Professional Communication

In this document, a Professional Communication Evaluator has identified representative examples of consistent, patterned writing errors.

Not all errors are marked.

To demonstrate competency in Professional Communication, please correct the representative examples and revise your work to amend unmarked errors similar to the identified examples.

In resubmissions, a Professional Communication Evaluator may note different examples of remaining issues. If more than one document was submitted, those documents might require similar revisions.

Performance Assessment | D206 Data Cleaning

Ryan L. Buchanan
Student ID: 001826691
Masters Data Analytics (12/01/2020)
Program Mentor: Dan Estes
(385) 432-9281 (MST)
rbuch49@wgu.edu

Part I: Research Question¶

A. Question or Decision:¶

Can we determine which individual customers are at high risk of churn? And, can we determine which features are most significant to churn?

**A1. Alternative Question:¶

Also, Are there certain responses to survey that correlate with customer churn?

Commented [PCEV18-1]:
Parts of Speech: Missing article
→Parts of speech errors recur.

WGU's Guide to Academic Writing
Link: Module 8.12: Article Usage

B. Required Variables:¶

The data set is 10,000 customer records of a popular telecommunications company. The dependent variable (target) in question is whether or not each customer has continued or discontinued service within the last month. This column is titled "Churn."
Independent variables or predictors that may lead to identifying a relationship with the dependent variable of "Churn" within the dataset include:

1. Services that each customer signed up for (for example, multiple phone lines, technical support add-ons or streaming media)
2. Customer account information (customers' tenure with the company, payment methods, bandwidth usage, etc.)
3. Customer demographics (gender, marital status, income, etc.).
4. Finally, there are eight independent variables that represent responses customer-perceived importance of company services and features.

The data is both numerical (as in the yearly GB bandwidth usage; customer annual income) and categorical (a "Yes" or "No" for Churn; customer job).

Part II: Data-Cleaning Plan¶

C1. Plan to Find Anomalies:¶

My approach will include:
1. Back up my data and the process I am following as a copy to my machine and, since this is a manageable dataset, to GitHub using command line and gitbash.

2. Read the data set into Python using Pandas read_csv command.
3. Evaluate the data struture to better understand input data.
4. Naming the dataset as a the variable "churn_df" and subsequent useful slices of the dataframe as "df".
5. Examine potential misspellings, awkward variable namimg & missing data.
6. Find outliers that may create or hide statistical significance using histograms.
7. Imputing records missing data with meaningful measures of central tendency (mean, median or mode) or simply remove outliers that are several standard deviations above the mean.

C2. Justification of Approach:¶

Though the data seems to be inexplicably missing quite a bit of data (such as the many NAs in customer tenure with the company) from apparently random columns, this approach seems like a good first approach in order to put the data in better working order without needing to involve methods of initial data collection or querying the data-gatherers on reasons for missing information. Also, this the first dataset that I've clean, so I followed the procedures practice in the performance lab as well as tips from StackOverflow and other tutorial resources.

C3. Justification of Tools:¶

I will use the Python programming language as I have a bit of a background in Python having studied machine learning independently over the last year before beginning this masters program and its ability to perform many things right "out of the box." Python provides clean, intuitive and readable syntax that has become ubiquitous across in the data science industry. Also, I find the Jupyter notebooks a convenient way to run code visually, in its attractive single document markdown format, the ability to display results of code and graphic visualizations and provide crystal-clear running documentation for future reference. A thorough installation and

Commented [PCEV18-2]:
Conventions: Missing punctuation
→Conventions errors recur.

WGU's Guide to Academic Writing
Link: Module 7.10: Apostrophes

Commented [PCEV18-3]:
Conventions: Misspelled word
→Conventions errors recur.

WGU's Guide to Academic Writing
Link: Module 7.13: Spelling

importation of Python packages and libraries will provide specially designed code to perfom complex data science tasks rather than personally building them from scratch. This will include:

- NumPy - to work with arrays
- Pandas - to load datasets
- Matplotlib - to plot charts
- Scikit-learn - for machine learning model classes
- SciPy - for mathematical problems, specifically linear algebra transformations
- Seaborn - for high-level interface and atttractive visualizations

A quick, precise example of loading a dataset and creating a variable efficiently is using to call the Pandas library and its subsequent "read_csv" function in order to manipulate our data as a dataframe:

```
import pandas as pd
df = pd.read_csv('Data.csv')
```

C4. Provide the Code:¶

In [1]:

```
# Install necessary packages
!pip install pandas
!pip install numpy
!pip install scipy
!pip install sklearn
!pip install matplotlib
```

```
Requirement already satisfied: pandas in c:\users\vreed\anaconda3\lib\site-packages (1.0.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\vreed\anaconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\vreed\anaconda3\lib\site-packages (from pandas) (1.18.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\vreed\anaconda3\lib\site-packages (from pandas) (2019.3)
Requirement already satisfied: six>=1.5 in c:\users\vreed\anaconda3\lib\site-packages (from python-dateutil>=2.6.1->pandas) (1.14.0)
Requirement already satisfied: numpy in c:\users\vreed\anaconda3\lib\site-packages (1.18.1)
```

```
Requirement already satisfied: scipy in c:\users\vreed\anaconda3\lib\site-packages (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\vreed\anaconda3\lib\site-packages
(from scipy) (1.18.1)
Requirement already satisfied: sklearn in c:\users\vreed\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\vreed\anaconda3\lib\site-packages
(from sklearn) (0.22.1)
Requirement already satisfied: scipy>=0.17.0 in c:\users\vreed\anaconda3\lib\site-packages
(from scikit-learn->sklearn) (1.4.1)
Requirement already satisfied: numpy>=1.11.0 in c:\users\vreed\anaconda3\lib\site-packages
(from scikit-learn->sklearn) (1.18.1)
Requirement already satisfied: joblib>=0.11 in c:\users\vreed\anaconda3\lib\site-packages
(from scikit-learn->sklearn) (0.14.1)
Requirement already satisfied: matplotlib in c:\users\vreed\anaconda3\lib\site-packages
(3.1.3)
Requirement already satisfied: cycler>=0.10 in c:\users\vreed\anaconda3\lib\site-packages
(from matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\vreed\anaconda3\lib\site-
packages (from matplotlib) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\vreed\anaconda3\lib\site-packages (from matplotlib) (2.4.6)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\vreed\anaconda3\lib\site-
packages (from matplotlib) (2.8.1)
Requirement already satisfied: numpy>=1.11 in c:\users\vreed\anaconda3\lib\site-packages
(from matplotlib) (1.18.1)
Requirement already satisfied: six in c:\users\vreed\anaconda3\lib\site-packages (from
cycler>=0.10->matplotlib) (1.14.0)
Requirement already satisfied: setuptools in c:\users\vreed\anaconda3\lib\site-packages
(from kiwisolver>=1.0.1->matplotlib) (45.2.0.post20200210)
```

In [2]:

```
# Standard imports
import numpy as np
import pandas as pd
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
# Increase Jupyter display cell-width
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:75% !important;
}</style>"))
```

In [4]:

```
# Load data set into Pandas dataframe
churn_df = pd.read_csv('churn_raw_data.csv')
```

In [5]:

```
# Display Churn dataframe
churn_df
```

Out[5]:

	Unnamed: 0	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	...	MonthlyCh
0	1	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	...	171.449762
1	2	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	...	242.948015
2	3	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	...	159.440398
3	4	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	-117.24798	...	120.249493
4	5	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	...	150.761216
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	9996	M324793	45deb5a2-ae04-4518-	Mount Holly	VT	Rutland	5758	43.43391	-72.78734	...	159.828800

	Unnamed: 0	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	...	MonthlyCh
				bf0b-c82db8dbe4a4								
9996	9997	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	Clarksville	TN	Montgomery	37042	36.56907	-87.41694	...	208.856400
9997	9998	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	Mobeetie	TX	Wheeler	79061	35.52039	-100.44180	...	168.220900
9998	9999	9999	I641617	3775ccfc-0052-4107-81ae-9657f81ecdf3	Carrollton	GA	Carroll	30117	33.58016	-85.13241	...	252.628600
9999	10000	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	Clarkesville	GA	Habersham	30523	34.70783	-83.53648	...	218.371000

10000 rows × 52 columns

In [6]:

```
# List of Dataframe Columns
df = churn_df.columns
print(df)
```

```
Index(['Unnamed: 0', 'CaseOrder', 'Customer_id', 'Interaction', 'City',
       'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
       'Timezone', 'Job', 'Children', 'Age', 'Education', 'Employment',
       'Income', 'Marital', 'Gender', 'Churn', 'Outage_sec_perweek', 'Email',
       'Contacts', 'Yearly_equip_failure', 'Techie', 'Contract', 'Port_modem',
       'Tablet', 'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'Tenure',
       'MonthlyCharge', 'Bandwidth_GB_Year', 'item1', 'item2', 'item3',
       'item4', 'item5', 'item6', 'item7', 'item8'],
      dtype='object')
```

In [7]:

```
# Remove redundant "Unnamed" column at beginning & display first five records
```

```
df = churn_df.drop(churn_df.columns[0], axis = 1)
df.head()
```

Out[7]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCharge	Ban
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	133.37571	38	...	171.449762	904
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	84.24080	10446	...	242.948015	800
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	123.24657	3735	...	159.440398	205
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	117.24798	13863	...	120.249493	216
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	95.80673	11352	...	150.761216	271

