Lora Milam
Western Governors University
D212 Data Mining II
27 February 2023

D212 Performance Assessment Task 1

# 1 Introduction

Lora Milam Masters Data Analytics (2/19/2023) Program Mentor:d212@wgu.edu

## 1.1 Research Question

This analysis will cover a readmission dataset for a popular medical hospital. Utilizing a collection of patient characteristics, it will investigate the extent of readmission within this chain of hospitals.

This analysis will be performed by utilizing K-Means clustering. Once created, the model's accuracy will be tested by silhouette scoring.

## 1.2 Research Goal

The goal of this analysis is to determine if there are key patient characteristics that are associated with readmission.

# 2 Technique Justification

## 2.1 Explanation of Clustering Technique

In summary, the K-Means clustering algorithm is an unsupervised clustering algorithm. This means that the algorithm analyzes the data and formulates its own conclusions based on the connections and patterns it detects.

Since the algorithm does require a predetermined number of cluster centroids, it is best practice to analyze a range of cluster centroids. This can be accomplished by comparing the models' inertia. Utilizing graphical means, locating the "elbow" in the graph can aid in determining a desirable number of cluster centroids.

Once the desirable number of cluster centroids has been determined, the readmission rates of the clusters can be compared to determine if a cluster-readmission relationship exists.

## 2.2 Summary of Technique Assumption

One assumption of the K-Means clustering algorithm is that all clusters are spherical (Nagar,2020).

**2.3 Packages/Libraries List**

| Package | Justification |
|---|---|
| Numpy | Advanced mathematics |
| Pandas | Arrange and filter data |
| Seaborn | Styling of plots |
| Matplotlib.pyplot | Result visualization |
| Sklearn | Scaling and clustering implementation. Ex: StandardScaler, KMeans, Silhouette scoring |

# 3 Data Preparation

## 3.1 Data Preprocessing

One data preprocessing goal relevant to the KMeans clustering technique is creating a dataset that only includes continuous variables. This can be done by removing categorical and less meaningful variables from the dataset. Additionally, the dataset will need to scaled so that the model will not be impacted by variables with large ranges of values

## 3.2 Dataset Variables

| Variable | Type | Used in KMeans |
|---|---|---|
| Children | Continuous | Yes |
| Age | Continuous | Yes |
| Income | Continuous | Yes |
| VitD_levels | Continuous | Yes |
| Doc_visits | Continuous | Yes |
| Full_meals_eaten | Continuous | Yes |

| vitD_supp | Continuous | Yes |
|---|---|---|
| Initial_days | Continuous | Yes |
| TotalCharge | Continuous | Yes |
| Additional_charges | Continuous | Yes |
| ReAdmis | Categorical | No |

## 3.3 Steps for Analysis

```python
# Libraries
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
In [2]: ▶ # Import dataset into Pandas dataframe
          df = pd.read_csv('medical_clean_D212.csv')
          df
```

Out[2]:

