Performance Assessment for
D208: Predictive Modeling
Task 2 Attempt 2

Drew Mendez
MSDA Western Governors University
D208: Predictive Modeling
Dr. Taylor Jensen
July 8, 2024

D208_PA_MendezD_Task2_Attempt2

July 8, 2024

1 Part I: Research Question

1.1 A. Purpose of the Data Analysis

1.1.1 A1. Research Question

The data set selected for this performance assessment is the **churn** data set. The research question for this assessment is:

What variables contribute to whether or not a customer will churn?

1.1.2 A2. Goals of the Data Analysis

The goal of this data analysis is to determine influential explanatory variables to develop a logistic regression model that can be used by stakeholders to predict the likelihood of the target variable `Churn`, a binary categorical variable. This model could then be used by stakeholders to determine what services or customer attributes are associated with the probability that they will churn.

2 Part II: Method Justification

2.1 B. Multiple Logistic Regression Methods

2.1.1 B1. Four Assumptions of Logistic Regression

- independence of errors
- log odds (logit) of the dependent variable is a linear combination of the explanatory variables
- no multicollinearity
- no strongly influential outliers

(Stoltzfus, 2011)

2.1.2 B2. Benefits of using Python

Python was chosen for the functionality of the many packages available and for the ability to write functions that can be reused throughout the project. The libraries and packages that are essential to this analysis are:

- From **Pandas**, the `.isnull()`, `.duplicated()`, and `.sum()` methods provide some important basic functionality. Additionally, `.quantile()` is used in the detection of outliers and `.value_counts()` is used to count unique elements in data frames.
- From **Matplotlib**, `pyplot` is used to generate univariate graphs of variables to observe their distributions and outliers.

- From **Seaborn**, `boxplot()` and `countplot()` are used to generate the bivariate graphs.
- From **Statsmodels**, `.add_constant()` and `Logit()` are necessary to produce the logistic regression model.
- From **sklearn.metrics**, `confusion_matrix()` and `accuracy_score()` were necessary to respond to E2.
- From **scipy**, `stats.chi2.sf()` was necessary to calculate the test statistic for the Likelihood Ratio Test.

2.1.3 B3. Why Logistic Regression is Appropriate

The research question posed here has a binary categorical variable as the target variable. Since the goal of this analysis is to construct a model using multiple explanatory variables to predict the likelihood of a categorical response variable, a logistic regression model is appropriate.

3 Part III: Data Preparation

3.1 C. Summary of the Data Preparation Process

3.1.1 C1. Data Cleaning Goals

Before constructing the model, first the data set will be cleaned and treated. The plan to clean the data set involves detecting and treating duplicates, missing values, and outliers, and the re-expression of categorical variables. The steps and techniques necessary to perform these tasks is given:

- Duplicates:
  - Duplicates are detected below by chaining the `.duplicated()` and `.sum()` methods from the `Pandas` library and calling them on the data frame, returning the total count of duplicate observations. It is shown below that there are no duplicate rows.
- Missing Values:
  - Missing values are detected below by chaining the `.isnull()` and `.sum()` methods from the `Pandas` library and calling them on the data frame, returning the total count of missing values for each variable. Since the `InternetService` variable has `None` as one of its options, the 2129 erroneously identified null values are imputed with `None` to avoid being interpreted as nulls. It is shown below that there are no other missing values.
- Outliers:
  - The outliers of the following thirteen **quantitative variables** are identified using the `boxplot()` function from the `matplotlib` library and counted below using a user-defined function. It is shown below that there are no unacceptable/unreasonable outliers, so all outliers shown here will be retained.
    * Income: Annual income of customer (**continuous numeric data**)
    * Lat: GPS coordinates of the latitude of the customer residence (**continuous numeric data**)
    * Lng: GPS coordinates of the longitude of the customer residence (**continuous numeric data**)
    * Population: Population within a mile radius of customer (**discrete numeric data**)
    * Children: Number of children in customer's household (**discrete numeric data**)
    * Age: Age of customer (**continuous numeric data**)

    * Outage_sec_perweek: Average number of seconds per week of system outages in the customer's neighborhood (**continuous numeric data**)
    * Email: Number of emails sent to the customer in the last year (marketing or correspondence) (**discrete numeric data**)
    * Contacts: Number of times customer contacted technical support (**discrete numeric data**)
    * Yearly_equip_failure: The number of times customer's equipment failed and had to be reset/replaced in the past year (**discrete numeric data**)
    * Tenure: Number of months the customer has stayed with the provider (**continuous numeric data**)
    * MonthlyCharge: The amount charged, on average, per customer monthly (**continuous numeric data**)
    * Bandwidth_GB_Year: The average amount of data used, in GB, in a year by the customer (**continuous numeric data**)
- Re-expression of Categorical Variables:
  - The following thirteen **binary nominal categorical variables** are re-expressed below using binary encoding:
    * `Churn`: Whether the customer discontinued service within the last month (yes, no)
    * `Techie`: Whether the customer considers themselves technically inclined (yes, no)
    * `Port_modem`: Whether the customer has a portable modem (yes, no)
    * `Tablet`: Whether the customer owns a tablet such as iPad, Surface, etc. (yes, no)
    * `Phone`: Whether the customer has a phone service (yes, no)
    * `Multiple`: Whether the customer has multiple lines (yes, no)
    * `OnlineSecurity`: Whether the customer has an online security add-on (yes, no)
    * `OnlineBackup`: Whether the customer has an online backup add-on (yes, no)
    * `DeviceProtection`: Whether the customer has device protection add-on (yes, no)
    * `TechSupport`: Whether the customer has a technical support add-on (yes, no)
    * `StreamingTV`: Whether the customer has streaming TV (yes, no)
    * `StreamingMovies`: Whether the customer has streaming movies (yes, no)
    * `PaperlessBilling`: Whether the customer has paperless billing (yes, no)
  - The following six **nominal categorical variables** are re-expressed below using one-hot encoding:
    * `Area`: Area type (rural, urban, suburban)
    * `Marital`: Marital status of customer
    * `Gender`: Customer self-identification as male, female, or nonbinary
    * `Contract`: The contract term of the customer (month-to-month, one year, two year)
    * `InternetService`: Customer's internet service provider (DSL, fiber optic, None)
    * `PaymentMethod`: The customer's payment method (electronic check, mailed check, bank (automatic bank transfer), credit card (automatic))
  - If variables have too many unique values, re-expressing them will increase the dimensionality of the model (Middleton, 2022). For this reason, the following categorical variables will be omitted from the model:
    * `City`: Customer city of residence
    * `State`: Customer state of residence
    * `County`: Customer county of residence
    * `Zip`: Customer zip code of residence
    * `TimeZone`: Time zone of customer residence based on customer sign-up information
    * `Job`: Job of the customer/invoiced person

[1]:
```python
## C1 The following cells include the annotated code used to clean the data.
# See code attached, in D208_PA_MendezD_Task2_Attempt2.ipynb



# Import the Pandas library, then load the data into a data frame with Pandas' .
↪read_csv() function
import pandas as pd
df = pd.read_csv('/Users/drewmendez/Documents/WGU/D208/churn_d208/churn_clean.
↪csv')

def printDupesNulls(data_frame):
# Detect duplicates with Pandas' .duplicated method chained with .sum() method.
# Identify missing values in the data frame with Pandas' .isnull() method,
# then sum the resulting series with the .sum() method

    duplicate_count = data_frame.duplicated().sum()
    missing_values_count = data_frame.isnull().sum()
    print('Number of duplicate rows:', duplicate_count)
    print("Number of missing values per variable:")
    print(missing_values_count)

printDupesNulls(df)
```

```
Number of duplicate rows: 0
Number of missing values per variable:
CaseOrder                  0
Customer_id                0
Interaction                0
UID                        0
City                       0
State                      0
County                     0
Zip                        0
Lat                        0
Lng                        0
Population                 0
Area                       0
TimeZone                   0
Job                        0
Children                   0
Age                        0
Income                     0
Marital                    0
Gender                     0
Churn                      0
Outage_sec_perweek         0
Email                      0
Contacts                   0
```

```
Yearly_equip_failure       0
Techie                     0
Contract                   0
Port_modem                 0
Tablet                     0
InternetService         2129
Phone                      0
Multiple                   0
OnlineSecurity             0
OnlineBackup               0
DeviceProtection           0
TechSupport                0
StreamingTV                0
StreamingMovies            0
PaperlessBilling           0
PaymentMethod              0
Tenure                     0
MonthlyCharge              0
Bandwidth_GB_Year          0
Item1                      0
Item2                      0
Item3                      0
Item4                      0
Item5                      0
Item6                      0
Item7                      0
Item8                      0
dtype: int64
```

[2]:
```
## C1 Treatment of NAs

# Since the 'InternetService' variable has 'None' as one of its options,
# it is necessary to impute 'None'

df['InternetService'].fillna('None', inplace=True)

# Verify that 'None' no longer appears as 'Null'
print('Number of `InternetService` nulls:', df['Tenure'].isnull().sum())
```

```
Number of `InternetService` nulls: 0
```

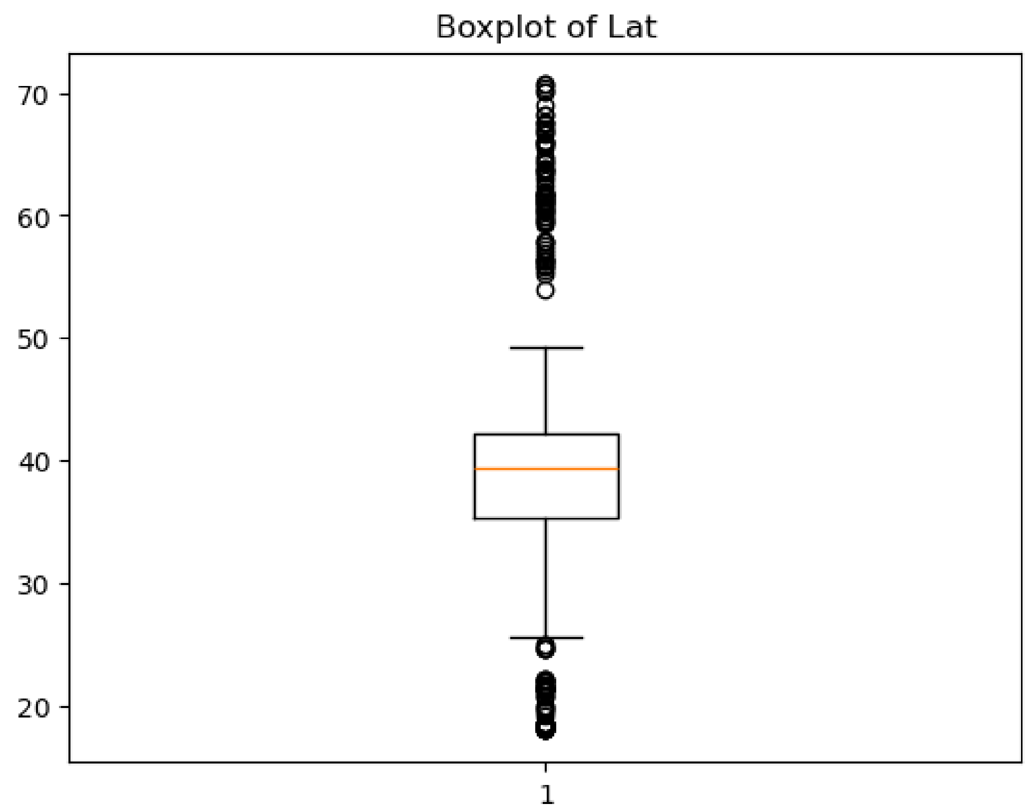
[3]:
```
## C1 Detect and Count Outliers of Numeric Variables

import matplotlib.pyplot as plt

def boxplotOutliers(data_frame, col_name):
# Visualize outliers using boxplot() from matplotlib
# First and third quartiles, Q1 and Q3, are found using .quantile() from Pandas,
```

```python
# then the interquartile range is found using IQR = Q3 - Q1.
# The upper whisker of the boxplot is found using max = Q3 + 1.5 * IQR.
# The lower whisker of the boxplot is found using min = Q1 - 1.5 * IQR.
# The .sum() method returns the count of observations greater than the max or␣
↪less than the min.
# The .round() method rounds the outlier count to two decimals.
# If loop to print corresponding outputs

    plt.boxplot(data_frame[col_name])
    plt.title(f'Boxplot of {col_name}')
    plt.show()

    Q1 = data_frame[col_name].quantile(0.25)
    Q3 = data_frame[col_name].quantile(0.75)
    IQR = Q3 - Q1
    maximum = round(Q3 + 1.5 * IQR, 2)
    minimum = round(Q1 - 1.5 * IQR, 2)
    outlier_count_up = (data_frame[col_name] > maximum).sum()
    outlier_count_low = (data_frame[col_name] < minimum).sum()


    if outlier_count_up > 0:
        if outlier_count_low > 0:
            print(f'For the `{col_name}` variable, all observations greater␣
↪than {maximum} or less than {minimum} are considered outliers.')
            print(f'The count of observations greater than {maximum} is␣
↪{outlier_count_up}.')
            print(f'The count of observations less than {minimum} is␣
↪{outlier_count_low}.')
        if outlier_count_low == 0:
            print(f'For the `{col_name}` variable, all observations greater␣
↪than {maximum} are considered outliers.')
            print(f'The count of observations greater than {maximum} is␣
↪{outlier_count_up}.')
    if outlier_count_up == 0:
        if outlier_count_low > 0:
            print(f'For the `{col_name}` variable, all observations less than␣
↪{minimum} are considered outliers.')
            print(f'The count of observations less than {minimum} is␣
↪{outlier_count_low}.')
        if outlier_count_low == 0:
            print(f'There are no outliers for the `{col_name}` variable.')
```

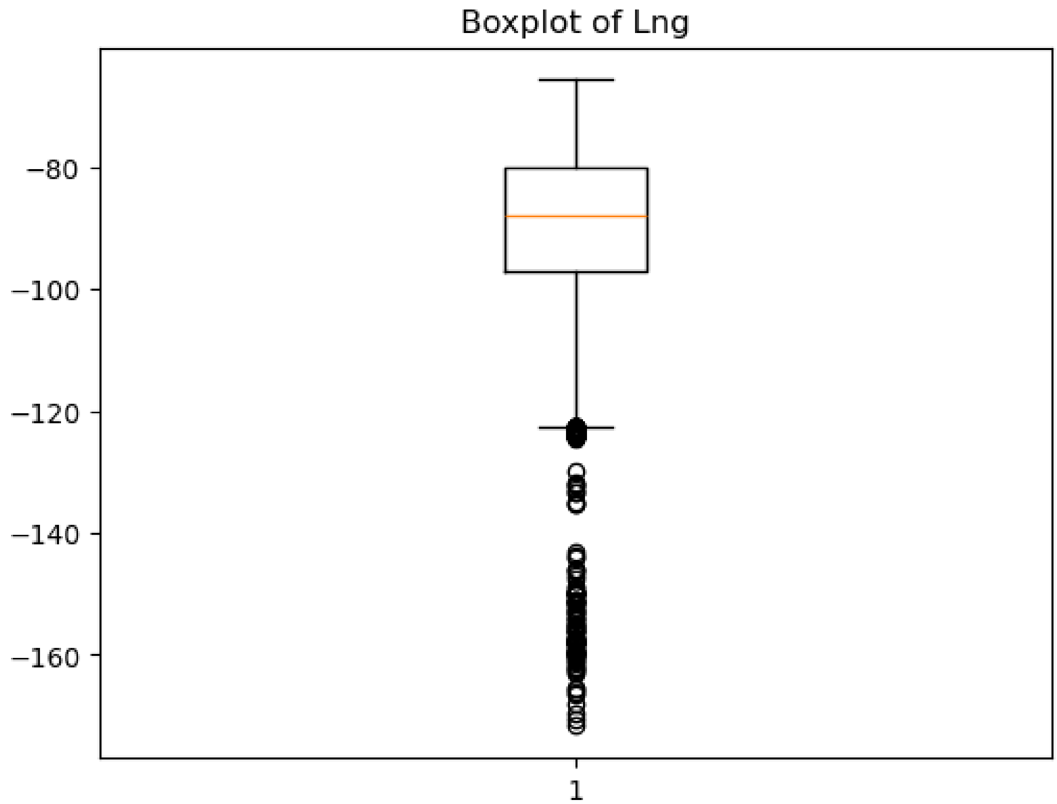
[4]:

```python
# C1 Detection of Outliers for 13 Numeric Variables
```

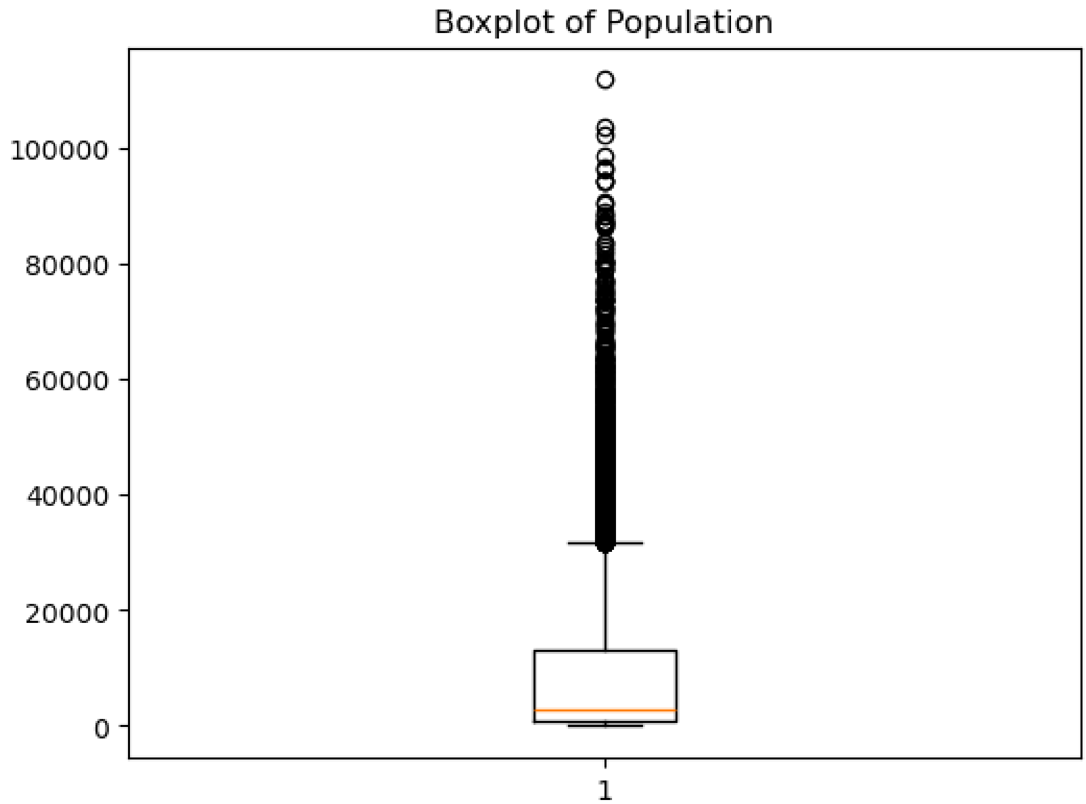
```python
numericVars = df[['Lat', 'Lng', 'Population', 'Children', 'Age', 'Income',
    'Outage_sec_perweek', 'Email',
            'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
    'Bandwidth_GB_Year']]

for col in numericVars:
    boxplotOutliers(df, col)
```



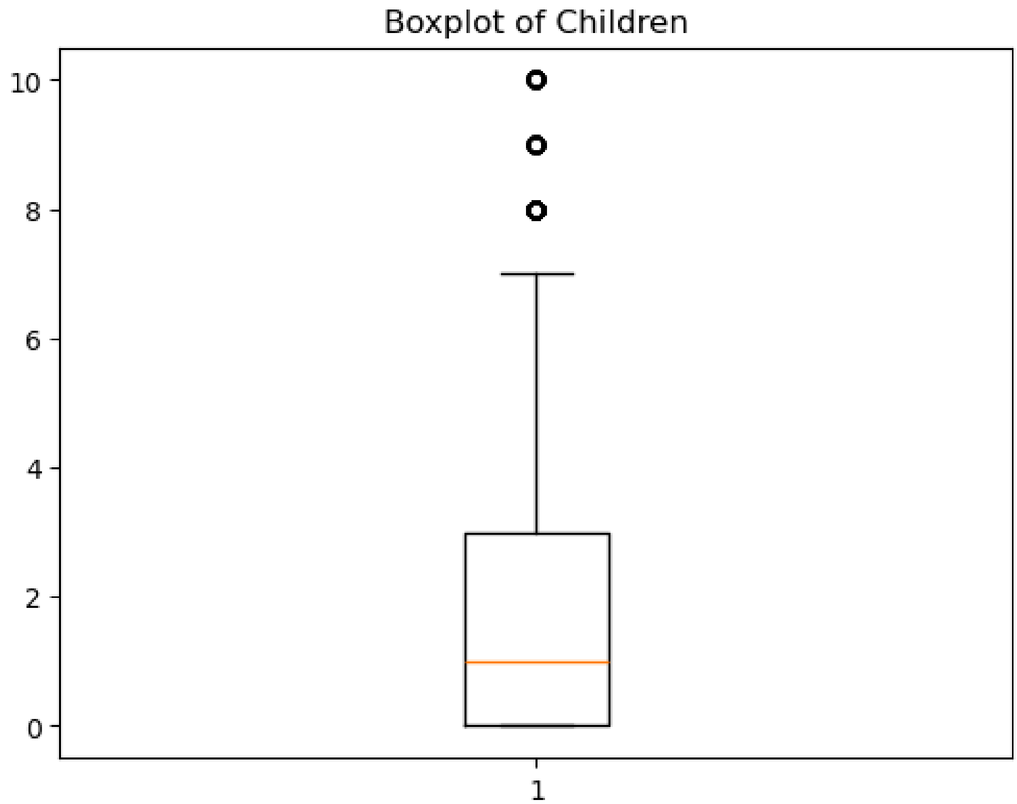
For the `Lat` variable, all observations greater than 52.25 or less than 25.19
are considered outliers.
The count of observations greater than 52.25 is 77.
The count of observations less than 25.19 is 81.



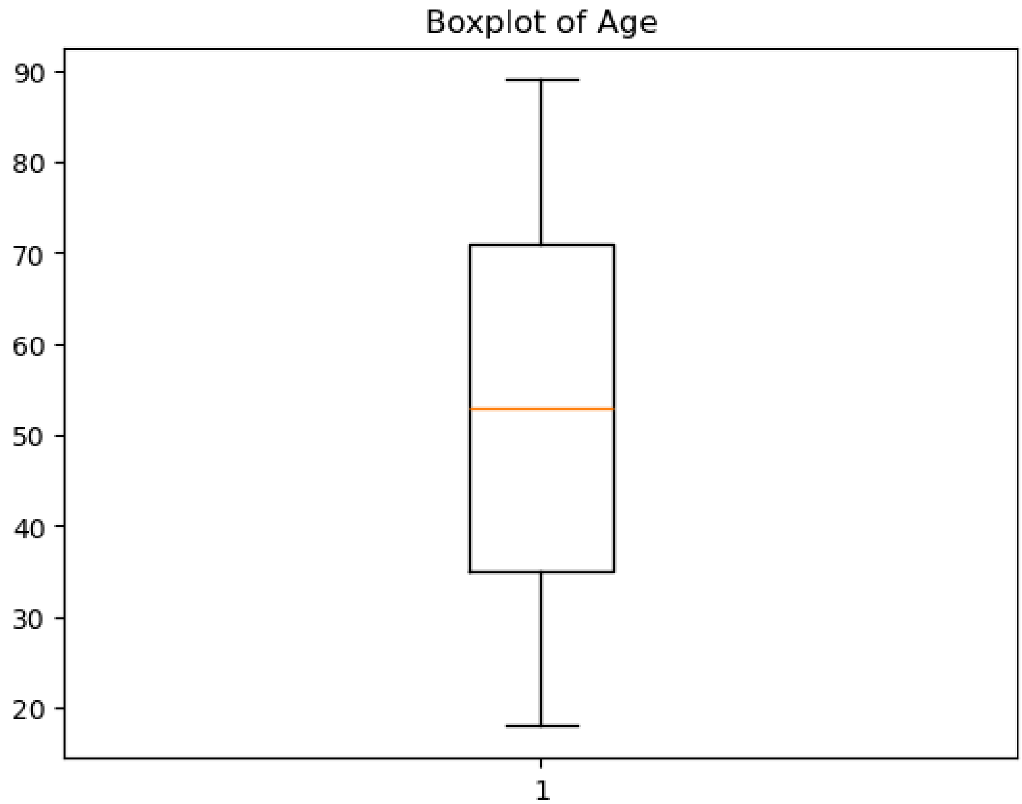

```
For the `Lng` variable, all observations less than -122.57 are considered
outliers.
The count of observations less than -122.57 is 273.
```



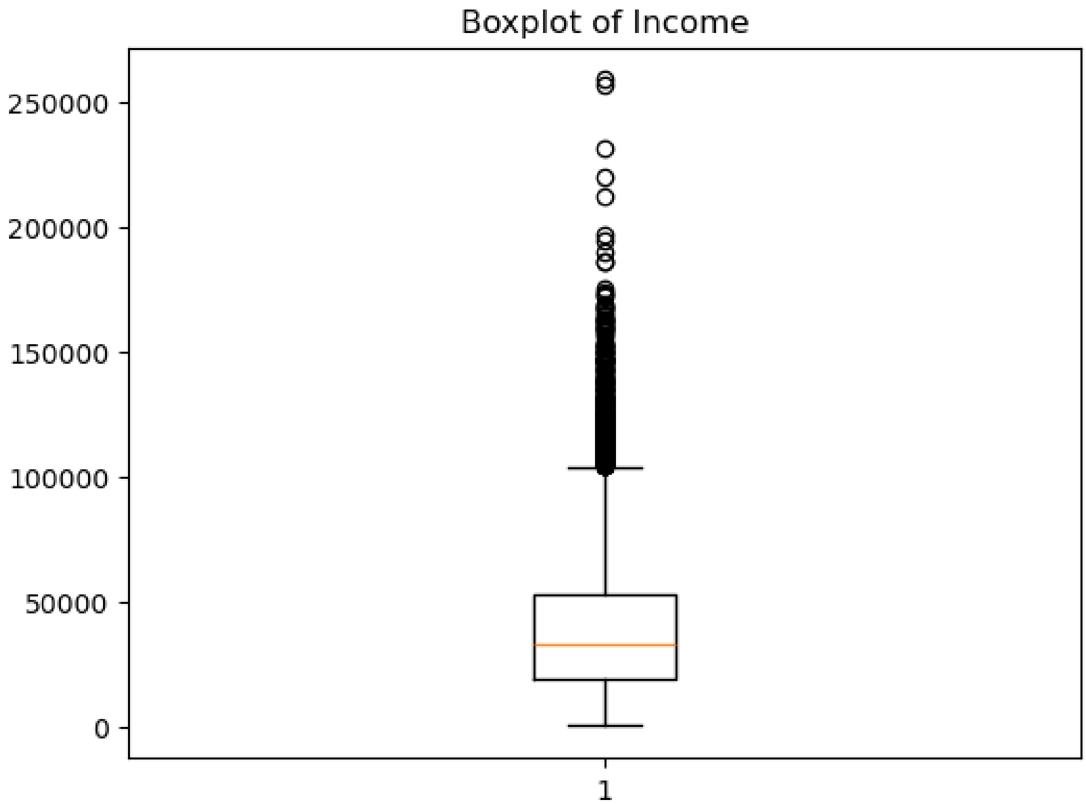
For the `Population` variable, all observations greater than 31813.0 are considered outliers.
The count of observations greater than 31813.0 is 937.



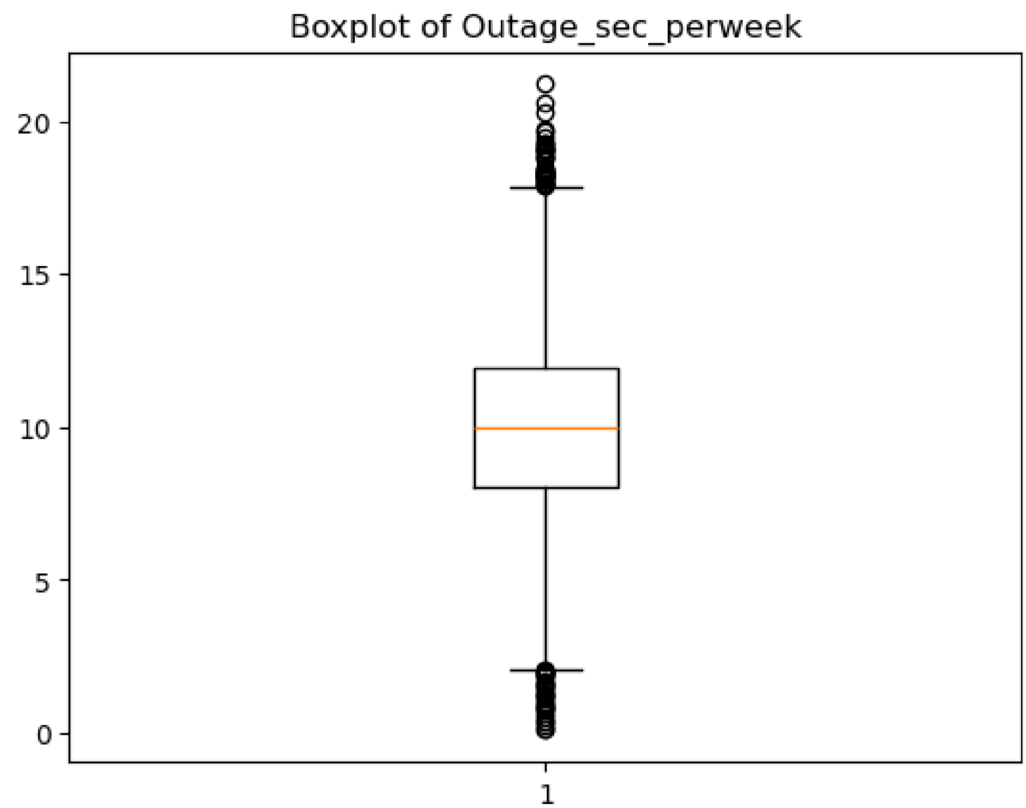

```
For the `Children` variable, all observations greater than 7.5 are considered
outliers.
The count of observations greater than 7.5 is 401.
```



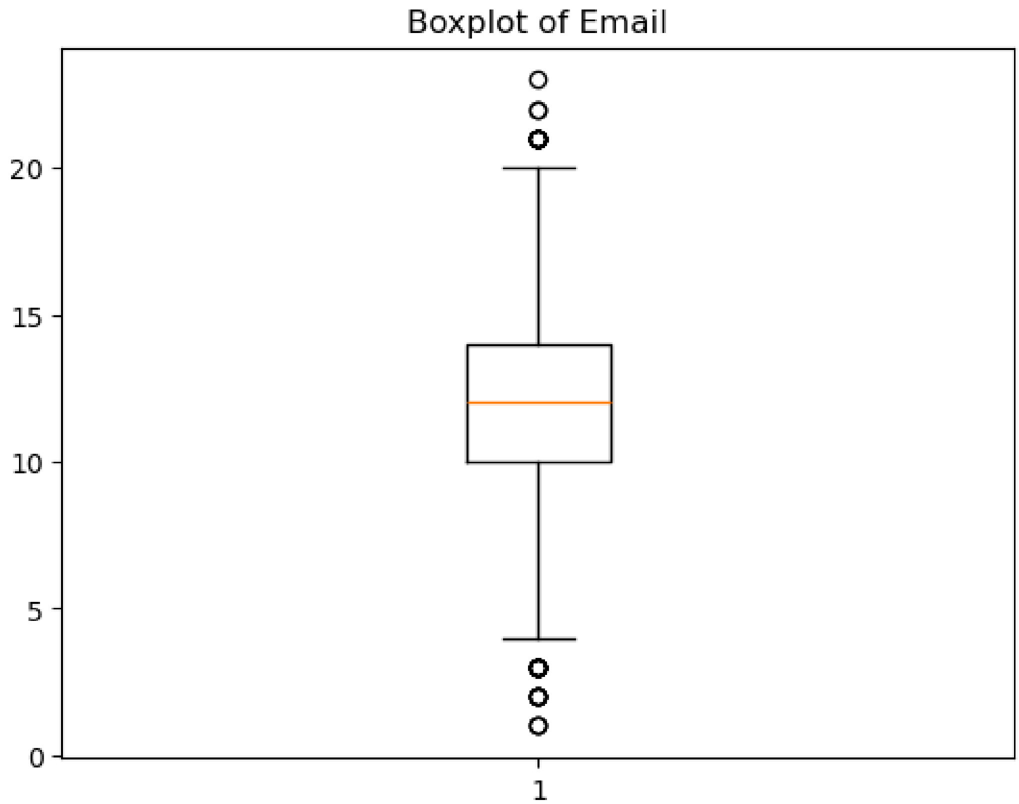
There are no outliers for the `Age` variable.



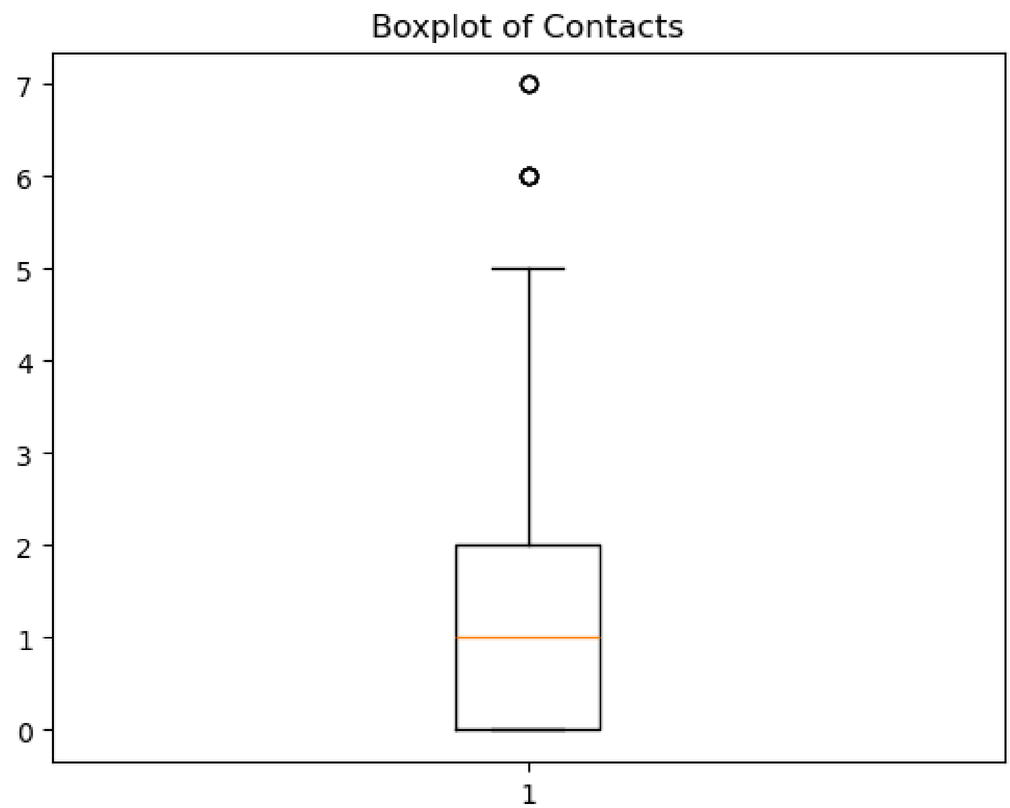

```
For the `Income` variable, all observations greater than 104278.35 are
considered outliers.
The count of observations greater than 104278.35 is 336.
```



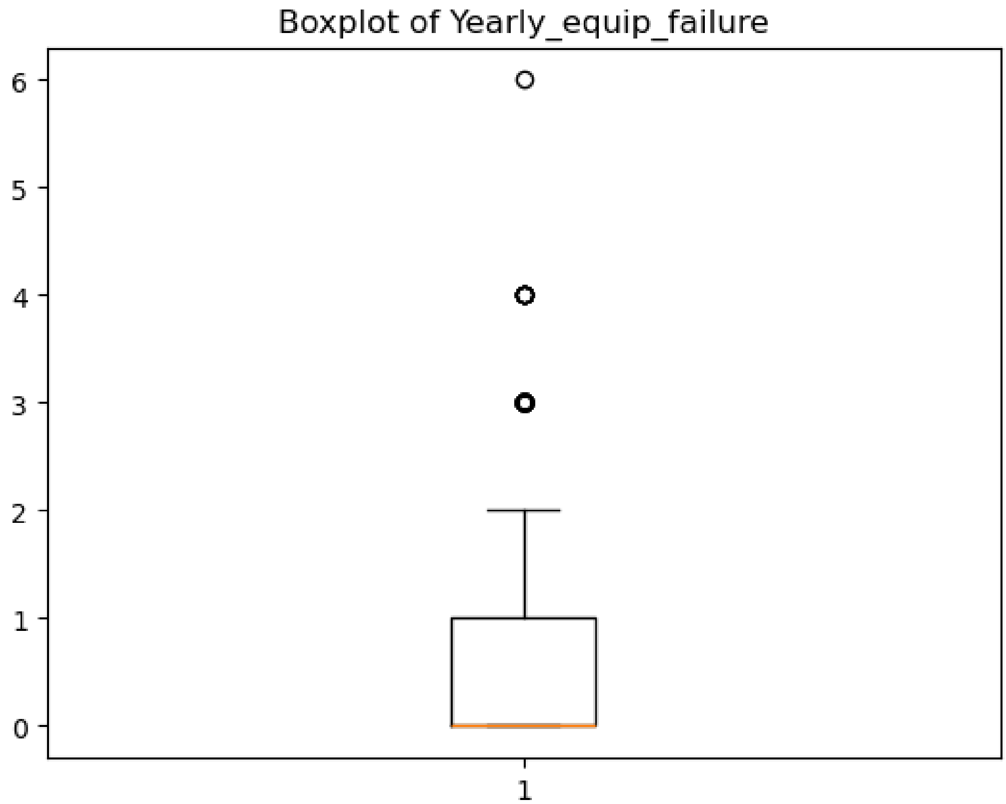
For the `Outage_sec_perweek` variable, all observations greater than 17.9 or less than 2.09 are considered outliers.
The count of observations greater than 17.9 is 43.
The count of observations less than 2.09 is 33.



