|  |
| --- |
| Data Cleaning |
| D206 |

|  |
| --- |
| Shantel Johnson  3-16-2023 |

Contents

[Part One: Research Question 4](#_Toc129866975)

[Section A: Do customers enrolled in more service offerings display lower levels of churn? 4](#_Toc129866976)

[Relevance 4](#_Toc129866977)

[Section B: Data Dictionary 4](#_Toc129866978)

[Part Two: Data Cleaning Plan 6](#_Toc129866979)

[Section C.1: Techniques & Steps 6](#_Toc129866980)

[Techniques 6](#_Toc129866981)

[Duplicate Rows 7](#_Toc129866982)

[Missing Values 7](#_Toc129866983)

[Outliers 7](#_Toc129866984)

[Section C.2: Justification of Methods 7](#_Toc129866985)

[Duplicate Rows 7](#_Toc129866986)

[Missing Values 7](#_Toc129866987)

[Outliers 7](#_Toc129866988)

[Section C.3: Programming Environment 8](#_Toc129866989)

[Programming Language 8](#_Toc129866990)

[Libraries 8](#_Toc129866991)

[Integrated Development Environment 8](#_Toc129866992)

[Section C.4: Code 8](#_Toc129866993)

[Duplicated Rows 11](#_Toc129866994)

[Missing Values 11](#_Toc129866995)

[Outliers 14](#_Toc129866996)

[Part Three: Data Cleaning 22](#_Toc129866997)

[Section D.1: Data Quality Findings 22](#_Toc129866998)

[Duplicates 22](#_Toc129866999)

[Missing Values 23](#_Toc129867000)

[Outliers 23](#_Toc129867001)

[Section D.2: Methods 23](#_Toc129867002)

[Duplicates 23](#_Toc129867003)

[Missing Values 23](#_Toc129867004)

[Outliers 24](#_Toc129867005)

[Section D.3: Code 25](#_Toc129867006)

[Handling Missing Values 25](#_Toc129867007)

[Handling Outliers 29](#_Toc129867008)

[Re-Expression of Categorical Variables 31](#_Toc129867009)

[Principal Component Analysis 36](#_Toc129867010)

[Scree Plot 40](#_Toc129867011)

[Section D.4: Verification and Outcome 40](#_Toc129867012)

[Duplicates 40](#_Toc129867013)

[Missing Values 40](#_Toc129867014)

[Outliers 41](#_Toc129867015)

[Section D.5: Output File 41](#_Toc129867016)

[Section D.6: Limitations 41](#_Toc129867017)

[Duplicates 41](#_Toc129867018)

[Missing Values 41](#_Toc129867019)

[Outliers 42](#_Toc129867020)

[Section D.7: Effect on the Research Question 42](#_Toc129867021)

[Section E.1: Principal Component Analysis 42](#_Toc129867022)

[Section E.2: Kaiser Rule 42](#_Toc129867023)

[Section E.3: Benefits of PCA 43](#_Toc129867024)

[Part Four: Supporting Documents 43](#_Toc129867025)

[Section F: Panopto Video 43](#_Toc129867026)

[Section G: Web Sources 43](#_Toc129867027)

[Section H: References 43](#_Toc129867028)

Table of Figures

[Figure 1: Matrix of Missing Values 13](#_Toc129866839)

[Figure 2: Histograms of Children, Age, Income, Tenure, and Bandwidth\_GB\_Year 14](#_Toc129866840)

[Figure 3: Boxplot of Children 15](#_Toc129866841)

[Figure 4: Boxplot of Age 15](#_Toc129866842)

[Figure 5: Boxplot of Income 16](#_Toc129866843)

[Figure 6: Boxplot of Outage\_sec\_perweek 16](#_Toc129866844)

[Figure 7: Boxplot of Email 17](#_Toc129866845)

[Figure 8: Boxplot of Contacts 17](#_Toc129866846)

[Figure 9: Boxplot of Yearly\_equip\_failure 18](#_Toc129866847)

[Figure 10: Boxplot of Tenure 18](#_Toc129866848)

[Figure 11: Boxplot of MonthlyCharge 19](#_Toc129866849)

[Figure 12: Boxplot of Bandwidth\_GB\_Year 20](#_Toc129866850)

[Figure 13: Histogram of Age 27](#_Toc129866851)

[Figure 14: Histograms of Children, Income, Tenure, and Bandwidth\_GB\_Year 28](#_Toc129866852)

[Figure 15: Matrix of Missing Values 29](#_Toc129866853)

[Figure 16: Boxplot of MonthlyCharge 30](#_Toc129866854)

[Figure 17: Boxplot of Email 30](#_Toc129866855)

[Figure 18: Scree Plot 40](#_Toc129866856)

Part One: Research Question[¶](" \l "Part-One:-Research-Question)

## Section A: Do customers enrolled in more service offerings display lower levels of churn?

The dataset for this project consists of AnyTelecom Company's observations on customers and customer churn. As the project's data analyst, my task is to prepare the data for exploratory data analysis and ultimately answer the following research question: Do customers enrolled in more service offerings display lower levels of churn?

### Relevance[¶](" \l "Relevance)

Churn (the measure of customers who stop using a product or service) can have a significant impact on a company's business performance. It is in AnyTelecom Company's best interest to examine the potential relationship between the number of service offerings a customer is enrolled in and the rate of churn.

## Section B: Data Dictionary[¶](" \l "Section-B:-Data-Dictionary)

|  | **Variable Name** | **Data Type** | **Description** | **Example** |
| --- | --- | --- | --- | --- |
| **0** | CaseOrder | Quantitative | preserves order of raw data | 1 |
| **1** | Customer\_id | Qualitative | unique customer id | K409198 |
| **2** | Interaction | Qualitative | unique id associated with customer transactions | aa90260b-4141-4a24-8e36-b04ce1f4f77b |
| **3** | City | Qualitative | city of residence | Point Baker |
| **4** | State | Qualitative | state of residence | AK |
| **5** | County | Qualitative | county of residence | Prince of Wales-Hyder |
| **6** | Zip | Quantitative | zip code of residence | 99927 |
| **7** | Lat | Quantitative | gps latitude coordinates | 56.251 |
| **8** | Lng | Quantitative | gps longitude coordinates | 56.251 |
| **9** | Population | Quantitative | population within mile radius | 38 |
| **10** | Area | Quantitative | area type | Urban |
| **11** | TimeZone | Qualitative | customer time zone | America/Sitka |
| **12** | Job | Qualitative | customer job | Environmental health practitioner |
| **13** | Children | Qualitative | number of children in household | NaN |
| **14** | Age | Quantitative | customer age | 68 |
| **15** | Education | Quantitative | highest degree earned by customer | Master's Degree |
| **16** | Employment | Qualitative | customer employment status | Part Time |
| **17** | Income | Qualitative | customer annual income | 28561.99 |
| **18** | Marital | Quantitative | customer marital status | Widowed |
| **19** | Gender | Qualitative | customer gender identification | Male |
| **20** | Churn | Qualitative | whether service discontinued in last month | No |
| **21** | Outage\_sec\_perweek | Qualitative | system outages seconds per week | 6.972566093 |
| **22** | Email | Quantitative | number of emails sent to customer in past year | 10 |
| **23** | Contacts | Quantitative | number of times customer contacted technical s... | 0 |
| **24** | Yearly\_equip\_failure | Quantitative | number of times customer equipment failed in p... | 1 |
| **25** | Techie | Quantitative | customer technical aptitude | No |
| **26** | Contract | Qualitative | contract term | One year |
| **27** | Port\_modem | Qualitative | customer has portable modem | Yes |
| **28** | Tablet | Qualitative | customer owns tablet | Yes |
| **29** | InternetService | Qualitative | customer internet service type | Fiber Optic |
| **30** | Phone | Qualitative | customer phone service | Yes |
| **31** | Multiple | Qualitative | customer has multiple lines | No |
| **32** | OnlineSecurity | Qualitative | customer has online security | Yes |
| **33** | OnlineBackup | Qualitative | customer has online backup | Yes |
| **34** | DeviceProtection | Qualitative | customer has device protection | No |
| **35** | TechSupport | Qualitative | customer has technical support | No |
| **36** | StreamingTV | Qualitative | customer has tv streaming | No |
| **37** | StreamingMovies | Qualitative | customer has movie streaming | Yes |
| **38** | PaperlessBilling | Qualitative | customer has paperless billing | Yes |
| **39** | PaymentMethod | Qualitative | customer payment method | Credit Card (automatic) |
| **40** | Tenure | Qualitative | number of months customer stayed with provider | 6.795512947 |
| **41** | MonthlyCharge | Quantitative | amount charged to customer monthly | 171.4497621 |
| **42** | Bandwidth\_GB\_Year | Quantitative | average amount of data used in past year | 904.5361102 |
| **43** | Item1 | Quantitative | timely response survey result | 5 |
| **44** | Item2 | Quantitative | timely fixes survey result | 5 |
| **45** | Item3 | Quantitative | timely replacements survey result | 5 |
| **46** | Item4 | Quantitative | reliability survey result | 3 |
| **47** | Item5 | Quantitative | options survey result | 4 |
| **48** | Item6 | Quantitative | respectful response survey result | 4 |
| **49** | Item7 | Quantitative | courteous exchange survey result | 3 |

Part Two: Data Cleaning Plan

## Section C.1: Techniques & Steps[¶](" \l "Section-C.1:-Techniques-&-Steps)

### Techniques[¶](" \l "Techniques)

The quality of the data will be assessed using the following techniques:

* Duplicate Detection
* Missing Value Detection
* Outlier Detection

### Duplicate Rows[¶](" \l "Duplicate-Rows)

The pandas.DataFrame.duplicated() function will be applied on the dataset to identify 1) duplication across entire records and 2) duplication across Customer\_id values.