| | CaseOrder | Customer_id | Interaction | UID | City | State | County | Zip | Lat | Lng | ... | TotalCha |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | C412403 | 8cd49b13-f45a-4b47-a2bd-173ffa932c2f | 3a83ddb66e2ae73798bdf1d705dc0932 | Eva | AL | Morgan | 35621 | 34.34960 | -86.72508 | ... | 3726.702 |
| 1 | 2 | Z919181 | d2450b70-0337-4406-bdbb-bc1037f1734c | 176354c5eef714957d486009feabf195 | Marianna | FL | Jackson | 32446 | 30.84513 | -85.22907 | ... | 4193.190 |
| 2 | 3 | F995323 | a2057123-abf5-4a2c-abad-8ffe33512562 | e19a0fa00aeda885b8a436757e889bc9 | Sioux Falls | SD | Minnehaha | 57110 | 43.54321 | -96.63772 | ... | 2434.234 |
| 3 | 4 | A879973 | 1dec528d-eb34-4079-adce-0d7a40e82205 | cd17d7b6d152cb6f23957346d11c3f07 | New Richland | MN | Waseca | 56072 | 43.89744 | -93.51479 | ... | 2127.830 |
| 4 | 5 | C544523 | 5885f56b-d6da-43a3-8760-83583af94266 | d2f0425877b10ed6bb381f3e2579424a | West Point | VA | King William | 23181 | 37.59894 | -76.88958 | ... | 2113.073 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | 9996 | B863060 | a25b594d-0328-486f-a9b9-0567eb0f9723 | 39184dc28cc038871912ccc4500049e5 | Norlina | NC | Warren | 27563 | 36.42886 | -78.23716 | ... | 6850.942 |
| 9996 | 9997 | P712040 | 70711574-f7b1-4a17-b15f-48c54564b70f | 3cd124ccd43147404292e883bf9ec55c | Milmay | NJ | Atlantic | 8340 | 39.43609 | -74.87302 | ... | 7741.690 |
| 9997 | 9998 | R778890 | 1d79569d-8e0f-4180-a207-d67ee4527d26 | 41b770aeee97a5b9e7f69c906a8119d7 | Southside | TN | Montgomery | 37171 | 36.36655 | -87.29988 | ... | 8276.481 |
| 9998 | 9999 | E344109 | f5a68e69-2a60-409b-a92f-ac0847b27db0 | 2bb491ef5b1beb1fed758cc8885c167a | Quinn | SD | Pennington | 57775 | 44.10354 | -102.01590 | ... | 7644.483 |
| 9999 | 10000 | I569847 | bc482c02-f8c9-4423-99de-3db5e62a18d5 | 95663a202338000abdf7e09311c2a8a1 | Coraopolis | PA | Allegheny | 15108 | 40.49998 | -80.19959 | ... | 7887.553 |

10000 rows × 50 columns

```
In [3]: ▶ # Review dataset
          # Variables within dataset
          df.columns
```

```
Out[3]: Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
               'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
               'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
               'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
               'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
               'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
               'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
               'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
               'TotalCharge', 'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4',
               'Item5', 'Item6', 'Item7', 'Item8'],
              dtype='object')
```

```
# Summary stats of variables
df.describe()
```

Out[4]:

| | CaseOrder | Zip | Lat | Lng | Population | Children | Age | Income | VitD_levels | Doc_visits | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 10000.00000 | 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 | ... |
| std | 2886.89568 | 27469.588208 | 5.403085 | 15.205998 | 14824.758614 | 2.163659 | 20.638538 | 28521.153293 | 2.017231 | 1.045734 | ... |
| min | 1.00000 | 610.000000 | 17.967190 | -174.209700 | 0.000000 | 0.000000 | 18.000000 | 154.080000 | 9.806483 | 1.000000 | ... |
| 25% | 2500.75000 | 27592.000000 | 35.255120 | -97.352982 | 694.750000 | 0.000000 | 36.000000 | 19598.775000 | 16.626439 | 4.000000 | ... |
| 50% | 5000.50000 | 50207.000000 | 39.419355 | -88.397230 | 2769.000000 | 1.000000 | 53.000000 | 33768.420000 | 17.951122 | 5.000000 | ... |
| 75% | 7500.25000 | 72411.750000 | 42.044175 | -80.438050 | 13945.000000 | 3.000000 | 71.000000 | 54296.402500 | 19.347963 | 6.000000 | ... |
| max | 10000.00000 | 99929.000000 | 70.560990 | -65.290170 | 122814.000000 | 10.000000 | 89.000000 | 207249.100000 | 26.394449 | 9.000000 | ... |

```
In [5]: # Determine if there are any missing values within dataset
        df.isnull().sum()
```

