**D206- Data Cleaning Performance Assessment**

**Isiasha Gordon**

**Western Governs University**

**Table of Contents**

[Part I: Research Question and Variables 3](#_Toc126087327)

[Section A. Research Question 3](#_Toc126087328)

[Section B. Required Variables 3](#_Toc126087329)

[Part II: Data-Cleaning Plan (Detection) 5](#_Toc126087330)

[Section C1: Plan to find Anomalies 5](#_Toc126087331)

[Section C2: Justification of Approach 6](#_Toc126087332)

[Section C3: Justification of Tools 7](#_Toc126087333)

[Section C4: Provide the Code 8](#_Toc126087334)

[Part III: Data Cleaning (Treatment) 11](#_Toc126087335)

[Section D1: Cleaning Findings 11](#_Toc126087336)

[Section D2: Justification of Mitigation Methods 13](#_Toc126087337)

[Section D3: Summary of the Outcomes 17](#_Toc126087338)

[Section D4: Mitigation Code 19](#_Toc126087339)

[Section D5: Cleaning Data 24](#_Toc126087340)

[Section D6: Limitations 24](#_Toc126087341)

[Section D7.  Impact of the Limitations 24](#_Toc126087342)

[Part IV: PCA 25](#_Toc126087343)

[Section E1: Principal Components 25](#_Toc126087344)

[Section E2: Criteria Used 26](#_Toc126087345)

[Section E3: Benefits 27](#_Toc126087346)

[Section F: Video 27](#_Toc126087347)

[Section G: Third-Party Code References 28](#_Toc126087348)

[Section H: References 28](#_Toc126087349)

## 

## Part I: Research Question and Variables

### Section A. Research Question

Do customers experiencing the most outages with their service and equipment failure increase the cancellation of service?

### Section B. Required Variables

The following variables were in the churn data set:

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** | **Example** |
| Case Order | Qualitative | Placeholder for order of data | 15 |
| Customer\_id |  | Distinct customer identification | H68068 |
| Interaction | Qualitative | Specific identification # for each customer interaction | 8dc7ad15-2f59-4c77-9640-6f2c0000b3fc |
| City | Qualitative | Customer city of residence | Hillside |
| State | Qualitative | Customer state of residence | IL |
| County | Qualitative | Customer county of residence | Cook |
| Zip | Qualitative | Customer zip code of residence | 60612 |
| Lat | Qualitative | GPS coordinates of residence | 41.86752 |
| Lng | Qualitative | GPS coordinates of residence | -87.90222 |
| Population | Quantitative | Population of customer residence | 8165 |
| Area | Qualitative | Area type (rural, urban, suburban) | Urban |
| TimeZone | Qualitative | Time zone of customer residence | America/Chicago |
| Job | Qualitative | Customer job title | Automotive engineer |
| Children | Quantitative | Number of children of customer | 5 |
| Age | Quantitative | Age of customer | 30 |
| Education | Qualitative | Customer highest level of education | Associate's Degree |
| Employment | Qualitative | Customer employment status | Full Time |
| Income | Quantitative | Customer annual income reported | 64256.81 |
| Martial | Qualitative | Customer marital status | Separated |
| Gender | Qualitative | Customer gender | Male |
| Churn | Qualitative | If the customer discontinued services in the last month | Yes |
| Outage\_sec\_perweek | Quantitative | Avg number of seconds per week of system outages in customer’s neighborhood | 12.63069124 |
| Email | Quantitative | Number of emails sent to customer over past year | 10 |
| Contacts | Quantitative | Number of times customer contacted technical support | 3 |
| Yearly\_equip\_failure | Quantitative | Number of times customer’s equipment failed and replaced | 0 |
| Techie | Qualitative | If customer considers themselves technically inclined | No |
| Contract | Qualitative | Contract term of the customer | Month-to-month |
| Port\_modem | Qualitative | If customer have portable modem | No |
| Tablet | Qualitative | If customer own a tablet | No |
| InternetService | Qualitative | Customer’s internet service provider | DSL |
| Phone | Qualitative | If customer has phone service | No |
| Multiple | Qualitative | If customer has multiple lines | Yes |
| OnlineSecurity | Qualitative | If customer has online security add-on | No |
| OnlineBackup | Qualitative | If customer has online backup add-on | No |
| DeviceProtection | Qualitative | If customer has device protection add-on | No |
| TechSupport | Qualitative | If customer has technical support add-on | Yes |
| StreamingTV | Qualitative | If customer has streaming TV | No |
| StreamingMovies | Qualitative | If customer has streaming movies | No |
| PaperlessBilling | Qualitative | If customer has paperless billing | Yes |
| PaymentMethod | Qualitative | Customer’s payment method | Bank Transfer(automatic) |
| Tenure | Quantitative | # of months customer has stayed with provider | 10.06019902 |
| MonthlyCharge | Quantitative | Amount charged to customer monthly | 160.8055418 |
| Bandwidth\_GB\_Year | Quantitative | Avg amount of data customer used (GB) in a year | 1948.694497 |
|  |  |  |  |
|  |  | *Below are survey responses of importance of various factors/surfaces on a scale of 1 to 8 (1 =mostly important; 8 = least important):* |  |
| Item1 | Qualitative | Survey response- Timely response | 1 |
| Item2 | Qualitative | Survey response- Timely fixes | 3 |
| Item3 | Qualitative | Survey response- Timely replacements | 4 |
| Item4 | Qualitative | Survey response- Reliability | 2 |
| Item5 | Qualitative | Survey response- Options | 3 |
| Item6 | Qualitative | Survey response- Respectful response | 4 |
| Item7 | Qualitative | Survey response- Courteous exchange | 4 |
| Item8 | Qualitative | Survey response-Evidence of active listening | 2 |

## Part II: Data-Cleaning Plan (Detection)

### Section C1: Plan to find Anomalies

Python was used to detect duplicate data, missing values, outliers, and any other data quality issues in the churn data set. Utilizing the “import pandas as pd” command Panda was imported into Python. The read\_cvs() function was used to read the churn data on my local hard drive. This file was assigned to the variable “df” for easy reference. To determine the data types included in the churn data the df.info(file\_path). The data type of each variable is needed information due to certain functions working only with specific functions. This includes the column names and the number of non-null values for each column. Once datatypes are known the data could be cleaned. Cleaning data included detecting duplicates, and identifying missing values and outlies.

