WGU D212 Data Mining II

Task 2 - Dimensionality Reduction Methods

Ednalyn C. De Dios

August 24, 2023

Environment

- Python: 3.9.9
- Jupyter: 7.0.2

Part I - Research Question

A1. Propose one question relevant to a real-world organizational situation that you will answer by using principal component analysis (PCA).

> ***Can the principal components of the customer base be identified using Principal Component Analysis?***

A2. Define one goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

The ultimate goal of this data analysis is to reduce operating costs by increasing the efficiency of the organization's marketing efforts. We will use Principal Component Analysis (PCA) as a dimensionality reduction technique to identify the principal components of the data set. This will inform the decisions of stakeholders in matters where customer retention is involved, for example. Knowing which principal components accounts for the variance will provide the organization with advanced insight towards that customer's characteristics or behavior. Thereby increasing the effectiveness of marketing campaigns.

Part II - Method Justification

B1. Explain how PCA analyzes the selected data set. Include expected outcomes.

PCA analyzes the selected by first stardardizing the data set and fitting it. Then, the variances of each component were plotted to discern the optimal number of k clusters using the elbow method. After choosing the number of k clusters, we then fit and transform the data set again with designated k number of clusters. The variances for each of the principal components were noted and then added to calculate the total variance captured by the principal components.

B2. Summarize one assumption of PCA.

PCA or Principal Component Analysis makes multiple assumptions about the data set but the biggest in my opinion is that variable should have correlation (Sharma, 2021). Data sets with high correlation between variables works best for PCA so that it can reduce the number of dimensions in the data set. Otherwise, there will be no reduction in dimensionality that can take place.

Part III - Data Preparation

C1. Identify the continuous dataset variables that you will need in order to answer the PCA question proposed in part A1.

In [1]:
```
columns = ['Population',
           'Children',
           'Age',
           'Income',
           'Outage_sec_perweek',
           'Email',
           'Contacts',
           'Yearly_equip_failure',
           'Tenure',
           'MonthlyCharge',
           'Bandwidth_GB_Year',
          ]
```

In [2]:
```
# setting the random seed for reproducibility
import random
random.seed(493)

# for manipulating dataframes
import pandas as pd
import numpy as np

# for visualizations
```

```python
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
from IPython.display import Image

# for modeling
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# to print out all the outputs of the cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# set display options
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', None)
```

In [3]:

```python
# read the csv file
df = pd.read_csv('churn_clean.csv')
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 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   UID                   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              10000 non-null  int64  
 15  Age                   10000 non-null  int64  
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object 
 18  Gender                10000 non-null  object 
 19  Churn                 10000 non-null  object 
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64  
 22  Contacts              10000 non-null  int64  
 23  Yearly_equip_failure  10000 non-null  int64  
 24  Techie                10000 non-null  object 
 25  Contract              10000 non-null  object 
 26  Port_modem            10000 non-null  object 
 27  Tablet                10000 non-null  object 
 28  InternetService       7871 non-null   object 
 29  Phone                 10000 non-null  object 
 30  Multiple              10000 non-null  object 
 31  OnlineSecurity        10000 non-null  object 
 32  OnlineBackup          10000 non-null  object 
 33  DeviceProtection      10000 non-null  object 
 34  TechSupport           10000 non-null  object 
 35  StreamingTV           10000 non-null  object 
 36  StreamingMovies       10000 non-null  object 
 37  PaperlessBilling      10000 non-null  object 
 38  PaymentMethod         10000 non-null  object 
 39  Tenure                10000 non-null  float64
 40  MonthlyCharge         10000 non-null  float64
 41  Bandwidth_GB_Year     10000 non-null  float64
 42  Item1                 10000 non-null  int64  
 43  Item2                 10000 non-null  int64  
 44  Item3                 10000 non-null  int64  
 45  Item4                 10000 non-null  int64  
 46  Item5                 10000 non-null  int64  
 47  Item6                 10000 non-null  int64  
 48  Item7                 10000 non-null  int64  
 49  Item8                 10000 non-null  int64  
```

```
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

Out[3]:

	CaseOrder	Customer_id	Interaction	UID	City	S
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	

In [4]:

```python
for col in columns:
    # show outliers
    plt.boxplot(df[col])
    plt.title(col)
    fig = plt.figure(figsize =(10, 7))
```

Out[4]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a546f77760>,
  <matplotlib.lines.Line2D at 0x1a546f77a30>],
 'caps': [<matplotlib.lines.Line2D at 0x1a546f77cd0>,
  <matplotlib.lines.Line2D at 0x1a546f77f70>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a546f774c0>],
 'medians': [<matplotlib.lines.Line2D at 0x1a546f97250>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a546f974f0>],
 'means': []}
```

Out[4]:

```
Text(0.5, 1.0, 'Population')
```

Out[4]:

