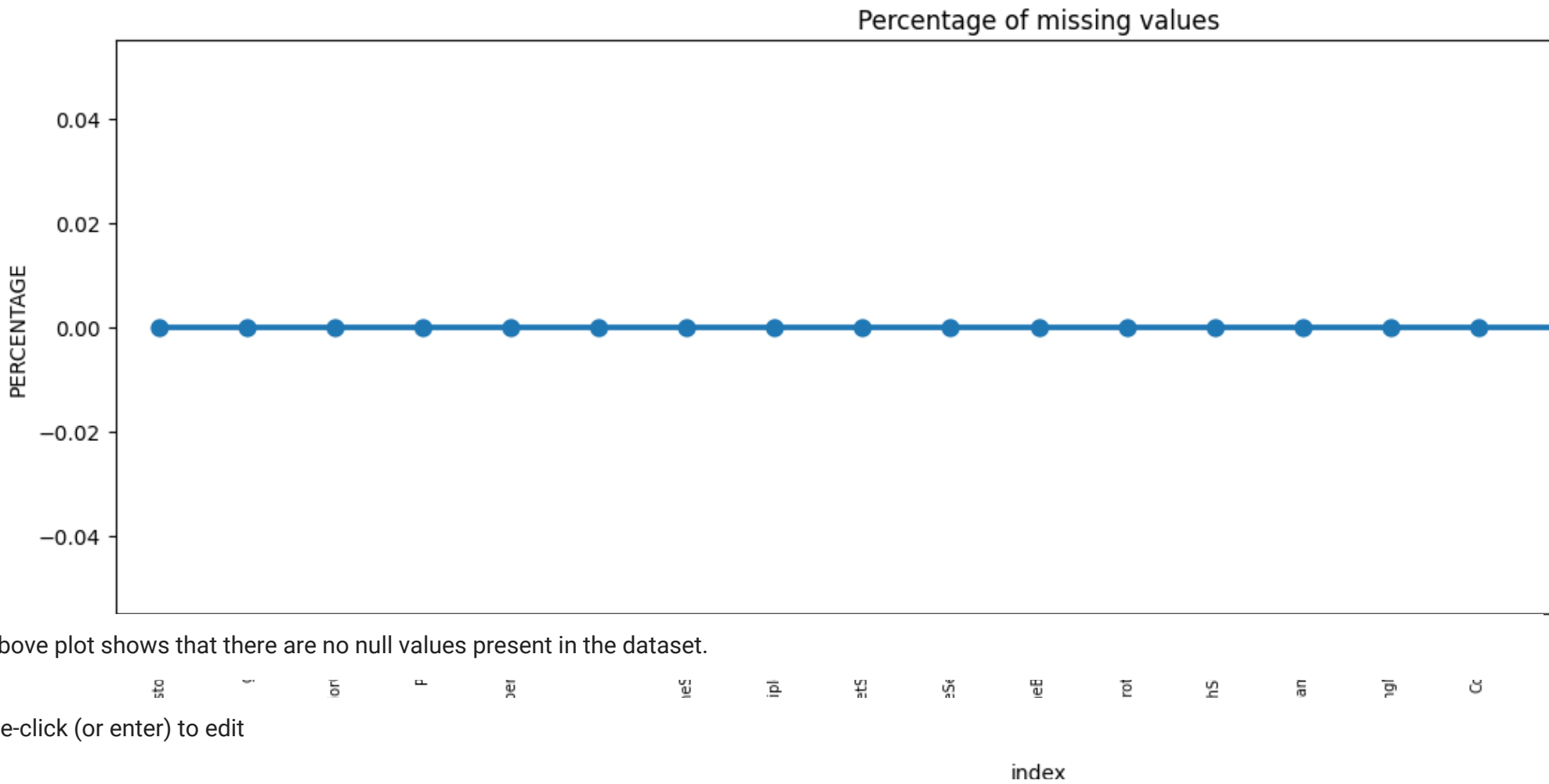



```
df["Churn"].value_counts() / len(df['Churn']) * 100
```

```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

The above percentage statistics shows that only 26.5% left the company or churned.

```
missing = pd.DataFrame((df.isnull().sum() * 100 / df.shape[0]).reset_index())

plt.figure(figsize=(16, 5))
ax = sns.pointplot(x="index", y=0, data=missing)  # Specify 'x' and 'y' arguments
plt.xticks(rotation=90, fontsize=7)
plt.title("Percentage of missing values")
plt.ylabel("PERCENTAGE")
plt.show()
```



The above plot shows that there are no null values present in the dataset.

Double-click (or enter) to edit

Data Cleaning

Creating a copy of this base dataset for data maanipulation and processing

```
new_df = df.copy()
```

We should note that total charges column should be numeric,so we will have to convert it to numerical or float datatype

```
# using the pd.to_numeric function
new_df.TotalCharges = pd.to_numeric(new_df.TotalCharges, errors = 'coerce')
new_df.isnull().sum()
```

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

There are 11 null values in the TotalCharges column, since the proportion of the missing values is significantly low, it is of no harm if we delete them.

```
#removing missing values
new_df.dropna(how = "any", inplace = True)
```

```
print(new_df['tenure'].max())
```

```
72
```

```
labels = [" {0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]
print(labels)
```

```
[' 1 - 12', ' 13 - 24', ' 25 - 36', ' 37 - 48', ' 49 - 60', ' 61 - 72']
```

```
#grouping the tenure in the bins of 12 months
labels =[" {0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]
new_df["tenure_group"] = pd.cut(new_df.tenure, range(1,80,12), right = False, labels = labels )
```

```
new_df["tenure_group"].value_counts()
```

```
1 - 12     2175
61 - 72    1407
13 - 24    1024
25 - 36     832
49 - 60     832
37 - 48     762
Name: tenure_group, dtype: int64
```

Remove columns not required for processing