5 rows × 51 columns

In [8]:

```
# Rename last 8 survey columns for better description of
variables
df.rename(columns = {'item1':'Responses',
                    'item2':'Fixes',
                     'item3':'Replacements',
                     'item4':'Reliability',
                     'item5':'Options',
                     'item6':'Respectful',
                     'item7':'Courteous',
                     'item8':'Listening'},
          inplace=True)
```

In [9]:

```
df.columns
```

Out[9]:

```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'City', 'State', 'County',
       'Zip', 'Lat', 'Lng', 'Population', 'Area', 'Timezone', 'Job',
       'Children', 'Age', 'Education', 'Employment', 'Income', 'Marital',
       'Gender', 'Churn', 'Outage_sec_perweek', 'Email', 'Contacts',
       'Yearly_equip_failure', 'Techie', 'Contract', 'Port_modem', 'Tablet',
       'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'Tenure',
       'MonthlyCharge', 'Bandwidth_GB_Year', 'Responses', 'Fixes',
       'Replacements', 'Reliability', 'Options', 'Respectful', 'Courteous',
       'Listening'],
      dtype='object')
```

In [10]:

```
df.head()
```

Out[10]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCharge	Ban
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	133.37571	38	...	171.449762	904
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	...	242.948015	800
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	123.24657	3735	...	159.440398	205
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	117.24798	13863	...	120.249493	216
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	...	150.761216	271

5 rows × 51 columns

In [11]:

```
# Find number of records and columns of dataset
df.shape
```

Out[11]:

```
(10000, 51)
```

In [12]:

```
# Describe Churn dataset statistics
df.describe()
```

Out[12]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_perweek	
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	7505.000000	7525.000000	7510.000000	10000.000000	10 00
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.095936	53.275748	39936.762226	11.452955	12
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.154758	20.753928	28358.469482	7.025921	3.0
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.000000	18.000000	740.660000	-1.348571	1.0
25%	2500.75000	26292.500000	35.341828	-97.082813	738.000000	0.000000	35.000000	19285.522500	8.054362	10
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.000000	53.000000	33186.785000	10.202896	12
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.000000	71.000000	53472.395000	12.487644	14
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.000000	89.000000	258900.700000	47.049280	23

8 rows × 23 columns

In [13]:

```
# Remove less meaningful variables from statistics description
df_stats = df.drop(columns=['CaseOrder', 'Zip', 'Lat', 'Lng'])
df_stats.describe()
```

Out[13]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	
count	10000.000000	7505.000000	7525.000000	7510.000000	10000.000000	10000.000000	10000.000000	10000.000000	906 00
mean	9756.562400	2.095936	53.275748	39936.762226	11.452955	12.016000	0.994200	0.398000	34.4
std	14432.698671	2.154758	20.753928	28358.469482	7.025921	3.025898	0.988466	0.635953	26.4
min	0.000000	0.000000	18.000000	740.660000	-1.348571	1.000000	0.000000	0.000000	1.00
25%	738.000000	0.000000	35.000000	19285.522500	8.054362	10.000000	0.000000	0.000000	7.89
50%	2910.500000	1.000000	53.000000	33186.785000	10.202896	12.000000	1.000000	0.000000	36.1
75%	13168.000000	3.000000	71.000000	53472.395000	12.487644	14.000000	2.000000	1.000000	61.4
max	111850.000000	10.000000	89.000000	258900.700000	47.049280	23.000000	7.000000	6.000000	71.9

In [14]:

```
# Calculate Churn Rate
df.Churn.value_counts() / len(df)
```

Out[14]:

```
No     0.735
Yes    0.265
Name: Churn, dtype: float64
```

In [15]:

```
# Review data types (numerical => "int64" & "float64"; &
categorical => "object") in data set
df.dtypes
```

Out[15]:

```
CaseOrder              int64
Customer_id           object
Interaction           object
City                  object
State                 object
County                object
Zip                    int64
Lat                  float64
Lng                  float64
```

```
Population                int64
Area                     object
Timezone                 object
Job                      object
Children                float64
Age                     float64
Education                object
Employment               object
Income                  float64
Marital                  object
Gender                   object
Churn                    object
Outage_sec_perweek      float64
Email                     int64
Contacts                  int64
Yearly_equip_failure      int64
Techie                   object
Contract                 object
Port_modem               object
Tablet                   object
InternetService          object
Phone                    object
Multiple                 object
OnlineSecurity           object
OnlineBackup             object
DeviceProtection         object
TechSupport              object
StreamingTV              object
StreamingMovies          object
PaperlessBilling         object
PaymentMethod            object
Tenure                  float64
MonthlyCharge           float64
Bandwidth_GB_Year       float64
Responses                 int64
Fixes                     int64
Replacements              int64
Reliability               int64
Options                   int64
Respectful                int64
Courteous                 int64
Listening                 int64
dtype: object
```

In [16]:

```
# Re-validate column data types and missing values
df.columns.to_series().groupby(df.dtypes).groups
```

Out[16]:

```
{dtype('int64'): Index(['CaseOrder', 'Zip', 'Population', 'Email', 'Contacts',
        'Yearly_equip_failure', 'Responses', 'Fixes', 'Replacements',
        'Reliability', 'Options', 'Respectful', 'Courteous', 'Listening'],
       dtype='object'),
 dtype('float64'): Index(['Lat', 'Lng', 'Children', 'Age', 'Income', 'Outage_sec_perweek',
        'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'],
       dtype='object'),
 dtype('O'): Index(['Customer_id', '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'],
       dtype='object')}
```

In [17]:

```
# Display non-null fields within each columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 51 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Customer_id           10000 non-null  object
 2   Interaction           10000 non-null  object
 3   City                  10000 non-null  object
 4   State                 10000 non-null  object
 5   County                10000 non-null  object
 6   Zip                   10000 non-null  int64
 7   Lat                   10000 non-null  float64
 8   Lng                   10000 non-null  float64
 9   Population            10000 non-null  int64
 10  Area                  10000 non-null  object
 11  Timezone              10000 non-null  object
 12  Job                   10000 non-null  object
 13  Children              7505 non-null   float64
 14  Age                   7525 non-null   float64
```

```
 15  Education             10000 non-null  object
 16  Employment            10000 non-null  object
 17  Income                7510 non-null   float64
 18  Marital               10000 non-null  object
 19  Gender                10000 non-null  object
 20  Churn                 10000 non-null  object
 21  Outage_sec_perweek    10000 non-null  float64
 22  Email                 10000 non-null  int64
 23  Contacts              10000 non-null  int64
 24  Yearly_equip_failure  10000 non-null  int64
 25  Techie                7523 non-null   object
 26  Contract              10000 non-null  object
 27  Port_modem            10000 non-null  object
 28  Tablet                10000 non-null  object
 29  InternetService       10000 non-null  object
 30  Phone                 8974 non-null   object
 31  Multiple              10000 non-null  object
 32  OnlineSecurity        10000 non-null  object
 33  OnlineBackup          10000 non-null  object
 34  DeviceProtection      10000 non-null  object
 35  TechSupport           9009 non-null   object
 36  StreamingTV           10000 non-null  object
 37  StreamingMovies       10000 non-null  object
 38  PaperlessBilling      10000 non-null  object
 39  PaymentMethod         10000 non-null  object
 40  Tenure                9069 non-null   float64
 41  MonthlyCharge         10000 non-null  float64
 42  Bandwidth_GB_Year     8979 non-null   float64
 43  Responses             10000 non-null  int64
 44  Fixes                 10000 non-null  int64
 45  Replacements          10000 non-null  int64
 46  Reliability           10000 non-null  int64
 47  Options               10000 non-null  int64
 48  Respectful            10000 non-null  int64
 49  Courteous             10000 non-null  int64
 50  Listening             10000 non-null  int64
dtypes: float64(9), int64(14), object(28)
memory usage: 3.9+ MB
```

In [18]:

```
# Find missing values
df.isnull()
```

Out[18]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCharge	Bandwidth_GB_Y
0	False	False	False	False	False	False	False	False	False	False	...	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	...	False	False
9996	False	False	False	False	False	False	False	False	False	False	...	False	False
9997	False	False	False	False	False	False	False	False	False	False	...	False	False
9998	False	False	False	False	False	False	False	False	False	False	...	False	False
9999	False	False	False	False	False	False	False	False	False	False	...	False	False

10000 rows × 51 columns

In [19]:

```
# Access only rows from dataframe containing missing values
df.isnull().any(axis=1)
```

Out[19]:

```
0        True
1       False
2        True
3       False
4       False
        ...
9995     True
9996     True
9997     True
9998    False
9999     True
Length: 10000, dtype: bool
```

In [20]:

```
# Woah, lots of empty fields!  Immediately noticeable as "True" in columns of "Children", "Age", "Income", "Techie", "Phone", "Tenure"
# Display the specific columns with NAs
df.isna().any()
```

