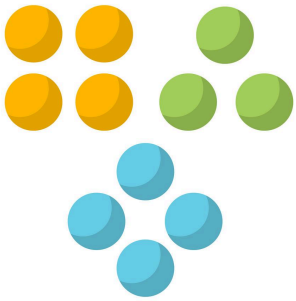


Project 1

Clustering Algorithms with MIMIC – III

IE6400 Foundation for Data Analytics Engineering

Spring 2025

Palak Tanwar

(001523508)

Table of Contents

Introduction

MIMIC (Medical Information Mart for Intensive Care) is a database that contains anonymized medical data of patients admitted to the Critical Care Units of the Beth Israel Deaconess Medical Center over the years. In this project, we will be conducting a thorough study of the MIMIC - III database which contains the de-identified medical data of over forty thousand patients, collected over the span of 2001 to 2012 from the Metavision and CarVue bedside monitors of the patients. We start by leveraging the exploratory data analysis techniques, to understand the structure of the datasets, followed by handling any missing values and using clustering algorithms such as K-Means and Hierarchical (Agglomerative) algorithms that help us to understand the data, deal with outliers and perform dimensionality reduction to keep the most important data for studying insights. In the last step, we will visualize the clustered data and study the trends and patterns. We have used Google Colab for K-Means clustering and for hierarchical clustering we have used a laptop with Intel(R) Core(™)-i9-14900 HX 2.20 GHz and 32.0 GB RAM (31.7 GB usable).

Data Preparation

In this project, we studied nineteen datasets, our initial step was to upload these datasets to the Google Colab platform, following this we started with Data Cleaning, which involved inspecting data for any missing values, duplicates and performing techniques to handle it. We start by performing an outer join on the datasets on the ‘hadm_id’ which is the unique id common to all datasets. The result of the join was the main dataframe called ‘main_df’, on which we performed data analysis throughout the project. Further, we studied the structure of the main dataframe and looked for the null values. Next, we dropped the columns where over 42% of the data was missing and 20% of the rows where any data was missing. There was no data imputation performed as healthcare data is sensitive, imputing data points can lead to incorrect analyses. We know that the clustering algorithms are sensitive to the scale of input values, hence, we separate the numerical dimensions of the dataframe and scale them to a range of [0,1] by using min-max scaling. The remaining dimensions that are categorical in nature such as 'gender', 'marital_status', 'religion' and 'ethnicity' can not go through clustering directly, they are encoded to binary values by using one hot encoding.

Why choose one hot encoding over label encoding?

Label Encoding converts categorical values into numerical labels (e.g., "Male" to 0, "Female" to 1). It is efficient in terms of memory and speed, especially for large datasets. It is suitable for ordinal data. But it comes with the biggest con of implying an ordinal relationship between

categories that may not exist. It can distort distance calculations in K-Means, leading to poor clustering.

On the other hand, One-Hot Encoding (OHE) creates binary columns for each category (e.g., "Male" to [1, 0], "Female" to [0, 1]). It helps to prevent the algorithm from assuming an ordinal relationship. It works well for nominal categorical variables (where there is no inherent ranking). It has its own cons like increased dimensionality, especially if there are many unique categories and can lead to sparsity, making clustering computationally expensive. Hence, choosing OHE for our use case makes more sense.

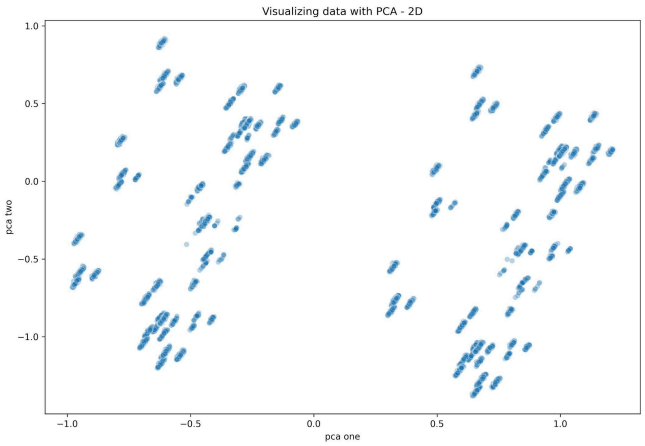
Data Visualization

Visualizing high dimensional data

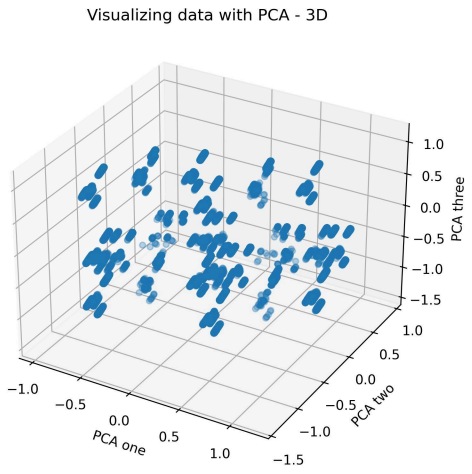
Principal Component Analysis: This is a dimension reduction technique which is used to convert an n dimension data fit into a lower dimension subspace by reducing the number of features, however, preserving the most important information.

t-SNE (t-Distributed Stochastic Neighbor Embedding): This is a nonlinear dimension reduction technique used for visualization of high dimension data into a 2-D or 3-D space, it preserves the local structure of data points.

