D212: Data Mining II – Task 1 Clustering Techniques

Part I: Interactive Data Dashboard

A1. Research Question

Is it possible to identify any distinct segments based on the customers' monthly charges, tenure, and the bandwidth usage per year using k-means clustering technique?

A2. Defined Goal

The goal of this analysis is to use the k-means clustering technique to identify groups of customers based on similarities between the customers' monthly charges, tenure, and the bandwidth usage per year. In doing so, this analysis will help the marketing team make more strategic decisions for the telecommunications company.

Part II: Technique Justification

B1. Clustering Technique Explanation

K-means clustering technique is an unsupervised machine learning model that is used to separate data points into subgroups called clusters. A cluster is grouped together based on their similarities. This technique identifies *k* the number of centroids, which is the mean of the cluster, and then distributes the data points to the nearest cluster. The goal is to keep the centroids as small as possible (LEDU 2018).

The expected outcome after performing this technique would mean all of the centroids have been stabilized and/or the optimal number of clusters have been met.

B2. Summary of the Technique Assumption

One assumption of the k-means clustering technique is that the clusters are assumed to be spherical-shaped and isotropic. This would mean their radius is equal in every direction, however this is not always the case since the centroid is determined by the mean of the data points within that cluster. Therefore, the clusters can be non-spherical or even elongated (*Demonstration of K-means assumptions*).

B3. Packages/Libraries List

I will be using the following libraries and packages for my analysis:

- pandas- to load datasets
- NumPy- to work with arrays
- Sci-kit Learn- for machine learning and to transform our data
- Matplotlib- for basic plotting generally consisting of bars, lines, pies, scatter plots, and graphs
- Seaborn- for a variety of visualization patterns

Part III: Data Preparation

C1. Data Preprocessing

One data preprocessing goal relevant to the clustering technique is to normalize the data. In this case, we have to normalize the data using the z-score with StandardScaler from sklearn.

C2. Dataset Variables

Variable Name	Continuous or Categorical
MonthlyCharge	Continuous
Tenure	Continuous
Bandwidth_GB_Year	Continuous

C3. Analysis Steps

First, perform the basic data cleaning steps that have been performed for every course including exploring the dataset, calculating any null or missing values, and dropping any unnecessary columns that won't be used for the analysis. Then, create a boxplot to visualize if there are any outliers. Lastly are the data preprocessing steps, which include normalizing the data by using the z-score with StandardScaler from sklearn.

The code segment and each step can be found in the screenshots below:

In [1]:
```
# Import the necessary packages & libraries
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import seaborn as sns
import matplotlib.pyplot as plt

from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import silhouette_samples

# Ignore Warning Code
import warnings
warnings.filterwarnings('ignore')

# Load the data set into the pandas data frame by using read_csv command
df = pd.read_csv(r'C:\Users\ashle\Downloads\D212\churn_clean.csv', keep_default_na=False)
```

In [2]:
```
# Explore the dataset in order to determine how to evaluate the input data by using the head() command
df.head()
```

Out[2]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	MonthlyCharge	Bandwidth_GB_Year	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	...	172.455519	904.536110	5	5	5	3	4	4	3	4
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	...	242.632554	800.982766	3	4	3	3	4	3	4	4
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	...	159.947583	2054.706961	4	4	2	4	4	3	3	3
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	...	119.956840	2164.579412	4	4	4	2	5	4	3	3
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	...	149.948316	271.493436	4	4	4	3	4	4	4	5

5 rows × 50 columns

In [3]:
```
df.info()
```

```
<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       10000 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
```

In [4]:

```
# Calculate total null values and total duplicate values in the dataset
total_nulls = df.isna().sum().sum()
total_dupes = df.duplicated().sum()

print(f"Total Nulls: {total_nulls}\nTotal Duplicate Records: {total_dupes}")
```

```
Total Nulls: 0
Total Duplicate Records: 0
```

In [5]:

```
# Drop columns that are unnecessary for the analysis
to_drop = ['CaseOrder','Customer_id','Interaction','UID','City','State','County','Zip',
           'Lat','Lng','Population','Area','TimeZone','Job','Children','Age','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','Item1','Item2',
           'Item3', 'Item4','Item5','Item6','Item7','Item8']
df.drop(columns=to_drop,inplace=True)

# Print column names to see what columns are left
print(df.columns)
```

```
Index(['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'], dtype='object')
```

In [6]:

```
# VISUALIZE THE DATA FOR FURTHER EXPLANATION
# Create boxplots of columns to check for outliers
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 1))

plt.subplot(1, 3, 1)
sns.boxplot(x='Bandwidth_GB_Year', data = df)

plt.subplot(1, 3, 2)
sns.boxplot(x='MonthlyCharge', data = df)

plt.subplot(1, 3, 3)
sns.boxplot(x='Tenure', data = df)
```

Out[6]:

```
<Axes: xlabel='Tenure'>
```

In [7]:

```
# DATA PREPROCESSING
cluster_data = df[['Bandwidth_GB_Year','MonthlyCharge','Tenure']].describe().round(2)
cluster_data
```

Out[7]:

	Bandwidth_GB_Year	MonthlyCharge	Tenure
count	10000.00	10000.00	10000.00
mean	3392.34	172.62	34.53
std	2185.29	42.94	26.44
min	155.51	79.98	1.00
25%	1236.47	139.98	7.92
50%	3279.54	167.48	35.43
75%	5586.14	200.73	61.48
max	7158.98	290.16	72.00

In [8]:

```
# Normalize data using z-score with StandardScaler from sklearn
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df[['Bandwidth_GB_Year', 'MonthlyCharge', 'Tenure']])
scaled_df = pd.DataFrame(scaled_df, columns = ['Bandwidth_GB_Year', 'MonthlyCharge', 'Tenure'])
scaled_df.head()
```

Out[8]:

	Bandwidth_GB_Year	MonthlyCharge	Tenure
0	-1.138487	-0.003943	-1.048746
1	-1.185876	1.630326	-1.262001
2	-0.612138	-0.295225	-0.709940
3	-0.561857	-1.226521	-0.659524
4	-1.428184	-0.528086	-1.242551

In [9]:

```
# Save to new file
df.to_csv('D212_Task1.csv')
```

C4. Cleaned Dataset

The cleaned dataset is attached as "D212_Task1.csv".

Part IV: Analysis

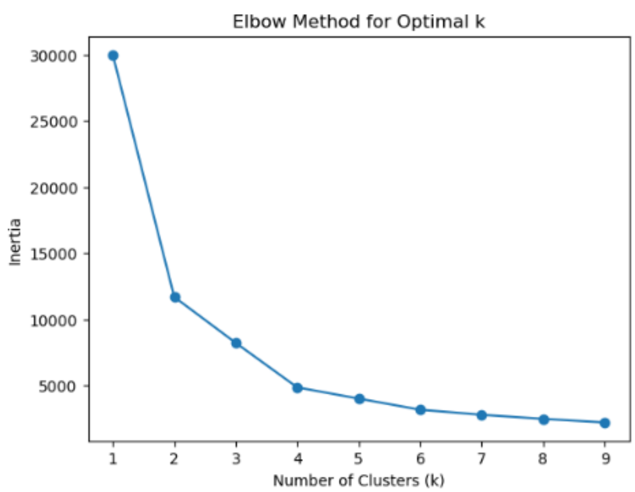
D1. Output & Intermediate Calculations

An initial k-means clustering model is built with *k* values ranging from 1 to a chosen maximum of clusters. Then, inertia is added to the model. Inertia measures how well the dataset has been clustered depending on the distance between the data points and their centroids. To determine the *k* optimal number of clusters in the data set, the elbow method is used in which you plot an "elbow" graph and locate the point where the decrease in inertia starts to flatten. That point will be the *k* of the analysis. The code used can found in part D2.

D2. Code Execution

```
In [10]: inertia = []
         for k in range(1,10):
             k_model = KMeans(n_clusters=k, n_init=10)
             k_model.fit(scaled_df)
             inertia.append(k_model.inertia_)

         # Plot the elbow graph
         plt.plot(range(1,10), inertia, '-o')
         plt.xlabel('Number of Clusters (k)')
         plt.ylabel('Inertia')
         plt.title('Elbow Method for Optimal k')
         plt.show()
```




```
In [11]: # Create the k-means clustering model using k=4
         model = KMeans(n_clusters=4)
         model.fit(scaled_df)
         model.labels_

Out[11]: array([3, 0, 3, ..., 1, 2, 2])
```

```
In [12]: model.cluster_centers_

Out[12]: array([[-0.88789675,  0.98886275, -0.95681772],
                [ 0.90465205, -0.67406416,  0.95587643],
                [ 1.02365257,  1.01729384,  0.96710276],
                [-0.99689915, -0.68107432, -0.96341863]])
```

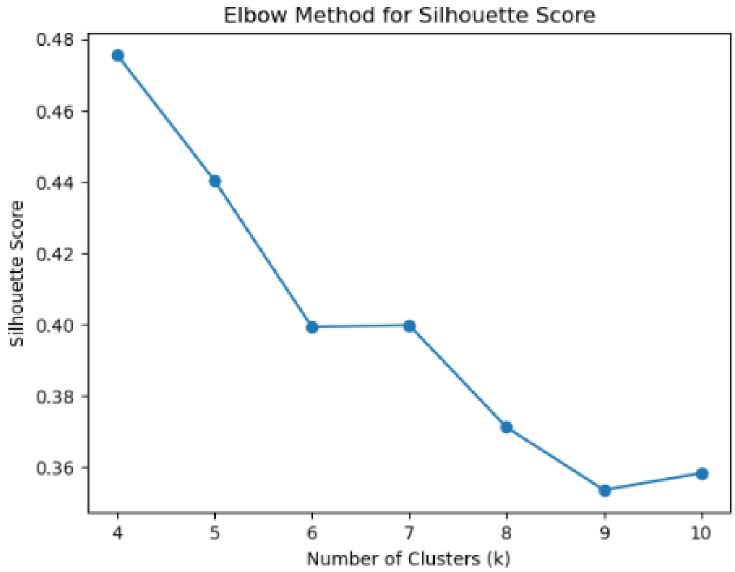
```
In [13]: # Calculate the silhouette score
         cluster_range = range(4,11)
         sil_score = []

         for k in cluster_range:
             k_model = KMeans(n_clusters=k, n_init=10)
             k_model.fit(scaled_df)
             sil_average = silhouette_score(scaled_df, k_model.labels_)
             sil_score.append(sil_average)
             print(f"For n_clusters = {k}, the silhouette score is {sil_average}")

         # Plot the silhouette score
         plt.plot(cluster_range, sil_score, '-o')
         plt.xlabel('Number of Clusters (k)')
         plt.ylabel('Silhouette Score')
         plt.title('Elbow Method for Silhouette Score')
```

```
plt.show()
```

```
For n_clusters = 4, the silhouette score is 0.47558303153855375
For n_clusters = 5, the silhouette score is 0.4403318684071761
For n_clusters = 6, the silhouette score is 0.3993304346882913
For n_clusters = 7, the silhouette score is 0.39976548079270097
For n_clusters = 8, the silhouette score is 0.37114622341619213
For n_clusters = 9, the silhouette score is 0.35355120752505265
For n_clusters = 10, the silhouette score is 0.3582842972392715
```



