

Task 1 - Clustering Techniques

Ednalyn C. De Dios

August 24, 2023

Environment

- Python: 3.9.9
- Jupyter: 7.0.2

Part I - Research Question

A1. Propose one relevant question and choose a clustering technique.

What are the different groups within the organization's customer base?

We will endeavor to answer the above question by using the K-means method.

A2. Define one goal of the data analysis.

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 the K-means clustering technique identify the different groups within the organization's customer base and segment them accordingly. The organization will benefit from knowing the similarities and differences between customer groups because the knowledge will greatly influence how marketing is conducted. This will inform the decisions of stakeholders in matters where customer retention is involved, for example. Knowing which group a particular customer belong in will provide the organization with advanced insight towards that customer's characteristics or behavior. Thereby increasing the effective of marketing campaigns.

This goal is within the scope of the scenario and is represented in the available data.

Part II - Method Justification

B1. Explain how the clustering technique you chose analyzes the selected dataset. Include expected outcomes.

I chose K-means it's relatively easy to implement, scales to large data sets, and guarantees convergence. (Google.com, 2023). K-means works by taking data points as input and groups them into groups or k number of clusters. It starts the training process of grouping by selecting a number of starting center points called centroids. The algorithm then takes each data point and assigns them to the nearest centroid. Based on the new assignments, the centroids are then recalculated and the whole process repeats until they converge.

I expect the model to show the groupings of k number of clusters that are similar to each other (the data points are similar, not necessarily the clusters themselves.) I also expect a summary of the model's performance, such as silhouette score.

B2. Summarize one assumption of the chosen clustering technique.

One core assumption of K-means is that the "scales of different variables are specified so they can be reasonably combined using sum-of-squares as the measure to be minimised" (Henry, 2022). In other words, using K-means assumes that the radius of each cluster is the same.

B3. List the packages or libraries you have chosen for Python or R, and justify how each item on the list supports the analysis.

The packages/libraries I've chosen are:

- sys: for running the executable file (d209_task1.py)
- random: set random seed for reproducibility of the experiment
- pandas: for manipulating dataframes
- numpy: for performing mathematical computations
- matplotlib and seaborn: for visualizations
- Image: display image from a URL
- StandardScaler: to scale the data set
- kmeans: clustering algorithm
- silhouette_score: evaluating the model

Part III - Data Preparation

C1. Describe one data preprocessing goal relevant to the clustering technique from part A1.

One preprocessing goal is to standardize the variables to satisfy the spherical assumption of k-means. We will accomplish this using StandardScaler.

C2. Identify the initial data set variables that you will use to analyze the classification question from part A1, and classify each variable as continuous or categorical.

Independent Variables:

- Children, Continuous, Quantitative
- Age, Continuous, Quantitative
- Income, Continuous, Quantitative
- Tenure, Continuous, Quantitative
- MonthlyCharge, Continuous, Quantitative
- Bandwidth_GB_Year, Continuous, Quantitative

C3. Explain each of the steps used to prepare the data for the analysis. Identify the code segment for each step.

- Load and view the data

```
use pandas to read the csv use head or tail methods use info()
```

- Remove irrelevant data for the analysis

```
use the drop method
```

- Perform data cleaning

```
remove duplicates, missing values, outliers
```

- Pre-process data

```
scale the features
```

- Perform K-means clustering

```
determine optimal number of k clusters select best model get  
characteristics evaluate model
```

```
In [1]: # setting the random seed for reproducibility
import random
random.seed(493)

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

# for visualizations
%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.cluster import KMeans
from sklearn.metrics import silhouette_score

# 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 [2]: # 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[2]:

	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 [3]: columns = ['Children',  
                  'Age',  
                  'Income',  
                  'MonthlyCharge',  
                  'Bandwidth_GB_Year',  
                  'Tenure']
```

```
In [4]: for col in columns:  
        # show outliers  
        plt.boxplot(df[col])  
        fig = plt.figure(figsize =(10, 7))
```

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x2bc017ded90>,
 <matplotlib.lines.Line2D at 0x2bc01801070>],
 'caps': [<matplotlib.lines.Line2D at 0x2bc01801310>,
 <matplotlib.lines.Line2D at 0x2bc018015b0>],
 'boxes': [<matplotlib.lines.Line2D at 0x2bc017deac0>],
 'medians': [<matplotlib.lines.Line2D at 0x2bc01801850>],
 'fliers': [<matplotlib.lines.Line2D at 0x2bc01801af0>],
 'means': []}