### Missing Values[¶](" \l "Missing-Values)

The pandas.DataFrame.isnull.sum() function will be applied to identify the number of missing values in each column of the dataset. Columns that are found to contain missing values will be visualized using the missingno.matrix() function.

### Outliers[¶](" \l "Outliers)

Outliers will be identified in two stages. First, boxplots will be used to visually identify outliers for relevant numeric variables in the dataset; these boxplots will be produced using the pandas.DataFrame.boxplot() function. Next, the number of outliers in each feature will be calculated by applying the scipy.stats.zscore() function on each column of the dataset.

## Section C.2: Justification of Methods[¶](" \l "Section-C.2:-Justification-of-Methods)

### Duplicate Rows[¶](" \l "Duplicate-Rows)

The pandas.DataFrame.duplicated() function returns a Series that describes which records in the dataset are duplicated and which records are not (pandas, 2023). If a record is found to be duplicated, these results can be queried to determine the record's index to drop the row from the dataset.

### Missing Values[¶](" \l "Missing-Values)

The pandas.DataFrame.isnull.sum() function returns the sum of missing values for each column in the dataset (pandas, 2023). The missingno.matrix() function provides a visualization of the distribution of missing values within a given column (Larose & Larose, 2019).

### Outliers[¶](" \l "Outliers)

Boxplots are used to represent the spread of data and allow one to visually assess the presence of outliers in a sample. The pandas.DataFrame.boxplot() function can be used to produce boxplots for numerical data in the dataset (pandas, 2023).

In statistics, z-scores describe the numeric relationship between a value and the mean. When combined with the standard cutoff value (+/-3), z-scores can also determine if a value is an outlier (Larose & Larose, 2019). The scipy.stats.zscore() function computes the z-score of each value in a sample (The SciPy Community, 2023).

## Section C.3: Programming Environment[¶](" \l "Section-C.3:-Programming-Environment)

### Programming Language[¶](" \l "Programming-Language)

Python will be used to clean the dataset. Python is an open-source, robust programming language that includes several libraries and packages used to perform data analytics, data science, data engineering, and machine learning.

### Libraries[¶](" \l "Libraries)

The following libraries/packages will be used in conjunction with Python:

|  |  |
| --- | --- |
| **Library/Package** | **Purpose** |
| pandas | data manipulation and analysis |
| missingno.matrix | visualization of missing values |
| scipy.stats | z-score computation |
| numpy.random | random number generation |
| scikit-learn.preprocessing | data standardization |
| scikit-learn.decomposition | data reduction |
| matplotlib.pyplot | plotting and visualization |

### Integrated Development Environment[¶](" \l "Integrated-Development-Environment)

JupyterLab will be used as the Integrated Development Environment. JupyterLab is a web-based, open-source data science tool that allows users to write and share code in the form of a notebook.

## Section C.4: Code[¶](" \l "Section-C.4:-Code)

In [2]:

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

# import pandas library

import pandas as pd

In [3]:

# read churn data into DataFrame

df = pd.read\_csv('source files/churn\_raw\_data.csv', header='infer')

In [4]:

# get shape (rows & columns) of the DataFrame

df.shape

Out[4]:

(10000, 52)

In [5]:

# get header and first 5 rows of the DataFrame

df.head(5)

Out[5]:

|  | **Unnamed: 0** | **CaseOrder** | **Customer\_id** | **Interaction** | **City** | **State** | **County** | **Zip** | **Lat** | **Lng** | **...** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 1 | K409198 | aa90260b-4141-4a24-8e36-b04ce1f4f77b | Point Baker | AK | Prince of Wales-Hyder | 99927 | 56.25100 | -133.37571 | ... |
| **1** | 2 | 2 | S120509 | fb76459f-c047-4a9d-8af9-e0f7d4ac2524 | West Branch | MI | Ogemaw | 48661 | 44.32893 | -84.24080 | ... |
| **2** | 3 | 3 | K191035 | 344d114c-3736-4be5-98f7-c72c281e2d35 | Yamhill | OR | Yamhill | 97148 | 45.35589 | -123.24657 | ... |
| **3** | 4 | 4 | D90850 | abfa2b40-2d43-4994-b15a-989b8c79e311 | Del Mar | CA | San Diego | 92014 | 32.96687 | -117.24798 | ... |
| **4** | 5 | 5 | K662701 | 68a861fd-0d20-4e51-a587-8a90407ee574 | Needville | TX | Fort Bend | 77461 | 29.38012 | -95.80673 | ... |

5 rows × 52 columns

In [6]:

# view information about the DataFrame

df.info()

<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)

memory usage: 4.0+ MB

### Duplicated Rows[¶](" \l "Duplicated-Rows)

In [7]:

# return duplicated rows

df[df.duplicated() == True]

Out[7]:

|  | **Unnamed: 0** | **CaseOrder** | **Customer\_id** | **Interaction** | **City** | **State** | **County** | **Zip** | **Lat** | **Lng** | **...** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

0 rows × 52 columns

In [8]:

# check for records with duplicates in Customer\_id column

df[df.duplicated(subset='Customer\_id') == True]

Out[8]:

|  | **Unnamed: 0** | **CaseOrder** | **Customer\_id** | **Interaction** | **City** | **State** | **County** | **Zip** | **Lat** | **Lng** | **...** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

0 rows × 52 columns

### Missing Values[¶](" \l "Missing-Values)

In [9]:

# find the sum of missing values for each column

df.isnull().sum()

Out[9]:

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 2495

Age 2475

Education 0

Employment 0

Income 2490

Marital 0

Gender 0

Churn 0

Outage\_sec\_perweek 0

Email 0

Contacts 0

Yearly\_equip\_failure 0

Techie 2477

Contract 0

Port\_modem 0

Tablet 0

InternetService 0

Phone 1026

Multiple 0

OnlineSecurity 0

OnlineBackup 0

DeviceProtection 0

TechSupport 991

StreamingTV 0

StreamingMovies 0

PaperlessBilling 0

PaymentMethod 0

Tenure 931

MonthlyCharge 0

Bandwidth\_GB\_Year 1021

item1 0

item2 0

item3 0

item4 0

item5 0

item6 0

item7 0

item8 0

dtype: int64

In [10]:

# install missingno using pip

# !pip install missingno

In [11]:

# Title: ResidentMario/missingno

# Author: Bilogur, et al.

# Date: 2018

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.1184723

# import missingno to visualize missing values

import missingno as msno

In [12]:

# isolate columns with missing values and add them to a new DataFrame

df\_missing = df.loc[:, df.isnull().sum() > 0]

df\_missing.isnull().sum()

Out[12]:

Children 2495

Age 2475

Income 2490

Techie 2477

Phone 1026

TechSupport 991

Tenure 931

Bandwidth\_GB\_Year 1021

dtype: int64

In [13]:

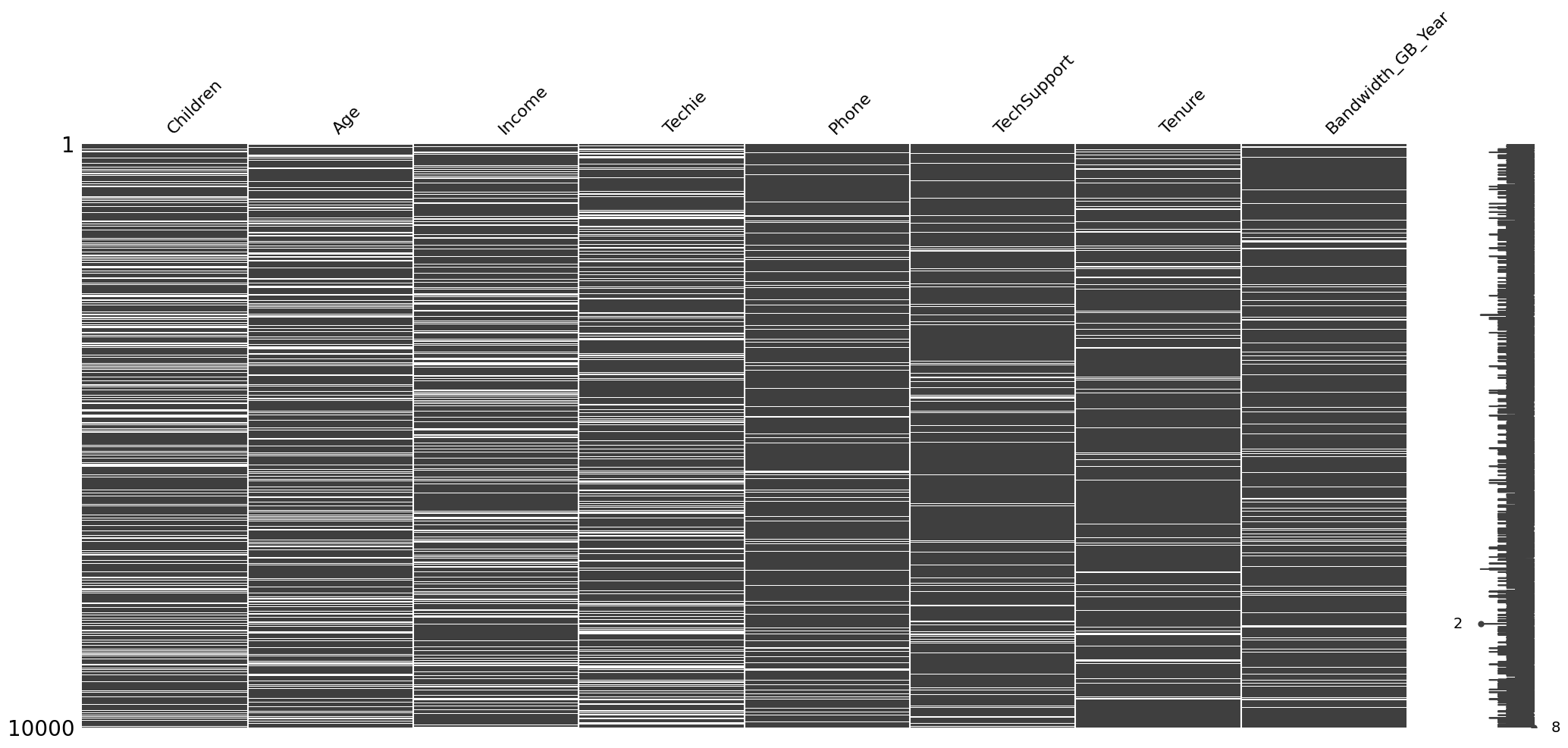
# generate matrix of missing values

msno.matrix(df\_missing)

Out[13]:

<Axes: >

Figure 1: Matrix of Missing Values



In [14]:

# use histograms to check columns in df\_missing for skewness

df\_missing.hist(figsize=(15,2), layout=(1,5))

Out[14]:

array([[<Axes: title={'center': 'Children'}>,

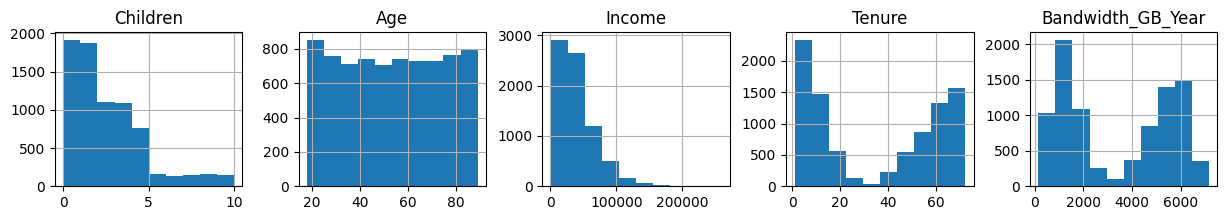
<Axes: title={'center': 'Age'}>,

<Axes: title={'center': 'Income'}>,

<Axes: title={'center': 'Tenure'}>,

<Axes: title={'center': 'Bandwidth\_GB\_Year'}>]], dtype=object)

Figure 2: Histograms of Children, Age, Income, Tenure, and Bandwidth\_GB\_Year



### Outliers[¶](" \l "Outliers)

In [15]:

# isolate columns with numerical data relevant for outlier analysis

df\_numericals = df[['Children',

'Age',

'Income',

'Outage\_sec\_perweek',

'Email',

'Contacts',

'Yearly\_equip\_failure',

'Tenure',

'MonthlyCharge',

'Bandwidth\_GB\_Year']]

df\_numericals.head(5)

Out[15]:

|  | **Children** | **Age** | **Income** | **Outage\_sec\_perweek** | **Email** | **Contacts** | **Yearly\_equip\_failure** | **Tenure** | **MonthlyCharge** | **…** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 68.0 | 28561.99 | 6.972566 | 10 | 0 | 1 | 6.795513 | 171.449762 | … |
| **1** | 1.0 | 27.0 | 21704.77 | 12.014541 | 12 | 0 | 1 | 1.156681 | 242.948015 | … |
| **2** | 4.0 | 50.0 | NaN | 10.245616 | 9 | 0 | 1 | 15.754144 | 159.440398 | … |
| **3** | 1.0 | 48.0 | 18925.23 | 15.206193 | 15 | 2 | 0 | 17.087227 | 120.249493 | … |
| **4** | 0.0 | 83.0 | 40074.19 | 8.960316 | 16 | 2 | 1 | 1.670972 | 150.761216 | … |

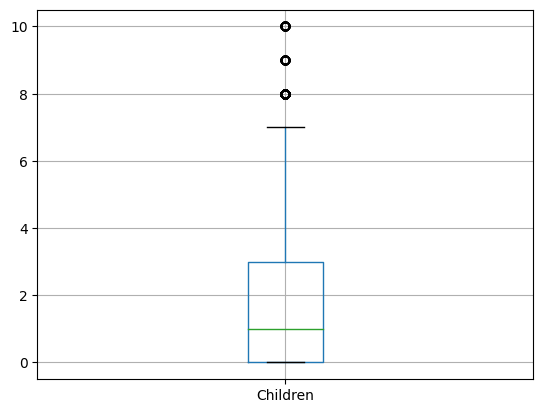
In [16]:

df\_numericals.boxplot(column='Children')

Out[16]:

<Axes: >

Figure 3: Boxplot of Children



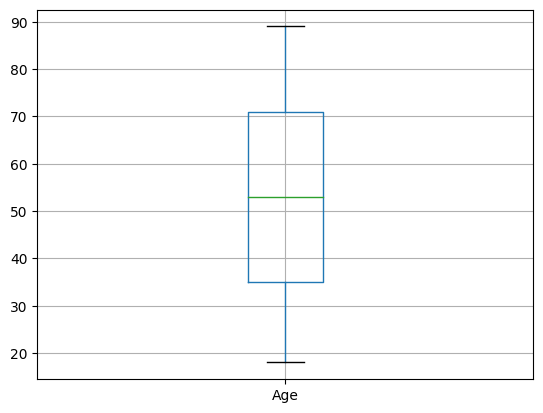
In [17]:

df\_numericals.boxplot(column='Age')

Out[17]:

<Axes: >

Figure 4: Boxplot of Age



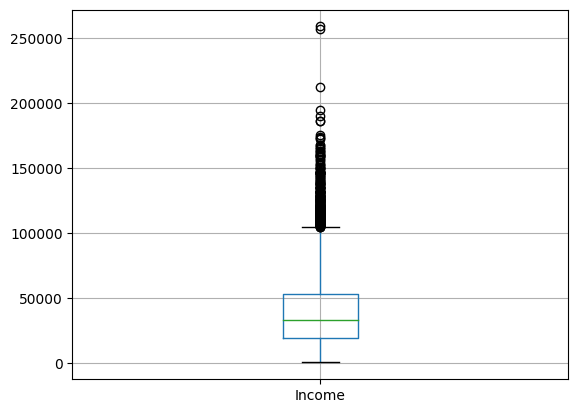
In [18]:

df\_numericals.boxplot(column='Income')

Out[18]:

<Axes: >

Figure 5: Boxplot of Income



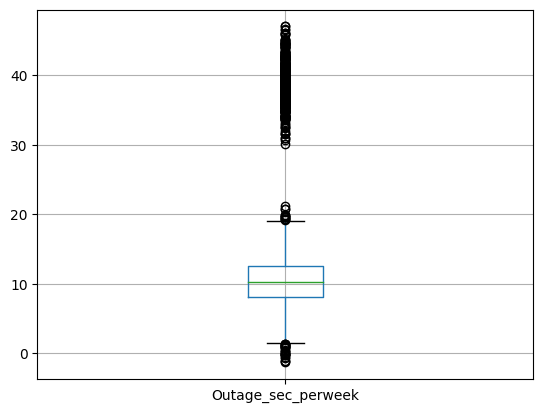
In [19]:

df\_numericals.boxplot(column='Outage\_sec\_perweek')

Out[19]:

<Axes: >

Figure 6: Boxplot of Outage\_sec\_perweek



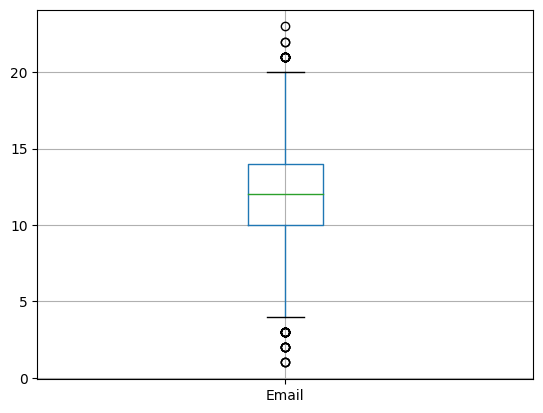
In [20]:

df\_numericals.boxplot(column='Email')

Out[20]:

<Axes: >

Figure 7: Boxplot of Email



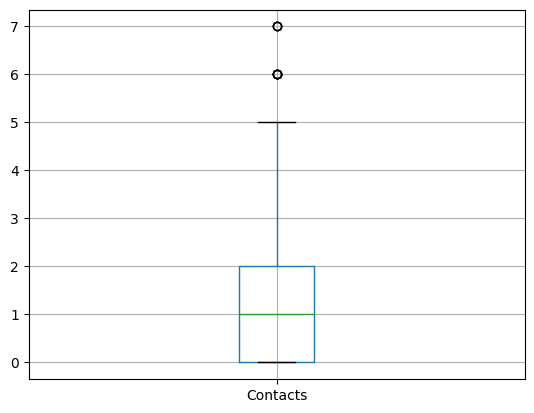
In [21]:

df\_numericals.boxplot(column='Contacts')

Out[21]:

<Axes: >

Figure 8: Boxplot of Contacts



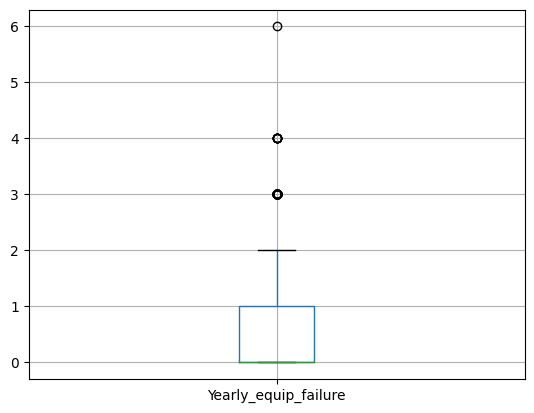
In [22]:

df\_numericals.boxplot(column='Yearly\_equip\_failure')

Out[22]:

<Axes: >

Figure 9: Boxplot of Yearly\_equip\_failure



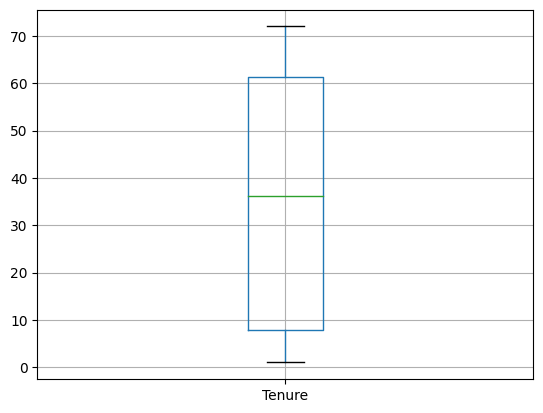
In [23]:

df\_numericals.boxplot(column='Tenure')

Out[23]:

<Axes: >

Figure 10: Boxplot of Tenure



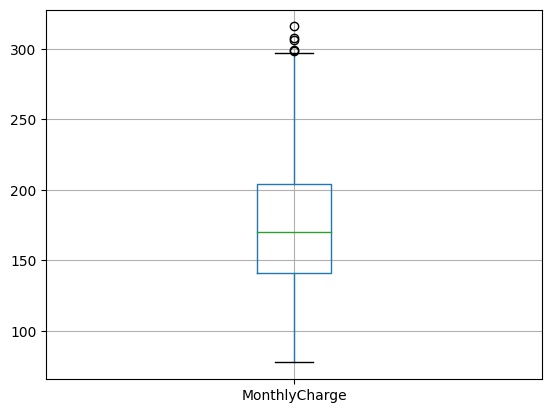
In [24]:

df\_numericals.boxplot(column='MonthlyCharge')

Out[24]:

<Axes: >

Figure 11: Boxplot of MonthlyCharge



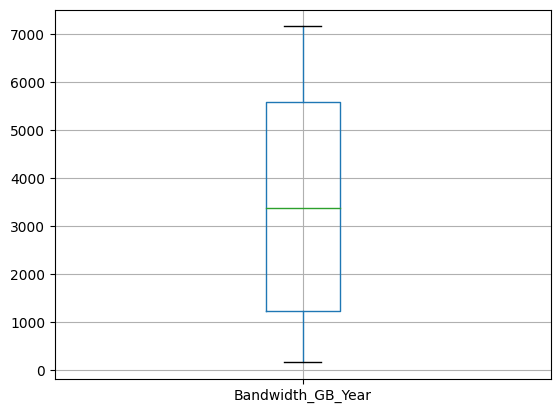
In [25]:

df\_numericals.boxplot(column='Bandwidth\_GB\_Year')

Out[25]:

<Axes: >

Figure 12: Boxplot of Bandwidth\_GB\_Year



In [26]:

# Title: scipy/scipy: Scipy

# Author: Gommers, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7655153

# import scipy.stats to calculate z-scores

from scipy import stats

In [27]:

# create a new DataFrame to hold z-scores for columns that contain outliers

data = {

'Children':[],

'Income':[],

'Outage\_sec\_perweek':[],

'Email':[],

'Contacts':[],

'Yearly\_equip\_failure':[],

'MonthlyCharge':[]

}

df\_zscores = pd.DataFrame(data)

In [28]:

# add z-scores to df\_zscores

df\_zscores['Children'] = stats.zscore(df\_numericals['Children'], nan\_policy='omit')

df\_zscores['Income'] = stats.zscore(df\_numericals['Income'], nan\_policy='omit')

df\_zscores['Outage\_sec\_perweek'] = stats.zscore(df\_numericals['Outage\_sec\_perweek'], nan\_policy='omit')

df\_zscores['Email'] = stats.zscore(df\_numericals['Email'], nan\_policy='omit')

df\_zscores['Contacts'] = stats.zscore(df\_numericals['Contacts'], nan\_policy='omit')

df\_zscores['Yearly\_equip\_failure'] = stats.zscore(df\_numericals['Yearly\_equip\_failure'], nan\_policy='omit')

df\_zscores['MonthlyCharge'] = stats.zscore(df\_numericals['MonthlyCharge'], nan\_policy='omit')

df\_zscores.head(5)

Out[28]:

|  | **Children** | **Income** | **Outage\_sec\_perweek** | **Email** | **Contacts** | **Yearly\_equip\_failure** | **MonthlyCharge** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | -0.401133 | -0.637726 | -0.666282 | -1.005852 | 0.946658 | -0.060613 |
| **1** | -0.508646 | -0.642955 | 0.079935 | -0.005288 | -1.005852 | 0.946658 | 1.589348 |
| **2** | 0.883715 | NaN | -0.171849 | -0.996779 | -1.005852 | 0.946658 | -0.337752 |
| **3** | -0.508646 | -0.740976 | 0.534225 | 0.986203 | 1.017588 | -0.625864 | -1.242158 |
| **4** | -0.972766 | 0.004846 | -0.354795 | 1.316700 | 1.017588 | 0.946658 | -0.538041 |

In [29]:

# return records where absolute value of z-score is greater than or equal to 3, Children column

children\_outliers = df\_zscores[df\_zscores['Children'].abs() > 3]['Children']

children\_outliers.count()

Out[29]:

144

In [30]:

# return records where absolute value of z-score is greater than or equal to 3, Income column

income\_outliers = df\_zscores[df\_zscores['Income'].abs() > 3]['Income']

income\_outliers.count()

Out[30]:

110

In [31]:

# return records where absolute value of z-score is greater than or equal to 3, Outage\_sec\_perweek column

outage\_outliers = df\_zscores[df\_zscores['Outage\_sec\_perweek'].abs() > 3]['Outage\_sec\_perweek']

outage\_outliers.count()

Out[31]:

491

In [32]:

# return records where absolute value of z-score is greater than or equal to 3, Email column

email\_outliers = df\_zscores[df\_zscores['Email'].abs() > 3]['Email']

email\_outliers.count()

Out[32]:

12

In [33]:

# return records where absolute value of z-score is greater than or equal to 3, Contacts column

contacts\_outliers = df\_zscores[df\_zscores['Contacts'].abs() > 3]['Contacts']

contacts\_outliers.count()

Out[33]:

165

In [34]:

# return records where absolute value of z-score is greater than or equal to 3, Yearly\_equip\_failure column

failure\_outliers = df\_zscores[df\_zscores['Yearly\_equip\_failure'].abs() > 3]['Yearly\_equip\_failure']

failure\_outliers.count()

Out[34]:

94

In [35]:

# return records where absolute value of z-score is greater than or equal to 3, MonthlyCharge column

charge\_outliers = df\_zscores[df\_zscores['MonthlyCharge'].abs() > 3]['MonthlyCharge']

charge\_outliers.count()

Out[35]:

3

Part Three: Data Cleaning[¶](" \l "Part-Three:-Data-Cleaning)

## Section D.1: Data Quality Findings[¶](" \l "Section-D.1:-Data-Quality-Findings)

### Duplicates[¶](" \l "Duplicates)

No duplicates were found.

### Missing Values[¶](" \l "Missing-Values)

Missing values were found for the following features:

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

### Outliers[¶](" \l "Outliers)

The following features contained outliers:

|  |  |
| --- | --- |
| **Feature** | **Number of Outliers** |
| Children | 144 |
| Income | 110 |
| Outage\_sec\_perweek | 491 |
| Email | 12 |
| Contacts | 165 |
| Yearly\_equip\_failure | 94 |
| MonthlyCharge | 3 |

## Section D.2: Methods[[1]](#footnote-1)[¶](" \l "Section-D.2:-Methods)

### Duplicates[¶](" \l "Duplicates)

Since no duplicates were found, it was not necessary to implement methods to handle duplicate values.

### Missing Values[¶](" \l "Missing-Values)

Depending on the data type and characteristics of the data, different methods were used to handle missing values.

