Decoding KMeans Clustering

Table of Contents

Student Name	**Riffat Munaf**
Student ID	22076284
Github Repository	https://github.com/riffatmunaf/ML_Assignment-KMeans-

KMeans Clustering

KMeans is a type of unsupervised machine learning algorithm which is most commonly used to cluster numerical data. Unlike supervised learning methods it does not require labeled data; instead, it takes in data and groups them under similar clusters. This makes KMeans very useful in areas like customer classification, image compression and identification of outliers. Indeed, the iterative nature of the algorithm guarantees that clusters are well defined and data points within the same cluster are more similar to each other than to data points belonging to other clusters.

In this report, I give a detailed description of the KMeans algorithm, its mechanism, how it is used, and some of the considerations to be made. To be able to better describe the workflow of KMeans, I rewrote the algorithm from scratch to be able to better explain what each step is doing.

Dataset

To show the real-world usage of KMeans, I used the algorithm to the Mall Customers dataset, which is available on Kaggle. This dataset includes some important customer attributes that are most useful for clustering in a retail perspective. The features include:

- **Age**: The age of the customer.
- **Annual Income (k$)**: The yearly income of the customer in thousands of dollars.
- **Spending Score (1-100)**: A score assigned to customers based on their spending behavior and habits.

I used 3 columns of this dataset so that I can make 3D charts as well to show the working mechanism of KMeans model as well. I analyzed the distribution of customers across clusters based on their age, income, and spending habits, providing meaningful insights into consumer segmentation. The subsequent sections discuss the detailed steps, mathematical foundations, and statistical properties of KMeans clustering process.

Mechanics of KMeans

The objective function of K-Means clustering is to minimize the **sum of squared distances** (inertia) between each data point and the centroid of its assigned cluster (Oti *et al.*, 2021). Mathematically, the objective function is expressed as:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu|^2$$

Where:

- k is the number of clusters.
- C_i is the set of data points assigned to cluster i.
- x represents a data point.

- u is the centroid (mean) of cluster i.
- $\|x-\mu i\|^2$ is the squared Euclidean distance between a data point x and its cluster centroid μ.

KMeans clustering aims to partition n data points into k clusters. Each cluster is represented by its centroid, and the algorithm iteratively updates these centroids to minimize intra-cluster distances also known as sum of squared distance (Ikotun *et al.*, 2022).

Working flow of KMeans is given below.

1. **Initialization**
   - Select k, the number of clusters.
   - Randomly initialize k centroids from the data points.
2. **Assignment Step**
   - Calculate the distance of each data point to all centroids.
   - Assign each data point to the nearest centroid based on the chosen distance metric (e.g., Euclidean, Manhattan, Minkowski).
3. **Update Step**
   - For each cluster, compute the new centroid by taking the mean of all data points assigned to that cluster.
   - Centroids represent the "center" of their respective clusters.
4. **Convergence Check**
   - Compare the old centroids with the updated centroids.
   - If the centroids' movements are less than a predefined tolerance epsilon, or if the maximum number of iterations is reached, stop the process.
   - Otherwise, return to the assignment step and repeat.
5. **Output**
   - Final cluster centroids and labels for all data points.
   - The clusters formed reflect the inherent groupings in the data.

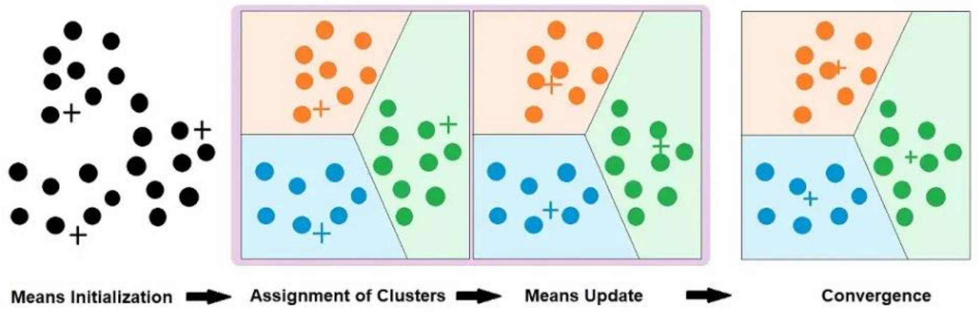


Figure 1: KMeans working (Arya and Arya, 2023)

Here's a detailed step-by-step breakdown:

1. Initialization

Centroid initialization is crucial to the algorithm's performance. Initial centroids are chosen randomly from the dataset, and the effectiveness of this choice significantly impacts the convergence speed and cluster quality. Poor initialization may lead to:

- Increased iterations for convergence.
- Suboptimal cluster assignments.

