
PERFORMANCE ASSESSMENT

KAILI HAMILTON

Masters of Science in Data Analytics, Western Governors University

Course: D206 Data Acquisition

Instructor: Dr. Keiona Middleton

Program Mentor: Krissy Bryant

April, 2023

PART I: RESEARCH QUESTION

A

Research question: Which variables most determine if a customer will discontinue service (i.e., churn)?

B

The table describes the index, data type, description, and example for each variable in the data set.

The rows highlighted in yellow indicate that the datatype was converted to its proper form from the original dataset.

<i>Index & Variable</i>	<i>Data type</i>	<i>Description</i>	<i>Example</i>
0. CaseOrder	Category	Place holder variable used to preserve the original order of the raw data set	2
1. Customer_id	Object	Unique customer ID to identify each customer	S120509
2. Interaction	Object	Another Unique customer ID, longer in length than Customer_id, that identifies transactions, technical support, and signups	fb76459f-c047-4a9d-8af9-e0f7d4ac2524
3. City	Object	Customer's city of residence	West Branch
4. State	Object	Customer's state of residence	MI
5. County	Object	Customer's county of residence	Ogemaw
6. Zip	Category	Customer's zip code of residence	48661
7. Lat	Float64	GPS coordinate, latitude, of customer's residence	44.32893
8. Lng	Float64	GPS coordinate, longitude, of customer's residence	-84.24080
9. Population	Int64	Population of customer's residence based on census data, within a mile radius	10446
10. Area	Object	Rural, urban, or suburban area type based on census data for customer's residence	Urban
11. Timezone	Object	Time zone of customer's residence, based on their signup data	America/Detroit
12. Job	Object	Customer's (or invoiced person's) self reported job	Programmer, multimedia
13. Children	Int64	Number of children of customer as reported by customer	1
14. Age	Int64	Age of customer as reported by customer	27

15. Education	Object	Highest degree of customer as reported by customer	Regular High School Diploma
16. Employment	Object	Employment status of customer as reported by customer	Retired
17. Income	Float64 (round 2)	Annual income of customer as reported by customer	21704.77
18. Marital	Object	Marital status of customer as reported by customer	Married
19. Gender	Object	Gender of customer as reported by customer (male, female, nonbinary)	Female
20. Churn	Object	Yes or no indicating if the customer discontinued service within the last month	Yes
21. Outage_sec_perweek	Float64 (round 2)	Average number of seconds per week of system outages in the customer's neighborhood	12.01
22. Email	Int64	Number of emails sent to the customer in the last year, including marketing or correspondence	12
23. Contacts	Int64	Number of times the customer contacted technical support	0
24. Yearly equip_failure	Int64	The number of times customer's equipment failed and had to be reset/replaced in the past year	1
25. Techie	Object	Yes or no indicating if the customer considers themselves technically inclined	Yes
26. Contract	Object	Type of contract (month-to-month, one year, two year)	Month-to-month
27. Port_modem	Object	If the customer has a portable modem (yes or no)	No
28. Tablet	Object	If the customer owns a table device (yes, no)	Yes
29. InternetService	Object	Customer's type of internet service (DSL, fiber optic, None)	Fiber Optic
30. Phone	Object	If the customer has a phone service (yes or no)	Yes
31. Multiple	Object	If the customer has multiple lines (yes, no)	Yes
32. OnlineSecurity	Object	If the customer has online security add-ons (yes, no)	Yes
33. OnlineBackup	Object	If the customer has online backup add-ons (yes, no)	No
34. DeviceProtection	Object	If the customer has device protection add-ons (yes, no)	No
35. TechSupport	Object	If the customer has technical support add-ons (yes, no)	No
36. StreamingTV	Object	If the customer has streaming tv (yes, no)	Yes

37. SteamingMovies	Object	If the customer has streaming movies (yes, no)	Yes
38. PaperlessBilling	Object	If the customer has paper billing (yes, no)	Yes
39. PaymentMethod	Object	The customer's payment method (electronic check, mailed check, bank (automatic bank transfer), credit card (automatic))	Bank Transfer(automatic)
40. Tenure	Float64 (round 2)	The number of months the customer has stayed with the provider	1.16
41. MonthlyCharge	Float64 (round 2)	Average monthly charge for the customer	242.95
42. Bandwidth_GB_Year	Float64 (round 2)	Average amount of data used, in GB, in a year used by the customer	800.98
43. item1	Category	How important "timely response" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	3
44. item2	Category	How important "timely fixes" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	4
45. item3	Category	How important "timely replacements" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	3
46. item4	Category	How important "reliability" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	3
47. item5	Category	How important "options" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	4
48. item6	Category	How important "respectful response" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	3
49. item7	Category	How important "courteous exchange" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	4
50. item8	Category	How important "evidence of active listening" is to the customer a scale of 1 to 8 (1 = most important, 8 = least important)	4

PART II: DATA-CLEANING PLAN

C1

In the following paragraphs, you will find my plan for how I cleaned the `churn_raw_data.csv` dataset and prepared it for future modeling usage. The justification for each step is found in parts C2, C3, and D2. The findings from each step in the data cleaning process are described in parts D1 and D3. Limitations of my data cleaning process are detailed in parts D6 and D7. Annotated code is provided in an executable script to assess and mitigate data quality issues as required by parts C4 and D4.

First, I engaged in a pre-cleaning process to try to get a big picture of what is in the data. To do this I imported some initial libraries and packages, imported the dataset, looked at the head of the data, got the shape of the data, obtained information about the data types and nullity of each variable, and got some summary statistics of the data. I also reset the index of the rows to begin at 0.

Once I had a sense of the data I began to assess and treat the data for quality issues. I detected and treated the data for duplicates, missing values, converted variables to their proper data types, then looked for and treated outliers (Middleton, n.d., Webinar 2). At this stage of the data cleaning process, I also rounded some of the float data types to make it more readable. This ended the data cleaning process.