```
new_df.drop(columns = ["customerID", 'tenure'], axis = 1, inplace = True)
```

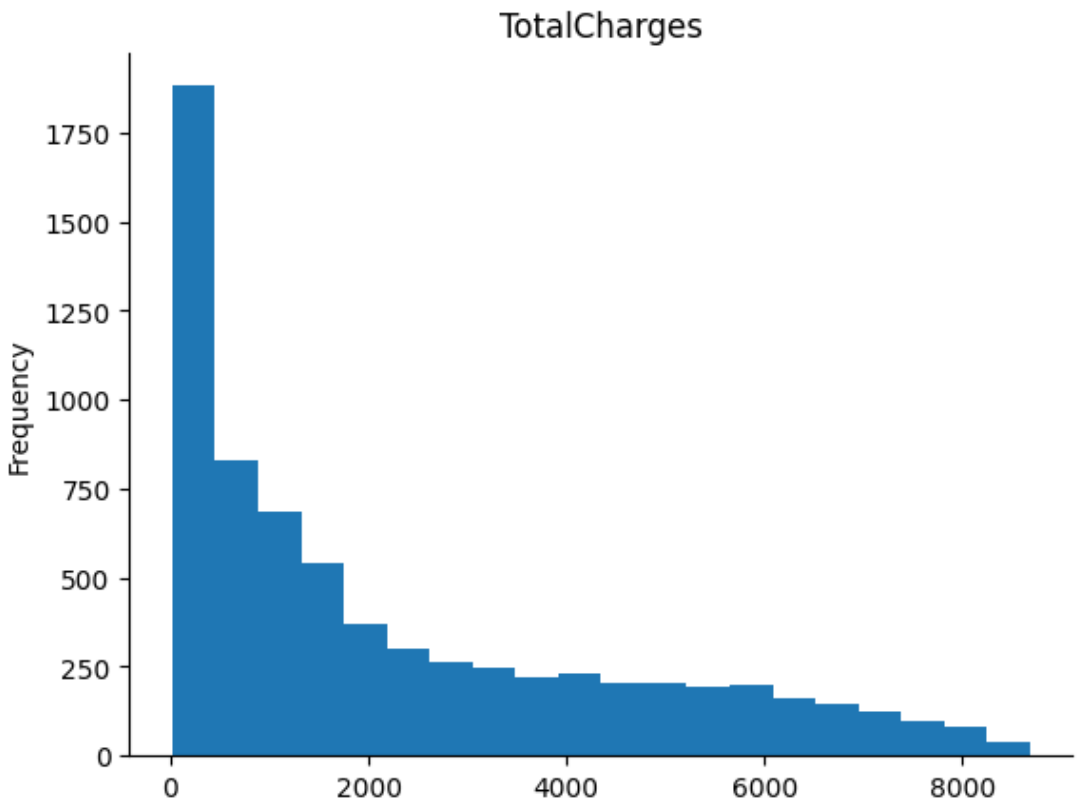
```
new_df.head()
```

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtecti
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	
1	Male	0	No	No	Yes	No	DSL	Yes	No	Y
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Y
4	Female	0	No	No	Yes	No	Fiber optic	No	No	

Next steps: Generate code with new_df View recommended plots

TotalCharges

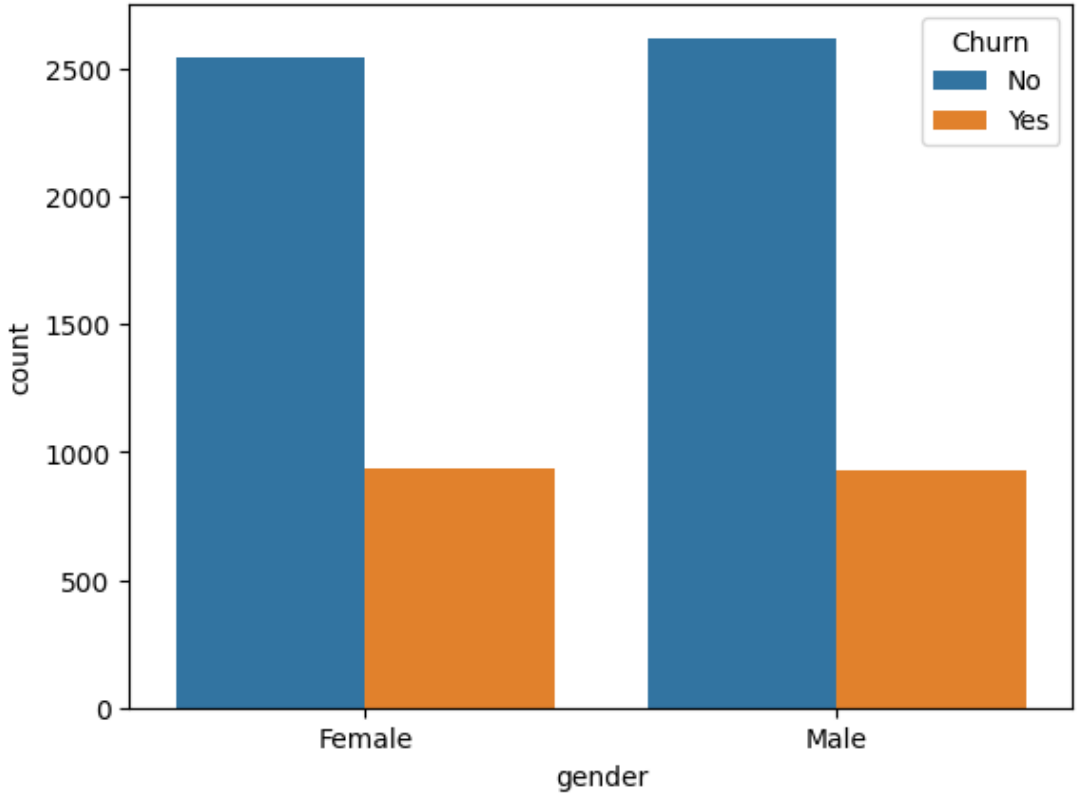
```
# @title TotalCharges

from matplotlib import pyplot as plt
new_df['TotalCharges'].plot(kind='hist', bins=20, title='TotalCharges')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