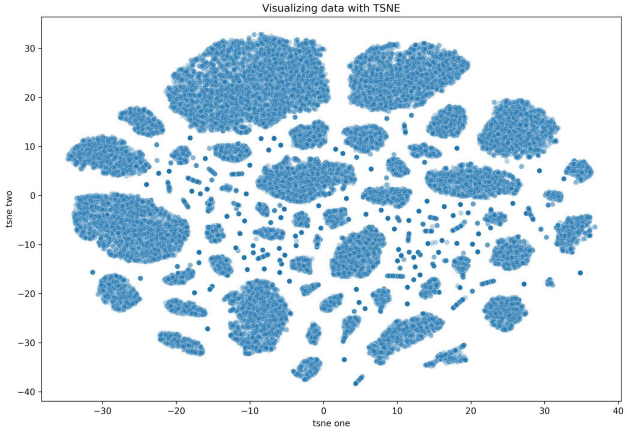
The image below shows the Principal Component Analysis of the main dataframe before applying any clustering algorithms, as we can see there aren't any clusters visible in this image as this is a two dimensional representation of over forty dimensions that our data had.



Upon visualizing the Principal Component Analysis in the three dimensional space in the image below, we observed that there may be clusters available when viewed by increasing the dimension.



Similarly, the t-SNE plot below, maps the points of the dataset such that huge groups are visible hence we can expect bigger clusters to exist in this data spread.



Clustering

K-Means clustering

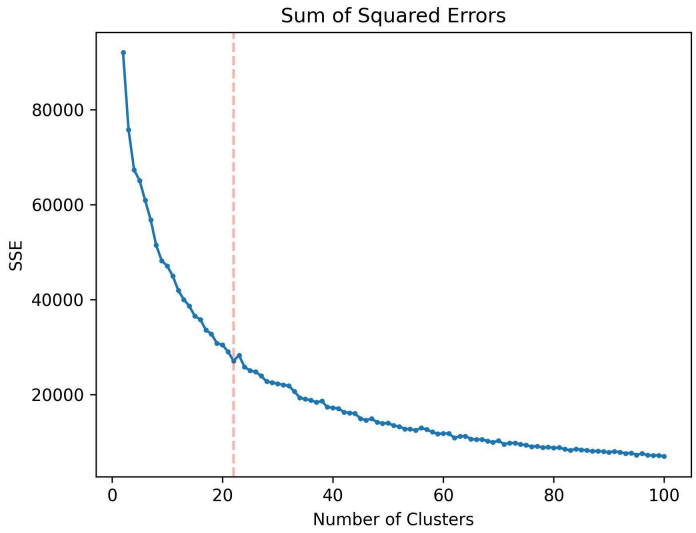
K-Means clustering is an unsupervised learning algorithm which groups data into clusters such that all the data points within those clusters have similar properties. The algorithm begins by

randomly picking some centre points known as the centroids. Next, the distance between every data point and these centroids is calculated, based on this distance, the data points are assigned to the centroid to which they are the closest, hence clusters are generated. The centroids are then moved within the clusters to a point such that it is at a similar distance to all data points within the cluster. The distance of each data point is calculated again to all the centroids and the clusters are updated. This algorithm converges when the data points stop changing their clusters.

In this project, we made a list of feature columns on which the model could be trained. For initial testing the model was looped through 10 clusters to check if an elbow point can be achieved on the line graph. As the data is quite large, no elbow was achieved and then the model was trained to loop through a value of 100 cluster points.

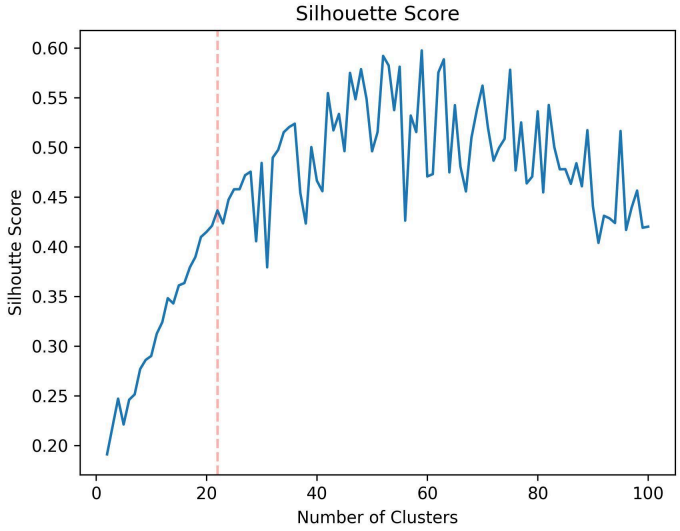
Following this, the line plot between Within Cluster Sum of Squares (WCSS) and number of clusters showed a dip at near 22 after which the clusters started increasing again. Hence we chose the number of clusters to be 22.

Optimal WCSS: 28015 (approx)

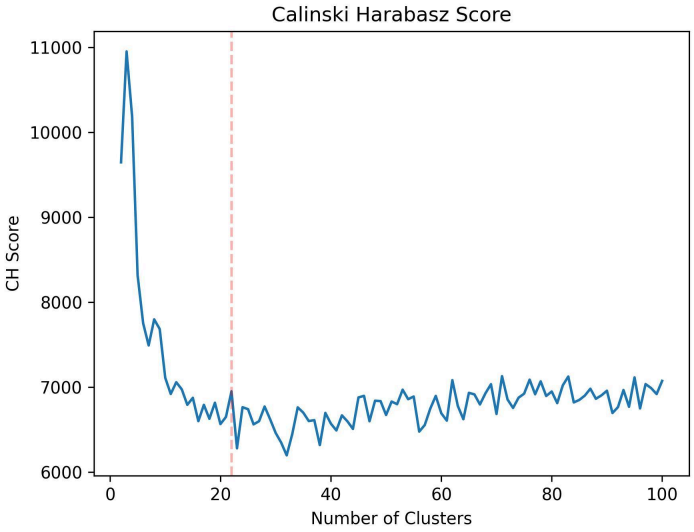


Upon plotting the silhouette score, we observed that the score gradually increases till 0.43 after which there are a lot of sporadic changes that take place.We trained the data for 22 clusters and get the following scores:

Optimal Silhouette Score: 0.43(approx)

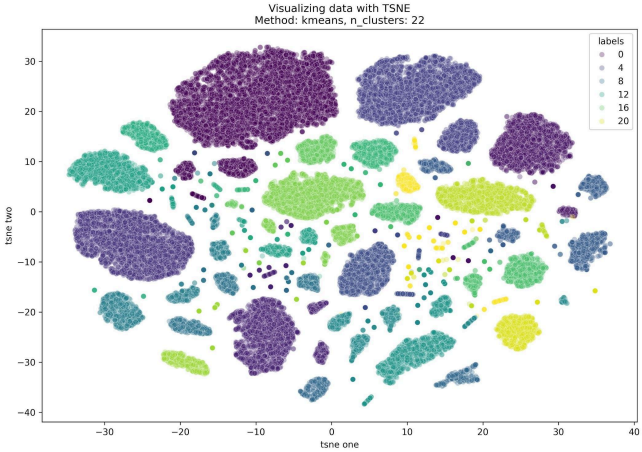


Optimal CH Ratio: 6661(approx)

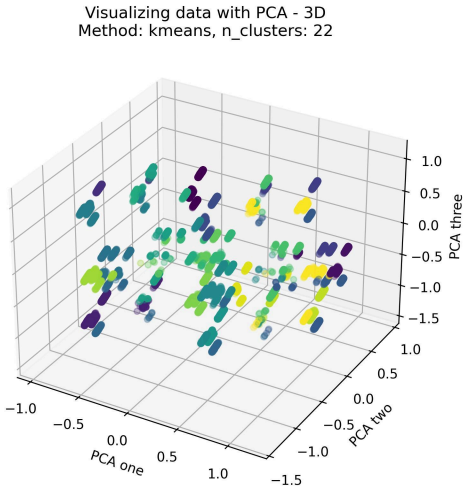


Cluster analysis

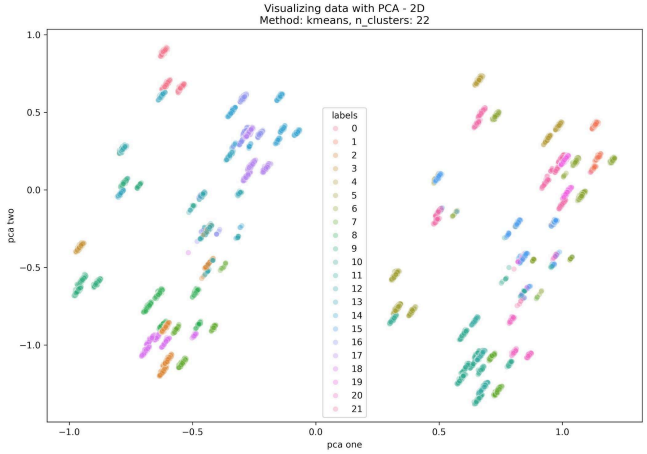
After performing the K-means clustering algorithms the below t-SNE plot is achieved with different colors representing the cluster number of data points.



The figure below represents the three dimensional Principal Component Analysis of the dataset after clustering.



The figure below shows the two dimensional Principal Component Analysis of the dataset with each datapoint representing the cluster it is a part of through its color.



