K-means Clustering of Telcom Customer Churn Data

Mike Mattinson

Western Governors University

D212: Data Mining II

Task 1: Clustering Analysis

Dr. Kesselly Kamara

May 5, 2022

Revision 6

Abstract

Telecom customer data is broken down into groups with similar attributes using K-means clustering analysis. Data source: Wgu.edu Telecom Churn data (N: 10,000). The focus is on lost customers (n: 2,650) defined where the 'Churn' variable is 'Yes'.

Keywords: Telecom. Churn. Data Mining. K-means Clustering.

List of Tables

List of Figures

K-means Clustering of Telcom Customer Churn Data

Scenario 1. Conduct data analysis for a telecommunications company that wants to better understand the characteristics of its customers.

Part I: Research Question

A. Describe the purpose of this data mining report by doing the following:

A1. Propose one question relevant to a real-world organizational situation that you will answer using k-means

What is a simple way to group lost customers of a telecom company based on key numerical features such as ‘MonthlyCharge’ or ‘Tenure’? ‘MonthlyCharge’ is defined as how much the customer is paying for services for the month. ‘Tenure’ is the number of years the customers has been loyal to the company. A simple clustering analysis might consist of a 2D scatter plot broken down into 3 or 4 sets of similar data.

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.

Use K-means clustering analysis on unlabeled customer data to group lost customers. The primary dataset consists of 10,000 customer records. The analysis will focus on lost customers, defined when ‘Churn’ = ‘Yes’.

Part II: Technique Justification

B. Explain the reasons for your chosen clustering technique from part A1 by doing the following:

B1. Explain how the clustering technique analyzes the select dataset. Include expected outcomes.

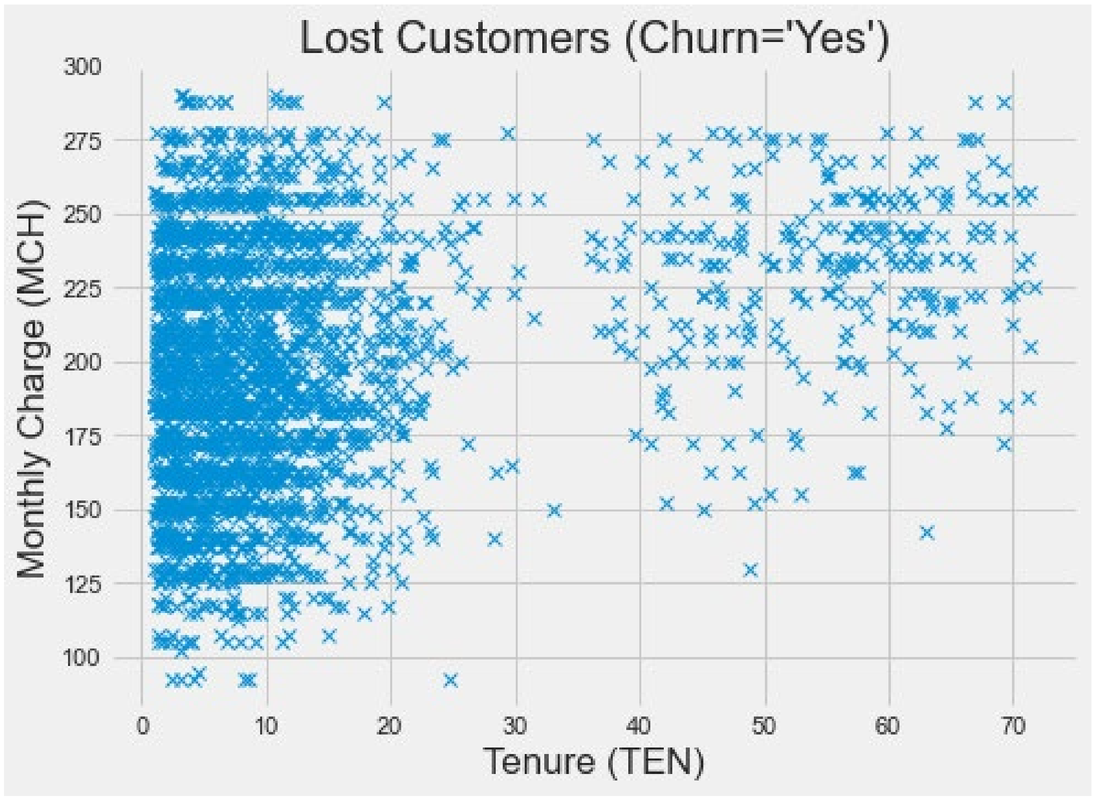
From the text “Practical Statistics for Data Scientist” (Bruce, Bruce, & Gedeck, 2020, p. 200), K-means clustering is a technique to divide data into different groups, where the records in each group are similar to one another. A goal of clustering is to identify significant and meaningful groups of data. The groups can be used directly, analyzed in more depth, or passed as a feature or an outcome to a predictive regression or classification model. K-means was the first clustering method to be developed; it is still widely used, owing its popularity to the relative simplicity of the algorithm and its ability to scale to large data sets.

