PERFORMANCE ASSESSMENT

Task 2

KAILI HAMILTON

Masters of Science in Data Analytics, Western Governors University

Course: D208 Predictive Modeling

Instructor: Dr. Daniel Smith

Program Mentor: Krissy Bryant

November, 2023

TABLE OF CONTENTS

A1

My research question is, “What factors influence if customers churn?”

A2

The goals of the data analysis is to determine which factors influence if customers churn. This will be accomplished through logistic regression because the target variable, “Churn”, is a categorical variable with response 0 for no and 1 for yes. This will give the company insight into how to prevent customer churn.

B1

Assumptions of multiple logistic regression include (“Assumptions of Logistic Regression”, 2023):

- The target variable must be categorical.
- The observations need to be independent of each other.
- Little to no multicollinearity among the independent variables.
- The independent variables are linearly related to the log odds.
- A large sample size is needed.

B2

Two benefits of using Python in support of various phases of the analysis include, but are not limited to, computing power to calculate and fit a logistic model using vast amounts of data and creating myriad data visualizations to inform the analysis.

B3

Multiple logistic regression is an appropriate technique to answer the research question (summarized in parts A1 and A2) because the target variable is a categorical variable. It has with two possible outcomes, 0 for no and 1 for yes. Specifically, I want to predict which factors influence if customers churn.

C1

My data cleaning goals are designed to help me answer my research question (Part A1). My data cleaning goals are to treat null values, remove extreme outliers, re-express categorical variables for use in multiple linear regression, and to determine which variables to use in my initial regression model (Middleton, nd).

Treated null values.

There were no null values in the data.

Check for null values

```
df.isna().sum()
```

```
Population             0
Age                    0
Income                 0
Churn                  0
Outage_sec_perweek     0
Yearly_equip_failure   0
Techie                 0
Port_modem             0
Tablet                 0
InternetService        0
Phone                  0
Multiple               0
OnlineSecurity         0
OnlineBackup           0
StreamingTV            0
StreamingMovies        0
Tenure                 0
MonthlyCharge          0
Bandwidth_GB_Year      0
dtype: int64
```

no null values

Removed outliers.

Regression is sensitive to outliers. However, due to the natural variation of the distribution of some variables, I decided to keep most outliers.

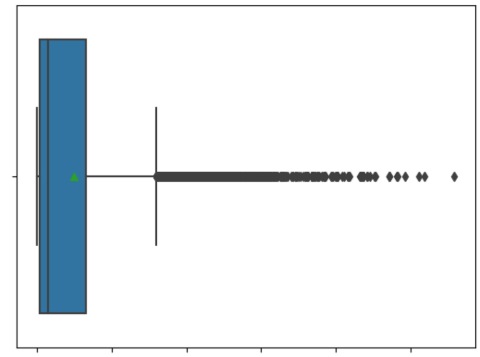
The variables “Population”, “Income”, “Outage_sec_perweek”, and “Yearly_equip_failure” all contain outliers. I created boxplots for all numerical variables and calculated the number of outliers for each to inform whether to drop outliers. I did not want to change the shape of the distribution so I opted to only drop the most extreme outliers.

An example of what was described is shown with the variable Population. See the copy of the code for further details regarding code.

POPULATION

```
sns.boxplot(x='Population', data=df, showmeans=True)
```

```
<Axes: xlabel='Population'>
```




```
def find_outliers_IQR(df):

   q1=df.quantile(0.25)

   q3=df.quantile(0.75)

   IQR=q3-q1

   outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]

   return outliers
```

```
outliers = find_outliers_IQR(df['Population'])
print("number of outliers: " + str(len(outliers)))
print('max outlier value: ' + str(outliers.max()))
print('min outlier value: ' + str(outliers.min()))
```

```
number of outliers: 937
max outlier value: 111850
min outlier value: 31816
```

Outliers that were dropped:

- Population > 100,000

```
df.drop(df[df['Population'] > 100000].index, inplace=True)
```

```
df.shape
```

```
(9991, 19)
```

```
df['Population'].describe()
```

```
count     9991.000000
mean      9730.486037
std      14341.542493
min          0.000000
25%        737.500000
50%       2905.000000
75%      13161.000000
max      98660.000000
Name: Population, dtype: float64
```

- Income > 200,000

```
df['Income'].describe()
```

```
count      9994.000000
mean      39710.411833
std       27861.200412
min         348.670000
25%       19219.835000
50%       33156.205000
75%       53226.895000
max      196746.000000
Name: Income, dtype: float64
```

```
df.drop(df[df['Population'] > 100000].index, inplace=True)
```

```
df.shape
```

```
(9991, 19)
```

- Yearly_equip_failure = 6

```
df.drop(df[df['Yearly_equip_failure'] == 6].index, inplace=True)
```

```
df.shape
```

```
(9999, 19)
```

```
df['Yearly_equip_failure'].describe()
```

```
count    9999.000000
mean        0.397440
std         0.633512
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         4.000000
Name: Yearly_equip_failure, dtype: float64
```

Re-expressed categorical data types.

First I re-expressed ordinal categorical data types using ordinal encoding. These included variables with a "yes" or "no" value. All "yes" values were replaced with "1" and all "no" values were replaced with "0". Then I made sure the datatype changed from object to int64.

```
df['Churn'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

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

```
0    7341
1    2650
Name: Churn, dtype: int64
```

```
df['Techie'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['Port_modem'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

```
df['Tablet'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['Phone'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

```
df['Multiple'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['OnlineSecurity'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['OnlineBackup'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['StreamingTV'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['StreamingMovies'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

```
df['StreamingMovies'].value_counts()
```

```
0    5104
1    4887
Name: StreamingMovies, dtype: int64
```

```
df.dtypes
```

```
Population              int64
Age                     int64
Income                float64
Churn                   int64
Outage_sec_perweek    float64
Yearly_equip_failure    int64
Techie                  int64
Port_modem              int64
Tablet                  int64
InternetService        object
Phone                   int64
Multiple                int64
OnlineSecurity          int64
OnlineBackup            int64
StreamingTV             int64
StreamingMovies         int64
Tenure                float64
MonthlyCharge         float64
Bandwidth_GB_Year     float64
dtype: object
```

Next, I re-expressed <u>nominal</u> categorical data types for the variable “InternetService” using pd.getdummies(). “InternetService” has three values, with no inherent rank.

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

```
array(['Fiber Optic', 'DSL', 'None'], dtype=object)
```

```
df = pd.get_dummies(df, columns=['InternetService'])
```

```
df.head()
```

Port_modem	Tablet	Phone	...	OnlineSecurity	OnlineBackup	StreamingTV	StreamingMovies	Tenure	MonthlyCharge	Bandwidth_GB_Year	InternetService_DSL	InternetService_Fiber Optic	InternetService_None
1	1	1	...	1	1	0	1	6.795513	172.455519	904.536110	0	1	0
0	1	1	...	1	0	1	1	1.156681	242.632554	800.982766	0	1	0
1	0	1	...	0	0	0	1	15.754144	159.947583	2054.706961	1	0	0
0	0	1	...	1	0	1	0	17.087227	119.956840	2164.579412	1	0	0
1	0	0	...	0	0	1	0	1.670972	149.948316	271.493436	0	1	0

Determined which variables to use in my initial model.

I initially kept all variables that I thought would influence the response variable “Churn”. “Job” was one of the variables I thought to include, but there 639 unique values, so I excluded it from the analysis. I also dropped “Martial” for the same reason, too many unique values. I also dropped “Children” so the focus is just the customer, not the household. This is also discussed in Part C4.

	Job
count	10000
unique	639
top	Occupational psychologist
freq	30

```
# drop "Job" variable - too many unique observations
df.drop(columns='Job', inplace=True)
```

```
df['Marital'].value_counts()
```

```
Divorced         2092
Widowed          2027
Separated        2014
Never Married    1956
Married          1911
Name: Marital, dtype: int64
```

```
# drop "Marital" variable - too many unique observations
df.drop(columns='Marital', inplace=True)
df.head()
```

```
df.drop(columns='Children', inplace=True)
df.head()
```

The explanatory variables used in my initial regression are:

- Population
- Age
- Income
- Bandwidth_GB_Year
- Outage_sec_perweek
- Yearly_equip_failure
- Techie
- Port_modem
- Tablet
- InternetService
- Phone
- Multiple
- OnlineSecurity
- OnlineBackup
- StreamingTV
- StreamingMovies
- Tenure
- MonthlyCharge

C2

The dependent (aka response) variable for the data analysis is "Churn". Out of 9991 customers in the dataset, 73.5% of customers churned within the last month, leaving 26.5% of customers staying with the company.