Out[20]:

```
CaseOrder               False
Customer_id             False
```

```
Interaction             False
City                    False
State                   False
County                  False
Zip                     False
Lat                     False
Lng                     False
Population              False
Area                    False
Timezone                False
Job                     False
Children                 True
Age                      True
Education               False
Employment              False
Income                   True
Marital                 False
Gender                  False
Churn                   False
Outage_sec_perweek      False
Email                   False
Contacts                False
Yearly_equip_failure    False
Techie                   True
Contract                False
Port_modem              False
Tablet                  False
InternetService         False
Phone                    True
Multiple                False
OnlineSecurity          False
OnlineBackup            False
DeviceProtection        False
TechSupport              True
StreamingTV             False
StreamingMovies         False
PaperlessBilling        False
PaymentMethod           False
Tenure                   True
MonthlyCharge           False
Bandwidth_GB_Year        True
Responses               False
Fixes                   False
Replacements            False
```

```
Reliability             False
Options                 False
Respectful              False
Courteous               False
Listening               False
dtype: bool
```

In [21]:

```
# Confirm missing observations numbers
data_nulls = df.isnull().sum()
print(data_nulls)
```

```
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
Responses                  0
Fixes                      0
Replacements               0
Reliability                0
Options                    0
Respectful                 0
Courteous                  0
Listening                  0
dtype: int64
```

In [22]:

```
# Store rows with missing values in a new variable
rows_with_missing_values = df.isnull().any(axis=1)
df[rows_with_missing_values]
```

Out[22]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCh
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	...	171.44976
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	...	159.44039
5	6	W303516	2b451d12-6c2b-4cea-a295-ba1d6bced078	Fort Valley	GA	Peach	31030	32.57032	-83.89040	17701	...	184.40155
6	7	U335188	6630d501-838c-4be4-	Pioneer	TN	Scott	37847	36.43420	-84.27892	2535	...	200.06488

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	MonthlyCh
			a59c-6f58c814ed6a									
7	8	V538685	70ddaa89-b726-49dc-9022-2d655e4c7936	Oklahoma City	OK	Oklahoma	73109	35.43313	-97.52463	23144	...	114.75411
...	...	...	...	...	...	...	...	...	...	...	...	...
9994	9995	P175475	c60df12b-a50b-4397-ae57-98381a0d3960	West Kill	NY	Greene	12492	42.18491	-74.33574	210	...	143.68790
9995	9996	M324793	45deb5a2-ae04-4518-bf0b-c82db8dbe4a4	Mount Holly	VT	Rutland	5758	43.43391	-72.78734	640	...	159.82880
9996	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	Clarksville	TN	Montgomery	37042	36.56907	-87.41694	77168	...	208.85640
9997	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	Mobeetie	TX	Wheeler	79061	35.52039	-100.44180	406	...	168.22090
9999	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	Clarkesville	GA	Habersham	30523	34.70783	-83.53648	12230	...	218.37100

7867 rows × 51 columns

In [23]:

```
# Examine columns for misspellings in categorical variables using unique() method
df['Employment'].unique()
```

Out[23]:

```
array(['Part Time', 'Retired', 'Student', 'Full Time', 'Unemployed'],
      dtype=object)
```

In [28]:

```
df['Area'].unique()
```

Out[28]:

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

In [29]:

```
df['Timezone'].unique()
```

Out[29]:

```
array(['America/Sitka', 'America/Detroit', 'America/Los_Angeles',
       'America/Chicago', 'America/New_York', 'America/Puerto_Rico',
       'America/Denver', 'America/Menominee', 'America/Phoenix',
       'America/Indiana/Indianapolis', 'America/Boise',
       'America/Kentucky/Louisville', 'Pacific/Honolulu',
       'America/Indiana/Petersburg', 'America/Nome', 'America/Anchorage',
       'America/Indiana/Knox', 'America/Juneau', 'America/Toronto',
       'America/Indiana/Winamac', 'America/Indiana/Vincennes',
       'America/North_Dakota/New_Salem', 'America/Indiana/Tell_City',
       'America/Indiana/Marengo', 'America/Ojinaga'], dtype=object)
```

