## PART I: RESEARCH QUESTION

### PROPOSAL OF QUESTION

Using the K-means clustering technique, what are the distinct patient segments based on their Total Charge and Income?

### DEFINED GOAL

This analysis aims to identify homogeneous patient groups using k-means clustering based on TotalCharge and Income. This enables the hospital to tailor healthcare services, allocate resources efficiently, and develop targeted financial assistance programs.

## PART II: TECHNIQUE JUSTIFICATION

### EXPLANATION OF CLUSTERING TECHNIQUE

K-means clustering can be employed to analyze the dataset, which consists of patient data regarding TotalCharge and Income. Initially, the variables TotalCharge and Income are preprocessed. This may involve standardizing the data to ensure that both variables contribute equally to the clustering process. Before applying k-means clustering, the elbow method and/or silhouette analysis are used to determine the optimal number of clusters (k). Then, random initial centroids are assigned to each cluster. These centroids represent the mean of the data points within each cluster. Then, each data point is assigned to the cluster whose centroid is closest to it based on a distance metric. After the initial assignment, the centroids of the clusters are recalculated by taking the mean of all data points assigned to each cluster. Assigning data points and updating centroids are repeated iteratively until convergence. Finally, convergence is achieved, the clustering process is finalized, and each patient is assigned to a specific cluster based on the final centroids.

Expected outcomes of K-means clustering are the segmentation of patients and homogenous patient groups. The patients will be segmented into distinct groups based on their TotalCharge and Income characteristics. Moreover, each cluster will represent a group of patients with similar TotalCharge and Income profiles.

### SUMMARY OF THE TECHNIQUE ASSUMPTION

An assumption of the k-means clustering technique is that clusters are spherical and of similar size. This means that the algorithm assumes that the data points within each cluster are tightly grouped around the cluster centroid in a spherical shape and that the clusters are roughly equal in size.

### PACKAGE OR LIBRARIES LIST

| Packages & Libraries | Usage |
|---|---|
| **csv** | For reading and writing CSV files |
| **pandas** | For data manipulation, provides data structure like DataFrame and Series |
| **warnings** | For controlling warning messages by suppressing certain warning messages that may not be critical |

| | |
|---|---|
| **platform** | Provides information about the platform and/or version of the Python running |
| **numpy** | For numerical computing, also for mathematical functions that enables arrays to efficiently operate |
| **matplotlib.pylot** | A plotting library; used to create histograms, bar charts and ROC curves |
| **seaborn** | A statistical data visualization based on matplotlib; used to create heatmaps and correlation matrices |
| **sklearn (scikit-learn)** | A machine learning library that provides tools for data mining and data analysis. |
| **sklearn.preprocessing.StandardScaler** | Provides a method to scale features by removing the mean and scaling to unit variance |
| **sklearn.cluster.KMeans** | A clustering algorithm that partitions data into K clusters based on the Euclidean distance between data points and cluster centroids |
| **sklearn.metrics.silhouette_score** | A score to evaluate the quality of clustering. Quantifies how well each data point fits into its assigned cluster relative to other clusters |

## PART III: DATA PREPARATION

### DATA PREPROCESSING

One common data preprocessing technique for clustering is the standardization or normalization of the data. Standardization involves scaling the features to have a mean of 0 and a standard deviation of 1. Normalization is scaling them to a range between 0 and 1. This technique ensures that all variables contribute equally to the clustering process. It is also essential to prevent features with larger scales from dominating the distance calculations.

### DATA SET VARIABLES

| Variable | Data type |
|---|---|
| **TotalCharge** | Continuous |
| **Income** | Continuous |

## STEPS FOR ANALYSIS

The preprocessing steps for this analysis:

1. Initialize all libraries and packages to be used

```
# Data Handling
import csv
import pandas as pd

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")

# System Information
import platform

# Numeric Computation and Array Handling
import numpy as np

# Data Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning - Clustering
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Machine Learning - Preprocessing
from sklearn.preprocessing import StandardScaler
```

