What personal data factors influence the churn rate in businesses?

By: Kevin Sandoval

Part 1: Research Question and Variables

A.

**1.** My research question will be, “What personal data factors influence the churn rate in companies?” This broad question will allow me to look at various variables in the churn data set and thus see what will impact the churn rate most significantly.

**2.** List of all variables, data type, description, and an example of the data.

1. **Name:** CaseOrder

**Type:** **Type:** Qualitative

**Description:** Keeps the initial order of the data

**EX:** Row 115 would have integer value “114” since the first row is all of the columns so the first instance is in the 2nd row of the spreadsheet

1. **Name:** Customer\_id

**Type:** Qualitative

**Description:** ID for each customer that is unique

**EX:** G106476, Z281810, K591268

1. **Name:** Interaction

**Type:** Qualitative

**Description:** Unique IDs that correspond to a customers information including sign-ups, tech support, and transactions

**EX:** 69fbc82f-1ea2-47a2-a056-d6f8d428b687

1. **Name:** City

**Type:** Qualitative

**Description:** The city where the customer lives per their billing statement

**EX:** San Diego, Columbus, Peoria

1. **Name:** State

**Type:** Qualitative

**Description:** The state where the customer lives per their billing statement

**EX:** CA, OR, NC

1. **Name:** County

**Type:** Qualitative

**Description:** The county where the customer lives per their billing statement

**EX:** Palo Alto, Washington, Orange

1. **Name:** Zip

**Type:** Qualitative

**Description:** The zip code where the customer lives per their billing statement

**EX:** 92014, 45176, 54558

1. **Name\*:** Lat

**Type:** Qualitative

**Description:** Latitude GPS coordinate of the customers address

**EX:** 34.49127, 44.48072, 40.7955

1. **Name\*:** Long

**Type:** Qualitative

**Description:** Latitude GPS coordinate of the customers address

**EX:** -120.082, -105.685, -73.9297

**\*NOTE:** The excel sheet has latitude and longitude in two separate columns, but the data dictionary lists them together. This is why I will end up having 51 variables as opposed to the 50 listed on the data dictionary.