To determine if there were duplicate entries in the data the df.duplicated() function was completed. This function returns columns with TRUE or FALSE values. If the column returnes a TRUE value there are duplicate records but if a FALSE value is returned there are no duplicates. The results of this function indicated all FALSE values meaning there were no duplicates values in the data set. To verify and count all entries that were FALSE the print(df.duplicated().value\_counts()) was used. If duplicates were present the df.drop\_duplicates() function could have been used to drop duplicate values.

The next step included determining if there were missing values. Missing values are usually represented in the form of nan, null, or none in the dataset. The df.isnull().sum() function was used. This function counted how many missing values were present in each column. The following columns included missing values: children, age, income, techie, phone, tech support, tenure, bandwidth\_GB\_year. A visualization of the missing data was completed using the missingno package. The install missigno and import missigno were used to import into jupyter notebook. The function msno.matrix(df) was used to create a visualization of the missing data. The matplotlib package was also imported to serve as a visualization of the data. The function plt.hist(df) was used as an additional visual to create a histogram and to examine the distribution of the missing values in the children, age, and income columns of the data set. This function was only performed on the children, age, tenure, bandwidth\_gb\_year, and income columns because it can only be applied to quantitative data. Once distribution was shown in each histogram the imputation methods could be determined.

The columns techie, phone, and techsupport also had missing data. These columns have qualitative data consisting of YES/NO values. Ordinal encoding was used to re-express values as numeric values. The function mapping\_dict = {'No': 0, 'Yes': 1}, df['Churn'] = df['Churn'].replace(mapping\_dict) was used to reassign values to 0 and 1. The function plt.hist(df[') was then used to visualize the missing data like previous numeric categories.

Univariate imputation was then used to treat missing values for all quantitative data. The mean and median were calculated to replace missing values. The function df.isnull().sum() was then used again to verify the above columns with missing values had been treated. Once duplicate and missing values were detected and treated outliers were then determined for all quantitative variables. All the quantitative variables from the churn data were plotted on boxplots to visualize outliers. For those variables that did not show obvious outliers z-scores were computed. This was completed after importing the seaborn library with the function boxplot=seaborn.boxplot(). The exact number of outliers was determined using the z-values since the boxplot could not provide an exact number. Outliers were then treated using the retention method).

Lastly, PCA was performed with the numerical variables from the data set for increased data compression and visualization.

### Section C2: Justification of Approach

The methods discussed above were used to clean the raw data set of the churn data. Cleaning data is important in drawing accurate conclusions when analyzing data. It allows for different sorting options, filtering, and modification to the data set. Using the methods above helps detect duplicates while maintaining the integrity of the data. It is necessary to detect and treat duplicate data because duplicate entries can lead to miscalculations or misrepresentations of the data. When detecting missing values (represented in the form of nan, null, or none) the df.isnull().sum() function counts the missing values. This gives a count of the missing values for each variable. The missingno function allows for visualizations of the missing data in each variable. The library matplotlib.pyplot allows for visualization (i.e. box plots, z-scores, scatterplots, and histograms) of the functions imputed. Methods such as calculating the mean and mode can fix missing data and provide a better representation of the data. This allows one to know the exact location of the missing data for treatment.

Outliers need to be detected and treated because outliers can provide incorrect/inconsistent collusions from the data. Outliers come from data entry errors, measurement errors, experimental errors, sampling errors, or novelties in the data (Lacrose & Lacrose, 2019). Box plots, z-scores, and histograms were used to detach outliers. The scipy.stats function was used to calculate z-cores. Different methods of visualization were used based on the number of outliers. When outliers were not readily visible from box plots z-scores were values to determine value and volume. There were two types of data from this data set, quantitative and qualitative data. Qualitative or categorical data (i.e. yes/no) requires re-expression or encoding of numbers to perform statical modeling. (Lacrose & Lacrose, 2019). Ordinal encoding was thus used to transform categorical data into nominal data. Education was a qualitative data variable that need to be transformed into a quantitative data set. A numerical value was assigned by creating a data dictionary and mapping the categorical responses to a numeric value, making it easier to operate on. For columns such as churn, techie, and tablet which are yes/no answers, we can assign binary values through re-expression to make them easier to work with when creating

visual tools.

### Section C3: Justification of Tools

Data cleaning for the churn data set was completed utilizing Python. Python is an open-sourced programing language used for analysis and development. Python has a consistent syntax that makes coding and debugging user-friendly for beginners. Python is flexible and has the ability to import packages and to tailor data. The following packages were imported and used for their advantages.

|  |  |
| --- | --- |
| **Packages/Libraries** | **Purpose** |
| pandas | Main package for data uploading and manipulation |
| numpy | Main package for working with arrays |
| Matplotlib.pyplot | Visualization |
| Missingno | Missing values visualization |
| Seaborn | Advanced visualiziation |
| scipy.stats | Normalization and statistics |
| sklearn.decompositionimport PCA | PCA analysis |

### Section C4: Provide the Code

The following functions were used for the detection of duplicates, missing data, and outliers. Also, see detailed code attached.

