# D208: Predictive Modeling

**Scott Sullivan | 010547531** **June 9, 2023**

## Task 1

## Part I: Research Question

### A1: Summary Questions

What factors influence high bandwidth (as described by the attribute “Bandwidth\_GB\_Year”) used by customers during a year? The author proposes to find the factors (if any), in the dataset that could potentially have an impact on bandwidth (and by extension, the health and profitability of the network).

### A2: Define Goals

The first goal of this analysis is to understand customer behavior, specifically, factors that influence their bandwidth consumption. Through this, the author will better understand the customer base and usage patterns. These insights will be able to be used to develop more effective marketing strategies to increase customer satisfaction and retention.

Also, through the understanding of which factors influence 'Bandwidth\_GB\_Year', we can improve our product to better meet customer needs. For example, if certain attributes or behaviors correlate with higher bandwidth, the company could potentially develop new products or modify existing products or service bundles that would be aimed at this customer segment and increase revenue.

## Part II: Method Justification

### B1: Four assumptions regarding multiple linear regression models

Below are multiple assumptions the author is making in regards to the multiple linear regression model.

1. **Linear Relationship**: Our variables of interest must be shown to have a linear relationship. This will be verified with a scatterplot.
2. **Target Variable is Continuous**: We must ensure that the dependent variable (our y-axis) is continuous.
3. **Multiple independent variables**: While the dependent variable must be continuous, the independent variables may be continuous or categorical means, but we must have multiple (2+) variables.
4. **Independence**: Our fourth assumption is to assume that the observations are independent of each other.

### B2: Python Benefits Regarding This Analysis

1. Python (via Jupyter Notebooks) has a broad range of libraries and frameworks that can be utilized by the author to analyze large datasets. Additionally, it is cross-platform, thereby allowing the code to be run on multiple platforms.
2. Python is also very powerful in visualization of datasets. Prior to analyzing and during the analysis, it will be beneficial to get a better view of the data, and Python allows us to graph and visualize our datasets to better extract information.

### B3: What makes multiple linear regression appropriate for this technique

The author believes that making use of multiple linear regression will be appropriate for a number of reasons. Primarily, the dependent variable being used for this analysis (bandwidth) is continuous, which is a key requirement for this model.

Additionally, multiple linear regression excels because while it will identify the independent and dependent variable relationships, it is also capable of quantifying the relationships. As a result, the author will be able to determine the predicted change in 'Bandwidth\_GB\_Year' as a direct result of a change in one of the independent variables while maintaining the other variables constant. This can be useful when planning and quantifying potential courses of actions.

## Part III: Preparation of Data

### C1: Data cleaning goals & steps to clean data (includes annotated code)

The author plans to clean the data by searching for Null values (using Pandas), outliers (using histograms), removing unnecessary data attributes, converting categorical values (such as yes or no) into numerical values (1 or 0 respectively) and finally relabeling attributes to be more descriptive.

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.

import numpy as np  
import pandas as pd  
  
# Read the CSV  
na\_values = ['NaN', 'nan', '']  
csvfile = '\_data/churn\_clean.csv'  
df = pd.read\_csv(csvfile, keep\_default\_na=False, na\_values=na\_values)  
  
# df = pd.read\_csv('\_data/churn\_clean.csv')  
  
# Copy the dataframe to RAW in case we need to come back to the raw data  
df\_raw = df  
df.head(2)

CaseOrder Customer\_id Interaction \  
0 1 K409198 aa90260b-4141-4a24-8e36-b04ce1f4f77b   
1 2 S120509 fb76459f-c047-4a9d-8af9-e0f7d4ac2524   
  
 UID City State County \  
0 e885b299883d4f9fb18e39c75155d990 Point Baker AK Prince of Wales-Hyder   
1 f2de8bef964785f41a2959829830fb8a West Branch MI Ogemaw   
  
 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   
  
 Item2 Item3 Item4 Item5 Item6 Item7 Item8   
0 5 5 3 4 4 3 4   
1 4 3 3 4 3 4 4   
  
[2 rows x 50 columns]

#### Duplicates

We now examine the dataframe for duplicates:

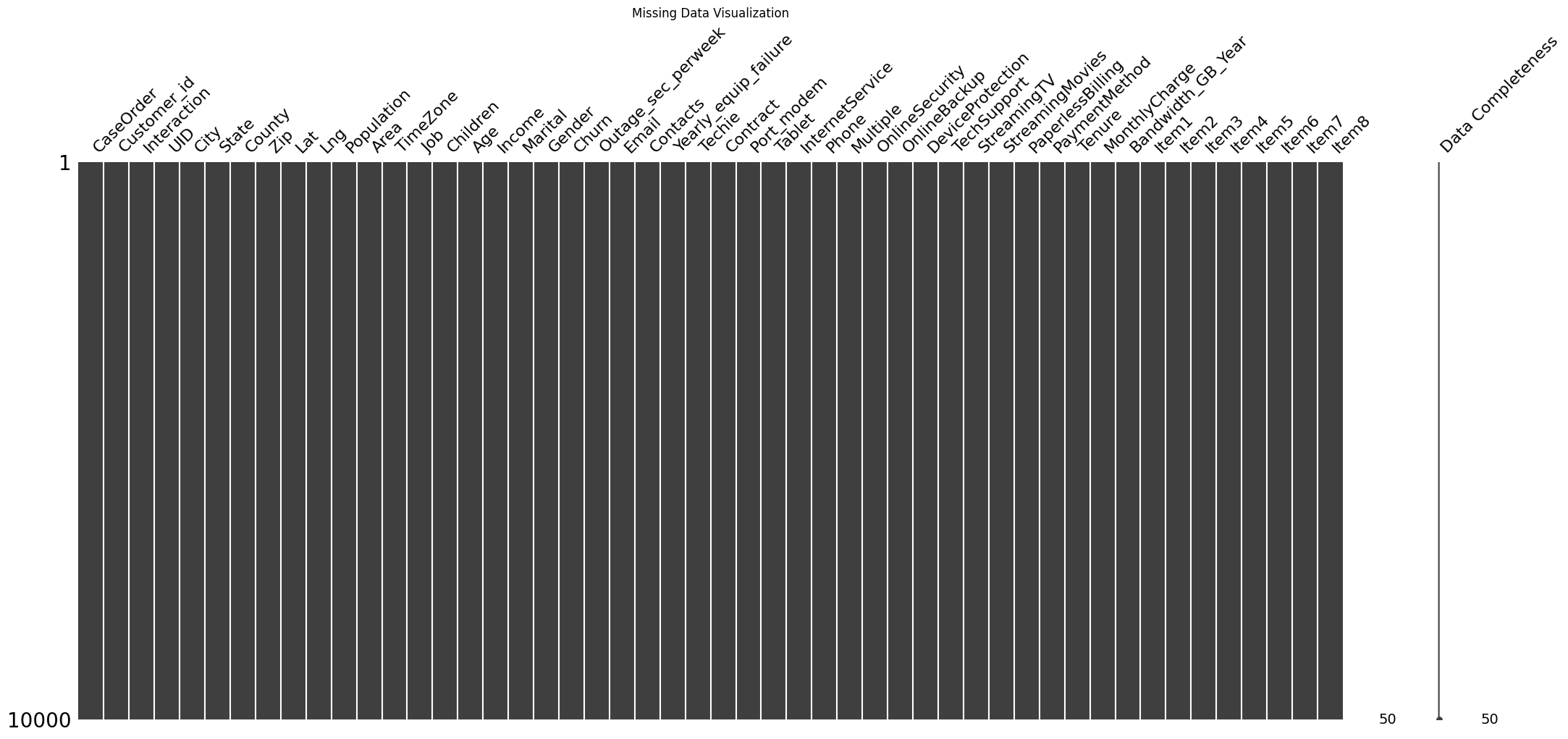
num\_duplicates = len(df) - len(df.duplicated())  
print(f"Duplicates: {num\_duplicates} of {len(df)} records.\n")

Duplicates: 0 of 10000 records.