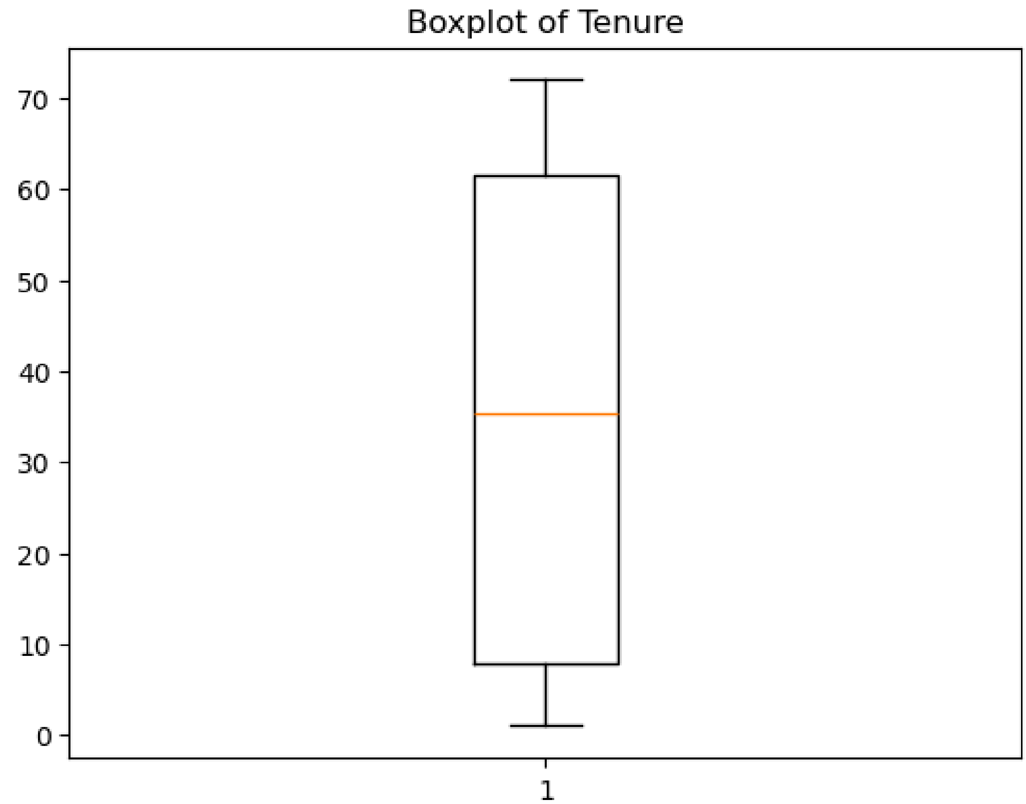

```
For the `Email` variable, all observations greater than 20.0 or less than 4.0
are considered outliers.
The count of observations greater than 20.0 is 15.
The count of observations less than 4.0 is 23.
```



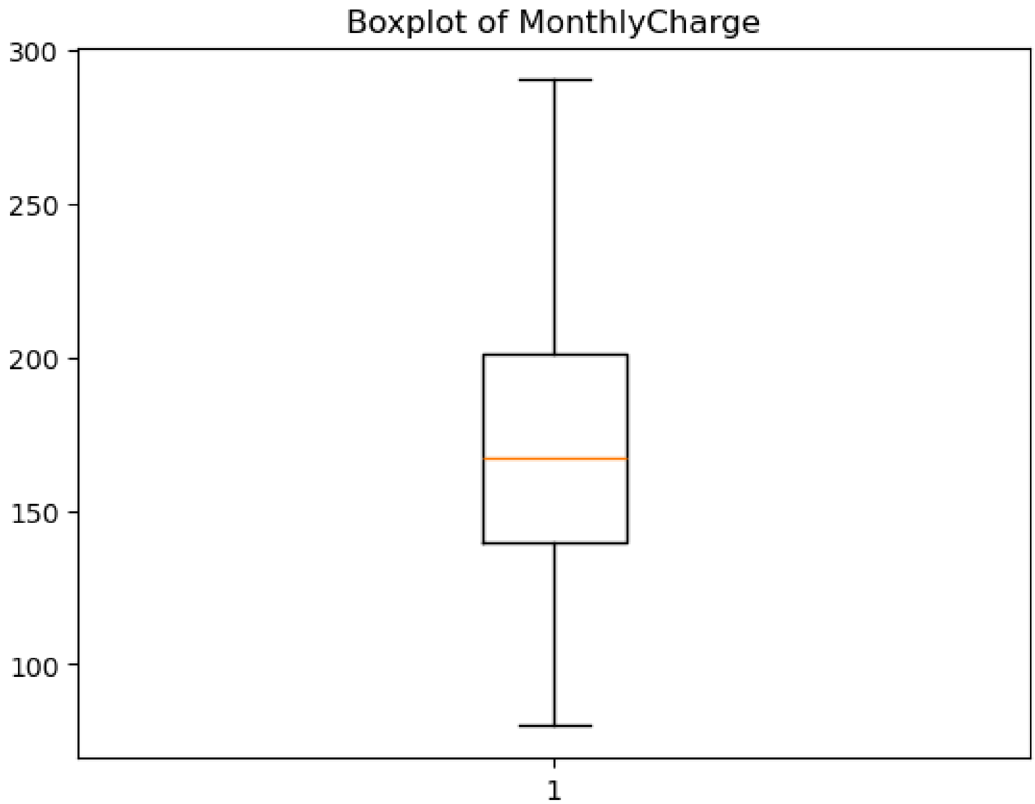
For the `Contacts` variable, all observations greater than 5.0 are considered outliers.
The count of observations greater than 5.0 is 8.



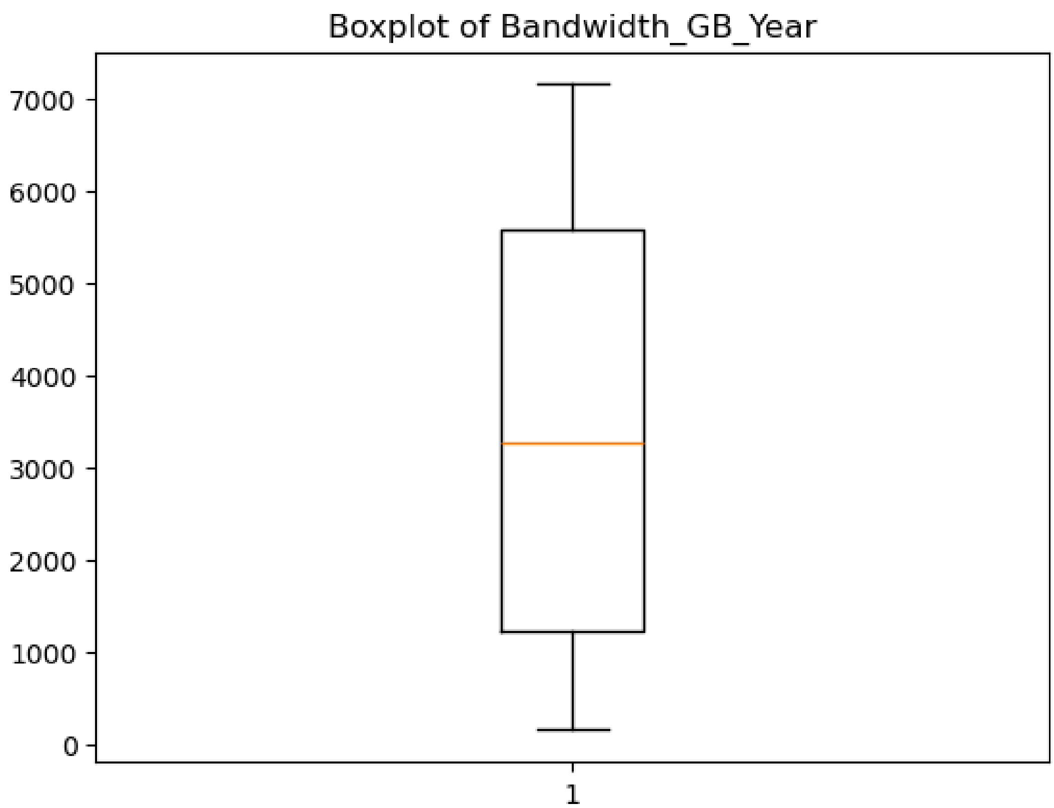
For the `Yearly_equip_failure` variable, all observations greater than 2.5 are considered outliers.
The count of observations greater than 2.5 is 94.



There are no outliers for the `Tenure` variable.



There are no outliers for the `MonthlyCharge` variable.



There are no outliers for the `Bandwidth_GB_Year` variable.

[5]:
```
## C1 Binary Encoding Re-expression of the Thirteen Binary Variables

# It was necessary to keep a copy of the Churn variable as strings for the␣
↪bivariate graphs
df['ChurnStr'] = df['Churn'].copy()

# Create a list of the columns that will be encoded
binaryList = ['Churn', 'Techie', 'Port_modem', 'Tablet', 'Phone',
              'Multiple', 'OnlineSecurity', 'OnlineBackup',␣
↪'DeviceProtection',
              'TechSupport', 'StreamingTV', 'StreamingMovies',␣
↪'PaperlessBilling']

# Run a loop that replaces all 'Yes' with 1 and 'No' with 0 for each column in␣
↪the list above
for col in binaryList:
    df[col] = df[col].replace({'Yes': 1, 'No': 0})

binaryVars = df[['Techie', 'Port_modem', 'Tablet', 'Phone',
```

```
                 'Multiple', 'OnlineSecurity', 'OnlineBackup',␣
 ↪'DeviceProtection',
                 'TechSupport', 'StreamingTV', 'StreamingMovies',␣
 ↪'PaperlessBilling']]
```

[6]:
```
## C1 One-Hot Encoding Re-expression of the Six Categorical Variables

# Data frame of categorical variables
catVars = df[['Area', 'Marital', 'Gender', 'Contract', 'InternetService',␣
 ↪'PaymentMethod']]

# Create additional data frame from variables being re-expressed
oneHotVars = catVars

# Apply one-hot encoding, dropping first column to avoid multi-collinearity in␣
 ↪the model
oneHotVars = pd.get_dummies(oneHotVars, drop_first = True, dtype = int)
```

[7]:
```
## C1 Type Casting Ordinal and Binary Variables

# This was necessary for graphs below to behave correctly
ordinalVars = df[['Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6',␣
 ↪'Item7', 'Item8']]

binaryVars = binaryVars.astype('category')

df['Churn'] = df['Churn'].astype('category')

# assign `Churn` to depVar
depVar = df['Churn']
```

[8]:
```
## C1 Prepared Data Frame

df_prep = pd.concat([depVar, numericVars, binaryVars, oneHotVars, ordinalVars],␣
 ↪axis = 1)
```

3.1.2 C2. Summary Statistics of Dependent Variable and Independent Variables

The dependent variable for this logistic regression model will be the binary categorical variable `Churn`.

The independent variables for the initial model will be:

- all thirteen numeric variables
  - `Lat`, `Lng`, `Population`, `Children`, `Age`, `Income`, `Outage_sec_perweek`, `Email`, `Contacts`, `Yearly_equip_failure`, `Tenure`, `MonthlyCharge`, `Bandwidth_GB_Year`
- all twelve re-expressed binary variables
  - `Techie`, `Port_modem`, `Tablet`, `Phone`, `Multiple`, `OnlineSecurity`,

  `OnlineBackup`, `DeviceProtection`, `TechSupport`, `StreamingTV`, `StreamingMovies`, `PaperlessBilling`
- all six categorical variables re-expressed with one-hot encoding
  - `Area`, `Marital`, `Gender`, `Contract`, `InternetService`, `PaymentMethod`
- all eight ordinal variables
  - `Item1`, `Item2`, `Item3`, `Item4`, `Item5`, `Item6`, `Item7`, `Item8`

The written descriptions of the summary statistics for these variables are shown below.

[9]:
```python
## C2 Summaries of Numeric and Ordinal Variables

print('Summaries of Numeric Variables')
print(numericVars.describe())

print('Summaries of Ordinal Variables')
print(ordinalVars.describe())
```

```
Summaries of Numeric Variables
                Lat           Lng     Population    Children           Age  \
count  10000.000000  10000.000000   10000.000000  10000.0000  10000.000000
mean      38.757567    -90.782536    9756.562400      2.0877     53.078400
std        5.437389     15.156142   14432.698671      2.1472     20.698882
min       17.966120   -171.688150       0.000000      0.0000     18.000000
25%       35.341828    -97.082812     738.000000      0.0000     35.000000
50%       39.395800    -87.918800    2910.500000      1.0000     53.000000
75%       42.106908    -80.088745   13168.000000      3.0000     71.000000
max       70.640660    -65.667850  111850.000000     10.0000     89.000000

              Income  Outage_sec_perweek         Email      Contacts  \
count   10000.000000        10000.000000  10000.000000  10000.000000
mean    39806.926771           10.001848     12.016000      0.994200
std     28199.916702            2.976019      3.025898      0.988466
min       348.670000            0.099747      1.000000      0.000000
25%     19224.717500            8.018214     10.000000      0.000000
50%     33170.605000           10.018560     12.000000      1.000000
75%     53246.170000           11.969485     14.000000      2.000000
max    258900.700000           21.207230     23.000000      7.000000

       Yearly_equip_failure        Tenure  MonthlyCharge  Bandwidth_GB_Year
count          10000.000000  10000.000000   10000.000000       10000.000000
mean               0.398000     34.526188     172.624816        3392.341550
std                0.635953     26.443063      42.943094        2185.294852
min                0.000000      1.000259      79.978860         155.506715
25%                0.000000      7.917694     139.979239        1236.470827
50%                0.000000     35.430507     167.484700        3279.536903
75%                1.000000     61.479795     200.734725        5586.141370
max                6.000000     71.999280     290.160419        7158.981530
Summaries of Ordinal Variables
              Item1         Item2         Item3         Item4         Item5  \
```

```
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
mean       3.490800      3.505100      3.487000      3.497500      3.492900
std        1.037797      1.034641      1.027977      1.025816      1.024819
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000      3.000000      3.000000
50%        3.000000      4.000000      3.000000      3.000000      3.000000
75%        4.000000      4.000000      4.000000      4.000000      4.000000
max        7.000000      7.000000      8.000000      7.000000      7.000000

              Item6         Item7         Item8
count  10000.000000  10000.000000  10000.000000
mean       3.497300      3.509500      3.495600
std        1.033586      1.028502      1.028633
min        1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000
50%        3.000000      4.000000      3.000000
75%        4.000000      4.000000      4.000000
max        8.000000      7.000000      8.000000
```

[10]:
```python
## C2 Summaries of Numeric Variables

def quantDesc(data_frame, col_name):
# Provide written description of the statistical summary output of .describe()

    count, mean, std, minimum, quarter, half, seventyfive, maximum = data_frame[col_name].describe()

    print(f'For the variable `{col_name}`:')
    print(f'There are {count} observations.')
    print(f'On average, the data tends towards the mean, which is {round(mean, 2)}.')
    print(f'The standard deviation {round(std, 2)} is the amount of variation, or how much the data differs from the mean.')
    print(f'The smallest observation is {round(minimum, 2)}.')
    print(f'25% of the data falls below {round(quarter, 2)}.')
    print(f'50% of the data falls below {round(half, 2)}.')
    print(f'75% of the data falls below {round(seventyfive, 2)}.')
    print(f'100% of the data falls below the largest observation, {round(maximum, 2)}.')
    print('\n')
```

[11]:
```python
## C2 Summaries of 13 Numeric Variables

for col in numericVars:
    quantDesc(df_prep, col)
```

```
For the variable `Lat`:
There are 10000.0 observations.
```

```
On average, the data tends towards the mean, which is 38.76.
The standard deviation 5.44 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 17.97.
25% of the data falls below 35.34.
50% of the data falls below 39.4.
75% of the data falls below 42.11.
100% of the data falls below the largest observation, 70.64.


For the variable `Lng`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is -90.78.
The standard deviation 15.16 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is -171.69.
25% of the data falls below -97.08.
50% of the data falls below -87.92.
75% of the data falls below -80.09.
100% of the data falls below the largest observation, -65.67.


For the variable `Population`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 9756.56.
The standard deviation 14432.7 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 0.0.
25% of the data falls below 738.0.
50% of the data falls below 2910.5.
75% of the data falls below 13168.0.
100% of the data falls below the largest observation, 111850.0.


For the variable `Children`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 2.09.
The standard deviation 2.15 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 0.0.
25% of the data falls below 0.0.
50% of the data falls below 1.0.
75% of the data falls below 3.0.
100% of the data falls below the largest observation, 10.0.


For the variable `Age`:
There are 10000.0 observations.
```

```
On average, the data tends towards the mean, which is 53.08.
The standard deviation 20.7 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 18.0.
25% of the data falls below 35.0.
50% of the data falls below 53.0.
75% of the data falls below 71.0.
100% of the data falls below the largest observation, 89.0.


For the variable `Income`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 39806.93.
The standard deviation 28199.92 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 348.67.
25% of the data falls below 19224.72.
50% of the data falls below 33170.6.
75% of the data falls below 53246.17.
100% of the data falls below the largest observation, 258900.7.


For the variable `Outage_sec_perweek`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 10.0.
The standard deviation 2.98 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 0.1.
25% of the data falls below 8.02.
50% of the data falls below 10.02.
75% of the data falls below 11.97.
100% of the data falls below the largest observation, 21.21.


For the variable `Email`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 12.02.
The standard deviation 3.03 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 1.0.
25% of the data falls below 10.0.
50% of the data falls below 12.0.
75% of the data falls below 14.0.
100% of the data falls below the largest observation, 23.0.


For the variable `Contacts`:
There are 10000.0 observations.
```

On average, the data tends towards the mean, which is 0.99.
The standard deviation 0.99 is the amount of variation, or how much the data differs from the mean.
The smallest observation is 0.0.
25% of the data falls below 0.0.
50% of the data falls below 1.0.
75% of the data falls below 2.0.
100% of the data falls below the largest observation, 7.0.

For the variable `Yearly_equip_failure`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 0.4.
The standard deviation 0.64 is the amount of variation, or how much the data differs from the mean.
The smallest observation is 0.0.
25% of the data falls below 0.0.
50% of the data falls below 0.0.
75% of the data falls below 1.0.
100% of the data falls below the largest observation, 6.0.

For the variable `Tenure`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 34.53.
The standard deviation 26.44 is the amount of variation, or how much the data differs from the mean.
The smallest observation is 1.0.
25% of the data falls below 7.92.
50% of the data falls below 35.43.
75% of the data falls below 61.48.
100% of the data falls below the largest observation, 72.0.

For the variable `MonthlyCharge`:
There are 10000.0 observations.
On average, the data tends towards the mean, which is 172.62.
The standard deviation 42.94 is the amount of variation, or how much the data differs from the mean.
The smallest observation is 79.98.
25% of the data falls below 139.98.
50% of the data falls below 167.48.
75% of the data falls below 200.73.
100% of the data falls below the largest observation, 290.16.

For the variable `Bandwidth_GB_Year`:
There are 10000.0 observations.

```
On average, the data tends towards the mean, which is 3392.34.
The standard deviation 2185.29 is the amount of variation, or how much the data
differs from the mean.
The smallest observation is 155.51.
25% of the data falls below 1236.47.
50% of the data falls below 3279.54.
75% of the data falls below 5586.14.
100% of the data falls below the largest observation, 7158.98.
```

[12]:

```
## C2 Summaries of Binary Variables

def binarySummary(data_frame, col_name):
# Get the counts, convert counts to percentages,
# and calculate and display summary statistics

    counts = data_frame[col_name].value_counts()
    percentages = counts / counts.sum() * 100

    summary_stats = pd.DataFrame({'Count': counts, 'Percentage': percentages})
    summary_stats = summary_stats.rename(index={0: 'no', 1: 'yes'})

    print(f'Summary of `{col_name}`')
    print(summary_stats)
    print('\n')
```

[13]:

```
## C2 Summaries of 13 Binary Variables including Dependent Variable

print('Summary of dependent variable `Churn`')
binarySummary(df_prep, 'Churn')

for col in binaryVars:
    binarySummary(df, col)
```

```
Summary of dependent variable `Churn`
Summary of `Churn`
       Count  Percentage
Churn
no      7350        73.5
yes     2650        26.5


Summary of `Techie`
        Count  Percentage
Techie
no       8321       83.21
yes      1679       16.79
```

```
Summary of `Port_modem`
            Count  Percentage
Port_modem
no           5166       51.66
yes          4834       48.34


Summary of `Tablet`
        Count  Percentage
Tablet
no       7009       70.09
yes      2991       29.91


Summary of `Phone`
       Count  Percentage
Phone
yes     9067       90.67
no       933        9.33


Summary of `Multiple`
          Count  Percentage
Multiple
no         5392       53.92
yes        4608       46.08


Summary of `OnlineSecurity`
                Count  Percentage
OnlineSecurity
no               6424       64.24
yes              3576       35.76


Summary of `OnlineBackup`
              Count  Percentage
OnlineBackup
no             5494       54.94
yes            4506       45.06


Summary of `DeviceProtection`
                  Count  Percentage
DeviceProtection
no                 5614       56.14
```

```
yes                 4386       43.86


Summary of `TechSupport`
             Count  Percentage
TechSupport
no            6250        62.5
yes           3750        37.5


Summary of `StreamingTV`
             Count  Percentage
StreamingTV
no            5071       50.71
yes           4929       49.29


Summary of `StreamingMovies`
                 Count  Percentage
StreamingMovies
no                5110        51.1
yes               4890        48.9


Summary of `PaperlessBilling`
                  Count  Percentage
PaperlessBilling
yes                5882       58.82
no                 4118       41.18
```

[14]:
```python
## C2 Summaries of Categorical Variables

def catSummary(data_frame, col_name):
# Get the counts, convert counts to percentages,
# and calculate and display summary statistics

    counts = data_frame[col_name].value_counts()
    percentages = counts / counts.sum() * 100

    summary_stats = pd.DataFrame({'Count': counts, 'Percentage': percentages})

    print(f'Summary of `{col_name}`')
    print(summary_stats)
    print('\n')
```

[15]:
```python
## C2 Summaries

# Summaries of Six Categorical Variables
for col in catVars:
    catSummary(df, col)

# Summaries of Eight Ordinal Variables
for col in ordinalVars:
    catSummary(df_prep, col)
```

```
Summary of `Area`
          Count  Percentage
Area
Suburban   3346       33.46
Urban      3327       33.27
Rural      3327       33.27



Summary of `Marital`
               Count  Percentage
Marital
Divorced        2092       20.92
Widowed         2027       20.27
Separated       2014       20.14
Never Married   1956       19.56
Married         1911       19.11



Summary of `Gender`
           Count  Percentage
Gender
Female      5025       50.25
Male        4744       47.44
Nonbinary    231        2.31



Summary of `Contract`
                Count  Percentage
Contract
Month-to-month   5456       54.56
Two Year         2442       24.42
One year         2102       21.02



Summary of `InternetService`
                 Count  Percentage
InternetService
Fiber Optic       4408       44.08
```

DSL	3463	34.63
None	2129	21.29

Summary of `PaymentMethod`

PaymentMethod	Count	Percentage
Electronic Check	3398	33.98
Mailed Check	2290	22.90
Bank Transfer(automatic)	2229	22.29
Credit Card (automatic)	2083	20.83

Summary of `Item1`

Item1	Count	Percentage
3	3448	34.48
4	3358	33.58
2	1393	13.93
5	1359	13.59
1	224	2.24
6	199	1.99
7	19	0.19

Summary of `Item2`

Item2	Count	Percentage
3	3415	34.15
4	3412	34.12
5	1368	13.68
2	1360	13.60
1	217	2.17
6	215	2.15
7	13	0.13

Summary of `Item3`

Item3	Count	Percentage
3	3435	34.35
4	3410	34.10
2	1424	14.24
5	1313	13.13
6	203	2.03
1	202	2.02
7	12	0.12
8	1	0.01

```
Summary of `Item4`
      Count  Percentage
Item4
4       3452       34.52
3       3430       34.30
2       1350       13.50
5       1335       13.35
1        221        2.21
6        203        2.03
7          9        0.09
```

```
Summary of `Item5`
      Count  Percentage
Item5
3       3462       34.62
4       3417       34.17
2       1378       13.78
5       1321       13.21
1        206        2.06
6        204        2.04
7         12        0.12
```

```
Summary of `Item6`
      Count  Percentage
Item6
3       3445       34.45
4       3333       33.33
2       1427       14.27
5       1382       13.82
6        210        2.10
1        190        1.90
7         12        0.12
8          1        0.01
```

```
Summary of `Item7`
      Count  Percentage
Item7
4       3456       34.56
3       3446       34.46
5       1335       13.35
2       1309       13.09
6        224        2.24
1        219        2.19
```

```
7          11         0.11


Summary of `Item8`
       Count  Percentage
Item8
3       3461       34.61
4       3400       34.00
2       1378       13.78
5       1335       13.35
1        206        2.06
6        205        2.05
7         14        0.14
8          1        0.01
```

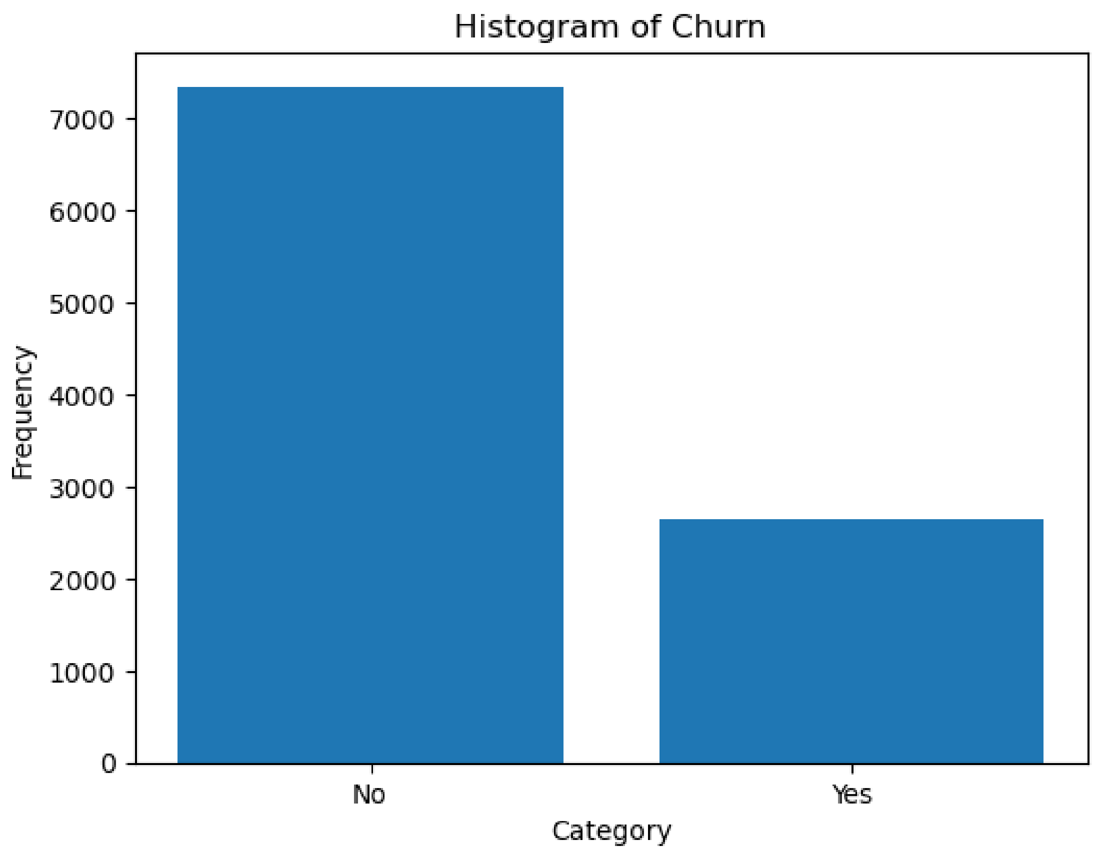
3.1.3 C3. Univariate and Bivariate Visualizations

The code below generates the univariate and bivariate visualizations of the distributions of the dependent and independent variables. The bivariate visualizations consider `Churn` as the dependent variable.

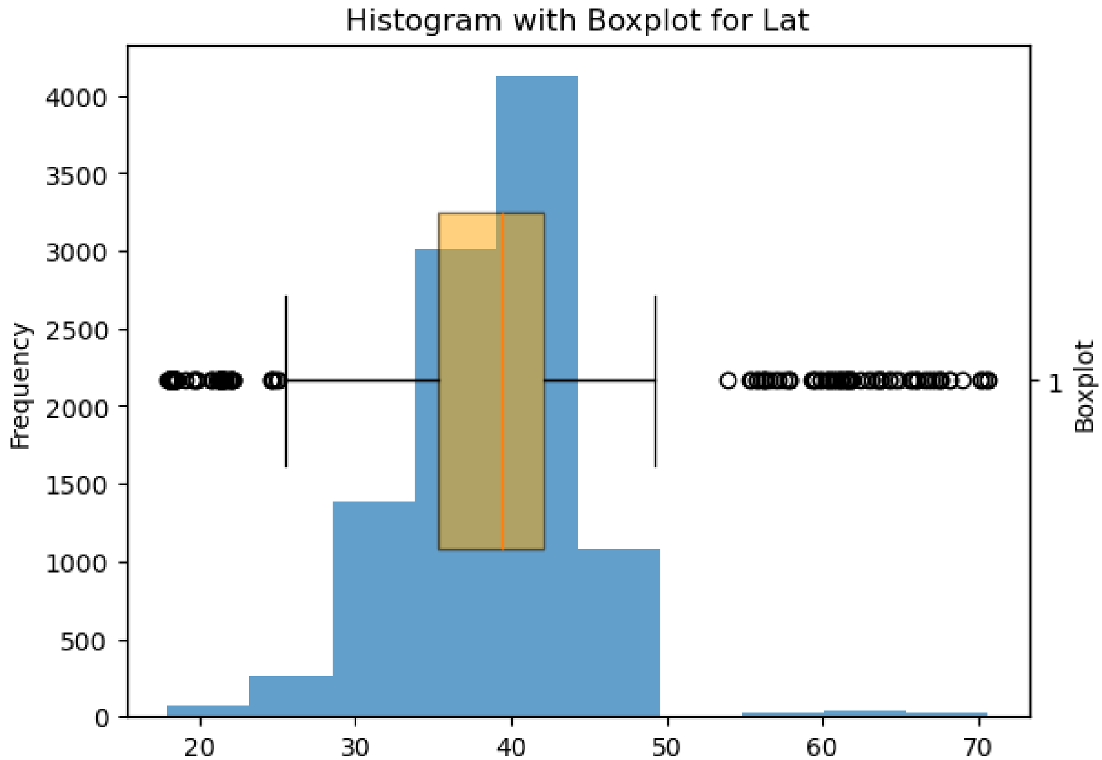
[16]:
```python
## C3 Univariate Visualizations

import matplotlib.pyplot as plt


def plot_histobox(data_frame, col_name):
# Overlays a transparent boxplot over a histogram
    data = data_frame[col_name].values
    fig, ax1 = plt.subplots()

    ax1.hist(data, bins = 10, alpha = 0.7, label = 'Histogram')
    ax2 = ax1.twinx()
    ax2.boxplot(data, vert = False, widths = 0.5, patch_artist = True,
                boxprops = dict(facecolor = 'orange', alpha = 0.5))

    ax1.set_ylabel('Frequency')
    ax2.set_ylabel('Boxplot')

    plt.title(f'Histogram with Boxplot for {col_name}')
    plt.show()


def plot_binaryhist(data_frame, col_name):
# Histogram of a binary variable
    catCounts = data_frame[col_name].value_counts()
```

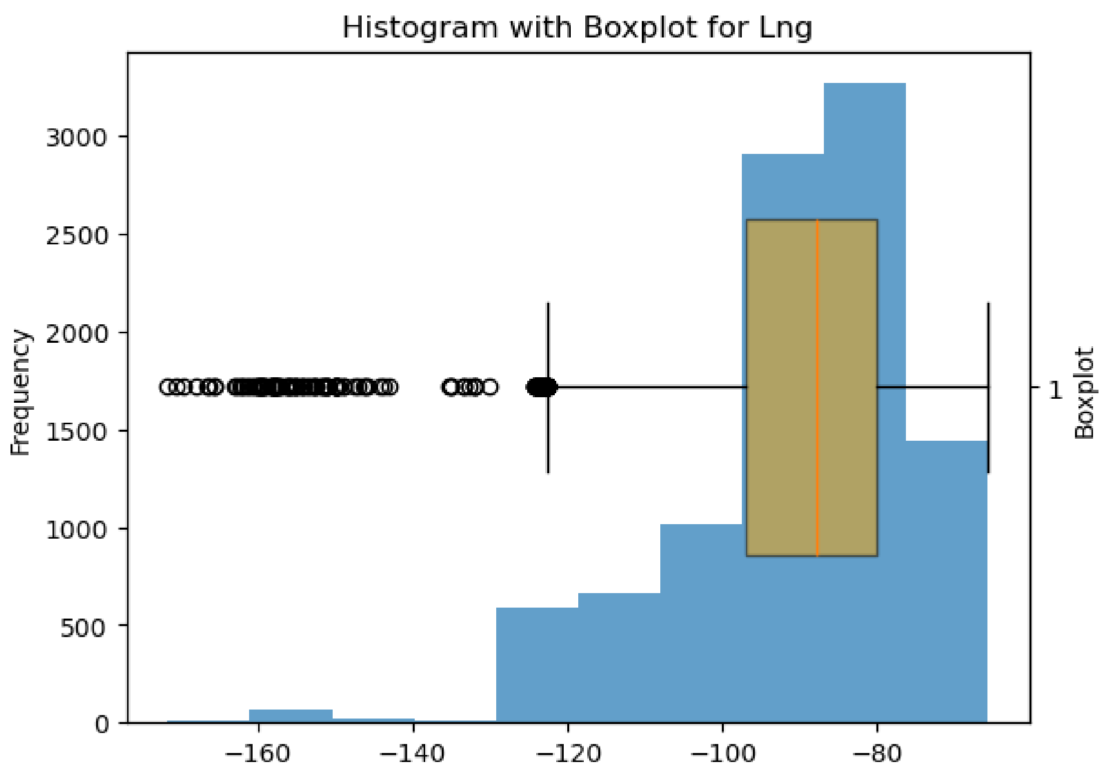
```python
    cats = catCounts.index.tolist()
    counts = catCounts.values.tolist()

    plt.bar(cats, counts)
    plt.xticks(ticks = [0, 1], labels = ['No', 'Yes'])
    plt.title(f'Histogram of {col_name}')
    plt.xlabel('Category')
    plt.ylabel('Frequency')
    plt.show()


def plot_cathist(data_frame, col_name):
# Histogram of a categorical variable
    catCounts = data_frame[col_name].value_counts()
    cats = catCounts.index.tolist()
    counts = catCounts.values.tolist()

    plt.bar(cats, counts)
    plt.title(f'Histogram of {col_name}')
    plt.xlabel('Category')
    plt.ylabel('Frequency')
    plt.show()
```

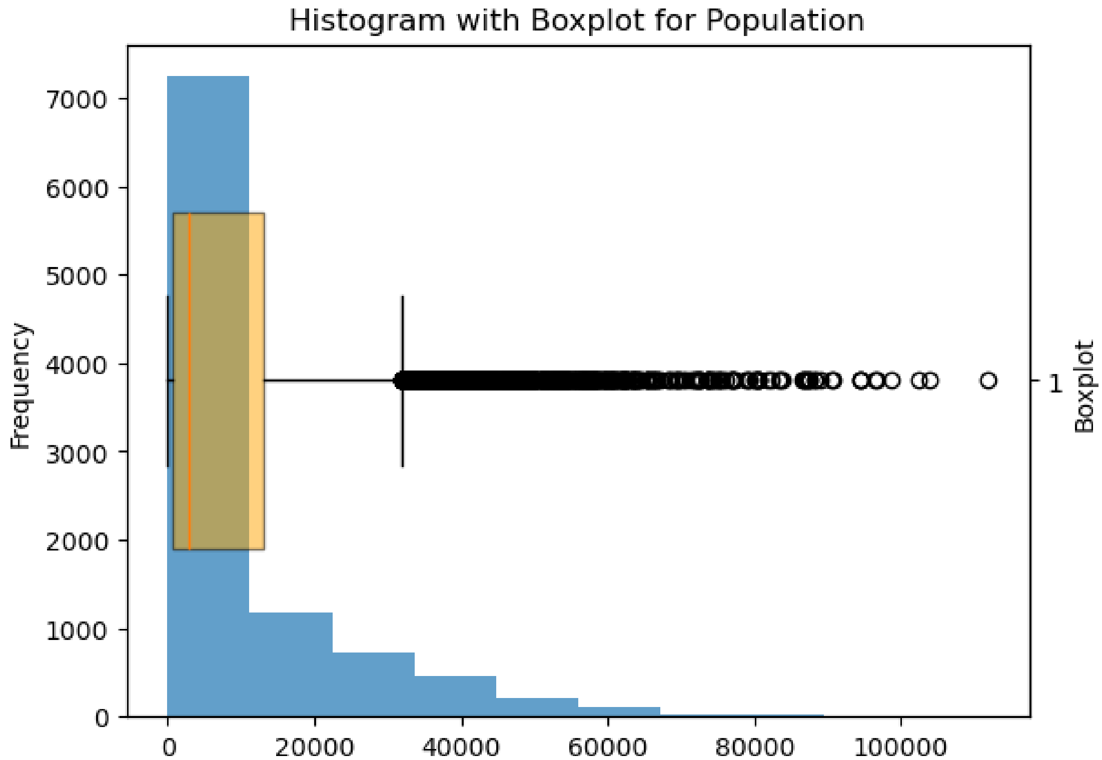
[17]:
```python
## C3 Univariate Visualizations

# Histograms + Boxplots for the dependent variable
plot_binaryhist(df_prep, 'Churn')



# Histograms + Boxplots for 13 Numeric Variables
for col in numericVars:
    plot_histobox(df_prep, col)

# Histograms for 12 Binary Variables
for col in binaryVars:
    plot_binaryhist(df_prep, col)

# Histograms for 6 Categorical Variables
for col in catVars:
    plot_cathist(df, col)

# Histograms for 8 Ordinal Variables
for col in ordinalVars:
    plot_cathist(df_prep, col)
```



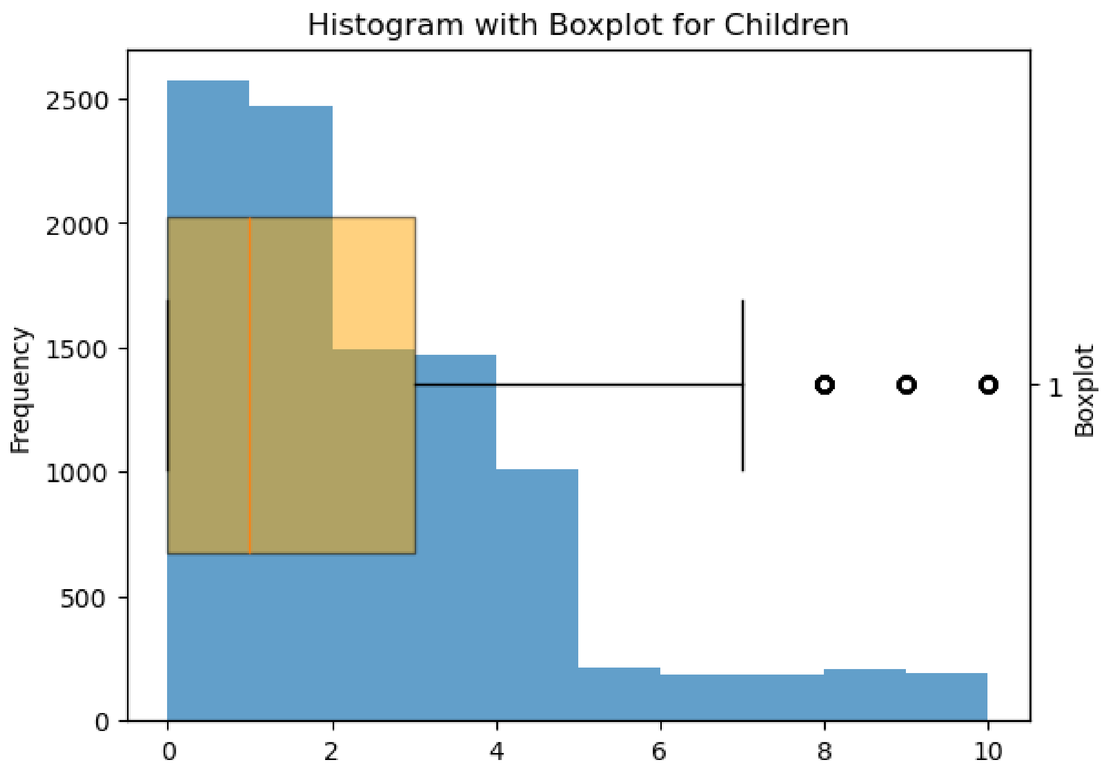
Histogram of Churn
Frequency
Category
0
1000
2000
3000
4000
5000
6000
7000
No
Yes


Histogram with Boxplot for Lat
Frequency
Boxplot
0
500
1000
1500
2000
2500
3000
3500
4000
20
30
40
50
60
70
1

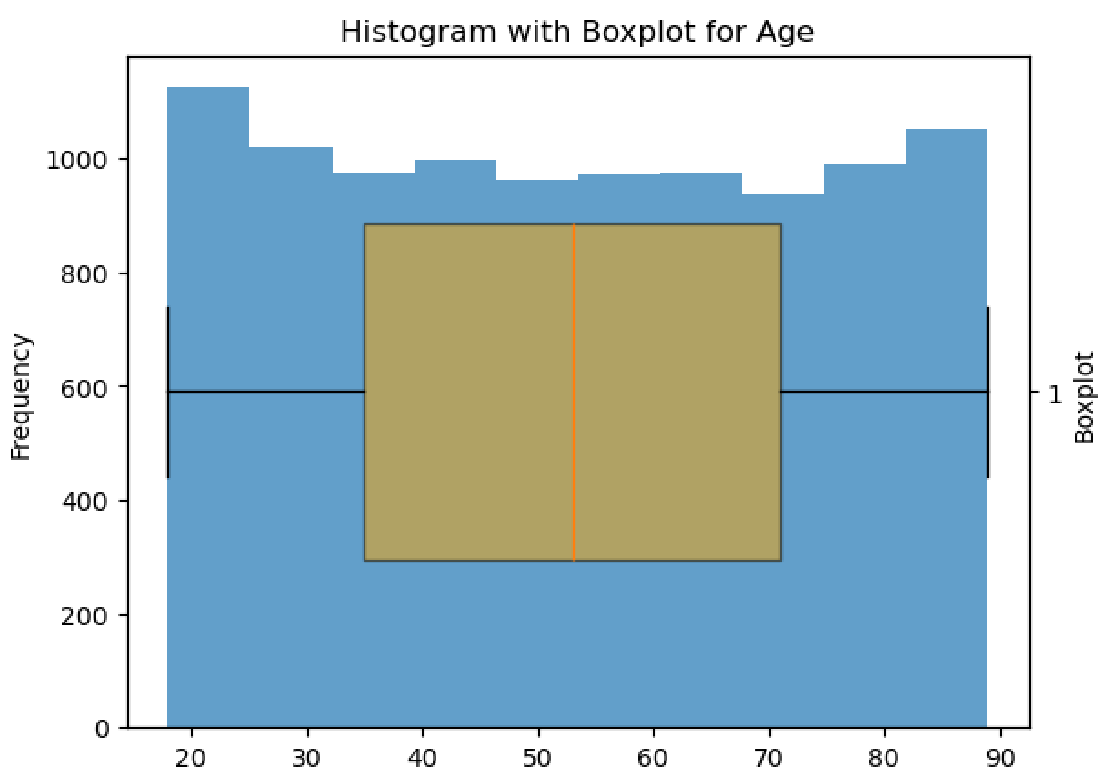

Histogram with Boxplot for Lng
Frequency
Boxplot
0
500
1000
1500
2000
2500
3000
−160
−140
−120
−100
−80
1



