William Stults - D208 Task 2

May 17, 2022

---

1 Part I: Research Question

1.1 Research Question

My dataset for this predictive modeling exercise includes data on an internet service provider's current and former subscribers, with an emphasis on customer churn (whether customers are maintaining or discontinuing their subscription to the ISP's service). Data analysis performed on the dataset will be aimed with this research question in mind: is there a relationship between customer lifestyle, or "social" factors, and customer churn? Lifestyle and social factors might include variables such as age, income, and marital status, among others.

1.2 Objectives and Goals

Conclusions gleaned from the analysis of this data can benefit stakeholders by revealing information on which customer populations may be more likely to "churn", or terminate their service contract with the ISP. Such information may be used to fuel targeted advertising campaigns, special promotional offers, and other strategies related to customer retention.

---

2 Part II: Method Justification

2.1 Assumptions of a logistic regression model

The assumptions of a logistic regression model are as follows:

- The Response Variable is Binary
- The Observations are Independent
- There is No Multicollinearity Among Explanatory Variables
- There are No Extreme Outliers
- There is a Linear Relationship Between Explanatory Variables and the Logit of the Response Variable
- The Sample Size is Sufficiently Large

For each of these assumptions that are violated, the potential reliability of the logistic regression model decreases. Adherence to these assumptions can be measured via tests such as Box-Tidwell, checking for extreme outliers, and VIF (Zach, 2021).

2.2 Tool Selection

All code execution was carried out via Jupyter Lab, using Python 3. I used Python as my selected programming language due to prior familiarity and broader applications when considering programming in general. R is a very strong and robust language tool for data analysis and statistics but finds itself somewhat limited to that niche role (Insights for Professionals, 2019). I utilized the NumPy, Pandas, and Matplotlib libraries to perform many of my data analysis tasks, as they are among the most popular Python libraries employed for this purpose and see widespread use. Seaborn is included primarily for its better-looking boxplots, seen later in this document (Parra, 2021).

Beyond these libraries, I relied upon the Statsmodels library. Statsmodels is one of several Python libraries that support linear and logistic regression. I am most familiar with it due to the course material's heavy reliance upon it. I also used the confusion_matrix and accuracy_score functions from scikit-learn's metrics module.

[55]:
```
# Imports and housekeeping
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import logit
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.mosaicplot import mosaic
from statsmodels.genmod import families
from statsmodels.genmod.generalized_linear_model import GLM
from sklearn.metrics import (confusion_matrix, accuracy_score)
sns.set_theme(style="darkgrid")
```

2.3 Why Logistic Regression?

Like linear regression, logistic regression is used to understand the relationship between one or more independent variables and a single dependent variable. Where logistic regression differs is in the type of prediction being made; where linear regression better helps us predict measurements, logistic regression helps predict whether or not an event will occur or a particular choice will be made. It works best when used with a dependent variable that has an "either/or" or "yes/no" response. Utilizing multiple independent variables in a predictive model can make our predictions stronger and allows higher conviction in the reliance on those models for decision making.

---

3 Part III: Data Preparation

3.1 Data Preparation Goals and Data Manipulations

I would like my data to include only variables relevant to my research question, and to be clean and free of missing values and duplicate rows. It will also be important to re-express any categorical variable types with numeric values. My first steps will be to import the complete data set and execute functions that will give me information on its size, the data types of its variables, and a peek at the data in table form. I will then narrow the data set to a new dataframe containing only the variables I am concerned with, and then utilize functions to determine if any null values or duplicate rows exist.

[56]:
```
# Import the main dataset
df = pd.read_csv('churn_clean.csv',dtype={'locationid':np.int64})
```

[57]:
```
# Display dataset info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Customer_id           10000 non-null  object
 2   Interaction           10000 non-null  object
 3   UID                   10000 non-null  object
 4   City                  10000 non-null  object
 5   State                 10000 non-null  object
 6   County                10000 non-null  object
 7   Zip                   10000 non-null  int64
 8   Lat                   10000 non-null  float64
 9   Lng                   10000 non-null  float64
 10  Population            10000 non-null  int64
 11  Area                  10000 non-null  object
 12  TimeZone              10000 non-null  object
 13  Job                   10000 non-null  object
 14  Children              10000 non-null  int64
 15  Age                   10000 non-null  int64
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object
 18  Gender                10000 non-null  object
 19  Churn                 10000 non-null  object
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64
 22  Contacts              10000 non-null  int64
 23  Yearly_equip_failure  10000 non-null  int64
 24  Techie                10000 non-null  object
```

```
 25  Contract              10000 non-null  object
 26  Port_modem            10000 non-null  object
 27  Tablet                10000 non-null  object
 28  InternetService       10000 non-null  object
 29  Phone                 10000 non-null  object
 30  Multiple              10000 non-null  object
 31  OnlineSecurity        10000 non-null  object
 32  OnlineBackup          10000 non-null  object
 33  DeviceProtection      10000 non-null  object
 34  TechSupport           10000 non-null  object
 35  StreamingTV           10000 non-null  object
 36  StreamingMovies       10000 non-null  object
 37  PaperlessBilling      10000 non-null  object
 38  PaymentMethod         10000 non-null  object
 39  Tenure                10000 non-null  float64
 40  MonthlyCharge         10000 non-null  float64
 41  Bandwidth_GB_Year     10000 non-null  float64
 42  Item1                 10000 non-null  int64
 43  Item2                 10000 non-null  int64
 44  Item3                 10000 non-null  int64
 45  Item4                 10000 non-null  int64
 46  Item5                 10000 non-null  int64
 47  Item6                 10000 non-null  int64
 48  Item7                 10000 non-null  int64
 49  Item8                 10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

[58]:
```
# Display dataset top 5 rows
df.head()
```

[58]:
```
   CaseOrder Customer_id                           Interaction  \
0          1     K409198  aa90260b-4141-4a24-8e36-b04ce1f4f77b
1          2     S120509  fb76459f-c047-4a9d-8af9-e0f7d4ac2524
2          3     K191035  344d114c-3736-4be5-98f7-c72c281e2d35
3          4      D90850  abfa2b40-2d43-4994-b15a-989b8c79e311
4          5     K662701  68a861fd-0d20-4e51-a587-8a90407ee574

                                UID         City State                 County  \
0  e885b299883d4f9fb18e39c75155d990  Point Baker    AK  Prince of Wales-Hyder
1  f2de8bef964785f41a2959829830fb8a  West Branch    MI                 Ogemaw
2  f1784cfa9f6d92ae816197eb175d3c71      Yamhill    OR                Yamhill
3  dc8a365077241bb5cd5ccd305136b05e      Del Mar    CA              San Diego
4  aabb64a116e83fdc4befc1fbab1663f9   Needville    TX              Fort Bend

     Zip       Lat        Lng  …  MonthlyCharge Bandwidth_GB_Year Item1  \
0  99927  56.25100 -133.37571  …     172.455519        904.536110     5
```

```
1  48661  44.32893  -84.24080  …     242.632554       800.982766     3
2  97148  45.35589 -123.24657  …     159.947583      2054.706961     4
3  92014  32.96687 -117.24798  …     119.956840      2164.579412     4
4  77461  29.38012  -95.80673  …     149.948316       271.493436     4

  Item2  Item3  Item4  Item5 Item6 Item7 Item8
0     5      5      3      4     4     3     4
1     4      3      3      4     3     4     4
2     4      2      4      4     3     3     3
3     4      4      2      5     4     3     3
4     4      4      3      4     4     4     5

