A Survey on K-means Clustering Algorithms: Theoretical Analysis and Performance Comparison

Yiming Liu*
liuym23@mails.tsinghua.edu.cn
Tsinghua University, Beijing, China

Siqiao Huang*
huangsq23@mails.tsinghua.edu.cn
Tsinghua University, Beijing, China

January 7, 2025

Abstract

We present a comprehensive theoretical analysis of k-means clustering algorithms, a fundamental family of partitional clustering methods that minimize distances between data points and their assigned cluster centers. While k-means clustering has been extensively studied, a systematic theoretical analysis and comparison of different k-means variants remains lacking. Our work first establishes the NP-completeness of k-means clustering. We then characterize the structure of locally optimal solutions and prove convergence properties for the k-means algorithm. We further examine various k-means paradigms, such as K-Medoids, Fuzzy K-Means, and IK-Means. Through rigorous theoretical analysis and experimental evaluation, we provide insights into the trade-offs among these variants. Our results offer theoretical guidance for selecting appropriate k-means variants and suggest directions for future algorithmic improvements in k-means clustering. Project Page

1 Introduction

"Do not forget that clusters are, in large part, on the eye of the beholder." [19]

Clustering, as a fundamental problem in computer science, focuses on partitioning a set of objects into groups (clusters) such that objects within the same cluster are more similar to each other than to those in different clusters [27]. Among various clustering approaches, k-means clustering [32] has emerged as one of the most widely studied and applied methods, owing to its simplicity, efficiency, and theoretical guarantees. The k-means clustering problem, which aims to minimize the sum of squared distances between data points and their assigned cluster centers, presents significant computational challenges and has been proven to be NP-hard when the number of clusters exceeds one [30, 16].

Within the broader context of k-means methodologies, various algorithmic variants have been developed to address different aspects of the clustering challenge. These variants can be broadly classified based on their initialization strategies (e.g., k-means++ [6]), distance metrics (e.g., spherical k-means [21]), kernel transformations (e.g., kernel k-means [38]), and computational

*Both authors contributed equally to this research.

efficiency improvements (e.g., mini-batch k-means [42] , online k-means [13]). Each variant presents unique theoretical properties and computational characteristics that warrant careful analysis.

This survey aims to provide a systematic examination of k-means clustering algorithms and their variants from an algorithmic perspective. We focus on three key aspects: theoretical foundations, algorithmic complexity and convergence analysis, and empirical performance evaluation.

The main contributions of this work include:

- A complete theoretical framework including NP-completeness proof and local optimality analysis
- A rigorous theoretical analysis of different k-means variants, examining their computational complexities and convergence properties
- An experimental framework for comparing the practical performance of different k-means variants, with a focus on convergence speed and clustering quality

The remainder of this paper is organized as follows: Section 2 establishes the theoretical foundations and formal definitions of k-means clustering. Section 3 proves that the k-means clustering problem is NP-complete even for k=2, highlighting the fundamental computational challenges in finding globally optimal solutions. Section 4 analyzes the structure of local optimal solutions and establishes convergence properties of k-means algorithm, providing theoretical guarantees for practical implementations. Section 5 presents a detailed analysis of various k-means variants, examining their theoretical properties and computational complexities. Section 6 describes our experimental methodology and presents comparative results and detailed analysis. Finally, Section 7 concludes with a summary of our theoretical analysis.

2 Theoretical Framework

What is clustering? Clustering is a fundamental problem in unsupervised learning that aims to partition data points into groups based on their similarity. Before delving into specific algorithms, we first present a formal mathematical framework.

2.1 Problem Formulation

We first define a class of k-means-type clustering problems. Given a dataset $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and a distance function $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, we seek:

- A partition $C_1, \ldots, C_k$ where $\bigcup_{i=1}^{k} C_i = \mathcal{X}$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. This ensures each data point belongs to exactly one cluster.
- Centers $c_1, \ldots, c_k$ that minimize the total distance based on function d. These centers represent the "prototypical" points of their respective clusters.

2.2 Objective Function

The objective function for k-means-type problems captures the total distortion within clusters:

$$G((\mathcal{X}, d), (C_1, \ldots, C_k)) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, c_i) \tag{1}$$

This function measures the sum of distances between each point and its assigned cluster center. The classical k-means algorithm emerges as a special case where $d(x, y) = \|x - y\|^2$, chosen for its mathematical properties that enable efficient optimization.

2.3 Center Update Rule

For k-means-type problems, we define a center update rule ϕ_d that maps a cluster to its representative center:

$$\phi_d : 2^{\mathcal{X}} \to \mathbb{R}^d \tag{2}$$

Here, $2^{\mathcal{X}}$ denotes the power set of $\mathcal{X}$. The specific form of ϕ_d depends on the problem structure and distance function d. For example, in classical k-means with squared Euclidean distance, ϕ_d computes the arithmetic mean.

2.4 Optimization Formulation

The clustering problem can be formulated as the following optimization problem:

$$\begin{aligned}
\text{Problem P:} \quad \text{minimize} \quad & \sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij} d(x_j, c_i) \\
\text{subject to} \quad & \sum_{i=1}^{k} w_{ij} = 1, \quad \forall j = 1, \ldots, n \\
& w_{ij} \in \{0, 1\}, \quad \forall i, j \\
& c_i = \phi_d(\{x_j : w_{ij} = 1\}), \quad \forall i = 1, \ldots, k
\end{aligned}$$

The binary variables w_{ij} encode cluster assignments, where $w_{ij} = 1$ indicates point x_j belongs to cluster i. The constraints ensure each point is assigned to exactly one cluster.

2.5 The Lloyd-type Algorithm

For general k-means-type problems, we extend Lloyd's algorithm as Algorithm 1.

Having established the theoretical framework for k-means-type clustering problems, we now turn our attention to the computational complexity of these problems. While the Lloyd-type algorithm provides an intuitive approach to solving clustering problems, it is important to understand the fundamental computational challenges involved. In particular, we will show that even for the simplest case of k=2, finding the optimal clustering solution is computationally intractable.

3 Computational Complexity of k-means Clustering

The optimization formulation presented in the previous section raises a natural question: how difficult is it to find an optimal solution to the k-means clustering problem? Despite the problem's simple geometric nature and the widespread use of heuristic algorithms like Lloyd's method, the following analysis reveals that finding a globally optimal solution is computationally challenging. We prove that the k-means clustering problem is NP-complete even when restricted to $k = 2$ [15, 16].

Input: Dataset $\mathcal{X}$, number of clusters k, distance function d, center update function ϕ_d
Output: Clusters $C_1, \ldots, C_k$ and centers $c_1, \ldots, c_k$
Initialize centers $c_1, \ldots, c_k$ randomly;
repeat
  `/* Assignment step - assigns each point to nearest center */`
  for $i \leftarrow 1$ **to** n **do**
    $j^* \leftarrow \arg\min_{j \in \{1,\ldots,k\}} d(x_i, c_j)$;
    Assign x_i to cluster C_{j^*};
  end
  `/* Update step - recomputes centers using` ϕ_d `*/`
  for $j \leftarrow 1$ **to** k **do**
    $c_j \leftarrow \phi_d(C_j)$;
  end
until *convergence criterion is met*;

Algorithm 1: Lloyd-type Algorithm

The proof proceeds by reduction from a variant of Not-All-Equal 3SAT (NaeSat*, Defination 3.1), which we know is NP-complete [14].

Definition 3.1 (NaeSat*). *Input: A Boolean formula $\phi(x_1, \ldots, x_n)$ in 3CNF where:*

- *Each clause contains exactly three literals*
- *Each pair of variables appears together in at most two clauses:*
  - *Once as either $\{x_i, x_j\}$ or $\{\overline{x_i}, \overline{x_j}\}$*
  - *Once as either $\{x_i, \overline{x_j}\}$ or $\{\overline{x_i}, x_j\}$*

Output: true if there exists an assignment where each clause has exactly one or two true literals, false otherwise.

We want to prove the following theorem.

Theorem 3.2. *2-means clustering is NP-complete.*

To prove this theorem, we follow the standard approach for establishing NP-completeness. First, we show that 2-means clustering belongs to NP by demonstrating that a solution can be verified in polynomial time. Then, we prove hardness through a reduction from NaeSat*. This reduction will map boolean formulas to clustering instances in such a way that satisfiable formulas correspond exactly to clusterings with cost below a carefully chosen threshold.

3.1 Membership in NP

Lemma 3.3. *k-means clustering is in NP.*

Proof. Given a solution $(C_1, \ldots, C_k)$, we can verify in polynomial time:

1. The partition is valid: each point appears in exactly one cluster

2. The cost is optimal for the given partition by checking each centroid is at the right position
3. The total cost is below the specified threshold

The verification takes $O(nk)$ time where n is the number of points. □

3.2 Reduction Construction

Given an instance ϕ of NaeSat* with n variables and m clauses, we construct a 2-means clustering instance that captures the satisfiability properties of ϕ. The construction proceeds as follows:

1. For each variable x_i in ϕ, we create two corresponding points in our metric space: one representing the positive literal x_i and one representing the negative literal $\overline{x_i}$. This gives us $2n$ points in total.
2. We define a distance matrix D that encodes the logical structure of ϕ. For any two literals α and β:

$$D_{\alpha\beta} = \begin{cases} 0 & \text{if } \alpha \text{ and } \beta \text{ are identical} \\ 1+\Delta & \text{if } \alpha \text{ is the negation of } \beta \\ 1+\delta & \text{if } \alpha \text{ and } \beta \text{ appear together in a clause} \\ 1 & \text{otherwise} \end{cases}$$

   where the parameters δ and Δ satisfy:

   - $0 < \delta < \Delta < 1$
   - $4\delta m < \Delta \leq 1 - 2\delta n$
   - A concrete valid setting is $\delta = \frac{1}{5m+2n}$ and $\Delta = 5\delta m$, we can also choose other settings.

3. We set our target clustering cost threshold to $c(\phi) = n - 1 + \frac{2\delta m}{n}$

3.3 Correctness of the Reduction

We prove the reduction is correct through two key lemmas that establish the equivalence between NaeSat* solutions and low-cost clusterings.

Lemma 3.4 (Forward Direction)**.** *If ϕ has a satisfying assignment for NaeSat*, then there exists a 2-means clustering with cost exactly $c(\phi)$.*

Proof. Given a satisfying assignment for ϕ, we construct an optimal clustering as follows:

1. Let C_1 contain all literals that are true in the satisfying assignment
2. Let C_2 contain all literals that are false in the satisfying assignment

   This clustering has several important properties:

   - Each cluster contains exactly n points (one literal from each variable)
   - By the NaeSat* property, each clause must have literals assigned both true and false

- Therefore, each clause contributes exactly one pair of related literals ($\alpha \sim \beta$) to each cluster

The total clustering cost can be calculated as:

$$\begin{aligned}\text{cost} &= \frac{1}{2n}\sum_{i,i' \in C_1} D_{ii'} + \frac{1}{2n}\sum_{i,i' \in C_2} D_{ii'} \\ &= 2 \cdot \frac{1}{n}\left((\binom{n}{2} - m) \cdot 1 + m \cdot (1+\delta) \right) \\ &= n - 1 + \frac{2\delta m}{n} = c(\phi)\end{aligned}$$

□

Lemma 3.5 (Backward Direction)**.** *If there exists a 2-means clustering with cost* $\leq c(\phi)$*, then* ϕ *is satisfiable.*

Proof. We prove this through a sequence of structural properties:

1. First, we prove that no cluster can contain both a literal and its negation:
   - Suppose C_1 contains n' points including a complementary pair
   - The minimum possible cost would be:
   $$\frac{1}{n'}((\binom{n'}{2} - 1) \cdot 1 + 1 \cdot (1+\Delta)) + \frac{1}{2n-n'}\binom{2n-n'}{2} \cdot 1 \geq n - 1 + \frac{\Delta}{2n} > c(\phi)$$
   - This contradicts our assumption about the clustering cost
2. This implies that each cluster must contain exactly n points, with exactly one literal from each variable pair
3. The cost formula then simplifies to:
   $$\frac{2}{n}\left(\binom{n}{2} + \delta \sum_{\text{clauses}} (1 \text{ if split}, 3 \text{ otherwise}) \right)$$
4. For this to be $\leq c(\phi)$, every clause must be "split" between clusters
5. Therefore, assigning true to all literals in C_1 and false to all literals in C_2 gives a valid NaeSat* solution

□

3.4 Geometric Realization of the Distance Matrix

The previous lemmas established that our construction yields a valid reduction from NaeSat* to 2-means clustering if we can represent the distances in Euclidean space. We now show this is possible using theorems established by Schoenberg [40].

Theorem 3.6 (Geometric Realization). *Given our distance matrix $D(\phi)$, there exist points $\{x_\alpha\}_{\alpha \in literals} \subset \mathbb{R}^{2n}$ such that $D_{\alpha\beta} = \|x_\alpha - x_\beta\|^2$ for all literals α, β.*

Proof. The proof proceeds in two steps:

1. First, we show that $D(\phi)$ satisfies Schoenberg's criterion for embeddability into ℓ_2^2
2. Then, we explain how to explicitly construct the embedding using classical multidimensional scaling

Step 1: Verifying Schoenberg's Criterion

By Schoenberg's theorem [40], a symmetric matrix D can be realized as squared Euclidean distances if and only if $-HDH$ is positive semidefinite, where $H = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ is the centering matrix.

We will show this by proving that $v^T HDHv \leq 0$ for all vectors $v \in \mathbb{R}^N$. Let $u = Hv$. Note that $u \cdot \mathbf{1} = 0$ for any such u, because $H\mathbf{1} = 0$. Therefore, it suffices to show $u^T Du \leq 0$ for all vectors u with $u \cdot \mathbf{1} = 0$.

