S1W1P2

Project Scenario, Roles, and Dataset

Customer Churn Analysis for Telecommunications Company

Student Name: Safi Ullah Nasir

Student ID: 316623

Table of Contents

Table of Figures

Data Preparation and Preprocessing

Load and preprocess the dataset

To load the dataset from the system, read the CSV file by giving the file path from the system. Load dataset as data frame by importing “pandas” library. Also import some useful libraries like numpy, matplotlib, and seaborn.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

```
# Read file from system:

df= pd.read_csv(r'C:\Users\hp\Desktop\Churn.csv')
df
```

	gender	SeniorCitizen	Dependents	tenure	PhoneService	MultipleLines	InternetService	Contract	MonthlyCharges	Churn
0	Female	0	No	1	No	No	DSL	Month-to-month	29.85	No
1	Male	0	No	34	Yes	No	DSL	One year	56.95	No
2	Male	0	No	2	Yes	No	DSL	Month-to-month	53.85	Yes
3	Male	0	No	45	No	No	DSL	One year	42.30	No
4	Female	0	No	2	Yes	No	Fiber optic	Month-to-month	70.70	Yes
...	...	...	...	...	...	...	...	...	...	...
7038	Male	0	Yes	24	Yes	Yes	DSL	One year	84.80	No
7039	Female	0	Yes	72	Yes	Yes	Fiber optic	One year	103.20	No
7040	Female	0	Yes	11	No	No	DSL	Month-to-month	29.60	No
7041	Male	1	No	4	Yes	Yes	Fiber optic	Month-to-month	74.40	Yes
7042	Male	0	No	66	Yes	No	Fiber optic	Two year	105.65	No

7043 rows × 10 columns

Figure 1: Data loading

Handle missing data points and encode categorical variables.

There are no missing data or other issues in the given dataset. These are checked by using the “info” and “is null()” functions.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   gender           7043 non-null   object 
 1   SeniorCitizen    7043 non-null   int64  
 2   Dependents       7043 non-null   object 
 3   tenure           7043 non-null   int64  
 4   PhoneService     7043 non-null   object 
 5   MultipleLines    7043 non-null   object 
 6   InternetService  7043 non-null   object 
 7   Contract         7043 non-null   object 
 8   MonthlyCharges   7043 non-null   float64
 9   Churn            7043 non-null   object 
dtypes: float64(1), int64(2), object(7)
memory usage: 550.4+ KB
```

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

```
gender            0
SeniorCitizen     0
Dependents        0
tenure            0
PhoneService      0
MultipleLines     0
InternetService   0
Contract          0
MonthlyCharges    0
Churn             0
dtype: int64
```

Figure 2: Checking data issues

To encode variables, using the “binary encoding method and label encoding methods”. The binary encoding method is used for “gender, Internet Service, phone services, multiple lines, churn, and dependents” due to having only two values but the “Contract” variable has three values so, used label encoding.

```
# Variable encoding:
# Use binary encoding for gender, Phone services, MultipleLines, InternetService and churn values due to have only two categorical values:

df['gender'] = df['gender'].map({'Male': 0, 'Female': 1})
df['InternetService'] = df['InternetService'].map({'DSL': 0, 'Fiber optic': 1})
df['PhoneService'] = df['PhoneService'].map({'No': 0, 'Yes': 1})
df['MultipleLines'] = df['MultipleLines'].map({'No': 0, 'Yes': 1})
df['Churn'] = df['Churn'].map({'No': 0, 'Yes': 1})
df['Dependents'] = df['Dependents'].map({'No': 0, 'Yes': 1})
df
```

	gender	SeniorCitizen	Dependents	tenure	PhoneService	MultipleLines	InternetService	Contract	MonthlyCharges	Churn
0	1	0	0	1	0	0	0	Month-to-month	29.85	0
1	0	0	0	34	1	0	0	One year	56.95	0
2	0	0	0	2	1	0	0	Month-to-month	53.85	1
3	0	0	0	45	0	0	0	One year	42.30	0
4	1	0	0	2	1	0	1	Month-to-month	70.70	1
...	...	...	...	...	...	...	...	...	...	...
7038	0	0	1	24	1	1	0	One year	84.80	0
7039	1	0	1	72	1	1	1	One year	103.20	0
7040	1	0	1	11	0	0	0	Month-to-month	29.60	0
7041	0	1	0	4	1	1	1	Month-to-month	74.40	1
7042	0	0	0	66	1	0	1	Two year	105.65	0

7043 rows × 10 columns

