D212 – Data Mining II

WGU M.S. Data Analytics

Lyssa Kline

March 3, 2025

A, Part 1: Research Question

A1, Research Question and Technique

The research question in scope for this analysis is " Can we segment patients based on their medical spending patterns and health-related behaviors?" This question is valuable for hospitals, insurance companies, and healthcare providers to identify high-risk patient groups that require more frequent medical attention. Analyzing the factors affecting patients could help optimize healthcare costs by tailoring insurance plans or intervention programs and assist in developing targeted wellness programs to improve health outcomes.

A2, Analysis Goal

In this analysis, we will use a K-Means clustering technique. K-Means clustering is an unsupervised machine learning technique to partition data into k clusters based on feature similarity. When applied to continuous variables, K-Means aims to minimize intra-cluster variance while maximizing inter-cluster distance.

A K-means analysis will allow us to cluster patients based on financial and medical behavior, identify distinct patient groups, and use cases of the insights. The overall goal of the data analysis is to segment patients into distinct groups based on their healthcare spending and medical visit patterns using K-Means clustering to identify high-cost, moderate-cost, and low-cost patient groups for better resource allocation and personalized care management.

This goal will allow us to understand healthcare utilization while helping hospitals, insurance companies, and policymakers optimize resources and develop targeted intervention programs. Additionally, the dataset has multiple continuous variables suitable for clustering without requiring complex feature engineering.

B, Part 2: Technique Justification

B1, Clustering Technique and Outcomes

K-Means clustering technique was chosen for this analysis due to its ability to group patients based on similarities in their healthcare spending and visit behaviors. K-Means analyzes the dataset by selecting relevant continuous variables for data preprocessing, once selected you can standardize the variables to ensure equal weightage. The next step is to use the elbow method to determine the best number of clusters by analyzing the inertia. After this, a K-Means algorithm is applied by assigning each patient to a cluster based on their feature similarities. Once the model is built, you can use the results to analyze the clusters. These get split into high-cost groups, moderate-cost groups, and low-cost groups.

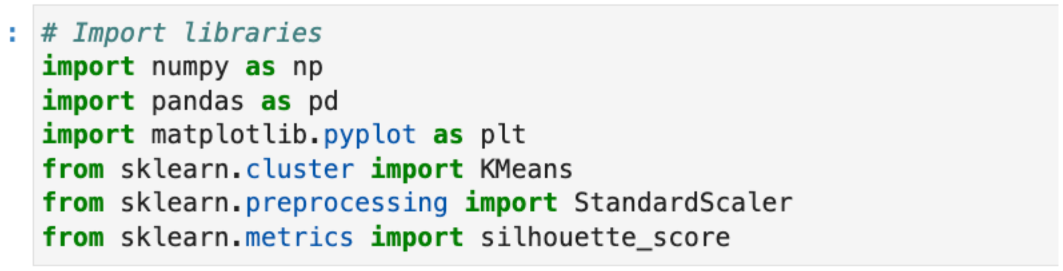
This analysis's expected outcome is to clearly segment patients into 3-5 groups based on medical expenses and visits. This segmentation of groups allows us to identify cost-heavy patient groups for targeted interventions and optimized healthcare resource allocation.

B2, Assumption

One assumption of a K-Means clustering method is that clusters are spherical and equally sized. K-Means assumes that clusters are compact, equally sized, and well separated into feature space. This may not always be true in real-world data, so preprocessing (e.g. scaling and outlier removal) helps improve results.

B3, Packages and Libraries

The following packages and libraries were chosen for this analysis.



```
: # Import libraries
  import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  from sklearn.cluster import KMeans
  from sklearn.preprocessing import StandardScaler
  from sklearn.metrics import silhouette_score
```

- Pandas – used for data handling to load, clean, and structure the dataset.
- Numpy – used for numerical computation for efficient mathematical operations on arrays.
- Matplotlib.pyplot – used for data visualization to create elbow plots and scatter plots for insights.
- Sklearn.cluser.KMeans – used for the clustering model to implement the K-Means clustering algorithm.
- Sklearn.preprocessing.StandardScaler – used for data normalization to ensure all features have equal weight.
- Sklearn.metrics.silhouette_score – used for cluster evaluation to measure how well-separated the clusters are.

C, Part 3: Data Preparation

C1, Data Preprocessing Goal

One data preprocessing goal relevant to K-Means clustering is to ensure all selected continuous variables are properly scaled to prevent bias in clustering.

Since K-Means clustering is distance-based, features with larger numeric values can dominate the clustering process. The solution to this is to standardize all the continuous variables using StandardScaler(), which transforms values to have a mean of 0 and a standard deviation of 1.

C2, Variables

The following continuous variables were selected for analysis.

- TotalCharge (continuous) - represented the overall medical expenses.
- Additional_charges (continuous) – captures extra healthcare costs
- Doc_visits (continuous) – indicates the frequency of hospital visits.
- Age (continuous) – affects healthcare needs and spending patterns

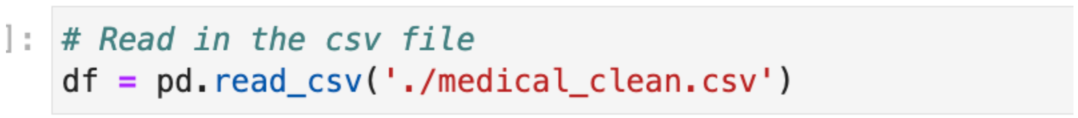
- VitD_levels (continuous) – represents nutritional/health status, which may impact medical visits.
- Full_meals_eaten (continuous) – could be linked to health conditions and medical visits.