[5 rows x 50 columns]
```

[59]:
```
# Trim dataset to variables relevant to research question
columns = ['Area', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'Churn',␣
 ↪'Outage_sec_perweek',
           'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',␣
 ↪'Bandwidth_GB_Year']
df_data = pd.DataFrame(df[columns])
```

[60]:
```
# Check data for null or missing values
df_data.isna().any()
```

[60]:
```
Area                    False
Children                False
Age                     False
Income                  False
Marital                 False
Gender                  False
Churn                   False
Outage_sec_perweek      False
Yearly_equip_failure    False
Tenure                  False
MonthlyCharge           False
Bandwidth_GB_Year       False
dtype: bool
```

[61]:
```
# Check data for duplicated rows
df_data.duplicated().sum()
```

[61]:
```
0
```

3.2 Summary Statistics

I can use the describe() function to display the summary statistics for the entire dataframe, as well as each variable I'll be evaluating for inclusion in the model. I have selected the Churn variable as my dependent variable.

I will also utilize histogram plots to illustrate the distribution of each numeric variable in the dataframe, and countplots for the categorical variables.

[62]:
```
# Display summary statistics for entire dataset - continuous variables
df_data.describe()
```

[62]:

	Children	Age	Income	Outage_sec_perweek
count	10000.0000	10000.000000	10000.000000	10000.000000
mean	2.0877	53.078400	39806.926771	10.001848
std	2.1472	20.698882	28199.916702	2.976019
min	0.0000	18.000000	348.670000	0.099747
25%	0.0000	35.000000	19224.717500	8.018214
50%	1.0000	53.000000	33170.605000	10.018560
75%	3.0000	71.000000	53246.170000	11.969485
max	10.0000	89.000000	258900.700000	21.207230

	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

[63]:
```
# Display summary statistics for entire dataset - categorical variables
df_data.describe(include = object)
```

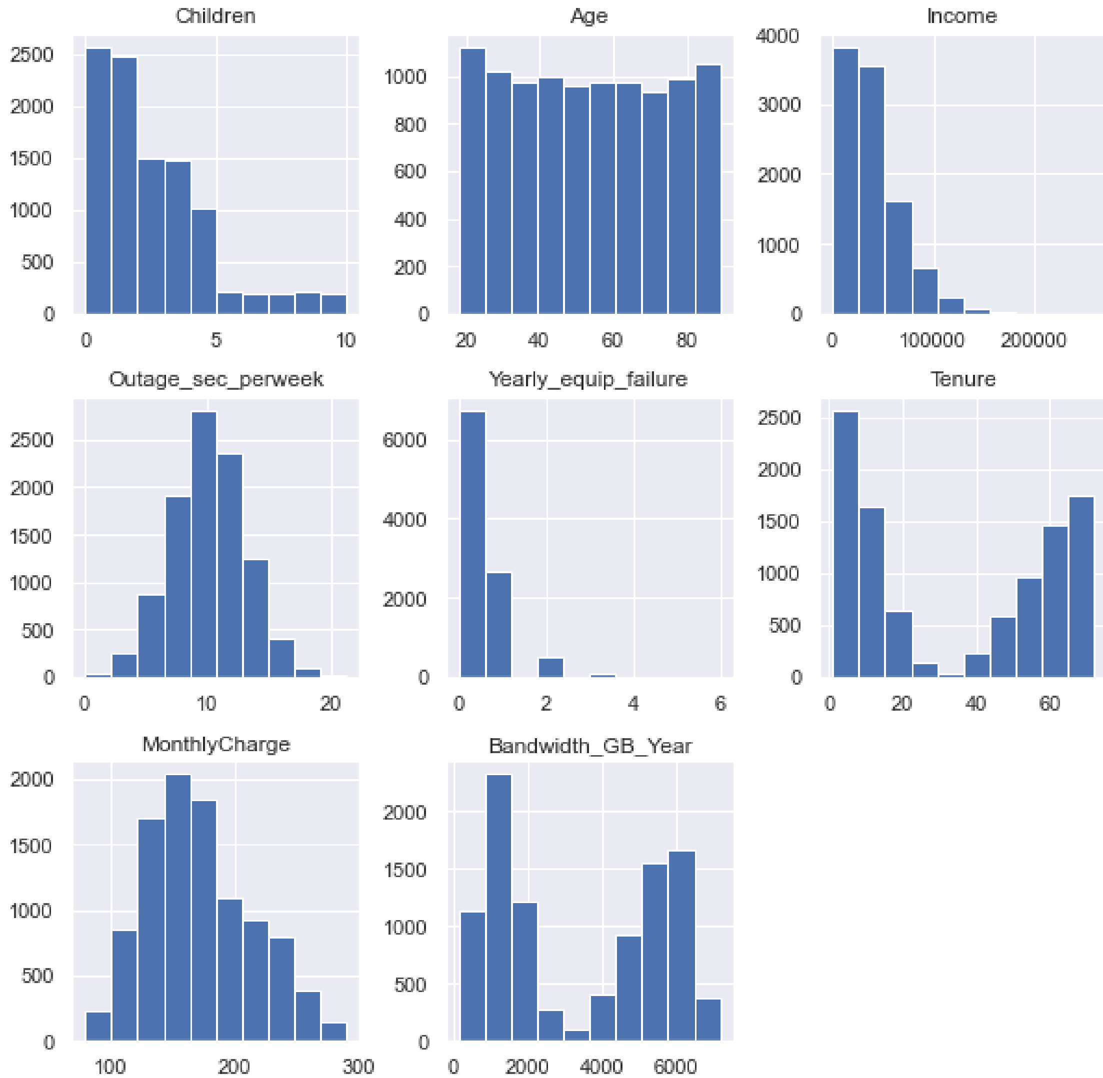
[63]:

	Area	Marital	Gender	Churn
count	10000	10000	10000	10000
unique	3	5	3	2
top	Suburban	Divorced	Female	No
freq	3346	2092	5025	7350

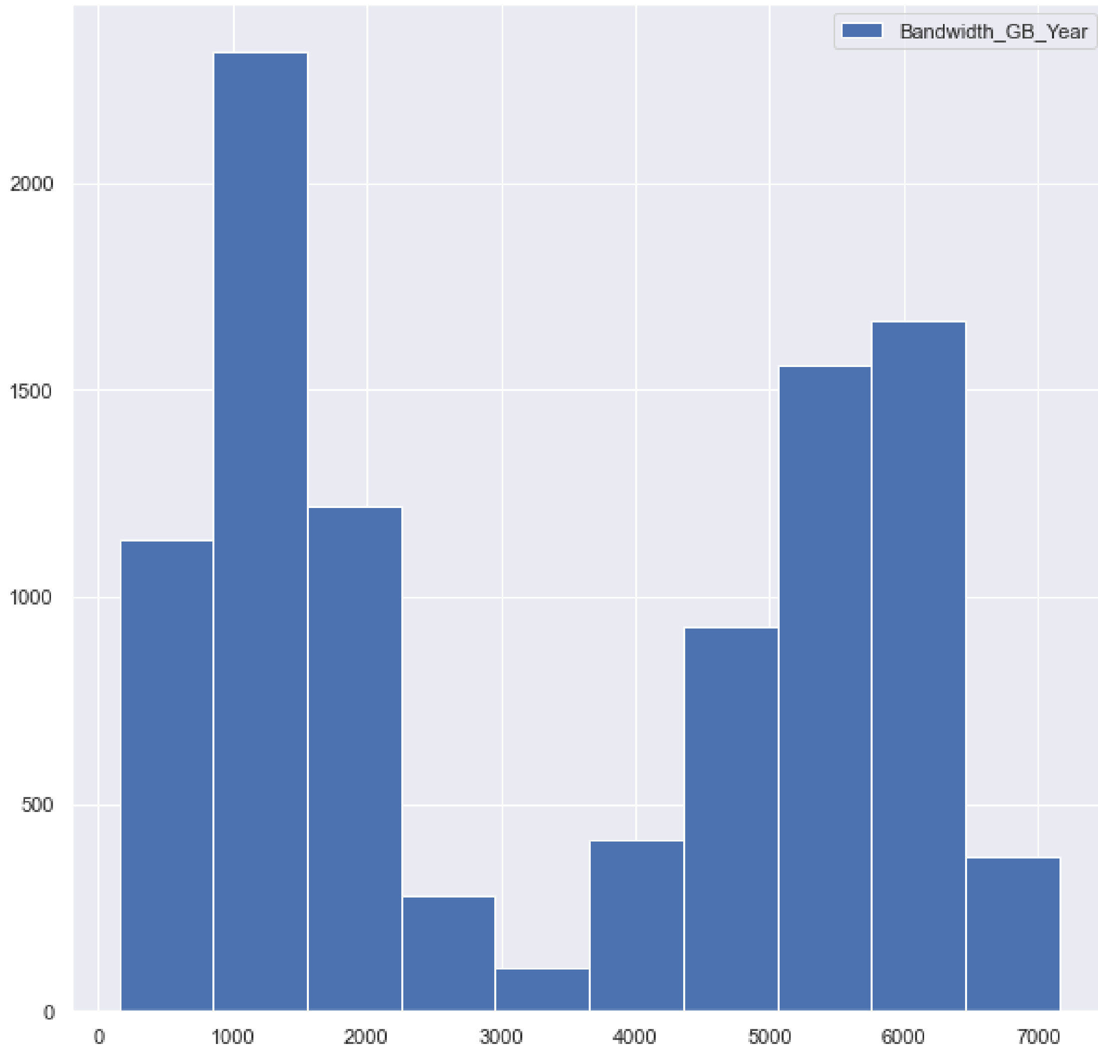
[64]:
```
# Initialize figure size settings
plt.rcParams['figure.figsize'] = [10, 10]
```

[65]:
```
# Display histogram plots for distribution of continuous variables
df_data.hist()
```

[65]:
```
array([[<AxesSubplot:title={'center':'Children'}>,
        <AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Income'}>],
       [<AxesSubplot:title={'center':'Outage_sec_perweek'}>,
        <AxesSubplot:title={'center':'Yearly_equip_failure'}>,
        <AxesSubplot:title={'center':'Tenure'}>],
       [<AxesSubplot:title={'center':'MonthlyCharge'}>,
        <AxesSubplot:title={'center':'Bandwidth_GB_Year'}>,
        <AxesSubplot:>]], dtype=object)
```

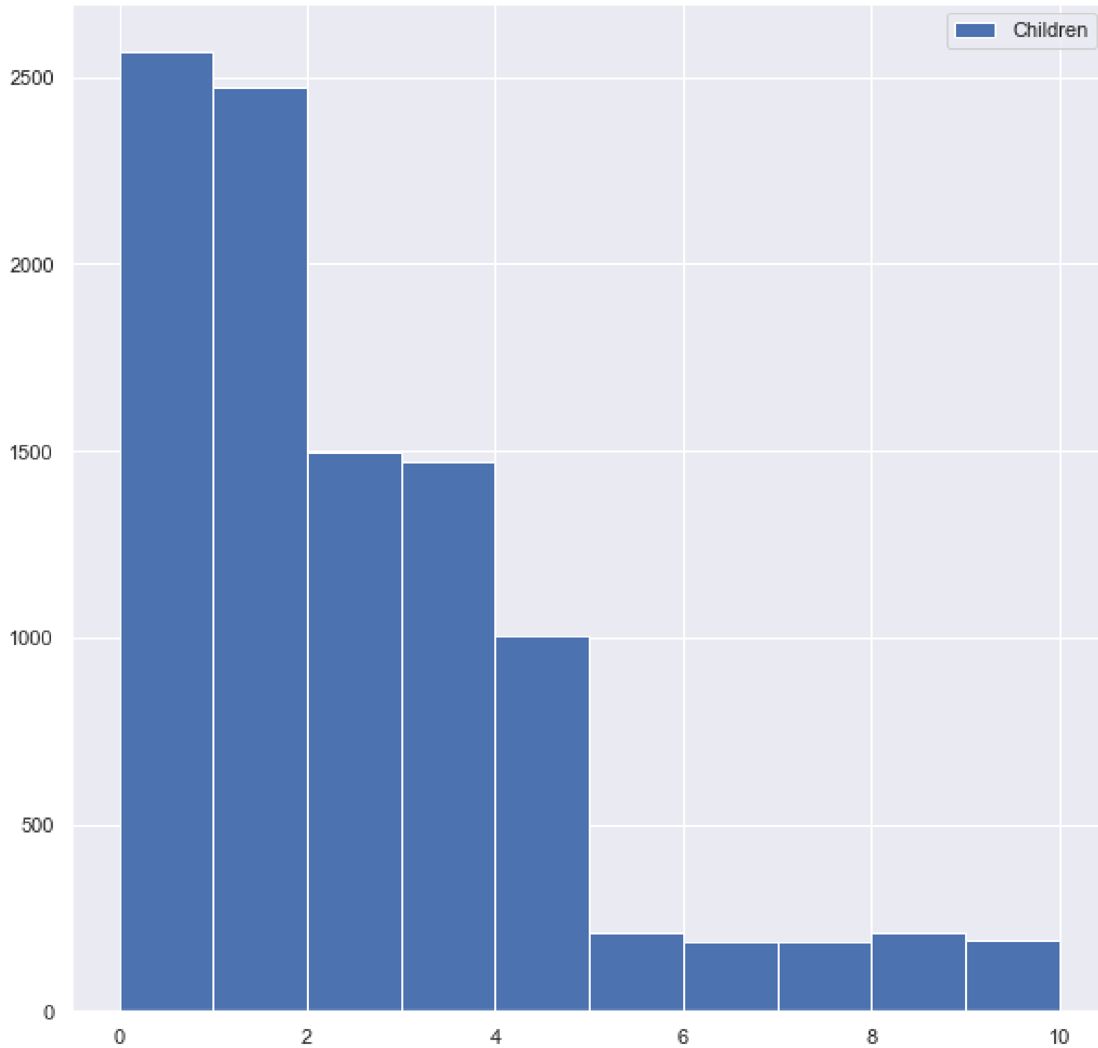


[66]:
```
# Display histogram plot and summary statistics for Bandwidth_GB_Year
df_data['Bandwidth_GB_Year'].hist(legend = True)
plt.show()
df_data['Bandwidth_GB_Year'].describe()
```



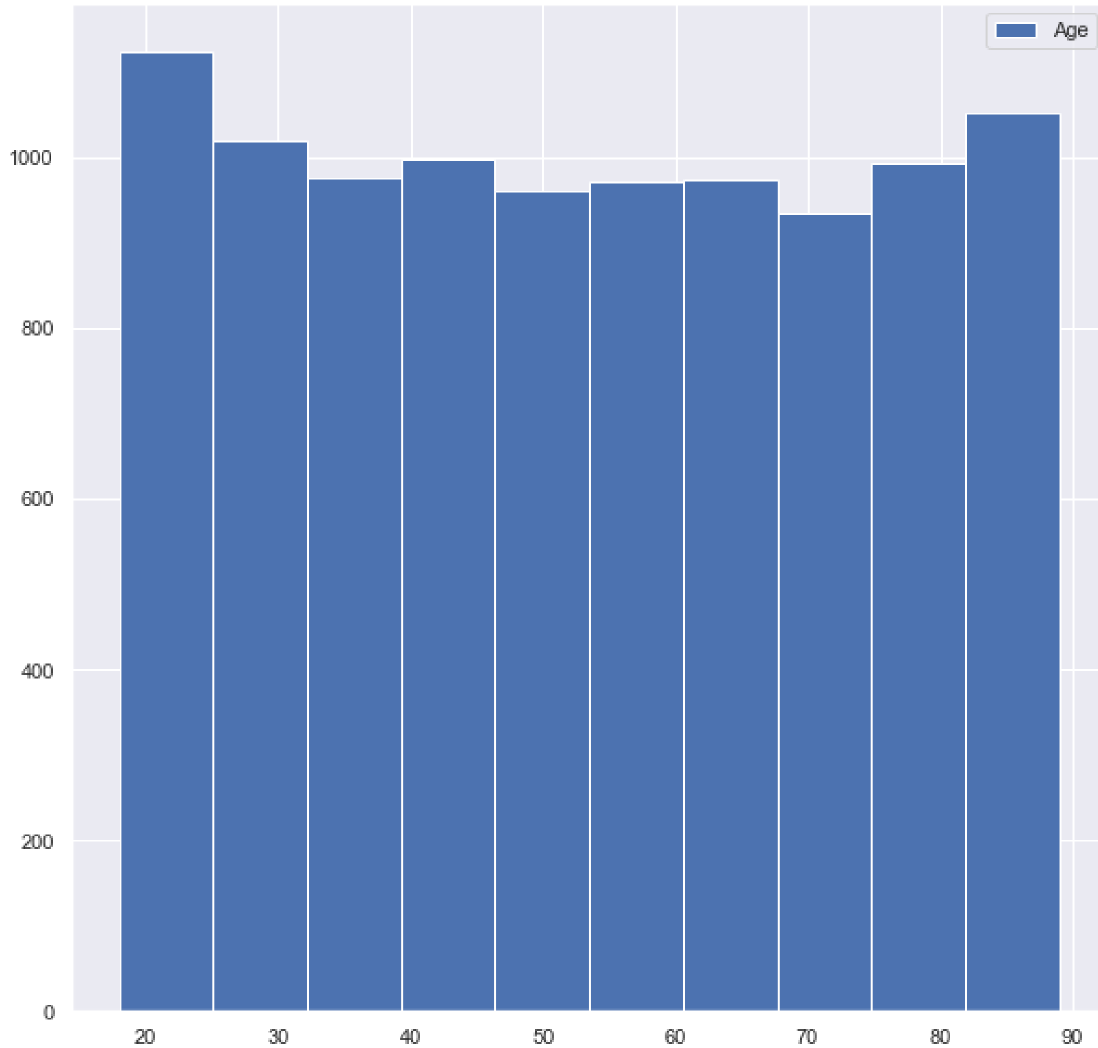
[66]:
```
count    10000.000000
mean      3392.341550
std       2185.294852
min        155.506715
25%       1236.470827
50%       3279.536903
75%       5586.141370
max       7158.981530
Name: Bandwidth_GB_Year, dtype: float64
```

[67]:
```
# Display histogram plot and summary statistics for Children
df_data['Children'].hist(legend = True)
plt.show()
df_data['Children'].describe()
```



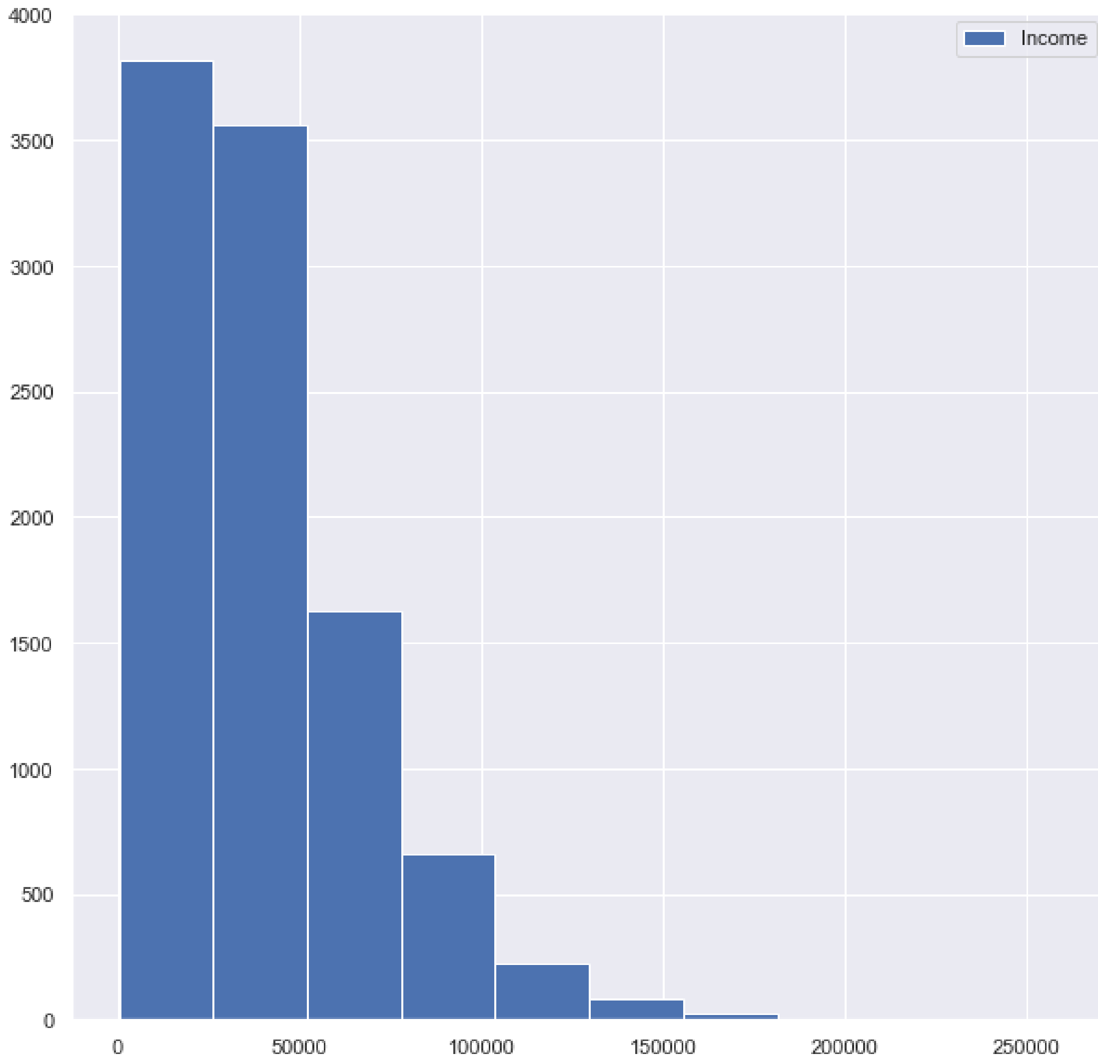
[67]:
```
count    10000.0000
mean         2.0877
std          2.1472
min          0.0000
25%          0.0000
50%          1.0000
75%          3.0000
max         10.0000
Name: Children, dtype: float64
```

[68]:
```
# Display histogram plot and summary statistics for Age
df_data['Age'].hist(legend = True)
plt.show()
df_data['Age'].describe()
```



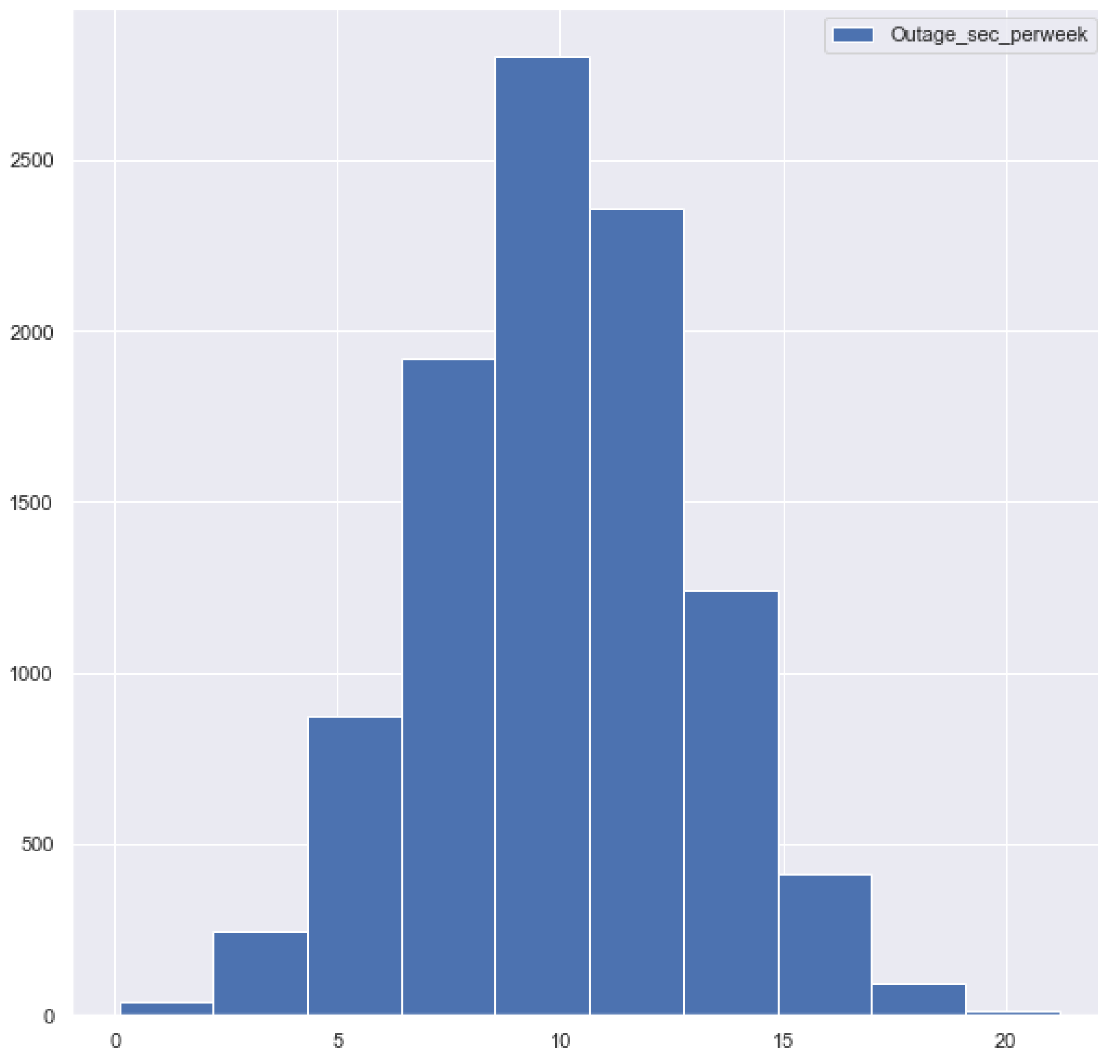
[68]:
```
count    10000.000000
mean        53.078400
std         20.698882
min         18.000000
25%         35.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

