D212: Data Mining II

Task 1: K-Means Clustering Analysis of WGU Telecom Customer Churn Habits

Andrew Mecchi

Masters of Data Analytics, Western Governors University

The following report covers Task 1 of the Performance Assessment for D212 Data Mining II. This document is categorized by the questions defined in the rubric.

A: Research Question

A1. Propose research question using k-means clustering data mining method — Page 2

A2. Define one goal of k-means clustering analysis — Page 2

B: Justification of K-means Clustering Method

B1. Explain how k-means clustering analyzes data with expected outcome — Page 2

B2. Summarize one assumption of k-means clustering — Page 3

B3. Table of python libraries and packages used — Page 3

C. Data Preparation

C1. Define one pre-processing goal of analysis — Page 4

C2. Identification of variables used in analysis — Page 4

C3. Step-by-step code and explanation of k-means clustering — Page 4

C4. Cleaned Dataset — SEE ATTACHED — Page 9

D. Analysis

D1. Description of analysis technique applied to k-means clustering — Page 9

D2. Copy of code used for analysis — Page 12

E. Data Summary and Implications

E1. Accuracy of analysis using silhouette score — Page 13

E2. Discuss results and implications of analysis — Page 13

E3. Discuss one limitation of analysis — Page 15

E4. Recommended course of action based on results — Page 16

F. Panopto Presentation

F1. Panopto Video — SEE ATTACHED — Page 16

G: Code References

G1. Coding Sources – Page 17

H: Literary References

H1. Literary Sources – Page 17

Part I: Research Question

A1. Propose one question relevant to WGU Telecom customer churn data using k-means clustering.

The telecommunications industry is highly competitive and retaining subscribers is paramount knowing customer churn rates can be as high as 25% as described in the WGU Telecom data dictionary. Customer retention leads to long-term profitability, therefore, a better understanding of customer habits and characteristics facilitates the ability to create marketing campaigns designed to encapsulate customer needs and wants. In other words, are there observable trends in the provided dataset that can account for customer habits.

Based on selected features, what clusters can be found using the k-means clustering algorithm and what are the characteristics of those clusters?

A2. Define one goal of the k-means clustering analysis.

The primary goal of this analysis is to apply the k-means clustering technique on specific features and glean valuable insight relative to customer characteristics and habits. To achieve this goal, features are selected, optimal clusters are explored, and enacted in the model. The variables chosen to better understand WGU Telecom customers are: Age, Children, Income, Monthly Charges, Tenure, and Bandwidth (gigabytes per year). Implementing the k-means clustering model on these variables will help identify common habits among these features, classify datapoints into clusters based on similarities, and further analysis of cluster centroids explored for common characteristics within each respective cluster.

Part II: Technique Justification

B1. Explain how k-means clustering technique analyzes the churn dataset with expected outcomes.

K-means clustering is an unsupervised machine learning algorithm that analyzes unlabeled data and creates groups (or clusters) with similar characteristics. The cluster analysis partitions a set of data in a way that the sum of the squared distances between objects and their assigned cluster mean is minimized (Sharma, 2023). The k-means algorithm starts with the first group of randomly selected centroids, which are used as starting points for each cluster, then the algorithm performs iterative calculations to optimize the positions of the centroids until the distance of cluster means are minimal or a predetermined number of iterations is achieved (Education Ecosystem LEDU, 2018). To define the optimal number of clusters, hyperparameter tests using elbow and silhouette methods were employed and determined the best performing model be based on two clusters.

When the k-means model is applied to the churn data, the model randomly selects the centroid of the first cluster minimizing cluster mean distances before iterating the second centroid and repeating the iterative process. The expected outcome will define all data points into two clusters that have minimized the distance of cluster means relative to their respective centroids. These results will effectively separate the churn data into two representative groups, each group will reflect classification of cluster based on similar characteristics.

B2. Summarize one assumption of the clustering technique.