Out[5]:
```
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
```

```
In [6]:  ▶  # Review variable types
            df.dtypes
```

```
Out[6]:  CaseOrder               int64
         Customer_id            object
         Interaction            object
         UID                    object
         City                   object
         State                  object
         County                 object
         Zip                     int64
         Lat                   float64
         Lng                   float64
         Population              int64
         Area                   object
         TimeZone               object
         Job                    object
         Children                int64
         Age                     int64
         Income                float64
         Marital                object
         Gender                 object
         ReAdmis                object
         VitD_levels           float64
         Doc_visits              int64
         Full_meals_eaten        int64
         vitD_supp               int64
         Soft_drink             object
         Initial_admin          object
         HighBlood              object
         Stroke                 object
         Complication_risk      object
         Overweight             object
         Arthritis              object
         Diabetes               object
         Hyperlipidemia         object
         BackPain               object
         Anxiety                object
         Allergic_rhinitis      object
         Reflux_esophagitis     object
         Asthma                 object
         Services               object
         Initial_days          float64
         TotalCharge           float64
         Additional_charges    float64
         Item1                   int64
         Item2                   int64
         Item3                   int64
         Item4                   int64
         Item5                   int64
         Item6                   int64
         Item7                   int64
         Item8                   int64
         dtype: object
```

```
In [7]:   # Once you review the dataset
          # Remove less meaningful and categorical variables
          df=df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
                  'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Marital', 'Gender',
                  'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk',
                  'Overweight', 'Diabetes', 'Hyperlipidemia', 'BackPain',
                  'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
                  'Services', 'Arthritis', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',
                  'Item6', 'Item7', 'Item8'])
          df.columns

Out[7]:   Index(['Children', 'Age', 'Income', 'ReAdmis', 'VitD_levels', 'Doc_visits',
                 'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge',
                 'Additional_charges'],
                dtype='object')
```

```
In [8]:   # Save the ReAdmis values to compare against clustering results in final analysis
          readmis_values = df['ReAdmis']
```

```
In [9]:   # Now drop ReAdmis column so you only have continous variables
          df.drop(columns = ['ReAdmis'], inplace = True)
          df.columns

Out[9]:   Index(['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
                 'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge',
                 'Additional_charges'],
                dtype='object')
```

```python
# Determine any outliers or discrepancies by reviewing univariate and bivariate graphs
# Univariate
# Ouliers seem to be within reason
fig, axes = plt.subplots(ncols=len(df.columns))

# Create the boxplot with Seaborn
for column, axis in zip(df.columns, axes):
    sns.boxplot(data=df[column], ax=axis)
    axis.set_title(column)

# Show the plot
plt.tight_layout()
plt.show()
```

```
C:\Users\Mel Milam\AppData\Local\Temp\ipykernel_14244\3980962426.py:12: UserWarning: Tig
cannot make axes width small enough to accommodate all axes decorations
  plt.tight_layout()
```
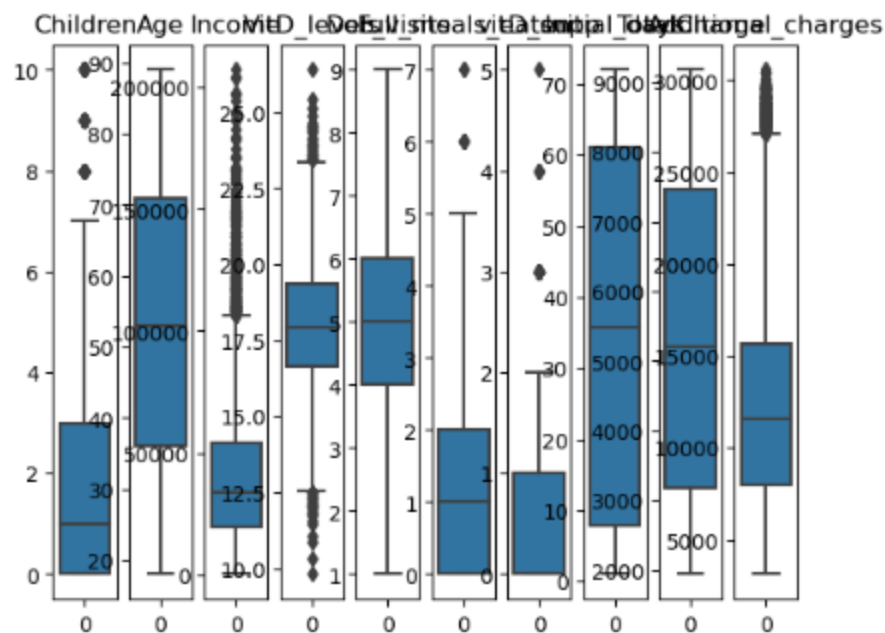
```
In [11]:  ▶  # Bivariate
              sns.pairplot(data=df)
              plt.show()
```



```
In [12]:  ▶  # Scale dataset
              ss = StandardScaler()
              ss.fit(df)
              ss_data_array = ss.transform(df)
              ss_data = pd.DataFrame(ss_data_array, columns = df.columns)
              ss_data.head()
```

