Task 2:

Clustering Techniques

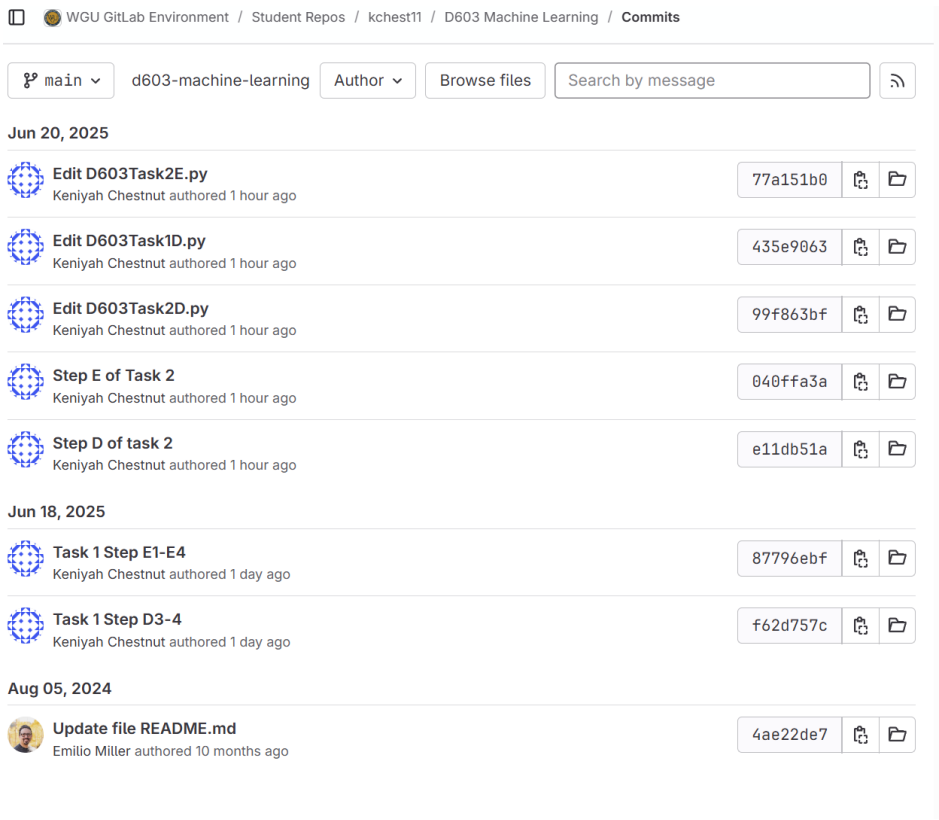
Keniyah Chestnut

Machine Learning — D603

SID: 012601305

A: GITLAB REPOSITORY

I created a subgroup and project in GitLab using the provided link. I cloned the repo to my computer, worked in it locally, and pushed my changes as I completed each part of the task.



B1: PROPOSAL OF QUESTION

Can patients be meaningfully grouped based on their demographic and medical history to identify patterns in healthcare needs and behaviors? What attributes define these clusters, and how can they inform patient care strategies?

B2: DEFINED GOAL

The goal of this analysis is to use k-means clustering to segment patients by shared characteristics. This segmentation aims to uncover underlying patterns in the data that can guide targeted healthcare strategies, improve patient outcomes, and optimize hospital resource allocation.

C1: EXPLANATION OF CLUSTERING TECHNIQUE

K-means clustering is a machine learning algorithm that divides data into k distinct, non-overlapping groups based on similarity. Each group, or cluster, contains patients whose attributes are more similar to each other than to those in other clusters. The goal is to minimize intra-cluster variance while maximizing inter-cluster differences. The expected outcome is a set of well-defined patient clusters that reveal common demographic or medical patterns.

C2: SUMMARY OF TECHNIQUE ASSUMPTION

K-means assumes that clusters are spherical and of similar size. It also expects variables to be scaled similarly to prevent features with larger magnitudes from dominating the analysis. The technique works best when the data is continuous and free from extreme outliers.