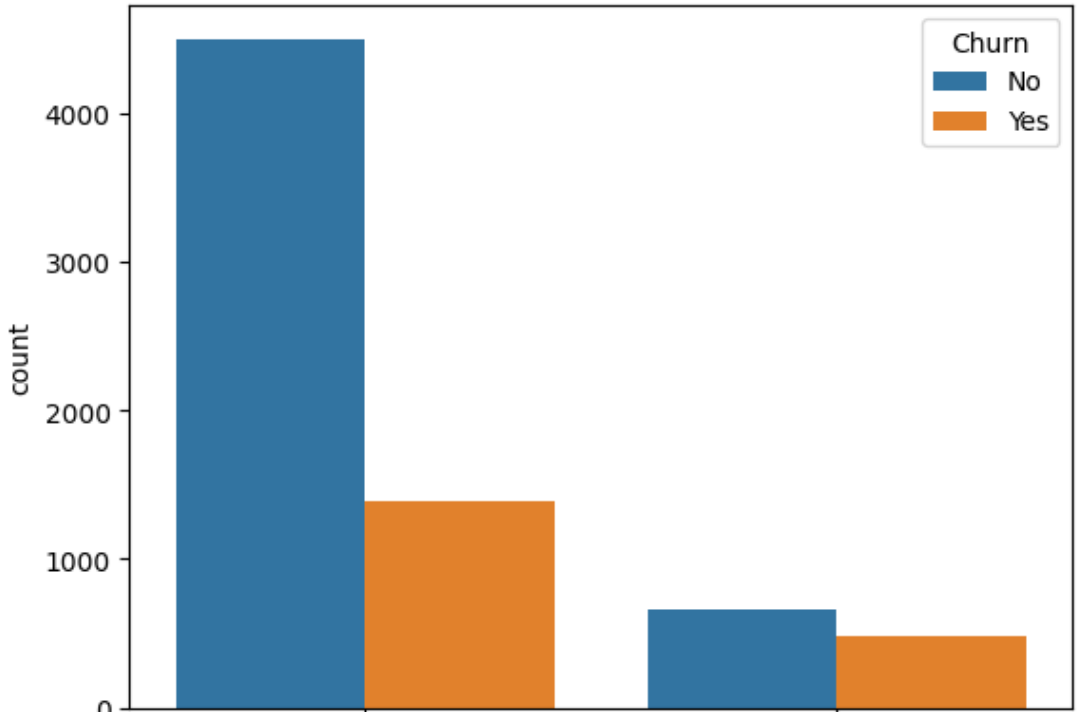


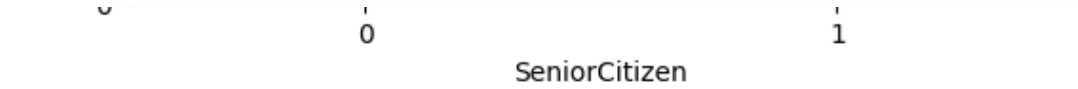
Data Exploration

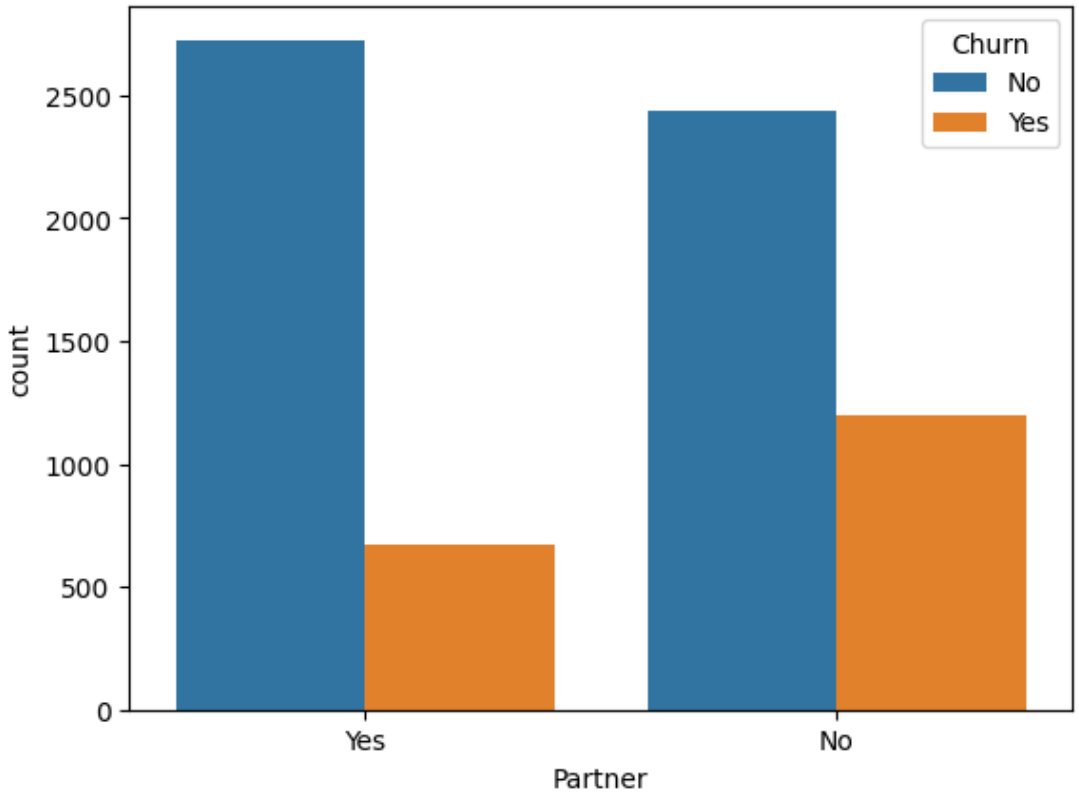
Univariate Analysis

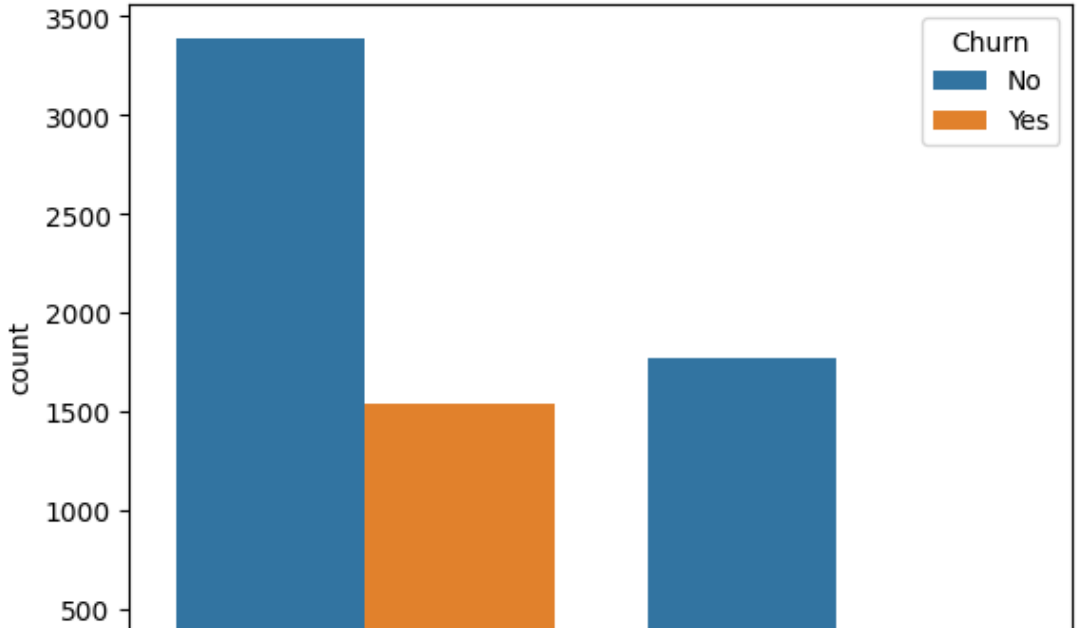
```
for i, predictor in enumerate(new_df.drop(columns = ['Churn', 'TotalCharges', 'MonthlyCharges'])):
  plt.figure(i)
  sns.countplot(data = new_df, x = predictor, hue = 'Churn')
```