Out[12]:

|   | Children | Age | Income | VitD_levels | Doc_visits | Full_meals_eaten | vitD_supp | Initial_days | TotalCharge | Additional_charges |
|---|----------|-----|--------|-------------|------------|------------------|-----------|--------------|-------------|--------------------|
| 0 | -0.507129 | -0.024795 | 1.615914 | 0.583603 | 0.944647 | -0.993387 | -0.634713 | -0.907310 | -0.727185 | 0.765005 |
| 1 | 0.417277 | -0.121706 | 0.221443 | 0.483901 | -0.967981 | 0.990609 | 0.956445 | -0.734595 | -0.513228 | 0.715114 |
| 2 | 0.417277 | -0.024795 | -0.915870 | 0.046227 | -0.967981 | -0.001389 | -0.634713 | -1.128292 | -1.319983 | 0.698635 |
| 3 | -0.969332 | 1.186592 | -0.026263 | -0.687811 | -0.967981 | -0.001389 | -0.634713 | -1.244503 | -1.460517 | 0.009004 |
| 4 | -0.507129 | -1.526914 | -1.377325 | -0.260366 | -0.011667 | -0.993387 | 2.547602 | -1.261991 | -1.467285 | -1.408991 |

### 3.4 Cleaned Dataset

```
In [13]:  ▶  # Save prepared dataset for further analysis
             ss_data.to_csv('D212_Part1_Scaled_Data.csv', index = False)
             readmis_values.to_csv('D212_Part1_ReAdmis_Data.csv', index = False)
```

# 4 Analysis

## 4.1 Output and Intermediate Calculations

```
In [14]:  ▶  # Open prepared dataset
             df_scaled = pd.read_csv('D212_Part1_Scaled_Data.csv')
             df_readmis = pd.read_csv('D212_Part1_ReAdmis_Data.csv')
```

```
In [15]:  ▶  df_scaled.head()
```

Out[15]:

|   | Children | Age | Income | VitD_levels | Doc_visits | Full_meals_eaten | vitD_supp | Initial_days | TotalCharge | Additional_charges |
|---|----------|-----|--------|-------------|------------|------------------|-----------|--------------|-------------|--------------------|
| 0 | -0.507129 | -0.024795 | 1.615914 | 0.583603 | 0.944647 | -0.993387 | -0.634713 | -0.907310 | -0.727185 | 0.765005 |
| 1 | 0.417277 | -0.121706 | 0.221443 | 0.483901 | -0.967981 | 0.990609 | 0.956445 | -0.734595 | -0.513228 | 0.715114 |
| 2 | 0.417277 | -0.024795 | -0.915870 | 0.046227 | -0.967981 | -0.001389 | -0.634713 | -1.128292 | -1.319983 | 0.698635 |
| 3 | -0.969332 | 1.186592 | -0.026263 | -0.687811 | -0.967981 | -0.001389 | -0.634713 | -1.244503 | -1.460517 | 0.009004 |
| 4 | -0.507129 | -1.526914 | -1.377325 | -0.260366 | -0.011667 | -0.993387 | 2.547602 | -1.261991 | -1.467285 | -1.408991 |

```
In [16]:  ▶  df_readmis.head()
```

Out[16]:

|   | ReAdmis |
|---|---------|
| 0 | No |
| 1 | No |
| 2 | No |
| 3 | No |
| 4 | No |

