# D208: Predictive Modeling

**Scott Sullivan | 010547531** **June 12, 2023**

## Task 2

## Part I: Research Question

### A1: Summary Questions

Do the number of children, amount of the monthly charge, tenure, as well as whether the customer has television service or tech shupport have an influence on the likelihood (or probabilty) of dropping the company's service (aka churn).

### A2: Goal Definitions

The author proposes to create a model that can predict with a strong level of confidence whether a customer will churn (drop the service) or stay a customer based on the above characteristics.

## Part II: Justification of Method

### B1: Summarize Logistic Regression Model Assumptions

1. **Categorical/Binary**: The dependent (y) variable in logistic regression is binary. With a binary outcome, the outcome either is or is not a result. In our example, the customer either churns or does not churn.
2. The model is a **logistic (s-shaped) curve** as opposed to a linear model. This represents the probability of the dependent variable being true or not. Generally, a value higher than 0.5 results in a high likelihood of the dependent variable being true.
3. The **value will always be between 0 and 1**.
4. **No outliers**: There will be an absence of strongly influential outliers because they will alter and distort the model's accuracy and, as a result, the outcome (*Web Source 1: Leung*).

### B2: Benefits of Python as the Programming Language of Choice

1. The author chose Python as the model creation language due to the large amount of logistic regression frameworks and simplicity of integrating them into a virtually real-time medium by using Jupyter Notebook (technically JupyterLab).
2. The second benefit of using Python is the generally universal adoption of Python. Knowing that Python is as ubiquitous as it is allows the author to be assured that the model will be able to be shared with any required stakeholders or research colleages.

### B3: Appropriateness of Logistic Regression Technique for Question in A1

The author determined that using logistic regression is an optimal use case for determining the probability of a customer churning due to the binary nature of the outcome. The customer either will or will not churn based on a number of factors.

## Part III: Preparation of Data

### C1. Data Cleaning Goals

The author will clean the provided data several items, including Null values (with Pandas), outliers (with histograms), then dropping unneeded data attributes, converting string values to numerical categorical values (changing a yes/no to 1/0.

Initial examination shows 10,000 records and 50 attributes. The code to import the data and show the initial attributes are as follows. One note, for the first import, the author discovered that one of the three categorical values for Internet Services was "None" (meaning the customer had no internet service). As a result, during the import, pandas incorrectly classified it as a Null value. To correct this, the author excluded "None" as a possible value to trigger the NaN flag.

For each case of Null Values, Duplicates, and Outliers, the author will elaborate on the method and python code used, broken down by section.

import numpy as np  
import pandas as pd  
  
# Read CSV File  
null\_values = ['nan', 'NaN', '']  
file\_path = '\_data/churn\_clean.csv'  
df = pd.read\_csv(file\_path, keep\_default\_na=False, na\_values=null\_values)  
  
# Create dataframe backup  
df\_backup = df  
df.head(3)

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   
  
 UID City State County \  
0 e885b299883d4f9fb18e39c75155d990 Point Baker AK Prince of Wales-Hyder   
1 f2de8bef964785f41a2959829830fb8a West Branch MI Ogemaw   
2 f1784cfa9f6d92ae816197eb175d3c71 Yamhill OR Yamhill   
  
 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   
  
 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 rows x 50 columns]

Before going through the data, the author will first drop variables with demographics that are less meaningful (such as 'CaseOrder', 'Customer\_id', etc.) that will not impact our research qustion. The author will also rename the Item1, Item2, etc to more meaningful names. Finally, the author will change all yes/no and true/false categories to be numerical 1/0 values.

# Dropping variables about demographics that are less meaningful  
# This includes variables such as customer\_id, city, caseOrder, etc  
columns\_to\_drop = ['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'Population',   
 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Marital', 'Area',   
 'TimeZone', 'Job']  
for column in df.columns:  
 if column in columns\_to\_drop:  
 df.drop(column, axis=1, inplace=True)  
 print(f"Dropping {column}")  
print("\nRemaining Attributes:\n")  
df.dtypes

Dropping CaseOrder  
Dropping Customer\_id  
Dropping Interaction  
Dropping UID  
Dropping City  
Dropping State  
Dropping County  
Dropping Zip  
Dropping Lat  
Dropping Lng  
Dropping Population  
Dropping Area  
Dropping TimeZone  
Dropping Job  
Dropping Marital  
  
Remaining Attributes:

Children int64  
Age int64  
Income float64  
Gender object  
Churn object  
Outage\_sec\_perweek float64  
Email int64  
Contacts int64  
Yearly\_equip\_failure int64  
Techie object  
Contract object  
Port\_modem object  
Tablet object  
InternetService object  
Phone object  
Multiple object  
OnlineSecurity object  
OnlineBackup object  
DeviceProtection object  
TechSupport object  
StreamingTV object  
StreamingMovies object  
PaperlessBilling object  
PaymentMethod object  
Tenure float64  
MonthlyCharge float64  
Bandwidth\_GB\_Year float64  
Item1 int64  
Item2 int64  
Item3 int64  
Item4 int64  
Item5 int64  
Item6 int64  
Item7 int64  
Item8 int64  
dtype: object

attr\_to\_make\_binary = ['Techie', 'Churn', 'Port\_modem', 'Phone','Tablet','Multiple','OnlineSecurity', 'DeviceProtection',  
 'OnlineBackup','StreamingTV','TechSupport', 'StreamingMovies', 'PaperlessBilling']  
attr\_to\_rename = {'Item1':'Timely\_Response', 'Item2':'Timely\_Fixes','Item3':'Timely\_Replacements','Item4':'Reliability','Item5':'Options',  
 'Item6':'Respectful','Item7':'Courteous','Item8':'Active\_Listening'}  
  
for stringToChange in attr\_to\_make\_binary:  
 print(f"Changing: {stringToChange}\t")  
 df[stringToChange] = df[stringToChange].replace({"Yes": 1, "No": 0, "True": 1, "False": 0})  
 df[stringToChange] = df[stringToChange].astype(np.int64)  
   
df = df.rename(columns=attr\_to\_rename)  
df.head(5)

