D206 Data Cleaning, Performance Assessment

Alexa R. Fisher

Western Governors University

Degree: M.S. Data Analytics

Table of Contents

Part I. Research Question

A. Research Question

The research question selected for this thesis is, "Does the amount of telecommunication services affect the yearly service failures per household size?" This is an analysis of the customer household size in relation to the quantity of services and the number of failures noted. To solve this research question, the customers' marital status, children, yearly equipment failures, contacts to tech support, outage seconds per week, annual GB bandwidth usage, internet services, and any additional addons are utilized for analysis. The research question can provide insight into the failure rates due to multiple services based on total GB usage. The household size can provide reference to the overall usage and service selections.

B. Description of Dataset

The telecommunications churn dataset variables are provided in the below chart. There was a total of fifty-two variables in the original dataset. The variables were found by utilizing the .info() function within the Jupyter Notebook Python environment to display all available columns representing the variables. Please see in the below chart for the variable name, data type, description, and an example of values. The data types are noted as either quantitative or qualitative. Quantitative variables are numerical variables that allow elements to be mathematically computed (Larose & Larose, 2019). Qualitative variables are categorical variables that allow elements to be classified or categorized (Larose & Larose, 2019). In this dataset there are the following quantitative variables: Population, Children, Age, Income, MonthlyCharge, Email, Contacts,

Outage_sec_perweek, Yearly_ equip_failure, Tenure, and Bandwidth_GB_Year. The remaining variables are all qualitative or categorical.

Variable Name	Data Type Quantitative ~ Numerical Qualitative ~ Categorical	Description	Example
Unnamed: 0	Qualitative ~ Categorical	This is an unnamed variable with the numbers correlating to the row number.	1,2,3,4
CaseOrder	Qualitative ~ Categorical	This is the variable that is a placeholder of the original order within the raw data file.	1,2,3,4
Customer_id	Qualitative ~ Categorical	The unique identification number of the customer that starts with a letter followed by numbers.	“K409198”, “S120509”, “K191035”
Interaction	Qualitative ~ Categorical	The identification number of the customer related to the transactions initiated, technical support contacted, and any various sign-ups.	“aa90260b-4141-4a24-8e36-b04ce1f4f77b”, “fb76459f-c047-4a9d-8af9-e0f7d4ac2524”, “344d114c-3736-4be5-98f7-c72c281e2d35”
City	Qualitative ~ Categorical	The customer’s city of residence as noted on their billing statements.	“Point Baker”, “West Branch”, “Yamhill”

State	Qualitative ~ Categorical	The customer's state of residence as noted on their billing statements.	NJ", "NY", "AK", "MI"
County	Qualitative ~ Categorical	The customer's county of residence as noted on their billing statements.	"Prince of Wales-Hyder", "Ogemaw", "Yamhill"
Zip	Qualitative ~ Categorical	The customer's zip code of residence as noted on their billing statements.	"99927", "48661", "97148"
Lat	Qualitative ~ Categorical	The latitude GPS coordinates of their billing address.	"56.25100", "44.32893", "45.35589"
Lng	Qualitative ~ Categorical	The longitude GPS coordinates of their billing address.	"-133.37571", "-84.24080", "-123.24657"
Population	Quantitative ~ Numerical	The total population within a mile of the billing address based on census data.	"38", "10446", "3735"
Area	Qualitative ~ Categorical	The type of area the billing address is classified as based on census data.	Urban, Rural, Suburban
Timezone	Qualitative ~ Categorical	The time zone of the billing address based the customer's information provided at sign-up.	'America/Anchorage', 'America/Boise', 'America/Chicago'

Job	Qualitative ~ Categorical	The job title of the customer as reported in the initial sign-up.	'Academic librarian', 'Accommodation manager', 'Accountant, chartered', 'Accountant, chartered certified'
Children	Quantitative ~ Numerical	The total number of children a customer has based on the information provided in initial sign-up.	0.0, 1.0, 2.0, … 10.0
Age	Quantitative ~ Numerical	The current age of the customer based on the information provided at initial sign-up.	18.0, 19.0, …89.0
Education	Qualitative ~ Categorical	The highest degree completed as provided by the customer at initial sign-up.	'9th Grade to 12th Grade, No Diploma', "Associate's Degree", "Bachelor's Degree", 'Doctorate Degree'
Employment	Qualitative ~ Categorical	The current employment status of the customer as provided by the customer at the initial sign-up.	'Full Time', 'Part Time', 'Retired', 'Student', 'Unemployed'
Income	Quantitative ~ Numerical	The total annual income as provided by the customer at initial sign-up.	740.66, 901.21, 938.81, ..., 256998.4 , 258900.7
Marital	Qualitative ~ Categorical	The current marital status as provided by the customer at initial sign up.	'Divorced', 'Married', 'Never Married', 'Separated', 'Widowed'
Gender	Qualitative ~ Categorical	The gender that customer identifies as.	'Female', 'Male', 'Prefer not to answer'

Churn	Qualitative ~ Categorical	The response to if a customer has discontinued a service within the last month.	No, Yes
Outage_sec_perweek	Quantitative ~ Numerical	The average amount of seconds a week the system has an outage that are noted within a customer's immediate radius.	-1.348571 , -1.19542834, -1.099934 , ..., 46.64180573, 47.02766 , 47.04928
Email	Quantitative ~ Numerical	The total amount of emails sent to a customer in the last year; be it for marketing or general correspondence.	1, 2, 3, 4, 5, 6
Contacts	Quantitative ~ Numerical	The total amount of times a customer contacted technical support.	0, 1, 2, 3, 4, 5, 6, 7
Yearly_equip_failure	Quantitative ~ Numerical	The total amount of times a customer's equipment has failed within the past year that resulted in it being reset or replaced.	0, 1, 2, 3, 4, 5, 6
Techie	Qualitative ~ Categorical	The result of the customer questionnaire in response to if they are technically inclined.	No, Yes
Contract	Qualitative ~ Categorical	The term of the customer's contract.	'Month-to-month', 'One year', 'Two Year'

Port_modem	Qualitative ~ Categorical	The confirmation of whether a customer has a portable modem or not.	No, Yes
Tablet	Qualitative ~ Categorical	The confirmation of whether a customer has a tablet.	No, Yes
InternetService	Qualitative ~ Categorical	The customer's internet service provider.	'DSL', 'Fiber Optic', 'None'
Phone	Qualitative ~ Categorical	The confirmation if a customer has a phone service.	No, Yes
Mulitple	Qualitative ~ Categorical	The confirmation if a customer has multiple lines.	No, Yes
OnlineSecurity	Qualitative ~ Categorical	The confirmation if a customer has an online security add-on service.	No, Yes
OnlineBackup	Qualitative ~ Categorical	The confirmation if a customer has an online backup add-on service.	No, Yes
DeviceProtection	Qualitative ~ Categorical	The confirmation if a customer has a device protection add-on service.	No, Yes
TechSupport	Qualitative ~ Categorical	The confirmation if a customer has a technical support add-on service.	No, Yes
StreamingTV	Qualitative ~ Categorical	The confirmation if a customer has streaming tv.	No, Yes
StreamingMovies	Qualitative ~ Categorical	The confirmation if a customer has streaming movies.	No, Yes
PaperlessBilling	Qualitative ~ Categorical	The confirmation if a customer has	No, Yes

		paperless billing statements.	
PaymentMethod	Qualitative ~ Categorical	The customer preferred payment method.	'Bank Transfer(automatic)', 'Credit Card (automatic)', 'Electronic Check', 'Mailed Check'
Tenure	Quantitative ~ Numerical	The total number of months a customer has been with a provider.	1.00025934, 1.005104 , 1.0185196 , ..., 71.99418
MonthlyCharge	Quantitative ~ Numerical	The average amount the customer is charged per month.	77.50523 , 77.92452 , 78.05288 , ..., 306.268 , 307.5281242, 315.8786
Bandwidth_GB_Year	Quantitative ~ Numerical	The average amount of data used by the customer annually as shown in GB.	155.5067148, 169.3992798, 223.4765826, ..., 7138.309 , 7158.982
item1	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Timely Responses.	1, 2, 3, 4, 5, 6, 7
item2	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Timely Fixes.	1, 2, 3, 4, 5, 6, 7
Item3	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of	1, 2, 3, 4, 5, 6, 7, 8

		Timely Replacements.	
Item4	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Reliability.	1, 2, 3, 4, 5, 6, 7
Item5	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Options.	1, 2, 3, 4, 5, 6, 7
Item6	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Respectful Responses.	1, 2, 3, 4, 5, 6, 7, 8
Item7	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Courteous exchange.	1, 2, 3, 4, 5, 6, 7
Item8	Qualitative ~ Categorical	The questionnaire response to the customer's rate of importance from 1-8 of Evidence of active listening.	1, 2, 3, 4, 5, 6, 7, 8

Part II. Data-Cleaning Plan : Detection

C1. Detection Methods

The telecommunication dataset had ten thousand entries. The first step in the data cleaning process was to detect any anomalies. The anomalies included data quality issues, missing values, inherent missing values, outliers, and duplicates. The detection methods included the use of various functions and visualizations.

The first action of the detection process was to detect any duplicates. The .info() function was used to visually inspect the variable names for any duplicates. Once it was confirmed that the variables were duplicate-free, the .duplicated() function was executed on the data frame. This function returned all rows with a Boolean value to identify if a duplicate was found. False referred to a result of no duplicate found. True referred to a duplicate value being located (Bowne-Anderson, et al., n.d.). The .duplicated() function was run again specifying a categorical variable to confirm that there were no incomplete duplicated rows.