Before utilizing the k-means clustering algorithm, it is important to understand the assumptions of the model. One important assumption to consider is how the k-means approach assumes the shape of the clusters to spherical and only considers distance from centroid. Meaning, when the algorithm calculates distance from the centroid, it does so in a spherical-like shape and only considers distance of data relative to the centroid (doesn't account for clusters with different size or densities), therefore, k-means cannot cluster complex geometrical shapes and assumes that features within a cluster have equal variance (Nagar, 2020). Taking this distance-based assumption into account, the data to be clustered by the model is preprocessed using a standard scaler which sets each feature to a mean equal to zero with a standard deviation of one.

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

Packages & Libraries	Support of Data Analysis
pandas	Create and manipulate dataframes, import/export csv files
seaborn	Visualization of data: histograms of cluster analysis
matplotlib.pyplot	Visualizations: k-cluster optimization analysis
from sklearn.preprocessing import StandardScaler	Scaling of variables before running k-means cluster model.
from sklearn.cluster import KMeans	K-means cluster model
from sklearn.metrics import silhouette_score	K-cluster optimization analysis and assess accuracy of model
from sklearn.pipeline import make_pipeline	Create pipeline to scale data and run k-means model

Part III: Data Preparation

C1. Describe one data preprocessing goal relevant to k-means clustering technique.

As previously discussed, the k-means model is a distance-based machine learning algorithm, therefore the data imputed into the model must undergo feature scaling. Implementing a distance-based model, one must understand that various features have different units of measurements. Features with different metrics can effectively "sway" the model as different measurements carry different weight or influence on distance measurements. For example, data sets with different units such as height (inches) and weight (pounds), a machine learning algorithm would consider weight more influential than height because the values for weight are larger and have higher variability from person to person (Arvai, n.d.). Therefore, previous to running the k-means model, the churn data underwent feature scaling using the standard scaler which sets mean values to zero with a standard deviation of one.

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.

Feature	Data Type	Use
Age	Continuous	K-means analysis
Children	Continuous	K-means analysis
Income	Continuous	K-means analysis
MonthlyCharge	Continuous	K-means analysis
Bandwidth_GB_Year	Continuous	K-means analysis
Tenure	Continuous	K-means analysis

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

1) Upload churn data (csv file) into dataframe and create backup.

```
# Upload CSV file and create backup
data =
pd.read_csv(r'C:\Users\andrew\Desktop\WGU_MSDA\D212_Data_Mining_II\Data
Sets\churn\churn_clean.csv')

data_backup = data.copy(deep = True)
```

2) Rename Item columns 1-8 to their corresponding names found in data dictionary

```
# Rename Item columns to represent their corresponding survey category
data.rename(columns = {'Item1':'Timely_response',
                       'Item2': 'Timely_fixes',
                       'Item3':'Timely_replacements',
                       'Item4':'Reliability',
                       'Item5':'Options',
                       'Item6':'Respectful_response',
                       'Item7':'Courteous_staff',
                       'Item8':'Active_listeners'}, inplace = True)
```

3) Search for duplicate records using Customer Id, defined in data dictionary as unique values

```
# Confirm no duplicate entries for data by checking for unique patient ids

data['Customer_id'].duplicated().value_counts()
```

4) Search data for missing data or null values

```
# Search for null values
data.isnull().sum()
```

5) Explore data frame, data types, and info

```
# Explore data types, identify target variables
# Search df info
data.info()
```

6) Create data frame for k-means model

```
# What features are to be used in the cluster analysis?
# CONTINUOUS -- age, children, income, monthly charges, bandwidth/gb/year,
tenure
t1 = data[['Age', 'Children', 'Income', 'MonthlyCharge',
'Bandwidth_GB_Year', 'Tenure']]
```

7) View summary statistics of numeric data

```
# View summary statistics - data to be scaled using standard scaler
t1.describe()
```

8) Create dataframe for analysis named → df_km

```
# Create copy of t1 df for analysis
df_km = t1.copy(deep = True)

# View first 3 rows of new dataframe
df_km.head(3)

# Export data frame used for analysis to csv file
df_km.to_csv(r'C:\\\\Users\\\\andrew\\\\Desktop\\\\WGU_MSDA\\\\D212_Data_M
ining_II\\\\PA\\\\Task_1\\\\df_km_clean_data_set.csv')
```