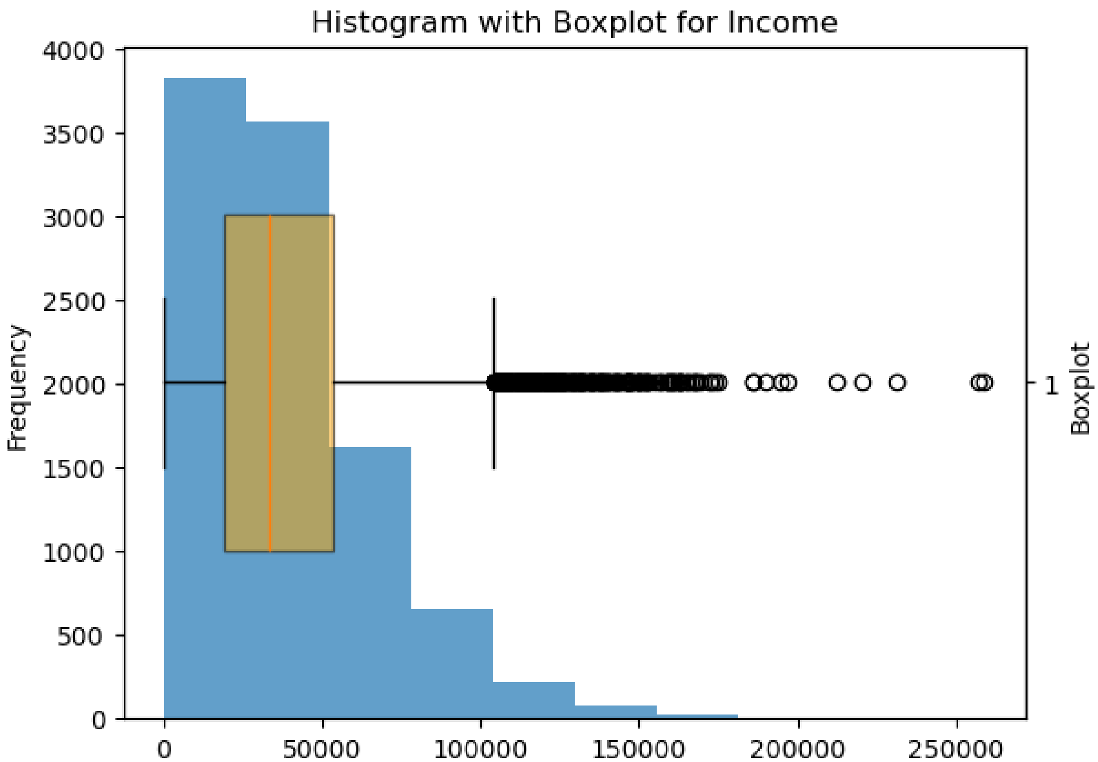
Histogram with Boxplot for Population
Frequency
Boxplot
0
1000
2000
3000
4000
5000
6000
7000
0
20000
40000
60000
80000
100000
1

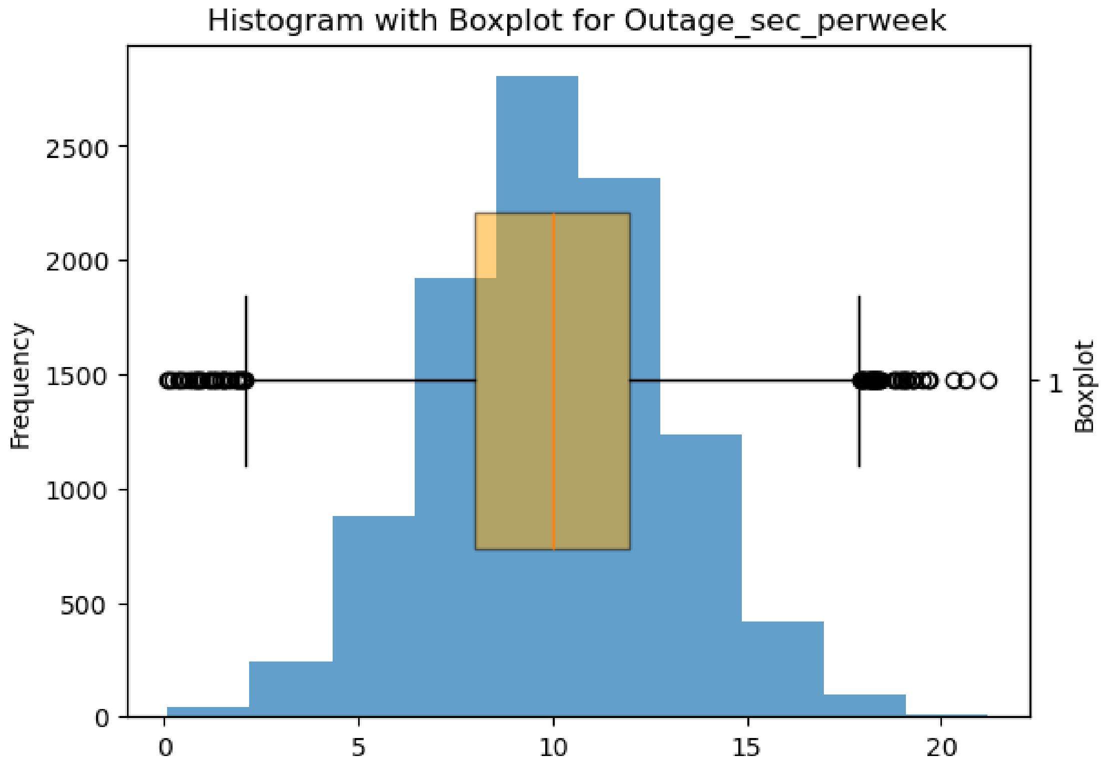


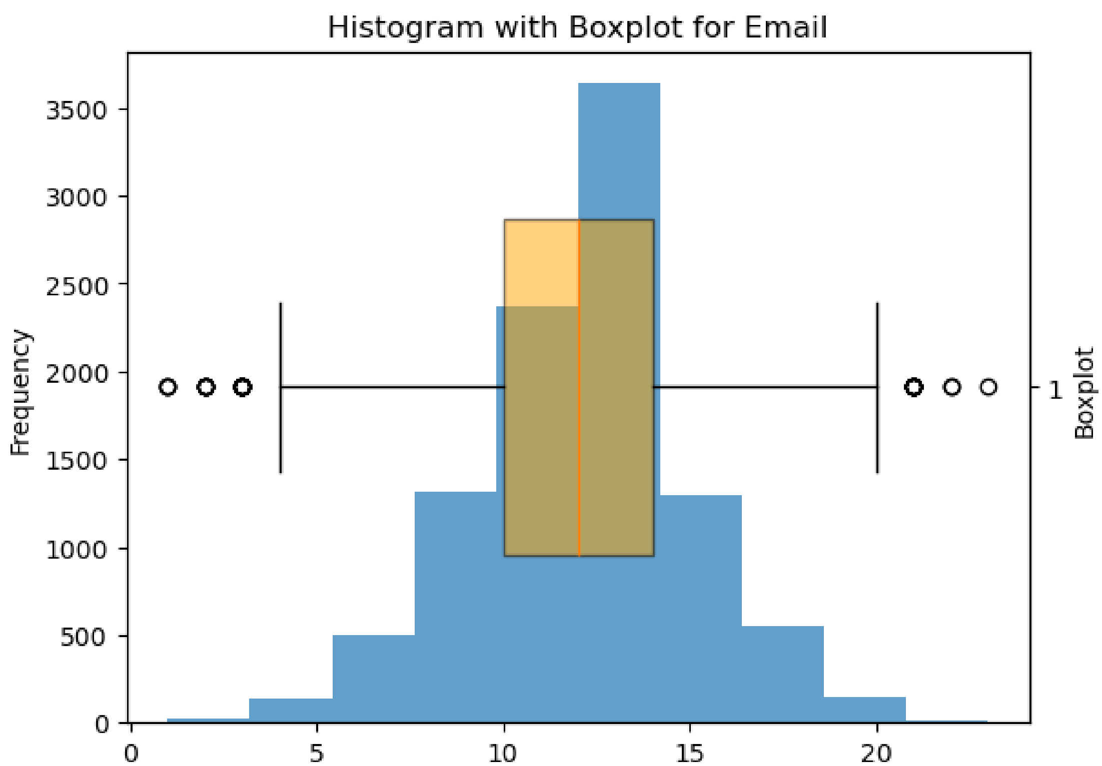
Histogram with Boxplot for Children
Frequency
Boxplot
0
500
1000
1500
2000
2500
0
2
4
6
8
10
1

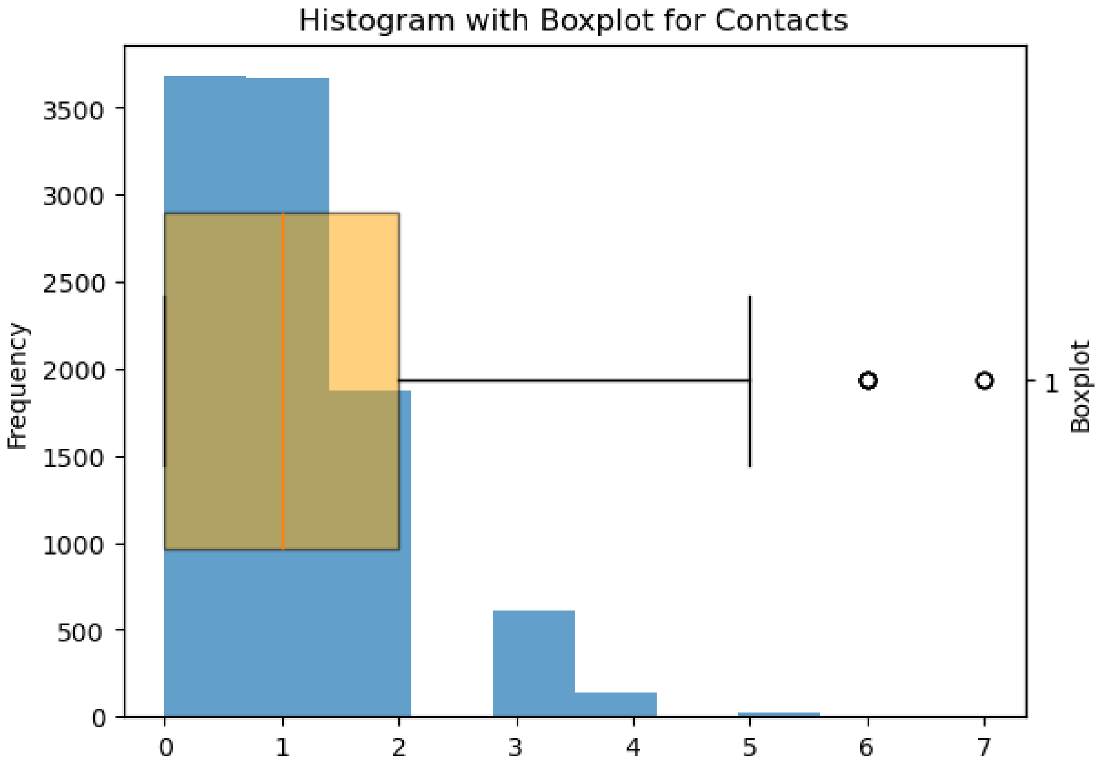


Histogram with Boxplot for Age
Frequency
Boxplot
0
200
400
600
800
1000
20
30
40
50
60
70
80
90
1

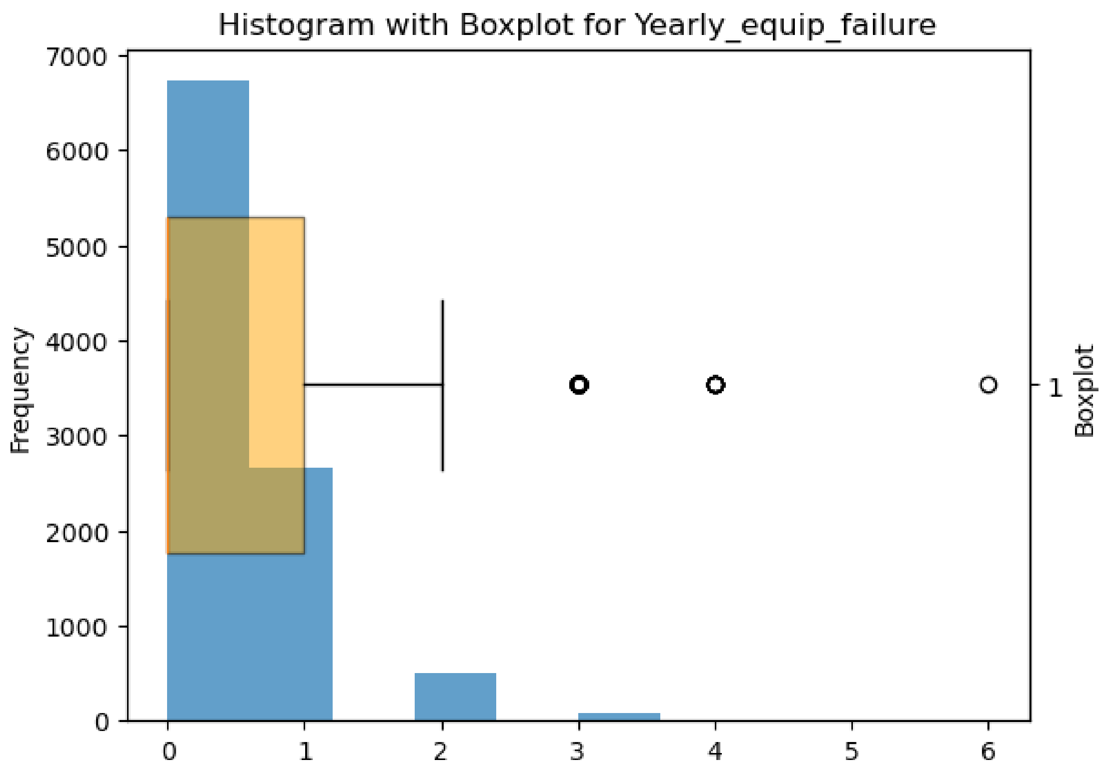

Histogram with Boxplot for Income
Frequency
Boxplot
0
500
1000
1500
2000
2500
3000
3500
4000
50000
100000
150000
200000
250000
1

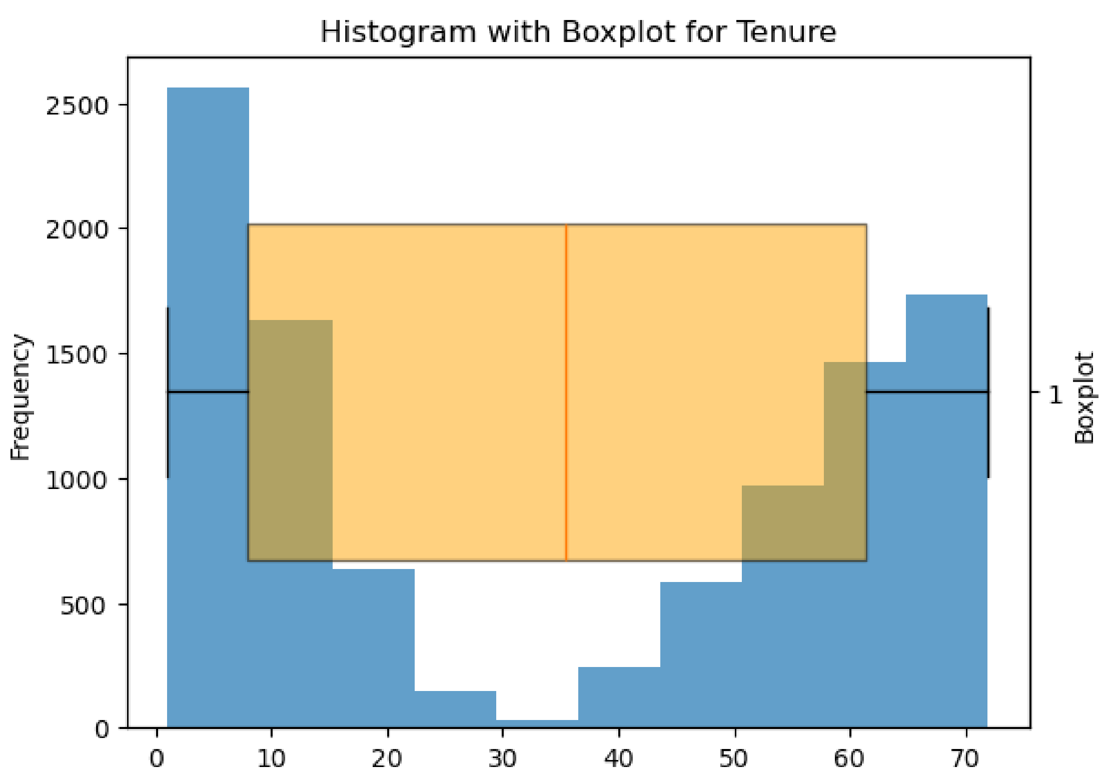

Histogram with Boxplot for Outage_sec_perweek
Frequency
Boxplot
0
500
1000
1500
2000
2500
0
5
10
15
20
1

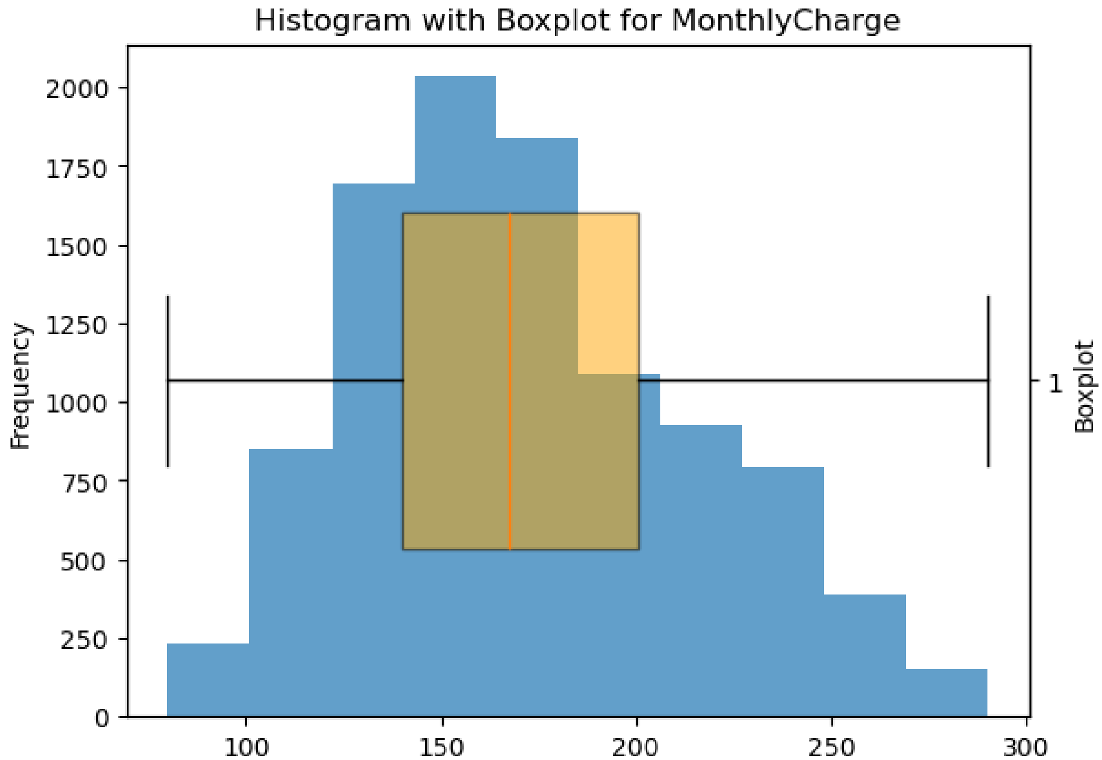

Histogram with Boxplot for Email
Frequency
Boxplot
0
500
1000
1500
2000
2500
3000
3500
0
5
10
15
20
1



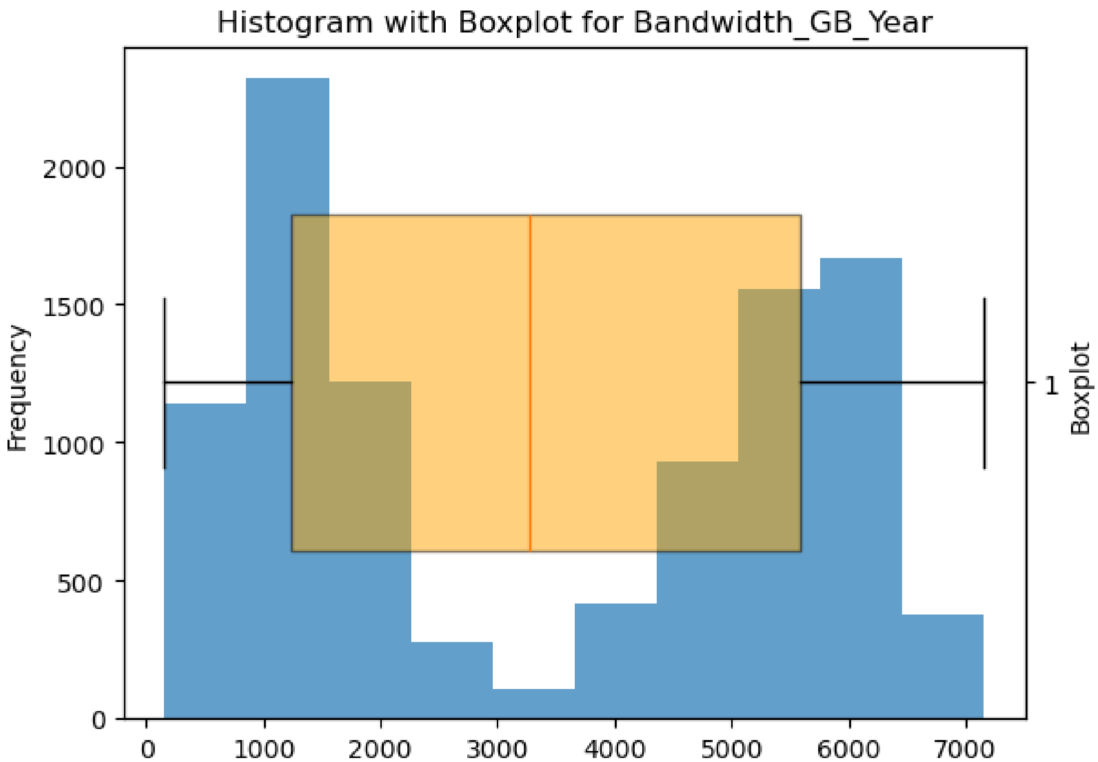
Histogram with Boxplot for Contacts
Frequency
Boxplot
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
1

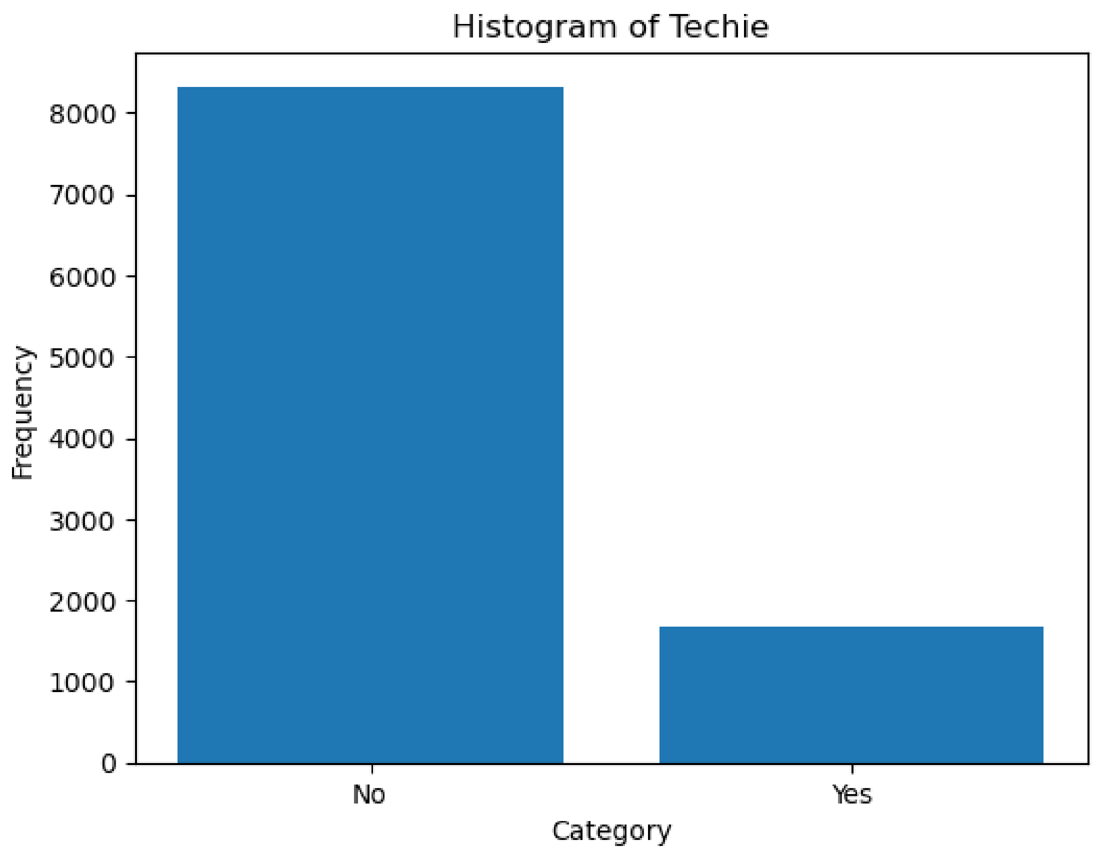

Histogram with Boxplot for Yearly_equip_failure
Frequency
Boxplot
0
1000
2000
3000
4000
5000
6000
7000
0
1
2
3
4
5
6
1

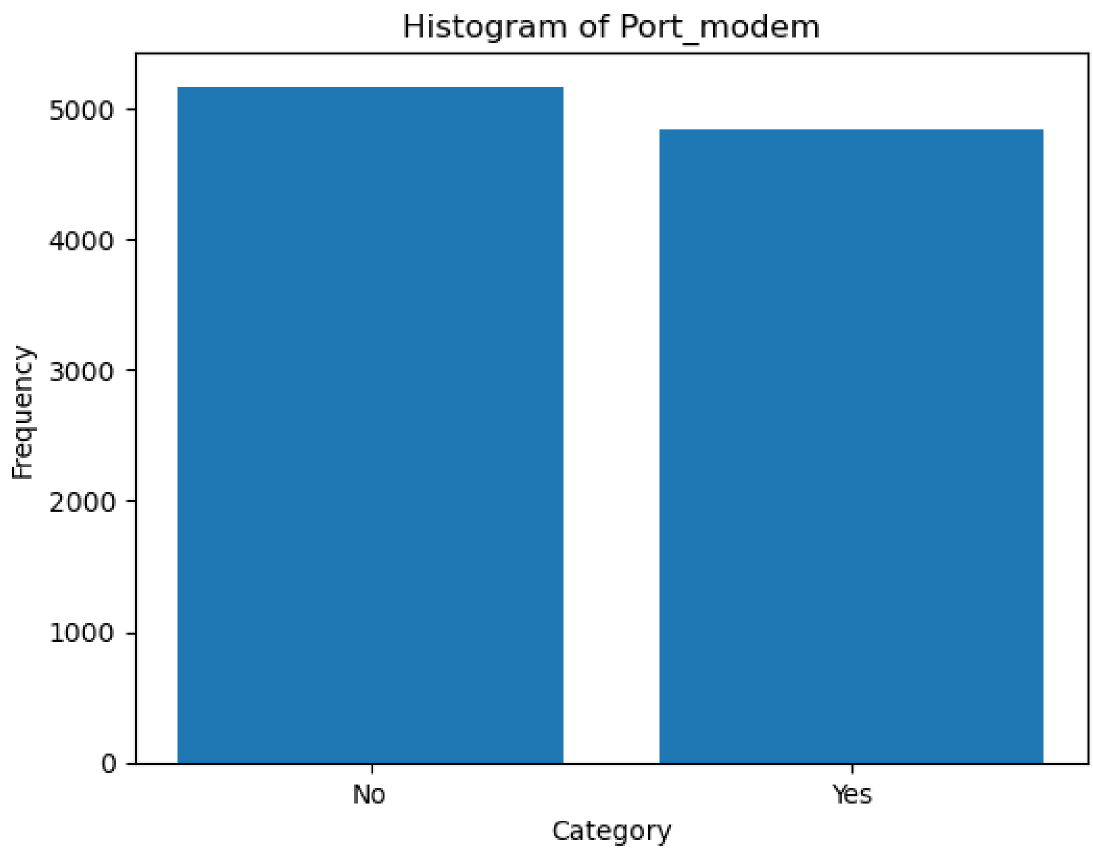

Histogram with Boxplot for Tenure
Frequency
Boxplot
0
500
1000
1500
2000
2500
0
10
20
30
40
50
60
70
1

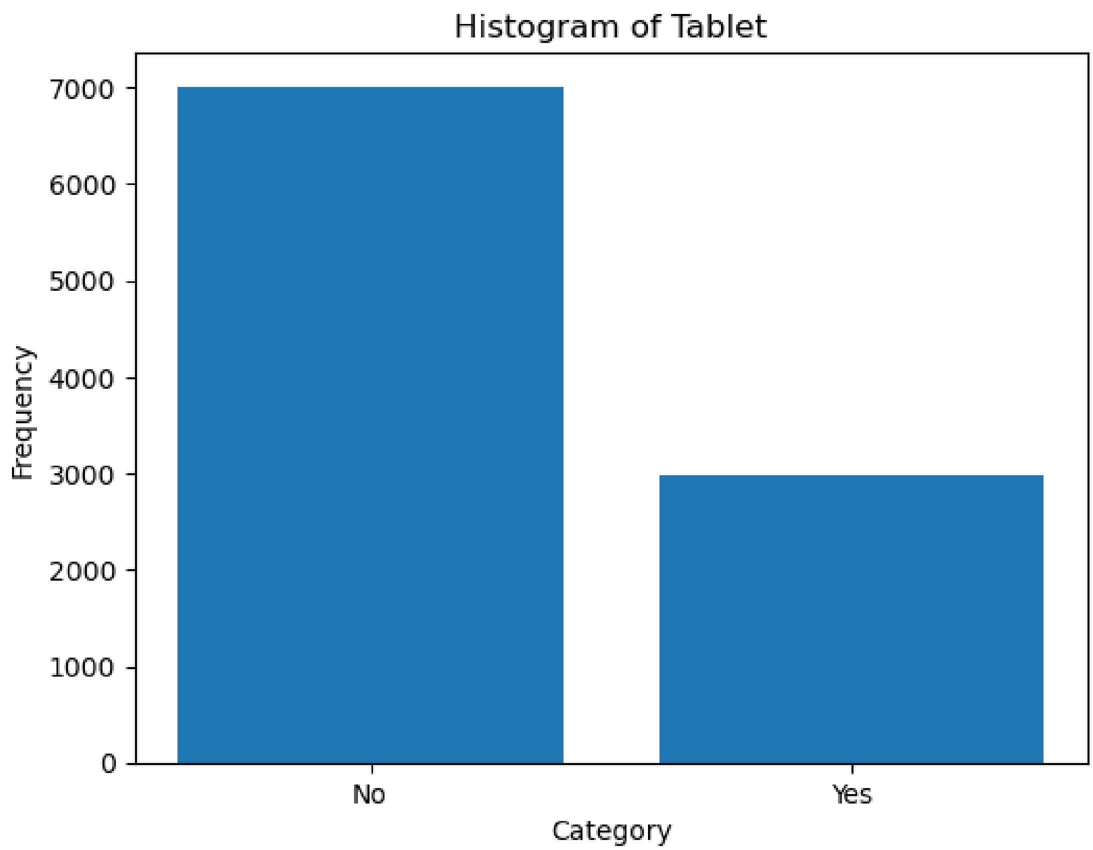


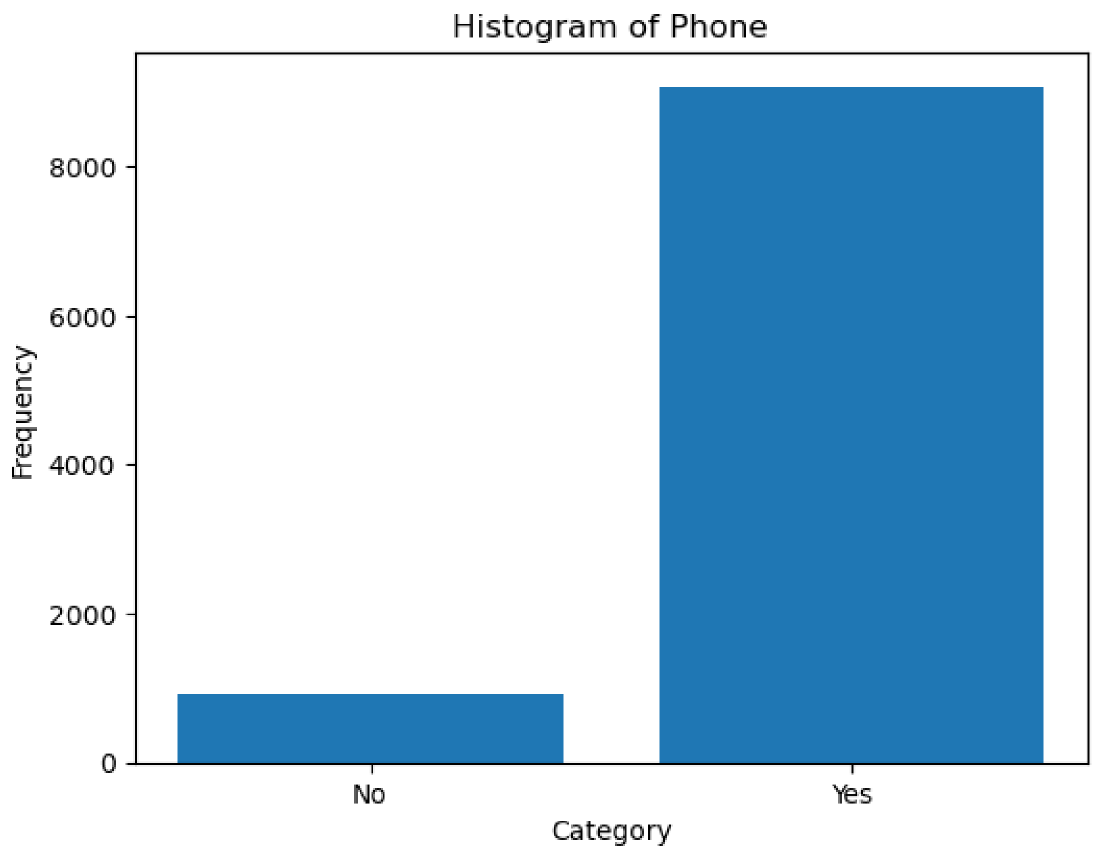
Histogram with Boxplot for MonthlyCharge
Frequency
Boxplot
0
250
500
750
1000
1250
1500
1750
2000
100
150
200
250
300
1


Histogram with Boxplot for Bandwidth_GB_Year
Frequency
Boxplot
0
500
1000
1500
2000
0
1000
2000
3000
4000
5000
6000
7000
1

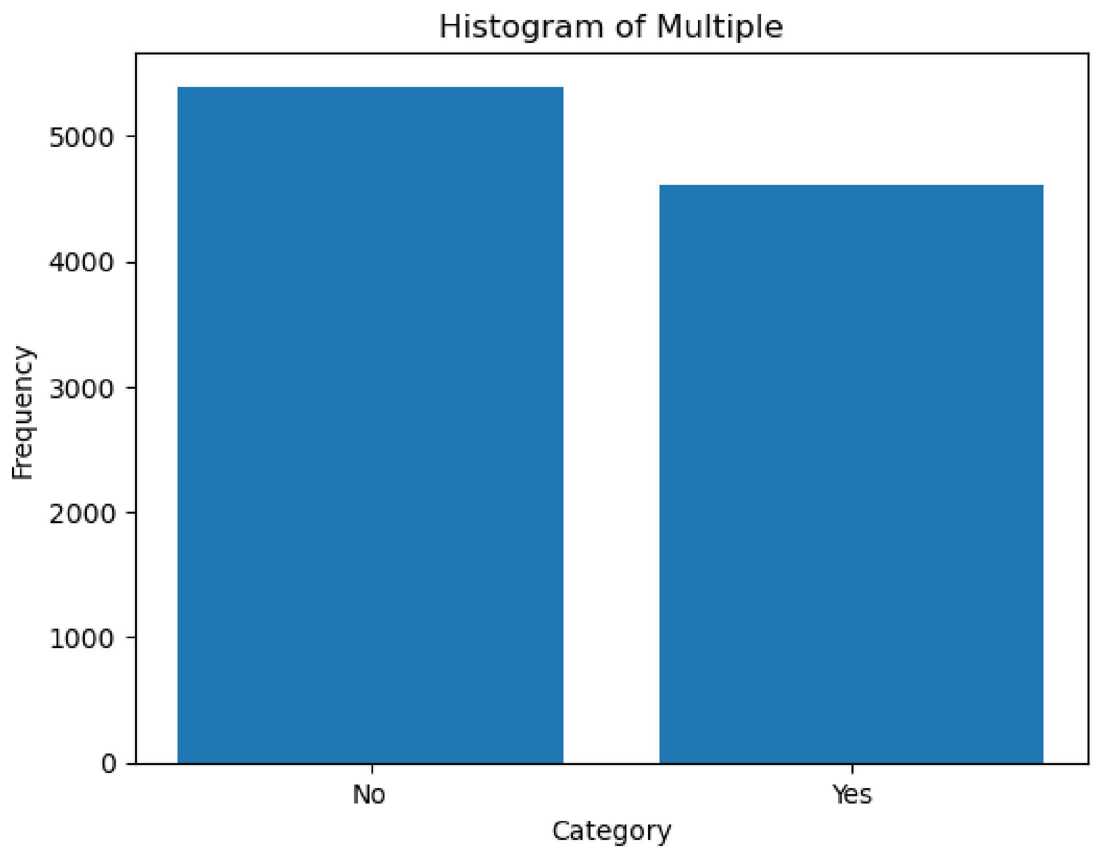

Histogram of Techie
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
No
Yes
Category


Histogram of Port_modem
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

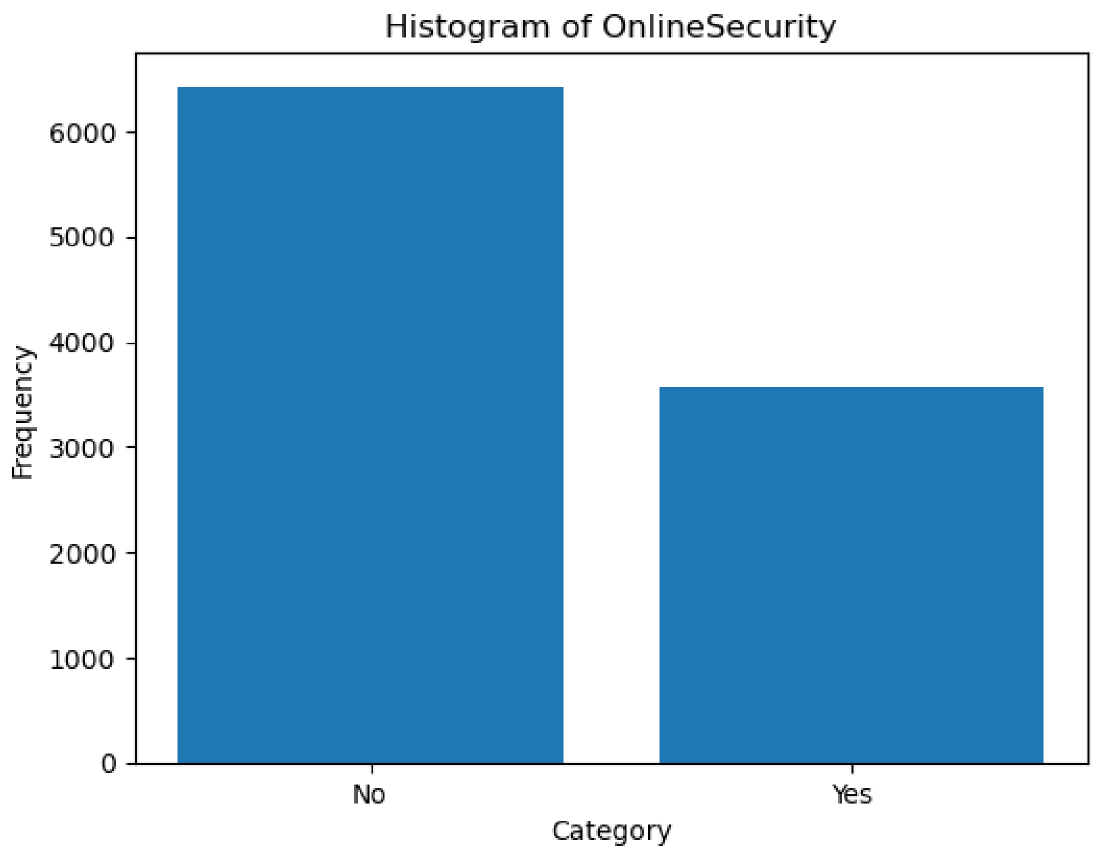

Histogram of Tablet
7000
6000
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