Churn
No
Yes
2500
2000
1500
1000
500
0
count
Female
Male
gender

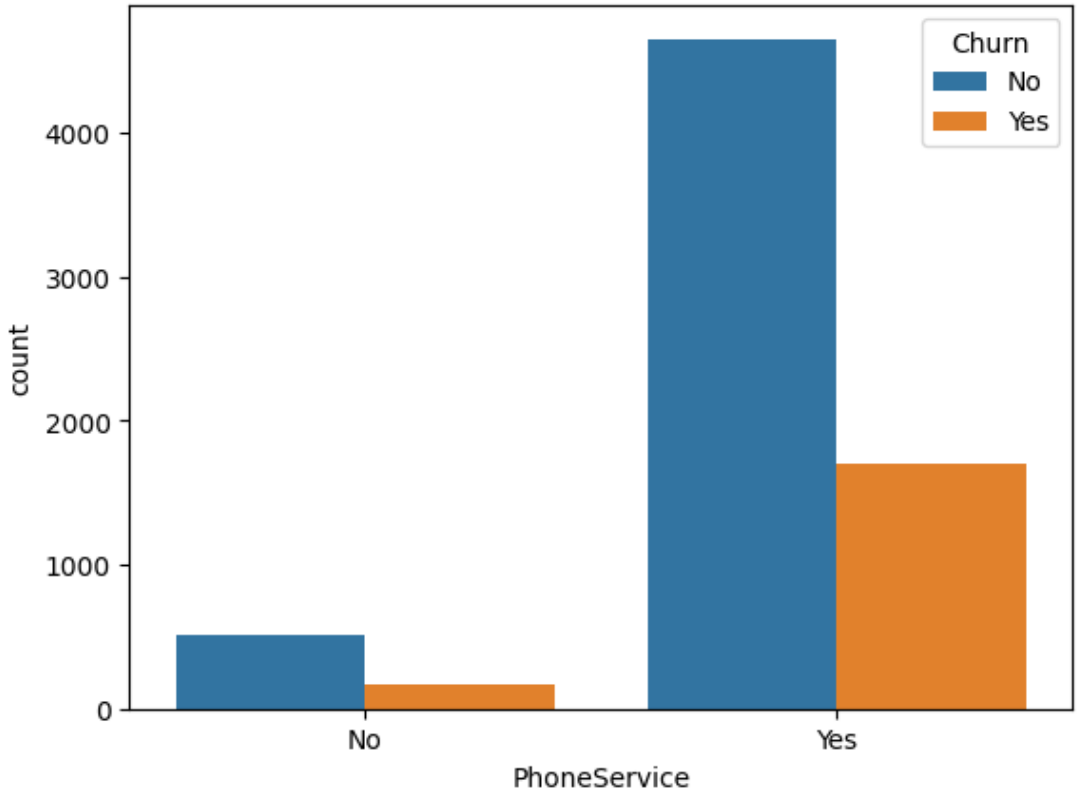

Churn
No
Yes
4000
3000
2000
1000
0
count

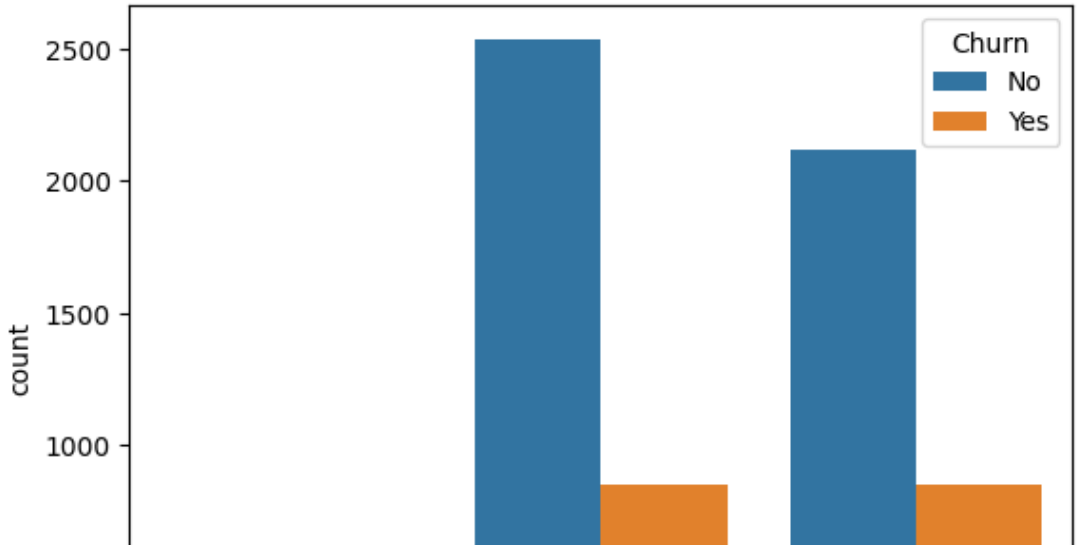

0
1
SeniorCitizen

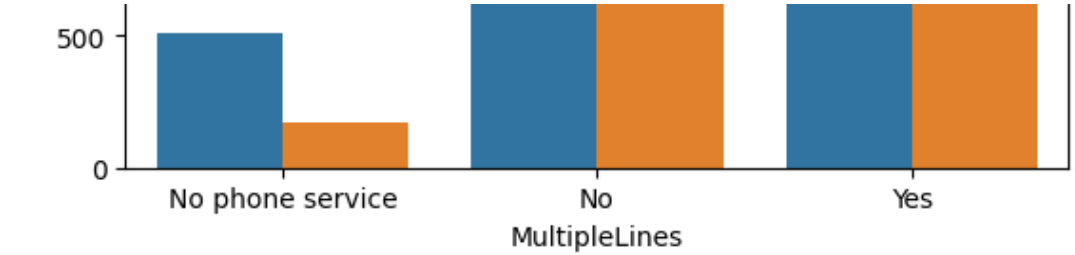

Churn
No
Yes
count
2500
2000
1500
1000
500
0
Yes
No
Partner

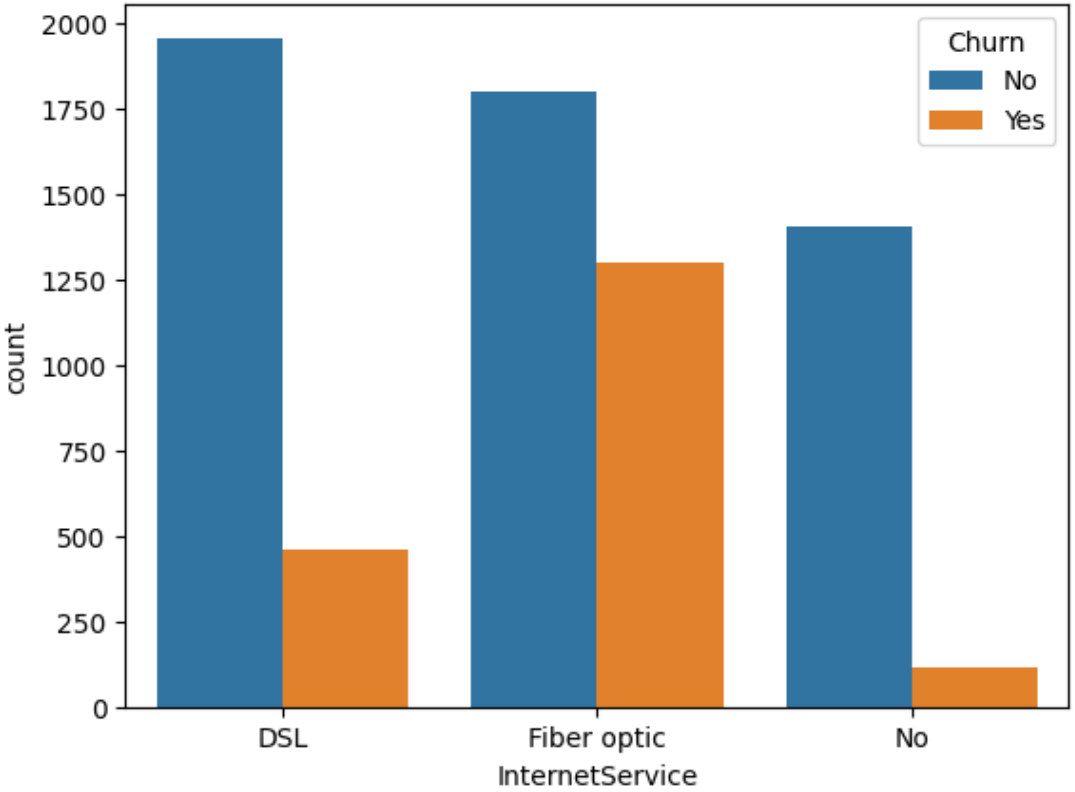

Churn
No
Yes
count
3500
3000
2500
2000
1500
1000
500