1. *Loading Data*:

File\_path = "/Users/igmark/Desktop/WGU Data Files/churn\_raw\_data.csv"

df = pd.read\_csv(file\_path)

1. *Determine data types:*

df.info(file\_path)

1. *Detecting duplicates:*

df.duplicated()

1. *Missing data:*

df.isnull().sum() – detection

msno.matrix(df) – visualization

plt.hist(df[''])- visualiziation

1. *Detecting Outliers:*

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

df['Z\_Score\_Population'] = stats.zscore(df['Population'])

df[['Population','Z\_Score\_Population']].head()

plt.hist(df['Z\_Score\_Population'])

plt.show()

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

df['Z\_Score\_Email'] = stats.zscore(df['Email'])

df[['Email,'Z\_Score\_Email']].head()

plt.hist(df['Z\_Score\_Email'])

plt.show()

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

df['Z\_Score\_Children'] = stats.zscore(df['Children'])

df[[' Children ','Z\_Score\_Children']].head()

plt.hist(df['Z\_Score\_Children'])

plt.show()

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

df['Z\_Score\_Age'] = stats.zscore(df['Age'])

df[['Age','Z\_Score\_Age']].head()

plt.hist(df['Z\_Score\_Age'])

plt.show()

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

df['Z\_Score\_Income'] = stats.zscore(df['Income'])

df[['Income','Z\_Score\_Income']].head()

plt.hist(df['Z\_Score\_Income'])

plt.show()

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

df['Z\_Score\_Outage\_sec\_perweek'] = stats.zscore(df['Outage\_sec\_perweek'])

df[[''Outage\_sec\_perweek'','Z\_Score\_Outage\_sec\_perweek']].head()

plt.hist(df['Z\_Score\_Outage\_sec\_perweek'])

plt.show()

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

df['Z\_Score\_Contacts'] = stats.zscore(df['Contacts'])

df[[' Contacts,'Z\_Score\_Contacts']].head()

plt.hist(df['Z\_Score\_Contacts'])

plt.show()

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

df['Z\_Score\_Yearly\_equip\_failure'] = stats.zscore(df['Yearly\_equip\_failure'])

df[[' Yearly\_equip\_failure'','Z\_Score\_Yearly\_equip\_failure']].head()

plt.hist(df['Z\_Score\_Yearly\_equip\_failure'])

plt.show()

boxplot=seaborn.boxplot(x= ‘Tenure’, data=df)

df['Z\_Score\_Tenure'] = stats.zscore(df['Tenure'])

df[[' Tenure','Z\_Score\_Tenure']].head()

plt.hist(df['Z\_Score\_Tenure'])

plt.show()

boxplot=seaborn.boxplot(x= ‘MonthlyCharge’, data=df)

df['Z\_Score\_MonthlyCharge'] = stats.zscore(df['MonthlyCharge'])

df[[' MonthlyCharge'','Z\_Score\_MonthlyCharge']].head()

plt.hist(df['Z\_Score\_MonthlyCharge'])

plt.show()

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

df['Z\_Score\_Bandwidth\_GB\_Year'] = stats.zscore(df['Bandwidth\_GB\_Year'])

df[[''Bandwidth\_GB\_Year' ','Z\_Score\_Bandwidth\_GB\_Year']].head()

plt.hist(df['Z\_Score\_Bandwidth\_GB\_Year'])

plt.show()

1. *Expressing values for ‘Education’ and ‘Churn’:*

dict\_ed = {"Education\_num":

{"Regular High School Diploma": 0,

"Bachelor's Degree": 1,

"Some College, 1 or More Years, No Degree": 2,

"9th Grade to 12th Grade, No Diploma":3,

"Master's Degree": 4,

"Associate's Degree": 5,

"Some College, Less than 1 Year": 6,

"Nursery School to 8th Grade": 7,

"GED or Alternative Credential": 8,

"Professional School Degree": 9

"No Schooling Completed": 10,

"Doctorate Degree": 11}}

df["Education\_num"] = df["Education"]

df.replace(dict\_ed, inplace = True)

counts = df["Education\_num"].value\_counts()

print(counts)

plt.hist(df['Education'])

mapping\_dict = {'No': 0, 'Yes': 1}

df['Churn'] = df['Churn'].replace(mapping\_dict)

print(df.head())

1. *Expressing the number of outliers and values for outliers:*

zscores = df[['Population', 'Z\_Score\_Population']]

data = df[['Population']]

num\_outliers, outliers = count\_and\_value\_outliers(zscores, data)

print("Number of outliers:", num\_outliers)

print("Outlier values:", outliers)

df\_outliers = df.query('(Z\_Score\_Children > 3) | (Z\_Score\_Children < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Children'].to\_dict().values())

df\_outliers = df.query('(Z\_Score\_Income > 3) | (Z\_Score\_Income < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Income'].to\_dict().values())

df\_outliers = df.query('(Z\_Score\_Outage\_sec\_perweek > 3) | (Z\_Score\_Outage\_sec\_perweek < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Outage\_sec\_perweek'].to\_dict().values())

df\_outliers = df.query('(Z\_Score\_Email > 3) | (Z\_Score\_Email < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Email'].to\_dict().values())

df\_outliers = df.query('(Z\_Score\_Contacts > 3) | (Z\_Score\_Contacts < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Contacts'].to\_dict().values())

df\_outliers = df.query('(Z\_Score\_Yearly\_equip\_failure > 3) | (Z\_Score\_Yearly\_equip\_failure < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Yearly\_equip\_failure'].to\_dict().values())

df\_outliers = df.query('(Z\_Score\_MonthlyCharge > 3) | (Z\_Score\_MonthlyCharge < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['MonthlyCharge'].to\_dict().values())

## Part III: Data Cleaning (Treatment)

### Section D1: Cleaning Findings

The data cleaning process began by determining the data types with the function df.info(file\_path). The following data types were present in the churn data set:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 52 columns):