Changing: Techie   
Changing: Churn   
Changing: Port\_modem   
Changing: Phone   
Changing: Tablet   
Changing: Multiple   
Changing: OnlineSecurity   
Changing: DeviceProtection   
Changing: OnlineBackup   
Changing: StreamingTV   
Changing: TechSupport   
Changing: StreamingMovies   
Changing: PaperlessBilling

Children Age Income Gender Churn Outage\_sec\_perweek Email \  
0 0 68 28561.99 Male 0 7.978323 10   
1 1 27 21704.77 Female 1 11.699080 12   
2 4 50 9609.57 Female 0 10.752800 9   
3 1 48 18925.23 Male 0 14.913540 15   
4 0 83 40074.19 Male 1 8.147417 16   
  
 Contacts Yearly\_equip\_failure Techie ... MonthlyCharge \  
0 0 1 0 ... 172.455519   
1 0 1 1 ... 242.632554   
2 0 1 1 ... 159.947583   
3 2 0 1 ... 119.956840   
4 2 1 0 ... 149.948316   
  
 Bandwidth\_GB\_Year Timely\_Response Timely\_Fixes Timely\_Replacements \  
0 904.536110 5 5 5   
1 800.982766 3 4 3   
2 2054.706961 4 4 2   
3 2164.579412 4 4 4   
4 271.493436 4 4 4   
  
 Reliability Options Respectful Courteous Active\_Listening   
0 3 4 4 3 4   
1 3 4 3 4 4   
2 4 4 3 3 3   
3 2 5 4 3 3   
4 3 4 4 4 5   
  
[5 rows x 35 columns]

#### Duplicates

First, the author checked for duplicates. This was done using the built in capabilities of pandas using the *duplicated()* method. This method will list any duplicated files, and how many there are.

duplicate\_entries = len(df) - len(df.duplicated())  
print(f"Duplicate Entries: {duplicate\_entries} duplicates, of {len(df)} total.\n")

Duplicate Entries: 0 duplicates, of 10000 total.

#### Nulls

We will now check for any Null values. Instead of using missingno (as the author did for the previous task), we will use the built in pandas *isnull()* function. This function lists each attribute followed by the number of null values in that attribute.

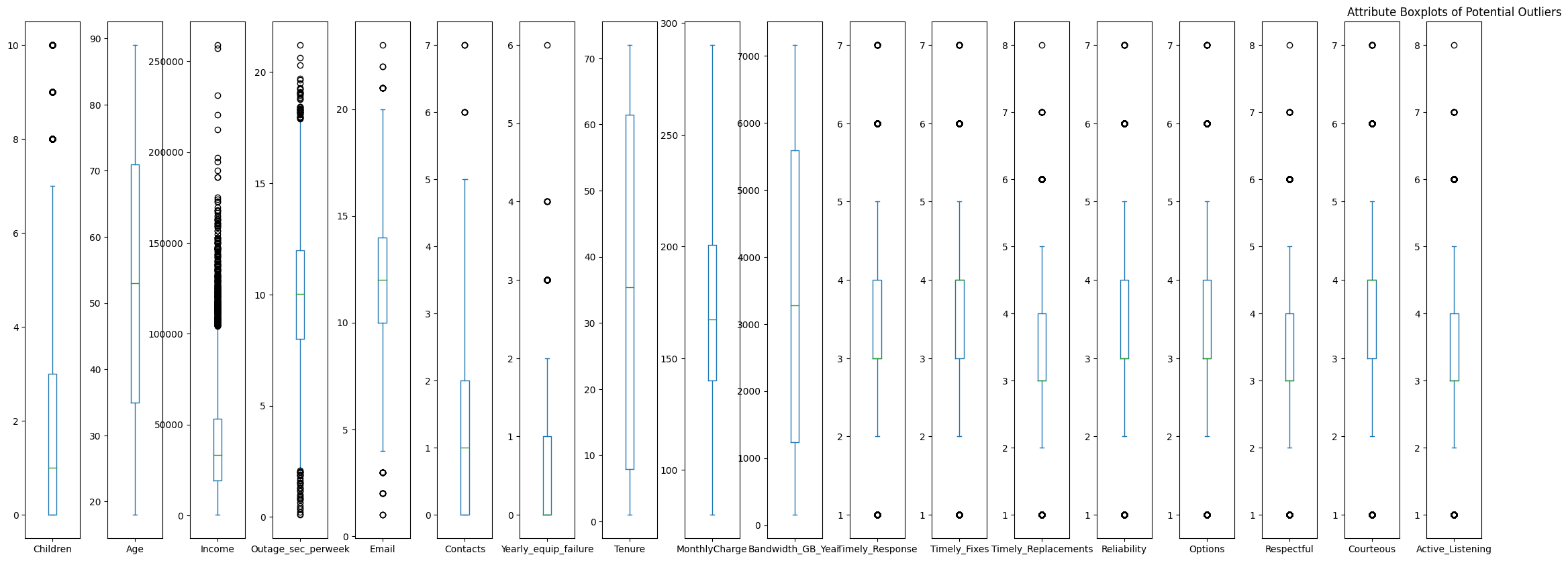
df.isnull().sum()

Children 0  
Age 0  
Income 0  
Gender 0  
Churn 0  
Outage\_sec\_perweek 0  
Email 0  
Contacts 0  
Yearly\_equip\_failure 0  
Techie 0  
Contract 0  
Port\_modem 0  
Tablet 0  
InternetService 0  
Phone 0  
Multiple 0  
OnlineSecurity 0  
OnlineBackup 0  
DeviceProtection 0  
TechSupport 0  
StreamingTV 0  
StreamingMovies 0  
PaperlessBilling 0  
PaymentMethod 0  
Tenure 0  
MonthlyCharge 0  
Bandwidth\_GB\_Year 0  
Timely\_Response 0  
Timely\_Fixes 0  
Timely\_Replacements 0  
Reliability 0  
Options 0  
Respectful 0  
Courteous 0  
Active\_Listening 0  
dtype: int64

