**Performance Assessment**

OFM4 — OFM4 Task 2: Dimensionality Reduction Methods

Bader Ale

Department of Information Technology, Western Governors University

D212 Data Mining II

2024

Table of Contents

[Part I: Research Question 3](#_Toc179451279)

[Part II: Technique Justification 3](#_Toc179451280)

[Part III: Data Preparation 5](#_Toc179451281)

[Part IV: Analysis 7](#_Toc179451282)

[Part V: Data Summary and Implications 7](#_Toc179451283)

# Part I: Research Question

The purpose of the analysis using k-means clustering is to answer the following “*In the context of a major hospital chain, can we identify clusters of patients with similar demographic and clinical profiles who share patterns in their hospital readmission experiences?".*  For this task, k-means clustering was used on the given medical data set. This unsupervised learning algorithm was used to group patients according to the following common features: Age, Vitamin D levels during hospital stay and Readmission status. These clusters can help medical administrators understand patterns in these clusters that show them a high risk for readmission, which would form a proper basis on which effective targeted management interventions could be based.

# Part II: Method Justification

The K-means algorithm is a partitioning method that groups data points—in this case, patients—based on their similarities in selected characteristics, such as hospital billing and length of stay. For this analysis, the clustering focuses on three continuous variables: TotalCharge (average daily amount billed), Additional\_charges (miscellaneous service costs), and Initial\_days (duration of initial hospital admission). These features enable grouping patients with similar hospital usage patterns, potentially helping administrators make cost-effective decisions tailored to each group.

Each patient is represented as a data point within a three-dimensional space corresponding to TotalCharge, Additional\_charges, and Initial\_days. Initially, the algorithm assigns each data point to randomly chosen centroids, calculated using Euclidean distance. Through iterative steps, the centroids are updated to reflect the average values of patients within each cluster, repeating until the centroids stabilize and the clusters become fixed. This analysis aims to reveal clusters showing distinct financial and hospital utilization patterns. For example, one cluster might consist of patients with higher TotalCharge and Additional\_charges and longer Initial\_days, indicating longer-term, higher-cost patients, while another might consist of those with lower costs and shorter stays, representing less intensive hospital use. Identifying these patterns can guide hospital administrators in understanding and optimizing resource utilization and patient care costs. K-means clustering assumes that data points within each cluster form a spherical shape with a central centroid, where points are evenly spread around the center and distances between points and their centroid remain relatively consistent across clusters. This assumption fits best when clusters are symmetrical but can be less effective with data of varying shapes or sizes

(scikit-learn developers, n.d.)

For this assessment, pandas was used to handle the dataset, seaborn for visualizations, and scikit-learn for applying the K-means algorithm. The scikit-learn package was also employed to scale data, preventing unit disparities among variables from impacting clustering outcomes.

# Part III: Data Preparation

A primary objective in preparing data for k-means clustering is to ensure that the variables are on comparable scales. Since k-means clustering relies on calculating distances, variables with broader ranges can unduly affect clustering outcomes. For instance, a variable such as TotalCharge, with a potentially large financial range, might inadvertently influence clustering more than Initial\_days if left unscaled. By standardizing continuous variables, we enable each feature to contribute equally to the analysis, preventing clustering results from being skewed by differing measurement scales.

For this analysis, three continuous variables were selected: TotalCharge, Additional\_charges, and Initial\_days. TotalCharge represents the average daily charge incurred by a patient during their initial hospital stay, providing insight into the financial demands of treatment. Additional\_charges reflect additional costs from other services, treatments, or procedures, which can indicate care complexity. Lastly, Initial\_days captures the length of the initial stay, a critical indicator of the intensity of medical care provided. These variables collectively offer a well-rounded basis for clustering, allowing for an understanding of patient patterns in both resource utilization and financial impact.

To prepare the data for k-means clustering, the first step involved importing the dataset using pandas. This allowed for an initial inspection of the data and identification of relevant features for the analysis. The dataset was loaded with the following code:

1. df = pd.read\_csv('D:\GitHub Repos\WGU\_MSDA\D212\_Data Mining\medical\_clean.csv')

2. df.head()

Once the data was loaded, the next step was data cleaning. This involved removing unnecessary columns, such as customer identifiers and geographic details, to focus on the key variables relevant to clustering. These irrelevant columns were dropped using the following code:

1. df\_kmeans = df.copy()

2. df\_kmeans.drop(['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'Zip', 'Lat', 'Lng',

3. 'Population', 'Area', 'TimeZone', 'State', 'County', 'Job', 'Marital',

4. 'Gender', 'Soft\_drink', 'Initial\_admin', 'HighBlood', 'Stroke', 'Complication\_risk',

5. 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',

6. 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Asthma', 'Services', 'Item1', 'Item2'], axis=1, inplace=True)

7. df\_kmeans.head()

8.

Next, from the newly created dataframe *df\_kmeans* only the continuous variable were selected using the code shown below. The columns were also saved in order to use them in the dataframe.

A copy of the cleaned dataset is provided for review.

1. # Selecting continuous variables

2. continuous\_vars = ['Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'vitD\_supp', 'Initial\_days', 'TotalCharge', 'Additional\_charges']

3.

4. # Creating a new dataframe with only continuous variables

5. df\_kmeans = df[continuous\_vars]

6.