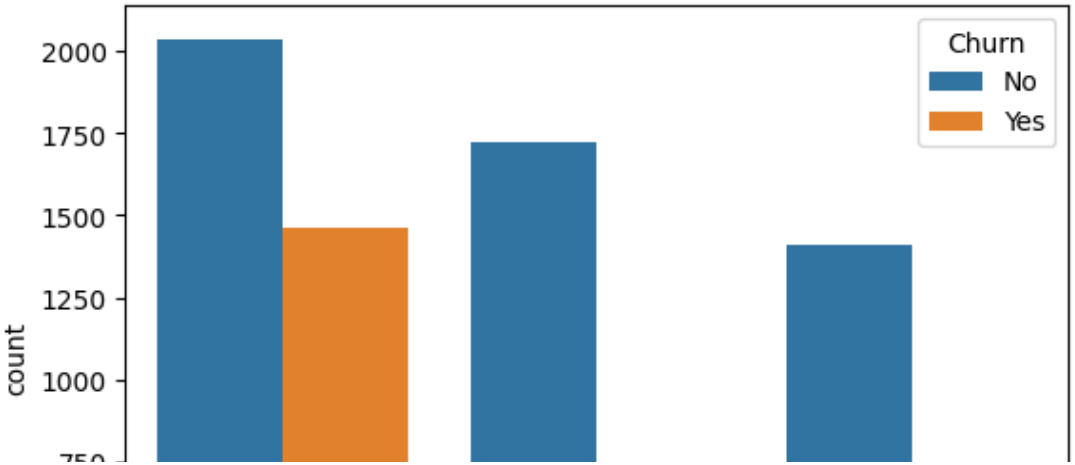

0
No
Yes
Dependents


Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PhoneService

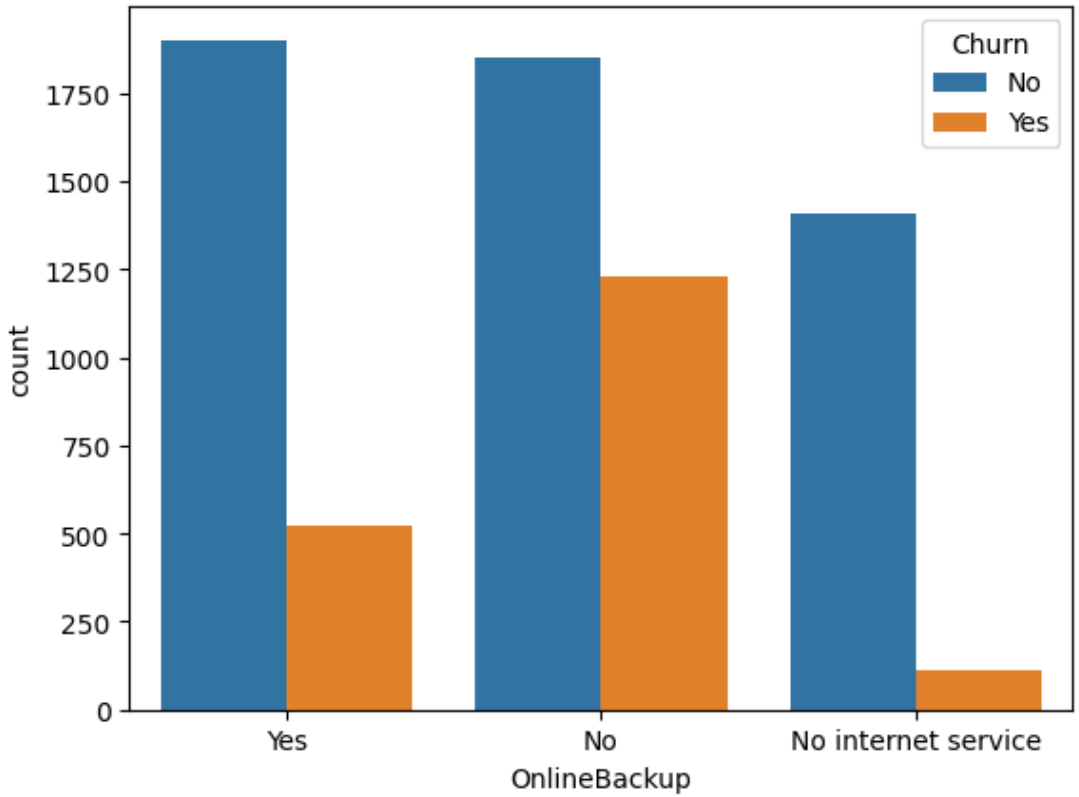

Churn
No
Yes
2500
2000
1500
1000
count

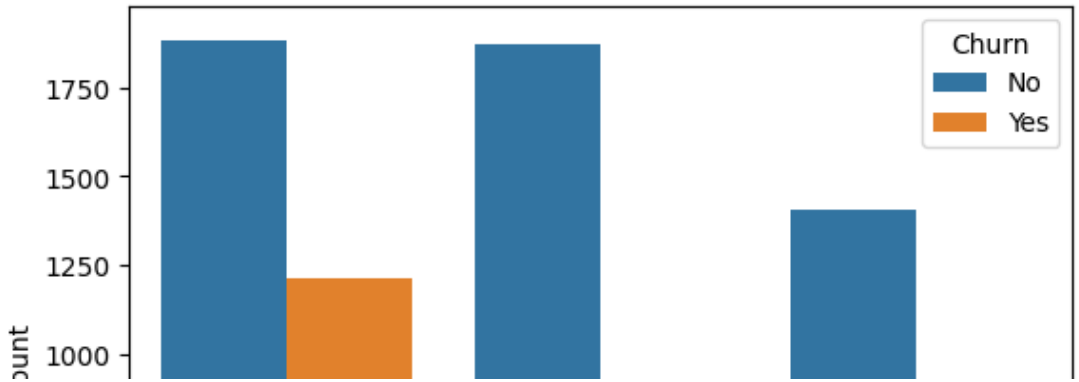

500
0
No phone service
No
Yes
MultipleLines

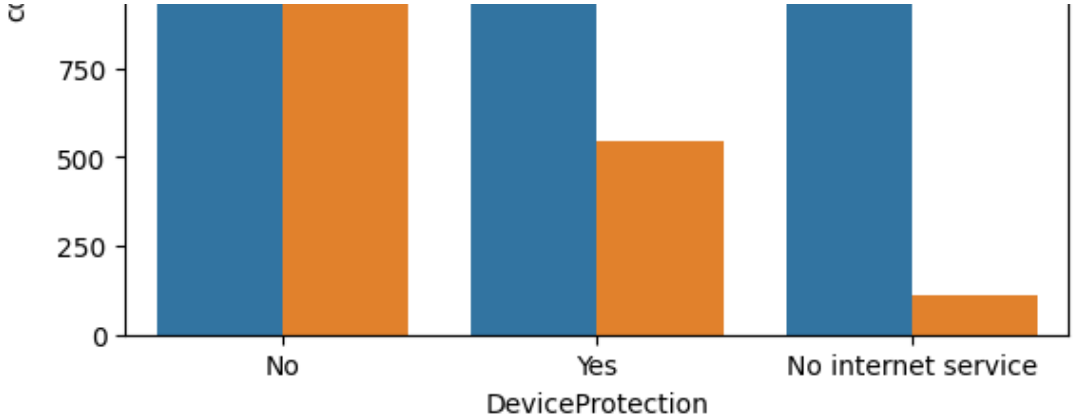

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
DSL
Fiber optic
No
InternetService