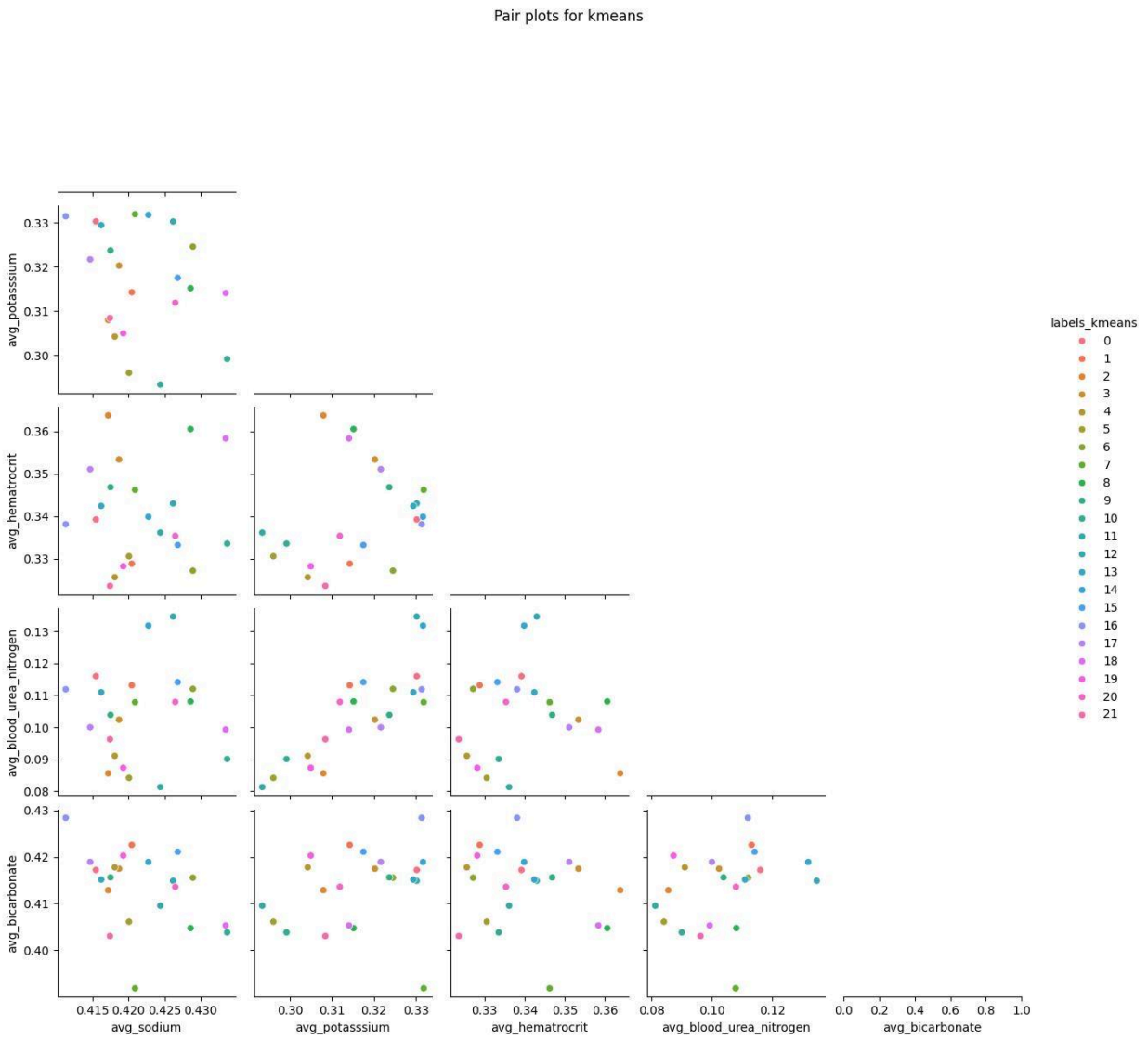
Observation

After studying the plots above, we observe that t-SNE captures the clusters better for visual representation and it can validate that the clusters have better grouping. However, in the case of PCA we can not properly visualize clusters. From both the plots we observe that the points that are in small groups are affecting the performance of the model.

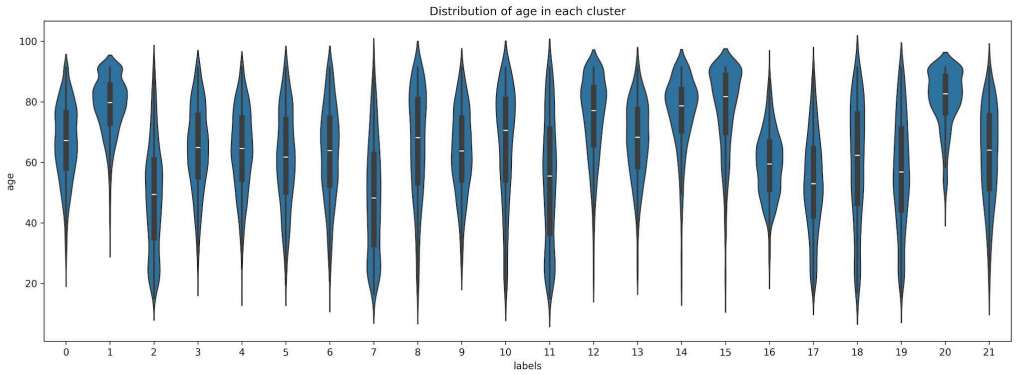
K-means interpretation

The below representation shows a pair plot of centroids of the total number of clusters formed after k-means clustering. The centroid is calculated by taking the average of all the data points within each cluster. In the below pair plots, we are consuming the features that are showing the highest variation in the centroids' in those dimensions. Example, centroids are showing highest variance in avg_potassium, avg_sodium, avg_hematrocrit. If we look at the pair plots, K-means is able to cluster points with good separation. In this example, if we look at avg_blood_urea_nitrogren and avg_hematrocrit, we are able to have centroids with high variance and minimal overlaps giving our clusters better separation.

If we compare pair plots alongside SSE, SS and CH scores, there is a lot of flexibility available to choose the optimal number of clusters. Depending on the use case, we can choose how granular we want to go.



Violin plots show the distribution of numeric data by using density plots. The below represented violin plots show the distribution of the variables age, marital_status and ethnicity in each cluster. We observed that there are more old aged patients present in the 2nd, 13th, 15th, 16th and 20th clusters when compared to other clusters. This explains that old aged patients tend to show similar vital behaviours compared to other ages, which if we think logically, there is a higher tendency of older patients to have vital related problems. On the other hand, we see other ages are mixed and don't show clear age based grouping, or trends in those groups.



Implemented K-means clustering algorithm from scratch

Custom implementation of K-means clustering algorithm is also included in the code files. After completing the project with available libraries. I implemented the k-means algorithm from scratch. Hence, tested it only with custom demo data.

Hierarchical clustering

Hierarchical clustering groups data into tree clusters. It starts by treating every data point as a single cluster and then it identifies two similar clusters and combines them into one cluster. This continues till all clusters are merged.

Training process

This clustering model was also trained on the same feature columns as in K-means. As the dataset is very large, it would crash the Google Colab Session. Hence, this clustering algorithm was executed on a laptop with Intel(R) Core(™)-i9-14900 HX 2.20 GHz and 32.0 GB RAM (31.7 GB usable). This algorithm started by considering each point as a cluster with over forty thousand data points in the dataset. Each method took around 15 minutes of training time on an average.

How to consume dendogram for optimal clusters?

We check the linkages and identify the one with the highest vertical dip. We take that distance that is represented on the y-axis and plot a horizontal line cutting the dendogram into clusters. After cutting the dendogram, we check for the number of unique clusters below the cutting line and consider that as the optimal number of clusters.