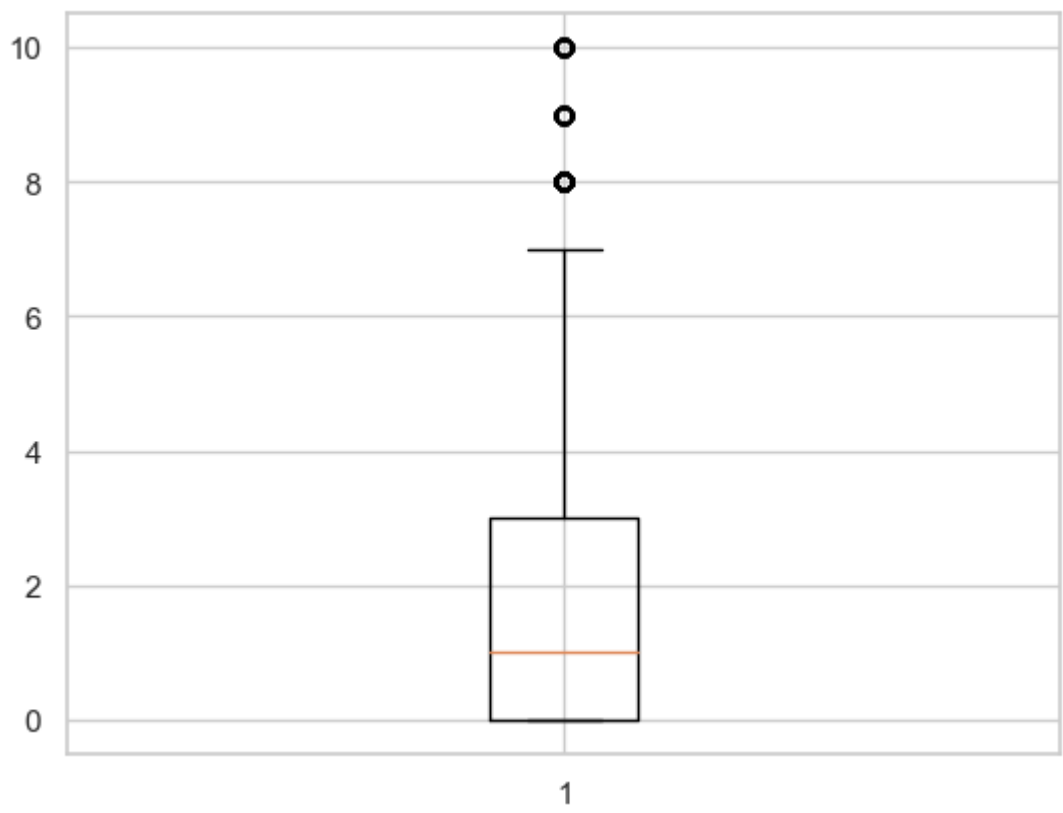
```
Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x2bc01849b20>,
<matplotlib.lines.Line2D at 0x2bc01849dc0>],
'caps': [<matplotlib.lines.Line2D at 0x2bc01849fd0>,
<matplotlib.lines.Line2D at 0x2bc018582b0>],
'boxes': [<matplotlib.lines.Line2D at 0x2bc01849880>],
'medians': [<matplotlib.lines.Line2D at 0x2bc01858550>],
'fliers': [<matplotlib.lines.Line2D at 0x2bc018587f0>],
'means': []}

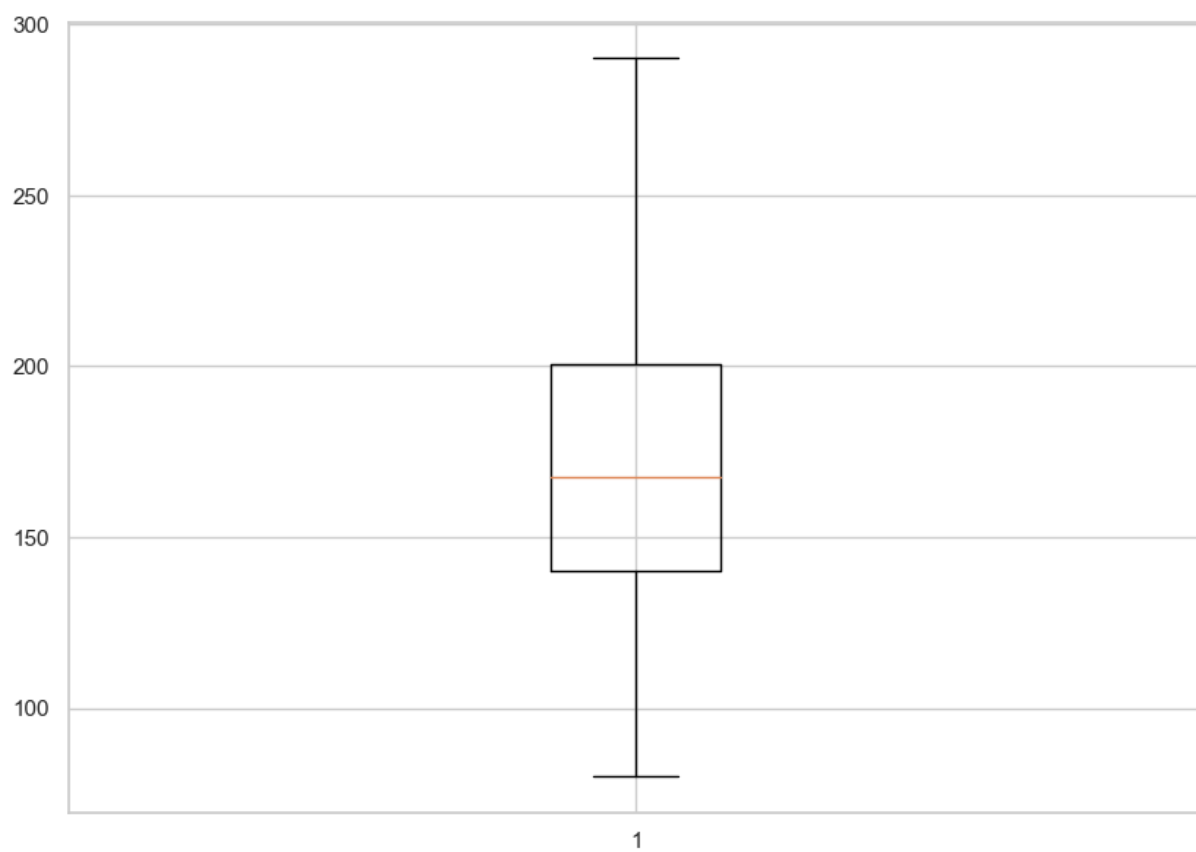
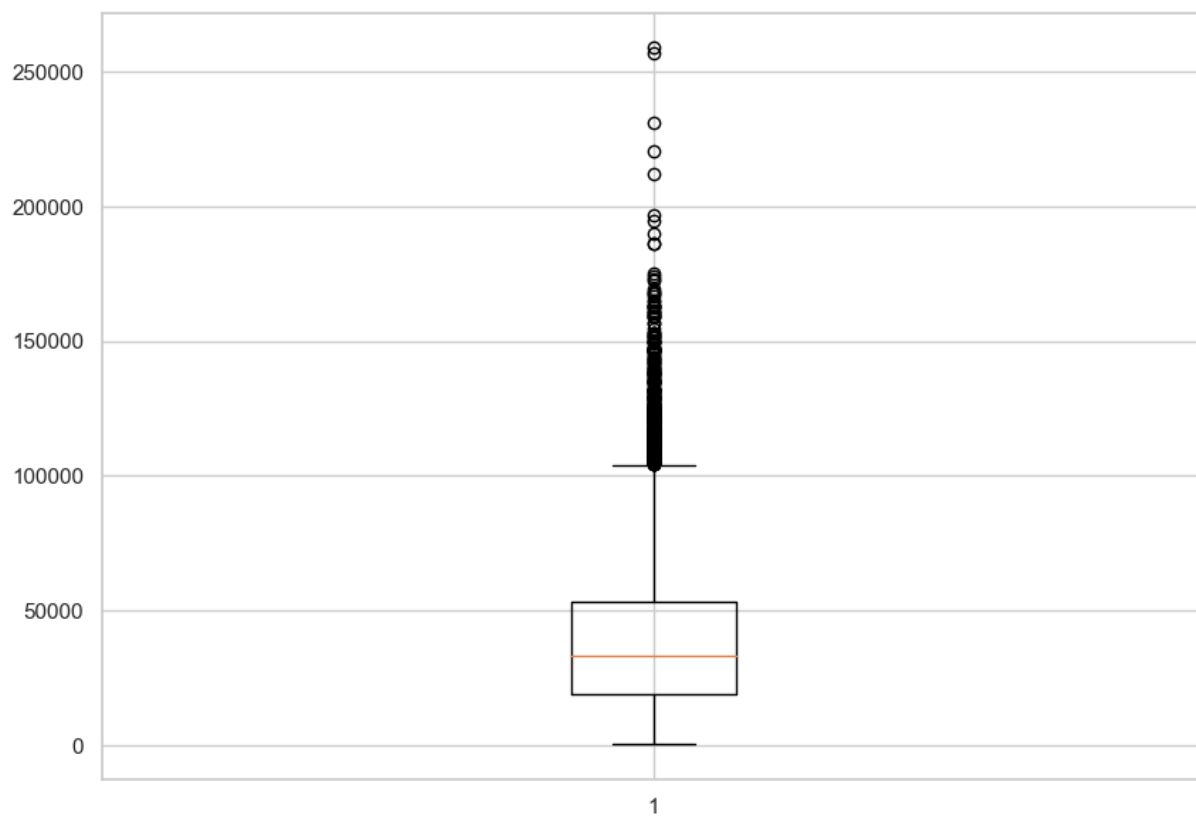
Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x2bc018a16d0>,
<matplotlib.lines.Line2D at 0x2bc01890280>],
'caps': [<matplotlib.lines.Line2D at 0x2bc018a1af0>,
<matplotlib.lines.Line2D at 0x2bc018a1d90>],
'boxes': [<matplotlib.lines.Line2D at 0x2bc018a1430>],
'medians': [<matplotlib.lines.Line2D at 0x2bc018ae070>],
'fliers': [<matplotlib.lines.Line2D at 0x2bc018ae310>],
'means': []}

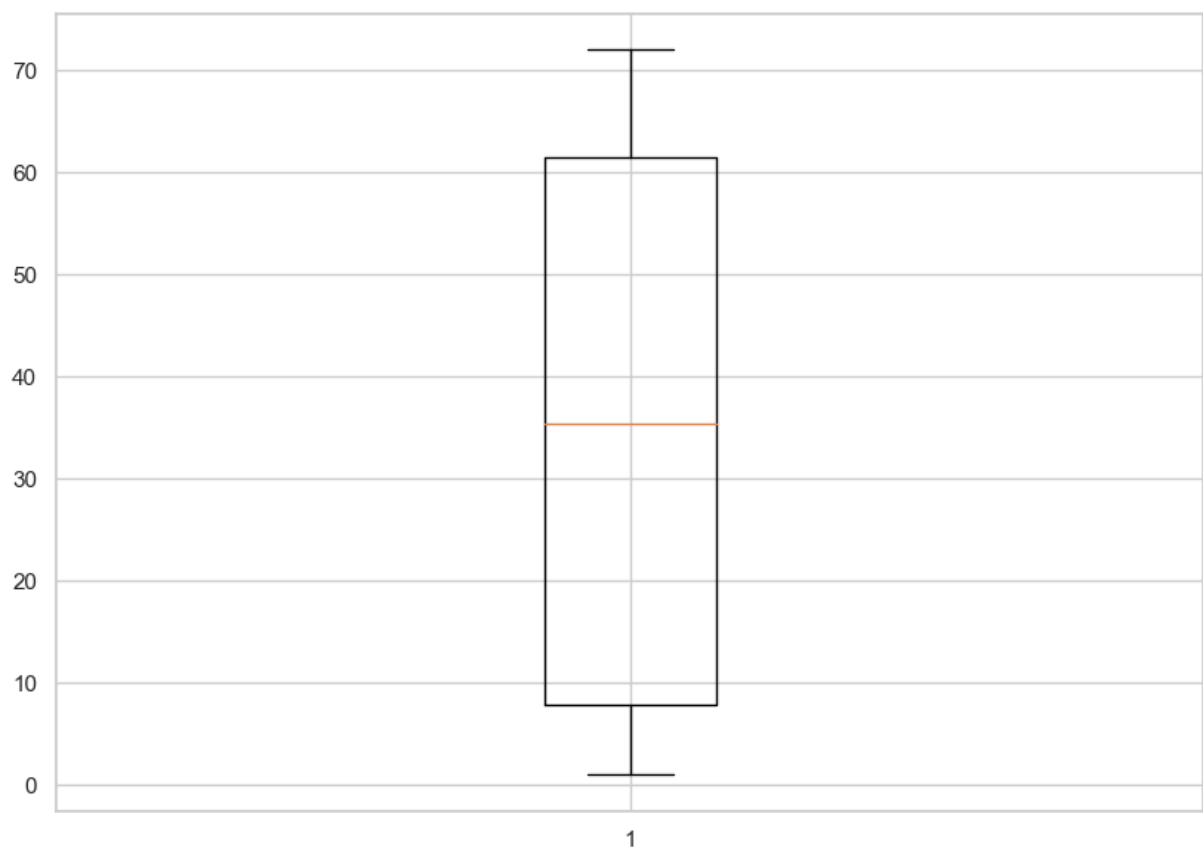
Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x2bc018e5ee0>,
<matplotlib.lines.Line2D at 0x2bc018f80a0>],