#### Qualitative Data[¶](#Qualitative-Data)

Features of the qualitative data type with missing values were:

* Techie
* Phone
* TechSupport

As it is not possible to compute the mean or median on categorical data, missing values for these features were imputed using the mode (Larose & Larose, 2019). To do this, the pandas.DataFrame.fillna() function was applied on the Techie, Phone, and TechSupport columns. The pandas.DataFrame.fillna() function is used to fill NA/NaN values with user-specified values. In this case, NA/NaN values were replaced using the mode of each feature. To calculate the mode, the pandas.Series.mode() function was applied on each column (pandas, 2023).

#### Quantitative Data: Uniform Distribution[¶](#Quantitative-Data:-Uniform-Distribution)

Age was the only uniform feature of the quantitative data type with missing values. To preserve the shape of this feature, missing values were imputed using random values. This data cleaning step was performed in several phases. First, the numpy.random.randint() function was used to create an array of random integers within the range of Age. Next, a new DataFrame was created; this DataFrame contained the records where the Age variable was null. Finally, the random array of integers was inserted into the new DataFrame, and the values of the DataFrame were used to replace NA/NaN values in Age by applying the pandas.DataFrame.fillna() function on the column.

#### Quantitative Data: Skewed & Bi-Modal Distribution[¶](#Quantitative-Data:-Skewed-&-Bi-Modal-Di)

Features of the quantitative data type with missing values were:

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

To avoid further skewing the distribution of these features, the median, rather than the mean, was used to impute missing values for these features (Larose & Larose, 2019). The pandas.DataFrame.fillna() function was applied on the Children, Income, Tenure, and Bandwidth\_GB\_Year columns. NA/NaN values were replaced using the median of each feature, and this value was calculated by applying the pandas.Series.median() function on each column.

### Outliers[¶](" \l "Outliers)

If there was indication that a set of outliers were the result of error or illegitimate entry, records containing these outliers were removed from the dataset (Larose & Larose, 2019). Otherwise, outliers were retained.

#### Indication of Illegitimacy[¶](#Indication-of-Illegitimacy)

For this project, illegitimacy was determined based on the volume of outliers found in each feature. It was assumed that if a feature contained a sparse number of outliers, then those outliers were illegitimate. In Section C.4: Outliers, queries were performed to identify the number of outliers in each feature.

#### Illegitimate Outliers[¶](#Illegitimate-Outliers)

The features MonthlyCharge and Email contained relatively few outliers (3 and 12, respectively). Records containing these outliers were removed from the dataset. To do this, the pandas.DataFrame.drop() function was applied on the MonthlyCharge and Email columns for records that contained outliers.

#### Legitimate Outliers[¶](#Legitimate-Outliers)

Each of the following features contained more than 100 outliers:

* Children
* Income
* Outage\_sec\_perweek
* Email
* Contacts
* Yearly\_equip\_failure
* MonthlyCharge

These features were retained in the dataset.

## Section D.3: Code[¶](" \l "Section-D.3:-Code)

### Handling Missing Values[¶](" \l "Handling-Missing-Values)

In [36]:

# fill missing values for the Techie, Phone, and TechSupport using the mode

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

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

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

In [37]:

# verify Techie, Phone, and TechSupport do not contain missing values

df[['Techie', 'Phone', 'TechSupport']].isnull().sum()

Out[37]:

Techie 0

Phone 0

TechSupport 0

dtype: int64

In [38]:

# get rage of Age feature

df['Age'].describe()

Out[38]:

count 7525.000000

mean 53.275748

std 20.753928

min 18.000000

25% 35.000000

50% 53.000000

75% 71.000000

max 89.000000

Name: Age, dtype: float64

In [39]:

# Title: Numpy – Annual Update

# Author: Inessa Pawson

# Date: 2021

# Code Version: latest

# Availability: https://doi.org/10.25080/majora-1b6fd038-026

# import numpy to generate random integers

import numpy as np

In [40]:

# fill missing values for Age using random values within the range

age\_min = 18

age\_max = 90

size = df['Age'].isnull().sum()

# create random array using numpy

rand\_array = np.random.randint(age\_min, age\_max, size)

# create DataFrame to fillna with

values = df[df['Age'].isnull()][['Age']]

values.insert(0, 'rand\_int', rand\_array)

df['Age'].fillna(value=values['rand\_int'], inplace=True)

In [41]:

# verify Age does not contain missing values

df['Age'].isnull().sum()

Out[41]:

0

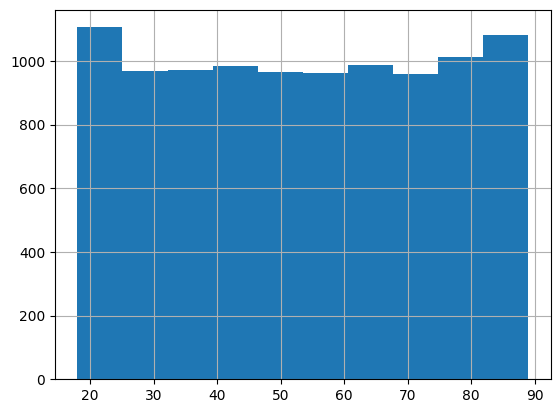
In [42]:

df['Age'].hist()

Out[42]:

<Axes: >

Figure 13: Histogram of Age



In [43]:

# fill missing values for Children, Income, Tenure, Bandwidth\_GB\_Year using the median

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

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

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

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

In [44]:

# verify Children, Income, Tenure, and Bandwidth\_GB\_Year do not contain missing values

df[['Children', 'Income', 'Tenure', 'Bandwidth\_GB\_Year']].isnull().sum()

Out[44]:

Children 0

Income 0

Tenure 0

Bandwidth\_GB\_Year 0

dtype: int64

In [45]:

df[['Children', 'Income', 'Tenure', 'Bandwidth\_GB\_Year']].hist()

Out[45]:

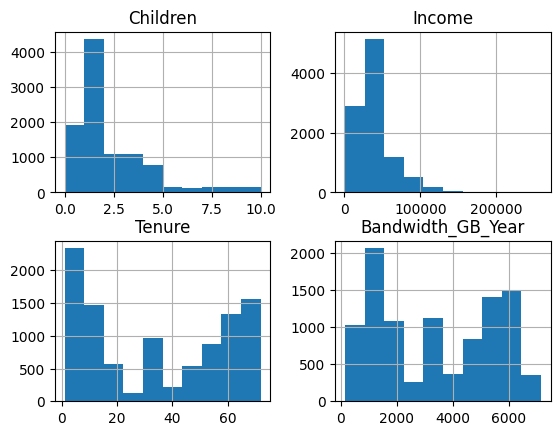
array([[<Axes: title={'center': 'Children'}>,

<Axes: title={'center': 'Income'}>],

[<Axes: title={'center': 'Tenure'}>,

<Axes: title={'center': 'Bandwidth\_GB\_Year'}>]], dtype=object)

Figure 14: Histograms of Children, Income, Tenure, and Bandwidth\_GB\_Year



In [46]:

# verify that all missing values have been filled

msno.matrix(df[['Children',

'Age',

'Income',

'Techie',

'Phone',

'TechSupport',

'Tenure',

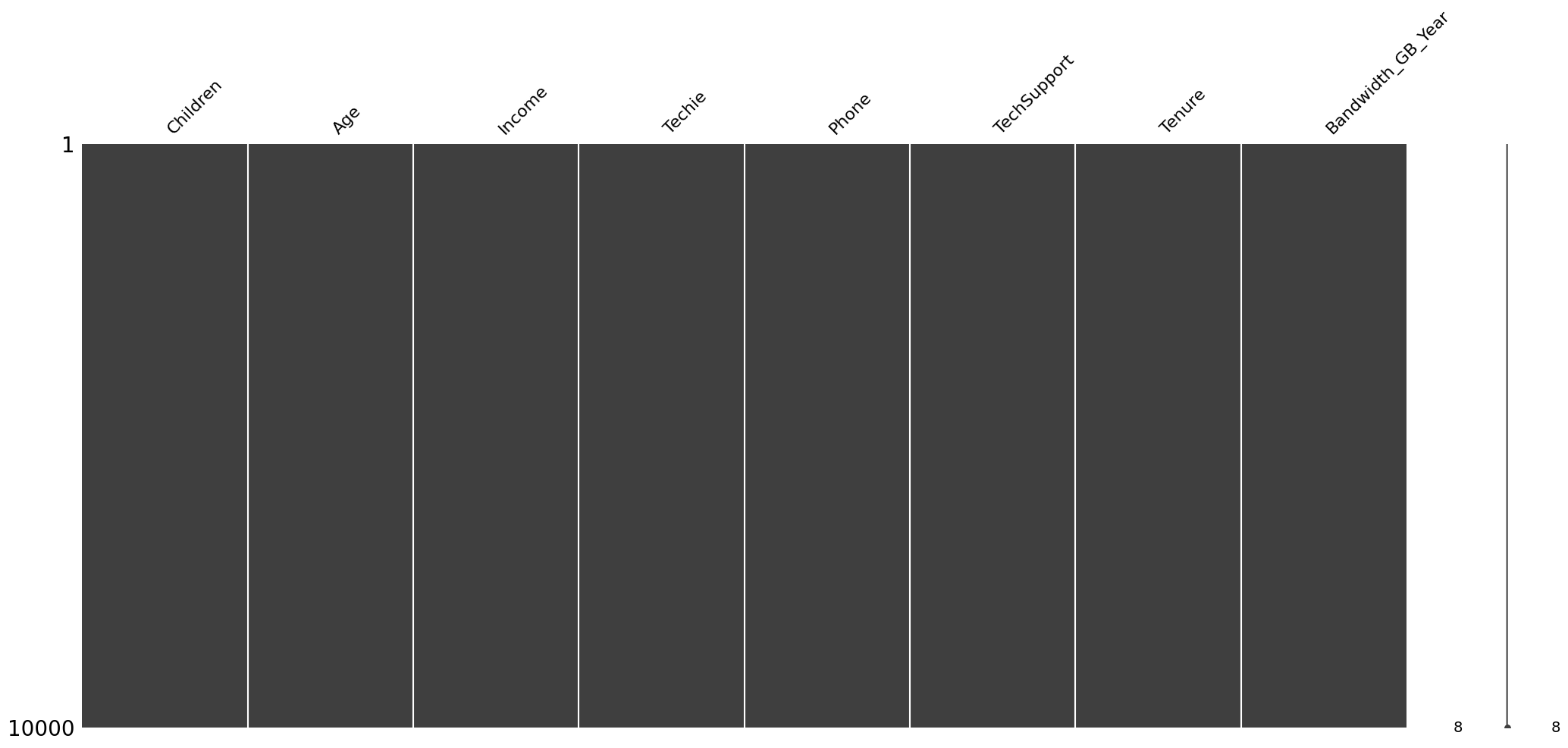
'Bandwidth\_GB\_Year'