```
# varibale encoding:
# Lable encoding for Contract variable due to have more than two values:

df['Contract'] = df['Contract'].map({'Month-to-month': 0, 'One year': 1, 'Two year': 2})

df
```

	gender	SeniorCitizen	Dependents	tenure	PhoneService	MultipleLines	InternetService	Contract	MonthlyCharges	Churn
0	1	0	0	1	0	0	0	0	29.85	0
1	0	0	0	34	1	0	0	1	56.95	0
2	0	0	0	2	1	0	0	0	53.85	1
3	0	0	0	45	0	0	0	1	42.30	0
4	1	0	0	2	1	0	1	0	70.70	1
...	...	...	...	...	...	...	...	...	...	...
7038	0	0	1	24	1	1	0	1	84.80	0
7039	1	0	1	72	1	1	1	1	103.20	0
7040	1	0	1	11	0	0	0	0	29.60	0
7041	0	1	0	4	1	1	1	0	74.40	1
7042	0	0	0	66	1	0	1	2	105.65	0

7043 rows × 10 columns

Figure 3: Data encoding

Perform feature scaling and normalization.

The code standardizes the data by centering it around the mean and scaling it to unit variance. Standardsclaer() calculates the mean and standard deviation for each feature. fit_transform(pdf) then applies this scaling, transforming the data to have a mean of 0 and a standard deviation of 1

```
# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
```

Figure 4: Standardization

Ensure data integrity and consistency.

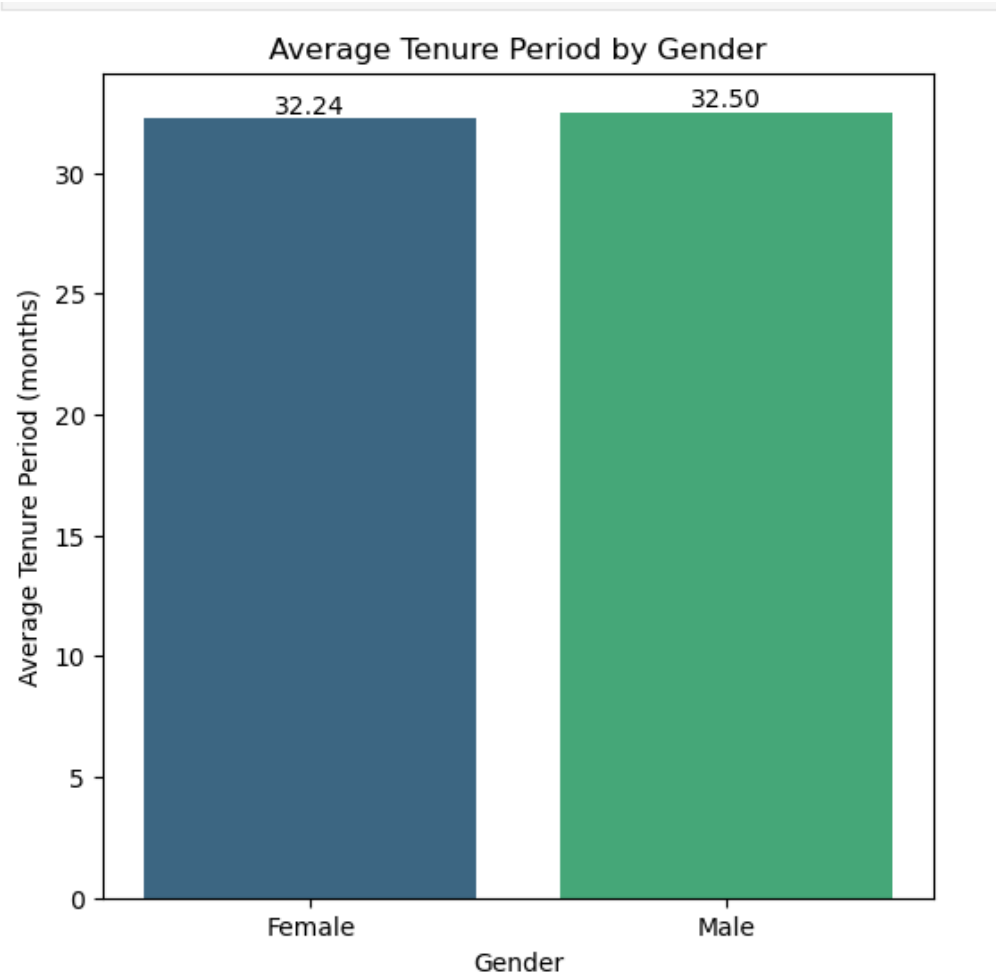
Ensuring data integrity and consistency involves validating data formats, checking for outliers or anomalies, and confirming that all necessary columns or features are present and correctly formatted. This step ensures the reliability and accuracy of the dataset for subsequent analysis or modeling tasks.

EDA