The following is the output of this new dataframe:



Since k-means clustering works solely with continuous numerical data, standardization of TotalCharge, Additional\_charges, and Initial\_days was necessary. This scaling process was performed using StandardScaler from the scikit-learn library, ensuring that each variable contributed equally to the Euclidean distance computations. The standardization was implemented as follows:

1. #Scaling our dataset for the KMeans algorithm

2. from sklearn.preprocessing import StandardScaler

3.

4. scaler = StandardScaler()

5. df\_means\_columns = df\_kmeans.columns

6. df\_kmeans\_scaled = pd.DataFrame(scaler.fit\_transform(df\_kmeans), columns=df\_means\_columns)

7.

A new dataframe called *clusterdata* was created to be used only in the clustering algorithm. From the bigger *df\_means* dataframe, *clusterdata* contained the continuous variables of interest *Initial\_days, TotalCharge and Additional\_charges*. The code shown below was used to perform this task.

1. # Creating a new dataframe with just the variables of interest for k-means clustering

2. clusterdata = df\_kmeans\_scaled[['Initial\_days', 'TotalCharge', 'Additional\_charges']]

3. clusterdata.head()

4.

The clustering algorithm was first initialized and the dataframe *clusterdata* was passed. Initially, a parameter of *n\_cluster* was set to 3 in order to be run the code.

1. # Initializing KMeans

2. from sklearn.cluster import KMeans

3.

4. k\_model = KMeans(n\_clusters=3, n\_init= 25, random\_state=42)

5.

6. k\_model.fit(clusterdata)

7.

The centroids were also calculated from the attribute “cluster\_centers” as shown below.

1. # Calculating centroids

2. centroids = pd.DataFrame(k\_model.cluster\_centers\_, columns = clusterdata.columns)

3. centroids

4.

A screenshot of a cell phone

Description automatically generated

Finally, a scatterplot with the centroids was graphed to visually show the centroids.

1. plt.figure(figsize=(12, 10));

2. ax = sns.scatterplot(data=clusterdata,

3.                      x='TotalCharge',

4.                      y='Initial\_days',

5.                      hue=k\_model.labels\_,

6.                      palette='colorblind',

7.                      alpha=0.9,

8.                      s=150,

9.                      legend=True)

10.

11. sns.scatterplot(data=centroids,

12.                 x='TotalCharge',

13.                 y='Initial\_days',

14.                 hue=centroids.index,

15.                 palette='colorblind',

16.                 s=900,

17.                 marker='D',

18.                 edgecolor='black',

19.                 legend=False);

20.

A screen shot of a graph

Description automatically generated

A copy of the cleaned dataset is provided for review.

# Part IV: Analysis

The Elbow Method was chosen to determine the most appropriate number of clusters (3) in the dataset. This technique works by running the k-means algorithm over a range of cluster values and calculating the within-cluster sum of squares (WCSS) for each option. The WCSS reflects how well data points fit within each cluster, with lower values signifying tighter groupings. As the number of clusters increases, the WCSS naturally decreases because more clusters provide more flexibility in grouping data points. However, after a certain point, the reduction in WCSS slows down, forming an "elbow" in the graph. This elbow point suggests the ideal number of clusters, as adding more clusters beyond this point yields minimal improvement in the tightness of the groupings. This method provides a visual way to assess the optimal number of clusters and is commonly used due to its simplicity and clarity. While the exact point of the elbow can sometimes be subjective, it offers a useful starting point for determining cluster numbers, especially when used in combination with other metrics like the Silhouette Score to validate the results (Basil, 2024).

A graph with a line

Description automatically generated

1. from sklearn.cluster import KMeans

2.

3. # Create an empty list to store inertia values

4. inertia = []

5.

6. # Fit the k-means model for cluster values ranging from 1 to 10

7. for k in range(1, 11):

8.     kmeans = KMeans(n\_clusters=k, random\_state=42)

9.     kmeans.fit(df\_kmeans\_scaled)

10.     inertia.append(kmeans.inertia\_)

11.

12. # Plot inertia vs. number of clusters (k)

13. plt.figure(figsize=(8,5))

14. plt.plot(range(1, 11), inertia, marker='o')

15. plt.title('Elbow Method for Optimal k')

16. plt.xlabel('Number of Clusters (k)')

17. plt.ylabel('Inertia')

18. plt.show();

19.

This code is designed to help determine the optimal number of clusters for a k-means clustering model using the Elbow Method. It starts by creating an empty list to store the inertia values, which measure how tightly data points are grouped within clusters. Lower inertia values indicate better, more compact clustering. The code then fits the k-means algorithm for a range of cluster numbers, from 1 to 10, by looping through different values of `k` (the number of clusters). For each value of `k`, the model calculates the inertia and stores it in the list. Finally, the inertia values are plotted against the number of clusters, with the resulting graph displaying the "elbow" point, which helps identify the optimal number of clusters. The elbow is the point where the reduction in inertia slows down, indicating that adding more clusters beyond this point provides little improvement in clustering performance.

# 

Works Cited

Basil. (2024, September 27). *Elbow Method for Optimal Cluster Number in K-Means*. Retrieved October 2024, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/

scikit-learn developers. (n.d.). *2.3. Clustering*. Retrieved October 2024, from Unsupervised Learning: https://scikit-learn.org/stable/modules/clustering.html#k-means