K-means divides the data into K clusters by minimizing the sum of the squared distances of each record to the mean of its assigned cluster. This is referred to as the within-cluster sum of squares or within-cluster SS. K-means does not ensure the clusters will have the same size by finds the clusters that are the best separated.

Figure 1 shows the raw distribution of churned customers by monthly charge and tenure, the final analysis should look similar to this but should break the data into groups of similar attribute.

Figure 1

Tenure vs Monthly Charge Scatter Plot of Lost Customers (data: cleaned)



Notes.

Here is the code used to create Figure 1:

```
# create scatter plot of lost customer data
fig, ax = plt.subplots(figsize =(7, 5))
plt.plot(df["TEN"], df["MCH"], marker="x",
linestyle="")
plt.xlabel("Tenure")
plt.ylabel("Monthly Charge")
plt.title("Lost Customers (Churn='Yes')")
fig.savefig("figures/fig_1", dpi=150)
```

B2. Summarize one assumption of the clustering technique.

Data scientist blogger Sayak Paul (Paul, 2022) "K-Means clustering method considers two assumptions regarding the clusters – first that the clusters are spherical and second that the clusters are of similar size. Spherical assumption helps in separating the clusters when the algorithm works on the data and forms clusters. If this assumption is violated, the clusters formed may not be what one expects. On the other hand, assumption over the size of clusters helps in deciding the boundaries of the cluster. This assumption helps in calculating the number of data points each cluster should have. This assumption also gives an advantage. Clusters in K-means are defined by taking the mean of all the data points in the cluster. With this assumption, one can start with the centers of clusters anywhere. Keeping the starting points of the clusters anywhere will still make the algorithm converge with the same final clusters as keeping the centers as far apart as possible."

B3. List the packages or libraries chosen and justify how each supports the analysis.

Data preparation will be completed using Python/Jupyter interface on a Windows 10 computer. Python's pandas package (pandas.pydata.org, 2022) provide a nice way to load and manipulate data.

Data visualization will be completed using matplotlib.pyplot package.

K-means clustering analysis will be completed using the Python's sklearn.cluster, KMeans package.

Data standardization will be completed using sklearn.preprocessing, StandarScaler package.

Part III: Data Preparation

C. Perform data preparation for the chosen dataset by doing the following:

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

Clustering analysis requires all variables to be standardized, otherwise, variables with extreme values will tend to dominate the analysis. Prior to completing the clustering analysis, the numerical data to be used must be standardized. Therefore, one data preprocessing goal is to determine the correct number of variables to use, then apply some scaler process to standardize the data.

C2. Identify the initial dataset variables that you will use to perform the analysis for the clustering question from part A1, and label each as continuous or categorical.

Using data preparation and exploratory data analysis, the following list of variables were determined to be relevant to the analysis. Using a helper function, these numerical variables are described showing whether it is continuous or categorical data:

```
# describe variables as continuous or categorical
describe_dataframe_type(df_numerical)

1. INC is numerical (CONTINUOUS) - type: float64.
  Min: 348.670  Max: 189938.400  Std: 28623.988

2. OUT is numerical (CONTINUOUS) - type: float64.
  Min: 0.232  Max: 21.207  Std: 2.970

3. TEN is numerical (CONTINUOUS) - type: float64.
  Min: 1.000  Max: 71.646  Std: 15.577

4. MCH is numerical (CONTINUOUS) - type: float64.
  Min: 92.455  Max: 290.160  Std: 41.268

5. BAN is numerical (CONTINUOUS) - type: float64.
  Min: 248.179  Max: 7096.495  Std: 1375.370
```

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

The following steps were followed in order to prepare data for the clustering analysis:

Step 1. Df_raw ←import raw customer data

```
# import raw customer data
df_raw = pd.read_csv('data/churn_clean.csv')

Out[]: (10000,50)
```

Step 2. Df_cleaned←remove unwanted data

```
# remove unwanted data
df_cleaned = df_raw.drop(columns=[
    'CaseOrder','UID', 'County',
    'Interaction', 'City',
    'Job', 'Zip','Population',
    'Lat', 'Lng','Item1','Item2',
    'Item3','Item4','Item5','Item6',
    'Item7','Item8'
])
df_cleaned.shape

Out[]: (10000,32)
```

Step 3. Df_churn←filter for lost customers

```
# filter for lost customers
df_churn = df_cleaned.loc[(df_cleaned.Churn=="Yes")]
df_churn.shape

Out[]: (2650,32)
```

Step 4. Df_numerical←filter numerical float variables

```
# filter numerical float variables
df_numerical = df_churn.select_dtypes(include="float")
df_numerical.info()
df_numerical.shape

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2650 entries, 1 to 9979
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Income               2650 non-null   float64
 1   Outage_sec_perweek   2650 non-null   float64
 2   Tenure               2650 non-null   float64
 3   MonthlyCharge        2650 non-null   float64
 4   Bandwidth_GB_Year    2650 non-null   float64
dtypes: float64(5)
memory usage: 124.2 KB

Out[]: (2650, 5)
```