9) Scale data using standard scaler as k-means clustering is a distance-based algorithm

```
# Instantiate standard scaler
scaler = StandardScaler()

# Scale data for optimal cluster analysis
scaled_opt = scaler.fit_transform(df_km)

# View array of scaled data --- first three values
scaled_opt[:3]
```

10) Hyperparameter tuning: observe optimal clusters for k-means approach with a range of clusters 1 through 9. Employ the elbow method using loop to iterate different number clusters based on inertia values.

```
# Find optimal number of clusters
# Use Elbow method identifying upto 10 clusters
# Optimal number determined by "elbow" in plot

# Define Cluster range and empty list of inertia values
```

```
cluster = range(1, 10)
inertia_value = []

# Create loop to iterate number of clusters 1-10
for k in cluster:
  km = KMeans(n_clusters = k, init = 'k-means++', n_init = 1, random_state
= 61)
  km.fit(scaled_opt)
  inertia_value.append(km.inertia_)
```

11) Plot results of elbow method loop.

```
# Elbow Plot of optimal clusters
plt.plot(cluster, inertia_value, '-o')
plt.title("Optimal number of Clusters --- SCALED")
plt.xlabel("Number of Clusters, k")
plt.ylabel("Inertia Value")
plt.xticks(cluster)
plt.show()
```

12) Analysis of elbow method plot not clear as to optimal number of clusters, therefore, additional hyperparameter tuning is needed using silhouette score method.

```
# Elbow plot unclear if 2, 3, or 4 clusters are optimal
# Assess values of "k" using Silhouette Method
#create loop for silhouette scores
clusters = range(2, 10)
silhouette_scores = []

for k in clusters:
 kmeans = KMeans(n_clusters = k, init = 'k-means++', n_init = 1,
random_state = 61)
 kmeans.fit_transform(scaled_opt)
 labels = kmeans.labels_
 silhouette_scores.append(silhouette_score(scaled_opt, labels))
```

13) Plot results of silhouette score analysis

```
# plot silhouette scores
plt.plot(clusters, silhouette_scores, '-o')
plt.xlabel("Values of K")
plt.ylabel("Silhouette score")
plt.title("Silhouette analysis For Optimal k")
plt.show()
```

14) Print coefficient results of silhouette score analysis

```
# Silhouette Score Values
# Initialize kmeans model with n_clusters value and a random generator
# seed of 10 for reproducibility.
clusters = range(2, 10)

for k in clusters:

kmeans = KMeans(n_clusters = k, init = 'k-means++', n_init = 1,
random_state = 61)

labels = kmeans.fit_predict(scaled_opt)

# The silhouette_score gives the average value for all the samples.
# This gives a perspective into the density and separation of the formed
# clusters
  sil_score = silhouette_score(scaled_opt, labels)
  print("For n_clusters =", k, "The average silhouette_score is :",
round(sil_score, 3))
```

15) Define optimal clusters as 2. Prepare pipeline to scale data, instantiate k-means model, and define pipeline.

```
# Optimal number of clusters confirmed to be 2, with the highest
silhouette score
# 2 clusters has silhouette score of 0.302
# Define clusters as 2
# Instantiate scaler and define k-means model with number of clusters = 2
scaler = StandardScaler()
```

```
km_model = KMeans(n_clusters = 2, init = 'k-means++', n_init = 1,
random_state = 61)

# Create pipeline
pipeline = make_pipeline(scaler, km_model)
```

16) Fit pipeline to dataframe and identify labels

```
# Fit the pipeline to df_km
pipeline.fit(df_km)

# Identify Labels
labels = pipeline.predict(df_km)
```

17) Create results dataframe and add cluster results to new data frame for results analysis

```
# Create new dataframe from original df to add cluster labels
results = df_km.copy(deep = True)

# Add cluster labels back to df with readmissions included for comparison
of features
# Add labels to km results df
results['clusters'] = labels
```

C4. Provide a copy of the cleaned dataset.

SEE ATTACHED: df_km_clean_data_set.csv

Part IV: Analysis

D1. Describe the analysis technique you used to appropriately analyze the data.

