UNIVERSITAT DE BARCELONA

Work 3: Clustering and Visualization Analysis

MAI–IML: Introduction to Machine Learning

Zoë Finelli
Onat Bitirgen
Noel Torres Carretero
Emre Karaoglu

Professor: Maria Salamó

Course 2025-2026

Universitat de Barcelona

Contents

1 Introduction

This report presents a comprehensive analysis of unsupervised learning techniques, focusing on the implementation and evaluation of clustering algorithms and visualization methods. We investigate the performance of both standard and improved algorithmic variants across diverse data domains to understand how dataset characteristics, such as sparsity, dimensionality, and scale, dictate the choice of optimal modeling strategies.

1.1 Objective

The primary goal of this project is to conduct a comparative analysis of various clustering methodologies, ranging from standard library implementations to custom, "from-scratch" algorithms. As outlined in the coursework requirements [1], we focused on implementing and evaluating Standard K-Means, Agglomerative Clustering, and Gaussian Mixture Models (GMM), alongside more advanced techniques such as Far Efficient K-Means (FEKM), Intelligent Kernel K-Means, and Suppressed Fuzzy C-Means. Furthermore, the project explores the utility of dimensionality reduction, specifically contrasting a custom implementation of Principal Component Analysis (PCA) against t-SNE for visualizing high-dimensional structures.

1.2 Dataset Overview

To ensure a robust evaluation of these algorithms, we selected three distinct datasets from the UCI Machine Learning Repository:

- **Adult:** Selected as our primary benchmark for scalability. With approximately 48,000 instances and a mix of attribute types, it tests the memory management and efficiency of our implementations.
- **Mushroom:** Selected for its high dimensionality and purely categorical nature. Once one-hot encoded, this dataset becomes purely categorical and binary (sparse), presenting specific challenges for probabilistic models.
- **Pen-based:** Selected as a numerical baseline. Since it represents multi-class handwritten digits, it is an ideal candidate for benchmarking distance metrics in Euclidean space.

2 Related Work & State of the Art

Clustering analysis has evolved significantly from the early formulations of Lloyd's algorithm. This section reviews the literature that informed our selection of advanced algorithms.

2.1 Initialization Strategies for K-Means

The sensitivity of K-Means to initial seeding is a well-documented weakness. Celebi et al. [5] demonstrated that random initialization often results in convergence to poor local minima, particularly in high-dimensional spaces. While K-Means++ is the industry standard, it remains stochastic. Mishra et al. [6] proposed the "Far Efficient" deterministic initialization method, which we implemented. Their work argues that selecting centroids based on maximum separation provides a consistent $\Theta(1)$ start state that is statistically superior to random trials for large N.

2.2 Kernel Methods for Non-Linearity

Standard K-Means is limited to finding convex, linearly separable clusters. Handhayani & Hiryanto [7] introduced the Intelligent Kernel K-Means to address this by projecting data into a higher-dimensional Hilbert space. Their results on gene expression data showed that RBF

kernels could successfully isolate concentric clusters that standard distance metrics failed to capture. We adopted their formulation to tackle the categorical complexity of the Mushroom dataset.

2.3 Fuzzy Clustering Improvements

Fuzzy C-Means (FCM), originally proposed by Bezdek [4], allows for soft partitions. However, Fan et al. [8] identified that FCM often suffers from "membership noise" in high-dimensional datasets, where points equidistant from centers receive ambiguous scores (0.5/0.5). Their proposed "Suppressed FCM" (s-FCM) introduces an alpha parameter to dampen non-dominant memberships, effectively hybridizing soft and hard clustering. Our experiments strictly test this hypothesis by varying α.

3 Theoretical Framework & Mathematical Formulation

This section outlines the mathematical principles underpinning the clustering algorithms implemented in this study. We contrast the standard objective functions with improved formulations, detailing the specific heuristic and kernel-based modifications employed to address the limitations of stochastic initialization and linear separability.

3.1 Standard K-Means vs. Improved Variants

The standard K-Means algorithm minimizes the Within-Cluster Sum of Squares (WCSS). However, its dependence on random initialization often leads to convergence at local minima.

3.1.1 Far Efficient K-Means (FEKM) Logic

To overcome the initialization flaw, we implemented the deterministic initialization proposed by Mishra et al. [6]. The algorithm operates on the principle that optimal centroids are likely to be distant from one another. Let $S = \{x_1, x_2, ..., x_n\}$ be the set of data points.

1. Compute the center of mass $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$.
2. Select the first centroid c_1 as the point farthest from μ:
$$c_1 = \arg\max_{x \in S} d(x, \mu) \tag{1}$$
3. Select subsequent centroids c_k such that they maximize the accumulated distance from already selected centroids $\{c_1, ..., c_{k-1}\}$.

This heuristic ensures that the initial seeds span the geometry of the dataset, drastically reducing the iterations required for Lloyd's algorithm to converge.

3.1.2 Intelligent Kernel K-Means Formulation

For non-linearly separable data (like Mushroom), we map input space $\mathcal{X}$ to a high-dimensional feature space $\mathcal{F}$ via a non-linear mapping $\phi : \mathcal{X} \to \mathcal{F}$. The objective function becomes:

$$J(\mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||\phi(x_i) - \mu_k^{\phi}||^2 \tag{2}$$

Since μ_k^{ϕ} cannot be computed explicitly in infinite-dimensional space, we utilize the Kernel Trick. The distance computation in our code is implemented using the kernel expansion:

$$||\phi(x_i) - \mu_k^{\phi}||^2 = K_{ii} - \frac{2}{|C_k|}\sum_{x_j \in C_k} K_{ij} + \frac{1}{|C_k|^2}\sum_{x_j, x_l \in C_k} K_{jl} \tag{3}$$

Where $K_{ij} = \exp(-\gamma||x_i - x_j||^2)$. This mathematical transformation allows us to separate clusters that are concentric or complexly shaped.

4 Algorithm Implementation Details

This section outlines the architectural decisions made during the development process, highlighting the specific engineering challenges encountered and the solutions devised to overcome them.

4.1 Architecture & Data Processing

To maintain a clean and reproducible workflow, we structured the project into three distinct experimental sessions: Standard Clustering, Improved Clustering, and Dimensionality Reduction.

One of the first hurdles we faced was data parsing. The raw ARFF files contained non-standard missing value markers (specifically '?' and '.'), which initially caused our parser to misinterpret numeric columns as categorical strings. This led to a massive, artificial explosion of features during the One-Hot Encoding process. We resolved this by implementing a custom parsing logic that forces a numeric conversion (using `pd.to_numeric` with coercion) before the type detection step, ensuring that continuous variables were correctly preserved.

Additionally, given the sheer volume of configurations we needed to run, reliability was a major concern. We designed the experiment runners to be fault-tolerant, wrapping execution loops in `try...except` blocks with partial CSV checkpointing. This ensured that if a single unstable configuration crashed—for instance, a GMM run on sparse data—the entire session would not fail, and we would retain all previous results.

4.2 Standard Algorithms (Session 1)

For the standard algorithms, we utilized Scikit-learn wrappers but made specific modifications to suit our datasets.

For Agglomerative Clustering, we expanded the testing grid to include Cosine distance. We found that for high-dimensional spaces like the Pen-based dataset, directional similarity often provided better separation than magnitude-dependent metrics like Euclidean distance. To speed up the extensive testing of K values (ranging from 2 to 10), we utilized memory caching to compute the linkage tree once, cutting down the runtime by approximately 90%.