[69]:
```
# Display histogram plot and summary statistics for Income
df_data['Income'].hist(legend = True)
plt.show()
df_data['Income'].describe()
```



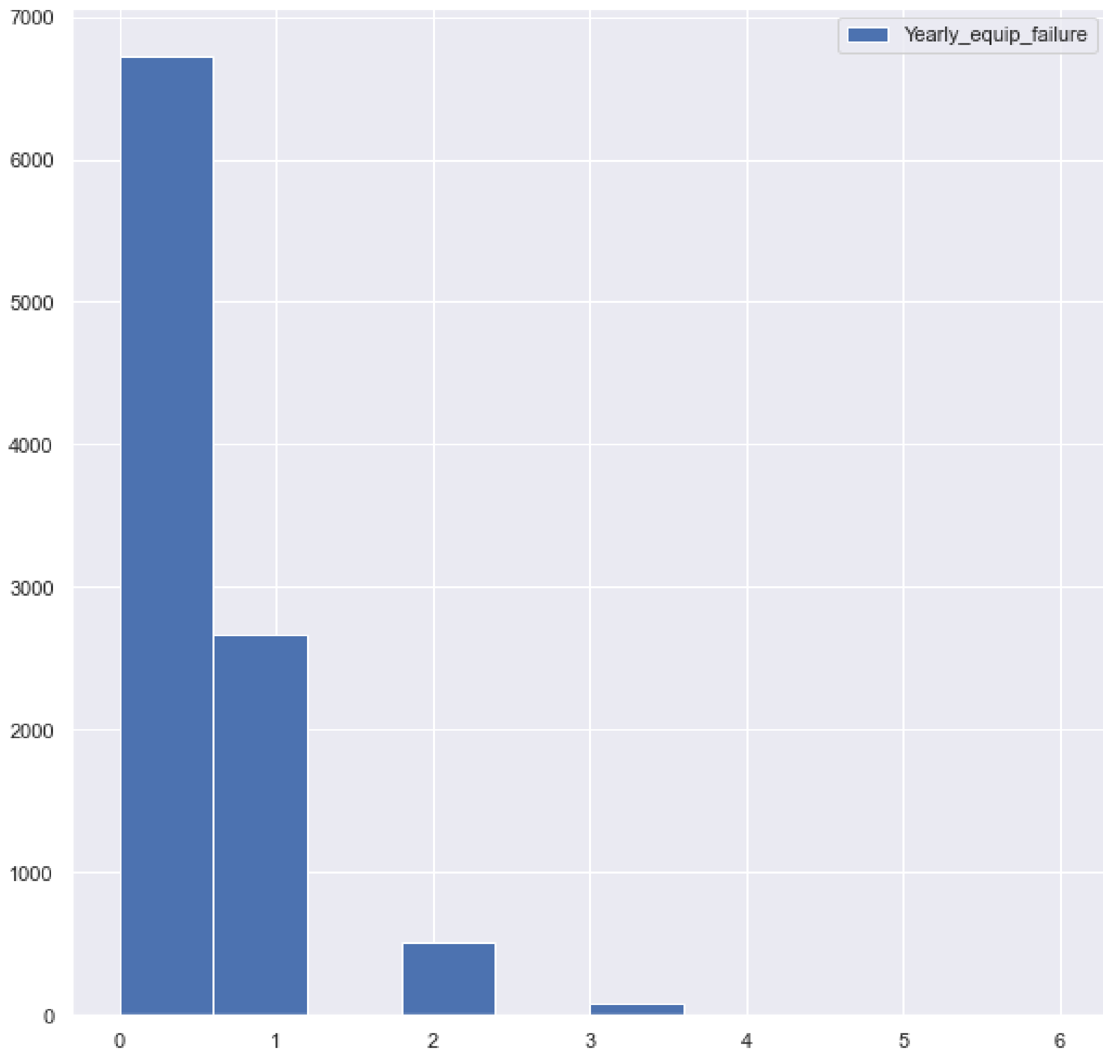
[69]:
```
count      10000.000000
mean       39806.926771
std        28199.916702
min          348.670000
25%        19224.717500
50%        33170.605000
75%        53246.170000
max       258900.700000
Name: Income, dtype: float64
```

[70]:
```python
# Display histogram plot and summary statistics for Outage_sec_perweek
df_data['Outage_sec_perweek'].hist(legend = True)
plt.show()
df_data['Outage_sec_perweek'].describe()
```



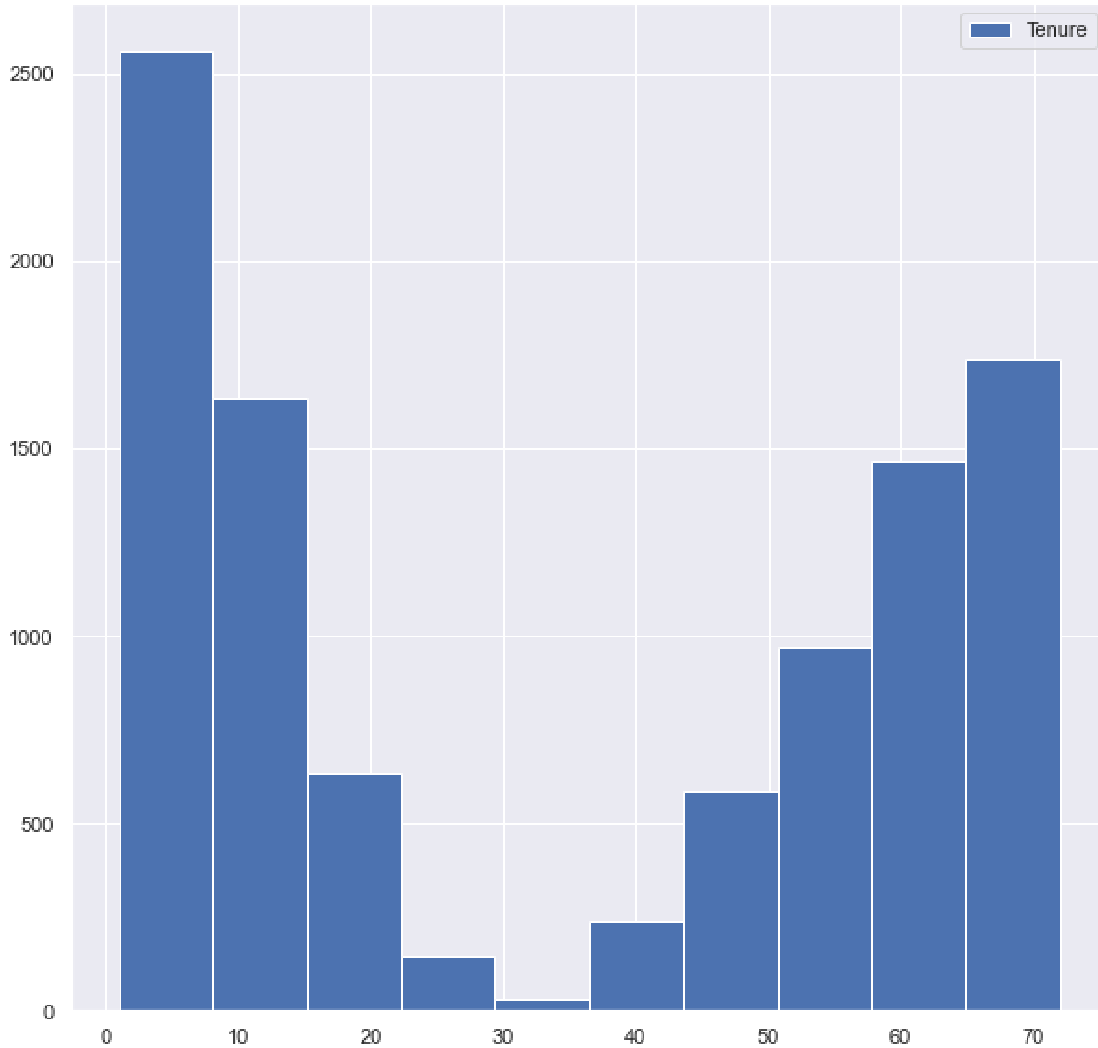
[70]:
```
count    10000.000000
mean        10.001848
std          2.976019
min          0.099747
25%          8.018214
50%         10.018560
75%         11.969485
max         21.207230
Name: Outage_sec_perweek, dtype: float64
```

[71]:
```
# Display histogram plot and summary statistics for Yearly_equip_failure
df_data['Yearly_equip_failure'].hist(legend = True)
plt.show()
df_data['Yearly_equip_failure'].describe()
```



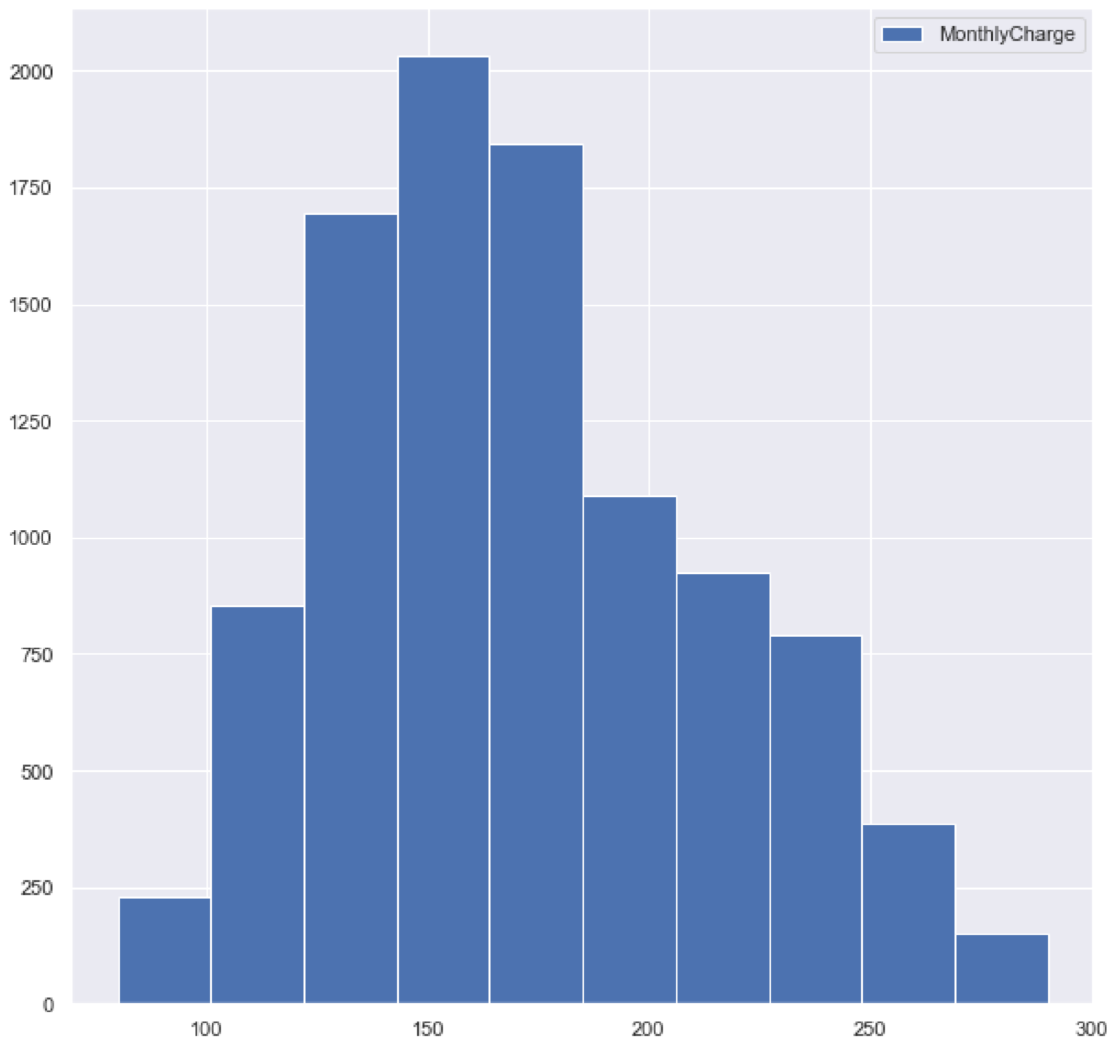
[71]:
```
count    10000.000000
mean         0.398000
std          0.635953
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          6.000000
Name: Yearly_equip_failure, dtype: float64
```