#### Nulls

And now we use missingno to check for nulls.

import missingno as msno  
%matplotlib inline  
import matplotlib.pyplot as plt  
msno.matrix(df.sample(10000), labels=True)  
plt.title('Missing Data Visualization')  
plt.show()



As mentioned in the initial part of this document, the author had to import the CSV file using a special import which excluded "None" from appearing as a Null value. Initially, it appeared as if there were a significant number of null values for this attribute, but upon closer examplination, the realization was apparent that the value was, in fact, a third categorical option of "None."

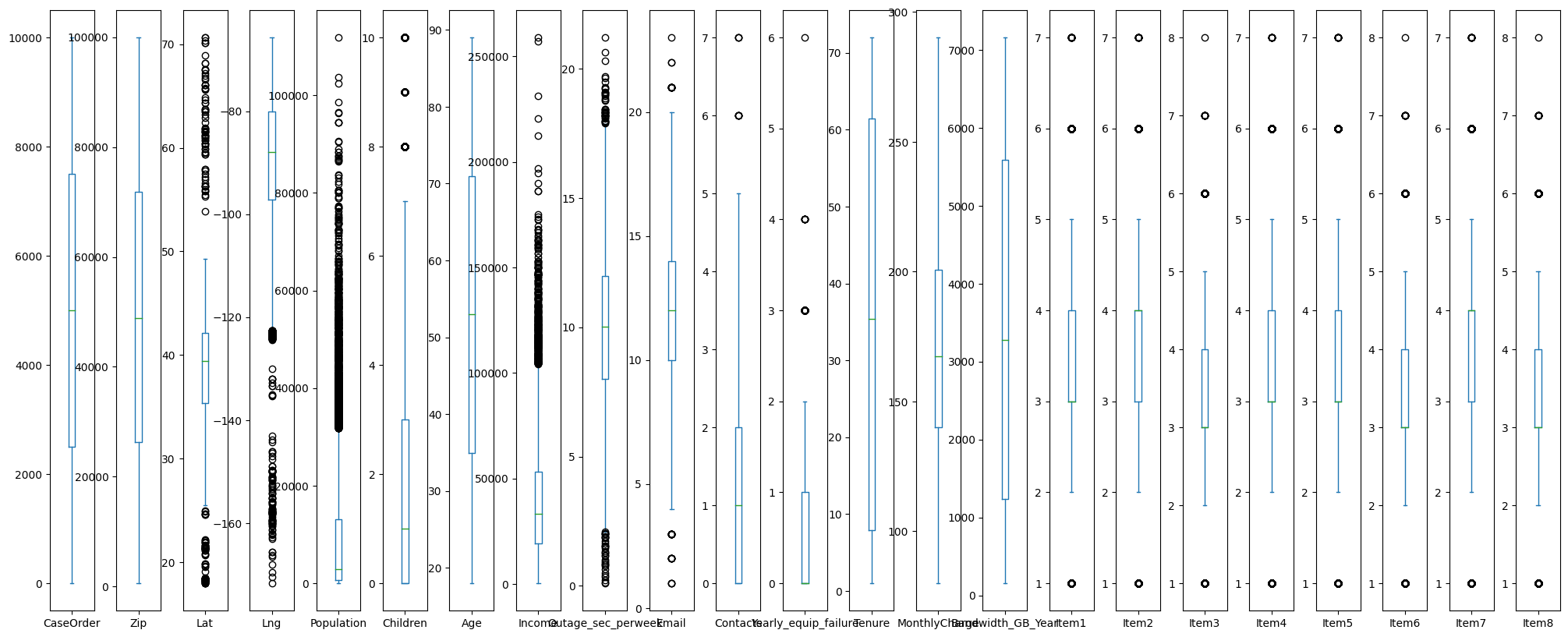
# Verify there are no null values  
df.isnull().sum()

CaseOrder 0  
Customer\_id 0  
Interaction 0  
UID 0  
City 0  
State 0  
County 0  
Zip 0  
Lat 0  
Lng 0  
Population 0  
Area 0  
TimeZone 0  
Job 0  
Children 0  
Age 0  
Income 0  
Marital 0  
Gender 0  
Churn 0  
Outage\_sec\_perweek 0  
Email 0  
Contacts 0  
Yearly\_equip\_failure 0  
Techie 0  
Contract 0  
Port\_modem 0  
Tablet 0  
InternetService 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  
Item1 0  
Item2 0  
Item3 0  
Item4 0  
Item5 0  
Item6 0  
Item7 0  
Item8 0  
dtype: int64

#### Outliers

The author now looked at the outliers by printing a boxplot of the appropriate variables.

# Plot boxplot for each attribute.   
# NOTE: this code was previously written and used by the author for an earlier class, D206  
import plotnine as p9  
  
df.plot(  
 kind='box',  
 subplots=True,  
 sharey=False,  
 figsize=(25, 10)  
)  
# increase spacing between subplots  
plt.subplots\_adjust(wspace=0.5)  
plt.show()



Regarding the outliers, the author found that although there were several attributes that may have outliers. That said, on a closer look, only "Children" and "Outages per week" had what were actual outliers. The former outlier included customers with 7-10 children. Outliers for the latter showed a cluster of outliers between 35 and 45 outages per week.

Therefore, the author decided to keep the outlier data since it was likely that either one or both may play a role in finding insights with our bandwidth analysis.

#### One Hot Encoding

For the attributes that were not a simple boolean Yes/No, the author performed a One Hot Encoding to each of the attributes, such as Gender, to quantify the values. These newly encoded columns were added to the dataframe and the originals dropped.

