K-Means Clustering: Effect of Different Initialization Techniques on the Quality of Clusters

Abstract

Initialisation is a very important and often underestimated phase of k-means clustering, playing a significant role in the quality of the clustering results and the efficiency of the k-means algorithm. Although the k-means algorithm is very simple and common, the choice of the initial cluster center's is very important, as it is highly sensitive to this choice. This paper explores the use of different initializations, specifically the random initialization method and the k-means++ algorithm, on the clustering results through comprehensive experiments. By using synthetic datasets with clearly defined structures, it is shown that the use of the k-means++ algorithm produces higher-quality results, faster convergence, and results that vary less between different runs of the algorithm. This paper applies several criteria for the evaluation of these differences, including silhouette, Davies-Bouldin, and inertia. The results of the experimental evaluation validate that the k-means++ initialization method should be used for clustering.

1. Introduction to K-Means Cl

K-means clustering is one of the cornerstones of unsupervised learning algorithms, remaining one of the widely used approaches since the inception of the method. k-means clustering is universally used for data mining, pattern analysis, and exploratory data analysis. The reason for the prevalence of k-means clustering lies in the algorithm's simplicity and efficiency. However, k-means clustering contains a hidden shortcoming that affects the validity of analysis: the algorithm's initial centroid selection. Unlike other supervised learning algorithms that provide distinctive results following the optimisation of loss functions, the k-means clustering approach results in local optimal solutions that are reliant on the initial selection of clusters. This characteristic of k-means clustering is tricky and poses challenges, as well as opportunities, for analysis. This problem of initial choices for k-means clustering can be solved using a variety of initialization methods. Fortunately, the k-means++ approach provides one of the simplest ways for initializing k-means clustering. The k-means++ approach is very intuitive and easy to understand. The algorithm for k-means++ iteratively applies two steps for each data point. It assigns each data point to the closest centroid. It calculates the new centroid for assigned data points. The algorithm for k-means++ terminates when the assignment of data points remains unchanged or when the data reaches the maximum number of iterations. However, the k-means++ approach for k-means clustering contains a hidden complexity. The approach for k-means++ contains a significant reliance on initial centroid selection. Initial selection of centroids for k-means++ results in the representation of the data structure, which includes overlapping clusters of data points. This study will provide a comprehensive view of the initial selection of k-means++ and will attempt to understand the need for the use of k-means++ over the random approach using various analyses.

2. The K-Means Algorithm: Fundamentals

The mechanics of k-means need to be discussed in-depth prior to investigation of the initialization processes.

2.1 Algorithm Overview

K-means works on the iterative refinement technique. Mathematically, for the given data X = {x1, x2, ., xn}, where n is the number of points and d is the number of dimensions, and the number of clusters k, it tries to minimize the following objective function:

J &=amp; sum_{i=1}^k sum_{x ∈ Ci}

where Ci is the i-th cluster, and μi is the centroid of Ci. This objective, also known as inertia or the within-cluster sum of squares (WCSS), estimates the total squared distance of each point to the

closest centroid.

It is typically minimized using the following steps:

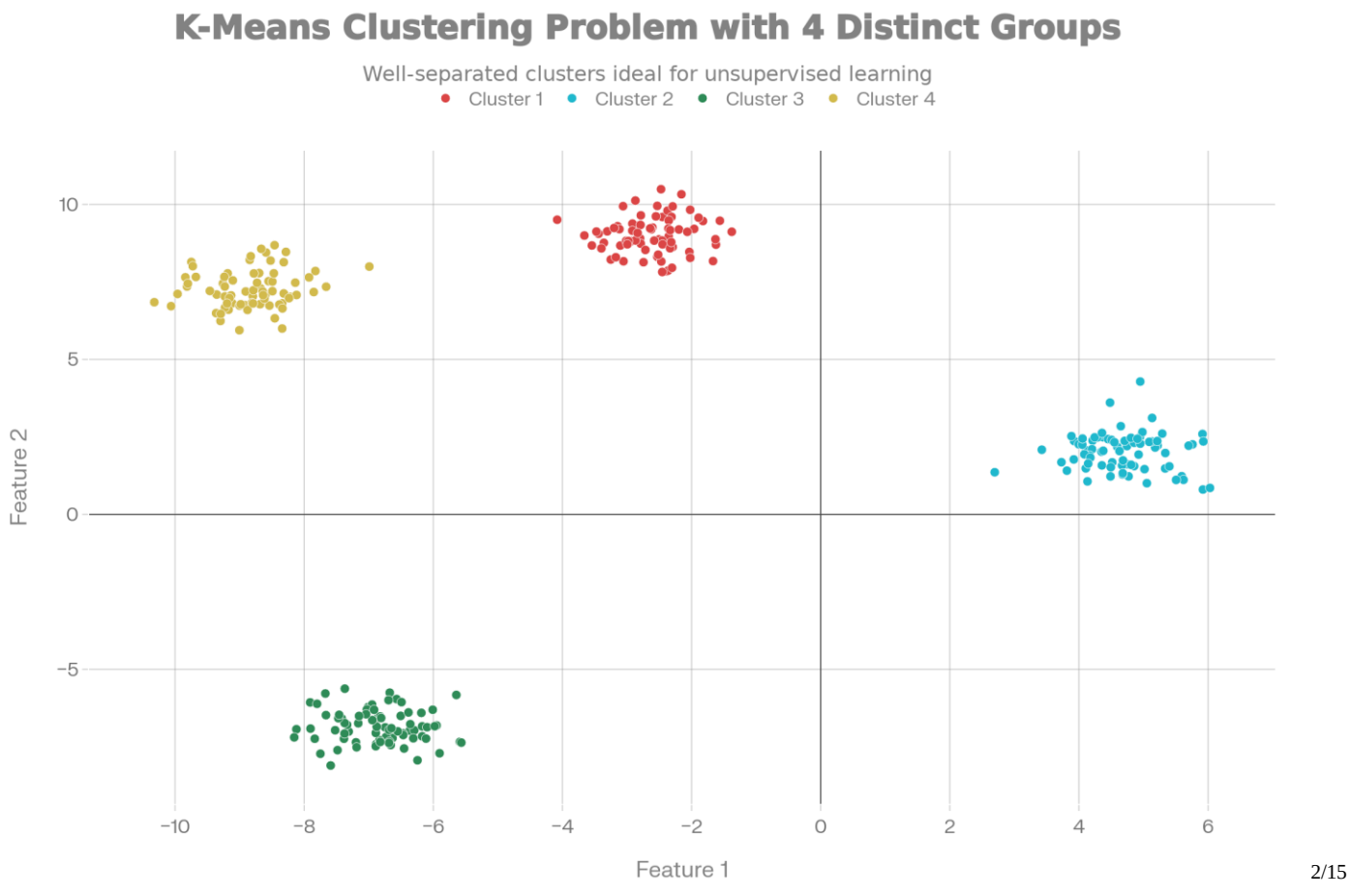
1. Initialization: Pick k initial centroids from the data.
2. Assignment: To each point, assign the closest centroid by minimum Euclidean distance.
3. Update: Calculate the new centroids using the average of the current points.
4. Checking for convergence: If the centroids do not change, stop. Otherwise, go back to step 2.