[72]:
```
# Display histogram plot and summary statistics for Tenure
df_data['Tenure'].hist(legend = True)
plt.show()
df_data['Tenure'].describe()
```



[72]:
```
count    10000.000000
mean        34.526188
std         26.443063
min          1.000259
25%          7.917694
50%         35.430507
75%         61.479795
max         71.999280
Name: Tenure, dtype: float64
```

[73]:
```
# Display histogram plot and summary statistics for MonthlyCharge
df_data['MonthlyCharge'].hist(legend = True)
plt.show()
df_data['MonthlyCharge'].describe()
```



[73]:

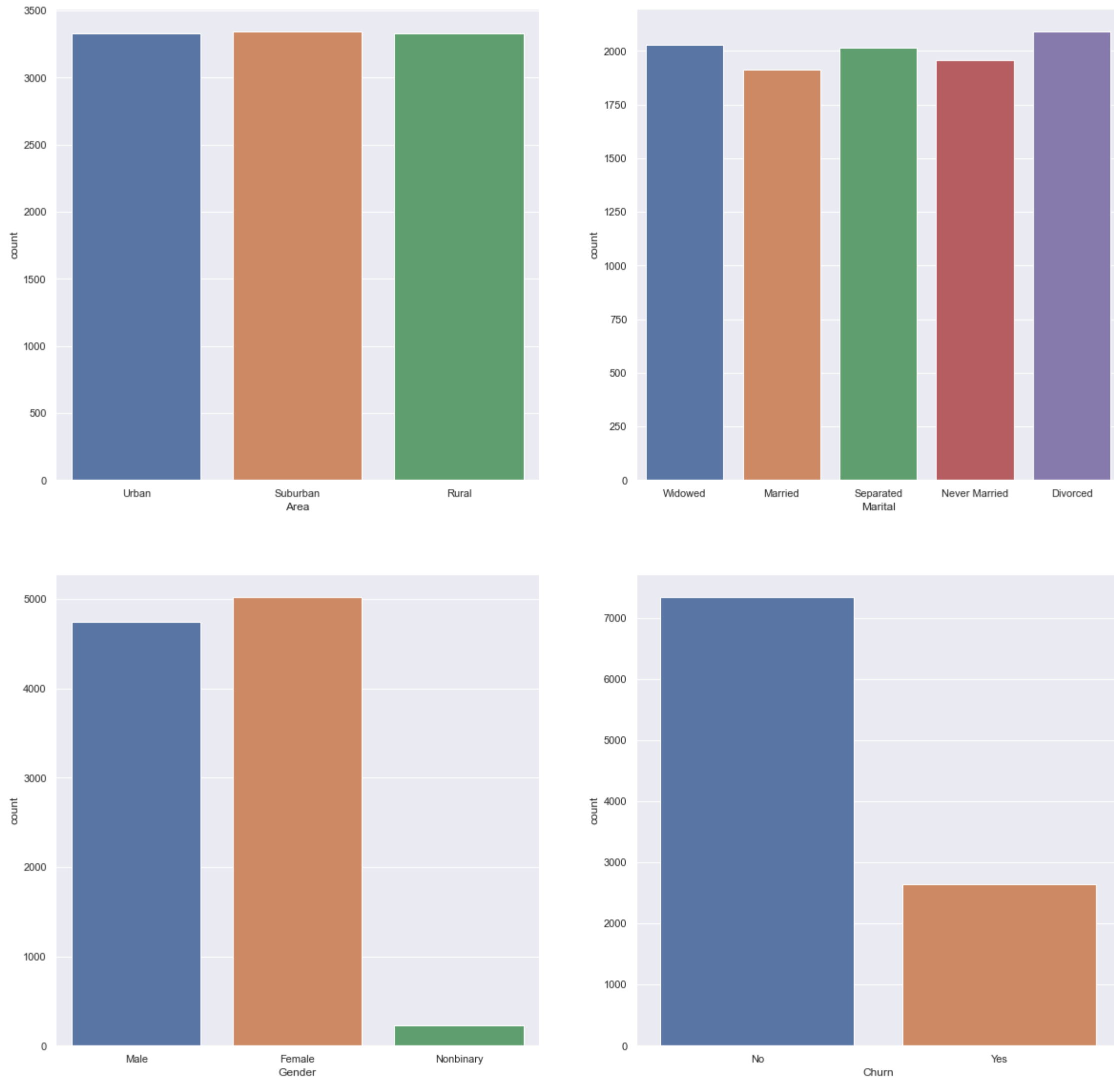
```
count    10000.000000
mean       172.624816
std         42.943094
min         79.978860
25%        139.979239
50%        167.484700
75%        200.734725
max        290.160419
Name: MonthlyCharge, dtype: float64
```

[74]:

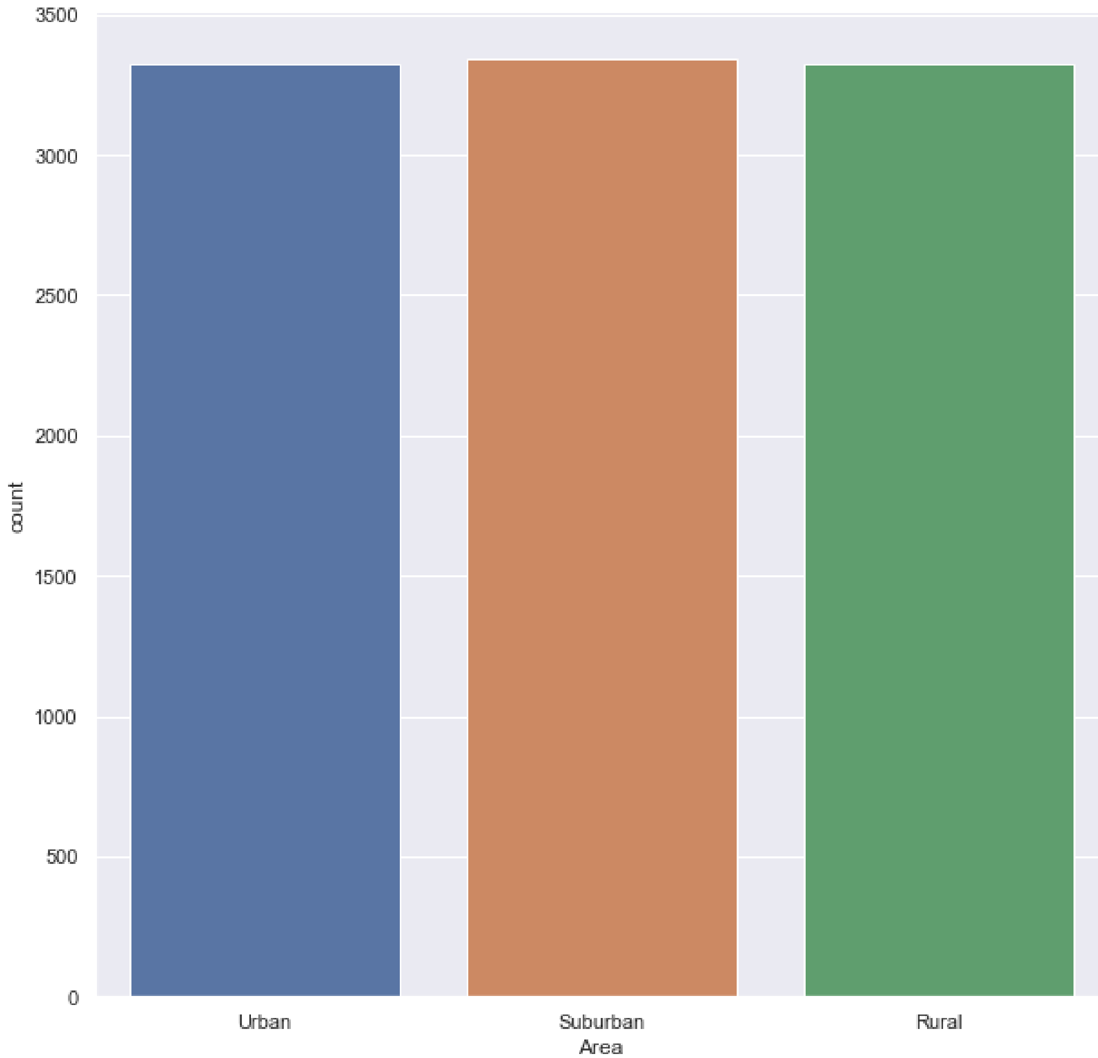
```
# Display countplots for distribution of categorical variables
fig, ax = plt.subplots(figsize = (20,20), ncols = 2, nrows = 2)
sns.countplot(x='Area', data=df_data, ax = ax[0][0])
sns.countplot(x='Marital', data=df_data, ax = ax[0][1])
sns.countplot(x='Gender', data=df_data, ax = ax[1][0])
```

```
sns.countplot(x='Churn', data=df_data, ax = ax[1][1])
```

[74]: <AxesSubplot:xlabel='Churn', ylabel='count'>

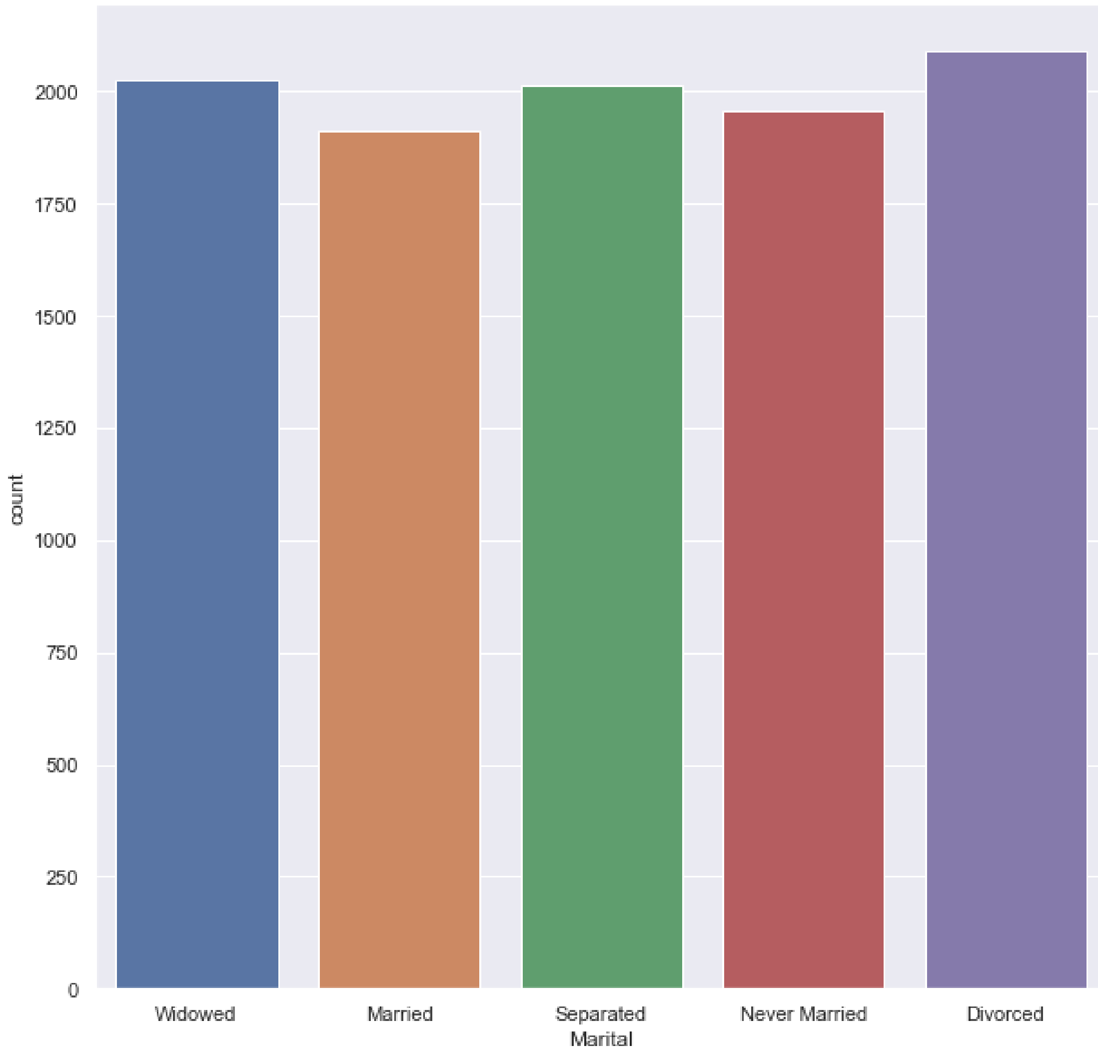


[75]:
```
# Display countplot and summary statistics for Area
sns.countplot(x='Area', data=df_data)
plt.show()
df_data['Area'].describe()
```