]])

Out[46]:

<Axes: >

Figure 15: Matrix of Missing Values



### Handling Outliers[¶](" \l "Handling-Outliers)

In [47]:

# remove outliers from the MonthlyCharge and Email columns

df.drop(index=charge\_outliers.index, inplace=True)

df.drop(index=email\_outliers.index, inplace=True)

In [48]:

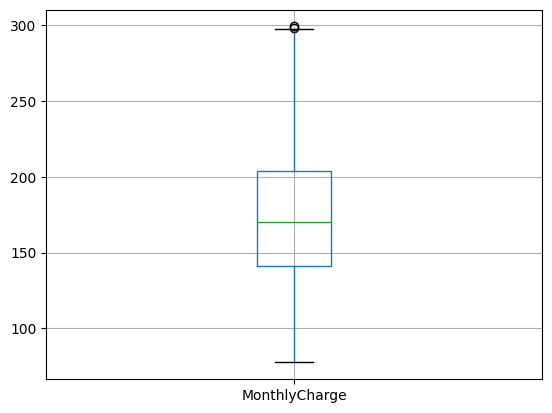
# verify outliers have been removed from MonthlyCharge

df[['MonthlyCharge']].boxplot()

Out[48]:

<Axes: >

Figure 16: Boxplot of MonthlyCharge



In [49]:

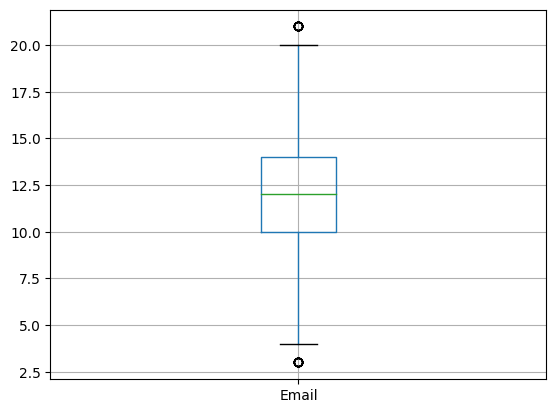
# verify outliers have been removed from Email

df[['Email']].boxplot()

Out[49]:

<Axes: >

Figure 17: Boxplot of Email



### Re-Expression of Categorical Variables[¶](" \l "Re-Expression-of-Categorical-Variables)

In [50]:

# view categorical variables

df[[

'Interaction',

'City',

'State',

'County',

'Area',

'Timezone',

'Job',

'Education',

'Employment',

'Marital',

'Gender',

'Churn',

'Techie',

'Contract',

'Port\_modem',

'Tablet',

'InternetService',

'Phone',

'Multiple',

'OnlineSecurity',

'OnlineBackup',

'DeviceProtection',

'TechSupport',

'StreamingTV',

'StreamingMovies',

'PaperlessBilling',

'PaymentMethod'

|  | **Interaction** | **City** | **State** | **County** | **Area** | **Timezone** | **Job** | **Education** | **Employment** | **...** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | aa90260b-4141-4a24-8e36-b04ce1f4f77b | Point Baker | AK | Prince of Wales-Hyder | Urban | America/Sitka | Environmental health practitioner | Master's Degree | Part Time | ... |
| **1** | fb76459f-c047-4a9d-8af9-e0f7d4ac2524 | West Branch | MI | Ogemaw | Urban | America/Detroit | Programmer, multimedia | Regular High School Diploma | Retired | ... |
| **2** | 344d114c-3736-4be5-98f7-c72c281e2d35 | Yamhill | OR | Yamhill | Urban | America/Los\_Angeles | Chief Financial Officer | Regular High School Diploma | Student | ... |
| **3** | abfa2b40-2d43-4994-b15a-989b8c79e311 | Del Mar | CA | San Diego | Suburban | America/Los\_Angeles | Solicitor | Doctorate Degree | Retired | ... |
| **4** | 68a861fd-0d20-4e51-a587-8a90407ee574 | Needville | TX | Fort Bend | Suburban | America/Chicago | Medical illustrator | Master's Degree | Student | ... |

]].head(5)

Out[50]:

5 rows × 27 columns

In [51]:

# count unique values in each categorical variable

# will be used to determine which variables can be re-expressed as numerical data

# variables with 3 or fewer unique values will be re-expressed as numerical data

df[[

'Interaction',

'City',

'State',

'County',

'Area',

'Timezone',

'Job',

'Education',

'Employment',

'Marital',

'Gender',

'Churn',

'Techie',

'Contract',

'Port\_modem',

'Tablet',

'InternetService',

'Phone',

'Multiple',

'OnlineSecurity',

'OnlineBackup',

'DeviceProtection',

'TechSupport',

'StreamingTV',

'StreamingMovies',

'PaperlessBilling',

'PaymentMethod'

]].nunique()

Out[51]:

Interaction 9985

City 6054

State 52

County 1620

Area 3

Timezone 25

Job 639

Education 12

Employment 5

Marital 5

Gender 3

Churn 2

Techie 2

Contract 3

Port\_modem 2

Tablet 2

InternetService 3

Phone 2

Multiple 2

OnlineSecurity 2

OnlineBackup 2

DeviceProtection 2

TechSupport 2

StreamingTV 2

StreamingMovies 2

PaperlessBilling 2

PaymentMethod 4

dtype: int64

In [52]:

# apply ordinal encoding: yes/no values

df.replace(to\_replace='No', value=0, inplace=True)

df.replace(to\_replace='Yes', value=1, inplace=True)

df[[

'Area',

'Gender',

'Churn',

'Techie',

'Contract',

'Port\_modem',

'Tablet',

'InternetService',

'Phone',

'Multiple',

'OnlineSecurity',

'OnlineBackup',

'DeviceProtection',

'TechSupport',

'StreamingTV',

'StreamingMovies',

'PaperlessBilling'

]].head(5)

Out[52]:

|  | **Area** | **Gender** | **Churn** | **Techie** | **Contract** | **Port\_modem** | **Tablet** | **InternetService** | **Phone** | **Multiple** | **…** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Urban | Male | 0 | 0 | One year | 1 | 1 | Fiber Optic | 1 | 0 | … |
| **1** | Urban | Female | 1 | 1 | Month-to-month | 0 | 1 | Fiber Optic | 1 | 1 | … |
| **2** | Urban | Female | 0 | 1 | Two Year | 1 | 0 | DSL | 1 | 1 | … |
| **3** | Suburban | Male | 0 | 1 | Two Year | 0 | 0 | DSL | 1 | 0 | … |
| **4** | Suburban | Male | 1 | 0 | Month-to-month | 1 | 0 | Fiber Optic | 0 | 0 | … |

In [53]:

# apply ordinal encoding: Contract variable

contract\_values = {'Month-to-month':0,

'One year':1,

'Two Year':2

}

df.replace(contract\_values, inplace=True)

df[['Contract']].head(5)

Out[53]:

|  | **Contract** |
| --- | --- |
| **0** | 1 |
| **1** | 0 |
| **2** | 2 |
| **3** | 2 |
| **4** | 0 |

In [54]:

# label encoding: Area and Gender variables

# return unique values from Area and Gender

df['Area'].unique(), df['Gender'].unique()

Out[54]:

(array(['Urban', 'Suburban', 'Rural'], dtype=object),

array(['Male', 'Female', 'Prefer not to answer'], dtype=object))

In [55]:

# apply label encoding for Area and Gender

area\_values = {

'Rural':0,

'Suburban':1,

'Urban':2

}

gender\_values = {

'Female':0,

'Male':1,

'Prefer not to answer':2

}

df.replace(area\_values, inplace=True)

df.replace(gender\_values, inplace=True)

df[['Area', 'Gender']].head(5)