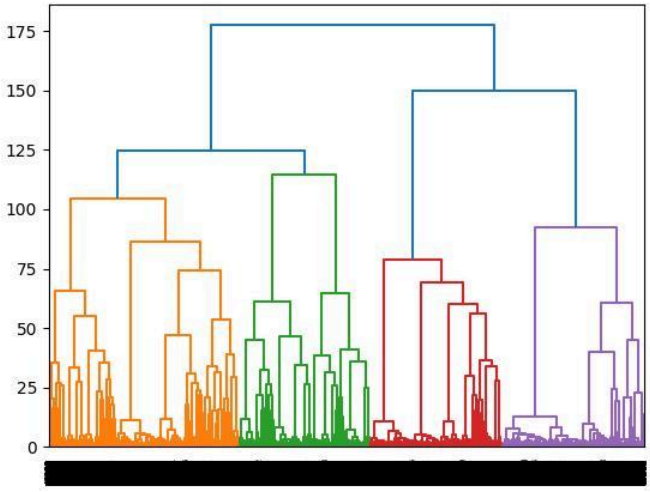
Hierarchical clustering methods

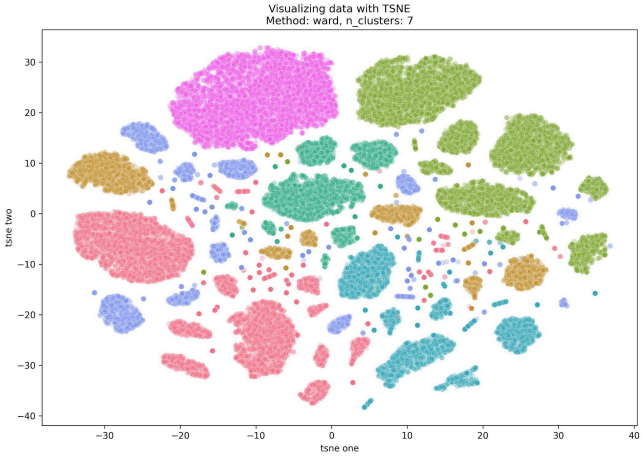
WARD LINKAGE

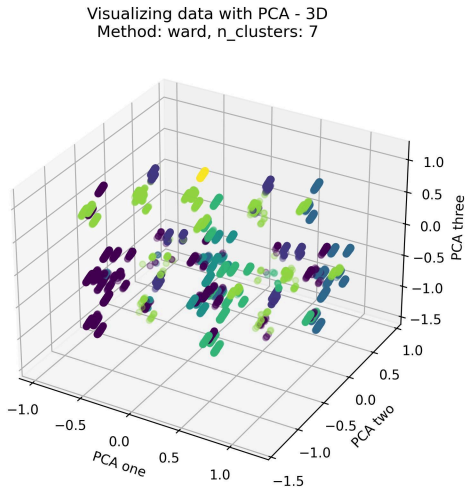
This method minimizes the increase in total within-cluster variance when merging two clusters.It considers the sum of squared differences within the cluster and merges clusters that result in the smallest increase in variance. It tends to create clusters of similar size and shape (compact and spherical).

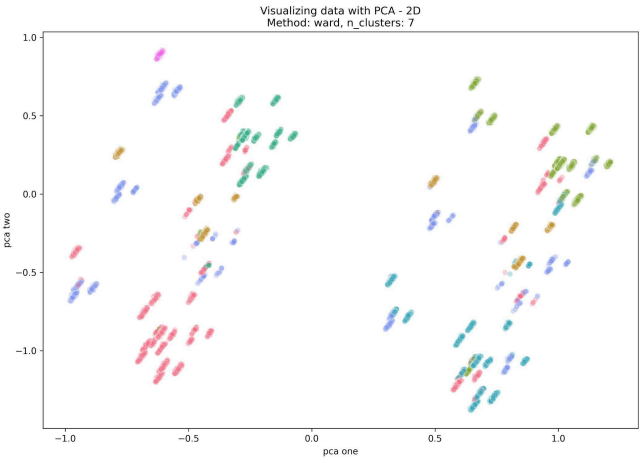
We ran this method for our dataset and got the dendogram. Through visual inspection and confirmed distance coordinates we drew a horizontal line and cut the y axis at 92.41. As a result we get 7 clusters. The Silhouette Score was calculated to be 0.2213 (0.22 approx) and the CH Ratio was 6738.47 (6738 approx.). As there are only 7 clusters formed, we can conclude that this method focused on forming bigger clusters, based on the initial similarities it noticed between the data points. It does not dive deeper into making more specific clusters, hence we can conclude that the clusters are quite generalised.

The figures below show the data visualizations after clustering with ward linkage method:




Visualizing data with TSNE
Method: ward, n_clusters: 7
tsne two
tsne one


Visualizing data with PCA - 3D
Method: ward, n_clusters: 7
PCA one
PCA two
PCA three