The second action in the detection process was finding the data anomalies. Inherent missing values and data anomalies were detected using general domain knowledge after viewing the results of the .describe() function (Bowne-Anderson, et al., n.d.). The .describe() function provided the count, mean, standard deviation, minimum value, quantile information for each percentage bracket, and maximum value of each variable. Using the provided information, visualizations could be made using those statistics and general knowledge. Observations would include negative values in a variable that elements cannot be lower than zero. Another instance would be a zero value where zero cannot be a legitimate value based on variable subject reference. Domain knowledge utilization helped locate these inherent missing and inaccurate data.

The third action in the detection process was locating missing values or nulls. The null values within each variable were found using the .isnull() function. The .isnull() function returned True and False values comparably to the .duplicated() function. The values that were missing or null were shown as True with non-missing values as False. This function was used in conjunction with the .sum() function to provide a complete count of missing values per variable. The .mean() function was used with the .isnull() function to provide a percentage of null values. This provided the volume of missing data per variable.

The last action in the detection process was to identify outliers within the dataset. Outliers were detected via a created function that provided the total count of outliers per quantitative variable. This allowed the representation of the volume of outliers as well as notate the maximum and minimum outlier values for each variable. The function was defined as "find_outliers". It was used to utilize the interquartile range(IQR) method. The third quantile minus the first quantile defined the IQR. The lower bound value for the variable elements was set equal to the first quantile variable minus 1.5 multiplied by the IQR. It was shown as the following mathematical equation. IQR = Q3-Q1, Q1 – 1.5 * IQR = Lower Bound Value. The upper bound value for the variable elements was set equal to the third quantile added to 1.5 multiplied by the IQR. This was mathematically shown as Q3 + 1.5 * IQR = Upperbound value. The IQR method notated any values less than the lower bound and greater than the upper bound values were outliers.

Completing the above techniques allowed a successful examination of the dataset. These detection methods provided insightful information into the data anomalies. It gave

an overview of what needed to be corrected before treatment could be made and any further assessment.

C2: Justification for Detection Methods

The detection methods selected were based on the distinctive characteristics of the dataset, telecommunications churn. Before cleaning any data, the dataset must be examined. The methods were utilized to provide that exploration. The telecommunication file contained intuitive information regard to the queried customers' personal information, services selected, payment and charges, services failures as well as their overall satisfaction with their interactions with their service provider. As the churn data sample size was somewhat large, the methods of detection were ideal to find missing values or nulls, outliers, inherent missing values, and duplicates.

In the Datacamp resource, the video notated that the .()duplicated function was the optimal procedure to find cloned rows and values (Bowne-Anderson, et al., n.d.). The .duplicated() method was preferred as it gave an overall idea of if duplicated values were complete or incomplete duplicates. Complete duplicates were rows of data that values were identical across all columns (Bowne-Anderson, et al., n.d.). The .duplicated() function was specified on the Customer_id variable as it was a unique identifier for each customer. A variable specification with the .duplicated() function allowed for the results to show incomplete duplicates. Incomplete duplicates are rows of data that only some columns are identical with a discrepancy in one or more of the columns (Bowne-Anderson, et al., n.d.)

The use of general domain knowledge to locate invalid data was used in this detection method. General domain knowledge was documented as a strong method to

identify discrepancies as understanding the variables' limitations can pinpoint invalid values (Bowne-Anderson, et al., n.d.). The .describe() function was utilized to provide the statistical data for each variable (Larose & Larose, 2019). The statistical data included the mean, standard deviation, quantile brackets, minimum, maximum, and total count of values of a given quantitative variable (Larose & Larose, 2019). This statistical data in conjunction with general knowledge identified inherent missing values within the variables. In this dataset, the Population variable was referenced as the population within a mile radius of the customer's billing address based on the census data information. With that information, it is logically impossible for the minimum value returned in the .describe() function for that variable to be zero as the census data would have included the referenced customer. Another variable that had data anomalies was the Outage_ sec_perweek. Using the same method of domain knowledge and the .describe() function, the minimum value for seconds can only be zero. Seconds cannot have a value of less than zero. This justified my approach to identifying the inherent data validity.

The detection method used for identifying outliers was the Interquartile Method. This approach was justified as it accounted for the variation in the spread of the data. The method was noted to be less affected by extreme outlier values (Larose & Larose, 2019). In the telecommunications churn dataset, there was a vast range of values within some variables. For example, the Income amounts ranged from making under $1,000 to over $250,000. The extensive range of Income data skewed the distribution of data and could make the analysis appear unclear. The IQR method considered the quantile points with the data and calculated the upper and lower bounds that would qualify the values as

being outliers. The function utilized in the code was able to show the volume of outliers as well as the maximum and minimum value of the outliers (Kleppen, 2022).

Lastly, the missing values were detected using the ().isnull function. This method was the best method of detection for null values as it returned a Boolean value to find the missing values within each variable (Larose & Larose, 2019). Similarly, to the .duplicated() function, the .isnull() function notated a True value on fields that were missing or null. At a quick glance, it is viewed that the Children variable noted a null on the first row. The combination of the other functions to this .isnull() function added clearer information to the missing data that the Boolean value did not specify. The added .sum() function provided the best insight into the volume of missing data per quantitative variable. This combination provided the total amount of null values. The .mean() function added to the .isnull() function was utilized to show the ratio of missing values compared to the total values in each variable (Bowne-Anderson, et al., n.d.). Multiplication of this result and 100 provided the percentage. The function was ideal in detecting the missing values in this particular dataset as it provided insight on nulls within important data variables such as children, income, tech support, and bandwidth per year. It showed that these variables had at least 10 percent or more of data missing.

C3: Justification for Program Language

The programming language selected for this data cleaning process was Python. As it is noted, Python is a general-purpose language that has flexibility with packages to adapt the data to the needs of the analyst (Larose & Larose, 2019). Some of the benefits of this are the ease of readability and deployment. Most codes in Python are based on actual words describing the function. It was the primary reason for my selection as I

could understand what was being done for each function. Another benefit of the selection of Python was the wide range of packages and libraries that can be used.

There was a variety of packages and libraries available for use to perform complex tasks on the data without having to write the code out (Larose & Larose, 2019). Packages and libraries that were used within the data cleaning process were the following: Pandas, NumPy, Matplotlib.pyplot, Sklearn.decomposition, Missingno, and Seaborn. The Pandas package provided data structures that could make working with relational data easier. It was ideal in the case of this dataset as the data was structured in a table-like fashion with labeling. Importing the CSV file and using the set option feature was a prime example of the use of Pandas. It read the file and set it to display all the columns in the file. NumPy is a package that hosts a library of array objects and the tools that process those arrays. Importing this package was ideal in this instance as it provided the functions to explore the arrays within each variable. For example, utilizing the NumPy where syntax helped in searching the array of values within the variable. The NumPy where returns values where specified conditions are met. Matplotlib.pyplot is a package within the matplotlib library that helps graphically visualize data within python. It was ideal to create basic graphs that are used to show the distribution of the data. This package was used to show a histogram of the quantitative variables in the telecommunication dataset. The histogram allows analysis of the data to see if it was in an ideal distribution. Sklearn is used for machine learning. The decomposition package within Sklearn is used in statistical modeling which includes linear regression (Larose & Larose, 2019). It was primarily used in computing the principal component analysis(PCA). Missingno is a library that is used to visually explore missing data in

Python (Bowne-Anderson, et al., n.d.). In the data, it was used to easily identify the missing data by visually showing where the missing data was sparse. Also, Missingno has a Heatmap feature that showed the correlation of the variables with null values in relation to each other. It provided insight on if the missing data were missing completely at random(MCAR), missing at random(MAR), and missing not at random(MNAR) (Bowne-Anderson, et al., n.d.). Lastly Seaborn is another visualization package that collaborates well with pandas. I utilized Seaborn to show boxplots for outliers in my dataset.

Overall Python worked best in my opinion for this dataset. It managed the larger dataset well as opposed to the other languages where it would have been an issue. It allowed for great readability (Larose & Larose, 2019). Packages and libraries could be added with ease to improve the basic features of Python.