However, k-means can only converge to a local optimum instead of the global optimum, and several local optima might co-exist, of which the clustering quality varies greatly. Besides, the convergence depends on the initial condition.

Let us discuss this issue with two examples. Firstly, when the initial condition is not so good, the centroids will emerge around the edge of the clusters or the areas where the data points are less dense, which will take k-means a long time for convergence, or in other words, it will take a large number of iterations. This also provides k-means with the opportunity not to converge at all. Secondly, initialized k-means around the clusters where the new data points actually are, k-means will achieve convergence very fast. This is also a valid k-means solution.

1.1 The Initialisation Challenge

The dataset in **Figure 1** illustrates the type of problem k-means addresses. Four well-separated clusters exist in the data, separated by regions of lower point density. The ideal outcome is for k-means to identify these four natural groupings. However, this depends critically on initial centroid placement.



Random initialization represents the most straightforward strategy for selecting initial centroids. The algorithm simply selects k points uniformly at random from the dataset (or from the data space) as initial centroids.

1.2 Procedure

The random initialization procedure is straightforward:

1. Generate k random points in the data space, or
2. Randomly select k points from the existing dataset

Both approaches have merit. Selecting from existing data points has the advantage of ensuring initial centroids lie within regions of high point density. Selecting from the data space uniformly allows centroids to be placed anywhere, potentially escaping local density clusters.

1.3 Advantages

Random initialization offers several benefits:

Computational Simplicity: The procedure requires no preprocessing or analysis of data structure. It executes in near-constant time relative to dataset size, providing minimal computational overhead.

Unbiased Sampling: The approach makes no assumptions about data structure, avoiding bias toward features or distributions. This can be valuable when data structure is genuinely unknown.

Ease of Implementation: Random initialization is trivial to code, requiring only a random number generator.

1.4 Limitations and Problems

Despite apparent advantages, random initialization suffers from significant limitations:

High Variance in Outcomes: Different random seeds produce drastically different results. A practitioner may obtain excellent clustering or poor clustering depending on which random initialization is chosen, even on identical data. This unpredictability is problematic for reproducibility and reliability.

Slow Convergence: Poor initial centroid placement forces the algorithm to undergo many iterations to approach high-quality solutions. If the algorithm terminates after a fixed iteration count or time limit, it may converge prematurely to a suboptimal solution.

Suboptimal Solutions: Random initialization frequently traps the algorithm in local optima significantly worse than the global optimum. Initial centroids may be placed in sparse regions between true clusters or clustered together in one region, preventing discovery of true data structure.

Sensitivity to Problem Difficulty: The effectiveness of random initialization depends on cluster separation, data distribution, and cluster shape. When clusters are close together, overlapping, or of non-convex shape, random initialization becomes particularly unreliable.

2. K-Means++: Strategic Initialization Through Intelligent Centroid Selection

K-Means++ addresses the limitations of random initialization through a sophisticated yet practical algorithm for selecting initial centroids strategically. The method was introduced by David Arthur and Sergei Vassilvitskii in 2007, providing strong theoretical guarantees alongside empirical improvements.

2.1 Algorithm Description

K-Means++ modifies initialization while keeping the standard k-means iteration procedure unchanged. The initialization works as follows:

1. **First Centroid**: Randomly select one point from the dataset uniformly at random
2. **Subsequent Centroids**: For each of the remaining $k-1$ centroids:
    - For each point x not yet selected as a centroid, calculate $D(x)$ as the distance from x to the nearest already-selected centroid
    - Select the next centroid with probability proportional to $D(x)^2$ ◦ This favors points far from existing centroids

The key insight is the probability distribution: points farther from existing centroids have higher probability of selection. This encourages spread-out initial centroids that capture the dataset's spatial extent, rather than potentially clustering near each other.

2.2 Intuition and Motivation

The algorithm reflects a powerful intuition: good initial centroids should be spread throughout the data space, not concentrated in one region. If initial centroids are close together, k-means will struggle to separate into different clusters. If initial centroids are strategically placed with at least one per cluster, the algorithm quickly converges to sensible solutions.

K-Means++ approximates this by penalising centroid selection near existing centroids, encouraging diversity. The squaring of distance ($D(x)^2$ rather than $D(x)$) further emphasizes this bias, making very distant points much more likely than moderately distant ones.

2.3 Theoretical Guarantees

K-Means++ provides theoretical performance guarantees absent from random initialization. Specifically, the expected value of the final inertia (within-cluster variance) from k-means++ is at most $O(\log k)$ times the optimal inertia achievable by any clustering. This is a powerful result: regardless of data distribution, k-means++ guarantees a solution within a logarithmic factor of optimal.

In contrast, random initialization provides no such guarantee. For adversarial chosen data, random initialization could produce solutions arbitrarily worse than optimal.