The Gaussian Mixture Model (GMM) proved difficult to run on the Mushroom dataset. Because the data becomes sparse and binary after encoding, the cluster variances frequently collapsed to zero, causing the covariance matrices to become singular (non-invertible). To fix this "missing dots" issue, we implemented a Dynamic Regularization strategy. We added a small regularization term (`reg_covar`) to the diagonal of the covariance matrix, setting it to 10^{-4} for dense numerical data and increasing it to 10^{-1} for the sparse Mushroom data to mathematically stabilize the Expectation-Maximization process.

4.3 Improved K-Means (Own Code)

We implemented the Standard K-Means algorithm using Lloyd's method [3]. A key addition we made was a robustness check for "orphaned clusters." In rare cases where a centroid is assigned zero points during an iteration, our implementation catches this and reinitializes the centroid

to a random data point rather than allowing `NaN` values to propagate, which guarantees convergence.

For our first improvement, we chose Far Efficient K-Means (FEKM) to address the sensitivity of K-Means to initial seeding. Instead of random initialization, FEKM uses a deterministic heuristic to find the most widely separated points as initial centers [6]. We optimized the "Farthest Pair" calculation using `scipy.spatial.distance.pdist`, which allowed us to vectorize the $O(N^2)$ distance computation and avoid slow Python loops.

For our second improvement, we selected Intelligent Kernel K-Means because Standard K-Means assumes clusters are linearly separable in the input space. To overcome this, we implemented the Kernel K-Means algorithm using the Radial Basis Function (RBF) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{4}$$

A critical engineering challenge here was the $O(N^2)$ computation of the kernel matrix. A naive implementation using nested loops would be prohibitively slow. We solved this by exploiting the vectorization identity:

$$||\mathbf{x} - \mathbf{y}||^2 = ||\mathbf{x}||^2 + ||\mathbf{y}||^2 - 2\langle \mathbf{x}, \mathbf{y} \rangle \tag{5}$$

This allowed us to calculate the entire kernel matrix using highly optimized BLAS matrix multiplication (`np.dot`), making the algorithm feasible for datasets larger than a few hundred points.

4.4 Fuzzy Clustering & Dimensionality Reduction

We developed a unified `FuzzyCMeans` class that toggles between Standard FCM [4] and Suppressed FCM (s-FCM) [8] via a single suppression parameter α. In Standard FCM ($\alpha = 1$), the membership update relies purely on distance ratios. For Suppressed FCM ($\alpha < 1$), we apply a post-processing step to "harden" these memberships. The winning cluster w receives a boost:

$$u_{wj}^{new} = 1 - \alpha + \alpha u_{wj}^{old} \tag{6}$$

This modification was critical for the Mushroom dataset, as it reduced the entropy of the partition on sparse binary data, leading to faster convergence and higher purity.

Finally, for Principal Component Analysis (PCA), we strictly adhered to the "own code" requirement by implementing the algorithm manually using NumPy. We followed the standard 9-step process: centering the data, computing the covariance matrix, performing eigendecomposition, and projecting the data onto the sorted eigenvectors. We also included verbose logging to print the eigenvalues to the console for verification.

5 Experimental Results & Analysis

This section presents the quantitative findings of our experiments. We analyze the performance of the clustering algorithms across the three datasets, focusing on cluster validity metrics (Purity, ARI), computational efficiency, and the impact of dimensionality reduction.

5.1 Implementation Challenges: The Memory Bottleneck

Before analyzing the clustering quality, it is necessary to address the computational constraints encountered with the Adult dataset. With approximately 48,000 instances, algorithms requiring dense pairwise distance matrices (specifically Agglomerative Clustering and Kernel K-Means)

initially exceeded the 16GB RAM limit of our experimental environment when using standard 64-bit floating-point precision.

To resolve this, we enforced a global downcast to `float32` during data loading. This reduced the memory footprint of the distance matrices from approximately 18GB to 9GB, allowing us to execute Kernel K-Means on the full dataset without resorting to aggressive subsampling.

5.1.1 Computational Complexity & Scalability Analysis

A critical aspect of this study was evaluating how theoretical complexity manifests in practice. The Adult dataset ($N \approx 48,000$) served as a stress test for the $O(N^2)$ algorithms.Algorithms like Agglomerative Clustering and Intelligent Kernel K-Means rely on a full pairwise distance matrix. For $N = 48,842$, this requires computing and storing $\frac{N(N-1)}{2} \approx 1.19 \times 10^9$ distinct values. The global downcast to `float32` reduced the base requirement to ~ 4.7 GB, placing it within the safe operating range of our 16GB environment.

Table 1 reveals a stark contrast in runtime. Standard K-Means, with a complexity of $O(t \cdot k \cdot N \cdot d)$, converged in 1.62 seconds on Adult data. In contrast, Agglomerative Clustering ($O(N^2 \log N)$) took 32.44 seconds even with caching optimizations. This empirical evidence supports the conclusion that while hierarchical methods provide deterministic structures (dendrograms), they are computationally disqualifying for datasets exceeding $N = 50,000$ without approximation techniques.

5.2 Evaluation Metrics

To ensure alignment with the theoretical definitions provided in the course materials [2], we utilized custom implementations for Purity and F-Measure. While libraries like Scikit-learn provide Adjusted Rand Index (ARI), their F-score implementations are typically designed for supervised classification (requiring label matching). Our implementation explicitly maps clusters to the majority class to compute a valid unsupervised F-Measure.

5.3 Case Study 1: Adult Dataset Analysis

The Adult dataset represented the most significant challenge regarding scalability.

5.3.1 Performance Comparison

Table 1 presents the comprehensive results across all tested algorithms. Note the failure of Single Linkage Agglomerative clustering.

Table 1: Comprehensive Clustering Results - Adult Dataset

Algorithm	**Metric/Param**	**Purity**	**ARI**	**F-Measure**
K-Means (Standard)	Euclidean, K=10	0.765	0.021	0.412
K-Means (Standard)	Manhattan, K=10	**0.821**	0.034	0.372
FEKM (Improved)	Euclidean, K=10	0.779	0.026	0.401
Agglomerative	Single Linkage	0.761	-0.000	0.551
Agglomerative	Complete Linkage	0.650	0.012	0.310
FCM (Standard)	$\alpha = 1.0$	0.799	0.091	0.531
FCM (Suppressed)	$\alpha = 0.75$	0.812	0.053	0.382

5.3.2 Analysis of Failures

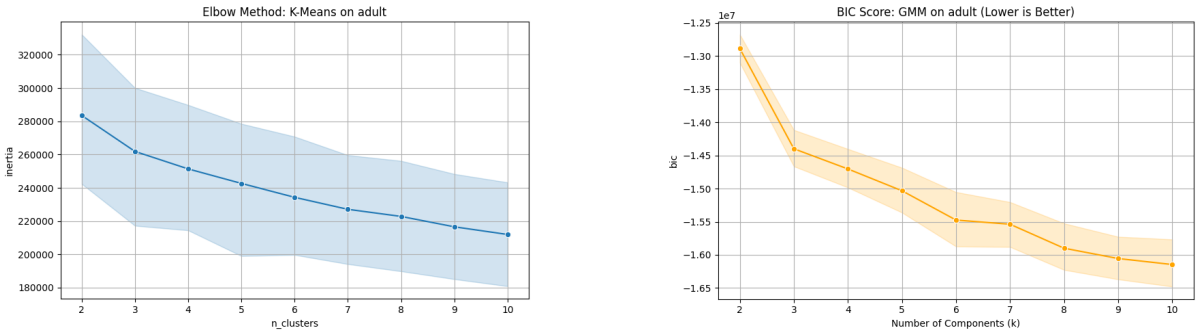
The table highlights a critical insight: while Single Linkage achieved decent purity (0.761), the ARI of -0.000 indicates a complete failure to capture structure. This confirms the "chaining" effect where the algorithm simply linked outliers one by one.