Now that my data was cleaned, I engaged in some data wrangling techniques to prepare the dataset for future modeling and exploration. I used ordinal encoding on categorical variables with yes or no responses, where “no” is 0 and “yes” is 1 (Middleton, n.d., Webinar 3). Also, I applied principal component analysis and identified significant features of the data set to reduce the dimensionality of the dataset (Middleton, n.d., Webinar 4). The PCA is described in section E.

C2

Here I will describe and justify my approach for assessing quality of the data for each characteristic of the data.

Precleaning. Before cleaning the data, I needed a sense of what I was working with in the `churn_raw_data.csv` dataset. To do this I imported some preliminary libraries including `pandas`, `numpy`, and `matplotlib`, read in the data set, looked at the head of the dataset to see what the value of each variable looked like. I also got the shape of the dataset to see how many rows and columns I was working with. There were 50 variables and 10,000 entries.

Duplicates. Once I had an idea of what the dataset was like, I assessed the quality of the data by looking for duplicates. I called `df.duplicated()` and discovered that there were not any duplicate rows. I would have removed any duplicate rows if there were any.

Missing values. Next, I assessed the quality of the data by looking for missing values. To do this I called `df.info()` on the data frame so I could see the data types and the number of null values for each variable. I discovered that there were nine variables that had missing values. For all the non-numeric data types I imputed the missing values with the most frequently occurring value already present in the dataset. In this way, the mode still remained the mode. For the numeric data types I imputed with the median with the exception of the variables age and income, which I imputed using KNN. Using the median preserved the shape of each variable's distribution. Age and income have a relationship, the older you are the more likely it is that your income is higher. Because of this I used the more advanced imputation technique of KNN to impute the missing values.

Data types. Now that all the variables did not contain any null values I addressed the data types of each variable. I changed the 'CaseOrder', 'Zip', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', and 'item8' variables from int64 to categorical data types. Even though the value entered for these is a number, they are actually ordinal values, except for 'Zip' which is a nominal variable. Next, I changed 'Children' and 'Age' to int64 data types. There weren't any decimal values in these columns and it made sense to use a whole number for the number of children and the age of a customer. At this stage I also rounded all float data types, except latitude and longitude, to 2 decimal places for better readability. I kept latitude and longitude to preserve accuracy for the location values.

Outliers. Lastly, I assessed the dataset for volume and values of outliers on the numerical data types so I would know how to treat them by removing them, retaining them, excluding them, or replacing them (Middleton, n.d., Webinar 2). These variables are: 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', and 'Bandwidth_GB_Year'. Most of the data is not Normally distributed so I opted to use boxplots rather than z-scores, to determine if a variable has outliers. After viewing the boxplots for these numeric variables, I chose to retain all the outliers for variables except for 'Income'. I retained outliers because I think these could affect why customers churn and are important to our question of interest. I chose to drop customers with incomes greater than \$200,000 based on how extreme these values are compared to the overall distribution. In addition, these extremely high outliers will affect the effectiveness of PCA, another reason to drop these values.

C3

The programming language I used to clean the data is Python. Although Python has somewhat weak graphics and visualization capabilities, it is extremely readable and its simplicity makes it easy to learn.

“Python is great for building data science pipelines and machine learning products integrated with web frameworks at scale. However, be mindful of dependencies when installing Python libraries” (WGU, 2023).

Ultimately, I chose Python because I had previously learned it in a data analyst certification program I did prior to my time here at WGU. I plan to learn R in the future so I can have multiple arrows in my data science quiver.

Here are the libraries and packages I used and why I used them in my data cleaning process:

- Pandas – for working with data frames
- Numpy – for multidimensional arrays and numerical data types
- Matplotlib – for data visuals including histograms, boxplots, density curves, and scatterplots
- Missingno – for visualization of missingness of variables
- Datetime – for working with variables that are dates or times
- Seaborn – for visualizations
- Statsmodels – for statistics and PCA
- Scikit learn, KNN Imputer – for imputing missingness of data
- Scikit learn, Iterative Imputer – for imputing missingness of data
- Scikit learn, decomposition, PCA – for principal component analysis

C4

The annotated code that shows what I used for assessing the quality of the data is provided in an executable script file with the submission of this performance assessment.

PART III: DATA CLEANING

D1

Here I will describe the findings for the data quality issues that I found when I executed my plan as described in part C1. The executable script file included for part C4 and D4 contains the code and visuals for these findings.

Precleaning. I imported some preliminary libraries including pandas, numpy, and matplotlib, read in the data set. I looked at the head of the dataset and observed that the first column of the file was an index column so I reimported the data without that column. I next realized that the row index was counting at 1, 'CaseOrder', which is serving as a place holder variable used to preserve the original order of the raw data set. I decided to reset the index to start counting at 0 for coding purposes and kept the 'CaseOrder' column while I was cleaning the data.

I next called up the shape of the dataset because I wanted to see how many rows and columns I was working with. I found that there were 50 features about a customer (columns) and 10,000 customers (rows). The 'CaseOrder' does create a 51st column, but that is not a feature of a customer.

Also, by looking at the head of the data, I got a sense of what the value of each variable looked like. I noticed values that would make more sense to round, such as 'Income' and 'MonthlyCharge'. After I imputed missing values and fixed data types, I rounded all float data types, except latitude and longitude, to 2 decimal places for better readability. I kept latitude and longitude to preserve accuracy for the location values.

Also, I called `df.info()` and `df.describe()` so I could see what type each variable was, if there were any null values, and to see some summary statistics for each variable.