C3: PACKAGES OR LIBRARIES LIST

- pandas and numpy: For data manipulation and numeric operations.
- sklearn.preprocessing.StandardScaler: To standardize features before clustering.
- sklearn.cluster.KMeans: To perform the clustering.
- sklearn.decomposition.PCA: To reduce dimensionality for visualizing the clusters.
- matplotlib.pyplot and seaborn: For plotting the elbow graph and cluster visualization.

D1: DATA PREPROCESSING

Before beginning the analysis, I cleaned the dataset to ensure it was ready for clustering.

```
print(data.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
```

First, I checked for and removed any missing values to avoid errors during processing. I also stripped whitespace from all column headers to prevent issues when referencing column names in the code.

```
# Strip extra spaces from column names
data.columns = data.columns.str.strip()

# Display the first few rows to confirm it's loaded correctly
data.head()
```

[28]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	TotalCharge	Additional_cha
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	35621	34.34960	-86.72508	...	3726.702860	17939.40
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Jackson	32446	30.84513	-85.22907	...	4193.190458	17612.99
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minnehaha	57110	43.54321	-96.63772	...	2434.234222	17505.19
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	Waseca	56072	43.89744	-93.51479	...	2127.830423	12993.43
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	King William	23181	37.59894	-76.88958	...	2113.073274	3716.52

5 rows × 50 columns

```
data.describe()
```

[30]:

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

0 rows × 23 columns

Next, I reviewed the dataset and removed columns unrelated to patient demographics or medical history. Since k-means clustering requires all input features to be numeric, I encoded all categorical variables using appropriate techniques. This step ensured the data was both clean and in the correct format for the clustering algorithm.

D2: DATASET VARIABLES

To focus the clustering analysis on patient demographics and medical history, I removed any columns that were not relevant to these aspects. The remaining variables were then categorized as either categorical or continuous based on their data type and role in the analysis.

While some numerical variables could be interpreted as categorical, I opted to treat all purely numeric variables as continuous since they do not require encoding and can be used directly in the clustering algorithm.

Categorical Variables (non-numerical):

- Marital
- Gender
- ReAdmis

- Soft_drink
- Initial_admin
- HighBlood
- Stroke
- Overweight
- Arthritis
- Diabetes
- Hyperlipidemia
- BackPain
- Anxiety
- Allergic_rhinitis
- Reflux_esophagitis
- Asthma
- Services
- Complication_risk

Continuous Variables (numerical):

- Income
- VitD_levels
- Doc_visits
- Initial_days
- TotalCharge

- Additional_charges
- Full_meals_eaten
- vitD_supp
- Children
- Age

D3: STEPS FOR ANALYSIS

The first step in the analysis was to remove all columns that were not directly related to patient demographics or medical history. This helped focus the dataset and reduced the dimensionality of the data.

```
# D2: Drop irrelevant columns safely
drop_cols = ['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
             'County', 'Zip', 'Lat', 'Lng', 'Population', 'TimeZone', 'Job',
             'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8',
             'Weight', 'SurveyScore', 'Insurance']

# Keep only the columns that are present
existing_cols = [col for col in drop_cols if col in df.columns]
df.drop(columns=existing_cols, inplace=True)
```

Next, I encoded the categorical variables. Many of the features were already boolean, so they were simply converted into binary values (0 and 1). For the Gender column, a special encoding approach was required because the values included not just male and female, but also a 'nonbinary' option. Nominal variables with more than two categories were encoded using one-hot encoding to ensure proper numerical representation.

```
6]: # D3: Encode binary categorical variables safely
    binary_cols = ['ReAdmis', 'Gender', 'Change', 'Complete']
    binary_map = {'Yes': 1, 'No': 0, 'Female': 1, 'Male': 0}
    for col in binary_cols:
        if col in df.columns:
            df[col] = df[col].map(binary_map)


    #2. Special binary encoding just for Gender
    df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
    df['ReAdmis'] = df['ReAdmis'].map({'No': 0, 'Yes': 1})

8]: #3. One-Hot Encoding for Initial_admin and Services (nominal)
    df = pd.get_dummies(df, columns=['Initial_admin', 'Services'], drop_first=True)
```