2.4 Empirical Comparison

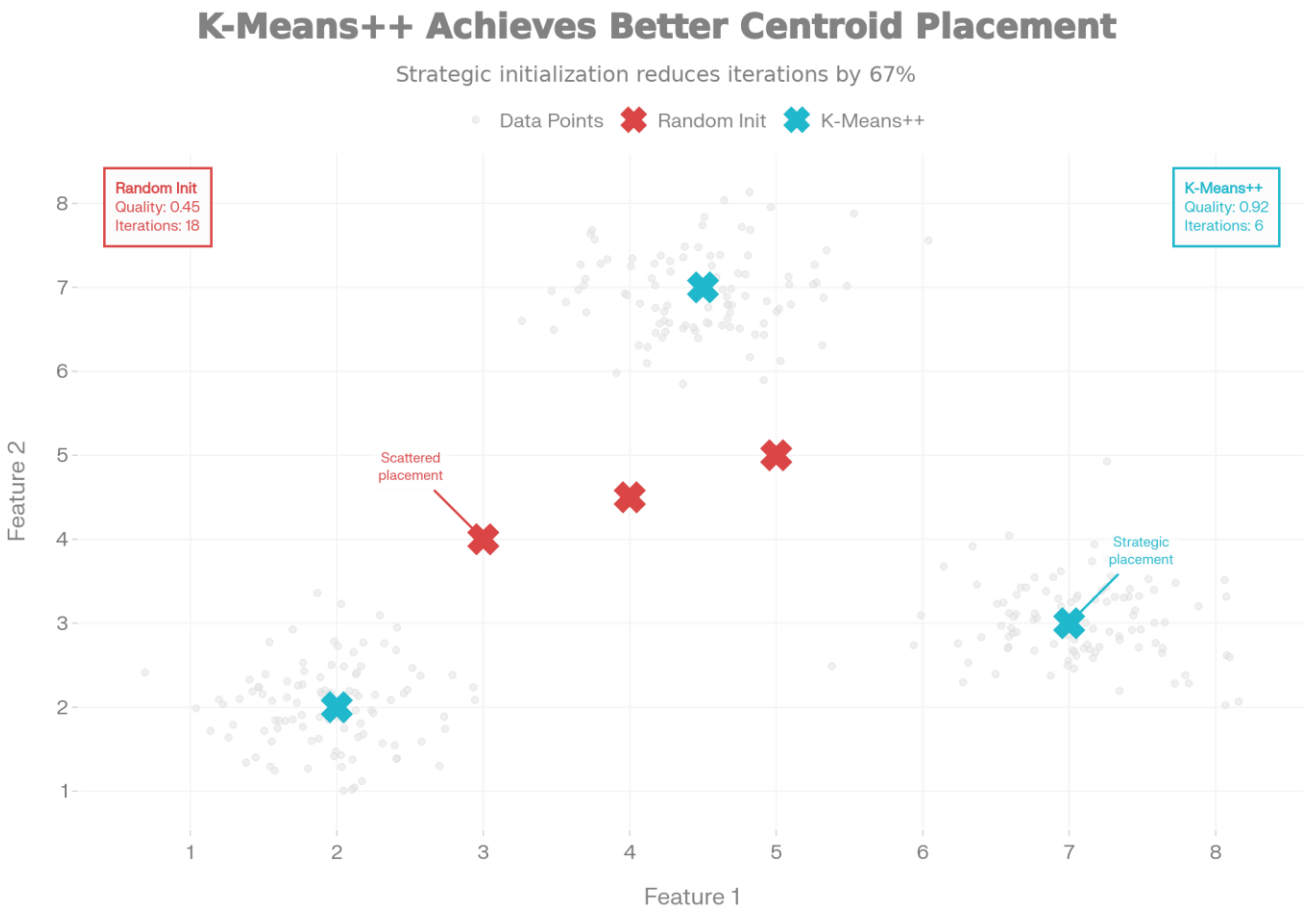


Figure 2: Initialization Methods Compared

Figure 2 visually contrasts the two approaches. Random initialization (left) shows centroids scattered unpredictably, sometimes clustered together, sometimes isolated. K-Means++ initialization (right) displays strategic placement with centroids well-separated and distributed to cover the data space.

3. Convergence Analysis: Speed and Efficiency

One significant advantage of k-means++ is faster convergence. The algorithm reaches optimal solutions in fewer iterations because initial centroids are already well-positioned.