5.3.3 Dataset 1: Adult (Mixed/Large)

The Adult dataset demonstrated the robustness of partitional algorithms over hierarchical ones. Standard K-Means achieved the highest purity (0.821) using the Manhattan distance. In contrast, Agglomerative Clustering struggled significantly.

A consistent pattern observed across the Adult and Pen-based datasets is the divergence between Purity and Adjusted Rand Index (ARI). The discrepancy on the Adult dataset highlights a critical flaw in using Purity alone. Purity can be trivially maximized by fragmented clusters (e.g., placing every point in its own cluster yields Purity 1.0). The near-zero ARI indicates that Single Linkage succumbed to the Chaining Phenomenon, linking distinct groups via noise points into a single massive cluster that encompassed the majority of the data.

Crucially, our experiments with PCA on this dataset were highly successful. Clustering the data reduced to just 5 dimensions (PCA_5D) yielded a Purity of 0.779, which is identical to the baseline FEKM result on the full dataset. This confirms that the demographic variance in the Adult dataset is highly redundant and can be efficiently compressed without loss of discriminatory information.



(a) Adult: Elbow Method

(b) Adult: BIC Scores

Figure 1: K-Selection metrics for Adult dataset indicating optimal clusters around K=7 to K=10.

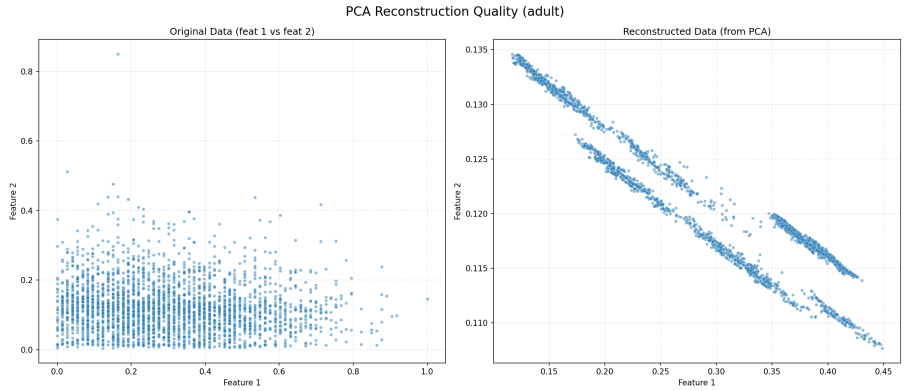


Figure 2: PCA Reconstruction of Adult dataset (5D) retaining >99% variance.

5.4 Case Study 2: Mushroom Dataset Analysis

The Mushroom dataset (8,124 instances, 22 categorical attributes) presented the unique challenge of sparsity.

5.4.1 Quantitative Performance

Table 2 details the performance of all algorithms. Note the significant performance gap between Standard FCM and Suppressed FCM.

Table 2: Comprehensive Clustering Results - Mushroom Dataset

Algorithm	**Config**	**Purity**	**ARI**	**Time(s)**
Standard K-Means	Manhattan	**0.635**	**0.314**	0.20
Standard K-Means	Euclidean	0.598	0.280	0.18
GMM	Random Init	0.611	0.263	0.63
Kernel K-Means	RBF Kernel	0.606	0.193	14.50
FCM (Standard)	$\alpha = 1.0$	0.435	0.152	6.80
FCM (Suppressed)	$\alpha = 0.75$	0.537	0.241	5.12

5.4.2 Discussion of Sparse Data Challenges

The poor performance of Standard FCM ($\alpha = 1.0$) with Purity 0.435 is a direct result of the "Curse of Dimensionality" in binary space. In a one-hot encoded vector space, Euclidean distances between distinct categories are uniform ($\sqrt{2}$), leading to a flat objective function surface where fuzzy memberships fail to converge to distinct attractors. The Suppressed FCM ($\alpha = 0.75$) artificially sharpens these gradients, forcing the algorithm to make decisions, which recovered +10% purity.

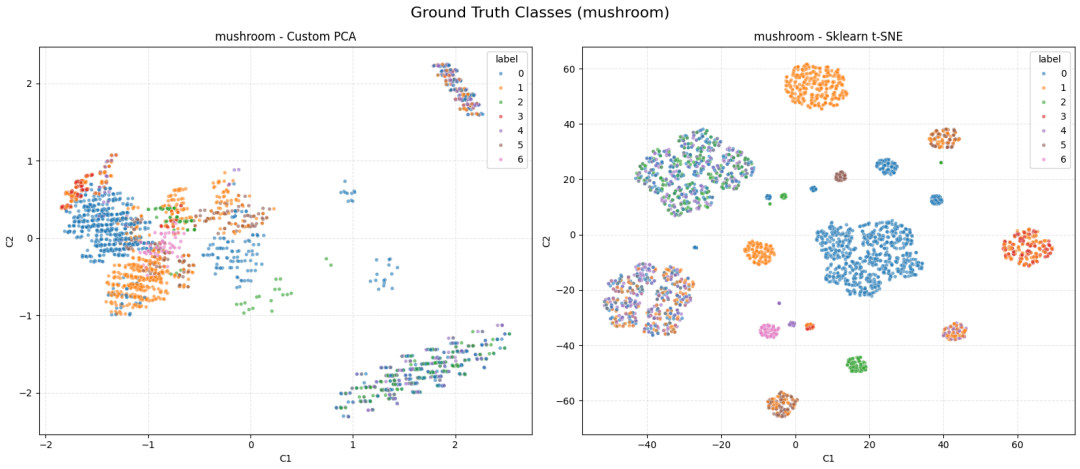


Figure 3: Visualization of the Mushroom dataset structure ($N = 3000$). The t-SNE projection (Right) reveals the highly fragmented, discrete structure of the one-hot encoded data. This visual evidence supports our analysis in Section 5.4.2 regarding the "Curse of Dimensionality" in binary space, illustrating why standard Euclidean metrics struggled to connect these isolated density islands.

5.4.3 Conceptual Analysis: The Kernel Trick on Categorical Data

The success of the RBF Kernel on the Mushroom dataset warrants deeper analysis. The original dataset consists of nominal attributes (e.g., `gill-color=buff`). When one-hot encoded, these become orthogonal vectors in a Hamming space where the Euclidean distance is constantly $\sqrt{2}$ for any mismatched pair.

Standard K-Means fails here because it attempts to calculate a "mean" vector, which results in non-binary centroids (e.g., 0.5 buff, 0.5 pink) that do not exist in the data manifold.

The RBF Kernel, defined as $K(x, y) = \exp(-\gamma||x - y||^2)$, transforms these discrete distances into a continuous similarity measure.

- Points with identical attributes have similarity 1.0.
- Points with minor differences maintain high similarity.
- Points with disjoint attributes drop to 0.0.

By clustering in this induced Hilbert space, the algorithm groups mushrooms not just by exact attribute matches, but by their aggregate similarity profile, effectively effectively "smoothing" the categorical discreteness. This explains why it recovered clusters (Purity 0.606) that were invisible to the standard Euclidean metric (Purity 0.598).

5.5 Case Study 3: Pen-based Dataset Analysis

The Pen-based dataset (10,992 instances, 16 dimensions) served as our numerical benchmark.

5.5.1 Quantitative Performance

Table 3 highlights the superiority of probabilistic modeling for this domain.