Out[55]:

|  | **Area** | **Gender** |
| --- | --- | --- |
| **0** | 2 | 1 |
| **1** | 2 | 0 |
| **2** | 2 | 0 |
| **3** | 1 | 1 |
| **4** | 1 | 1 |

In [56]:

# return unique values from InternetService variable

# will be used to perform one-hot encoding

df['InternetService'].unique()

Out[56]:

array(['Fiber Optic', 'DSL', 'None'], dtype=object)

In [57]:

# create new columns (FiberOptic & DSL) to perform one-hot encoding

df['DSL'] = df['InternetService'].str.contains('DSL')

df['FiberOptic'] = df['InternetService'].str.contains('Fiber Optic')

df.drop(columns='InternetService', inplace=True)

df[['DSL', 'FiberOptic']].head()

Out[57]:

|  | **DSL** | **FiberOptic** |
| --- | --- | --- |
| **0** | False | True |
| **1** | False | True |
| **2** | True | False |
| **3** | True | False |
| **4** | False | True |

In [58]:

# apply ordinal encoding: True/False values

df.replace(to\_replace=False, value=0, inplace=True)

df.replace(to\_replace=True, value=1, inplace=True)

df[['DSL', 'FiberOptic']].head()

Out[58]:

|  | **DSL** | **FiberOptic** |
| --- | --- | --- |
| **0** | 0 | 1 |
| **1** | 0 | 1 |
| **2** | 1 | 0 |
| **3** | 1 | 0 |
| **4** | 0 | 1 |

In [59]:

# data cleaning and data wrangling complete

# save results to csv

# df.to\_csv('output files/churn\_cleaned\_data.csv', header=True)

### Principal Component Analysis[¶](" \l "Principal-Component-Analysis)

In [60]:

# install sklearn

# pip install scikit-learn

In [61]:

# Title: scikit-learn/scikit-learn: Scikit-learn

# Author: Grisel, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7711792

# import libraries from sklearn to perform PCA

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

In [62]:

# create DataFrame with numerical values for PCA

variables = df[['Zip',

'Lat',

'Lng',

'Population',

'Children',

'Age',

'Income',

'Outage\_sec\_perweek',

'Email',

'Contacts',

'Yearly\_equip\_failure',

'Tenure',

'MonthlyCharge',

'Bandwidth\_GB\_Year',

'item1',

'item2',

'item3',

'item4',

'item5',

'item6',

'item7',

'item8']]

variables.head(5)

Out[62]:

|  | **Zip** | **Lat** | **Lng** | **Population** | **Children** | **Age** | **Income** | **Outage\_sec\_perweek** | **Email** | **Contacts** | **...** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 99927 | 56.25100 | -133.37571 | 38 | 1.0 | 68.0 | 28561.990 | 6.972566 | 10 | 0 | ... |
| **1** | 48661 | 44.32893 | -84.24080 | 10446 | 1.0 | 27.0 | 21704.770 | 12.014541 | 12 | 0 | ... |
| **2** | 97148 | 45.35589 | -123.24657 | 3735 | 4.0 | 50.0 | 33186.785 | 10.245616 | 9 | 0 | ... |
| **3** | 92014 | 32.96687 | -117.24798 | 13863 | 1.0 | 48.0 | 18925.230 | 15.206193 | 15 | 2 | ... |
| **4** | 77461 | 29.38012 | -95.80673 | 11352 | 0.0 | 83.0 | 40074.190 | 8.960316 | 16 | 2 | ... |

5 rows × 22 columns

In [63]:

# standardize df\_pca using StandardScaler

scaler = StandardScaler()

scaled\_variables = scaler.fit\_transform(variables)

In [64]:

# apply PCA

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

pca.fit(scaled\_variables)

df\_pca = pd.DataFrame(pca.transform(scaled\_variables))

#### Component Loadings Matrix[¶](#Component-Loadings-Matrix)

In [65]:

loadings = pd.DataFrame(pca.components\_.T,

columns = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11',

'PC12','PC13','PC14','PC15','PC16','PC17','PC18','PC19','PC20','PC21','PC22'],

index=variables.columns)

loadings

Out[65]:

Table 1: Component Loading Matrix

**PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 ... PC13 PC14 PC15 PC16 PC17 PC18 PC19 PC20 PC21 PC22**

**Zip** -0.019116 0.600737 0.362174 0.045941 0.025099 0.010736 0.005911 0.002608 0.005987 0.010813 ... -0.014603 -0.091734 -0.075165 -0.013420 0.012051 -0.002998 -0.003844 -0.011517 0.038715 -0.698097

**Lat** -0.001275 0.035135 -0.003037 -0.004258 -0.703311 -0.062578 -0.058239 -0.079983 0.005248 0.017169 ... 0.029083 0.518079 0.450483 0.039628 -0.028215 -0.006929 0.014427 0.007467 0.007990 -0.117250

**Lng** 0.017064 -0.603891 -0.361951 -0.043544 0.061476 -0.000519 0.002413 0.013376 -0.009448 -0.007822 ... 0.003941 -0.020241 -0.018524 0.002366 -0.006369 0.012015 -0.006799 0.008736 0.040580 -0.703302

**Population** -0.002907 0.052810 0.038574 0.023750 0.675478 0.075009 0.054625 0.063100 0.053312 -0.044313 ... 0.040914 0.585722 0.397697 0.051240 -0.000762 -0.000975 0.000142 0.006246 0.001253 -0.030959

**Children** 0.001200 -0.027884 -0.019021 0.006778 -0.052752 0.004383 0.618178 0.026195 0.233271 0.275482 ... 0.017570 -0.000869 0.036781 -0.035640 0.017925 0.011433 0.018016 0.008124 -0.018488 0.000591

**Age** 0.003812 0.005345 -0.027214 -0.015545 0.025882 -0.018872 -0.403601 0.546014 -0.066468 -0.133755 ... 0.107425 0.091274 -0.032657 0.012712 0.006029 -0.015860 -0.003689 -0.009744 0.019560 0.001359

**Income** -0.000764 -0.002810 0.006632 0.023936 -0.060391 -0.009277 0.127235 0.422815 0.773733 0.053258 ... -0.074420 -0.028976 -0.002353 -0.060704 0.005916 0.000604 0.013878 -0.004736 0.001121 0.001557

**Outage\_sec\_perweek** -0.012834 -0.007901 0.019366 -0.048386 -0.086943 0.698695 0.048328 -0.000007 0.021856 0.016149 ... 0.689405 0.034190 -0.121056 0.007236 0.013501 -0.017510 0.010525 0.004356 0.000803 -0.000420

**Email** 0.008301 -0.000331 -0.021031 -0.004353 0.147656 0.063675 -0.073841 -0.585775 0.161615 0.332985 ... -0.031745 -0.004336 0.075825 -0.021246 0.017482 0.007571 -0.010501 -0.002135 0.005588 -0.002255

**Contacts** -0.009195 -0.008833 -0.002327 -0.011547 0.026083 0.008642 -0.528664 0.081181 0.119871 0.737195 ... 0.020634 -0.033021 0.027283 -0.033703 0.003469 -0.026186 0.021171 0.000095 -0.002578 0.002228

**Yearly\_equip\_failure** -0.007947 0.001629 0.019971 0.007852 -0.011618 0.068137 0.365324 0.396131 -0.541092 0.476411 ... -0.117038 0.018885 0.050222 0.008301 0.012543 -0.001025 0.006993 0.021546 -0.002347 0.002824

**Tenure** -0.011166 -0.357274 0.602793 -0.070295 -0.002352 -0.060907 -0.018589 -0.001626 -0.000247 -0.005503 ... 0.039228 0.006605 0.006625 -0.009017 -0.007846 -0.012220 0.005619 -0.004574 -0.703731 -0.041122

**MonthlyCharge** -0.000493 -0.036303 0.030581 -0.026467 -0.064068 0.694935 -0.093904 0.014718 0.019963 -0.096429 ... -0.686072 -0.004627 0.051426 0.009089 0.014360 -0.000205 0.022194 0.012930 -0.048297 0.000268

**Bandwidth\_GB\_Year** -0.013105 -0.358116 0.604486 -0.072702 -0.008043 -0.013011 0.002002 -0.008723 0.006063 0.000599 ... -0.010233 -0.001283 0.015863 0.003134 -0.003656 -0.001634 -0.007585 -0.007721 0.705798 0.039252

**item1** 0.458843 -0.031940 0.018104 0.279372 -0.016332 0.030085 0.004587 0.001286 -0.019097 0.017313 ... -0.006450 0.056520 -0.043136 -0.117461 0.047846 0.025440 -0.240748 -0.792638 -0.003356 0.002142

**item2** 0.434088 -0.021752 0.036935 0.282285 -0.019949 0.017290 -0.016258 -0.000028 -0.000048 0.009401 ... -0.001782 0.073680 -0.078217 -0.169873 0.069643 0.071636 -0.590359 0.574366 -0.002574 0.003399

**item3** 0.400897 -0.030673 0.022620 0.280210 -0.002219 -0.014487 -0.003157 -0.029126 -0.024524 -0.016610 ... -0.009815 0.113096 -0.135118 -0.248292 0.147807 -0.395856 0.674390 0.175762 0.014583 -0.005206

**item4** 0.145778 0.056814 -0.023044 -0.565751 -0.002230 -0.034638 -0.003847 -0.005423 -0.024217 -0.009672 ... -0.021938 0.143279 -0.107621 -0.475279 0.446686 0.431525 0.088801 -0.017993 0.001485 0.001896