For any such vector u, let u^+ denote its first n coordinates and u^- its last n coordinates. Then:

$$\begin{aligned}
u^T Du &= \sum_{\alpha,\beta} D_{\alpha\beta} u_\alpha u_\beta \\
&= \sum_{\alpha,\beta} u_\alpha u_\beta (1 - \mathbf{1}(\alpha = \beta) + \Delta \cdot \mathbf{1}(\alpha = \overline{\beta}) + \delta \cdot \mathbf{1}(\alpha \sim \beta)) \\
&= \sum_{\alpha,\beta} u_\alpha u_\beta - \sum_\alpha u_\alpha^2 + \Delta \sum_\alpha u_\alpha u_{\overline{\alpha}} + \delta \sum_{\alpha \sim \beta} u_\alpha u_\beta \\
&\leq \left(\sum_\alpha u_\alpha\right)^2 - \|u\|^2 + 2\Delta(u^+ \cdot u^-) + \delta \sum_{\alpha,\beta} |u_\alpha||u_\beta| \\
&\leq -\|u\|^2 + \Delta(\|u^+\|^2 + \|u^-\|^2) + \delta \left(\sum_\alpha |u_\alpha|\right)^2 \\
&\leq -(1-\Delta)\|u\|^2 + 2\delta\|u\|^2 n \leq 0
\end{aligned}$$

The last inequality follows from our choice of parameters where $2\delta n \leq 1 - \Delta$.

Step 2: Constructing the Embedding

Given that D satisfies Schoenberg's criterion, we can explicitly construct the embedding points $\{x_\alpha\}$ as follows:

1. Form the matrix $B = -\frac{1}{2}HDH$
2. Compute the eigendecomposition $B = U\Lambda U^T$
3. The embedding points are given by the rows of $U\Lambda^{1/2}$

This construction is guaranteed to work because B is positive semidefinite, and the resulting points $x_\alpha \in \mathbb{R}^{2n}$ will satisfy $\|x_\alpha - x_\beta\|^2 = D_{\alpha\beta}$ for all pairs of literals α, β. □

This geometric realization completes our reduction by showing that the abstract distance matrix $D(\phi)$ can be realized by actual points in Euclidean space, making our reduction valid for the standard k-means clustering problem.

Having established the computational hardness of finding globally optimal solutions to the k-means clustering problem, we now turn our attention to understanding the structure of partial optimal solutions and local minima. This analysis is crucial because while finding global optima is NP-hard, most practical algorithms (including Lloyd's algorithm) aim to find high-quality local solutions. We begin by introducing the reduced function formulation, which will allow us to analyze the geometric and algorithmic properties of k-means clustering more deeply.

Specifically, we will:

1. Characterize the structure of partial optimal solutions using optimization theory
2. Establish convergence properties of the k-means algorithm
3. Analyze special cases where partial optimal solutions coincide with local minima

4 Local Optimality and Convergence Analysis

4.1 Reduced Function

To analyze the problem structure more deeply, we use reduced function (Definition 4.1) that eliminates the center variables defined by Selim [43]:

Definition 4.1. *The reduced function $F(W)$ of Problem P is defined by:*

$$F(W) = \min_{c_1,\ldots,c_k \in \mathbb{R}^d} \sum_{i=1}^{k} \sum_{j=1}^{n} w_{ij} d(x_j, c_i) \tag{3}$$

where $W = [w_{ij}]$ is a $k \times n$ matrix of assignment variables.

This leads to the reduced problem:

$$\begin{aligned} \text{Problem RP:} \quad \text{minimize} \quad & F(W) \\ \text{subject to} \quad & \sum_{i=1}^{k} w_{ij} = 1, \quad \forall j = 1, \ldots, n \\ & w_{ij} \geq 0, \quad \forall i, j \end{aligned}$$

To understand the properties of this reduced problem, we first analyze the structure of the reduced function $F(W)$ under different conditions on the distance function d.

Theorem 4.2. *When $d(x, y)$ is convex in y for fixed x, the following properties hold:*

- *$F(W)$ is concave in W*

- *For $w_{ij} \in [0,1]$, if $d(x,y)$ is strictly convex in y, then optimal solutions satisfy $w_{ij} \in \{0,1\}$*
- *If $d(x,y)$ is concave in y, then optimal w_{ij} may be fractional*

Proof. The concavity follows from:

$$\begin{aligned} F(\gamma W^1 + (1-\gamma)W^2) &= \min_c \sum_{i,j} (\gamma w_{ij}^1 + (1-\gamma) w_{ij}^2) d(x_j, c_i) \\ &= \min_c \{\gamma \sum_{i,j} w_{ij}^1 d(x_j, c_i) + (1-\gamma) \sum_{i,j} w_{ij}^2 d(x_j, c_i)\} \\ &\geq \gamma \min_c \sum_{i,j} w_{ij}^1 d(x_j, c_i) + (1-\gamma) \min_c \sum_{i,j} w_{ij}^2 d(x_j, c_i) \\ &= \gamma F(W^1) + (1-\gamma) F(W^2) \end{aligned}$$

The binary property follows from strict convexity of d, which ensures that any fractional solution can be improved by moving to a vertex of the feasible region.

□

Lemma 4.3. *For classical k-means with non-empty clusters and $d(x,y) = \|x-y\|^2$:*

$$F(W) = \sum_{j=1}^{n} \sum_{i=1}^{k} w_{ij} \|x_j - \frac{\sum_{l=1}^{n} w_{il} x_l}{\sum_{l=1}^{n} w_{il}}\|^2 \tag{4}$$

where $\sum_{l=1}^{n} w_{il} > 0$ for all i.

While the reduced function provides a useful reformulation of our problem, to develop efficient algorithms we need to understand the structure of optimal solutions. This leads us to consider partial optimal solutions, which play a crucial role in understanding the convergence properties of k-means-type algorithms.

4.2 Partial Optimal Solutions

Definition 4.4. *A point (W^*, Z^*) is called a partial optimal solution if:*

$$\begin{aligned} f(W^*, Z^*) &\leq f(W, Z^*) \quad \text{for all } W \in S \\ f(W^*, Z^*) &\leq f(W^*, Z) \quad \text{for all } Z \in \mathbb{R}^{nk} \end{aligned}$$

where S is the feasible set of cluster assignment matrices.

To find partial optimal solutions, we consider two subproblems:

- Problem P_1: Given $\hat{Z}$, minimize $f(W, \hat{Z})$ subject to $W \in S$
- Problem P_2: Given $\hat{W}$, minimize $f(\hat{W}, Z)$ subject to $Z \in \mathbb{R}^{nk}$

To better understand the structure of these partial optimal solutions, we can characterize them using the Kuhn-Tucker optimality conditions from nonlinear programming theory.

Theorem 4.5 (Equivalence of Partial Optimal Solutions and Kuhn-Tucker Points)**.** *Under differentiability conditions, a point* (W^*, Z^*) *is a partial optimal solution if and only if it satisfies the Kuhn-Tucker conditions:*

$$\begin{aligned}
&i) \quad \nabla_{w_j} f(W,Z) + \Pi^t E \geq 0 \quad j = 1,\ldots,m \\
&ii) \quad (\nabla_{w_j} f(W,Z) + \Pi^t E) w_j = 0 \quad j = 1,\ldots,m \\
&iii) \quad E w_j = 1, w_j \geq 0 \quad j = 1,\ldots,m \\
&iv) \quad \nabla_{z_i} f(W,Z) = 0 \quad i = 1,\ldots,k
\end{aligned}$$

where E *is a vector of ones and* Π *is the vector of Lagrange multipliers.*

Proof. ($\Rightarrow$) First, assume (W^*, Z^*) is a partial optimal solution.

1. Since W^* minimizes $f(W, Z^*)$ over $W \in S$, it must satisfy the KKT conditions for Problem P_1:
   - Stationarity: $\nabla_{w_j} f(W, Z^*) + \Pi^t E \geq 0$
   - Complementary slackness: $(\nabla_{w_j} f(W, Z^*) + \Pi^t E) w_j = 0$
   - Primal feasibility: $E w_j = 1, w_j \geq 0$
2. Since Z^* minimizes $f(W^*, Z)$ over $Z \in \mathbb{R}^{nk}$, we have:
   - $\nabla_{z_i} f(W^*, Z^*) = 0$ (unconstrained minimization)

($\Leftarrow$) Now assume (W^*, Z^*) satisfies conditions i)-iv).

1. Conditions i)-iii) are the KKT conditions for Problem P_1, implying W^* minimizes $f(W, Z^*)$ over $W \in S$
2. Condition iv) implies Z^* is a stationary point of $f(W^*, Z)$, and since this is unconstrained and f is convex in Z, Z^* minimizes $f(W^*, Z)$

Therefore, (W^*, Z^*) satisfies both conditions of a partial optimal solution. □

Theorem 4.6 (K-means Convergence)**.** *Algorithm converges to a partial optimal solution of Problem P in a finite number of iterations.*

Proof. Let's prove by contradiction that the algorithm visits each extreme point of S at most once.

Assume there exist iterations r_1 and r_2 ($r_1 \neq r_2$) where the algorithm visits the same extreme point:

$$W^{r_1} = W^{r_2} \tag{5}$$

Following the algorithm steps:

$$\begin{aligned}
&\text{At iteration } r_1 : \text{We solve } P_2 \text{ with } W = W^{r_1} \text{ to get } Z^{r_1+1} \\
&\text{At iteration } r_2 : \text{We solve } P_2 \text{ with } W = W^{r_2} \text{ to get } Z^{r_2+1}
\end{aligned}$$

Since $W^{r_1} = W^{r_2}$, we have:

$$\begin{aligned}
f(W^{r_1}, Z^{r_1+1}) &= f(W^{r_2}, Z^{r_1+1}) && \text{(same W)} \\
&= f(W^{r_2}, Z^{r_2+1}) && \text{(from step ii)}
\end{aligned}$$

However, by the construction of the algorithm, we know that:

$$f(W^{r_1}, Z^{r_1}) > f(W^{r_1}, Z^{r_1+1}) > \cdots > f(W^{r_2}, Z^{r_2+1}) \tag{6}$$

This creates a contradiction because we cannot have:

$$f(W^{r_1}, Z^{r_1+1}) = f(W^{r_2}, Z^{r_2+1}) \tag{7}$$

while also having a strictly decreasing sequence between these points. Therefore, the algorithm cannot visit the same extreme point twice.

Since the feasible region has a finite number of extreme points and the algorithm never revisits the same extreme point, it must terminate in a finite number of iterations. When the algorithm terminates, the solution satisfies partial optimality conditions. □

To further characterize local optimality, we need two results from optimization theory. For brevity, we states two lemmas without proof. For more details, please refer to [31]

Lemma 4.7 (Directional Derivatives). *Let $f(W, Z)$ be defined for all W and for all $Z \in V$ where:*

(i) V is compact and

(ii) f and the partial derivatives $\partial f / \partial w_{ij}$ are continuous.

Let $F(W) = \min\{f(W, Z) : Z \in V\}$ and define

$$A(W^*) = \{Z : Z \text{ minimizes } f(W^*, Z), Z \in V\}. \tag{8}$$

The one-sided directional derivative of F at W^* in direction d is given by:

$$DF(W^*; d) = \lim_{\alpha \to 0^+} \frac{F(W^* + \alpha d) - F(W^*)}{\alpha} \tag{9}$$

Then $DF(W^*; d)$ exists for any d at any point W^* and is given by:

$$DF(W^*; d) = \min\{\nabla_w^t f(W^*, Z) \cdot d : Z \in A(W^*)\} \tag{10}$$

where $\nabla_w f(W^*, Z)$ is the vector of partial derivatives $\partial f(W, Z)/\partial w_{ij}$ evaluated at $W = W^*$.

Lemma 4.8 (Optimality Conditions). *Consider Problem RP: $\min\{F(W) : W \in S\}$ where S is convex. Let $W^* \in S$. Then W^* is a local minimum of RP if and only if the directional derivatives satisfy:*

$$DF(W^*; d) \geq 0 \tag{11}$$

for each feasible direction d at W^*.

Using these lemmas, we can now establish a complete characterization of local optimality for k-means clustering.

Theorem 4.9 (Local Optimality Characterization). *For a partial optimal solution (W^*, Z^*) where W^* is an extreme point of the polyhedral set S and $Z^* \in A(W^*)$, W^* is a local minimum of Problem RP if and only if:*

$$F(W^*) = f(W^*, Z^*) \leq \min\{f(W, Z) : W \in S \text{ for all } Z \in A(W^*)\} \tag{12}$$

where $A(W^*) = \{Z : Z \text{ minimizes } f(W^*, Z), Z \in V\}$ and V is a compact set containing Z^*.

Proof. Consider Problem RP and assume the inequality holds. For any $Z \in A(W^*)$, by assumption:

$$f(W^*, Z^*) = f(W^*, Z) \leq \min\{f(W, Z) : W \in S\}$$

This implies that for any feasible direction d at W^*:

$$\nabla_w f(W^*, Z) \cdot d \geq 0$$

Since this holds for any $Z \in A(W^*)$:

$$\min\{\nabla_w f(W^*, Z) \cdot d : Z \in A(W^*)\} \geq 0$$

By Lemma 4.7, the left-hand side equals $DF(W^*; d)$, the directional derivative. Therefore, by Lemma 4.8, W^* is a local minimum of Problem RP.

The converse follows similarly by reversing these steps. □

A particularly important special case occurs when the set A(W*) contains exactly one element.