A unique case was the Complication_risk column, which was ordinal in nature. I applied ordinal encoding here, assigning a value of 0 to ‘Low’, 1 to ‘Medium’, and 2 to ‘High’, as this preserved the natural ranking of risk levels.

```
40]: #4.Ordinal Encoding for Complication_risk
     risk_map = {'Low': 0, 'Medium': 1, 'High': 2}
     df['Complication_risk'] = df['Complication_risk'].map(risk_map)
```

Once all variables were numerical, I selected the columns necessary for clustering. Before proceeding, I applied feature scaling using StandardScaler to normalize the values. This step was

important because features like Income had much larger scales and could otherwise disproportionately influence the clustering.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
#D4
df.to_csv("cleaned_medical_data.csv", index=False)
```

D4: CLEANED DATASET

A cleaned, numeric dataset with 50 columns was created. All categorical variables were encoded, and the features were scaled. This dataset was used as the input for clustering and visualization.


```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

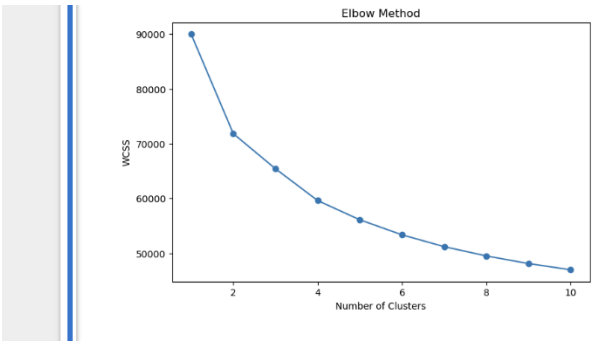
```
#D4
df.to_csv("cleaned_medical_data.csv", index=False)
```


E1: OUTPUT AND INTERMEDIATE CALCULATIONS

After scaling and validating the cleaned dataset, I used the Elbow Method to determine the optimal number of clusters. I plotted the Within-Cluster Sum of Squares (WCSS) against various values of k to visualize the point where increasing the number of clusters provided diminishing returns. The resulting graph showed a clear elbow at k = 3, indicating this was the most appropriate number of clusters for the analysis. This suggests that three clusters balance both cohesion and separation effectively, making it a strong candidate for segmenting the dataset.

```
#E1: Determine Optimal Number of Clusters - Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



To further support this decision, I calculated the Silhouette Score for each value of k between 2 and 10. The silhouette score measures how well each data point fits within its assigned cluster compared to other clusters. A higher score indicates better-defined clusters. The results showed that k = 3 had one of the higher silhouette scores, confirming that this value was a strong choice for segmenting the dataset.

These two methods together gave confidence that k = 3 would be the most appropriate number of clusters for this analysis.

[50]:
```
#Silhouette Score
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    sil_score = silhouette_score(X_scaled, labels)
    print(f"Silhouette Score for k={k}: {sil_score}")
```