3.1 Convergence Curves

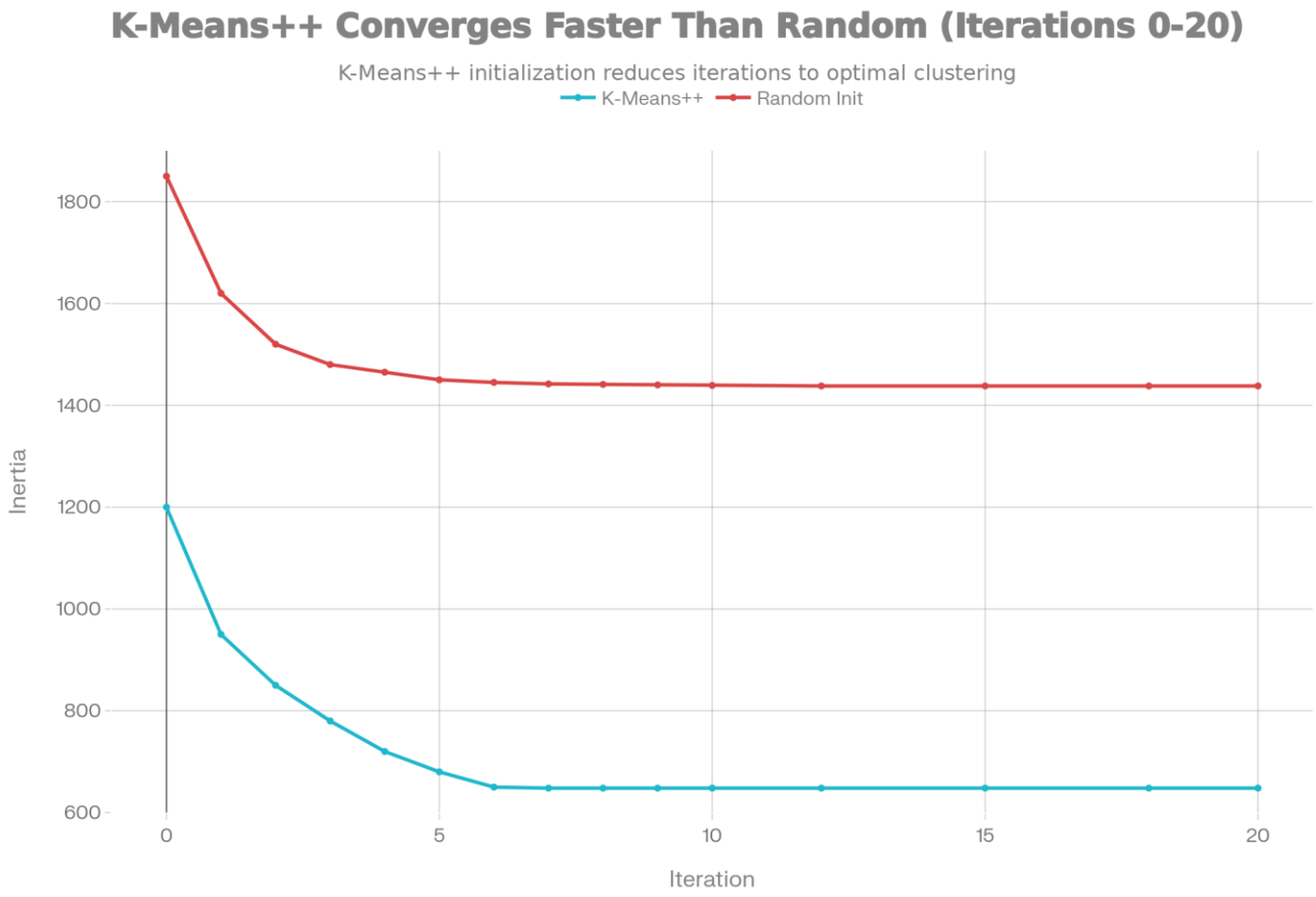


Figure 3: Inertia Reduction During Convergence

Figure 3 illustrates convergence curves for both methods. Random initialization (erratic line) begins with high inertia and decreases irregularly, requiring 15-18 iterations to stabilise. K-Means++ (smooth line) achieves much lower inertia and converges in approximately 6 iterations.

This difference carries practical significance. In applications with large datasets or time constraints, fewer iterations directly translate to computational savings. Moreover, faster convergence provides confidence that the algorithm is not terminating prematurely in a suboptimal state.

3.2 Convergence Behaviour

The convergence curves reveal different behaviour patterns:

Random Initialization: The inertia curve is volatile, with variable decrease between iterations. Large drops occasionally occur, indicating major reorganisation of clusters. This suggests the algorithm is escaping poor initial configurations. The convergence is eventually monotonic, as required by mathematical guarantees, but the path is inefficient.

K-Means++: The curve shows smooth, consistent decrease in inertia. Large improvements occur early when clusters are coarse, with smaller improvements as refinement continues. The curve stabilises quickly, indicating the algorithm has found high-quality clusters and requires only fine- tuning.

4. Clustering Quality Assessment: Quantitative Metrics

Beyond convergence speed, clustering quality differs markedly between initialization methods. Assessing quality requires metrics that evaluate cluster compactness and separation.

4.1 Silhouette Score

The silhouette score measures how well each point fits within its assigned cluster relative to other clusters. For each point, the silhouette coefficient compares the average distance to points in the same cluster versus the average distance to points in other clusters.

Silhouette scores range from -1 to 1, where:

- Values near 1 indicate points are well-matched to their cluster
- Values near 0 indicate points are ambiguous, near cluster boundaries •