Since K-Means only works with continuous data, categorical variables won't be included in clustering but may help interpret results post-analysis.

C3, Preparation Steps

The medical dataset was selected for use during this analysis. To ensure that the data is clean and ready to use, several steps must be followed.

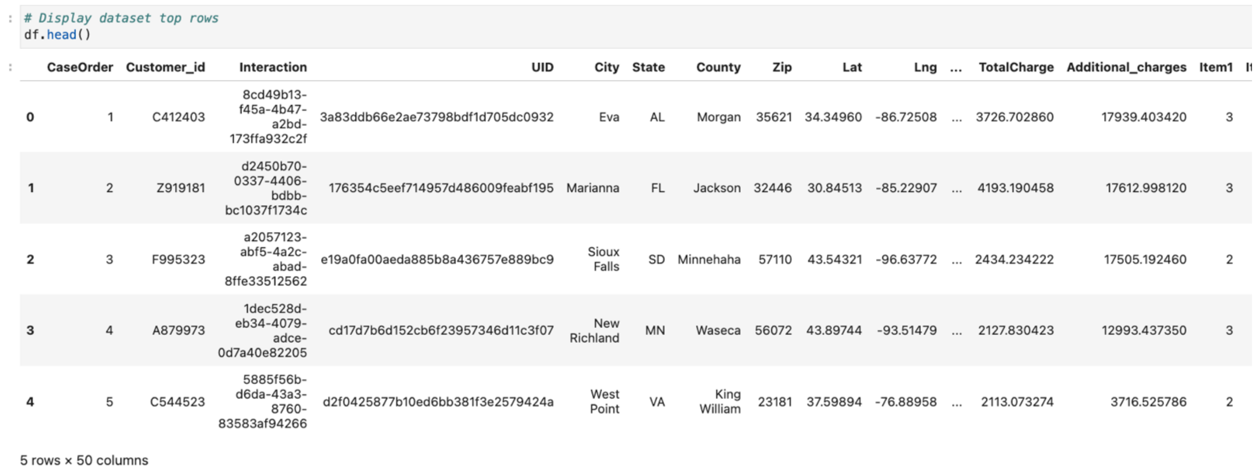
The first step is to load the dataset into the workbook for analysis. A read_csv function was called to do this.



```
]: # Read in the csv file
   df = pd.read_csv('./medical_clean.csv')
```

Once the dataset is loaded into the workbook, you can begin to evaluate and clean the data structure. During this step, there are a few functions that can be called which allow you to examine the data structure. In doing so, you can begin to identify the numerical, categorical, and text columns. It is key for a K-Means analysis to remove null values, missing data, and outliers. It helps in identifying features that may need conversion or removal.

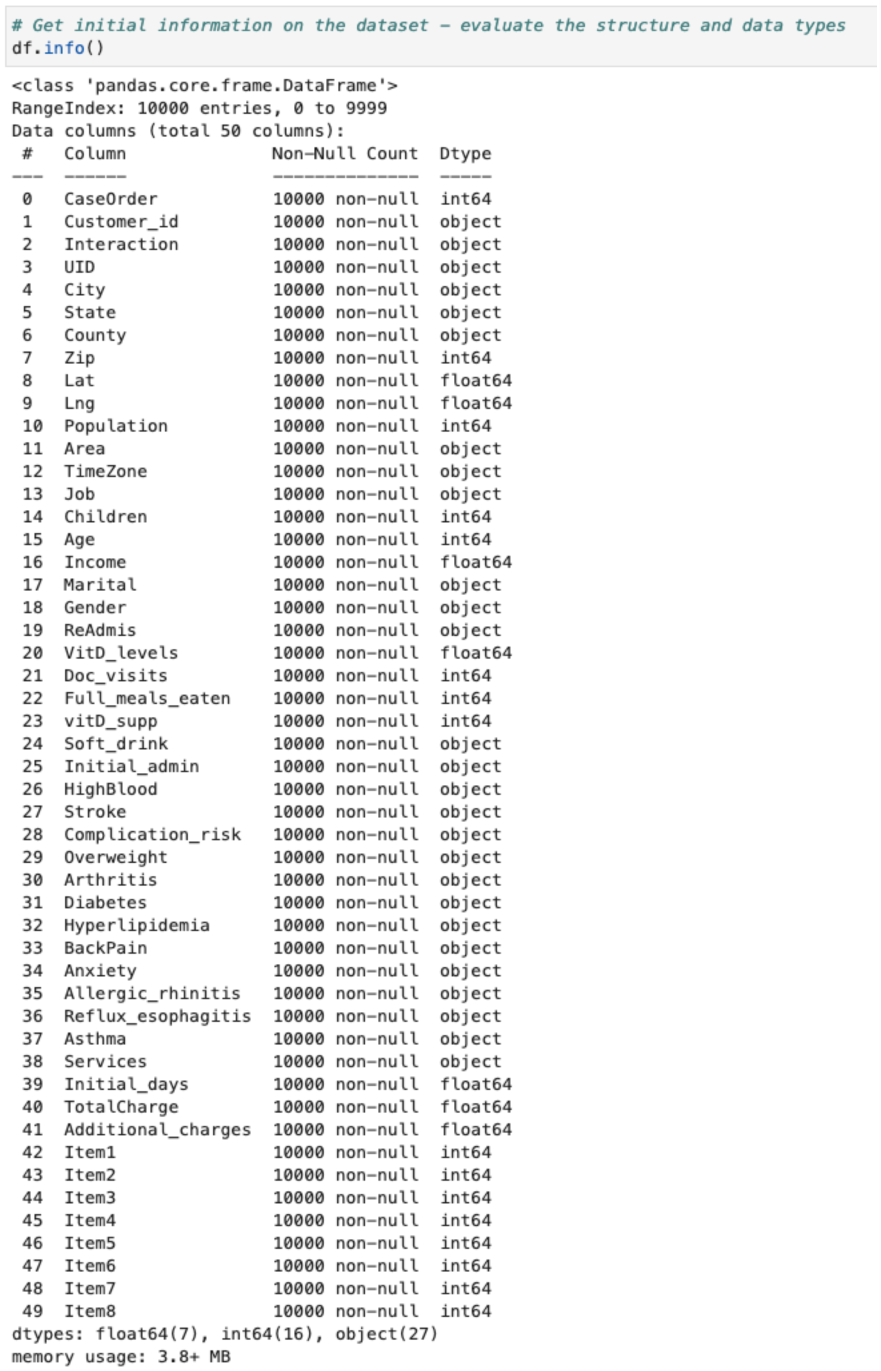
First, a .head() and a .info() were called to display an overview of the data structure, including the datatypes and field names.



```
: # Display dataset top rows
  df.head()
```