Theorem 4.10 (Singleton Local Optimality)**.** *Let (W^*, Z^*) be a partial optimal solution of Problem P and let $A(W^*)$ given by equation (8) be singleton; then W^* is a local minimum of Problem RP.*

Proof. Since W^* is a partial optimal solution, then by definition, $f(W^*, Z^*) \leq \min\{f(W, Z^*) : W \in S\}$. But this is precisely the condition of local optimality of W^* if $A(W^*)$ is singleton, as shown in Theorem 4.9. Hence the proof is complete. □

Theorem 4.11 (Convergence under Quadratic Metrics)**.** *Consider Problem P where $D(x_j, c_i) = (x_j - c_i)^t(x_j - c_i)$. Then any partial optimal solution is a local minimum point.*

Proof. Let (W^*, Z^*) be a partial optimal solution. For quadratic metrics:

$$c_i^* = \frac{\sum_{j=1}^n w_{ij}^* x_j}{\sum_{j=1}^n w_{ij}^*}$$

This value is unique for each cluster i, therefore $A_i(W_i^*)$ is singleton for all i. Consequently, $A(W^*)$ is singleton as well.

By Theorem 4.10, when $A(W^*)$ is singleton, any partial optimal solution is automatically a local minimum point of Problem RP. Therefore, under quadratic metrics, all partial optimal solutions are local minima. □

Finally, we can apply these results to analyze the special case of quadratic distance metrics, which includes the classical k-means algorithm.

Corollary 4.12 (K-means Convergence)**.** *The K-means algorithm with quadratic metrics converges to a local minimum in a finite number of iterations.*

Proof. By Theorem 4.6, the K-means algorithm converges to a partial optimal solution in finite iterations. By Theorem 4.11, this solution must be a local minimum when using quadratic metrics. □

Having established the theoretical properties of the classical k-means algorithm, including its convergence guarantees and local optimality conditions, we now explore how these foundations can be extended and modified to address various practical challenges. While k-means is widely used, its limitations – such as sensitivity to initialization, restricted use of Euclidean distance, and the requirement for numeric data – have motivated numerous variants of the algorithm.

5 Variants of K-Means

In this section, we summarize some of the well-known variants of K-Means, which include different problem setups, different algorithm. We introduce their problem formulation, practical algorithms, relation to K-Means and some theoretical results. Specifically, we introduce some of the most well-established variants in K-Means with variations in the following aspects: (1) Representative Elements and Distance Generating Function for the clusters (K-Medoids, K-Medians, K-Modes, Mean Shift Clustering), (2) Feature Transformation (Fuzzy K-Means, Kernel K-Means, Weighted K-Means) , (3) Initial Representative Elements Assignment (IK-Means), (4) Algorithm Paradigm (Bisecting K-Means Clustering).

5.1 K-Medoids Clustering

K-Mediods Clustering [41] is a variant of K-Means Clustering. Similar to K-Means, it also aims to find a set of $\{C_1, C_2, \cdots, C_k\}$ that minimizes a pre-defined loss function. However, the problem formulation is different in the following two ways:

1. The representative points of each cluster $\{m_1, m_2, \cdots, m_k\}$ are restricted to be one of the data points in the dataset. i.e. $\{m_1, m_2, \cdots, m_k\} \subseteq \{x_1, x_2, \cdots, x_n\}$.
2. Instead of minimizing the SSE loss in K-Means, the loss function for K-Medoids is defined using Absolute Error:

$$G_{\text{k-medoids}}((\mathcal{X}, d), (C_1, \ldots, C_k)) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\| \tag{13}$$

where the representative element μ_i denotes the *medoid* of the cluster.

$$m_i = \min_{m \in C_i} \sum_{x \in C_i} \|x - m_i\| \tag{14}$$

Intuitively, as l_2 norm squares each term's difference compared to l_1 norm, the model trained on l_1 norm will be more robust to outliers. Also, the property of all representative elements are points within the given input set $\mathcal{X}$ makes it useful in scenarios where more "natural" solutions are desired [39].

Similar to K-Means, the K-Medoid clustering algorithms also follow the paradigm of iteratively updating the representative points and the clusters until convergence. A Basic K-Medoid clustering algorithm is provided in 2.

In the naive implementation of k-Medoids (Option 1 of Algorithm 2), after we select the initial medoids, we need to compute the total cost of swapping the representative object m with x_i for all the non-representative objects x_i. This is a $O(n^2)$ operation.

To address this issue, a modification of the k-Medoids algorithm, Partition Around Medoids (PAM) is proposed in [1]. The algorithm first builds a dissimilarity matrix on the given dataset, refered to in literature as BUILD operation. The second part of the algorithm, refered to as SWAP operations, iteratively selectes a random non-representative point and calculating whether swapping the representative object with it yields better loss function, as illustrated in Option 2 of Algorithm 2.

Compared to k-Means Clustering, k-Medoids Clustering:

Input: Dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$, number of clusters K
Output: Clusters $C_1, \ldots, C_K$ and medoids $m_1, \ldots, m_K$
Select K points as the initial representative objects medoids;
repeat
  `/* Assign each point to the cluster with the nearest medoid */`
  for $i \leftarrow 1$ **to** n **do**
    $j^* \leftarrow \arg\min_{j \in \{1,\ldots,K\}} \|x_i - m_j\|$;
    Assign x_i to cluster C_{j^*};
  end
  `/* Option 1: Iterate to find the best medoid of each cluster */`
  for $i \leftarrow 1$ **to** K **do**
    $m_i \leftarrow \arg\min_{x \in C_i} \sum_{y \in C_i} \|x - y\|$;
  end
  `/* Option 2:Update medoids by testing replacements */`
  Randomly select a non-representative object x_i;
  Compute the total cost S of swapping the representative object m with x_i;
  if $S < 0$ **then**
    Swap m with x_i to form the new set of K representative objects;
  end
until *convergence*;

Algorithm 2: K-Medoids Clustering

- Provides more natural solutions by restricting representative points on the input set $\mathcal{X}$
- Are more robust to noise and outliers

However the computational complexity of k-Medoids is higher, restricting it's application to larger datasets. Classical works like Clustering LARge Application (CLARA) algorithm [26] extends the idea of PAM to larger datasets through sampling methods.

5.2 K-Medians Clustering

K-Medians Clustering [24] selects the medians of the clusters, as opposed to the means in K-Means Clustering. For each cluster C_i, the median med is defined as

$$med_{ij} = median_{x \in C_i}(x_j) \tag{15}$$

where med_{ij} is the j-th feature of the median point in the cluster C_i, and x_j is the j-th feature of data point x in the cluster C_i. The loss function for K-Medians is defined as:

$$\sum_{i=1}^{k} \sum_{x \in C_i} \sum_{j=1}^{d} |x_j - med_{ij}| \tag{16}$$

K-medians is more robust to outliers compared to K-means. One should not confuse K-Medians' representative elements to be restricted on points in $\mathcal{X}$. As the median is selected feature-wise, the resulting representative element may not be a point in the given set of points. For instance, given cluster $(0, 0), (1, -1), (2, 1)$, the representative point is $(1, 0)$, which is not an element in the cluster.

Classical algorithm uses Lloyd-style iteration which iterates between an expectation step and a maximization step, as shown in k-Means. In the expectation step, input data points are assigned to their nearest median. In the maximization step, the medians are recomputed by using the median in each single dimension.

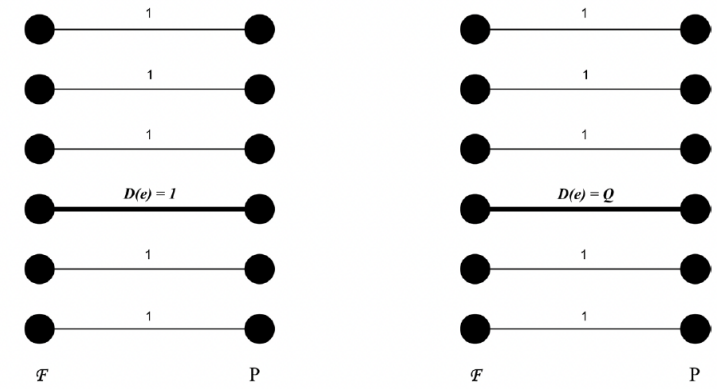

Figure 1: An illustration of the constructed instance in Theorem 5.1's proof [8]

Even in metric space, k-median is known to be NP-hard. For approximation algorithms, there exists a $O(1)$-approximation algorithm that runs in $\tilde{O}(kn)$ time [34] ,and it's been proved that there is no $O(1)$-approximation algorithm for metric k-median with $o(kn)$ runtime [8].

Theorem 5.1. *For any $\rho \geq 1$, every approximation algorithm (including randomized algorithm) with approximation ratio ρ for estimating the cost of metric k-median for $k = \frac{n}{2}$ requires time $\Omega(n^2)$.*

Proof. For simplicity, we denote the number of input points as $2n$, and the number of clusters $k = \frac{2n}{2} = n$. We then partition the $2n$ points into two sets: n points in P and n points in F. Between these two set of points, we construct a perfect matching $\mathcal{M}$, and choose an edge $e \in \mathcal{M}$ at random. We then define the distances in $(P \cup F, D)$ according to the following:

- $\forall e^* \in \mathcal{M} \setminus \{e\}, D(e^*) = 1$,
- For one instance, $D(e) = 1$, for the other, $D(e) = Q = 2n(\rho - 1) + 3$
- $\forall (x, y) \notin \mathcal{M}, D(x, y) = n^3\rho$.

The metric space on $(P \cup F, D)$ is well-defined. For arbitrarily large n, we know that the optimal solution is the following clustering:

$$\{C_1, ...C_{k=n}\} = \{\{x, y\} | (x, y) \in E\} \tag{17}$$

If $D(e) = Q$, then the cost of the k-Median problem is $2n\rho + 1$, and if $D(e) = 1$, then the cost is $2n$. Hence, any ρ-factor approximation algorithm for the k-Median problem must distinguish between these two problem instances. However, this requires an oracle to check if there exists an edge of length Q, which is known to be $\Omega(n^2)$, even if a randomized algorithm is used. □

5.3 K-Modes Clustering

Definition 5.2. *Let $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$ denote a set of m distinct categorical attributes, each associated with a finite set $\mathcal{O}_j$ $(1 \leq j \leq d)$ as its domain, where $DOM(A_j) = \mathcal{O}_j$ $(\geq 2$ discrete values$)$.*

A categorical dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ comprises n categorical data points, where each object $\mathbf{x}_i \in \mathcal{D}$ $(1 \leq i \leq n)$ is a tuple $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \in \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_d$.

For categorical data, k-Means Clustering is not applicable as the l_2 distance between categorical data points is not well-defined. Intuitively, if we denote the categories as $0, 1, 2$, the distance between $(0, 2)$ and $(0, 1)$ should be the same, as they both represent two points of different categories. In other words, the distance should be invariant to the order of the categories, which is not the case for the l_2 distance.

Definition 5.3. *The l_0-norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, denoted by $\|\mathbf{x}\|_0$, is defined as the number of non-zero entries in $\mathbf{x}$. Mathematically, it is expressed as:*

$$\|\mathbf{x}\|_0 = \sum_{i=1}^{m} \mathbf{1}(x_i \neq 0),$$

where $\mathbf{1}(\cdot)$ is the indicator function, which equals 1 if the condition inside is true, and 0 otherwise. The L_0-norm is commonly used to measure the sparsity of a vector.

Instead, in K-Modes Clustering [11], we use l_0 norm to define the distance generating function. This implies that the difference between two data points x, y are defined by the number of features that the two disagree with. A Basic K-modes algorithm is described in Algorithm 3.

Input: Dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$, number of clusters K
Output: Clusters $C_1, \ldots, C_K$ and modes $m_1, \ldots, m_K$
Select K initial modes;
repeat
  `/* Assign points to the cluster with the nearest mode using the matching metric */`
  for $i \leftarrow 1$ **to** n **do**
    $j^* \leftarrow \arg\min_{j \in \{1,\ldots,K\}} \text{matching\_distance}(x_i, m_j)$;
    Assign x_i to cluster C_{j^*};
  end
  `/* Recompute the modes of the clusters */`
  for $j \leftarrow 1$ **to** K **do**
    Recompute the mode m_j of cluster C_j;
  end
until *convergence*;

Algorithm 3: K-Modes Clustering

The algorithm works by iteratively assigning data points to the nearest cluster mode, which is the most common value within the cluster.

5.4 Mean Shift Clustering

Mean Shift Clustering [12] is a popular non-parametric clustering method. Its objective is to discover the modes (local maxima) in the data distribution by iteratively shifting data points toward the direction of maximum density. This process is based on the Parzen window kernel density estimation method and performs gradient ascent until convergence. The stationary points of the estimated density represent the modes, which serve as the cluster centers.

Definition 5.4. *Parzen window kernel density estimate:*
Given N data points $\{x_i\}_{i=1}^N$ in a d-dimensional space $\mathbb{R}^d$, the multivariate Parzen window kernel density estimate $f(x)$ is computed using a kernel function $K(x)$ and window radius h as:

$$f(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right), \tag{18}$$

Definition 5.5. *Mean Shift Vector*
The mean shift vector $m_h(x)$, which points toward the direction of the maximum increase in density, is calculated as:

$$m_h(x) = \frac{\sum_{i=1}^{N} x_i \cdot g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}, \tag{19}$$

where $g(x)$ is the gradient of the kernel $K(x)$.

The Mean Shift Clustering algorithm iteratively computes $m_h(x)$ for each data point x and updates x until convergence to a stationary point. The algorithm is summarized in Algorithm 4.

