# Data Cleaning

# D206

# Amber Peacock

**Part I**

1. After careful review and consideration of the information within the Telecommunications Churn dataset, my research question is as follows: What customer factors influence a high risk of churn for the telecommunications company? There are 50 columns for customer information, and as the data analyst on this project, I want to know which factors make for a higher risk for churn. This research question is relevant to business needs because churn in this sense is the discontinuation of services which affects company profits both current and future.
2. There are a total of 50 variables within the churn data set. They are a mix of quantitative and qualitative data types. Below are the columns separated into quantitative vs. qualitative data as well as examples from the raw data file and definitions for each variable. Examples from the dataset are in parenthesis following the definitions which all correspond with the first customer in the data file. I reviewed Dr.Straw’s breakdown of data types pdf file to determine what each variable would be.

|  |  |
| --- | --- |
| Quantitative | Qualitative |
| * CaseOrder: raw data files original order. (1) * Lat: Latitude of the customer. (56.251) * Lng: Longitude of the customer. (-133.376) * Population: population within a mile based on consensus data. (38) * Children: # of children in the household per sign up information. (NA) * Age: customer age per sign up information. (68) * Income: customer income per sign up information. (28561.99) * Outage\_sec\_perweek: total time per week in seconds that customer experienced outages. (6.972566) * Email: # of emails sent to customer annually. (10) * Contacts: # of times customer contacted tech support. (0) * Yearly\_equip\_failure: # of times a customers equipment failed and needed reset/replaces this past year. (1) * Tenure: # of months customer has had services. (6.795513) * MonthlyCharge: amount charged monthly. (171.4498) * Bandwidth\_GB\_Year: average data used by the customer within a year. (904.5361) | * Customer\_id: identifier for each customer. (K409198) * Interaction: ID’s that are unique and in relation to transactions, tech support, and sign-ups. (aa90260b-4141-4a24-8e36-b04ce1f4f77b) * City: Residing city of the customer. (Point Baker) * State: Residing state of the customer. (AK) * County: Residing County of the customer. (Prince of Wales-Hyder) * Zip: Residing zip code of the customer. (99927) * Area: Type of area customer resides in. (Urban) * TimeZone: Time zone of customer. (America/Sitka) * Job: Customer’s occupation. (Environmental health practitioner) * Education: Customer’s highest earned degree. (Master’s Degree) * Employment: Employment status of the customer. (Part Time) * Marital: Marital status of the customer. (Widowed) * Gender: Customer’s self-identification. (Male) * Churn: Discontinuation of services within the past month. (No) * Techie: If the customer identified as tech savvy. (No) * Contract: Term of the contract. (One year) * Port\_modem: Whether the customer has a port modem. (Yes) * Tablet: Does the customer own a tablet? (Yes) * InternetService: type of internet service. * Phone: Does the customer have a phone service? (Fiber Optic) * Multiple: Does the customer have multiple lines? (No) * OnlineSecurity: Does the customer have the online security add-on feature? (Yes) * OnlineBackup: Does the customer have the online backup add-on feature? (Yes) * DeviceProtection: Does the customer have the device protection add-on feature? (No) * TechSupport: Does the customer have the tech support add-on feature? (No) * StreamingTV: Does the customer have TV streaming? (No) * StreamingMovies: Does the customer have movie streaming? (Yes) * PaperlessBilling: Is the customer enrolled in paperless billing? (Yes) * PaymentMethod: What the is customer’s payment method type? (Credit Card (automatic)) * Item1: [Survey Response] Timely response. (5) * Item2: [Survey Response] Timely fixes. (5) * Item3: [Survey Response] Timely replacements. (5) * Item4: [Survey Response] Reliability. (3) * Item5: [Survey Response] Options. (4) * Item6: [Survey Response] Respectful response. (4) * Item7: [Survey Response] Courteous exchange. (3) * Item8: [Survey Response] Evidence of active listening. (4) |

**Part II**

**C1.** I began the data cleaning process by running specific functions to detect duplicates, missing values, and outliers. I also assessed what values may need re-expression. I stared with detecting duplicates. To do this, I first imported pandas and numpy, and gave the command to read the csv file as df. I then ran the info() function.

Next, I ran the following code to check for duplicates: (Dr.Middleton, Getting Started With D206 - Duplicates, n.d.)