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	TotalCharge	Additional_charges	Item1	It
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	35621	34.34960	-86.72508	...	3726.702860	17939.403420	3	
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Jackson	32446	30.84513	-85.22907	...	4193.190458	17612.998120	3	
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minnehaha	57110	43.54321	-96.63772	...	2434.234222	17505.192460	2	
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	Waseca	56072	43.89744	-93.51479	...	2127.830423	12993.437350	3	
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	King William	23181	37.59894	-76.88958	...	2113.073274	3716.525786	2	

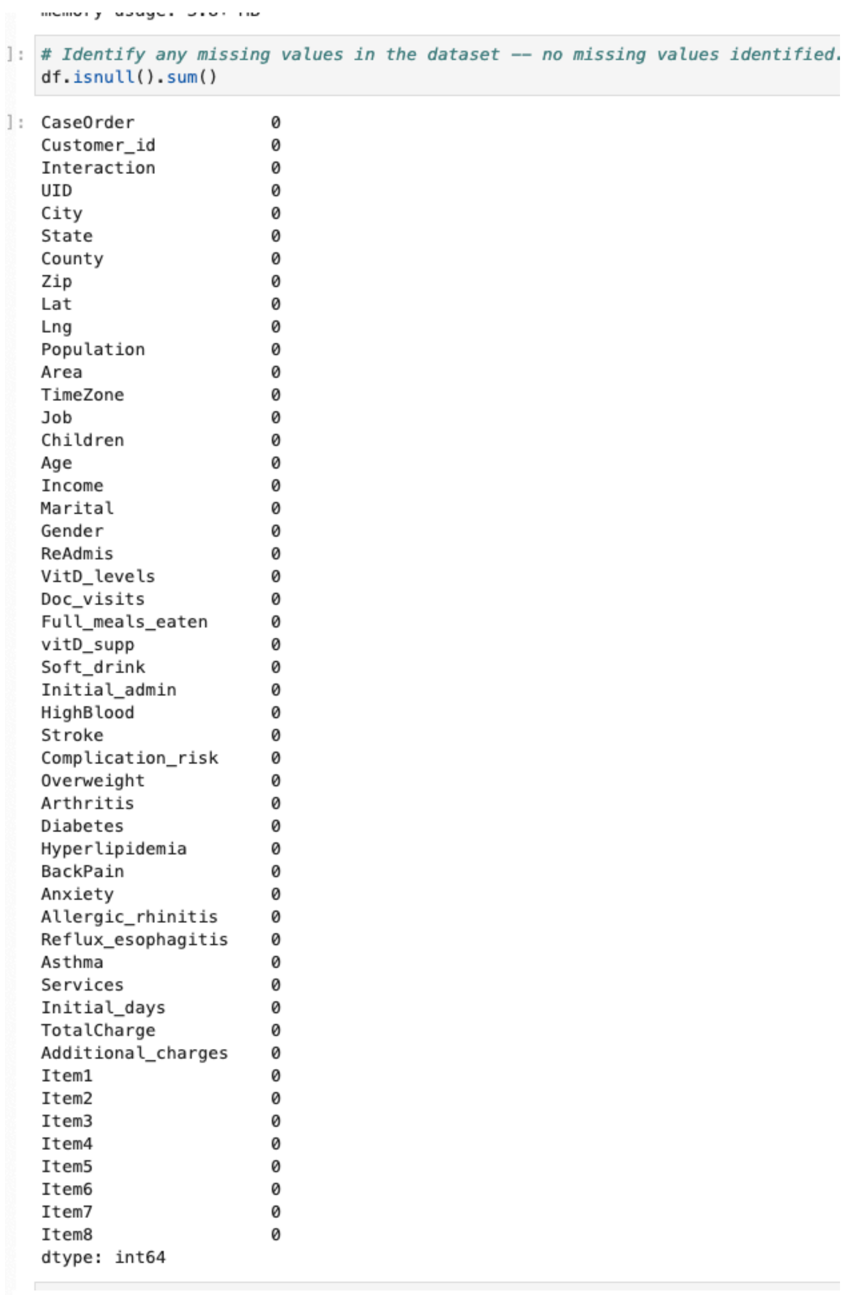
5 rows × 50 columns



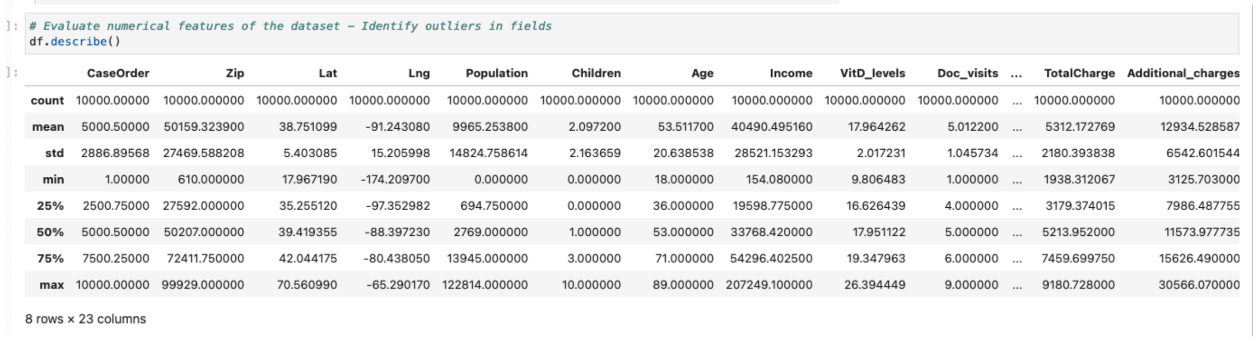
```
# Get initial information on the dataset - evaluate the structure and data types
df.info()
```

```
<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  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  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
```

Next, to identify missing values, nulls, and outliers, isnull() can be called to determine if there are any null or missing values. Additionally, .describe() can be called to identify the outliers. The output of this command indicated that there were no null values inside of the fields, as well as no outliers that stood out.



```
]: # Identify any missing values in the dataset -- no missing values identified.
   df.isnull().sum()

]: CaseOrder             0
   Customer_id           0
   Interaction           0
   UID                   0
   City                  0
   State                 0
   County                0
   Zip                   0
   Lat                   0
   Lng                   0
   Population            0
   Area                  0
   TimeZone              0
   Job                   0
   Children              0
   Age                   0
   Income                0
   Marital               0
   Gender                0
   ReAdmis               0
   VitD_levels           0
   Doc_visits            0
   Full_meals_eaten      0
   vitD_supp             0
   Soft_drink            0
   Initial_admin         0
   HighBlood             0
   Stroke                0