# One Hot Encoding  
one\_hot\_attr = ['Gender', 'Contract', 'InternetService', 'PaymentMethod']  
  
for attr in one\_hot\_attr:  
 if attr in df.columns:  
 # Added 'drop\_first=True' at end of parameters to drop one column to avoid multicoliniarity  
 # Also added astype to convert from boolean to numerical  
 y = pd.get\_dummies(df[attr], prefix=attr, drop\_first=True)  
 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   
 CaseOrder Customer\_id Interaction \  
0 1 K409198 aa90260b-4141-4a24-8e36-b04ce1f4f77b   
1 2 S120509 fb76459f-c047-4a9d-8af9-e0f7d4ac2524   
  
 UID City State County \  
0 e885b299883d4f9fb18e39c75155d990 Point Baker AK Prince of Wales-Hyder   
1 f2de8bef964785f41a2959829830fb8a West Branch MI Ogemaw   
  
 Zip Lat Lng ... Item8 Gender\_Male Gender\_Nonbinary \  
0 99927 56.25100 -133.37571 ... 4 True False   
1 48661 44.32893 -84.24080 ... 4 False False   
  
 Contract\_One year Contract\_Two Year InternetService\_Fiber Optic \  
0 True False True   
1 False False True   
  
 InternetService\_None PaymentMethod\_Credit Card (automatic) \  
0 False True   
1 False False   
  
 PaymentMethod\_Electronic Check PaymentMethod\_Mailed Check   
0 False False   
1 False False   
  
[2 rows x 55 columns]

CaseOrder int64  
Customer\_id object  
Interaction object  
UID object  
City object  
State object  
County object  
Zip int64  
Lat float64  
Lng float64  
Population int64  
Area object  
TimeZone object  
Job object  
Children int64  
Age int64  
Income float64  
Marital object  
Churn object  
Outage\_sec\_perweek float64  
Email int64  
Contacts int64  
Yearly\_equip\_failure int64  
Techie object  
Port\_modem object  
Tablet object  
Phone object  
Multiple object  
OnlineSecurity object  
OnlineBackup object  
DeviceProtection object  
TechSupport object  
StreamingTV object  
StreamingMovies object  
PaperlessBilling 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  
Gender\_Male bool  
Gender\_Nonbinary bool  
Contract\_One year bool  
Contract\_Two Year bool  
InternetService\_Fiber Optic bool  
InternetService\_None bool  
PaymentMethod\_Credit Card (automatic) bool  
PaymentMethod\_Electronic Check bool  
PaymentMethod\_Mailed Check bool  
dtype: object

#### YES / NO and Boolean attributes:

For each of the attributes with a Yes or No value, the author wrote the following python code to replace each string with the respective integer value with No being replaced with 0, and Yes being replaced with 1. Additionally, the attributes that are NOT pertaining to any correlation will be dropped. These include attributes such as Customer\_ID, Interaction, UID, etc.

#### Item1-8 Renaming

Items1-8 have also been renamed here for more user-friendly analysis. The replacement names are pulled from the data dictionary provided by the instructor.

attr\_to\_change = ['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'}  
attr\_to\_drop = ['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'Population', 'City', 'State', 'County',   
 'Zip', 'Lat', 'Lng', 'Marital', 'Area', 'TimeZone', 'Job']  
  
for a in attr\_to\_change:  
 print(f"Updating: {a}\t")  
 df[a] = df[a].replace({"Yes": 1, "No": 0, "True": 1, "False": 0})  
 df[a] = df[a].astype(np.int64) # Ensure data is saved as a 1 and 0 int, not a boolean "true"/"false"  
  
  
for a in attr\_to\_drop:  
 if a in df.columns: # if the attribute column exists error checking  
 df.drop(a, axis=1, inplace=True)  
 print(f"Dropped: {a}\t")  
  
df = df.rename(columns=attr\_to\_rename)  
  
df.head(5)

Updating: Techie   
Updating: Churn   
Updating: Port\_modem   
Updating: Phone   
Updating: Tablet   
Updating: Multiple   
Updating: OnlineSecurity   
Updating: DeviceProtection   
Updating: OnlineBackup   
Updating: StreamingTV   
Updating: TechSupport   
Updating: StreamingMovies   
Updating: PaperlessBilling   
Dropped: CaseOrder   
Dropped: Customer\_id   
Dropped: Interaction   
Dropped: UID   
Dropped: Population   
Dropped: City   
Dropped: State   
Dropped: County   
Dropped: Zip   
Dropped: Lat   
Dropped: Lng   
Dropped: Marital   
Dropped: Area   
Dropped: TimeZone   
Dropped: Job

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   
2 4 50 9609.57 0 10.752800 9 0   
3 1 48 18925.23 0 14.913540 15 2   
4 0 83 40074.19 1 8.147417 16 2   
  
 Yearly\_equip\_failure Techie Port\_modem ... Active\_Listening \  
0 1 0 1 ... 4   
1 1 1 0 ... 4   
2 1 1 1 ... 3   
3 0 1 0 ... 3   
4 1 0 1 ... 5   
  
 Gender\_Male Gender\_Nonbinary Contract\_One year Contract\_Two Year \  
0 True False True False   
1 False False False False   
2 False False False True   
3 True False False True   
4 True False False False   
  
 InternetService\_Fiber Optic InternetService\_None \  
0 True False   
1 True False   
2 False False   
3 False False   
4 True False   
  
 PaymentMethod\_Credit Card (automatic) PaymentMethod\_Electronic Check \  
0 True False   
1 False False   
2 True False   
3 False False   
4 False False   
  
 PaymentMethod\_Mailed Check   
0 False   
1 False   
2 False   
3 True   
4 True   
  
[5 rows x 40 columns]

#### Print sample row to verify values for each column have been updated

column\_names = df.columns  
dtypes = df.dtypes  
first\_row = df.iloc[1].values  
  
column\_names = df.columns  
dtypes = df.dtypes  
first\_row = df.iloc[1].values  
print(f"\033[1mName\t\tType\tValue\033[0m")  
for column\_name, dtype, value in zip(column\_names, dtypes, first\_row):  
 print(f"{column\_name:<15}\t{dtype}\t{value:<18}\n")

Name Type Value  
Children int64 1   
  
Age int64 27   
  
Income float64 21704.77   
  
Churn int64 1   
  
Outage\_sec\_perweek float64 11.69907956   
  
Email int64 12   
  
Contacts int64 0   
  
Yearly\_equip\_failure int64 1   
  
Techie int64 1   
  
Port\_modem int64 0   
  
Tablet int64 1   
  
Phone int64 1   
  
Multiple int64 1   
  
OnlineSecurity int64 1   
  
OnlineBackup int64 0   
  
DeviceProtection int64 0   
  
TechSupport int64 0   
  
StreamingTV int64 1   
  
StreamingMovies int64 1   
  
PaperlessBilling int64 1   
  
Tenure float64 1.156680997   
  
MonthlyCharge float64 242.632554   
  
Bandwidth\_GB\_Year float64 800.9827661   
  
Timely\_Response int64 3   
  
Timely\_Fixes int64 4   
  
Timely\_Replacements int64 3   
  
Reliability int64 3   
  
Options int64 4   
  
Respectful int64 3   
  
Courteous int64 4   
  
Active\_Listening int64 4   
  
Gender\_Male bool 0   
  
Gender\_Nonbinary bool 0   
  
Contract\_One year bool 0   
  
Contract\_Two Year bool 0   
  
InternetService\_Fiber Optic bool 1   
  
InternetService\_None bool 0   
  
PaymentMethod\_Credit Card (automatic) bool 0   
  
PaymentMethod\_Electronic Check bool 0   
  
PaymentMethod\_Mailed Check bool 0

### C2: Describe dependent and independent variables using summary statistics

The **dependent variable** for our question (see section A1) will be "Bandwidth\_GB\_Year", a continuous variable that captures the total bandwidth used by the customer for the year (in gigabytes). By having this as our dependent variable, the author will analyze the other attributes (or independent variables) in order to find which have a correlation affecting the dependent variable.