2. Print Python version

```
# Print Python version
print("Python Version:", platform.python_version())
```

3. Load the dataset into Pandas DataFrame

```
# Specifiy CSV file path
file_path = r'C:\Users\kolgi\OneDrive - Western Governors University\D212\medical_clean.csv'

# Open the CSV file and read it using DictReader
with open(file_path, 'r') as csvfile:
    csvreader = csv.DictReader(csvfile)

# Read the CSV file into a pandas DataFrame then open head
df = pd.read_csv(file_path)
```

4. View the first 5 rows of the DataFrame
   **df.head()**
5. View the index, column names, non-null count, and data types
   **df.info()**
6. Generate descriptive statistics such as the count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile and maximum values
   **df.describe()**
7. Check for null/missing values in the dataset, then count the null values for each column
   **df.isnull().sum()**
8. Check for duplicates in the data
   **df.duplicated()**
9. Create a new Dataframe named new_df containing only the variables TotalCharge and Income

```
# Create a new DataFrame with just 'TotalCharge' and 'Income'
new_df = df[['TotalCharge', 'Income']].copy()

# Display the new DataFrame
print(new_df)
```

10. Make a scatter plot to visualize TotalCharge and Income

```
# Plotting TotalCharge vs. Income
plt.figure(figsize=(8, 6))
plt.scatter(new_df['TotalCharge'], new_df['Income'], alpha=0.5)
plt.title('Scatter Plot of TotalCharge vs. Income')
plt.xlabel('TotalCharge')
plt.ylabel('Income')
plt.grid(True)
plt.savefig("t1_scatterplot.png")
plt.show()
```

11. Generate descriptive statistics for the new_df

```
# Call describe() method for the two variables
description = new_df.describe()

description
```

12. Use StandardScaler() to standardize TotalCharge and Income. Then, fit the scaler and transform the variables. Finally, create a new DataFrame for the scaled variables called scaled_df

```
# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler to the data and transform the variables
scaled_data = scaler.fit_transform(new_df)

# Create a new DataFrame with the scaled variables
scaled_df = pd.DataFrame(scaled_data, columns=['TotalCharge', 'Income'])

# Display the scaled DataFrame
scaled_df
```

13. Save the scaled dataframe into a .csv file

```
# Save the filtered DataFrame to a CSV file
scaled_df.to_csv('task1_scaled_df.csv', index=False)
```