1. **Name:** Population

**Type:** Quantitative

**Description:** The number of people living within one mile of the customer

**EX:** 20193, 381, 39649

1. **Name:** Area

**Type:** Qualitative

**Description:** The type of area that a customer lives in

**EX:** Rural, Urban, Suburban

1. **Name:** TimeZone

**Type:** Qualitative

**Description:** The timezone that the customer resides in

**EX:** America/Detroit, America/New\_York, Pacific/Honolulu

1. **Name:** Job

**Type:** Qualitative

**Description:** The occupation of the signed-up customer

**EX:** Surgeon, Materials engineer, Office manager

1. **Name:** Children

**Type:** Quantitative

**Description:** Children living with the signed-up customer

**EX:** 0, 7, 3

1. **Name:** Age

**Type:** Quantitative

**Description:** How old the signed-up customer is

**EX:** 23, 68, 49

1. **Name:** Education

**Type:** Qualitative

**Description:** Highest level of education achieved by the customer

**EX:** Master’s Degree, Regular High School Diploma, Bachelor’s Degree

1. **Name:** Employment

**Type:** Qualitative

**Description:** Current employment information for the customer

**EX:** Retired, Student, Full Time, Part Time, Unemployed

1. **Name:** Income

**Type:** Quantitative

**Description:** Current annual income for the customer

**EX:** 21704.77, 89061.45, 12558.83

1. **Name:** Marital

**Type:** Qualitative

**Description:** Current marital status for the customer

**EX:** Widowed, Married, Separated, Divorced, Never Married

1. **Name:** Gender

**Type:** Qualitative

**Description:** Stated gender of the customer

**EX:** Male, Female, Prefer not to answer

1. **Name:** Churn

**Type:** Qualitative

**Description:** If the customer cancelled their service within the previous month

**EX:** Yes, No

1. **Name:** Outage\_sec\_perweek

**Type:** Quantitative

**Description:** Average amount of time there were outages in the customer's neighborhood in a week in seconds

**EX:** 10.24561565, 6.520830558, 14.42916512

1. **Name:** Email

**Type:** Quantitative

**Description:** How many emails the customer received via marketing or correspondence in the previous year

**EX:** 9, 17, 13

1. **Name:** Contacts

**Type:** Quantitative

**Description:** How many times the customer contacted tech support

**EX:** 0, 2, 4

1. **Name:** Yearly\_equip\_failure

**Type:** Quantitative

**Description:** How many times the customer’s equipment failed in the past year and needed to be replaced or reset

**EX:** 0, 1, 2

1. **Name:** Techie

**Type:** Qualitative

**Description:** Question on if the customer considers themselves technically inclined

**EX:** Yes, No

1. **Name:** Contract

**Type:** Qualitative

**Description:** How long the customer is on the contract

**EX:** One year, Month-to-month, Two Year

1. **Name:** Port\_modem

**Type:** Qualitative

**Description:** If the customer has a portable modem

**EX:** Yes, No

1. **Name:** Tablet

**Type:** Qualitative

**Description:** If the customer owns their own iPad, surface, or any other type of tablet

**EX:** Yes, No

1. **Name:** InternetService

**Type:** Qualitative

**Description:** What the customer’s internet provider is

**EX:** DSL, None, Fiber Optic

1. **Name:** Phone

**Type:** Qualitative

**Description:** If the customer has phone service

**EX:** Yes, No

1. **Name:** Multiple

**Type:** Qualitative

**Description:** If the customer has multiple lines on their plan

**EX:** Yes, No

1. **Name:** OnlineSecurity

**Type:** Qualitative

**Description:** If the customer has an add-on for online security

**EX:** Yes, No

1. **Name:** OnlineBackup

**Type:** Qualitative

**Description:** If the customer has an add-on for online backup

**EX:** Yes, No

1. **Name:** DeviceProtection

**Type:** Qualitative

**Description:** If the customer has an add-on for device protection

**EX:** Yes, No

1. **Name:** TechSupport

**Type:** Qualitative

**Description:** If the customer has an add-on for tech support

**EX:** Yes, No

1. **Name:** StreamingTV

**Type:** Qualitative

**Description:** If the customer has a TV with streaming

**EX:** Yes, No

1. **Name:** StreamingMovies

**Type:** Qualitative

**Description:** If the customer has movies for streaming

**EX:** Yes, No

1. **Name:** PaperlessBilling

**Type:** Qualitative

**Description:** If the customer has decided to do paperless billing

**EX:** Yes, No

1. **Name:** PaymentMethod

**Type:** Qualitative

**Description:** How the customer has decided to pay

**EX:** Mailed Check, Bank Transfer(automatic), Credit Card(automatic)

1. **Name:** Tenure

**Type:** Quantitative

**Description:** How many months the customer has been with the provider

**EX:** 13.23677, 1.670972, 19.26726

1. **Name:** MonthlyCharge

**Type:** Quantitative

**Description:** How much is charged to the customer on a monthly basis

**EX:** 171.4498, 203.0696, 163.3124

1. **Name:** Bandwidth\_GB\_Year

**Type:** Quantitative

**Description:** How much data in GB is used by the customer per year on average

**EX:** 1948.694, 594.1054, 870.764

1. **Name:** Item1

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to timely responses

**EX:** 1, 4, 6

1. **Name:** Item2

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to timely fixes

**EX:** 5, 2, 6

1. **Name:** Item3

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to timely replacements

**EX:** 2, 3, 5

1. **Name:** Item4

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to reliability

**EX:** 3, 4, 7

1. **Name:** Item5

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to options provided

**EX:** 1, 2, 5

1. **Name:** Item6

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to respectful responses

**EX:** 3, 4, 5

1. **Name:** Item7

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to courteous exchanges

**EX:** 1, 2, 7

1. **Name:** Item8

**Type:** Qualitative

**Description:** From 1-8 with 1 being the most important and 8 being the least important, the customer’s response to evidence of active listening

**EX:** 3, 5, 7

Part 2: Data-Cleaning Plan (Detection)

B1.

Duplicates:

In order to find potential duplicate values in the data, I used the .shape function to determine the shape of the data frame and then used the .duplicated function that would output the number of False instances (no duplicates) and True instances (duplicates). For a sanity check, I also checked how many unique customer IDs there were by using the .nunique() function.