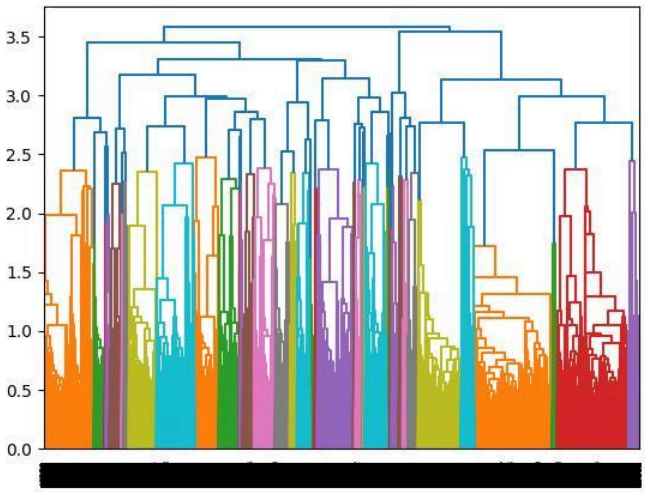

Visualizing data with PCA - 2D
Method: ward, n_clusters: 7
pca two
pca one

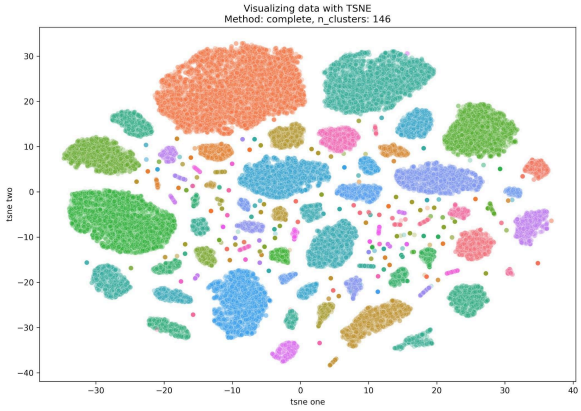
COMPLETE LINKAGE

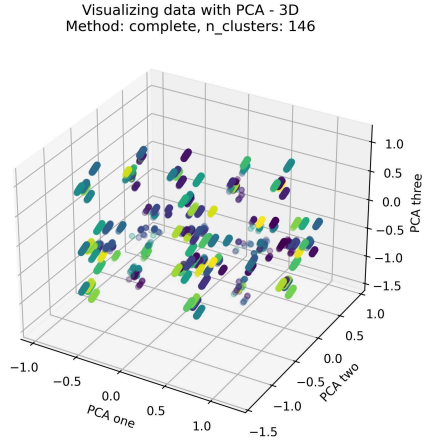
This method considers the distance between two clusters as the maximum distance between any two points, one from each cluster. It merges clusters where the largest pairwise distance is minimized. It produces compact clusters but can be sensitive to outliers, as one far away point can dominate the distance calculation.

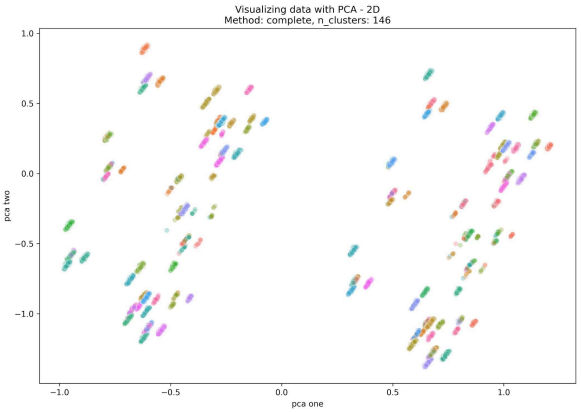
We ran this method for our dataset and got the dendogram below. Through visual inspection and confirmed distance coordinates we drew a horizontal line and cut the y axis at 1.737. As a result we get 146 clusters. The Silhouette Score was calculated to be 0.6855 (0.68 approx) and the CH Ratio was 5442.100 (5442 approx.). Since the complete linkage method merges two clusters by minimising the maximum distance between two points (each from a different cluster), this means it keeps the clusters apart for a longer time hence, it forms the maximum clusters out of the three linkage methods.

The figures below show the data visualizations after clustering with ward linkage:




Visualizing data with TSNE
Method: complete, n_clusters: 146
tsne two
tsne one


Visualizing data with PCA - 3D
Method: complete, n_clusters: 146
PCA one
PCA two
PCA three

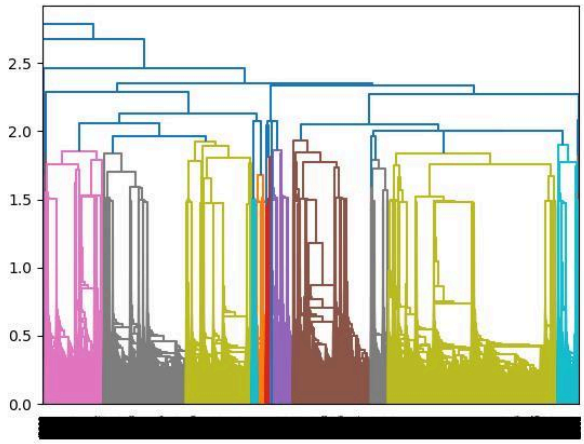

Visualizing data with PCA - 2D
Method: complete, n_clusters: 146
pca two
pca one

AVERAGE LINKAGE

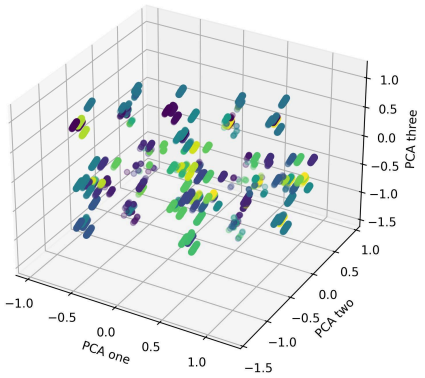
This method considers the distance between two clusters as the average of all pairwise distances between points in one cluster and points in the other. It merges clusters where the mean pairwise distance is minimized. It produces more balanced clusters compared to complete linkage but can still allow elongated shapes.