In [31]:

```
df['Job'].unique()
```

Out[31]:

```
array(['Environmental health practitioner', 'Programmer, multimedia',
       'Chief Financial Officer', 'Solicitor', 'Medical illustrator',
       'Chief Technology Officer', 'Surveyor, hydrographic',
       'Sales promotion account executive',
       'Teaching laboratory technician', 'Museum education officer',
       'Teacher, special educational needs', 'Maintenance engineer',
       'Engineer, broadcasting (operations)', 'Learning disability nurse',
       'Automotive engineer', 'Amenity horticulturist',
       'Applications developer', 'Immunologist', 'Engineer, electrical',
       'Broadcast presenter', 'Counsellor', 'Geophysical data processor',
       'Designer, multimedia', 'Event organiser',
       'Equality and diversity officer', 'Psychiatrist',
       'Surveyor, commercial/residential', 'Civil Service administrator',
       'Radiographer, diagnostic', 'Air traffic controller', 'Dietitian',
       'Therapist, occupational', 'Building services engineer',
       'Information officer', 'Outdoor activities/education manager',
       'Market researcher', 'Surveyor, insurance', 'Office manager',
       'Editorial assistant', 'Customer service manager',
       'Production designer, theatre/television/film',
       'Analytical chemist', 'Print production planner',
       'Conservation officer, nature', 'Librarian, public',
       'Financial adviser', 'Surveyor, building',
       'Horticulturist, amenity', 'Diagnostic radiographer',
```

```
'Doctor, general practice', 'Insurance risk surveyor',
'Heritage manager', 'Legal executive', 'Professor Emeritus',
'Radio producer', "Barrister's clerk", 'Engineer, automotive',
'Recruitment consultant', 'Commercial horticulturist',
'Pharmacist, community', 'Forest/woodland manager',
'Designer, graphic', 'Civil engineer, consulting',
'Science writer', 'Health and safety inspector',
'Administrator, Civil Service', 'Technical sales engineer',
'Special educational needs teacher', 'Sports therapist',
'Engineer, communications', 'Oceanographer', 'Archaeologist',
'Personal assistant', 'Animal nutritionist', 'Hydrologist',
'Arts development officer', 'Herpetologist',
'Medical sales representative',
'Scientist, research (physical sciences)',
'Higher education lecturer', 'Nurse, adult', 'Chiropodist',
'Therapeutic radiographer', 'Designer, television/film set',
'Education officer, environmental', 'Colour technologist',
'Academic librarian', 'Mudlogger', 'Designer, textile',
'Chief Strategy Officer', 'Loss adjuster, chartered',
'Pharmacologist', 'Hydrographic surveyor',
'Engineer, manufacturing', 'Research scientist (medical)',
'Wellsite geologist', 'Embryologist, clinical',
'Occupational psychologist', 'Sales professional, IT',
'Advertising copywriter', 'Radiographer, therapeutic',
'English as a second language teacher', 'Occupational therapist',
'Armed forces logistics/support/administrative officer',
'Technical author', 'Regulatory affairs officer',
'Optician, dispensing', 'Theme park manager', 'IT trainer',
'Contracting civil engineer', 'Psychologist, sport and exercise',
'Manufacturing engineer', 'Musician',
'Senior tax professional/tax inspector', 'Engineer, biomedical',
'Facilities manager', 'Osteopath', 'Corporate investment banker',
'Psychotherapist', 'Copywriter, advertising',
'Horticultural consultant', 'Microbiologist',
'Educational psychologist', 'Sport and exercise psychologist',
'Risk manager', 'Health visitor', 'Visual merchandiser',
'Clinical biochemist', 'Water quality scientist', 'Optometrist',
'Petroleum engineer', 'Building control surveyor',
'Financial planner', 'Theatre director', 'Secretary, company',
'Materials engineer', 'Civil Service fast streamer',
'Health service manager', 'Scientist, forensic',
'Immigration officer', 'Dealer',
'Planning and development surveyor', 'Broadcast engineer',
'Local government officer', 'Nature conservation officer',
```

```
'Private music teacher', 'Geologist, wellsite', 'Gaffer',
'Curator', 'Editor, commissioning', 'Barrister', 'TEFL teacher',
'Public relations account executive', 'Audiological scientist',
'Travel agency manager', 'Land', 'Music therapist',
'Librarian, academic', 'Film/video editor',
'Journalist, broadcasting', 'Waste management officer',
'Scientist, water quality', 'Sub', 'Neurosurgeon',
'Scientist, research (maths)', 'Public house manager',
'Building surveyor', 'Animator',
'Production assistant, television', 'Transport planner',
'Geneticist, molecular', 'Merchant navy officer',
'Research scientist (life sciences)',
'Engineer, building services', 'Solicitor, Scotland',
'Hospital pharmacist', 'Engineer, petroleum', 'Oncologist',
'IT technical support officer', 'Site engineer',
'Early years teacher', 'Plant breeder/geneticist',
'Chartered management accountant',
'Runner, broadcasting/film/video', 'Newspaper journalist',
'Naval architect', 'Agricultural engineer', 'Meteorologist',
'Designer, ceramics/pottery', 'Environmental education officer',
'Textile designer', 'Engineer, materials', 'Magazine journalist',
'Conference centre manager', 'Dance movement psychotherapist',
'Warden/ranger', 'Teacher, English as a foreign language',
'Producer, television/film/video', 'Make', 'Pharmacist, hospital',
'Therapist, horticultural', 'Journalist, newspaper',
'Retail merchandiser', 'Nurse, mental health', 'Chief of Staff',
'Systems analyst', 'Electronics engineer', 'Quantity surveyor',
'Minerals surveyor', 'Scientist, research (life sciences)',
'Archivist', 'Brewing technologist',
'Investment banker, operational',
'Accountant, chartered certified', 'Surveyor, minerals',
'Hospital doctor', 'Theatre stage manager',
'Operational researcher', 'Tax inspector',
'Television camera operator', 'Arts administrator',
'Patent attorney', 'Bonds trader', 'Ship broker',
'Furniture conservator/restorer', 'Media planner',
'Radio broadcast assistant', 'Mental health nurse',
'Purchasing manager', 'Scientist, biomedical', 'Photographer',
'Sports coach', 'Environmental manager', 'Estate agent',
'Public librarian', 'Designer, blown glass/stained glass',
'Occupational hygienist', 'Surgeon', 'Youth worker',
'Hotel manager', 'Programmer, systems', "Politician's assistant",
'Social researcher', 'Publishing copy', 'Tax adviser',
'Quarry manager', 'Buyer, industrial', 'Production manager',
```

```
'Police officer', 'Theatre manager', 'Sports administrator',
'Research scientist (maths)', 'Therapist, music', 'Soil scientist',
'Holiday representative', 'Producer, radio',
'Intelligence analyst', 'Geochemist', 'Probation officer',
'Fish farm manager', 'Chartered accountant', 'Architect',
'Psychiatric nurse', 'Farm manager', 'Geoscientist',
'Lecturer, further education', 'Horticulturist, commercial',
'Surveyor, quantity', 'Clothing/textile technologist',
'Technical brewer', 'Landscape architect',
'Information systems manager', 'Sales executive',
'Exercise physiologist', 'Administrator, arts', 'Careers adviser',
'Lobbyist', 'Claims inspector/assessor', 'Recycling officer',
'Product/process development scientist', 'Paramedic',
'Fine artist', 'Teacher, secondary school',
'Data processing manager', 'Government social research officer',
'Product manager', 'Accounting technician', 'Engineer, land',
'Lawyer', 'Restaurant manager', 'Catering manager', 'Contractor',
'Research officer, government', 'Medical secretary', 'Podiatrist',
'Phytotherapist', 'Surveyor, building control', 'Comptroller',
'Lighting technician, broadcasting/film/video', 'Paediatric nurse',
'Designer, furniture', 'Adult guidance worker',
'Clinical molecular geneticist', 'Games developer', 'Metallurgist',
'Armed forces technical officer', 'Risk analyst',
'Careers information officer', 'Garment/textile technologist',
'Multimedia specialist', 'Trade union research officer',
'Museum/gallery exhibitions officer',
'Armed forces operational officer', 'Air broker',
'Mechanical engineer', 'Ceramics designer', 'Airline pilot',
'Trading standards officer', 'Advice worker', 'Music tutor',
'Leisure centre manager', 'Surveyor, rural practice',
'Scientist, physiological', 'Fisheries officer',
'Research officer, trade union', 'Licensed conveyancer',
"Nurse, children's", 'Museum/gallery curator',
'Psychologist, occupational', 'Astronomer', 'Therapist, drama',
'Therapist, speech and language', 'Surveyor, land/geomatics',
'Production assistant, radio', 'Human resources officer',
'Fast food restaurant manager', 'Orthoptist',
'Public relations officer', 'Bookseller',
'Health and safety adviser', 'Clinical cytogeneticist',
'Ergonomist', 'Psychologist, prison and probation services',
'Actuary',
'Scientist, clinical (histocompatibility and immunogenetics)',
'Community development worker', 'Consulting civil engineer',
'Television production assistant', 'Veterinary surgeon',
```

```
'Teacher, adult education', 'Civil engineer, contracting',
'Architectural technologist', 'Volunteer coordinator',
'Primary school teacher', 'Insurance underwriter',
'Research officer, political party',
'Radiation protection practitioner', 'Psychotherapist, child',
'Interior and spatial designer', 'Therapist, nutritional',
'Jewellery designer', 'Press sub',
'Clinical scientist, histocompatibility and immunogenetics',
'Administrator, sports', 'Insurance account manager',
'Museum/gallery conservator', 'Furniture designer',
'Haematologist', 'Associate Professor', 'Physicist, medical',
'Pathologist', 'Chartered public finance accountant', 'Printmaker',
'Surveyor, mining', 'Chief Marketing Officer',
'General practice doctor', 'Chemical engineer',
'Forensic scientist', 'Marketing executive', 'Art gallery manager',
'Therapist, sports', 'Insurance claims handler',
'Secondary school teacher',
'Development worker, international aid', 'Quality manager',
'Conservator, furniture', 'Tour manager',
'Control and instrumentation engineer', 'Adult nurse',
'Diplomatic Services operational officer', 'Cartographer',
'Chiropractor', 'Land/geomatics surveyor', 'Statistician',
'Financial trader', 'Special effects artist',
'Clinical psychologist', 'Further education lecturer',
'Engineer, water', 'Energy manager', 'Education administrator',
'Art therapist', 'Television floor manager', 'Legal secretary',
'Merchandiser, retail', 'Web designer',
'Nurse, learning disability',
'International aid/development worker', 'Warehouse manager',
'Engineer, mining', 'Exhibition designer',
'Administrator, local government', 'Water engineer',
'Physiotherapist', 'Engineer, electronics', 'Equities trader',
'Telecommunications researcher', 'Hydrogeologist',
'Community education officer', 'Engineer, energy',
'Scientist, audiological', 'Patent examiner', 'Retail manager',
'Engineer, aeronautical', 'Engineer, site',
'Engineer, civil (contracting)', 'Proofreader',
'Scientist, marine', 'Speech and language therapist',
'IT sales professional', 'Buyer, retail', 'Network engineer',
'Commercial art gallery manager',
'Chartered legal executive (England and Wales)',
'Presenter, broadcasting', 'Surveyor, planning and development',
'Research scientist (physical sciences)', 'Commissioning editor',
'Operational investment banker', 'Seismic interpreter',
```

```
'Charity officer', 'English as a foreign language teacher',
'Scientist, research (medical)', 'Designer, interior/spatial',
'Lexicographer', 'Therapist, art', 'Clinical embryologist',
'Child psychotherapist', 'Midwife', 'Pensions consultant',
'Tree surgeon', 'Health physicist', 'Artist', 'Company secretary',
'Fashion designer', 'IT consultant', 'Teacher, early years/pre',
'Geographical information systems officer',
'Tourist information centre manager', 'Biomedical engineer',
'Biomedical scientist', 'Financial risk analyst',
'Multimedia programmer', 'Engineer, control and instrumentation',
'Insurance broker', 'Drilling engineer',
'Development worker, community', 'Designer, industrial/product',
'Medical technical officer', 'Advertising account executive',
'Counselling psychologist', 'Tourism officer', 'Dancer',
'Social research officer, government', 'Teacher, music',
'Translator', 'Race relations officer',
'Engineer, civil (consulting)',
'Historic buildings inspector/conservation officer',
'Financial manager', 'Financial controller', 'Designer, jewellery',
'Retail banker',
'Administrator, charities/voluntary organisations',
'Magazine features editor', 'Psychotherapist, dance movement',
'Barista', 'Passenger transport manager', 'Mining engineer',
'Administrator, education',
'Programme researcher, broadcasting/film/video', 'Ranger/warden',
'Actor', 'Pension scheme manager', 'Investment analyst',
'Physiological scientist', 'Advertising art director',
'Sports development officer', 'Manufacturing systems engineer',
'Accommodation manager', 'Television/film/video producer',
'Accountant, chartered', 'Engineer, agricultural',
'Horticultural therapist', 'Economist',
'Training and development officer', 'Engineer, maintenance',
'Logistics and distribution manager', 'Psychologist, clinical',
'Accountant, chartered management', 'Rural practice surveyor',
'Biochemist, clinical', 'Set designer', 'Nutritional therapist',
'Illustrator', 'Designer, exhibition/display',
'Armed forces training and education officer', 'Location manager',
'Charity fundraiser', 'Community pharmacist',
'Geophysicist/field seismologist', 'Designer, fashion/clothing',
'Computer games developer', 'Acupuncturist',
'Database administrator', 'Stage manager', 'Operations geologist',
'Marine scientist', 'Glass blower/designer', 'Corporate treasurer',
'Ecologist', 'Structural engineer', 'Housing manager/officer',
'Chief Operating Officer', 'Engineer, manufacturing systems',
```

```
'Herbalist', 'Editor, film/video', 'Retail buyer',
'Doctor, hospital', 'Prison officer', 'Ophthalmologist',
'Chemist, analytical', 'Chartered certified accountant',
'Industrial buyer', 'Video editor', 'Publishing rights manager',
'Engineer, drilling', 'Food technologist', 'Arboriculturist',
'Engineer, technical sales', 'Systems developer', 'Firefighter',
'Education officer, museum', 'Media buyer', 'Records manager',
'Aid worker', 'Pilot, airline', 'Advertising account planner',
'Psychologist, counselling', 'Environmental consultant', 'Copy',
'Trade mark attorney', 'Psychologist, forensic', 'Social worker',
'Administrator', 'Agricultural consultant',
'Education officer, community', 'Management consultant',
'Field trials officer', 'Graphic designer',
'Teacher, primary school', 'Homeopath', 'Cabin crew',
'Editor, magazine features', 'Medical physicist',
'Medical laboratory scientific officer', 'Press photographer',
'Field seismologist', 'Estate manager/land agent',
'Industrial/product designer', 'Software engineer',
'Air cabin crew', 'Freight forwarder', 'Engineer, structural',
'Fitness centre manager', 'Interpreter',
'Scientific laboratory technician', 'Data scientist',
'Electrical engineer', 'Clinical research associate',
'Engineering geologist', 'Call centre manager',
'Psychologist, educational', 'Conservator, museum/gallery',
'Emergency planning/management officer', 'Communications engineer',
'Conservation officer, historic buildings', 'Cytogeneticist',
'Personnel officer', 'Dramatherapist',
'Investment banker, corporate', 'Camera operator',
'Chartered loss adjuster', 'Health promotion specialist',
'Scientist, product/process development', 'Learning mentor',
'Lecturer, higher education',
'Sound technician, broadcasting/film/video',
'Restaurant manager, fast food', 'Engineer, maintenance (IT)',
'Energy engineer', 'Dispensing optician',
'Chief Executive Officer', 'Ambulance person',
'Public affairs consultant', 'Product designer',
'Community arts worker', 'Higher education careers adviser',
'Dentist', 'Exhibitions officer, museum/gallery', 'Futures trader',
'Commercial/residential surveyor', 'Engineer, production',
'Animal technologist', 'Banker', 'Programmer, applications',
'Best boy', 'Secretary/administrator', 'Journalist, magazine',
'Production engineer', 'Accountant, chartered public finance',
'Geologist, engineering', 'Aeronautical engineer',
'Engineer, chemical', 'Forensic psychologist',
```

```
       'Broadcast journalist', 'Town planner', 'Toxicologist', 'Writer'],
      dtype=object)
```