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

```
Churn
0    7341
1    2650
Name: count, dtype: int64
```

The independent (aka explanatory) variables in this analysis are listed below along with their summary statistics. There are 9,991 entries for each variable.

- Population. The distribution of Population is highly skewed right as evidenced by the mean population of 9730 being much larger than the median population, 2905. The middle 50% of customers live in a one mile radius population size between 737 and 13,161 people.

```
df['Population'].describe()
```

```
count     9991.000000
mean      9730.486037
std      14341.542493
min          0.000000
25%        737.500000
50%       2905.000000
75%      13161.000000
max      98660.000000
Name: Population, dtype: float64
```

- Age. The mean and median age for customers is 53, with the oldest customer 89 years old and the youngest 20 years old.

```
df['Age'].describe()
```

```
count    9991.000000
mean       53.082174
std        20.702085
min        18.000000
25%        35.000000
50%        53.000000
75%        71.000000
max        89.000000
Name: Age, dtype: float64
```

- Income. On average, customers have a yearly income of about $39,716 (median value $33,169). The income of the middle 50% of customers ranges between $19,215 and $53,228.

```
df['Income'].describe()
```

```
count      9991.000000
mean      39715.501730
std       27863.826217
min         348.670000
25%       19214.740000
50%       33168.880000
75%       53227.795000
max      196746.000000
Name: Income, dtype: float64
```

- Bandwidth_GB_Year. From the summary statistics displayed below we see that there are 9991 values with a range of approximately 7003 GB. On average, customers use about 3391 GB per year with a median value of 3261 GB. The middle 50% of customers use between 1236 and 5585 GB per year.

```
df['Bandwidth_GB_Year'].describe()
```

```
count    9991.000000
mean     3390.795380
std      2185.252241
min       155.506715
25%      1236.046551
50%      3260.745232
75%      5584.704954
max      7158.981530
Name: Bandwidth_GB_Year, dtype: float64
```

- Outage_sec_perweek. On average, customers' neighborhood's experience 10 seconds of outage time per week.

```
df['Outage_sec_perweek'].describe()
```

```
count    9991.000000
mean       10.002278
std         2.976494
min         0.099747
25%         8.019310
50%        10.019720
75%        11.971418
max        21.207230
Name: Outage_sec_perweek, dtype: float64
```

- Yearly_equip_failure. The average number of times per year a customer's equipment failed and replaced was 0.4, with a median of 0 times. The maximum yearly equipment failure a customer experienced was 4.

```
df['Yearly_equip_failure'].describe()
```

```
count    9991.000000
mean        0.397157
std         0.633253
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         4.000000
```

- Techie. Out of 9991 customers, only 16.8% of customers described themselves as technically inclined.

```
df['Techie'].value_counts()
```

```
Techie
0    8314
1    1677
Name: count, dtype: int64
```

- Port_modem. Out of 9991 customers, 51.6% has a portable modem.

```
df['Port_modem'].value_counts()
```

```
Port_modem
0    5158
1    4833
Name: count, dtype: int64
```

- Tablet. Out of 9991 customers, only 29.9% reported owning a tablet.

```
df['Tablet'].value_counts()
```

```
Tablet
0    7006
1    2985
Name: count, dtype: int64
```

- InternetService. Customer's internet service provider could be fiber optics, DSL, or none. 2128 out of 9991 customers (21.3%) have neither fiber optics nor DSL. More customers have fiber optics than DSL (44.1% vs. 34.6%).

```
df['InternetService_Fiber Optic'].value_counts()
```

```
InternetService_Fiber Optic
0    5587
1    4404
Name: count, dtype: int64
```

```
df['InternetService_DSL'].value_counts()
```

```
InternetService_DSL
0    6532
1    3459
Name: count, dtype: int64
```

- Phone. 90.7% of customers own a phone.

```
df['Phone'].value_counts()
```

```
Phone
1    9058
0     933
Name: count, dtype: int64
```

- Multiple. Less than half of customers have multiple phone lines, 46.1%.

```
df['Multiple'].value_counts()
```

```
Multiple
0    5387
1    4604
Name: count, dtype: int64
```

- OnlineSecurity. Just over a third of customers (35.8%) have an online security addon.

```
df['OnlineSecurity'].value_counts()
```

```
OnlineSecurity
0    6418
1    3573
Name: count, dtype: int64
```

- OnlineBackup. Just under half of all customers, 45.1%, have an addon for online backup.

```
df['OnlineBackup'].value_counts()
```

```
OnlineBackup
0    5489
1    4502
Name: count, dtype: int64
```

- StreamingTV. About half of all customers (49.3%) stream TV.

```
df['StreamingTV'].value_counts()
```

```
StreamingTV
0    5068
1    4923
Name: count, dtype: int64
```

- StreamingMovies. About half of all customers (48.9%) stream movies.

```
df['StreamingMovies'].value_counts()
```

```
StreamingMovies
0    5104
1    4887
```

- Tenure. The average tenure of customers is 34 months. The middle 50% of customers have been with the provider between 8 and 61 months. The most tenured customer has been with the provider for 72 months, while the least tenured is 1 month.

```
df['Tenure'].describe()
```

```
count    9991.000000
mean       34.508248
std        26.442933
min         1.000259
25%         7.916107
50%        33.196120
75%        61.473295
max        71.999280
Name: Tenure, dtype: float64
```

- MonthlyCharge. The average monthly charge per customer is $172.62 (median $167.48). The middle 50% of customers are charged between $139.98 and $200.17 monthly.

```
df['MonthlyCharge'].describe()
```

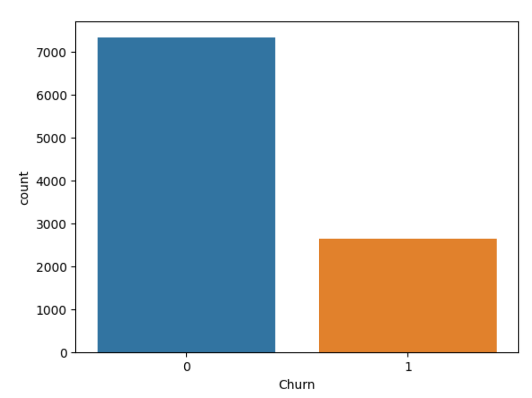
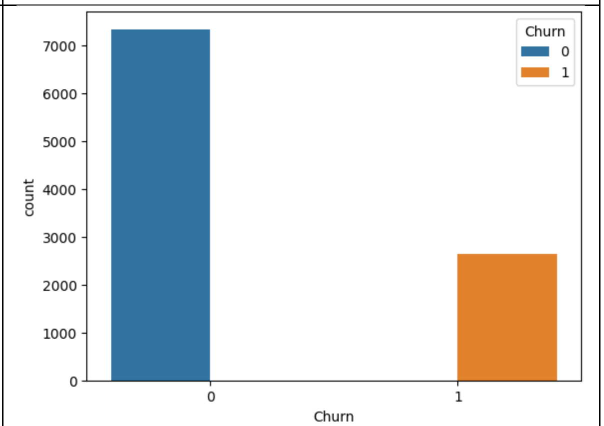
```
count    9991.000000
mean      172.618895
std        42.937793
min        79.978860
25%       139.979239
50%       167.484700
75%       200.165200
max       290.160419
Name: MonthlyCharge, dtype: float64
```

C3

Univariate and bivariate visualizations are for all variables are presented here. “Churn” is the only dependent variable. Bivariate visualizations are grouped by “Churn”.