'caps': [<matplotlib.lines.Line2D at 0x2bc018f8340>,
<matplotlib.lines.Line2D at 0x2bc018f85e0>],
'boxes': [<matplotlib.lines.Line2D at 0x2bc018e5c40>],
'medians': [<matplotlib.lines.Line2D at 0x2bc018f8880>],
'fliers': [<matplotlib.lines.Line2D at 0x2bc018f8b20>],
'means': []}

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x2bc0193d9a0>,
<matplotlib.lines.Line2D at 0x2bc0193db20>],
'caps': [<matplotlib.lines.Line2D at 0x2bc0193ddc0>,
<matplotlib.lines.Line2D at 0x2bc0194e0a0>],
'boxes': [<matplotlib.lines.Line2D at 0x2bc0193d700>],
'medians': [<matplotlib.lines.Line2D at 0x2bc0194e340>],
'fliers': [<matplotlib.lines.Line2D at 0x2bc0194e5e0>],
'means': []}

Out[4]: {'whiskers': [<matplotlib.lines.Line2D at 0x2bc01992460>,
<matplotlib.lines.Line2D at 0x2bc01992700>],
'caps': [<matplotlib.lines.Line2D at 0x2bc019929a0>,
<matplotlib.lines.Line2D at 0x2bc01992c40>],
'boxes': [<matplotlib.lines.Line2D at 0x2bc019922e0>],
'medians': [<matplotlib.lines.Line2D at 0x2bc01992ee0>],
'fliers': [<matplotlib.lines.Line2D at 0x2bc019a21c0>],
'means': []}
```