Previous to running the k-means model, the analysis starts with hyperparameter optimization of selecting k-clusters for the algorithm. Employing the elbow and silhouette methods as evaluation tools assists with defining the best number of clusters for the k-means model. Provided k-means is a distance-based technique, the data was scaled using a standard scaler

before being entered into the optimal cluster analysis. The hyperparameter loop initializes the k-means algorithm with the k-means++ method, which selects initial cluster centroids using sampling based on an empirical probability distribution of the points'contribution to the overall inertia, improving accuracy and convergence (Scikit-learn, n.d.). The hyperparameter loop also iterates different values of clusters (range 1-9) for the elbow method, whereby inertia values are evaluated on a plot. Inertia measures the effectiveness of the k-means clusters by measuring the distance between each data point and its centroid, squaring the distance, and summon the squares in a cluster; with a good model reflecting low inertia and low number of clusters (Code Academy, n.d.). However, the results of the elbow plot didn't reflect a definitive number of clusters to define the k-means model, therefore, silhouette coefficients were explored (Fig. 1).

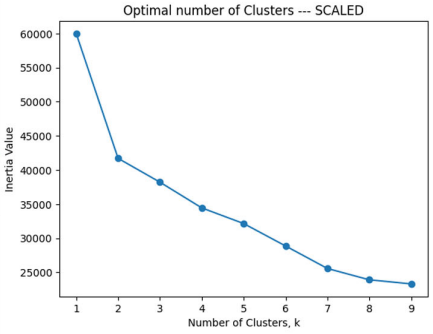
Figure 1: Elbow Method for Optimal "k" Cluster: Code and Plot

```
[13] # Find optimal number of clusters
     # Use Elbow method identifying upto 9 clusters
     # Optimal number determined by "elbow" in plot

     # Define Cluster range and empty list of inertia values
     cluster = range(1, 10)
     inertia_value = []

     # Create loop to iterate number of clusters 1-9
     for k in cluster:
       km = KMeans(n_clusters = k, init = 'k-means++', n_init = "auto", random_state = 61)
       km.fit(scaled_opt)
       inertia_value.append(km.inertia_)
```

```
     # Elbow Plot of optimal clusters
     plt.plot(cluster, inertia_value, '-o')
     plt.title("Optimal number of Clusters --- SCALED")
     plt.xlabel("Number of Clusters, k")
     plt.ylabel("Inertia Value")
     plt.xticks(cluster)
     plt.show()
```



The silhouette coefficients were calculated to assist with accuracy analysis by measuring the cluster cohesion and separation. The silhouette coefficient quantifies how well a data point fits into its assigned cluster based on two factors, how close the data point is to other points in the cluster and how far away the data point is from points in other clusters (Arvai, n.d.). The scores range from -1 to 1 with larger values representing clustered samples having a stronger association of their placement relative to their cluster as opposed to other clusters. After viewing the hyperparameter plots and silhouette scores, it was determined that the optimal number of clusters is two (Figs. 2 & 3).

Figure 2: Silhouette Method for Optimal "k" Cluster: Code and Plot

```
     # Elbow plot unclear if 2, 3, or 4 clusters are optimal
     # Assess values of "k" using Silhouette Method
     # SOURCE --- (Banerji, 2023)

     # Import Libraries
     from sklearn.cluster import KMeans
     from sklearn.metrics import silhouette_score

     #create loop for silhouette scores
     clusters = range(2, 10)
     silhouette_scores = []

     for k in clusters:
      kmeans = KMeans(n_clusters = k, init = 'k-means++', n_init = "auto", random_state = 61)
      kmeans.fit_transform(scaled_opt)
      labels = kmeans.labels_
      silhouette_scores.append(silhouette_score(scaled_opt, labels))
```

```
[16]  # plot silhouette scores
      plt.plot(clusters, silhouette_scores, '-o')
      plt.xlabel("Values of k")
      plt.ylabel("Silhouette score")
      plt.title("Silhouette analysis For Optimal k")
      plt.show()
```