* df.**duplicated**()
* **print**(df.**duplicated**().**value\_counts**())

Next, I checked for missing values using the following code (Dr.Middleton, Getting Started With D206 - Missing Values, n.d.):

* df.**isnull**().**sum**()

To detect outliers, I ran the next code (Dr.Middleton, Getting Started With D206 - Outliers, n.d.):

* boxplot=seaborn.**boxplot**(x=’columnname’, data=df)

Lastly, I looked back at the info() output to assess which categorical fields needed re-expression in the data frame. To re-express these I used the following code (Dr.Middleton, Getting Started With D206 - Re-expression of Categorical Variables, n.d.):

* df.columnname.**unique**()

(Chantal D. Larose, 2019)

**C2**. In C1, I discussed the methods used to detect data quality issues. Now, I will discuss why I used these methods to detect different types of data quality issues. To start, I imported the any libraries/packages I may need to import the csv file and read the file as well as show me information about the variables and values within the data frame. I chose pandas and numpy to complete this task. With these libraries, my IDE was able to pull information from the csv file when given simple commands like the info() and duplicated() functions. When I performed the duplicated() function on the churn data set, I then ran a value\_counts() on the duplicated files to give me a count of how many duplicated values would be present in the file. Next, I completed steps to detect any missing values in the file. To do this, I ran the isnull().sum() function. When running this output on a data frame, you gain information on the total number of missing values grouped by each variable. Next, I wanted to detect any potential outliers, so I installed the package seaborn. After that, I used the command boxplot=seaborn.boxplot(x=’columnname’, data=df) to pull a box plot for each column. I did this with each variable that had a qualitative data type as determined in part B. Lastly, I took each categorical variable that needed re-expression and ran the unique() function to find how many unique values each variable contained that would need replaced.

**C3.** I used Python to start the cleaning process in this project. I chose Python because the syntax was easy to work with for the cleaning process as it is consistent across packages and was very simple to utilize its programming capabilities. (R or Python, 2023) I installed Anaconda onto my device and utilized Jupyter Notebooks to make importing libraries and packages easy and accessible. I originally imported pandas (pd) and numpy (np) to be able to run info(), duplicated(), isnull(), and unique() on the data frame. I then imported seaborn from matplotlib to create boxplot visualizations to identify outliers easily.

**C4.** Below is a copyable text version of the code I used as well as annotations used. Attached is also a file with the code from Python.

#Import initial libraries/packages and csv as df

import pandas as pd

import numpy as np

df = pd.read\_csv('churn\_raw\_data.csv')

df.head()

--------------------------------------------------------------------------------------------------------------------------------

#Run info() function to evaluate how many entries and non-null values

df.info()

--------------------------------------------------------------------------------------------------------------------------------

#Print current duplicates

df.duplicated()

print(df.duplicated().value\_counts())

--------------------------------------------------------------------------------------------------------------------------------

#Detect missing values

df.isnull().sum()

--------------------------------------------------------------------------------------------------------------------------------

#Import seaborn and create boxplots for outliers

Import seaborn

#Boxplot to detect outliers CaseOrder

boxplot=seaborn.boxplot(x='CaseOrder', data=df)

#Boxplot to detect outliers Lat

boxplot=seaborn.boxplot(x='Lat', data=df)

#Boxplot to detect outliers Lng

boxplot=seaborn.boxplot(x='Lng', data=df)

#Boxplot to detect outliers Population

boxplot=seaborn.boxplot(x='Population', data=df)

#Boxplot to detect outliers Children

boxplot=seaborn.boxplot(x='Children', data=df)

#Boxplot to detect outliers Age

boxplot=seaborn.boxplot(x='Age', data=df)

#Boxplot to detect outliers Income

boxplot=seaborn.boxplot(x='Income', data=df)

#Boxplot to detect outliers Outage\_sec\_perweek

boxplot=seaborn.boxplot(x='Outage\_sec\_perweek', data=df)

#Boxplot to detect outliers Email

boxplot=seaborn.boxplot(x='Email', data=df)

#Boxplot to detect outliers Contacts

boxplot=seaborn.boxplot(x='Contacts', data=df)

#Boxplot to detect outliers Yearly\_equip\_failure

boxplot=seaborn.boxplot(x='Yearly\_equip\_failure', data=df)