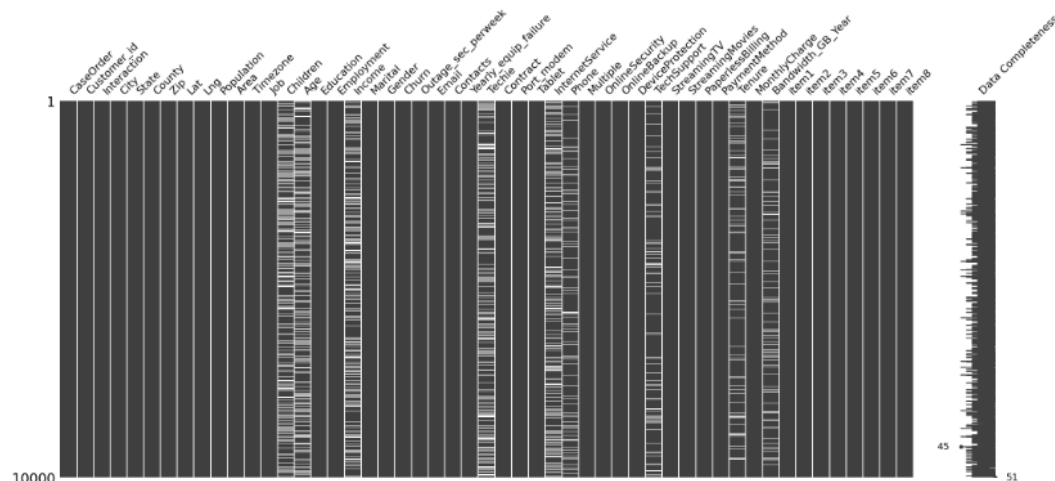
Duplicates. I sought to detect and treat any duplicate rows via `df.duplicated()` in the data set but there weren't any so I was able to move on to imputing missing values.

Missing values. Next, I addressed missing values by calling `df.isna().sum()` to obtain a count of null values for each variable. I found that there were nine variables that had missing values: 'Children', 'Age', 'Income', 'Techie', 'InternetService', 'Phone', 'TechSupport', 'Tenure', and 'Bandwidth_GB_Year'.

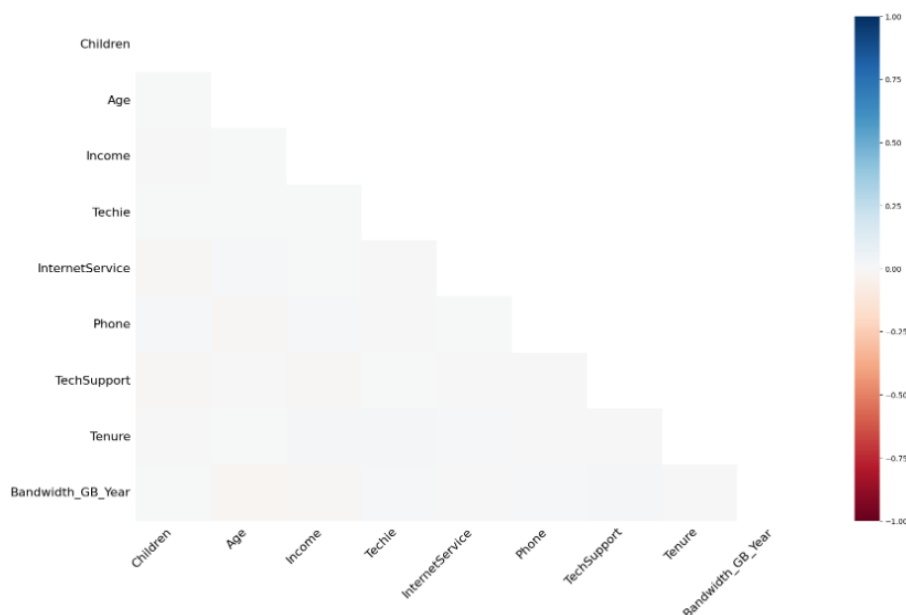
Once I knew which variables contained NaNs, I used the missingno library and obtained `msno.matrix()`, `msno.heatmap()`, and `msno.dendrogram()` to detect what kind of missingness I was dealing with. The heatmap showed no correlation amongst missingness of any pair of variables. The matrix also confirmed this. When I sorted the missing values matrix on a particular variable, such as 'Children', there wasn't a change in the positions of missingness for any other variable. I concluded that the missing values were

missing completely at random (MCAR). I proceeded to impute the missing values using univariate measures, expect in the case of 'Age' and 'Income', which I will detail subsequently.

Missing Values Matrix



Heatmap for missing values



For the variables that are a pandas object type (i.e., strings), I imputed the missing values using the most frequently occurring variable, which was found by calling, for example, `df['TechSupport'].describe()`. Imputing the missing values with the mode best preserved the distribution, with the mode still remaining as the mode. The variables imputed with the mode are:

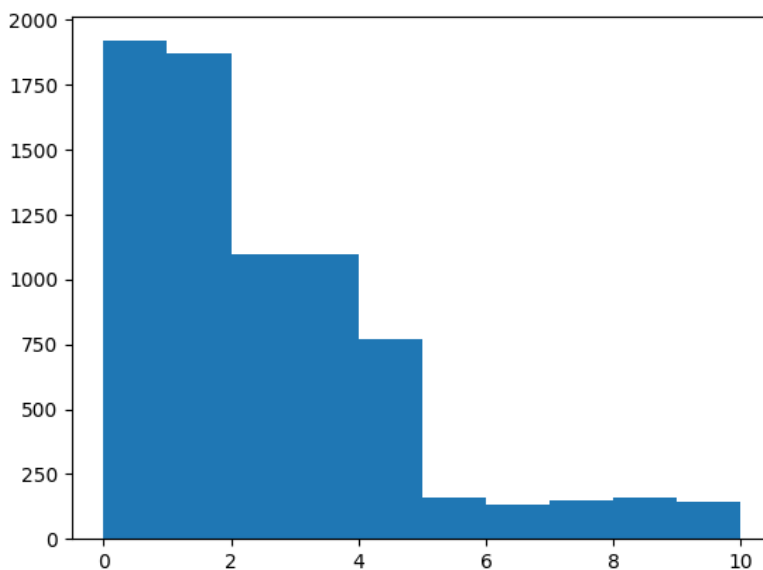
Variable	Mode with missing values	Mode without missing values
Techie	No = 83.3% frequency	No = 84.4% frequency
InternetService	Fiber Optic = 56% frequency	Fiber Optic = 65.4% frequency

Phone	Yes = 90.6% frequency	Yes = 91.5% frequency
TechSupport	No = 62.5% frequency	No = 66.3% frequency

Next, I imputed the missing values for the numeric data types. 'Children' and 'Tenure' were imputed with the median while 'Bandwidth_GB_Year' was imputed the mean. 'Age' and 'Income' were imputed with KNN imputer. To determine which technique to use for imputation I made a histogram to view the distribution of each numeric variable using `plt.hist(df)`. My aim for imputing missing values was to preserve the distribution as best as I could.