# Column Non-Null Count Dtype

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

0 Unnamed: 0 10000 non-null int64

1 CaseOrder 10000 non-null int64

2 Customer\_id 10000 non-null object

3 Interaction 10000 non-null object

4 City 10000 non-null object

5 State 10000 non-null object

6 County 10000 non-null object

7 Zip 10000 non-null int64

8 Lat 10000 non-null float64

9 Lng 10000 non-null float64

10 Population 10000 non-null int64

11 Area 10000 non-null object

12 Timezone 10000 non-null object

13 Job 10000 non-null object

14 Children 7505 non-null float64

15 Age 7525 non-null float64

16 Education 10000 non-null object

17 Employment 10000 non-null object

18 Income 7510 non-null float64

19 Marital 10000 non-null object

20 Gender 10000 non-null object

21 Churn 10000 non-null object

22 Outage\_sec\_perweek 10000 non-null float64

23 Email 10000 non-null int64

24 Contacts 10000 non-null int64

25 Yearly\_equip\_failure 10000 non-null int64

26 Techie 7523 non-null object

27 Contract 10000 non-null object

28 Port\_modem 10000 non-null object

29 Tablet 10000 non-null object

30 InternetService 10000 non-null object

31 Phone 8974 non-null object

32 Multiple 10000 non-null object

33 OnlineSecurity 10000 non-null object

34 OnlineBackup 10000 non-null object

35 DeviceProtection 10000 non-null object

36 TechSupport 9009 non-null object

37 StreamingTV 10000 non-null object

38 StreamingMovies 10000 non-null object

39 PaperlessBilling 10000 non-null object

40 PaymentMethod 10000 non-null object

41 Tenure 9069 non-null float64

42 MonthlyCharge 10000 non-null float64

43 Bandwidth\_GB\_Year 8979 non-null float64

44 item1 10000 non-null int64

45 item2 10000 non-null int64

46 item3 10000 non-null int64

47 item4 10000 non-null int64

48 item5 10000 non-null int64

49 item6 10000 non-null int64

50 item7 10000 non-null int64

51 item8 10000 non-null int64

dtypes: float64(9), int64(15), object(28)

There were 9 floats, 15 integers, and 28 strings present in the data set. Different data types have different methods for handling missing values (Lacrose & Lacrose, 2019). Duplicates in the data set could then be determined with function df.duplicated(). This function provided TRUE/FALSE values, where TRUE means duplicates were present and FALSE means duplicates were not present (Lacrose & Lacrose, 2019). The results below indicate there were no duplicate data entries in the data set.

0 False

1 False

2 False

3 False

4 False

...

9995 False

9996 False

9997 False

9998 False

9999 False

Length: 10000, dtype: bool

### 

### Missing values were then detected using the df.isnull().sum(). The following variables had missing values.

|  |  |
| --- | --- |
| Variable | # of Missing Values |
| Children | 2495 |
| Age | 2475 |
| Income | 2490 |
| Techie | 2477 |
| Phone | 1026 |
| TechSupport | 991 |
| Tenure | 931 |
| Bandwidth\_GB\_Year | 1021 |

### Missing data could also be visualized by importing the library and using the function: import missingno as msno msno.matrix(df).

Box plots and z-scores were used to detect outliers in the data set. Outliers were detected from all the quantitative variables in the data set. The quantitative variables included Population, Children, Age, Income, Outage\_sec\_perweek, Email, Contacts, Yearly\_equip\_failure, Tenure, MonthlyCharge, and Bandwidth\_GB\_Year. Box plots were completed for each of the individual variables after importing the seaborn library with the function: import seaborn

boxplot=seaborn.boxplot(x='Children',data=df). See attached code for each variable. After examining the box plots of each variable the following variables that exhibited outliers were Population, Children, Income, Outage\_sec\_perweek, Email, Contacts, Yearly\_equip\_failure, and MonthlyCharge. To determine the number and values of the outliers in these variables z-scores were then taken. Values for each are printed in the code attached.

|  |  |
| --- | --- |
| **Variable** | **# of Outliers** |
| Population | 219 |
| Children | 302 |
| Income | 193 |
| Outage\_sec\_perweek | 491 |
| Email | 12 |
| Contacts | 165 |
| Yearly\_equip\_failure | 94 |
| MonthlyCharge | 3 |

Data wrangling includes transforming the data type into another format more appropriate for use. The re-expression of categorical data to numeric data allows for easier filtration and manipulation of the raw data. The variable ‘Education’ was originally expressed as categorical data. When the variable transformed into numeric data the following results were shown:

0 2421 Regular High School Diploma

1 1703 Bachelor's Degree

2 1562 Some College, 1 or More Years, No Degree

3 870 9th Grade to 12th Grade, No Diploma

4 764 Master's Degree

5 760 Associate's Degree

6 652 Some College, Less than 1 Year

7 449 Nursery School to 8th Grade

8 387 GED or Alternative Credential

9 198 Professional School Degree

10 118 No Schooling Completed

11 116 Doctorate Degree

Name: Education\_num, dtype: int64

### Section D2: Justification of Mitigation Methods

There were no duplicates in the data set so the treatment of the data focused on treating the missing values and outliers. To treat missing values univariate statistical imputation was completed by calculating the mean median, or mode. This technique was used for replacing the missing values in order to retain most of the data from the original data set (Lacrose & Lacrose, 2019).

The function df.isnull().sum() was then ran again to verify missing data was removed. Result:

Unnamed: 0 0

CaseOrder 0

Customer\_id 0

Interaction 0

City 0

State 0

County 0

Zip 0

Lat 0

Lng 0

Population 0

Area 0

Timezone 0

Job 0

Children 0

Age 0

Education 0

Employment 0

Income 0

Marital 0

Gender 0

Churn 0

Outage\_sec\_perweek 0

Email 0

Contacts 0

Yearly\_equip\_failure 0

Techie 0

Contract 0

Port\_modem 0

Tablet 0

InternetService 0

Phone 0

Multiple 0

OnlineSecurity 0

OnlineBackup 0

DeviceProtection 0

TechSupport 0

StreamingTV 0

StreamingMovies 0

PaperlessBilling 0

PaymentMethod 0

Tenure 0

MonthlyCharge 0

Bandwidth\_GB\_Year 0

item1 0

item2 0

item3 0

item4 0

item5 0

item6 0

item7 0

item8 0

Z\_Score\_Population 0

Z\_Score\_Children 10000

dtype: int64

### After using box plots and z-scores to identify the outliers the Inter Quantile Range (IQR) was used to treat the outliers. The IQR is a measure of variability based on dividing a set into quartiles. The IQR is the range between the first quartile (25th percentile) and the third quartile (75th percentile) of the data (D, 2022).. The IQR identifies values that are significantly higher or lower than the majority of the data (D, 2022). The ICR was computed on the following variables Population, Children, Income, Outage\_sec\_perweek, Email, Contacts, Yearly\_equip\_failure, and MonthlyCharge due to data shown on the box plots as outliers. The following code was used to determine the outliers in each variable (‘population’ is shown below, all other variables seen in code attached).