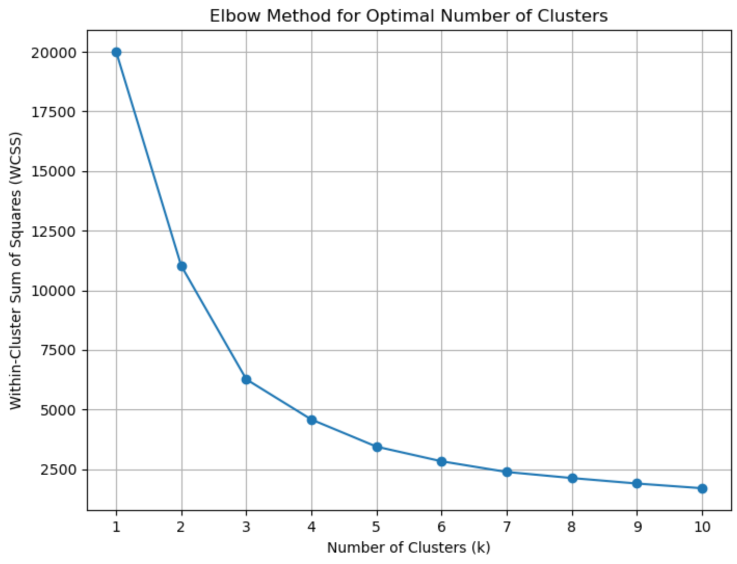
## CLEANED DATA SET

*File attached as task1_scaled_df**.csv***

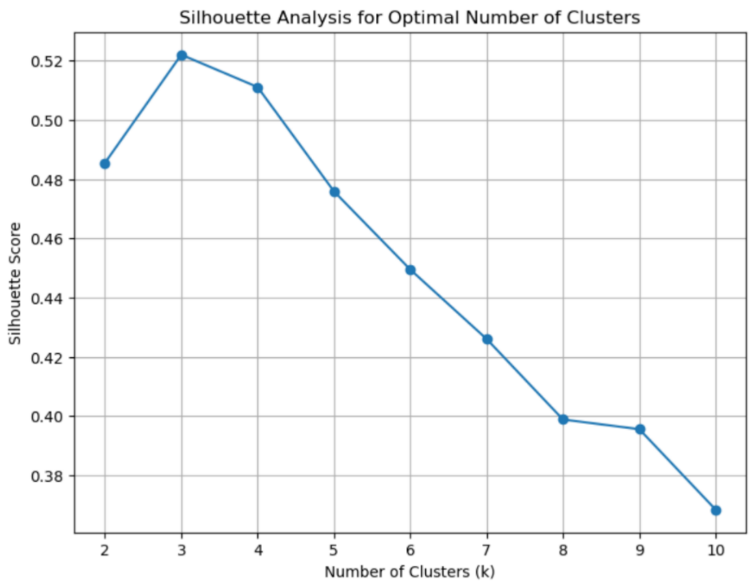
## PART IV: ANALYSIS

### OUTPUT & INTERMEDIATE CALCULATION

For this analysis, 2 methods were used to find the optimal number of clusters. The methods used are the elbow method and silhouette analysis. Below are graphs from each method generated to better visualize the number of clusters (k).



In the elbow method, the x-axis represents the number of clusters (k), while the y-axis represents the within-cluster sum of squares (WCSS). WCSS is a measure of the variability within each cluster.
As stated in the method's name, the "elbow" point in the plot is essential to interpreting the graph. The "elbow" point is where the rate of decrease in WCSS. This indicates the optimal number of clusters. This point represents a balance between maximizing the number of clusters while minimizing the WCSS. In this case, 3 is the optimal number of clusters as seen where the WCSS starts to level off.



```
Optimal number of clusters: 3
```

Silhouette score measures how similar an object is to its own cluster compared to other clusters. Higher silhouette score indicates better-defined clusters. The x-axis represents the number of clusters (k), while the y-axis represents the silhouette score. The optimal number of clusters is determined by finding the peak on the graph. In this case, the silhouette analysis suggests that the optimal number of clusters for the dataset is 3.

## CODE EXECUTION

1. For the elbow method: Calculate the WCSS for different values of k using the KMeans algorithm. Then, plot the elbow curve to visualize the WCSS.

```
# Calculate within-cluster sum of squares for different values of k
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_df)
    wcss.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='-')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.xticks(range(1, 11))
plt.grid(True)
plt.savefig("t1_elbowmethod.png")
plt.show()
```

2. For the silhouette score: From a range of 2 to 10, compute the silhouette score for each value of k using a for loop. Then, plot the silhouette score.

```
# Initialize an empty list to store silhouette scores
silhouette_scores = []

# Range of clusters to try
min_clusters = 2
max_clusters = 10

# Compute silhouette score for each value of k
for k in range(min_clusters, max_clusters + 1):
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(scaled_df)
    silhouette_avg = silhouette_score(scaled_df, cluster_labels)
    silhouette_scores.append(silhouette_avg)

# Plot the silhouette scores
plt.figure(figsize=(8, 6))
plt.plot(range(min_clusters, max_clusters + 1), silhouette_scores, marker='o', linestyle='-')
plt.title('Silhouette Analysis for Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.xticks(range(min_clusters, max_clusters + 1))
plt.grid(True)
plt.savefig("t1_silhouette.png")
plt.show()

# Find the optimal number of clusters with the highest silhouette score
optimal_k = np.argmax(silhouette_scores) + min_clusters
print("Optimal number of clusters:", optimal_k)
```

3. Initialize KMeans clustering model with 3 clusters and then fit the model to the scaled data.

```
# Initialize KMeans with 3 clusters
kmeans = KMeans(n_clusters=3, n_init=25, random_state=300)

# Fit KMeans to the scaled data
kmeans.fit(scaled_df)
```

```
▾                    KMeans
KMeans(n_clusters=3, n_init=25, random_state=300)
```