#Boxplot to detect outliers Tenure

boxplot=seaborn.boxplot(x='Tenure', data=df)

#Boxplot to detect outliers MonthlyCharge

boxplot=seaborn.boxplot(x='MonthlyCharge', data=df)

#Boxplot to detect outliers Bandwidth\_GB\_Year

boxplot=seaborn.boxplot(x='Bandwidth\_GB\_Year', data=df)

#Re-expression: Detecting unique values in Area

df.Area.unique()

#Detecting unique values in Employment

df.Employment.unique()

#Detecting unique values in Education

df.Education.unique()

#Detecting unique values in Marital

df.Marital.unique()

#Detecting unique values in Gender

df.Gender.unique()

#Detecting unique values in Churn

df.Churn.unique()

#Detecting unique values in Techie

df.Techie.unique()

#Detecting unique values in Contract

df.Contract.unique()

#Detecting unique values in Port\_modem

df.Port\_modem.unique()

#Detecting unique values in Tablet

df.Tablet.unique()

#Detecting unique values in InternetService

df.InternetService.unique()

#Detecting unique values in Phone

df.Phone.unique()

#Detecting unique values in Multiple

df.Multiple.unique()

#Detecting unique values in OnlineSecurity

df.OnlineSecurity.unique()

#Detecting unique values in OnlineBackup

df.OnlineBackup.unique()

#Detecting unique values in DeviceProtection

df.DeviceProtection.unique()

#Detecting unique values in TechSupport

df.TechSupport.unique()

#Detecting unique values in StreamingTV

df.StreamingTV.unique()

#Detecting unique values in StreamingMovies

df.StreamingMovies.unique()

#Detecting unique values in PaperlessBilling

df.PaperlessBilling.unique()

#Detecting unique values in PaymentMethod

df.PaymentMethod.unique()

**Part III**

**D1.** The first code I ran in efforts to detect issues in the data was duplicated() to detect duplicated. When running this code, there were 10,000 entries that populated as False and 0 that populated as True. Below is a picture of that code and the output which can also be found in the attached file. (Dr.Middleton, Getting Started With D206 - Duplicates, n.d.)

A screenshot of a computer code

Description automatically generated

The next code I ran to detect any issues in the data was isnull().sum(). When I ran this code, a few of the variables indeed showed some missing data values. Below is a picture of the code as well as the output. (Dr.Middleton, Getting Started With D206 - Missing Values, n.d.)

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

As can be seen above, different variables such as children, age, income, techie, internet service, phone, tech support, tenure, and bandwidth had some missing values. Next, I ran code for outliers. I performed a similar command for each quantitative variable in the data frame. Below are some pictures of some of the outputs. (Dr.Middleton, Getting Started With D206 - Outliers, n.d.)These can be viewed in the attached file as well.

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screen shot of a graph

Description automatically generated

As can be seen above, some outputs of this code like for CaseOrder showed no outliers, while Lat showed many. Meanwhile, others like Children showed few. The outputs for the remaining variables can be viewed in the attached code file. For re-expressing categorical variables, I found that some variables contained 3 or less unique values that would need numeric re-expression and other that had 5+. Below are pictures of that code and output (which is also in the attached code file).

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

**D2.** There were no duplicates in this data set. When using the code df.duplicated()

print(df.duplicated().value\_counts()) , there were no True populations. Therefore, there was no treatment for duplicates for this project.

For missing values, I followed the process laid out in Dr.Middleton’s video. I began by finding missing values, then I performed imputation on the variable, then verified the missing values for that variable was gone and that the distribution was similar/reasonable. I completed this process for the following quantitative variables:

* Children
* Age
* Income
* Tenure
* Bandwidth\_GB\_Year

‘Children’ and ‘Income’ had a skewed right distribution, so I performed a median imputation for both of those variables. ‘Age’ was uniform, so I performed a mean imputation for that variable. ‘Tenure’ and ‘Bandwidth\_GB\_Year’ were both bimodal so I used median imputation for both of those variables. After performing the imputation code according to distribution from Dr.Middleton’s video, I re-ran the isnull().sum() code to check for missing values in each of those variables (confirming zero) and re-ran the distribution check to make sure the distribution was similar to before imputation was performed. For categorical variables I had to use a different approach. I received an error when trying to find the distribution for these because they are objects, so since they are qualitative, I imputated them using the mode. I ran the same isnull().sum() function code after imputing them on the mode to verify all null values were filled. I used this process for ‘Techie’, ‘Phone’, ‘InternetSupport’, and ‘TechSupport’.