Input: $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, kernel function $K(x)$, window radius h
Output: Cluster centers corresponding to the modes of the data distribution
Step 1: Initialization. Select initial points from the dataset as the starting modes;
repeat
  foreach *point* $x \in \mathcal{X}$ **do**
    `/* Step 2: Compute mean shift vector */`
    Compute the mean shift vector $m_h(x)$ as:
    $m_h(x) = \frac{\sum_{i=1}^{N} x_i \cdot g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}$
    `/* Step 3: Update point location */`
    Update the point x as:
    $x \leftarrow m_h(x)$.
  end
until *Modes converge*;

Algorithm 4: Mean Shift Clustering

The Mean Shift Clustering Algorithm has broad applications in pattern recognition and computer vision, for example image segmentation, object tracking, and feature space analysis. However, its performance is sensitive to the choice of the kernel function and the bandwidth h.

5.5 Fuzzy K-Means Clustering

Standard K-Means restrict each data points' membership to only one cluster. However, for many real world applications, the underlying structure contains overlapping clusters. Fuzzy K-Means Clustering, or often referred to as Fuzzy C-Means clustering, relaxes the unitary-membership restriction to confront this issue.

In fuzzy C-means clustering algorithm (FCM) [37] , the loss function for a clustering solution $\mathcal{C}$ is given by:

$$G_{\text{fuzzy}}(\mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} w_{xik}^{\beta} \|x_i - c_k\|^2 \tag{20}$$

Here, w_{xik} is the membership weight of point x_i belonging to cluster C_k. This weight is used during the clustering process.

The membership weight w_{xik} is computed as:

$$w_{xik} = \frac{1}{\sum_{j=1}^{K} \left(\frac{\|x_i - c_k\|}{\|x_i - c_j\|} \right)^{\frac{2}{\beta-1}}} \tag{21}$$

The cluster center c_k is calculated as:

$$c_k = \frac{\sum_{x_i \in C_k} w_{xik}^{\beta} x_i}{\sum_{x_i \in C_k} w_{xik}} \tag{22}$$

Then, starting from a set of initial centroids and repeatedly applying formulas 21 and 22, a computational algorithm has been proven to converge to a local optimum of criterion 20. The algorithm is presented in Algorithm 5.

Although can be seen as a generalized version of K-Means, it still suffers some of the most significant issues with K-Means, namely being sensitive to outliers and converges to local minimum.

Further works that extends this idea include: Rough C-means [33], Possibilistic C-means [29] and Kernel FCM (KFCM) [18]. For instance, KFCM (Kernel Fuzzy C-Means) is an improved version of the FCM algorithm, leveraging kernel methods to map data into a higher-dimensional feature space. It's key modifications include:

1. **Kernel Mapping** Each data point $x_i \in \mathcal{X}$ is mapped into a higher-dimensional feature space using a kernel function ϕ, where the kernel function is defined as:

$$K(x_i, x_j) = \phi(x_i)^{\top} \phi(x_j) \tag{23}$$

   Here, $K(x_i, x_j)$ represents the inner product in the higher-dimensional feature space.

2. **Loss Function** Accordingly, the loss function is modified to the following form:

$$J(U, C) = \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}^{\beta} \|\phi(x_i) - c_k\|^2 \tag{24}$$

   where u_{ik} is the membership degree of data point x_i to cluster C_k, $\beta > 1$ is the fuzziness parameter, c_k is the center of cluster C_k in the feature space.

Input: Dataset $\mathcal{X}$, number of clusters K, fuzziness parameter β, stopping criteria ϵ
Output: Clusters $\{C_1, C_2, \ldots, C_K\}$ and cluster centers $\{c_1, c_2, \ldots, c_K\}$
Initialize: Randomly select initial cluster centers $c_1, c_2, \ldots, c_K$. Initialize membership weights w_{xik} for all $x_i \in \mathcal{X}$ and C_k;
repeat
  `/* Step 1: Update membership weights */`
  foreach $x_i \in \mathcal{X}$ *and* C_k **do**
    $w_{xik} \leftarrow \frac{1}{\sum_{j=1}^{K} \left(\frac{\|x_i - c_k\|}{\|x_i - c_j\|} \right)^{\frac{2}{\beta-1}}}$
  end
  `/* Step 2: Update cluster centers */`
  foreach C_k **do**
    $c_k \leftarrow \frac{\sum_{x_i \in C_k} w_{xik}^{\beta} x_i}{\sum_{x_i \in C_k} w_{xik}^{\beta}}$
  end
  `/* Step 3: Compute Loss Function */`
  $G_{\text{fuzzy}}(\mathcal{C}) \leftarrow \sum_{k=1}^{K} \sum_{x_i \in C_k} w_{xik}^{\beta} \|x_i - c_k\|^2$;
  `/* Step 4: Check convergence */`
  Stop if change in $G_{\text{fuzzy}}(\mathcal{C})$ is less than ϵ;
until *convergence or change in SSE* $< \epsilon$;

Algorithm 5: Fuzzy C-means clustering algorithm (FCM)

3. **Membership Weights** The membership values u_{ik} are updated using the kernel distances:

$$u_{ik} = \frac{1}{\sum_{j=1}^{K} \left(\frac{K(x_i,x_i) - 2\sum_{x_h \in C_k} u_{hk}^{\beta} K(x_i,x_h) + \sum_{x_g,x_h \in C_k} u_{gk}^{\beta} u_{hk}^{\beta} K(x_g,x_h)}{K(x_i,x_i) - 2\sum_{x_h \in C_j} u_{hj}^{\beta} K(x_i,x_h) + \sum_{x_g,x_h \in C_j} u_{gj}^{\beta} u_{hj}^{\beta} K(x_g,x_h)} \right)^{\frac{1}{\beta-1}}} \tag{25}$$

4. **Cluster Centers** The cluster centers c_k in the feature space are calculated as:

$$c_k = \frac{\sum_{i=1}^{n} u_{ik}^{\beta} \phi(x_i)}{\sum_{i=1}^{n} u_{ik}^{\beta}} \tag{26}$$

5. **Stopping Criterion** The algorithm stops when the change in the objective function $J(U, C)$ or the cluster centers c_k falls below a predefined threshold ϵ.

The algorithm can be found in Algorithm 6.

KFCM is an extension of FCM by using the Kernel trick to map low-dimensional and possibly non-linearly seperable data to a latent space where linear seperation is possible. It is particularly effective in scenarios with non-linearly separable clusters, such as those found in pattern recognition or biomedical applications. As we will see later in this section, KFCM is actually an intergration of Fuzzy K-Means Clustering and Kernel K-Means Clustering.

Input: Dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, number of clusters K, kernel function $K(x, y)$, fuzziness parameter β, stopping criteria ϵ
Output: Clusters $C_1, C_2, \ldots, C_K$ and cluster centers $c_1, c_2, \ldots, c_K$ in the feature space
Step 1: Kernel Mapping: Map each data point x_i into the feature space using kernel function $K(x_i, x_j)$;
Step 2: Initialization: Assign initial values for membership degrees u_{ik} and initialize cluster centers c_k;
repeat
  `/* Step 3: Update Membership Weights */`
  foreach $x_i \in \mathcal{X}$ *and* C_k **do**
    Update u_{ik} using the kernel-based distance formula:
    $u_{ik} = \frac{1}{\sum_{j=1}^{K} \left(\frac{\text{dist}_k(x_i)}{\text{dist}_j(x_i)} \right)^{\frac{1}{\beta-1}}}$
  end
  `/* Step 4: Update Cluster Centers */`
  foreach C_k **do**
    Update cluster center c_k using:
    $c_k = \frac{\sum_{i=1}^{n} u_{ik}^{\beta} \phi(x_i)}{\sum_{i=1}^{n} u_{ik}^{\beta}}$
  end
until *convergence (change in* $J(U, C) < \epsilon$*)*;

Algorithm 6: Kernel Fuzzy C-Means (KFCM) Algorithm

5.6 Kernel K-Means Clustering

Kernel K-Means [38] is an extension of the standard K-Means algorithm that works by projecting the data into a high-dimensional kernel space, where clusters can be more easily separated. This projection is achieved using kernel functions, however as we only care about distances between two points in the kernel space, an explicit representation of the projection function ϕ is not needed. Common kernel functions include polynomial, Gaussian (RBF), and sigmoid kernels, as shown in Table 1. We present Theorem 5.6 without proof.

Theorem 5.6. *(Mercer's Theorem) Let $K(x, y)$ be a continuous, symmetric, and positive semi-definite kernel function defined on a compact domain $\mathcal{X} \times \mathcal{X}$. Then, $K(x, y)$ can be expressed as:*

$$K(x, y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(y), \tag{27}$$

where:

- $\{\phi_i(x)\}$ *is an orthonormal set of eigenfunctions in* $L^2(\mathcal{X})$,
- $\{\lambda_i\}$ *are the non-negative eigenvalues associated with the eigenfunctions.*

The objective function for Kernel K-Means minimizes the sum of squared errors (SSE) in the high-dimensional feature space:

$$G_{\text{kernel}}(\mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|\phi(x_i) - c_k\|^2, \tag{28}$$

where c_k is the centroid of cluster C_k in the kernel space.

Kernel Type	**Kernel Function**
Polynomial Kernel	$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$
Gaussian Kernel (RBF)	$K(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{\|\mathbf{a}-\mathbf{b}\|^2}{2\sigma^2}\right)$
Sigmoid Kernel	$K(\mathbf{a}, \mathbf{b}) = \tanh(c(\mathbf{a} \cdot \mathbf{b}) + \theta)$

Table 1: Kernel Functions

The steps for Kernel K-Means clustering are summarized in Algorithm 7.

Input: Dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, number of clusters K, kernel function $K(x, y)$, stopping criteria ϵ
Output: Clusters $C_1, C_2, \ldots, C_K$
Step 1: Initialize. Randomly assign each point x_i to one of the K clusters;
repeat
`/* Step 2: Compute kernel distances */`
For each cluster C_k, compute the distance of each point x_i to the centroid in the kernel space:

$$\|\phi(x_i) - c_k\|^2 = K_{x_i x_i} - \frac{2}{|C_k|} \sum_{x_j \in C_k} K_{x_i x_j} + \frac{1}{|C_k|^2} \sum_{x_j, x_l \in C_k} K_{x_j x_l}.$$

`/* Step 3: Update cluster assignments */`
Assign each point x_i to the cluster with the nearest centroid in the kernel space;
`/* Step 4: Update centroids */`
Recompute cluster centroids c_k using the kernel function:

$$c_k = \frac{\sum_{x_i \in C_k} \phi(x_i)}{|C_k|}.$$

until *Convergence or change in* $G_{kernel}(C) < \epsilon$;

Algorithm 7: Kernel K-Means Clustering

The primary difference between standard K-Means and Kernel K-Means lies in the use of the kernel function $\phi(x)$, resulting in clustering in a high-dimensional kernel space. While this enables the clustering of non-linearly separable data, the computational complexity increases significantly, as the kernel matrix must be computed and stored. As a popular extensions of this algorithm, Weighted Kernel K-Means[17] incorporates weights into the clustering process, yielding nice theoretical properties and empirical results.

In Weighted Kernel K-Means , we assign a weight $w(x)$ for each data point x, allowing the algorithm to account for the varying importance of data points in the clustering process. Using a kernel function $K(x_i, x_j)$, which maps points into a high-dimensional feature space $\phi(x)$, the objective of Weighted Kernel K-Means is to minimize the following distortion function:

$$G_{\text{Weighted Kernel}}(\{C_j\}_{j=1}^k) = \sum_{j=1}^{k} \sum_{x \in C_j} w(x) \|\phi(x) - c_j\|^2, \tag{29}$$

where C_j represents the j-th cluster, and c_j is the weighted mean of the j-th cluster in the kernel-induced feature space, defined as:

$$c_j = \frac{\sum_{x \in C_j} w(x)\phi(x)}{\sum_{x \in C_j} w(x)}. \tag{30}$$

Definition 5.7. *(Distortion). We define the **distortion** of a cluster C_j as*

$$d(C_j) = \sum_{x \in C_j} w(x)\|\phi(x) - c_j\|^2 \tag{31}$$

The mean c_j is chosen such that it minimizes the intra-cluster distortion, making it the optimal cluster representative. Specifically,

$$c_j = \arg\min_{\mathbf{z}} \sum_{x \in C_j} w(x)\|\phi(x) - \mathbf{z}\|^2. \tag{32}$$

To compute the weighted distance between a point x and the cluster mean c_j, we expand the squared Euclidean distance in the kernel space as:

$$\|\phi(a) - c_j\|^2 = K(a,a) - \frac{2\sum_{b \in C_j} w(b)K(a,b)}{\sum_{b \in C_j} w(b)} + \frac{\sum_{b,c \in C_j} w(b)w(c)K(b,c)}{\left(\sum_{b \in C_j} w(b)\right)^2}, \tag{33}$$

The first term $K(a,a)$ is a constant for each point a and does not affect the cluster assignment, leaving only the second and third terms to be computed for clustering. The Weighted Kernel K-Means algorithm is described in Algorithm 8.