C4: Detection Code

Please see below for a copy of my code to detect anomalies within my data.

```
##detection of duplicates, first by visual of variables within dataset.

churn.info()

#use the duplicated function to find duplicates within the rows.

churn.duplicated()

duplicates = churn.duplicated('Customer_id', keep=False)
churn_duplicated =churn[duplicates].sort_values(by='Customer_id')
print(churn_duplicated[['City', 'State']])

#locate NA/missing values within the data frame
#use describe function to overview for missing data
#review values to detect inherent missing/invalid values within data frame
```

```
churn.describe()

#analyze the missing values.

churn_nullity = churn.isnull()
churn_nullity.head()

#find the total of the missing values

churn_nullity.sum()

#find the percentage of missing values

churn_nullity.mean() * 100

# locating values and volumes of outliers :
#used IQR method, Based on Code from the following website, Careerfoundry.
(Kleppen, 2022)

def find_outliers(df, var):
    q1 = df[var].quantile(0.25)
    q3 = df[var].quantile(0.75)
    IQR = q3 - q1
    lowerbound = q1-(1.5*IQR)
    upperbound = q3+(1.5*IQR)
    outliers = df[var][((df[var] < (lowerbound)) | (df[var] > (upperbound)))]
    return outliers

#run the newly created function on each quantitative variable.

outliers = find_outliers(churn, 'Children')
print("number of outliers in Children: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Age')
print("number of outliers in Age: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Income')
print("number of outliers in Income: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Population')
```

```
print("number of outliers in Population: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'MonthlyCharge')
print("number of outliers in MonthlyCharge: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Email')
print("number of outliers in Email: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn.Outage_sec_perweek)
print("number of outliers in Outage seconds per week: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Tenure')
print("number of outliers in Tenure: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Bandwidth_GB_Year')
print("number of outliers in Bandwidth GB per Year: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Outage_sec_perweek')
print("number of outliers in Outage seconds per week: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(churn, 'Yearly_equip_failure')
print("number of outliers in Yearly equipment failure: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
outliers = find_outliers(churn, 'Contacts')
print("number of outliers in Contacts: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
```

NOTE: See code / script attached: AFCodeD206.ipynb

Part III. Data Cleaning: Treatment

D1. Discussion of Findings

The process of detection in the data cleaning of the telecommunication churn dataset provided a variety of anomalies. The method of detection for finding duplicates showed no duplicates were found. The inherent missing data was found in two variables, Population and Outage_sec_perweek. The totals for these variables were captured during the treatment phase once they were converted to nulls. There was a total of 97 inherent missing values in Population and 11 inherent missing values in Outage_sec_perweek. The missing data during the detection phase was noted in the following variables: Children, Age, Income, Techie, Phone, TechSupport, Tenure, and Bandwidth_GB_Year. The noted variables had the following total of missing data.

Children : 24.95 %, 2495 nulls

Age: 24.75 , 2475 nulls

Income: 24.90%, 2490 nulls

Techie: 24.77%, 2477 nulls

Phone: 10.26%, 1026 nulls

TechSupport: 9.91%, 991 nulls

Tenure: 9.31%, 931 nulls

Bandwidth_GB_Year: 10.21%, 1021 nulls

The outliers in this dataset were found in a total of eight quantitative variables out of the eleven available ones. The eight variables included the following: Children, Population, Income, Email, MonthlyCharge, Outage_sec_perweek, Yearly_equip_failure,

and Contacts. These variables had the following total amount of outliers and those outliers had the following minimum and maximum range of values.

Children: 302 outliers, max outlier value: 10, min outlier value: 8

Population: 937 outliers, max outlier value: 111850, min outlier value: 31816

Income: 249 outliers, max outlier value: 258900.7, min outlier value: 104867.5

Email: 38 outliers, max outlier value: 23, min outlier value: 1

MonthlyCharge: 5 outliers, max outlier value: 315.8786, min outlier value: 298.1730235

Outage_sec_perweek: 539 outliers, max outlier value: 47.04928, min outlier value: -1.348571

Yearly_equip_failure: 94 outliers, max outlier value: 6, min outlier value: 3

Contacts: 8 outliers, max outlier value: 7, min outlier value: 6

The detection methods utilized in this dataset provided an astute exploration of the available data in this data frame. It showed the numerous errors and non-uniformed values that made the full picture of the dataset incomplete. The techniques used highlighted the anomalies in the dataset and provided a stepping stone on how to clean the data for further exploration.

D2. Treatment Methods and Justification

The treatment methods for this dataset were determined based on the detected anomalies found. As no duplicates were noted within the observed data, treatment was not applied. The inherent missing data were treated with the .replace() function and NumPy .where() function. In the churn data frame variable, Population, domain knowledge confirmed that zero found as a minimum could not be a valid value. The

.replace() function was used to change the value to a NaN. The decision to replace the value was due to the zero being a missing value in disguise (Bowne-Anderson, et al., n.d.). Similarly, to the observed Population variable, it was noted that the Outage_sec_per week variable was showing a negative value for a minimum. As seconds cannot be an integer below zero. This would also be notated as a discrepancy in the data (Bowne-Anderson, et al., n.d.). To alleviate guessing if it was a data entry or mathematical error, the decision was made to note this value as null or NaN as well. The function .where() was used to locate the values of the variable that were less than zero and set it to NaN. Once the inherent missing values were treated, the missing values were treated using a combination of methods.

See the example of code used as well as the resulted null values for these two variables after the execution.

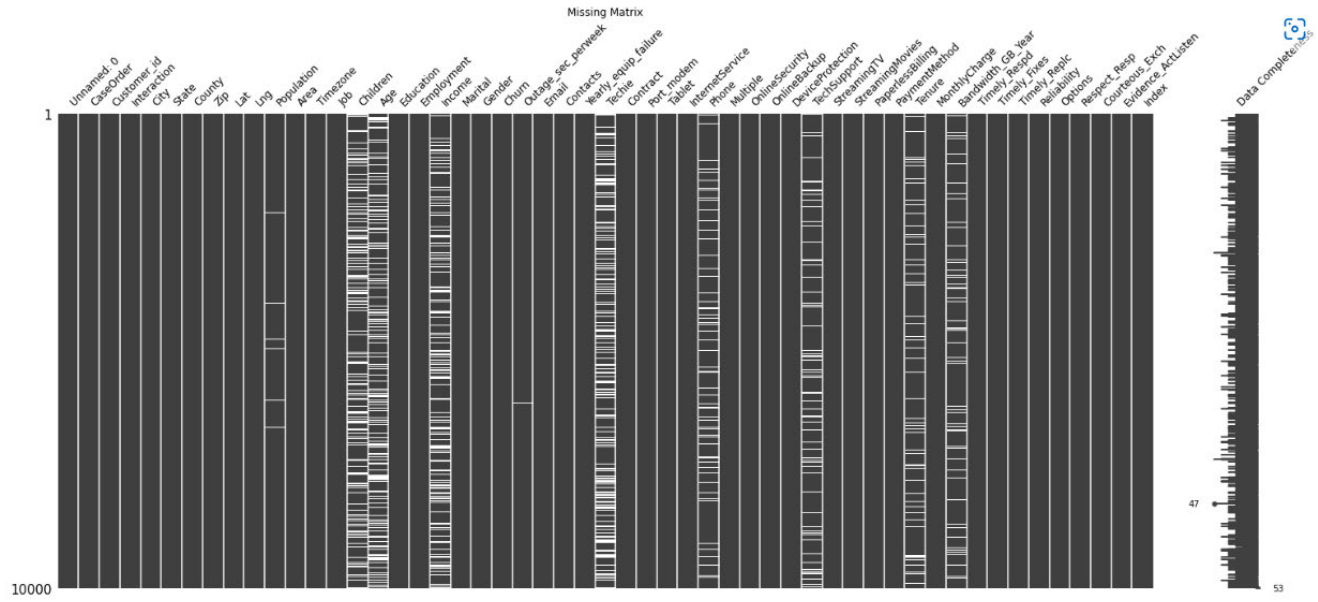
```
#treat the inherent missing values/data anomalies as noted during detection stage.
churn.Population = churn.Population.replace(0, np.nan)
churn.Outage_sec_perweek = np.where(churn.Outage_sec_perweek < 0,
                                    np.nan, churn.Outage_sec_perweek)
```

```
Lat                    0
Lng                    0
Population            97
Area                   0
Timezone               0
Job                    0
Children            2495
Age                 2475
Education              0
Employment             0
Income              2490
Marital                0
Gender                 0
Churn                  0
Outage_sec_perweek    11
Email                  0
Contacts               0
Yearly_equip_failure   0
```

The missing values or nulls were treated by first using visualizations provided by the Missingno matrix and heatmaps to find out any insights as to why the missing data occurred. The missingness can be broken down into three types: Missing completely at random, missing at random, and missing not at random.

Missing completely at random(MCAR) is where there was no relationship between any values, nulls, or non-nulls (Larose & Larose, 2019). This was shown in columns Populations and Outage_sec_perweek as the values seem scarce and randomly scattered.

```
msno.matrix(churn, fontsize = 12, labels=True)
plt.title('Missing Matrix')
plt.show()
```



Missing at random(MAR) is where there is a random loss of data, but the missingness is confined to only certain groupings (Larose & Larose, 2019). The sort_values added to the missingno matrix sorted the values to see if there was any correlation to other columns. In the cleaning process, I selected the variable Children to

sort by as it had a higher number of null values. Upon sorting, the variables did not seem to be missing at random due to no noticeable shift in the data.

```
#sorting to detect what type of missing data.

sorted = churn.sort_values('Children')
msno.matrix(sorted, fontsize = 12, labels=True)
plt.show()
```