In [33]:

```
# Well then, how many unique jobs are there and will this variable help us out much?
len(df['Job'].unique())
```

Out[33]:

```
639
```

In [34]:

```
df['Children'].unique()
```

Out[34]:

```
array([nan,  1.,  4.,  0.,  3.,  2.,  7.,  5.,  9.,  6., 10.,  8.])
```

In [35]:

```
df['Age'].unique()
```

Out[35]:

```
array([68., 27., 50., 48., 83., nan, 49., 86., 23., 56., 30., 39., 63.,
       60., 61., 52., 75., 77., 47., 70., 69., 45., 40., 82., 26., 25.,
       66., 72., 41., 44., 43., 84., 59., 31., 51., 58., 73., 33., 42.,
       81., 87., 54., 67., 46., 24., 20., 71., 32., 29., 80., 53., 79.,
       65., 35., 34., 74., 55., 76., 57., 38., 78., 19., 36., 88., 62.,
       37., 28., 22., 85., 89., 18., 21., 64.])
```

In [36]:

```
# Examine age range
age_range = df['Age'].unique()
print(sorted(age_range))
```

```
[23.0, 25.0, 26.0, 27.0, 30.0, 31.0, 39.0, 40.0, 41.0, 43.0, 44.0, 45.0, 47.0, 48.0, 49.0,
50.0, 51.0, 52.0, 59.0, 61.0, 68.0, 83.0, nan, 18.0, 19.0, 20.0, 21.0, 22.0, 24.0, 28.0,
29.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 42.0, 46.0, 53.0, 54.0, 55.0, 56.0, 57.0,
58.0, 60.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0,
76.0, 77.0, 78.0, 79.0, 80.0, 81.0, 82.0, 84.0, 85.0, 86.0, 87.0, 88.0, 89.0]
```

In [37]:

```
df['Education'].unique()
```

Out[37]:

```
array(["Master's Degree", 'Regular High School Diploma',
       'Doctorate Degree', 'No Schooling Completed', "Associate's Degree",
       "Bachelor's Degree", 'Some College, Less than 1 Year',
       'GED or Alternative Credential',
       'Some College, 1 or More Years, No Degree',
       '9th Grade to 12th Grade, No Diploma',
       'Nursery School to 8th Grade', 'Professional School Degree'],
      dtype=object)
```

In [38]:

```
df['Employment'].unique()
```

Out[38]:

```
array(['Part Time', 'Retired', 'Student', 'Full Time', 'Unemployed'],
      dtype=object)
```

In [39]:

```
df['Marital'].unique()
```

Out[39]:

```
array(['Widowed', 'Married', 'Separated', 'Never Married', 'Divorced'],
      dtype=object)
```

In [40]:

```
df['Gender'].unique()
```

Out[40]:

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

In [41]:

```
df['Contract'].unique()
```

Out[41]:

```
array(['One year', 'Month-to-month', 'Two Year'], dtype=object)
```

In [42]:

```
df['PaymentMethod'].unique()
```

Out[42]:

```
array(['Credit Card (automatic)', 'Bank Transfer(automatic)',
       'Mailed Check', 'Electronic Check'], dtype=object)
```

In [43]:

```
# Display any duplicate rows in the dataframe.
data_duplicates = df.loc[df.duplicated()]
print(data_duplicates)
```

```
Empty DataFrame
Columns: [CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng,
Population, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marital,
Gender, Churn, Outage_sec_perweek, Email, Contacts, Yearly_equip_failure, Techie, Contract,
Port_modem, Tablet, InternetService, Phone, Multiple, OnlineSecurity, OnlineBackup,
DeviceProtection, TechSupport, StreamingTV, StreamingMovies, PaperlessBilling,
PaymentMethod, Tenure, MonthlyCharge, Bandwidth_GB_Year, Responses, Fixes, Replacements,
Reliability, Options, Respectful, Courteous, Listening]
Index: []

[0 rows x 51 columns]
```

In [44]:

```
# Identify the standard deviation of every numeric column in the
dataset
data_std = df_stats.std()
print(data_std)
```

```
Population              14432.698671
Children                    2.154758
Age                        20.753928
Income                  28358.469482
Outage_sec_perweek          7.025921
Email                       3.025898
Contacts                    0.988466
Yearly_equip_failure        0.635953
Tenure                     26.438904
MonthlyCharge              43.335473
Bandwidth_GB_Year        2187.396807
Responses                   1.037797
Fixes                       1.034641
Replacements                1.027977
Reliability                 1.025816
Options                     1.024819
Respectful                  1.033586
Courteous                   1.028502
Listening                   1.028633
dtype: float64
```

In [45]:

```
data_nulls = df_stats.isnull().sum()
print(data_nulls)
```

```
Customer_id                0
Interaction                0
City                       0
State                      0
County                     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
```

```
Responses               0
Fixes                   0
Replacements            0
Reliability             0
Options                 0
Respectful              0
Courteous               0
Listening               0
dtype: int64
```

In [46]:

```
# Impute missing fields for variables Children, Age, Income,
Tenure and Bandwidth_GB_Year with median or mean

df_stats['Children'] =
df['Children'].fillna(df['Children'].median())

df_stats['Age'] = df['Age'].fillna(df['Age'].median())

df_stats['Income'] = df['Income'].fillna(df['Income'].median())

df_stats['Tenure'] = df['Tenure'].fillna(df['Tenure'].median())

df_stats['Bandwidth_GB_Year'] =
df['Bandwidth_GB_Year'].fillna(df['Bandwidth_GB_Year'].median())
```

In [47]:

```
data_nulls = df_stats.isnull().sum()

print(data_nulls)
```

```
Customer_id             0
Interaction             0
City                    0
State                   0
County                  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                    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                       0
MonthlyCharge                0
Bandwidth_GB_Year            0
Responses                    0
Fixes                        0
Replacements                 0
Reliability                  0
Options                      0
Respectful                   0
Courteous                    0
Listening                    0
dtype: int64
```