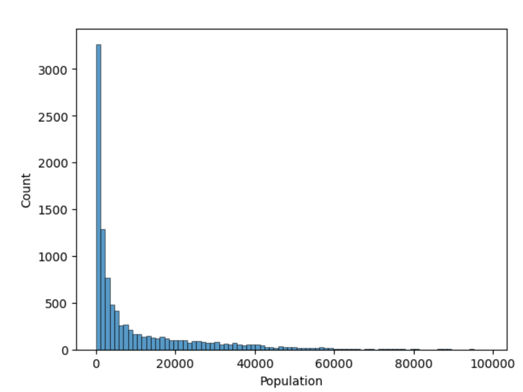
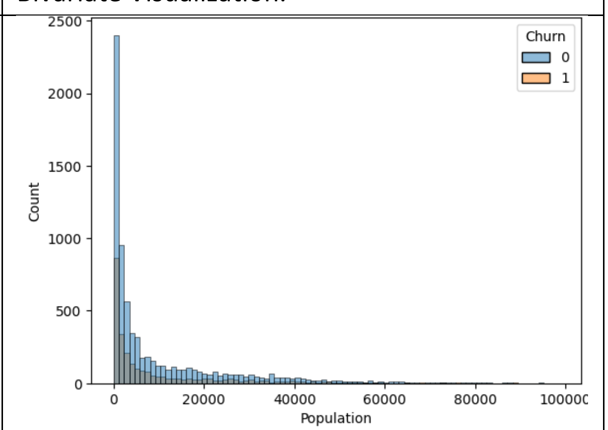
INDEPENDENT VARIABLE: CHURN

- Churn. 0 = No. 1 = Yes. (Dependent/target variable)

<table>
<tr><td>Univariate Visualization</td><td>Bivariate Visualization.</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

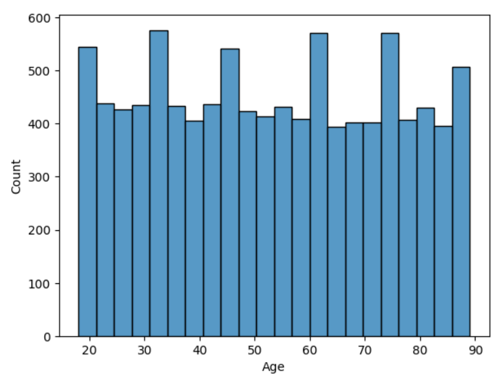
DEPENDENT VARIABLES:

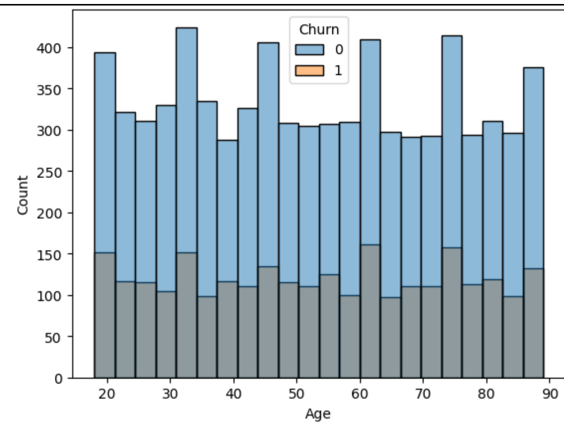
- Population

<table>
<tr><td>Univariate Visualization</td><td>Bivariate Visualization.</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

- Age

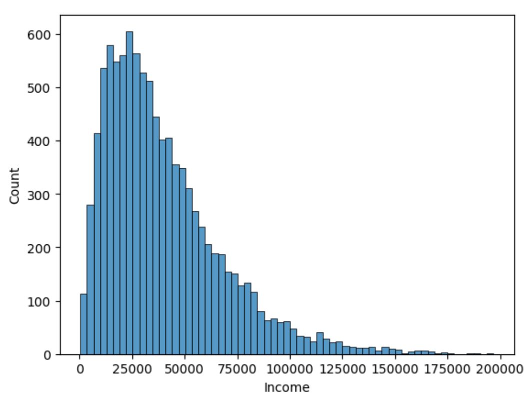
Univariate Visualization	Bivariate Visualization.

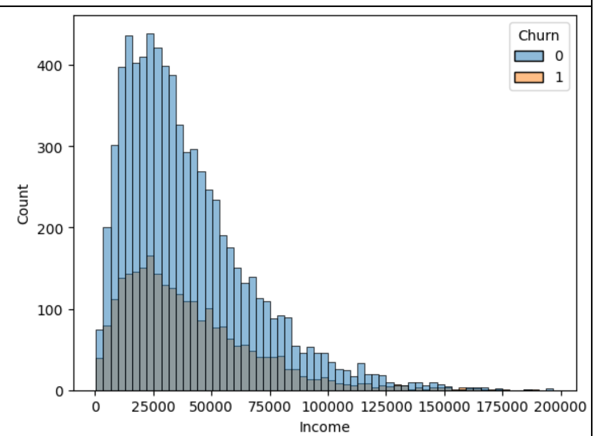




- Income

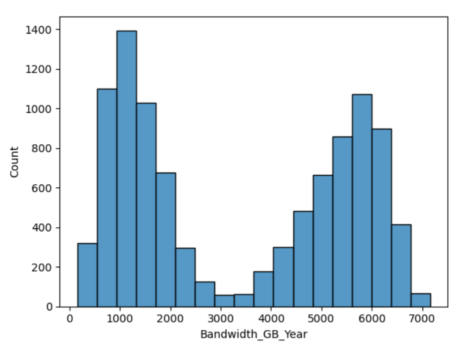
Univariate Visualization	Bivariate Visualization.

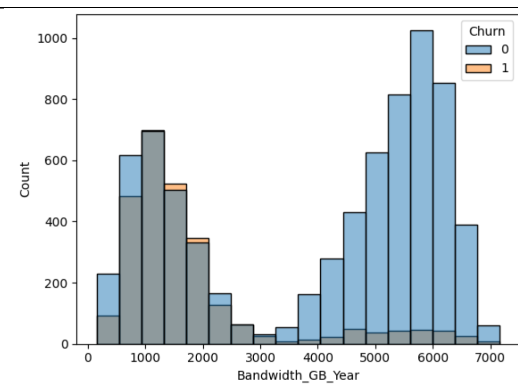




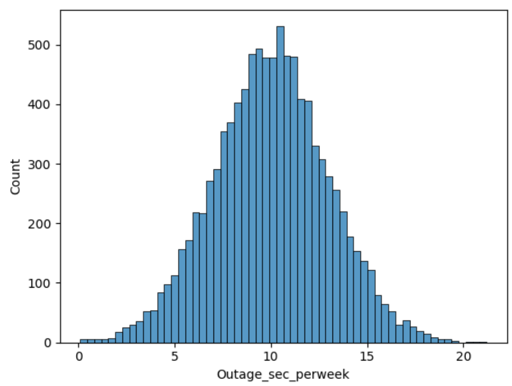
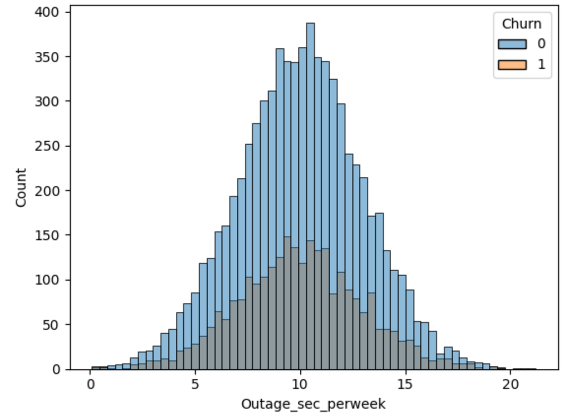
- Bandwidth_GB_Year

Univariate Visualization	Bivariate Visualization.

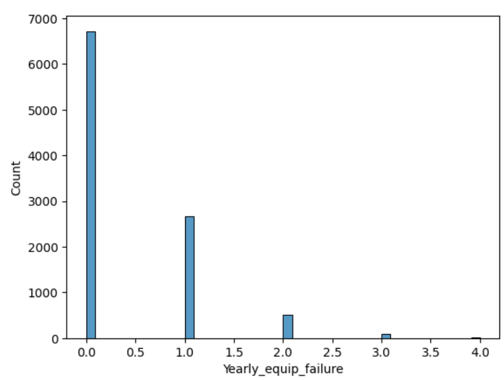
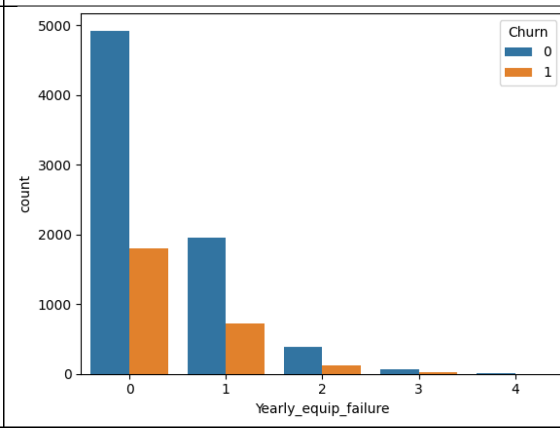