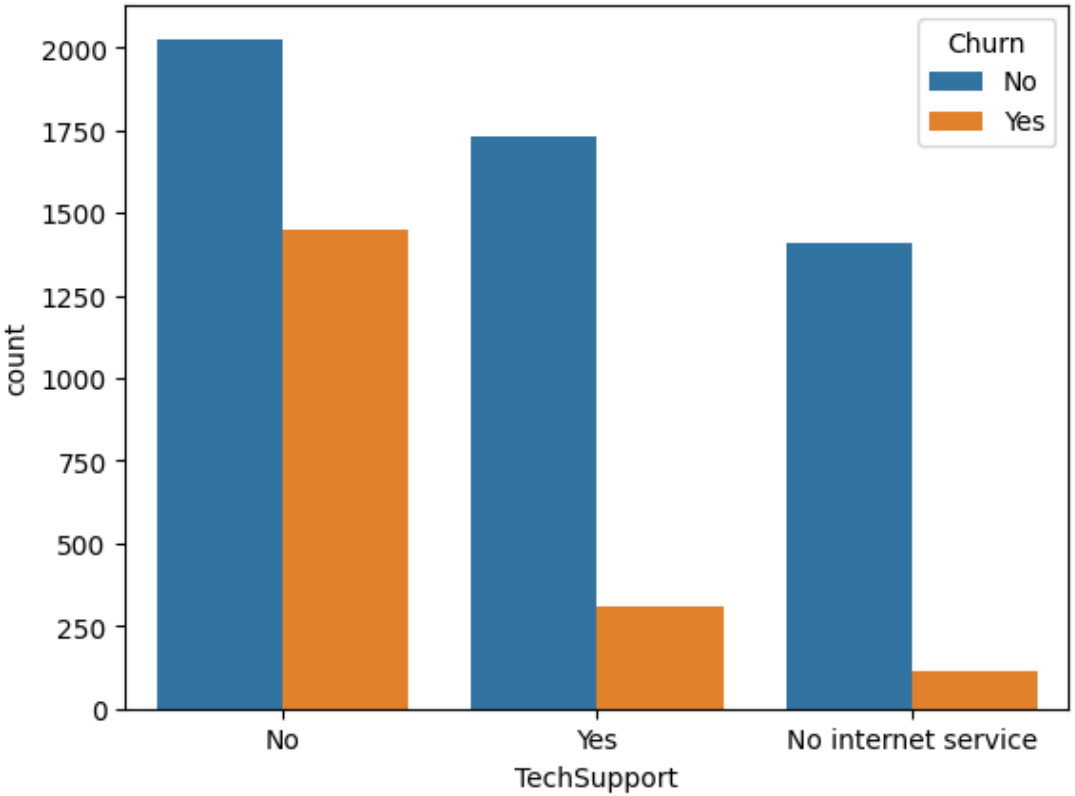

Churn
No
Yes
2000
1750
1500
1250
1000
count


750
500
250
0
No
Yes
No internet service
OnlineSecurity

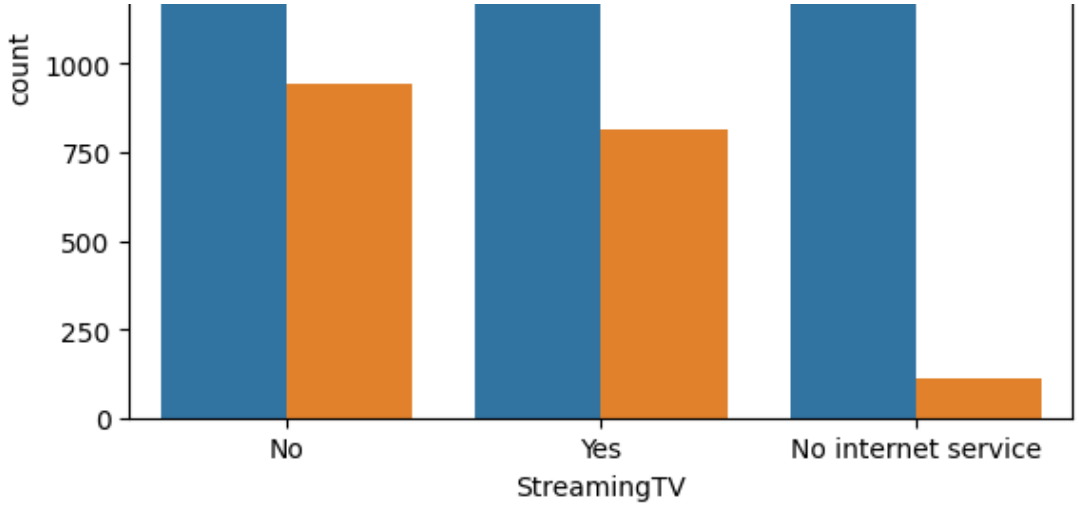

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
Yes
No
No internet service
OnlineBackup