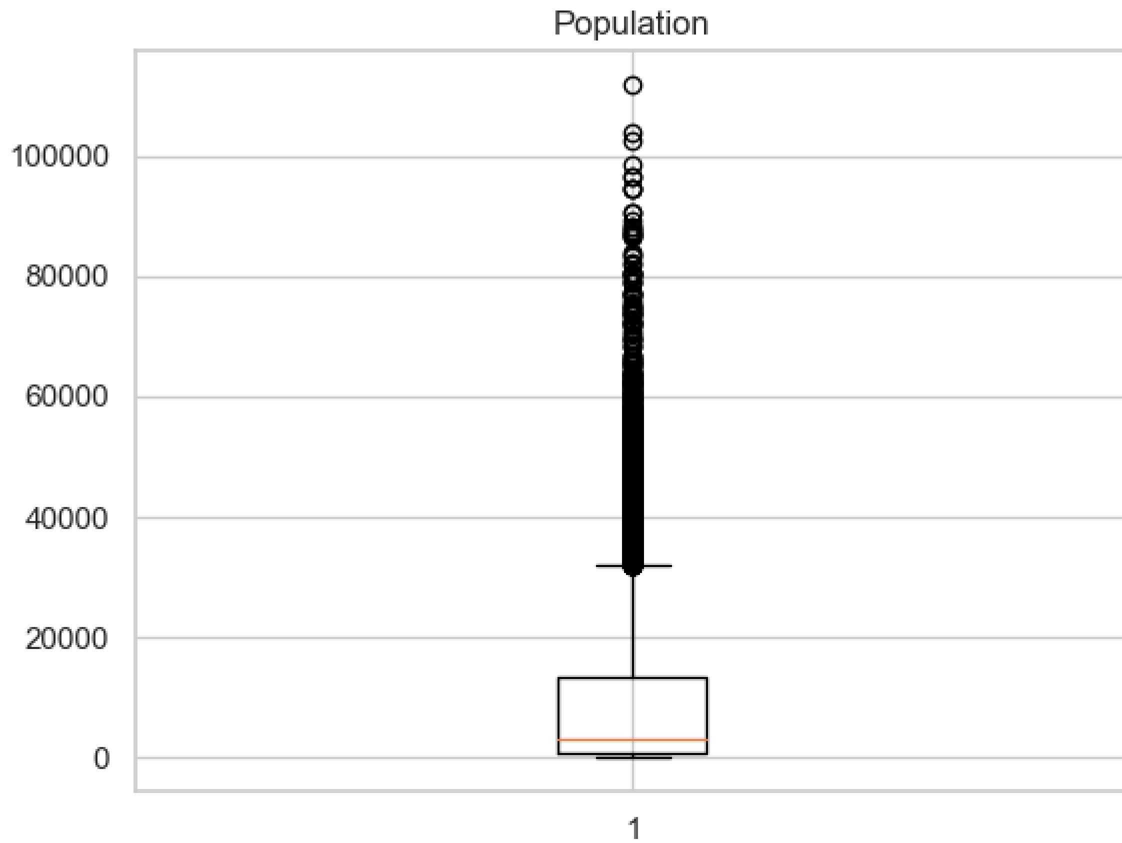
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a546fe43a0>,
  <matplotlib.lines.Line2D at 0x1a546fe4640>],
 'caps': [<matplotlib.lines.Line2D at 0x1a546fe48e0>,
  <matplotlib.lines.Line2D at 0x1a546fe4b80>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a546fe4100>],
 'medians': [<matplotlib.lines.Line2D at 0x1a546fe4e20>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a546ff4100>],
 'means': []}
```

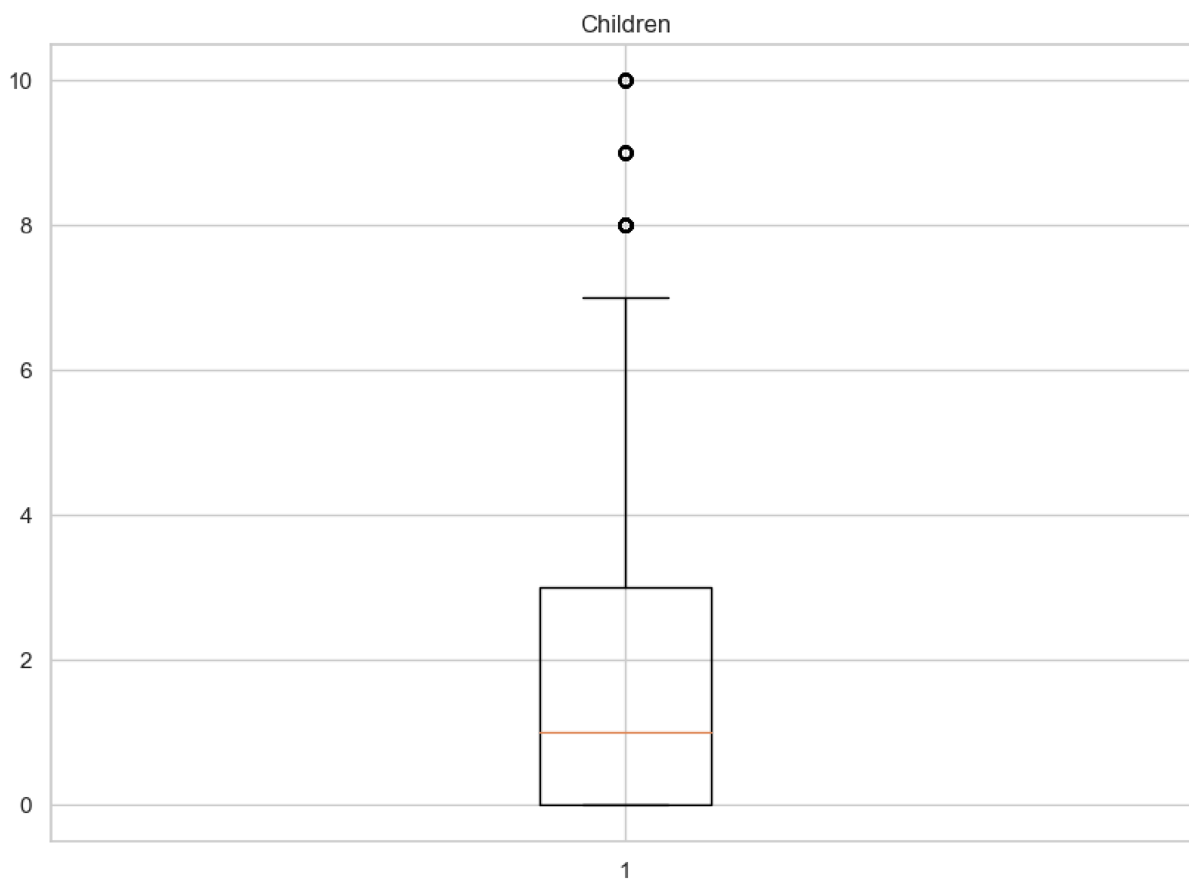
Out[4]:

```
Text(0.5, 1.0, 'Children')
```

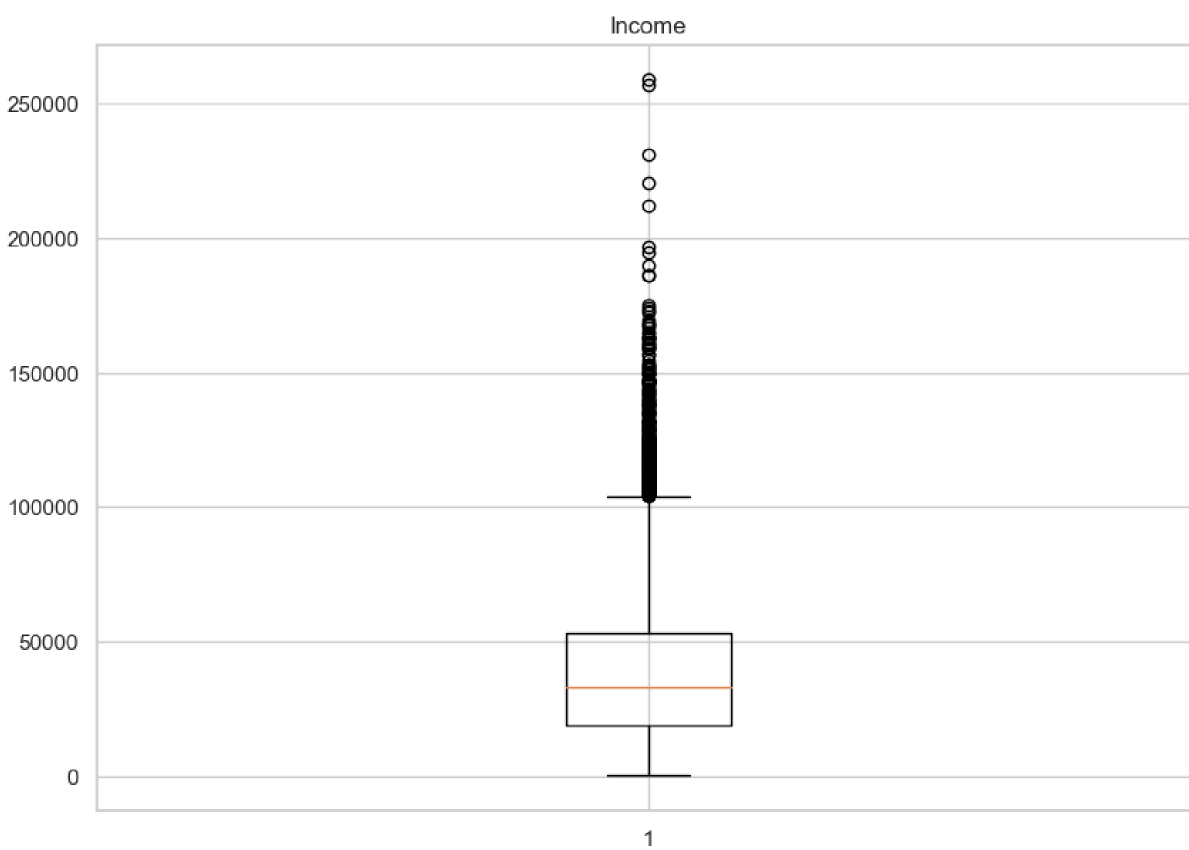
```
Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a547038100>,
          <matplotlib.lines.Line2D at 0x1a5470383a0>],
         'caps': [<matplotlib.lines.Line2D at 0x1a546fd3040>,
          <matplotlib.lines.Line2D at 0x1a547038370>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a547027e20>],
         'medians': [<matplotlib.lines.Line2D at 0x1a547038700>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a5470389a0>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Age')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a547080910>,
          <matplotlib.lines.Line2D at 0x1a547080a60>],
         'caps': [<matplotlib.lines.Line2D at 0x1a547080d30>,
          <matplotlib.lines.Line2D at 0x1a547080fd0>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a547080670>],
         'medians': [<matplotlib.lines.Line2D at 0x1a54708d2b0>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a54708d550>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Income')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a5470d53d0>,
          <matplotlib.lines.Line2D at 0x1a5470d5550>],
         'caps': [<matplotlib.lines.Line2D at 0x1a5470d57f0>,
          <matplotlib.lines.Line2D at 0x1a5470d5a90>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a5470d5130>],
         'medians': [<matplotlib.lines.Line2D at 0x1a5470d5d30>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a5470d5fd0>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Outage_sec_perweek')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a5470f5be0>,
          <matplotlib.lines.Line2D at 0x1a547119e50>],
         'caps': [<matplotlib.lines.Line2D at 0x1a54712d130>,
          <matplotlib.lines.Line2D at 0x1a54712d3d0>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a547119c70>],
         'medians': [<matplotlib.lines.Line2D at 0x1a54712d670>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a54712d910>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Email')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a547170730>,
          <matplotlib.lines.Line2D at 0x1a5471709d0>],
         'caps': [<matplotlib.lines.Line2D at 0x1a547170c70>,
          <matplotlib.lines.Line2D at 0x1a547170f10>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a5471705b0>],
         'medians': [<matplotlib.lines.Line2D at 0x1a54717f1f0>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a54717f490>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Contacts')
```

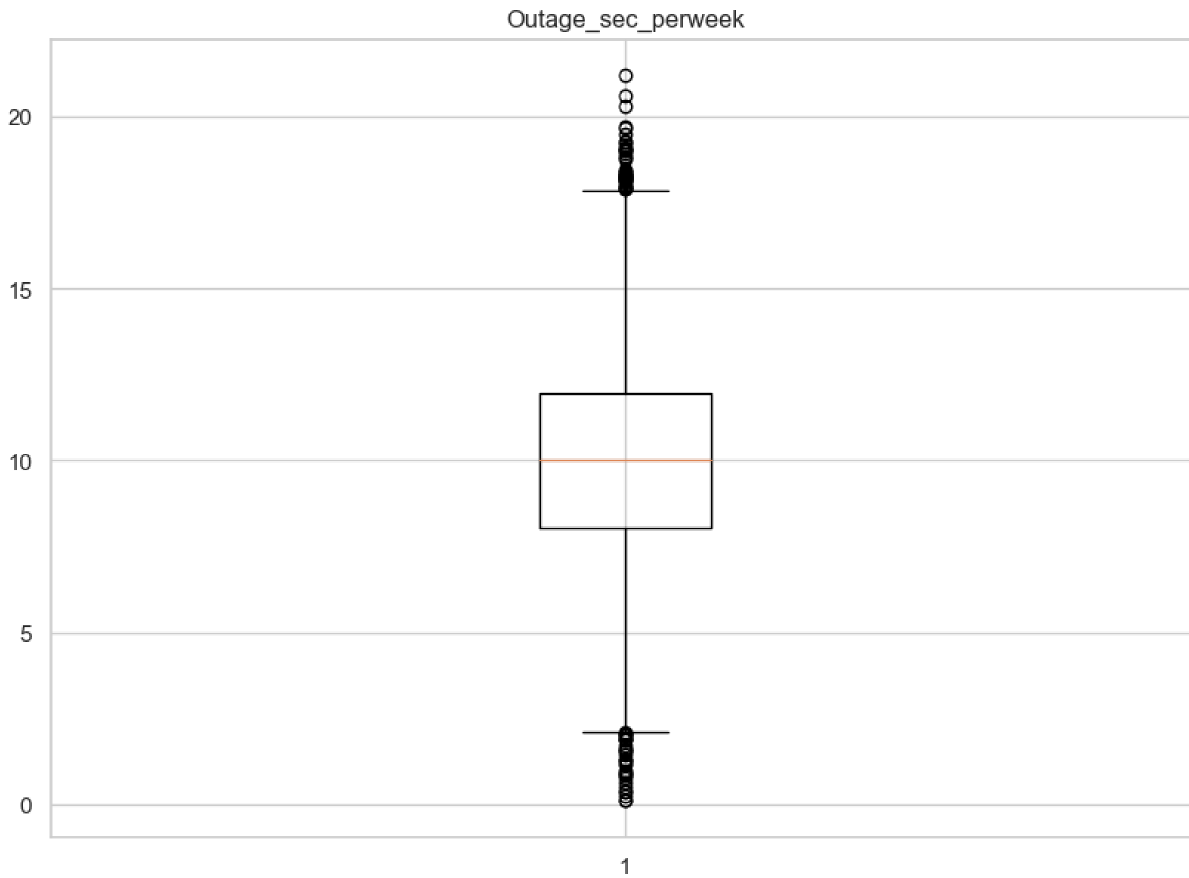
```
Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a5471c71c0>,
          <matplotlib.lines.Line2D at 0x1a5471c7460>],
         'caps': [<matplotlib.lines.Line2D at 0x1a5471c7700>,
          <matplotlib.lines.Line2D at 0x1a5471c79a0>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a5471b8ee0>],
         'medians': [<matplotlib.lines.Line2D at 0x1a5471c7c40>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a5471c7ee0>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Yearly_equip_failure')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a54720cb20>,
          <matplotlib.lines.Line2D at 0x1a54720cdc0>],
         'caps': [<matplotlib.lines.Line2D at 0x1a54721e0a0>,
          <matplotlib.lines.Line2D at 0x1a54721e340>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a54720c880>],
         'medians': [<matplotlib.lines.Line2D at 0x1a54721e5e0>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a54721e880>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Tenure')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a5472636d0>,
          <matplotlib.lines.Line2D at 0x1a547263970>],
         'caps': [<matplotlib.lines.Line2D at 0x1a547263c10>,
          <matplotlib.lines.Line2D at 0x1a547263eb0>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a547263430>],
         'medians': [<matplotlib.lines.Line2D at 0x1a547273190>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a547273430>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'MonthlyCharge')