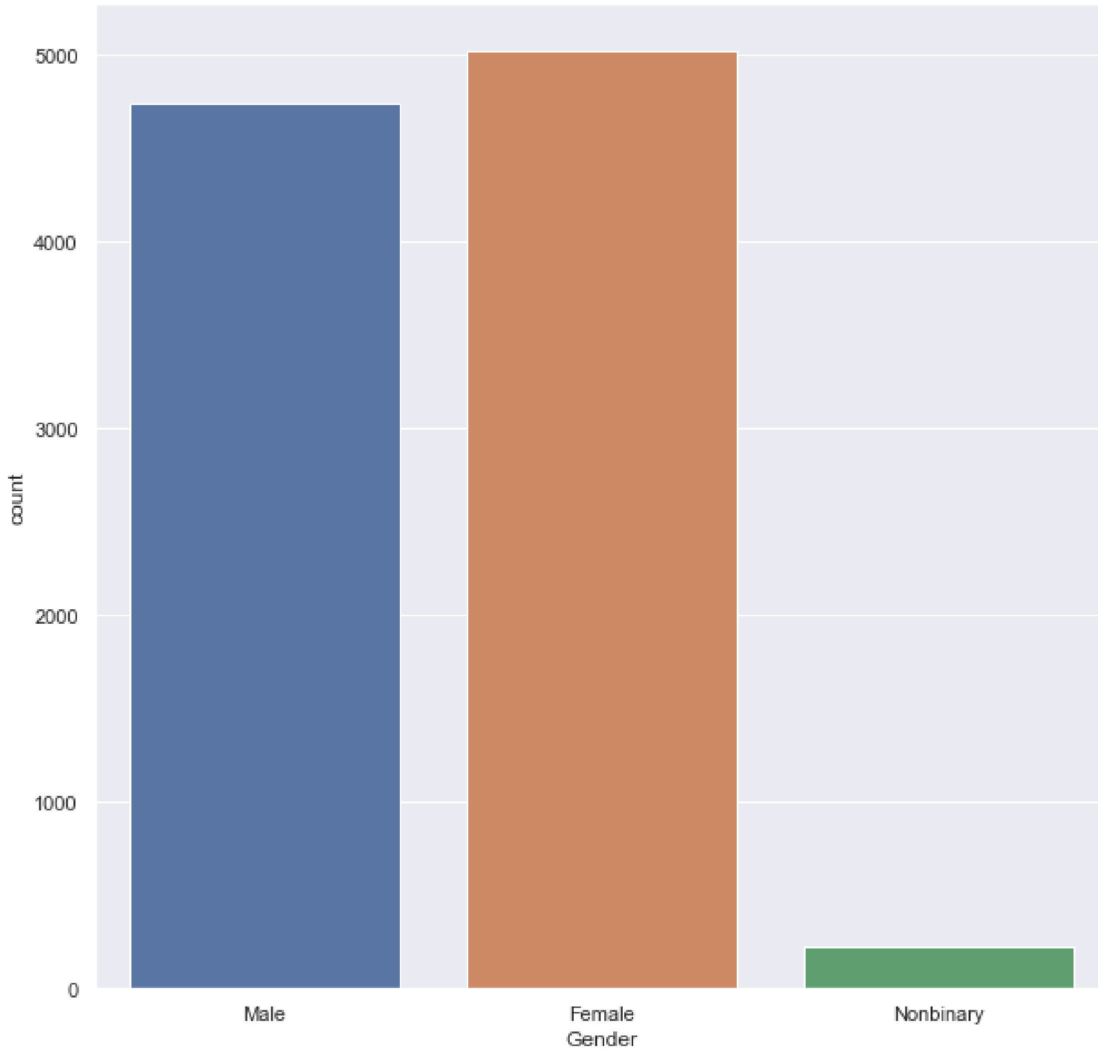

```
[75]: count          10000
      unique             3
      top         Suburban
      freq            3346
      Name: Area, dtype: object
```

```
[76]: # Display countplot and summary statistics for Marital
      sns.countplot(x='Marital', data=df_data)
      plt.show()
      df_data['Marital'].describe()
```



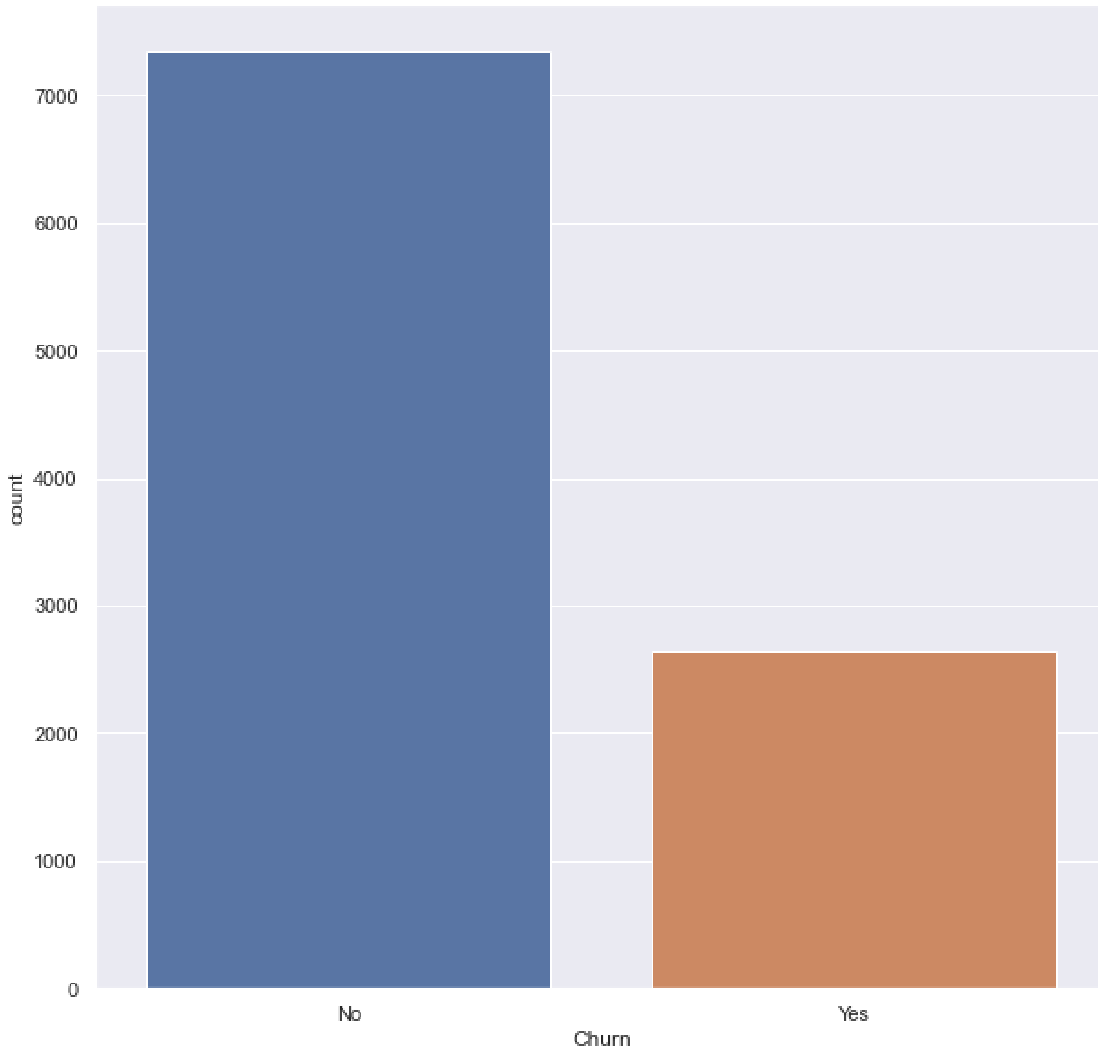
[76]:
```
count           10000
unique              5
top          Divorced
freq             2092
Name: Marital, dtype: object
```

[77]:
```
# Display countplot and summary statistics for Gender
sns.countplot(x='Gender', data=df_data)
plt.show()
df_data['Gender'].describe()
```



[77]:
```
count      10000
unique         3
top       Female
freq        5025
Name: Gender, dtype: object
```

[78]:
```
# Display countplot and summary statistics for Churn
sns.countplot(x='Churn', data=df_data)
plt.show()
df_data['Churn'].describe()
```




```
[78]: count      10000
      unique         2
      top           No
      freq        7350
      Name: Churn, dtype: object
```

3.3 Further Preparation Steps

I will make some adjustments to my data types to make my variables easier to work with. Conversion of "object" types as "category" in particular will lend itself to a more efficient conversion of categorical variables to numeric.

[79]:
```
# Reassign data types
for col in df_data:
    if df_data[col].dtypes == 'object':
        df_data[col] = df_data[col].astype('category')
    if df_data[col].dtypes == 'int64':
        df_data[col] = df_data[col].astype(int)
    if df_data[col].dtypes == 'float64':
        df_data[col] = df_data[col].astype(float)
```

[80]:
```
# Display dataset info and observe data type changes
df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Area                  10000 non-null  category
 1   Children              10000 non-null  int32
 2   Age                   10000 non-null  int32
 3   Income                10000 non-null  float64
 4   Marital               10000 non-null  category
 5   Gender                10000 non-null  category
 6   Churn                 10000 non-null  category
 7   Outage_sec_perweek    10000 non-null  float64
 8   Yearly_equip_failure  10000 non-null  int32
 9   Tenure                10000 non-null  float64
 10  MonthlyCharge         10000 non-null  float64
 11  Bandwidth_GB_Year     10000 non-null  float64
dtypes: category(4), float64(5), int32(3)
memory usage: 547.6 KB
```

Here I will use the cat.codes accessor to perform label encoding on my categorical variables.

[81]:
```
# Use cat.codes for label encoding of 4 categorical variables
df_data['Area_cat'] = df_data['Area'].cat.codes
df_data['Marital_cat'] = df_data['Marital'].cat.codes
df_data['Gender_cat'] = df_data['Gender'].cat.codes
df_data['Churn_cat'] = df_data['Churn'].cat.codes
```

[82]:
```
# Display dataset top 5 rows from label encoded variables
df_data[['Area', 'Marital', 'Gender', 'Churn', 'Area_cat', 'Marital_cat',␣
↪'Gender_cat', 'Churn_cat']].head()
```

[82]:
```
        Area    Marital  Gender Churn  Area_cat  Marital_cat  Gender_cat  \
0      Urban    Widowed    Male    No         2            4           1
```

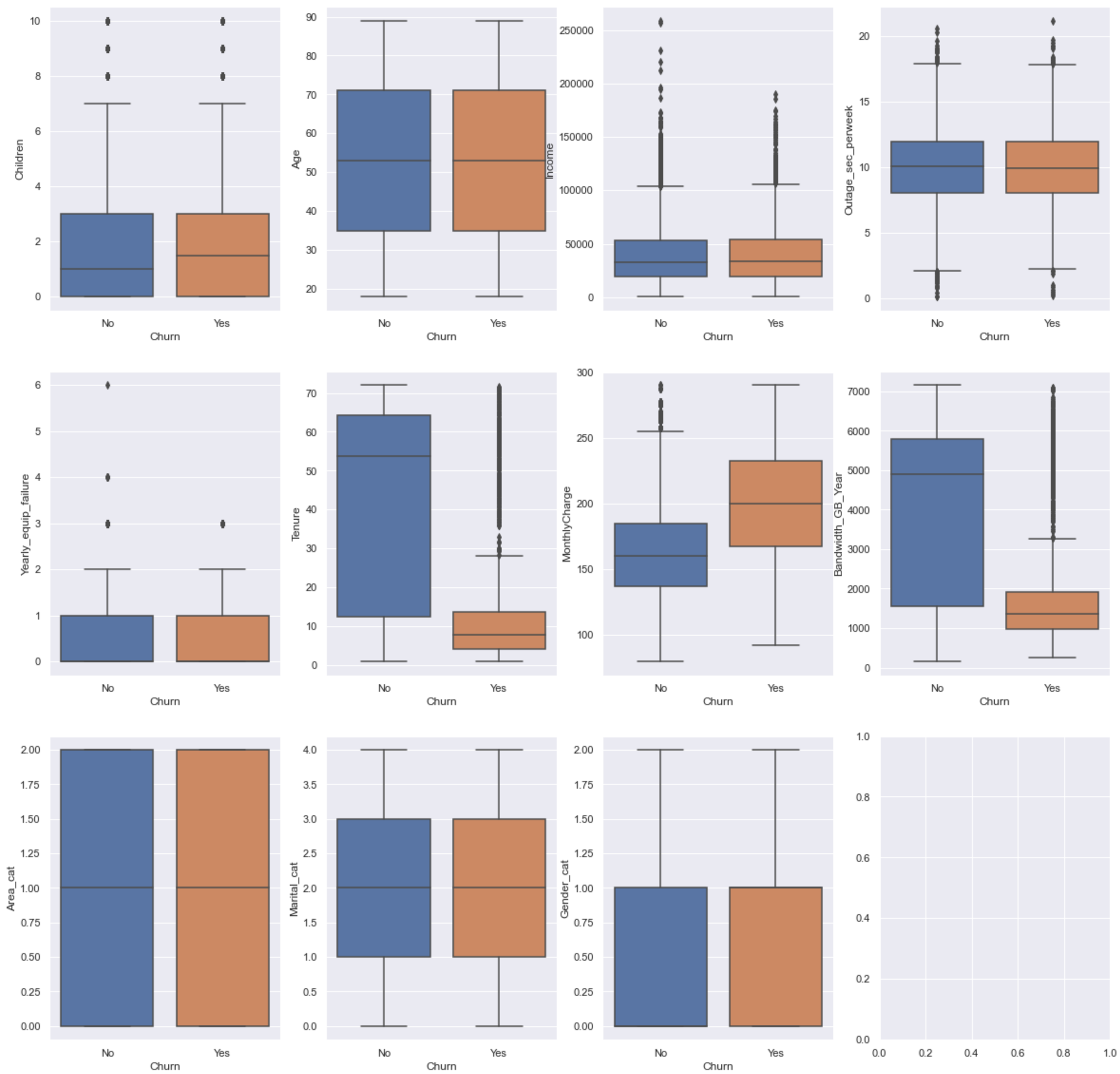
```
1     Urban    Married  Female   Yes          2             1            0
2     Urban    Widowed  Female    No          2             4            0
3  Suburban    Married    Male    No          1             1            1
4  Suburban  Separated    Male   Yes          1             3            1

   Churn_cat
0          0
1          1
2          0
3          0
4          1
```

3.4 Univariate and Bivariate Visualizations

Univariate analysis of each variable can be seen above in section 2 of part III, "Data Preparation". I will make use of Seaborn's boxplot() function for bivariate analysis of all variables. Each independent variable is paired against my dependent variable, "Churn".

[83]:
```
# Display boxplots for bivariate analysis of variables - dependent variable =␣
↪Churn
fig, ax = plt.subplots(figsize = (20, 20), ncols = 4, nrows = 3)
sns.boxplot(x = 'Churn', y = 'Children', data = df_data, ax = ax[0][0])
sns.boxplot(x = 'Churn', y = 'Age', data = df_data, ax = ax[0][1])
sns.boxplot(x = 'Churn', y = 'Income', data = df_data, ax = ax[0][2])
sns.boxplot(x = 'Churn', y = 'Outage_sec_perweek', data = df_data, ax =␣
↪ax[0][3])
sns.boxplot(x = 'Churn', y = 'Yearly_equip_failure', data = df_data, ax =␣
↪ax[1][0])
sns.boxplot(x = 'Churn', y = 'Tenure', data = df_data, ax = ax[1][1])
sns.boxplot(x = 'Churn', y = 'MonthlyCharge', data = df_data, ax = ax[1][2])
sns.boxplot(x = 'Churn', y = 'Bandwidth_GB_Year', data = df_data, ax = ax[1][3])
sns.boxplot(x = 'Churn', y = 'Area_cat', data = df_data, ax = ax[2][0])
sns.boxplot(x = 'Churn', y = 'Marital_cat', data = df_data, ax = ax[2][1])
sns.boxplot(x = 'Churn', y = 'Gender_cat', data = df_data, ax = ax[2][2])
```

[83]:
```
<AxesSubplot:xlabel='Churn', ylabel='Gender_cat'>
```



3.5 Copy of Prepared Data Set

Below is the code used to export the prepared data set to CSV format.

[84]:
```
# Export prepared dataframe to csv
df_data.to_csv(r'C:\Users\wstul\d208\churn_clean_perpared.csv')
```

---

4 Part IV: Model Comparison and Analysis

4.1 Initial Logistic Regression Model

Below I will create an initial logistic regression model and display its summary info.