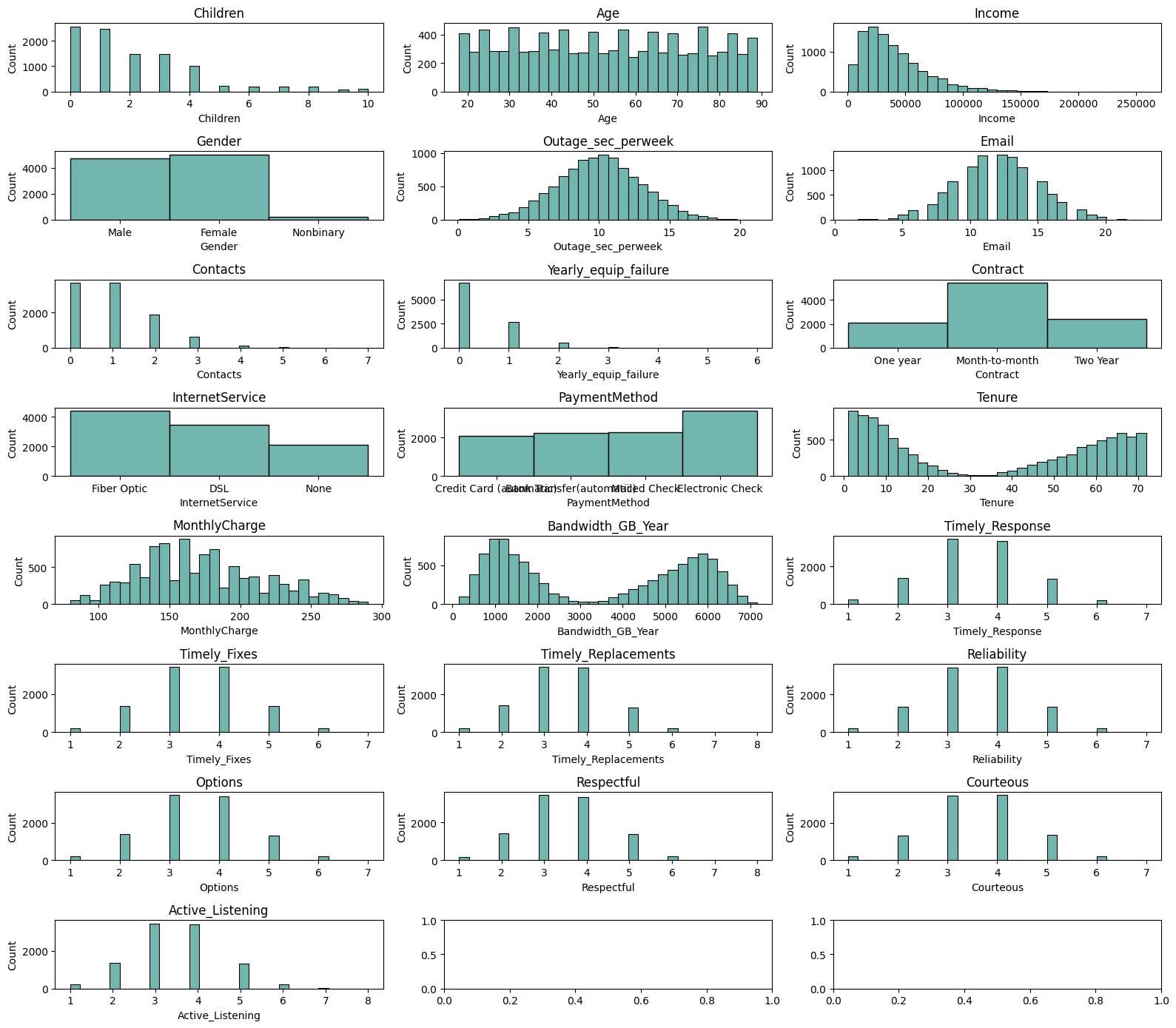
#### Outliers

We will now search for outliers. This will be achieved by creating a boxplot from each variable. As an aside, the author will loop through and ignore any of the binary categories changed above as those values will be limited to either 1 or 0.

# NOTE: this code was previously written and used by the author for an earlier class, D206 and task 1 of D208  
%matplotlib inline  
import matplotlib.pyplot as plt  
import plotnine as p9  
  
df\_tmp\_boxplot = pd.DataFrame()  
for attribute in df.columns:  
 if attribute not in attr\_to\_make\_binary:  
 df\_tmp\_boxplot[attribute] = df[attribute]  
  
df\_tmp\_boxplot.plot(  
 subplots=True,  
 kind='box',  
 sharey=False,  
 figsize=(28, 10)  
)  
# Add a little space for legibility  
plt.subplots\_adjust(wspace=0.5)  
plt.title('Attribute Boxplots of Potential Outliers')  
plt.show()



# Using the same sub-dataframe that skips binary attributes  
from plotnine import ggplot, aes, geom\_histogram  
import seaborn as sns  
  
fig, axs = plt.subplots(nrows=8, ncols=3, figsize=(16,14))  
for ax, var in zip(axs.flatten(), df\_tmp\_boxplot):  
 sns.histplot(data=df, x=var, bins=30, color='#429e94', ax=ax)  
 ax.set\_title(var)  
 ax.set\_xlabel(var)  
 ax.set\_ylabel('Count')  
plt.tight\_layout()  
plt.show()



Upon examination of the above histograms, the author found no outliers in the data. Each of the plots are either normally distributed, skewed (with no outliers), or had a bimodal distribution (with no outliers).