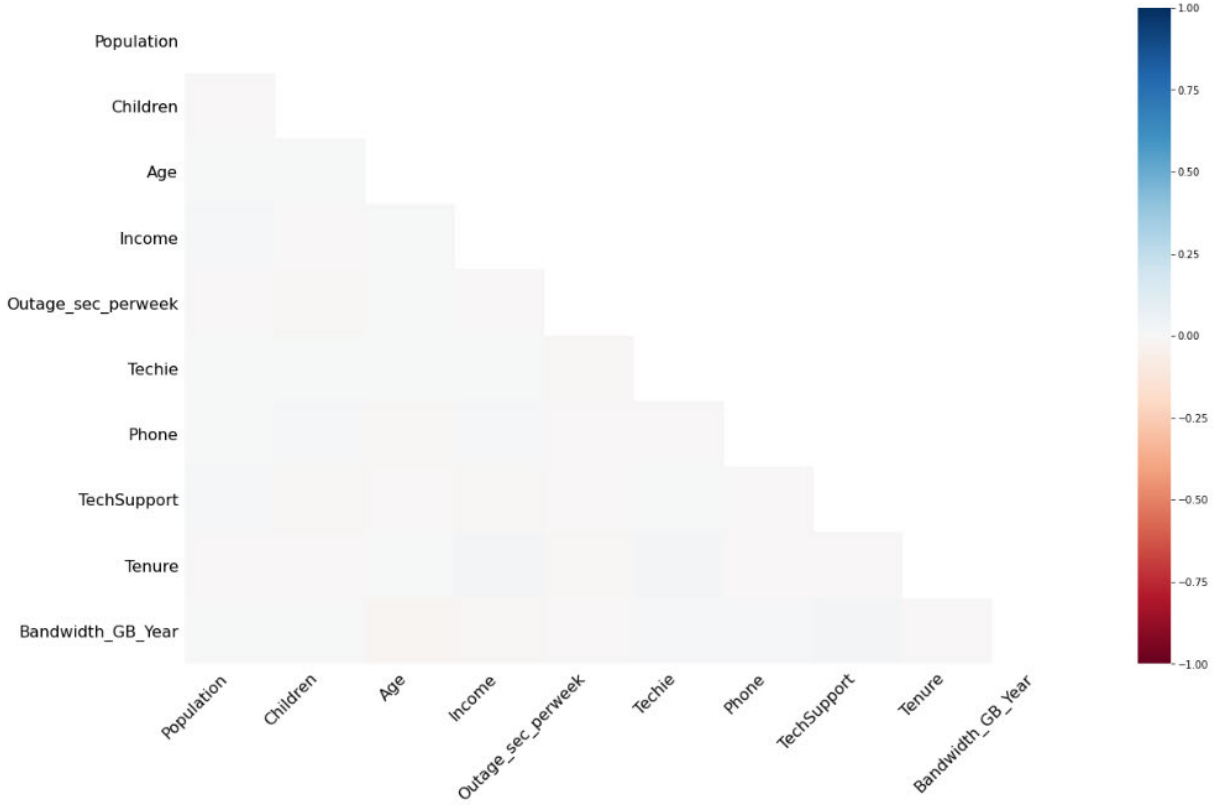
The last missingness type was Missing Not at Random(MNAR), which is the relationship between the missingness in the variable and its values missing or not (Larose & Larose, 2019). I could not find any concrete data to support this missing type, so I utilized the heatmap feature to find any correlation. The heatmap confirmed that the values were MCAR.

The missingno heatmap was used to find and identify relationships between the nullity or missing values of the different variables (Bowne-Anderson, et al., n.d.). Values that are close to positive one indicted that the values in one variable are related to a value in another column. As you can see in the heat map below this is not the case in this dataset. Values close to negative one are identified as missing values that are present when other non-null values are found in another variable and vice versa. Values close to

zero show that there is little to no relationship between the null values in one variable compared to another. This was proven in the case of the telecommunications dataset. All eight variables with missing data were shown to have values close to zero in all cases. This confirmed my hypothesis that the missing type was MCAR.

See below for the heatmap generated for this dataset.

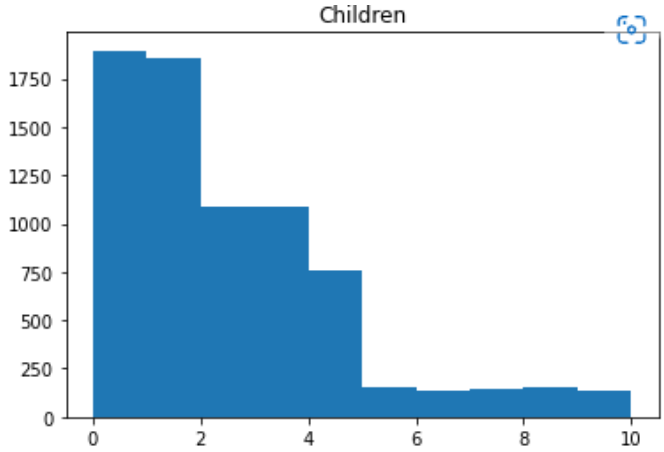
```
: #visualizing the correlation between the missing values.

msno.heatmap(churn)
plt.show()
```



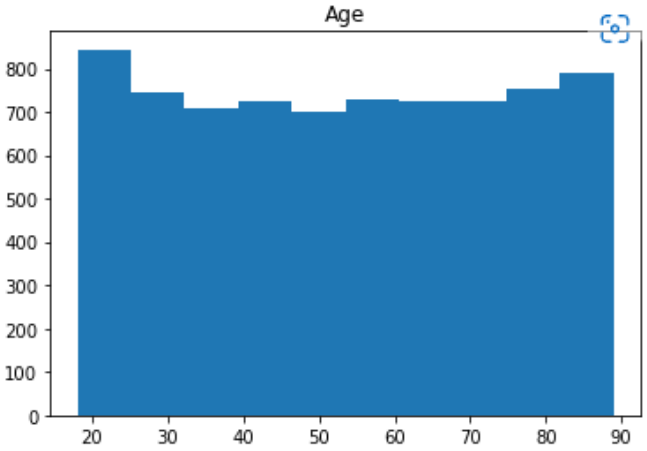
After identifying the missingness type, the decision was made to remove the missing values in the Population and Outage_sec_perweek as the nullity of the data was under 1%. The .dropna() function was applied there. Once these were removed, the remaining quantitative variables of Children, Age, Income, Tenure, and Bandwidth_GB_Year were plotted out in a histogram with the package Matplotlib.pyplot

to show the distribution of the data. This provided the observation on how to treat the variables using the Univariate Imputation method (Larose & Larose, 2019). The Univariate Imputation method stated that if the data had a normal distribution to use the mean to complete the missing values if skewed distribution to use the median, and for any categorial missing values to utilize the mode. The treatment of the null elements was based on these factors. As TechSupport, Techie, and Phone were categorical variables, they were not plotted and noted to use the mode to treat the missing values. The results of the distributions for each quantitative variable were as follows:

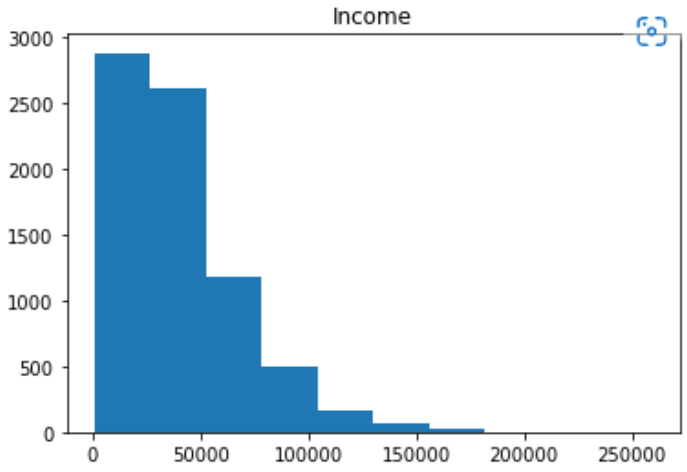
Children: Right Skewed, positively skewed



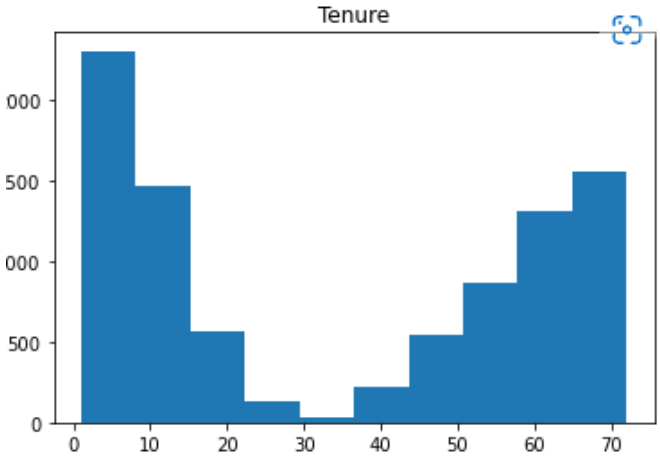
Age: Uniform, equally spread



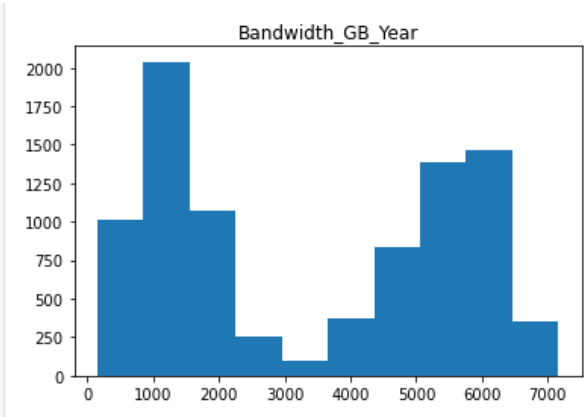
Income: Right Skewed, positively skewed



Tenure: Bimodal non-symmetric



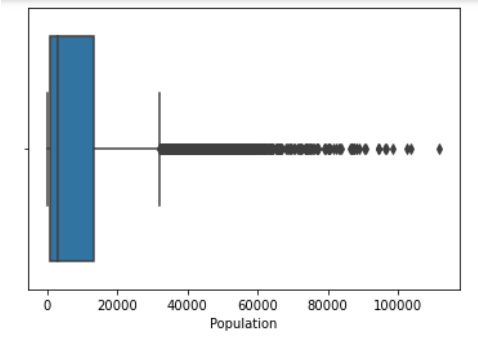
Bandwidth_GB_Year: Bimodal non-symmetric

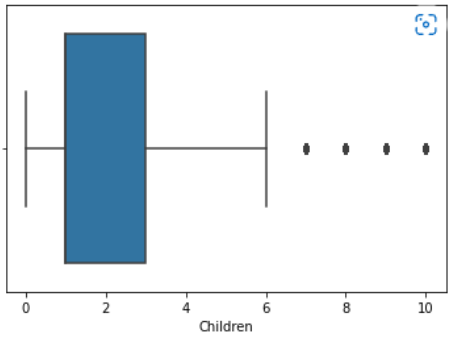


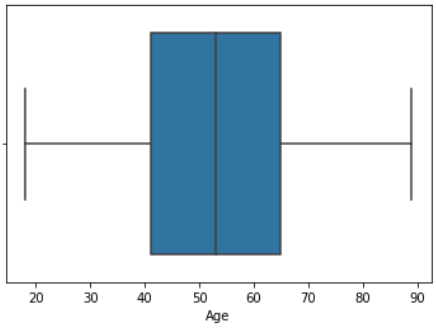
As Children, Age, Income, Tenure, and Bandwidth were all skewed, the median was computed to complete the null entries in the variables. The .fillna() function was applied to compute the median and mode for each respective variable. After all the null values were resolved the outliers were treated.