Input: Kernel matrix K, number of clusters k, weights $w(x)$ for each point
Output: Clusters $C_1, \ldots, C_k$
Step 1: Initialization. Randomly assign initial clusters $C_1^{(0)}, \ldots, C_k^{(0)}$;
Step 2: Iterative optimization. Set $t = 0$;
repeat
  foreach *point* $x \in \mathcal{X}$ **do**
    Assign x to the nearest cluster:
    $j^*(a) = \arg\min_j \|\phi(a) - c_j\|^2$, using Equation 33.
  end
  Update clusters: $C_j^{(t+1)} = \{a : j^*(a) = j\}$.
  Update $t = t + 1$;
until *Convergence*;
Step 3: Return $C_1, \ldots, C_k$;

Algorithm 8: Weighted Kernel K-Means

The computational cost of the algorithm is dominated by the calculation of weighted distances. The second term in the weighted distance computation requires $O(n)$ operations per data point and is computed once per iteration. The third term is cluster-dependent and requires $O(n^2)$ operations per iteration, as it involves summing over all points within each cluster. The overall time complexity

is $O(n^2\tau)$, where τ denotes the total number of iterations. Initial computation of the kernel matrix K adds an additional cost of $O(n^2 m)$, where m is the dimensionality of the original data points.

Theorem 5.8. *(Weighted Kernel K-means is equivalent to trace maximization)*
Given Kernel Matrix K, weight matrix W, the objective of Weighted Kernel K-means Clustering algorithm can be expressed as $\arg\max_{Y \in \mathbb{R}^{n\times k}} trace(Y^T W^{1/2} K W^{1/2} Y)$.

Proof. For a cluster C_j, denote the sum of the weights w of the points in C_j to be s_j; in other words, $s_j = \sum_{x\in C_j} w(x)$. Then, let $W \in \mathbb{R}^{n\times n}$ be the diagonal matrix of the w weights for each data point, and $W_j \in \mathbb{R}^{|C_j|\times|C_j|}$ denotes the diagonal matrix of w weights with restriction of the data points are in C_j. Then we can rewrite the representative element c_j as

$$c_j = \Phi_j \frac{W_j \mathbf{e}}{s_j}, \tag{34}$$

where Φ_j is the diagonal matrix of points associated with cluster C_j (after the ϕ mapping), i.e., $\Phi = diag[\phi(x_1), \phi(x_2), \ldots, \phi(x_n)]$, and $\mathbf{e}$ is the vector of all ones of size $|C_j|$.

We can rewrite the distortion of cluster C_j to be:

$$\begin{aligned} d(C_j) &= \sum_{x\in C_j} w(x)\|\phi(x) - c_j\|^2 && (35)\\ &= \sum_{x\in C_j} w(x) \left\|\phi(x) - \Phi_j \frac{W_j\mathbf{e}}{s_j}\right\|^2 && (36)\\ &= \left\|(\Phi_j - \Phi_j \frac{W_j \mathbf{e}\mathbf{e}^T}{s_j}) W_j^{1/2}\right\|_F^2 && (37)\\ &= \left\|\Phi_j W_j^{1/2}\left(I - \frac{W_j^{1/2}\mathbf{e}\mathbf{e}^T W_j^{1/2}}{s_j}\right)\right\|_F^2. && (38) \end{aligned}$$

As $\text{trace}(AA^T) = \text{trace}(A^T A) = \|A\|_F^2$, and $I - \frac{W_j^{1/2}\mathbf{e}\mathbf{e}^T W_j^{1/2}}{s_j} = P$ is an orthogonal projection matrix, (because $s_j = \mathbf{e}^T W_j \mathbf{e}$, therefore $P^2 = P$), we get that

$$d(C_j) = \text{trace}\left(W_j^{1/2}\Phi_j^T\Phi_j W_j^{1/2}\right) - \frac{\mathbf{e}^T W_j^{1/2}}{\sqrt{s_j}}\Phi_j^T\Phi_j\frac{W_j^{1/2}\mathbf{e}}{\sqrt{s_j}}. \tag{39}$$

Rearranging the order of the data points, we can represent the full matrix of points as $\Phi = diag[\Phi_1, \Phi_2, \ldots, \Phi_k]$, then we have that

$$\begin{aligned} G_{\text{Weighted Kernel}}(\{C_j\}_{j=1}^k) &= \text{trace}(W^{1/2}\Phi^T\Phi W^{1/2}) - \text{trace}(Y^T W^{1/2}\Phi^T\Phi W^{1/2} Y) && (40)\\ &= \text{trace}(\Phi W \Phi^T) - \text{trace}(Y^T W^{1/2}\Phi^T\Phi W^{1/2} Y) && (41) \end{aligned}$$

where

$$Y = \begin{bmatrix} \frac{W_1^{1/2}\mathbf{e}}{\sqrt{s_1}} & & & \\ & \frac{W_2^{1/2}\mathbf{e}}{\sqrt{s_2}} & & \\ & & \ddots & \\ & & & \frac{W_k^{1/2}\mathbf{e}}{\sqrt{s_k}} \end{bmatrix}. \tag{42}$$

$Y \in \mathbb{R}^{n\times k}$ is an orthonormal matrix, i.e. $Y^TY = I$.

Since trace$(\Phi W \Phi^T)$ is a constant, minimizing the loss function, as expressed in Equation 41 is equivalent maximizing trace$(Y^T W^{1/2}\Phi^T \Phi W^{1/2} Y)$. As the kernel matrix of the data points $K = \Phi^T\Phi$, we can rewrite this expression as

$$\arg\max_{Y\in\mathbb{R}^{n\times k}} \text{trace}(Y^T W^{1/2} K W^{1/2} Y). \tag{43}$$

For the relaxed version of this problem, i.e. we allow Y to be an arbitrary orthonormal matrix, we can obtain an optimal Y by taking the top k eigenvectors of $W^{1/2}KW^{1/2}$. This is a standard algebra result, and the detailed proof is omitted. □

As it is a well-celebrated result that Spectral Graph Clustering [7] is also equivalent to trace maximization [46], therefore this theorem establishes the theoretical connection between weighted kernel k-means and spectral graph clustering.

5.7 Weighted K-Means Clustering

Weighted K-Means (WK-Means) [22] is an extension of the standard K-Means algorithm that introduces a feature weighting mechanism to account for the varying importance of features. Through iteratively learns weights for different features, WK-Means improves clustering performance, especially in datasets where certain features are more informative than others. A user-defined parameter β controls the sensitivity of the algorithm to feature weights.

In WK-Means algorithm, the distance between a data point x_i and a cluster centroid c_k is defined using a weighted Euclidean distance as:

$$d(x_i, c_k) = \sum_{v=1}^{M} w_v^{\beta} \|x_{iv} - c_{kv}\|^2. \tag{44}$$

At each iteration, WK-Means first assigns data points to the nearest cluster based on this weighted distance and updating the cluster centroids. Once the centroids are updated, the feature weights w_v are recalculated using Equation 45.

$$w_v = \frac{1}{\sum_{u=1}^{M} \left(\frac{D_v}{D_u}\right)^{\frac{1}{\beta-1}}}, \tag{45}$$

where D_v is the sum of within-cluster variances for feature v, defined as:

$$D_v = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_{iv} - c_{kv})^2. \tag{46}$$

The weight update ensures that the weights w_v are normalized such that:

$$\sum_{v=1}^{M} w_v = 1. \tag{47}$$

In other words, the WK-Means algorithm starts by initializing random centroids and assigning equal weights to all features. During each iteration, the algorithm alternates between assigning data points to clusters based on weighted distances and updating the centroids and feature weights. The full algorithm is shown in Algorithm 9.

Input: Dataset $\mathcal{X}$ with M features, number of clusters K, user-defined parameter β, stopping criteria
Output: Clusters $C_1, C_2, \ldots, C_K$, feature weights w_v, and centroids c_k
Step 1: Initialization. Choose K random centroids and initialize M feature weights such that they sum to 1;
repeat
  `/* Step 2: Assign points to clusters */`
  Assign each data point x_i to the closest centroid c_k based on $d(x_i, c_k)$;
  `/* Step 3: Update centroids */`
  Recompute the centroids c_k for each cluster;
  `/* Step 4: Update feature weights */`
  Update the feature weights using w_v;
until *Convergence criterion is met*;

Algorithm 9: Weighted K-Means Clustering (WK-Means)

By incorporating feature importance, WK-Means empirically achieves better clustering results compared to standard K-Means, especially for datasets with heterogeneous feature importance. However, the computational cost of WK-Means is higher than traditional K-Means due to the additional weight learning step.

5.8 Intelligent K-Means Clustering

Intelligent K-Means (IK-Means) [10] is a clustering method that follows the principle: *the farther a point is from the centroid, the more we care about it*. Unlike traditional K-Means, IK-Means introduces a deterministic way of selecting initial centroids by considering points that are farthest from the existing centroid, enabling better clustering for datasets with wide scatter. This method leverages the ideas of principal component analysis (PCA) to identify points with maximum data scatter and uses them as potential cluster centroids. These centroids often represent anomalous patterns in the data, and the clusters derived from them are referred to as *anomalous pattern clusters*.

The first step of IK-Means involves calculating the centroid for the entire dataset, referred to as the center of gravity c_g.

$$c_g = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{48}$$

Once c_g is determined, the algorithm identifies the point c farthest from c_g:

$$c = \arg\max_{x_i \in \mathcal{X}} \|x_i - c_g\|^2. \tag{49}$$

This point c becomes a candidate for an anomalous cluster centroid. The algorithm then creates a cluster S_{iter} by assigning all points x_i to S_{iter} if they are closer to c than to c_g. Namely:

$$S_{\text{iter}} = \{x_i \in \mathcal{X} \mid d(x_i, c) < d(x_i, c_g)\}, \tag{50}$$

where $d(x_i, c)$ denotes the distance between x_i and c. Once the cluster S_{iter} is formed, the centroid of this cluster, s_g, is updated as the mean of the points in S_{iter}:

$$s_g = \frac{1}{|S_{\text{iter}}|} \sum_{x_i \in S_{\text{iter}}} x_i. \tag{51}$$

The algorithm iteratives between these steps, until one of the stopping criteria is met. The steps of IK-Means are summarized in Algorithm 10.

Input: Dataset $\mathcal{X}$
Output: Clusters $C_1, C_2, \ldots, C_K$ and centroids $c_1, c_2, \ldots, c_K$
Step 1: Initialize. $c_g = \frac{1}{n} \sum_{i=1}^{n} x_i$.
repeat
  `/* Step 2: Select new centroid */`
  $c = \arg\max_{x_i \in \mathcal{X}} \|x_i - c_g\|^2$.
  `/* Step 3: Cluster assignment */`
  $S_{\text{iter}} = \{x_i \in \mathcal{X} \mid d(x_i, c) < d(x_i, c_g)\}$.
  `/* Step 4: Update centroid */`
  $s_g = \frac{1}{|S_{\text{iter}}|} \sum_{x_i \in S_{\text{iter}}} x_i$. Set $c_g = s_g$.
  `/* Step 5: Prune small clusters */`
  Discard clusters smaller than a pre-specified threshold.
until *Stopping criterion is met*;

Algorithm 10: Intelligent K-Means (IK-Means) Clustering

Unlike traditional K-Means, which is non-deterministic due to random initialization, IK-Means is a deterministic algorithm. It is particularly effective in scenarios where clusters are spread widely across the dataset.

5.9 Bisecting K-Means Clustering

Bisecting K-means clustering [44] builds a binary tree and iteratively splits the leaf nodes C into two child nodes C_1, C_2 by running 2-means. The algorithm for Bisecting K-means clustering is given in Algorithm 11.

In line 2, the parent cluster to be split is initialized. Then, the algorithn runs a 2-means clustering algorithm for I times. The resulting partition C_1, C_2 will be used to split the current node and the larger among the split clusters is made as the new parent for further splitting.

This algorithm intergrates divisive hierarchical clustering method with K-Means. However, Bisecting K-means has much higher computation complexity compared to standard K-Means.

Input: Dataset $\mathcal{X}$, desired number of clusters K, maximum iterations I
Output: Clusters $C_1, C_2, \ldots, C_K$
Step 1: Initialize. Start with the entire dataset as a single cluster C;
repeat
  `/* Step 2: Select cluster to split */`
  Choose the parent cluster C to be split;
  `/* Step 3: Bisect the cluster */`
  repeat
    Randomly initialize two centroids within C;
    Assign points in C to the nearest centroid;
    Recompute centroids and evaluate inter-cluster dissimilarity;
  until *I iterations or convergence*;
  Split C into two subclusters C_1 and C_2;
  `/* Step 4: Update parent cluster */`
  Choose the larger subcluster (if applicable) as the next parent cluster;
until *The number of clusters equals K*;

Algorithm 11: Bisecting K-Means Clustering

6 Experiments

6.1 Experimental Setup

6.1.1 Datasets

We conducted extensive experiments on four well-known benchmark datasets from the UCI Machine Learning Repository [28]:

- **Breast Cancer Wisconsin** [45]: A binary classification dataset with $n = 569$ samples and $d = 30$ features
- **Iris** [20]: A dataset containing $n = 150$ samples with $d = 4$ features categorized into three classes
- **Wine** [2]: Consists of $n = 178$ samples with $d = 13$ features divided into three classes
- **Digits (0-4)** [5]: A subset of the handwritten digits dataset containing the first five digits, with $d = 64$ features

6.1.2 Algorithms

We implemented and compared ten variants of K-means clustering we discussed earlier:

- **Standard K-means** [32]: The classical K-means algorithm with Euclidean distance
- **K-medoids** [41]: A more robust variant using actual data points as centroids
- **K-medians** [24]: Uses Manhattan distance and median centers
- **K-modes** [11]: Adapted for handling continuous data through discretization

- **Mean Shift** [12]: A density-based approach with adaptive bandwidth
- **Fuzzy K-means** [37]: Allows soft cluster assignments
- **Kernel K-means** [38]: Incorporates non-linear relationships using RBF kernel
- **Weighted K-means** [22]: Assigns feature importance based on variance
- **Intelligent K-means** [10]: Uses strategic initialization of centroids
- **Bisecting K-means** [44]: Implements hierarchical divisive clustering

6.1.3 Data Preprocessing

To ensure reliable and consistent clustering results, we implemented a comprehensive preprocessing pipeline:

Outlier Treatment We employed the Interquartile Range (IQR) method [9] to handle outliers:

$$\begin{aligned} \text{IQR} &= Q_3 - Q_1 \\ \text{Lower Bound} &= Q_1 - 1.5 \times \text{IQR} \\ \text{Upper Bound} &= Q_3 + 1.5 \times \text{IQR} \end{aligned}$$

where Q_1 and Q_3 represent the first and third quartiles respectively. Values outside these bounds were clipped to the respective boundaries.