In [14]:

```
# Calculate the WCSS (Within Cluster Sum of Squares); another name for Inertia
wcss = model.inertia_

# Calculate the silhouette score
labels = model.labels_
silhouette_average = silhouette_score(scaled_df, labels)

# Calculate the Davies-Bouldin Index
db_index = davies_bouldin_score(scaled_df, labels)

# Print the calculations
print("WCSS/Inertia:", wcss)
print("Silhouette Score:", silhouette_average)
print("Davies-Bouldin Index:", db_index)
```

```
WCSS/Inertia: 4882.841678857103
Silhouette Score: 0.47562871470453655
Davies-Bouldin Index: 0.7579076724393521
```

Part V: Data Summary & Implications

E1. Quality of Clustering Technique

To evaluate the quality of the clusters created, I calculated the silhouette score, WCSS/inertia, and Davies-Bouldin index.

- Silhouette score: This metric measures the quality by determining how similar the data points are to each other in every cluster. The average score is between -1 and 1 in which the closer the score is to 1, the better the cluster. According to my analysis, the silhouette score is 0.48. A score between 0.25 and 0.5 is considered reasonable clustering and a score above 0.5 is considered good clustering. Therefore, the analysis has reasonable clustering but is very close to being good.
- WCSS/Inertia: This metric calculates the sum of squared distances between each data point and the centroid of their cluster. Although this metric does not necessarily measure the quality of clusters, it evaluates how compact the clusters are. The lower the value, the closer the data points are to each other. The value of 4882.84 shows the data points are reasonably compact.

- Davies-Bouldin index: This index calculates the average similarity ratio between each cluster and a cluster that is alike. A lower index is preferred because it shows the clusters are well-separated. This would mean the score of 0.76 represents decent separation across clusters (Surajsutar, 2023).

E2. <u>Results & Implications</u>

As mentioned in part A1, my research question is, "Is it possible to identify any distinct segments based on the customers' monthly charges, tenure, and the bandwidth usage per year using k-means clustering technique?" It is definitely possible to identify segments based on the customers' monthly charges, tenure, and the bandwidth usage per year because we got 4 clusters. The quality of the clusters was considered to be a reasonable cluster; however, I believe there would need to be further analysis done to be able to pass this information along to he marketing team in the telecommunications company because it is not considered a "good" cluster. Although changing the optimal number of clusters would solve this problem, it would to be contrary to the calculations and results of the elbow method.

E3. <u>Limitation</u>

One limitation of the data analysis is that the k value has to be chosen correctly. If not, the results will be much different because of all the different values. For example, if I were to manually choose k to be 3, based on the trend of the provided graph above of the silhouette scores, we can assume it would be a silhouette score of around 0.5 which is considered good clustering. However, we used the elbow method to determine the k value to be 4 which brings the score down slightly. This could show the results to be inaccurate.

E4. <u>Course of Action</u>

I recommend adding in other variables into the analysis such as age and income. A disadvantage to this clustering method is how categorical variables cannot be included, especially when the data provided consists of mainly categorical variables. I wanted to try to focus on the three continuous variables that have been the main focus of past assignments when trying to retain customers. But I do believe including additional variables may help to detect more distinct patterns since there will be a larger sample size.

Part VI: Demonstration

F. <u>Panopto Video of Code/Programs</u>

G. <u>Sources for Third-Party Code</u>

N/A

H. <u>Sources</u>

(LEDU), E. E. (2018, September 12). Understanding K-means clustering in machine learning. Medium. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

GeeksforGeeks. (2023, December 9). Demonstration of K-means assumptions. GeeksforGeeks. https://www.geeksforgeeks.org/demonstration-of-k-means-assumptions/

Surajsutar. (2023, May 31). Evaluation metrics for clustering algorithms. Medium. https://medium.com/@surajsutar37/evaluation-metrics-for-clustering-algorithms-c9baee50e328

I. **Professional Communication**