import numpy as np

outliers = []

def detect\_outliers\_iqr(data):

data = sorted(data)

q1 = np.percentile(data, 25)

q3 = np.percentile(data, 75)

IQR = q3-q1

lwr\_bound = q1-(1.5\*IQR)

upr\_bound = q3+(1.5\*IQR)

for i in data:

if (i<lwr\_bound or i>upr\_bound):

outliers.append(i)

return outliers

Population\_column = df['Population']

Population\_outliers = detect\_outliers\_iqr(Population\_column)

print("Outliers from IQR method: ", Population\_outliers)

A new array was then created for each variable. Example code below. New arrays for each variable seen in code attached.

tenth\_percentile = np.percentile(df['Population'], 10)

ninetieth\_percentile = np.percentile(df['Population'], 90)

b = np.where(df['Population']<tenth\_percentile, tenth\_percentile, df['Population'])

b = np.where(b>ninetieth\_percentile, ninetieth\_percentile, b)

print("New array:",b)

Once a new array is created a box plot was then created for each variable to verify the removal of all outliers.

df['Population\_without\_outliers'] = b

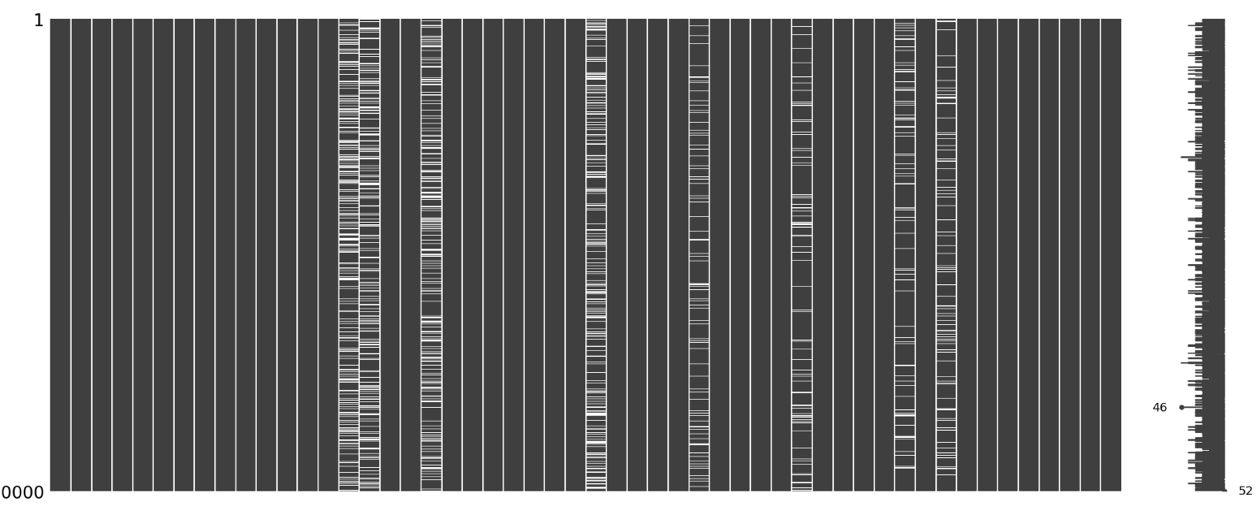
plt.figure(figsize=(8,3))

seaborn.boxplot(x=df['Population\_without\_outliers'])

plt.show()

### Section D3: Summary of the Outcomes

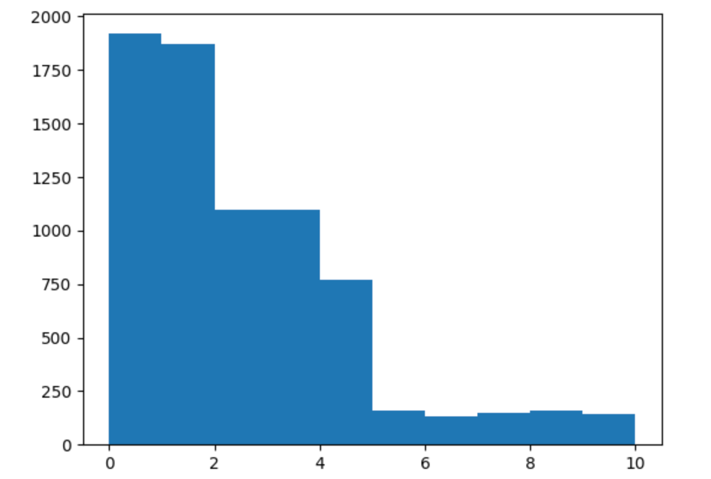
Visualizations are used to get a better view of duplicate data, missing data, and outliers. Visuals allow one to quickly identify patterns and trends. Visuals also easily detect the presence of unusual or unexpected values. Missing data was visualized with function: msno.matrix(df) prior to the treatment of the data and after the treatment of the missing data to verify no missing data remained in the data set.



Background pattern

Description automatically generated

After determining the location of the missing data the treatment of the missing values could be done. Missing data was treated by replacing them with mean or medium. Each variable could be visualized with a histogram to view the distribution of the data variable. A histogram was created for the variable ‘children’, ‘age’, and ‘income’. The original distribution of the variable was viewed prior to and after the removal of the missing data. This was done to ensure the distribution of data was maintained after treatment.