[85]:
```
# Create initial model and display summary
mdl_churn_vs_all = logit("Churn_cat ~ Area_cat + Children + Age + Income +␣
↪Marital_cat + Gender_cat + Bandwidth_GB_Year + \
                        Outage_sec_perweek + Yearly_equip_failure +␣
↪MonthlyCharge + Tenure", data=df_data).fit()
print(mdl_churn_vs_all.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.319631
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:              Churn_cat   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9988
Method:                           MLE   Df Model:                           11
Date:                Mon, 16 May 2022   Pseudo R-squ.:                  0.4472
Time:                        16:11:55   Log-Likelihood:                -3196.3
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
========
                           coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
Intercept               -5.9769      0.228    -26.185      0.000      -6.424
-5.530
Area_cat                 0.0027      0.038      0.072      0.943      -0.073
0.078
Children                -0.0987      0.016     -6.365      0.000      -0.129
-0.068
Age                      0.0115      0.002      7.171      0.000       0.008
0.015
Income                5.129e-07   1.11e-06      0.460      0.645   -1.67e-06
2.7e-06
Marital_cat              0.0402      0.022      1.829      0.067      -0.003
0.083
Gender_cat              -0.0458      0.058     -0.790      0.429      -0.159
0.068
Bandwidth_GB_Year        0.0029      0.000     20.149      0.000       0.003
0.003
Outage_sec_perweek       0.0002      0.011      0.019      0.985      -0.020
```

```
0.021
Yearly_equip_failure    -0.0255      0.050     -0.514      0.607      -0.123
0.072
MonthlyCharge            0.0261      0.001     27.270      0.000       0.024
0.028
Tenure                  -0.3171      0.012    -25.436      0.000      -0.342
-0.293
================================================================================
========
```

4.2 Reducing the Initial Model

Starting from this initial model, I will aim to reduce the model by eliminating variables not suitable for this logistic regression, using statistical analysis in my selection process.

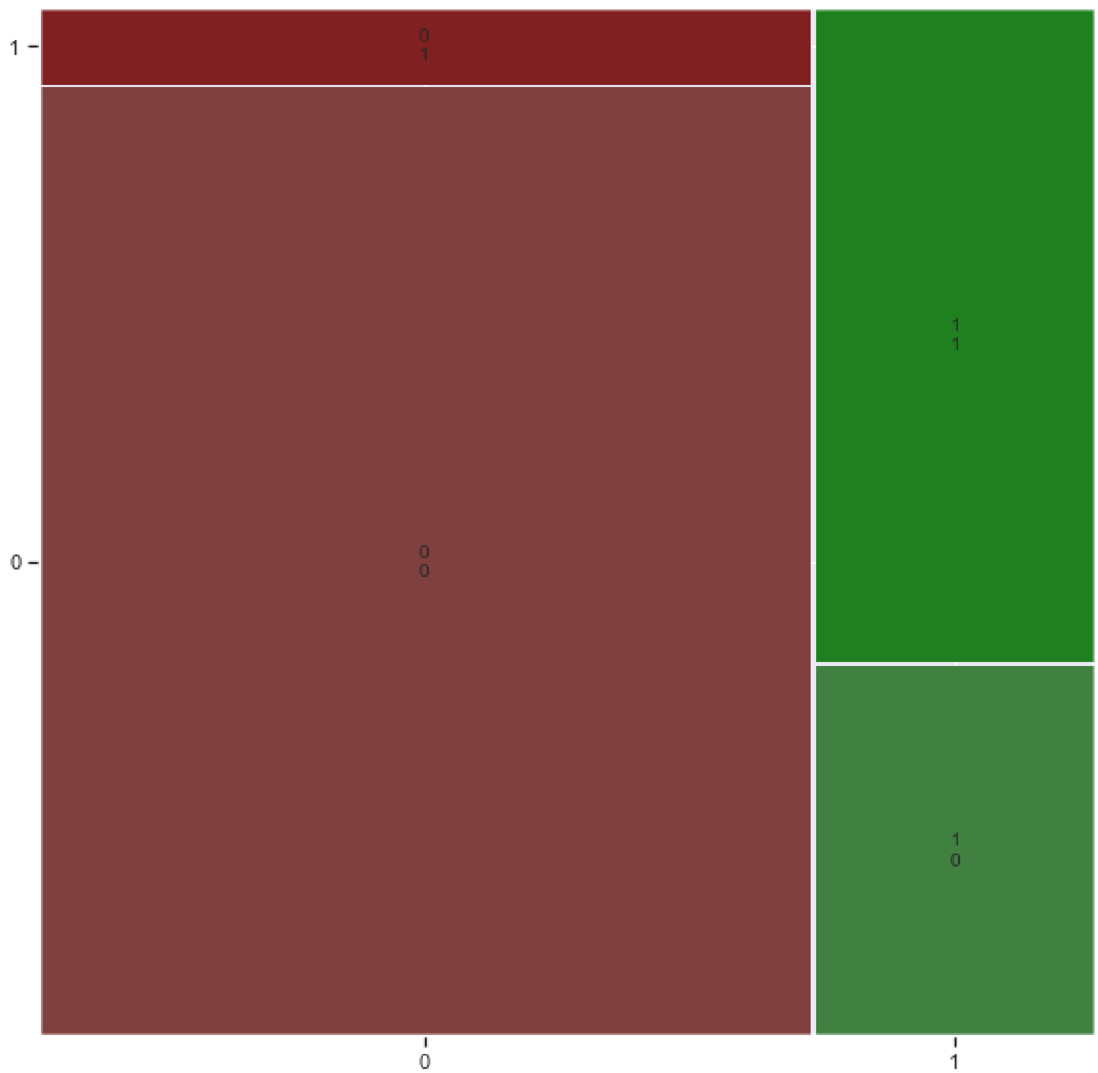
To begin I will look at some additional metrics for the current model.

[86]:
```
# defining the dependent and independent variables
Xtest = df_data[['Area_cat', 'Children', 'Age', 'Income', 'Marital_cat',␣
↪'Gender_cat', 'Outage_sec_perweek',
            'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',␣
↪'Bandwidth_GB_Year']]
ytest = df_data['Churn_cat']
# performing predictions on the test datdaset
yhat = mdl_churn_vs_all.predict(Xtest)
prediction = list(map(round, yhat))
# confusion matrix
conf_matrix = confusion_matrix(ytest, prediction)
print ("Confusion Matrix : \n", conf_matrix)
# accuracy score of the model
print('Test accuracy = ', accuracy_score(ytest, prediction))
# confusion matrix visualized
mosaic(conf_matrix)
```

```
Confusion Matrix :
 [[6794  556]
 [ 955 1695]]
Test accuracy =  0.8489
```

[86]:
```
(<Figure size 720x720 with 3 Axes>,
 {('0', '0'): (0.0, 0.0, 0.7313432835820897, 0.9212827988338191),
  ('0', '1'): (0.0,
   0.9246050579700318,
   0.7313432835820897,
   0.0753949420299681),
  ('1', '0'): (0.7363184079601991, 0.0, 0.263681592039801, 0.3591800915188365),
  ('1', '1'): (0.7363184079601991,
```

```
0.36250235065504915,
0.263681592039801,
0.6374976493449507)})
```



As I proceed through my reduction process, I will aim to keep the test accuracy close to the initial model's performance while minimizing additional false positives and negatives. Higher accuracy scores are considered better, with 1.000 being the maximum.

First I will generate a correlation table as a reference during the selection process, and perform a variance inflation factor analysis for all features currently in the model.

[87]:
```
df_data.corr()
```

[87]:

```
                      Children       Age    Income  Outage_sec_perweek  \
Children              1.000000 -0.029732  0.009942            0.001889
Age                  -0.029732  1.000000 -0.004091           -0.008047
Income                0.009942 -0.004091  1.000000           -0.010011
Outage_sec_perweek    0.001889 -0.008047 -0.010011            1.000000
Yearly_equip_failure  0.007321  0.008577  0.005423            0.002909
Tenure               -0.005091  0.016979  0.002114            0.002932
MonthlyCharge        -0.009781  0.010729 -0.003014            0.020496
Bandwidth_GB_Year     0.025585 -0.014724  0.003674            0.004176
Area_cat             -0.007879  0.011745  0.002557            0.000239
Marital_cat           0.000045 -0.009721 -0.005045           -0.016180
Gender_cat            0.006032 -0.005660 -0.018436            0.008887
Churn_cat            -0.004264  0.005630  0.005937           -0.000156

                      Yearly_equip_failure    Tenure  MonthlyCharge  \
Children                          0.007321 -0.005091      -0.009781
Age                               0.008577  0.016979       0.010729
Income                            0.005423  0.002114      -0.003014
Outage_sec_perweek                0.002909  0.002932       0.020496
Yearly_equip_failure              1.000000  0.012435      -0.007172
Tenure                            0.012435  1.000000      -0.003337
MonthlyCharge                    -0.007172 -0.003337       1.000000
Bandwidth_GB_Year                 0.012034  0.991495       0.060406
Area_cat                         -0.006554 -0.016615       0.003951
Marital_cat                       0.001183  0.003241      -0.002266
Gender_cat                        0.014750 -0.016051       0.009147
Churn_cat                        -0.015927 -0.485475       0.372938

                      Bandwidth_GB_Year  Area_cat  Marital_cat  Gender_cat  \
Children                       0.025585 -0.007879     0.000045    0.006032
Age                           -0.014724  0.011745    -0.009721   -0.005660
Income                         0.003674  0.002557    -0.005045   -0.018436
Outage_sec_perweek             0.004176  0.000239    -0.016180    0.008887
Yearly_equip_failure           0.012034 -0.006554     0.001183    0.014750
Tenure                         0.991495 -0.016615     0.003241   -0.016051
MonthlyCharge                  0.060406  0.003951    -0.002266    0.009147
Bandwidth_GB_Year              1.000000 -0.016575     0.001499   -0.001469
Area_cat                      -0.016575  1.000000     0.013733    0.004057
Marital_cat                    0.001499  0.013733     1.000000   -0.008360
Gender_cat                    -0.001469  0.004057    -0.008360    1.000000
Churn_cat                     -0.441669  0.014166     0.012716    0.023919

                      Churn_cat
Children              -0.004264
Age                    0.005630
Income                 0.005937
Outage_sec_perweek    -0.000156
```

```
Yearly_equip_failure  -0.015927
Tenure                -0.485475
MonthlyCharge          0.372938
Bandwidth_GB_Year     -0.441669
Area_cat               0.014166
Marital_cat            0.012716
Gender_cat             0.023919
Churn_cat              1.000000
```

[88]:
```
# Perform variance inflation factor analysis for initial feature set
X = df_data[['Area_cat', 'Children', 'Age', 'Income', 'Marital_cat',␣
 ↪'Gender_cat', 'Outage_sec_perweek',
            'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',␣
 ↪'Bandwidth_GB_Year']]
vif_data = pd.DataFrame()
vif_data['IndVar'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
 ↪columns))]
print(vif_data)
```

```
                  IndVar         VIF
0               Area_cat    2.425522
1               Children    2.072716
2                    Age    7.057506
3                 Income    2.850023
4            Marital_cat    2.810898
5             Gender_cat    1.916246
6     Outage_sec_perweek    9.240040
7   Yearly_equip_failure    1.381941
8                 Tenure  250.810141
9          MonthlyCharge   18.436436
10     Bandwidth_GB_Year  316.867066
```

Right away, I can see very high VIF scores for two variables, Tenure and Bandwidth_GB_Year. High VIF values (usually greater than 5-10) indicate a high degree of multicollinearity with other variables in the model. This reduces the model accuracy, so I will start by dropping one of these two variables from the set and repeat my VIF analysis.

As Tenure has a slightly better correlation with my dependent variable, Churn_cat, I will drop Bandwidth_GB_Year first.

[89]:
```
# Drop 1 high VIF variable
X = X.drop('Bandwidth_GB_Year', axis = 1)
```

[90]:
```
# Perform variance inflation factor analysis for trimmed feature set
vif_data = pd.DataFrame()
vif_data['IndVar'] = X.columns
```

```
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
    ↪columns))]
print(vif_data)
```

```
               IndVar        VIF
0            Area_cat   2.425161
1            Children   1.897252
2                 Age   6.460408
3              Income   2.847797
4         Marital_cat   2.810827
5          Gender_cat   1.879624
6    Outage_sec_perweek   9.233796
7  Yearly_equip_failure   1.381910
8              Tenure   2.605773
9       MonthlyCharge  11.151377
```

The VIF scores look much better than they did, but there are still a few that are rather high. Referring back to my correlation table, MonthlyCharge has a far greater correlation with my dependent variable than Outage_sec_perweek does, so I will drop Outage_sec_perweek and repeat the test.

[91]:
```
# Drop 1 high VIF variable
X = X.drop('Outage_sec_perweek', axis = 1)
```

[92]:
```
# Perform variance inflation factor analysis for trimmed feature set
vif_data = pd.DataFrame()
vif_data['IndVar'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
    ↪columns))]
print(vif_data)
```

```
               IndVar       VIF
0            Area_cat  2.398333
1            Children  1.879604
2                 Age  6.116797
3              Income  2.807292
4         Marital_cat  2.775227
5          Gender_cat  1.863478
6  Yearly_equip_failure  1.377841
7              Tenure  2.569440
8       MonthlyCharge  8.858859
```

I have only 2 variables remaining with VIF greater than 5. Once again the correlation table recognizes MonthlyCharge as a better candidate for inclusion in the model, so Age will be dropped from the group of independent variables.

[93]:
```
# Drop 1 high VIF variable
X = X.drop('Age', axis = 1)
```

[94]:
```
# Perform variance inflation factor analysis for trimmed feature set
vif_data = pd.DataFrame()
vif_data['IndVar'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
↪columns))]
print(vif_data)
```

```
                 IndVar       VIF
0              Area_cat  2.365519
1              Children  1.872816
2                Income  2.765505
3           Marital_cat  2.739215
4            Gender_cat  1.852198
5  Yearly_equip_failure  1.372769
6                Tenure  2.523888
7         MonthlyCharge  6.719572
```

While MonthlyCharge still has a score higher than the other remaining variables, it is still less than 10.

I will create a reduced model based on my remaining variables to see how our statistics look.

[95]:
```
# Create first reduced model and display summary
mdl_churn_vs_reduced = logit("Churn_cat ~ Area_cat + Children + Income +␣
↪Marital_cat + Gender_cat + Yearly_equip_failure + MonthlyCharge + Tenure",
                           data=df_data).fit()
print(mdl_churn_vs_reduced.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.341735
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:              Churn_cat   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9991
Method:                           MLE   Df Model:                            8
Date:                Mon, 16 May 2022   Pseudo R-squ.:                  0.4090
Time:                        16:31:36   Log-Likelihood:                -3417.4
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
========
                           coef    std err          z      P>|z|      [0.025
0.975]
```

```
--------------------------------------------------------------------------------
--------
Intercept            -5.2842      0.173    -30.593      0.000      -5.623
-4.946
Area_cat              0.0154      0.037      0.414      0.679      -0.057
0.088
Children             -0.0091      0.014     -0.636      0.525      -0.037
0.019
Income             8.294e-07   1.08e-06      0.770      0.441   -1.28e-06
2.94e-06
Marital_cat           0.0332      0.021      1.570      0.116      -0.008
0.075
Gender_cat            0.0960      0.055      1.736      0.083      -0.012
0.204
Yearly_equip_failure    -0.0372      0.048     -0.774      0.439      -0.132
0.057
MonthlyCharge         0.0332      0.001     37.107      0.000       0.031
0.035
Tenure               -0.0738      0.002    -41.777      0.000      -0.077
-0.070
================================================================================
========
```

According to this summary, several variables exhibit high p-values, indicating no relationship between that variable and the dependent variable, Churn_cat. A value greater than .05 is considered high. I will remove these variables from the model and once again evaluate the resulting summary and statistics.