<Figure size 1000x700 with 0 Axes>

```
In [5]: # remove outliers in "Children"
df = df[df['Children'] < 8]
df.shape
```

Out[5]: (9599, 50)

```
In [6]: # drop unnecessary columns not used in the analysis
df = df[columns]
```

```
In [7]: # select rows that are duplicated based on all columns
dup = df[df.duplicated()]

# find out how many rows are duplicated
dup.shape
```

Out[7]: (0, 6)

```
In [8]: def show_missing(df):
        """
        Takes a dataframe and returns a dataframe with stats
        on missing and null values with their percentages.
        """

        null_count = df.isnull().sum()
        null_percentage = (null_count / df.shape[0]) * 100
        empty_count = pd.Series(((df == ' ') | (df == '')).sum())
        empty_percentage = (empty_count / df.shape[0]) * 100
        nan_count = pd.Series(((df == 'nan') | (df == 'NaN')).sum())
        nan_percentage = (nan_count / df.shape[0]) * 100
        dfx = pd.DataFrame({'num_missing': null_count, 'missing_percentage': null_perce
                            'num_empty': empty_count, 'empty_percentage': empty_perce
                            'nan_count': nan_count, 'nan_percentage': nan_percentage})

        return dfx

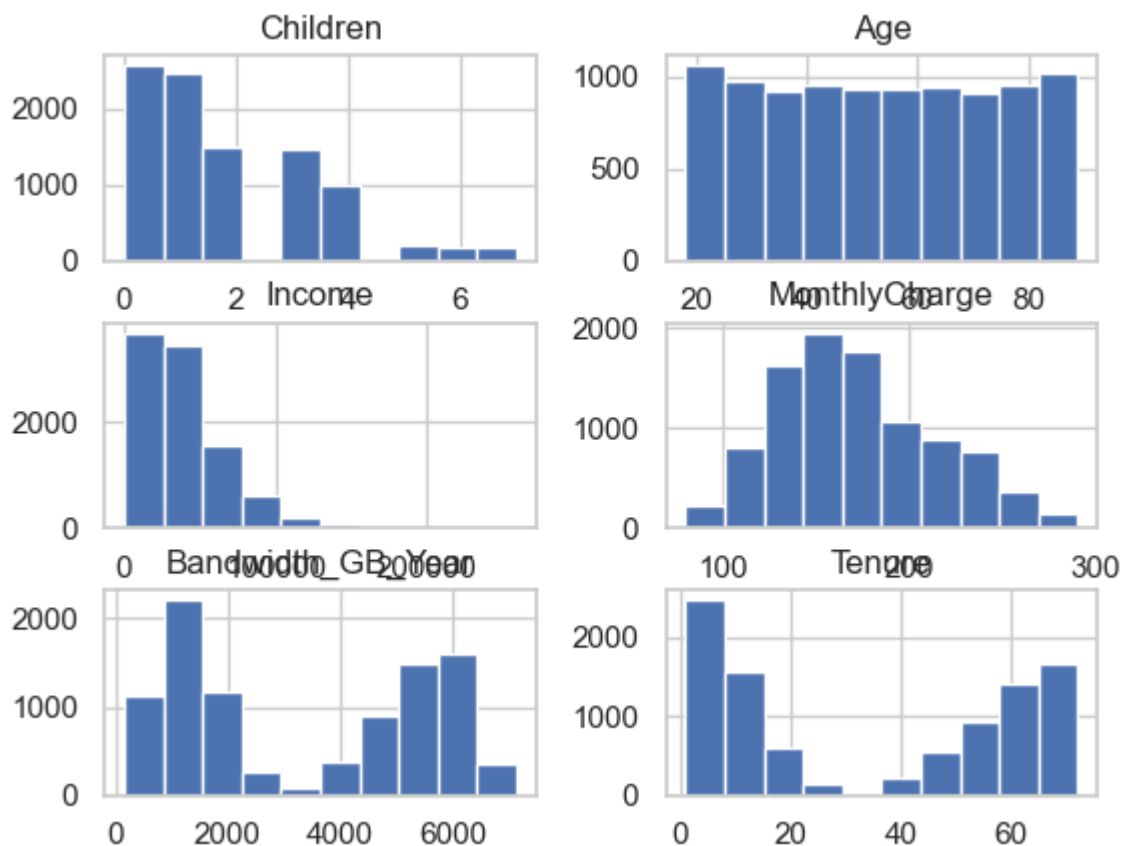
show_missing(df)
```

Out[8]:

	num_missing	missing_percentage	num_empty	empty_percentage	nan
Children	0	0.0	0	0.0	
Age	0	0.0	0	0.0	
Income	0	0.0	0	0.0	
MonthlyCharge	0	0.0	0	0.0	
Bandwidth_GB_Year	0	0.0	0	0.0	
Tenure	0	0.0	0	0.0	

```
In [9]: # make histograms and save the plot
df[columns].hist()
```