   Complication_risk     0
   Overweight            0
   Arthritis             0
   Diabetes              0
   Hyperlipidemia        0
   BackPain              0
   Anxiety               0
   Allergic_rhinitis     0
   Reflux_esophagitis    0
   Asthma                0
   Services              0
   Initial_days          0
   TotalCharge           0
   Additional_charges    0
   Item1                 0
   Item2                 0
   Item3                 0
   Item4                 0
   Item5                 0
   Item6                 0
   Item7                 0
   Item8                 0
   dtype: int64
```



```
]: # Evaluate numerical features of the dataset - Identify outliers in fields
   df.describe()
```

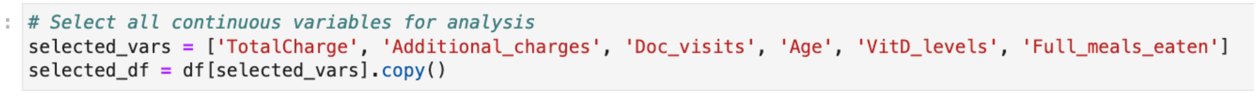
]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	VitD_levels	Doc_visits	...	TotalCharge	Additional_charges
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	...	10000.000000	10000.000000
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.097200	53.511700	40490.495160	17.964262	5.012200	...	5312.172769	12934.528587
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.163659	20.638538	28521.153293	2.017231	1.045734	...	2180.393838	6542.601544
min	1.00000	610.000000	17.967190	-174.209700	0.000000	0.000000	18.000000	154.080000	9.806483	1.000000	...	1938.312067	3125.703000
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	36.000000	19598.775000	16.626439	4.000000	...	3179.374015	7986.487755
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53.000000	33768.420000	17.951122	5.000000	...	5213.952000	11573.977735
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71.000000	54296.402500	19.347963	6.000000	...	7459.699750	15626.490000
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89.000000	207249.100000	26.394449	9.000000	...	9180.728000	30566.070000

8 rows × 23 columns

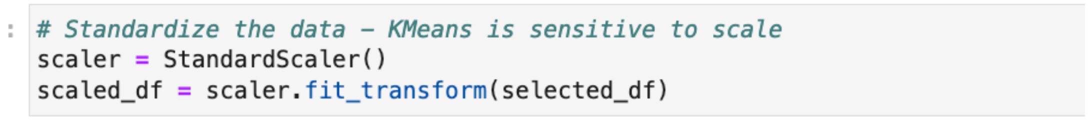
Once the dataset has been assessed and cleaned for inconsistencies, we can begin to select features for the analysis.

An initial list of all variables selected for analysis was inputted into a 'selected_df' data frame; this includes continuous values deemed adequate for the analysis. These variables are described and classified in the above step.