#### One Hot Encoding

The last step to cleaning the data includes changing any of the above attributes that are categorical into a numerical attribute so that it can be incorporated in the calculations. As part of the One Hot Encoding process of accomplishing this, for each possible value in the categories, one will be dropped to prevent multicoliniarity. For example, in the above historgrams, we can see that InternetService has three options, Fiber Optic, DSL, and None. Using One Hot Encoding, each is converted to a column, and one of the three columns is dropped. This is accomplished using the *drop\_first()* function that the author uses below.

# One Hot Encoding  
one\_hot\_attr = ['Gender', 'Contract', 'InternetService', 'PaymentMethod']  
  
for attr in one\_hot\_attr:  
 if attr in df.columns:  
 # Added astype to convert from boolean to numerical  
 y = pd.get\_dummies(df[attr], prefix=attr, drop\_first=True).astype(np.int64)  
 df.drop(attr, axis=1, inplace=True)  
 print(f"Dropped: {attr}\t")  
 df = pd.concat([df, y], axis="columns")  
  
print(df.head(2))  
df.dtypes

Dropped: Gender   
Dropped: Contract   
Dropped: InternetService   
Dropped: PaymentMethod   
 Children Age Income Churn Outage\_sec\_perweek Email Contacts \  
0 0 68 28561.99 0 7.978323 10 0   
1 1 27 21704.77 1 11.699080 12 0   
  
 Yearly\_equip\_failure Techie Port\_modem ... Active\_Listening \  
0 1 0 1 ... 4   
1 1 1 0 ... 4   
  
 Gender\_Male Gender\_Nonbinary Contract\_One year Contract\_Two Year \  
0 1 0 1 0   
1 0 0 0 0   
  
 InternetService\_Fiber Optic InternetService\_None \  
0 1 0   
1 1 0   
  
 PaymentMethod\_Credit Card (automatic) PaymentMethod\_Electronic Check \  
0 1 0   
1 0 0   
  
 PaymentMethod\_Mailed Check   
0 0   
1 0   
  
[2 rows x 40 columns]

Children int64  
Age int64  
Income float64  
Churn int64  
Outage\_sec\_perweek float64  
Email int64  
Contacts int64  
Yearly\_equip\_failure int64  
Techie int64  
Port\_modem int64  
Tablet int64  
Phone int64  
Multiple int64  
OnlineSecurity int64  
OnlineBackup int64  
DeviceProtection int64  
TechSupport int64  
StreamingTV int64  
StreamingMovies int64  
PaperlessBilling int64  
Tenure float64  
MonthlyCharge float64  
Bandwidth\_GB\_Year float64  
Timely\_Response int64  
Timely\_Fixes int64  
Timely\_Replacements int64  
Reliability int64  
Options int64  
Respectful int64  
Courteous int64  
Active\_Listening int64  
Gender\_Male int64  
Gender\_Nonbinary int64  
Contract\_One year int64  
Contract\_Two Year int64  
InternetService\_Fiber Optic int64  
InternetService\_None int64  
PaymentMethod\_Credit Card (automatic) int64  
PaymentMethod\_Electronic Check int64  
PaymentMethod\_Mailed Check int64  
dtype: object

### C2. Dependent and Independent Variables

The dependent variable for this research project is the binary categorical attribute, "Churn." It is possible to be either yes or no in value and signifies whether or not an individual leaves as a customer.

The independent variables include the attributes that are remaining in the pandas dataframe. Specifically, the attributes that will be described with summary statistics are as follows:

'Children': number of children of the customer, 'Age': age of the customer, 'Income': income of the customer (annual), 'Email': how many emails sent to the customer (in last year), 'Contacts': how many times the customer was contacted (in last year), 'Techie': does the customer feel techie?, 'Outage\_sec\_perweek': average seconds per week the system was out, 'Port\_modem': do they have a port modem?, 'Yearly\_equip\_failure': number of customer equipment fails in the past year, 'Tablet': do they own an iPad/surface?, 'Multiple': multple phone lines?, 'OnlineSecurity' does the customer have online security as an add-on service, 'Phone': does the customer have phone service, 'DeviceProtection': did they sign up for device protection service?, 'TechSupport': whether the customer has signed up for tech support service, 'StreamingTV': did the customer sign up for streaming tv, 'StreamingMovies': did they sign up for streaming movies, 'OnlineBackup': did the customer have online backup service, 'PaperlessBilling': did they opt into paperless billing, 'Tenure': how many months has the customer been with the company, 'MonthlyCharge': amount charged each month, 'Timely\_Response': from survey, 'Timely\_Fixes': from survey, 'Timely\_Replacements': from survey, 'Reliability': from survey, 'Options': from survey, 'Respectful': from survey, 'Courteous',: from survey 'Active\_Listening': from survey. 'Gender': customer's gender (split into one hot encoding), 'Contract': the customer's contract term (split into one hot encoding) 'PaymentMethod': how the customer pays (split into one hot encoding) 'Bandwidth\_GB\_Year': total bandwidth used by customer per month

Below is the code and summary statistic describing all of the independent variables that are necessary for answering the research question from A1.

# Independent variables  
import plotnine as p9  
  
independent\_variable\_df = pd.DataFrame()  
independent\_variable\_df = df.drop('Churn', axis=1)  
  
print(independent\_variable\_df.describe())

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   
  
 Email Contacts Yearly\_equip\_failure Techie \  
count 10000.000000 10000.000000 10000.000000 10000.000000   
mean 12.016000 0.994200 0.398000 0.167900   
std 3.025898 0.988466 0.635953 0.373796   
min 1.000000 0.000000 0.000000 0.000000   
25% 10.000000 0.000000 0.000000 0.000000   
50% 12.000000 1.000000 0.000000 0.000000   
75% 14.000000 2.000000 1.000000 0.000000   
max 23.000000 7.000000 6.000000 1.000000   
  