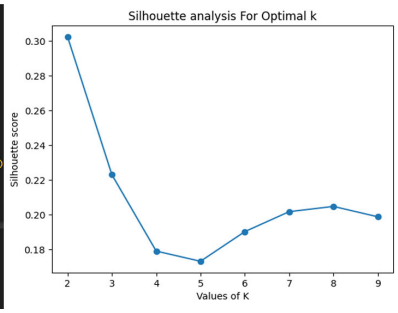


Figure 3: Silhouette Score Hyperparameter Loop Coefficients

```
[17] # Silhouette Score Values
     # SOURCE --- (Scikit-learn, n.d)
     clusters = range(2, 10)

     for k in clusters:
        kmeans = KMeans(n_clusters = k, init = 'k-means++', n_init = "auto", random_state = 61)
        labels = kmeans.fit_predict(scaled_opt)
        sil_score = silhouette_score(scaled_opt, labels)
        print("For n_clusters =", k, "The average silhouette_score is :", round(sil_score, 3))

     For n_clusters = 2 The average silhouette_score is : 0.302
     For n_clusters = 3 The average silhouette_score is : 0.223
     For n_clusters = 4 The average silhouette_score is : 0.179
     For n_clusters = 5 The average silhouette_score is : 0.173
     For n_clusters = 6 The average silhouette_score is : 0.19
     For n_clusters = 7 The average silhouette_score is : 0.202
     For n_clusters = 8 The average silhouette_score is : 0.205
     For n_clusters = 9 The average silhouette_score is : 0.199
```

With the k-means algorithm properly defined with two clusters, a pipeline was created to examine the churn data. The pipeline included a standard scaler and the k-means algorithm (Fig. 4). After the model is fit to the dataset using the pipeline, the predictive clusters are confirmed and added to a new dataframe for further analysis. To quantify placement of clusters, a value count was applied to the new data frame resulting in an almost perfect 50/50 split; 5002 customers were placed in cluster zero while the remaining 4998 were classified to cluster one (Fig. 5).

Figure 4: K-Means Model and Standard Scaler Pipeline

```
[18] # Optimal number of clusters confirmed to be 2, with the highest silhouette score
     # 2 clusters has silhouette score of 0.302
     # Define clusters as 2

     from sklearn.cluster import KMeans
     from sklearn.preprocessing import StandardScaler
     from sklearn.pipeline import make_pipeline

     # Instantiate scaler and define k-means model with number of clusters = 2
     scaler = StandardScaler()
     km_model = KMeans(n_clusters = 2, init = 'k-means++', n_init = "auto", random_state = 61)

     # Create pipeline
     pipeline = make_pipeline(scaler, km_model)

[19] # Fit the pipeline to df_km
     pipeline.fit(df_km)

     # Identify Labels
     labels = pipeline.predict(df_km)
```

Figure 5: Create Results Dataframe - Cluster Value Counts

```
     # Create new dataframe from original df to add cluster labels

     results = df.copy(deep = True)

     # Add cluster labels back to df with readmissions included for comparison of features
     # Add labels to km results df
     results['clusters'] = labels

[39] # View value counts of cluster analysis
     results['clusters'].value_counts()

     0    5002
     1    4998
     Name: clusters, dtype: int64
```

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

```
# Optimal number of clusters confirmed to be 2, with the highest
silhouette score
# 2 clusters has silhouette score of 0.302
# Define clusters as 2

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

# Instantiate scaler and define k-means model with number of clusters = 2
scaler = StandardScaler()
km_model = KMeans(n_clusters = 2, init = 'k-means++', n_init = 1,
random_state = 61)

# Create pipeline
pipeline = make_pipeline(scaler, km_model)
```

```
# Fit the pipeline to df_km
pipeline.fit(df_km)

# Identify Labels

labels = pipeline.predict(df_km)
```

```
# Create new dataframe from original df to add cluster labels
results = df.copy(deep = True)

# Add cluster labels back to df with readmissions included for comparison
of features
# Add labels to km results df
results['clusters'] = labels
```

```
# View value counts of cluster analysis
results['clusters'].value_counts()
```

```
0    5002
1    4998
Name: clusters, dtype: int64
```

Part V: Data Summary and Implications

E1. Explain the accuracy of your clustering technique.