And the reason to this is that when all centroid is selected from the same cluster then KMeans will take time to move the centroids belonging to other clusters away to their actual position. Advanced centroid initialization techniques like selecting totally different and variated centroids can mitigate these issues (Arthur and Vassilvitskii, 2007).

2. Points Assignment

Each data point is assigned to the nearest centroid. This is determined by calculating distances between the data point and all centroids and then the data points are assigned to their nearest centroid. The choice of distance metric plays a critical role:

- **Euclidean Distance** (most common): Measures straight-line distance.
- **Manhattan Distance**: Sum of absolute differences, more robust to outliers.
- **Minkowski Distance**: A generalization of distance formulas.

The choice of distance metric should align with the nature of the data:

- For **continuous data** without outliers, use **Euclidean Distance**.
- For **high-dimensional or outlier-prone data**, opt for **Manhattan Distance**.
- For **mixed needs or experimentation**, consider the tunable **Minkowski Distance**.

Selecting the right distance metric ensures meaningful clusters and reduces algorithmic inefficiencies.

3. Centroid Update

Centroids are updated by calculating the mean of all data points assigned to each cluster:

$$C_j = \frac{1}{n_j} \sum_{i \in C_j} X_i$$

Where:

- C_j is the updated centroid of cluster j.
- n_j is the number of points in cluster j.
- X_i are the points in cluster j.

4. Convergence Check

Convergence is the point at which the KMeans algorithm determines that the clustering process is complete and no further iterations are needed. The algorithm relies on two stopping criteria: **centroid stabilization** (using a tolerance threshold epsilon) and a **maximum number of iterations**.

Epsilonϵ (Tolerance)

- epsilon is a small predefined value that acts as the convergence threshold.
- It represents the maximum allowable movement of centroids between iterations for the algorithm to consider the clusters stable.
- Mathematically, the algorithm checks: $\|C_{new} - C_{old}\| \leq \epsilon$
- **Significance**:
    - A smaller epsilon ensures more precise convergence but may increase computational time.
    - A larger epsilon speeds up the process but risks early termination, potentially yielding suboptimal clusters.

Iterations

- An iteration is a single cycle through the KMeans algorithm steps:
    1. Assigning data points to the nearest centroids.
    2. Updating centroids based on cluster assignments.
- The algorithm repeats these steps until convergence or the maximum number of iterations is reached.

Stopping Criteria

The KMeans algorithm stops clustering under either of the following conditions:

Centroid Stabilization (epsilon):

- The centroids' positions do not change significantly between consecutive iterations.
- This indicates that the data points are no longer reassigning to different clusters, and the clusters have stabilized.

Maximum Number of Iterations:

- The algorithm is capped at a predefined number of iterations, even if centroids have not fully stabilized.
- This is a fail-safe mechanism to prevent infinite loops or excessive computation on complex datasets.

Why Two Stopping Criteria?

- **Centroid Stabilization** ensures the algorithm stops when a solution is found.
- **Maximum Iterations** acts as a safeguard to prevent continuous processing, especially on datasets that are hard to cluster or where perfect convergence is impractical.

5. Output Step

After convergence is achieved, the KMeans algorithm produces the following outputs:

Final Centroids:

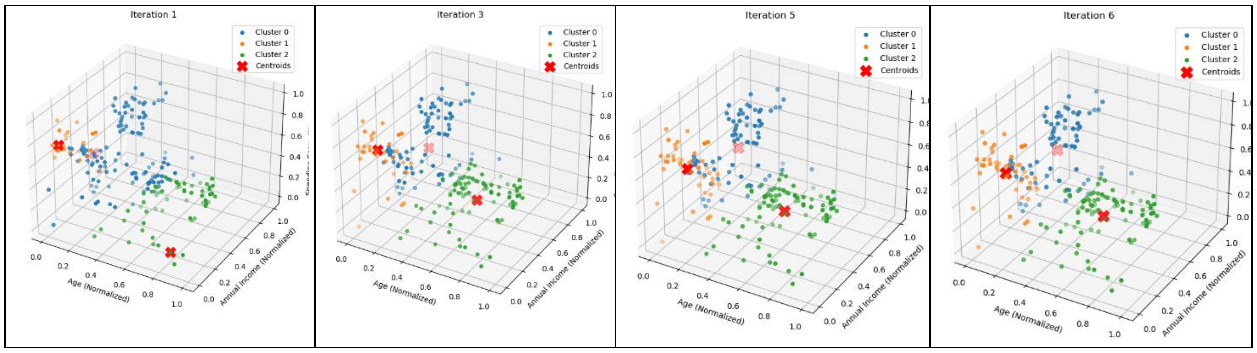
- The coordinates of the centroids for each cluster.
- These centroids represent the "center" or "mean" of the data points in each cluster.