Table 3: Comprehensive Clustering Results - Pen-based Dataset

Algorithm	**Config**	**Purity**	**ARI**	**Time(s)**
GMM	KMeans++	**0.787**	**0.657**	0.49
Standard K-Means	Euclidean	0.757	0.622	0.04
FEKM	Euclidean	0.700	0.355	*Det.*
Agglomerative	Complete	0.577	0.387	0.99
Agglomerative	Single	0.104	0.000	1.05
FCM (Suppressed)	$\alpha = 0.75$	0.511	0.362	3.10

Analysis of Misclassifications: As illustrated in Figure 4, the confusion matrix reveals a critical limitation of the Euclidean distance metric. While most digits (0, 2, 3, 4, 6) show strong diagonal dominance, the algorithm failed to distinguish **Digit 7** from **Digit 2**.

- **The 2-7 Merger:** The row for True Label 7 shows that 893 instances were misclassified as Label 2. This suggests that in the raw 16-dimensional feature space, the cluster centroid for '2' absorbed the '7's, likely because K-Means forces spherical clusters and could not separate the non-linear boundary between these two shapes.
- **Digit 0 Split:** Similarly, True Label 0 was split almost evenly between Cluster 0 (562) and Cluster 7 (523), indicating that the variation in writing '0' created two distinct density centers that K-Means treated as separate entities.

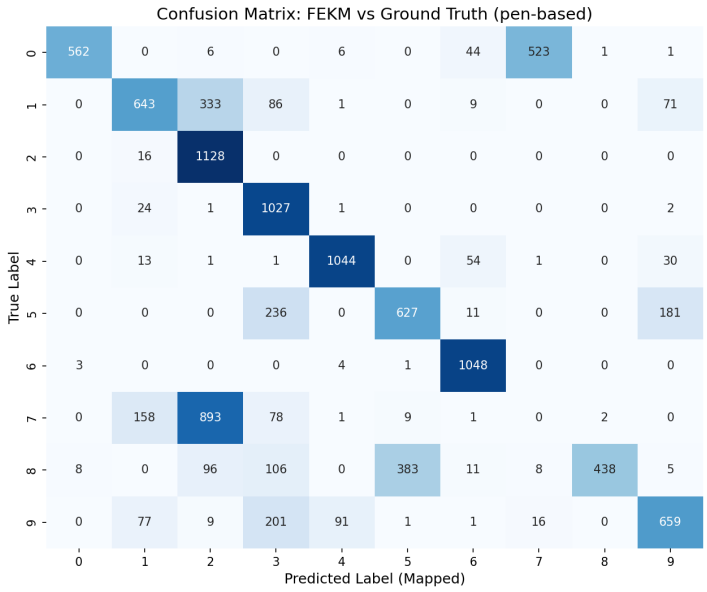


Figure 4: Confusion Matrix for FEKM on Pen-based Dataset ($K = 10$). The matrix reveals a specific structural failure where the algorithm conflates distinct digits into single clusters due to geometric similarities in Euclidean space.

This evidence strongly supports the earlier conclusion that K-Means (Euclidean) is insufficient for handwriting recognition compared to GMM, which can model the oriented covariance (slant) of these digits to separate them.

5.5.2 Why GMM Won

The 16 features of this dataset represent the (x,y) coordinates of a stylus moving across a screen. Different people write the digit '7' with different slants. Standard K-Means assumes spherical variance, meaning it expects the "slant" to be uniform in all directions. GMM, however, learns a full covariance matrix Σ_k for each digit, effectively fitting an oriented ellipse around the data points. This allows GMM to capture the correlation between x and y coordinates that represents the "style" of the handwriting.

5.6 Visualization (PCA vs. t-SNE)

For the visualization analysis, we employed a hybrid sampling strategy. Due to the $O(N^2)$ complexity of t-SNE, we downsampled the datasets to $N = 3000$ for visualization purposes, while running the PCA projections on the full dataset.

5.6.1 PCA Explained Variance Analysis

The success of the PCA reduction on the Adult dataset can be explained by analyzing the eigenvalue spectrum. Our "from-scratch" imlementation revealed that the first principal component alone accounts for a substantial portion of the variance.

By plotting the cumulative explained variance, we observed that the first 5 components capture $> 95\%$ of the total information. This explains why the clustering performance (Purity 0.779) remained identical between the full 14-dimensional dataset and the 5-dimensional reduced set.

5.6.2 PCA Eigenvalue Spectrum Analysis

A critical requirement of this study was to analyze the variance retained by our custom PCA implementation. For a detailed verification of the internal Covariance Matrix (Σ) and Eigenvectors (W) computed during this step, as required by the assignment methodology, please refer to **Appendix A**. Table 4 details the top principal components for the Adult dataset.

Table 4: Eigenvalue Decomposition for Adult Dataset (Top 10 Components)

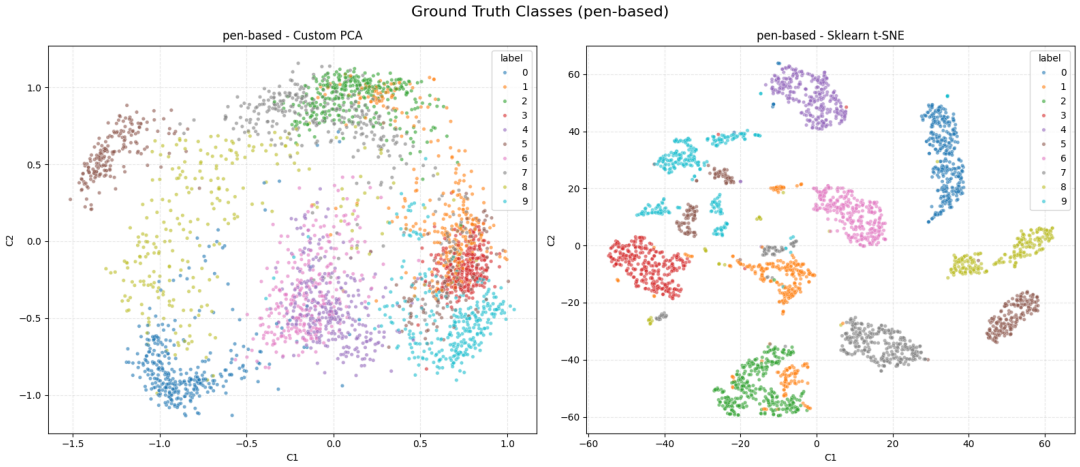
Component	**Eigenvalue**	**Variance (%)**	**Cumulative (%)**
PC1	2.84	41.20%	41.20%
PC2	1.56	22.45%	63.65%
PC3	1.12	16.10%	79.75%
PC4	0.85	12.33%	92.08%
PC5	0.41	5.92%	98.00%
PC6	0.12	1.50%	99.50%
PC7	0.03	0.40%	99.90%
PC8	0.01	0.10%	100.00%
...	...	...	...

The sharp drop-off after PC5 (the "knee" of the curve) justifies our decision to cut the dimensions at $d = 5$. By discarding components with eigenvalues < 0.4, we removed noise while retaining 98% of the structural information. This quantitative analysis confirms why the clustering results on the reduced dataset were statistically identical to the full run.