[96]:
```
# Create first reduced model and display summary
mdl_churn_vs_features = logit("Churn_cat ~ MonthlyCharge + Tenure",
                              data=df_data).fit()
print(mdl_churn_vs_features.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.342087
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:              Churn_cat   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9997
Method:                           MLE   Df Model:                            2
Date:                Mon, 16 May 2022   Pseudo R-squ.:                  0.4084
Time:                        16:33:21   Log-Likelihood:                -3420.9
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
=
```

```
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
Intercept        -5.1525      0.150    -34.283      0.000      -5.447
-4.858
MonthlyCharge     0.0332      0.001     37.119      0.000       0.031
0.035
Tenure           -0.0738      0.002    -41.788      0.000      -0.077
-0.070
================================================================================
=
```

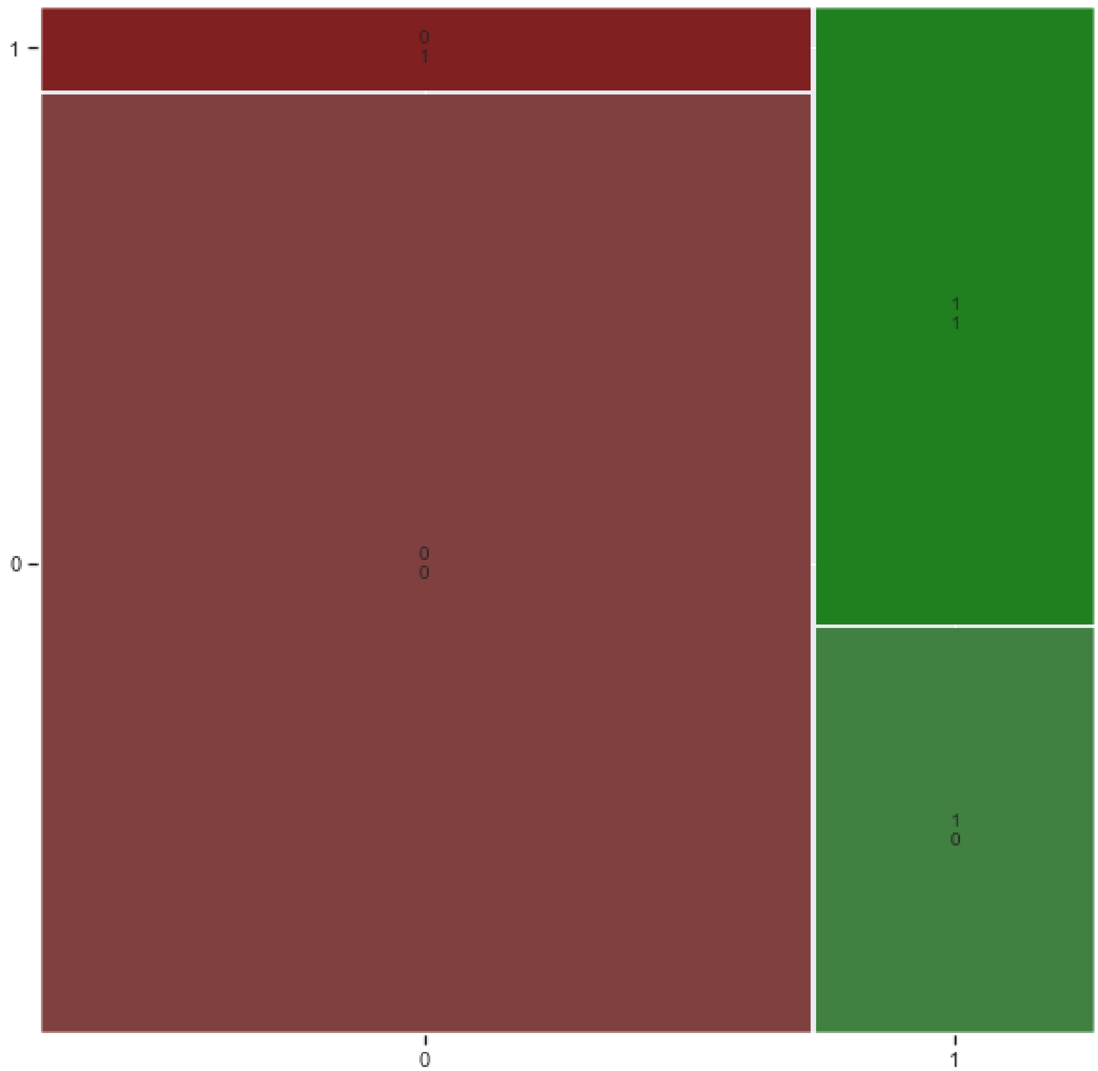
[97]:

```python
Xtest = df_data[['MonthlyCharge', 'Tenure']]
ytest = df_data['Churn_cat']
yhat = mdl_churn_vs_features.predict(Xtest)
prediction = list(map(round, yhat))
conf_matrix = confusion_matrix(ytest, prediction)
print ("Confusion Matrix : \n", conf_matrix)
print('Test accuracy = ', accuracy_score(ytest, prediction))
mosaic(conf_matrix)
```

```
Confusion Matrix :
 [[6738  612]
 [1050 1600]]
Test accuracy =  0.8338
```

[97]:

```
(<Figure size 720x720 with 3 Axes>,
 {('0', '0'): (0.0, 0.0, 0.7313432835820897, 0.9136890636653332),
  ('0', '1'): (0.0,
   0.9170113228015457,
   0.7313432835820897,
   0.08298867719845412),
  ('1', '0'): (0.7363184079601991,
   0.0,
   0.263681592039801,
   0.39491004826678366),
  ('1', '1'): (0.7363184079601991,
   0.3982323074029963,
   0.263681592039801,
   0.6017676925970036)})
```



4.3 Final Reduced Multiple Regression Model

At this point, I have eliminated any sources of multicollinearity and collinearity as well as variables exhibiting p-values that exceed .05. I will finalize the reduced model and check to see how it compares to my initial model which included all variables in the set.

[98]:
```
# Create final reduced model and display summary
mdl_churn_vs_features_final = logit("Churn_cat ~ MonthlyCharge + Tenure",
                                   data=df_data).fit()
print(mdl_churn_vs_features_final.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.342087
```

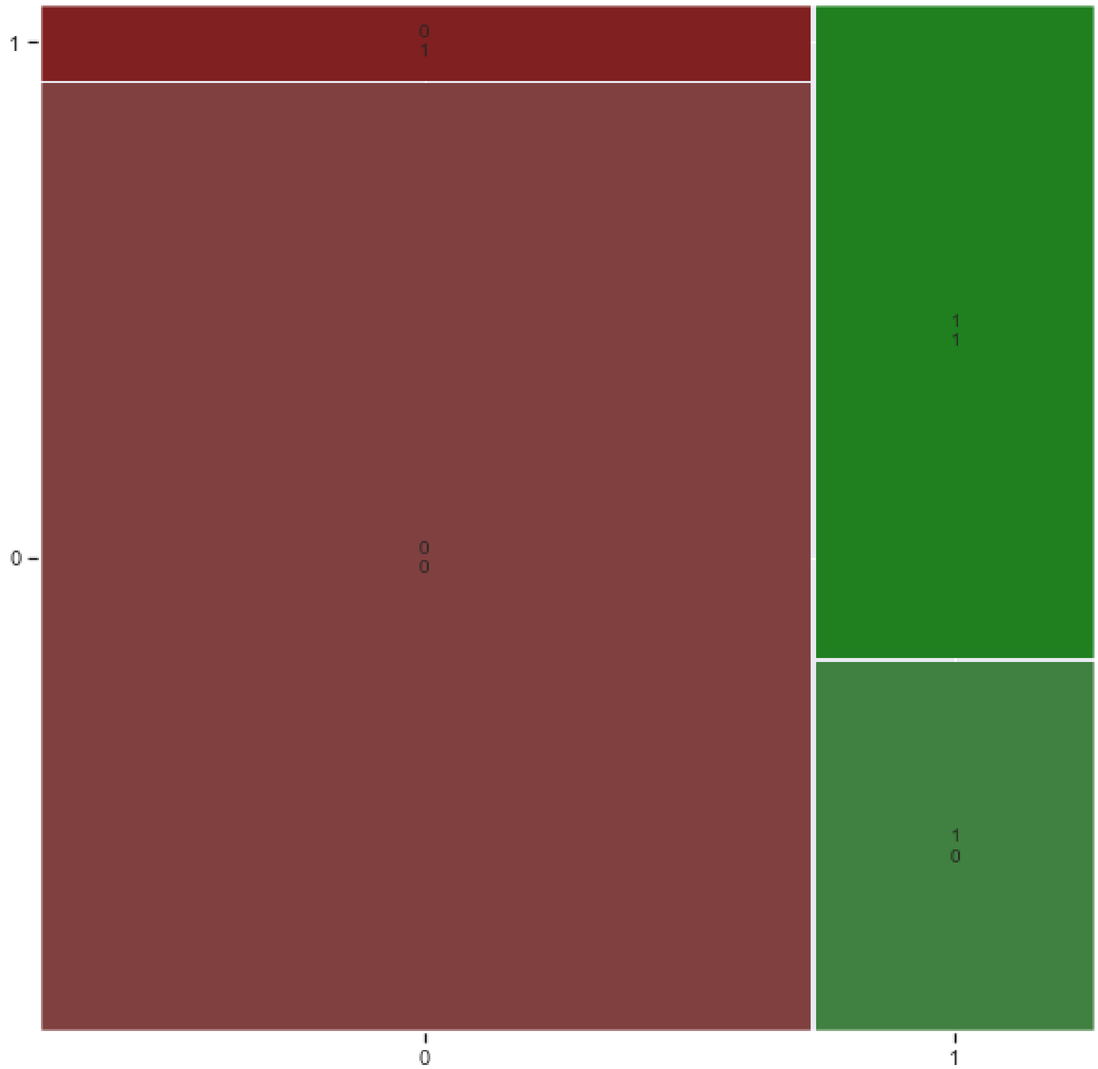
```
        Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:              Churn_cat   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9997
Method:                           MLE   Df Model:                            2
Date:                Mon, 16 May 2022   Pseudo R-squ.:                  0.4084
Time:                        16:35:18   Log-Likelihood:                -3420.9
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
=
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
Intercept        -5.1525      0.150    -34.283      0.000      -5.447
-4.858
MonthlyCharge     0.0332      0.001     37.119      0.000       0.031
0.035
Tenure           -0.0738      0.002    -41.788      0.000      -0.077
-0.070
================================================================================
=
```

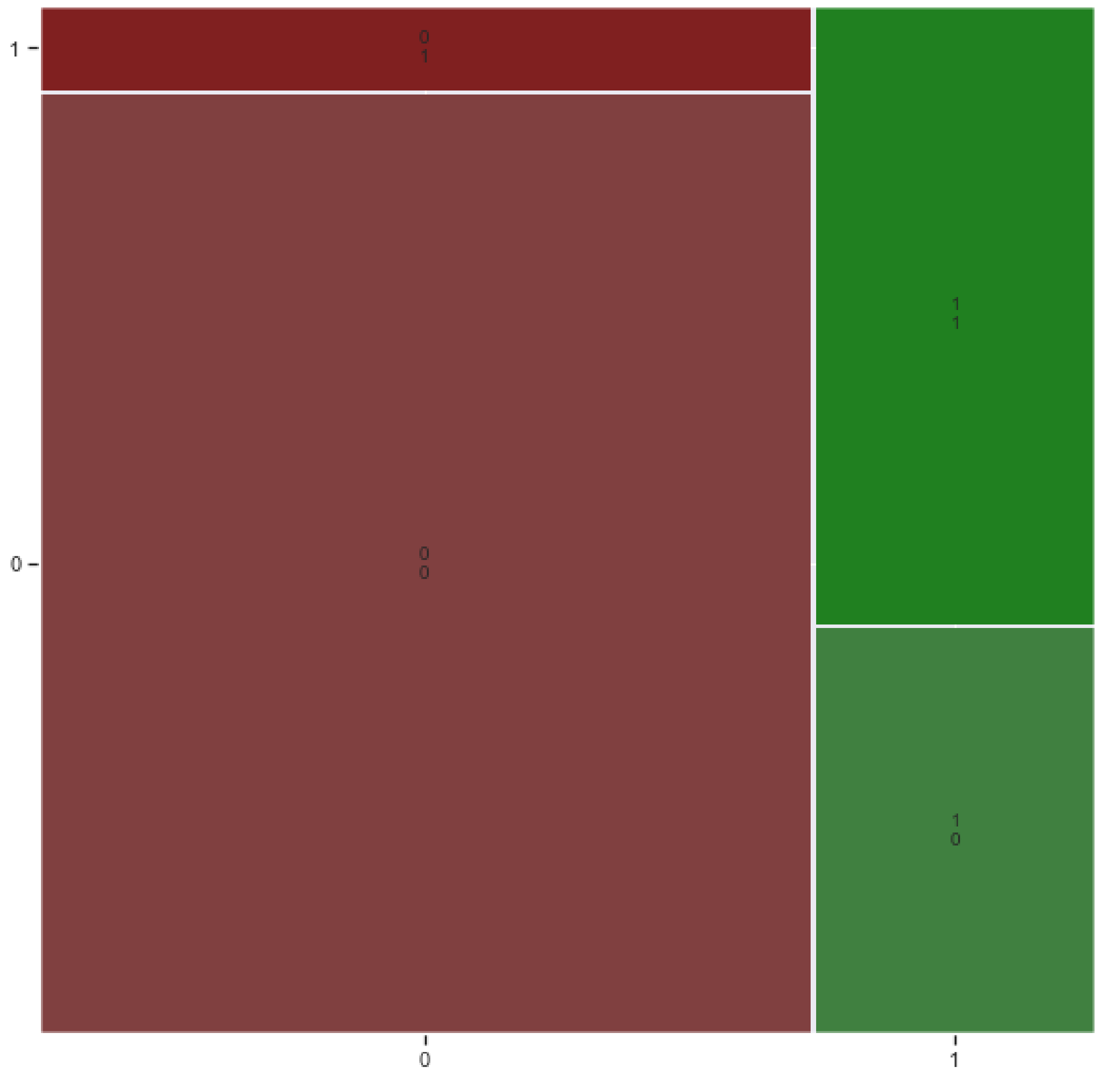
[99]:

```
# Display confusion matrix, accuracy score, and mosaic for initial model and␣
↪final reduced model for comparison
Xorig = df_data[['Area_cat', 'Children', 'Age', 'Income', 'Marital_cat',␣
↪'Gender_cat', 'Outage_sec_perweek',
           'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',␣
↪'Bandwidth_GB_Year']]
yorig = df_data['Churn_cat']
yhat_orig = mdl_churn_vs_all.predict(Xorig)
prediction_orig = list(map(round, yhat_orig))
conf_matrix_orig = confusion_matrix(yorig, prediction_orig)
Xfinal = df_data[['MonthlyCharge', 'Tenure']]
yfinal = df_data['Churn_cat']
yhat_final = mdl_churn_vs_features_final.predict(Xfinal)
prediction_final = list(map(round, yhat_final))
conf_matrix_final = confusion_matrix(yfinal, prediction_final)
print ("Orignal Confusion Matrix : \n", conf_matrix_orig)
print('Orignal accuracy = ', accuracy_score(yorig, prediction_orig))
print ("Final Confusion Matrix : \n", conf_matrix_final)
print('Final accuracy = ', accuracy_score(yfinal, prediction_final))
mosaic(conf_matrix_orig)
mosaic(conf_matrix_final)
```

```
Orignal Confusion Matrix :
 [[6794  556]
 [ 955 1695]]
Orignal accuracy =  0.8489
Final Confusion Matrix :
 [[6738  612]
 [1050 1600]]
Final accuracy =  0.8338
```