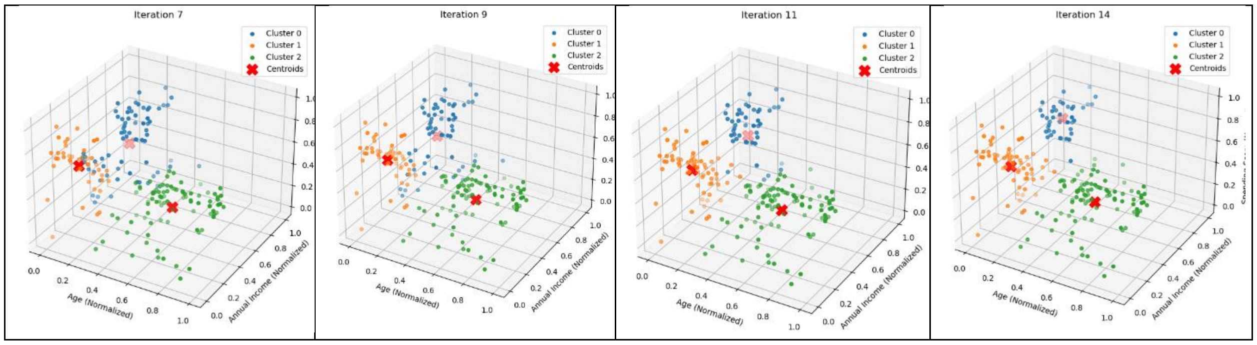
Cluster Labels:

- An array indicating the cluster assignment for each data point.
- These labels can be used for analysis, visualization, or further machine learning tasks.

Implementation

On the “Mall Customers” dataset, the KMeans clustering algorithm was applied on Age, Annual Income, and Spending Score to group the customers. The features were first normalized (0-1) so that all features contributed equally in the clustering process. Basic functions were defined for distance measurement, cluster allocation, centroid update, and convergence. The algorithm repeatedly created clusters of data points with the nearest centroids and adjusted the centroids until convergence either when the centroids became stable with a tolerance of epsilon = 0.0001 or when the maximum number of iterations was 300. The results were final cluster centroids and labels, of which statistics were done to analyze the nature of the clusters obtained. The 3D clusters were produced to show how clusters changed through iterations, which gave a clear insight of the KMeans process and its use. KMeans took 14 iterations to converge to the optimal point and for each iteration a visualization has been made that can be seen below.





Over 14 iterations, the algorithm successfully stabilized the clusters. Each cluster highlights a unique group of customers.

Table 1: Statistical Properties for each cluster

Cluster	Mean Age	Mean Income	Mean Spending	Std Age	Std Income	Std Spending
0	32.87	86.10	81.52	3.81	16.13	9.87
1	25.14	43.79	55.81	5.51	19.29	21.94
2	50.48	61.09	33.40	10.13	24.77	17.57

- Cluster 0 primarily consists of **young adults** with **high annual income** and **high spending scores**, representing affluent customers who frequently engage in high-value purchases. This segment is likely the primary focus for premium products and loyalty programs, as these customers exhibit both the financial capability and willingness to spend.
- Cluster 1 captures a group of relatively **younger individuals** with **moderate income** and **moderate spending scores**. These customers are more budget-conscious compared to Cluster 0. Targeted promotions, discounts, or value-for-money offerings would resonate well with this group, as they are less inclined toward luxury purchases but still show moderate spending behavior.
- Cluster 2, on the other hand, represents **older customers** with **moderate income** and **low spending scores**. This group likely includes infrequent shoppers who prioritize essential purchases over indulgences. Engaging these customers may require tailored strategies, such as offering senior-friendly discounts, products that cater to their specific needs, or personalized incentives to encourage more frequent visits.