Our **independent variables** are the remainder of the attributes in our dataframe (after cleaning). Specifically, we have a number of categorical, boolean, and continuous variables, while at the same time, dropping the attributes that are administrative and would not have any impact on bandwidth. For example, "CaseOrder", "Customer\_id", etc. are removed as attributes.

Once those attributes are removed, the following attributes are used as independent variables: 'Children': number of children of the customer, 'Age': age of the customer, 'Churn': customers who left in the last month, 'Income': income of the customer (annual), 'Outage\_sec\_perweek': average seconds per week the system was out, 'Email': how many emails sent to the customer (in last year), 'Contacts': how many times the customer was contacted (in last year), 'Yearly\_equip\_failure': number of customer equipment fails in the past year, 'Techie': does the customer feel techie?, 'Port\_modem': do they have a port modem?, 'Tablet': do they own an iPad/surface?, 'Phone': does the customer have phone service, 'Multiple': multple phone lines?, 'OnlineSecurity' does the customer have online security as an add-on service, 'DeviceProtection': did they sign up for device protection service?, 'OnlineBackup': did the customer have online backup 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, '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, 'Contract': the customer's contract term (monthly, annual, or biannual)

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  
ind\_vars = ['Children', 'Age', 'Churn', 'Income', 'Outage\_sec\_perweek', 'Email', 'Contacts',   
 'Yearly\_equip\_failure', 'Techie', 'Port\_modem', 'Tablet', 'Phone', 'Multiple',   
 'OnlineSecurity', 'DeviceProtection', 'OnlineBackup', 'TechSupport', 'StreamingTV',  
 'StreamingMovies', 'PaperlessBilling', 'Tenure', 'MonthlyCharge',  
 'Timely\_Response', 'Timely\_Fixes', 'Timely\_Replacements', 'Reliability', 'Options',  
 'Respectful', 'Courteous', 'Active\_Listening']   
bool\_vars = ['Gender\_Male', 'Gender\_Nonbinary', 'Contract\_One year','Contract\_Two Year', 'InternetService\_Fiber Optic',  
 'InternetService\_None','PaymentMethod\_Credit Card (automatic)','PaymentMethod\_Electronic Check',  
 'PaymentMethod\_Mailed Check']  
vars = ind\_vars + bool\_vars  
import plotnine as p9  
  
# Note: each of the bool\_vars is missing one column, which is dropped on purpose from earlier  
# during the one hot encoding to prevent multicollinearity  
# Convert boolean values to integers for model fitting later  
for var in vars:  
 if var in bool\_vars:  
 print(var + " Is boolean")  
 df[var] = df[var].astype('int64')  
  
print(df[vars].describe())

Gender\_Male Is boolean  
Gender\_Nonbinary Is boolean  
Contract\_One year Is boolean  
Contract\_Two Year Is boolean  
InternetService\_Fiber Optic Is boolean  
InternetService\_None Is boolean  
PaymentMethod\_Credit Card (automatic) Is boolean  
PaymentMethod\_Electronic Check Is boolean  
PaymentMethod\_Mailed Check Is boolean  
 Children Age Churn Income \  
count 10000.0000 10000.000000 10000.000000 10000.000000   
mean 2.0877 53.078400 0.265000 39806.926771   
std 2.1472 20.698882 0.441355 28199.916702   
min 0.0000 18.000000 0.000000 348.670000   
25% 0.0000 35.000000 0.000000 19224.717500   
50% 1.0000 53.000000 0.000000 33170.605000   
75% 3.0000 71.000000 1.000000 53246.170000   
max 10.0000 89.000000 1.000000 258900.700000   
  
 Outage\_sec\_perweek Email Contacts Yearly\_equip\_failure \  
count 10000.000000 10000.000000 10000.000000 10000.000000   
mean 10.001848 12.016000 0.994200 0.398000   
std 2.976019 3.025898 0.988466 0.635953   
min 0.099747 1.000000 0.000000 0.000000   
25% 8.018214 10.000000 0.000000 0.000000   
50% 10.018560 12.000000 1.000000 0.000000   
75% 11.969485 14.000000 2.000000 1.000000   
max 21.207230 23.000000 7.000000 6.000000   
  
 Techie Port\_modem ... Active\_Listening Gender\_Male \  
count 10000.000000 10000.000000 ... 10000.000000 10000.000000   
mean 0.167900 0.483400 ... 3.495600 0.474400   
std 0.373796 0.499749 ... 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% 0.000000 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 and bivariate visualizations (dependent and independent variables)

The univariate and bivariate visualizations were created by breaking down each into a separate list and then graphic with histograms or scatterplots.

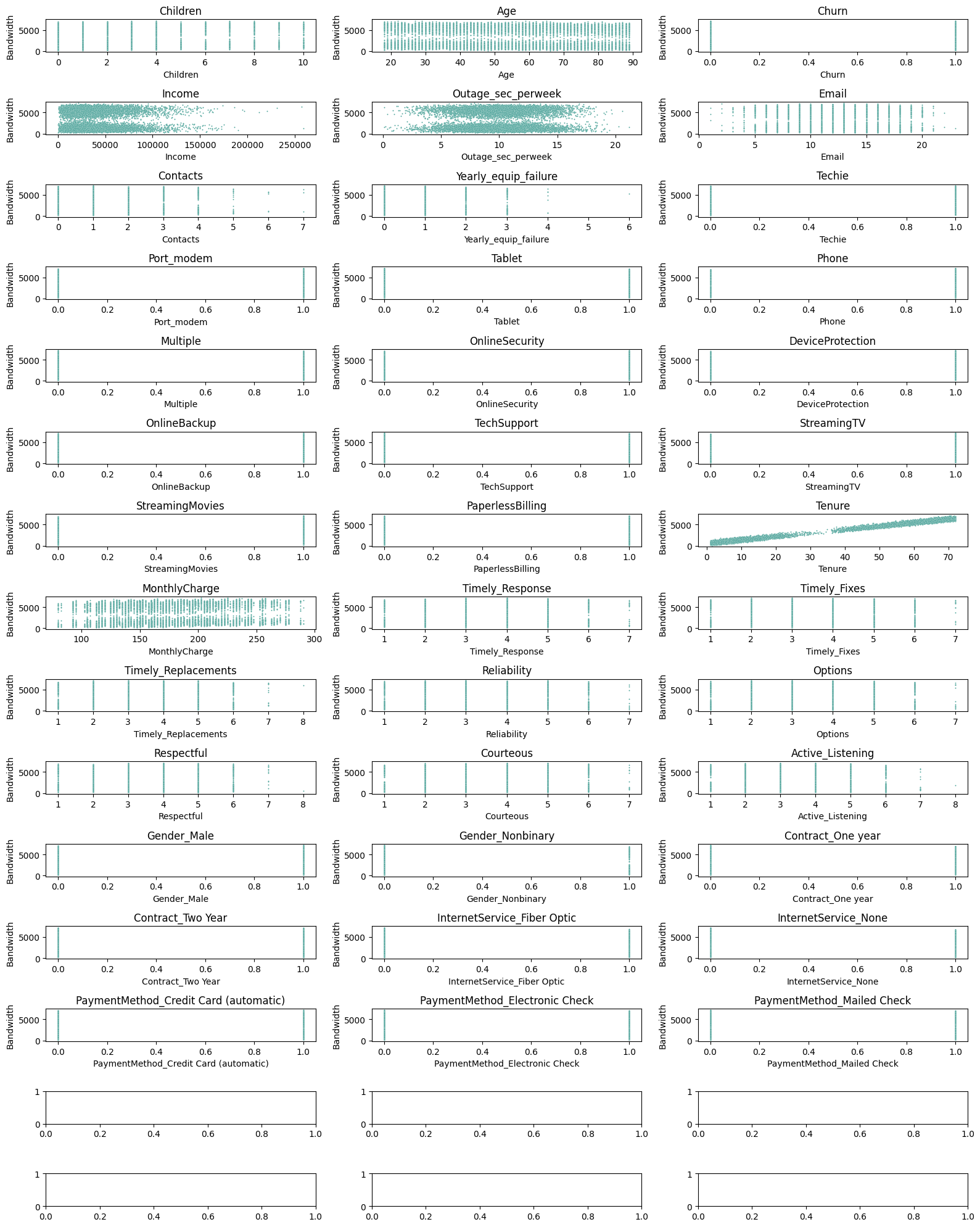
import plotnine as p9  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# UNIVARIATE Histograms Plotting  
print("Univariate Histograms\n")  
  