For the distribution for 'Children', which is skewed right, I chose to impute with the median. 'Bandwidth_GB_Year' was also imputed with the mean because the distribution is bimodal, non symmetric and the median would preserve this instead of the mean. The distribution for 'Tenure' is bimodal, symmetric, so I chose to impute with the mean.

Histogram of 'Children'

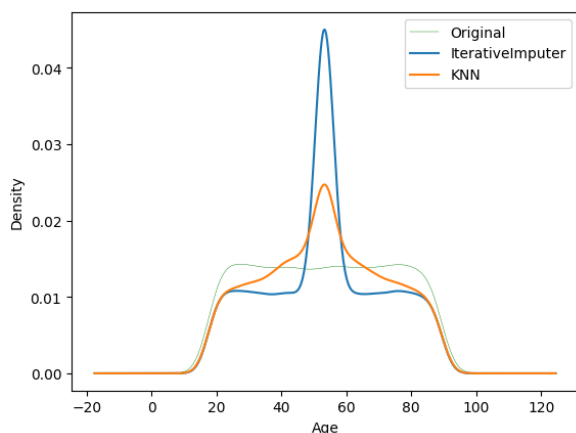


I used more advanced imputation techniques on 'Age' and 'Income' because it made sense that there is a direct relationship amongst these two variables, the older a customer is, the more likely it is they have a higher income. The dendrogram for missing values also lead me to see a relationship between these two variables.

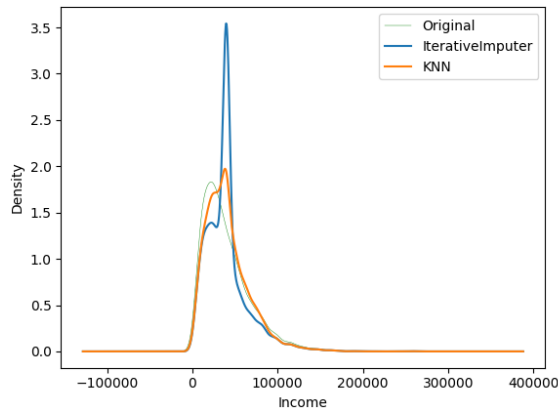
To impute 'Age' and 'Income' I compared KNNImputer and IterativeImputer to see which method better preserves the distributions. After importing these libraries and packages I fit and transformed the imputations, only using 'Age' and 'Income', onto a copy of the original dataset. I also made a density plot comparing the original variable's distribution, IterativeImputer's distribution, and KNN's distribution.

After viewing the density plots, I chose to impute 'Age' and 'Income' using KNN imputation with $n=3$ nearest neighbors. IterativeImputer changed the shape of the distributions too much (Datacamp, n.d.).

Density plots for imputations for 'Age'



Density plots for imputations for 'Income'



Data types. After all variables had no missing values, I addressed data types. I changed the 'CaseOrder', 'Zip', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', and 'item8' variables from int64 to categorical data types. Even though the value entered for these is a number, they are actually ordinal values, except for 'Zip' which is a nominal variable. Next, I changed 'Children' and 'Age' to int64 data types. There weren't any decimal values in these columns and it made sense to use a whole number for the number of children and the age of a customer. At this stage I also rounded all float data types, except latitude and longitude, to 2 decimal places for better readability. I kept latitude and longitude to preserve accuracy for the location values.

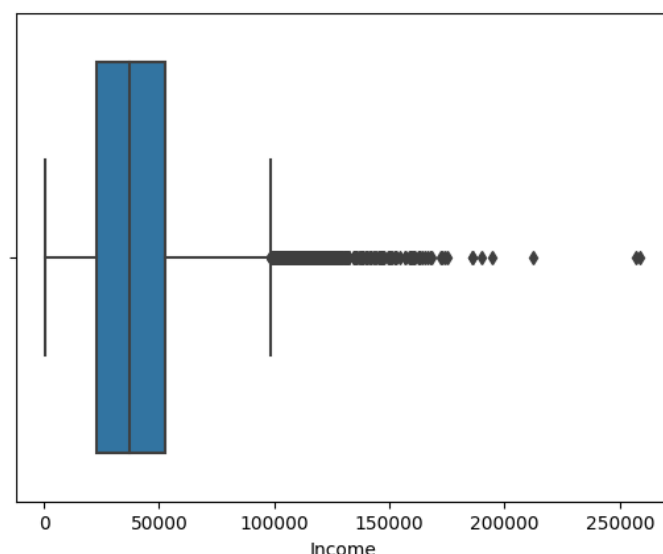
Outliers. Lastly, I assessed the dataset for volume and values of outliers on the numerical data types. These variables are: 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Yearly equip_failure', 'Tenure', 'MonthlyCharge', and 'Bandwidth_GB_Year'. Most of the data is not Normally distributed so I opted to use boxplots rather than z-scores, to determine if a variable has outliers. After viewing the boxplots for these numeric variables, I chose to retain all the outliers.

Variables that did not contain outliers are 'Age', 'Tenure', and 'Bandwidth_GB_Year'. 'MonthlyCharge', 'Yearly equip_failure', and 'Children' had only three to six outliers, none of which was very extreme, so I chose to retain all the outliers for these variables.