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a5475c9160>,
          <matplotlib.lines.Line2D at 0x1a5475c9400>],
         'caps': [<matplotlib.lines.Line2D at 0x1a5475c96a0>,
          <matplotlib.lines.Line2D at 0x1a5475c9940>],
         'boxes': [<matplotlib.lines.Line2D at 0x1a5472a8e80>],
         'medians': [<matplotlib.lines.Line2D at 0x1a5475c9be0>],
         'fliers': [<matplotlib.lines.Line2D at 0x1a5475c9e80>],
         'means': []}

Out[4]: Text(0.5, 1.0, 'Bandwidth_GB_Year')
```


Population
100000
80000
60000
40000
20000
0
1


Children
10
8
6
4
2
0
1

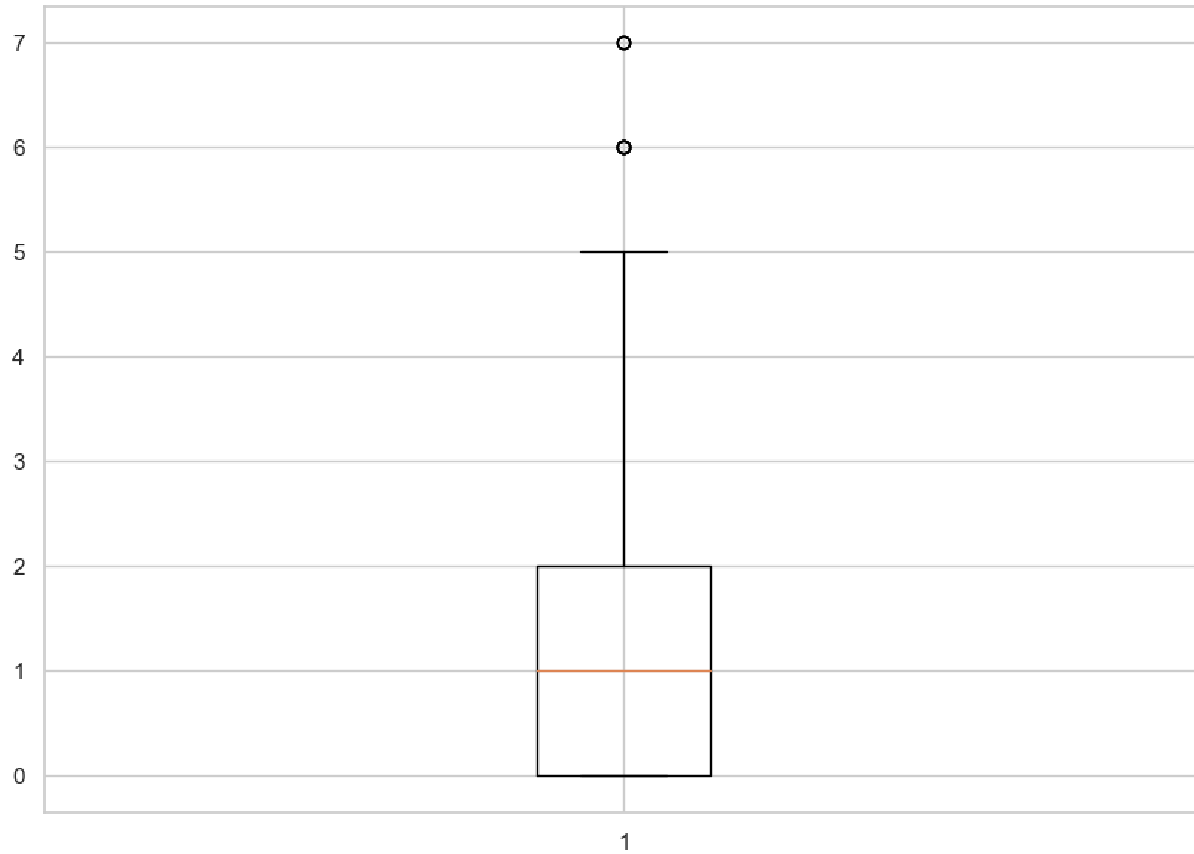

Age
90
80
70
60
50
40
30
20
1


Income
250000
200000
150000
100000
50000
0
1

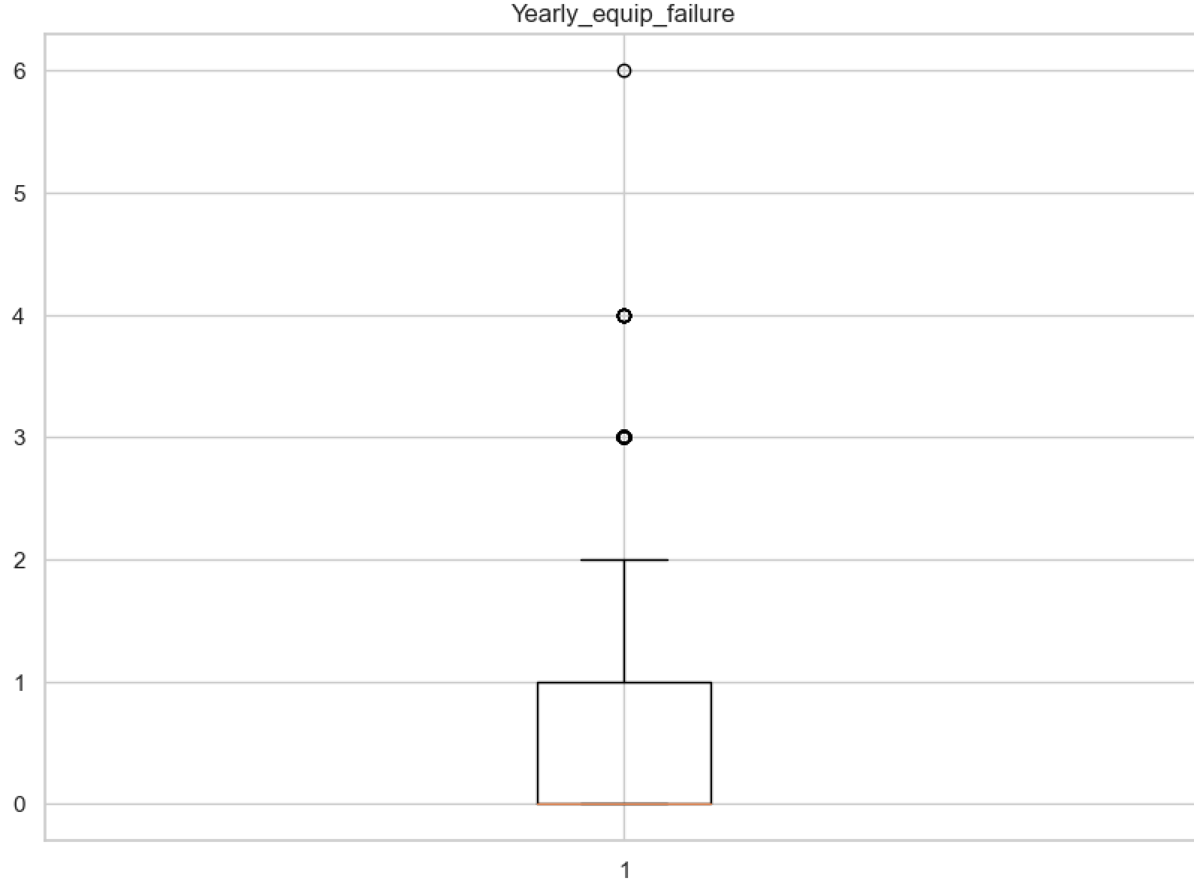

Outage_sec_perweek
20
15
10
5
0
1


Email
20
15
10
5
0
1

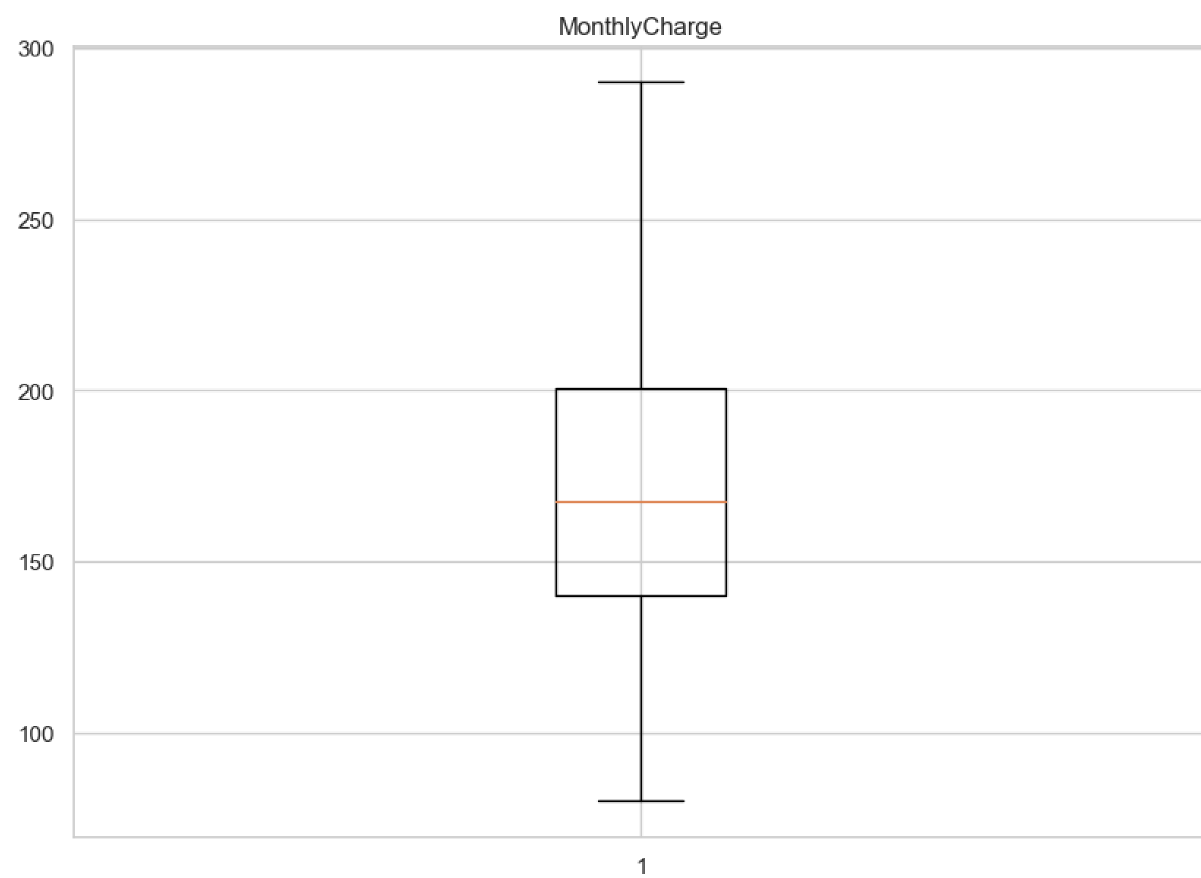
Contacts

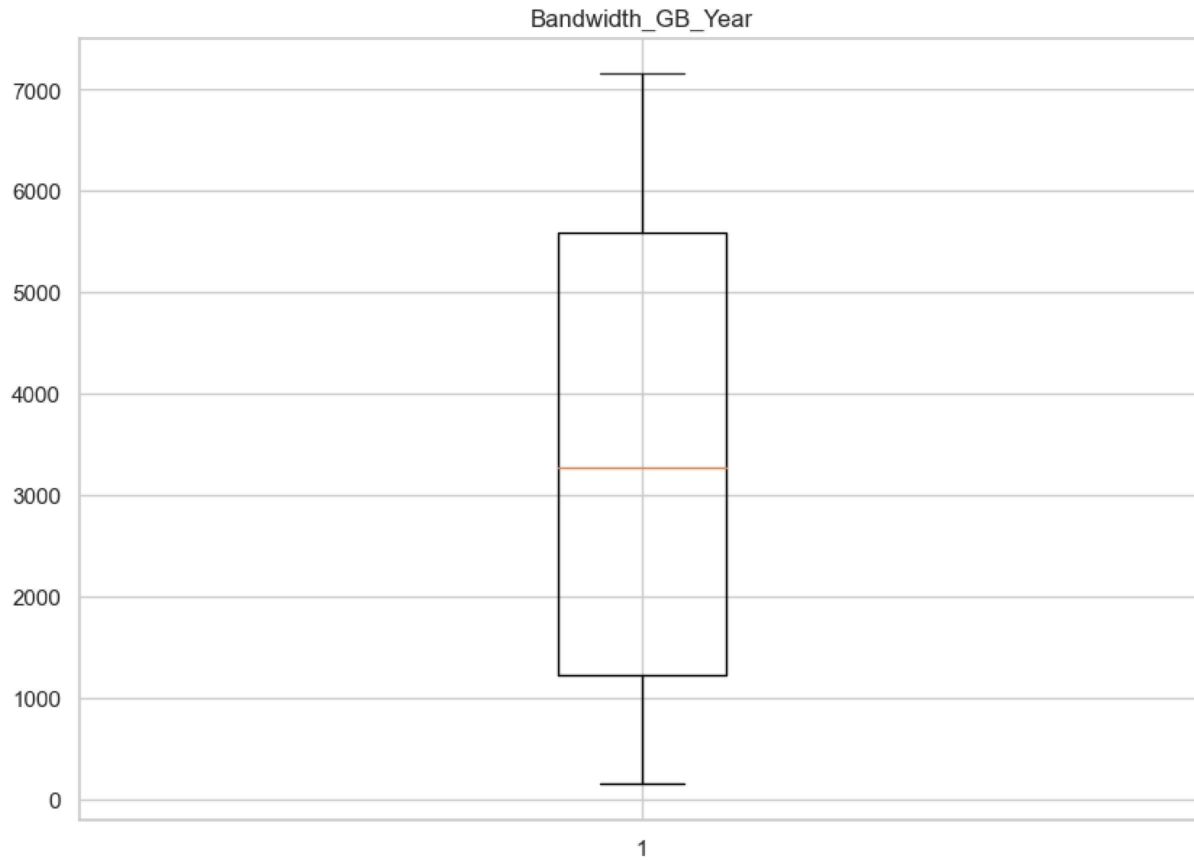


Yearly_equip_failure




Tenure
70
60
50
40
30
20
10
0
1


MonthlyCharge
300
250
200
150
100
1