‘Children’:

Chart, histogram

Description automatically generated

‘Age’:

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

### 

Chart, histogram

Description automatically generated‘Income’ :Chart, histogram

Description automatically generated

### Outliers were then detected and treated. Outliers could not be adequately detected prior to addressing missing data because the presence of missing values could affect the calculation of measures of central tendency and could lead to a distortion of the data and affect the imputation methods used to fill in the missing values (Bonthu, 2022). Outliers were detected using box plots. Once box plots were created and outliers were visible z-cores were then calculated. From the z-scores, the IQR was calculated to remove outliers. An example of this can be seen by examining the variables ‘children’, ‘income’ and ‘population’. before and after outliers are removed.

‘Children’:

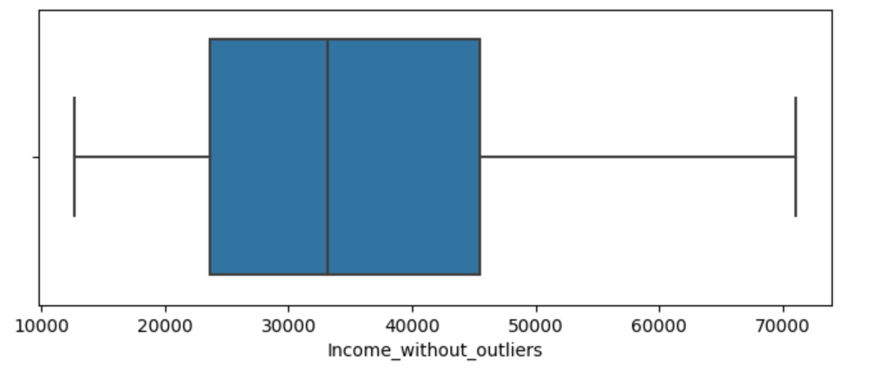
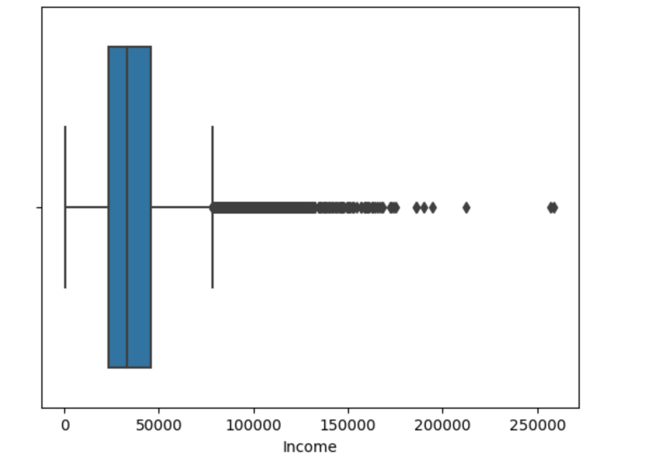
Chart, box and whisker chart

Description automatically generated

Chart

Description automatically generated

‘Income’:



‘Population’:

Chart, box and whisker chart

Description automatically generated

Chart

Description automatically generated

### Section D4: Mitigation Code

The following functions were used for the treatment of missing data, and outliers. See detailed code attached.

1. *Treat the missing values:*

df['Children'].fillna(df['Children'].median(), inplace=True)

df['Age'].fillna(df['Age'].mean(),inplace=True)

df['Income'].fillna(df['Income'].median(), inplace=True)

df['Tenure'].fillna(df['Tenure'].mean(),inplace=True)

df['Bandwidth\_GB\_Year'].fillna(df['Bandwidth\_GB\_Year'].mean(),inplace=True)

df['Techie'].fillna(df['Techie'].mode().iat[0], inplace=True)

df['Phone'].fillna(df['Phone'].mode().iat[0], inplace=True)

df['TechSupport'].fillna(df['TechSupport'].mode().iat[0], inplace=True)

1. *Code to view histograms before & after removing missing data:*

#plt.hist(df['Children'])

plt.hist(df['Children'])

plt.xlabel('Children')

plt.ylabel('Frequency')

plt.title('Histogram of Children after removing missing values')

plt.show()

#plt.hist(df['Age'])

plt.hist(df['Age'])

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.title('Histogram of Age after removing missing values')

plt.show()

#plt.hist(df['Income'])

plt.hist(df['Income'])

plt.xlabel('Income')

plt.ylabel('Frequency')

plt.title('Histogram of Age after removing missing values')

plt.show()

1. Verifying missing data:

msno.matrix(df)

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

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

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

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

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

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

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

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

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

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

1. Determining z-scores:

df['Z\_Score\_Population'] = stats.zscore(df['Population'])

df[['Population','Z\_Score\_Population']].head()

zscores = df[['Population', 'Z\_Score\_Population']]

data = df[['Population']]

num\_outliers, outliers = count\_and\_value\_outliers(zscores, data)

print("Number of outliers:", num\_outliers)

print("Outlier values:", outliers)

df['Z\_Score\_Children'] = stats.zscore(df['Children'])

df[['Children','Z\_Score\_Children']].head()

df\_outliers = df.query('(Z\_Score\_Children > 3) | (Z\_Score\_Children < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Children'].to\_dict().values())

df['Z\_Score\_Income'] = stats.zscore(df['Income'])

df[['Children','Z\_Score\_Income']].head()

df\_outliers = df.query('(Z\_Score\_Income > 3) | (Z\_Score\_Income < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Income'].to\_dict().values())

df['Z\_Score\_Outage\_sec\_perweek'] = stats.zscore(df['Outage\_sec\_perweek'])

df[['Children','Z\_Score\_Outage\_sec\_perweek']].head()

df\_outliers = df.query('(Z\_Score\_Outage\_sec\_perweek > 3) | (Z\_Score\_Outage\_sec\_perweek < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Outage\_sec\_perweek'].to\_dict().values())

df['Z\_Score\_Email'] = stats.zscore(df['Email'])