Feature Standardization All features were standardized using Standard Scaler [4] to ensure equal feature contribution:

$$x_i' = \frac{x_i - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the feature.

6.1.4 Evaluation Metrics

To comprehensively evaluate the clustering performance, we employed the following metrics [3, 35, 25]:

Clustering Accuracy This metric quantifies the agreement between predicted cluster labels and true class labels:

$$\text{Accuracy} = \frac{\text{Correctly identified class}}{\text{Total number of class}} \times 100$$

The accuracy is computed after finding the optimal one-to-one mapping between predicted clusters and true classes using the Hungarian algorithm [36].

Adjusted Rand Index (ARI) The ARI [23] measures the similarity between two clustering partitions. For a dataset S with α elements divided into two partitions $Y = \{Y_1, Y_2, \ldots, Y_b\}$ and $X = \{X_1, X_2, \ldots, X_c\}$, the ARI is calculated as:

$$\text{ARI} = \frac{\sum_{ij} \binom{a_{ij}}{2} - [\sum_i \binom{r_i}{2} \sum_j \binom{s_j}{2}] / \binom{\alpha}{2}}{\frac{1}{2}[\sum_i \binom{r_i}{2} + \sum_j \binom{s_j}{2}] - [\sum_i \binom{r_i}{2} \sum_j \binom{s_j}{2}] / \binom{\alpha}{2}}$$

where:

- a_{ij} is the number of elements in common between partitions X_i and Y_j
- r_i is the sum of the i-th row
- s_j is the sum of the j-th column

The ARI score ranges from 0 (random clustering) to 1 (perfect clustering agreement).

6.1.5 Experimental Procedure

The experiments were conducted following this protocol:

1. For each dataset, we generated multiple subsets of varying sizes: $\{10\%, 25\%, 50\%, 75\%, 100\%\}$ of the original data
2. Each algorithm was run with the true number of clusters (k) known from the ground truth
3. For stability assessment, each experiment was conducted with fixed random seed (42)
4. Performance metrics were collected across different data sizes to analyze scalability
5. Results were visualized using PCA-reduced 2D projections and performance heat maps

6.2 Experiments Results and Analysis

For comprehensive visualization of our experimental results, we present additional plots in Appendix A.1, A.2 and A.3, including performance scaling, runtime analysis and clustering visualizations across different datasets. The main results and analysis are discussed in detail in the following sections.

6.2.1 Performance Analysis Across Datasets

As illustrated in Figure 2, our experimental evaluation on four standard datasets—Breast Cancer, Iris, Wine, and Digits—highlights several notable trends in clustering performance. On structured datasets like Wine and Breast Cancer, most algorithms demonstrated strong accuracy, with values exceeding 0.85 in most cases. KMedians and KMedoids consistently delivered stable results across all datasets. Among these, the Wine dataset emerged as the most conducive to clustering, with Fuzzy K-means achieving an accuracy of 0.967, closely followed by KMedians at 0.965.

In contrast, the Digits dataset posed significant challenges, with accuracy rates ranging widely from 0.222 to 0.797. Certain algorithms, such as IKMeans (0.228), Fuzzy K-means (0.402), and

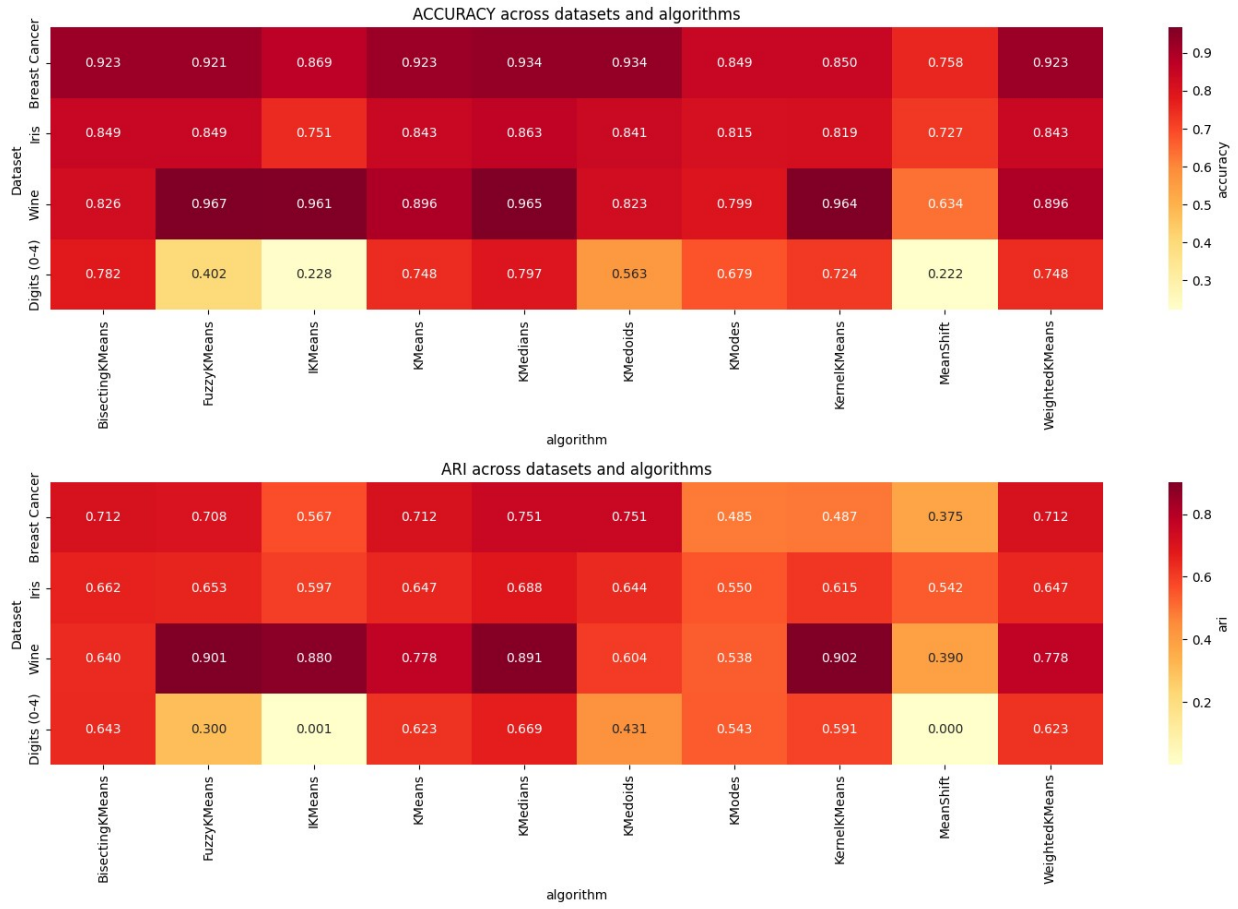

Figure 2: Performances across four datasets (Breast Cancer, Iris, Wine, and Digits)

MeanShift (0.222), exhibited notably poor performance. This decline in accuracy on the high-dimensional image dataset likely stems from the inherent complexity of handwritten digit variations and the sensitivity of these algorithms to high-dimensional feature spaces. These findings suggest that preprocessing techniques like dimensionality reduction could be instrumental in improving clustering performance on such complex datasets.

6.2.2 Analysis of Performance Scaling with Dataset Size

Figure 3 illustrates how different clustering algorithms scale with increasing dataset size on the Breast Cancer dataset (results for other datasets are provided in Appendix A.1). Several key patterns emerge from this analysis:

First, traditional K-means variants (KMeans, KMedians, KMedoids) demonstrate remarkable stability across different dataset sizes, maintaining consistently high performance (accuracy > 0.90) even as the dataset grows. This stability suggests these algorithms are well-suited for real-world applications where data volume may vary.

Second, while algorithms show considerable performance variation at smaller dataset sizes (100-200 samples), their performance tends to stabilize as the dataset size increases beyond 300

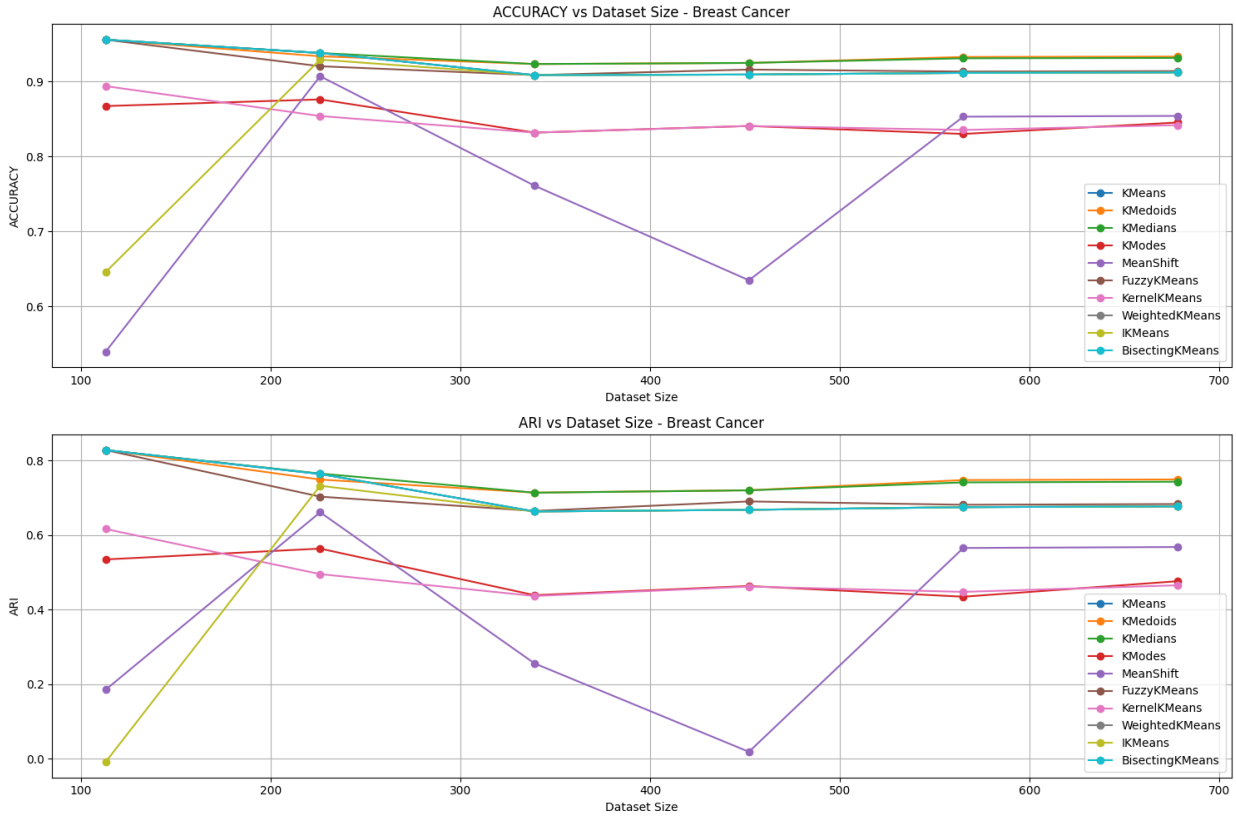

Figure 3: Accuracy and ARI comparison of Breast Cancer Datasets different sizes

samples. This is particularly evident in the ARI scores, where the performance gap between algorithms narrows at larger dataset sizes.

However, not all algorithms exhibit such stability. MeanShift, for instance, shows significant performance degradation as the dataset size increases, with its accuracy dropping from 0.90 at around 200 samples to below 0.70 at around 400 samples. This suggests that certain algorithms may require careful parameter tuning or may be better suited for specific dataset size ranges.

Notably, the correlation between accuracy and ARI scores remains consistent across dataset sizes, though ARI scores are generally lower. This consistent gap between accuracy and ARI (approximately 0.2-0.3) indicates that while algorithms may achieve high accuracy, their cluster assignments might not fully capture the underlying data structure.

The remaining three datasets show similar patterns but with dataset-specific variations (see Appendix A.1). These results highlight the importance of considering dataset size when selecting clustering algorithms for practical applications.