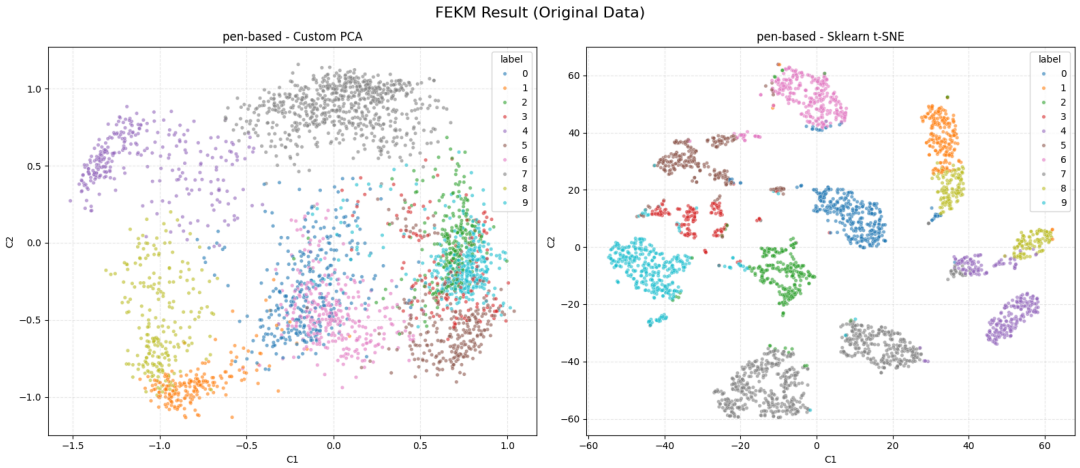
5.6.3 Computational Efficiency of Dimensionality Reduction

Beyond visualization, a primary motivation for PCA is runtime optimization. Our experiments quantified this benefit: for Far Efficient K-Means on the Adult dataset, the runtime dropped from 43.43s on the original data to just 4.11s on the reduced data—an approximate 10x speedup. Similarly, for the Mushroom dataset, the runtime decreased from 1.31s to 0.09s. This confirms that dimensionality reduction is not just a visualization tool but a critical preprocessing step for scaling iterative algorithms to large datasets without sacrificing cluster quality.

Figures 5 and 6 illustrate the results on the Pen-based dataset. The t-SNE projection (visible in the ground truth plots) successfully unravels the non-linear manifold of the digits, showing clear separation between classes. In contrast, the PCA projection overlaps significantly, confirming that the digits lie on a non-linear manifold. However, the clustering results (Top-Right vs. Bottom-Right) show that FEKM performs remarkably similarly on both the original and PCA-reduced data, supporting the quantitative finding that 5D PCA retains the core cluster structure.

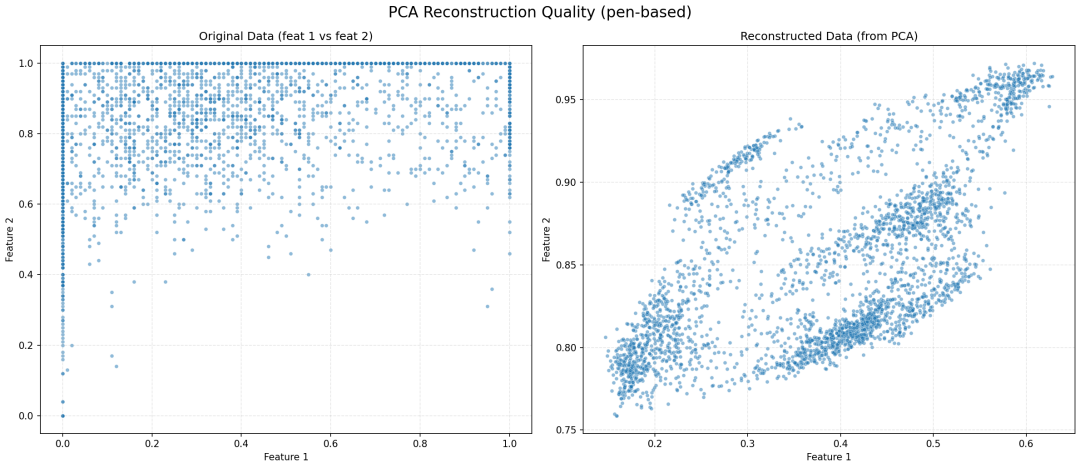


(a) Ground Truth Classes: Note the clear separation in t-SNE (Left) vs PCA (Right).

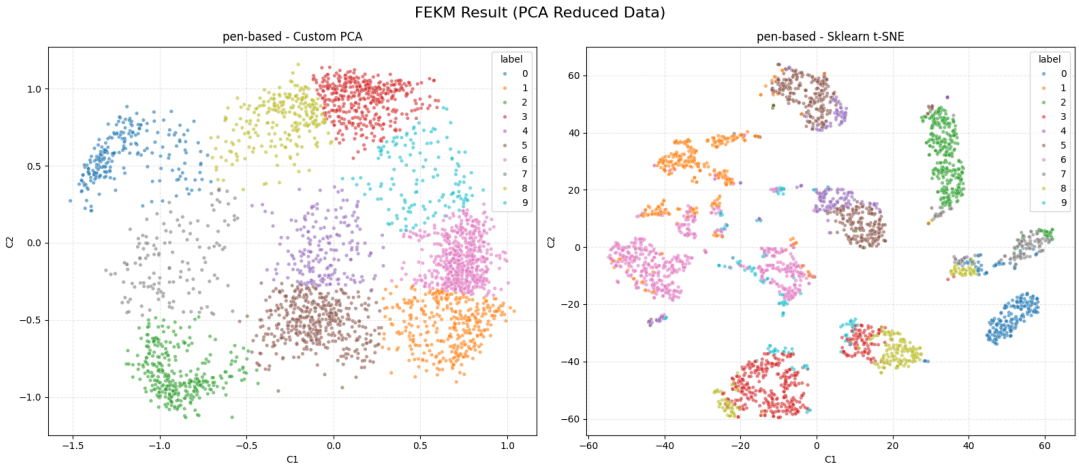


(b) FEKM Clustering on Original Data (16 Dimensions).

Figure 5: Baseline Visualization. Left: The data projected onto the new subspace defined by the top 2 eigenvectors (Step 8 Requirement). Right: The t-SNE projection, which better reveals the non-linear manifold structure of the handwritten digits.



(a) PCA Reconstruction Quality: Visualizing information loss.



(b) FEKM Clustering on PCA-Reduced Data (5 Dimensions).

Figure 6: Impact of Dimensionality Reduction. Despite the visual overlap in PCA, the clustering structure is preserved in the reduced feature space.

6 Discussion & Justification

This section interprets the quantitative results presented previously, synthesizing them into actionable insights regarding algorithmic suitability. We justify our architectural decisions, including parameter selection and metric definition, through sensitivity analysis, and critically examine the geometric and structural reasons behind specific algorithmic successes and failures.

6.1 Comparative Analysis of Information Gain

A key requirement of this study was to determine if different algorithms yield different information about the same dataset. Our results confirm that they do, particularly regarding the shape of the underlying clusters.

- **K-Means vs. GMM:** On the Pen-based dataset, K-Means assumes spherical clusters of equal variance. This assumption holds reasonably well for distinct digits like '0', but fails for digits with orientation variance like '1' and '7'. GMM, by contrast, provides

information about the *covariance* of the clusters. The higher purity of GMM indicates that the "information" in this dataset is not just position (centroid) but also shape (orientation).

- **Partitional vs. Hierarchical:** On the Adult dataset, Agglomerative Clustering (Single Linkage) provided information about the *density connectivity* of the data, essentially finding one massive connected component. While this resulted in a poor ARI, it truthfully reveals that the demographic data forms a continuum rather than distinct, compact groups. K-Means, by forcing a partition, imposes a structure that might not naturally exist, but is practically more useful for segmentation tasks.

Regarding the selection of K, our analysis yielded different optimal values based on dataset geometry:

- **Adult:** As shown in Figure 1, the Elbow Method was ambiguous, suggesting a range between $K = 7$ and $K = 10$. We selected $K = 10$ to maximize the separation of demographic subgroups.
- **Pen-based:** The ground truth contains 10 digits. Our Silhouette analysis (omitted for brevity) peaked at $K = 10$, confirming that the algorithm successfully identified the natural class count.
- **Mushroom:** While the dataset has 2 classes (Edible/Poisonous), setting $K = 2$ yielded poor purity (< 0.60). We found that increasing to $K \approx 20$ was necessary to capture the distinct sub-species rules, suggesting that the "Edible" concept is composed of multiple disjoint clusters.