```
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

Figure 5: Descriptive statistics



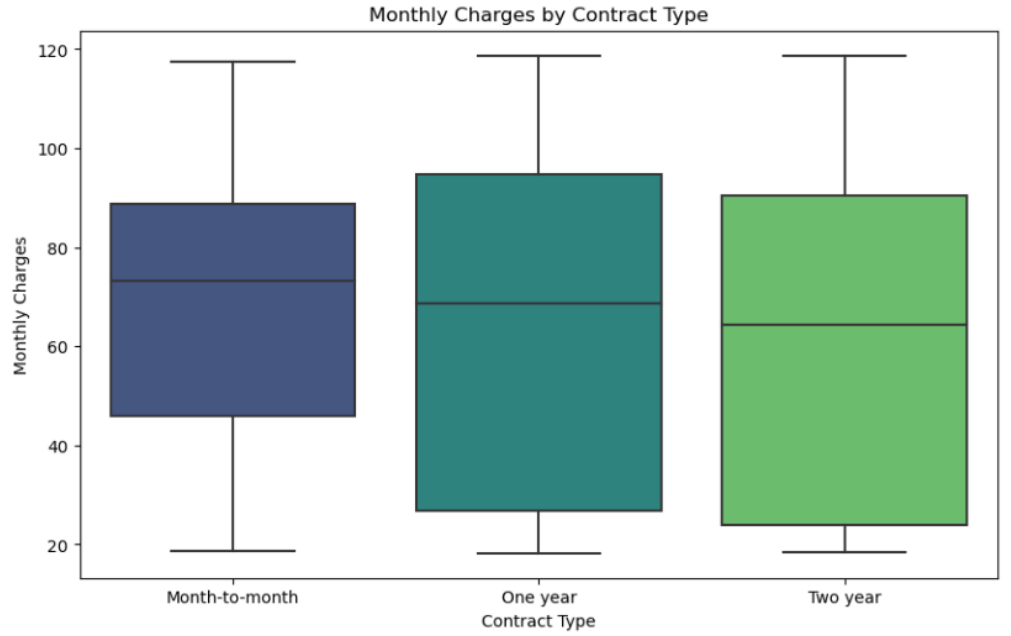


Figure 6:Tenure according to gender and Monthly charges by contract type

The above graphs are for “tenure according to gender” and another box plot graph for “relationship between monthly charges and contract”. Females served more tenure periods than males. “Month-to-month” charges are more costly than others.