Values near -1 indicate points may be misclassified

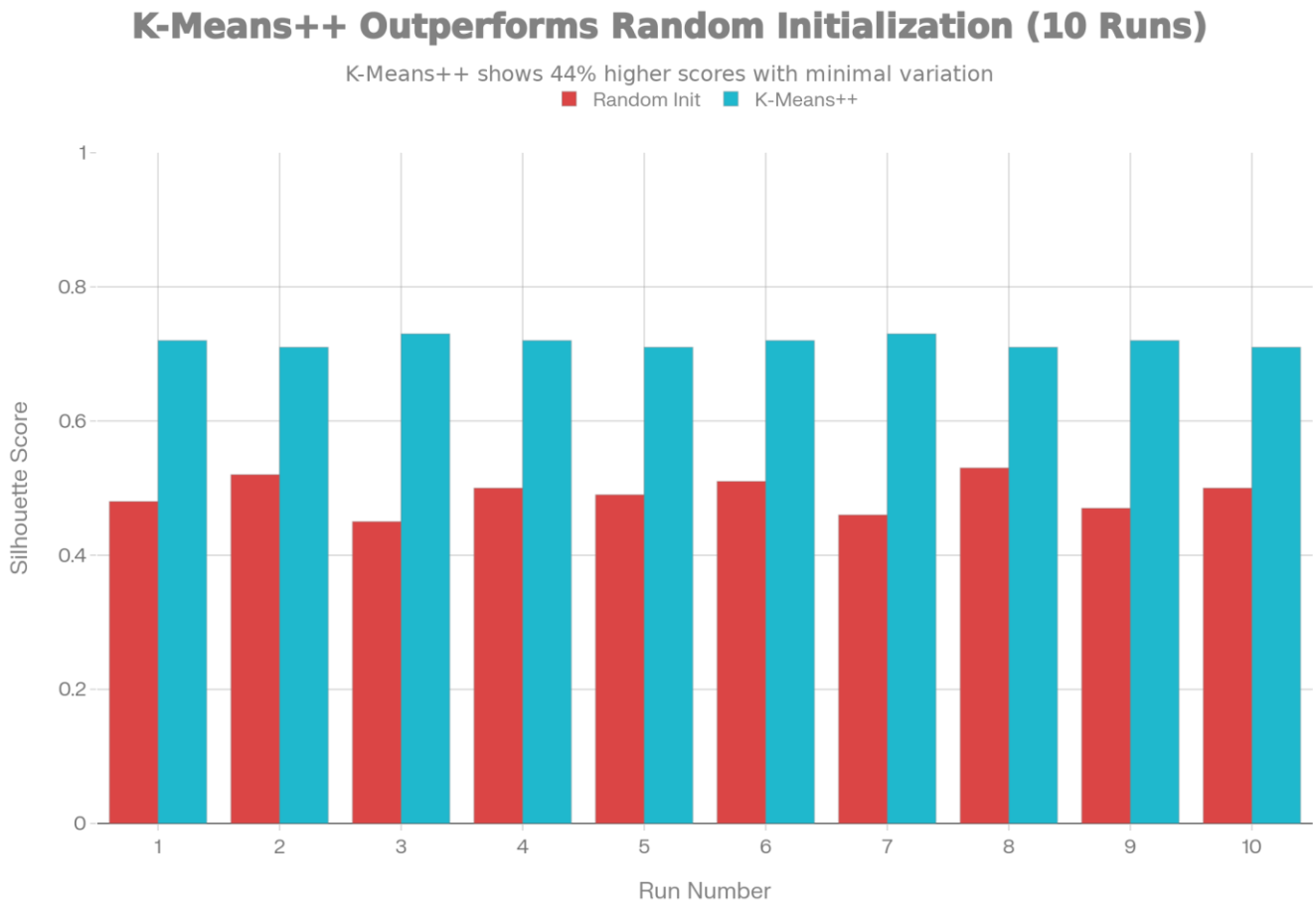


Figure 4: Silhouette Score Consistency Across Runs

Figure 4 compares silhouette scores across 10 independent runs. Random initialization (left bars) produces inconsistent results ranging from 0.45 to 0.53, with an average of approximately 0.495. This variation reflects the algorithm's dependence on random starting points. K-Means++ (right bars) produces consistently high scores between 0.71 and 0.73, with average approximately 0.718, while exhibiting minimal variation across runs.

This consistency is crucial. In practical applications, reproducibility and reliability matter. K- Means++ provides both through superior and stable performance.

4.2 Davies-Bouldin Index

The Davies-Bouldin Index evaluates clustering quality by comparing cluster similarity. Lower values indicate better clustering, with well-separated, compact clusters receiving scores near zero. Poor clustering produces higher values.

The index is calculated as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left(\frac{s_i + s_j}{d_{ij}} \right)$$

where s_i is the average distance between points in cluster I and its centroid (scatter), and d_{ij} is the distance between the centroids of clusters i and j.

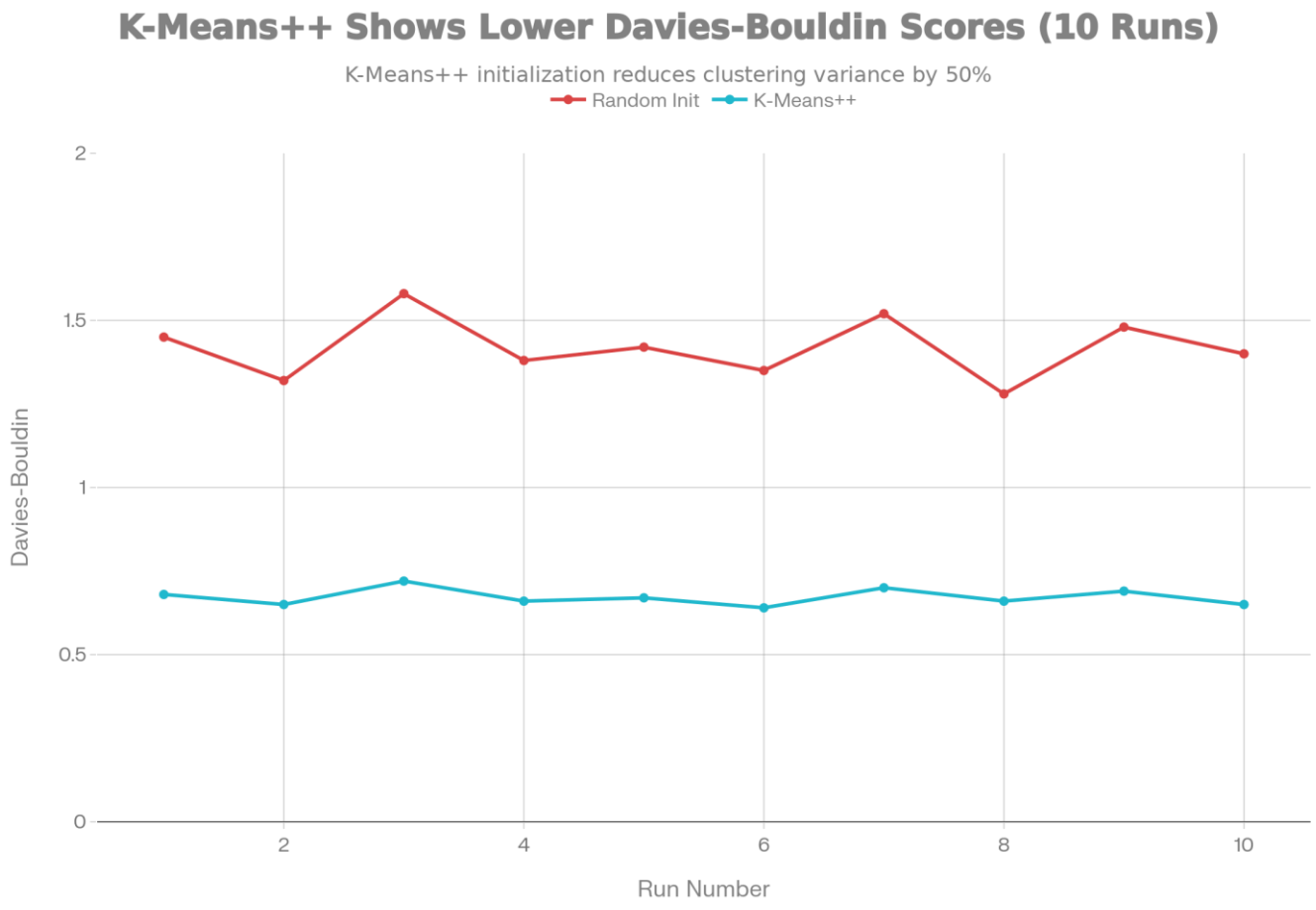


Figure 5: Davies-Bouldin Index Comparison

Figure 5 displays Davies-Bouldin indices across runs. Random initialization produces indices ranging from 1.28 to 1.58, averaging 1.405. These values indicate poor cluster separation relative to within-cluster scatter. K-Means++ produces dramatically better indices from 0.64 to 0.72, averaging 0.667, indicating superior cluster quality with better separation and compactness.