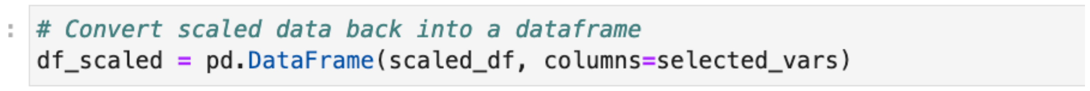


```
# Select all continuous variables for analysis
selected_vars = ['TotalCharge', 'Additional_charges', 'Doc_visits', 'Age', 'VitD_levels', 'Full_meals_eaten']
selected_df = df[selected_vars].copy()
```

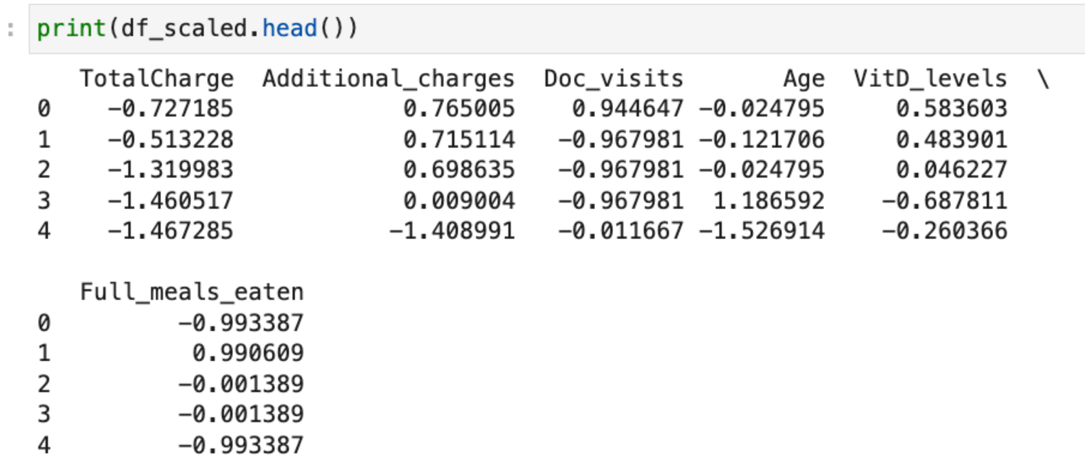
Once selected the variables need to be standardized, standardization is a scaling technique that transforms continuous variables to have a Mean = 0 and a Standard Deviation = 1. Standardization is important for K-Means because K-Means measures similarity using distances between points. Features with larger ranges will dominate the clustering process. Without standardization, K-Means would give more important variables to variables with larger numerical values, even if they are not the most relevant. Each variable contributes equally to clustering and leads to more meaningful and balanced cluster formation. Standardization is applied by taking the initially selected variables and applying the StandardScaler and fit_transform functions on them. This was then put back into a data frame called 'df_scaled' that can now be used for the analysis.



```
# Standardize the data - KMeans is sensitive to scale
scaler = StandardScaler()
scaled_df = scaler.fit_transform(selected_df)
```



```
# Convert scaled data back into a dataframe
df_scaled = pd.DataFrame(scaled_df, columns=selected_vars)
```



```
print(df_scaled.head())
```

```
   TotalCharge  Additional_charges  Doc_visits       Age  VitD_levels  \
0    -0.727185            0.765005    0.944647 -0.024795     0.583603
1    -0.513228            0.715114   -0.967981 -0.121706     0.483901
2    -1.319983            0.698635   -0.967981 -0.024795     0.046227
3    -1.460517            0.009004   -0.967981  1.186592    -0.687811
4    -1.467285           -1.408991   -0.011667 -1.526914    -0.260366

   Full_meals_eaten
0         -0.993387
1          0.990609
2         -0.001389
3         -0.001389
4         -0.993387
```

C4, Cleaned Dataset

A prepared dataset outputted in CSV format can be found within the attached 'prepared_medical_task1.csv' file.