df[['Email','Z\_Score\_Email']].head()

df\_outliers = df.query('(Z\_Score\_Email > 3) | (Z\_Score\_Email < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Email'].to\_dict().values())

df['Z\_Score\_Contacts'] = stats.zscore(df['Contacts'])

df[['Contacts','Z\_Score\_Contacts']].head()

df\_outliers = df.query('(Z\_Score\_Contacts > 3) | (Z\_Score\_Contacts < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Contacts'].to\_dict().values())

df['Z\_Score\_Yearly\_equip\_failure'] = stats.zscore(df['Yearly\_equip\_failure'])

df[['Yearly\_equip\_failure','Z\_Score\_Yearly\_equip\_failure']].head()

df\_outliers = df.query('(Z\_Score\_Yearly\_equip\_failure > 3) | (Z\_Score\_Yearly\_equip\_failure < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['Yearly\_equip\_failure'].to\_dict().values())

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

df['Z\_Score\_MonthlyCharge'] = stats.zscore(df['MonthlyCharge'])

df[['MonthlyCharge','Z\_Score\_MonthlyCharge']].head()

df\_outliers = df.query('(Z\_Score\_MonthlyCharge > 3) | (Z\_Score\_MonthlyCharge < -3)')

print("Number of outliers:", df\_outliers.shape[0])

print("Outlier values:", df\_outliers['MonthlyCharge'].to\_dict().values())

1. *Treating Outliers Ihe following was the treatment of the ‘Population’ variable only. See attached code for all other variables with outliers.* (D, 2022)*.*

outliers = []

def detect\_outliers\_iqr(data):

data = sorted(data)

q1 = np.percentile(data, 25)

q3 = np.percentile(data, 75)

IQR = q3-q1

lwr\_bound = q1-(1.5\*IQR)

upr\_bound = q3+(1.5\*IQR)

for i in data:

if (i<lwr\_bound or i>upr\_bound):

outliers.append(i)

return outliers

Population\_column = df['Population']

Population\_outliers = detect\_outliers\_iqr(Population\_column)

print("Outliers from IQR method: ", Population\_outliers)

tenth\_percentile = np.percentile(df['Population'], 10)

ninetieth\_percentile = np.percentile(df['Population'], 90)

b = np.where(df['Population']<tenth\_percentile, tenth\_percentile, df['Population'])

b = np.where(b>ninetieth\_percentile, ninetieth\_percentile, b)

print("New array:",b)

df['Population\_without\_outliers'] = b

plt.figure(figsize=(8,3))

seaborn.boxplot(x=df['Population\_without\_outliers'])

plt.show()

### Section D5: Cleaning Data

### 

See the attached file of the cleaned data set.

Code used to extract file:

df.to\_csv(r'/Users/igmark/Desktop/WGU Data Files/churn\_clean\_data.csv’)

### Section D6: Limitations

Data cleaning is an important step in the data analysis process. This step is required to obtain an accurate picture of data outcomes. However, data cleaning can have limitations. One limitation of the data learning process is that it is time consuming. The data cleaning process requires a significant amount of time and resources to identify and clean data issues (Lacrose & Lacrose, 2019). Human error is also a limitation of data cleaning. Human error is related to a lack of standardization and subjectivity. There is no set methodology to clean data. Different methods can be used by different analysts and different data sets required different methods for cleaning data.

A limitation of utilizing the imputation method in replacing missing values includes possible distortion or distribution of the data (Lacrose & Lacrose, 2019). Limitations in removing outliers after detection could significantly reduce the sample size and eliminate data points from the observation that may be considered valuable (Lacrose & Lacrose, 2019).

### Section D7.  Impact of the Limitations

Challenges data analysts may encounter if they were to take the recently cleaned churn data set to analyze include getting inaccurate and inconsistent results. Human error can lead to incorrectly cleaning data. Due to the cleaning process not being standardized data cleaning can be subjective and lead to inconsistencies in results. Removing outliers and/or using imputation methods can lead to the loss of or incomplete data can limit the scope of the data.

## Part IV: PCA

### Section E1: Principal Components

Principal component analysis (PCA) is a statistical technique used to analyze the structure of a dataset by identifying patterns in the data (Mirshra et al. 2016). The goal of PCA is to identify a new set of uncorrelated variables that can explain variability in the data. PCA can only be completed with quantitative variables after missing data has been treated.

*Code used for analysis:*

from sklearn.decomposition import PCA

test\_pca = df[['Age', 'Children','Income', 'Population', 'Outage\_sec\_perweek', 'Email', 'Contacts', 'Tenure', 'Bandwidth\_GB\_Year', 'Yearly\_equip\_failure']]

test\_pca\_normalized=(test\_pca-test\_pca.mean())/test\_pca.std()

pca = PCA(n\_components=test\_pca.shape[1])

pca.fit(test\_pca\_normalized)

PCA(n\_components=11)