```
Silhouette Score for k=2: 0.203757399809246
Silhouette Score for k=3: 0.16065584960897258
Silhouette Score for k=4: 0.15004471168205924
Silhouette Score for k=5: 0.15400320038943854
Silhouette Score for k=6: 0.1536467983386844
Silhouette Score for k=7: 0.14757377550060777
Silhouette Score for k=8: 0.13387932073414358
Silhouette Score for k=9: 0.12295285592800442
Silhouette Score for k=10: 0.11429347913446318
```

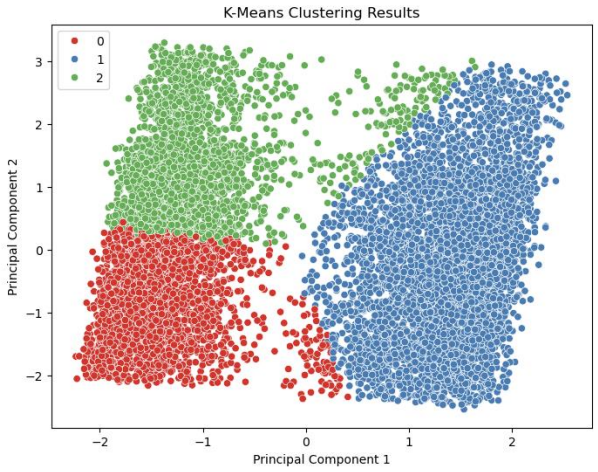
F1: QUALITY OF CLUSTERING TECHNIQUE

To visualize the clustering results, I applied Principal Component Analysis (PCA) to reduce the scaled dataset to two dimensions. This allowed me to create a scatterplot showing the distribution of patients across the three clusters identified using the K-Means algorithm. The plot clearly shows distinct clusters, although there is some minor overlap. The separation between groups indicates that the model was reasonably successful in identifying underlying patterns in the data. While not perfectly clean, the clustering suggests meaningful groupings that could reflect different patient profiles.

```
# Apply k=3 from silhouette/elbow methods
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(X_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:,0], y=X_pca[:,1], hue=labels, palette='Set1')
plt.title('K-Means Clustering Results')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```



The clustering results appear moderately successful based on the PCA scatterplot. One of the clusters is clearly defined and visually distinct from the others, which suggests that the algorithm effectively grouped similar patients together in that case. However, the other two clusters overlap slightly, which means the boundaries between them are not as well defined. This overlap could be due to the natural similarities in patient attributes or a limitation of using PCA to project high-dimensional data into two dimensions. Despite this, the clusters are still reasonably separated, and the model shows potential in identifying general groupings of patients with shared traits. Overall, while the clustering is not perfect, it provides a solid foundation for further refinement and insight.

F2: RESULTS AND IMPLICATIONS

After applying the k-means clustering model and using PCA for dimensionality reduction, the model successfully separated the patient data into three distinct clusters. The scatterplot shows that while one cluster is clearly defined, the other two share some overlapping areas. This indicates that the model found some meaningful structure in the data, but the separation is not perfect. Even so, these clusters provide a starting point for identifying patterns in patient demographics and medical history. With more investigation into what defines each cluster, healthcare providers could use these groupings to customize treatment plans or identify at-risk populations. The model allows for an initial segmentation of patients that could lead to better decision-making in terms of resource allocation and targeted care.

F3: LIMITATION

One clear limitation of this model is the number of clusters we decided to use. While k=3 seemed reasonable based on both the elbow method and silhouette scores, the scatterplot shows that one of the clusters might be split in an unusual way. It's possible that choosing 4 or even 5 clusters could lead to more meaningful groupings. Another limitation is the sensitivity of k-means to outliers and non-spherical shapes. Our data may include patient types that don't fit neatly into round clusters, which k-means isn't great at handling. On top of that, the PCA dimensionality reduction step might be hiding some of the more subtle differences in the data since we're compressing a lot of variables down into just two components for visualization. Lastly, using a large number of variables can lead to noise that weakens the quality of the clustering, so some feature selection might help improve the model.

F4: COURSE OF ACTION

The next step should be to analyze each cluster in detail to see what traits patients in each group share. This includes reviewing factors like age, diagnosis patterns, and medical charges. Reducing the number of variables used in the model may help reveal clearer cluster boundaries. It would also be helpful to test the model using a different number of clusters to compare the results. If further testing shows this model is valid, it can be used to support health care planning and delivery. For example, hospitals could use the clustering to identify groups of patients who need more preventative care or who are more likely to be readmitted. This information could improve care plans and help medical staff provide more effective and efficient treatment across patient groups.

References

All datasets and instructional materials were provided by Western Governors University (WGU) for academic use.