Step 5. Rename columns to facilitate output

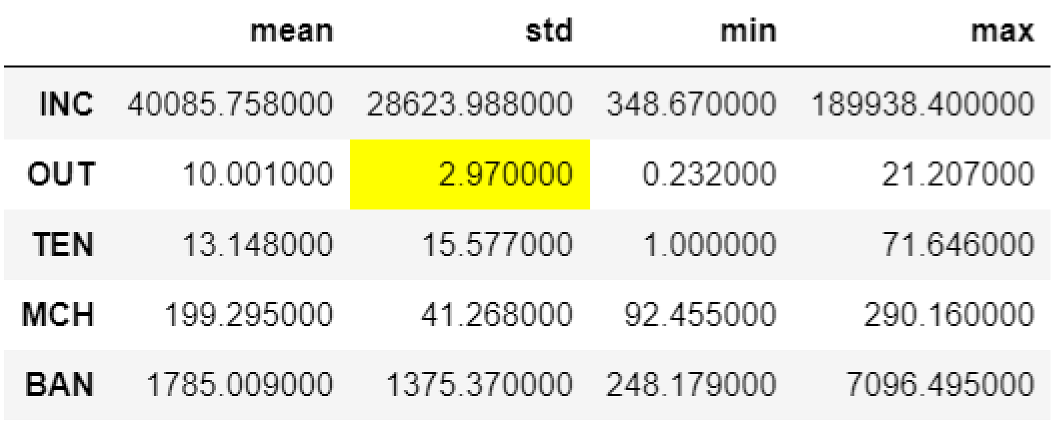
```
# rename columns to facilitate output
df_numerical.rename(columns = {
    'Income':'INC',
    'Outage_sec_perweek':'OUT',
    'Tenure':'TEN',
    'MonthlyCharge':'MCH',
    'Bandwidth_GB_Year':'BAN'
}, inplace = True)
df_numerical.info()
df_numerical.shape

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2650 entries, 1 to 9979
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   INC     2650 non-null   float64
 1   OUT     2650 non-null   float64
 2   TEN     2650 non-null   float64
 3   MCH     2650 non-null   float64
 4   BAN     2650 non-null   float64
dtypes: float64(5)
memory usage: 124.2 KB

Out[]: (2650, 5)
```

Table 1

Step 6. Describe numerical data (data: df_numerical)



	mean	std	min	max
INC	40085.758000	28623.988000	348.670000	189938.400000
OUT	10.001000	2.970000	0.232000	21.207000
TEN	13.148000	15.577000	1.000000	71.646000
MCH	199.295000	41.268000	92.455000	290.160000
BAN	1785.009000	1375.370000	248.179000	7096.495000

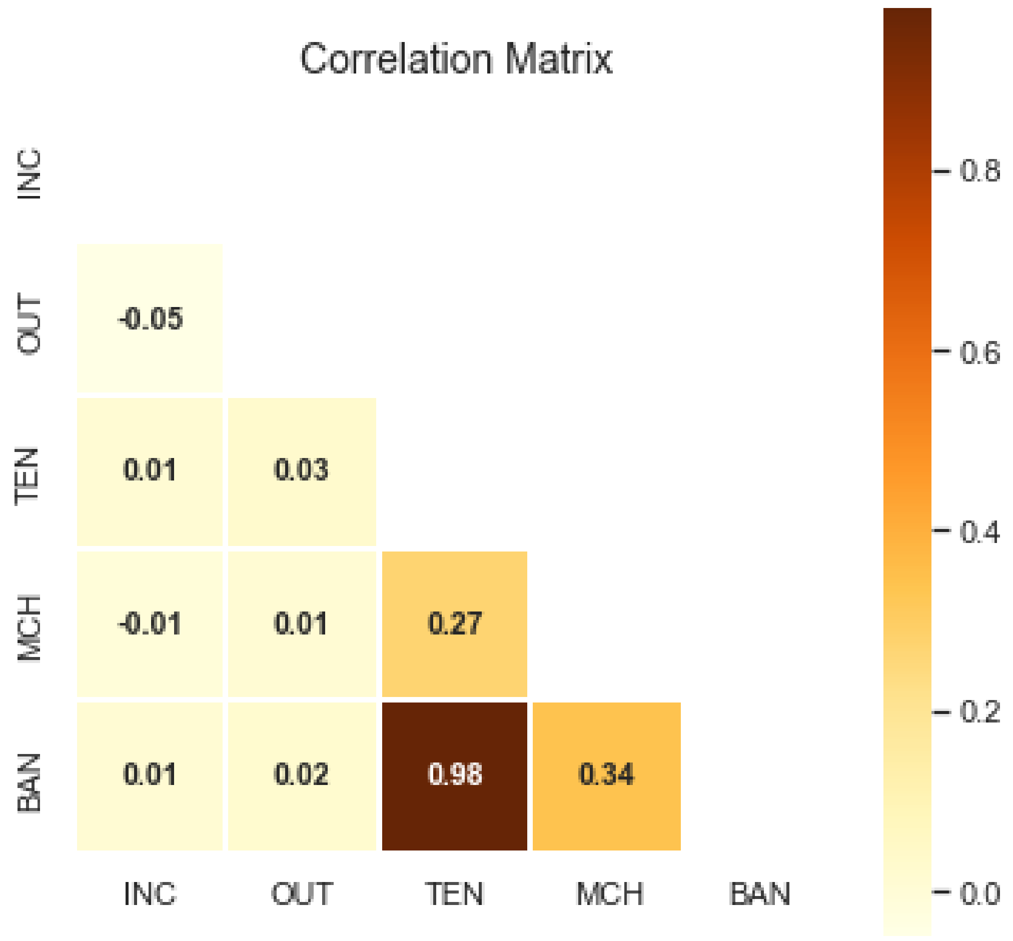
Notes.