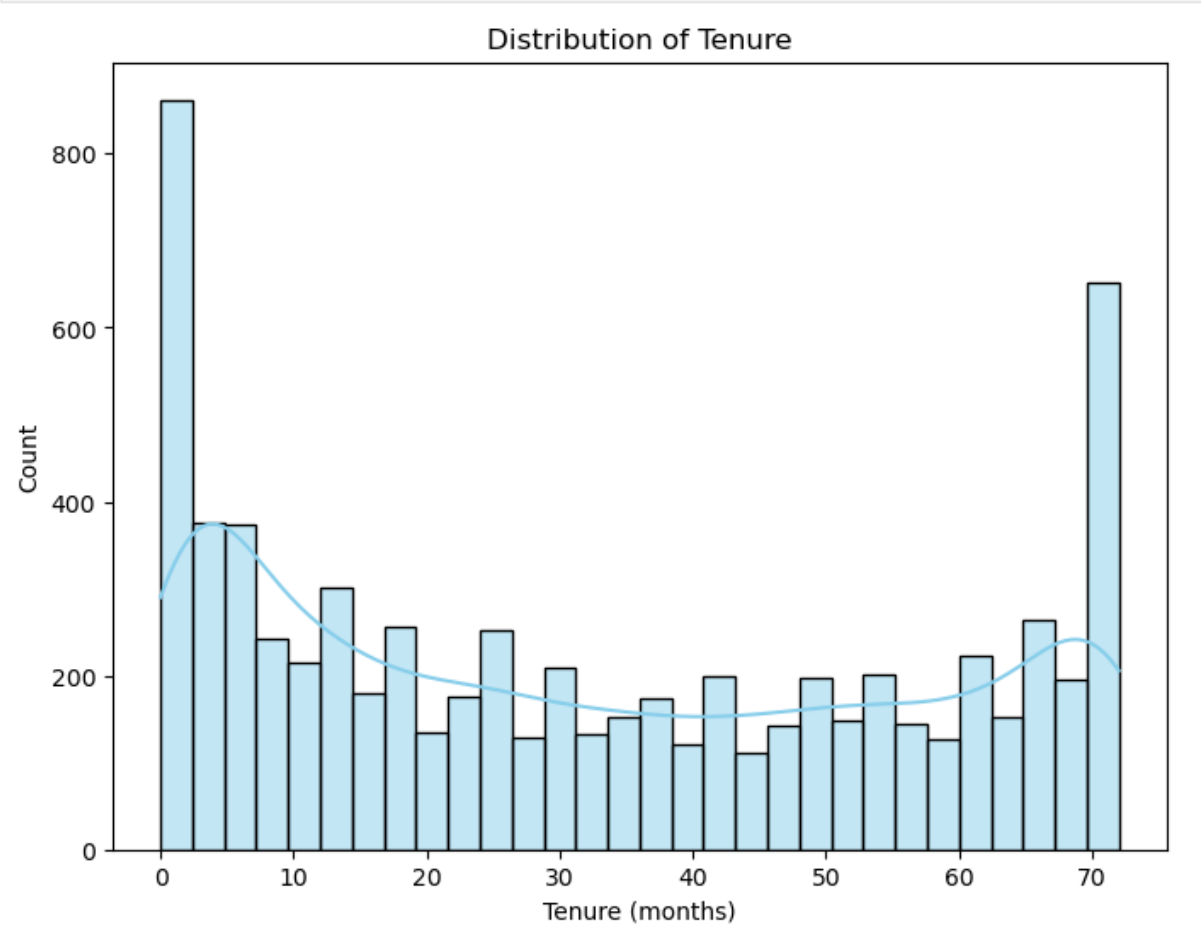


Figure 7: Histogram for Tenure

The above graphs show the “histogram for knowing the distribution of tenure”. According to this graph, it can be easily seen that 0 and 70 values have the highest count and others are normally distributed.