Provided k-means clustering is an unsupervised machine learning algorithm, traditional accuracy scores do not apply as the data is unlabeled. Therefore, an effective technique to best address clustering accuracy is to calculate the silhouette score of the model. While calculating silhouette scores during hyperparameter tuning assists with defining the optimal number of clusters to fit the k-means model, it is expected that the defined model retains the same accuracy score. The value of the silhouette coefficient ranges from -1 to 1, where 1 denotes the best, or the data point is very compact within the cluster to which it belongs and far away from other clusters, while the worst value is -1, and values near 0 denote overlapping clusters (Banerji, 2023).

The average silhouette score for the 2-cluster k-means algorithm resulted with a coefficient of 0.302, or, the same value found through hyperparameter tuning (Fig. 6). Therefore, it can be determined the model is fairly accurate at classifying clusters. The silhouette coefficient is relatively close to zero, thus, it must be noted that some clusters may have potential overlap. Although, when observing the relationship of the cluster analysis relative to customer churn habits, the moderately accurate model provided results worthy of deeper analysis.

Figure 6: Results Accuracy Score calculating Silhouette Coefficient

```
# Analyze k-means cluster model
scaler = StandardScaler()
km_model = KMeans(n_clusters = 2, init = 'k-means++', n_init = "auto", random_state = 61)
X = scaler.fit_transform(df_km)
labels = km_model.fit_predict(X)


# The silhouette_score gives the average value for all the samples.
# This gives a perspective into the density and separation of the formed clusters
silhouette_avg = silhouette_score(X, labels, metric = 'euclidean')
print("For the 2-cluster k-means model, the Silhouette Score is: ", round(silhouette_avg, 3))
```

```
For the 2-cluster k-means model, the Silhouette Score is:  0.302
```

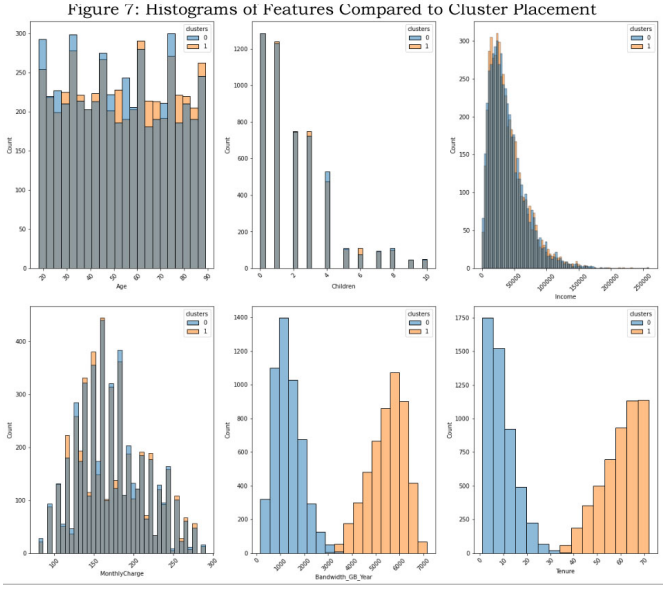
E2. Discuss the results and implications of your clustering analysis.

K-means clustering is a distance-based algorithm that aims to place data points relative to the mean difference between the data point and a cluster's centroid. Through an iterative process,

data is organized into groups where each member of the cluster has more in common with other members of the same cluster than with members of other groups with the most represenatvie point within the group being the centroid (Dahiya, 2021). The k-means clustering model returned with a mediocre performance, resulting in a silhouette coefficient of 0.302, indicative of a moderately successful classification of unlabeled data. The algorithm assessed the data and placed datapoints into two separate clusters, of which approximately equaled a 50/50 split of all data records; 5002 in cluster zero and 4998 in cluster one. While these unlabeled clusters by themselves are not descriptive of customer habits, further analysis of each cluster provides insight into the characteristics which determined cluster placement.