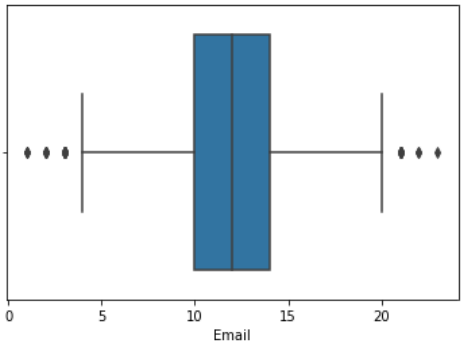
The outliers were corrected using the following treatment methods of boxplots, IQR method to find the lower and upper limit of each variable, and then capped at those limits (Larose & Larose, 2019). I choose to cap the outliers at their limits so as to not lose valuable data. Seaborn boxplots were utilized first to show where the outliers were located. As boxplots can only be used on quantitative variables, only the following variables were plotted: Population, Age, Children, Email, Contacts, Tenure, MonthlyCharge, Income, Yearly_equip_failure, Bandwidth_GB_Year, and Outage_sec_perweek.

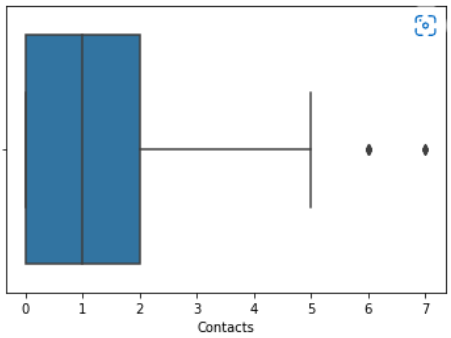
See below for the results of the boxplots.

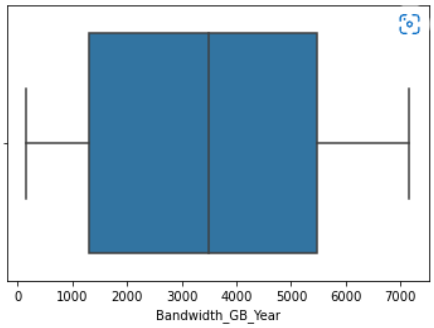


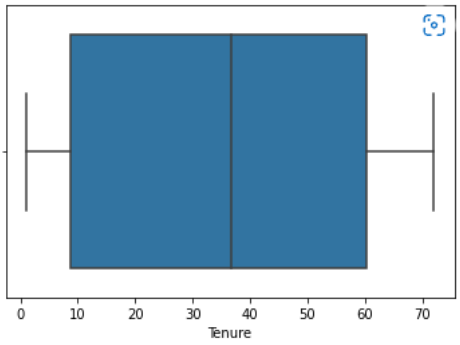


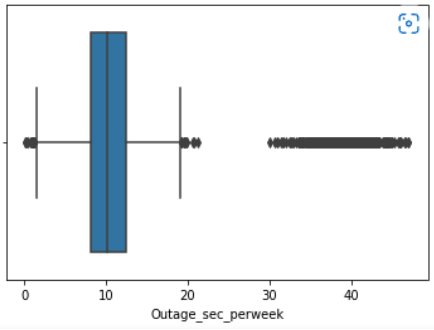


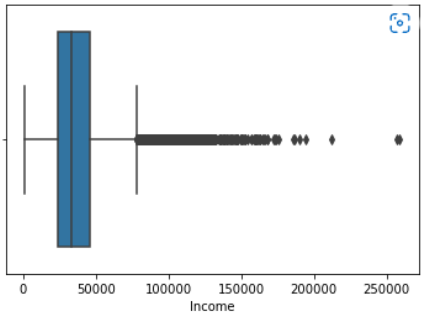


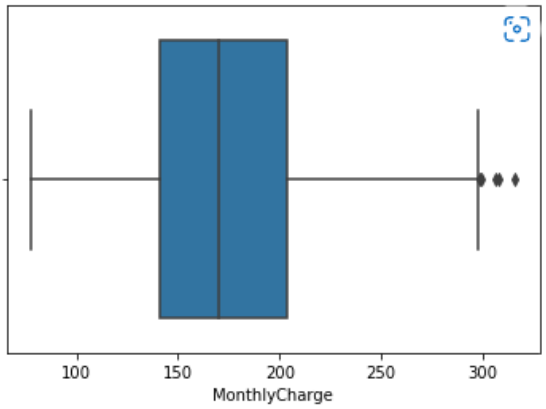


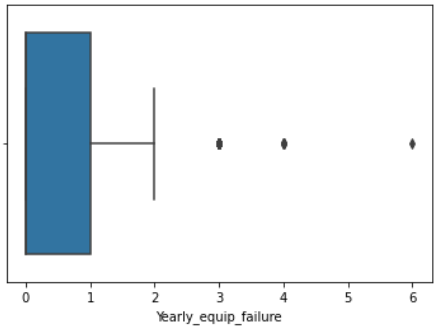












Each of the variables above had outliers, excluding Age, Tenure, and Bandwidth_GB_Year. A function, “find boundaries”, was created to find the boundaries for each outlier to calculate the lower and upper bounds. These boundaries were used

within the NumPy where function to cap the outliers to the nearby bound. The outliers found less than the lower bound were set to the lower bound. The outliers found greater than the upper bound were set to the upper bound.

D3. Summary of Treatment

The data treatment of the telecommunication dataset can be summarized in the following steps. All the inherent missing values were first converted to null values. This only included the Population and Outage_sec_perweek variables. Once these were converted to null, the missing data was visualized utilizing the missingno matrix and heatmaps to determine missingness types. After the missingness types were observed, the determination of how to treat was confirmed. Population and Outage_sec_perweek were deleted using the .dropna() function due to the percentage of nullity being so limited and under 1% of their respective variable. The missingno matrix was repopulated to show that the Population and Outage_sec_perweek were deleted.

See below for the Missingno matrix values after the deletion of null values from the Population and Outage_sec_perweek variables were executed.

The remaining variables with null values were treated with Univariate Imputation which uses distribution types to ascertain if mean or median computations should be applied to fill the missing values. Mode calculations were applied to all categorical variables to populate their null values. The computations were executed and the .isnull() function was run again to show that all were cleaned. The count value of zero for null values in all variables deemed this successfully cleaned.

See below for a copy of the results of the .isnull().sum() function after the .fillna() median and mode were performed.

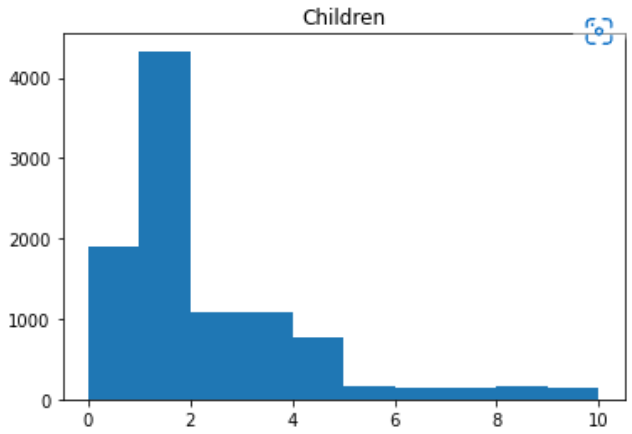
```
churn.isnull().sum()
```

```
Unnamed: 0            0
CaseOrder             0
Customer_id           0
Interaction           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
Education             0
Employment            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
Timely_Respd          0
Timely_Fixes          0
Timely_Replc          0
Reliability           0
Options               0
Respect_Resp          0
Courteous_Exch        0
Evidence_ActListen    0
Index                 0
dtype: int64
```

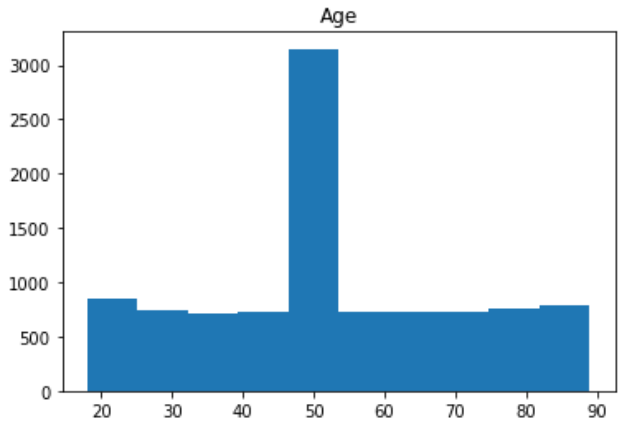
After confirming that the variables were cleaned, new histograms are run to ensure that the distributions of the variables have not changed. This confirms that the overall native state of the dataset remains the same. Any changes in the distribution state can impact any statistical analysis. In this case, all the distributions remained the same.