Variables that had many outliers are: 'Population', 'Income', and 'Outage_sec_perweek'. I chose to retain outliers for 'Population' and 'Outage_sec_perweek' because I believe these outliers could provide important insight into customer churn. Are the customers who churn the most contained in these outliers? For 'Income', I chose to drop customers with incomes greater than \$200,000 based on how

extreme these values are compared to the overall distribution. Three rows out of 10,000 were dropped. I made a new dataframe to house only customers with incomes greater than or equal to \$200,000. In addition, these extremely high outliers will affect the effectiveness of PCA, another reason to drop these values.

Boxplot for 'Income'



D2

Here I will present my justifications for mitigating the data quality issues that I found when I executed my plan as described in part C1. The executable script file included for part C4 and D4 contains the code and visuals for these findings.

Precleaning. I imported some preliminary libraries including pandas, numpy, and matplotlib, read in the data set. I looked at the head of the dataset and observed that the first column of the file was an index column so I reimported the data without that column. I next realized that the row index was counting at 1, 'CaseOrder', which is serving as a place holder variable used to preserve the original order of the raw data set. I decided to reset the index to start counting at 0 for coding purposes and kept the 'CaseOrder' column while I was cleaning the data.

I next called up the shape of the dataset because I wanted to see how many rows and columns I was working with. I found that there were 50 features about a customer (columns) and 10,000 customers (rows). The 'CaseOrder' does create a 51st column, but that is not a feature of a customer.

Also, by looking at the head of the data, I got a sense of what the value of each variable looked like. I noticed values that would make more sense to round, such as 'Income' and 'MonthlyCharge'. After I

imputed missing values and fixed data types, I rounded all float data types, except latitude and longitude, to 2 decimal places for better readability. I kept latitude and longitude to preserve accuracy for the location values.

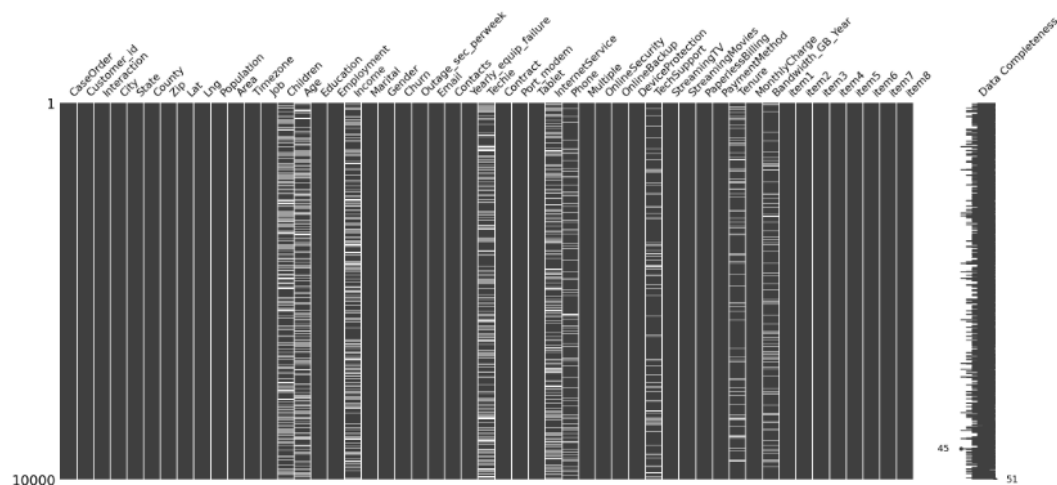
Also, I called `df.info()` and `df.describe()` so I could see what type each variable was, if there were any null values, and to see some summary statistics for each variable.

Duplicates. I sought to detect and treat any duplicate rows via `df.duplicated()` in the data set but there weren't any so I was able to move on to imputing missing values.

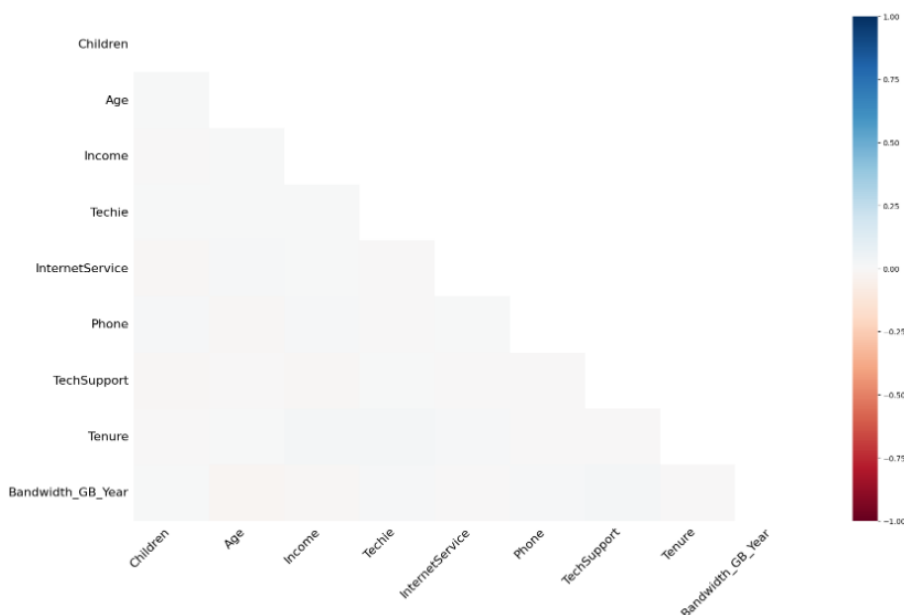
Missing values. Next, I addressed missing values by calling `df.isna().sum()` to obtain a count of null values for each variable. I found that there were nine variables that had missing values: 'Children', 'Age', 'Income', 'Techie', 'InternetService', 'Phone', 'TechSupport', 'Tenure', and 'Bandwidth_GB_Year'.