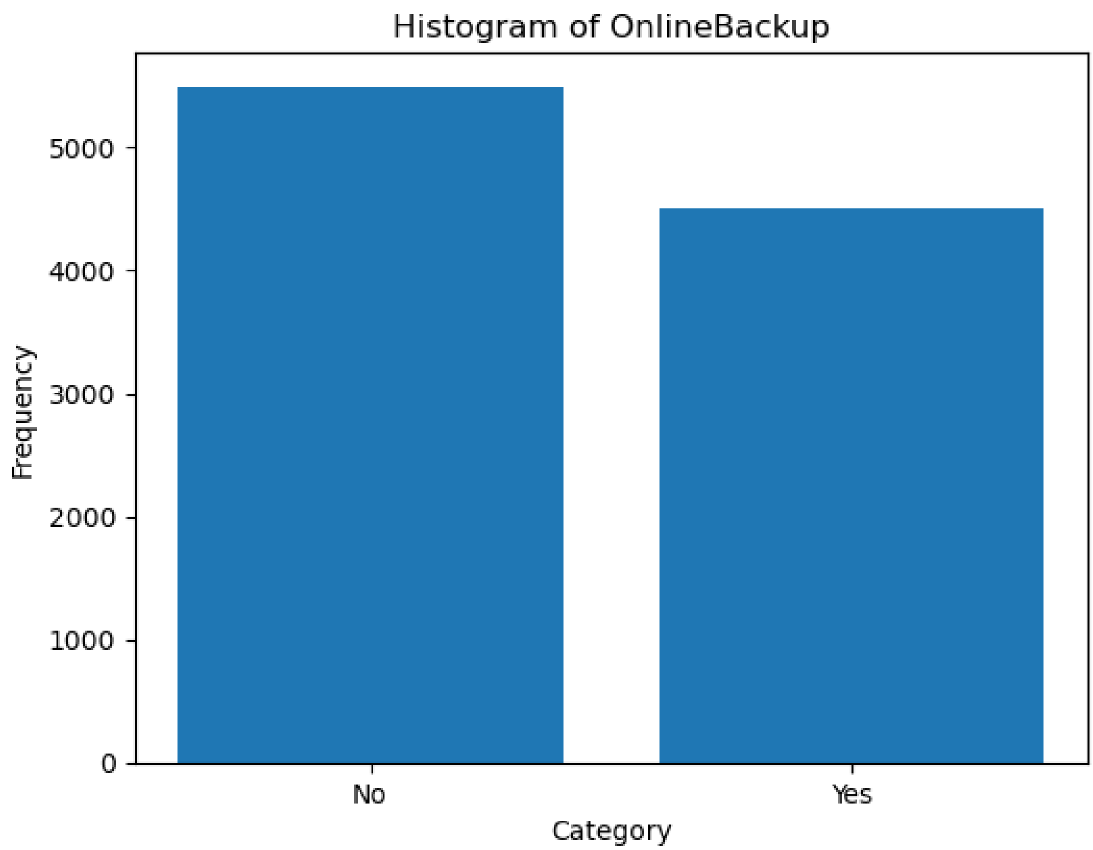


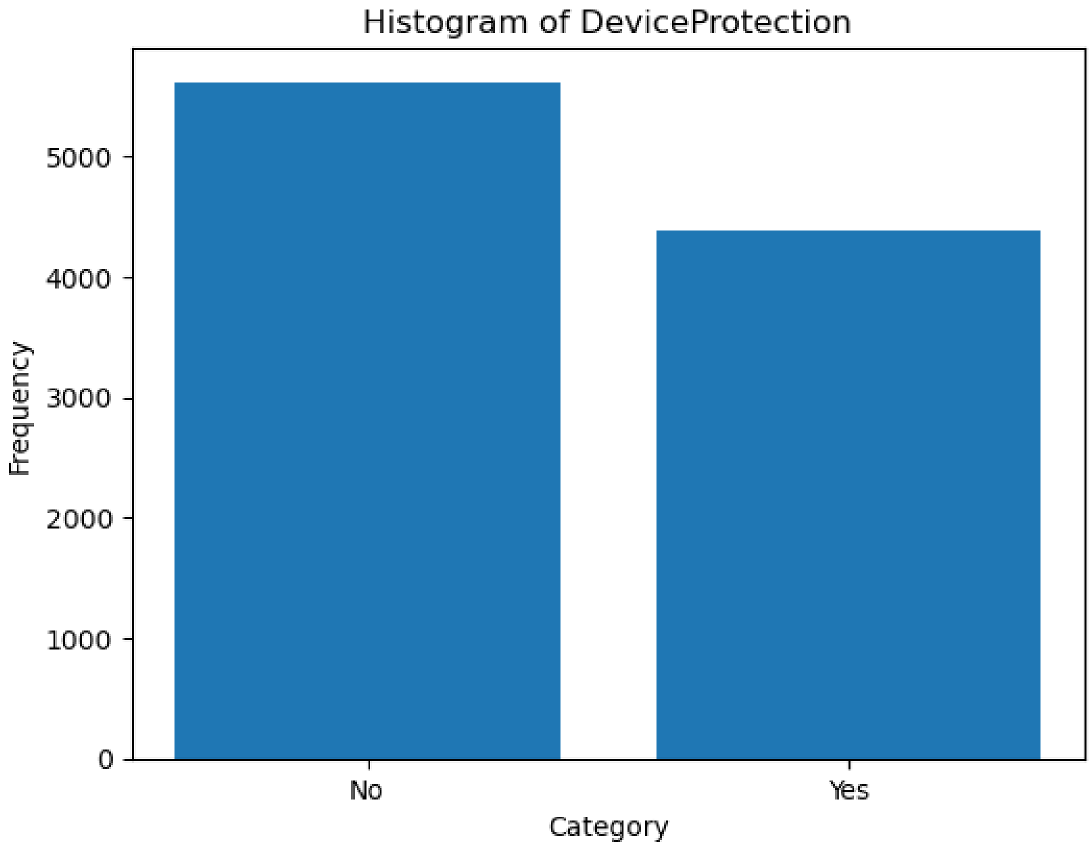
Histogram of Phone
8000
6000
4000
2000
0
Frequency
No
Yes
Category



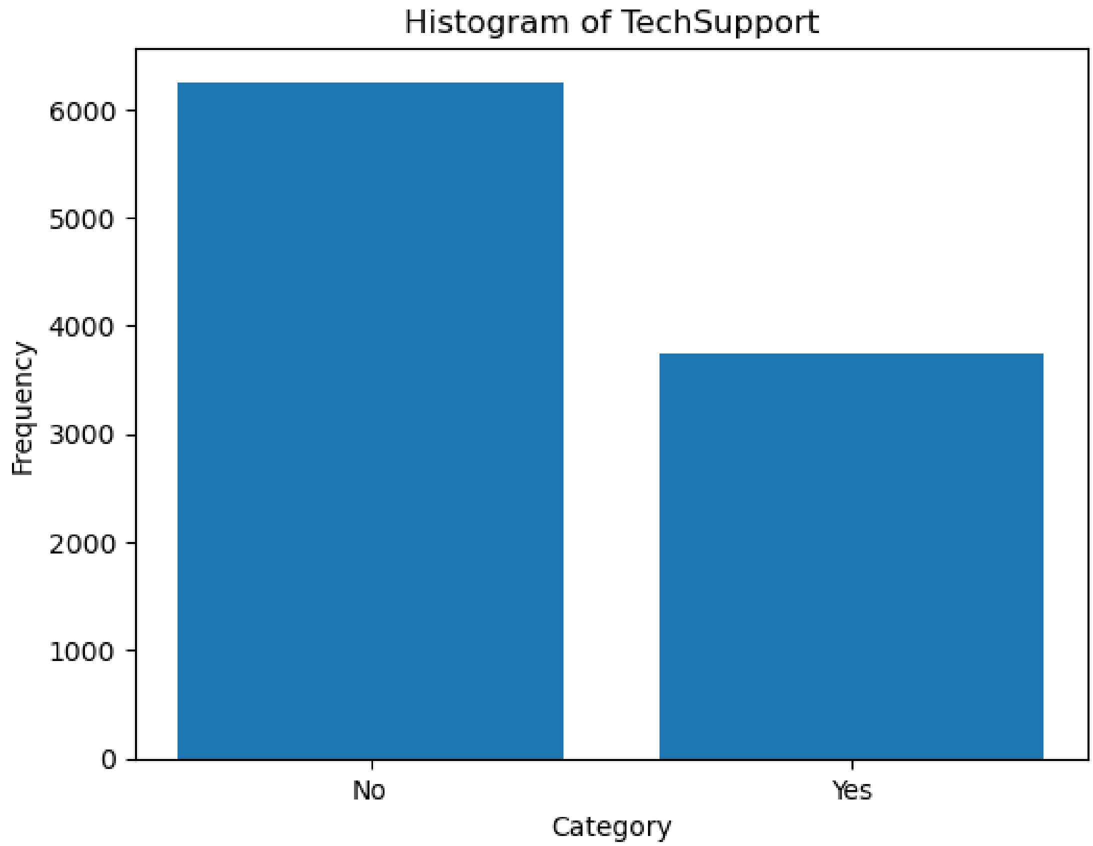
Histogram of Multiple
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

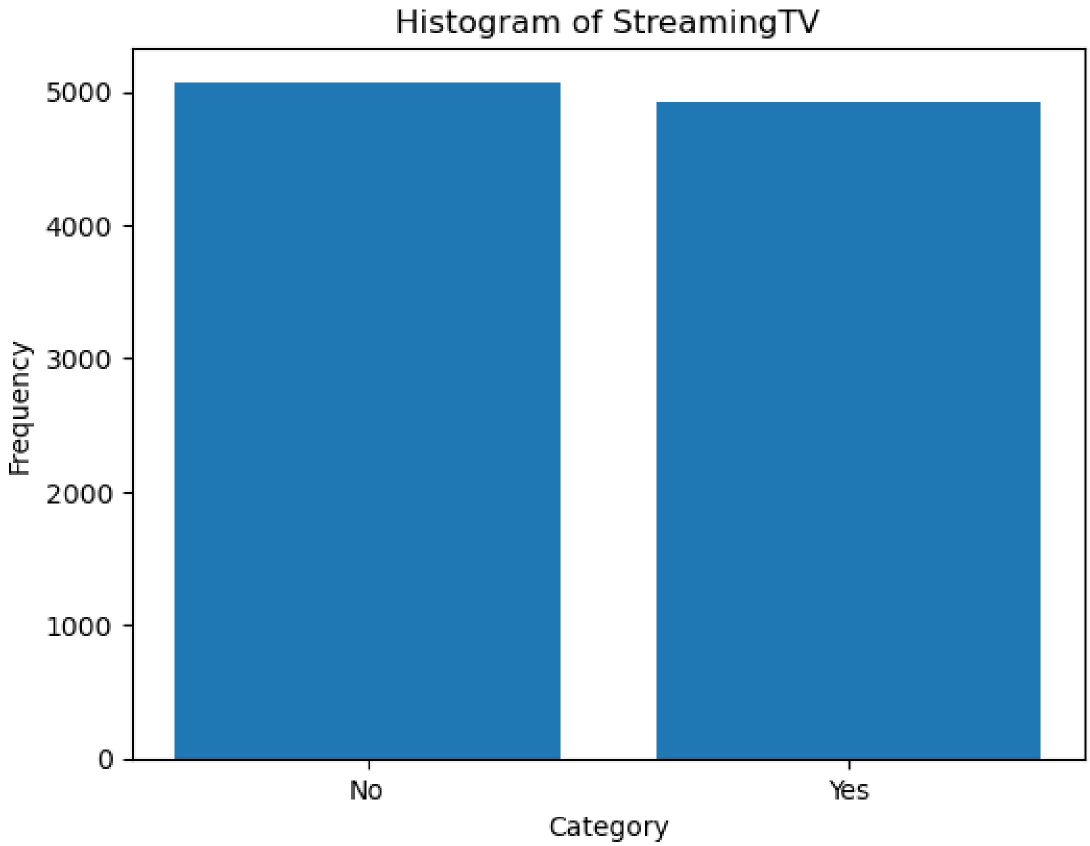

Histogram of OnlineSecurity
Frequency
0
1000
2000
3000
4000
5000
6000
No
Yes
Category

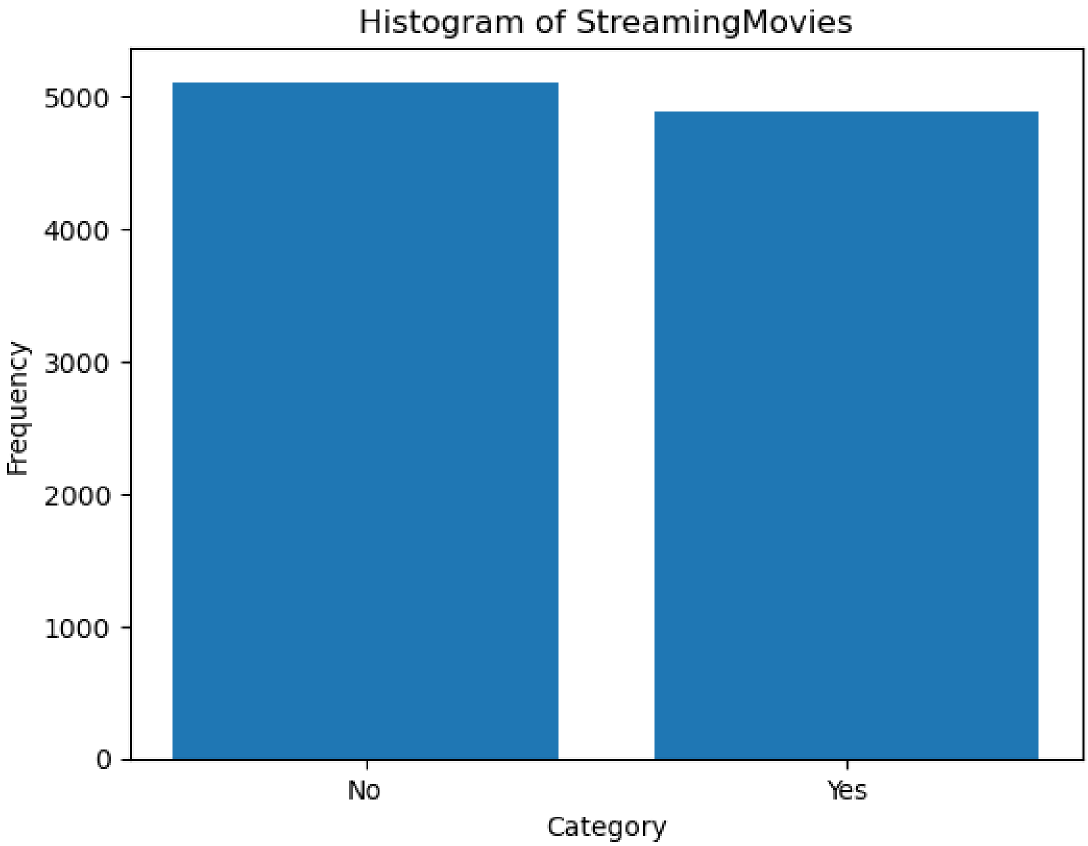

Histogram of OnlineBackup
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

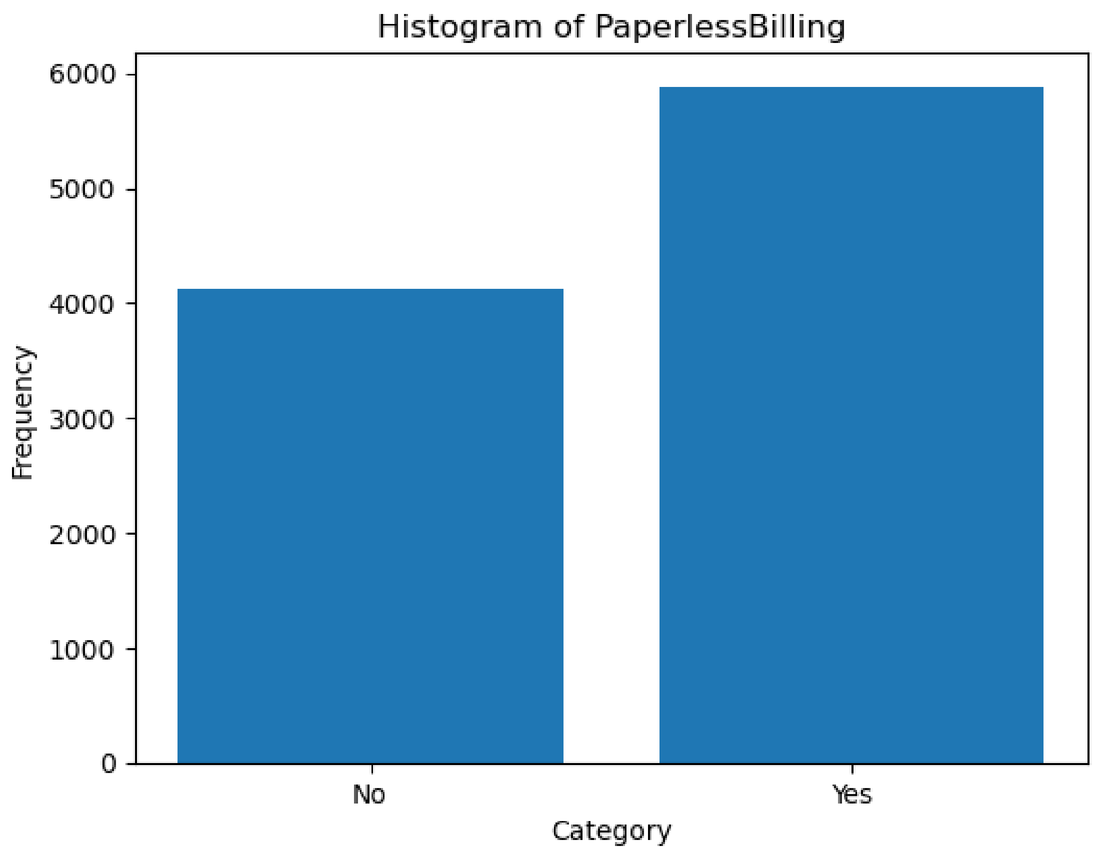


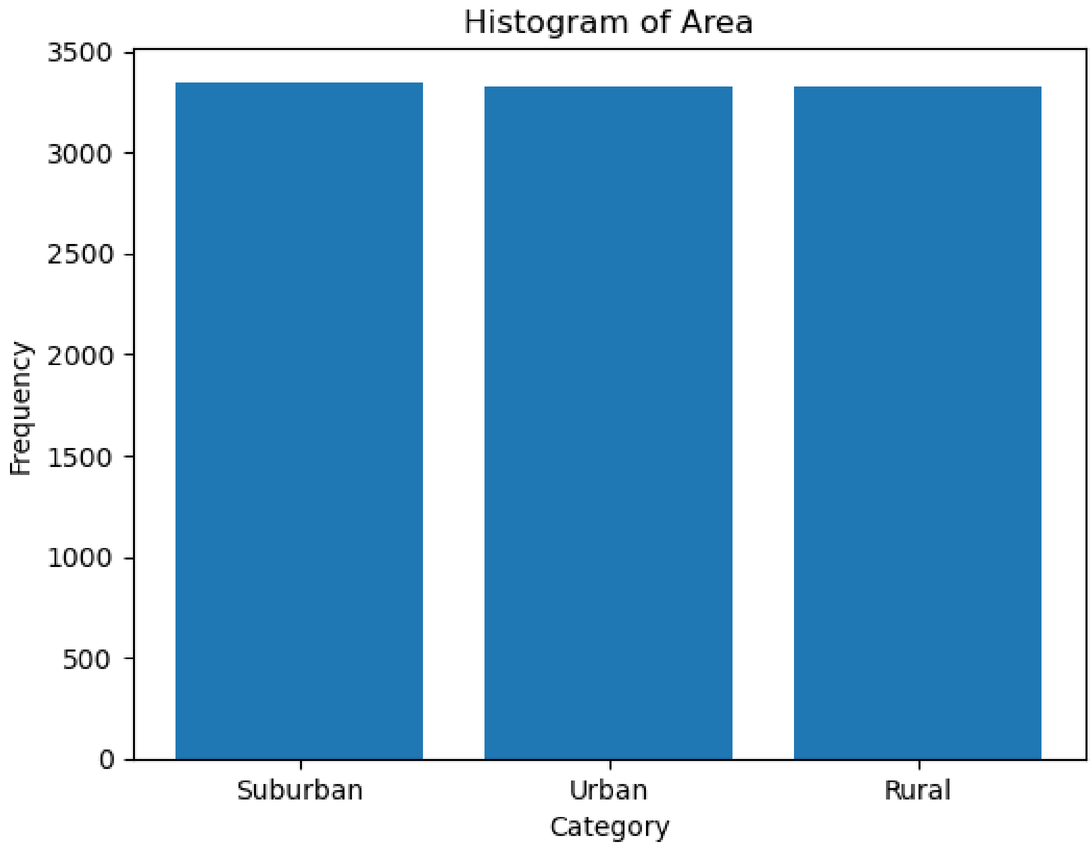
Histogram of DeviceProtection
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

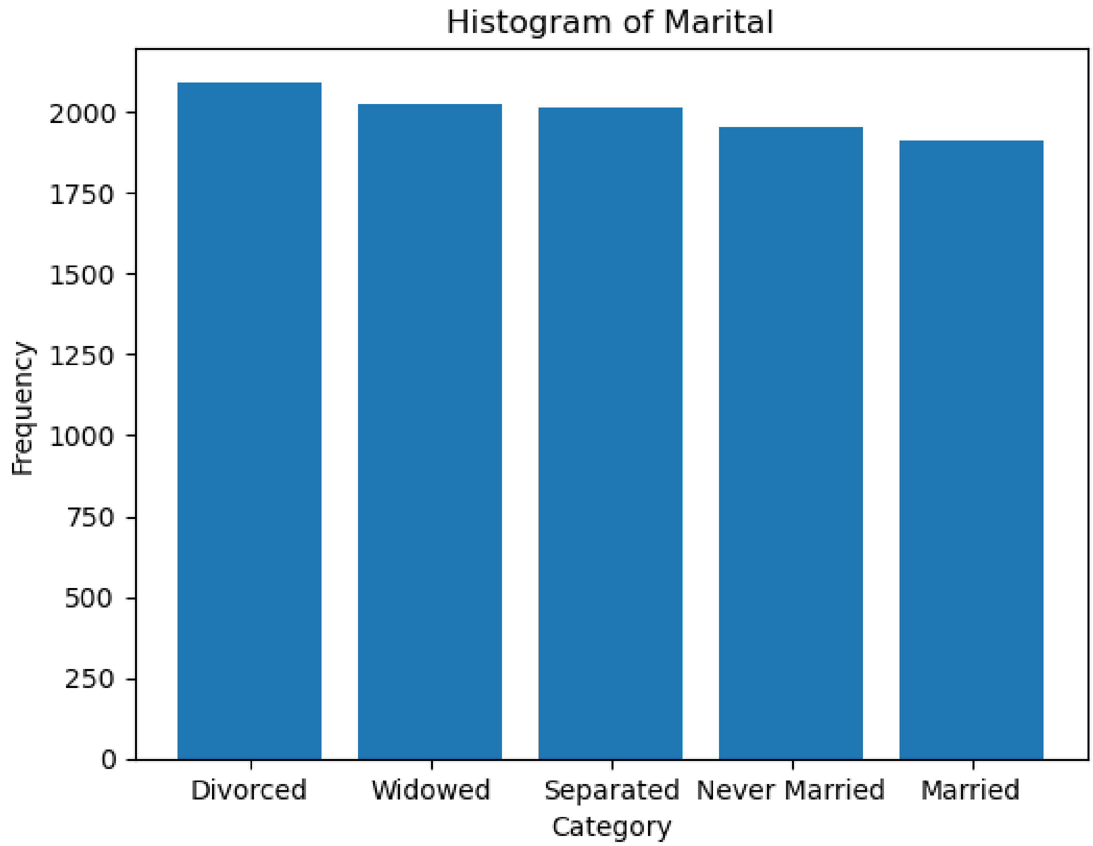

Histogram of TechSupport
6000
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

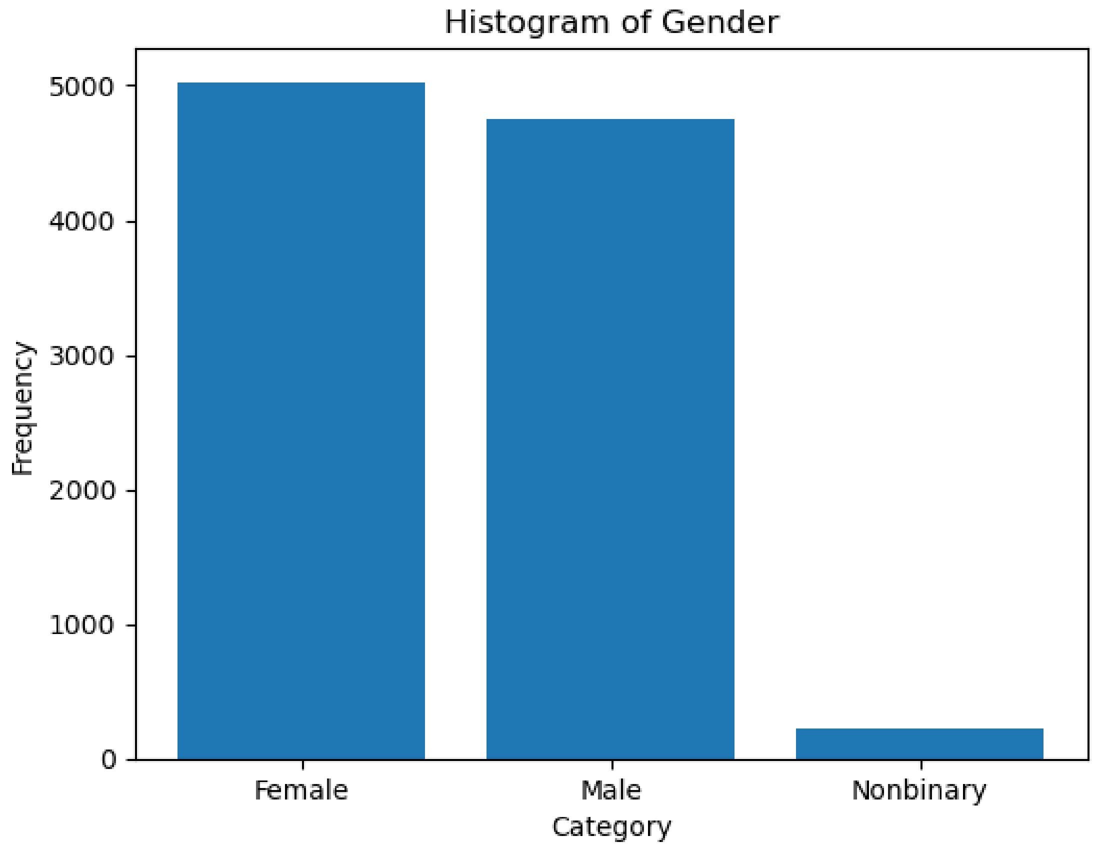

Histogram of StreamingTV
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

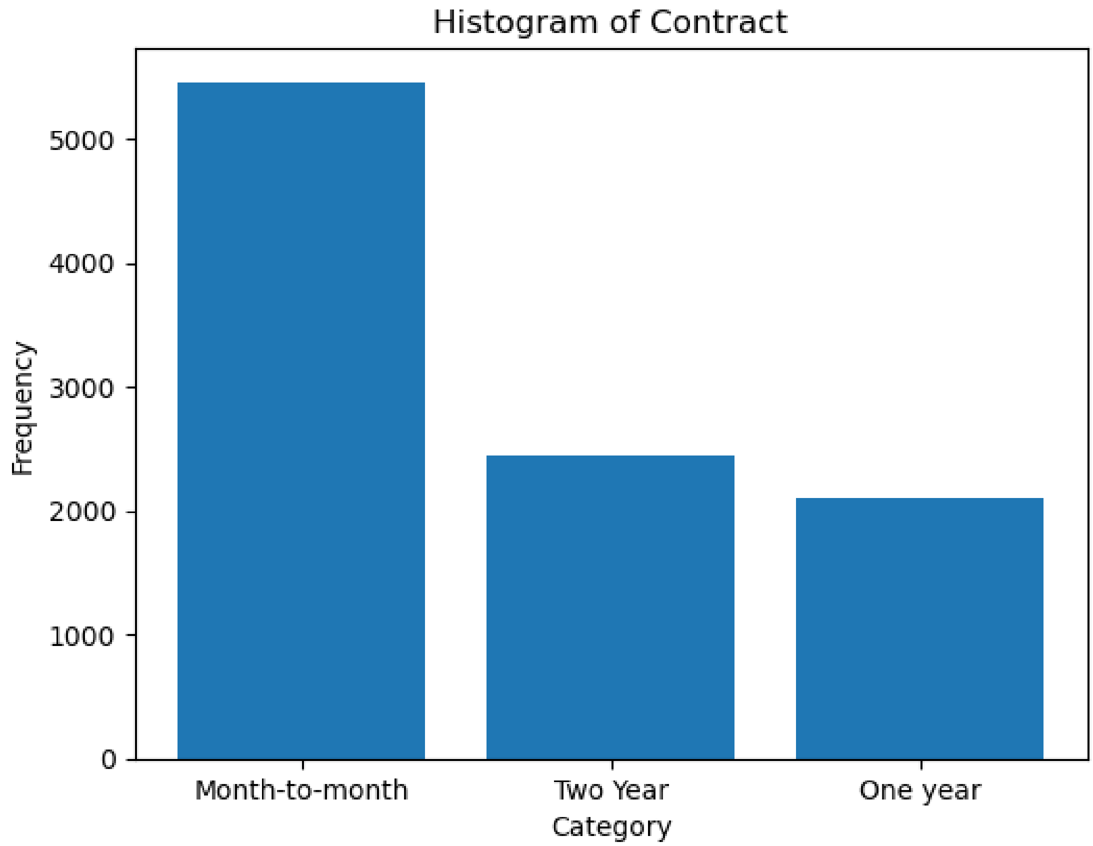

Histogram of StreamingMovies
Frequency
5000
4000
3000
2000
1000
0
No
Yes
Category


Histogram of PaperlessBilling
6000
5000
4000
3000
2000
1000
0
Frequency
No
Yes
Category

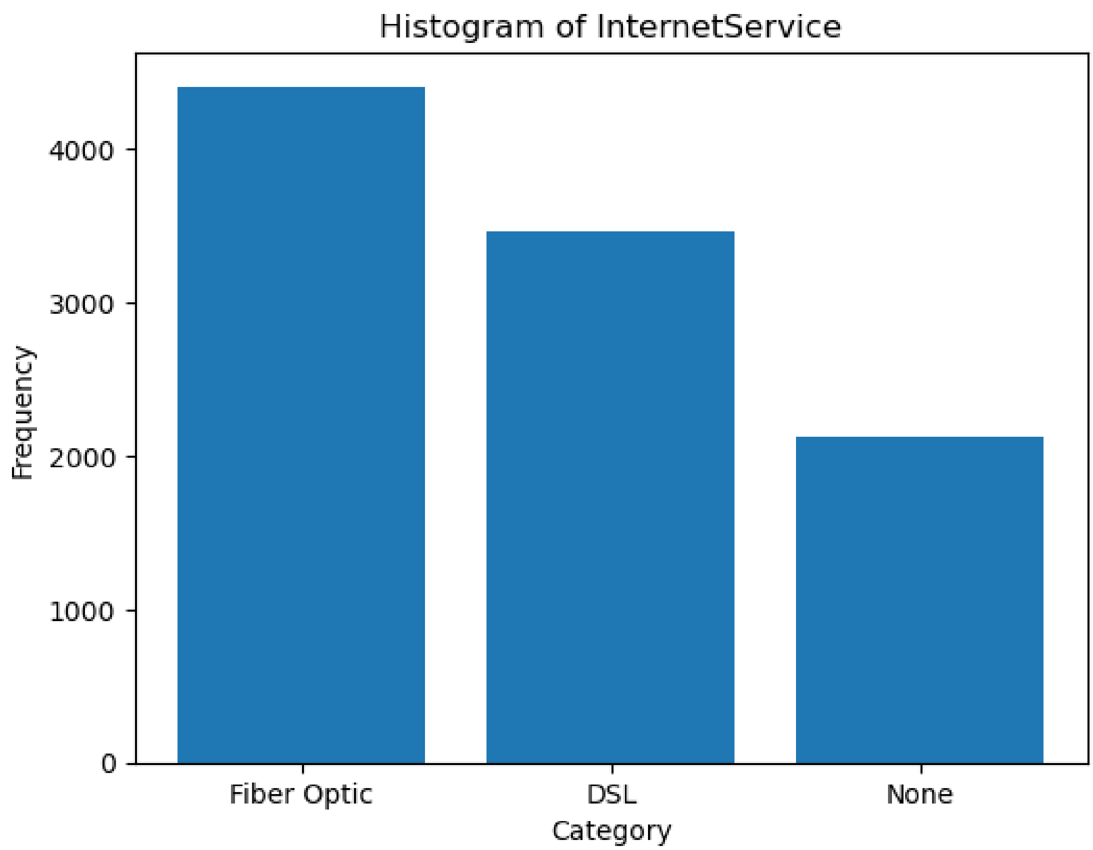

Histogram of Area
3500
3000
2500
2000
1500
1000
500
0
Frequency
Suburban
Urban
Rural
Category

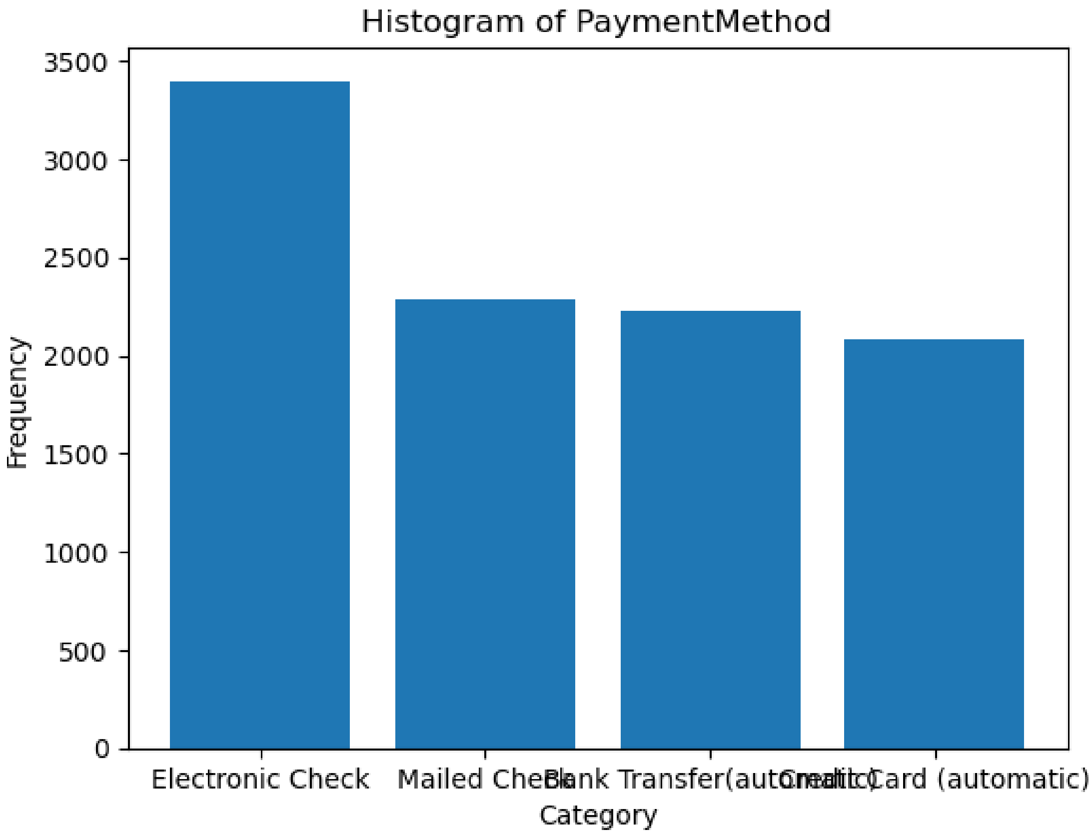

Histogram of Marital
Frequency
0
250
500
750
1000
1250
1500
1750
2000
Divorced
Widowed
Separated
Never Married
Married
Category

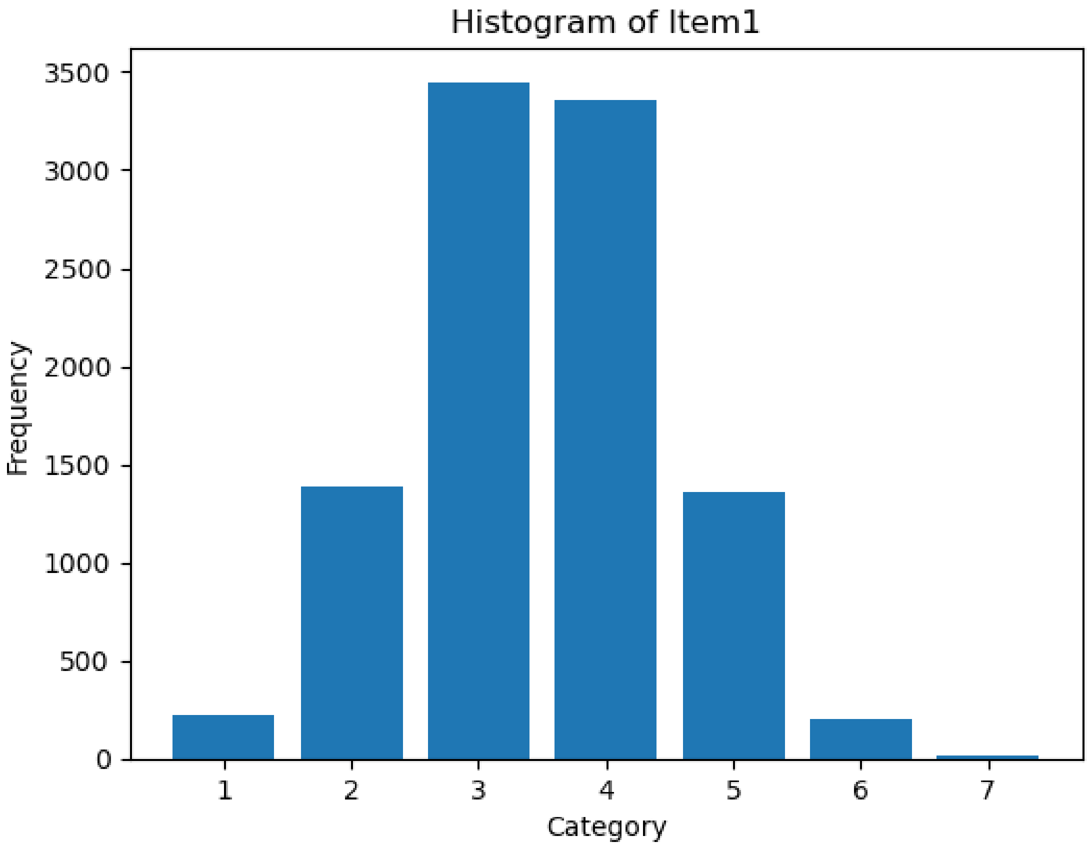

Histogram of Gender
Frequency
0
1000
2000
3000
4000
5000
Female
Male
Nonbinary
Category