 Port\_modem Tablet ... Active\_Listening Gender\_Male \  
count 10000.000000 10000.000000 ... 10000.000000 10000.000000   
mean 0.483400 0.299100 ... 3.495600 0.474400   
std 0.499749 0.457887 ... 1.028633 0.499369   
min 0.000000 0.000000 ... 1.000000 0.000000   
25% 0.000000 0.000000 ... 3.000000 0.000000   
50% 0.000000 0.000000 ... 3.000000 0.000000   
75% 1.000000 1.000000 ... 4.000000 1.000000   
max 1.000000 1.000000 ... 8.000000 1.000000   
  
 Gender\_Nonbinary Contract\_One year Contract\_Two Year \  
count 10000.000000 10000.000000 10000.000000   
mean 0.023100 0.210200 0.244200   
std 0.150229 0.407471 0.429633   
min 0.000000 0.000000 0.000000   
25% 0.000000 0.000000 0.000000   
50% 0.000000 0.000000 0.000000   
75% 0.000000 0.000000 0.000000   
max 1.000000 1.000000 1.000000   
  
 InternetService\_Fiber Optic InternetService\_None \  
count 10000.000000 10000.000000   
mean 0.440800 0.212900   
std 0.496508 0.409378   
min 0.000000 0.000000   
25% 0.000000 0.000000   
50% 0.000000 0.000000   
75% 1.000000 0.000000   
max 1.000000 1.000000   
  
 PaymentMethod\_Credit Card (automatic) PaymentMethod\_Electronic Check \  
count 10000.000000 10000.000000   
mean 0.208300 0.339800   
std 0.406113 0.473665   
min 0.000000 0.000000   
25% 0.000000 0.000000   
50% 0.000000 0.000000   
75% 0.000000 1.000000   
max 1.000000 1.000000   
  
 PaymentMethod\_Mailed Check   
count 10000.00000   
mean 0.22900   
std 0.42021   
min 0.00000   
25% 0.00000   
50% 0.00000   
75% 0.00000   
max 1.00000   
  
[8 rows x 39 columns]

### C3. Univariate / Bivariate Visualizations

The visualizations for the univariate and bivariate graphs below were generated by iterating the dataframe attributes into lists and then creating the graphic using matplotlib to show the histograms or scatterplots.

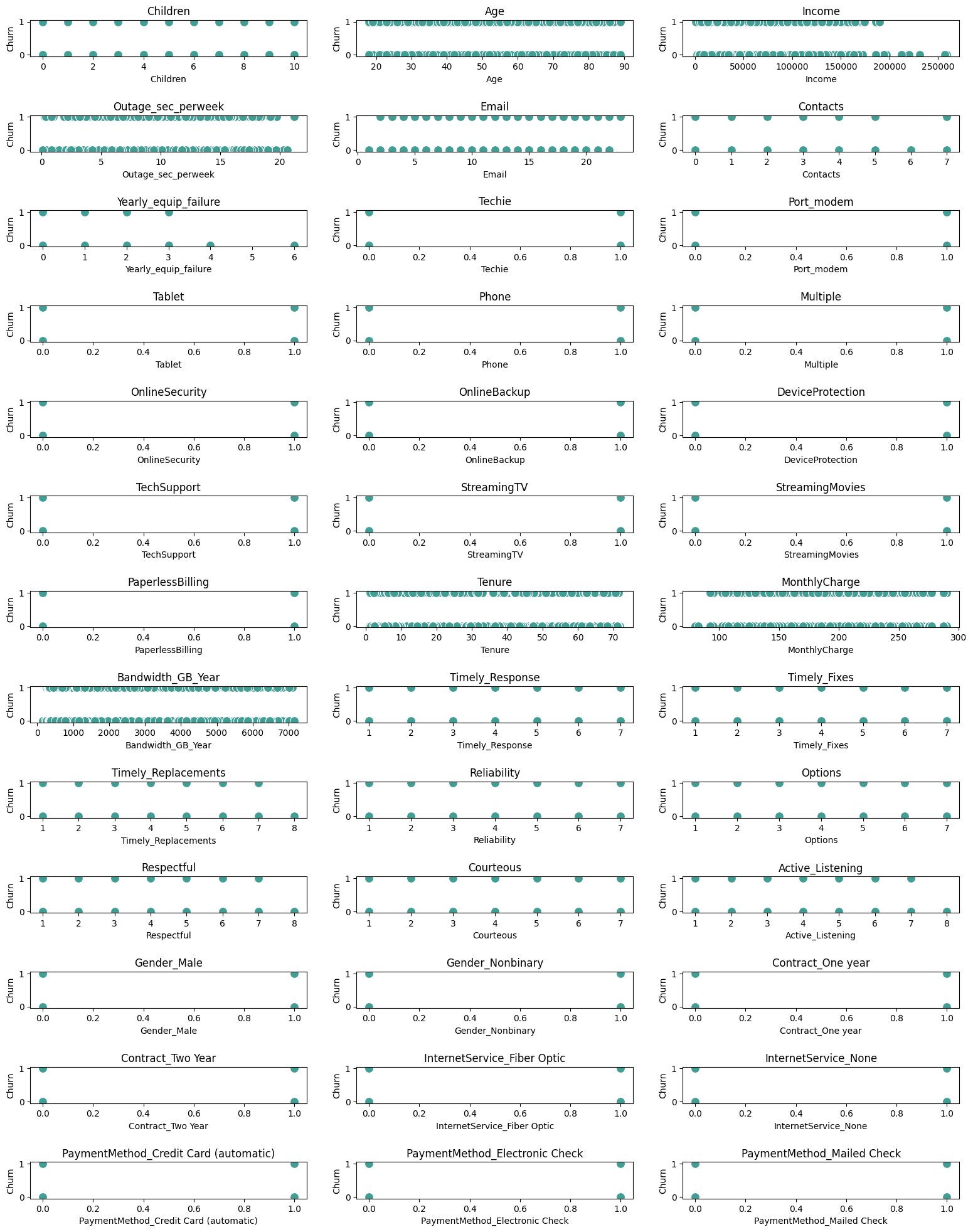
import plotnine as p9  
#import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Creation of Univariate Graphs (using histograms)  
print("Univariate Histograms\n")  
  
independent\_variable\_list = independent\_variable\_df.columns  
  
fig, axs = plt.subplots(nrows=13, ncols=3, figsize=(16,20))  
for ax, var in zip(axs.flatten(), independent\_variable\_list):  
 sns.histplot(data=independent\_variable\_df, x=var, bins=30, color='#429e94', ax=ax)  
 ax.set\_title(var)  
 ax.set\_xlabel(var)  
 ax.set\_ylabel('Count')  
plt.tight\_layout()  
plt.show()