Here is code used to create Table 1:

```
# describe numerical data - highlight small std
df = df_numerical.describe().round(3).T
def highlight_cells(val, color_if_true):
    color = color_if_true if val <= 3  else ''
    return 'background-color: {}'.format(color)
df[['mean','std','min','max']].style.applymap(highli
ght_cells,
    color_if_true='yellow', subset=['std'])
```

Figure 2

Step 7. Find highly correlated features using correlation matrix (data: df_numerical)



Notes. BAN and TEN are highly correlated with a value of 0.98, so that one of them should be removed from the clustering analysis.

Here is the code to generate the correlation matrix in Figure 2:

```
# use heatmap graph to identify highly correlated variables
def Generate_heatmap_graph(corr, chart_title,
mask_uppertri=False ):
    """ Based on features , generate correlation matrix """
    mask = np.zeros_like(corr)
    mask[np.triu_indices_from(mask)] = mask_uppertri
    fig,ax = plt.subplots(figsize=(6,6))
    sns.heatmap(corr
                , mask = mask
                , square = True
                , annot = True
                , annot_kws={'size': 10.5, 'weight' : 'bold'}
                , cmap=plt.get_cmap("YlOrBr")
                , linewidths=.1)
    plt.title(chart_title, fontsize=14)
    plt.show()

Generate_heatmap_graph(
    round(df_numerical.corr(),2),
    chart_title = 'Correlation Matrix',
    mask_uppertri = True)
```

Step 8. Df_final←remove highly correlated variables. From Step 7, it was determined that either BAN or TEN should be removed, so the following code segment will remove BAN.

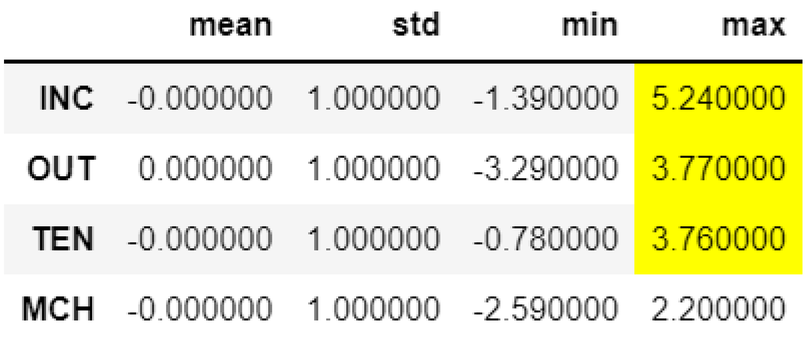
```
# remove highly correlated variables
df_final = df_numerical.drop(columns=['BAN'])
df_final.info()
df_final.shape

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2650 entries, 1 to 9979
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   INC     2650 non-null   float64
 1   OUT     2650 non-null   float64
 2   TEN     2650 non-null   float64
 3   MCH     2650 non-null   float64
dtypes: float64(4)
memory usage: 103.5 KB

Out[]: (2650, 3)
```

Table 2

Step 9. Describe standardized data (data: df_standardized)



	mean	std	min	max
INC	-0.000000	1.000000	-1.390000	5.240000
OUT	0.000000	1.000000	-3.290000	3.770000
TEN	-0.000000	1.000000	-0.780000	3.760000
MCH	-0.000000	1.000000	-2.590000	2.200000