Out[9]: array([[<Axes: title={'center': 'Children'}>,
 <Axes: title={'center': 'Age'}>],
 [<Axes: title={'center': 'Income'}>,
 <Axes: title={'center': 'MonthlyCharge'}>],
 [<Axes: title={'center': 'Bandwidth_GB_Year'}>,
 <Axes: title={'center': 'Tenure'}>]], dtype=object)



```
In [10]: df.head()
```

```
Out[10]:
```

	Children	Age	Income	MonthlyCharge	Bandwidth_GB_Year	Tenure
0	0	68	28561.99	172.455519	904.536110	6.795513
1	1	27	21704.77	242.632554	800.982766	1.156681
2	4	50	9609.57	159.947583	2054.706961	15.754144
3	1	48	18925.23	119.956840	2164.579412	17.087227
4	0	83	40074.19	149.948316	271.493436	1.670972

```
In [11]: # scale the data
scaler = StandardScaler()

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

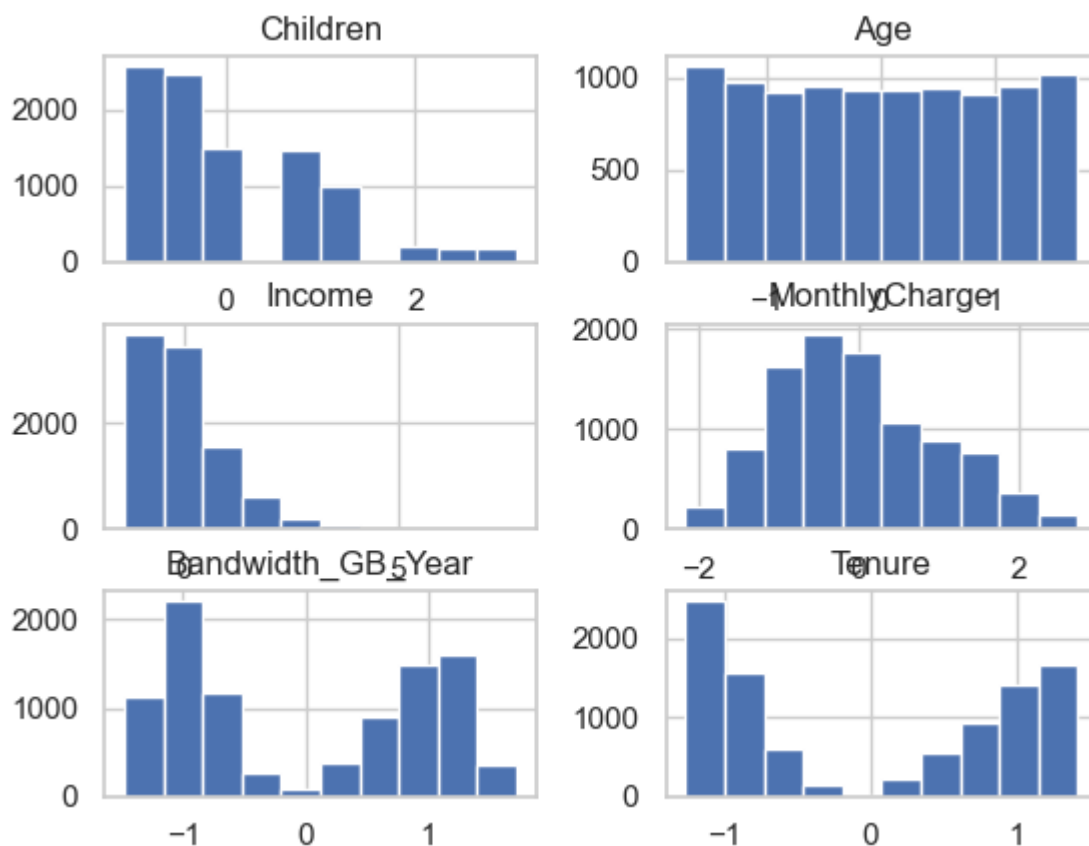
scaled.head()
```

```
Out[11]:
```

	Children	Age	Income	MonthlyCharge	Bandwidth_GB_Year	Tenure
0	-1.071056	0.713714	-0.398974	-0.006432	-1.135196	-1.049171
1	-0.479476	-1.270369	-0.642528	1.629571	-1.182594	-1.262376
2	1.295264	-0.157347	-1.072124	-0.298023	-0.608743	-0.710444
3	-0.479476	-0.254131	-0.741251	-1.230307	-0.558452	-0.660040
4	-1.071056	1.439598	0.009914	-0.531131	-1.424951	-1.242931

```
In [12]: # make histograms and save the plot
scaled[colums].hist()
```

```
Out[12]: array([[<Axes: title={'center': 'Children'}>,
<Axes: title={'center': 'Age'}>],
[<Axes: title={'center': 'Income'}>,
<Axes: title={'center': 'MonthlyCharge'}>],
[<Axes: title={'center': 'Bandwidth_GB_Year'}>,
<Axes: title={'center': 'Tenure'}>]], dtype=object)
```



C4. Provide a copy of the cleaned dataset.

```
In [13]: # save the prepared data set
scaled.to_csv('churn_prepared1.csv', index=False)
```

Part IV - Analysis

D1. Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

Multiple k-means models were used to conduct the analysis. Each model had a different number of k clusters, ranging from 1 to 10. The inertia score was plotted against the number of k clusters for each model, and the elbow method, the optimal number of k clusters was selected through visual inspection of the graph.

The selected number of k clusters was two and scikit learn's implementation of k-means clustering was used to label each data point's cluster. The cluster labels were then assigned back to the unscaled dataframe and aggregated by mean, median, and standard deviation. Finally, the characteristics of each centroids were displayed and reviewed.