Univariate Histograms



import numpy as np  
import plotnine as p9  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Creation of Bivariate Graphs (using histograms)  
print("Bivariate Visualizations\n")  
  
fig, axs = plt.subplots(nrows=13, ncols=3, figsize=(16,20))  
for ax, var in zip(axs.flatten(), independent\_variable\_list):  
 sns.scatterplot(data=df, x=var, y='Churn', s=100, color='#429e94', ax=ax)  
 ax.set\_title(var)  
 ax.set\_xlabel(var)  
 ax.set\_ylabel('Churn')  
plt.subplots\_adjust(hspace = 0.15)  
plt.tight\_layout(pad=2.0)  
plt.show()

Bivariate Visualizations



### C4.Transformation Goals

For the second task, we focus on the what variables (if any) can predict the probability of the 'churn' of a customer. This attribute (whether or not a customer dropped the service in the past month) is our dependent variable. This is a categorical attribute (specifically, binary) indicating whether or not the customer has churned, indicated by a 1 or 0 representing that they have or have not churned, respectively.

**Data transformation used:** First, the author cleaned the data in preparation. Any null values were removed, "yes/no" and "true"/"false" attribute values were replaced with numerical values of 1/0, respectively. For the boolean true/false and yes/no cases, these strings were replaced using python to numerical 1 and 0 values (using a *replace()* on each column with these values. This code was used in C1:

df[stringToChange] = df[stringToChange].replace({"Yes": 1, "No": 0, "True": 1, "False": 0})

Missing values were also searched for and prepared to be cleaned. However, there were no missing values and so no cleaning was needed for this transformation.

Categorical data was transformed with one-hot encoding so that instead of strings, we had numerical values (in this case 1 or 0 for each dummy column added). In order to transform this data, the author used the pandas code from Section C1 above:

pd.get\_dummies(df[attr], prefix=attr, drop\_first=True).astype(np.int64)

**This transformation** left us with a clean set of data, ready for proper analysis on the dependent variable, 'Churn.' The author also checekd that all the needed data for this analysis was numerical and not in an object/string encoding. The author also checked for outliers, though none were found, using a histogram to inspect the data distribution.

In addition to the code listed in the preceeding paragraphs, full annotated code is included in section C1, in the respective transformation sub-section.

### C5. Data Export

The prepared data is exported to a CSV using the pandas function *to\_csv()* as shown here:

# Export Prepared Data to CSV  
file\_name = '\_data/d208\_churn\_task\_2\_cleaned.csv'  
df.to\_csv(file\_name, index=False)  
print(f"Exported to {file\_name}")

Exported to \_data/d208\_churn\_task\_2\_cleaned.csv

## Part IV: Analysis and Comparison of Model

### D1. Initial Logistic Regression Model

Construct an initial logistic regression model from all independent variables that were identified in part C2.

from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  
  
X = independent\_variable\_df  
y = df['Churn']  
  
X\_tr, X\_te, y\_tr, y\_te = train\_test\_split(X, y, test\_size=0.25, random\_state=3)  
log\_reg = LogisticRegression(max\_iter=500)  
  
log\_reg.fit(X\_tr, y\_tr) # Fit model on x\_ and y\_train

LogisticRegression(max\_iter=500)

#### Confusion Matrix Generation

y\_pred = log\_reg.predict(X\_te) # Calculate predictions from x\_te  
  
print(f"Accuracy: {accuracy\_score(y\_te, y\_pred)}\n\nConfusion Matrix:\n{confusion\_matrix(y\_te, y\_pred)}\n")  
print(f"Classification Report:\n{classification\_report(y\_te, y\_pred)}")

Accuracy: 0.8328  
  
Confusion Matrix:  
[[1687 160]  
 [ 258 395]]  
  
Classification Report:  
 precision recall f1-score support  
  
 0 0.87 0.91 0.89 1847  
 1 0.71 0.60 0.65 653  
  
 accuracy 0.83 2500  
 macro avg 0.79 0.76 0.77 2500  
weighted avg 0.83 0.83 0.83 2500

### D2. Dataset Reduction

The author examined the data and performed a Recursive Feature Elimination (RFE) in order to find the more important attributes affecting the churn variable.

The author experimented with reducing the model down to different sizes and found that reducing it anywhere from 4-10 features resulted in a very similar final accuracy score. That said, the reduced model was less accurate, at 77% as seen below, where the full feature set had an accuracy score of 87% as seen above. For this reason, the author determined that the model could be reduced to 4 features and retain a very similar accuract as if we had reduced it to 10 features.

from sklearn.feature\_selection import RFE  
  
reg\_model = LogisticRegression(solver='liblinear')  
rfe = RFE(reg\_model, n\_features\_to\_select=4)  
fit\_model = rfe.fit(X, y)  
  
print(f"Number of chosen features: {fit\_model.n\_features\_}")

Number of chosen features: 4

#### Build temp dataframe in order to display rankings with column names

feature\_ranking = pd.DataFrame({'Feature': X.columns,   
 'Rank': fit\_model.ranking\_,   
 'IsSelected': fit\_model.support\_})  
print(feature\_ranking.sort\_values(by='Rank'))