To garner better insight, an analysis of feature charactaristics was performed relative to cluster placement. As mentioned, the centroids are the best “representative” of a given cluster, thus, data closest to centroid proximity are put in the same cluster, while maintaining different clusters be dissimilar to each other. To analyze features relative to cluster placement, a loop was used to plot histograms of the features included in the analysis compared to cluster classification (Fig. 7). Given the silhouette coefficient of 0.302, overlap of clusters was expected and was reflected in the histogram plots. When observing cluster placement of Age, Children, Income, and Monthly Charges, there is obvious overlapping of clusters as evidenced by their respective histograms (Fig. 7).

Figure 7: Histograms of Features Compared to Cluster Placement



However, closer examination of Bandwidth (gb/year) and Tenure showed more definitive partitioning of cluster analysis (Figs. 8 & 9). The implications of these distinct cluster separations indicate that the features contributing most to the segregation of cluster analysis are likely Bandwidth (gb/year) and Tenure. Overall, the implications of the k-means application reflect a model that produces distinct clusters with low confidence based on the imputed attributes as evidenced by the relatively low silhouette score.

Figure 8: Closer Analysis of Cluster Placement: Bandwidth

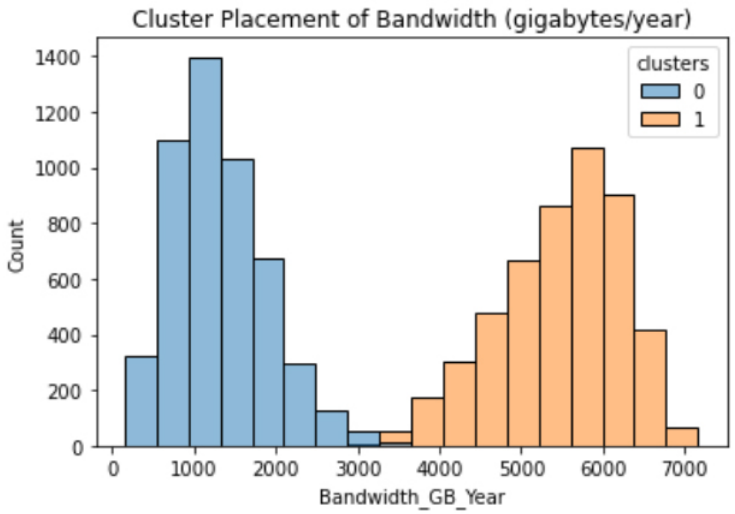
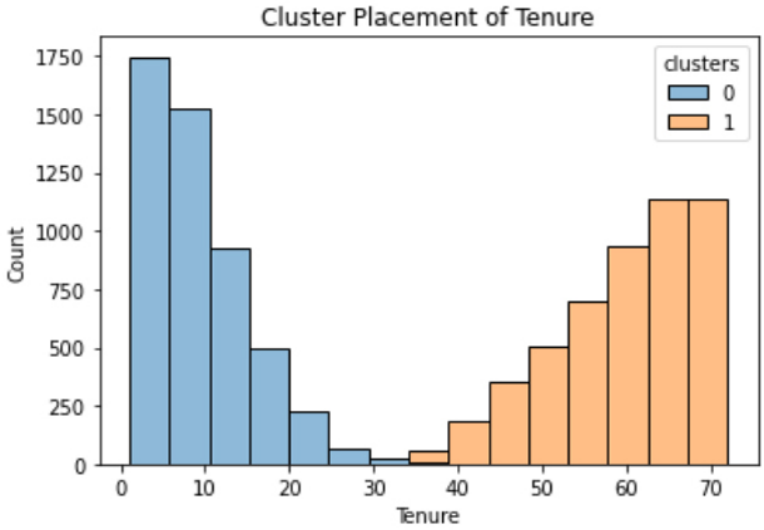


Figure 9: Closer Analysis of Cluster Placement: Tenure



E3. Discuss one limitation of your data analysis.