D2. Provide the code used to perform the clustering technique from part D1.

```
In [14]: # determine the best value for k
inertia = np.array([])
k_vals = range(1,10)

for k in k_vals:
    kmeans = KMeans(n_clusters=k, random_state=493)
    kmeans.fit(scaled)
    inertia = np.append(inertia, kmeans.inertia_)

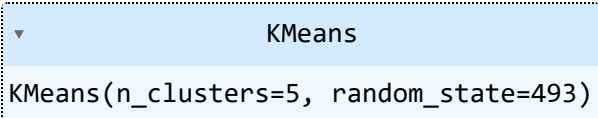
inertia_vals = pd.DataFrame(inertia, index=k_vals, columns=['Inertia'])
```

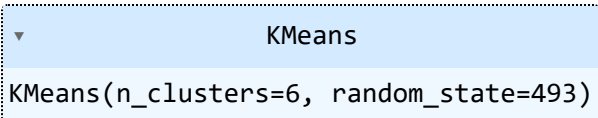
```
Out[14]: ▼ KMeans
KMeans(n_clusters=1, random_state=493)
```

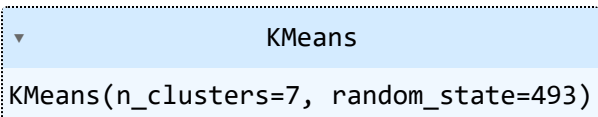
```
Out[14]: ▼ KMeans
KMeans(n_clusters=2, random_state=493)
```

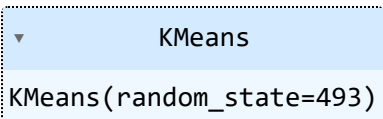
```
Out[14]: ▼ KMeans
KMeans(n_clusters=3, random_state=493)
```

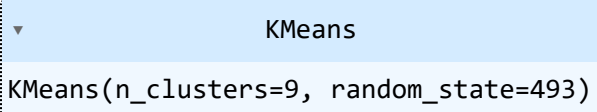
```
Out[14]: ▼ KMeans
KMeans(n_clusters=4, random_state=493)
```

Out[14]:  KMeans
KMeans(n_clusters=5, random_state=493)

Out[14]:  KMeans
KMeans(n_clusters=6, random_state=493)

Out[14]:  KMeans
KMeans(n_clusters=7, random_state=493)

Out[14]:  KMeans
KMeans(random_state=493)

Out[14]:  KMeans
KMeans(n_clusters=9, random_state=493)

```
In [15]: # plot the inertia values
sns.barplot(x=inertia_vals.index, y=inertia_vals.Inertia, alpha=0.25)
sns.lineplot(x=inertia_vals.index-1, y=inertia_vals.Inertia, marker='o', markersize
plt.title('Elbow Method')
plt.xticks(inertia_vals.index-1)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
for i in inertia_vals.index:
    plt.text(
        x=i-1,
        y=inertia_vals.Inertia[i]+1000,
        s=round(inertia_vals.Inertia[i], 2)
    )
plt.grid(alpha=0.25)
```

Out[15]: <Axes: ylabel='Inertia'>

Out[15]: <Axes: ylabel='Inertia'>

Out[15]: Text(0.5, 1.0, 'Elbow Method')

```
Out[15]: ([<matplotlib.axis.XTick at 0x2bc02aacc70>,
<matplotlib.axis.XTick at 0x2bc02aacc40>,
<matplotlib.axis.XTick at 0x2bc02aac850>,
<matplotlib.axis.XTick at 0x2bc025c1ca0>,
<matplotlib.axis.XTick at 0x2bc025d9850>,
<matplotlib.axis.XTick at 0x2bc02a63c10>,
<matplotlib.axis.XTick at 0x2bc02bd2be0>,
<matplotlib.axis.XTick at 0x2bc025d9b20>,
<matplotlib.axis.XTick at 0x2bc02bdd430>],
[Text(0, 0, '1'),
Text(1, 0, '2'),
Text(2, 0, '3'),
Text(3, 0, '4'),
Text(4, 0, '5'),
Text(5, 0, '6'),
Text(6, 0, '7'),
Text(7, 0, '8'),
Text(8, 0, '9')])
```

```
Out[15]: Text(0.5, 0, 'Number of Clusters')
```

```
Out[15]: Text(0, 0.5, 'Inertia')
```

```
Out[15]: Text(0, 58593.999999999956, '57594.0')
```

```
Out[15]: Text(1, 41019.79419385704, '40019.79')
```

```
Out[15]: Text(2, 37385.21592901777, '36385.22')
```

```
Out[15]: Text(3, 33800.13300225327, '32800.13')
```

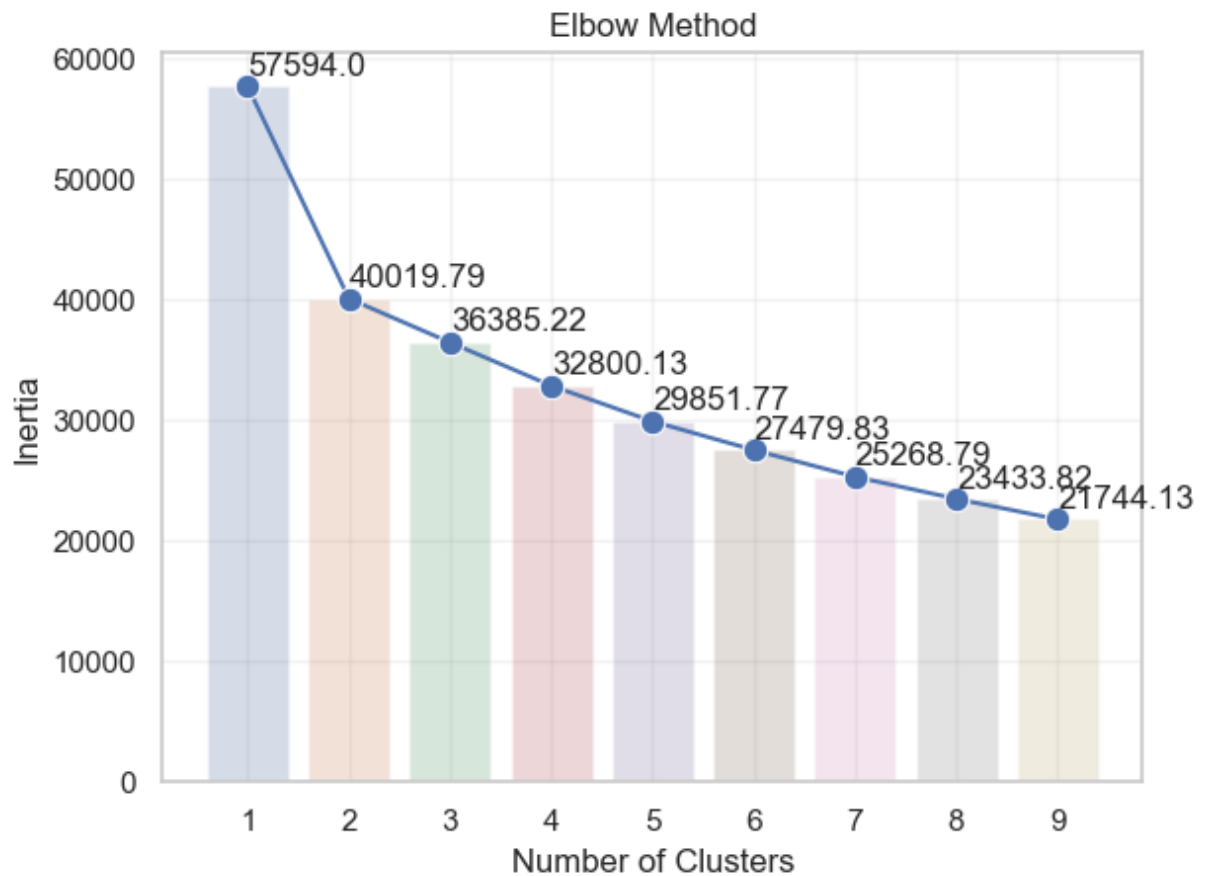
```
Out[15]: Text(4, 30851.765052565588, '29851.77')
```

```
Out[15]: Text(5, 28479.832289387083, '27479.83')
```

```
Out[15]: Text(6, 26268.785976365332, '25268.79')
```

```
Out[15]: Text(7, 24433.82109221359, '23433.82')
```

```
Out[15]: Text(8, 22744.129180902477, '21744.13')
```

```
In [16]: # do k-means clustering
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=493)
kmeans.fit(scaled)
```

```
Out[16]: KMeans
KMeans(n_clusters=2, random_state=493)
```

```
In [17]: # get predictions
predictions = kmeans.fit_predict(scaled)

# calculate the silhouette score
silhouette = silhouette_score(scaled, predictions)
print(f'Silhouette Score: {silhouette}, {n_clusters} clusters')
```

