Clustering is an unsupervised machine learning task in which the goal is to partition points within a dataset into distinct groups (or clusters) such that elements within the same group are similar to each other. K-means is an example of a hard-clustering method which means that each point can only be assigned to a single cluster and there are hard boundaries between clusters.

Though we told to use k=7 as value but if I was to justify or pick, this is what I will do…

A graph with a line

Description automatically generated

K-Means is sensitive to the initial placement of centroids, as demonstrated by running the algorithm multiple times with n\_init=1. The sum of squared distances (SSD) varied significantly across different random initializations. However, when n\_init=10 was used, the algorithm automatically selected the best initialization, resulting in the lowest SSD. This demonstrates the importance of using multiple initializations to ensure robust clustering.  
The effect of the n\_init parameter in K-Means clustering was evaluated by comparing the **Sum of Squared Distances (SSD)** across different values of n\_init. SSD measures the compactness of clusters by summing the squared Euclidean distances between data points and their respective cluster centroids. A lower SSD indicates better clustering, as data points are more tightly grouped within their clusters.

The results show that:

1. Using a single initialization (n\_init=1) results in a high SSD (~700,000), highlighting the sensitivity of K-Means to poor random centroid placement.
2. Increasing n\_init to 5 or 10 significantly reduces SSD, as the algorithm selects the best initialization from multiple random trials.
3. Beyond n\_init=10, the SSD stabilizes, suggesting diminishing returns for higher values. This balance highlights that n\_init=10 achieves robust clustering results efficiently.

K-Means clustering is sensitive to the initial placement of centroids, which are chosen randomly at the start of the algorithm. Poor initial centroids can lead to suboptimal clustering (local optima). To address this, the n\_init parameter allows the algorithm to run multiple times with different random initializations of centroids. For example, with n\_init=10, the algorithm performs 10 runs, each with a new initialization, and selects the clustering result with the lowest Sum of Squared Distances (SSD).

Using a fixed random\_state ensures that the sequence of random initializations is reproducible across runs, while still allowing the algorithm to test multiple centroid placements. This improves the robustness of K-Means and ensures that the best initialization is consistently chosen

A graph with a blue line

Description automatically generated

The default Euclidean distance metric was used in K-Means because it aligns with the algorithm's objective of minimizing the variance within clusters. This alignment ensures that clusters are compact and well-separated. Experiments comparing Euclidean distance with alternative metrics (e.g., Manhattan distance) show that K-Means achieves lower SSD and higher Silhouette Scores with Euclidean distance, validating its suitability for numerical datasets.

Euclidean Distance:

SSD: 7738681547.12, Silhouette Score: 0.33

Manhattan Distance:

SSD: 14893455.94, Silhouette Score: 0.28

The tolerance (tol) parameter in K-Means determines the precision of convergence by setting the threshold for acceptable changes in SSD between iterations. To evaluate its impact, experiments were conducted using tolerances ranging from 10−210^{-2}10−2 to 10−1010^{-10}10−10. The results show that:

1. **Higher Tolerances (e.g., 10−210^{-2}10−2)**: The algorithm converges quickly (fewer iterations), but at the cost of higher SSD, indicating less optimal clustering.
2. **Stricter Tolerances (e.g., 10−610^{-6}10−6 to 10−1010^{-10}10−10)**: The algorithm takes significantly more iterations to converge, but the improvement in SSD becomes negligible beyond 10−610^{-6}10−6.

The default tolerance of 10−410^{-4}10−4 achieves a good balance, allowing the algorithm to efficiently converge to compact and stable clusters without unnecessary computational cost.

A graph of a graph

Description automatically generated with medium confidence

Although the default max\_iter=300 is sufficient for most datasets, experiments demonstrate that the algorithm consistently converges within ~80 iterations for this dataset. Reducing max\_iter to 100 achieves the same clustering quality (SSD) while minimizing unnecessary computational overhead. Therefore, max\_iter=100 is an efficient and effective choice for this dataset.  
A graph of a graph

Description automatically generated with medium confidence

The init\_params parameter in GMM was set to random to allow flexible initialization of cluster parameters (means, covariances, and weights). Unlike the default kmeans initialization, which assumes spherical clusters, random initialization avoids potential biases and enables the model to explore diverse starting points. This choice is particularly useful for datasets with non-spherical or complex clusters, and combined with a fixed random\_state, it ensures reproducibility without limiting the variability of initial parameters

**How to Use This in the Report**

**Sample Explanation**

"Clustering errors were calculated by counting the number of pairs of datapoints with the same true class label but assigned to different clusters by the algorithm. The results show:

1. **K-Means**: Produced fewer errors compared to the random baseline, demonstrating its ability to form meaningful clusters.
2. **GMM**: Made fewer errors than K-Means, leveraging its probabilistic framework to capture more complex cluster structures.
3. **Random Baseline**: Expectedly performed the worst, as it assigns clusters randomly without considering the data structure."

The error counts highlight the strengths and weaknesses of each clustering algorithm:

1. **K-Means** made **15,430,954 errors**, which is fewer than the Random Baseline but more than GMM. This is because K-Means minimizes Euclidean distances and assumes spherical clusters. If the true clusters are non-spherical or overlapping, K-Means struggles to assign datapoints correctly, especially near cluster boundaries.
2. **GMM** performed the best, making **13,697,715 errors**, due to its ability to model elliptical cluster shapes using covariance and its probabilistic approach to soft clustering. This allows GMM to better handle overlapping clusters and datapoints near boundaries.
3. The **Random Baseline** made the most errors (**16,121,145**) because it assigns cluster labels without considering the structure of the data. The high error count reflects the lack of meaningful clustering.

These results demonstrate that structured clustering algorithms like K-Means and GMM significantly outperform random assignment. Furthermore, GMM's flexibility and probabilistic framework make it the most effective choice for datasets with complex cluster geometries.

K-Means Errors: 15430954 / 18807759

GMM Errors: 13697715 / 18807759

Random Baseline Errors: 16121145 / 18807759

K-Means Errors: 12792643 / 18807759

GMM Errors: 12794219 / 18807759

Random Baseline Errors: 16121145 / 18807759

K-Means Errors: 12792643 / 18807759

GMM Errors: 12794219 / 18807759

Random Baseline Errors: 16121145 / 18807759

**ample Explanation**

"A comparison was conducted between K-Means with k-means++ initialization and random initialization. The results showed:

* **K-Means with k-means++ Initialization** achieved a lower SSD and higher Silhouette Score compared to random initialization. This demonstrates that k-means++ improves centroid placement, leading to tighter and better-separated clusters.
* The random initialization method resulted in less compact clusters and lower clustering quality, as evidenced by higher SSD and lower Silhouette Score.

K-Means Errors: 12426423 / 18807759

The impact of the maximum number of iterations (max\_iter) on K-Means clustering was analyzed. Results showed that the algorithm converged within ~20 iterations for all tested values of max\_iter, provided that the tolerance (tol) was set to 1e-4. Increasing max\_iter beyond 20 had no effect on the number of iterations or clustering quality (as measured by SSD). This indicates that the algorithm's convergence is primarily governed by the tolerance parameter, making excessively large max\_iter values unnecessary.

Different covariance structures were tested to identify the best fit for the data. The 'full' covariance type provided the lowest BIC and was selected for clustering.