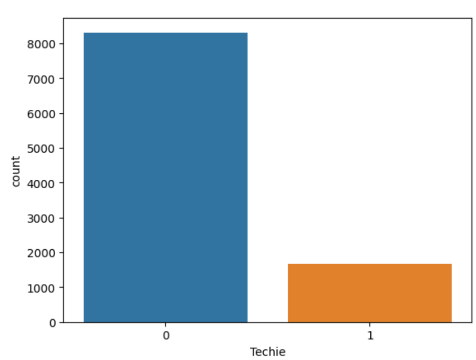
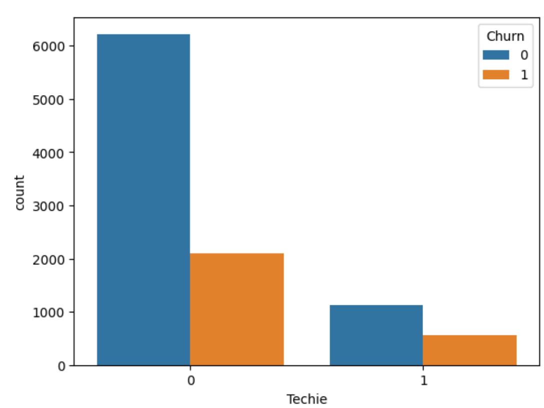
- Outage_sec_perweek

Univariate Visualization	Bivariate Visualization.
	

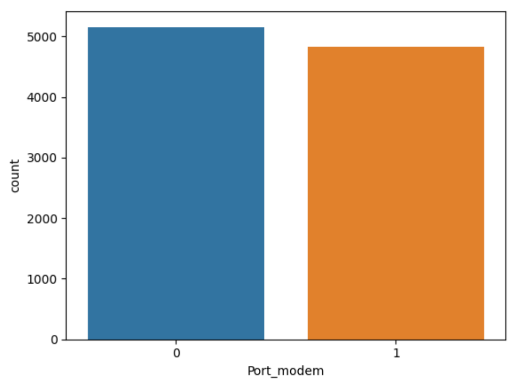
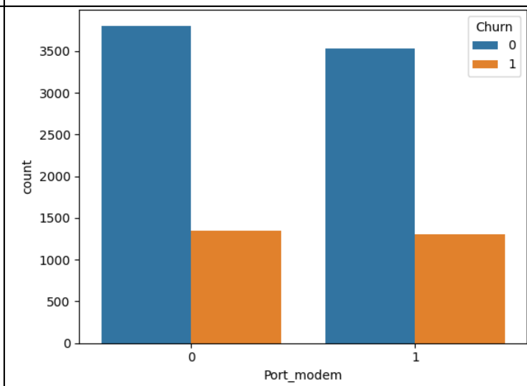
- Yearly_equip_failure

Univariate Visualization	Bivariate Visualization.
	

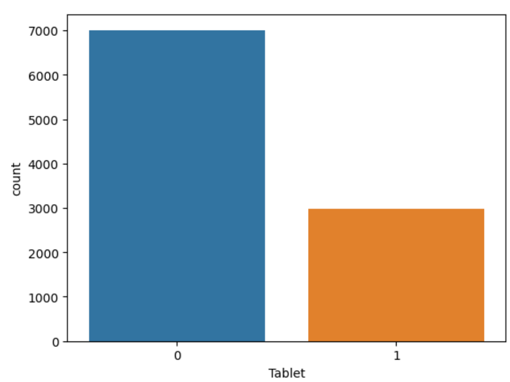
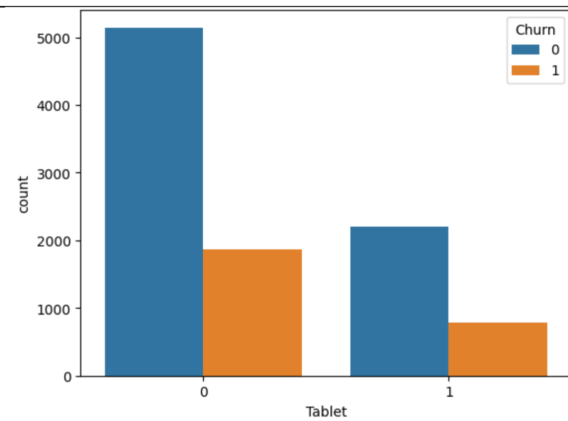
Techie. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

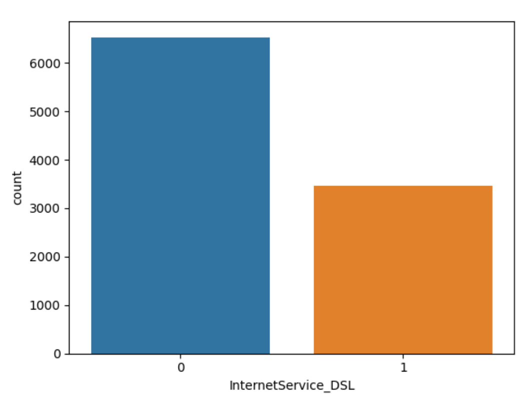
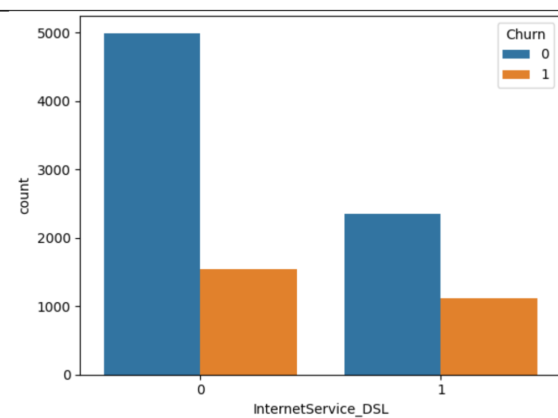
- Port_modem. 0 = No. 1 = Yes.

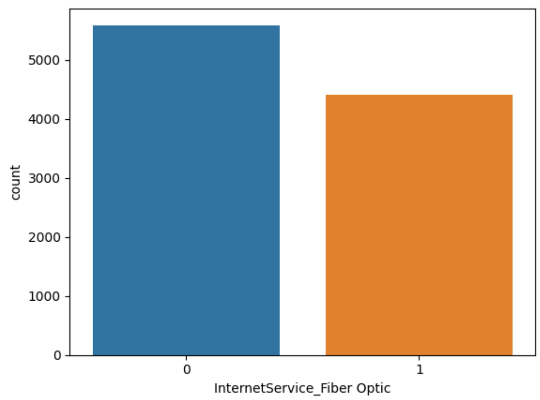
Univariate Visualization	Bivariate Visualization.
	

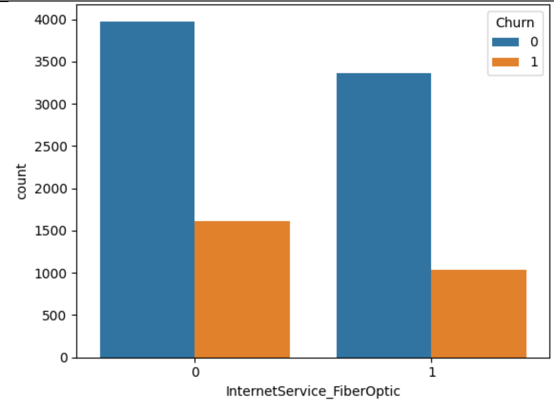
- Tablet. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

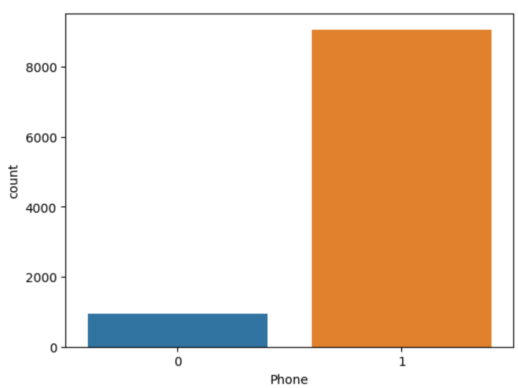
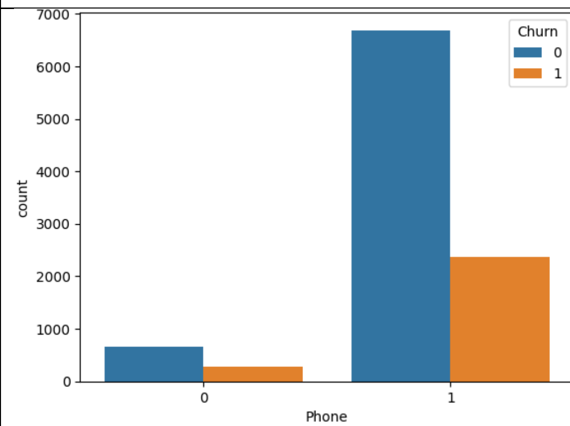
- InternetService. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