Silhouette Score: 0.29897803408788926, 2 clusters

```
In [18]: # assign cluster labels
df['Cluster'] = kmeans.labels_ + 1
```

```
In [19]: # calculate cluster summary
cluster = df.groupby('Cluster').agg(['mean', 'median', 'std']).transpose()
cluster.columns = ['Cluster 1', 'Cluster 2']
cluster
```

Out[19]:

		Cluster 1	Cluster 2
Children	mean	1.811446	1.809554
	median	1.000000	1.000000
	std	1.696421	1.684673
Age	mean	53.695109	52.806842
	median	54.000000	53.000000
	std	20.661250	20.662434
Income	mean	39974.132903	39615.565920
	median	33056.020000	33348.015000
	std	28456.884915	27853.631809
MonthlyCharge	mean	172.655381	172.807601
	median	167.456400	169.937833
	std	43.105339	42.692882
Bandwidth_GB_Year	mean	5463.462849	1301.088811
	median	5574.461345	1224.665872
	std	748.306325	568.844865
Tenure	mean	59.922603	9.106950
	median	61.417660	7.890303
	std	8.456071	6.028835

```
In [20]: # get the centroids
centroids = pd.DataFrame(
    scaler.inverse_transform(kmeans.cluster_centers_),
    index=['Cluster 1', 'Cluster 2'],
    columns=df.columns[:-1]
)
```

```
In [21]: # view the centroids
print('Cluster Centroids:\n')
for i in centroids.index:
    print(i)
    for col in centroids.columns:
        print(f'{col}: {round(centroids[col][i], 2)}')
    print()
```

Cluster Centroids:

Cluster 1

Children: 1.81

Age: 53.7

Income: 39974.13

MonthlyCharge: 172.66

Bandwidth_GB_Year: 5463.46

Tenure: 59.92

Cluster 2

Children: 1.81

Age: 52.81

Income: 39615.57

MonthlyCharge: 172.81

Bandwidth_GB_Year: 1301.09

Tenure: 9.11

```
In [22]: # plot the centroids
centr_x = [sum(df[df.Cluster==1].index)/len(df[df.Cluster==1]),
           sum(df[df.Cluster==2].index)/len(df[df.Cluster==2])]

titles = [
    'Number of Children',
    'Age',
    'Income',
    'Monthly Charge',
    'Bandwidth',
    'Tenure',
]
y_labels = [
    '',
    '',
    '',
    '',
    '',
    '',
]
fig, ax = plt.subplots(1, 6)
plt.suptitle('Cluster Characteristics and Centroids')
plt.rcParams['figure.figsize'] = (12,6)
plt.tight_layout()
for i in range(6):
    sns.scatterplot(x=df.index, y=df[df.columns[i]], hue=df.Cluster, ax=ax[i], legend=False)
    sns.scatterplot(x=centr_x, y=centroids[centroids.columns[i]], ax=ax[i])
    ax[i].set_title(titles[i])
    ax[i].set_ylabel(y_labels[i])
```