```
<Figure size 1000x700 with 0 Axes>
```

In [5]:
```
# remove outliers in "Population"
df = df[df['Population'] <= 30000]
df.shape
```

Out[5]: (8957, 50)

In [6]:
```
# remove outliers in "Children"
df = df[df['Children'] < 8]
df.shape
```

Out[6]: (8600, 50)

In [7]:
```
# remove outliers in "Income"
df = df[df['Income'] <= 100000]
df.shape
```

Out[7]: (8255, 50)

In [8]:
```
# remove outliers in "Outage_sec_perweek"
df = df[df['Outage_sec_perweek'] <= 17.5]
df.shape
```

Out[8]: (8198, 50)

In [9]:
```
# remove outliers in "Email"
df = df[df['Email'] <= 20]
```

```
df.shape
```

Out[9]: (8187, 50)

In [10]:
```
# remove outliers in "Contacts"
df = df[df['Contacts'] <= 5]
df.shape
```

Out[10]: (8181, 50)

In [11]:
```
# remove outliers in "Yearly_equip_failure"
df = df[df['Yearly_equip_failure'] <= 2]
df.shape
```

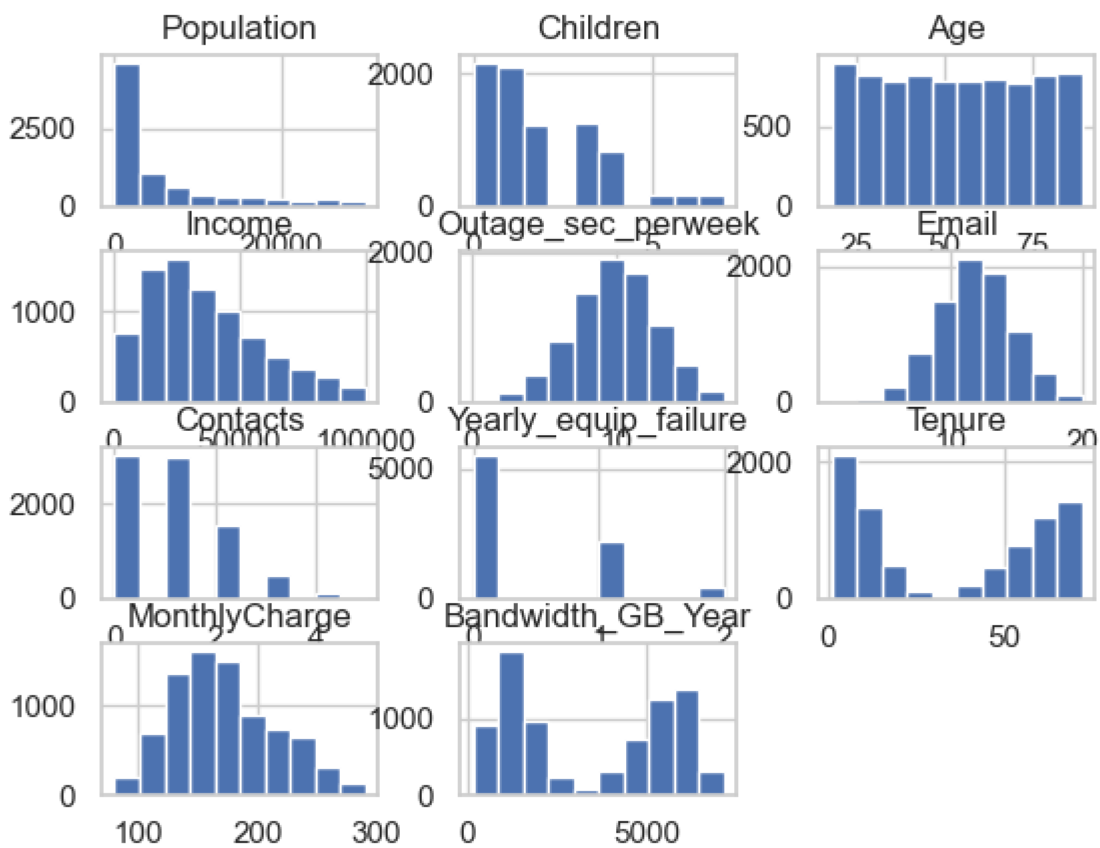
Out[11]: (8105, 50)

C2. Standardize the continuous dataset variables identified in part C1. Include a copy of the cleaned dataset.

In [12]:
```
df = df[columns]
```

In [13]:
```
# make historgrams and save the plot
df[df.columns].hist()
```

Out[13]:
```
array([[<Axes: title={'center': 'Population'}>,
        <Axes: title={'center': 'Children'}>,
        <Axes: title={'center': 'Age'}>],
       [<Axes: title={'center': 'Income'}>,
        <Axes: title={'center': 'Outage_sec_perweek'}>,
        <Axes: title={'center': 'Email'}>],
       [<Axes: title={'center': 'Contacts'}>,
        <Axes: title={'center': 'Yearly_equip_failure'}>,
        <Axes: title={'center': 'Tenure'}>],
       [<Axes: title={'center': 'MonthlyCharge'}>,
        <Axes: title={'center': 'Bandwidth_GB_Year'}>, <Axes: >]],
      dtype=object)
```



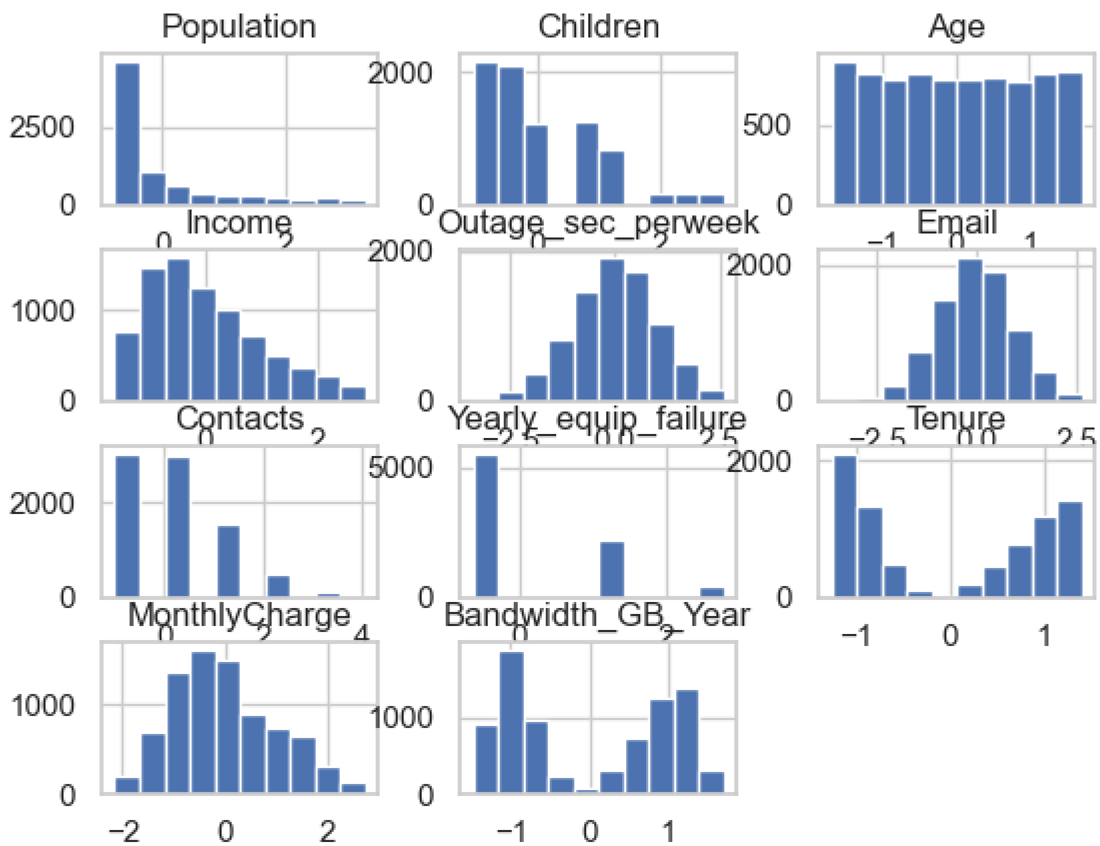
In [14]:
```
# scale the data
scaler = StandardScaler()