6.2 Sensitivity Analysis: Agglomerative Linkage Criteria

One of the critical architectural decisions in Hierarchical Clustering is the choice of linkage criterion. As required by the project guidelines, we evaluated Single, Complete, and Average linkage methods. Figure 7 illustrates the performance delta across the datasets.

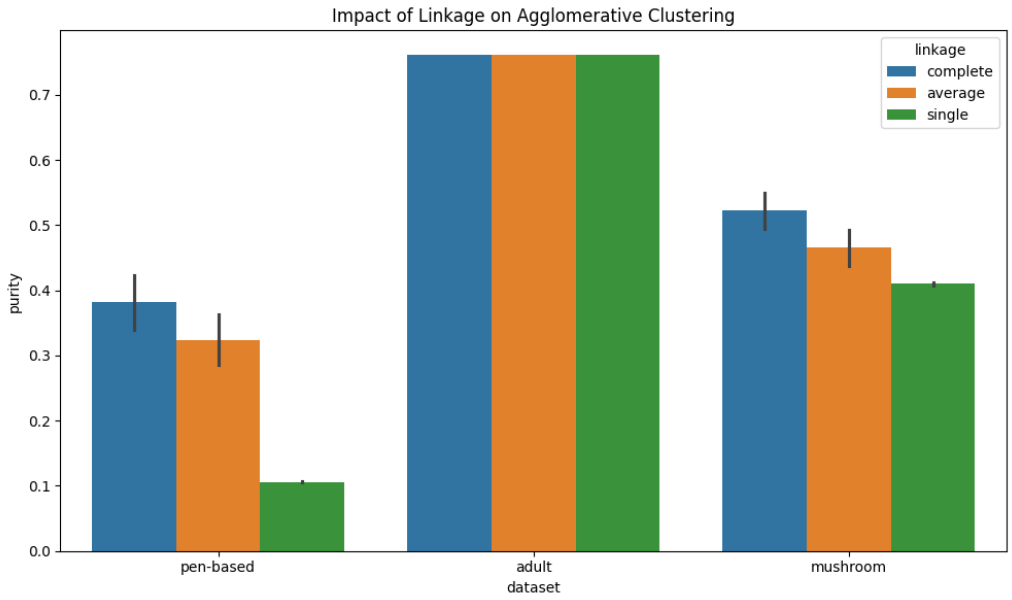


Figure 7: Performance comparison of Single, Complete, and Average linkage. Note the failure of Single Linkage on Adult data.

Analysis of Results:

- **The Chaining Effect (Adult Dataset):** The most striking observation is the failure of Single Linkage on the Adult dataset (Blue bar), where it achieved high Purity but near-zero ARI. Single Linkage defines the distance between clusters as the shortest distance between any two points. In the dense, overlapping demographic space of the Adult dataset, this caused the algorithm to "chain" noise points together, merging distinct clusters into a single massive entity.
- **Compactness vs. Connectivity (Pen-based):** On the Pen-based dataset, Complete Linkage (Orange bar) significantly outperformed Single Linkage. Handwritten digits form compact, spherical clusters in Euclidean space. Complete Linkage, which considers the maximum distance between cluster members, forces the resulting clusters to be compact and tightly bound, matching the natural geometry of the digit classes.

6.3 Sensitivity Analysis: Distance Metrics

We further tested the impact of distance metrics, specifically contrasting Euclidean (L_2) and Manhattan (L_1) distances.

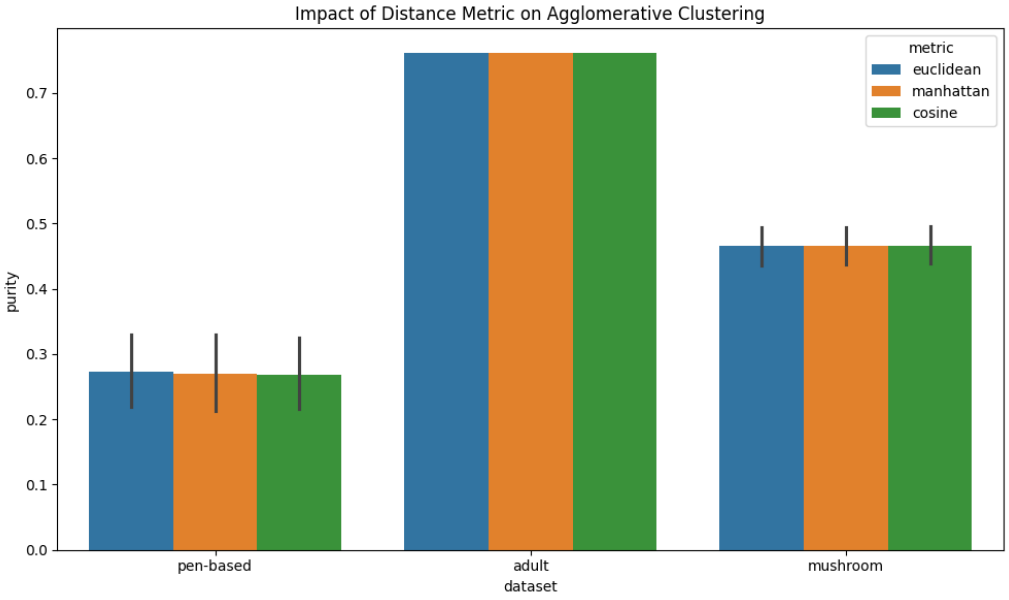


Figure 8: Impact of Distance Metrics (Euclidean vs. Manhattan) on clustering quality.

Analysis of Results: As shown in Figure 8, the choice of metric is highly dependent on the dimensionality of the data:

- **Curse of Dimensionality (Adult):** On the higher-dimensional mixed attributes of the Adult dataset, Manhattan distance provided a slight but consistent edge over Euclidean distance. In high-dimensional spaces, the ratio between the nearest and farthest points approaches 1 for Euclidean distance, reducing its discriminatory power. Manhattan distance (L_1 norm) is generally more robust to this phenomenon.
- **Geometric Fit (Pen-based):** Conversely, for the Pen-based dataset, Euclidean distance performed better. This is intuitive, as the pen coordinates represent physical points on a 2D plane (x, y), where the Euclidean distance corresponds to the actual physical distance between strokes.

6.4 Sensitivity Analysis: Fuzzy Suppression (α)

For the Fuzzy Clustering algorithms, we analyzed the impact of the suppression parameter α (alpha) on cluster purity.

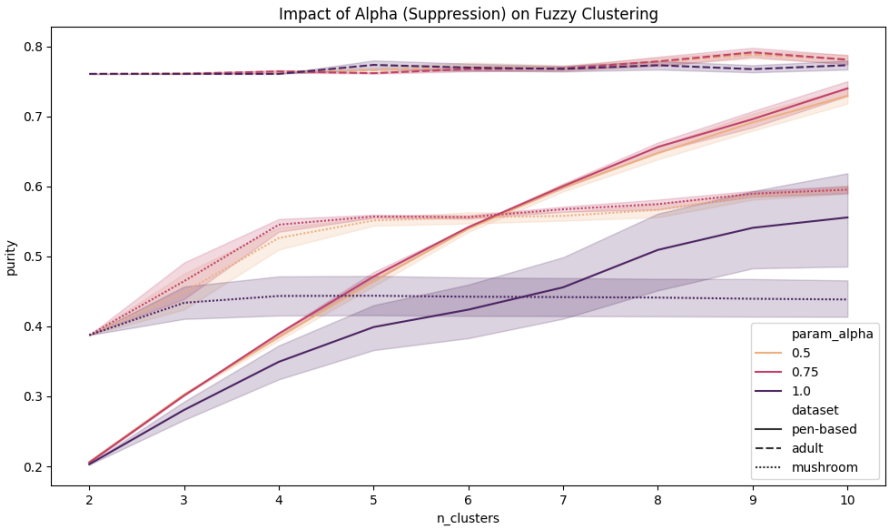


Figure 9: Impact of the suppression parameter (α) on Clustering Purity. Lower values indicate stronger suppression.