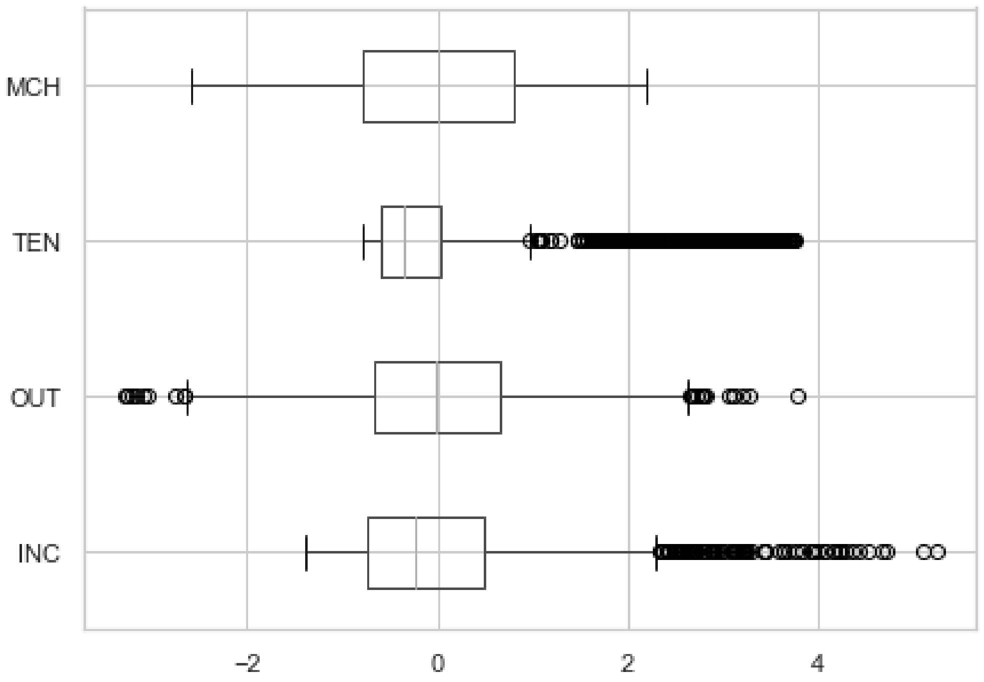
Notes. Max highlighted where max >= 3.

Here is code used to create Table 2:

```
# describe standardized data
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_final.values)
df_standardized = pd.DataFrame(scaled_features,
        index=df_final.index,
      columns=df_final.columns)
df = df_standardized.describe().round(2).T
def highlight_cells(val, color_if_true):
    color = color_if_true if val >= 3  else ''
    return 'background-color: {}'.format(color)
df[['mean','std','min','max']].style.applymap(highlight
_cells,
    color_if_true='yellow', subset=['max'])
```

Figure 3

Step 10. Looking for outliers using Boxplot (data: df_standardized)



Notes. TEN and INC both have outliers in the 3+ range. For the meantime, I am going to leave the data alone.

Here is the code to create the boxplot:

```
# use boxplot to look for outliers
fig, ax = plt.subplots(figsize =(7, 5))
ax = df_standardized.boxplot(vert=False)
```

Table 3

C4. Provide copy of cleaned dataset (data: cleaned.csv) (first 10 rows)

INC	OUT	TEN	MCH	BAN
21704.77	11.69908	1.156681	242.6326	800.9828
40074.19	8.147417	1.670972	149.9483	271.4934
11467.5	11.18272	13.23677	200.1185	1907.243
26759.64	7.791632	4.264255	114.9509	979.6127
64256.81	11.79073	10.0602	159.9656	1582.295
89061.45	10.79847	13.87001	177.6508	1840.014
31659.3	13.52285	15.78215	194.9663	2070.377
44142.81	9.831167	2.303331	202.6829	882.0986
19494.75	10.92246	12.80616	149.9447	1954.081
28520.32	13.74778	5.77039	162.5119	870.764

Source: cleaned.csv

Notes.

Part IV: Analysis

D. Perform the data analysis and report on the results by doing the following:

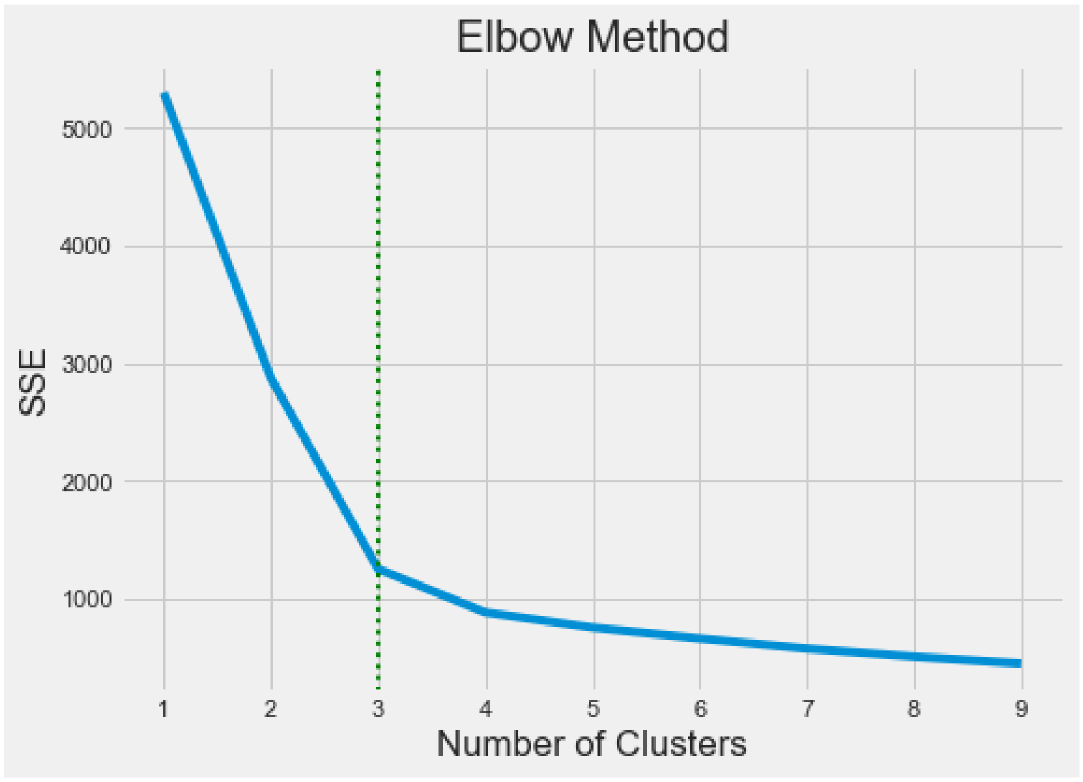
D1. Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