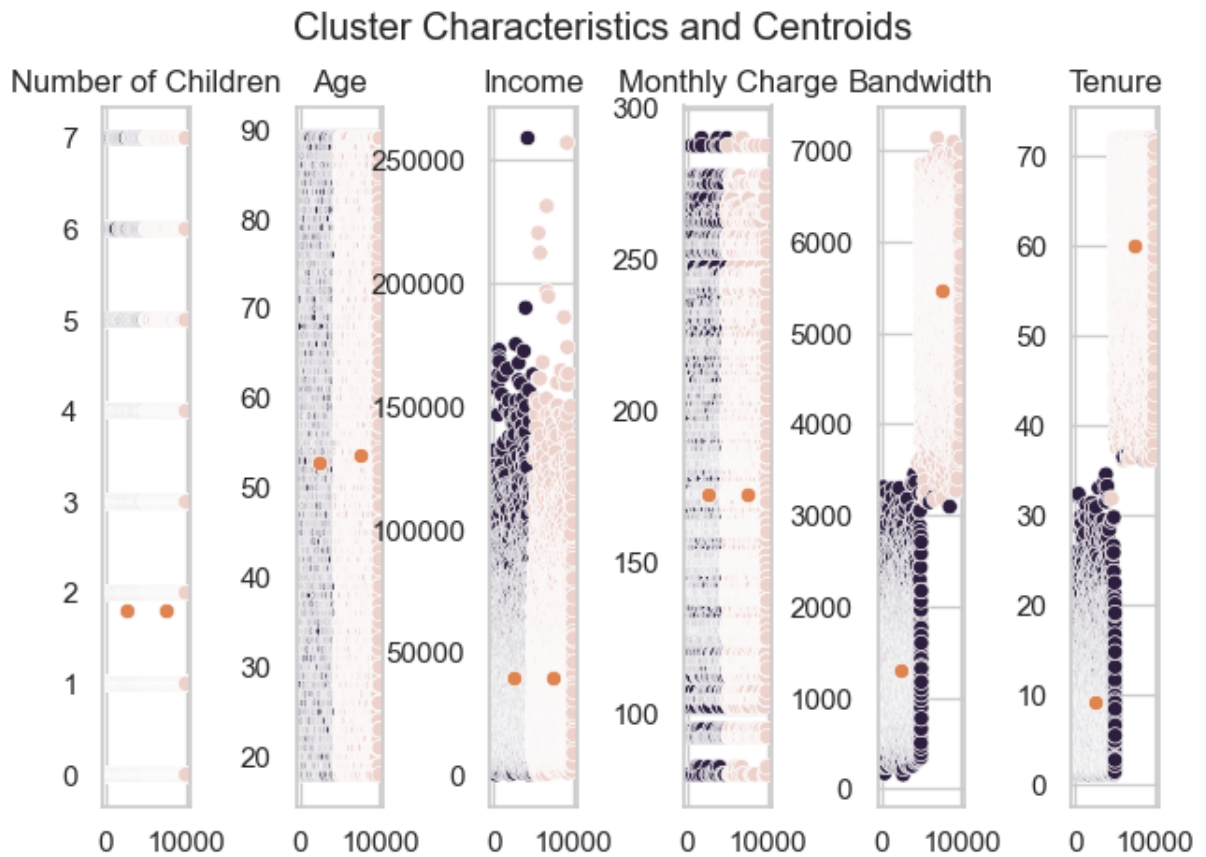
Out[22]: Text(0.5, 0.98, 'Cluster Characteristics and Centroids')

Out[22]: <Axes: ylabel='Children'>

Out[22]: <Axes: ylabel='Children'>

Out[22]: Text(0.5, 1.0, 'Number of Children')

```
Out[22]: Text(39.875, 0.5, '')
Out[22]: <Axes: ylabel='Age'>
Out[22]: <Axes: ylabel='Age'>
Out[22]: Text(0.5, 1.0, 'Age')
Out[22]: Text(124.90357142857145, 0.5, '')
Out[22]: <Axes: ylabel='Income'>
Out[22]: <Axes: ylabel='Income'>
Out[22]: Text(0.5, 1.0, 'Income')
Out[22]: Text(209.93214285714288, 0.5, '')
Out[22]: <Axes: ylabel='MonthlyCharge'>
Out[22]: <Axes: ylabel='MonthlyCharge'>
Out[22]: Text(0.5, 1.0, 'Monthly Charge')
Out[22]: Text(294.9607142857143, 0.5, '')
Out[22]: <Axes: ylabel='Bandwidth_GB_Year'>
Out[22]: <Axes: ylabel='Bandwidth_GB_Year'>
Out[22]: Text(0.5, 1.0, 'Bandwidth')
Out[22]: Text(379.98928571428576, 0.5, '')
Out[22]: <Axes: ylabel='Tenure'>
Out[22]: <Axes: ylabel='Tenure'>
Out[22]: Text(0.5, 1.0, 'Tenure')
Out[22]: Text(465.0178571428572, 0.5, '')
```



Part V - Data Summary and Implications

E1. Explain the accuracy of your clustering technique.

The final model with 2 clusters was evaluated using the silhouette score. A silhouette score is "a measure of how well each data point fits within its assigned cluster" (Castillo, 2023). The possible range of values for silhouette score is from -1 to 1. A silhouette score of 1 is the best, which indicates clearly defined clusters while -1 is the worst. A silhouette score of near 0 indicates overlapping clusters (Banerji, 2021). In this case, the silhouette score is 0.2989 which is near 0 so we can assume some overlap in our clusters.

E2. Discuss the results and implications of your clustering analysis.

The two most distinctive characteristics of the two clusters were the amount of bandwidth per year in gigabytes and tenure. Cluster 1's bandwidth is 5,463.46 GB while Cluster 2's bandwidth is only 1,301.09 GB. Respectively, Cluster 1's and Cluster 2's tenure were 59.92 and 9.11. Everything else, including the number of children, age, income, and monthly

charge, were pretty much similar in values. This leads me to believe that customers with the longest tenure tend to be the heavy users of bandwidth as well.

The implication of this is the materialization of customer segments: heavy and light users of bandwidth.

E3. Discuss one limitation of your data analysis.

One big limitation of this analysis is the exclusion of categorical variables because k-means clustering can only handle continuous variables. One categorical variable in mind is contract types. Previous analysis of the data set indicated that those customers with month-to-month contract types are more likely to churn than those with one and two-year contract types. As such, future analysis should utilize other clustering techniques that can deal with both continuous and categorical variables.

E4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

An easy course of action for the organization is for the marketing department to make use of the newly discovered customer segments. For example, a marketing campaign can be designed to target light users of bandwidth and upsell them some streaming services. Bundling services is also an option. The point is that the more bandwidth a customer uses, the more likely that they are going to stick around.

Part VI - Video Demonstration

F. Panopto recording

URL: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8fe99cc7-78d5-4f14-bcd2-b068006d2bfa>

G. 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://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering>
- https://github.com/jlopez873/Telecom_Churn_Analysis_Using_Clustering_Techniques_Kmean

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

- <https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages>
- <https://stats.stackexchange.com/questions/576812/what-are-the-k-means-algorithm-assumptions>
- <https://javilopezcastillo.medium.com/telecom-churn-analysis-using-clustering-techniques-for-customer-segmentation-4cdb7318f672>
- <https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters>

```
In [23]: print('Successful run!')
```

```
Successful run!
```