Missing Values:

In order to find any missing values, I used the .isna().sum() functions on my saved data frame and checked which columns had instances in them. I also used the missingno package and the msno.matrix function.

Outliers:

In order to check for outliers, I found every quantitative variable first and then made a note of them for easy reference later. Then for each variable I plotted a histogram using the plt.hist() and plt.show() functions to look at what the general distribution of the data looked like. For distributions that appeared normal, I used boxplots and z-scores to detect any outliers in the data. To find the z-scores I used the stats.zscore function, and then plotted those on a histogram as well as used the .query() and .info() functions to find the number of outliers. For distributions that were not normal I used boxplots with seaborn’s .boxplot() function to graph and I also calculated the interquartile range by using the .percentile() function and some basic arithmetic functions to find an exact range of non-outlier values. After I had the range of non-outliers, I used a comparison and the .count() function to determine the number of outliers that I had.

Re-Expression of Categorical Variables:

In order to determine whether or not to re-express some categorical variables, I looked through the raw data file of the churn data set. I looked for any ordinal variables that could have a dictionary assigned to them easily, and I also looked for how survey responses were recorded. In order to re-express my data I used the .unique() function to find the unique values, dictionary creation, and the .replace() function to replace the values as needed for the ordinal categorical variables.

B2.

Duplicates:

I used the .shape() function to determine the shape because it would show me how many columns/variables I had, while also showing me how many rows of data I would have as well. I used the .duplicated() function because it easily outputs how many rows it detects might be a duplicate value. And since I know the total number of rows from the .shape() function, it makes it easy to check everything is accounted for. I decided to use the .nunique() function just as a check that every row has a unique ID tied to it.

Missing Values:

I used the .isna().sum() functions because they easily pair together to output the count of missing values for each variable and displays it clearly. I decided to use the msno.matrix() package because it was an easy double check to see the missing values visually, and helped ensure that I did not miss anything in the .isna().sum() function.

Outliers:

I used the plt.hist and plt.show functions to create a graph of the histogram because histograms are a great way to look at the general distribution of the variables. It’s important to know if a quantitative variable is normal, bi-modal, or skewed because that will affect how we attempt to treat the data later. I used seaborn’s .boxplot() functions because they show a clear visual of outliers as every outlier is clearly depicted after the edge of the “whisker.” I used the .percentile() function because it was a great way to find the 25th and 75th percentiles of every distribution. After that I used some arithmetic so I could find the max value of the “whisker” before a value would become an outlier. I wanted to do this because it created an easy way for me to find the exact amount of outliers so I could keep data integrity. For distributions that were normal, I used the stats.zscore() function because it easily let me look for any values that were more than 3 standard deviations away from the mean. I then used the plt.hist() function again on the Z-scores to visually check for any values that might fall out of range and I also used the .info() function paired with the .query() function to show how many entries met the criteria of 3 standard deviations away from the mean or more.

Re-Expression of Categorical Variables:

In order to re-express categorical variables I used two techniques. The first ordinal encoding for variables that could be expressed in a natural order. I used the guide presented by Dr. Middleton in her lecture “D206 – Getting Started With D206 | Re-expression of Categorical Variables.” From that lecture I used the .unique() function to find all the values for the ordinal variable. I did this so I could easily make a dictionary with set values and it would replace every instance in the data set because everything would be accounted for. I used the .replace() function with inplace = True to do the actual replacing after I created the dictionary. I used this because it made the transition easy and it flowed through the file. Since Item1 through Item8 all used a number scale to show data, I felt that it would be necessary to convert all Yes/No responses into a 1/0 interpretation. To do this I used a simple .replace() function with Yes being 1 and No being 0. I ran that over the whole data frame so it would replace everything easily. I also preformed a check after changing these values with the same functions used in the Duplicate and Missing Values sections just to make sure everything remained consistent.

B3.