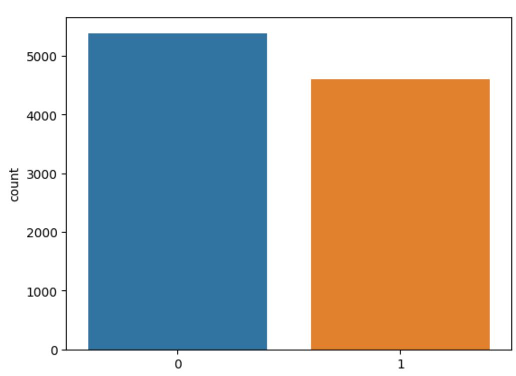
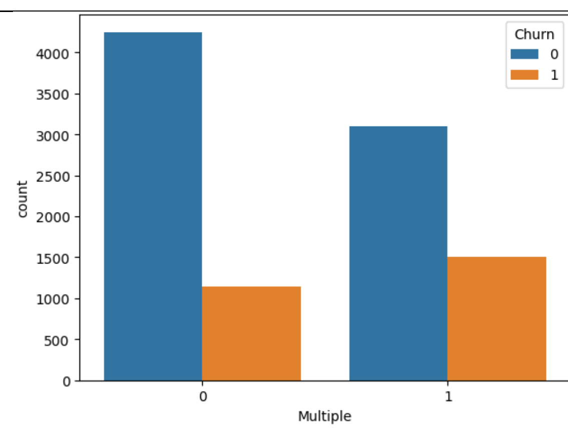




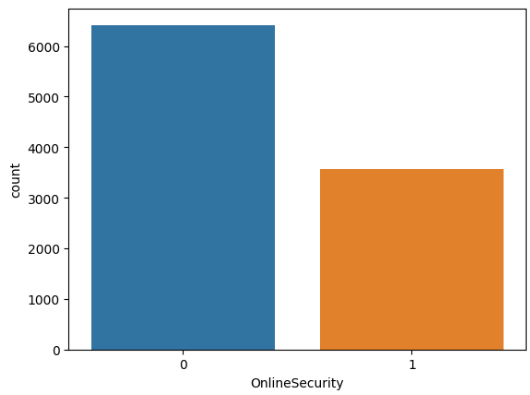
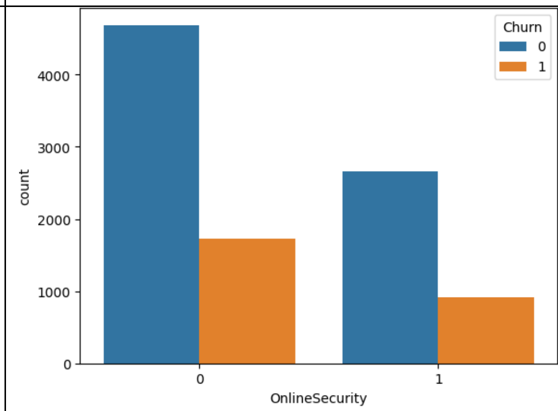
- Phone. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

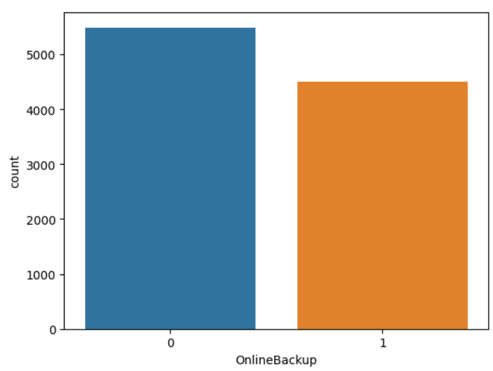
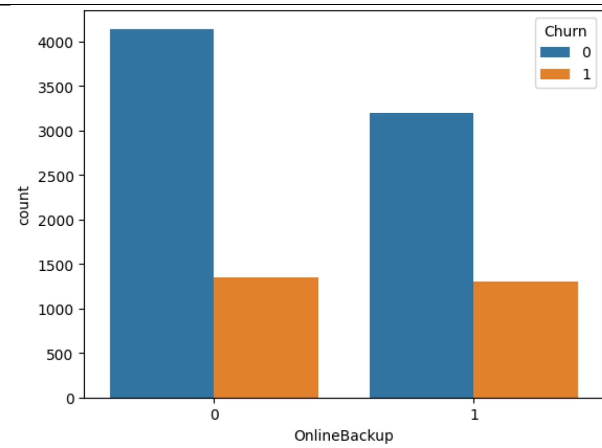
- Multiple. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

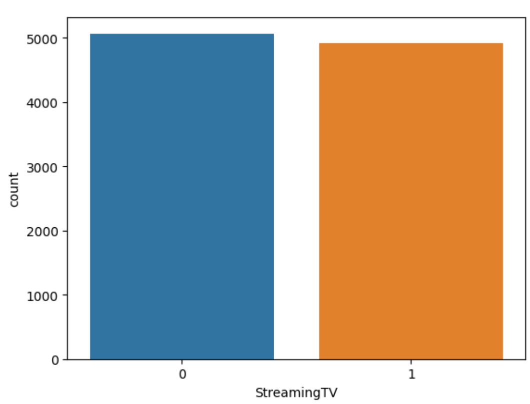
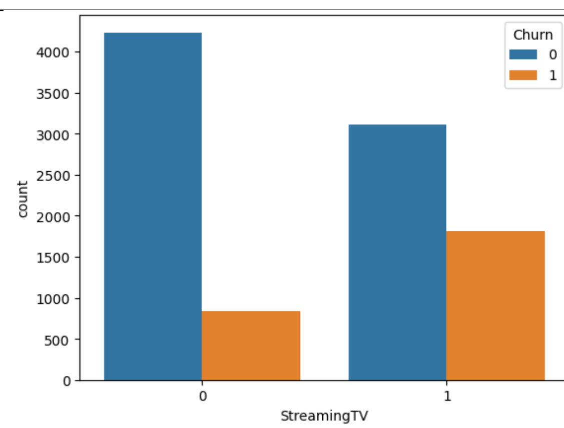
- OnlineSecurity. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

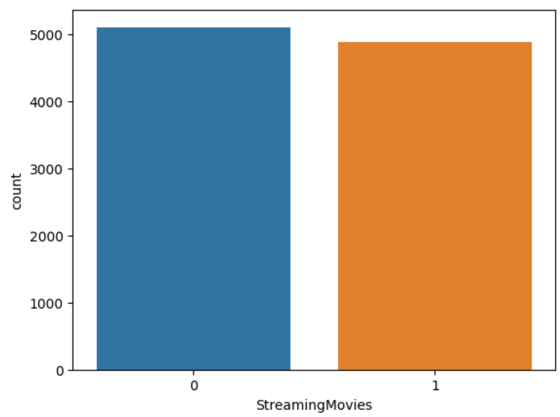
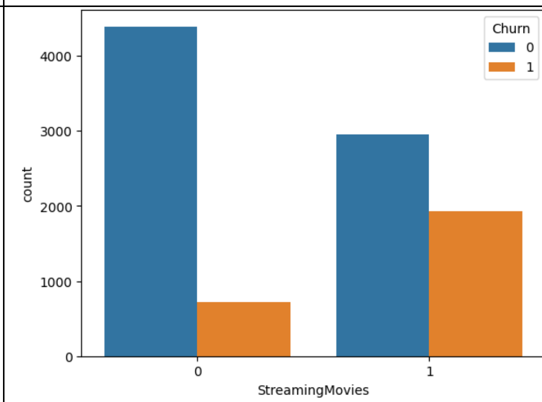
- OnlineBackup. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