The following steps were used to appropriately analyze the data using k-means clustering:

Step 1. Determine the recommended number of clusters. This is a loop that runs multiple k-means clustering for different values of k, then looking at the lowest SSE for each k, is able to plot and mark the recommended k value that yields the lowest SSE value. SSE is defined as "the sum of the squared Euclidean distance of each point to its closest centroid." (Arvai, 2022) There are two methods that help to determine appropriate number of clusters, the elbow method (Figure F) and the silhouette method (Figure 6).

Figure 4

Elbow Method



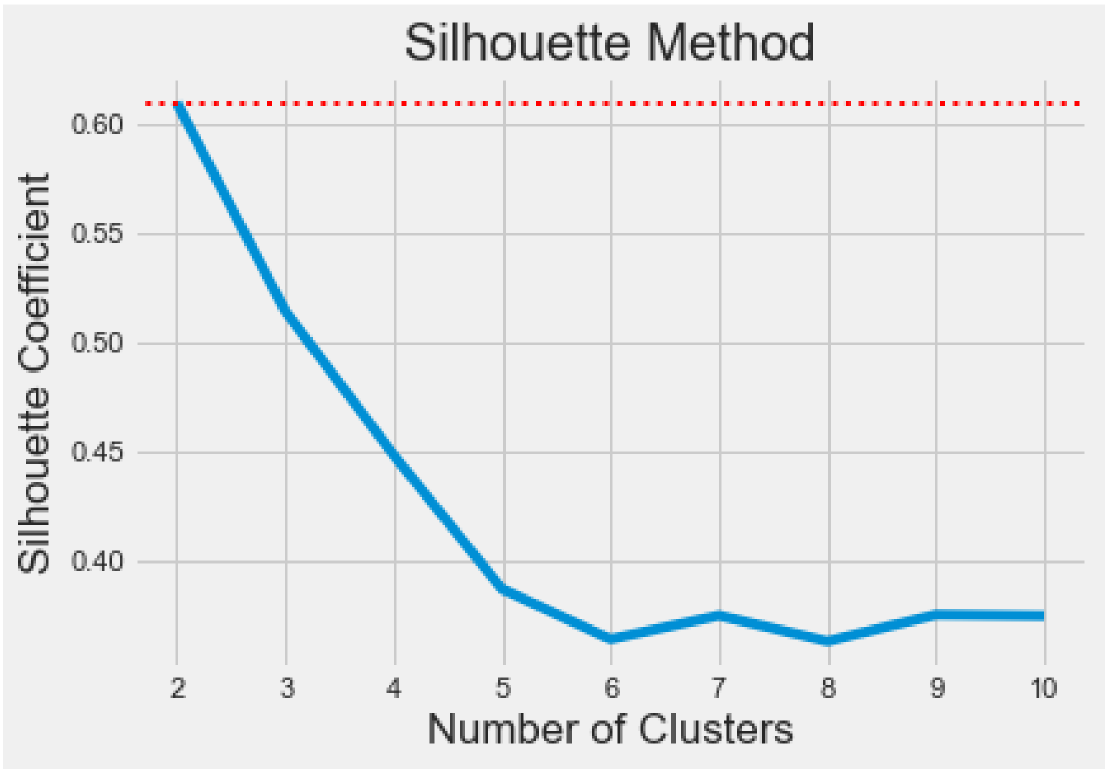
Notes. The elbow method shows how the SSE values decrease as the number of clusters increases. The elbow point is indicated by the green dotted line, and is the point where the SSE values decrease less abruptly, it looks like k=3 is the optimum value for this data.