See below for the results of the re-executed histograms. Please note that all though there were some peaks the overall distribution stayed the same.

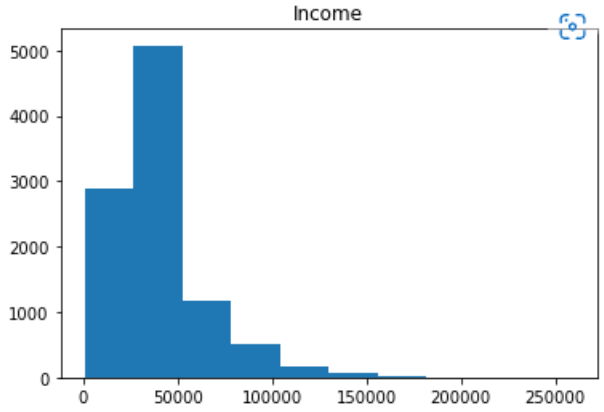
Children: Right Skewed, positively skewed



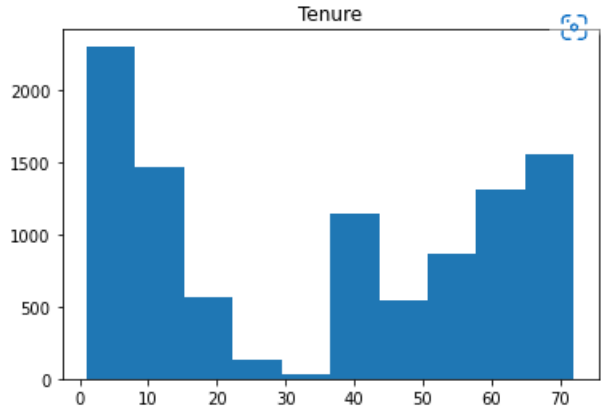
Age: Uniform, equally spread, peak shows, but generally still uniform distribution.



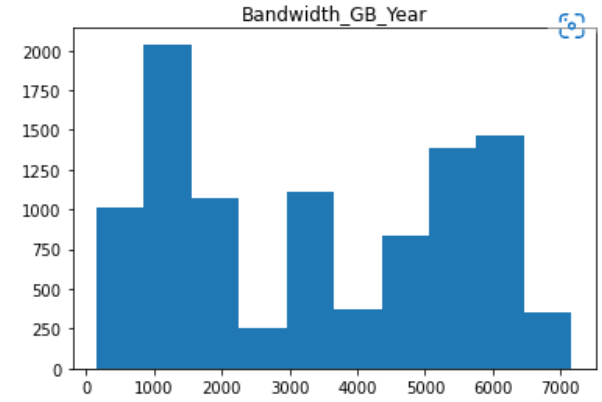
Income: Right Skewed, positively skewed



Tenure: Bimodal non-symmetric

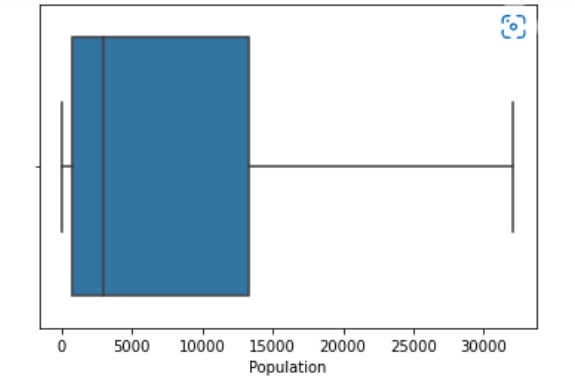


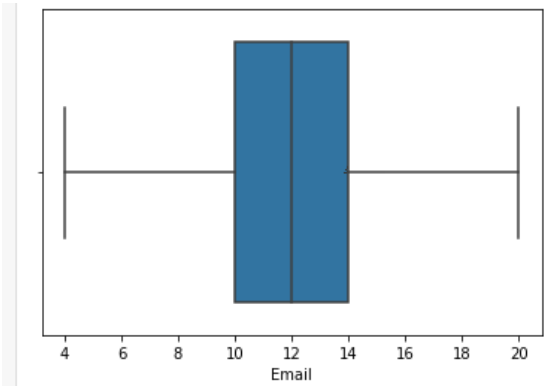
Bandwidth_GB_Year: Bimodal non-symmetric

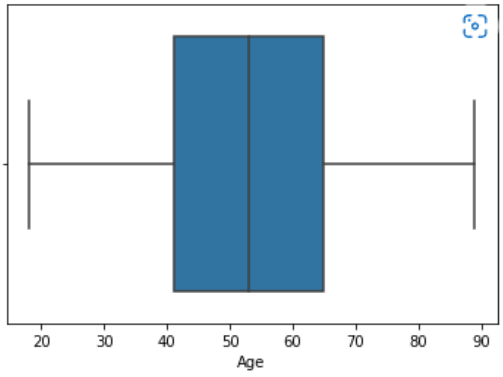


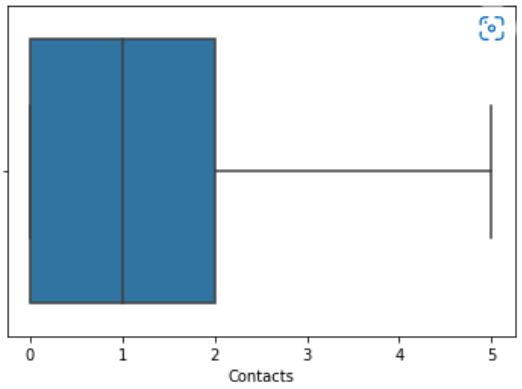
The outliers were treated next using boxplots, the IQR method, and capping each outlier. Once the outliers were cleaned, boxplots were re-executed to ensure that the outliers were successfully resolved.

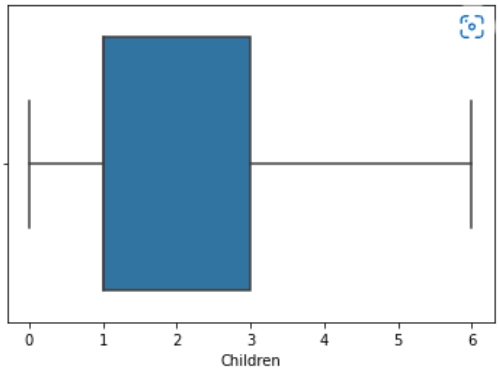
See below for the results of the treatment for outliers.