variables = vars  
fig, axs = plt.subplots(nrows=13, ncols=3, figsize=(16,20))  
for ax, var in zip(axs.flatten(), variables):  
 sns.histplot(data=df, 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()

Univariate Histograms



import plotnine as p9  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# BIVARIATE Histograms  
print("Bivariate Visualizations\n")  
variables = vars  
fig, axs = plt.subplots(nrows=15, ncols=3, figsize=(16,20))  
for ax, var in zip(axs.flatten(), variables):  
 sns.scatterplot(data=df, x=var, y='Bandwidth\_GB\_Year', s=2, color='#429e94', ax=ax)  
 ax.set\_title(var)  
 ax.set\_xlabel(var)  
 ax.set\_ylabel('Bandwidth')  
plt.tight\_layout()  
plt.show()

Bivariate Visualizations



### C4. Data Transformation Goals and Transformation Steps Taken

The author focused his attention around the bandwidth attribute (Bandwidth\_GB\_Year) as the dependent variable (it is the continuous target variable) to explore and analyze which factors (from the independent variables) showed a correlation with the bandwidth.

**Data was transformed using the following steps.** First, it needed to be cleaned in preparation for the training and testing of the dataset model. To clean the data, null values were removed, "yes/no" and "true"/"false" values were converted to 1/0, respectively. This was done by replacing null values with appropriate default values for the former case. For the latter cases, the boolean strings were converted to numerical booleans using a replace on each column with said data values.

Additionally, categorical data was encoded via one-hot encoding to multiple new attributes that were boolean values. This was accomplished using the pandas pd.get\_dummies(df[attribute], prefix=attribute) code from Section C1 above.

**This goal of data transformation** left us with a complete set of data that was prepared for proper analysis on the dependent variable. The author ensured that all required data for the analysis was in numerical form (for later regression analysis). Specifically, this included converting categorical data into numerical form (ie. one-hot encoding mentioned above). The author searched for missing values and depending on the nature of each data attribute, resolved these issues. Finally, the author checked for outliers (though the author determined that the nature of the outliers did not require they be removed due to the data insights they might hold).

All annotated code is illustrated in section C1, labeled according to each transformation. Due to this being a jupter notebook, the actual code will be kept in section C1.

### C5. Export Prepared Data Set as CSV File

Below is the code written by the author to export the prepared dataframe as a CSV file.

# Export Prepared Data to CSV  
file\_name = '\_data/cleaned\_and\_prepared\_dataframe.csv'  
df.to\_csv(file\_name, index=False)  
print(f"Exported to {file\_name}")

Exported to \_data/cleaned\_and\_prepared\_dataframe.csv

## Part IV: Model Comparison / Analysis

### D1: Initial Multiple Linear Regression Model (Using All Independent Variables from C2)

At this point, the author will create an initial multiple linear regression model that uses all the above independent variables except for the columns that were dropped to prevent multicollinearity as part of the one hot encoding. This was accomplished in section C1 by using the **drop\_first=True** paramenter to *get\_dummies()* and basically, met the condition of setting our columns to k-1.

The author chose to use Scikit Learn due to its popularity and the fact that the author has used this framework previously. The LinearRegression model is imported, and the attributes listed immediately above are loaded as the X (independent) axis and our bandwidth (Bandwith\_GB\_Year) is loaded as our dependent variable.

The Ordinary Least Squares was performed as the multiple linear regression. This model was chosen due to its ability to estimate the relationship between multiple independent variables (by minimizing the sum of squares). Since scikit-learn does not directly provide the required statistics for this, the author used Statsmodels for the statistical evaluation.

df.dtypes

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

import statsmodels.api as sm  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
  
# independent variables from above and the target variable (bandwidth used by the customer)  
X = df[ind\_vars]  
y = df['Bandwidth\_GB\_Year']  
  