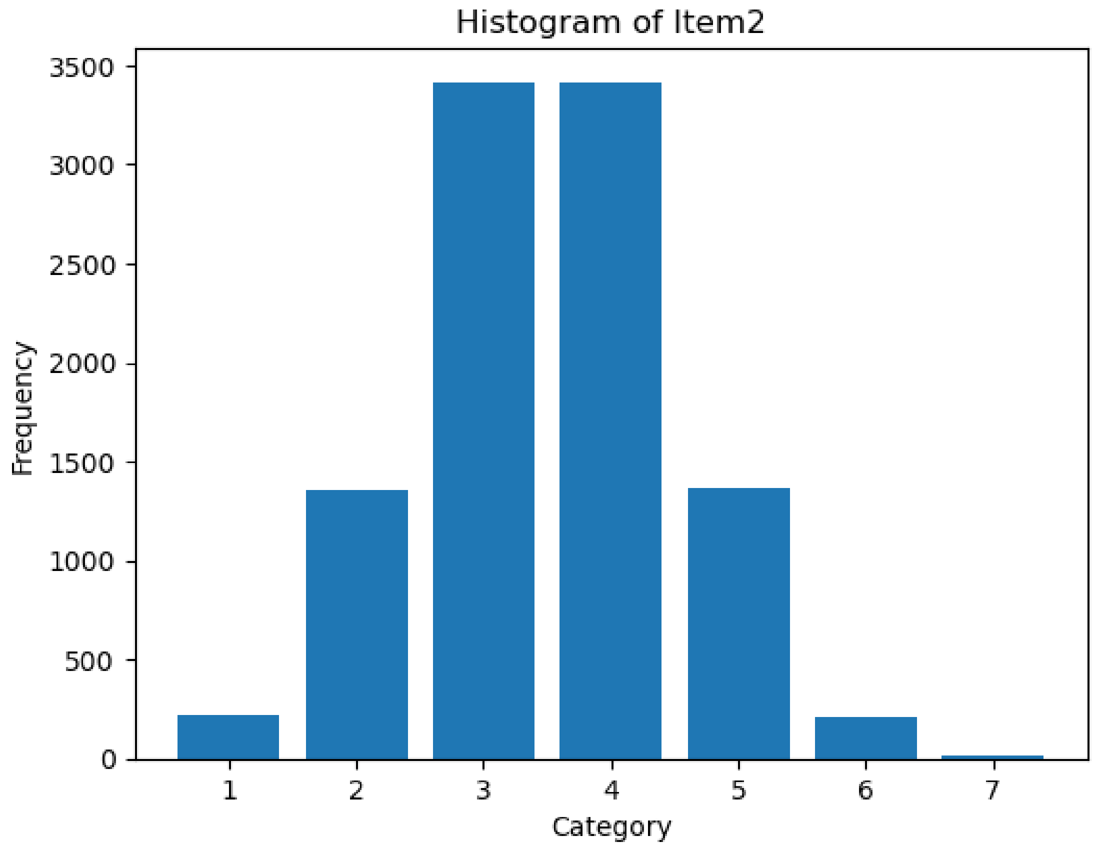


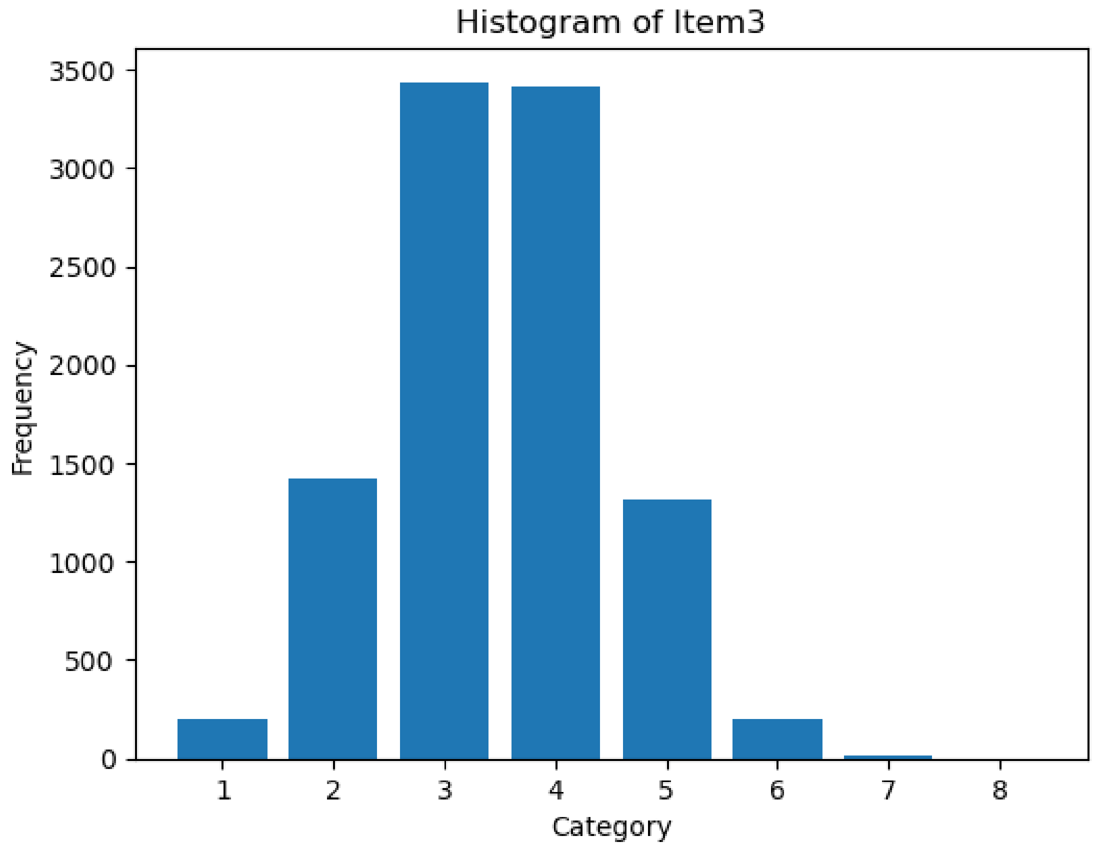
Histogram of Contract
Frequency
0
1000
2000
3000
4000
5000
Month-to-month
Two Year
One year
Category

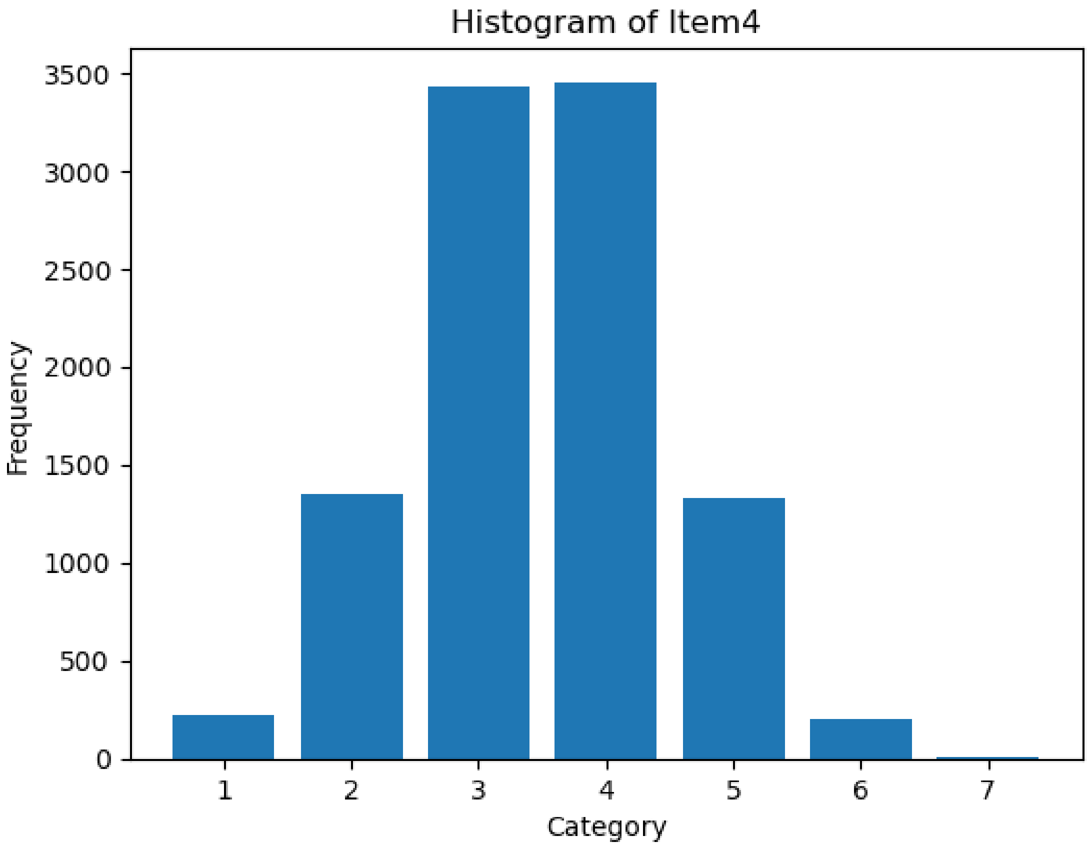

Histogram of InternetService
Frequency
4000
3000
2000
1000
0
Fiber Optic
DSL
None
Category

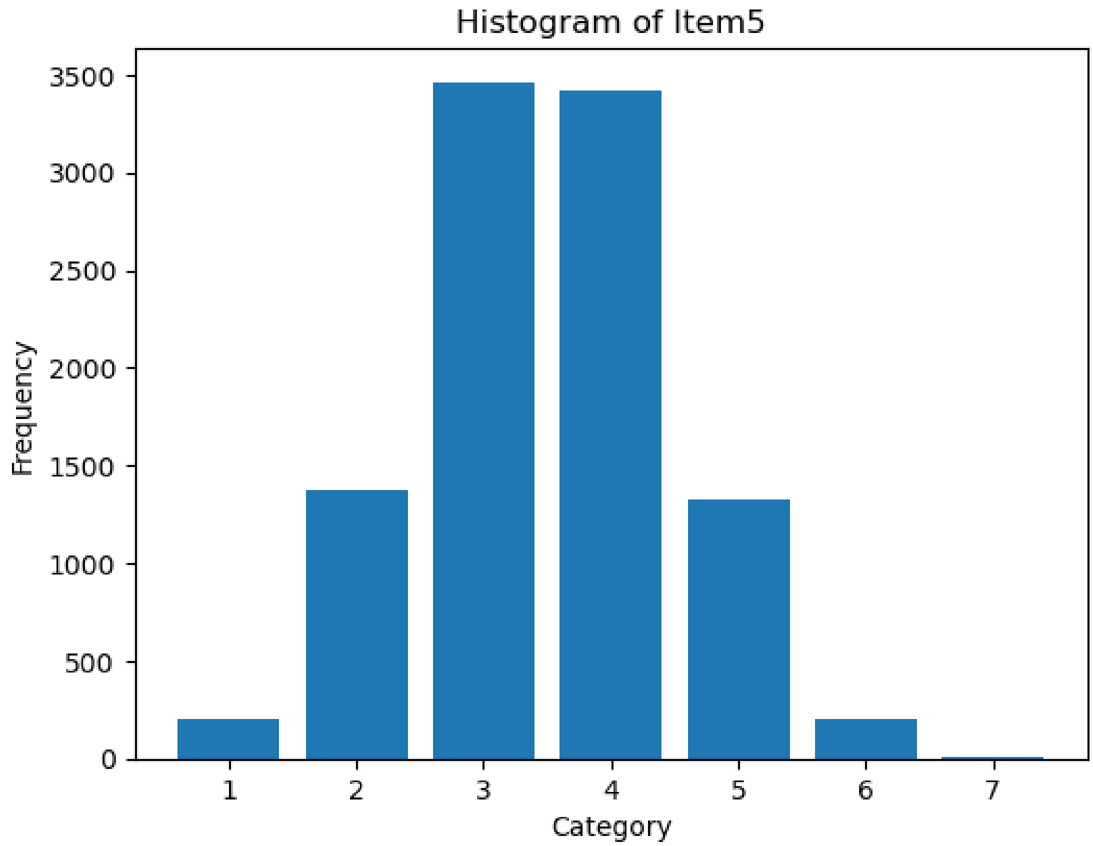

Histogram of PaymentMethod
Frequency
3500
3000
2500
2000
1500
1000
500
0
Electronic Check
Mailed Check
Bank Transfer(automatic)
Credit Card (automatic)
Category


Histogram of Item1
3500
3000
2500
2000
1500
1000
500
0
Frequency
1
2
3
4
5
6
7
Category

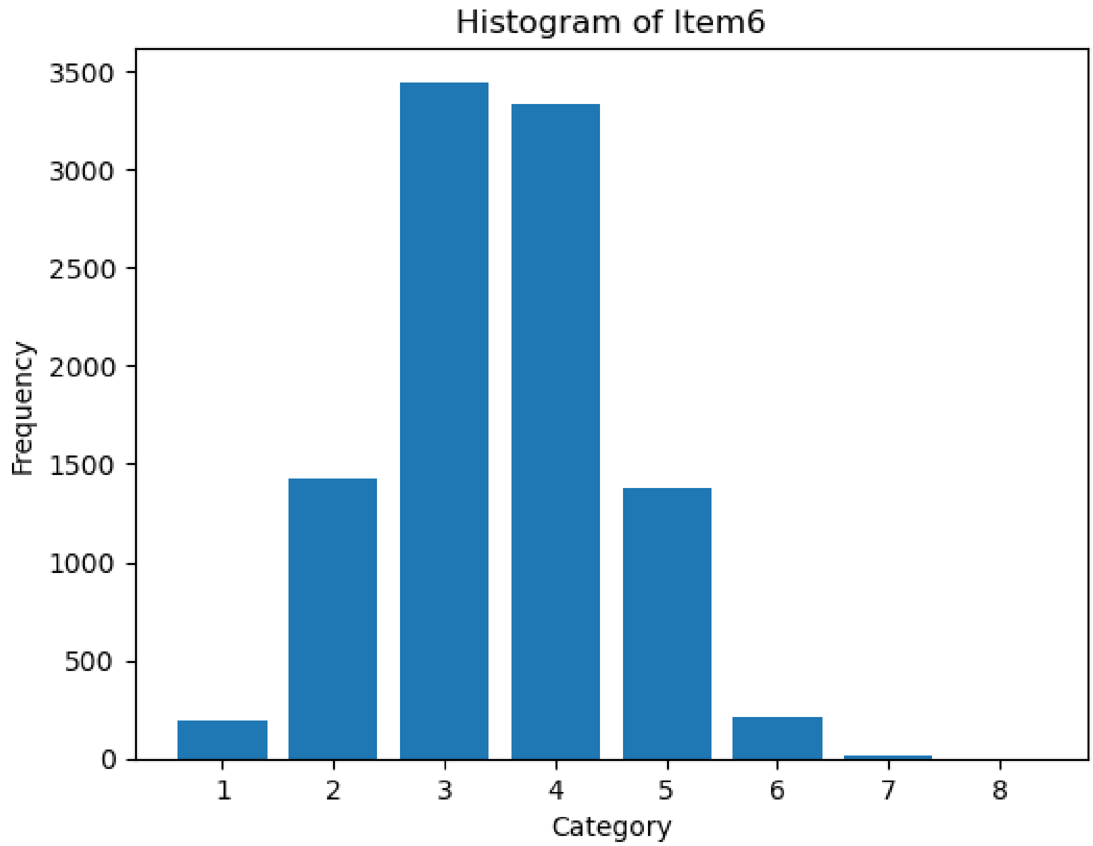

Histogram of Item2
3500
3000
2500
2000
1500
1000
500
0
Frequency
1
2
3
4
5
6
7
Category

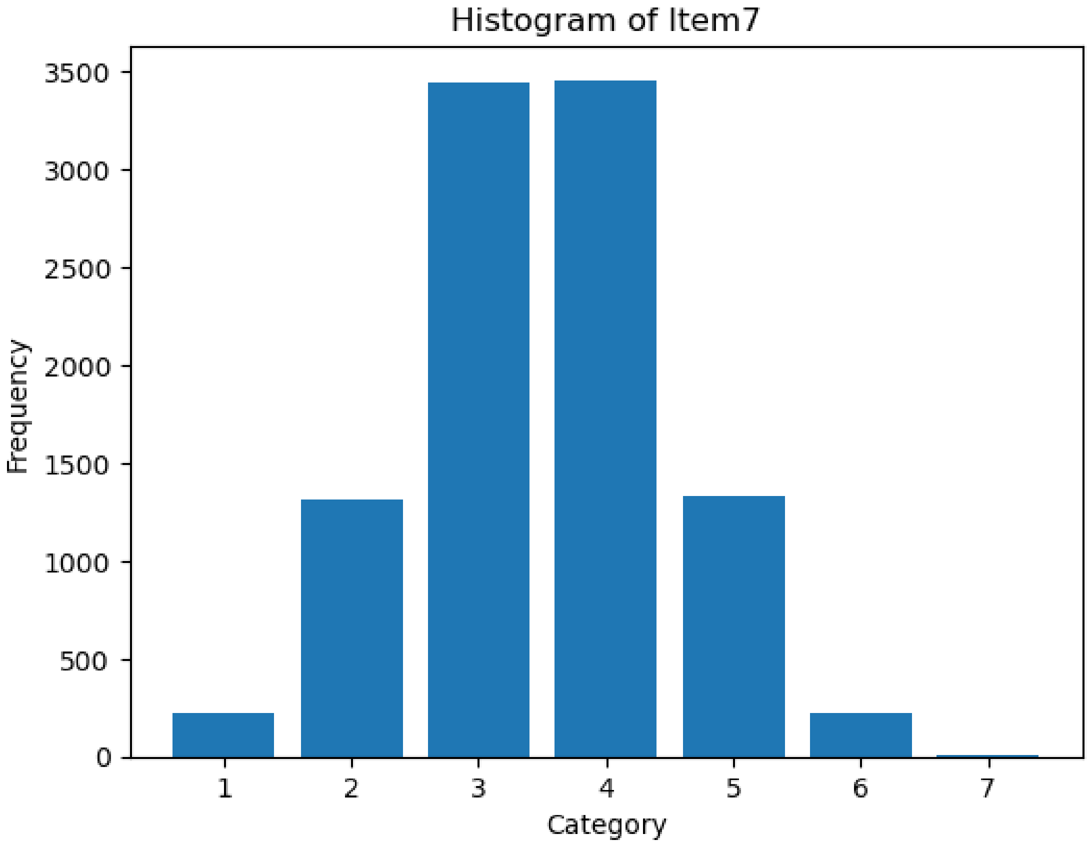

Histogram of Item3
Frequency
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
Category

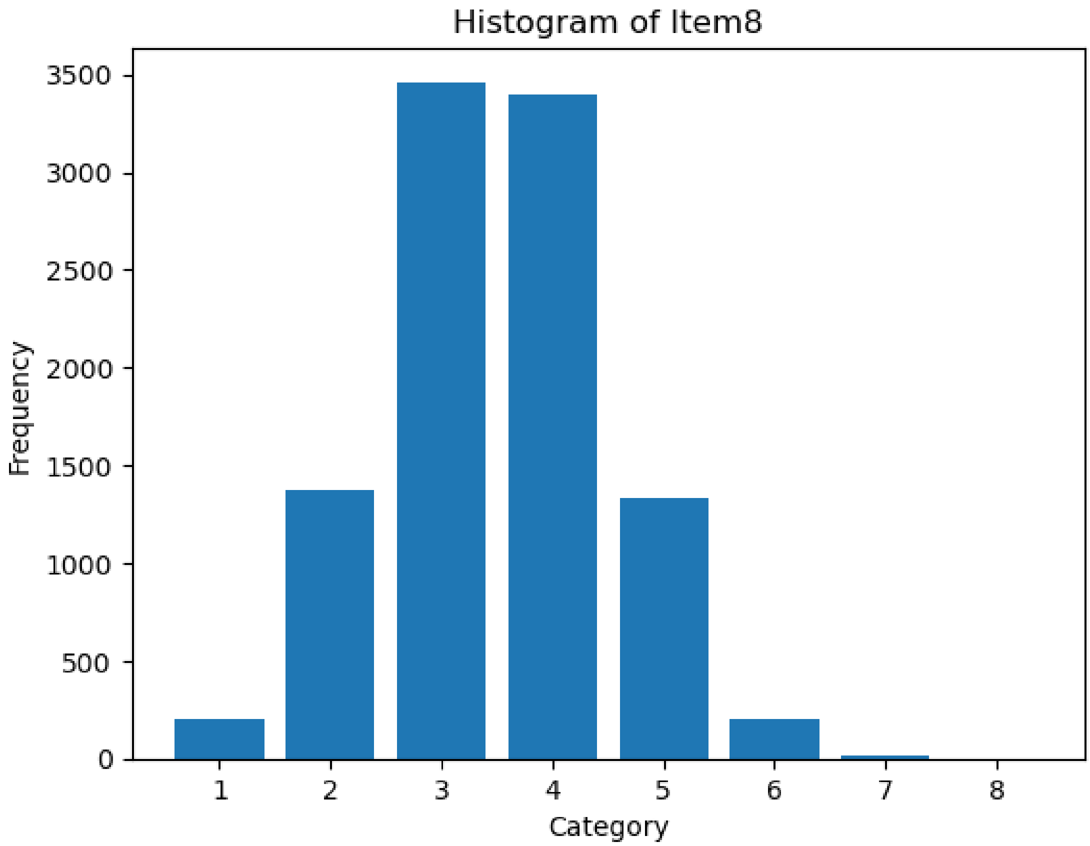

Histogram of Item4
Frequency
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
Category



Histogram of Item5
Frequency
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
Category


Histogram of Item6
Frequency
Category
0
500
1000
1500
2000
2500
3000
3500
1
2
3
4
5
6
7
8

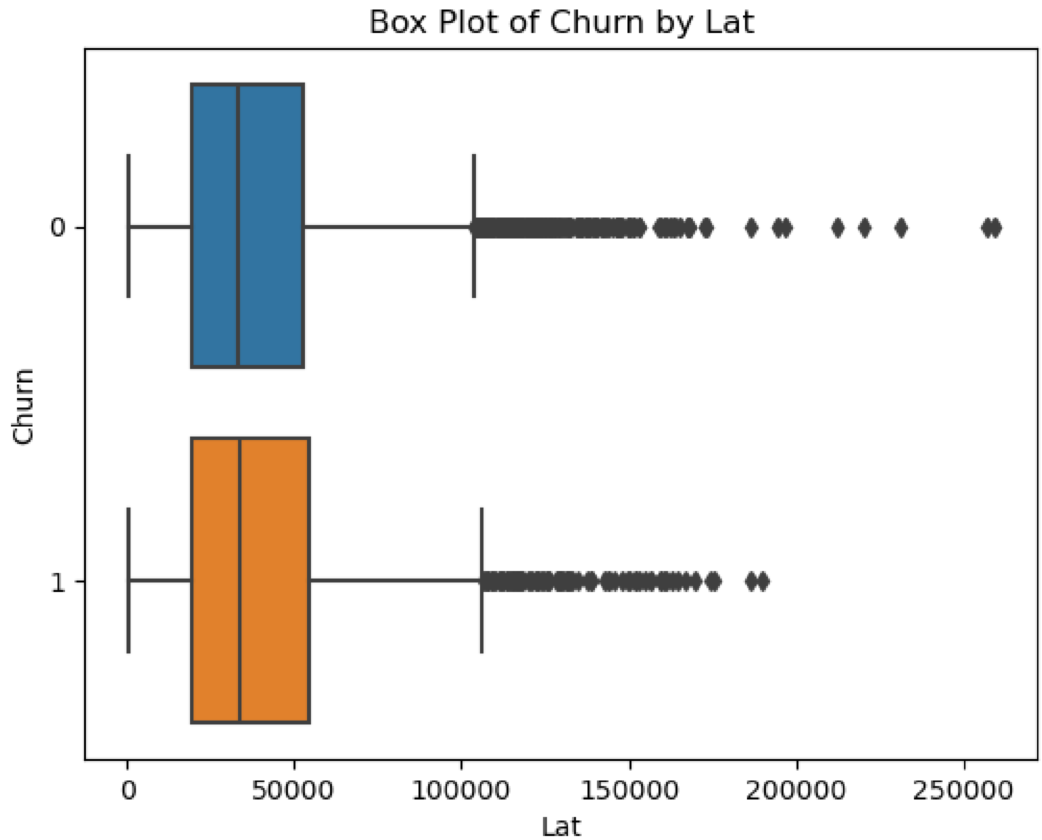

Histogram of Item7
3500
3000
2500
2000
1500
1000
500
0
Frequency
1
2
3
4
5
6
7
Category

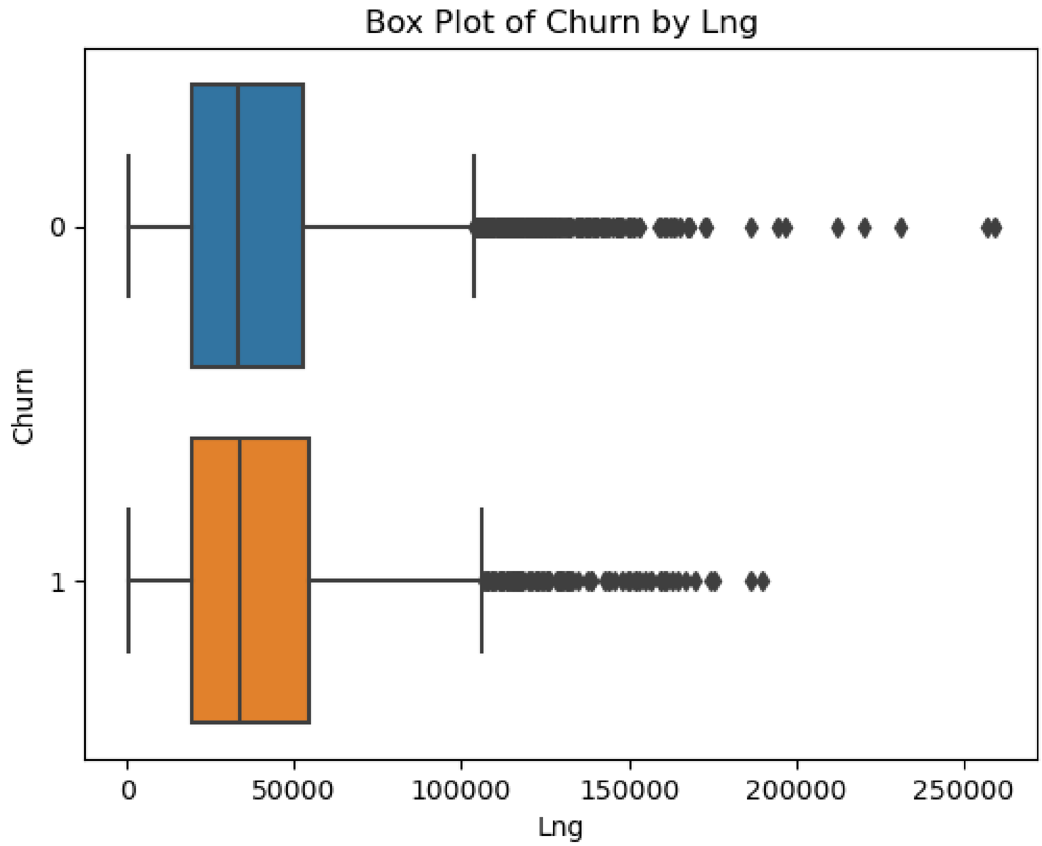


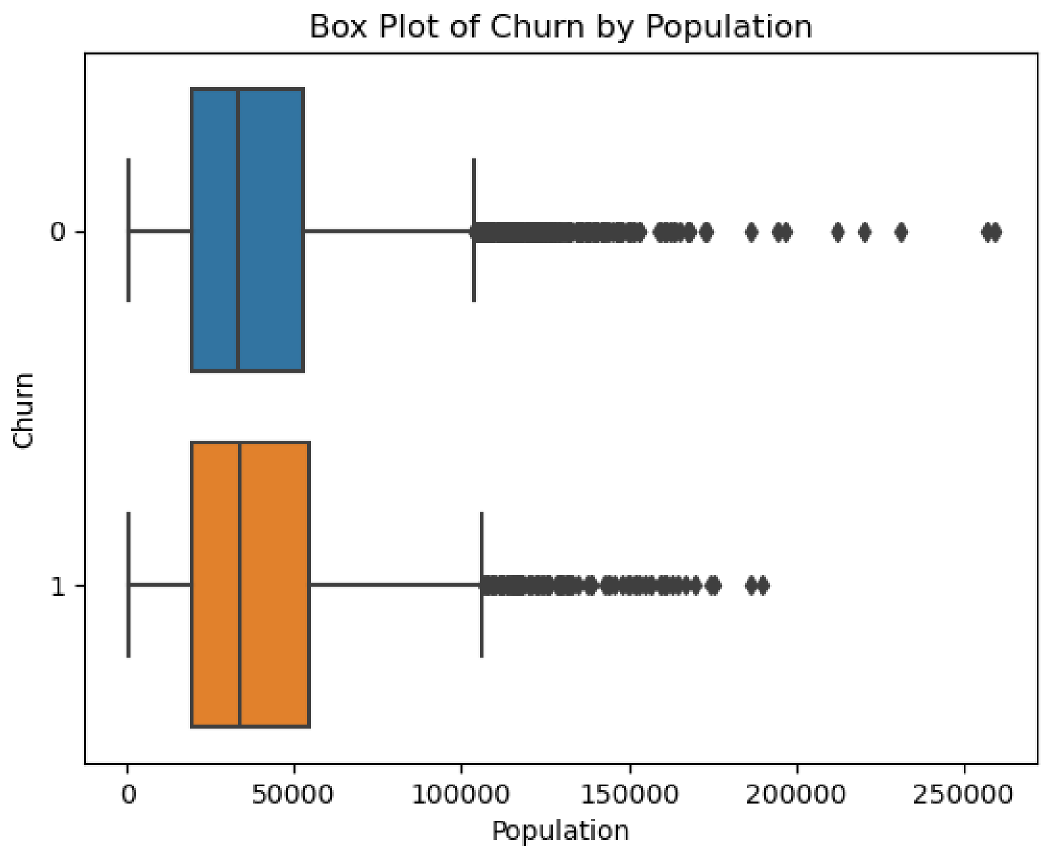
[18]:
```
## C3 Bivariate Visualizations

import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category = FutureWarning, module = "seaborn")

def plot_bivarcatH(data_frame, cont_col, cat_col):
# Plot bivariate boxplots of categorical vs continuous variables
    sns.boxplot(x = df['Income'], y = df['Churn'])
    plt.title(f'Box Plot of {cat_col} by {cont_col}')
    plt.xlabel(cont_col)
    plt.ylabel(cat_col)
    plt.show()

def plot_bivarBars(data_frame, exp, dep):
# Plot bivariate stacked bar plots of two categorical varibles
    counts = pd.crosstab(data_frame[exp], data_frame[dep])
    counts.plot(kind = 'bar', stacked = True)

    plt.title(f'Stacked Bar Charts of `{dep}` by `{exp}`')
```

```
    plt.xticks(rotation = 0)
    plt.xlabel(exp)
    plt.ylabel(dep)
    plt.show()

def plot_stackedBars(data_frame, exp, dep):
# Stacked bar charts of categorical vs ordinal
    sns.countplot(data = data_frame, x = exp, hue = dep)
    plt.title(f'Grouped Bar Charts of `{dep}` by `{exp}`')
    plt.xticks(rotation = 0)
    plt.xlabel(exp)
    plt.ylabel(dep)
    plt.show()
```

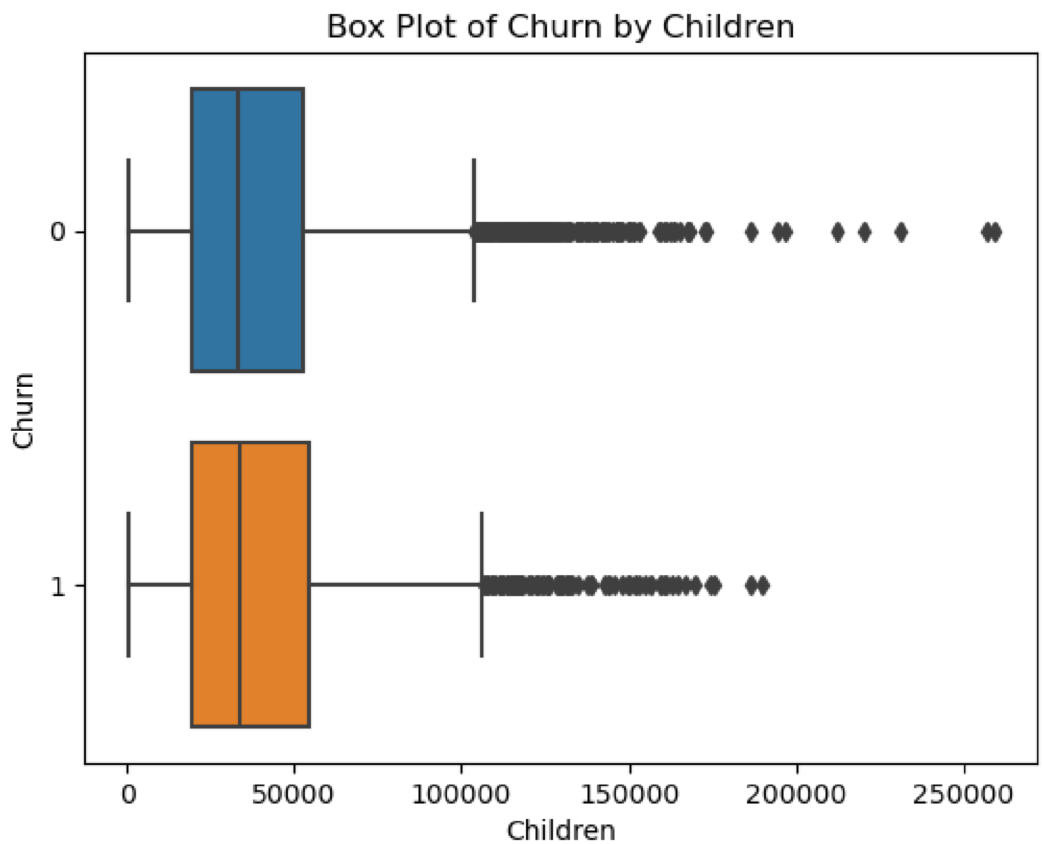
[19]:
```
## C3 Bivariate Visualizations

# Dependent Variable vs 13 Numeric Variables
for col in numericVars:
    plot_bivarcatH(df, col, 'Churn')

# Dependent Variable vs 12 Categorical Variables
for col in binaryVars:
    plot_bivarBars(df, col, 'Churn')

# Dependent Variable vs 6 Categorical Variables
for col in catVars:
    plot_bivarBars(df, col, 'Churn')

# Dependent Variable vs 8 Ordinal Variables
for col in ordinalVars:
    plot_stackedBars(df, col, 'ChurnStr')
```

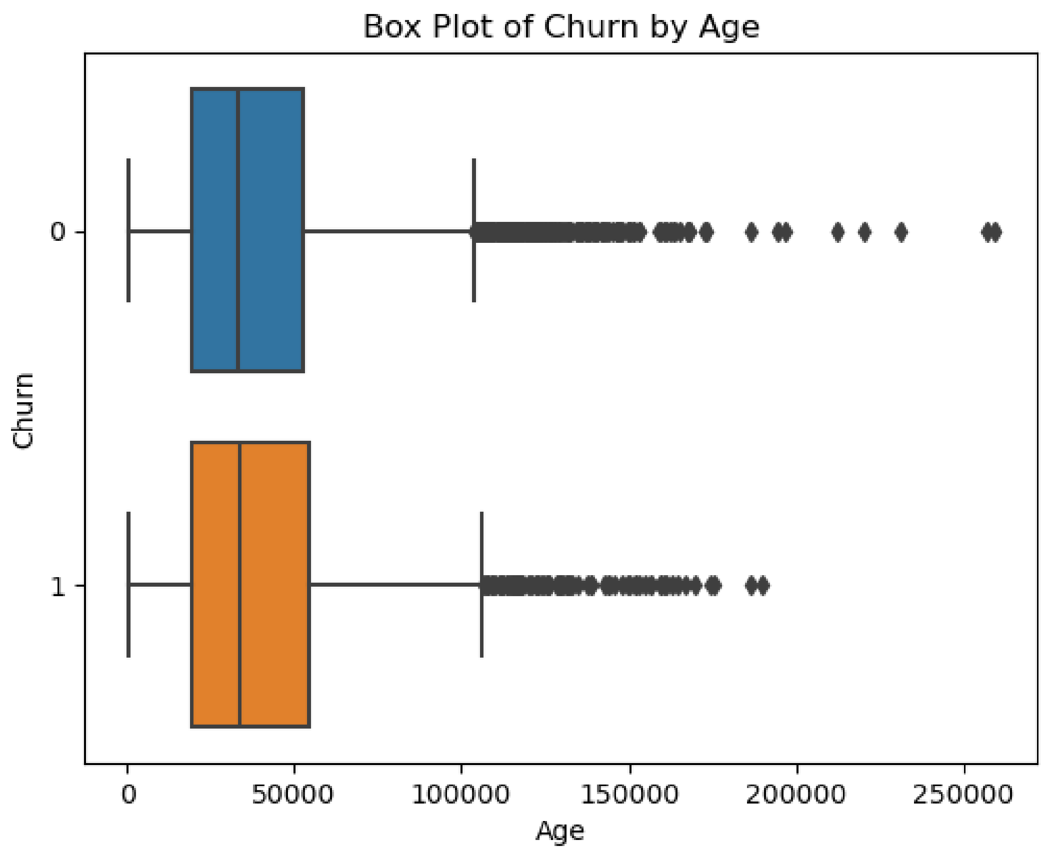

Box Plot of Churn by Lat
Churn
0
1
0
50000
100000
150000
200000
250000
Lat

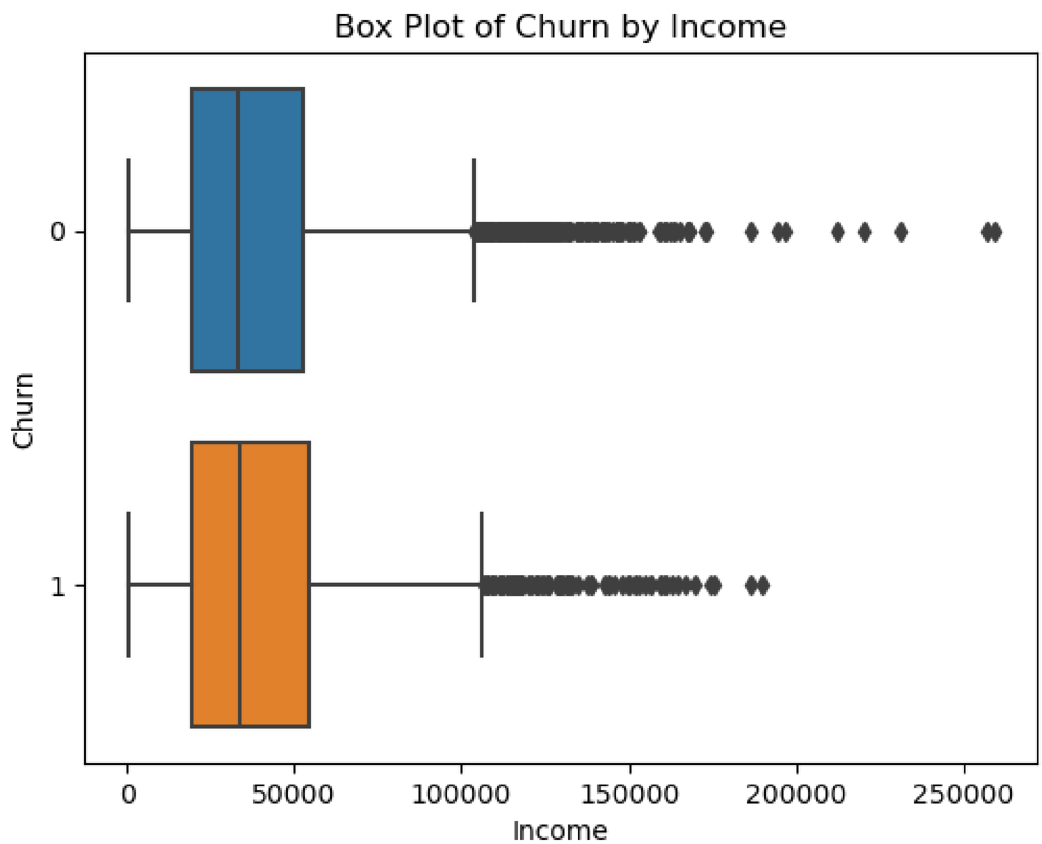

Box Plot of Churn by Lng
Churn
0
1
0
50000
100000
150000
200000
250000
Lng

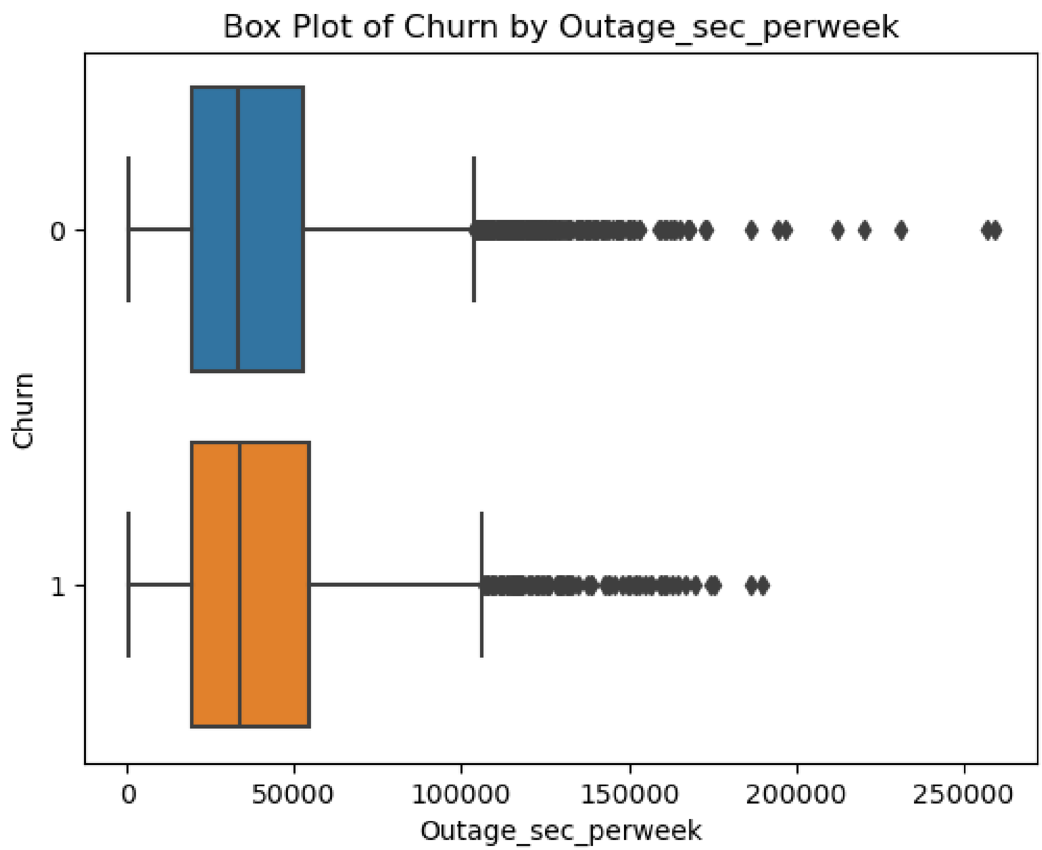


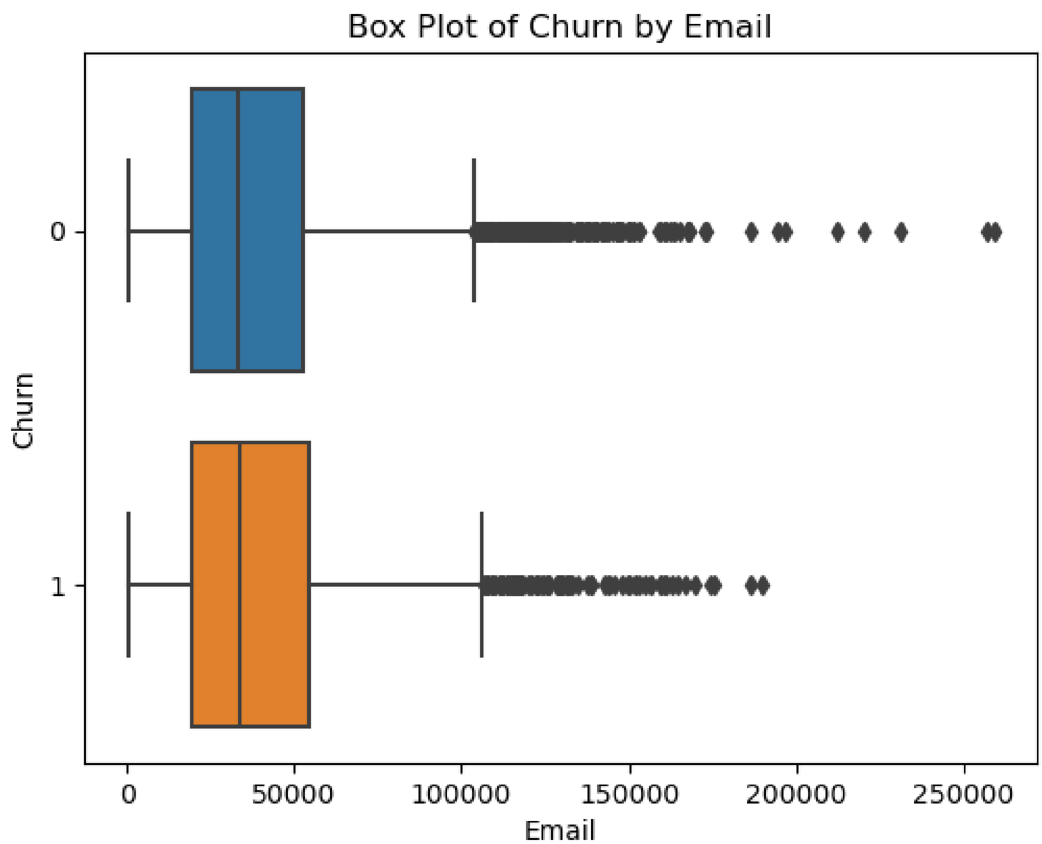
Box Plot of Churn by Population
Churn
0
1
0
50000
100000
150000
200000
250000
Population