In [17]: ▶
```python
# Utilize the graphical "elbow" to determine an appropriate number of clusters
ks = range(1,10)
silhouette_scores = []
inertias = []
for k in ks:
    model = KMeans(n_clusters = k)
    model.fit(df_scaled)
    inertias.append(model.inertia_)
plt.plot(ks, inertias, '-o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.xticks(ks)
plt.show()
```
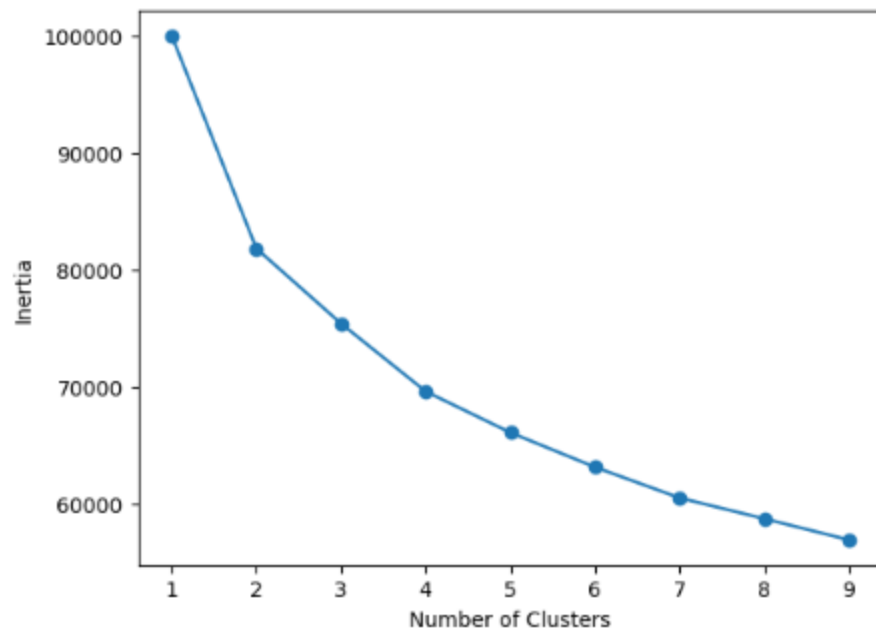
In [18]: 

```python
# Analyze Silhoutte Scores for 2+ Clusters
silhouette_scores = []
for i in range(2, 11):
    model = KMeans(n_clusters = i)
    model.fit(df_scaled)
    score = silhouette_score(df_scaled, model.labels_, metric = 'euclidean')
    silhouette_scores.append(score)
plt.plot(range(2,11), silhouette_scores, '-o')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.xticks(range(2,11))
plt.show()
```
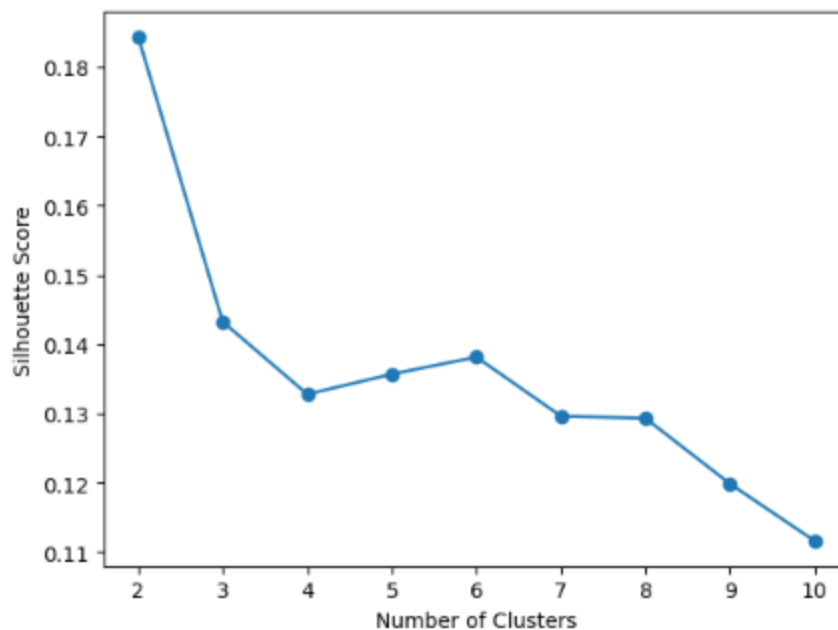
```
In [19]:  ▶  # The most accurate number of clusters is 2
             kmeans = KMeans(n_clusters=2)
             kmeans.fit(df_scaled)
             silhouette_score = silhouette_score(df_scaled, kmeans.labels_, metric = 'euclidean')
```

```
In [20]:  ▶  # Add the ReAdmis column
             df_scaled['readmis'] = df_readmis['ReAdmis'].eq('Yes').mul(1)
             df_scaled.head()
```