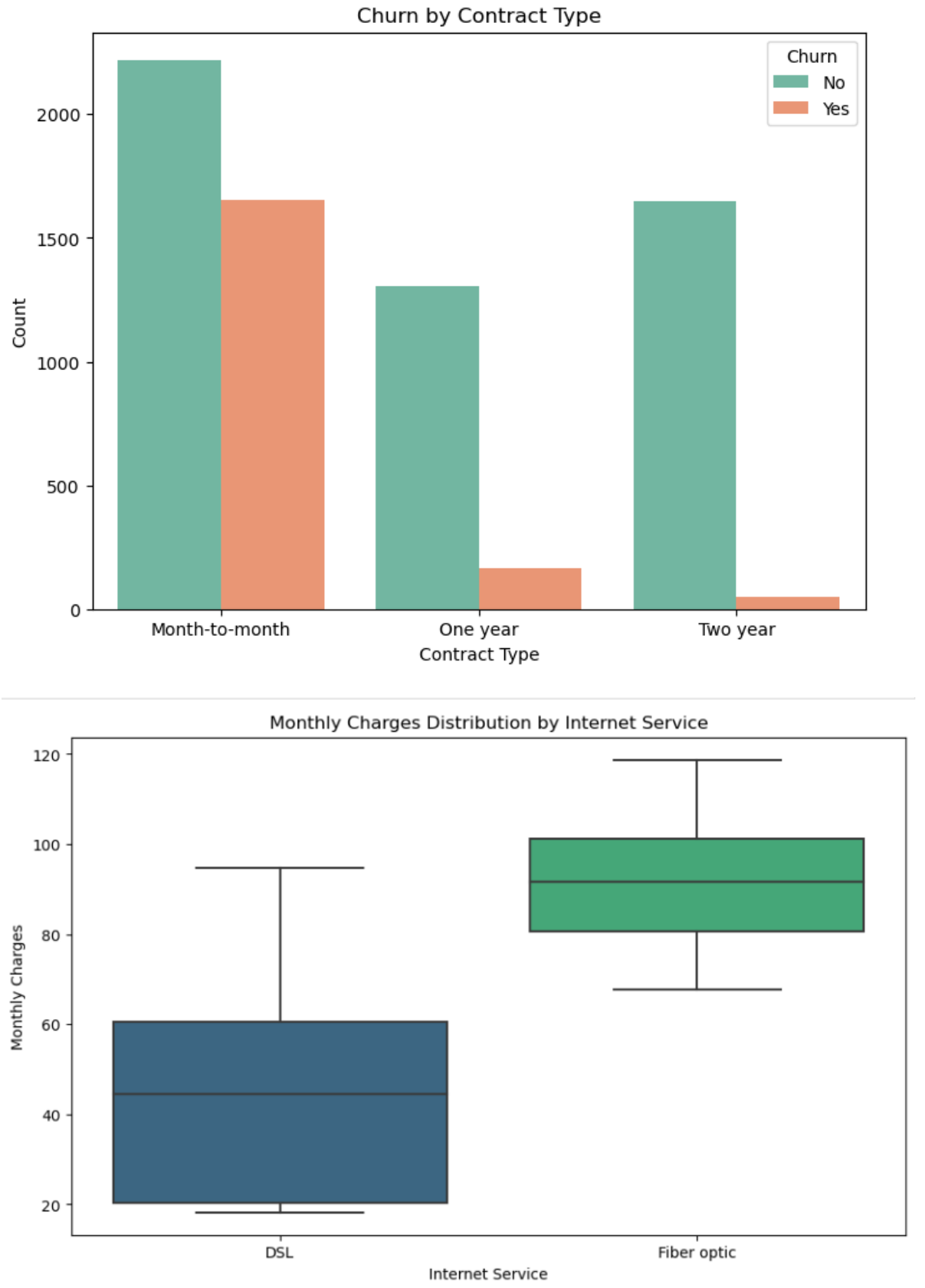


Figure 8: Churn by Contract and Monthly Charges for Internet Services

The above charts show the “churn by contract type” and “other shows the monthly charges distribution by internet service”. According to the churn graph, more customers are no churn in each contract type. The box plot shows that “Fiberoptic” has higher monthly charges than DSL.