# Linear Regression model  
model = sm.OLS(y,X)  
results = model.fit()  
print(results.summary())

OLS Regression Results   
=======================================================================================  
Dep. Variable: Bandwidth\_GB\_Year R-squared (uncentered): 0.998  
Model: OLS Adj. R-squared (uncentered): 0.998  
Method: Least Squares F-statistic: 1.359e+05  
Date: Fri, 09 Jun 2023 Prob (F-statistic): 0.00  
Time: 00:45:43 Log-Likelihood: -67136.  
No. Observations: 10000 AIC: 1.343e+05  
Df Residuals: 9970 BIC: 1.345e+05  
Df Model: 30   
Covariance Type: nonrobust   
========================================================================================  
 coef std err t P>|t| [0.025 0.975]  
----------------------------------------------------------------------------------------  
Children 32.2981 0.927 34.833 0.000 30.481 34.116  
Age -2.9576 0.095 -31.179 0.000 -3.144 -2.772  
Churn 91.8733 5.837 15.741 0.000 80.432 103.314  
Income 0.0003 7.04e-05 3.619 0.000 0.000 0.000  
Outage\_sec\_perweek 2.9839 0.646 4.618 0.000 1.717 4.251  
Email 4.1234 0.628 6.567 0.000 2.893 5.354  
Contacts 4.7884 2.017 2.374 0.018 0.834 8.742  
Yearly\_equip\_failure 5.6342 3.136 1.796 0.072 -0.514 11.782  
Techie 0.5970 5.359 0.111 0.911 -9.908 11.102  
Port\_modem 2.5816 3.988 0.647 0.517 -5.236 10.399  
Tablet 4.1497 4.362 0.951 0.341 -4.401 12.700  
Phone 32.1947 6.659 4.835 0.000 19.142 45.247  
Multiple 122.9794 5.601 21.957 0.000 112.000 133.958  
OnlineSecurity 89.0391 4.189 21.257 0.000 80.828 97.250  
DeviceProtection 104.3405 4.312 24.196 0.000 95.887 112.794  
OnlineBackup 128.2174 4.832 26.537 0.000 118.746 137.689  
TechSupport 33.4189 4.370 7.647 0.000 24.852 41.985  
StreamingTV 281.7902 6.405 43.998 0.000 269.236 294.345  
StreamingMovies 272.8372 7.355 37.096 0.000 258.420 287.254  
PaperlessBilling 1.2286 4.045 0.304 0.761 -6.700 9.158  
Tenure 82.8945 0.089 935.243 0.000 82.721 83.068  
MonthlyCharge -1.5214 0.113 -13.405 0.000 -1.744 -1.299  
Timely\_Response -6.9245 2.862 -2.420 0.016 -12.534 -1.315  
Timely\_Fixes 5.7051 2.682 2.127 0.033 0.449 10.962  
Timely\_Replacements 3.7956 2.446 1.552 0.121 -1.000 8.591  
Reliability 16.3423 2.037 8.022 0.000 12.349 20.336  
Options 25.7709 1.979 13.023 0.000 21.892 29.650  
Respectful 7.3437 2.328 3.154 0.002 2.780 11.908  
Courteous 8.2076 2.194 3.741 0.000 3.907 12.508  
Active\_Listening 10.9887 2.089 5.259 0.000 6.893 15.084  
==============================================================================  
Omnibus: 9779.697 Durbin-Watson: 1.975  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 1048.003  
Skew: 0.514 Prob(JB): 2.69e-228  
Kurtosis: 1.792 Cond. No. 2.54e+05  
==============================================================================  
  
Notes:  
[1] R² is computed without centering (uncentered) since the model does not contain a constant.  
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[3] The condition number is large, 2.54e+05. This might indicate that there are  
strong multicollinearity or other numerical problems.

### D2: Statistically Based Model Reduction with Justification

The author began by creating a duplicated dataframe (df\_reduced) to use as a dataframe to iteratively filter out various attributes based on statistically based justifications.

# Creation of the reduced df  
df\_reduced = pd.DataFrame()  
reduced\_vars = df.columns.tolist()

The first task was to check for Multicollinearity. VIF quantifies how much the variance of the coefficient is impacted from multicollinearity. When several attributes in the model are correlated with each other, this can often introduce inflated errors, and more unreliable results. This dataframe reduction was accomplished by calculating and using the VIF and dropping all attributes with a VIF value higher than 10. This is a standard value when factoring out multicollinearity in a dataframe.

To interpret the VIF, a value of 1 means there is no correlation (and no multicollinearity). In general, a VIF value larger than 5 or 10 is an indicator of multicollinearity, and should not be included. To find the VIF, the author used statsmodels and variance\_inflation\_factor. The value of 5 for the VIF cutoff was recommended by Jim Frost, of Statistics by Jim (source 3).

#### Multicollinearity Detection Using VIF

# MULTICOLLINEARITY DETECTION VIA VIF  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
MAX\_VIF = 5  
  
X = df[variables].copy()  
vif\_data = pd.DataFrame()  
  
for i in range(len(X.columns)):  
 vif = variance\_inflation\_factor(X.values, i)  
 # Conditionally add to reduced DataFrame only if VIF is less than 10   
 if vif < 10:  
 df\_reduced.loc[:, X.columns[i]] = X.loc[:, X.columns[i]]  
 print(f"VIF = {vif:4f} for {X.columns[i]}\t(Keeping)")   
 else:  
 print(f"VIF = {vif:4f} for {X.columns[i]}\t(Dropping: Over {MAX\_VIF})")

VIF = 1.943309 for Children (Keeping)  
VIF = 7.419598 for Age (Keeping)  
VIF = 2.671424 for Churn (Keeping)  
VIF = 2.978833 for Income (Keeping)  
VIF = 11.695667 for Outage\_sec\_perweek (Dropping: Over 5)  
VIF = 15.673962 for Email (Dropping: Over 5)  
VIF = 2.010821 for Contacts (Keeping)  
VIF = 1.393333 for Yearly\_equip\_failure (Keeping)  
VIF = 1.212860 for Techie (Keeping)  
VIF = 1.932576 for Port\_modem (Keeping)  
VIF = 1.431486 for Tablet (Keeping)  
VIF = 10.278844 for Phone (Dropping: Over 5)  
VIF = 6.009486 for Multiple (Keeping)  
VIF = 1.596081 for OnlineSecurity (Keeping)  
VIF = 2.414856 for DeviceProtection (Keeping)  
VIF = 3.806695 for OnlineBackup (Keeping)  
VIF = 2.132447 for TechSupport (Keeping)  
VIF = 9.275045 for StreamingTV (Keeping)  
VIF = 12.891114 for StreamingMovies (Dropping: Over 5)  
VIF = 2.424736 for PaperlessBilling (Keeping)  
VIF = 3.930832 for Tenure (Keeping)  
VIF = 255.364136 for MonthlyCharge (Dropping: Over 5)  
VIF = 27.315333 for Timely\_Response (Dropping: Over 5)  
VIF = 24.151197 for Timely\_Fixes (Dropping: Over 5)  
VIF = 19.975281 for Timely\_Replacements (Dropping: Over 5)  
VIF = 14.537063 for Reliability (Dropping: Over 5)  
VIF = 14.312457 for Options (Dropping: Over 5)  
VIF = 18.206547 for Respectful (Dropping: Over 5)  
VIF = 16.315169 for Courteous (Dropping: Over 5)  
VIF = 14.654509 for Active\_Listening (Dropping: Over 5)  
VIF = 1.942542 for Gender\_Male (Keeping)  
VIF = 1.048561 for Gender\_Nonbinary (Keeping)  
VIF = 1.505946 for Contract\_One year (Keeping)  
VIF = 1.591510 for Contract\_Two Year (Keeping)  
VIF = 4.031225 for InternetService\_Fiber Optic (Keeping)  
VIF = 1.809503 for InternetService\_None (Keeping)  
VIF = 1.925031 for PaymentMethod\_Credit Card (automatic) (Keeping)  
VIF = 2.510326 for PaymentMethod\_Electronic Check (Keeping)  
VIF = 2.015199 for PaymentMethod\_Mailed Check (Keeping)

# Updated Reduced Dataframe Attributes Remaining  
df\_reduced.dtypes

Children int64  
Age int64  
Churn int64  
Income float64  
Contacts int64  
Yearly\_equip\_failure int64  
Techie int64  
Port\_modem int64  
Tablet int64  
Multiple int64  
OnlineSecurity int64  
DeviceProtection int64  
OnlineBackup int64  
TechSupport int64  
StreamingTV int64  
PaperlessBilling int64  
Tenure float64  
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

#### Recursive Feature Elimination

The author next ran a recursive feature elimination on the remaining attributes to iteratively remove the worst performing attributes for each iteration. This is a standard part of the SciKit-Learn framework, hence the import of RFE. The author set upon 15 values as our final attribute count, and if needed, run it again with fewer. It was determined that 15 was a good value that prevented overfitting while at the same time didn't lose significant accuracy in the model.