6.2.3 Runtime Analysis with Dataset Size

The runtime analysis reveals significant differences in computational efficiency across algorithms (Figure 4). From the results on the Digits dataset (additional datasets shown in Appendix A.2), we observe three distinct scaling patterns:

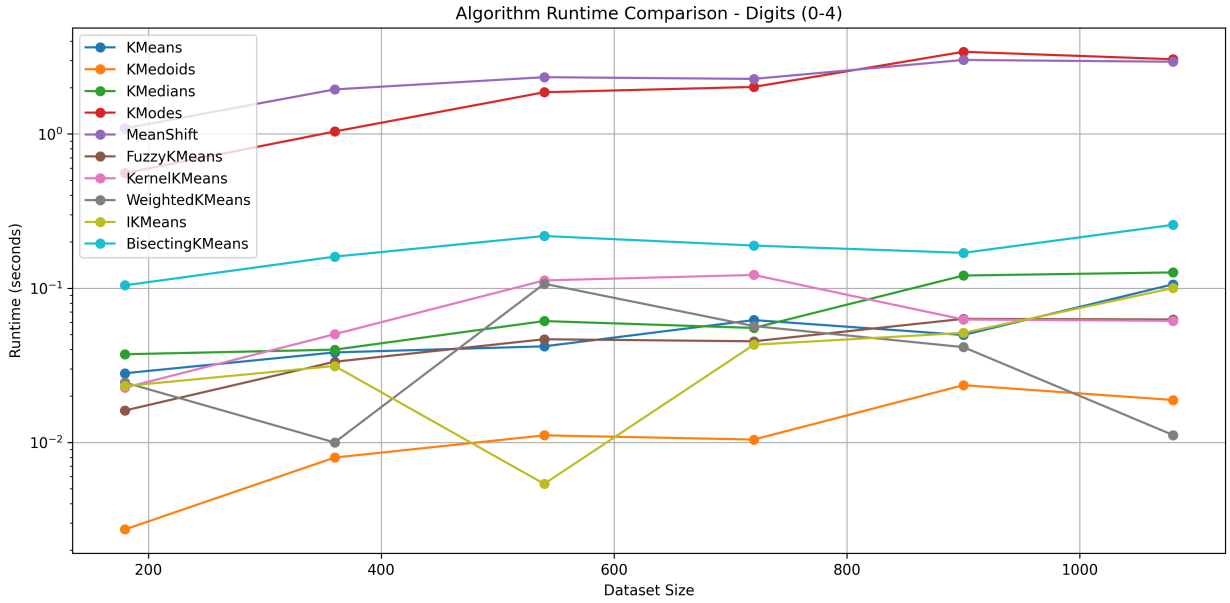

Figure 4: Runtime scaling with dataset size on Digits dataset.

First, most K-means variants demonstrate moderate scaling with dataset size, maintaining relatively stable performance around or less than 10^{-1} seconds even as the dataset grows. This efficiency makes them practical choices for many real-world applications.

Second, more sophisticated algorithms show notably different scaling behaviors. MeanShift and KModes exhibit substantially higher computational costs, with runtimes approaching and exceeding 1 second for larger datasets. This significant increase in runtime suggests careful consideration is needed when applying these algorithms to larger datasets.

Notably, while some algorithms show consistent scaling patterns, others exhibit more erratic behavior, particularly at transition points in dataset size. This variability highlights the importance of testing algorithms across a range of dataset sizes when computational efficiency is a concern.

6.3 Recommendations and Future Directions

Based on our comprehensive experimental analysis, we can provide several key recommendations for algorithm selection and usage:

For general-purpose clustering tasks, KMedians or KMedoids emerge as robust choices, demonstrating both consistent performance across different data types and efficient runtime scaling. These algorithms maintain high accuracy (> 0.90) while keeping computational costs manageable ($10^{-2} \sim 10^{-1}$ seconds for 1000 examples), making them particularly suitable for real-world applications with varying dataset sizes.

When working with well-structured datasets with clear cluster boundaries, Fuzzy K-means and Kernel K-means can provide superior accuracy (up to 0.967 on the Wine dataset), though at the cost of increased computational complexity. These algorithms are best suited for applications where accuracy is prioritized over computational efficiency.

For complex, high-dimensional datasets like Digits, our analysis reveals significant challenges

that require careful consideration, like the substantial performance variation (0.222 to 0.797) suggests the need for preprocessing steps such as dimensionality reduction and the disparity between accuracy and ARI scores indicates the need for more robust evaluation metrics.

6.3.1 Future Directions

Future research could focus on extending the analysis to a wider variety of datasets with diverse characteristics to enhance generalizability. It would also be valuable to investigate the impact of different preprocessing techniques, particularly for high-dimensional data, as well as to develop hybrid approaches that balance computational efficiency and clustering accuracy. Additionally, exploring adaptive parameter tuning strategies could improve the performance of size-sensitive algorithms. Finally, further examination of factors such as initialization and their influence on clustering outcomes could provide deeper insights into optimizing these methods [39].

The relationship between dataset characteristics and algorithm performance suggests that no single algorithm is universally superior. Instead, algorithm selection should be guided by specific application requirements, considering both performance metrics and computational constraints.

7 Conclusion

This study presents a systematic investigation of k-means clustering and its variants, offering a comprehensive analysis of their algorithmic properties and practical performance. Through theoretical analysis, we have elucidated the computational complexity and convergence properties of these algorithms, providing a mathematical general framework for understanding their behavior. Our experimental evaluations across diverse datasets validate these theoretical findings while practical insights into their strengths and limitations.

References

[1] *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley Sons, Ltd, 1990.

[2] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

[3] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8), 2020.

[4] Selim Aksoy and Robert M. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5):563–582, 2001. Image/Video Indexing and Retrieval.

[5] E. Alpaydin and C. Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50P49.

[6] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[7] Benjamin Auffarth. Spectral graph clustering. 01 2007.

[8] Mihai Badoiu, Artur Czumaj, Piotr Indyk, and Christian Sohler. Facility location in sublinear time. volume 3580, pages 866–877, 07 2005.

[9] H. Beyer. Tukey, john w.: Exploratory data analysis. addison-wesley publishing company reading, mass. — menlo park, cal., london, amsterdam, don mills, ontario, sydney 1977, xvi, 688 s. *Biometrical Journal*, 23(4):413–414, 1981.

[10] Johannes Blömer, Christiane Lammersen, Melanie Schmidt, and Christian Sohler. Theoretical analysis of the *k*-means algorithm - a survey. *arXiv preprint arXiv: 1602.08254*, 2016.

[11] Miguel Á. Carreira-Perpiñán and Weiran Wang. The k-modes algorithm for clustering. *arXiv preprint arXiv: 1304.6478*, 2013.

[12] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[13] Vincent Cohen-Addad, Benjamin Guedj, Varun Kanade, and Guy Rom. Online k-means clustering. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1126–1134. PMLR, 2021.

[14] Andreas Darmann and Janosch Döcker. On simplified np-complete variants of not-all-equal 3-sat and 3-sat. *arXiv preprint arXiv: 1908.04198*, 2019.

[15] Sanjoy Dasgupta. The hardness of k-means clustering. 2008.

[16] Harshada S. Deshmukh and P. L. Ramteke. Comparing the techniques of cluster analysis for big data. 2016.

[17] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means, spectral clustering and normalized cuts. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 551-556, 07 2004.

[18] Zhong dong Wu, Wei xin Xie, and Jian ping Yu. Fuzzy c-means clustering algorithm based on kernel method. In *Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2003*, pages 49–54, 2003.

[19] Vladimir Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1):65–75, 2002.

[20] R. A. Fisher. Iris. UCI Machine Learning Repository, 1936. DOI: https://doi.org/10.24432/C56C76.

[21] Kurt Hornik, Ingo Feinerer, Martin Kober, and Christian Buchta. Spherical k-means clustering. *Journal of Statistical Software*, 50:1–22, 09 2012.

[22] J.Z. Huang, M.K. Ng, Hongqiang Rong, and Zichen Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005.

[23] Lawrence J. Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[24] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[25] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[26] Leonard Kaufman and Peter Rousseeuw. *Finding Groups in Data: An Introduction To Cluster Analysis*. 01 1990.

[27] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[28] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository, https://archive.ics.uci.edu, 2025. Accessed 2025-01-07.

[29] R. Krishnapuram and J.M. Keller. The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[30] Yugal Kumar and Gadadhar Sahoo. A two-step artificial bee colony algorithm for clustering. *Neural Computing and Applications*, 28:537–551, 2017.

[31] Leon S. Lasdon and Daniel Tabak. Optimization theory of large systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:300–301, 1970.

[32] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[33] Pradipta Maji and Sankar Pal. Rough set based generalized fuzzy c -means algorithm and quantitative indices. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 37:1529–40, 12 2007.

[34] Ramgopal Mettu and Greg Plaxton. Optimal time bounds for approximate clustering. *arXiv preprint arXiv: 1301.0587*, 2012.

[35] Mehryar Mohri. Foundations of machine learning, 2018.

[36] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[37] Janmenjoy Nayak, Bighnaraj Naik, and Prof. Dr. H. Behera. *Fuzzy C-Means (FCM) Clustering Algorithm: A Decade Review from 2000 to 2014*, volume 32, pages 133–149. 01 2015.

[38] Debolina Paul, Saptarshi Chakraborty, Swagatam Das, and Jason Xu. Kernel k-means, by all means: Algorithms and strong consistency. *arXiv preprint arXiv: 2011.06461*, 2020.

[39] Chandan Reddy and Bhanukiran Vinzamuri. *A Survey of Partitional and Hierarchical Clustering Algorithms*, pages 87–110. 09 2018.

[40] I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44:522–536, 1938.

[41] Erich Schubert and Peter J. Rousseeuw. Faster k-medoids clustering: Improving the pam, clara, and clarans algorithms. *arXiv preprint arXiv: 1810.05691*, 2018.

[42] D. Sculley. Web-scale k-means clustering. In *The Web Conference*, 2010.

[43] Shokri Z Selim and Mohamed A Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on pattern analysis and machine intelligence*, (1):81–87, 1984.

[44] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. *Proceedings of the International KDD Workshop on Text Mining*, 06 2000.

[45] Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.

[46] Yu and Shi. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 313–319 vol.1, 2003.

A Experiments Results

A.1 Accuracy comparison

Detailed results of different algorithms' accuracy are shown in Figure 5, Figure 6 and Figure 7.

A.2 Runtime comparison

Detailed results for runtime comparison are shown in Figure 8, Figure 9 and Figure 10.

A.3 Visualizations

Visualization results are shown in Figure 11, Figure 12, Figure 13 and Figure 14.