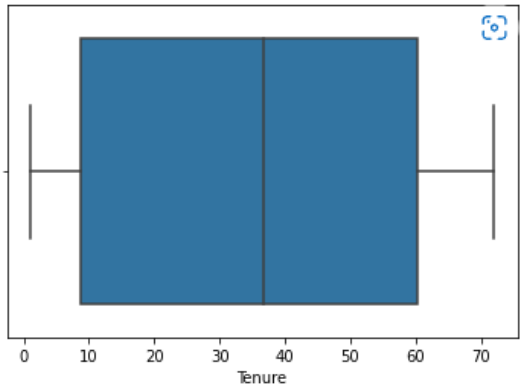


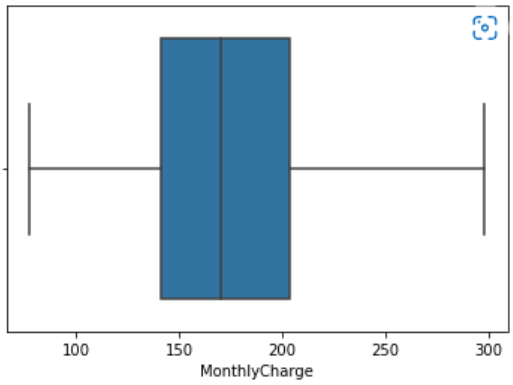


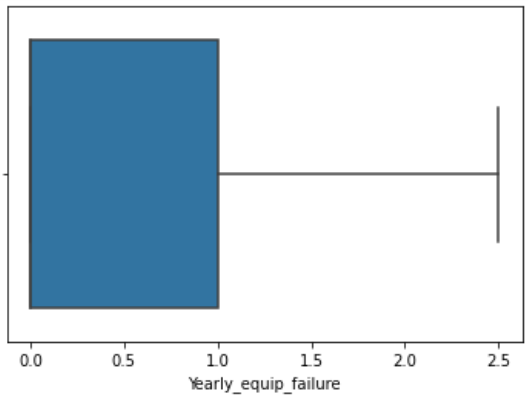


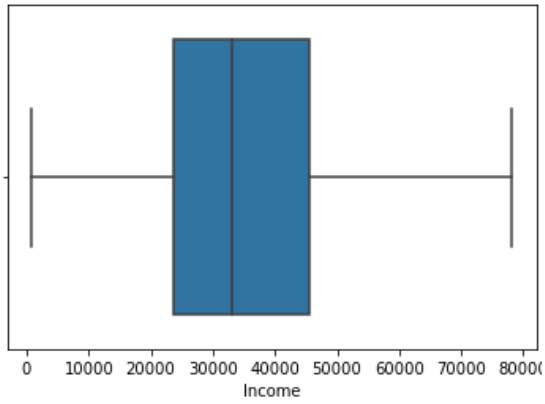


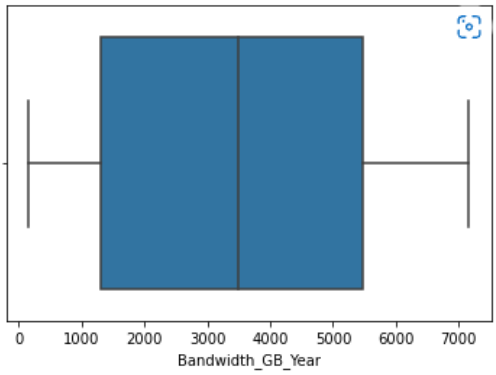


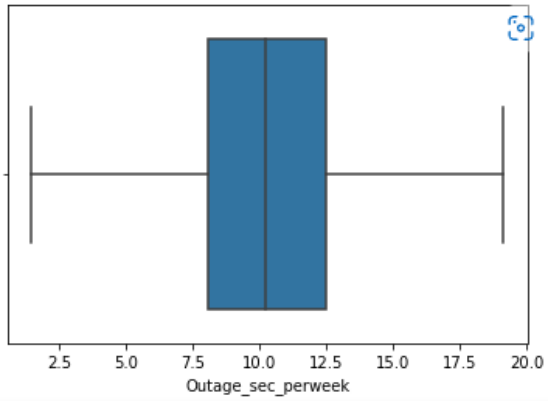












D4. Treatment Code

Please see below for a copy of my code to treat anomalies within my data.

```
#treat the inherent missing values/data anomalies as noted during the detection stage.
churn.Population = churn.Population.replace(0, np.nan)
```

```
churn.Outage_sec_perweek = np.where(churn.Outage_sec_perweek < 0,
                                    np.nan, churn.Outage_sec_perweek)

#visualize and treat missing data.

churn.isnull().sum()

#use the missingno matrix,

msno.matrix(churn, fontsize = 12, labels=True)
plt.title('Missing Matrix')
plt.show()

#sorting to detect what type of missing data.

sorted = churn.sort_values('Children')
msno.matrix(sorted, fontsize = 12, labels=True)
plt.show()

#visualizing the correlation between the missing values.

msno.heatmap(churn)
plt.show()

# deletion of Population/ Outage seconds per week rows with missing values due
to MCAR, % < 1% missing data
churn.dropna(subset=['Population'], how='all', inplace=True)
churn.dropna(subset=['Outage_sec_perweek'], how='all', inplace=True)

#visualizing this change.

msno.matrix(churn, fontsize = 12, labels=True)
plt.show()

# create a histogram of each quantitative variables that have missing data to
examine distribution shapes.

plt.hist(churn.Children)
plt.title('Children')
plt.show()
plt.hist(churn.Age)
plt.title('Age')
plt.show()
plt.hist(churn.Income)
plt.title('Income')
plt.show()
```

```
plt.hist(churn.Tenure)
plt.title('Tenure')
plt.show()
plt.hist(churn.Bandwidth_GB_Year)
plt.title('Bandwidth_GB_Year')
plt.show()


#using univariate imputation to fill missing data.
#using mean for normal distro, median for skewed, mode for
qualitative/categorical

churn.Children.fillna(churn.Children.median(), inplace=True)
churn.Age.fillna(churn.Age.median(), inplace=True)
churn.Income.fillna(churn.Income.median(), inplace=True)
churn.Tenure.fillna(churn.Tenure.median(), inplace=True)
churn.Bandwidth_GB_Year.fillna(churn.Bandwidth_GB_Year.median(),
inplace=True)

#categorical use the mode for missing values
churn.TechSupport = churn.TechSupport.fillna(churn.TechSupport.mode()[0])
churn.Techie = churn.Techie.fillna(churn.Techie.mode()[0])
churn.Phone = churn.Phone.fillna(churn.Phone.mode()[0])


#rerun the isnull to ensure all the missing data is resolved.

churn.isnull().sum()


#re-showing the distributions to make sure that the distribution types have not
changed.
plt.hist(churn.Children)
plt.title('Children')
plt.show()
plt.hist(churn.Age)
plt.title('Age')
plt.show()
plt.hist(churn.Income)
plt.title('Income')
plt.show()
plt.hist(churn.Tenure)
plt.title('Tenure')
plt.show()
plt.hist(churn.Bandwidth_GB_Year)
plt.title('Bandwidth_GB_Year')
```

```
plt.show()


#treat my outliers, first view the shape of the df to view total rows/columns

churn.shape


#boxplot all of the quantitative variables.

boxplot=sns.boxplot(x='Population',data=churn)
plt.show()
boxplot=sns.boxplot(x='Age',data=churn)
plt.show()
boxplot=sns.boxplot(x='Children',data=churn)
plt.show()
boxplot=sns.boxplot(x='Email',data=churn)
plt.show()
boxplot=sns.boxplot(x='Contacts',data=churn)
plt.show()
boxplot=sns.boxplot(x='Tenure',data=churn)
plt.show()
boxplot=sns.boxplot(x='MonthlyCharge',data=churn)
plt.show()
boxplot=sns.boxplot(x='Income',data=churn)
plt.show()
boxplot=sns.boxplot(x='Yearly_equip_failure',data=churn)
plt.show()
boxplot=sns.boxplot(x='Bandwidth_GB_Year',data=churn)
plt.show()
boxplot=sns.boxplot(x='Outage_sec_perweek',data=churn)
plt.show()


def find_boundary(df, var):
    Q1 = df[var].quantile(0.25)
    Q3 = df[var].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1-(1.5*IQR)
    upper = Q3+(1.5*IQR)
    return lower , upper

#run the function and cap the limits of the values to the upper and lower values.

lower_age, upper_age = find_boundary(churn, 'Age')
print("Upper limit for age is" , upper_age)
```

```
print("Lower limit for age is" , lower_age)
churn.Age = np.where(churn.Age > upper_age, upper_age, np.where(churn.Age <
lower_age, lower_age, churn.Age))

lower_eml, upper_eml = find_boundary(churn, 'Email')
print("Upper limit for email is" , upper_eml)
print("Lower limit for email is" , lower_eml)
churn.Email = np.where(churn.Email > upper_eml, upper_eml,
                       np.where(churn.Email < lower_eml, lower_eml, churn.Email))

lower_contct, upper_contct = find_boundary(churn, 'Contacts')
print("Upper limit for contacts is" , upper_contct)
print("Lower limit for contacts is" , lower_contct)
churn.Contacts = np.where(churn.Contacts > upper_contct, upper_contct,
                       np.where(churn.Contacts < lower_contct, lower_contct,
churn.Contacts))

lower_tnr, upper_tnr = find_boundary(churn, 'Tenure')
print("Upper limit for tenure is" , upper_tnr)
print("Lower limit for tenure is" , lower_tnr)
churn.Tenure = np.where(churn.Tenure > upper_tnr, upper_tnr,
                       np.where(churn.Tenure < lower_tnr, lower_tnr, churn.Tenure))

lower_mc, upper_mc = find_boundary(churn, 'MonthlyCharge')
print("Upper limit for MonthlyCharge is" , upper_mc)
print("Lower limit for MonthlyCharge is" , lower_mc)
churn.MonthlyCharge = np.where(churn.MonthlyCharge > upper_mc, upper_mc,
                       np.where(churn.MonthlyCharge < lower_mc, lower_mc,
churn.MonthlyCharge))

lower_inc, upper_inc = find_boundary(churn, 'Income')
print("Upper limit for Income is" , upper_inc)
print("Lower limit for Income is" , lower_inc)
churn.Income = np.where(churn.Income > upper_inc, upper_inc,
                       np.where(churn.Income < lower_inc, lower_inc,
churn.Income))

lower_yef, upper_yef = find_boundary(churn, 'Yearly_equip_failure')
print("Upper limit for Yearly_equip_failure is" , upper_yef)
print("Lower limit for Yearly_equip_failure is" , lower_yef)
churn.Yearly_equip_failure = np.where(churn.Yearly_equip_failure > upper_yef,
upper_yef,
                       np.where(churn.Yearly_equip_failure < lower_yef, lower_yef,
churn.Yearly_equip_failure))

lower_osp, upper_osp = find_boundary(churn, 'Outage_sec_perweek')
```

```
print("Upper limit for Outage_sec_perweek is" , upper_osp)
print("Lower limit for Outage_sec_perweek is" , lower_osp)
churn.Outage_sec_perweek = np.where(churn.Outage_sec_perweek > upper_osp,
upper_osp,
                    np.where(churn.Outage_sec_perweek < lower_osp, lower_osp,
churn.Outage_sec_perweek))

lower_bgy, upper_bgy = find_boundary(churn, 'Bandwidth_GB_Year')
print("Upper limit for Bandwidth_GB_Year is" , upper_bgy)
print("Lower limit for Bandwidth_GB_Year is" , lower_bgy)
churn.Bandwidth_GB_Year = np.where(churn.Bandwidth_GB_Year >
upper_bgy, upper_bgy,
                    np.where(churn.Bandwidth_GB_Year < lower_bgy,
lower_bgy, churn.Bandwidth_GB_Year))

lower_kid, upper_kid = find_boundary(churn, 'Children')
print("Upper limit for children is" , upper_kid)
print("Lower limit for children is" , lower_kid)
churn.Children = np.where(churn.Children > upper_kid, upper_kid,
                    np.where(churn.Children < lower_kid, lower_kid,
churn.Children))

lower_pop, upper_pop = find_boundary(churn, 'Population' )
print("Upper limit for population is" , upper_pop)
print("Lower limit for population is" , lower_pop)
churn.Population = np.where(churn.Population > upper_pop, upper_pop,
                    np.where(churn.Population < lower_pop, lower_pop,
churn.Population))

boxplot=sns.boxplot(x='Population',data=churn)
plt.show()
boxplot=sns.boxplot(x='Age',data=churn)
plt.show()
boxplot=sns.boxplot(x='Children',data=churn)
plt.show()
boxplot=sns.boxplot(x='Email',data=churn)
plt.show()
boxplot=sns.boxplot(x='Contacts',data=churn)
plt.show()
boxplot=sns.boxplot(x='Tenure',data=churn)
plt.show()
boxplot=sns.boxplot(x='MonthlyCharge',data=churn)
plt.show()
boxplot=sns.boxplot(x='Income',data=churn)
plt.show()
boxplot=sns.boxplot(x='Yearly_equip_failure',data=churn)
```

```
plt.show()
boxplot=sns.boxplot(x='Bandwidth_GB_Year',data=churn)
plt.show()
boxplot=sns.boxplot(x='Outage_sec_perweek',data=churn)
plt.show()

churn.shape
```