When using an unsupervised machine learning algorithm it is also important to understand the disadvantages and limitations of the analytical approach. With the distance-based k-means clustering algorithm, the model is highly sensitive to statistical outliers because the aim is to minimize the squared Euclidean distances between observation and the centroid of the cluster to which it belongs (Franklin, 2019). Therefore, sensitivity to outliers is compounded as an outlying data point is not proximate to the centroid and the squared Euclidean distance relative to the centroid becomes greater as a function of squaring the distance to the outlying data point. The greater the distance of outlying data, the greater the effect will have on the “mean distance calculation” thus resulting with sub-optimal centroids and cluster analysis. To counteract this

limitation, the dataset underwent feature scaling using the standard scaler previous to running the k-means model.

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.

The goal of the analysis is to identify distinct groups using the k-means clustering algorithm and observe characteristics summarized from the cluster classification. To achieve this goal, selected features were included in a k-means algorithm and determined if relationships could be found by clustering the unlabeled data. Through the use of hyperparameter tuning with elbow and silhouette methods, the optimal number of clusters of two was confirmed. To offset the k-means model's sensitivity to outliers, feature scaling through the use of a standard scaler was applied to ensure the algorithm performs as intended. Given the scaled data and k-means model instantiated with two predetermined clusters, the model resulted in a mediocre performance as reflected by the silhouette coefficient of 0.302 (range of 1 to -1). There are two courses of actions I would recommend. First, see if better cluster analysis can be established through the use of a principal component analysis prior to running the k-means algorithm, as opposed to the hand-selected feature approach used for this model. Second, given the silhouette score's proximity to zero (overlapping of clusters), I would suggest WGU Telecom not use this cluster analysis to gain valuable insight into customer habits.

Through the analysis of cluster classification, it was shown that some overlap exists between clusters within the following features; Age, Children, Income, and Monthly Charges, thus not great indicators of definitive customer habits based on cluster. However, as seen in figures eight and nine, cluster placements are as distinct as the bimodal distributions of the histograms with virtually no overlapping of clusters. This indicates that the strongest influence of centroid formation and customer segmentation resides in Bandwidth and Tenure. Therefore, I would not suggest WGU Telecom use this cluster analysis to create marketing campaigns as this model provided clusters with a low confidence based on the imputed attributes. As mentioned, the best course of action would be to perform a principal component analysis previous to applying a k-means clustering algorithm.

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.

SEE ATTACHED: Mecchi_Panopto_D212_Data_Mining_II_Task_1

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ab360d13-d071-4d93-ba2a-b059012c9820

CODE SOURCES

Banerji, A. (2023, August 3). K-Mean: Getting the Optimal Number of Clusters. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/.

Scikit-learn. (n.d.). KMeans.. Scikit-learn 1.3.0 - documentation. Retrieved August 3, 2023. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html.

REFERENCES

Arvai, K. (n.d.) K-Means Clustering in Python, A Practical Guide. Real Python. Retrieved August 3, 2023. https://realpython.com/k-means-clustering-python/#evaluating-clustering-performance-using-advanced-techniques

Code Academy. (n.d.) Intro to Machine Learning: Clustering: K-means. Code Academy. Retrieved August 3, 2023. https://www.codecademy.com/learn/machine-learning/modules/dspath-clustering/cheatsheet.

Dahiya, S. (2021 July 6). K-Means Clustering: Centroid. Programs Buzz. https://www.programsbuzz.com/article/k-means-clustering-centroid.

Education Ecosystem LEDU. (2018, September 12). Understanding K-Means Clustering in Machine Learning. Medium. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1.

Franklin, S.J. (2019, November 16). Effect of Outliers on K-Means Algorithm Using Python. Analytics Vidhya. https://medium.com/analytics-vidhya/effect-of-outliers-on-k-means-algorithm-using-python-7ba85821ea23.

Nagar, A. (2020, January 26). K-means Clustering – Everything You Need to Know. Analytics Vidhya. https://medium.com/analytics-vidhya/k-means-clustering-everything-you-need-to-know-175dd01766d5.

Scikit-learn. (n.d.). KMeans.. Scikit-learn 1.3.0 - documentation. Retrieved August 3, 2023. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.

Sharma, P. (2023, May 19). The Ultimate Guide to K-Means Clustering: Definition, Methods and Applications. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/.