# apply scaler() to all the continuous column
scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

scaled.head()
```

Out[14]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts
0	-0.763524	-1.069540	0.718481	-0.350998	-0.677041	-0.656913	-1.009708
1	0.640643	-0.478753	-1.266577	-0.657046	0.606316	0.007418	-1.009708
2	-0.264753	1.293609	-0.153008	-1.196872	0.279927	-0.989078	-1.009708
3	1.101638	-0.478753	-0.249840	-0.781101	1.715042	1.003913	1.024516
4	0.762874	-1.069540	1.444722	0.162809	-0.618718	1.336079	1.024516

In [15]:
```
# make historgrams and save the plot
scaled[scaled.columns].hist()
```

Out[15]:
```
array([[<Axes: title={'center': 'Population'}>,
        <Axes: title={'center': 'Children'}>,
        <Axes: title={'center': 'Age'}>],
       [<Axes: title={'center': 'Income'}>,
        <Axes: title={'center': 'Outage_sec_perweek'}>,
        <Axes: title={'center': 'Email'}>],
       [<Axes: title={'center': 'Contacts'}>,
        <Axes: title={'center': 'Yearly_equip_failure'}>,
        <Axes: title={'center': 'Tenure'}>],
       [<Axes: title={'center': 'MonthlyCharge'}>,
        <Axes: title={'center': 'Bandwidth_GB_Year'}>, <Axes: >]],
      dtype=object)
```



In [16]:
```python
# save the prepared data set
scaled.to_csv('churn_prepared2.csv', index=False)
```

Part IV. Analysis

D1. Determine the matrix of all the principal components.

In [17]:
```python
pca = PCA(n_components=6, random_state=493)
model = pca.fit_transform(scaled)
explained_ratio = pca.explained_variance_ratio_
explained_ratio
```

Out[17]: array([0.1813204 , 0.09673   , 0.09385725, 0.0920565 , 0.09105452,
        0.09085826])

In [18]:
```
print(pca.explained_variance_ratio_.cumsum())
```

[0.1813204 0.2780504 0.37190765 0.46396415 0.55501867 0.64587693]

In [19]:
```
matrix = pd.DataFrame(model, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6'])
```

In [20]:
```
matrix.head()
matrix.tail()
```

Out[20]:

	PC1	PC2	PC3	PC4	PC5	PC6
0	-1.519209	-0.540523	-0.473303	0.861094	-0.652050	-1.097087
1	-1.653816	0.260500	1.735917	-0.171267	1.103416	-1.350004
2	-0.877558	-1.236562	0.909397	0.782003	-0.135945	-1.523216
3	-0.926511	1.736037	0.776245	-1.107354	-1.012047	0.374659
4	-1.930431	1.365848	-0.985916	0.598114	-0.916003	-0.457744

Out[20]:

	PC1	PC2	PC3	PC4	PC5	PC6
8100	1.217256	1.689024	-0.249704	-1.031767	-1.481937	0.314613
8101	0.825883	-0.672417	-1.407049	-1.078539	-0.437635	0.631326
8102	1.866298	-0.797156	0.315489	-0.836525	-0.198837	1.478551
8103	0.569396	-0.993756	-1.129926	-0.385718	0.151184	0.082555
8104	1.554721	1.639923	1.075307	-1.916243	0.736876	-0.692976

D2. Identify the total number of principal components using the elbow rule or the Kaiser criterion. Include a screenshot of the scree plot.

In [21]:
```
plt.figure(figsize=(10, 8))
plt.plot(pca.explained_variance_ratio_)
plt.title('Scree Plot with Elbow')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance')
```

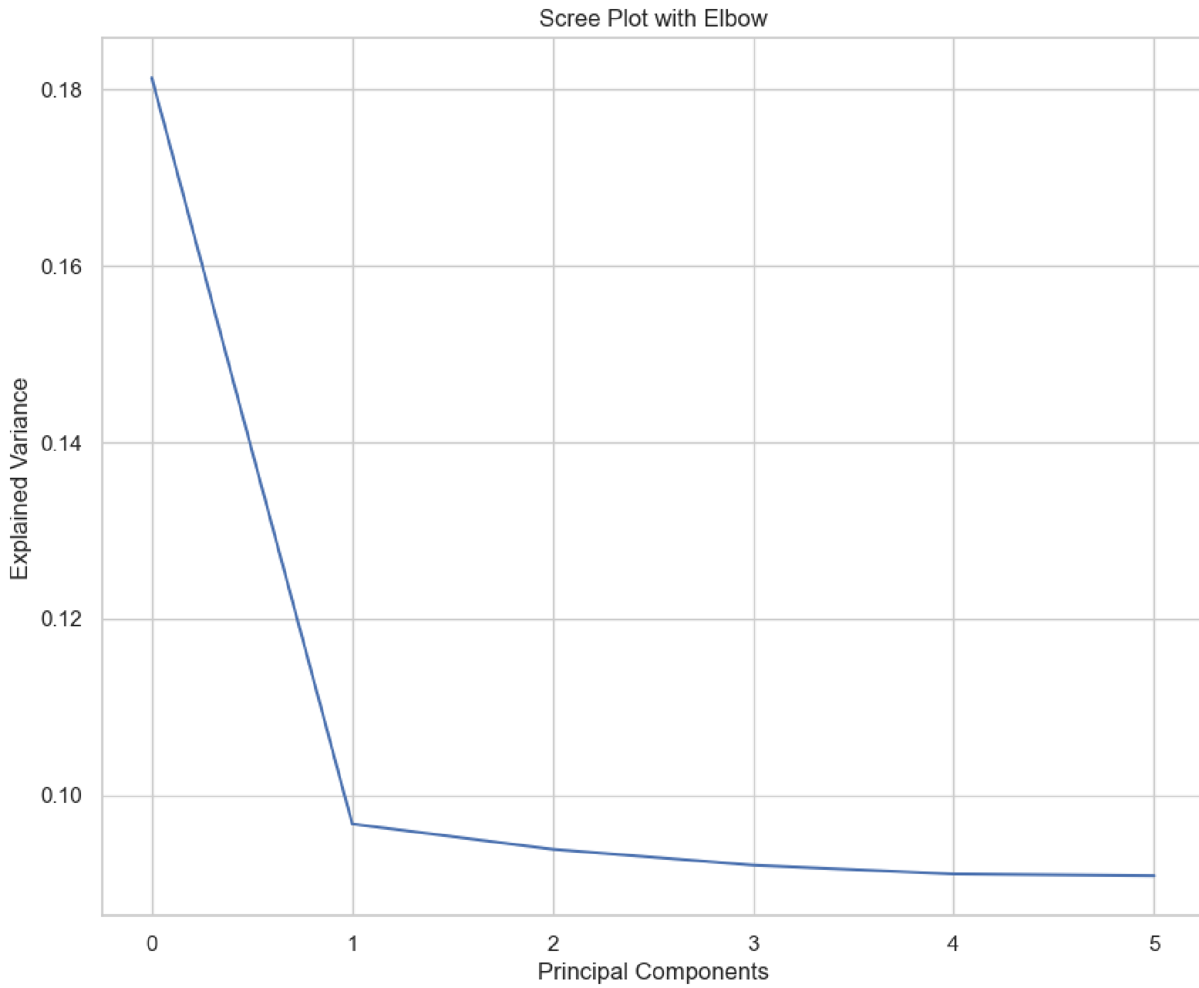
Out[21]: <Figure size 1000x800 with 0 Axes>

Out[21]: [<matplotlib.lines.Line2D at 0x1a5480d4550>]

Out[21]: Text(0.5, 1.0, 'Scree Plot with Elbow')

Out[21]: Text(0.5, 0, 'Principal Components')

Out[21]: Text(0, 0.5, 'Explained Variance')



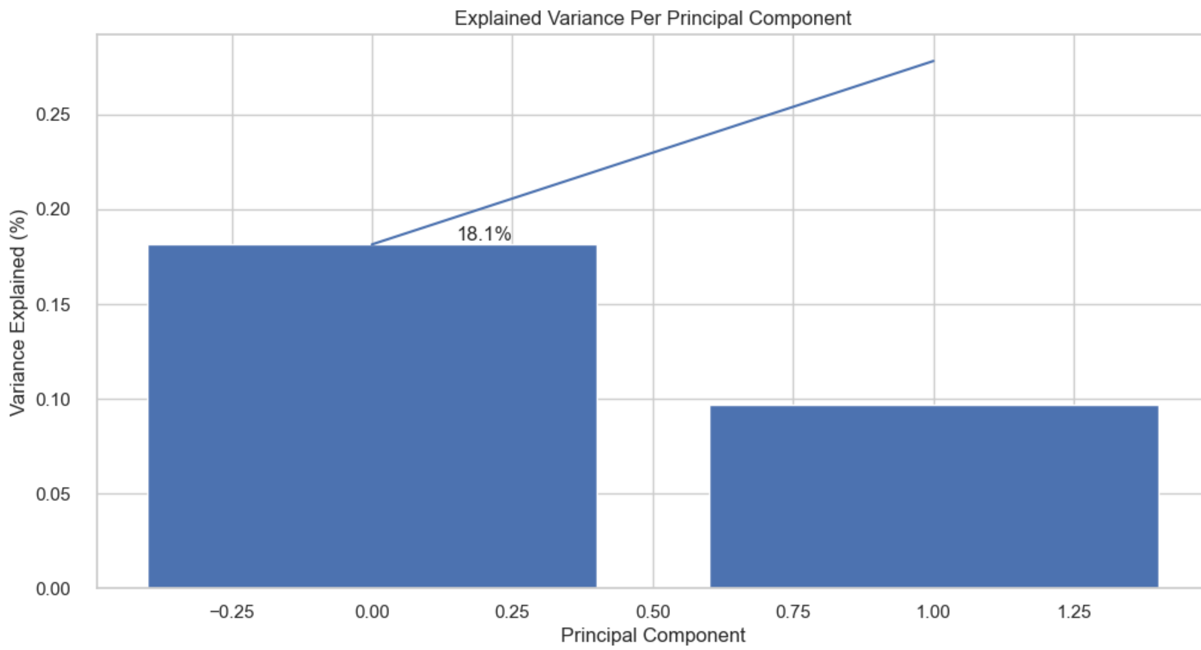
Based on the scree plot above, two principal components are significant.

D3. Identify the variance of each of the principal components identified in part D2.

In [22]:
```
def scree_plot(pca):
    num_components = len(pca.explained_variance_ratio_)
    ind = np.arange(num_components)
    vals = pca.explained_variance_ratio_