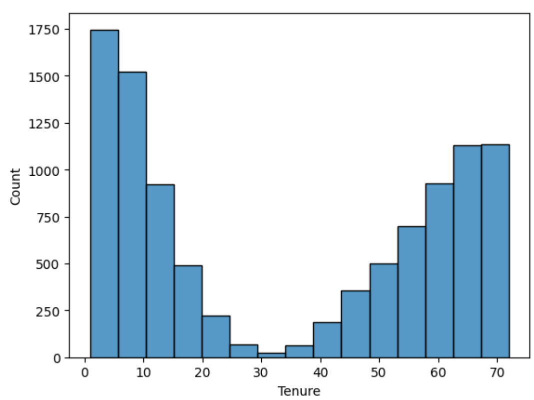
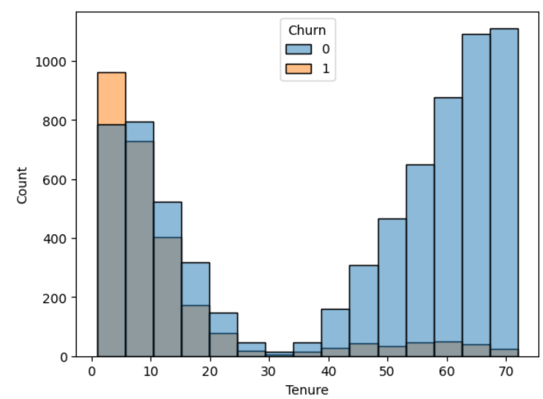
- StreamingTV. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

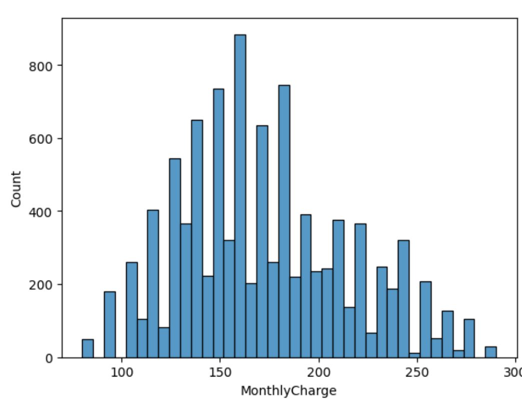
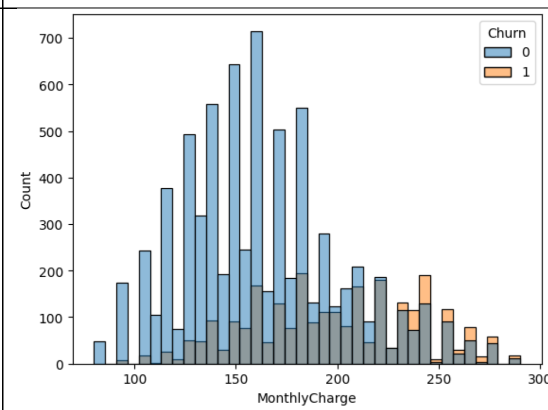
- StreamingMovies. 0 = No. 1 = Yes.

Univariate Visualization	Bivariate Visualization.
	

- Tenure

Univariate Visualization	Bivariate Visualization.
	

- MonthlyCharge

Univariate Visualization	Bivariate Visualization.
	

C4

My data transformation goals included determining which variables to keep. I initially kept all variables that I thought would influence the response variable "Churn". "Job" was one of the variables I thought to include, but there 639 unique values, so I excluded it from the analysis. I also dropped "Martial" for the same reason, too many unique values. I also dropped "Children" so the focus is just the customer, not the household.

	Job
count	10000
unique	639
top	Occupational psychologist
freq	30

```
# drop "Job" variable - too many unique observations
df.drop(columns='Job', inplace=True)
```

```
df['Marital'].value_counts()
```

```
Divorced         2092
Widowed          2027
Separated        2014
Never Married    1956
Married          1911
Name: Marital, dtype: int64
```

```
# drop "Marital" variable - too many unique observations
df.drop(columns='Marital', inplace=True)
df.head()
```

```
df.drop(columns='Children', inplace=True)
df.head()
```

The explanatory variables used in my initial logistic regression are:

- Population
- Age
- Income
- Bandwidth_GB_Year
- Outage_sec_perweek
- Yearly_equip_failure

- Techie
- Port_modem
- Tablet
- InternetService
- Phone
- Multiple
- OnlineSecurity
- OnlineBackup
- StreamingTV
- StreamingMovies
- Tenure
- MonthlyCharge

Other data transformation goals are to re-express categorical variables for use in the regression models. First I re-expressed ordinal categorical data types using ordinal encoding. These included variables with a "yes" or "no" value. All "yes" values were replaced with "1" and all "no" values were replaced with "0". Then I made sure the datatype changed from object to int64.

```
df['Churn'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

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

```
0    7341
1    2650
Name: Churn, dtype: int64
```

```
df['Techie'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['Port_modem'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

```
df['Tablet'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['Phone'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

```
df['Multiple'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['OnlineSecurity'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['OnlineBackup'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['StreamingTV'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
df['StreamingMovies'].replace({'No' : 0, 'Yes' : 1}, inplace=True)
```

```
df['StreamingMovies'].value_counts()
```

```
0    5104
1    4887
Name: StreamingMovies, dtype: int64
```

```
df.dtypes
```

```
Population               int64
Age                      int64
Income                 float64
Churn                    int64
Outage_sec_perweek     float64
Yearly_equip_failure     int64
Techie                   int64
Port_modem               int64
Tablet                   int64
InternetService         object
Phone                    int64
Multiple                 int64
OnlineSecurity           int64
OnlineBackup             int64
StreamingTV              int64
StreamingMovies          int64
Tenure                 float64
MonthlyCharge          float64
Bandwidth_GB_Year      float64
dtype: object
```

Next, I re-expressed nominal categorical data types for the variable "InternetService" using pd.getdummies(). "InternetService" has three values, with no inherent rank.

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