4. Calculate the count of patients in each cluster and store the result in cluster_counts

```
# Get cluster labels
cluster_labels = kmeans.labels_

# Add cluster labels to the scaled DataFrame
scaled_df_with_clusters = scaled_df.copy()
scaled_df_with_clusters['Cluster'] = cluster_labels

# Count patients per cluster
cluster_counts = scaled_df_with_clusters['Cluster'].value_counts()

# Display the count of patients per cluster
print(cluster_counts)
```

```
Cluster
1    4240
0    4219
2    1541
Name: count, dtype: int64
```

5. Extract the cluster centers obtained from the KMeans clustering algorithm and assign them to the columns TotalCharge and Income

```
# Extract centroids with TotalCharge and Income columns
centroid_df = pd.DataFrame(kmeans.cluster_centers_, columns=['TotalCharge', 'Income'])

# Display the centroids DataFrame
print(centroid_df)
```

```
   TotalCharge    Income
0    -0.948244 -0.336102
1     0.967796 -0.323370
2    -0.068533  1.808531
```

6. Plot the centroids annotated with cluster numbers

```
# Extract centroids
centroids = kmeans.cluster_centers_

# Plotting TotalCharge vs. Income with centroids annotated with cluster numbers
plt.figure(figsize=(8, 6))
plt.scatter(scaled_df['TotalCharge'], scaled_df['Income'], c=cluster_labels, cmap='viridis', alpha=0.5)

# Plot centroids and annotate with cluster numbers
for i, centroid in enumerate(centroids):
    plt.scatter(centroid[0], centroid[1], marker='${}$'.format(i), s=200, c='red', label='Centroids')

plt.title('Clustering with Centroids')
plt.xlabel('TotalCharge')
plt.ylabel('Income')
plt.legend()
plt.grid(True)
plt.savefig("t1_centroids.png")
plt.show()
```

7. Use one-hot encoding using get_dummies() on the Gender. Then, convert the No to 0 and Yes to 1

```python
# Perform one-hot encoding for the 'Gender' variable
df_encoded = pd.get_dummies(df, columns=['Gender'])

# Convert boolean columns to binary values (0 for 'No', 1 for 'Yes')
df_encoded [['Gender_Female', 'Gender_Male', 'Gender_Nonbinary']] = df_encoded[['Gender_Female', 'Gender_Male', 'Gender_Nonbinary']].astype(int)

# Display the DataFrame with one-hot encoded 'Gender' variable
print(df_encoded.head())
```

8. Add the columns Cluster, Age, Gender_Female, Gender_Male, and Gender_Nonbinary to scaled_df_with_clusters

```python
# Add 'Gender' and 'Age' to the DataFrame
scaled_df_with_clusters = scaled_df.copy()
scaled_df_with_clusters['Cluster'] = cluster_labels
scaled_df_with_clusters['Age'] = df['Age']
scaled_df_with_clusters['Gender_Female'] = df_encoded['Gender_Female']
scaled_df_with_clusters['Gender_Male'] = df_encoded['Gender_Male']
scaled_df_with_clusters['Gender_Nonbinary'] = df_encoded['Gender_Nonbinary']

# Display the DataFrame with labeled clusters and additional variables
scaled_df_with_clusters
```

9. Calculate the value counts and percentage distribution of Gender

```python
# Calculate value counts of gender
gender_counts = df['Gender'].value_counts()

# Calculate percentage distribution
gender_percentage = (gender_counts / len(df)) * 100

# Create a DataFrame to store the gender counts and percentages
gender_distribution = pd.DataFrame({'Counts': gender_counts, 'Percentage': gender_percentage})

# Display the gender distribution DataFrame
print(gender_distribution)
```

10. Calculate the mean and median values of Gender_Female, Gender_Male, Gender_Nonbinary, Age, Income, and Total Charge after grouping by Cluster.