Once I knew which variables contained NaNs, I used the `missingno` library and obtained `msno.matrix()`, `msno.heatmap()`, and `msno.dendrogram()` to detect what kind of missingness I was dealing with. The heatmap showed no correlation amongst missingness of any pair of variables. The matrix also confirmed this. When I sorted the missing values matrix on a particular variable, such as 'Children', there wasn't a change in the positions of missingness for any other variable. I concluded that the missing values were missing completely at random (MCAR). I proceeded to impute the missing values using univariate measures, except in the case of 'Age' and 'Income', which I will detail subsequently.

Missing Values Matrix



Heatmap for missing values

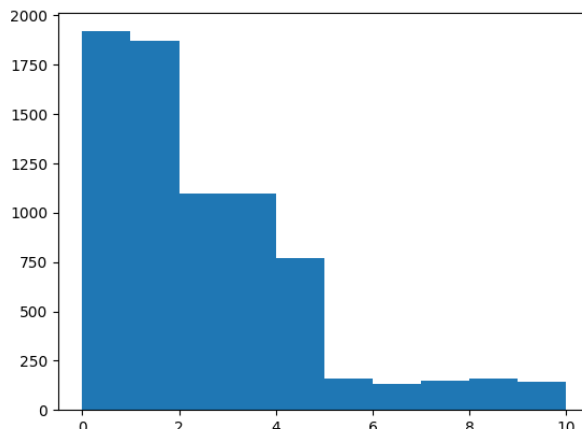


For the variables that are a pandas object type (i.e., strings), I imputed the missing values using the most frequently occurring variable, which was found by calling, for example, `df['TechSupport'].describe()`. Imputing the missing values with the mode best preserved the distribution, with the mode still remaining as the mode. The variables imputed with the mode are:

<i>Variable</i>	<i>Mode with missing values</i>	<i>Mode without missing values</i>
Techie	No = 83.3% frequency	No = 84.4% frequency
InternetService	Fiber Optic = 56% frequency	Fiber Optic = 65.4% frequency
Phone	Yes = 90.6% frequency	Yes = 91.5% frequency
TechSupport	No = 62.5% frequency	No = 66.3% frequency

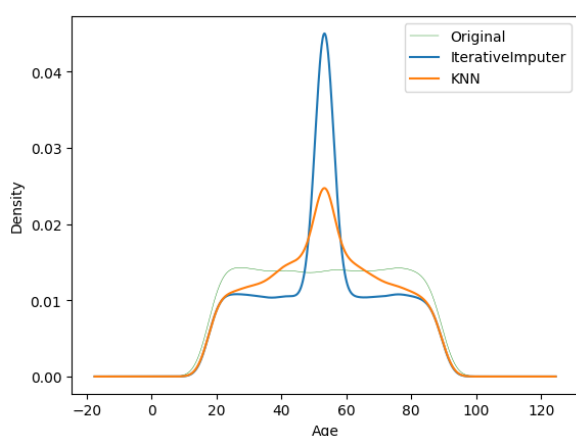
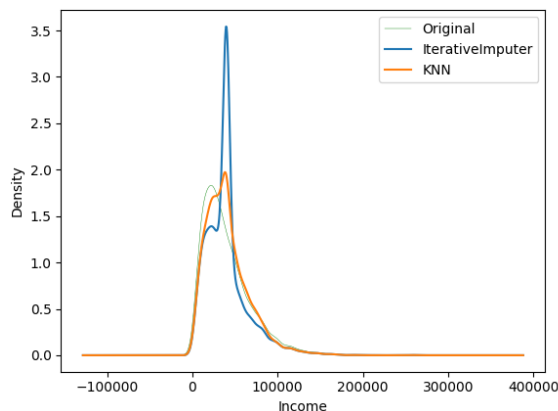
Next, I imputed the missing values for the numeric data types. 'Children' and 'Tenure' were imputed with the median while 'Bandwidth_GB_Year' was imputed the mean. 'Age' and 'Income' were imputed with KNN imputer. To determine which technique to use for imputation I made a histogram to view the distribution of each numeric variable using `plt.hist(df)`. My aim for imputing missing values was to preserve the distribution as best as I could.

For the distribution for 'Children', which is skewed right, I chose to impute with the median. 'Bandwidth_GB_Year' was also imputed with the mean because the distribution is bimodal, non symmetric and the median would preserve this instead of the mean. The distribution for 'Tenure' is bimodal, symmetric, so I chose to impute with the mean.

Histogram of 'Children'

I used more advanced imputation techniques on 'Age' and 'Income' because it made sense that there is a direct relationship amongst these two variables, the older a customer is, the more likely it is they have a higher income. The dendrogram for missing values also lead me to see a relationship between these two variables.

To impute 'Age' and 'Income' I compared KNNImputer and IterativeImputer to see which method better preserves the distributions. After importing these libraries and packages I fit and transformed the imputations, only using 'Age' and 'Income', onto a copy of the original dataset. I also made a density plot comparing the original variable's distribution, IterativeImputer's distribution, and KNN's distribution. After viewing the density plots, I chose to impute 'Age' and 'Income' using KNN imputation with $n=3$ nearest neighbors. IterativeImputer changed the shape of the distributions too much (Datacamp, n.d.).