```
array(['Fiber Optic', 'DSL', 'None'], dtype=object)
```

```
df = pd.get_dummies(df, columns=['InternetService'])
```

```
df.head()
```

Port_modem	Tablet	Phone	...	OnlineSecurity	OnlineBackup	StreamingTV	StreamingMovies	Tenure	MonthlyCharge	Bandwidth_GB_Year	InternetService_DSL	InternetService_Fiber Optic	InternetService_None
1	1	1	...	1	1	0	1	6.795513	172.455519	904.536110	0	1	0
0	1	1	...	1	0	1	1	1.156681	242.632554	800.982766	0	1	0
1	0	1	...	0	0	0	1	15.754144	159.947583	2054.706961	1	0	0
0	0	1	...	1	0	1	0	17.087227	119.956840	2164.579412	1	0	0
1	0	0	...	0	0	1	0	1.670972	149.948316	271.493436	0	1	0

C5

A copy of the prepared data set is submitted as a CSV file titled 'prepared_dataset_churn_D208.csv'.

D1

The initial logistic regression model from all independent variables that were identified in part C2 is expressed below.

$P(Y = 1)$ represents the probability that a customer will churn.

$$P(Y = 1) = \frac{1}{1 + e^{-(\boldsymbol{a})}}$$

Where $a = -4.6518$
$+ 0.0004 * Bandwidth\_GB\_Year$
$+ 0.228 * MonthlyCharge$
$+ - 0.000001606 * Population$
$+ 0.0039 * Age$
$+ 0.0000007691 * Income$
$+ - 0.0007 * Outage\_\sec\_perweek$
$+ - 0.0374 * Yearly\_equip\_failure$
$+ 0.7352 * Techie$
$+ 0.1135 * Port\_modem$
$+ - 0.0490 * Tablet$
$+ - 0.2681 * Phone$
$+ 0.3752 * Multiple$
$+ - 0.2455 * OnlineSecurity$
$+ 0.0295 * OnlineBackup$
$+ 1.10013 * StreamingTV$
$+ 1.2221 * StreamingMovies$
$+ - 0.1182 * Tenure$
$+ 0.4758 * InternetService\_DSL$
$+ - 0.7187 * InternetService\_FiberOptic$

D2

I reduced the initial model, represented in part D1, to better align with the research question by engaging in the following tasks.

First, I checked for multicollinearity among the variables. This is not a feature selection procedure, but instead is a method for reducing multicollinearity, an assumption of multiple linear regression. If $VIF > 10$, then there is a presence of multicollinearity and the associated variable was dropped iteratively. In the first iteration, the highest $VIF = 745.3$ for “Bandwidth_GB_Year”. This variable was dropped and VIF was calculated again. In the next iteration “MonthlyCharge” had $VIF = 12.1$ and was dropped. After this, all variables had $VIF < 10$.

Next, I ran the model without "Bandwidth_GB_Year" and "MonthlyCharge". On this step, I engaged in a statistically based feature selection procedure by selecting variables using an iterative backwards elimination method until all variables are statistically significant with a $p\ value\ <\ 0.05$. I removed the variable with the highest p-value greater than 0.05 first, recalculated the model, and repeated until all p-values are less than 0.05. The following variables were removed iteratively:

- "Outage_sec_perweek" $p = 0.850$
- "InternetService_FiberOptic" $p = 0.846$
- "Tablet" $p = 0.488$
- "Yearly_eqiup_failure" $p = 0.458$
- "Population" $p = 0.422$
- "Income" $p = 0.347$
- "Port_modem" $p = 0.098$
- "Age" $p = 0.071$

Now, all variables in the model are statistically significant with $p < 0.05$. The reduced model is provided in part D3.

D3

As a result of the feature selection process described in part D2, the reduced linear regression model is as follows:

$P(Y = 1)$ represents the probability that a customer will churn.

$$P(Y = 1) = \frac{1}{1 + e^{-(\boldsymbol{a})}}$$

$$\begin{aligned} \text{Where } a = &-2.5972 \\ &+ 0.7532 * Techie \\ &+ -0.2941 * Phone \\ &+ 1.1361 * Multiple \\ &+ -0.1416 * OnlineSecurity \\ &+ 0.5727 * OnlineBackup \\ &+ 2.0939 * StreamingTV \\ &+ 2.5250 * StreamingMovies \\ &+ -0.0806 * Tenure \\ &+ 0.9218 * InternetService\_DSL \end{aligned}$$

- Logit Regression Results | Initial Linear Regression Model:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	9991
Model:	Logit	**Df Residuals:**	9971
Method:	MLE	**Df Model:**	19
Date:	Mon, 20 Nov 2023	**Pseudo R-squ.:**	0.4745
Time:	21:35:31	**Log-Likelihood:**	-3037.0
converged:	True	**LL-Null:**	-5779.4
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-4.6518	0.305	-15.269	0.000	-5.249	-4.055
Bandwidth_GB_Year	0.0004	0.000	1.023	0.306	-0.000	0.001
MonthlyCharge	0.0228	0.003	7.440	0.000	0.017	0.029
Population	-1.606e-06	2.3e-06	-0.699	0.484	-6.1e-06	2.89e-06
Age	0.0039	0.002	1.870	0.062	-0.000	0.008
Income	7.691e-07	1.16e-06	0.665	0.506	-1.5e-06	3.04e-06
Outage_sec_perweek	-0.0007	0.011	-0.061	0.951	-0.022	0.021
Yearly_equip_failure	-0.0374	0.051	-0.730	0.465	-0.138	0.063
Techie	0.7352	0.084	8.771	0.000	0.571	0.900
Port_modem	0.1135	0.065	1.756	0.079	-0.013	0.240
Tablet	-0.0490	0.070	-0.698	0.485	-0.187	0.089
Phone	-0.2681	0.109	-2.464	0.014	-0.481	-0.055
Multiple	0.3752	0.111	3.381	0.001	0.158	0.593
OnlineSecurity	-0.2455	0.074	-3.335	0.001	-0.390	-0.101
OnlineBackup	0.0295	0.089	0.332	0.740	-0.145	0.204
StreamingTV	1.0013	0.141	7.099	0.000	0.725	1.278
StreamingMovies	1.2221	0.161	7.611	0.000	0.907	1.537
Tenure	-0.1182	0.033	-3.545	0.000	-0.183	-0.053
InternetService_DSL	0.4758	0.177	2.683	0.007	0.128	0.823
InternetService_FiberOptic	-0.7187	0.132	-5.425	0.000	-0.978	-0.459

- Logit Regression Results | <u>Reduced Linear Regression Model</u>:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	9991
Model:	Logit	**Df Residuals:**	9981
Method:	MLE	**Df Model:**	9
Date:	Wed, 22 Nov 2023	**Pseudo R-squ.:**	0.4668
Time:	13:49:30	**Log-Likelihood:**	-3081.4
converged:	True	**LL-Null:**	-5779.4
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-2.5972	0.138	-18.837	0.000	-2.867	-2.327
Techie	0.7532	0.084	8.991	0.000	0.589	0.917
Phone	-0.2941	0.108	-2.713	0.007	-0.507	-0.082
Multiple	1.1361	0.067	17.035	0.000	1.005	1.267
OnlineSecurity	-0.1416	0.067	-2.111	0.035	-0.273	-0.010
OnlineBackup	0.5727	0.065	8.803	0.000	0.445	0.700
StreamingTV	2.0939	0.074	28.449	0.000	1.950	2.238
StreamingMovies	2.5250	0.076	33.147	0.000	2.376	2.674
Tenure	-0.0806	0.002	-42.341	0.000	-0.084	-0.077
InternetService_DSL	0.9218	0.068	13.551	0.000	0.788	1.055

E1

In my data analysis process I developed a logisitc regression model to determine what factors influence if a customer will churn. I will use the pseudo R^2 value as a model evaluation metric to make a comparison of the initial and reduced logistic regression model. For the initial model pseudo $R^2 = 0.4745$ and for the reduced model pseudo $R^2 = 0.4668$. This indicates that the initial and the reduced models are a good fit of the data (McFadden's Pseudo-R^2 Interpretation, 2020).

Initial Model Calculations

No. Observations:	9991
Df Residuals:	9971
Df Model:	19
Pseudo R-squ.:	0.4745
Log-Likelihood:	-3037.0
LL-Null:	-5779.4
LLR p-value:	0.000

Reduced Model Calculations

No. Observations:	9991
Df Residuals:	9981
Df Model:	9
Pseudo R-squ.:	0.4668
Log-Likelihood:	-3081.4
LL-Null:	-5779.4
LLR p-value:	0.000

E2

The output and all calculations of the analysis I performed for the reduced model are outlined below. Included are the confusion matrix and the accuracy calculation.

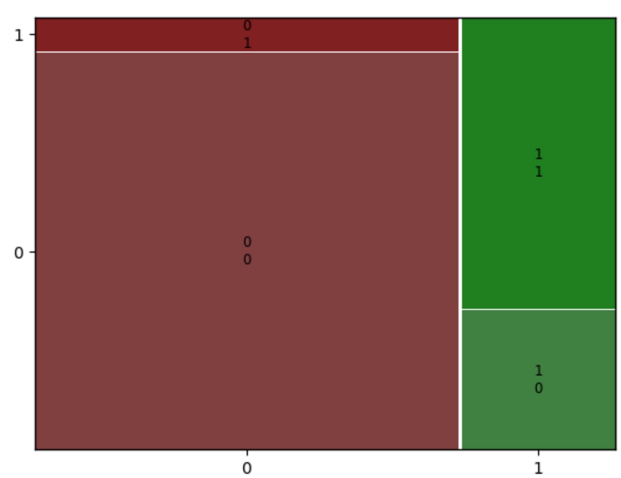
CONFUSION MATRIX

True negative 6758	*False positive* 583
False negative 856	*True positive* 1794

```
conf_matrix = model_reduced9.pred_table()
```

```
print(conf_matrix)
```

```
[[6758.  583.]
 [ 856. 1794.]]
```



ACCURACY CALCULATION

$$Accuracy = \frac{1794 + 6758}{6758 + 583 + 856 + 1764}$$

$$= 0.86$$

```
TN = conf_matrix[0,0]
TP = conf_matrix[1,1]
FN = conf_matrix[1,0]
FP = conf_matrix[0,1]
```

```
accuracy = (TP + TN) / (TN + TP + FN + FP)
print("accuracy = ", accuracy)
```