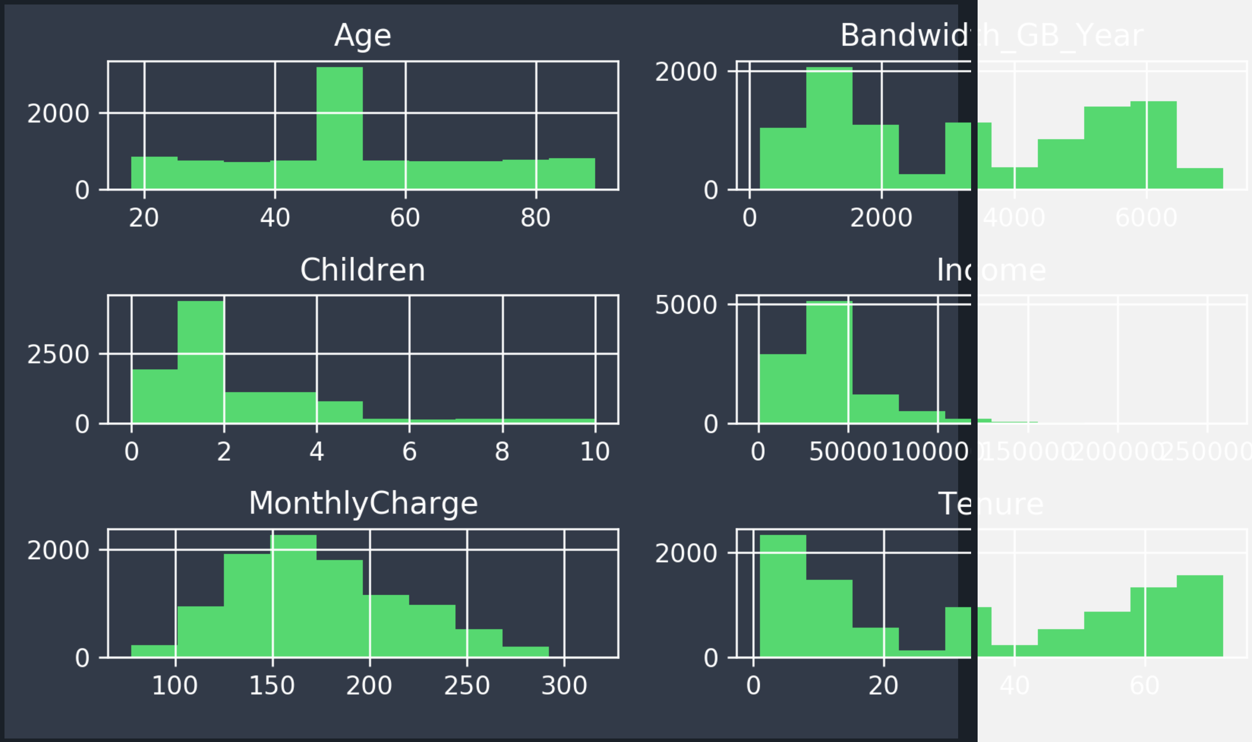
Anomaly Detection & Data Visualization

In [48]:

```
# Create histograms of important variables
df_stats[['Children', 'Age', 'Income', 'Tenure',
'MonthlyCharge', 'Bandwidth_GB_Year']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```

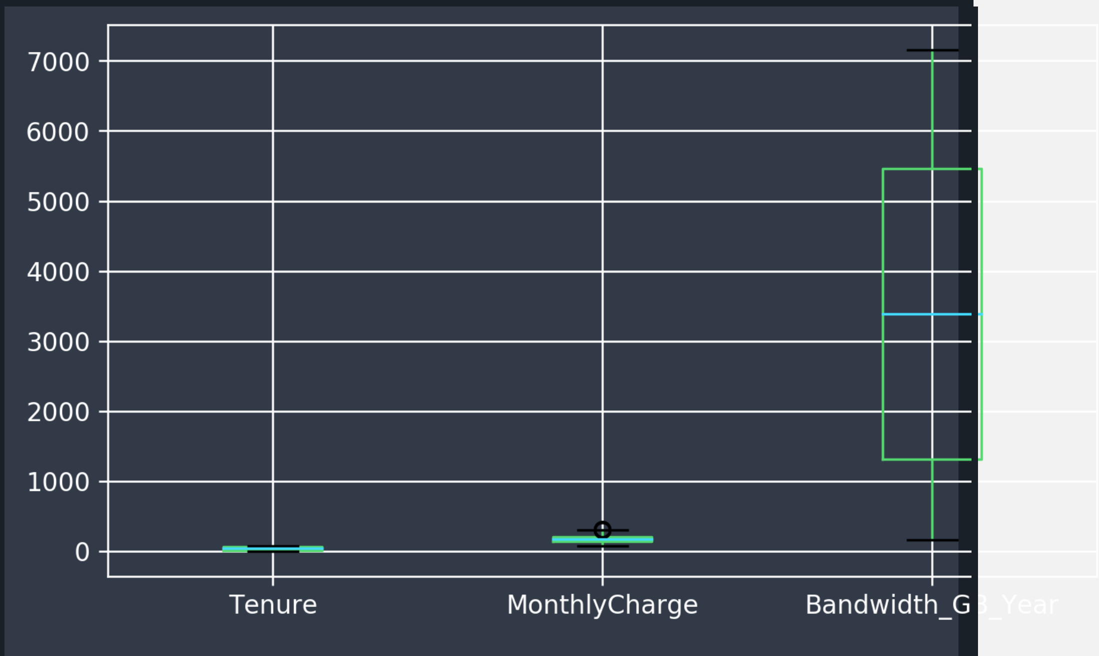
```
# plt.close()
```

```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
```



In [49]:

```
# Some odd distributions here, let's see some box plots for outliers
# Create a boxplot of user duration, payment & usage variables
df_stats.boxplot(['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'])
plt.savefig('churn_boxplots.jpg')
```

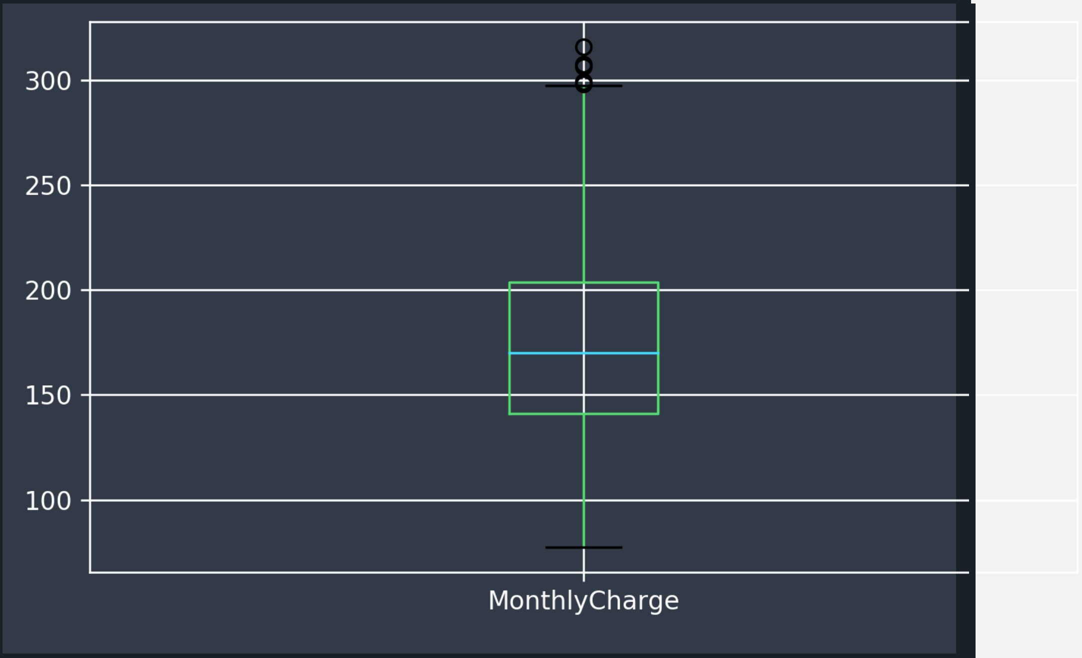


In [50]:

```
# Let's see monthly charge separately
df_stats.boxplot(['MonthlyCharge'])
# Definitely outliers but not sure how that effects PCA down the line
```

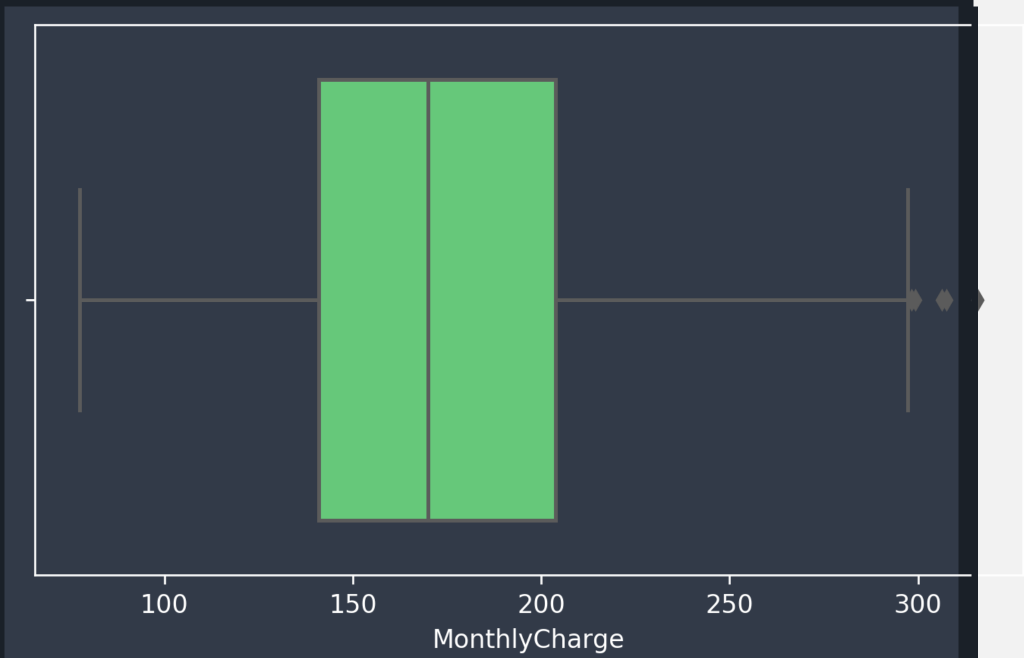
Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x150f465dc08>
```



In [51]:

```
# Let's see a Seaborn boxplot fee & bandwidth
sns.boxplot('MonthlyCharge', data = df_stats)
plt.show()
# Definitely outliers but not sure what to do with them
```



Extract Clean dataset to 'churn_clean.csv'¶

In [52]:

```
# Extract Clean dataset
df_stats.to_csv('churn_clean.csv')
```

In [53]:

```
# Reload cleaned data & remove all variable except user services
payment info and survey data
churn_user = pd.read_csv('churn_clean.csv')
```