Out[20]:

| | Children | Age | Income | VitD_levels | Doc_visits | Full_meals_eaten | vitD_supp | Initial_days | TotalCharge | Additional_charges | readmis |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.507129 | -0.024795 | 1.615914 | 0.583603 | 0.944647 | -0.993387 | -0.634713 | -0.907310 | -0.727185 | 0.765005 | 0 |
| 1 | 0.417277 | -0.121706 | 0.221443 | 0.483901 | -0.967981 | 0.990609 | 0.956445 | -0.734595 | -0.513228 | 0.715114 | 0 |
| 2 | 0.417277 | -0.024795 | -0.915870 | 0.046227 | -0.967981 | -0.001389 | -0.634713 | -1.128292 | -1.319983 | 0.698635 | 0 |
| 3 | -0.969332 | 1.186592 | -0.026263 | -0.687811 | -0.967981 | -0.001389 | -0.634713 | -1.244503 | -1.460517 | 0.009004 | 0 |
| 4 | -0.507129 | -1.526914 | -1.377325 | -0.260366 | -0.011667 | -0.993387 | 2.547602 | -1.261991 | -1.467285 | -1.408991 | 0 |

```
In [21]:  ▶  df_scaled['label'] = kmeans.labels_
             df_scaled.columns
```

Out[21]:  Index(['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
                'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge',
                'Additional_charges', 'readmis', 'label'],
               dtype='object')

```
In [22]:  ▶  print("Characteristics of the model:")
             print(kmeans.n_features_in_,' features')
             print('Labels: ',set(df_scaled['label']))
             print(len(df_scaled['label']),' observations')
             print('Inertia value for KMean analysis: ', kmeans.inertia_)
             print('Silhouette Score for KMean analysis: ', silhouette_score)
```

```
             Characteristics of the model:
             10  features
             Labels:  {0, 1}
             10000  observations
             Inertia value for KMean analysis:  81820.47911665741
             Silhouette Score for KMean analysis:  0.18433995873995446
```

```
In [23]:  ▶  # Compare ReAdmis rates by Cluster
             cluster_0_readmis_rate = (df_scaled[df_scaled['label'] == 0]['readmis']).sum() / (df_scaled[df_scaled['label'] == 0]['readm
             cluster_1_readmis_rate = (df_scaled[df_scaled['label'] == 1]['readmis']).sum() / (df_scaled[df_scaled['label'] == 1]['readm

             print("Cluster 0 readmission rate: ", cluster_0_readmis_rate)
             print("Cluster 1 readmission rate: ", cluster_1_readmis_rate)
```

```
             Cluster 0 readmission rate:  0.7338
             Cluster 1 readmission rate:  0.0
```

# 5 Data Summary and Implications

## 5.1 Accuracy of Clustering Technique

The accuracy of the model corresponds to a silhouette score of ~.18. This is a very low confidence of accuracy.

## 5.2 Results and Implications

With this specific set of variables, two clusters would be most accurate. Utilizing two primary clusters, it was found that patients in Cluster 0 were more likely to be readmitted.

Perhaps with more data points and further testing, a silhouette score closer to 1 and higher accuracy confidence can be produced. When this is achieved, the characteristics associated with high readmission can be recorded.

**5.3 Limitations**

A limitation of the current analysis is that k-means cannot utilize categorical variables. These other variables might be able to better cluster customers into categories.

**5.4 Course of Action**

A recommended course of action from the results of this analysis would be to:

1. Collect more data points
2. Test new dataset, comparing cluster sizes until high accuracy and a silhouette score close to 1 is achieved
3. Determine new clusters' readmission rates
4. Create a plan to target patients of cluster with high readmission rates to decrease readmission rates

Alternatively,

1. Reanalyse attributes to determine clustering with higher accuracy and a silhouette score closer to 1. This can be addressed by possibly converting categorical into numerical variables which would provide more insight concerning patients' characteristics.


# 6 Supporting Documentation

### 6.1 Video

This can be found within the attached file 'Panopto Recording'.

### 6.2 Sources

Nagar, A. (2020, January 26). K-means clustering‐everything you need to know. Medium. Retrieved February 28, 2023, from https://medium.com/analytics-vidhya/k-means-clustering-everything-you-need-to-know-175dd01766d5

Western Governors University. (n.d.). D212 Data Mining II. Salt Lake City.