For outliers, I constructed the boxplot using boxplot=seaborn.boxplot(x='columnname', data=df) then assessed the boxplot to see if there were any outliers. For each variable, if there was an outlier I determined whether that was reasonable for the data being collected. For variables where the outliers were reasonable, my treatment plan was to retain the outliers as they are important to the reliability of the data. For variables where it was clear that the outliers were not reasonable, I imputed them by their median. ‘CaseOrder’, ‘Age’, ‘Tenure’, ‘MonthlyCharge’, and ‘Bandwidth\_GB\_Year’ showed no outliers and therefore needed no treatment. ‘Lng’ and ‘Lat’ showed outliers but those are reasonable because these are variables based on geographic location so they will vary and it is important to keep those variables as they are so that any business questions regarding customer demographics are accurate. Next is population. This variable had many outliers but I kept these as well. Some areas will be more densely populated than others (rural, suburban, urban) so this information is likely accurate and will provide insights for business needs in the future, therefore it is important to not shorten the sample or remove reliability by imputing or deleting. For ‘Children’ I imputed by the median. There were few outliers in this variables boxplot and by taking the median I am not removing having children as a factor all together, but still accomplishing the cleaning out the outliers. For ‘Income’, there were many outliers present. I left these due to income being a very fluid and differentiating value. Income will vary amongst households based on many factors and this information could be pertinent to answering business questions. The next variable I tested was ‘Outage\_sec\_perweek’. This variable had many outliers. Some of which were below zero, which does not make logical sense. Therefore, I imputed by the median and defined any outputs as greater than 0. Next is ‘Email’. For this variable, I determined that it was reasonable for this value to differ. The reason this column exists is to monitor differing values for emails sent to customers, so it makes sense that some customers would receive more or less than the median amount. Similarly, ‘Contacts’ for some customers (especially ones experiencing more frequent issues) would contact support more, so I retained all values here as well. ‘Yearly\_equip\_failure’ was retained for the same reason, since some customers will experience more of this value. For ‘MonthlyCharge’ some of the outliers were a cause for concern. It didn’t seem reasonable that some customers were paying significantly more than all the others. I imputed this variable by the median to get a more central price point. Since the outliers were well over $250, I imputed and defined the limit as less than 250. Overall, I made determinations based on the context of the data (it’s a churn data set) as well as the definitions. I made decisions for treatment based on the most logical standpoint for each variable while trying to keep the integrity of the data.

The last data quality issue I addressed for this project is the re-expression of variables. To do this, I took each categorical variable that needed re-expression and gave numerical value to categorical ones. To do this, I first used unique() to find all unique values in each variable. Then, I created a new variable with column\_numeric for the name. This allows me to keep the categorical data in a qualitative form but gain access to the numerical content for the values (like 0 for “No”, 1 for “Yes”). Next, I created a dictionary for the new variable and assigned a numerical value to each of the unique values from the original variables. I ran a replace() function on the new variable to replace the previously specified phrase with the new phrase. The variables that were replaced can be viewed as well as the code used in the attached file. Below is a picture of all of the new variables after re-expression.

A screenshot of a computer program

Description automatically generated

**D3.** After confirming the detection of data quality issues, I began treatment. Since there were no duplicates in the data (zero True entries, 10000/10000 were False), I did not need to do any treatments for duplicates. Next, when looking at missing data, there were 9 variables that had missing values.. When treating missing values, your options are typically going to be deletion or imputation. I did not want to delete data because the total entries is only 10,000 and the missing values were between 1,000 and 2,500 for each of the columns. I did not want to remove that significant of a portion of the data causing a drastically reduced sample size and potentially biased results. Due to this, I decided to proceed with imputation. I used univariate imputation for the missing values in this data set. To start the imputation process, I examined the distribution of the variable. I did this by creating a histogram to see what the original distribution for each of these variables looked like. After determining that for each variable, I performed the imputation. For categorical variables with missing values I imuputed by the mode. Lastly, I verified that the missing values were resolved and that the distribution after imputation was relatively similar to the distribution prior to imputation. Below is the code as well as the output for the variable “Children”. This as well as the other variables can be found in can be found in the attached file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