I chose to use Python for this course. I have some coding experience in both Python and R but I find Python easier to use as the debugging and general coding felt easier. Python is more robust than R and although R has better packages for data science specific topics, I thought Python’s power and libraries were stronger and easier to use. Python made a lot of the mathematical computation I did very easy and the code cells are easy to identify and break up into individual sections for bug testing. The on caveat with Python is that the graphs and displays don’t show up very cleanly, and it tends to clutter the notebook and make it hard to scroll through. Despite that flaw, I still felt as though Python’s visualizations were more than enough to get the job done, and the ease of use with bug testing, computational power, and flexibility made it the choice for me.

Part 3: Data Cleaning (Treatment)

C1.

Duplicates:

When I used .shape() and the .duplicate() functions to determine any duplicates, the output was (10000, 52) for the shape and False 10000 for the duplicates. This showed that my original data set had no duplicates. I was slightly skeptical of this discovery and since there was a column dedicated to unique customer IDs, I though a good way to double check my finding was to see if there were 10000 unique IDs. I used the .nunique() function to output the number of unique instances and it still showed 10,000. Since these two techniques both showed no duplicates, I did not treat any.

Missing Values:

After using the .isna().sum() functions paired together and the msno.matrix visualization, I found 8 columns with missing values. Below are the columns listed with how many missing values I detected.

**Children:** 2,495   
**Age:** 2,475   
**Income:** 2,490   
**Tenure:** 931  
**Techie:** 2,477   
**Phone:** 1,026   
**TechSupport:** 991  
**Bandwidth\_GB\_Year:** 1,021

Outliers:

Of the 11 quantitative variables in the data set, I detected outliers in 8 of them. Below is a list of each that had outliers, as well as how many and the range of the outliers present.

**Population:** Contained 937 outliers with a range of 31,814 to 111,850 **Children:** Contained 451 outliers with a range of 7 to 10  **Income:** Contained 759 outliers with a range of $78,270 to $258,900.7 **Outage\_sec\_perweek:** Contained 539 outliers with a range of -1.348571s to 1.404438s AND 19.137567s to 47.04928s **Email:** Contained 38 outliers with a range of 1 to 3 emails AND 21 to 23 emails **Contacts:** Contained 8 outliers with a range of 6 to 7 contacts **Yearly\_equip\_failure:** Contained 94 outliers with a range of 3 to 6 yearly failures **MonthlyCharge:** Contained 5 outliers with a range of $298.84 to $315.88

As a note for Email and MonthlyCharge, I initially checked the Z-score values to look for outliers. After I determined the number of values that were more than 3 standard deviations away from the mean, I decided to check the boxplots as well. The boxplots showed more outliers than the Z-scores, so I decided to use the boxplot information to treat the outliers as I felt that was a better interpretation of the data.

C2.

Duplicates:

I found that my data set had no duplicate values and so I did not use any methods to treat them.

Missing Values:

**Children:** For this column I saw that the distribution was skewed to the right. Because of that I decided that I was going to impute the missing values with the median of the data set. This would ensure that the general distribution of the data would remain the same, and it made it seamless to add integers to the column of values as there can’t be partial children. I decided to impute because a quarter of the data in the column was missing and if I were to remove those rows the data set would start to shrink drastically.

**Age:** For the “Age” column I saw that the distribution was uniform. Through Dr. Middleton’s video lectures I knew that I would impute the missing value with the mean. One thing that I had to consider was the mean age was a float. Since every other data point in the column was an integer, I used the .astype(‘int64’) function when imputing the data to ensure that the imputed data was rounded to the nearest integer. Similarly to the Children data set, I decided to impute because a quarter of the data in the “Age” column was missing and the data set would have shrunk. I also checked the distribution after imputing the missing data and the distribution did not change the general shape, albeit with a huge jump at the mean which was to be expected.

**Income:** The distribution for the “Income” column was skewed to the right so I decided to impute the missing data with the median income. Very similar to the Children column, the distribution remained the same after imputing the missing data. Again, almost a quarter of the data would have been missing so imputing helped retain valuable information in my data set.

**Tenure:** The distribution looked to be bi-modal, so I decided to impute the missing values with the median. Since there were less missing values than the previous variables I checked, the imputation had less of an impact on the distribution. The distribution still looked generally bi-modal so I think it was worth keeping the data in.