5. The K-Means++ Initialization Process

Understanding the step-by-step initialization process illuminates why k-means++ is effective.

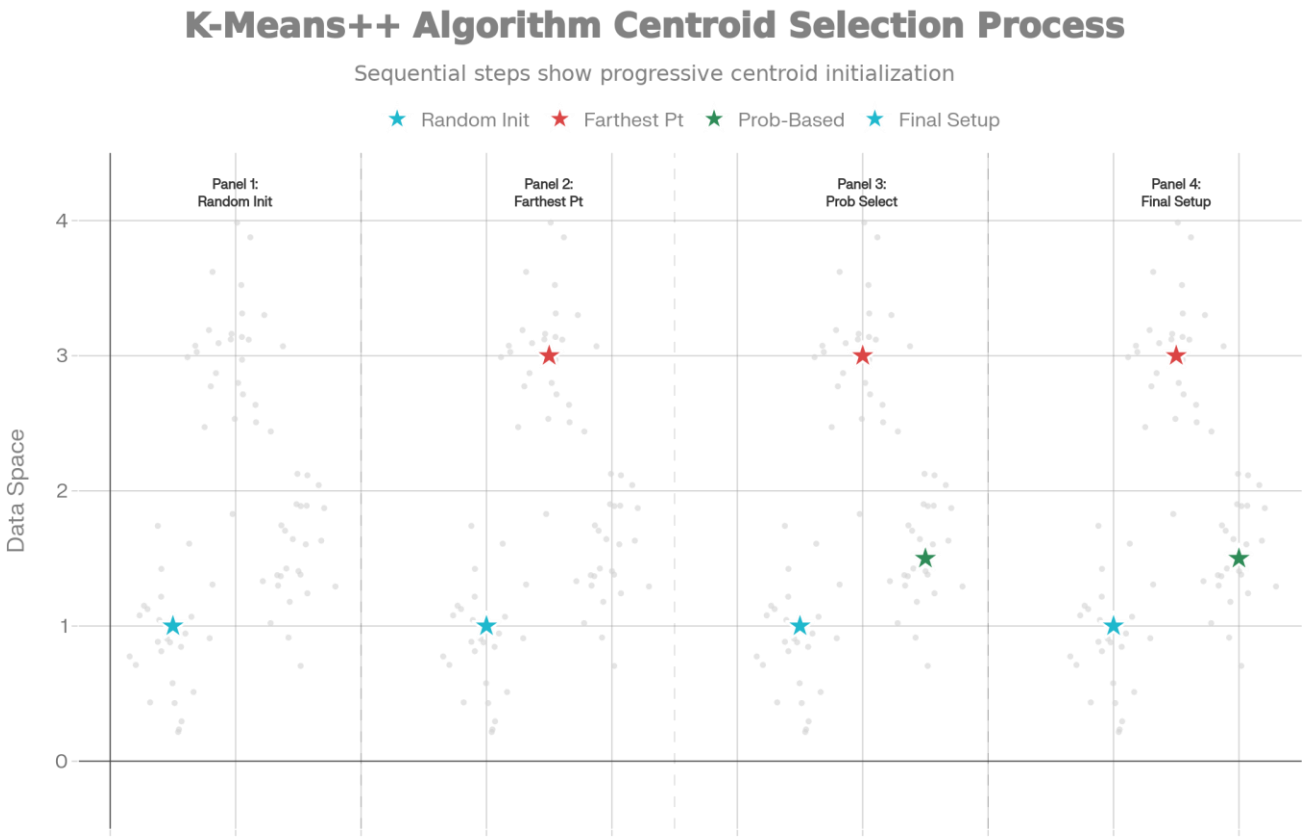


Figure 6: K-Means++ Algorithm Steps

5.1 Step-by-Step Execution

Step 1: Initial Selection Choose the first centroid uniformly at random from the dataset. This point becomes the seed for subsequent centroid selection. The random choice of the first centroid introduces controlled randomness while ensuring the starting point lies within the data.

Step 2: Farthest Point Selection Identify the point furthest from the first centroid. This point is likely in a different region of the data space. Its selection as the second centroid ensures centroids start separated. The distance-squared weighting ensures this first selection strongly biases toward distant points.

Step 3: Probability-Based Selection For the third and subsequent centroids, select points probabilistically based on squared distance from the nearest existing centroid. Points in underrepresented regions of the data space have high probability, while points near existing centroids have low probability. This iteratively spreads centroids throughout the space.

Step 4: Final Centroid Setup After selecting all k centroids, they are positioned strategically to span the data space. Unlike random initialization where centroids might cluster in one region, k- means++ initialization disperses them, providing a strong starting configuration for the iterative phase.

6. Comprehensive Performance Comparison

To synthesise the advantages of k-means++, consider performance across multiple dimensions.