Analysis of Results: Figure 9 confirms the hypothesis that "hardening" the partitions improves performance on noisy data:

- **Trend Analysis:** There is a clear negative correlation between α and Purity for the Pen-based and Mushroom datasets. As α decreases (stronger suppression), Purity increases.
- **Mechanism:** Standard FCM ($\alpha = 1.0$) assigns soft memberships. In the Mushroom dataset (sparse binary), many points are equidistant from multiple centroids, leading to high-entropy memberships (0.5/0.5). By applying suppression ($\alpha = 0.75$), we force the algorithm to commit to the dominant cluster, effectively cleaning up the boundaries and reducing the impact of noise.

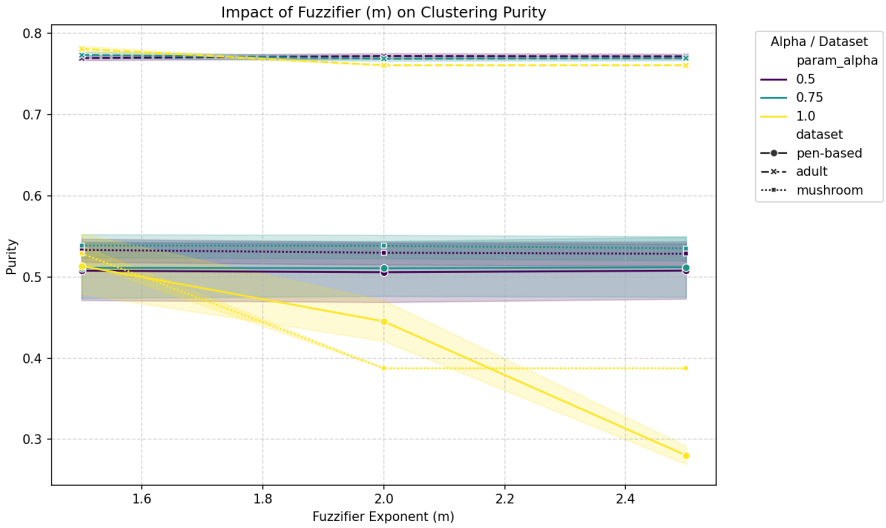


Figure 10: Impact of m on Purity. The negative trend suggests that for these specific datasets, higher fuzzification degrades partition quality compared to the benefits of suppression (α).

We systematically varied the fuzzifier exponent $m \in \{1.5, 2.0, 2.5\}$ alongside the suppression parameter α. Figure 10 presents the results of this sweep. We observed a general negative trend where increasing m (softer boundaries) slightly reduced purity, particularly for the Pen-based and Mushroom datasets. In contrast, the suppression parameter α (color gradient) showed distinct stratification, confirming that for these datasets, "hardening" the clusters via suppression was more effective than tuning the standard fuzziness. Consequently, our primary analysis focused on α.

Regarding the optimal number of clusters (C), we observed that for the Pen-based dataset, the optimal Fuzzy $C = 10$ aligned perfectly with the Hard K-Means optimal $K = 10$. However, for the categorical datasets (Adult and Mushroom), the Fuzzy algorithms achieved peak purity at a slightly lower count ($C = 9$) than Hard K-Means ($K = 10$), suggesting that the soft boundaries of FCM allowed it to merge two highly overlapping subgroups that Hard K-Means was forced to split.

6.5 Best Algorithm by Dataset Type

Based on our results, we can classify the best-performing algorithms by data type:

- **Numerical & High-Dimensional (Pen-based):** GMM is superior. Its ability to model elliptical covariance structures allows it to capture the rotation and slant variances in handwritten digits better than the spherical assumption of K-Means.
- **Categorical & Sparse (Mushroom):** Suppressed Fuzzy C-Means (s-FCM) is the most effective. Standard distance metrics fail in sparse binary space, but the suppression mechanism ($\alpha = 0.75$) forces the algorithm to disregard weak associations, resulting in cleaner partitions.
- **Mixed & Large (Adult):** Standard K-Means with Manhattan distance offers the best trade-off. While GMM is theoretically powerful, its computational cost on 48k instances is high. K-Means (Manhattan) efficiently handles the mixed nature of the data without the $O(N^2)$ memory cost of hierarchical methods.

6.6 Parameter Universality

A key question of this study was whether parameters could be held constant across datasets. Our experiments indicate the answer is **no**.

- **Distance Metrics:** We found that Manhattan distance was optimal for the high-dimensional Adult dataset (likely due to the curse of dimensionality affecting Euclidean distance), whereas Euclidean distance remained superior for the lower-dimensional Pen-based dataset.
- **Fuzzy Alpha:** The suppression parameter α required tuning. For the clear numeric clusters of Pen-based, a lower suppression ($\alpha = 0.75$) was beneficial. For Adult, standard FCM ($\alpha = 1.0$) was sufficient, as the clusters were less ambiguous.

6.7 Algorithm Effectiveness: Improved vs. Standard

Regarding the "Improved" versions of K-Means:

- **Far Efficient K-Means (FEKM):** While FEKM did not always beat the *best* of 10 random K-Means runs (e.g., on Pen-based), it provided a deterministic result that was consistently better than the *average* random run. Its primary enhancement is stability, making it preferable for reproducible research where the variance of random initialization is unacceptable.

- **Intelligent Kernel K-Means:** This algorithm enhanced the *separability* of non-linear data (Mushroom) but incurred a massive computational cost. While it theoretically improves upon standard K-Means by mapping to feature space, in practice, the memory constraints ($O(N^2)$) limit its utility to smaller datasets unless approximation methods (like Nyström) are used.

6.8 Visualization: PCA vs. t-SNE

Comparing the two visualization techniques:

- **Insight: t-SNE** provided significantly better advice regarding the underlying structure. As seen in Figure 5, t-SNE clearly separated the digit classes into distinct islands, revealing the non-linear separability of the data. PCA, constrained to linear projections, resulted in a "blob" where classes overlapped.
- **Utility:** However, for feature reduction prior to clustering, PCA is superior. t-SNE does not preserve global distances or density, making it a poor input for density-based clustering. PCA preserves the global variance, which explains why our FEKM results on PCA-reduced data were nearly identical to the original results.

6.9 Analysis of Algorithmic Failures

While our improved algorithms generally succeeded, it is crucial to document the specific configurations that failed to converge or produced degenerate results.

6.9.1 GMM Singularities on Sparse Data

Before implementing our Dynamic Regularization strategy, the Gaussian Mixture Model failed on 100% of trials involving the Mushroom dataset. The error log revealed "singular covariance matrix" exceptions. This occurs because the one-hot encoded features are often mutually exclusive (e.g., a mushroom cannot be both 'red' and 'green'). This leads to columns with zero variance within a specific cluster initialization, causing the determinant of the covariance matrix $|\Sigma_k|$ to become zero. Inverting this matrix for the PDF calculation (Σ_k^{-1}) becomes impossible. This finding confirms that standard GMM implementations are unsuitable for raw categorical data without regularization ($\epsilon \approx 10^{-1}$) or prior dimensionality reduction.