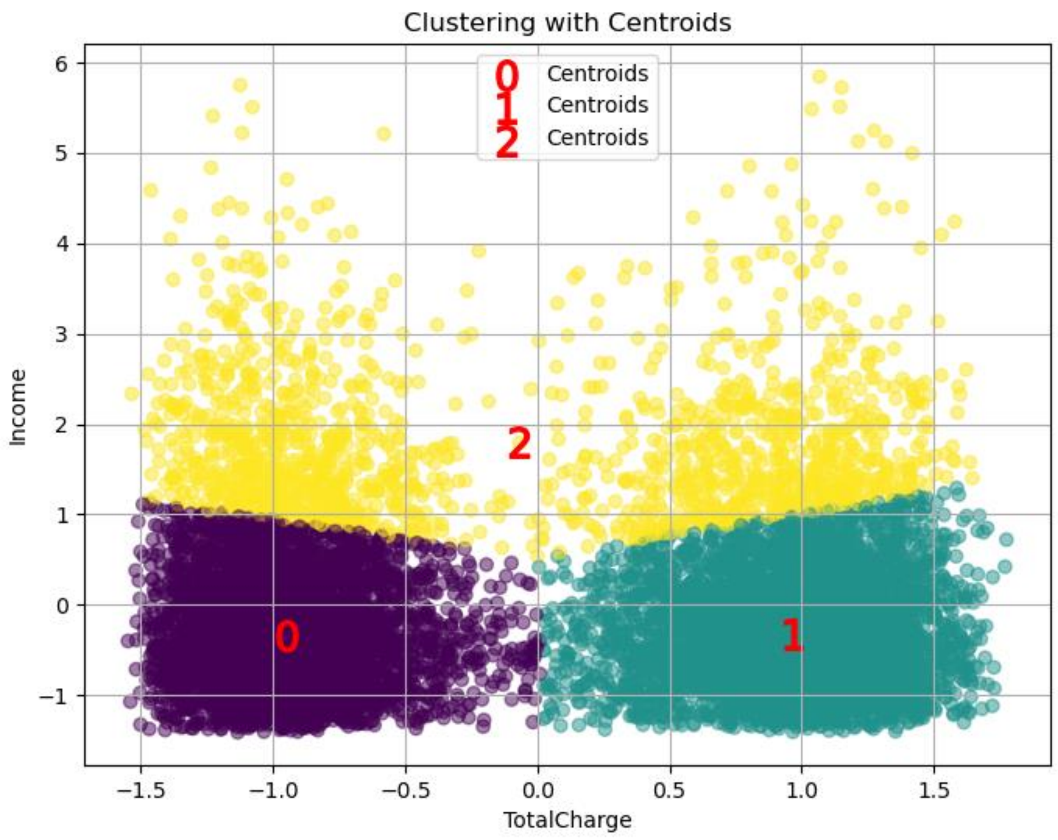
```python
# Group by 'Cluster' column and calculate mean and median for specified columns
cluster_stats = scaled_df_with_clusters.groupby('Cluster').agg({'Gender_Female': 'mean',
                                                              'Gender_Male': 'mean',
                                                              'Gender_Nonbinary': 'mean',
                                                              'Age': 'median',
                                                              'Income': 'median',
                                                              'TotalCharge': 'median'}).reset_index()

# Display the DataFrame with cluster statistics
cluster_stats
```

## PART V: DATA SUMMARY & IMPLICATIONS

### QUALITY OF THE CLUSTERING TECHNIQUE

There are many factors to consider when looking at the quality of the clustering technique. This analysis used silhouette analysis, visual inspection, and cluster cohesion.

The silhouette score is 0.52, indicating that the clustering solution has reasonably well-defined clusters. The score means that data points within each cluster are closer than data points in other clusters. This suggests that the clustering solution effectively separates patients into distinct groups.



Above is a graph of the clustering with centroids. Each cluster has distinct colors to distinguish between clusters visually. As an overall visual inspection, clusters are well-separated from each other and internally cohesive. Moreover, there are clear boundaries and minimal overlapping between neighboring clusters. Overall, clusters have distinct boundaries and minimal overlap, showing high-quality clustering.

| Cluster | Count |
|---|---|
| **0** | 4219 |
| **1** | 4240 |
| **2** | 1541 |

Above are the cluster counts of this analysis. The distribution is not perfectly balanced, but it is not heavily skewed either. It suggests that the clusters are of varying sizes, which is common in real-world data. However, it is essential to ensure that each cluster represents a meaningful segment of patients with distinct characteristics that can inform decision-making.