As you can see, the number of null values went from 2,495 to 0. The distribution also remained similar after imputation. I repeated this process for the remaining variables using mean vs median depending on the distribution type. After all quantitative missing values were remedied, I verified this by repeating the isnull().sum() function. The only variables with outputs greater than zero are categorical variables which I then imputed by the mode.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Next, I focused on outliers. As could be seen in D1, I used MatPlotLib to populate boxplots for each of the quantitative variables. I assessed each boxplot for outliers. Any variables where outliers were not reasonable were then imputated using the median. Then, I pulled the updated boxplot to verify the eradication of outliers. Below is an example of the code I used and the result. The remainder of the boxplots, my findings, my solutions, and outputs are in the attached file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Lastly, I re-expressed all categorical variables so that correlations and information could be easily assessed. The only variables that were not re-expressed are ones that were already in numerical form like likert scales. I did this by finding unique values, creating a new variable to hold the values, giving those unique values numerical value, then replacing old values with the new counterparts in the new variable. Below is an example of that as well as the output from info() after the code was ran to verify the addition of the new variable. All others are in the attached file.

A computer screen shot of a computer screen

Description automatically generated



I repeated the above process for each of the categorical values mentioned in previous sections. Now when info() is ran, the new variables populate at the bottom of the list.

A screenshot of a computer code

Description automatically generated

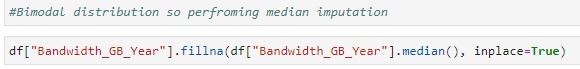
**D4**.

*Duplicates:*

A screenshot of a computer code

Description automatically generated

\*Note: There was no treatment because there were no duplicates.  
  
*Missing Values:*



A screenshot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

*Outliers:*

A screenshot of a computer

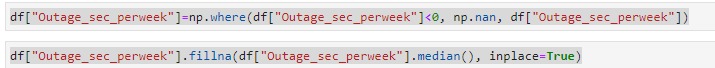
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a phone

Description automatically generated

A screenshot of a phone

Description automatically generated

\*Note: There were other values that were tested with boxplots aside from what is shown above. Those values did not have outliers OR were retained. Annotation and code for those variables can be found in the attached code.

*Re-expression:*

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a phone

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Note: As can be seen, the general structure for the re-expression remains the same across the variables. The main difference is in how many unique values needed re-expressed.   
  
\*\*For all code provided above, please see attached code.

**D5.** \*\* Check attached files! \*\*  
**D6.** There are disadvantages with any method used in the data cleaning process. For detecting duplicates, I used the duplicated() function. While this is very efficient in providing a list of duplicates, it doesn’t give other information about the duplicates, it only lists how many are present in the data. I did not have to treat duplicates in this project because there were no True results when running that function. Had there been True results, I would have used the distinct() function to remove duplicates. A disadvantage to that could have been a significantly shortened data set. (Dr.Middleton, Getting Started With D206 - Duplicates, n.d.)

For missing values, I used the isnull().sum() function for detection. This gives me a sum of how many missing values are in each variable. A disadvantage to this detection route is that it shows no visualization for where the missing values are. An alternative method that would show a visual (but alternatively, no exact value) of missing values would be installing missingno and using the code msno.matrix(df). As for treatment, I used univariate imputation. As with any other methods, there are disadvantages to this as well though this is my most preferred method. With univariate imputation, you could possibly distort data/distribution of the data which can lead to less accurate representation of the data in the future when modeling. (Dr.Middleton, Getting Started With D206 - Missing Values, n.d.)

For outliers, I used boxplots to detect any in the data. A disadvantage to this method is the lack of information on the outliers. Although it is a quick way to show outliers, it gives no other information and assumes symmetry in the data. For treatment, I used two methods depending on my evaluation. For some values I decided to retain values. A disadvantage to this method includes a decrease in normality if running statistical tests in future processes of the analysis. In others, I imputated using the median. The biggest disadvantage to this is that you’re essentially guesstimating values which could lead to bias in the data. (Dr.Middleton, Getting Started With D206 - Outliers, n.d.)