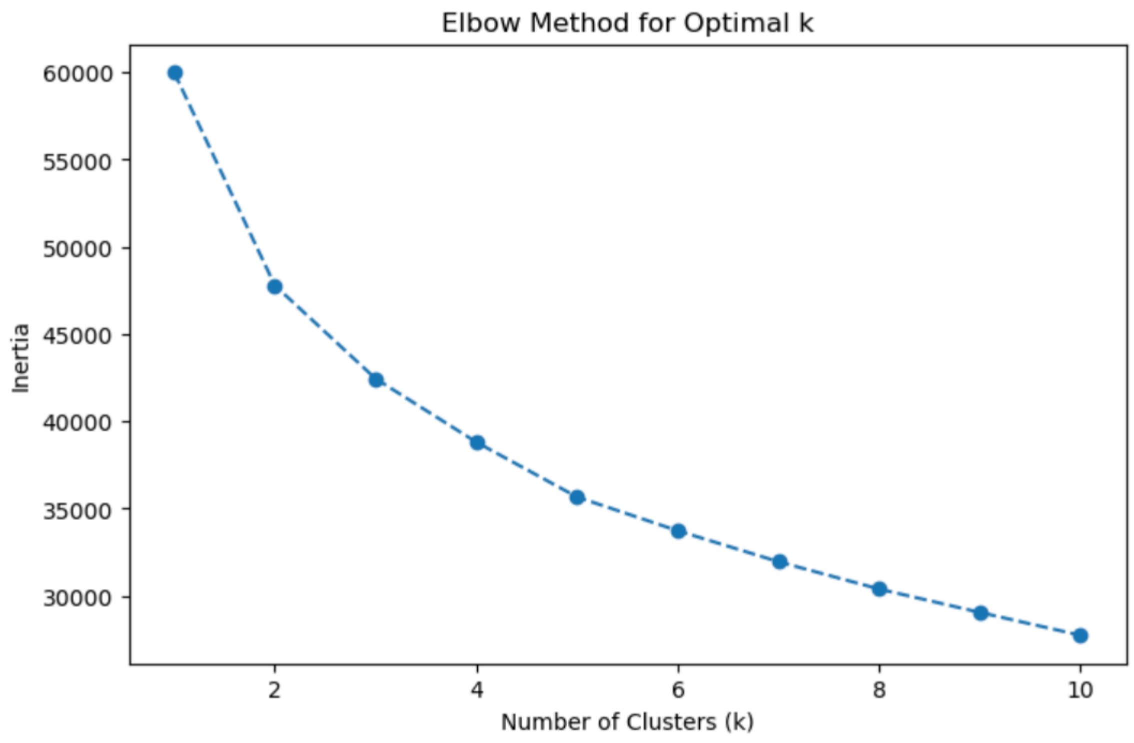
D, Part 4: Analysis

D1, Determining Clusters

The optimal number of clusters can be identified using the Elbow Method. The Elbow Method helps by plotting the inertia (within-cluster sum of squared distances) for different values of k. The plots on the Elbow Curve can be plotted and displayed using a scatter plot as shown below. The Elbow Point is where the inertia curve bends significantly. This suggests the best k for clustering. From the screenshot below we can observe a few key points.

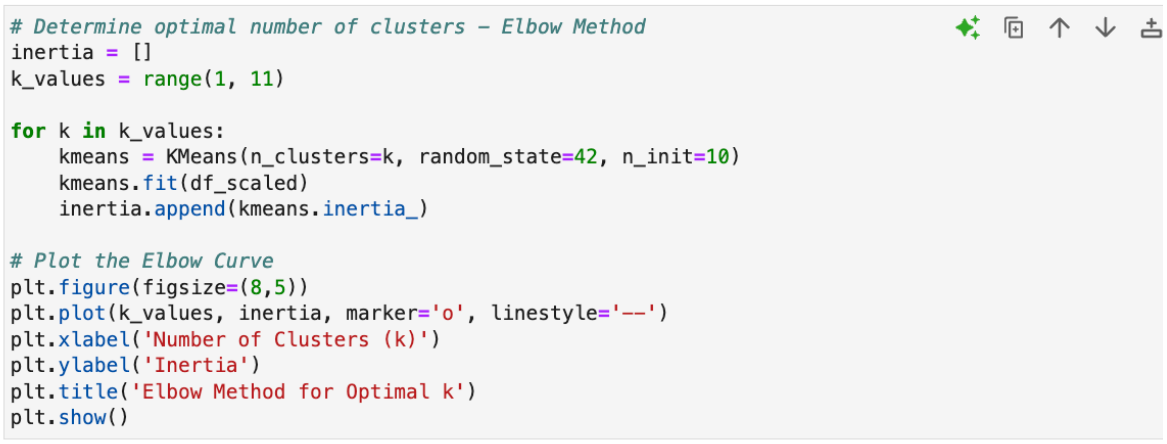
The inertia (sum of the squared distances) decreases sharply from k = 1 to k = 3, indicating significant improvement in clustering. After k = 3, the rate of decrease slows down, forming an 'elbow' at k = 3 or k = 4. Beyond k = 4, the reduction in inertia is gradual, meaning additional clusters are adding less significant value to segmentation.

The 'elbow' appears most prominent at k= 3 or k = 4. Due to this, it could be beneficial to try both k =3 and k = 4 and compare their silhouette scores to determine the best segmentation.

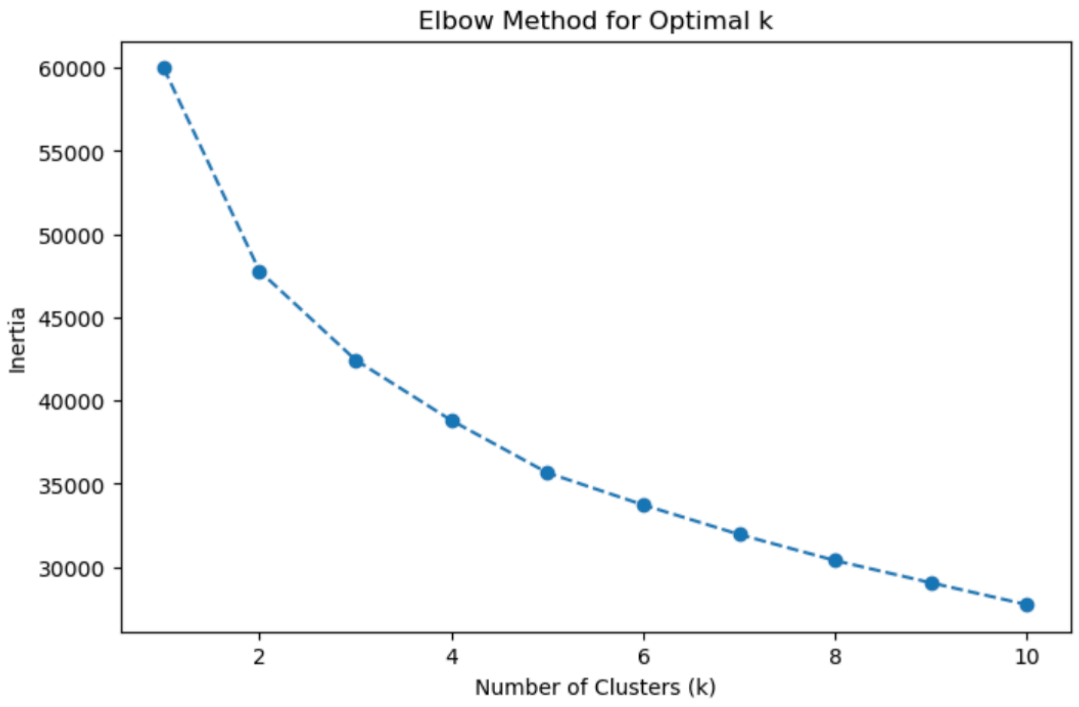


D2, Clustering Analysis Technique Code

Before applying K-Means, the first step is to find the best number of clusters (k). The Elbow Method is used by plotting the inertia. The plots on the Elbow Curve can be plotted and displayed using a scatter plot as shown below.