**item5** -0.175463 -0.064689 0.036925 0.584719 -0.010086 0.027063 -0.040206 0.008504 -0.012674 -0.011120 ... 0.041774 -0.082917 0.098022 0.063028 0.207228 0.693842 0.265534 0.042226 -0.003210 0.003164

**item6** 0.404561 0.036625 0.008633 -0.181670 0.015243 0.008010 0.003716 0.005575 0.007675 0.024595 ... -0.003672 0.025814 -0.062686 0.056080 -0.758909 0.401923 0.226828 0.064288 0.001496 -0.000415

**item7** 0.357786 0.020355 0.008126 -0.180606 -0.016924 -0.033971 0.011918 -0.003977 0.055943 0.047546 ... -0.025873 0.068868 -0.162668 0.806930 0.371479 0.068579 0.066199 0.040642 -0.006563 -0.002021

**item8** 0.308598 0.025249 -0.004541 -0.131236 0.043896 0.039449 -0.019265 0.062519 -0.020641 -0.079420 ... 0.118592 -0.559999 0.725298 -0.013143 0.110858 -0.043083 0.048862 0.042687 -0.003003 -0.002199

22 rows × 22 columns

In [67]:

# calculate covariance, define eigenvalues

cov\_matrix = np.dot(scaled\_variables.T, scaled\_variables)/variables.shape[0]

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

In [68]:

# Title: matplotlib/matplotlib

# Author: Caswell, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7697899

# import matplotlib for plots

import matplotlib.pyplot as plt

### Scree Plot

:In [69]:

# plot eigenvalues against number of components

plt.plot(eigenvalues)

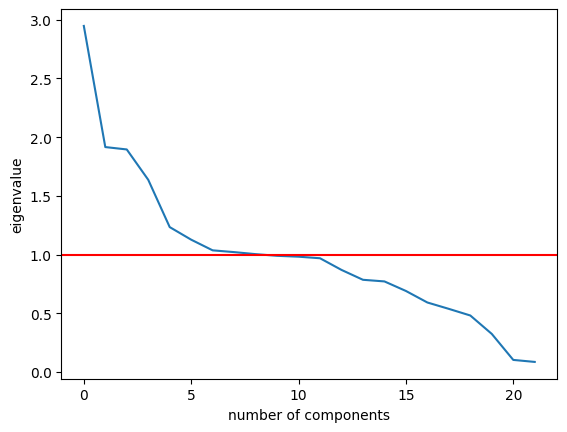
plt.xlabel('number of components')

plt.ylabel('eigenvalue')

plt.axhline(y=1, color='r')

plt.show()

Figure 18: Scree Plot



## Section D.4: Verification and Outcome[[2]](#footnote-2)[¶](" \l "Section-D.4:-Verification-and-Outcome)

### Duplicates[¶](" \l "Duplicates)

No steps were taken to handle duplicate values.

### Missing Values[¶](" \l "Missing-Values)

Depending on the datatype and characteristics of the data, missing values were filled using either the median, mode, or random values. Features of the qualitative datatype were filled using the mode. Quantitative data with a uniform distribution was imputed using random values. Quantitative data with a skewed distribution was imputed using the median.

Verification of this data cleaning step was performed in two stages. First, the pandas.DataFrame.isnull.sum() function was applied on all columns that originally contained missing values. For each feature that this function was applied on, zero was returned as the sum of null values, indicating that all missing values were filled.

Next, visualizations were produced to understand the changes in the dataset. Matrices of missing values were generated using the msno.matrix() function both before and after applying the data cleaning transformations (refer to Figure 1: Matrix of Missing Values and Figure 15: Matrix of Missing Values, respectively). While sparsity was observed in the original matrix, it was not observed in the subsequent matrix.

### Outliers[¶](" \l "Outliers)

Outliers were discarded from the dataset if it was determined that they were the result of error or illegitimate entry. In this case, the outliers found in the MonthlyCharge and Email features were removed from the dataset.

Visualizations were used to verify this data cleaning step. Boxplots were used to identify the outliers contained in the MonthlyCharge and Email features using the pandas.DataFrame.boxplot() function. After applying the data cleaning transformations, subsequent boxplots showed fewer outliers in both features (refer to Figure 11: Boxplot of MonthlyCharge, Figure 7: Boxplot of Email, Figure 16: Boxplot of MonthlyCharge, and Figure 17: Boxplot of Email).

## Section D.5: Output File[¶](" \l "Section-D.5:-Output-File)

To view the results of the data cleaning process, refer to the attached churn\_cleaned\_data.csv.

## Section D.6: Limitations[¶](" \l "Section-D.6:-Limitations)

### Duplicates[¶](" \l "Duplicates)

Given that the pandas.DataFrame.duplicated() function was only applied across records and Customer\_id values, it is possible that some instances of duplication were not detected. For example, if there was an instance of duplication that occurred across the combination of City, State, and Interaction features, the methods used in Section C.4: Code would not have identified this data quality issue.

### Missing Values[¶](" \l "Missing-Values)

Univariate imputation (filling NA/NaN values using the mean, median, and mode) was the dominant method for handling missing values. The drawback of univariate imputation is that it can distort the distribution of the data (Larose & Larose, 2019). This phenomenon was observed in the Children, Income, Tenure, and Bandwidth\_GB\_Year features; the histograms for each of these features changed shape after the data cleaning transformations were applied (refer to Figure 2: Histograms of Children, Age, Income, Tenure, and Bandwidth\_GB\_Year and Figure 14: Histograms of Children, Income, Tenure, and Bandwidth\_GB\_Year).

### Outliers[¶](" \l "Outliers)

Outliers that were deemed erroneous were dropped from the dataset, thus reducing the sample size, and potentially eliminating valuable data points from the dataset. For most features, however, all outliers were retained; the disadvantage of retaining outliers is that it can lead to the distortion of statistical models (Larose & Larose, 2019).

## Section D.7: Effect on the Research Question[¶](" \l "Section-D.7:-Effect-on-the-Research-Que)

As noted in Section A, the research question for this project is "Do customers enrolled in more service offerings display lower levels of churn?" To answer this research question, the data analyst will need to analyze the relevant data, which includes:

* Churn
* Port\_modem
* Tablet
* InternetService
* Phone
* Multiple
* OnlineSecurity
* OnlineBackup
* DeviceProtection
* TechSupport
* StreamingTV
* StreamingMovies

None of these features were determined to contain missing values, and since these features are of the categorical data type, cannot contain outliers. For this reason, the limitations discussed in Section D.6 regarding missing values and outliers would not affect the analysis of the research question. On the other hand, the limitation regarding the identification of duplicates could impact the analysis; if unidentified duplicates exist in the dataset, then conclusions drawn from the data might be distorted or inaccurate.

## Section E.1: Principal Component Analysis[¶](" \l "Section-E.1:-Principal-Component-Analys)

A Principal Component Analysis (PCA) was performed on all numeric data (refer to Table 1: Component Loading Matrix). The results of the PCA indicate that there are 6 principal components in the dataset.

## Section E.2: Kaiser Rule[¶](" \l "Section-E.2:-Kaiser-Rule)

The Kaiser Rule recommends retaining principal components with eigenvalues greater than 1 (Larose & Larose, 2019). According to the scree plot (refer to Figure 18: Scree Plot) principal components 1 through 6 should be retained as they fall above this threshold.

## Section E.3: Benefits of PCA[¶](" \l "Section-E.3:-Benefits-of-PCA)

PCA is a data mining technique used to reduce the dimensionality of a dataset. This is beneficial for several reasons. Firstly, dimensionality reduction prevents a modeling error in statistics called overfitting. Next, a reduced dataset means that machine learning algorithms take less time to train. Finally, PCA improves data visualization by reducing the number of features that need to be visualized (Larose & Larose, 2019).

Part Four: Supporting Documents[¶](" \l "Part-Four:-Supporting-Documents)

## Section F: Panopto Video[¶](" \l "Section-F:-Panopto-Video)

To view a walkthrough demonstration of the functionality of the code, refer to the Panopto link: ...

## Section G: Web Sources

Bilogur, et al. (2018, February). doi:10.5281/zenodo.1184723

Caswell, et al. (2023, March). doi:10.5281/zenodo.7697899

Gommers, et al. (2023, February). doi:10.5281/zenodo.7655153

Grisel, et al. (2023, March). doi:10.5281/zenodo.7711792

Pawson, I. (2021, August). doi:10.25080/majora-1b6fd038-026

The pandas development team. (2023, March). doi:zenodo.7741580

## [¶](#Section-G:-Web-Sources)Section H: References[¶](" \l "Section-H:-References)

Larose, D. T., & Larose, C. D. (2019). *Data Science Using Python and R.* Hoboken: Wiley. Retrieved March 2023

pandas. (2023). *API reference*. Retrieved March 2023, from pandas: https://pandas.pydata.org/docs/reference/index.html

The SciPy Community. (2023). *SciPy API*. Retrieved March 2023, from SciPy: https://docs.scipy.org/doc/scipy/reference/index.html

1. Refer to Section D.3: Code to review the full set of data cleaning steps. [↑](#footnote-ref-1)
2. Refer to Section D.3: Code to review the data cleaning results. [↑](#footnote-ref-2)