```
accuracy =  0.8559703733360025
```

```
sensitivity = TP / (FN + TP)
print("sensitivity = ", sensitivity)
```

```
sensitivity =  0.6769811320754717
```

```
specificity = TN / (TN + FP)
print("specificity = ", specificity)
```

```
specificity =  0.920583026835581
```

E3

A copy of the code used to support the implementation of the linear regression model is submitted. There are two code files, “D208 PA Task 1 – Code File Part 1” and “D208 PA Task 1 – Code File Part 4”. The Part 4 file uses the cleaned data set created from Part 1 to create data visualizations and to perform multiple logistic regression.

F1

Here I provide a summary of my findings and assumptions by discussing the logistic regression equation for the reduced model, an interpretation of the coefficients of the reduced model, the statistical and practical significance of the recued model, and the limitations of the data analysis.

Reduced Logistic Model

$P(Y = 1)$ represents the probability that a customer will churn.

$$P(Y = 1) = \frac{1}{1 + e^{-(\boldsymbol{a})}}$$

Where $a = -2.5972$

$$\begin{aligned} &+ 0.7532 * Techie \\ &+ -0.2941 * Phone \\ &+ 1.1361 * Multiple \\ &+ -0.1416 * OnlineSecurity \\ &+ 0.5727 * OnlineBackup \\ &+ 2.0939 * StreamingTV \\ &+ 2.5250 * StreamingMovies \\ &+ -0.0806 * Tenure \\ &+ 0.9218 * InternetService\_DSL \end{aligned}$$

Interpretation of the coefficients

An interpretation for the coefficients of the reduced model is given below where the odds ratio:

Odds Ratio (OR) for a variable $\boldsymbol{X_i}$ with a coefficient of $\boldsymbol{\beta_I}$ is given by $\boldsymbol{OR = e^{\beta_i}}$:

COEFFICIENT * VARIABLE	ODDS RATIO INTERPRETATION
$\boldsymbol{Intercept = -2.5972}$	$e^{-2.5972} = 0.07$ The odds that a customer will churn, when all other variables remain constant is 0.07. In probability terms, there is a 6.8% chance that a customer will churn assuming other variables remain constant.
$\boldsymbol{0.7532 * Techie}$	$e^{0.7532} = 2.12$ The odds that a customer will churn are about 2.12 times higher for customers who reported themselves as a techie, assuming other variables remain constant.

$-\mathbf{0.2941} * \boldsymbol{Phone}$	$e^{-0.2941} = 0.75$ The odds that a customer will churn are about 25% lower for customers who reported themselves as a techie, assuming other variables remain constant.
$\mathbf{1.1361} * \boldsymbol{Multiple}$	$e^{1.1361} = 3.11$ The odds that a customer will churn are about 3.11 times higher for customers who have multiple phone lines, assuming other variables remain constant.
$-\mathbf{0.1416} * \boldsymbol{OnlineSecurity}$	$e^{-0.1416} = 0.87$ The odds that a customer will churn are about 13% lower for customers who have an online security add on, assuming other variables remain constant.
$\mathbf{0.5727} * \boldsymbol{OnlineBackup}$	$e^{0.5727} = 1.77$ The odds that a customer will churn are about 1.77 times higher for customers have an online backup add on, assuming other variables remain constant.
$\mathbf{2.0939} * \boldsymbol{StreamingTV}$	$e^{2.0939} = 8.11$ The odds that a customer will churn are about 8.11 times higher for customers who stream TV, assuming other variables remain constant.
$\mathbf{2.5250} * \boldsymbol{StreamingMovies}$	$e^{2.5250} = 12.49$ The odds that a customer will churn are about 12.49 times higher for customers stream movies, assuming other variables remain constant.
$-\mathbf{0.0806} * \boldsymbol{Tenure}$	$e^{-0.0806} = 0.92$ For every one month increase in tenure, the odds of customers churning are about 8% lower.
$\mathbf{0.9218} * \boldsymbol{InternetService\_DSL}$	$e^{0.9218} = 2.51$ The odds that a customer will churn are about 2.51 times higher for customers who have DSL internet service, assuming other variables remain constant.

Statistical and Practical Significance

The statistical and practical significance of the reduced model are discussed here.

The pseudo R^2 value for the reduced model is 0.4668, displayed in part E1, indicates the model is a good fit for the data (McFadden's Pseudo-R^2 Interpretation, 2020). Additionally all coefficients of the variables in the model are statistically significant with p-values less than 0.05.

The model does have practical significance. We have nine variables that influence customer churn that the company can use to prevent customer churn.

Limitations

Multiple logistic regression has the following assumptions (Assumptions of Logistic Regression, 2023):

- The target variable must be categorical.
- The observations need to be independent of each other.
- Little to no multicollinearity among the independent variables.
- The independent variables are linearly related to the log odds.
- A large sample size is needed.

The limitations in the analysis include: The data set contains outliers which may have an impact on model performance. Also, the independent variables may not be related linearly to the log odds. Furthermore, no interactions amongst independent variables were used in model.

F2

My recommended course of action based on the results my analysis is as follows:

The model can be improved by removing outliers and testing for interactions amongst independent variables in the model.

The model does provide some insight into which factors influence the likelihood of customer churn. The company should put effort into these to prevent customer churn, since it is less expensive to keep a customer than to gain a new one. The variables that have the highest effect size on churn are customers who stream TV and stream movies. The company should look to improve these services and employ some incentives for customers who are streaming to stay with the company. Conversely, if a customer has a phone, this lowers the odds of that a customer will churn. The company should make a concerted effort to sell the phone plan to their customers.

G

A Panopto video recording is provided in the submission of this performance assessment.

H

Web sources used to acquire segments of code to support the application:

"Detecting Multicollinearity with VIF – Python." Geeks for Geeks. www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/

Van den Broeck, Maarten. "Introduction to Regression with statsmodels in Python." Datacamp, app.datacamp.com/learn/courses/introduction-to-regression-with-statsmodels-in-python

Van den Broeck, Maarten. "Intermediate Regression with statsmodels in Python." Datacamp, app.datacamp.com/learn/courses/intermediate-regression-with-statsmodels-in-python

I

I acknowledged sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

References:

"Assumptions of Logistic Regression". Complete Dissertation by Statistics Solutions. 2023. www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/

"McFadden's Pseudo-R^2 Interpretation". Stack Exchange. 2020. stats.stackexchange.com/questions/82105/mcfaddens-pseudo-r2-interpretation

Middleton, Keiona. "Getting Started with D208 Part I." Predictive Modeling – D208, nd, westerngovernorsuniversity.sharepoint.com/sites/DataScienceTeam/Shared%20Documents/Forms/AllItems.aspx?csf=1&web=1&e=9ccodm&cid=3911c4ec%2D7d83%2D4ba9%2Da54b%2D7da9cbb5d38f&FolderCTID=0x01200022092E63FD85A64A8ABFB4F5AEA4839A&id=%2Fsites%2FDataScienceTeam%2FShared%20Documents%2FGraduate%20Team%2FD208%2FStudent%20Facing%20Resources%2FDr%2E%20Middleton%20Getting%20Started%20with%20D208%28Part%20I%29COIT%2Epdf&parent=%2Fsites%2FDataScienceTeam%2FShared%20Documents%2FGraduate%20Team%2FD208%2FStudent%20Facing%20Resources

Middleton, Keiona. “Getting Started with D208 Part II.” Predictive Modeling – D208, nd, westerngovernorsuniversity.sharepoint.com/sites/DataScienceTeam/Shared%20Documents/Forms/AllItems.aspx?csf=1&web=1&e=9ccodm&cid=3911c4ec%2D7d83%2D4ba9%2Da54b%2D7da9cbb5d38f&FolderCTID=0x01200022092E63FD85A64A8ABFB4F5AEA4839A&id=%2Fsites%2FDataScienceTeam%2FShared%20Documents%2FGraduate%20Team%2FD208%2FStudent%20Facing%20Resources%2FDr%2E%20Middleton%20Getting%20Started%20with%20D208%20%28Part%20II%29COIT%2Epdf&parent=%2Fsites%2FDataScienceTeam%2FShared%20Documents%2FGraduate%20Team%2FD208%2FStudent%20Facing%20Resources

Nair, Aashish. “Targeting Multicollinearity With Python.” Medium. December 6, 2021, towardsdatascience.com/targeting-multicollinearity-with-python-3bd3b4088d0b

Van den Broeck, Maarten. “Introduction to Regression with statsmodels in Python.” Datacamp, app.datacamp.com/learn/courses/introduction-to-regression-with-statsmodels-in-python

Van den Broeck, Maarten. “Intermediate Regression with statsmodels in Python.” Datacamp, app.datacamp.com/learn/courses/intermediate-regression-with-statsmodels-in-python

J

Professional communication is demonstrated in the content and presentation of my Performance Assessment.