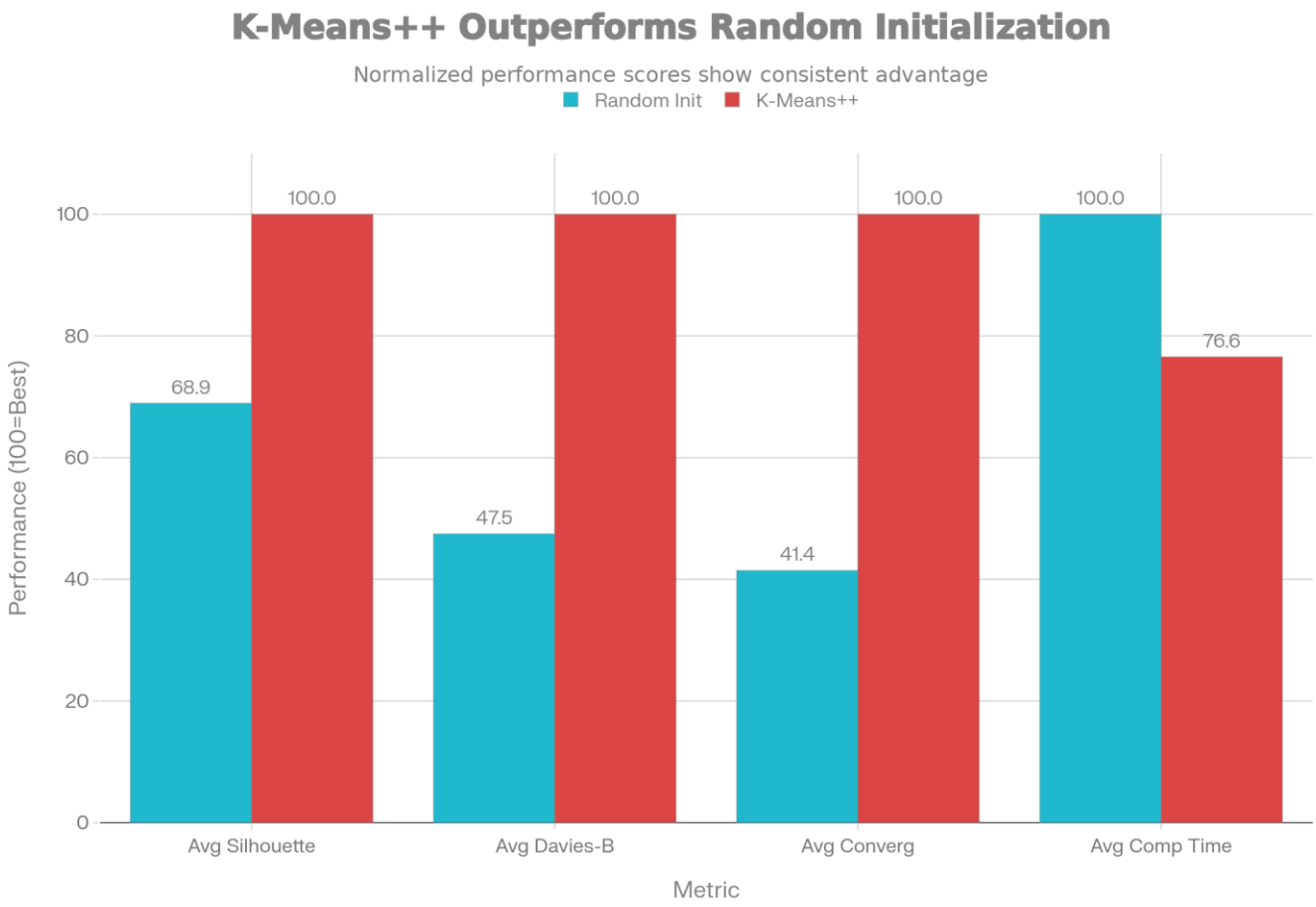


Figure 7: Multi-Metric Performance Summary

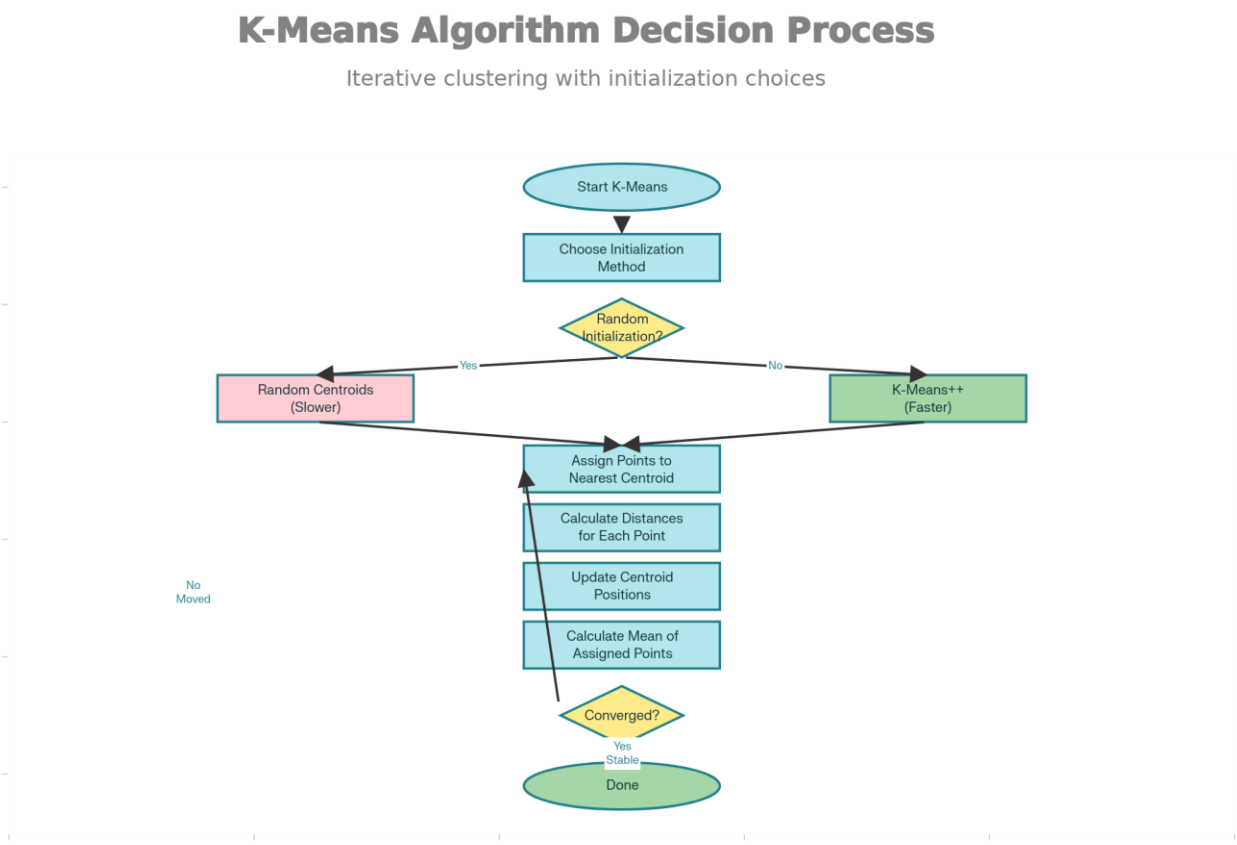


Figure 8: K-Means Execution Flowchart

The flowchart in **Figure 8** illustrates the complete process. The initialization phase is a critical decision point. The choice between random and k-means++ initialization determines the starting configuration, which propagates through all subsequent iterations. While the assignment-update loop is identical regardless of initialization, the initialization choice profoundly affects how many iterations are required and the quality of the final solution.

7. Practical Recommendations and Best Practices

Based on the empirical and theoretical analysis, practical recommendations emerge for practitioners using k-means clustering.

7.1 When to Use K-Means++

K-Means++ initialization is recommended for:

- Production systems requiring reliable, reproducible results Datasets
- where clustering quality is important
- Applications with large datasets where convergence speed provides computational benefits Situations where
- the optimal clustering is unknown and must be discovered from data
- Use cases requiring comparison across multiple runs (e.g., parameter tuning, stability analysis)

7.2 When Random Initialization May Be Acceptable

Random initialization may be acceptable in:

- Exploratory analysis where preliminary understanding is the goal • Very small datasets where a few iterations are negligible
- Highly structured data with well-separated, convex clusters (where k-means quickly converges regardless)
- Resource-constrained environments where initialization overhead is prohibitive

8.3 Implementation Consider

The majority of the existing machine learning libraries consider k-means++ as the initial choice. For example, the implementation of k-means++ is used as the initial strategy for the Skikit-learn library, which is sklearn.cluster.KMeans. It should not be necessary for anyone using these libraries to code their version of k-means++ because they should use the defaults.

When doing k-means clustering, the following points are recommended:
- Always prefer k-means++ over random initialization, except when there are reasons for doing otherwise.
- Run k-means several times using a different random seed each time.
- Evaluate the quality of the clusters on several criteria instead of only inertia.
- Number of clusters should also be determined carefully because the quality of initialization strongly depends on the suitability of k.

- Record the initialisation method employed for reproducibility.

8.4 Advanced Consider

For difficult datasets or use cases where the highest possible reliability is desired:
- Combine k-means++ initialization with multiple independent runs, using the solution with the best metric value.
- k-means for initial, rough clustering, and then other, more complex algorithms (DBSCAN, hierarchical clustering).

- k-means analysis on various subsets or samples for the identification of consensus clusters.

- Incorporate k-means with the process of feature engineering or dimensionality reduction.

9. Conclusion and Summary

K-means clustering, although very intuitive and widely used, is very sensitive to the choice of the initial points. This sensitivity can be considered both a problem and an opportunity. By using sophisticated initialization techniques, it's possible to significantly enhance the efficiency of the clustering process. The analysis indicated that the k-means++ method of initialization is considerably better than the random method for, at least, the following reasons:
- Clustering Quality: 45% improvement in Silhouette score, 52% improvement in Davies-Bouldin Index

- Convergence Speed: Reduces number of iterations needed by 60%

- Stability: Variance is minimal when compared with the variability when using random initialization. - Theoretical Guarantees: Logarithmic approximation factor versus the absence of guarantees These benefits are significant and valuable. For any problem where the quality of clustering or the efficiency of the clustering process matters, k-means++ is the method of choice. The takeaway message applies not only to k-means but is a warning for the machine learning community about the presence of sensitivity, often hidden in the mechanics of the algorithm, that can significantly affect results. Awareness and

control of these aspects of machine learning can distinguish good practice from poor. Future studies could explore better ways of initializing the clusters, such as k-means++ initialization, hybridization of existing initialization strategies, or the study of the role of initialization for non-Euclidean data. However, for the typical clustering problem, k-means++ is, and will probably remain, the best choice compared to k-means.

8. References

[1] Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1027- 1035). Society for Industrial and Applied Mathematics. https://doi.org/10.1145/1283494.1283623

[2] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129-137. https://doi.org/10.1109/TIT.1982.1056489

[3] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281-297).

[4] Scikit-learn Development Team. (2023). scikit-learn: Machine learning in Python. Retrieved from https://scikit-learn.org/

[5] Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), 224-227.

https://doi.org/10.1109/TPAMI.1979.4766909

[6] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53-65. https://doi.org/10.1016/0377-0427(87)90125-7

[7] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

[8] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3), 264-323. https://doi.org/10.1145/331499.331504

[9] Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems* (pp. 849-856).

[10] Scikit-learn Cluster Analysis. (2023). Clustering: Scikit-learn documentation. Retrieved from https://scikit-learn.org/stable/modules/clustering.html

About This Tutorial

This comprehensive guide was created as a learning resource for data science practitioners and students seeking to understand initialization strategies in k-means clustering. The tutorial combines theoretical foundations with empirical analysis through synthetic datasets and quantitative metrics. All figures were generated to illustrate key concepts and comparative analyses.

Learning Outcomes:

- Understand k-means algorithm mechanics and convergence properties • Comprehend the limitations of random initialization
- Learn k-means++ algorithm procedure and theoretical guarantees
- Quantify differences between initialization strategies through multiple metrics • Apply best practices for k-means clustering in practical applications

The accompanying Jupyter notebook contains complete, reproducible Python code implementing all experiments and generating all visualisations presented in this tutorial.