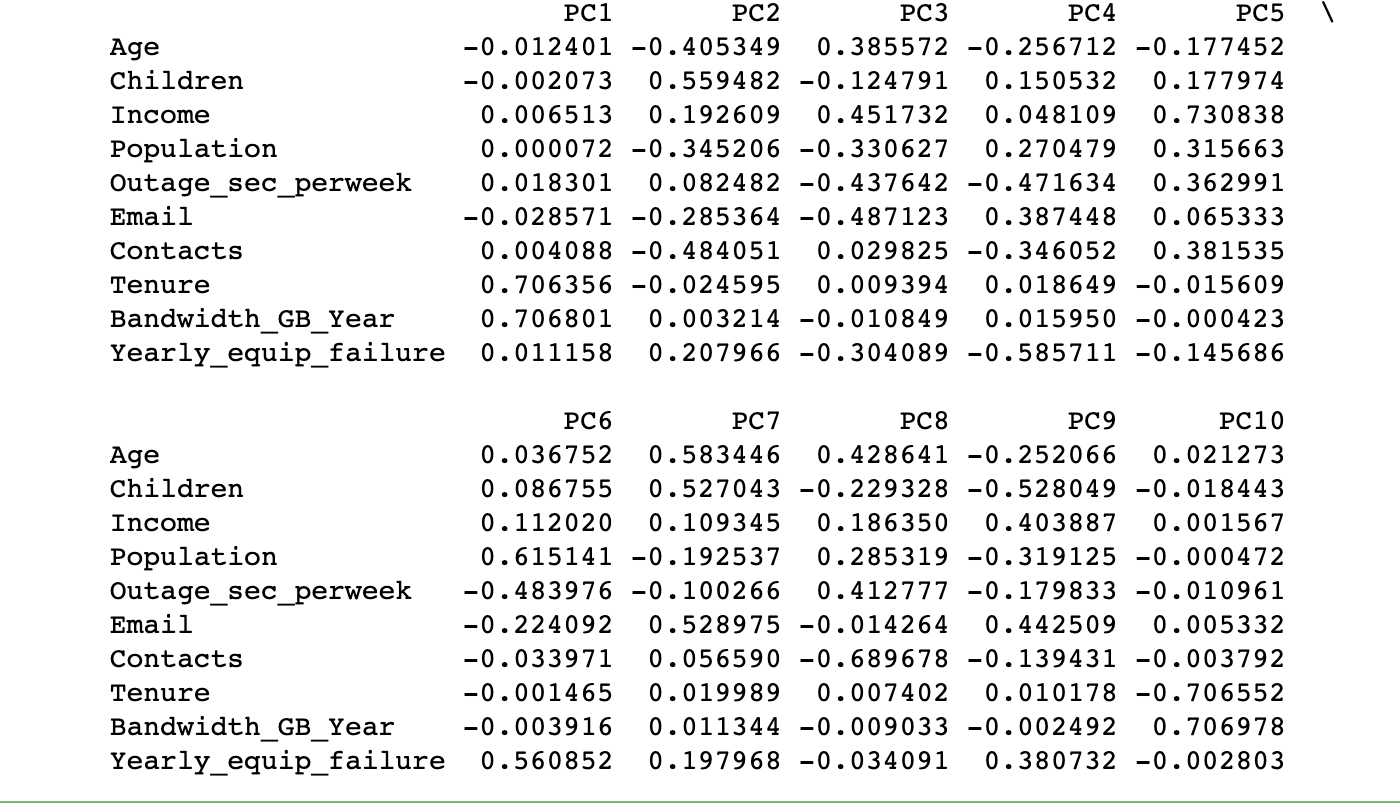
test\_pca2=pd.DataFrame(pca.transform(test\_pca\_normalized),columns=['PC1','PC2','PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10'])

loadings = pd.DataFrame(pca.components\_.T, columns=['PC1','PC2','PC3','PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10'], index=test\_pca\_normalized.columns)

print(test\_pca2)

print(loadings)

*PCA loading matrix:*



### Section E2: Criteria Used

According to the Kaiser Rule, the number of components to retain from PCA is equal to the number of eigenvalues greater than one (Lacrose & Lacrose, 2019). An eigenvalue greater than one suggests that the corresponding principal component captures meaningful information from the data. PCA below one is not significant. The larger the PCA the more significant it is in explaining variations in the dataset. A visual of the PCA using eigenvalues can be seen on a scree plot. Thus PCA 1, 2, and 3 should be kept.

*Code for Eigenvalues:*

cov\_matrix=np.dot(test\_pca\_normalized.T, test\_pca\_normalized) / test\_pca.shape[0]

eigenvalues=[np.dot(eigenvector.T, np.dot(cov\_matrix, eigenvector))for eigenvector in pca.components\_]

plt.plot(eigenvalues)

plt.xlabel('number of components')

plt.ylabel('eigenvalues')

plt.axhline(y=1, color="red")

plt.show()

*Scree Plot:*

Chart, line chart

Description automatically generated

### Section E3: Benefits

PCA can be beneficial in examining patterns in a data set. One benefit of PCA is that it reduces the dimensionally of a dataset leading to easier visualizations and interpretation. PCA removed noise and irrelevant information from the data resulting in a cleaner data set (Mirshra et al. 2016). PCA also results in the ability to see uncorrelated variables. The scree plot indicates PC 1, 2, and 3 are the most significant or correlated. The PCA loading matrix shows the principal components that have positive and negative relationships between variables. The principal component and corresponding variables with a positive relationship are correlated. The variables that are contributed to each principal component are listed below:

|  |  |
| --- | --- |
| **Variable** | **Contributing PC** |
| Age | PC3 |
| Children | PC2 |
| Income | PC1, PC2, PC3 |
| Population | PC1 |
| Outage\_sec\_perweek | PC1, PC2 |
| Email | PC2 |
| Contacts | PC1, PC3 |
| Tenure | PC1, PC3 |
| Bandwidth\_GB\_Year | PC1, PC2, PC3 |
| Yearly\_equip\_failure | PC1, PC2 |

### Section F: Panopto Video

*Link for Panopto Video:*

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=5de3730a-c507-4b28-a208-afa701616723#

### Section G: Third-Party Code References

Bonthu, H. (2022, November 30). *Detecting and Treating Outliers | Treating the odd one out!* Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/>

D, V. A. (2022, October 10). *Python Boxplot – How to create and interpret boxplots (also find outliers and summarize distributions)*. Machine Learning Plus. https://www.machinelearningplus.com/plots/python-boxplot/

### Section H: References

Larose, C. D., & Larose, D. T. (2019).Data science using Python and R. John Wiley& Sons. ISBN: 978-1-119-52684-1

Mishra, S. P., Sarkar, U. K., Taraphder, S., Datta, S. K., Swain, D. P., Saikhom, R., Panda, S., & Laishram, M. (2016). Multivariate Statistical Data Analysis- Principal Component Analysis (PCA) -. *International Journal of Livestock Research*, *7*(5), 60–78. https://www.bibliomed.org/?mno=261590