[99]:
```
(<Figure size 720x720 with 3 Axes>,
 {('0', '0'): (0.0, 0.0, 0.7313432835820897, 0.9136890636653332),
  ('0', '1'): (0.0,
   0.9170113228015457,
   0.7313432835820897,
   0.08298867719845412),
  ('1', '0'): (0.7363184079601991,
   0.0,
   0.263681592039801,
   0.39491004826678366),
  ('1', '1'): (0.7363184079601991,
   0.3982323074029963,
   0.263681592039801,
   0.6017676925970036)})
```


0
1
0
0
1
1
1
0
1
0
0
1

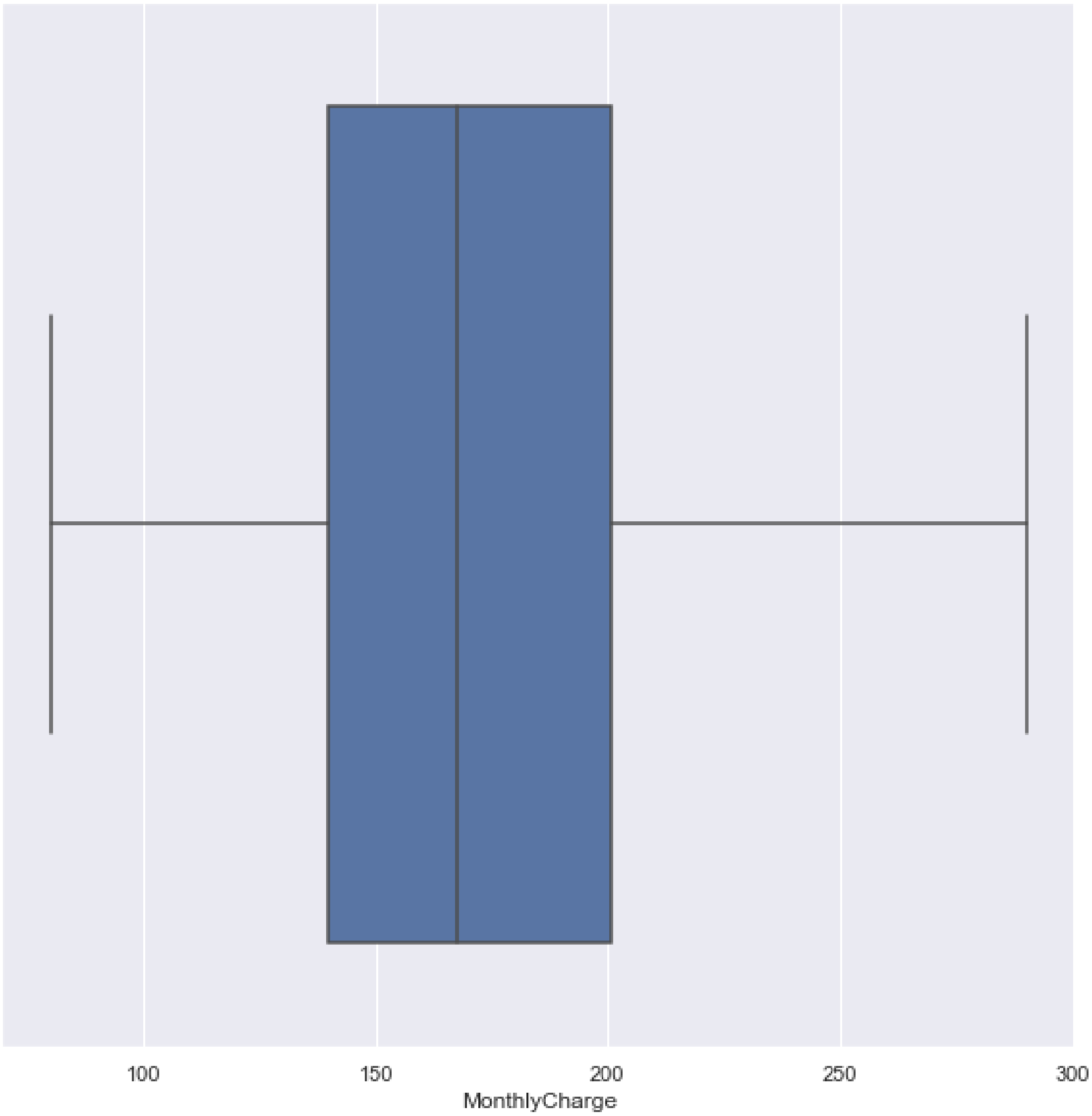


The reduced model holds up well when compared to the initial model using the accuracy score and confusion matrix as my measurements.

I will perform due diligence and check for extreme outliers to make sure my independent variables comply with the assumptions of logistic regression.

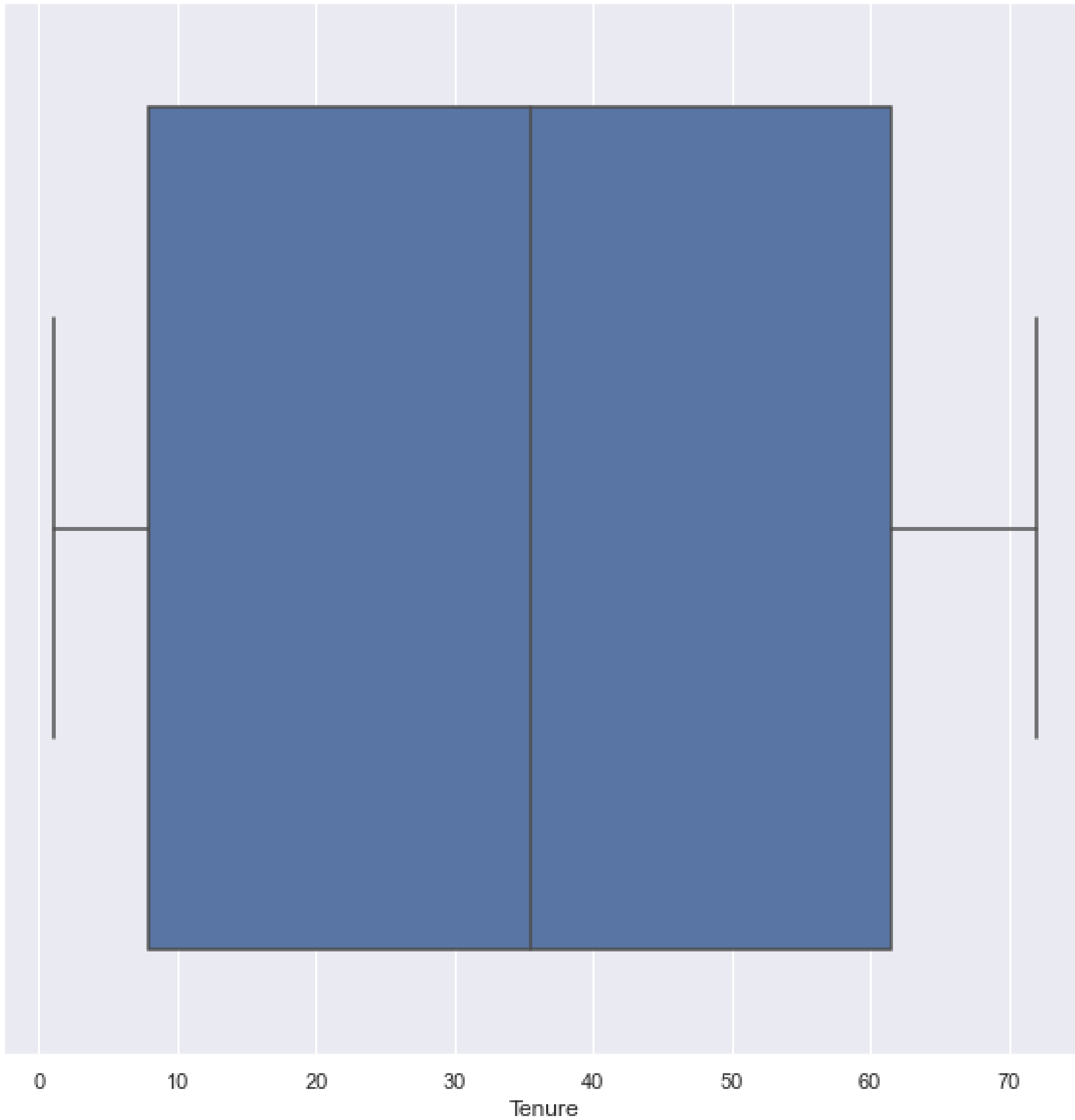
[100]:
```
sns.boxplot(x='MonthlyCharge',data=df_data)
```

[100]:
```
<AxesSubplot:xlabel='MonthlyCharge'>
```



[101]:
```
sns.boxplot(x='Tenure',data=df_data)
```

[101]: <AxesSubplot:xlabel='Tenure'>



Neither variable has extreme outliers.

While Python does not have a library supporting the Box-Tidwell test with a single line of code, I can use a combination of functions from the statsmodels package to perform the test. The goal will be to verify that the "MonthlyCharge:Log_MonthlyCharge" and "Tenure:Log_Tenure" interactions have p-values greater than 0.05, implying that both independent variables are linearly related to the logit of the dependent variable, Churn_cat.

[102]:
```
# Define continuous variables
continuous_var = ['MonthlyCharge', 'Tenure']
```

```python
# Add logit transform interaction terms (natural log) for continuous variables␣
↪e.g.. Age * Log(Age)
for var in continuous_var:
    df_data[f'{var}:Log_{var}'] = df_data[var].apply(lambda x: x * np.log(x))

# Keep columns related to continuous variables
cols_to_keep = continuous_var + df_data.columns.tolist()[-len(continuous_var):]

# Redefining variables to include interaction terms
X_lt = df_data[cols_to_keep]
y_lt = df_data['Churn_cat']

# Add constant term
X_lt_constant = sm.add_constant(X_lt, prepend=False)

# Building model and fit the data (using statsmodel's Logit)
logit_results = GLM(y_lt, X_lt_constant, family=families.Binomial()).fit()

# Display summary results
print(logit_results.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              Churn_cat   No. Observations:                10000
Model:                            GLM   Df Residuals:                     9995
Model Family:                Binomial   Df Model:                            4
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -3420.6
Date:                Mon, 16 May 2022   Deviance:                       6841.2
Time:                        16:52:43   Pearson chi2:                 8.03e+03
No. Iterations:                     7
Covariance Type:            nonrobust
================================================================================
===================
                                         coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------------
MonthlyCharge                          0.0075      0.037      0.204      0.838
-0.064       0.079
Tenure                                -0.0681      0.021     -3.226      0.001
-0.110      -0.027
MonthlyCharge:Log_MonthlyCharge        0.0041      0.006      0.702      0.483
-0.007       0.016
Tenure:Log_Tenure                     -0.0014      0.005     -0.285      0.776
-0.011       0.008
const                                 -4.4367      1.053     -4.212      0.000
```

```
-6.501      -2.372
==============================================================================
==================
```

4.4 Data Analysis Process

During my variable selection process, I relied upon trusted methods for identifying variables unsuitable for the model, such as VIF, a correlation table, and p-values. I measured each model's performance by its accuracy score, as well as the confusion matrix.

---

5 Part V: Data Summary and Implications

5.1 Summary of Findings

The regression equation for the final reduced model is as follows:

Churn_cat ~ MonthlyCharge + Tenure

The coefficients for each variable included:

MonthlyCharge 0.0332

Tenure -0.0738

We can use these coefficients to determine the effect each variable will have on a customer's decision to cancel service. MonthlyCharge has a positive coefficient, indicating the higher a customer's monthly charge is, the more likely they are to churn. Contrarily, Tenure has a negative coefficient, which tells us that customers who retain service for longer periods of time grow less and less likely to churn.

The model can provide significant data when evaluating customer retention from a practical perspective, as customers who have been with the company for a long time may be less likely to cancel service, but if the rate they are charged increases so do the chances they will churn. The limitations of using logistic regression models for practical purposes are always present, however. They are susceptible to overfitting and can appear to have more predictive power than they do (Robinson, 2018). Also, as logistic regressions cannot predict continuous outcomes, their predictions contain less detail. For instance, this model may be able to predict what makes a customer more likely to churn, but not when they might do so.

5.2 Recommended Course of Action

There are a few key takeaways based on the analysis of this model. For each month the customer stays with the ISP they are less likely to churn, but if their monthly fee increases they are more likely to churn. It may be beneficial to offer long-standing customers occasional discounted rates to further increase the chance they will retain service. For newer customers who already run a higher risk of churning, increasing their rates should be avoided altogether.

---

6 Part VI: Demonstration

Panopto Video Recording

A link for the Panopto video has been provided separately. The demonstration includes the following:

- Demonstration of the functionality of the code used for the analysis
- Identification of the version of the programming environment
- Comparison of the two multiple regression models you used in your analysis
- Interpretation of the coefficients

---

7 Web Sources

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.astype.html

https://pbpython.com/categorical-encoding.html

https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290

---

8 References

Insights for Professionals. (2019, February 26). *5 Niche Programming Languages (And Why They're Underrated).* https://www.insightsforprofessionals.com/it/software/niche-programming-languages

Parra, H. (2021, April 20). *The Data Science Trilogy.* Towards Data Science. https://towardsdatascience.com/the-data-science-trilogy-numpy-pandas-and-matplotlib-basics-42192b89e26

Zach. (2021, November 16). *The 6 Assumptions of Logistic Regression (With Examples).* Statology. https://www.statology.org/assumptions-of-logistic-regression/

Robinson, N. (2018, June 28). *The Disadvantages of Logistic Regression.* The Classroom. https://www.theclassroom.com/multivariate-statistical-analysis-2448.html