Density plots for imputations for 'Age'*Density plots for imputations for 'Income'*

Data types. After all variables had no missing values, I addressed data types. I changed the 'CaseOrder', 'Zip', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', and 'item8' variables from int64 to categorical data types. Even though the value entered for these is a number, they are actually

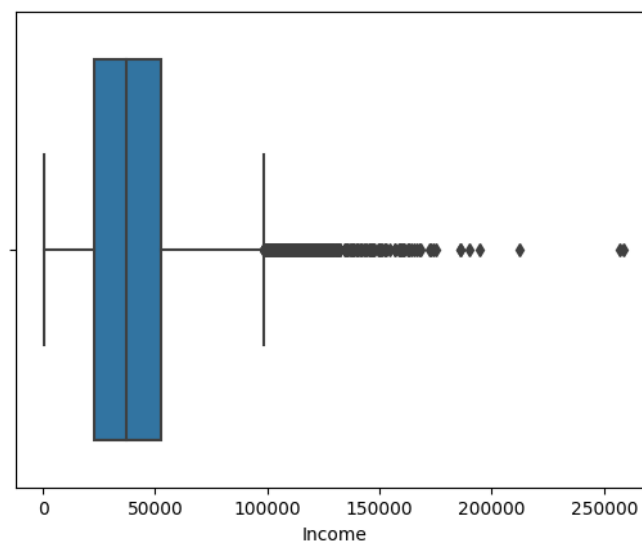
ordinal values, except for 'Zip' which is a nominal variable. Next, I changed 'Children' and 'Age' to int64 data types. There weren't any decimal values in these columns and it made sense to use a whole number for the number of children and the age of a customer. At this stage I also rounded all float data types, except latitude and longitude, to 2 decimal places for better readability. I kept latitude and longitude to preserve accuracy for the location values.

Outliers. Lastly, I assessed the dataset for volume and values of outliers on the numerical data types. These variables are: 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', and 'Bandwidth_GB_Year'. Most of the data is not Normally distributed so I opted to use boxplots rather than z-scores, to determine if a variable has outliers. After viewing the boxplots for these numeric variables, I chose to retain all the outliers.

Variables that did not contain outliers are 'Age', 'Tenure', and 'Bandwidth_GB_Year'. 'MonthlyCharge', 'Yearly_equip_failure', and 'Children' had only three to six outliers, none of which was very extreme, so I chose to retain all the outliers for these variables.

Variables that had many outliers are: 'Population', 'Income', and 'Outage_sec_perweek'. I chose to retain outliers for 'Population' and 'Outage_sec_perweek' because I believe these outliers could provide important insight into customer churn. Are the customers who churn the most contained in these outliers? For 'Income', I chose to drop customers with incomes greater than \$200,000 based on how extreme these values are compared to the overall distribution. Three rows out of 10,000 were dropped. I made a new dataframe to house only customers with incomes greater than or equal to \$200,000. In addition, these extremely high outliers will affect the effectiveness of PCA, another reason to drop these values.

Boxplot for 'Income'



D3

Here is a summary of the outcome from the implementation of each data cleaning step.

Precleaning.

- Imported libraries and packages
- Imported data
- Reset the index for each row so we start counting with 0
- Shape of the dataset is 10,000 rows (customers) with 50 variables (columns)
- Obtained the number of null values for each variable

Duplicates.

- There were no duplicate values

Missing values.

- Nine variables had missing values: 'Children', 'Age', 'Income', 'Techie', 'InternetService', 'Phone', 'TechSupport', 'Tenure', and 'Bandwidth_GB_Year'
- Determined types of missingness with the visualizations of the missingno matrix, heatmap, and dendrogram. All variables were MCAR.
- I imputed the missing values of all categorical variables, 'Techie', 'InternetService', 'Phone', 'TechSupport', with the mode of that variable.
- 'Children' and 'Tenure' were imputed using the median because of their skewed distributions.
- 'Bandwidth_GB_Year' was imputed using the mean because its distribution was roughly symmetric.
- I used KNNImputer to impute the missing values for 'Age' and 'Income' because of the relationship of these variables.

Data types.

- Cast the variables are the correct data types

Outliers.

- Plotted boxplots for each variable with a numeric data type to observe outliers.
- Determined to keep all outliers for all variables except for 'Income'.
- Customers with incomes greater than \$200,000 were dropped.

D4

The annotated code that shows what I used to mitigate data quality issues, including anomalies, is provided in an executable script file with the submission of this performance assessment.

D5

A copy of the cleaned dataset is provided with the submission of this performance assessment.

D6

Here I will summarize the limitations of the data cleaning process.

Imputing missing values poses limitations in the data cleaning process because there is no way to know what these missing values are for sure. We do our best to preserve the shape of the distribution of each variable and use advanced imputation techniques when possible to obtain missing values. But we are still limited because we simply don't know what those values were.

D7

Here I will discuss how the limitations summarized in D6 could affect the analysis of the research question from A.

For categorical variables, the imputation of the mode category will exaggerate the frequency of that value. Also, imputation of the univariate statistics and KNN Imputer on numerical variables, changes the shape of the distributions by imputing another mode in the middle of the dataset, when one isn't there originally.

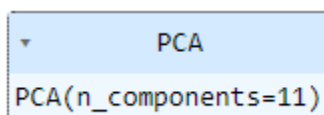
Furthermore, outliers will greatly affect PCA and machine learning techniques and skew results towards the outliers.

E1

I applied principal component analysis (PCA) to identify the significant features of the data set (Middleton, n.d., Webinar 4).

First, identifying the total number of principal components (PC). I imported the PCA package from sklearn, created a new dataframe containing numeric data only, and normalized the data. Then I applied PCA and found that there are 11 principal components.

Output for number of PCs



PCA

PCA(n_components=11)

Next, I obtained the PCA loadings for the 11 PCs.