Box Plot of Churn by Children
0
1
Churn
0
50000
100000
150000
200000
250000
Children

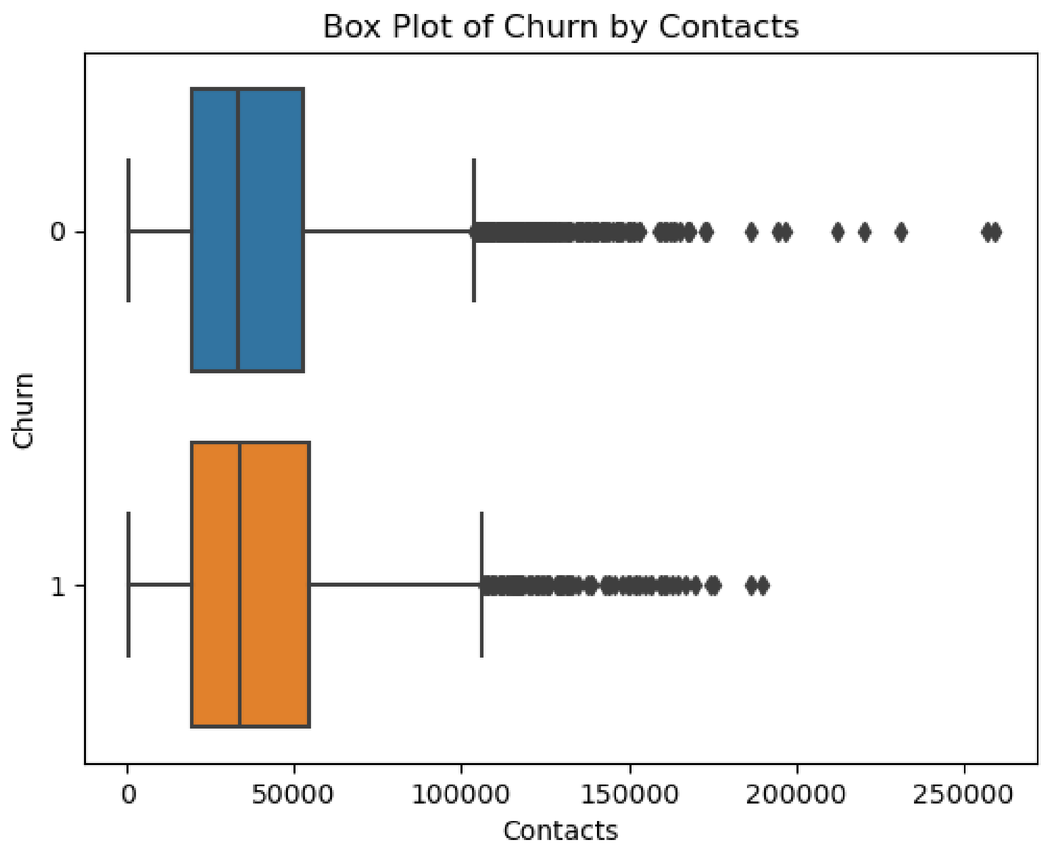

Box Plot of Churn by Age
Churn
0
1
0
50000
100000
150000
200000
250000
Age

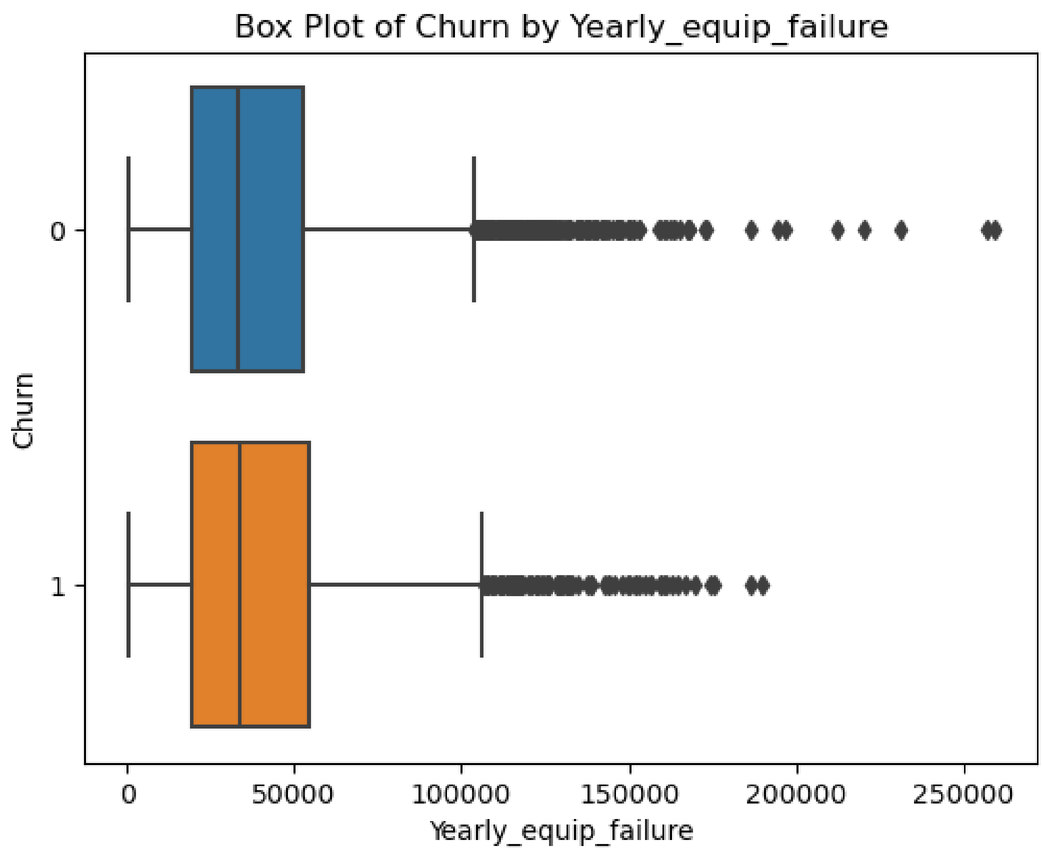

Box Plot of Churn by Income
Churn
0
1
0
50000
100000
150000
200000
250000
Income

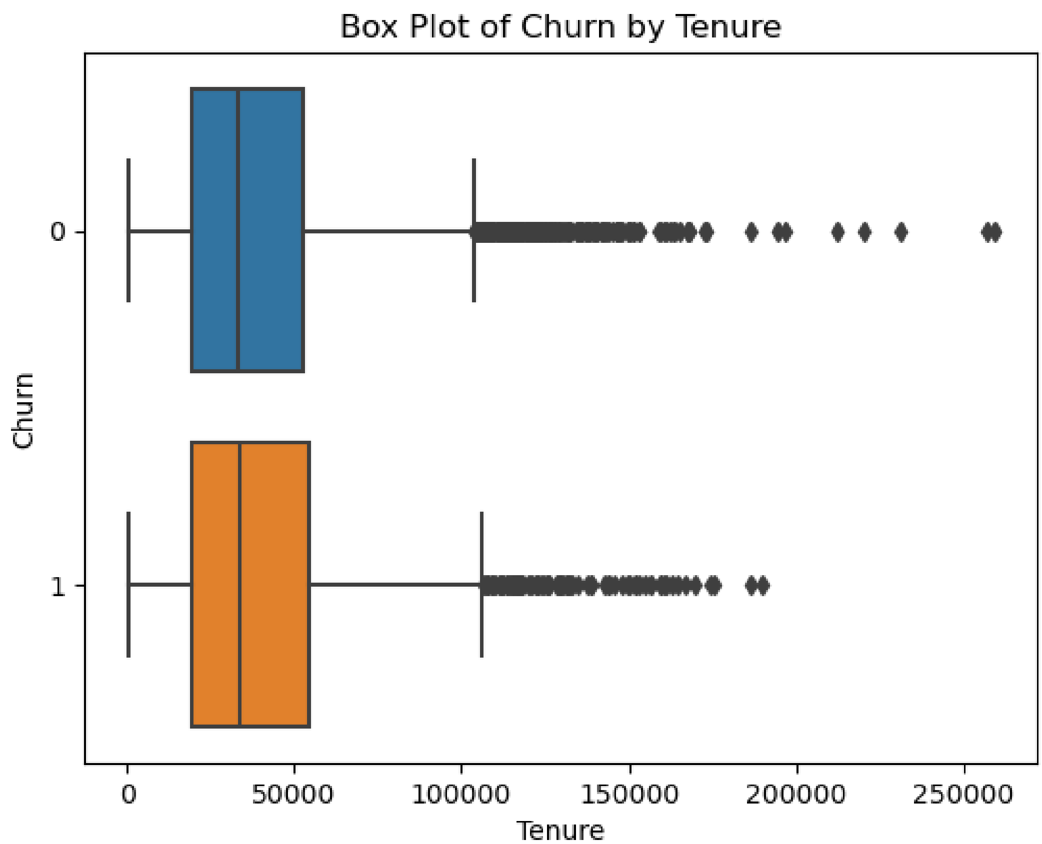

Box Plot of Churn by Outage_sec_perweek
Churn
0
1
0
50000
100000
150000
200000
250000
Outage_sec_perweek

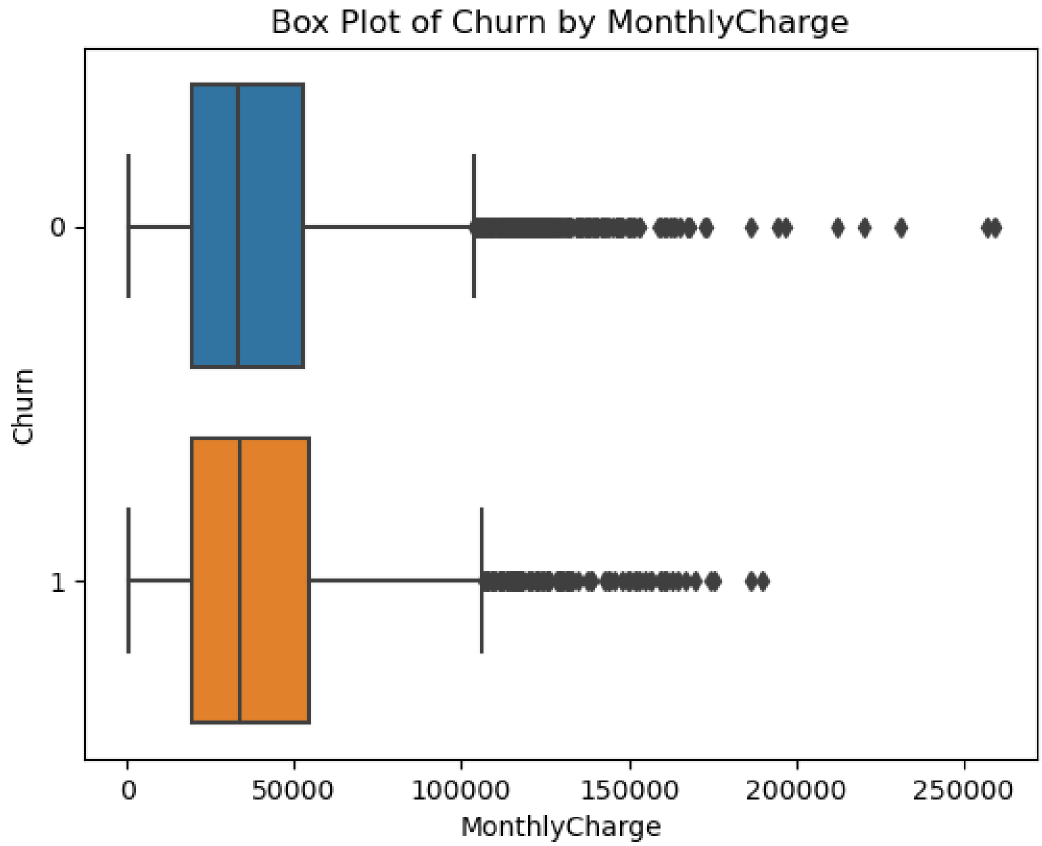


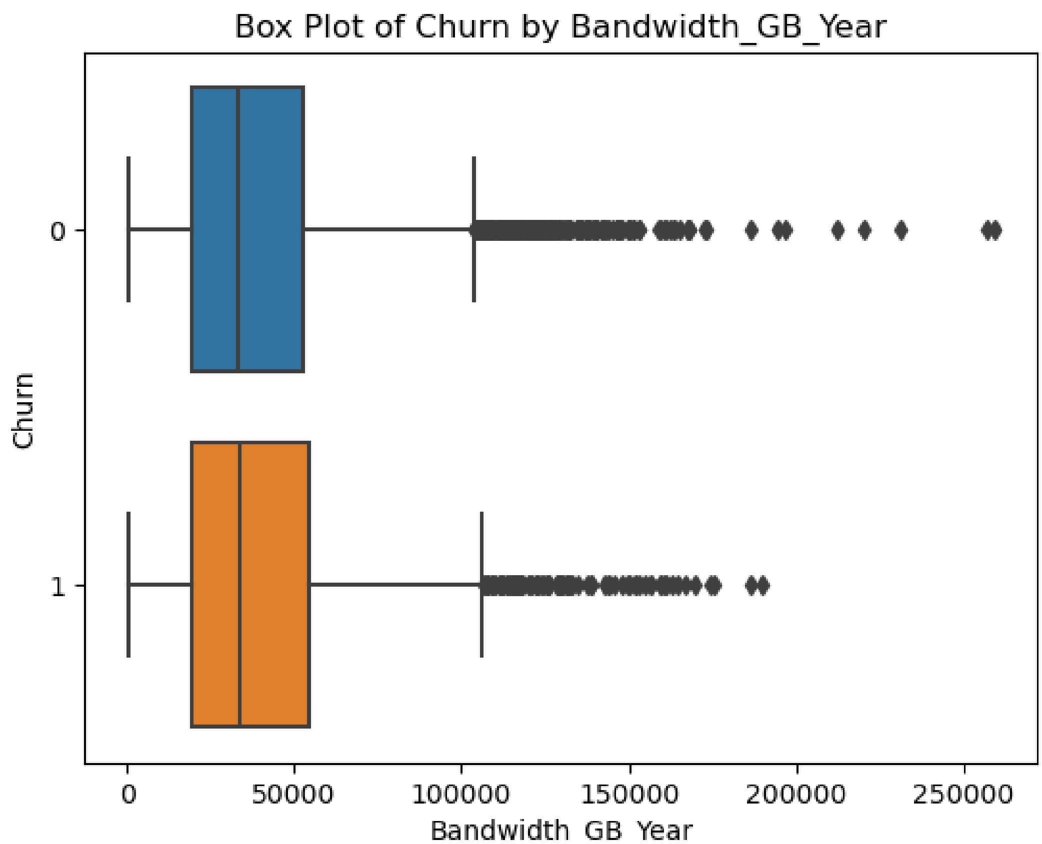
Box Plot of Churn by Email
Churn
0
1
0
50000
100000
150000
200000
250000
Email

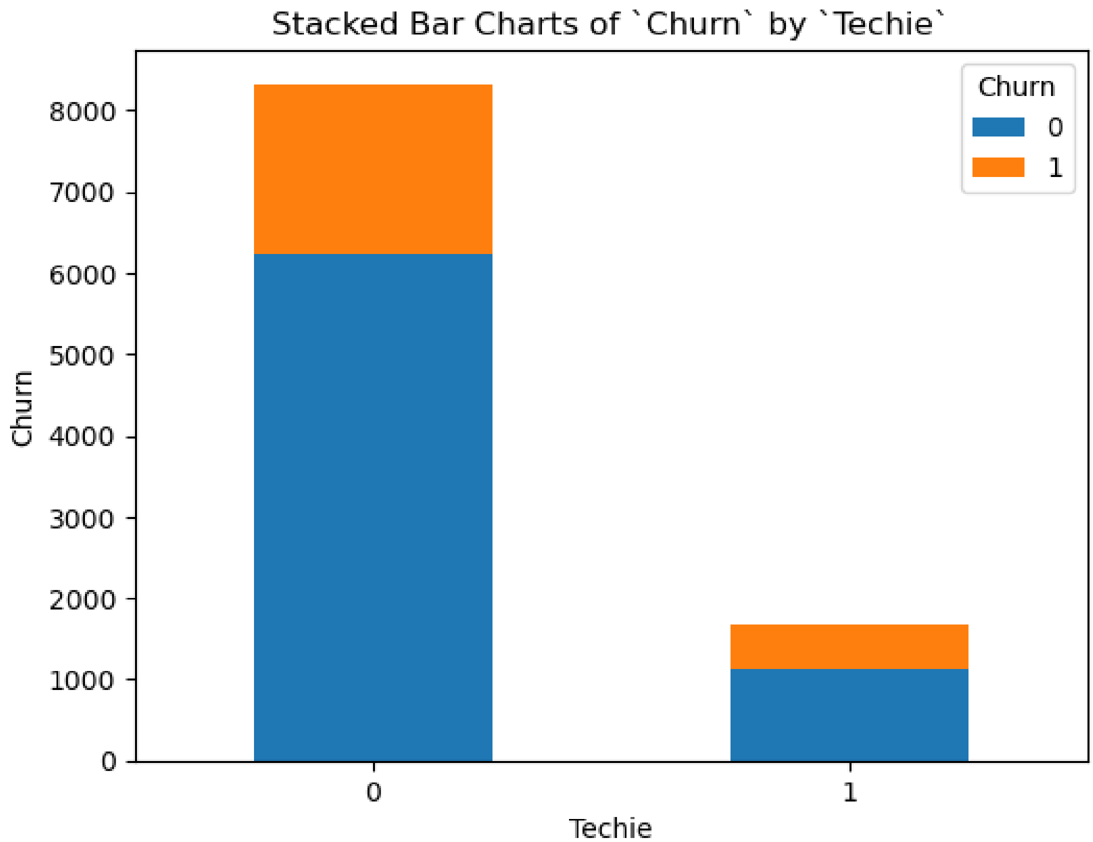

Box Plot of Churn by Contacts
Churn
0
1
Contacts
0
50000
100000
150000
200000
250000

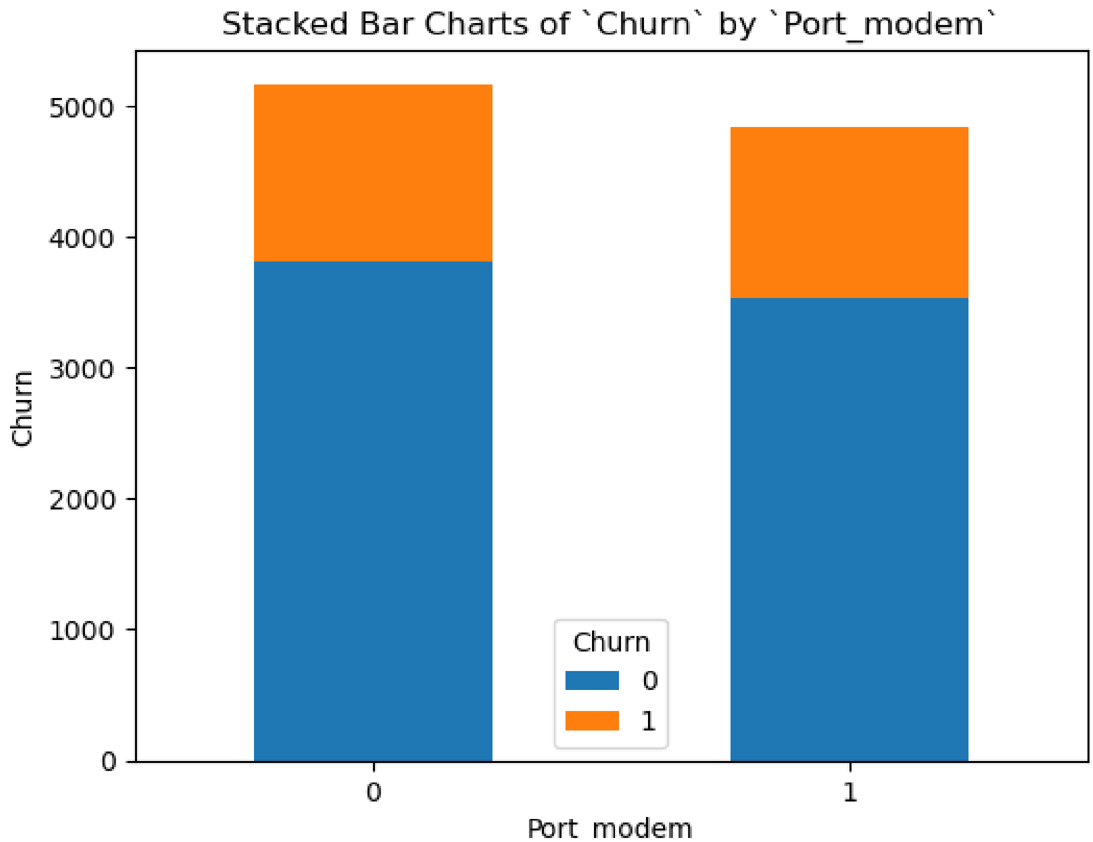

Box Plot of Churn by Yearly_equip_failure
Churn
0
1
0
50000
100000
150000
200000
250000
Yearly_equip_failure

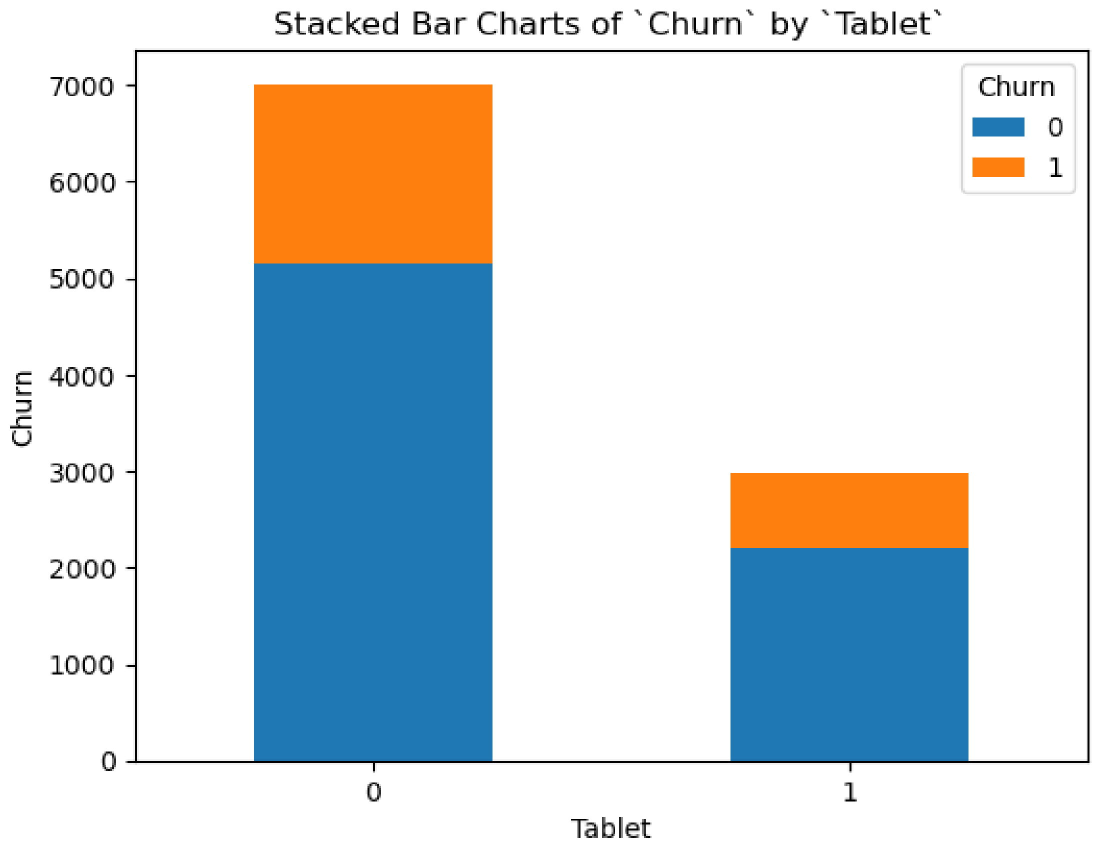

Box Plot of Churn by Tenure
Churn
0
1
Tenure
0
50000
100000
150000
200000
250000


Box Plot of Churn by MonthlyCharge
Churn
0
1
0
50000
100000
150000
200000
250000
MonthlyCharge

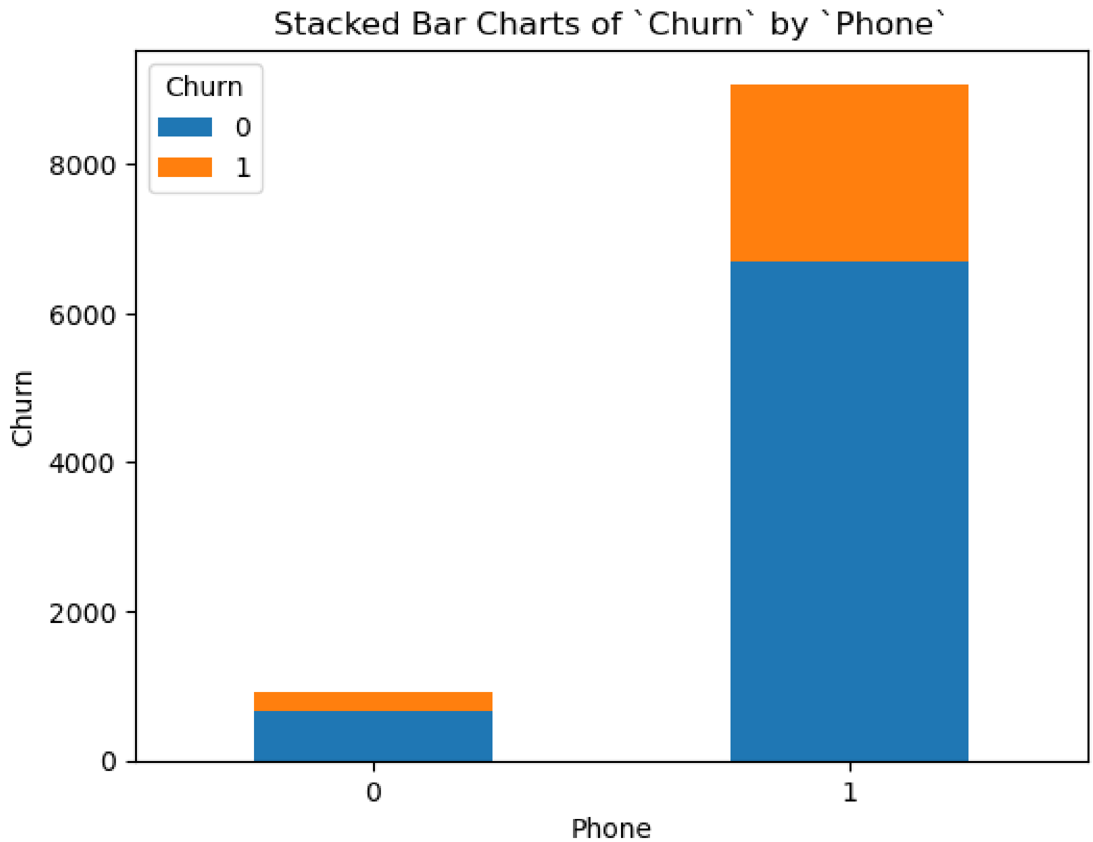

Box Plot of Churn by Bandwidth_GB_Year
Churn
0
1
0
50000
100000
150000
200000
250000
Bandwidth_GB_Year

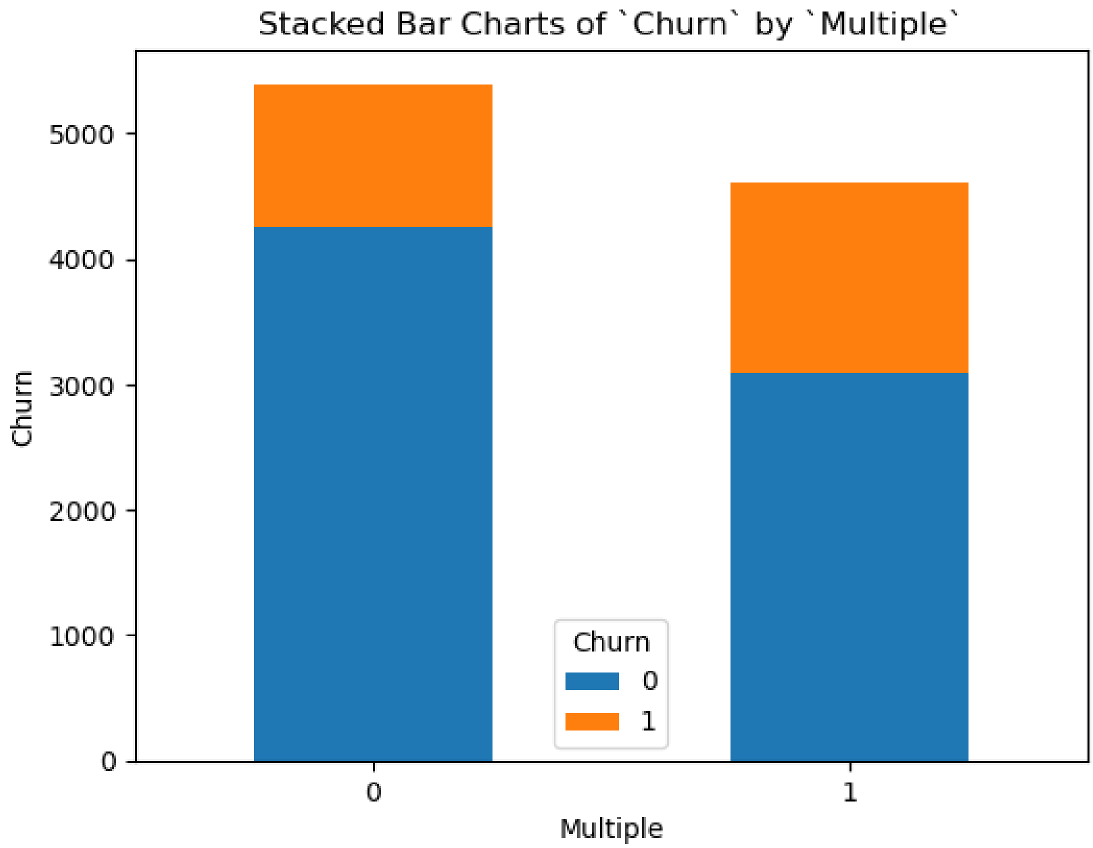

Stacked Bar Charts of `Churn` by `Techie`
Churn
0
1
8000
7000
6000
5000
4000
3000
2000
1000
0
Churn
0
1
Techie


Stacked Bar Charts of `Churn` by `Port_modem`
5000
4000
3000
2000
1000
0
Churn
Churn
0
1
0
1
Port_modem

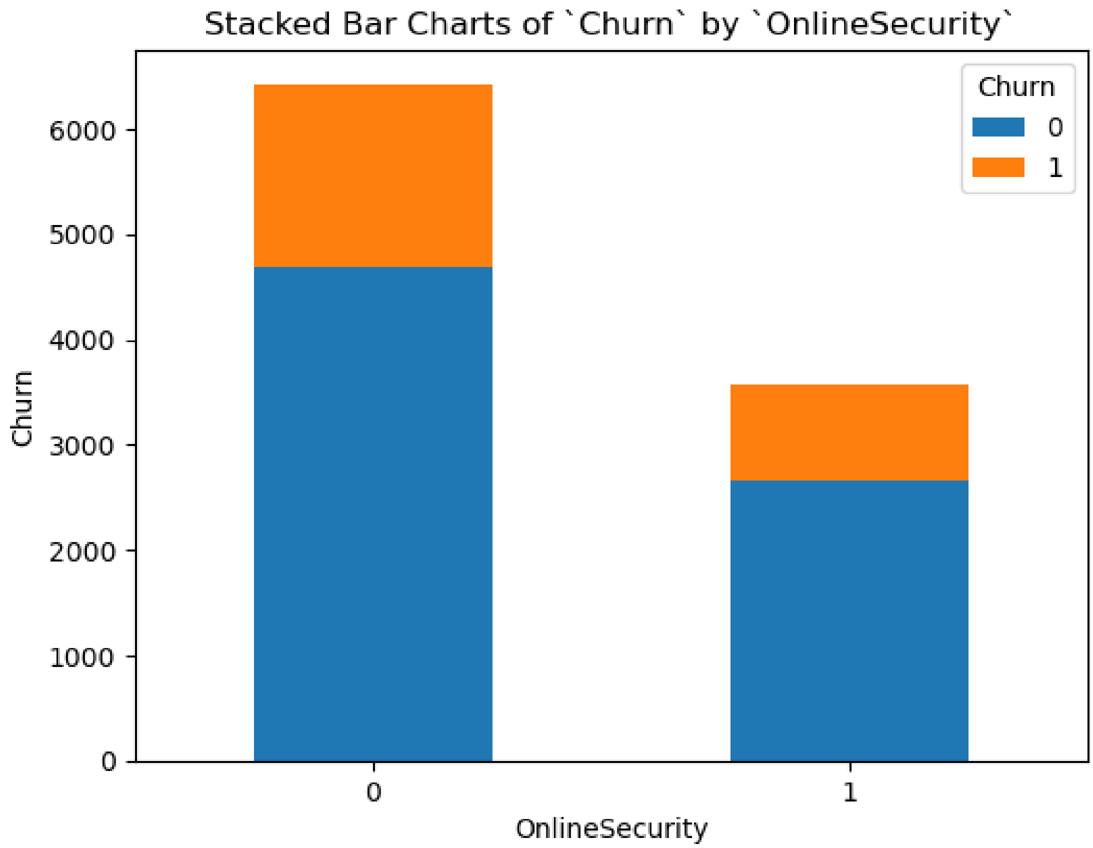


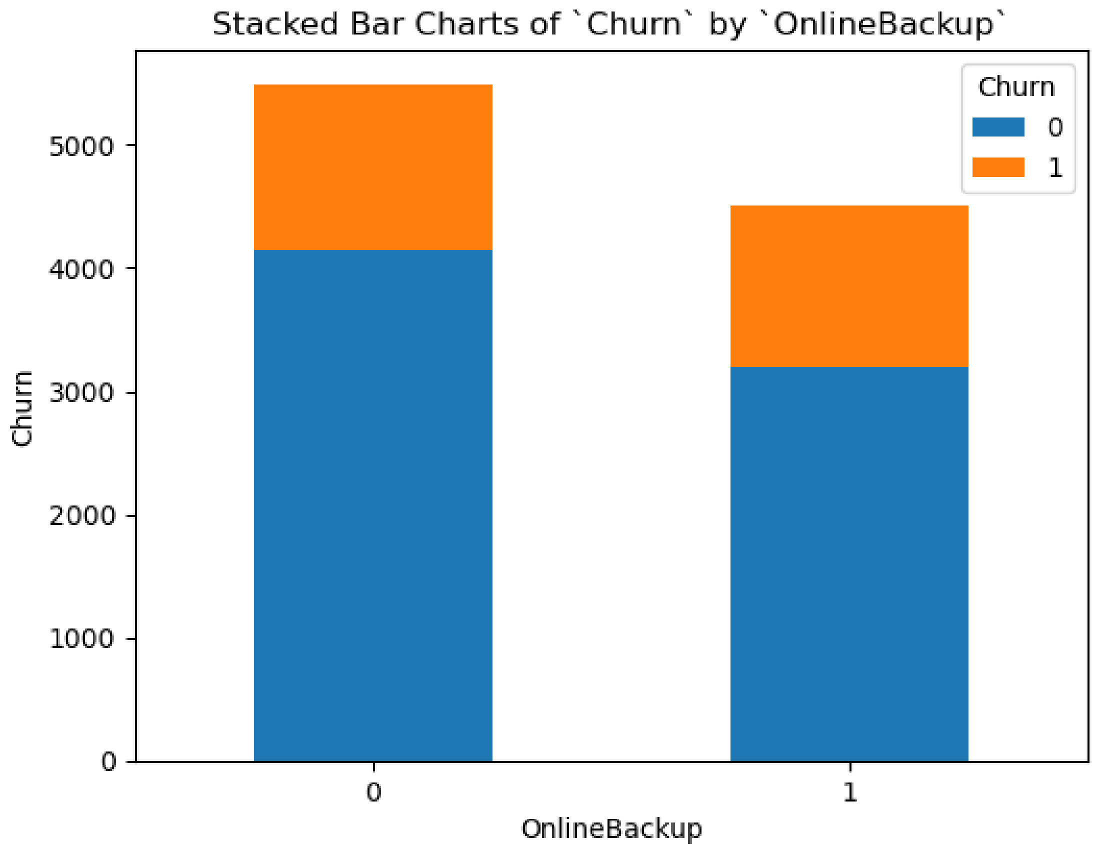
Stacked Bar Charts of `Churn` by `Tablet`
Churn
0
1
7000
6000
5000
4000
3000
2000
1000
0
Churn
0
1
Tablet