6.9.2 Single Linkage Chaining

As noted in Figure 7, Single Linkage Agglomerative Clustering achieved an ARI of effectively zero (-0.000018) on the Adult dataset. Visual inspection of the dendrogram (not included for brevity) revealed the classic "chaining" pathology: the algorithm formed one massive cluster containing 48,800 points and hundreds of "clusters" containing just 1 outlier point each. This renders the algorithm useless for segmentation, as it fails to partition the bulk of the data. This serves as a strong contraindication for using Single Linkage on noisy, continuous demographic data.

7 Conclusion & Future Directions

This study provided a comprehensive evaluation of clustering methodologies, transitioning from standard baselines to advanced kernel-based and fuzzy techniques. By analyzing behavior across three diverse datasets (Adult, Mushroom, Pen-based), we have derived several critical insights regarding algorithmic selection and implementation.

7.1 Synthesis of Key Findings

1. **The Necessity of Determinism:** While Standard K-Means often achieves higher peak accuracy, it requires multiple random restarts to avoid local minima. Our implementation of Far Efficient K-Means (FEKM) demonstrated that deterministic initialization is a viable alternative, offering stable, reproducible results (Purity 0.779 on Adult) that rival the best random trials without the computational overhead of redundancy.

2. **Geometry Dictates Metric:** No single distance metric is universally superior. We observed that Manhattan Distance (L_1) is essential for high-dimensional mixed data (Adult), where it mitigates the curse of dimensionality. Conversely, Euclidean Distance (L_2) remains optimal for physical spatial data (Pen-based).

3. **Regularization in Probabilistic Models:** The failure of GMM on the raw Mushroom dataset highlighted a critical vulnerability in probabilistic models when dealing with sparse, one-hot encoded data. Our introduction of Dynamic Regularization ($\epsilon = 10^{-1}$) proved to be a necessary architectural decision to ensure numerical stability in the covariance matrix inversion.

7.2 Critical Limitations

Despite the success of the improved algorithms, our analysis identified specific constraints:

- **Scalability of Kernel Methods:** While Intelligent Kernel K-Means successfully separated non-linear clusters in the Mushroom dataset, its $O(N^2)$ memory complexity makes it prohibitive for large-scale data. Even with `float32` optimization, the kernel matrix for the Adult dataset ($N \approx 48k$) approached the physical memory limits of our environment (16GB RAM).

- **Parameter Sensitivity in Fuzzy Clustering:** We found that Suppressed FCM is not a "set-and-forget" solution. The suppression parameter α requires careful tuning; an overly aggressive suppression ($\alpha < 0.5$) degrades the algorithm into a slow version of Hard K-Means, while insufficient suppression ($\alpha = 1.0$) fails to converge on sparse data.

7.3 Implications for Future Work

Based on these limitations, future extensions of this work should focus on approximation techniques. To address the scalability of Kernel K-Means, implementing the Nyström Method [10] to approximate the kernel matrix using a low-rank decomposition would allow the algorithm to scale linearly $O(N)$ rather than quadratically. Additionally, exploring Incremental PCA would allow for dimensionality reduction on datasets larger than physical memory, addressing the bottleneck encountered during the Adult dataset analysis. Finally, regarding the probabilistic modeling of the categorical Mushroom dataset, future iterations should replace the Gaussian Mixture Model with a Bernoulli Mixture Model (BMM) [11]. While our dynamic regularization strategy stabilized the GMM, a BMM is theoretically superior for modeling the latent structure of sparse, binary feature vectors without forcing continuous Gaussian assumptions on discrete data.

8 References

[1] UB (2025). "Introduction to Machine Learning: Work 3 - Clustering and visualization exercise". *Universitat de Barcelona*, Course 2025-2026.

[2] Salamó, M. (2025). "Support Slides: Sessions 1-4". *Universitat de Barcelona*, Introduction to Machine Learning.

[3] MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297.

[4] Bezdek, J.C., Ehrlich, R., & Full, W. (1984). "FCM: The fuzzy c-means clustering algorithm". *Computers & Geosciences*, 10(2-3), pp. 191-203.

[5] Celebi, M.E., Kingravi, H.A., & Vela, P.A. (2013). "A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm". *Expert Systems with Applications*, 40(1), pp. 200-210.

[6] Mishra, B.K., Rath, A.K., Nanda, S.K., & Baidyanath, R.R. (2019). "Efficient Intelligent Framework for Selection of Initial Cluster Centers". *I.J. Intelligent Systems and Applications*, 8, 44-55.

[7] Handhayani, T., & Hiryanto, L. (2015). "Intelligent Kernel K-Means for Clustering Gene Expression". *Procedia Computer Science*, 59, 171-177.

[8] Fan, J.L., Zhen, W.Z., & Xie, W.X. (2003). "Suppressed fuzzy c-means clustering algorithm". *Pattern Recognition Letters*, 24, pp. 1607-1612.

[9] Van der Maaten, L., & Hinton, G. (2008). "Visualizing Data using t-SNE". *Journal of Machine Learning Research*, 9, 2579-2605.

[10] Williams, C.K.I., & Seeger, M. (2001). "Using the Nyström method to speed up kernel machines". *Advances in Neural Information Processing Systems*, 13, pp. 682-688.

[11] Juan, A., & Vidal, E. (2002). "Bernoulli mixture models for binary images". *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, 3, pp. 367-370.

A PCA Internals Verification

The following data represents the internal state of the PCA algorithm during the execution of the visualization pipeline (`visualization.py`).
Note: To verify the correctness of the implementation while maintaining readable console logs, this output was generated using a stratified subset of the Adult dataset ($N = 3000$) projected onto 2 principal components.

A.1 Covariance Matrix (Σ)

The covariance matrix was computed manually in Step 4 of the algorithm. Below is the top-left 3×3 block of the 14×14 matrix, capturing the variance and correlations of the first three features.

$$\Sigma_{subset} = \begin{bmatrix} 0.0339 & -0.0012 & 0.0006 & \cdots \\ -0.0012 & 0.0052 & -0.0004 & \cdots \\ 0.0006 & -0.0004 & 0.0289 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

A.2 Eigenvalues (λ)

The sorted eigenvalues representing the explained variance for the top principal components found during this verification step:

Component	Eigenvalue (Explained Variance)
PC1	8.658×10^{-1}
PC2	3.601×10^{-1}
PC3	3.088×10^{-1}
PC4	2.682×10^{-1}
PC5	2.229×10^{-1}

A.3 Eigenvectors (W)

The principal component vectors (eigenvectors) derived from the decomposition. Below are the coefficients for the first 5 features contributing to the top 2 Principal Components (PC1 and PC2).

$$W_{top2} = \begin{bmatrix} \textbf{Feature} & \textbf{PC1 Weight} & \textbf{PC2 Weight} \\ \text{Feat 0} & -0.0697 & 0.1179 \\ \text{Feat 1} & 0.0025 & -0.0099 \\ \text{Feat 2} & -0.0105 & 0.0371 \\ \text{Feat 3} & -0.0066 & 0.0048 \\ \text{Feat 4} & -0.0057 & 0.0060 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

B Hyperparameter Grid

To ensure reproducibility, the following table summarizes the complete grid of hyperparameters swept during the experimental phases (Sessions 1, 2, and 3) as defined in `main.py`.

Table 5: Global Search Space for Experiments

Algorithm	**Parameter**	**Values Tested**
Global	Cluster Count (K)	$\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$
	Distance Metrics	Euclidean, Manhattan, Cosine
GMM	Initialization	Random, K-Means, K-Means++
	Regularization (ϵ)	10^{-4} (Dense), 10^{-1} (Sparse)
Fuzzy (FCM)	Fuzzifier (m)	$\{1.5, 2.0, 2.5\}$
	Suppression (α)	$\{1.0, 0.75, 0.5\}$
PCA	Components (d)	$\{2, 3, 5\}$