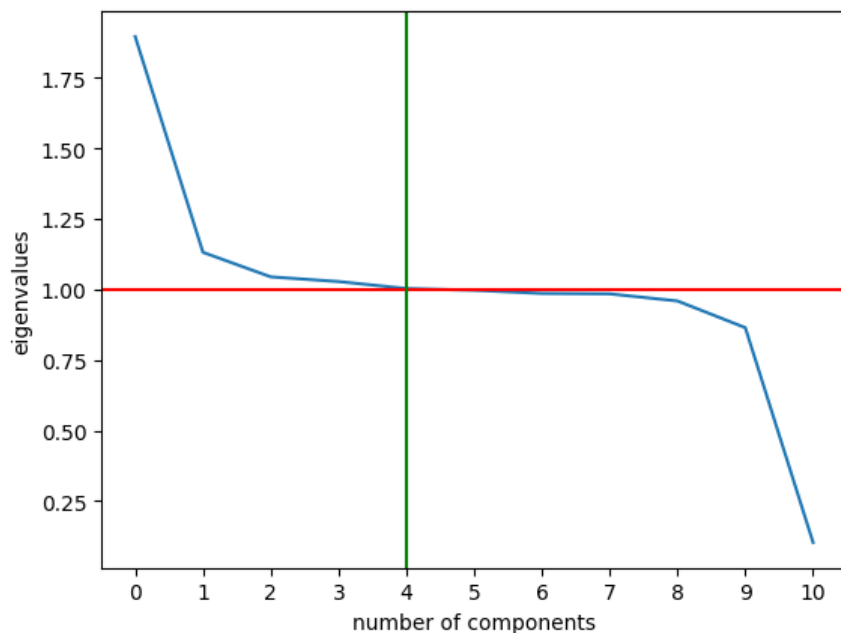
PC loadings

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
Population	-0.000199	-0.056723	-0.343169	-0.293078	0.223697	0.629945	0.504105	-0.058497	0.297837	0.000965	-0.000850
Children	-0.002693	0.023920	0.559683	-0.119431	0.393495	0.125996	-0.287683	0.370870	0.529537	0.011654	-0.018623
Age	-0.001220	-0.027167	-0.344716	0.551800	0.053796	0.237854	-0.493475	-0.344666	0.364270	0.145105	0.020470
Income	0.004545	0.029488	0.111829	0.488311	0.703628	0.073413	0.222148	0.018873	-0.440865	-0.060033	0.001280
Outage_sec_perweek	0.022377	0.706629	0.020424	-0.036120	-0.010801	0.036612	0.067607	0.025171	-0.058491	0.698835	0.000421
Email	-0.020471	0.053444	-0.335694	-0.480941	0.286501	0.177305	-0.592627	0.117098	-0.413828	-0.054818	0.005780
Contacts	0.005181	-0.005274	-0.440413	0.285407	-0.113126	-0.068696	0.054470	0.835727	0.074583	0.005513	-0.002880
Yearly equip_failure	0.015424	0.059561	0.348466	0.202423	-0.449424	0.696033	-0.105625	0.146936	-0.314526	-0.125251	-0.002323
Tenure	0.705048	-0.057973	-0.013850	-0.006311	0.006172	-0.000008	-0.016631	-0.009461	-0.013201	0.037488	-0.705216
MonthlyCharge	0.045984	0.696700	-0.094489	0.038692	0.004215	-0.057720	0.007160	-0.073156	0.156423	-0.683029	-0.048177
Bandwidth_GB_Year	0.706808	-0.009744	0.005940	-0.018242	0.009976	-0.006351	-0.004509	0.008139	0.004740	-0.013296	0.706777

E2

After generating the loadings for the PCs, I found the covariance matrix and eigenvalues for the PCs. I selected PCs that have eigenvalues greater than one, which we can see in the scree plot displayed below. The vertical line plotted at $x=4$ helps to see where the number of components have eigenvalues greater than one.

Scree plot of number of components and eigenvalues



From the scree plot I determined that there are four PCs (0 through 3) that have eigenvalues greater than one. So, I will use PC1, PC2, PC3, and PC4.

E3

The organization would benefit from Principal Component Analysis because there would be less variables to analyze and because PCA finds the most important related groupings of variables (WGU, 2020). Our PCA indicated the following four related groupings.

In PC1, 'Tenure' and 'Bandwidth_GB_Year' have the highest correlation coefficient, both about $r = 0.71$. So, PC1 relates how long a customer has been with the company and the average amount of data used by a customer over the course of a year.

For PC2, 'Outage_sec_perweek' has $r = 0.71$ and 'MonthlyCharge' has $r = 0.70$. So PC2 relates the number of seconds a customer experiences outages per week, and a customers monthly charge.

PC3 has 'Children' with $r = 0.56$, 'Age' with $r = -0.34$, 'Email' with $r = -0.34$, 'Contacts' with $r = -0.44$, and 'Yearly equip_failure' $r = 0.35$. We have a relationship with number of children, age of customer, number of times a customer has been contacted, and the number of yearly equipment failures.

PC 4 has 'Age' with $r = 0.55$, 'Income' with $r = 0.49$, and 'Email' with $r = -0.48$. There is a relationship amongst the age and income of the customer and the number of times they have been contacted via email.

PART IV: SUPPORTING DOCUMENTS

F

A Panopto video recording including a demonstration of the functionality of my code used for my analysis is provided with the submission of my Performance Assessment.

G

Datacamp. (n.d.). Dealing with Missing Data in Python. Datacamp.

<https://app.datacamp.com/learn/courses/dealing-with-missing-data-in-python>

Middleton, K. (n.d.). Webinar 2: Getting Started with Missing Values and Outliers [Webinar]. D206,

WGU. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=d3af9533-0b7f-4d42-9db2-af48002f4799>

Middleton, K. (n.d.). Webinar 3: Getting Started with Re-expression of Categorical Variables [Webinar].

D206, WGU. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=bfc98490-7757-4dbb-8794-af56013265ab>

Middleton, K. (n.d.). Webinar 4: Getting Started with PCA [Webinar]. D206, WGU.

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7b31791b-24e8-4077-ba1a-af5d0005144c>

Western Governors University. (2023). R or Python. [https://www.wgu.edu/online-it-](https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html)

[degrees/programming-languages/r-or-python.html](https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html)

Western Governors University. (2020). Data Cleaning. Lesson 7: Principal Component Analysis.

https://cgp-oex.wgu.edu/courses/course-v1:WGUx+OEX0026+v02/courseware/1f468770545f494fa657b4dc0ed3762f/2b5f23c5dad64357b352728993788677/1?activate_block_id=block-v1%3AWGUx%2BOEX0026%2Bv02%2Btype%40vertical%2Bblock%4063c14837d7e3469ca774872cd2fe0f03

H

Larose, C. D., & Larose D.T. (2019). Data Science Using Python and R (1st ed.). Wiley.



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