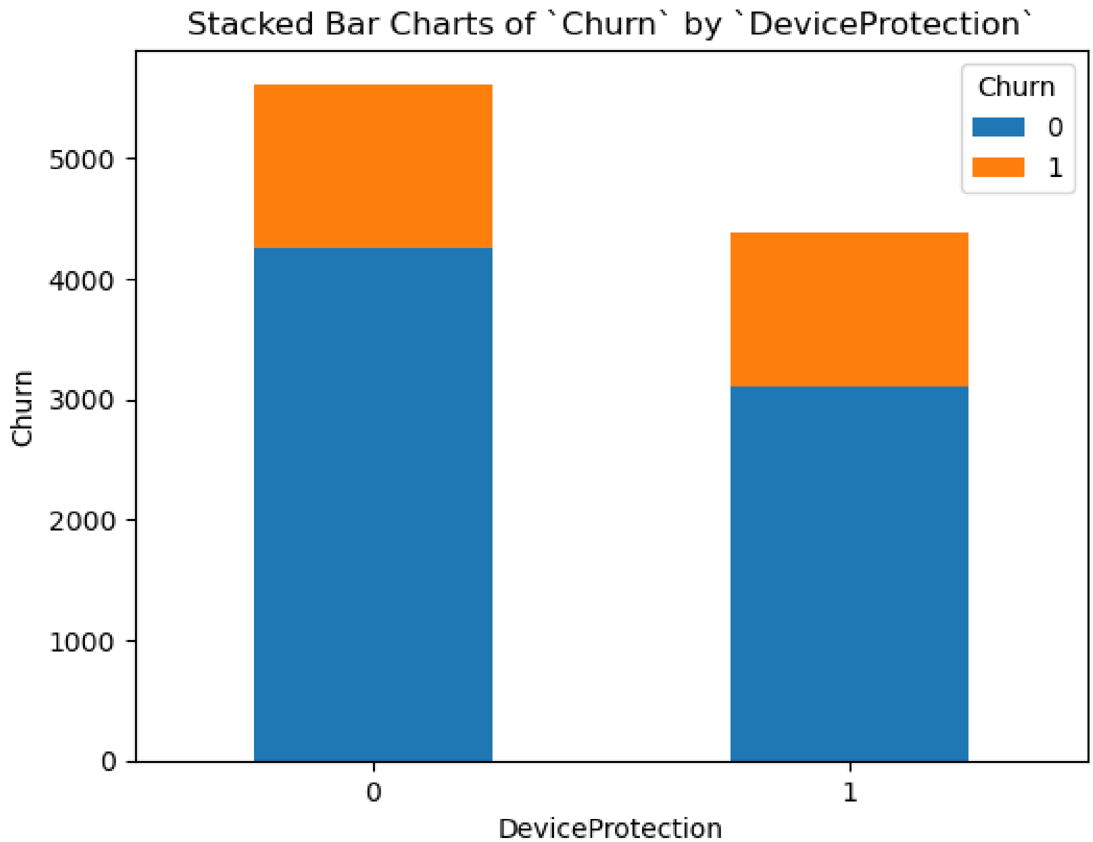

Stacked Bar Charts of `Churn` by `Phone`
Churn
0
1
8000
6000
4000
2000
0
Churn
0
1
Phone

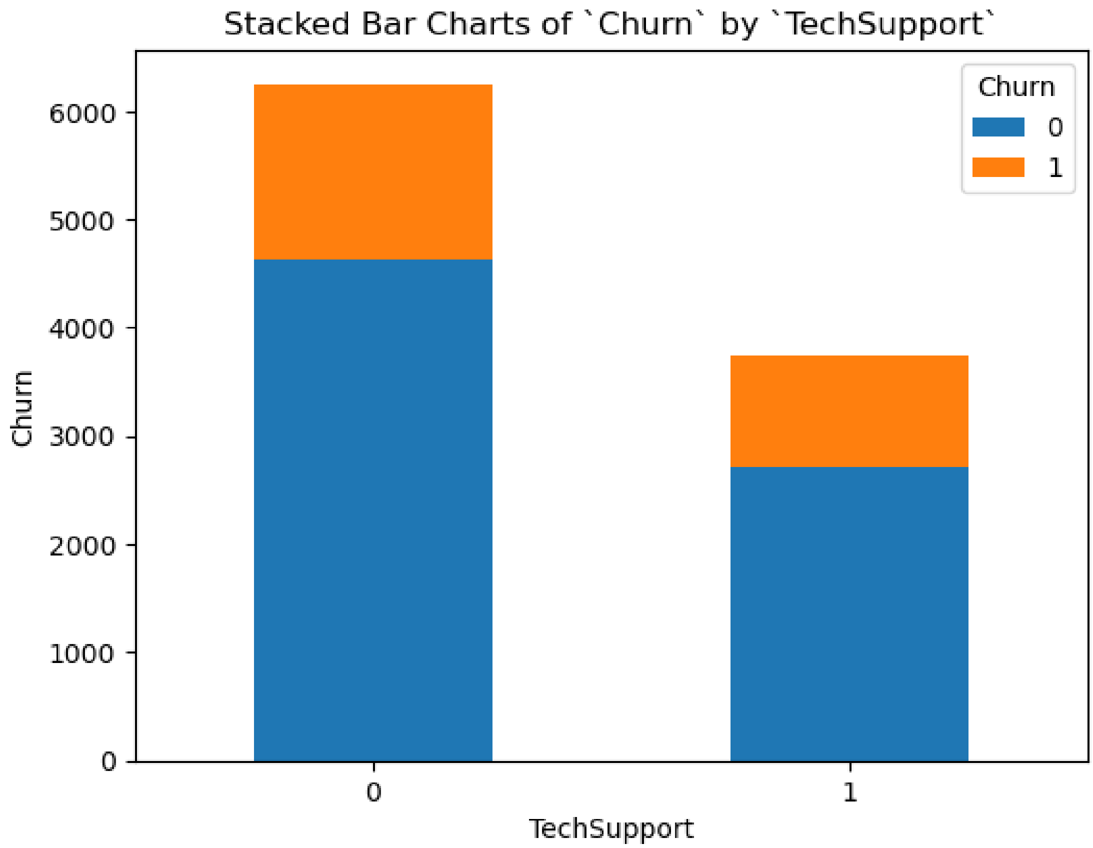


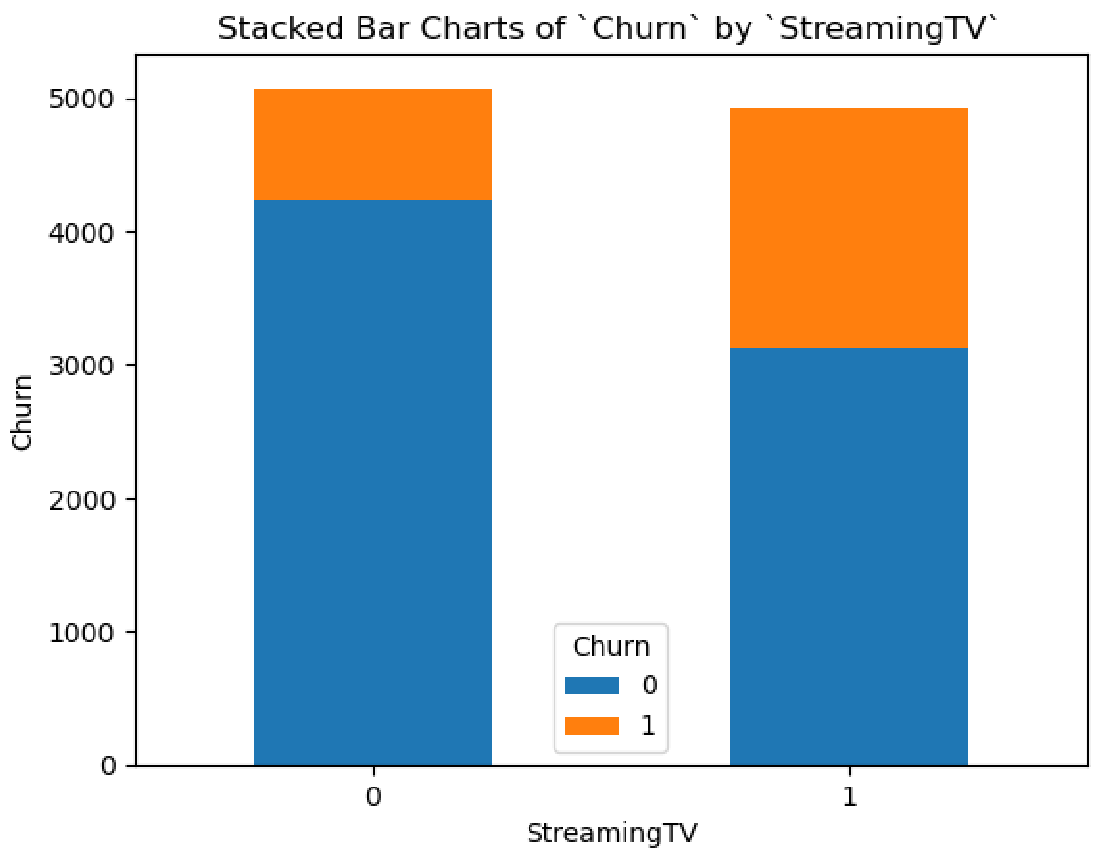
Stacked Bar Charts of `Churn` by `Multiple`
5000
4000
3000
2000
1000
0
Churn
0
1
Multiple
Churn
0
1


Stacked Bar Charts of `Churn` by `OnlineSecurity`
Churn
0
1
6000
5000
4000
3000
2000
1000
0
Churn
0
1
OnlineSecurity

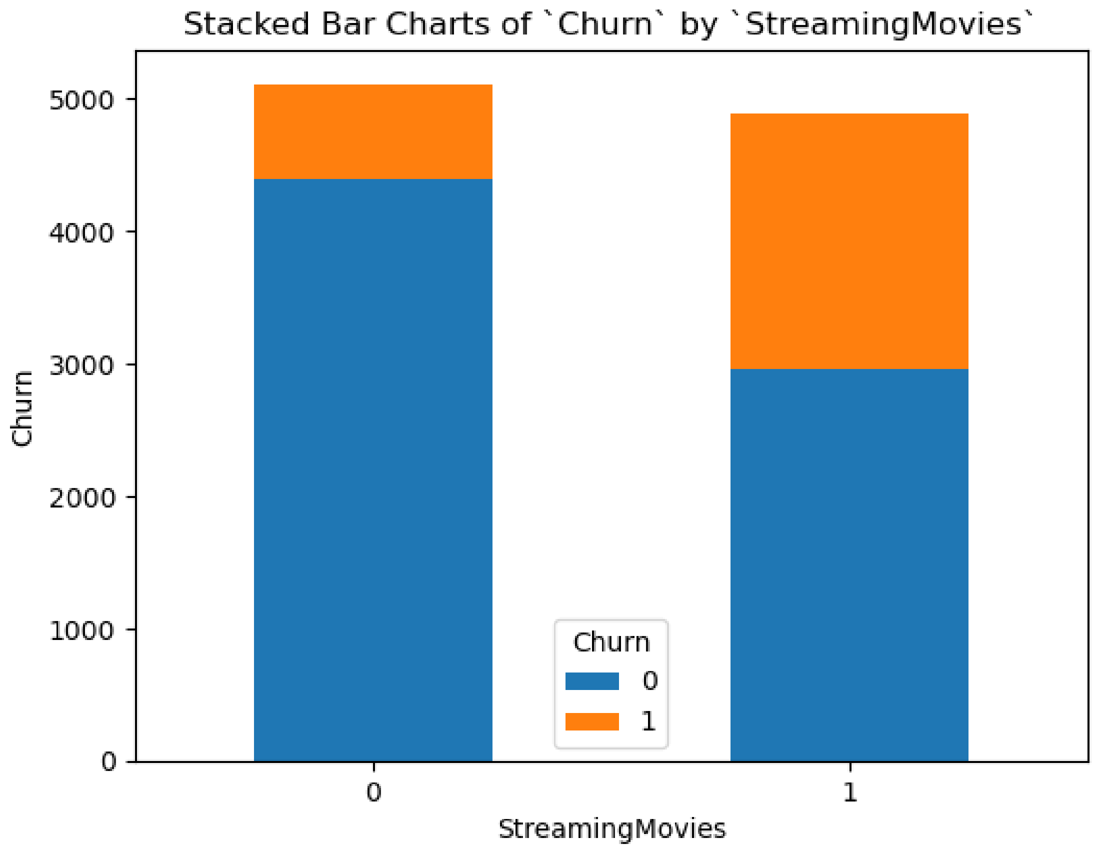

Stacked Bar Charts of `Churn` by `OnlineBackup`
Churn
0
1
5000
4000
3000
2000
1000
0
Churn
0
1
OnlineBackup

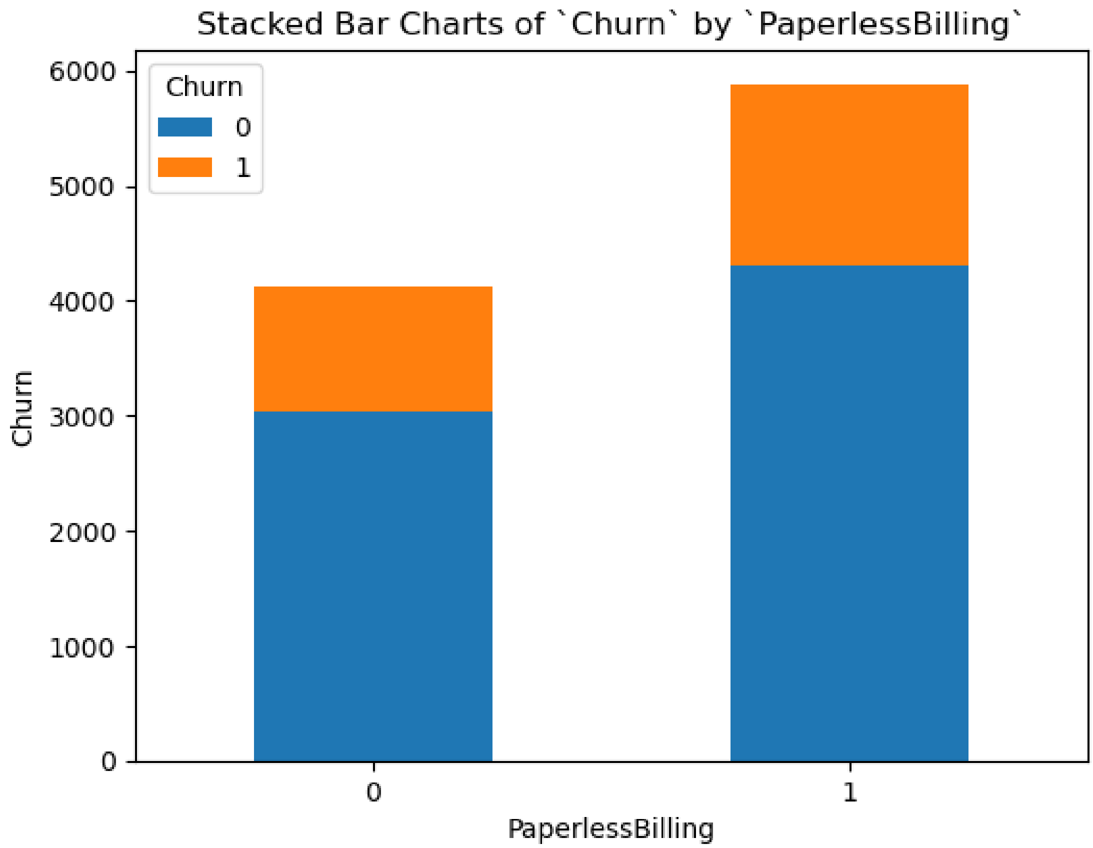

Stacked Bar Charts of `Churn` by `DeviceProtection`
Churn
0
1
5000
4000
3000
2000
1000
0
Churn
0
1
DeviceProtection

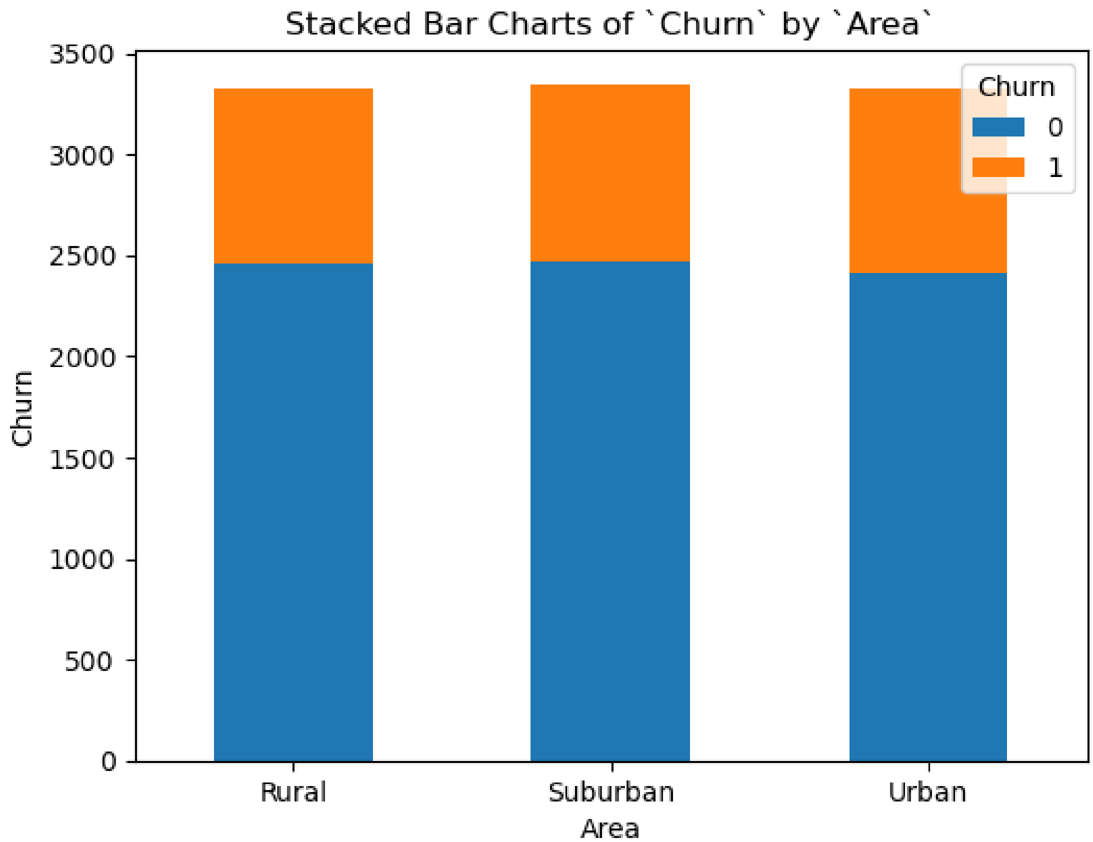

Stacked Bar Charts of `Churn` by `TechSupport`
Churn
0
1
6000
5000
4000
3000
2000
1000
0
Churn
0
1
TechSupport

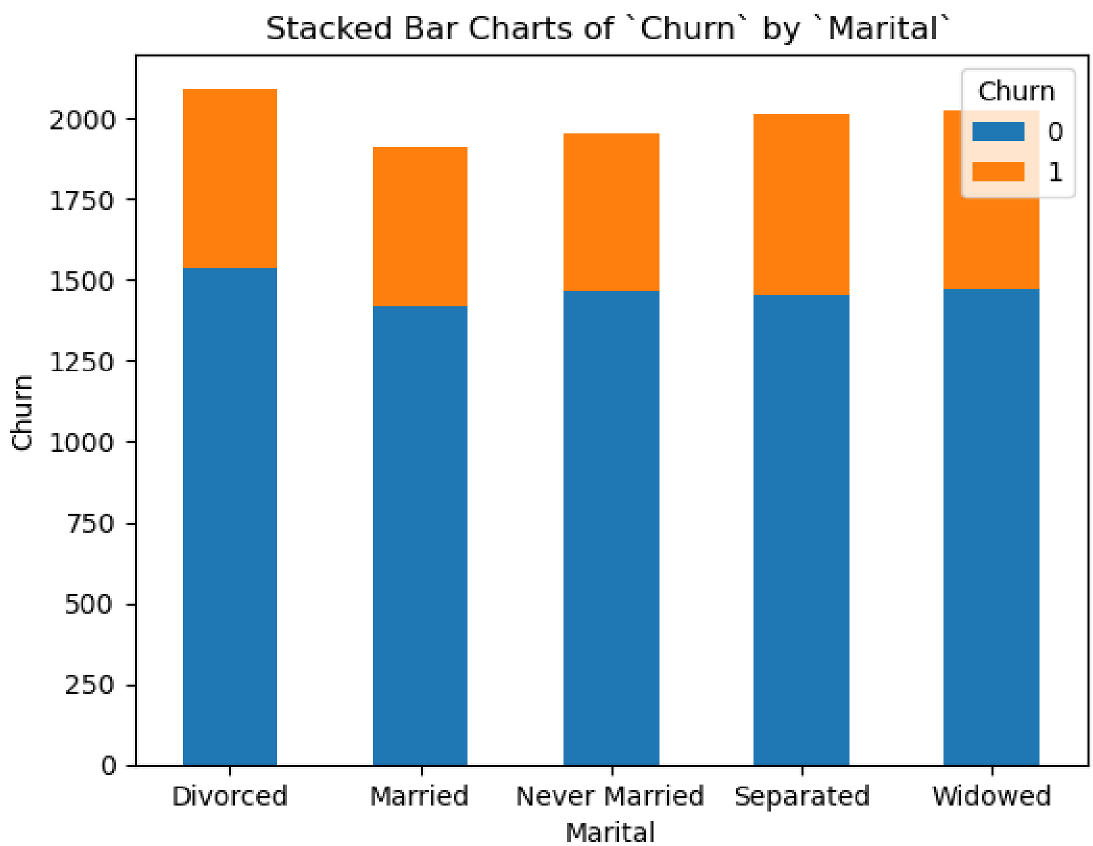


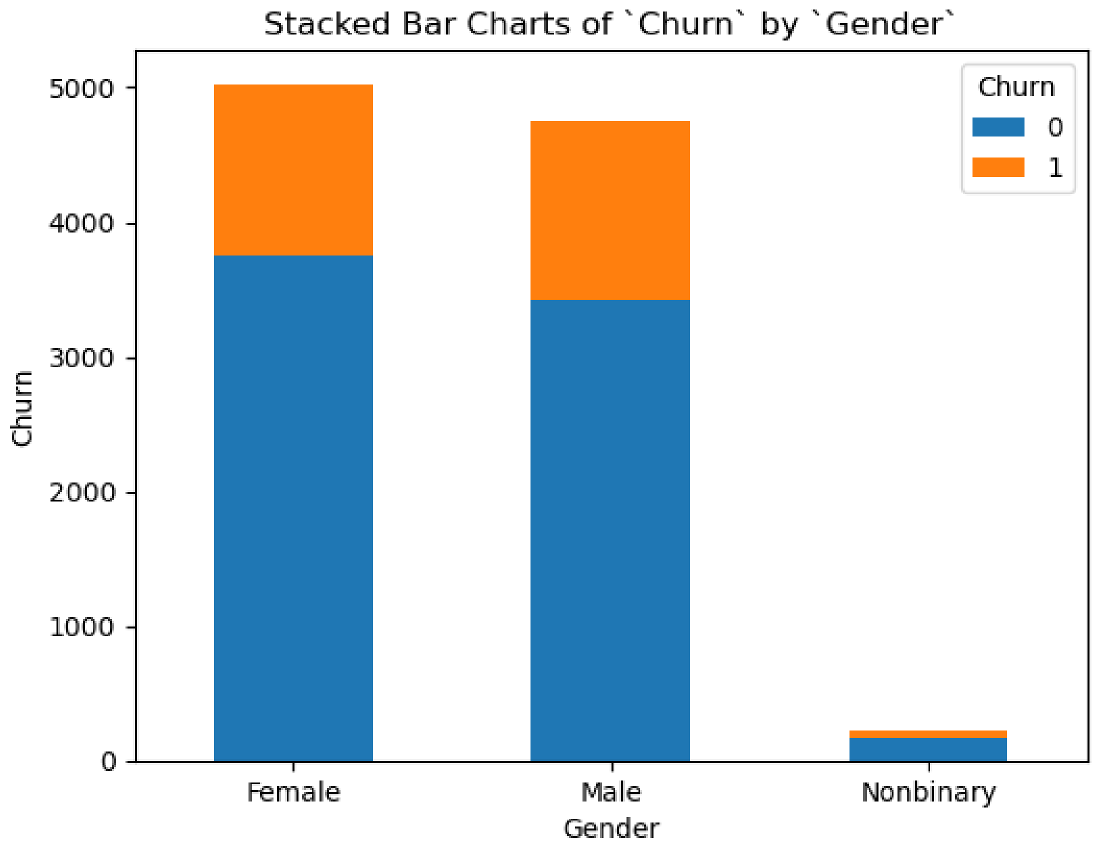
Stacked Bar Charts of `Churn` by `StreamingTV`
5000
4000
3000
2000
1000
0
Churn
Churn
0
1
0
1
StreamingTV

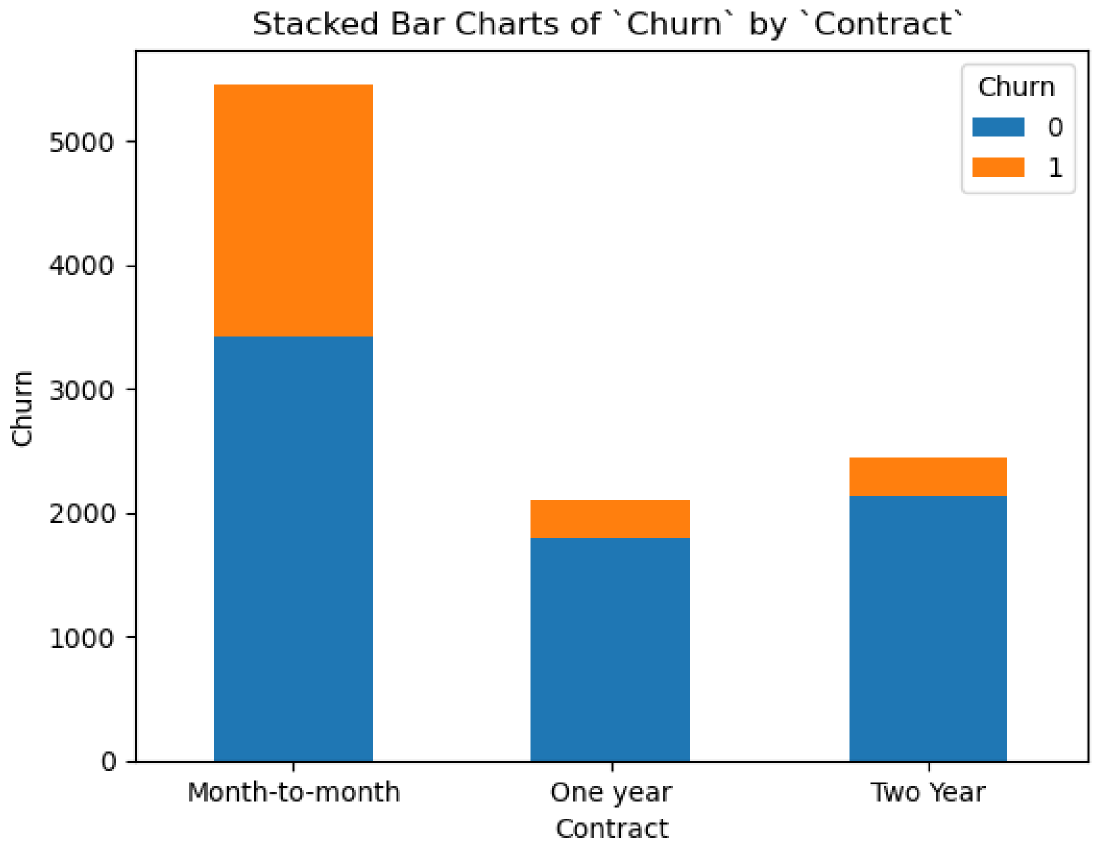

Stacked Bar Charts of `Churn` by `StreamingMovies`
Churn
0
1000
2000
3000
4000
5000
0
1
StreamingMovies
Churn
0
1


Stacked Bar Charts of `Churn` by `PaperlessBilling`
Churn
0
1
6000
5000
4000
3000
2000
1000
0
Churn
0
1
PaperlessBilling

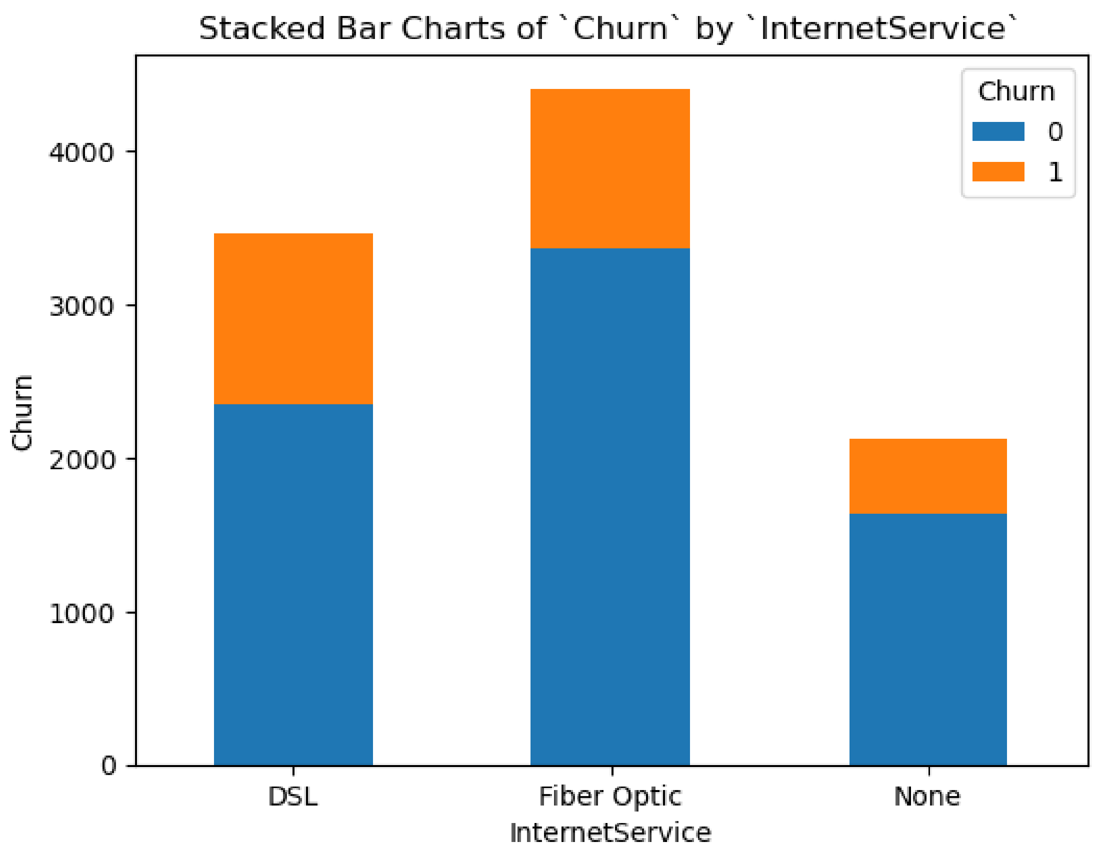

Stacked Bar Charts of `Churn` by `Area`
Churn
0
1
3500
3000
2500
2000
1500
1000
500
0
Churn
Rural
Suburban
Urban
Area

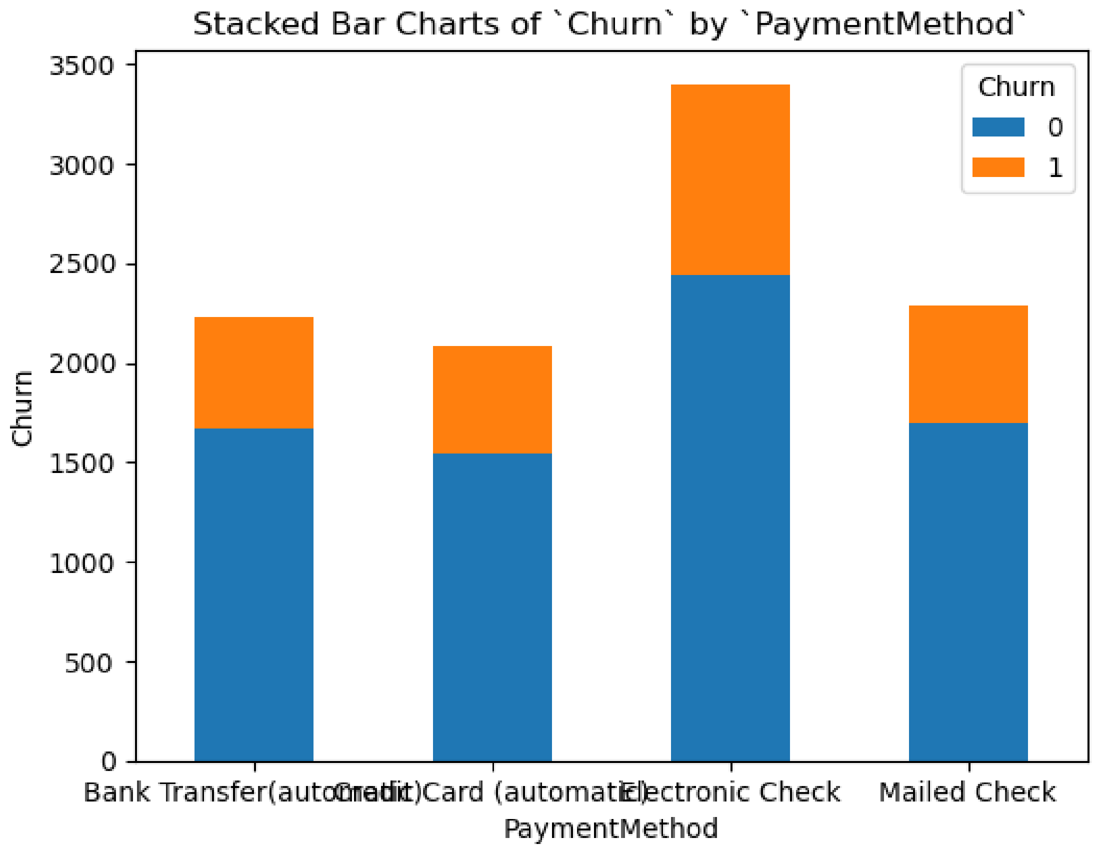

Stacked Bar Charts of `Churn` by `Marital`
Churn
0
1
2000
1750
1500
1250
1000
750
500
250
0
Churn
Divorced
Married
Never Married
Separated
Widowed
Marital

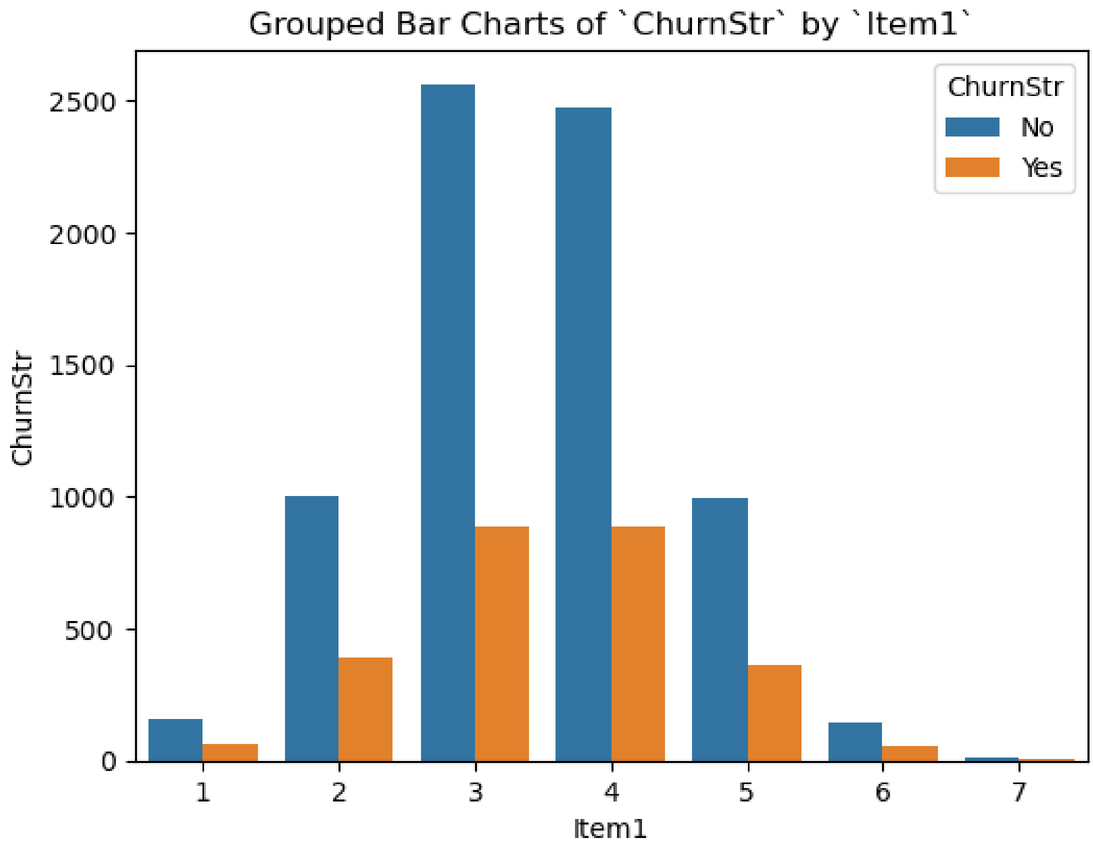

Stacked Bar Charts of `Churn` by `Gender`
Churn
0
1
5000
4000
3000
2000
1000
0
Churn
Female
Male
Nonbinary
Gender