**Bandwidth\_GB\_Year:** The distribution looked to also be bi-modal. Again, I decided that I would impute the missing values with the median and the distribution stayed the same.

**Techie, Phone, TechSupport:** These are all the qualitative/categorical variables that had missing data. I looked at the bar graph of each Yes/No response to gauge the rough distribution of answers to these 3 questions. “Techie” and “TechSupport” both had an overwhelming majority of response in “No” whereas “Phone” had an overwhelming majority in “Yes.” Because of the distribution of these responses, I decided that I would impute the mode of response for the respective missing values. I did this to preserve the integrity of the data as well as to not delete too many of the instances.

Outliers:

**Population:** For “Population” I chose to RETAIN the outliers in the data set. I found that almost 10% of the data would be considered an outlier and I felt like that would be too much of the data set to omit with just one parameter. I also considered what the “Population” variable represents, which is the surrounding population count for a customers 1 mile radius around them. This led to anyone in a major city, such as Chicago, would be seen as an outlier. I felt as though removing these outliers would not be representative of the true nature of the customer data so I decided to RETAIN.

**Children:** I chose to EXCLUDE the initially found outliers for the “Children” column variable. I chose to exclude because I felt as though the outliers in number of children did not accurately represent the general data of the customers. There were less than 5% of the data that met the criteria as an outlier, so excluding was the best option. I excluded instead of deleted because I feel like keeping track of the outliers is always important, and it allows for flexibility later on if we wanted to go back and look closer at the data. After I excluded the initial outliers, I replotted the boxplot to look for any more. I found that a new range of outliers had appeared from 4 to 6 children. There were 1,065 counts of outliers that met this criteria. I decided to RETAIN these new outliers. Excluding these would have reduced the total data set by 15% and I felt as though it was important to keep these data points. To do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the upper\_limit of the respective variable I found previously with the box plots.

**Income:** I decided to EXCLUDE the outliers in the “Income” column variable. Although there were a significant amount of outliers that I detected (around 7.5% of the data), I concluded that these outliers would not be indicative of the general population and so I chose to exclude them. After excluding the values I re-checked the boxplot and found additional outliers. I decided to drop them as well and re-check. I completed this process several times and realized that new outliers were consistently showing up whenever I edited the criteria. Because of this, I decided that I would RETAIN the outliers after the initial exclusion. This left the salary range from $0 to $78,269. To do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the upper\_limit of the respective variable I found previously with the box plots.

**Outage\_sec\_perweek:** I decided to EXCLUDE the outliers from the “Outage\_sec\_perweek” column variable. I decided to exclude these outliers as the distribution of them was heavily skewed towards longer outages. Interestingly, I noticed that there were a significant amount of outliers that were grouped towards the right of the boxplot, which is another contributing factor to why I decided to exclude and store the outliers to a variable. The variable allows anyone to reference the outliers in case we wanted to examine them further. As a note for the data, there were values that were less than 0. These obviously do not make sense contextually as the variable is represented in total amount of seconds in an outage. I decided that I did not want these to be included in my outlier variable, as they are erroneous data points and so I used the .drop() function on them before I created the variable for the outliers so they would not be included. For this column there were outliers on both ends of the boxplot, so I needed to include an additional line of code to exclude the outliers on the lower side and upper side. I used the .append() function to combine the lower outliers to the dataframe of the upper outliers. Python did throw me a warning that .append() is deprecated but for now I decided to keep the code in as it is working as intended. Similarly to the “Income” variable, outliers kept showing up when re-doing the boxplots, so I decided to RETAIN the outliers after the initial outlier exclusion. To do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the lower \_limit and upper\_limit of the respective variable I found previously with the box plots.

**Email:** I decided to EXCLUDE the outliers for the “Email” column variable. Although contextually the amount of emails received might make sense or seem reasonable, only 0.38% of the data sampled classified as such so it was a logical choice to remove these from the data. Similar to the other excluded outliers, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the lower \_limit and upper\_limit of the respective variable I found previously with the box plots. I used the .append() function to combine the lower outliers to the dataframe of the upper outliers. Python did throw me a warning that .append() is deprecated but for now I decided to keep the code in as it is working as intended. There were no outliers after excluding the initial ones so I felt comfortable finishing there. To do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the lower \_limit and upper\_limit of the respective variable I found previously with the box plots.