## RECURSIVE FEATURE ELIMINATION  
  
from sklearn.feature\_selection import RFE  
from sklearn.linear\_model import LinearRegression  
MAX\_ATTRIBUTES = 15  
  
X = df\_reduced  
y = df['Bandwidth\_GB\_Year']  
  
model = LinearRegression()  
rfe = RFE(estimator=model, n\_features\_to\_select=MAX\_ATTRIBUTES)  
rfe = rfe.fit(df\_reduced, y)  
  
# get the names of the remaining columns  
iteration2\_features = df\_reduced.columns[rfe.support\_]  
  
# Save reduced dataframe print the features  
df\_reduced = df\_reduced[iteration2\_features]  
print('Selected features:', iteration2\_features)  
df\_reduced.dtypes

Selected features: Index(['Children', 'Churn', 'Techie', 'Multiple', 'OnlineSecurity',  
 'DeviceProtection', 'OnlineBackup', 'StreamingTV', 'Tenure',  
 'Gender\_Male', 'Gender\_Nonbinary', 'Contract\_One year',  
 'Contract\_Two Year', 'InternetService\_Fiber Optic',  
 'InternetService\_None'],  
 dtype='object')

Children int64  
Churn int64  
Techie int64  
Multiple int64  
OnlineSecurity int64  
DeviceProtection int64  
OnlineBackup int64  
StreamingTV int64  
Tenure float64  
Gender\_Male int64  
Gender\_Nonbinary int64  
Contract\_One year int64  
Contract\_Two Year int64  
InternetService\_Fiber Optic int64  
InternetService\_None int64  
dtype: object

### D3: Reduced Linear Regression Model

Based on the analysis in D2, the author has reduced the model, taking into consideration multicollinearity and RFE to find a reduced model without lowering the accuracy.

Below is the creation of the reduced linear model with these attributes.

# Reduced linear regression model (from previous code section)  
df\_reduced = sm.add\_constant(df\_reduced)  
model = sm.OLS(df['Bandwidth\_GB\_Year'], df\_reduced)  
results = model.fit()  
print(results.summary())

OLS Regression Results   
==============================================================================  
Dep. Variable: Bandwidth\_GB\_Year R-squared: 0.997  
Model: OLS Adj. R-squared: 0.997  
Method: Least Squares F-statistic: 2.187e+05  
Date: Fri, 09 Jun 2023 Prob (F-statistic): 0.00  
Time: 00:58:26 Log-Likelihood: -62095.  
No. Observations: 10000 AIC: 1.242e+05  
Df Residuals: 9984 BIC: 1.243e+05  
Df Model: 15   
Covariance Type: nonrobust   
===============================================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------------------------  
const 425.5599 4.442 95.801 0.000 416.852 434.267  
Children 31.7781 0.561 56.599 0.000 30.677 32.879  
Churn 119.1347 3.515 33.891 0.000 112.244 126.025  
Techie -6.9144 3.234 -2.138 0.033 -13.255 -0.574  
Multiple 59.8528 2.453 24.402 0.000 55.045 64.661  
OnlineSecurity 80.7132 2.515 32.087 0.000 75.782 85.644  
DeviceProtection 82.6484 2.434 33.959 0.000 77.878 87.419  
OnlineBackup 87.5414 2.431 36.015 0.000 82.777 92.306  
StreamingTV 201.0221 2.520 79.786 0.000 196.083 205.961  
Tenure 82.8380 0.054 1543.462 0.000 82.733 82.943  
Gender\_Male 63.9098 2.441 26.181 0.000 59.125 68.695  
Gender\_Nonbinary -17.1002 8.112 -2.108 0.035 -33.002 -1.198  
Contract\_One year 28.8529 3.205 9.003 0.000 22.571 35.135  
Contract\_Two Year 28.2877 3.055 9.259 0.000 22.299 34.276  
InternetService\_Fiber Optic -403.9119 2.755 -146.632 0.000 -409.312 -398.512  
InternetService\_None -404.5391 3.336 -121.273 0.000 -411.078 -398.000  
==============================================================================  
Omnibus: 472.356 Durbin-Watson: 1.972  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 192.994  
Skew: 0.038 Prob(JB): 1.24e-42  
Kurtosis: 2.324 Cond. No. 294.  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

As we can see in the above OLS Regression Results, the R-squared is 0.998. So while we have eliminated almost all of the previous attributes, we have retained the correlation almost exactly, showing the strong correlation of the bandwidth used.

model = sm.OLS(df['Bandwidth\_GB\_Year'], df\_reduced)  
  
results = model.fit()  
print(results.summary())

OLS Regression Results   
==============================================================================  
Dep. Variable: Bandwidth\_GB\_Year R-squared: 0.997  
Model: OLS Adj. R-squared: 0.997  
Method: Least Squares F-statistic: 2.187e+05  
Date: Fri, 09 Jun 2023 Prob (F-statistic): 0.00  
Time: 00:58:37 Log-Likelihood: -62095.  
No. Observations: 10000 AIC: 1.242e+05  
Df Residuals: 9984 BIC: 1.243e+05  
Df Model: 15   
Covariance Type: nonrobust   
===============================================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------------------------  
const 425.5599 4.442 95.801 0.000 416.852 434.267  
Children 31.7781 0.561 56.599 0.000 30.677 32.879  
Churn 119.1347 3.515 33.891 0.000 112.244 126.025  
Techie -6.9144 3.234 -2.138 0.033 -13.255 -0.574  
Multiple 59.8528 2.453 24.402 0.000 55.045 64.661  
OnlineSecurity 80.7132 2.515 32.087 0.000 75.782 85.644  
DeviceProtection 82.6484 2.434 33.959 0.000 77.878 87.419  
OnlineBackup 87.5414 2.431 36.015 0.000 82.777 92.306  
StreamingTV 201.0221 2.520 79.786 0.000 196.083 205.961  
Tenure 82.8380 0.054 1543.462 0.000 82.733 82.943  
Gender\_Male 63.9098 2.441 26.181 0.000 59.125 68.695  
Gender\_Nonbinary -17.1002 8.112 -2.108 0.035 -33.002 -1.198  
Contract\_One year 28.8529 3.205 9.003 0.000 22.571 35.135  
Contract\_Two Year 28.2877 3.055 9.259 0.000 22.299 34.276  
InternetService\_Fiber Optic -403.9119 2.755 -146.632 0.000 -409.312 -398.512  
InternetService\_None -404.5391 3.336 -121.273 0.000 -411.078 -398.000  
==============================================================================  
Omnibus: 472.356 Durbin-Watson: 1.972  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 192.994  
Skew: 0.038 Prob(JB): 1.24e-42  
Kurtosis: 2.324 Cond. No. 294.  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

### E1: Data Analysis

The author's initial multiple linear regression model, given above, included a large number of predictors from the initial dataset. The author began with this model with the goal of understanding the inter-relationship between the various attributes which had a potential to drive the dependent attribute, Bandwidth\_GB\_Year. By using this large a number of attributes, though, increases the likelihood of overfitting the data and not being able to be a proper predictor with future data, thus making it a poor model.