NOTE: See code / script attached: AFCodeD206.ipynb

D5. Clean Dataset

Please see attached CSV data file: AFCode206_clean.csv to view the results of the cleaned data.

D6. Limitations

Each method of the data cleaning process has its own limitations. For the methods used in the treatment process above, these are the limitations noted. The conversion of invalid or inherent missing values to null could lead to distortion if the assumptions are not validated. In the case of the deletion of missing values, there is a risk of the loss of important data. The utilization of univariate imputation can possibly lead to the distortion of the distribution in the data depending on the variables it is applied on. It can later affect the statistical analysis. In the case of the outliers the IQR method, the 1.5 times the IQR is not ideal for treating variables in larger datasets because it can lead to skewed results deeming greater or lesser values as an outlier when they are not. All of these are prime examples of what each method can be limited to.

D7. Implications

The cleaned dataset can have a few challenges in its current state. One of the noted issues that arose with treating the null values and outliers was a restriction in the elements

of the Income, Children, Contacts, Yearly_equip_failure, and Outage_sec_perweek. This can limit the results in the analysis of service failures as it relates to income and household variables. The clean dataset does not allow for a wide range of income levels to be captured. Currently, the American economic statuses are broken up into three distinct income bracket groups, Upper, Middle, and Low (Snider, 2021). Upper denoting income over $156,000, Low denoting income less than $52,000, with Middle denoting income falling between the two. The currently cleaned dataset will not show any information on the Upper-income grouping as values are capped at a little under $80,000. The capping of outliers similarly affects the Outage_sec_perweek and Yearly_equip_failure as they will not capture the rise of failures after the capped amount. The treatment in nulls can give some misleading insights to the Bandwidth_GB_Year variable as it populates the values to be right around 3000-4000 GB, but there is no investigation on if this guesstimate is accurate. When we consider the usage to failure ratio, it can provide skewed information.

Part IV. PCA

E1: Variables and PCA Loadings

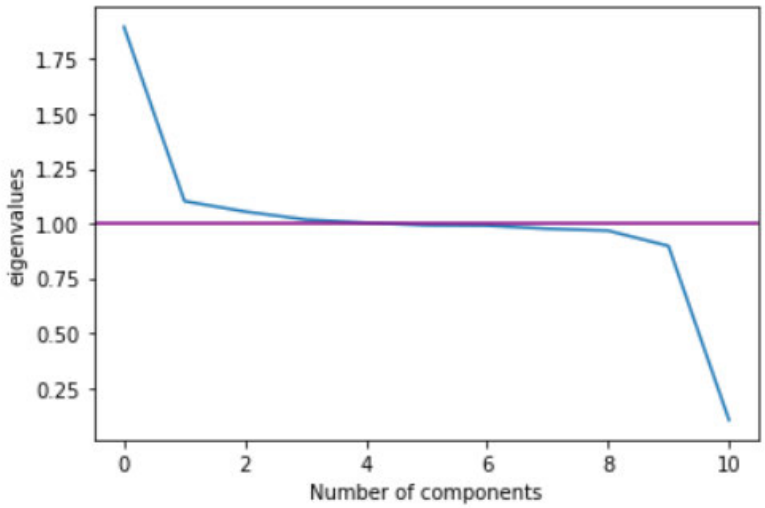
The variables used for the principal component analysis(PCA) were as following quantitative variables within the dataset: 'Population', 'Income', 'MonthlyCharge', 'Children', ‘Age’, 'Contacts', 'Email', 'Yearly_equip_failure', 'Outage_sec_perweek', 'Bandwidth_GB_Year', and ‘Tenure’.

Please see below for a screenshot of the PCA loadings.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
Population	-0.009518	-0.074293	-0.417858	-0.316784	0.051284	0.290441	0.620591	-0.261549	0.422752	-0.013608	0.000863
Income	0.001668	-0.085197	0.201347	0.206349	-0.791356	0.270167	0.241103	-0.273822	-0.272476	0.055054	0.000417
MonthlyCharge	0.046539	0.692582	-0.070491	0.040812	-0.061298	-0.032134	-0.051360	-0.128883	0.130955	0.684174	-0.047410
Children	-0.002177	0.045458	0.496351	-0.190809	-0.096682	0.542732	-0.191342	0.357609	0.496278	-0.009835	-0.019152
Age	-0.013039	-0.044869	-0.269786	0.593133	0.147840	0.389541	-0.409013	-0.399424	0.232993	-0.134412	0.021053
Contacts	0.005780	0.015477	-0.429081	0.440293	-0.184172	0.054764	0.202954	0.734281	0.051131	0.043079	-0.003312
Email	-0.021479	0.058592	-0.398242	-0.434689	0.003295	0.506976	-0.313883	0.110481	-0.527536	0.049717	0.004900
Yearly_equip_failure	0.013233	0.067401	0.342884	0.289990	0.546126	0.367419	0.445215	0.030538	-0.380900	0.127000	-0.002411
Outage_sec_perweek	0.020645	0.701071	0.008448	-0.000141	-0.062654	0.007315	0.109714	-0.011600	-0.056237	-0.699041	-0.005392
Bandwidth_GB_Year	0.706868	-0.006392	-0.000521	-0.016545	-0.003997	0.010300	-0.005785	0.006286	0.006482	0.007942	0.706911
Tenure	0.704846	-0.060951	-0.019702	-0.002784	0.006251	0.011846	-0.013957	-0.009851	-0.010648	-0.036717	-0.705088

E2: PCs Selection

The PC selections were made by utilizing the eigenvalues of the principal components and plotting them on a Scree plot. The Scree plot is used to determine the number of principal components to retain to make a principal component analysis. In the Scree plot below, the eigenvalues had the desired result only in the following principal components, PC1, PC2, and PC3. This observation was confirmed based on the eigenvalue of greater than one in only these three.



E3. Benefits

The benefits of PCA are that it reduces the multi-dimensionally of data, provides visualizations of the dimensions, and compresses information into components. Overall PCA is a great tool to de-clutter the data into more compact data points for analysis. PCA reduces the multi-dimensionally by retaining the data that show variance between the data. The variables are compressed into principal components. In the case of this dataset, there were only three principal components. With the first component, we could see the positive relationship between Bandwidth_GB_Year and Tenure. That can tell the organization that when the Bandwidth_GB_Year increases at the same time as the Tenure of the customer. In the second component, it is noted a positive relationship between Outage_sec_perweek and MonthlyCharge. It can provide insight on if these outages are causing an increase in the monthly charge of the customer. It would be something that would need to be explored further as an organization. Lastly, the third component is a multifaceted relationship between Children, Population, Email, Contacts, and Yearly_equip_failure. This can provide some insight to the organization because it notes that the children and Yearly_equip_failure increase their values at the same time meaning there can be a marked correlation.

Part V. Supporting Documents

F. Panopto Video

Please see attached Panopto video link. This is a video providing an overview of the Python code used to discover anomalies and the data cleaning process. The recording will

demonstrate the code's warning and error-free functionality as well as provide an overview of the programming environment used, Jupyter Notebook.

Link Found here: [redacted]

G. Third Party Web Sources

Kleppen, E. (2022, February 24). *How to Find Outlier in Data Using Python (and How to Handle Them)*. Retrieved from CareerFoundry: https://careerfoundry.com/en/blog/data-analytics/how-to-find-outliers/

H. References

Bowne-Anderson, H., Cornellssen, J., Schourwenaars, F., Nehme, A., Matsui, M., Donthi, S., . . . Castanedo, F. (n.d.). *DataCamp D206 Custom Track Data Cleaning*. Retrieved from DataCamp: https://app.datacamp.com/learn/custom-tracks/custom-d206-data-cleaning

Larose, C. D., & Larose, D. T. (2019). *Data Science Using Python And R.* Hoboken: John Wiley & Sons, Inc.

Snider, S. &. (2021, December 16). Where do I fall in the American economic class system. Retrieved April 28, 2022, from US News.: https://money.usnews.com/money/personal-finance/family-finance/articles/where-do-i-fall-in-the-american-economic-class-system