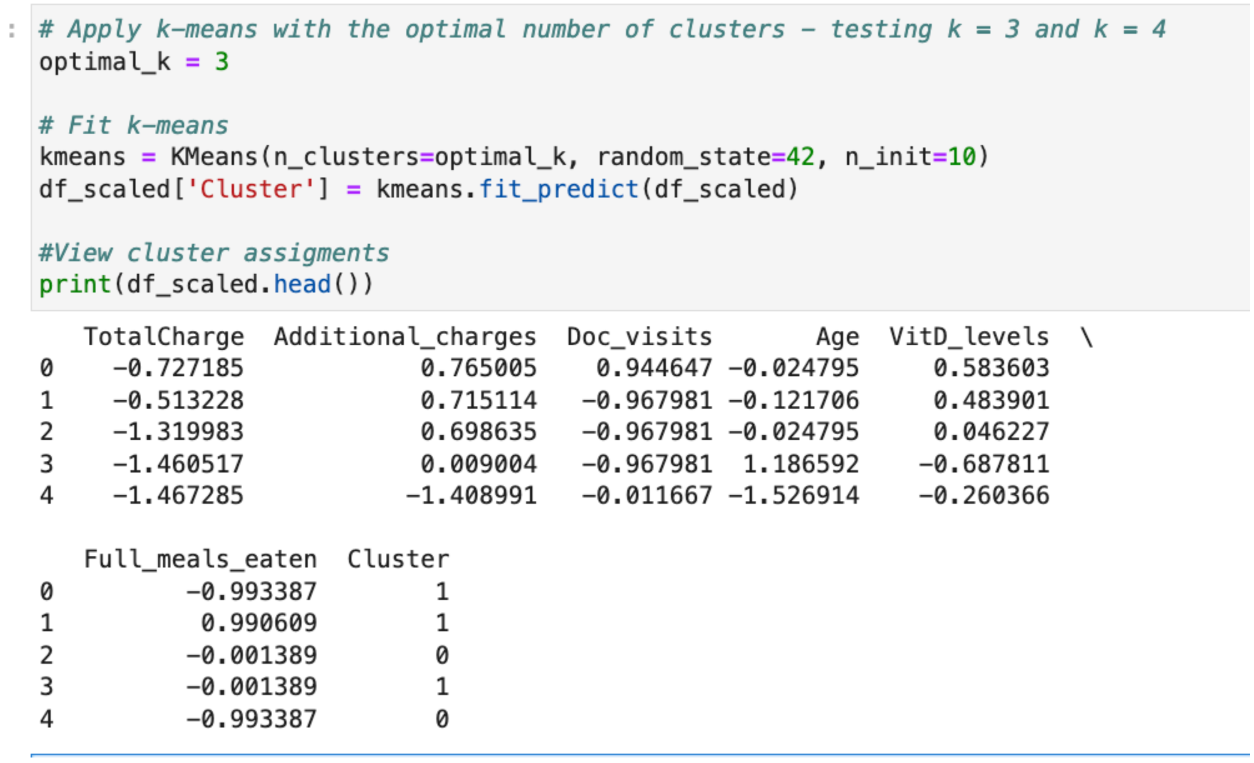
```
# Determine optimal number of clusters - Elbow Method
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Curve
plt.figure(figsize=(8,5))
plt.plot(k_values, inertia, marker='o', linestyle='--')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.show()
```



The below steps were repeated for k = 3 and k = 4 to identify which one has better defined clusters.

k = 3

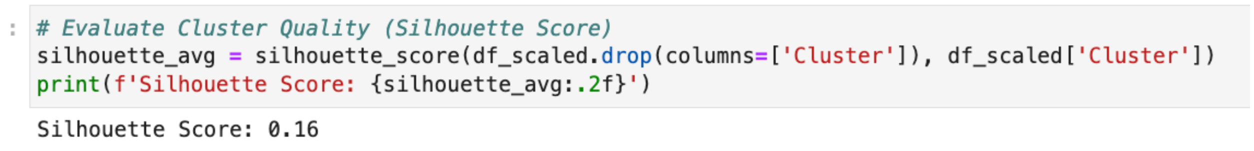
Once the optimal k has been identified, fit K-Means on df_scaled. This will apply K-Means with the optimal number of clusters and assign each record to a cluster based on feature similarity.



```
# Apply k-means with the optimal number of clusters - testing k = 3 and k = 4
optimal_k = 3

# Fit k-means
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
df_scaled['Cluster'] = kmeans.fit_predict(df_scaled)

#View cluster assigments
print(df_scaled.head())
```

```
   TotalCharge  Additional_charges  Doc_visits       Age  VitD_levels  \
0    -0.727185            0.765005    0.944647 -0.024795     0.583603
1    -0.513228            0.715114   -0.967981 -0.121706     0.483901
2    -1.319983            0.698635   -0.967981 -0.024795     0.046227
3    -1.460517            0.009004   -0.967981  1.186592    -0.687811
4    -1.467285           -1.408991   -0.011667 -1.526914    -0.260366

   Full_meals_eaten  Cluster
0         -0.993387        1
1          0.990609        1
2         -0.001389        0
3         -0.001389        1
4         -0.993387        0
```