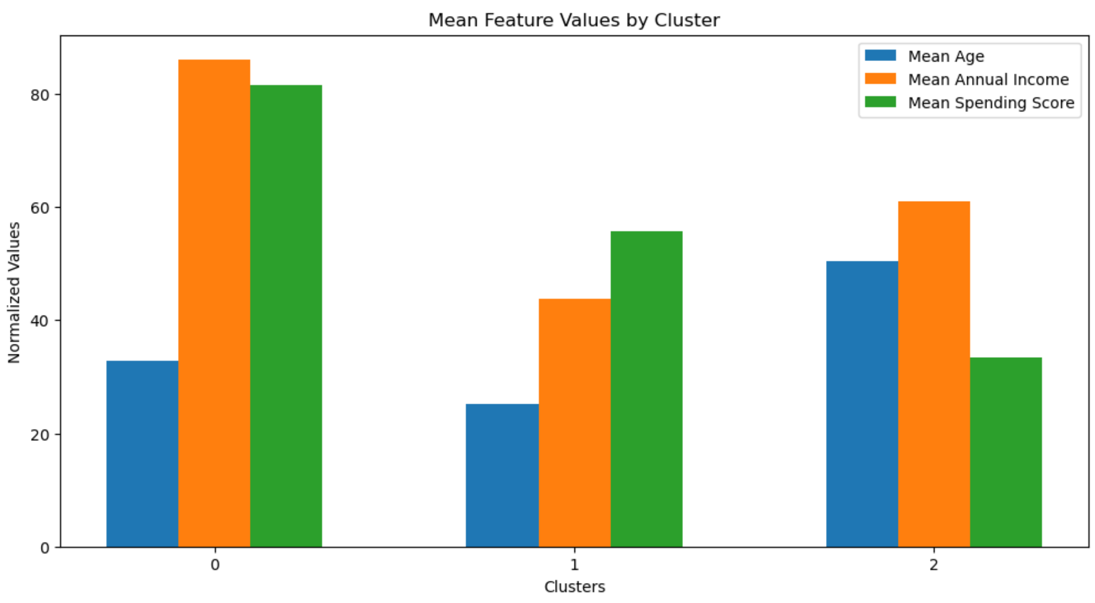


Figure 2: Mean Feature Values by Cluster

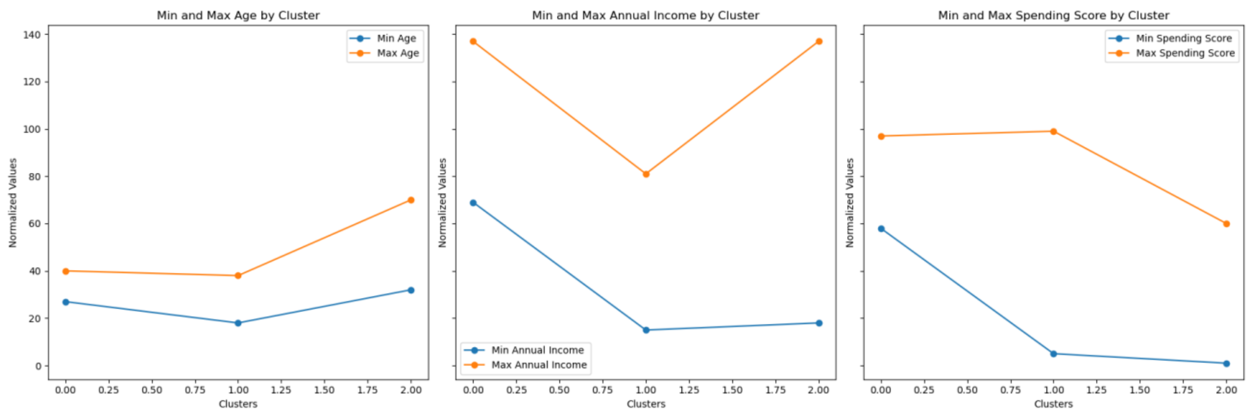


Figure 3: MinMax Feature Values by Cluster

Conclusion

KMeans is a simple and fast clustering algorithm that can be used in any type of data depending on the features. Its steps, which are centroid initialization, point assignment, centroid updating and convergence testing make it useful in finding hidden patterns. Some of the parameters that are selected include the number of clusters.

The choice of the kernel function (e.g., linear, polynomial or radial basis function or RBF (k) and distance metrics (Euclidean or Manhattan), has a major influence on its performance. Though it is sensitive to the initialization and not as effective with non-spherical clusters KMeans is still widely used because it is easy to understand and implement, fast and can be used in a wide range of tasks. They remain an indispensable instrument for analysis and decision making based on data.

References

Arthur, D. and Vassilvitskii, S. (2007) 'k-means++: the advantages of careful seeding,' *Symposium on Discrete Algorithms*, pp. 1027–1035. https://doi.org/10.5555/1283383.1283494.

Arya, N. and Arya, N. (2023) *K-Means Clustering for Unsupervised Machine Learning | EJable*. https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/k-means-clustering/.

Ikotun, A.M. *et al.* (2022) 'K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data,' *Information Sciences*, 622, pp. 178–210. https://doi.org/10.1016/j.ins.2022.11.139.

Oti, E.U. *et al.* (2021) 'Comprehensive Review of K-Means Clustering Algorithms,' *International Journal of Advances in Scientific Research and Engineering*, 07(08), pp. 64–69. https://doi.org/10.31695/ijasre.2021.34050.