The last process I ran was re-expression of categorical data. For detection, I reviewed the categorical variables from the chart I made in part B, and assessed which of those variables would be beneficial to be accessed as numerical variables. A disadvantage to this process is that it was long and tedious. As far as how I re-expressed the variables, I used ordinal encoding to give the items numerical value. A disadvantage to ordinal encoding is that it can lead to misleading machine learning models in future data analysis processes. (Dr.Middleton, Getting Started With D206 - Re-expression of Categorical Variables, n.d.)

**D7.** While I tried my best to keep the integrity of the data, there are always complications that arise from cleaning data. I did not delete any of the data since that would have required a large portion of the data to be removed completely. However, even when imputing to the mean or median, you are ultimately altering the data. With this being said, depending on the business question, the answer and future decision making may be made on altered data whereas a different conclusion may have been made with the raw data. The point of data analytics is to help stakeholders with decision-making based on data within the company. Changing, deleting, or altering data will always have an impact on the decision-making process and will influence the direction of future data. Regarding my specific research question, it is possible that certain outliers being imputated as well as missing values may alter my data especially regarding PCA’s. Conclusions could be drawn about correlations that otherwise would not have been made.

**Part IV**

**E1.** I used the following variables for my PCA configuration:

* Lat
* Lng
* Population
* Age
* Income
* Outage\_sec\_perweek
* Email
* Contacts
* Yearly\_equip\_failure
* Tenure
* MonthlyCharge
* Bandwidth\_GB\_Year

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**E2.** Next, I created a scree plot following the Kaiser Criterion and making a line at y=1. Below are pictures of my code as well as the scree plot results.

A screenshot of a computer

Description automatically generatedA graph showing the number of components

Description automatically generated

As can be seen, all meaningful PCA’s are located above the red line at 1 (eigenvalue). This line was placed in accordance with the Kaiser Criterion to give meaningful results to deliver powerful insights on correlations within variables of the data. As you can see, PC’s 1-7 (labeled 0-6 on the scree plot) are all above that line, indicating that those hold meaningful correlations for the data so those are the PC’s I will retain. (Dr.Middleton, Getting Started With D206 - PCA, n.d.)

**E3**. There are many benefits for organizations using PCA’s. It helps reduce dimensionality to provide easier and more functional models and visualizations, reduction of noise in the data, and maximizes the variance of data. (PCA: What, How, and Why?, 2023) An example of how PCA could be used to help a business is to know what factors correlate with an increase in churn as if to decrease future churn.

A screenshot of a computer

Description automatically generated

What is shown here is that in PC3 there is a correlation between a higher number of Outage\_sec\_perweek and MonthlyCharge. This may indicate that customers with more issues with outages are also experiencing a higher monthly charge. If we think of this from a logical standpoint, if customers are experiencing both of these conditions, they may look elsewhere for business.

**G.** No third-party code references were used. All content was from Dr.Middleton’s *Getting Started With D206* video series referenced in H.

# **H**: References

Chantal D. Larose, D. T. (2019). Data Science Using Python and R. Wiley.

*Cleaning Data In Python - Finding Duplicates*. (n.d.). Retrieved from DataCamp: https://campus.datacamp.com/courses/cleaning-data-in-python/common-data-problems-1?ex=10

Dr.Middleton. (n.d.). *Getting Started With D206 - Duplicates*. Retrieved from Panopto: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6eedfad4-240e-4c5c-8eab-b058003d3e6b

Dr.Middleton. (n.d.). *Getting Started With D206 - Missing Values*. Retrieved from Panopto: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=767749d2-ba19-4f94-bec8-b058017b2f5e

Dr.Middleton. (n.d.). *Getting Started With D206 - Outliers*. Retrieved from Panopto: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=19c24c56-0f37-408e-bb1f-b059002a77ac

Dr.Middleton. (n.d.). *Getting Started With D206 - PCA*. Retrieved from Panopto: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3bcc452f-fa35-43be-b69f-b05901356f95

Dr.Middleton. (n.d.). *Getting Started With D206 - Re-expression of Categorical Variables*. Retrieved from Panopto: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=bd7b8541-77ba-42e0-80a4-b059010bc790

*PCA: What, How, and Why?* (2023, February 8). Retrieved from Bigabid: https://www.bigabid.com/what-is-pca-and-how-can-i-use-it/

*R or Python*. (2023, July 7). Retrieved from Western Governors University: https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html#\_