Here is the adapted code used for the Elbow Method (Arvai, 2022):

```
# find number of clusters using elbow, adapted code (Arvai, 2022)M
= df_standardized[['TEN','MCH']]
sse = [] # list of SSE values for each k
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=10)
    kmeans.fit(M)
    sse.append(kmeans.inertia_)
fig, ax = plt.subplots(figsize =(7, 5))
knee = KneeLocator(range(1, 10), sse, curve="convex",
direction="decreasing")
plt.plot(range(1,10), sse)
plt.xticks(range(1,10))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Elbow Method")
plt.axvline(x=knee.elbow, color='green', ls=':', lw=2,)
fig.savefig("figures/fig_2a", dpi=150)
# optimum point on knee plot
print('Optimum: ({}, {:.3f})'.format(knee.elbow, sse[knee.elbow-
1]))
```

Figure 5

Silhouette Method



Notes. The optimum value of the silhouette coefficient is where it is maximized. The maximum point on the graph is marked by the red dotted line. It looks like the maximum silhouette coefficient is a bit over 0.60, which corresponds to the number of clusters of k=2. This method attempts to separate the clusters as much as possible, indicated by a value close to 1.00. For k=3 or more, the silhouette coefficient says that the clusters will become less separated and may even touch.

Here is the adapted code used for the silhouette method plot (Arvai, 2022):

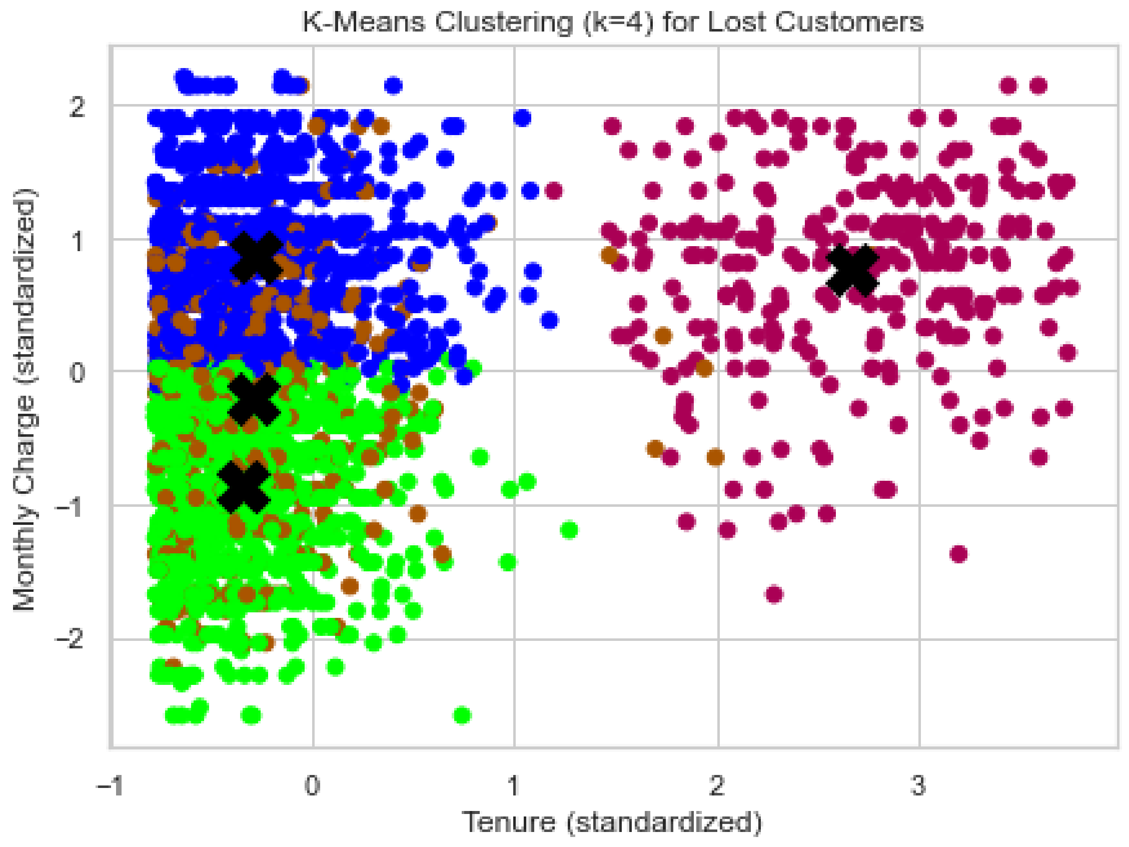
```
# find number of clusters using silhouette method, adapted code
(Arvai, 2022)
silhouette_coefficients = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=10)
    kmeans.fit(M)
    score = silhouette_score(M, kmeans.labels_)
    silhouette_coefficients.append(score)
plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients)
plt.xticks(range(2, 11))
plt.title("Silhouette Method")
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
ymax = np.amax(silhouette_coefficients)
print('Max element : ', ymax)
result = np.where(silhouette_coefficients ==
np.amax(silhouette_coefficients))
print('Returned tuple of arrays :', result)
print('List of Indices of maximum element :', result[0])
plt.axhline(y=ymax, color='red', ls=':', lw=2,)
fig.savefig("figures/fig_2b", dpi=150)
```

Step 2. Run k-means analysis using the k-value determined in Step 1. Using the knee plot and calculated SSE values, the optimum point on the graph is (4, 5326.264).