## RESULTS & IMPLICATIONS

The table below provides a summary of the characteristics of each cluster identified in the clustering analysis. The gender distribution is 50.18% female, 47.68% male, and 2.14% nonbinary.

| Cluster | Female | Male | Non-binary | Age | Income | Total Charge |
|---|---|---|---|---|---|---|
| 0 | 50.98% | 46.98% | 2.04% | 54.0 | -0.413698 | -0.981812 |
| 1 | 49.83% | 47.97% | 2.19% | 53.0 | -0.396655 | 1.006446 |
| 2 | 48.93% | 48.80% | 2.27% | 53.0 | 1.565609 | -0.378333 |

Cluster 0 represents a diverse group of patients with almost the same gender distribution as the dataset. The median age suggests that this cluster includes patients who may be in the middle to older age range. The negative median income indicates that patients in this cluster may have lower incomes on average. The negative median total charge suggests that patients in this cluster may have incurred lower healthcare charges on average. Healthcare services tailored for this cluster may address the needs of middle to older-aged individuals with lower incomes. Resource allocation efforts could prioritize financial assistance programs for patients in this cluster who may face financial barriers to healthcare access.

Cluster 1 represents another diverse group of patients in terms of gender, with a median age like Cluster 0. The median income suggests that patients in this cluster may have slightly higher incomes compared to 0. However, the positive median total charge indicates that patients in this cluster may have incurred higher healthcare charges on average. Healthcare services for this cluster may need to address the healthcare needs of patients with slightly higher incomes who still face significant healthcare expenses. Strategies for resource allocation could focus on providing services that address the specific healthcare needs of this group.

Cluster 2 represents another diverse group of patients in terms of gender and age, like the other clusters. However, this cluster stands out for its notably higher median income than the others. Despite the higher income, patients in this cluster have lower healthcare charges on average compared to the other clusters. Healthcare services for this cluster may need to consider the unique healthcare utilization patterns of patients with higher incomes but lower healthcare charges. Resource allocation efforts could focus on preventive care programs to promote healthy behavior and reduce healthcare costs over time.

Overall, the clustering analysis reveals distinct segments based patient segments based on demographic and financial characteristics. This is crucial in providing valuable insights for tailoring healthcare services, allocating resources efficiently, and developing financial assistance programs. Healthcare providers can use these insights to design targeted interventions that address the specific needs of each patient segment. This is essential in improving healthcare outcomes and enhancing the overall efficiency of healthcare delivery.

## LIMITATION

While the analysis focuses on total charge income, other factors could significantly influence healthcare utilization patterns and financial circumstances. Other factors, such as patient demographics, medical history, geographic location, or lifestyle factors, could provide a more comprehensive understanding of patient characteristics and behaviors. Without considering these additional variables, the clustering analysis may provide an incomplete picture of patient segmentation and may not capture all relevant aspects of patient diversity. As a result, the strategies developed based on the clustering results may not fully address the diverse needs of the patient population.

## COURSE OF ACTION

The organization can take the following actions: customize healthcare services, optimize resource allocation, and create financial aid programs.

Firstly, develop customized healthcare services tailored to the specific needs of each patient cluster identified in the analysis. The organization can design interventions and programs that address the healthcare priorities within each cluster. Importantly, consider their demographic and financial challenges.

Moreover, allocate resources efficiently by targeting interventions and services to patient clusters with the most significant healthcare needs and potential for impact. The organization can prioritize resource allocation based on healthcare utilization patterns and patients' financial circumstances within each cluster.

Lastly, develop targeted financial assistance programs to support patients facing financial barriers to healthcare access. The organization can tailor financial assistance initiatives to the unique needs and circumstances of patient clusters identified in the analysis. This ensures that resources are allocated effectively to those who need them most.