Next, to measure how well separated the clusters are, the silhouette score is calculated. This evaluates the cluster quality. A score closer to 1 indicates well-defined clusters. The Silhouette score printed for k = 3 was .16, this is far away from 1 indicating that the clusters are not well defined.



```
# Evaluate Cluster Quality (Silhouette Score)
silhouette_avg = silhouette_score(df_scaled.drop(columns=['Cluster']), df_scaled['Cluster'])
print(f'Silhouette Score: {silhouette_avg:.2f}')
```

```
Silhouette Score: 0.16
```

k = 4

Once the optimal k has been identified, fit K-Means on df_scaled. This will apply K-Means with the optimal number of clusters and assign each record to a cluster based on feature similarity.

```
# K = 4
# Apply k-means with the optimal number of clusters
optimal_k_4 = 4

# Fit k-means
kmeans_4 = KMeans(n_clusters=optimal_k_4, random_state=42, n_init=10)
df_scaled['Cluster_4'] = kmeans_4.fit_predict(df_scaled)

#View cluster assigments
print(df_scaled.head())
```

```
   TotalCharge  Additional_charges  Doc_visits       Age  VitD_levels  \
0    -0.727185            0.765005    0.944647 -0.024795     0.583603
1    -0.513228            0.715114   -0.967981 -0.121706     0.483901
2    -1.319983            0.698635   -0.967981 -0.024795     0.046227
3    -1.460517            0.009004   -0.967981  1.186592    -0.687811
4    -1.467285           -1.408991   -0.011667 -1.526914    -0.260366

   Full_meals_eaten  Cluster  Cluster_4
0         -0.993387        2          0
1          0.990609        2          0
2         -0.001389        2          0
3         -0.001389        2          0
4         -0.993387        1          2
```

Next, to measure how well separated the clusters are, the silhouette score is calculated. This evaluates the cluster quality. A score closer to 1 indicates well-defined clusters. The Silhouette score printed for k = 3 was .22, this is far away from 1 indicating that the clusters are not well defined.

```
# K = 4
# Evaluate Cluster Quality (Silhouette Score)
silhouette_avg_4 = silhouette_score(df_scaled.drop(columns=['Cluster_4']), df_scaled['Cluster_4'])
print(f'Silhouette Score: {silhouette_avg_4:.2f}')
```

```
Silhouette Score: 0.22
```

E, Part 5: Summary and Implications

E1, Clusters Quality

The clustering quality is measured by the Silhouette Score, which indicates how well-separated the clusters are.

- Silhouette Score for k=4 is 0.22
- Silhouette Score for k=3 s 0.16

These scores suggest weak clustering, meaning that the clusters are not well-separated, and some points are likely close to decision boundaries. The low scores indicate significant overlap between clusters. Increasing the number of clusters beyond k=4 does not significantly improve separation.

This suggests that the dataset may not have strongly distinct natural groupings, or K-Means may not be the best method.

E2, Results and Implications

The dataset forms clusters, but with weak separation, meaning that patient spending and healthcare usage patterns do not have very distinct segments. The clusters likely represent slight variations in medical spending and behavior, rather than entirely different patient groups.

Various implications could arise for different parties. For example, hospital & insurance providers may not find strong segmentation for clear-cut pricing or policy decisions. For the wellness programs clusters can still be useful for grouping patients into general spending categories for targeted healthcare programs.

It can be determined from the results that additional categorical variables (e.g., medical conditions, geographic location) may help refine the clusters further.

E3, Limitation of Analysis

A major limitation is only using continuous variables for clustering. Many key healthcare-related factors (e.g., medical conditions, lifestyle choices, insurance type) are categorical but were not included in the analysis. K-Means assumes spherical clusters, but real-world patient behavior may not fit this shape, leading to poor separation.

Alternative clustering methods, such as DBSCAN or Hierarchical Clustering, might be better for capturing non-spherical patterns in healthcare data.

E4, Course of Action

Based on the weak clustering results, the organization should expand feature selection, try alternative clustering methods, use clusters for broad grouping, or further validate with business insights.

Expanding feature selection to Include categorical variables (e.g., medical history, chronic diseases, insurance plans) using one-hot encoding. This could allow you to test new features, such as patient location or frequency of preventive care, to refine segmentation.

By trying alternative clustering methods. This might uncover better relationships in patient spending behavior or allow the model to work better for non-uniform, real-world medical data.

Using clusters for broad grouping, instead of strict pricing or medical policy segmentation, may allow you to use these clusters to identify trends in patient behavior and develop generalized intervention strategies, such as predicting high-cost patients before their medical expenses rise.

Lastly, by further validating with business insights, could allow the business to cross-check cluster patterns with hospital administrators or insurers to see if they match real-world patient types and refine analysis before making strategic decisions.

Based on the results of the analysis, we can determine that while the K-Means clustering did not yield highly distinct groups, it provides a foundation for further refinement. Incorporating categorical data and alternative clustering techniques can enhance the analysis, leading to better patient segmentation and improved healthcare decision-making.

F, Part 6: Demonstration

F1, Panopto Video Recording

The video recording for this assignment includes a vocalized demonstration of all the code, the code being executed, and the results of the code being represented inside this report.
The video recording for this project can be found inside the Panopto drop box titled "D212-Task1-Kline"

Panopto video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6bfad9db-7813-4ea4-a947-b2950161cd1d

G, Web Sources

No sources or segments of third-party code were used to acquire data or to support the report.

H, Acknowledge the Sources

I acknowledge that no segments of third-party sources were directly stated or copied from the web into this report.