**Contacts:** I chose to EXCLUDE the outliers for the “Contacts” column variable. There were very few instances of outliers (0.08%) so the exclusion for these variables was obvious. They would impact the data in a way that was not meaningful to the rest of the population. To do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the lower \_limit and upper\_limit of the respective variable I found previously with the box plots. No outliers were present in the re-graph of the boxplot of the data.

**Yearly\_equip\_failure:** I chose to EXCLUDE the outliers for the “Yearly\_equip\_failure” column variable. The process was almost identical to the “Contacts” variable. There was only 0.94% of the data that were classified as outliers, so the cost of excluding them was worth the affect they would have on the rest of the data. Again, to do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the lower \_limit and upper\_limit of the respective variable I found previously with the box plots. No outliers were present in the re-graph of the boxplot of the data.

**MonthlyCharge:** I chose to EXCLUDE the outliers for the “MonthlyCharge” column variable. The process was almost identical to the “Contacts” variable. There was only 0.05% of the data that were classified as outliers, so the cost of excluding them was worth the affect they would have on the rest of the data. Again, to do these edits, I created a variable to store the outliers respective to the column variable and then used the .drop() function with the parameter checking the lower \_limit and upper\_limit of the respective variable I found previously with the box plots. No outliers were present in the re-graph of the boxplot of the data.

**Re-expression of Categorical Variables:** An obvious variable that I chose to use ordinal encoding on was the “Education” column variable. I used the .unique() function to see the names of every education type as well as the .nunique() function to get the count. I then created a new column name and mapped a dictionary that attributed an integer value based on the level of education achieved. Afterwards I used the .replaced() function to edit the values as needed. Since Item1-Item8 all showed integer value representation for a survey response, I decided that the Yes/No answers also needed to be re-expressed to 1/0 respectively. To do this, I ran churned\_data = churned\_data.replace({“Yes”:1 , “No”:0}) to easily replace every instance of where they appeared in the data set. I re-checked the shape, duplicates, and missing values before using the .to\_csv() command to create my new data set.

C3.

In summary, I first used the .shape() and .duplicate() functions to test for any duplicate value in the data set. I found that I had none so I did not need to treat any duplicate values. After confirming no duplicates, I moved on to the missing values stage of the cleaning process. I used the .isna() function paired with the .sum() function to easily check how many missing values I had. I used the .fillna() command to fill the missing values as needed per each variable. Afterwards I checked for any missing values to confirm that everything had been edited. I then moved on to the outlier detection. I primarily used boxplots and IQR arithmetic to figure out what outliers I had as well as the range of values. I used the .boxplot() function for the visuals and the .info() function with a parameter check of the “whisker” range to get the exact value of the range. I then used the .drop() function to drop the outliers from the desired data set and re-used the .boxplot() function to check for any additional outliers that might have been created. Now that the data has been treated, I find that there are no duplicates and no missing values. There are still a few outliers for the variables “Income” and “Outage\_sec\_perweek”, but those came with depreciating value and so I chose to keep the outliers in.

A computer screen with text

Description automatically generated

From this screenshot we can see that there are no duplicate values. There are a few more columns because of Z-score addition and re-expression of variables.

 A screenshot of a computer screen

Description automatically generated

As you can see from the screenshot, there are no more missing values.

C4.

Duplicates:

For the duplicate checking of the data I used the .duplicate() and .nunique() functions. These output directly showed if there were any duplicate values detected. I think a limiation of using these methods would be assuming the code is working as intended and that it checks everything properly. Obviously I’m not going to check each row of 10,000 but I am putting faith in the code that it will detect any issues. That is partly why I wanted to use the .nunique() function on the customer IDs, as if that also outputted 10,000, then I felt a lot more comfortable assuming that there were in fact no duplicates detected. Even that function can come with some limitations as I’m only checking 1 column and perhaps someone received 2 “unique” IDs but the rest of their data was the same.