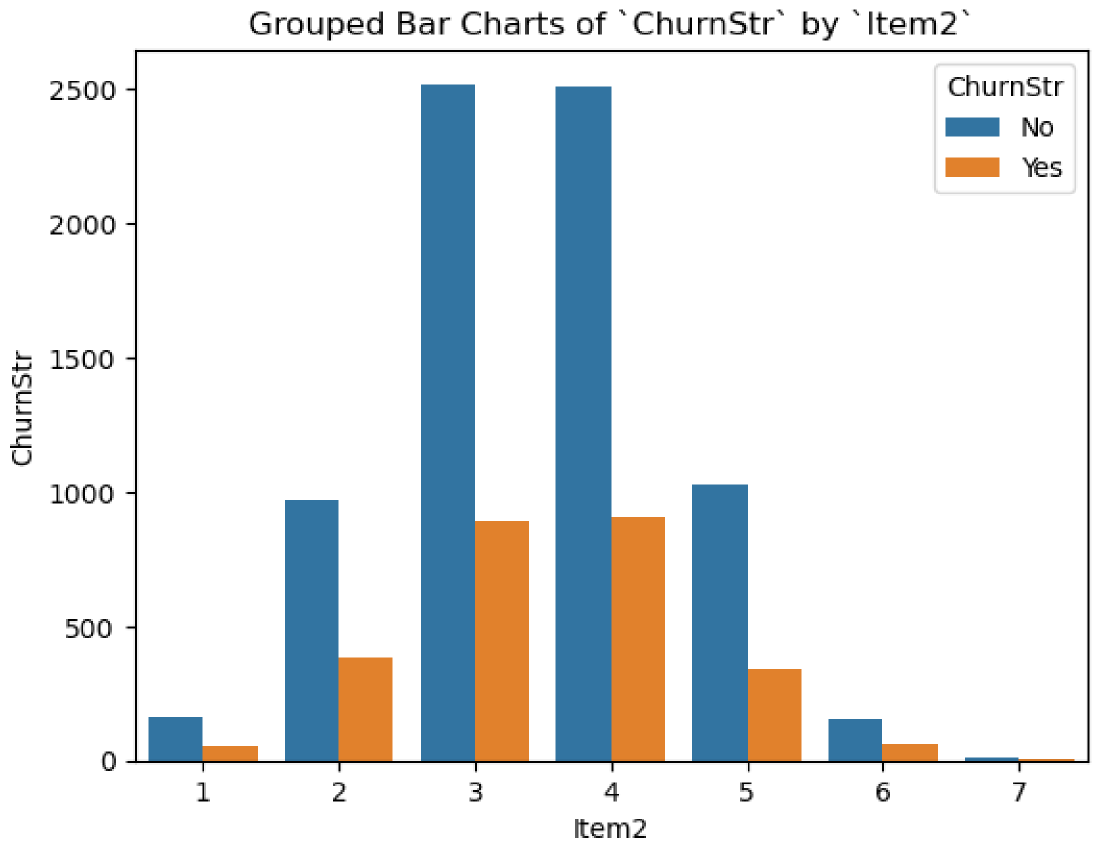


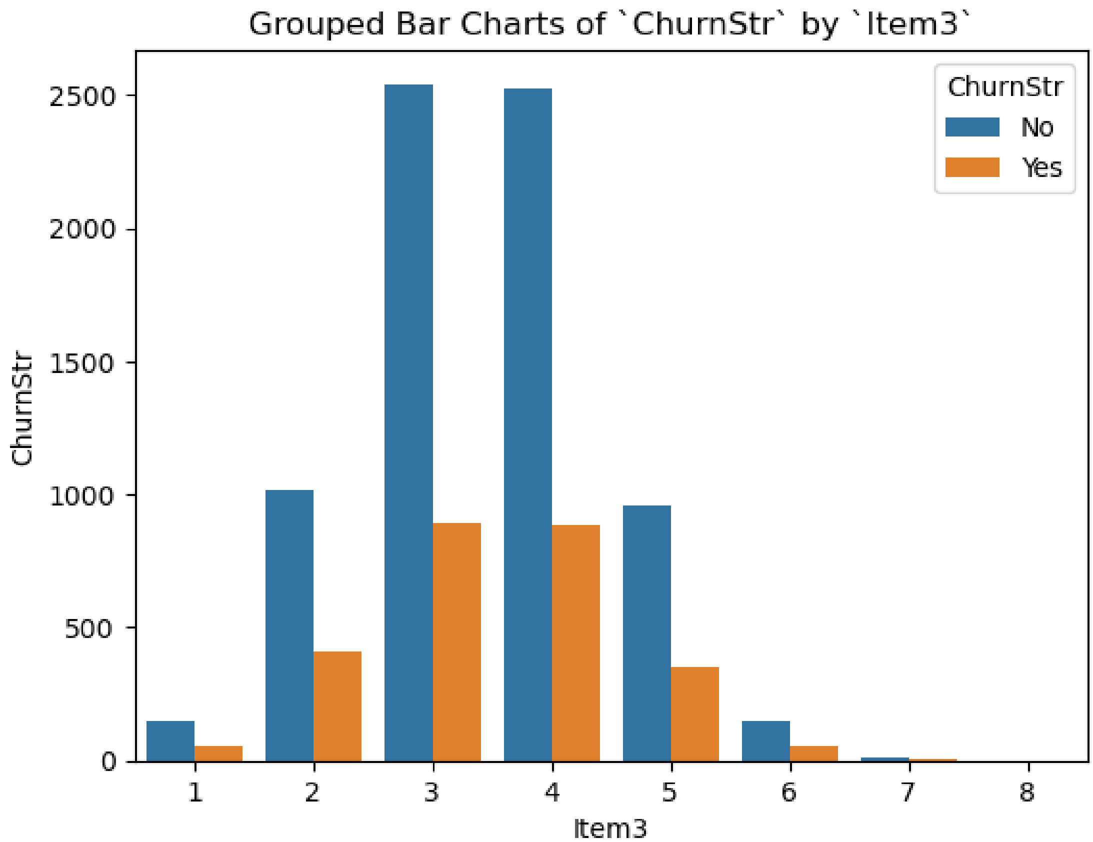
Stacked Bar Charts of `Churn` by `Contract`
Churn
0
1
5000
4000
3000
2000
1000
0
Churn
Month-to-month
One year
Two Year
Contract

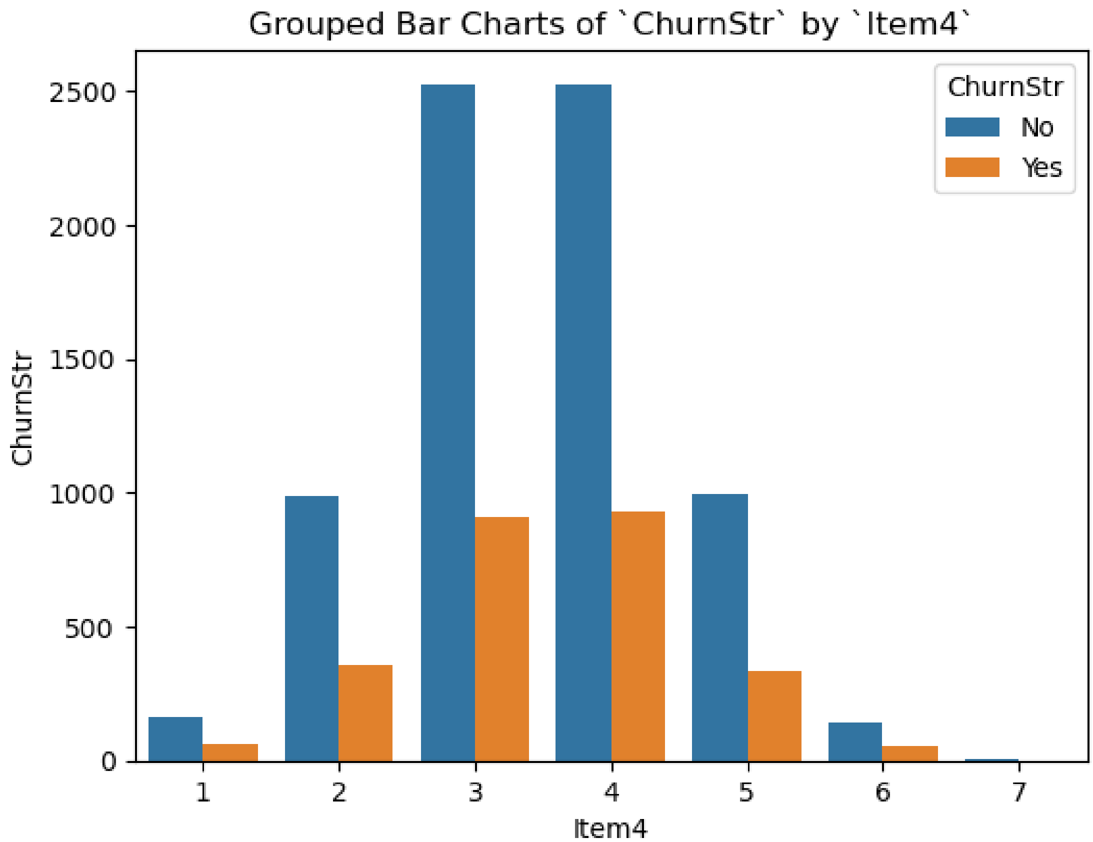

Stacked Bar Charts of `Churn` by `InternetService`
Churn
0
1
4000
3000
2000
1000
0
Churn
DSL
Fiber Optic
None
InternetService


Stacked Bar Charts of `Churn` by `PaymentMethod`
Churn
0
1
3500
3000
2500
2000
1500
1000
500
0
Churn
Bank Transfer(automatic)
Credit Card (automatic)
Electronic Check
Mailed Check
PaymentMethod

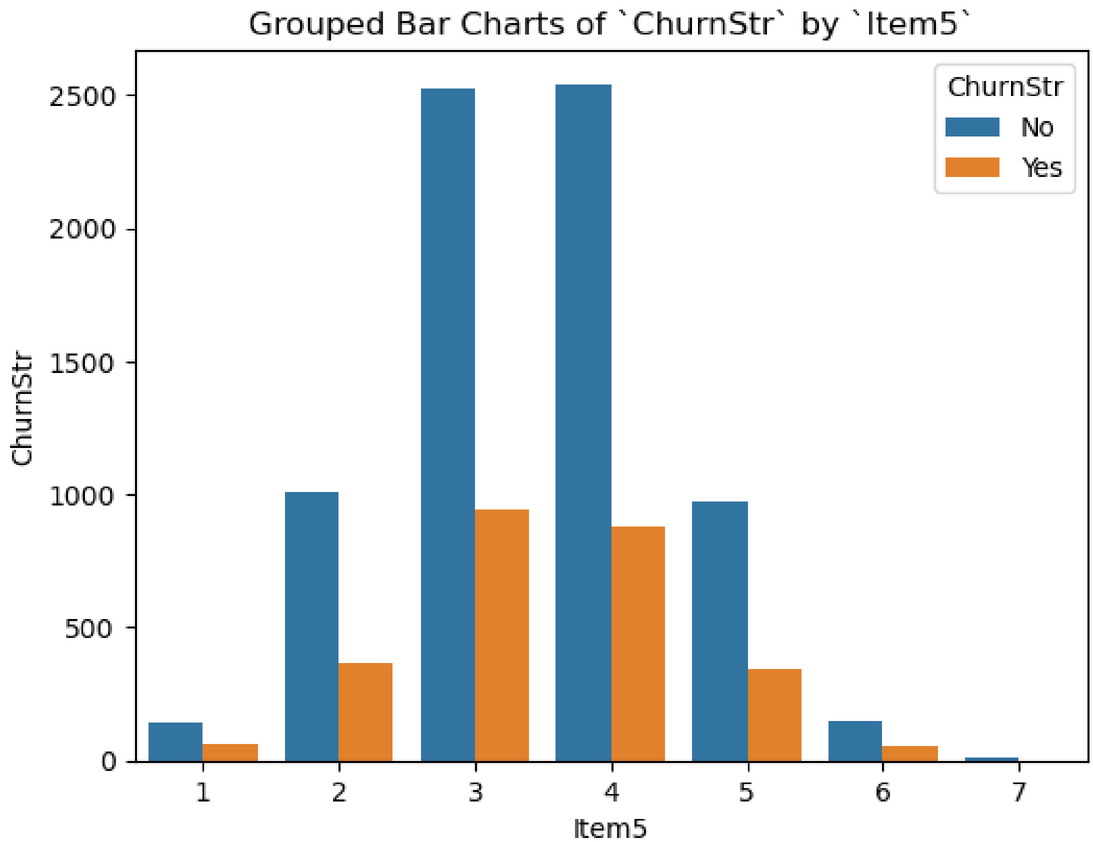

Grouped Bar Charts of `ChurnStr` by `Item1`
ChurnStr
No
Yes
2500
2000
1500
1000
500
0
ChurnStr
1
2
3
4
5
6
7
Item1

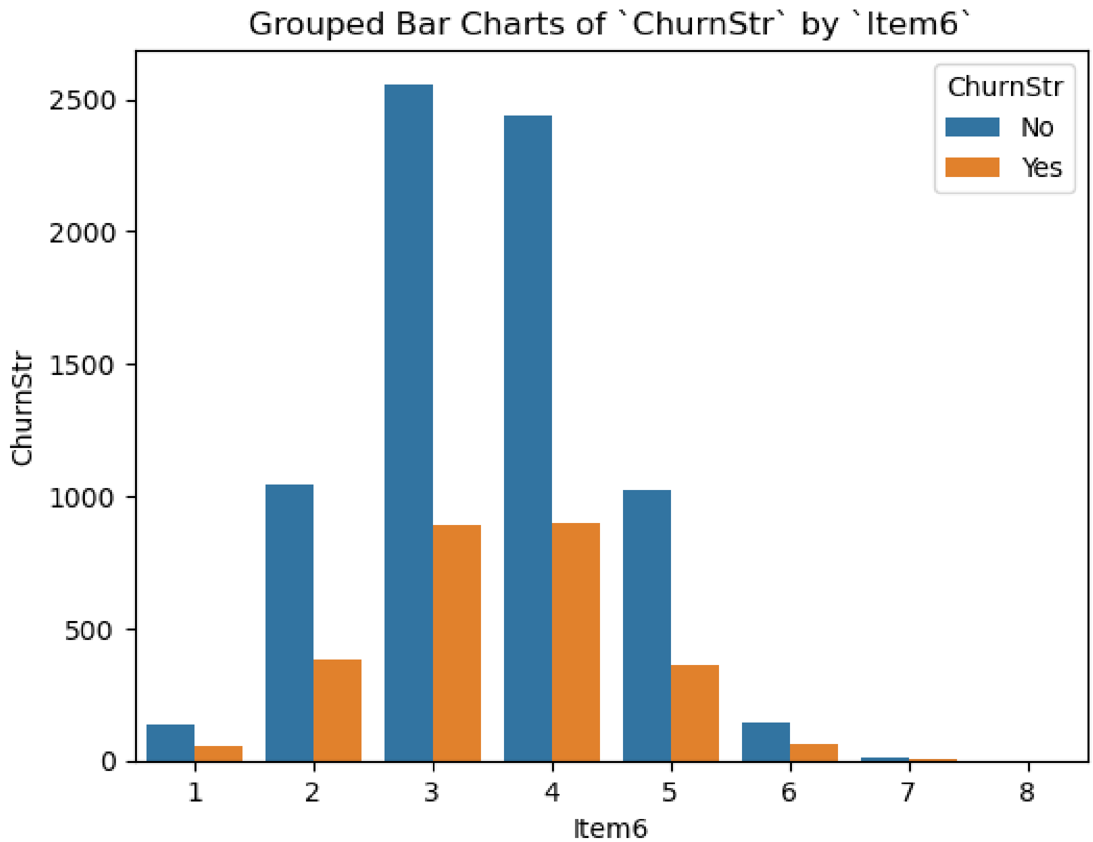

Grouped Bar Charts of `ChurnStr` by `Item2`
ChurnStr
No
Yes
0
500
1000
1500
2000
2500
1
2
3
4
5
6
7
Item2


Grouped Bar Charts of `ChurnStr` by `Item3`
ChurnStr
No
Yes
2500
2000
1500
1000
500
0
ChurnStr
1
2
3
4
5
6
7
8
Item3

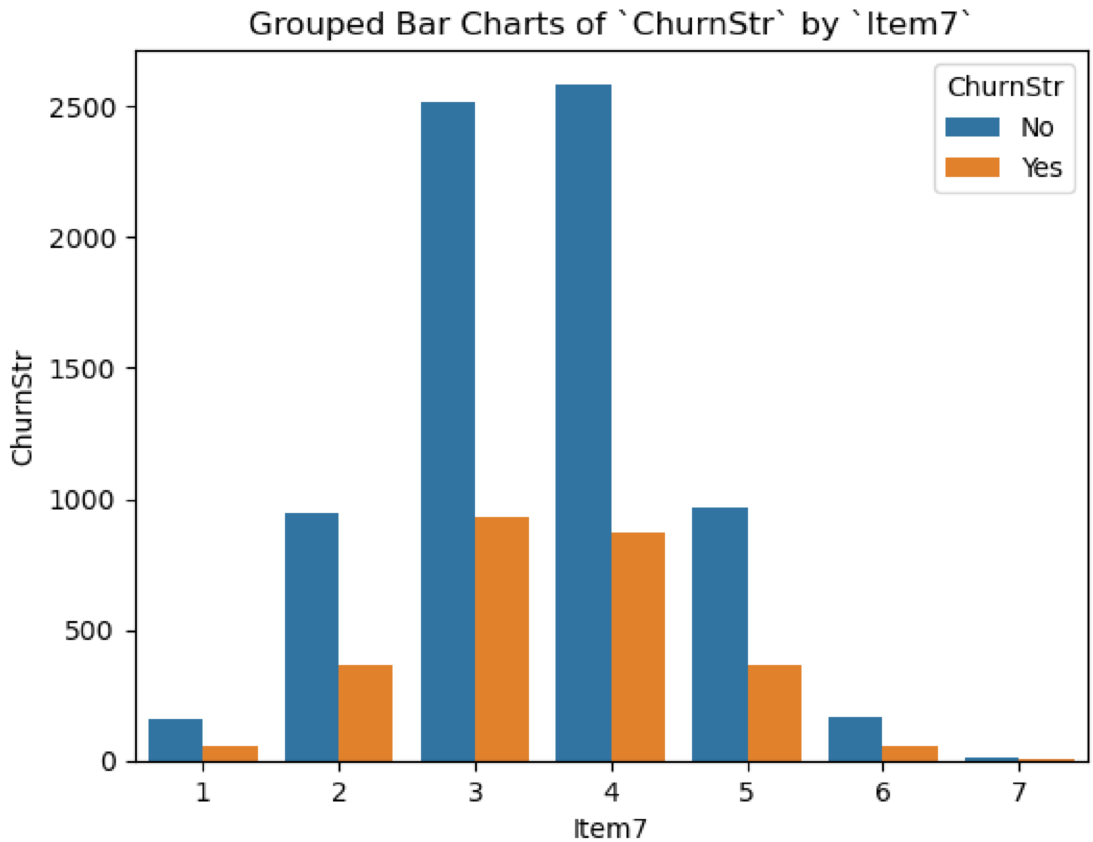


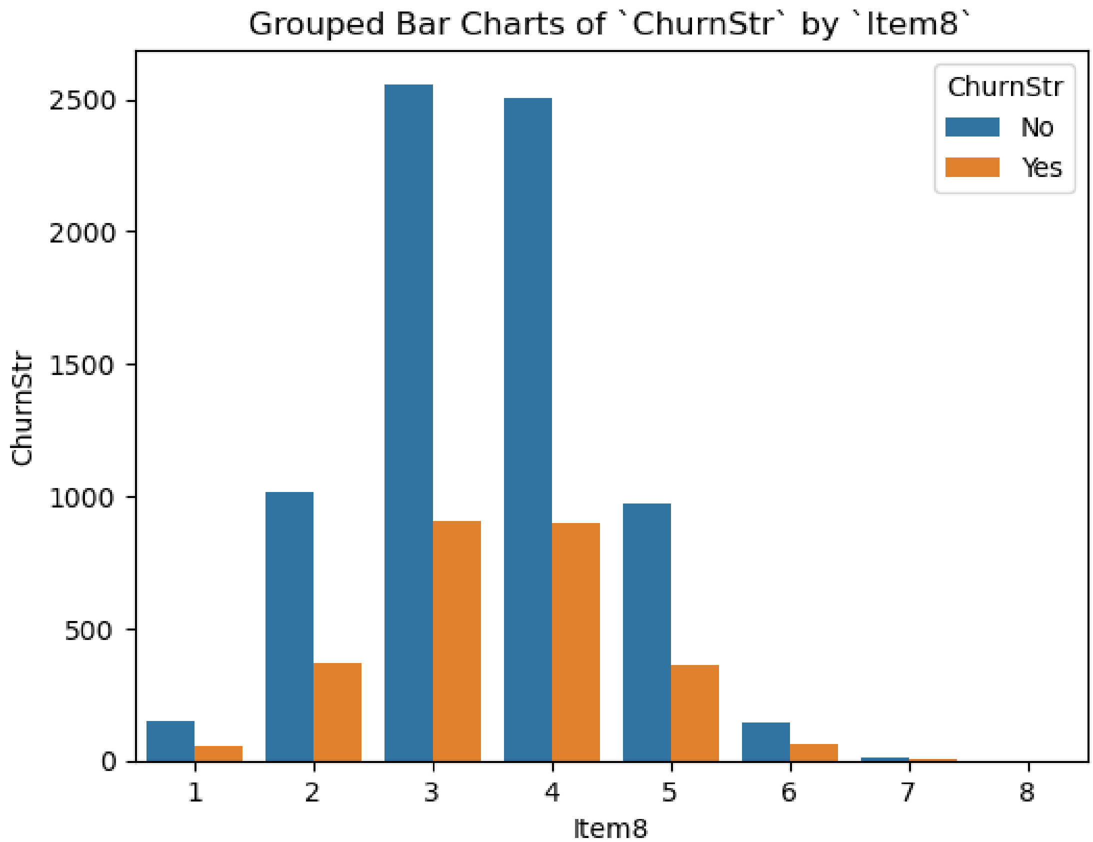
Grouped Bar Charts of `ChurnStr` by `Item4`
ChurnStr
No
Yes
2500
2000
1500
1000
500
0
ChurnStr
1
2
3
4
5
6
7
Item4


Grouped Bar Charts of `ChurnStr` by `Item5`
ChurnStr
No
Yes
2500
2000
1500
1000
500
0
ChurnStr
1
2
3
4
5
6
7
Item5



Grouped Bar Charts of `ChurnStr` by `Item6`
ChurnStr
No
Yes
2500
2000
1500
1000
500
0
ChurnStr
1
2
3
4
5
6
7
8
Item6


Grouped Bar Charts of `ChurnStr` by `Item7`
ChurnStr
No
Yes
2500
2000
1500
1000
500
0
ChurnStr
1
2
3
4
5
6
7
Item7



3.1.4 C4. Description of Data Transformation Goals and Steps to Achieve Goals

The data wrangling performed on the data set consisted of the following:

- Re-expression of the thirteen binary variables by encoding True and False as 1 and 0, respectively. This was done using a for loop that applies `.replace()` to specific columns using a dictionary. It was also necessary to type cast these with `.astype('category')` in order for the bivariate graphs to generate. The thirteen re-expressed binary variables:
  - `Churn`, `Techie`, `Port_modem`, `Tablet`, `Phone`, `Multiple`, `OnlineSecurity`, `OnlineBackup`, `DeviceProtection`, `TechSupport`, `StreamingTV`, `StreamingMovies`, `PaperlessBilling`
- Six categorical variable were re-expressed using one-hot encoding with `.get_dummies()` from Pandas. It was also necessary to type cast these with `.astype('category')` in order for the bivariate graphs to generate. The six re-expressed categorical variables:
  - `Area`, `Marital`, `Gender`, `Contract`, `InternetService`, `PaymentMethod`

See code attached, in D208_PA_MendezD_Task2_Attempt2.ipynb.

3.1.5 C5. Prepared Data Set as CSV file

[20]:
```
## C5 CSV Output

# df_prep.to_csv('D208_PA_MendezD_Task2_Attempt1.csv', sep = ',', encoding =
 ↪'utf-8', index = False)
```

4 Part IV: Model Comparison and Analysis

4.1 D. Comparison of Initial and Reduced Logistic Regression Models

4.1.1 D1. Initial Logistic Regression Model

The following cells contain the code necessary to construct the initial logistic regression model with all 39 variables identified in C2. The summary of the initial model is printed below.

[21]:
```
## D1 Initial Logistic Regression Model with 39 independent variables

import statsmodels.api as sm

# Assign 'Churn' as the dependent variable
depVar = df_prep['Churn']

# Create explanatory variable data frame
expVars = df_prep.drop('Churn', axis = 1)

# Add intercept to the model
expVars_intercept = sm.add_constant(expVars)

# Fit the intercept model
model_intercept = sm.Logit(depVar, expVars_intercept).fit()

# Print the model summary
print(model_intercept.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.217086
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9951
Method:                           MLE   Df Model:                           48
Date:                Sun, 07 Jul 2024   Pseudo R-squ.:                  0.6246
Time:                        12:04:48   Log-Likelihood:                -2170.9
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
=========================
```

```
                                       coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
-------------------------
const                                 -5.3777      1.546     -3.479
0.001      -8.408      -2.348
Lat                                    0.0032      0.007      0.452
0.651      -0.011       0.017
Lng                                   -0.0019      0.003     -0.735
0.462      -0.007       0.003
Population                          -1.64e-07    2.8e-06     -0.059
0.953   -5.65e-06    5.32e-06
Children                              -0.0055      0.137     -0.040
0.968      -0.274       0.263
Age                                    0.0028      0.015      0.194
0.846      -0.026       0.032
Income                              3.994e-07   1.38e-06      0.290
0.772    -2.3e-06     3.1e-06
Outage_sec_perweek                    -0.0027      0.013     -0.209
0.835      -0.028       0.023
Email                                 -0.0094      0.013     -0.738
0.460      -0.034       0.016
Contacts                               0.0626      0.039      1.606
0.108      -0.014       0.139
Yearly_equip_failure                  -0.0348      0.061     -0.569
0.569      -0.155       0.085
Tenure                                -0.1685      0.363     -0.464
0.642      -0.879       0.542
MonthlyCharge                          0.0394      0.014      2.851
0.004       0.012       0.067
Bandwidth_GB_Year                      0.0006      0.004      0.143
0.886      -0.008       0.009
Techie                                 1.0981      0.103     10.672
0.000       0.896       1.300
Port_modem                             0.1435      0.077      1.853
0.064      -0.008       0.295
Tablet                                -0.0508      0.085     -0.600
0.549      -0.217       0.115
Phone                                 -0.2988      0.133     -2.246
0.025      -0.560      -0.038
Multiple                               0.3602      0.202      1.786
0.074      -0.035       0.755
OnlineSecurity                        -0.2935      0.312     -0.942
0.346      -0.905       0.317
OnlineBackup                          -0.1172      0.181     -0.649
0.517      -0.471       0.237
DeviceProtection                      -0.1085      0.234     -0.464
0.642      -0.566       0.349
```

```
TechSupport                                  -0.2144      0.173     -1.237
0.216      -0.554       0.125
StreamingTV                                   1.1140      0.510      2.185
0.029       0.115       2.113
StreamingMovies                               1.2744      0.363      3.506
0.000       0.562       1.987
PaperlessBilling                              0.1661      0.079      2.108
0.035       0.012       0.320
Area_Suburban                                -0.0481      0.096     -0.503
0.615      -0.235       0.139
Area_Urban                                    0.0517      0.095      0.547
0.584      -0.134       0.237
Marital_Married                               0.1045      0.122      0.856
0.392      -0.135       0.344
Marital_Never Married                         0.0133      0.122      0.109
0.913      -0.226       0.253
Marital_Separated                             0.1179      0.120      0.979
0.327      -0.118       0.354
Marital_Widowed                               0.2603      0.121      2.158
0.031       0.024       0.497
Gender_Male                                   0.2276      0.288      0.790
0.430      -0.337       0.793
Gender_Nonbinary                             -0.0775      0.282     -0.275
0.784      -0.631       0.476
Contract_One year                            -3.4144      0.129    -26.542
0.000      -3.667      -3.162
Contract_Two Year                            -3.5185      0.127    -27.725
0.000      -3.767      -3.270
InternetService_Fiber Optic                  -1.9156      2.095     -0.914
0.361      -6.022       2.191
InternetService_None                         -0.7259      1.671     -0.434
0.664      -4.000       2.549
PaymentMethod_Credit Card (automatic)         0.2094      0.118      1.774
0.076      -0.022       0.441
PaymentMethod_Electronic Check                0.6308      0.106      5.946
0.000       0.423       0.839
PaymentMethod_Mailed Check                    0.2392      0.116      2.056
0.040       0.011       0.467
Item1                                        -0.0189      0.055     -0.346
0.730      -0.126       0.088
Item2                                        -0.0040      0.052     -0.077
0.939      -0.106       0.098
Item3                                         0.0206      0.047      0.437
0.662      -0.072       0.113
Item4                                        -0.0302      0.042     -0.718
0.473      -0.113       0.052
Item5                                        -0.0362      0.044     -0.815
0.415      -0.123       0.051
```

```
Item6                                       -0.0193     0.045     -0.427
0.670      -0.108       0.069
Item7                                       -0.0026     0.043     -0.061
0.951      -0.087       0.082
Item8                                       -0.0141     0.040     -0.348
0.728      -0.093       0.065
================================================================================
=========================
```

4.1.2 D2. Model Reduction Method and Justification

The model reduction method utilized below is backward stepwise elimination. Beginning with all explanatory variables, backwards elimination removes the least significant variable, as determined by p-value, at each iteration of the loop. This is iterated until no improvement is observed, or rather, no variables are observed to have $p > 0.05$ (Middleton, 2022).

[22]:

```
# D2 Model Reduction Method

def backward_elimination_Logit(data, target, siglevel):
# Function sourced from AnalyticsVidhya to perform Backwards Elimination
# adapted for Logistic Regression

    features = data.columns.tolist()
    while(len(features) > 1):
        features_with_constant = sm.add_constant(data[features])
        model = sm.Logit(target, features_with_constant)
        result = model.fit()

        p_values = result.pvalues[1:]
        max_p_value = p_values.max()

        if (max_p_value >= siglevel):
            excluded_feature = p_values.idxmax()
            features.remove(excluded_feature)
        else:
            break
    return features, result
```

[23]:

```
# D2 Model Reduction Method

selected_features, model_reduced = backward_elimination_Logit(expVars, depVar,␣
 ↪0.05)

print(selected_features)
```

```
Optimization terminated successfully.
         Current function value: 0.217086
```

```
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217086
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217086
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217086
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217086
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217087
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217089
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217093
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217098
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217105
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217114
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217120
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217131
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217143
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217161
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217181
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217204
```

```
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217230
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217251
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217269
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217299
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217306
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217336
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217347
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217384
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217424
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217465
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217510
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217631
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217787
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.217875
         Iterations 9
Optimization terminated successfully.
         Current function value: 0.218031
         Iterations 9
['Tenure', 'MonthlyCharge', 'Techie', 'Phone', 'Multiple', 'OnlineSecurity',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling',
```

'Marital_Widowed', 'Gender_Male', 'Contract_One year', 'Contract_Two Year', 'InternetService_Fiber Optic', 'InternetService_None', 'PaymentMethod_Electronic Check']

4.1.3 D3. Reduced Model

Upon applying Backwards Elimination, the explanatory variables of the reduced model are:

- `Tenure`
- `MonthlyCharge`
- `Techie`
- `Phone`
- `Multiple`
- `OnlineSecurity`
- `TechSupport`
- `StreamingTV`
- `StreamingMovies`
- `PaperlessBilling`
- `Marital_Widowed`
- `Gender_Male`
- `Contract_One year`
- `Contract_Two Year`
- `InternetService_Fiber Optic`
- `InternetService_None`
- `PaymentMethod_Electronic Check`

The corresponding reduced model was generated above in the function `backward_elimination_Logit()`. Below is the code that prints its summary.

[24]:
```python
# D3 Reduced Model

print(model_reduced.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9982
Method:                           MLE   Df Model:                           17
Date:                Sun, 07 Jul 2024   Pseudo R-squ.:                  0.6229
Time:                        12:08:14   Log-Likelihood:                -2180.3
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
==================
                                     coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
const                             -4.7283      0.314    -15.042      0.000
```

```
-5.344      -4.112
Tenure                              -0.1158      0.003    -39.343      0.000
-0.122      -0.110
MonthlyCharge                        0.0375      0.003     14.301      0.000
0.032       0.043
Techie                               1.0987      0.102     10.725      0.000
0.898       1.300
Phone                               -0.2898      0.132     -2.197      0.028
-0.548      -0.031
Multiple                             0.4516      0.112      4.049      0.000
0.233       0.670
OnlineSecurity                      -0.2388      0.081     -2.967      0.003
-0.397      -0.081
TechSupport                         -0.1907      0.085     -2.251      0.024
-0.357      -0.025
StreamingTV                          1.3345      0.137      9.735      0.000
1.066       1.603
StreamingMovies                      1.5051      0.157      9.582      0.000
1.197       1.813
PaperlessBilling                     0.1633      0.078      2.085      0.037
0.010       0.317
Marital_Widowed                      0.1927      0.096      2.006      0.045
0.004       0.381
Gender_Male                          0.2759      0.077      3.570      0.000
0.124       0.427
Contract_One year                   -3.4023      0.128    -26.665      0.000
-3.652      -3.152
Contract_Two Year                   -3.4998      0.126    -27.808      0.000
-3.747      -3.253
InternetService_Fiber Optic         -2.1360      0.110    -19.445      0.000
-2.351      -1.921
InternetService_None                -0.9998      0.113     -8.825      0.000
-1.222      -0.778
PaymentMethod_Electronic Check       0.4759      0.082      5.833      0.000
0.316       0.636
================================================================================
==================
```

4.2 E. Analysis Using Reduced Logistic Regression Model

4.2.1 E1. Model Comparison using Model Evaluation Metrics

The initial model can be compared with the reduced model using the Pseudo R^2 and AIC. The following comparisons can be drawn using the output of the code below.

- Pseudo R^2: The slight decrease in Pseudo R^2 from the initial model to the reduced model suggests that the initial model has slightly better bit (FAQ: What are pseudo R-squareds?).
- AIC: Since the reduced model has a higher AIC, it has a worse fit relative to the initial model.

[25]:
```python
## E1 Model Comparison

def modelCompare(initialModel, reducedModel):
# Prints a data frame as a table that compares model evaluation metrics for two␣
↪regression models

    evalMetrics = {
        'Criteria': ['Pseudo R2', 'AIC'],
        'Initial': [initialModel.prsquared, initialModel.aic],
        'Reduced': [reducedModel.prsquared, reducedModel.aic]
    }

    em = pd.DataFrame(evalMetrics)
    print(em)
```

[26]:
```python
## E1 Model Comparison

modelCompare(model_intercept, model_reduced)
```

```
    Criteria      Initial      Reduced
0  Pseudo R2     0.624564     0.622929
1        AIC  4439.713102  4396.618743
```

[27]:
```python
## E1 Model Comparison
# Sourced from Statology, Bobbitt, 2021

import scipy

def likelihoodRatioTest(full_model, reduced_model, sig_level):
# Computes the likelihood ratio test statistic for two regression models
    lr_stat = -2 * (reduced_model.llf - full_model.llf)
    df_diff = full_model.df_model - reduced_model.df_model
    p_value = scipy.stats.chi2.sf(lr_stat, df_diff)

    print(f"Likelihood Ratio Test Statistic: {lr_stat}")
    if p_value < sig_level:
        print('Reject null hypothesis. Evidence to conclude that the full model␣
↪offers a significantly better fit.')
    if p_value >= sig_level:
        print('Fail to reject null hypothesis. Evidence to conclude that the␣
↪nested model offers a significantly better fit.')
```

[28]:
```python
## E1 Model Comparison

likelihoodRatioTest(model_intercept, model_reduced, 0.05)
```

```
Likelihood Ratio Test Statistic: 18.905641219661447
Fail to reject null hypothesis. Evidence to conclude that the nested model
```

offers a significantly better fit.

4.2.2 E2. Confusion Matrix and Accuracy Calculation

Below is the code that generates the confusion matrix and performs the accuracy calculation.

[29]:
```
## E2 Confusion Matrix and Accuracy Calculation
# Code sourced from Geeksforgeeks

# Data frame of explanatory test data
expTest = df_prep[['Tenure', 'MonthlyCharge', 'Techie', 'Phone',
                    'Multiple', 'OnlineSecurity', 'TechSupport', 'StreamingTV',
                    'StreamingMovies', 'PaperlessBilling', 'Marital_Widowed',␣
  ↪'Gender_Male',
                    'Contract_One year', 'Contract_Two Year',␣
  ↪'InternetService_Fiber Optic',
                    'InternetService_None', 'PaymentMethod_Electronic Check']]
expTest = sm.add_constant(expTest)

# Data frame of dependent test data
depTest = df_prep['Churn']

yHat = model_reduced.predict(expTest)
prediction = list(map(round, yHat))
```

[30]:
```
## E2 Confusion Matrix and Accuracy Calculation

from sklearn.metrics import confusion_matrix, accuracy_score

# Confusion Matrix
cm = confusion_matrix(depTest, prediction)
print (f'Confusion Matrix: \n {cm}')

# Accuracy Score
print(f'Test accuracy = {accuracy_score(depTest, prediction)}')
```

```
Confusion Matrix:
 [[6908  442]
 [ 521 2129]]
Test accuracy = 0.9037
```

4.2.3 E3. Code

See code attached, in D208_PA_MendezD_Task2_Attempt2.ipynb

5 Part V: Data Summary and Implications

5.1 F. Summary of Findings

5.1.1 F1. Results of Data Analysis

- Logistic Regression Equation for the Reduced Model

Let the linear combination of predictor variables and their coefficients, $\mathbf{X}\beta$, be defined by

$$
\begin{aligned}
\mathbf{X}\beta = &- 4.73 - 0.12 \cdot \text{Tenure} + 0.04 \cdot \text{MonthlyCharge} + 1.1 \cdot \text{Techie} \\
&- 0.29 \cdot \text{Phone} + 0.45 \cdot \text{Multiple} - 0.24 \cdot \text{OnlineSecurity} - 0.19 \cdot \text{TechSupport} \\
&+ 1.33 \cdot \text{StreamingTV} + 1.51 \cdot \text{StreamingMovies} + 0.16 \cdot \text{PaperlessBilling} + 0.19 \cdot \text{MaritalWidowed} \\
&+ 0.28 \cdot \text{GenderMale} - 3.4 \cdot \text{ContractOneyear} - 3.5 \cdot \text{ContractTwoYear} \\
&- 2.14 \cdot \text{InternetServiceFiberOptic} - \text{InternetServiceNone} + 0.48 \cdot \text{PaymentMethodElectronicCheck}
\end{aligned}
$$

Then the logistic regression equation $\pi(\mathbf{X})$, where π is the probability, for the reduced model is

$$\pi(\mathbf{X}) = \frac{1}{1 + \exp(-\mathbf{X}\beta)}$$

(Logistic Regression, n.d.)

- Interpretation of the Coefficients of the Reduced Model
  - The intercept -4.73 is the baseline probability when all explanatory variables are zero.
  - All else constant, for each additional unit of tenure, the probability of Churn decreases by 0.12.
  - All else constant, for each additional unit of monthly charge, the probability of Churn increases by 0.04.
  - All else constant, being a techie increases the probability of Churn by 1.1 relative to not being a techie.
  - All else constant, having a phone decreases the probability of Churn by 0.29.
  - All else constant, having multiple phone lines increases the probability of Churn by 0.45.
  - All else constant, having online security decreases the probability of Churn by 0.24.
  - All else constant, having tech support decreases the probability of Churn by 0.19.
  - All else constant, having the streaming TV service increases the probability of Churn by 1.33.
  - All else constant, having the streaming movie service increases the probability of Churn by 1.50.
  - All else constant, having paperless billing increases the probability of Churn by 0.16.
  - All else constant, being widowed increases the probability of Churn by 0.19.
  - All else constant, being male increases the probability of Churn by 0.28.
  - All else constant, having a one year contract decreases the probability of Churn by 3.4.
  - All else constant, having a two year contract decreases the probability of Churn by 3.5.
  - All else constant, having fiber optic internet service decreases the probability of Churn by 2.14.
  - All else constant, having no internet service decreases the probability of Churn by 1.
  - All else constant, having electronic check as payment method increases the probability of Churn by 0.48.

- Statistical and Practical Significance of the Reduced Model
  - As shown in E1, with a p-value of $p = 18.9$, at $\alpha = 0.05$, there is evidence to conclude that the model is statistically significant and the additional explanatory variables do not significantly improve the model. However, the model may lack practical significance, as the model does not provide clear insights into why customers churn so it may not be entirely useful in retention strategies.
- Limitations of the Data Analysis
  - A limitation of this data analysis is the assumption that any of these explanatory variables have a causal relationship between them and the dependent variable. Additionally, with 17 explanatory variables, the model could be overfit and result in poor predictive performance.

[31]:
```python
## F1 Regression Equation Coefficients

coefficients = model_reduced.params
print('Coefficients:')
print(coefficients)
```

```
Coefficients:
const                            -4.728321
Tenure                           -0.115787
MonthlyCharge                     0.037535
Techie                            1.098711
Phone                            -0.289799
Multiple                          0.451621
OnlineSecurity                   -0.238849
TechSupport                      -0.190685
StreamingTV                       1.334490
StreamingMovies                   1.505083
PaperlessBilling                  0.163328
Marital_Widowed                   0.192669
Gender_Male                       0.275927
Contract_One year                -3.402297
Contract_Two Year                -3.499849
InternetService_Fiber Optic      -2.136037
InternetService_None             -0.999832
PaymentMethod_Electronic Check    0.475893
dtype: float64
```

5.1.2 F2. Course of Action

Since the Adjusted R^2 of the reduced model was only 0.62, my recommended course of action would be to seek to further improve the model by seeking alternative customer information that might be associated with Churn.

6 Part VI: Demonstration

6.1 G. Panopto Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=72c62ab6-f7c9-4da4-bf65-b1a6011944d8

6.2 H. Acknowledgement of Web Sources

Getting started - statsmodels 0.15.0 (+73). (n.d.). https://www.statsmodels.org/devel/gettingstarted.html

pandas.get_dummies — pandas 2.2.2 documentation. (n.d.). https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

warnings — Warning control — Python 3.10.0 documentation. (n.d.). Docs.python.org. https://docs.python.org/3/library/warnings.html#warnings.filterwarnings

Verma, V. (2020, October 24). Feature Selection using Wrapper Method - Python Implementation. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/

Logistic Regression using Statsmodels. (2020, July 17). GeeksforGeeks. https://www.geeksforgeeks.org/logistic-regression-using-statsmodels/

Bobbitt, Z. (2021, November 29). How to Perform a Likelihood Ratio Test in Python. Statology. https://www.statology.org/likelihood-ratio-test-in-python/

6.3 I. Acknowledgement of Sources

Middleton, K. (2022, November). D208 - Webinar Getting Started with D208 Part I. WGU. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=15e09c73-c5aa-439d-852f-af47001b8970

Middleton, K. (2022, November). D208 - Webinar Getting Started with D208 Part II. WGU. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=39bbe2db-de7d-4bf5-913b-af5c0003da9d

FAQ: What are pseudo R-squareds? UCLA: Statistical Consulting Group. https://stats.oarc.ucla.edu/other/mult-pkg/faq/general/faq-what-are-pseudo-r-squareds/

Stoltzfus, J. C. (2011). Logistic Regression: A Brief Primer. Academic Emergency Medicine. https://pubmed.ncbi.nlm.nih.gov/21996075/

Logistic Regression (n.d.). Online.stat.psu.edu. https://online.stat.psu.edu/stat462/node/207/

likelihood test? https://www.statology.org/likelihood-ratio-test-in-r/ https://sites.warnercnr.colostate.edu/gwhite/wp-content/uploads/sites/73/2017/04/LikelihoodRatioTests.pdf