    plt.figure(figsize=(12, 6))
    ax = plt.subplot(111)
    cumvals = np.cumsum(vals)
    ax.bar(ind, vals)
    ax.plot(ind, cumvals)
    for i in range(num_components):
      if(i%10==0):
        ax.annotate(r"%s%%" % ((str(vals[i]*100)[:4])), (ind[i]+0.2, vals[i]), va="

    ax.xaxis.set_tick_params(width=0)
    ax.yaxis.set_tick_params(width=2, length=12)

    ax.set_xlabel("Principal Component")
```

```
    ax.set_ylabel("Variance Explained (%)")
    plt.title('Explained Variance Per Principal Component')
```

In [23]:
```
# Re-apply PCA to the data while selecting for number of components to retain.
pca = PCA(n_components=2)
model = pca.fit_transform(scaled)
scree_plot(pca)
```



In [24]:
```
print('Explained variation per principal component: {}'.format(pca.explained_varian
```

```
Explained variation per principal component: [0.1813204 0.09673   ]
```

D4. Identify the total variance captured by the principal components identified in part D2.

In [25]:
```
print(str(round(sum(pca.explained_variance_ratio_)*100, 2)) + "%")
```

```
27.81%
```

D5. Summarize the results of your data analysis.

While we were able to identify the principal components of the data set, I believe that the result of the analysis is inconclusive. The total variance captured by the principal components identified in part D2 is only 27.81%. This is simply not enough for any actionable insight to base on.

Part V. Attachments

E. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

- https://github.com/ecdedios/code-snippets/blob/main/notebooks/master.ipynb
- https://github.com/ecdedios/d212-data-mining-ii/blob/main/D212%20Performance%20Assessment%20Task%201%20(Rev.%200).ipynb
- https://github.com/microbhai/CustomerChurnAnalysis/blob/master/PrincipalComponentAn
- https://www.datacamp.com/tutorial/principal-component-analysis-in-python

F. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

- https://medium.com/data-science-on-customer-churn-data/pca-or-principal-component-analysis-on-customer-churn-data-d18ca60397ed