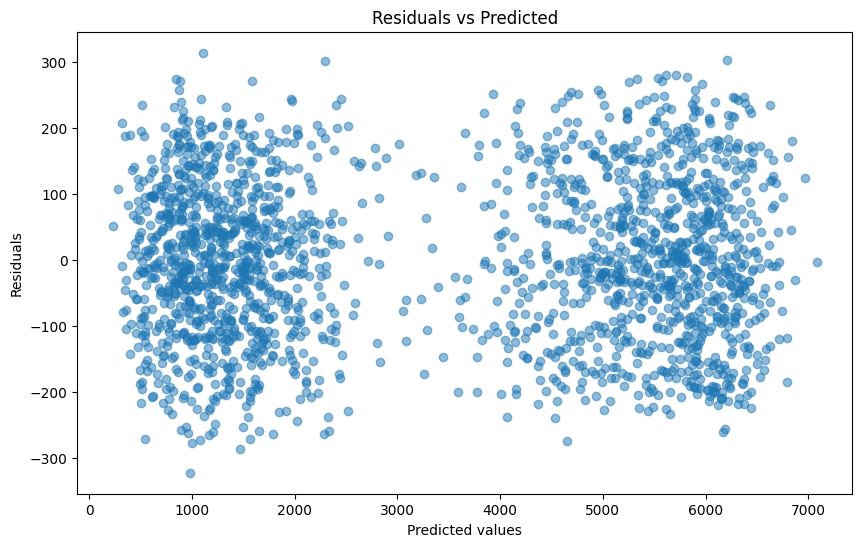
To remedy this, the author reduced negatively correlated attributes and only included positively correlated ones. This reduced dataset would reduce the complexity of our model and, thus, reduce the chance of overfitting. The author accomplished this by comparing coefficients from the regression model, p-values, and correlation values. Promising candidates for attribute removal were low coefficients, high p-values, and low/negative correlation values.

This reduced model became simpler and more likely to fit unknown data. As a final check, the R-squared value of the original- and reduced-model were compared and found to be statistically similar while reducing the model size. In both linear regression models, the R-squared models were 0.999 or higher (illustrating a higher probability of more variability in the data).

### E2: Output of all calculations

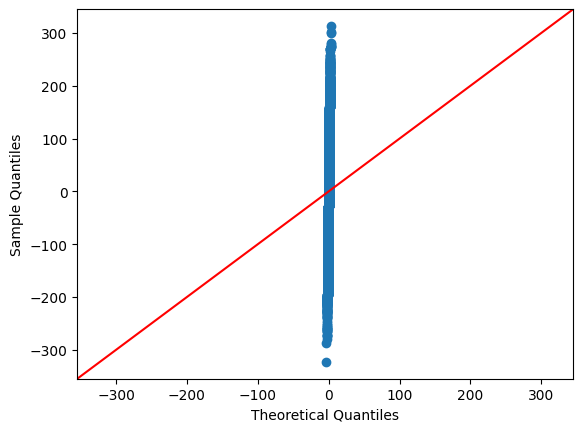
Below is the data summary from both the full dataset model and the reduced dataset model. Below is a residual plot of the reduced linear regression model followed by the residual standard error:

from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
  
X = df\_reduced  
y = df['Bandwidth\_GB\_Year']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=4)  
  
# Fit the model  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Predicting values  
y\_pred = model.predict(X\_test)  
  
# Compute residuals  
residuals = y\_test - y\_pred  
  
# Plotting the residuals  
plt.figure(figsize=(10,6))  
plt.scatter(y\_pred, residuals, alpha=0.5)  
plt.title('Residuals vs Predicted')  
plt.xlabel('Predicted values')  
plt.ylabel('Residuals')  
plt.show()



The scatter plot calculated above has an even and random distribution of points that are centered at the 0 residuals mark on the Y-axis. This is an indicator of a linear relationship connecting the independent and the dependent variables.

import statsmodels.api as sm  
import matplotlib.pyplot as plt  
  
sm.qqplot(residuals, line='45')  
plt.show()



from sklearn.metrics import mean\_squared\_error  
import numpy as np  
  
mse = mean\_squared\_error(y\_test, y\_pred)  
rse = np.sqrt(mse)  
error\_percentage = rse / df['Bandwidth\_GB\_Year'].mean() \* 100  
  
print(f"Residual Standard Error: {rse}")  
print(f"Percent Error: {error\_percentage:.2f}%")

Residual Standard Error: 121.30425449223196  
Percent Error: 3.58%

### E3: Working code for linear regression models using a Python

The author has included the error-free, working code in the entire section E1 and E2 as seen above.

## Part V: Data Summary Along With Implications

### F1: Findings Summary of Data Analysis

Upon analysis of the data, the author found that Tenure is the most significantly correlated attribute with the amount of bandwidth used.

**A regression equation for the reduced model** would follow the format: Y = b0 + b1X1 + b2X2 + ... + bn\*Xn + e. For the specific reduced model calculated by the author, the values to substitute in would be b1 (StreamingTv) = 201. While there were three other attributes, the author is not including them as part of the equation due to their respective insignificance in the calculation. Our constant from our OLS analysis (intercept) was 425. This would result in our Linear Regression equation as follows:

Bandwidth\_GB\_Year = 425 + 201\*Tenure

As the author **interprets the coefficients of the reduced model**, is that the StreamingTV coefficient of 201 shows there is a large impact on bandwidth from streaming television services (which does make sense). Additionally, there is a correlation between Tenure and bandwidth as well, with Tenure having coefficient of 82, but a much lower standard error.

The **statistical and practical significance of the reduced model** was shown to be a large reduction of the model size (and reduction in likelihood of overfitting) while the practical element shows that the linear regression equation could be drastically reduced to few variables.

Still, **the limitations of the data analysis** is seen in that the relationship must be a linear one. Thus, a more complicated relationship may not be able to be captured and modeled as accurately.

### F2: Recommended Course of Action Based on Analysis

The author recommends that the stakeholders work with marketers in creating a program among customers focused on bandwidth because of the positive correlation with tenure. An initial course of action could be to add a tiered program for long-term customers. For example, offering a higher-priced plan that offers higher internet speeds or higher bandwidth.

## Part VI: Demonstration

### G: Panopto Presentation

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=c4c0fd95-744f-46e6-b127-b01c004c0ad3>

### H: Web Sources

1. "Getting Started with D208" Part I, Dr. Keiona Middleton. Webinar video.
2. "Pandas get\_dummies (One-Hot Encoding) Explained". <https://datagy.io/pandas-get-dummies/>
3. "Variance Inflation Factors (VIFs)". Frost, Jim. <https://statisticsbyjim.com/regression/variance-inflation-factors/>

### I: In-text citations and references

1. Verbiest, Nele. "Introduction to Predictive Analytics in Python". Datacamp Interactive Course (Course textbook)