Feature Rank IsSelected  
33 Contract\_Two Year 1 True  
32 Contract\_One year 1 True  
17 StreamingMovies 1 True  
16 StreamingTV 1 True  
11 Multiple 2 False  
34 InternetService\_Fiber Optic 3 False  
35 InternetService\_None 4 False  
7 Techie 5 False  
37 PaymentMethod\_Electronic Check 6 False  
10 Phone 7 False  
30 Gender\_Male 8 False  
15 TechSupport 9 False  
12 OnlineSecurity 10 False  
18 PaperlessBilling 11 False  
38 PaymentMethod\_Mailed Check 12 False  
36 PaymentMethod\_Credit Card (automatic) 13 False  
19 Tenure 14 False  
8 Port\_modem 15 False  
31 Gender\_Nonbinary 16 False  
9 Tablet 17 False  
6 Yearly\_equip\_failure 18 False  
5 Contacts 19 False  
27 Respectful 20 False  
25 Reliability 21 False  
26 Options 22 False  
20 MonthlyCharge 23 False  
22 Timely\_Response 24 False  
14 DeviceProtection 25 False  
29 Active\_Listening 26 False  
4 Email 27 False  
28 Courteous 28 False  
13 OnlineBackup 29 False  
0 Children 30 False  
3 Outage\_sec\_perweek 31 False  
24 Timely\_Replacements 32 False  
23 Timely\_Fixes 33 False  
1 Age 34 False  
21 Bandwidth\_GB\_Year 35 False  
2 Income 36 False

#### Create dataframe of just the selected features from the previous codeblock

selected\_features = feature\_ranking[feature\_ranking['IsSelected'] == True]['Feature'].tolist()  
reduced\_df = independent\_variable\_df[selected\_features]  
reduced\_df.shape

(10000, 4)

### D3. Reduced Logistic Regression

Upon finding a reduced selection of features in the previous section, the author retrained the logistic regression model on this smaller model and ran the same accuracy score, confusion matrix, and classification report (Web Source 2: Brownlee)

X = reduced\_df  
y = df['Churn']  
  
X\_tr, X\_te, y\_tr, y\_te = train\_test\_split(X, y, test\_size=0.25, random\_state=3)  
log\_reg = LogisticRegression(max\_iter=1000)  
  
log\_reg.fit(X\_tr, y\_tr) # Fit model on x\_ and y\_train  
y\_pred = log\_reg.predict(X\_te) # Calculate predictions from x\_test  
  
print(f"Accuracy: {accuracy\_score(y\_te, y\_pred)}\n\nConfusion Matrix:\n{confusion\_matrix(y\_te, y\_pred)}\n")  
print(f"Classification Report:\n{classification\_report(y\_te, y\_pred)}")

Accuracy: 0.7756  
  
Confusion Matrix:  
[[1733 114]  
 [ 447 206]]  
  
Classification Report:  
 precision recall f1-score support  
  
 0 0.79 0.94 0.86 1847  
 1 0.64 0.32 0.42 653  
  
 accuracy 0.78 2500  
 macro avg 0.72 0.63 0.64 2500  
weighted avg 0.76 0.78 0.75 2500

#### Calculate and print the coefficients

The coefficients indicate the odds ratio for the change in change value (our dependent variable) in relation to the independent variable. A positive value indicates it drives it higher, and a negative value drives it lower. And the higher the number, the more it affects the dependent variable. In our example below, the StreamingMovies coefficient of 1.6 indicates that the odds of a customer churning is higher than if they only had StreamingTV service (though the coefficeints are close).

# Calculate and print the coefficients  
feature\_coef = pd.DataFrame({"Feature":X.columns,"Coefficients":np.transpose(log\_reg.coef\_[0])})  
print(feature\_coef)

Feature Coefficients  
0 StreamingTV 1.401067  
1 StreamingMovies 1.629446  
2 Contract\_One year -1.483701  
3 Contract\_Two Year -1.713168

As we can see, the Contract\_One year and Contract\_Two Year both have negative coefficients, which means that both will reduce the odds of a customer churning, with the latter option retaining a customer more than the former, with a coefficient of -1.4 and -1.7, respectively.

### E1. Data Analysis of Initial versus Reduced Model

The initial logistic regression model used above contained the full set of independent variables and was run against the dependent variable, 'Churn' to determine what the odds where of a customer churning from those attributes. We then generated plots with a univariate and bivariate histogram to see if there were any obvious relationships.

Finally, the author ran a confusion matrix and a classification report to understand if the initial model was accurate, and if so, how much. Additionally, it gave us a weighted average to compensate for the fact that there were over twice as many customers who did not churn as those who did. This was shown in the 'weighted avg' indicator. To avoid possible overfitting, the author reduced the model.

The model was reduced by running a Recursive Feature Elimination (RFE) to bring our list of attributes down to just a few. As mentioned above, the author experimented with *n\_features\_to\_select* values beginning with 10 and reduced the model further while checking to ensure the accuracy remained similar. The author found 4 to be the smallest number while retaining a relatively high accuracy.

#### Model Evaluation Metric

The primary evaluation metric was the accuracy of the model. As a recap, here are the results repeated from above.

print(f"Accuracy: {accuracy\_score(y\_te, y\_pred)}\n\nConfusion Matrix:\n{confusion\_matrix(y\_te, y\_pred)}\n")  
print(f"Classification Report:\n{classification\_report(y\_te, y\_pred)}")

Accuracy: 0.7756  
  
Confusion Matrix:  
[[1733 114]  
 [ 447 206]]  
  
Classification Report:  
 precision recall f1-score support  
  
 0 0.79 0.94 0.86 1847  
 1 0.64 0.32 0.42 653  
  
 accuracy 0.78 2500  
 macro avg 0.72 0.63 0.64 2500  
weighted avg 0.76 0.78 0.75 2500

As you can see, **accuracy** is the most basic evaluation metric. The accuracy of 0.78 was given above. However, it must be noted that this dataset was heavily weighted with twice as many customers who did not churn (1847 total) and 653 who did. The weighted average is 0.75 and should be noted that, by definition, this value favors the majority class (no churn).