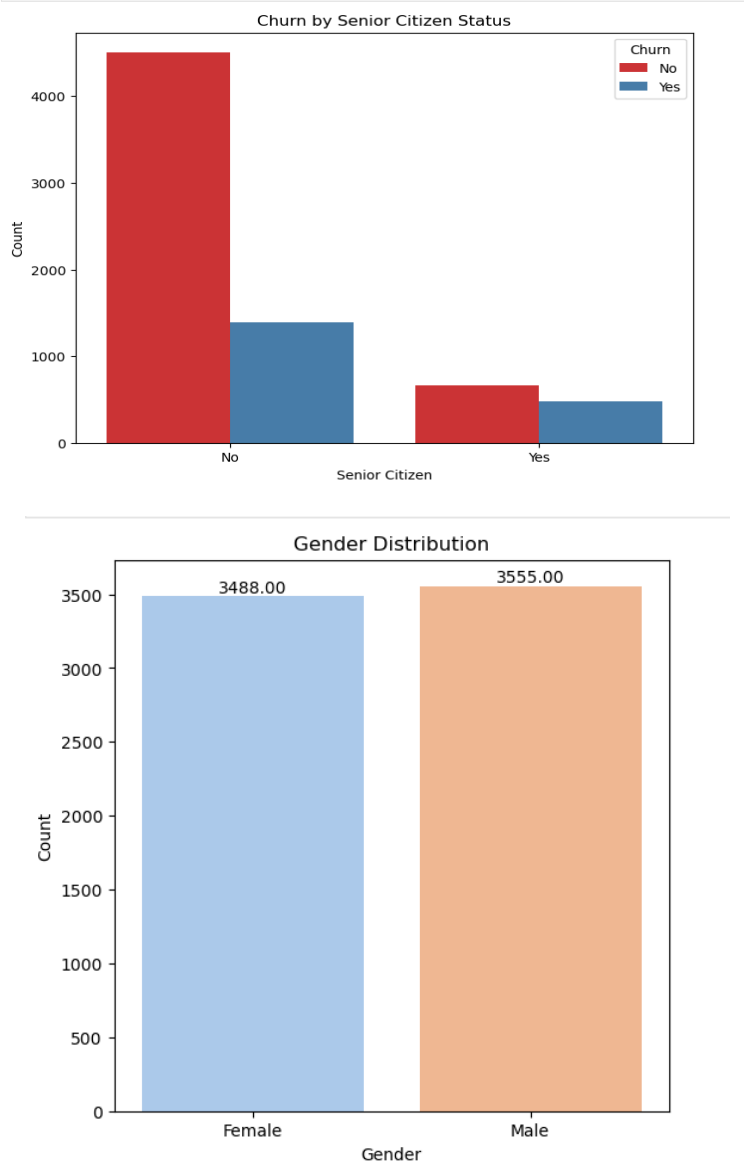


Figure 9: Churn by senior citizen and gender distribution

The above chart shows the “Churn behavior among senior citizens” and the other shows the “gender distribution”. According to this churn graph, it can be easily seen that most senior citizens are also not churnes as they like their telecom services. The gender distribution graph shows that “Male” customer is more connected to their telecom services (Mahadevan, 2022).

References

Mahadevan, M. (2022). Step-by-Step Exploratory Data Analysis (EDA) using Python. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/07/step-by-step-exploratory-data-analysis-eda-using-python/ [Accessed 5 Jul. 2024].