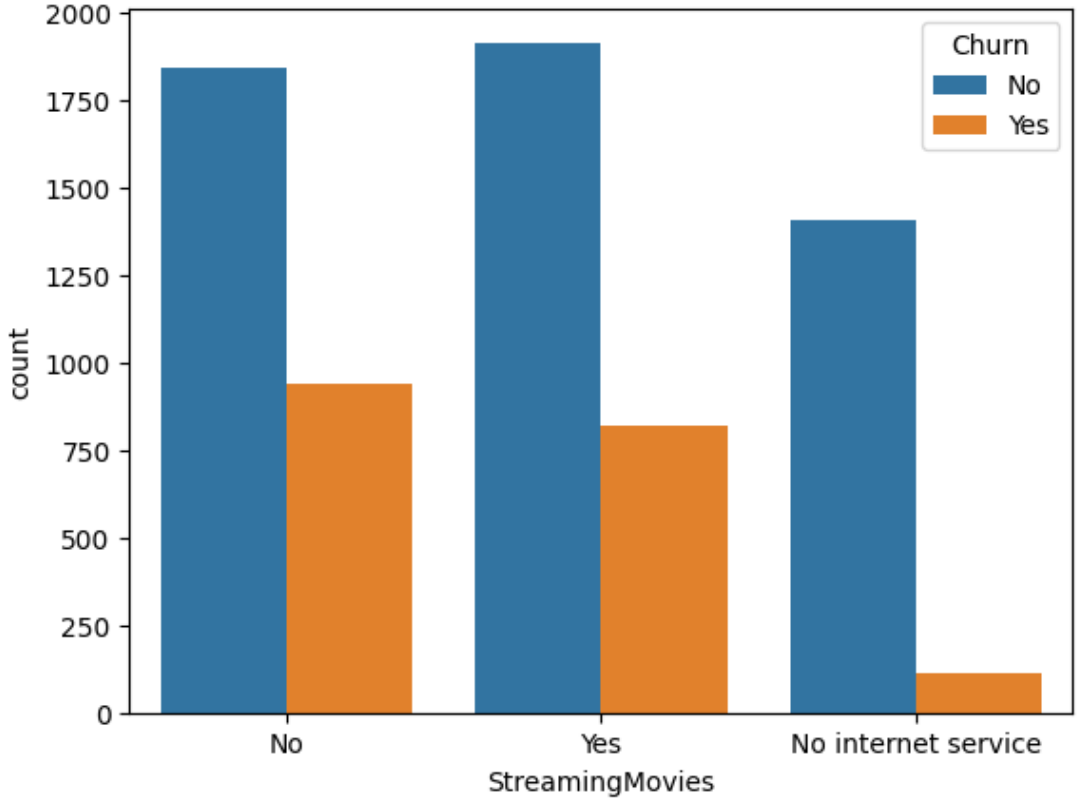

Churn
No
Yes
1750
1500
1250
1000


750
500
250
0
No
Yes
No internet service
DeviceProtection

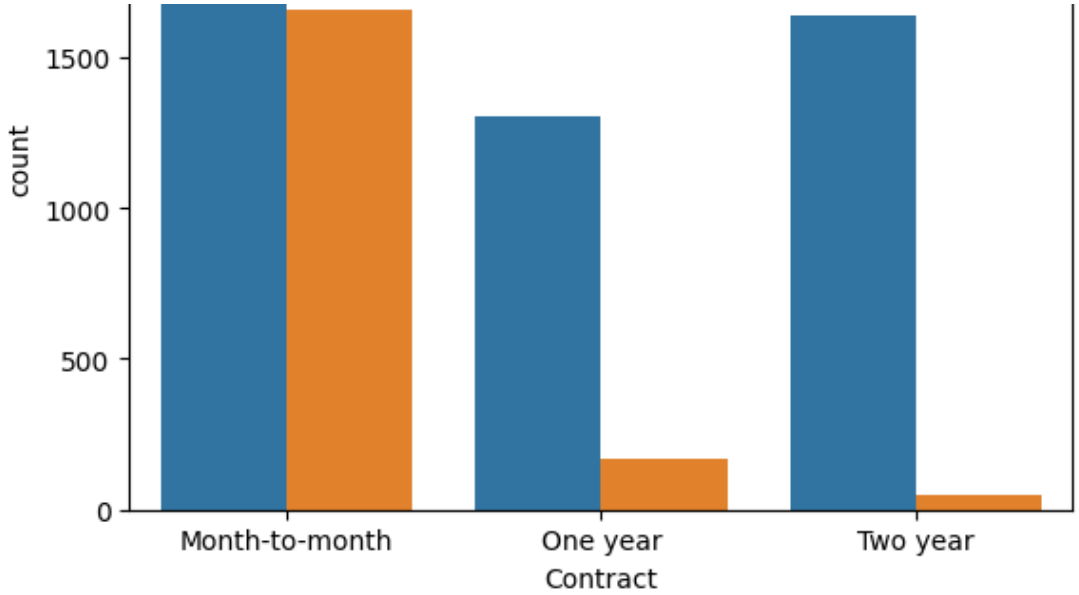

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
TechSupport

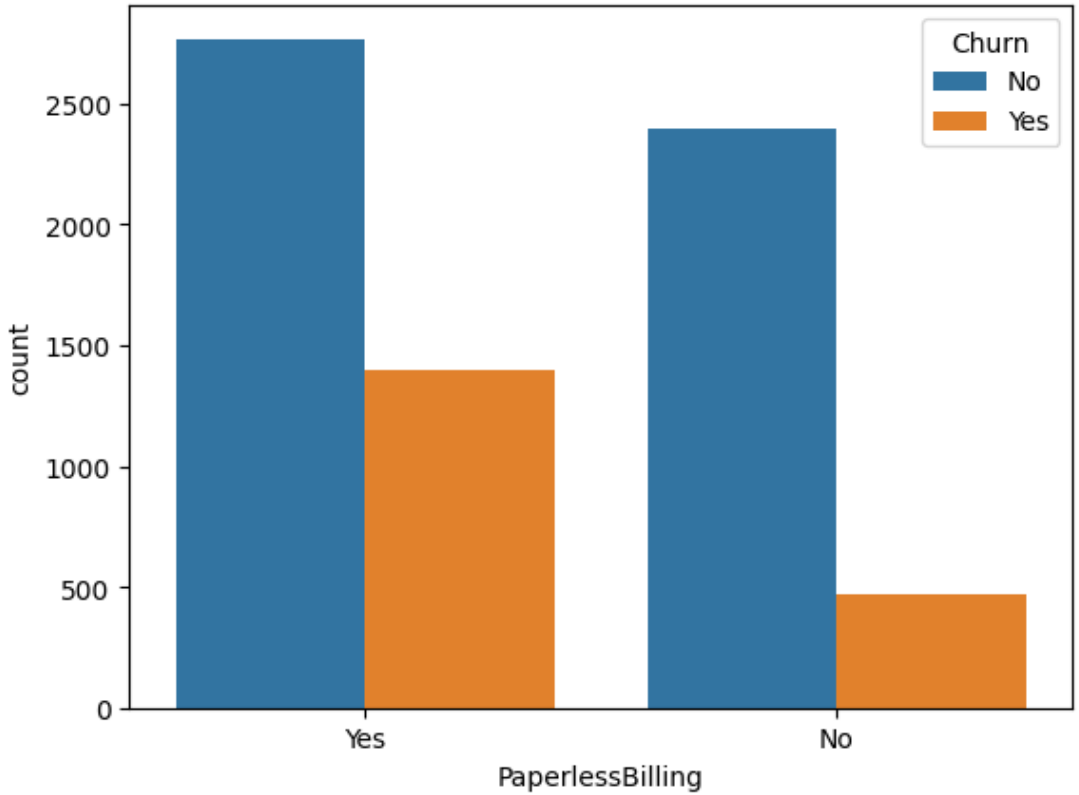

Churn
No
Yes
1750
1500
1250


count
1000
750
500
250
0
No
Yes
No internet service
StreamingTV

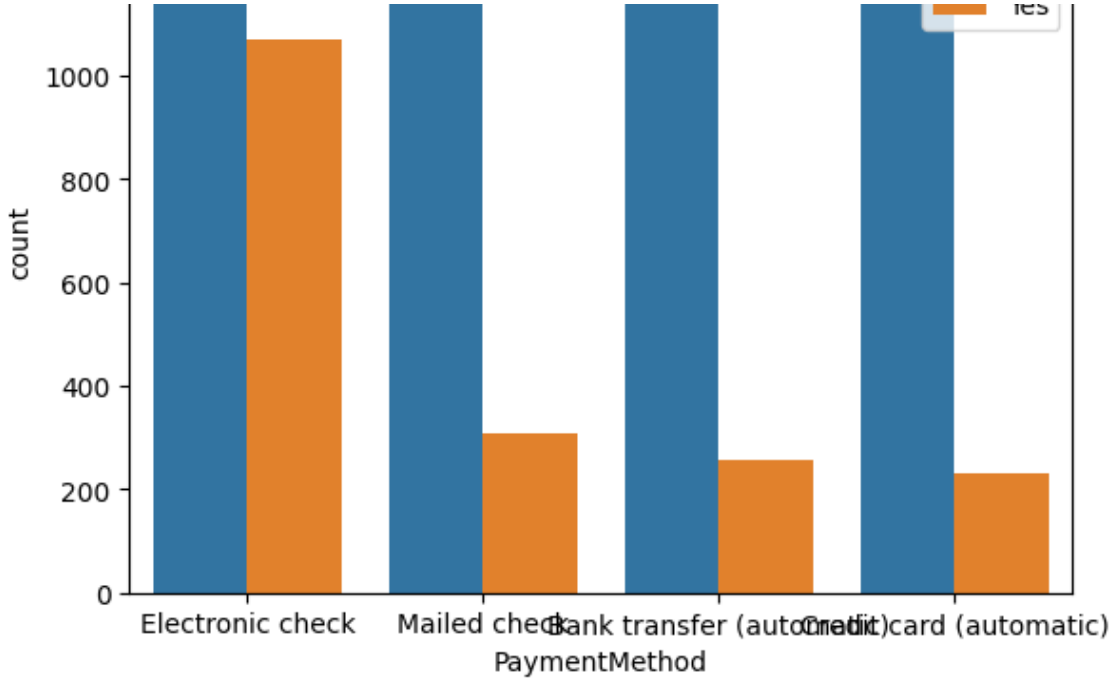

2000
1750
1500
1250
1000
750
500
250
0
count
Churn
No
Yes
No
Yes
No internet service
StreamingMovies