### E2. Output of Analysis

For the final reduced model for the output is included below (the code sections below are a copy of the original code above, and the original non-reduced model can be seen in section D1). The final reduced dataframe is as follows:

print(f"Dependent variable:\nChurn\t{df['Churn'].dtypes}\n")  
print(f"Reduced independent variables:\n{reduced\_df.dtypes}")

Dependent variable:  
Churn int64  
  
Reduced independent variables:  
StreamingTV int64  
StreamingMovies int64  
Contract\_One year int64  
Contract\_Two Year int64  
dtype: object

reduced\_df.head(5)

StreamingTV StreamingMovies Contract\_One year Contract\_Two Year  
0 0 1 1 0  
1 1 1 0 0  
2 0 1 0 1  
3 1 0 0 1  
4 1 0 0 0

#### Accuracy & Confusion Matrix

X = reduced\_df  
y = df['Churn']  
  
X\_tr, X\_te, y\_tr, y\_te = train\_test\_split(X, y, test\_size=0.25, random\_state=3)  
log\_reg = LogisticRegression(max\_iter=1000)  
  
log\_reg.fit(X\_tr, y\_tr) # Fit model on x\_ and y\_train  
y\_pred = log\_reg.predict(X\_te) # Calculate predictions from x\_test  
  
print(f"Accuracy: {accuracy\_score(y\_te, y\_pred)}\n\nConfusion Matrix:\n{confusion\_matrix(y\_te, y\_pred)}\n")  
print(f"Classification Report:\n{classification\_report(y\_te, y\_pred)}")

Accuracy: 0.7756  
  
Confusion Matrix:  
[[1733 114]  
 [ 447 206]]  
  
Classification Report:  
 precision recall f1-score support  
  
 0 0.79 0.94 0.86 1847  
 1 0.64 0.32 0.42 653  
  
 accuracy 0.78 2500  
 macro avg 0.72 0.63 0.64 2500  
weighted avg 0.76 0.78 0.75 2500

# Calculate and print the coefficients  
feature\_coef = pd.DataFrame({"Feature":X.columns,"Coefficients":np.transpose(log\_reg.coef\_[0])})  
print(feature\_coef)

Feature Coefficients  
0 StreamingTV 1.401067  
1 StreamingMovies 1.629446  
2 Contract\_One year -1.483701  
3 Contract\_Two Year -1.713168

### E3. Implementation of Python Code

The author has executed (and shown) error-free python code throughout this Jupyter Notebook file.

## Part V: Data Summary and Implications

### F1. Results of Analysis Findings

Through analysis of the dataset examining the factors that may play into and/or predict whether or not a customer will churn, the following calculations were made:

#### Reduced Model Regression Equation

Using the following coefficients:

* StreamingTV = 1.401067
* StreamingMovies = 1.629446
* Contract\_One year = -1.483701
* Contract\_Two Year = -1.713168 The following base equation can be written:
* log(odds of Churn) = b0 + b1 \* StreamingTV + b2 \* StreamingMovies + b3 \* Contract\_One year + b4 \* Contract\_Two year
* Substituting the numerical values for the variables above, we get:
* log(odds of Churn) = b0 + 1.4(TV) + 1.6(StreamingMovies) - 1.5(Contract\_One year) - 1.7(Contract\_Two year)

#### Reduced Model Coefficient Description

The details of the coefficients were touched upon in D3, but is included here for a more concise paper. The coefficients indicate the odds ratio for the change in change value (our dependent variable) in relation to the independent variable. A positive value indicates it drives it higher, and a negative value drives it lower. And the higher the number, the more it affects the dependent variable. In our model, the StreamingMovies coefficient of 1.6 indicates that the odds of a customer churning is higher than if they only had StreamingTV service (though the coefficeints are close).

#### Statistical & Practical Significance of Reduced Model

For the statistical significance, we will examine the BIC, or the Bayesian Information Criterion. This was chosen due to it being a part of statsmodels framework, a framework already used for this class.

import statsmodels.api as sm  
  
X = sm.add\_constant(X)  
log\_reg = sm.Logit(y, X).fit()  
  
# Calculate BIC  
bic = log\_reg.bic  
print(f"The BIC is {bic}")

Optimization terminated successfully.  
 Current function value: 0.459304  
 Iterations 6  
The BIC is 9232.128094366471

The practical significance in this case indicates the usefulness of our model, specifically, how it relates to real world data. In our specific reduced model, the practical significance relates to its ability to accurately predict churn, calculated on our features (StreamingTV, StreamingMovies, Contract\_One year, Contract\_Two year).

The accuracy of the model was approximately 0.78, meaning it correctly predicted churn in about 78% of cases in the test dataset. This level of accuracy may be very useful in a business context, as it can help the company to identify customers who are at risk of churn and possibly intervene to prevent it.

#### Limitations

One of the limitations found with this process was the possibility of underfitting or overfitting. The author attempted to reduce the model, reducing the liklihood of overfitting and not being able to predict new data. However, there is a possibility of underfitting, that we reduced the model too much, and the model no longer accurately models the data.

### F2. Recommendations

One immmediate recommendation, given the results of the logistic model above, would be to find a way to convert customers who are paying their bills monthly to one of the two annual payment plans, as customers on the annual payment plans showed significantly lower odds of churn.

## Part VI: Demonstration

### G. Panopto video

The panopto video can be found at the following link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6cc1a1cc-feb4-4a30-af5b-b0230022cece>

### H. Web Sources

List the web sources used to acquire data or segments of third-party code to support the application. Ensure the web sources are reliable.

Source 1:

* Leung, Kenneth. *"Assumptions of Logistic Regression, Clearly Explained"* <https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290>

Source 2:

* Brownlee, Jason, PhD. *"Recursive Feature Elimination (RFE) for Feature Selection in Python"* <https://machinelearningmastery.com/rfe-feature-selection-in-python/>

### I. In-text citations and references

Source 3:

* Verbiest, Nele. *"Introduction to Predictive Analytics in Python."* (Datacamp course material for D208)

### J. Professionalism

Illustrated throughout the Panopto video and throughout this document.