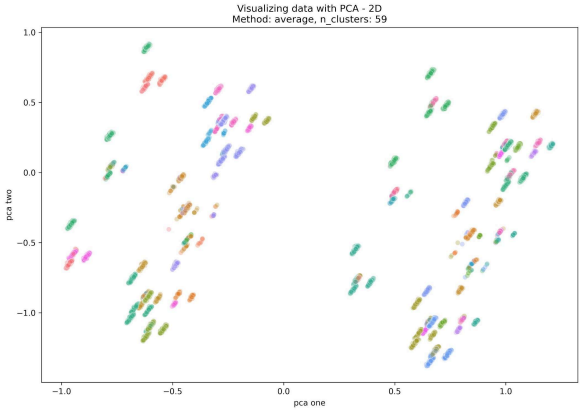
We ran this method for our dataset and got the dendogram below. Through visual inspection and confirmed distance coordinates we drew a horizontal line and cut the y axis at 1.697. As a result we get 59 clusters. The Silhouette Score was calculated to be 0.3619 (0.36 approx) and the CH Ratio was 2066.93 (2067 approx.). This method clusters by calculating the average distance between all the points in two clusters. It does not cluster based on early similarities (Ward Linkage) or keeps the clusters separated for a longer time (Complete Linkage) ; it creates a balance between these two scenarios.

The figures below show the data visualizations after clustering with ward linkage:



Visualizing data with PCA - 3D
Method: average, n_clusters: 59





Observation

Observing clustering through all the three types of linkage methods, we concluded that it is better to go ahead with the average linkage as it neither aggressively clusters data as in the case of ward linkage which generalises by clustering them into big clusters upon finding the initial similarity early on, nor it acts too specific to group data into large number of clusters as in the complete linkage. Hence, we choose the average linkage which forms clusters that are not too generalised or specific.