In [54]:

```
# Slice off all but last eleven service realted variables
data = churn_user.loc[:, 'Tenure':'Listening']
```

```
data.head()
```

Out[54]:

	Tenure	MonthlyCharge	Bandwidth_GB_Year	Responses	Fixes	Replacements	Reliability	Options	Respectful	Courteous
0	6.795513	171.449762	904.536110	5	5	5	3	4	4	3
1	1.156681	242.948015	800.982766	3	4	3	3	4	3	4
2	15.754144	159.440398	2054.706961	4	4	2	4	4	3	3
3	17.087227	120.249493	2164.579412	4	4	4	2	5	4	3
4	1.670972	150.761216	271.493436	4	4	4	3	4	4	4

PCA (D206 Performance Lab)¶

In [55]:

```
# Import Scikit Learn PCA application
from sklearn.decomposition import PCA
```

In [56]:

```
# Normalize the data
churn_normalized = (data - data.mean()) / data.std()
```

In [57]:

```
# Select number of components to extract
pca = PCA(n_components = data.shape[1])
```

In [58]:

```
# Create a list of PCA names
churn_numeric = data[['Tenure', 'MonthlyCharge',
'Bandwidth_GB_Year', 'Responses',
                       'Fixes', 'Replacements', 'Reliability',
'Options',
                       'Respectful', 'Courteous', 'Listening']]
pcs_names = []
for i, col in enumerate(churn_numeric.columns):
    pcs_names.append('PC' + str(i + 1))
print(pcs_names)
```

```
['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11']
```

In [59]:

```
# Re-Extract Clean dataset
churn_numeric.to_csv('churn_clean_final.csv')
```

In [60]:

```
churn_numeric.head()
```

Out[60]:

	Tenure	MonthlyCharge	Bandwidth_GB_Year	Responses	Fixes	Replacements	Reliability	Options	Respectful	Courteous
0	6.795513	171.449762	904.536110	5	5	5	3	4	4	3
1	1.156681	242.948015	800.982766	3	4	3	3	4	3	4
2	15.754144	159.440398	2054.706961	4	4	2	4	4	3	3
3	17.087227	120.249493	2164.579412	4	4	4	2	5	4	3
4	1.670972	150.761216	271.493436	4	4	4	3	4	4	4

In [61]:

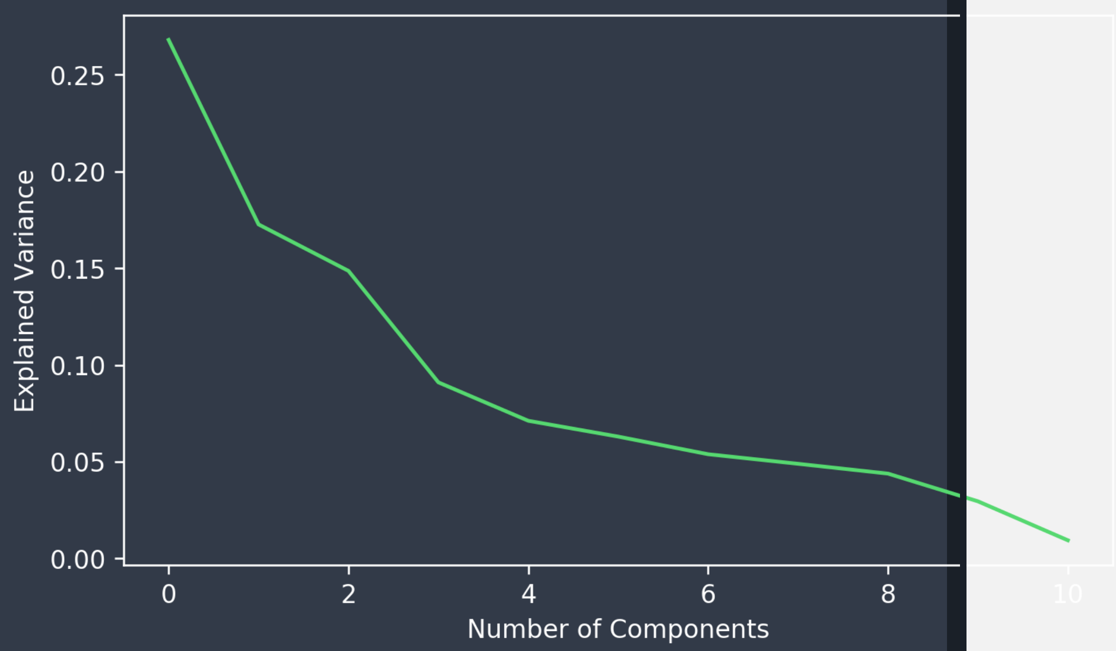
```
# Call PCA application & convert the dataset of  19 variables
into a dataset of 19 components
pca.fit(churn_normalized)
churn_pca = pd.DataFrame(pca.transform(churn_normalized),
                         columns = pcs_names)
```

In [62]:

```
# For a scree plot import matplotlib & seaborn libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

In [63]:

```
# Run the scree plot
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance')
plt.show();
```

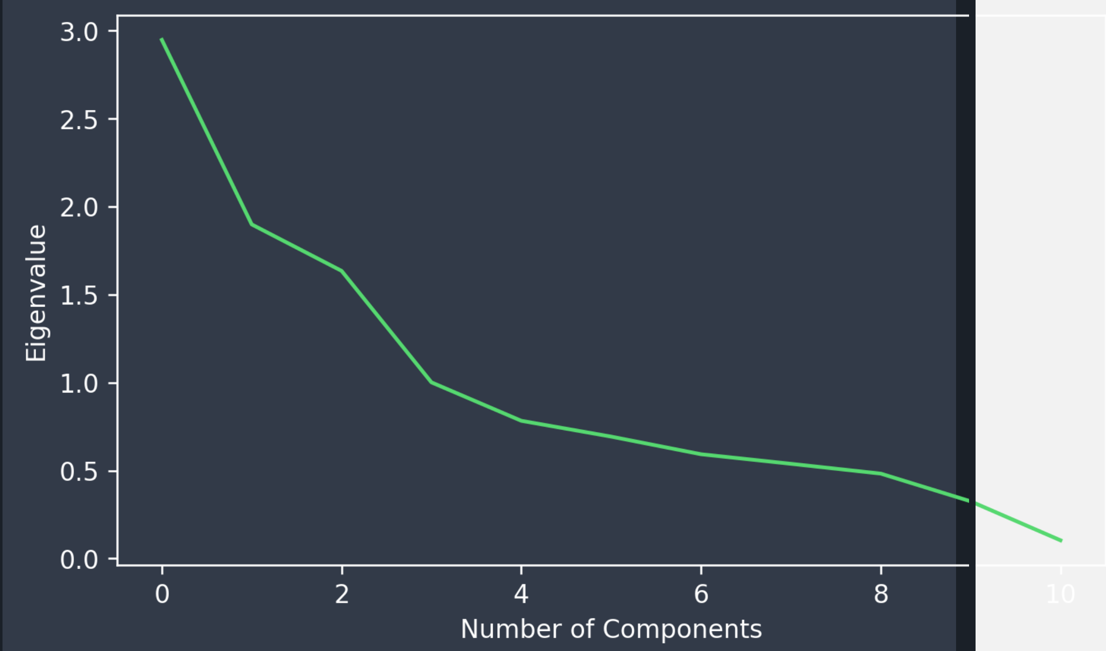


In [64]:

```
# Extract the eigenvalues
cov_matrix = np.dot(churn_normalized.T, churn_normalized) /
data.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix,
eigenvector)) for eigenvector in pca.components_]
```

In [65]:

```
# Plot the eigenvalues
plt.plot(eigenvalues)
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalue')
plt.show();
```



In [66]:

```
# Select the fewest components
for pc, var in zip(pcs_names,
np.cumsum(pca.explained_variance_ratio_)):
    print(pc, var)
```

```
PC1 0.26792660623963066
PC2 0.44053732880229357
PC3 0.5891444522248594
PC4 0.6801579842425083
PC5 0.7513217054372497
PC6 0.8143117094101318
PC7 0.8681970531672316
PC8 0.9171387362588829
PC9 0.9610122820002187
PC10 0.9905736363144831
PC11 1.0
```

In [67]:

```
# Above, we see that 89% of variance is explained by 14
components

# Create a rotation

rotation = pd.DataFrame(pca.components_.T, columns = pcs_names,
index = churn_numeric.columns)

print(rotation)
```

```
                       PC1       PC2       PC3       PC4       PC5       PC6  \
Tenure           -0.010403  0.701838 -0.072209 -0.063594  0.005683 -0.011155
MonthlyCharge     0.000317  0.041147 -0.014151  0.996995 -0.022136  0.015231
Bandwidth_GB_Year -0.012166  0.703079 -0.074222  0.004399  0.009590  0.003466
Responses         0.458932  0.031325  0.281154  0.018568 -0.070233 -0.119149
Fixes             0.434134  0.042559  0.282404  0.007508 -0.106632 -0.169752
Replacements      0.400639  0.034665  0.281118 -0.019631 -0.173742 -0.255336
Reliability       0.145799 -0.050367 -0.567815 -0.010310 -0.171334 -0.483328
Options          -0.175633  0.066334  0.587335 -0.000047  0.135949  0.060124
Respectful        0.405207 -0.012680 -0.183447  0.004596 -0.062342  0.064609
Courteous         0.358342 -0.003886 -0.181697 -0.027959 -0.182406  0.806166
Listening         0.308925 -0.017396 -0.131173  0.015574  0.931612 -0.011133