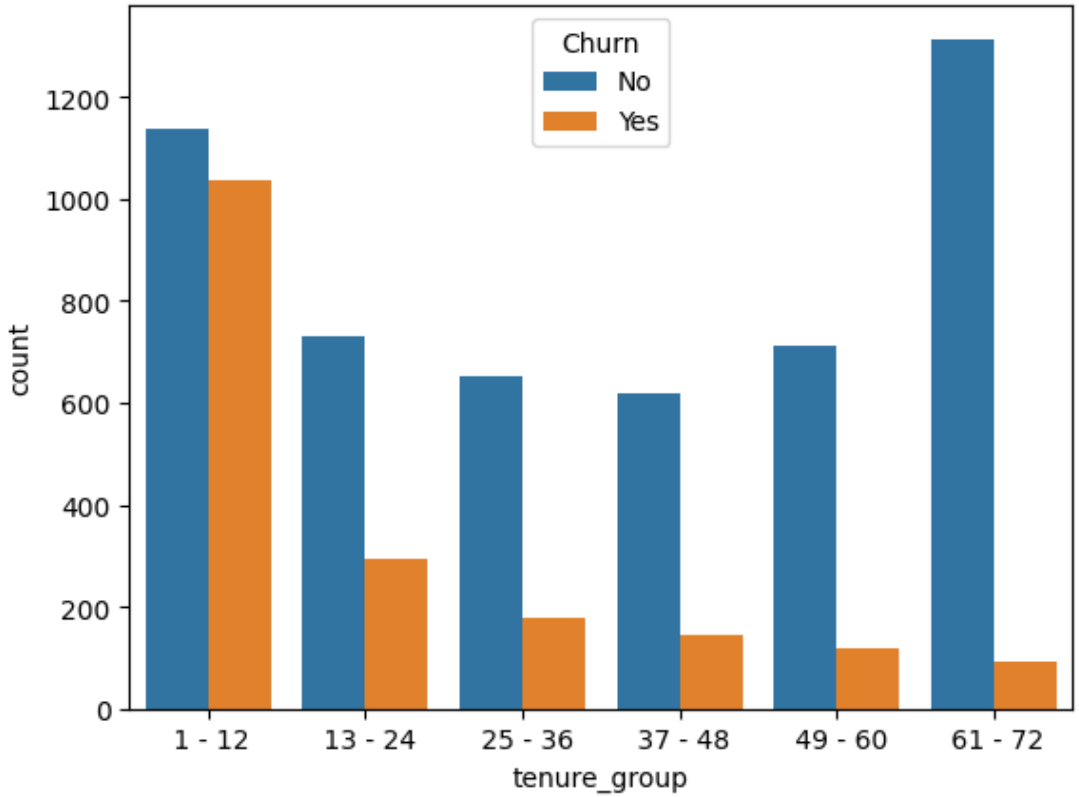

2000
Churn
No
Yes


count
1500
1000
500
0
Month-to-month
One year
Two year
Contract


Churn
No
Yes
count
2500
2000
1500
1000
500
0
Yes
No
PaperlessBilling


Churn
No
1200


count
0
200
400
600
800
1000
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod


Churn
No
Yes
count
0
200
400
600
800
1000
1200
1 - 12
13 - 24
25 - 36
37 - 48
49 - 60
61 - 72
tenure_group

Numerical Analysis

```
new_df.gender.value_counts()
```

```
Male      3549
Female    3483
Name: gender, dtype: int64
```

```
new_df_target1 = new_df[new_df["Churn"]=='No']
new_df_target2= new_df[new_df["Churn"]=='Yes']
```

```
new_df_target2.gender.value_counts()
```

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

```
pd.crosstab(new_df.PaymentMethod, new_df.Churn)
```

Churn	No	Yes
PaymentMethod		
Bank transfer (automatic)	1284	258
Credit card (automatic)	1289	232
Electronic check	1294	1071
Mailed check	1296	308

Convert target variable "churn" to numeric variable

```
new_df["Churn"] = np.where(new_df.Churn == 'Yes', 1, 0)
```

```
new_df.head()
```

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtecti
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	
1	Male	0	No	No	Yes	No	DSL	Yes	No	Y
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Y
4	Female	0	No	No	Yes	No	Fiber optic	No	No	

Next steps: Generate code with `new_df` View recommended plots

Convert categorical variables into dummy variables

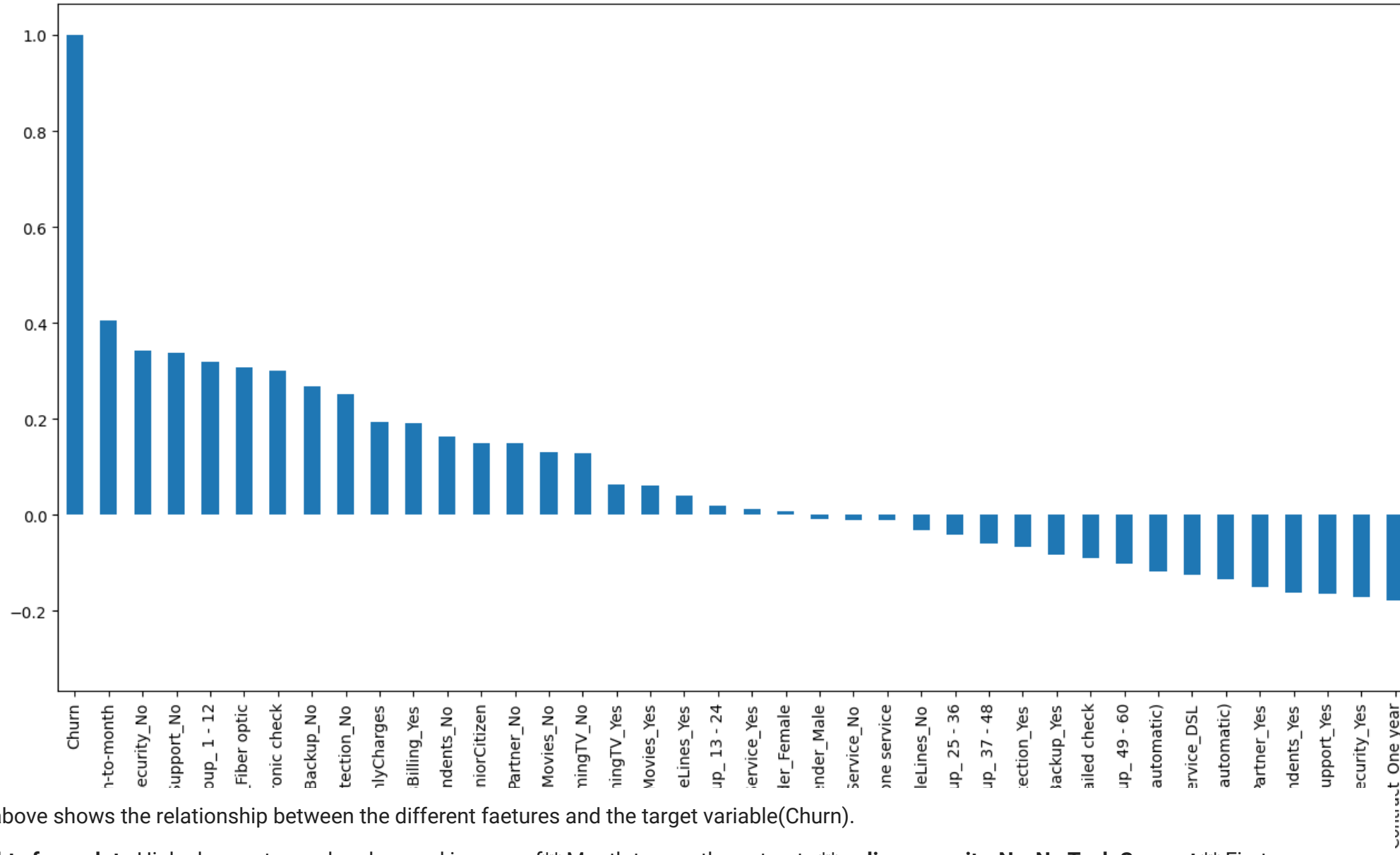
```
new_df_dummy = pd.get_dummies(new_df)
new_df_dummy.head()
```

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes
0	0	29.85	29.85	0	1	0	0	1	1	0
1	0	56.95	1889.50	0	0	1	1	0	1	0
2	0	53.85	108.15	1	0	1	1	0	1	0
3	0	42.30	1840.75	0	0	1	1	0	1	0
4	0	70.70	151.65	1	1	0	1	0	1	0

5 rows × 51 columns

```
plt.figure(figsize = (20, 8))
new_df_dummy.corr()['Churn'].sort_values(ascending = False).plot(kind = 'bar')
```

```
<Axes: >
```



The above shows the relationship between the different faetures and the target variable(Churn).

Insights from plots High churn rate can be observed in case of** Month to month contracts,** **online_security_No**, **No Tech Support**,** First year of subscription **and **Fibre Optics**

Low churn rate is possible in the cases of contracts of more than two years, no internet services and so on

Heat map representation *bold text*

```
plt.figure(figsize = (12,12))
sns.heatmap(new_df_dummy.corr(), cmap = "Paired")
```

```
<Axes: >
```