Step 3. Visualize analysis. Put it all together and visualize the final analysis.

Figure 6

Final Clustered Groups of Lost Customers (k=4) (data: df_standardized)



Notes.It really looks like it should be two (2) groups. The three (3) groups on left probably have a very low silhouette coefficient.

D2. Provide the code used to perform the clustering analysis technique from part 2.

Here is the adapted code used to create the Knee Plot (Arvai, 2022):

```
# create knee plot, adapted code (Arvai, 2022)
kmeans_kwargs = {
    "init": "random",
    "n_init": 10,
    "max_iter": 300,
    "random_state": 42 }
sse = [] # list of SSE values for each k
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(scaled_features)
    sse.append(kmeans.inertia_)
fig, ax = plt.subplots(figsize =(7, 5))
knee = KneeLocator(range(1, 11), sse, curve="convex",
direction="decreasing")
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Knee Plot")
plt.axvline(x=kl.elbow, color='green', ls=':', lw=2,)
fig.savefig("figures/fig_2", dpi=150)
```

Here is the adapted code to show the optimum point on knee plot (Arvai, 2022):

```
# optimum point on knee plot
'Optimum: ({}, {:.3f})'.format(knee.elbow, sse[knee.elbow-1])

Out[]: 'Optimum: (4, 5326.264)'
```

Here is the adapted code used to generate the final cluster plot (Arvai, 2022):

```
# final K-means analysis plot
fig, ax = plt.subplots(figsize =(7, 5))
title = 'K-Means Clustering (k=' + str(n_clusters) + ') for
Lost Customers'
ax.scatter(x=df_standardized['TEN'],y=df_standardized['MCH'],
    c=kmeans.labels_,cmap='brg')
ax.scatter(x=kmeans.cluster_centers_[:,2],
    y=kmeans.cluster_centers_[:,3],
    color='black', marker='X',s=400 )
ax.set_xlabel('Tenure (standardized)')
ax.set_ylabel('Monthly Charge (standardized)')
plt.title(title)
fig.savefig("figures/fig_3", dpi=150)
```

Part V: Data Summary and Implications

E. Summarize your data analysis by doing the following:

E1. Explain the accuracy of your clustering technique.

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

E3. Discuss one limitation of your data analysis.

Finding the right k

Impact of seeds when data is distributed…

Biased towards equal sized clusters

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.

Part VI: Demonstration

F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

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. (see References below)

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

I. Demonstrate professional communication in the content and presentation of your submission.

References

Arvai, K. (2022). K-Means Clustering in Python: A Practical Guide. Retrieved from https://realpython.com/k-means-clustering-python/#: :text=The SSE is defined as,try to minimize this value.&text=The purpose of this figure,centroids is an important step.

Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical Statistics for Data Scientists.* O'Reilly Media Inc.

Griffiths, D. (2009). *Head First Statistics.* O'Reilly Media Inc.

Kaloyanova, E. (2021, July 29). How to Combine PCA and K-means Clustering in Python? Retrieved from https://365datascience.com/tutorials/python-tutorials/pca-k-means/

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

McKinney, W. (2021, August). pandas: powerful Python data analysis toolkit, Release 1.3.1. *pandas: powerful Python data analysis toolkit, Release 1.3.1*. Retrieved from https://pandas.pydata.org/docs/pandas.pdf

Nagar, A. (2020, January 26). K-means Clustering — Everything you need to know. Retrieved from https://medium.com/analytics-vidhya/k-means-clustering-everything-you-need-to-know-175dd01766d5#f6a0

numpy.org. (2021). Getting Started. *Getting Started*. Retrieved from numpy.org

pandas.pydata.org. (2022). User Guide. *User Guide*. Retrieved from https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html

Paul, S. (2018, July 5). *K-Means Clustering in Python with scikit-learn*. Retrieved from K-Means Clustering in Python with scikit-learn: https://www.datacamp.com/community/tutorials/k-means-clustering-python

Paul, S. (2022, April 29). *K-Means Clustering in Python with scikit-learn*. Retrieved from datacamp.com: https://www.datacamp.com/community/tutorials/k-means-clustering-python

Publishers, S. A. (2022). Cluster Analysis: 10 Worst Pitfalls and Mistakes. Retrieved from http://www.statisticalassociates.com/clusteranalysis10.htm

Robinson, D. (2022). K-means clustering is not a free lunch. Retrieved from http://varianceexplained.org/r/kmeans-free-lunch/

scikit-learn.org. (2022). *K-means*. Retrieved from K-means: https://scikit-learn.org/stable/modules/clustering.html#k-means

Support, I. B. (2022). https://www.ibm.com/support/pages/clustering-binary-data-k-means-should-be-avoided. Retrieved from https://www.ibm.com/support/pages/clustering-binary-data-k-means-should-be-avoided

Tait, A. (2017, January 31). Assumptions Can Ruin Your K-Means Clusters. Retrieved from https://blog.learningtree.com/assumptions-ruin-k-means-clusters/