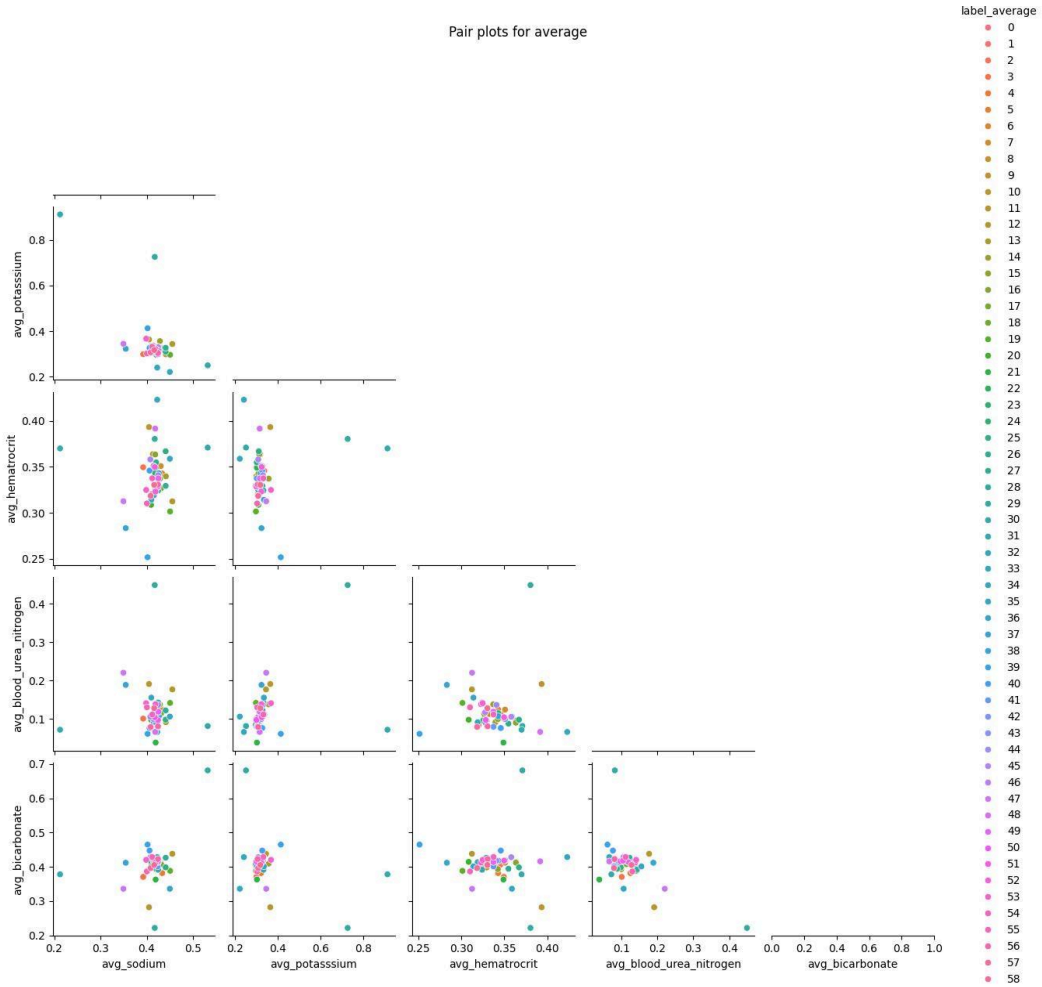
Hierarchical clustering analysis

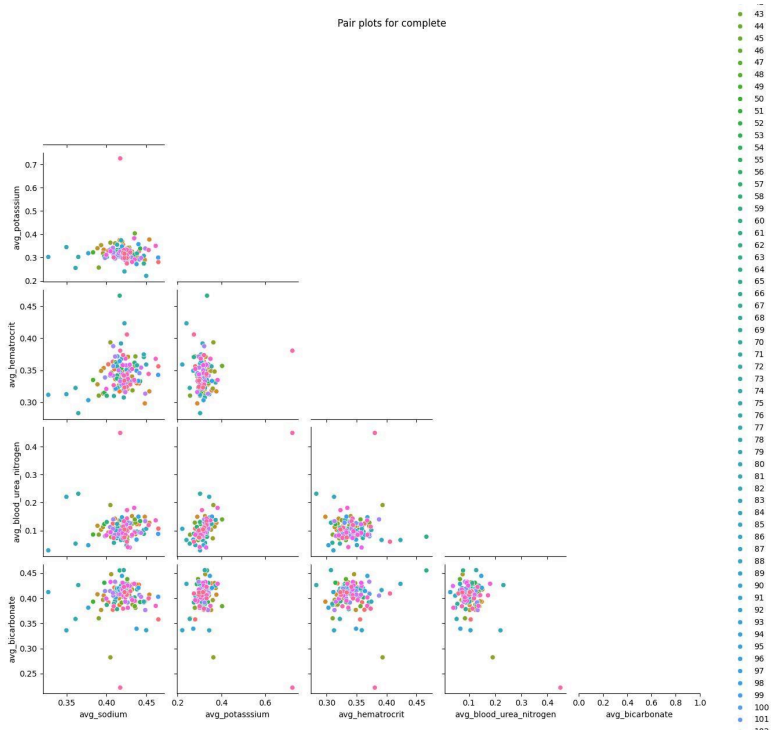
The below representation shows pair plots of centroids of the total number of clusters formed after hierarchical clustering. The centroid is calculated by taking the average of all the data points within each cluster.

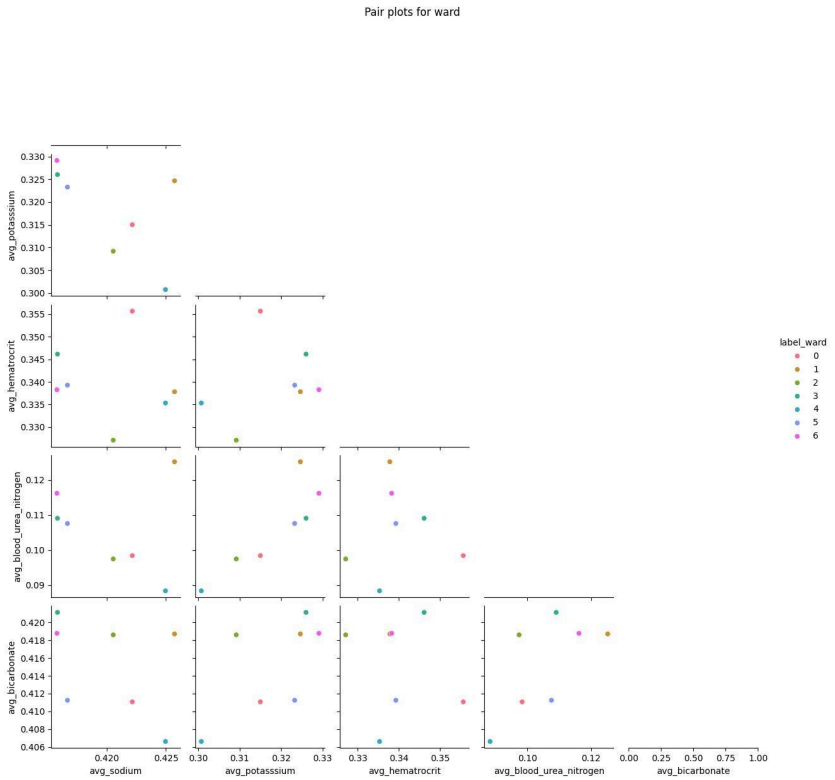
From the plots we observe that we are able to capture trends in different clusters. For instance, if we look closely at the avg_potasssium and avg_sodium (In method average) we are able to capture the patients with high avg_potasssium in a separate cluster which can be seen in the top left corner.

If we compare pair plots for average, ward, and complete, we see that (specifically in avg_potasssium and avg_sodium) ward is not able to capture the high avg_potasssium and the centroids are closely created. On the other hand, we see that the “complete” method is able to achieve fine granularity with many clusters and is able to create better clusters. Depending on the use case, we can choose between the methods. We are able to observe this same trend in other plots as well.

For example, if we are building this clustering to diagnose patients with high potassium consumption and want to alert them with fewer “false positives”, we should go with the “complete” method. If we are building this clustering for high level inference like finding demographics where high potassium is consumed, we can go ahead with the “ward” method. For a balanced use case, we can opt for the “average” method.




Pair plots for complete
avg_potassium
avg_hematrocrit
avg_blood_urea_nitrogen
avg_bicarbonate
avg_sodium
avg_potasssium
avg_hematrocrit
avg_blood_urea_nitrogen
avg_bicarbonate


Pair plots for ward
avg_potasssium
avg_hematrocrit
avg_blood_urea_nitrogen
avg_bicarbonate
label_ward
avg_sodium
avg_potasssium
avg_hematrocrit
avg_blood_urea_nitrogen
avg_bicarbonate

Conclusion

In this project, for the given datasets we would prefer k-means clustering over hierarchical clustering. This is because the computational requirements for the k-means algorithm are less than the hierarchical method and it is easier to find and analyse the number of clusters and further train the algorithm for the optimal number of clusters. However, the Silhouette Score calculated for the hierarchical clustering was better than the Silhouette Score of the k-means clustering, but we observed that the hierarchical method is computationally expensive and it was difficult to deduce the number of clusters from the dendograms of such a high dimensional data.

References

1. https://en.wikipedia.org/wiki/K-means_clustering
2. https://en.wikipedia.org/wiki/Hierarchical_clustering
3. https://en.wikipedia.org/wiki/Data_preparation
4. https://en.wikipedia.org/wiki/One-hot
5. https://en.wikipedia.org/wiki/Cluster_analysis