Missing Values:

I used the .isna() function with the .sum() function to count the instances of missing values in the data. I then used the .fillna() function to fix any missing values. One limitation of this method is data that shows up as “NONE” or “None.” One column did have some valid repsonses that were “None” and those should NOT be included in the missing values. I made sure to check for only values that were listed as NA.

Outliers:

A common method I used to treat outliers was to exclude them from the data set. This comes with some obvious limitations such as shrinking the data set, excluding some potentially interesting or unique values, and potentially changing the distribution of the data. I also chose to retain the data for the Population variable. This comes with some limitations as the data might be skewed towards more populated, more dense cities. It also could still potentially contain erroneous values.

Re-expression of Categorical Variables:

A limitation of changing both the Education column and all of the Yes/No columns is affecting the readability of the data set. When perusing the data set, you have to find for yourself that a “7” in the Education Numeric column is equivalent to a “Associate’s Degree” in the Education column. Similarly for the Yes/No columns, you must know what type of question the column is referring to as well as know that Yes equates to 1 and No to 0. It also could affect testing statistical modeling, as these numbers might seem like they are significant and worth testing on, but they are actually just placeholder values to help understand what the data is showing.

C5.

One challenge a data analyst might have when using my data set are deciphering the different columns. I created a few more for the Z-score testing as well as the re-expression of variables and it is imperative that they know what those variables represent. I also think that since I imputed missing values with the mean, median, or mode as needed, the data is slightly skewed towards each of those respective values and might lead to some slightly different results on testing the data. Since my question is to look at churn rate, the Churn column was edited from a Yes/No response to a 1/0 response. This might also affect how a data analyst would perceive the data. The editing and imputing of data could skew some results when specifically looking at the churn rate.

D1.

Uploaded.

D2.

Uploaded.

Part 4: PCA

E1.

The variables I used for the PCA are as follows:

**Income  
Outage\_sec\_perweek  
Tenure  
MonthlyCharge  
Bandwidth\_GB\_Year**

A screenshot of the loadings matrix is below:

A table with numbers and symbols

Description automatically generated

E2.

Using Dr. Middleton’s video lecture on PCA, I decided to use the eigenvalues as the determing factor for the PCs. I created a data frame with the 5 variables I would be using and then normalized them using the .mean() and .std() functions. I then used the pca.fit() function to check for the components. I continued with transforming back to a dataframe and named the columns PC1-PC5. Afterwards I created the loadings visual, covariance matrix, and the eigenvalues. Once I had the eigenvalues for the PCs I plotted them and checked the results. The PCs that should be retained are PC1, PC2, and PC3. They all have eigenvalues greater than 1 so based on the Kaiser rule, they should be retained. We can see this in the visual scree plot shown below.

A screen shot of a graph

Description automatically generated

E3.

An organization could benefit from the results of the PCA because it showed that there was benefit to the dimension reduction. This could result in easier, less computationally intense work later, and could help provide valuable insights. For example, we can see that PC3 and Income have a loadings matrix value of close to 1, so they are linked very closely. This could help provide insight into the nature of the variable “Income” and how it could affect a variety of business decisions.

F.

Uploaded.

G.

Bowne-Anderson, H. (n.d). *D206 Data Cleaning.* Introduction to Importing Data in Python. From <https://app.datacamp.com/learn/custom-tracks/custom-d206-data-cleaning>

Dr. Middleton, K. (n.d). *Getting Started with D206 Course Instruction.* Retrieved June 17th, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Data Cleaning Essentials.* Retrieved June 18h, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Data Types, Distributions, and Univariate Imputation.* Retrieved June 20th, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Detecting and Treating Duplicates.* Retrieved June 22h, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Detecting and Treating Missing Values.* Retrieved June 28th, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Detecting and Treating Outliers.* Retrieved June 30th, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Re-Expression of Categorical Variables.* Retrieved July 6th, 2024,From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Principal Component Analysis(PCA) .* Retrieved July 8th, 2024,From D206 Course Guide

H.

Dr. Middleton, K (n.d). *D206PAStepbyStepGuide (NUM4).* Retrieved June 17th, 2024, From D206 Course Guide

Dr. Middleton, K. (n.d). *Getting Started with D206 Principal Component Analysis(PCA) .* Retrieved July 8th, 2024,From D206 Course Guide