                       PC7       PC8       PC9      PC10      PC11
Tenure            0.007419 -0.011527  0.006935  0.003286 -0.705445
MonthlyCharge    -0.018038 -0.004316  0.023690 -0.013785 -0.047865
Bandwidth_GB_Year  0.003701 -0.002364 -0.008068  0.008529  0.706925
Responses        -0.045963  0.025431 -0.240574  0.793237 -0.004306
Fixes            -0.065414  0.074400 -0.592131 -0.573832 -0.002217
Replacements     -0.146887 -0.396333  0.673088 -0.177665  0.014933
Reliability      -0.443353  0.431528  0.087207  0.018301  0.002283
Options          -0.209767  0.693861  0.265474 -0.042012 -0.002514
Respectful        0.757954  0.402835  0.230319 -0.063972  0.001604
Courteous        -0.379136  0.067889  0.067293 -0.040946 -0.006875
Listening        -0.113297 -0.045132  0.046107 -0.042251 -0.002357
```

In [68]:

```
# Output loadings for components
loadings = pd.DataFrame(pca.components_.T,
                        columns = pcs_names,
                        index = data.columns)
loadings
```

Out[68]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
Tenure	-0.010403	0.701838	-0.0722090	-0.063594	0.005683	-0.011155	0.007419	-0.011527	0.006935	0.003286	-0.705445
MonthlyCharge	0.0003170	0.041147	-0.014151	0.996995	-0.022136	0.015231	-0.018038	-0.004316	0.023690	-0.013785	-0.047865
Bandwidth_GB_Year	-0.012166	0.703079	-0.074222	0.0043990	0.0095900	0.0034660	0.003701	-0.002364	-0.008068	0.0085290	0.706925
Responses	0.4589320	0.0313250	0.2811540	0.018568	-0.0702330	-0.1191490	-0.045963	0.025431	-0.240574	0.793237	-0.004306
Fixes	0.4341340	0.0425590	0.2824040	0.007508	-0.1066320	-0.1697520	-0.065414	0.074400	-0.592131	-0.573832	-0.002217
Replacements	0.4006390	0.0346650	0.281118	-0.0196310	-0.1737420	-0.2553360	-0.146887	-0.396333	0.673088	-0.177665	0.014933
Reliability	0.145799	-0.0503670	-0.5678150	-0.0103100	-0.1713340	-0.4833280	-0.443353	0.431528	0.087207	0.018301	0.002283
Options	-0.175633	0.0663340	0.587335	-0.000047	0.1359490	0.060124	-0.209767	0.693861	0.265474	-0.042012	-0.002514
Respectful	0.405207	-0.0126800	-0.183447	0.004596	-0.062342	0.0646090	0.757954	0.402835	0.230319	-0.063972	0.001604
Courteous	0.358342	-0.0038860	-0.181697	-0.0279590	-0.182406	0.806166	-0.379136	0.067889	0.067293	-0.040946	-0.006875
Listening	0.308925	-0.0173960	-0.131173	0.0155740	0.931612	-0.0111330	-0.113297	-0.045132	0.046107	-0.042251	-0.002357

In [69]:

```
# Finally, extract reduced dataset & print 3 components
churn_reduced = churn_pca.iloc[ : , 0:3]
print(churn_reduced)
```

```
           PC1       PC2       PC3
0     1.923875 -1.421955  1.903125
1    -0.199798 -1.706801  0.538766
2    -0.667923 -0.985940  0.227390
3     0.046465 -0.730628  2.282040
4     1.326741 -1.924880  0.825729
...        ...       ...       ...
9995 -2.097964  1.961837  0.104147
9996  1.917485  1.645946  0.611009
9997  1.431918  0.323573  0.028288
9998  2.011460  2.187756 -0.079864
9999 -2.266364  1.591986 -0.819973
```

```
[10000 rows x 3 columns]
```

Part III: Data Cleaning¶

D. Data Cleaning Summary¶

D1.Cleaning Findings: There was much missing data with meaningful variable fields including Children, Age, Income, Tenure and Bandwidth_GB_Year. Given mean and variance of these variables, it seemed reasonable to impute missing values with median values. Many categorical (such as whether or not the customer was "Techie") & non-numeric (columns for customer ID numbers & related customer transaction IDs) data were not included in analysis given they seemed less meaningful to interpretation and decision-making. The anomalies discovered were not significant & were mitigated as follows.
D2.Justification of Mitigation Methods: Mitigated missing values with imputation using median values. MonthlyCharge variable shows outliers so left alone. Does not seem significant to this analysis.
D3.Summary of Outcomes: Cleaned dataset to leave remaining variables describing customer tenure, monthly service charge, yearly bandwidth usage & responses to survey. Seems pretty straightforward at this point.
D4.Mitigation Code: (see code above and Panopto recording)
D5.Clean Data: (see attached file 'churn_clean.csv')
D6.Limitations: Limitations given the telecom company data set are that the data are not coming from an warehouse. In this scenario, it is as though I initiated and gathered the data. So, I am not able to reach out to the staff that organized & gathered this information to ask them why certain NAs are there, why are fields such as age or yearly bandwidth usage missing information that might be relevant to answering questions about customer retention or churn. In a real world project, you would be able to go down to the department where these folks worked and fill in the empty fields or discover why fields are left blank.

Commented [PCEV18-4]:
Sentence Fluency: Sentence fragment
→Sentence fluency concerns recur.

WGU's Guide to Academic Writing
Link: Module 5.24: Sentence Fragments

Commented [PCEV18-5]:
Parts of Speech: Incorrect article
→Parts of speech errors recur.

WGU's Guide to Academic Writing
Link: Module 8.12: Article Usage

D7. The accurate factual data for missing fields that might be recoverable given the ability to access the staff in the data warehouse, such as company tenure with the company, may give a slighty different overall picture to the analysis. While imputation may provide a path to move forward and give decision-makers reasonable answers, there really is no reason for these data to be missing. The limitations here could be remedied by instituting stricter data acquisition procedures, follow-ups & feedback.

E. PCA Application¶

E1. Principal Components: The principal components, and what I determined to be "most important", in this dataset include survey responses to:

- **Timely Responses**
- **Timely Fixes**
- **Timely Replacements**
- **Respectful Response**

E2. Criteria Used: I had a sneaking intuitition that these might be the most important features to predict why a customer might or might not leave a company. However, confirmation bias is not what confirmed this forethought. I used a scree plot & extracted the eigenvalues for visualization of where the "elbow was bending". The elbow bent at about 3 but kept an eigenvalue above 1 until the tenth component. Then, the fewest number of components were selected based on the 89% explanation using the Numpy cumsum method. A rotation & loadings were created which suggested the "most important" features of the dataset.
E3. Benefits: Their is as the loadings suggest the variables involved in timely action with regard to customer satisfaction should be given greater emphasis and hopefully help reduce the the churn rate from the large number of 27% & "increase the retention period of customers" by targeting more resources in the direction prompt customer service (Ahmad, 2019).

Commented [PCEV18-6]:
Word Choice: Repeated words
→Word choice errors recur.

WGU's Guide to Academic Writing
Link: Module 8.27: Vocabulary in Academic Writing

Again, this seems an intuitive result but now decision-makers in the company of reasonable verification of what might have been a "hunch".

Part IV: Supporting Documents¶

F. Video¶

(see Panopto recording)

G. Sources for Third-Party Code¶

Cmdline. (2018, March 20). *How To Change Column Names and Row Indexes in Pandas?* Python and R Tips. https://cmdlinetips.com/2018/03/how-to-change-column-names-and-row-indexes-in-pandas/

Larose, C. D. & Larose, D. T. (2019). *Data Science: Using Python and R.* John Wiley & Sons, Inc.

Poulson, B. (2016). *Data Science Foundations: Data Mining* LinkedIn Learning. https://www.linkedin.com/learning/data-science-foundations-data-mining/anomaly-detection-in-python?u=2045532

Sree. (2020, October 26). *Predict Customer Churn in Python.* Towards Data Science. https://towardsdatascience.com/predict-customer-churn-in-python-e8cd6d3aaa7

VanderPlas, J. (2017). *Python Data Science Handbook: Essential Tools for Working with Data.* O'Reilly Media, Inc.

H. Sources¶

Ahmad, A. K., Jafar, A & Aljoumaa, K. (2019, March 20). *Customer churn prediction in telecom using machine learning in big data platform*. Journal of Big Data.

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0191-6

Altexsoft. (2019, March 27). *Customer Churn Prediction Using Machine Learning: Main Approaches and Models*. Altexsoft. https://www.altexsoft.com/blog/business/customer-churn-prediction-for-subscription-businesses-using-machine-learning-main-approaches-and-models/

Frohbose, F. (2020, November 24). *Machine Learning Case Study: Telco Customer Churn Prediction*. Towards Data Science. https://towardsdatascience.com/machine-learning-case-study-telco-customer-churn-prediction-bc4be03c9e1d

Mountain, A. (2014, August 11). *Data Cleaning*. Better Evaluation. https://www.betterevaluation.org/en/evaluation-options/data_cleaning