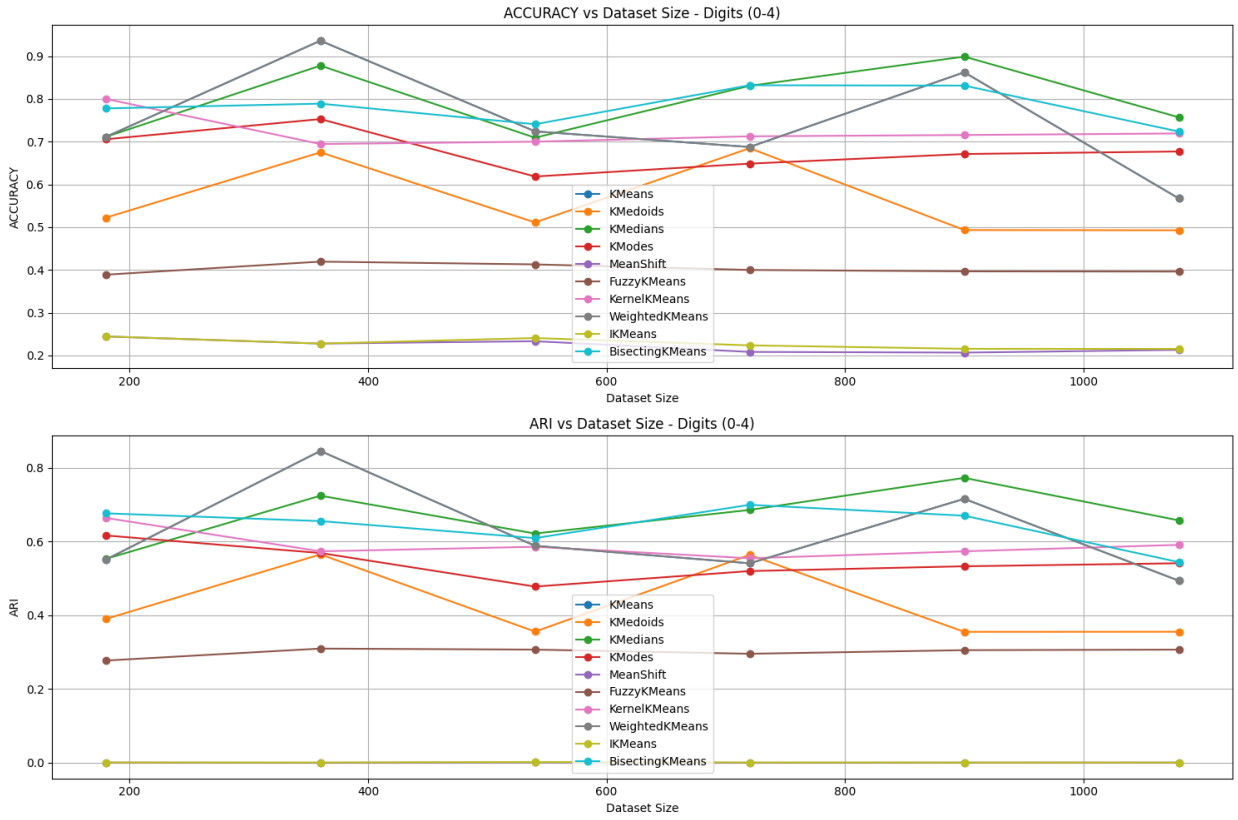

Figure 5: Accuracy and ARI comparison of Digits Datasets

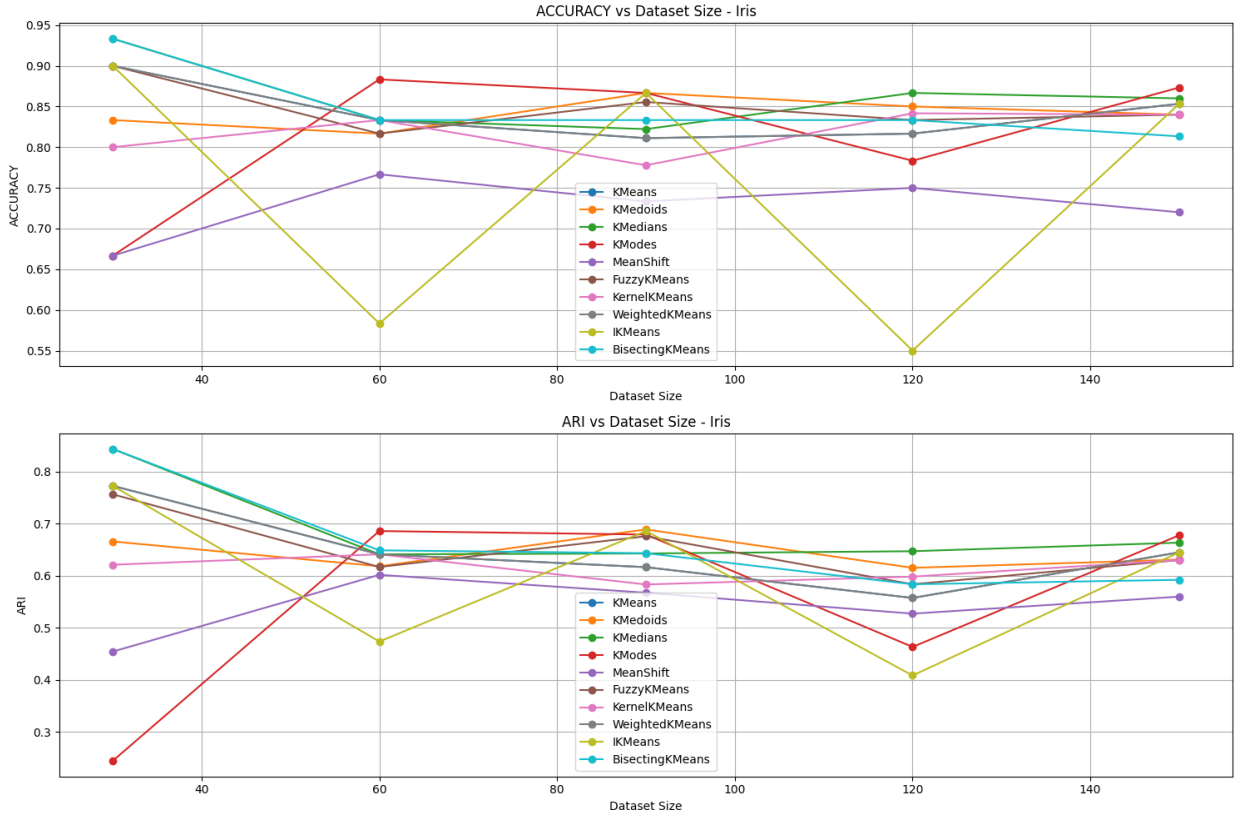

Figure 6: Accuracy and ARI comparison of Iris Datasets

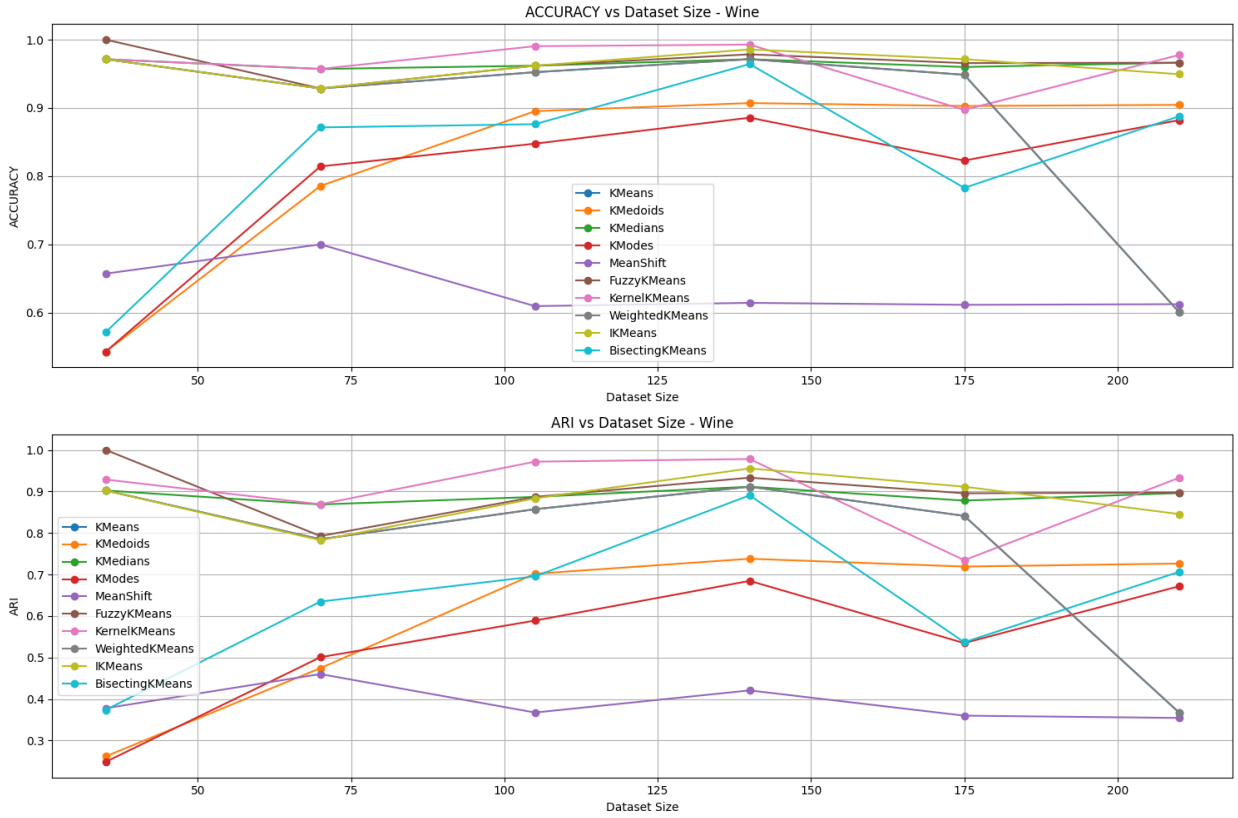

Figure 7: Accuracy and ARI comparison of Wine Datasets

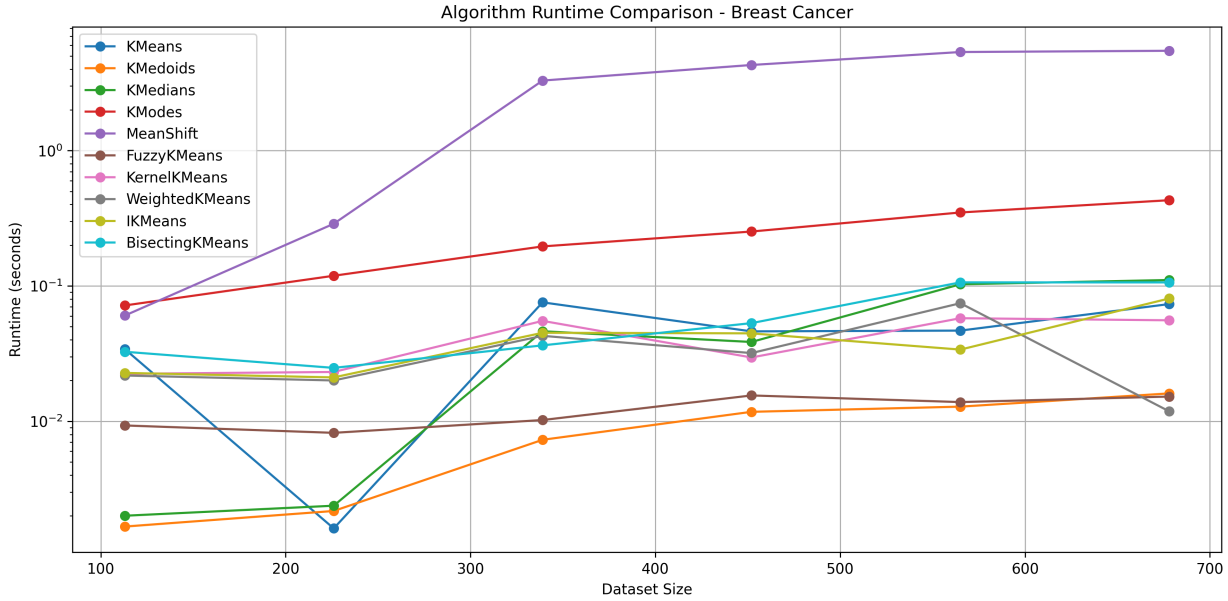

Figure 8: Runtime comparison of Breast Cancer Datasets

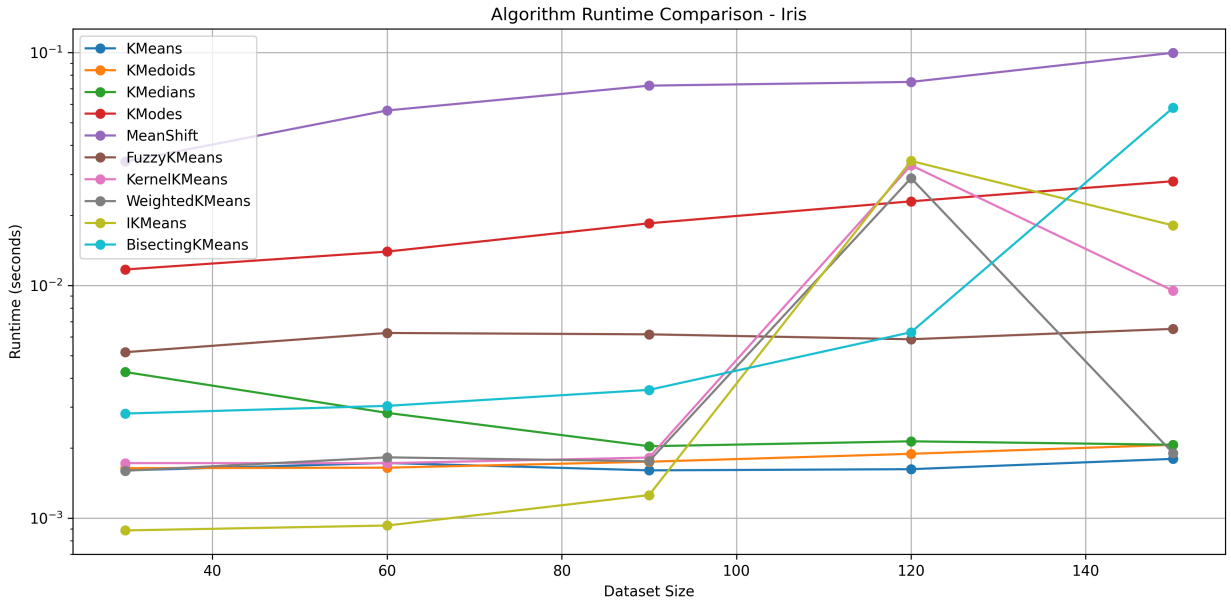

Figure 9: Runtime comparison of Iris Datasets

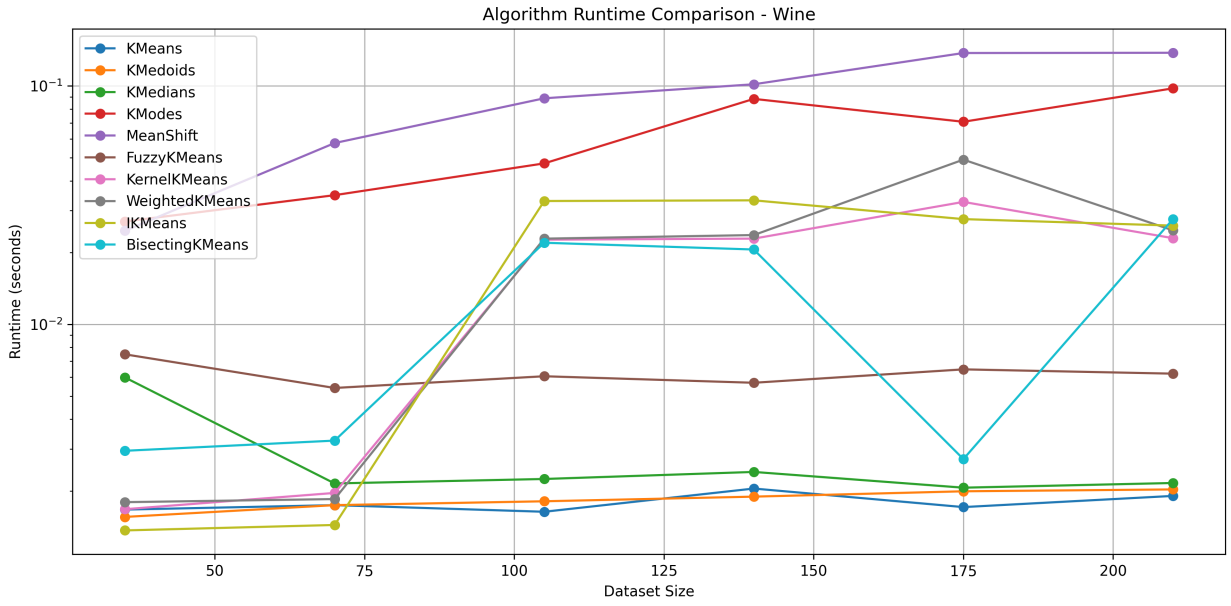

Figure 10: Runtime comparison of Wine Datasets

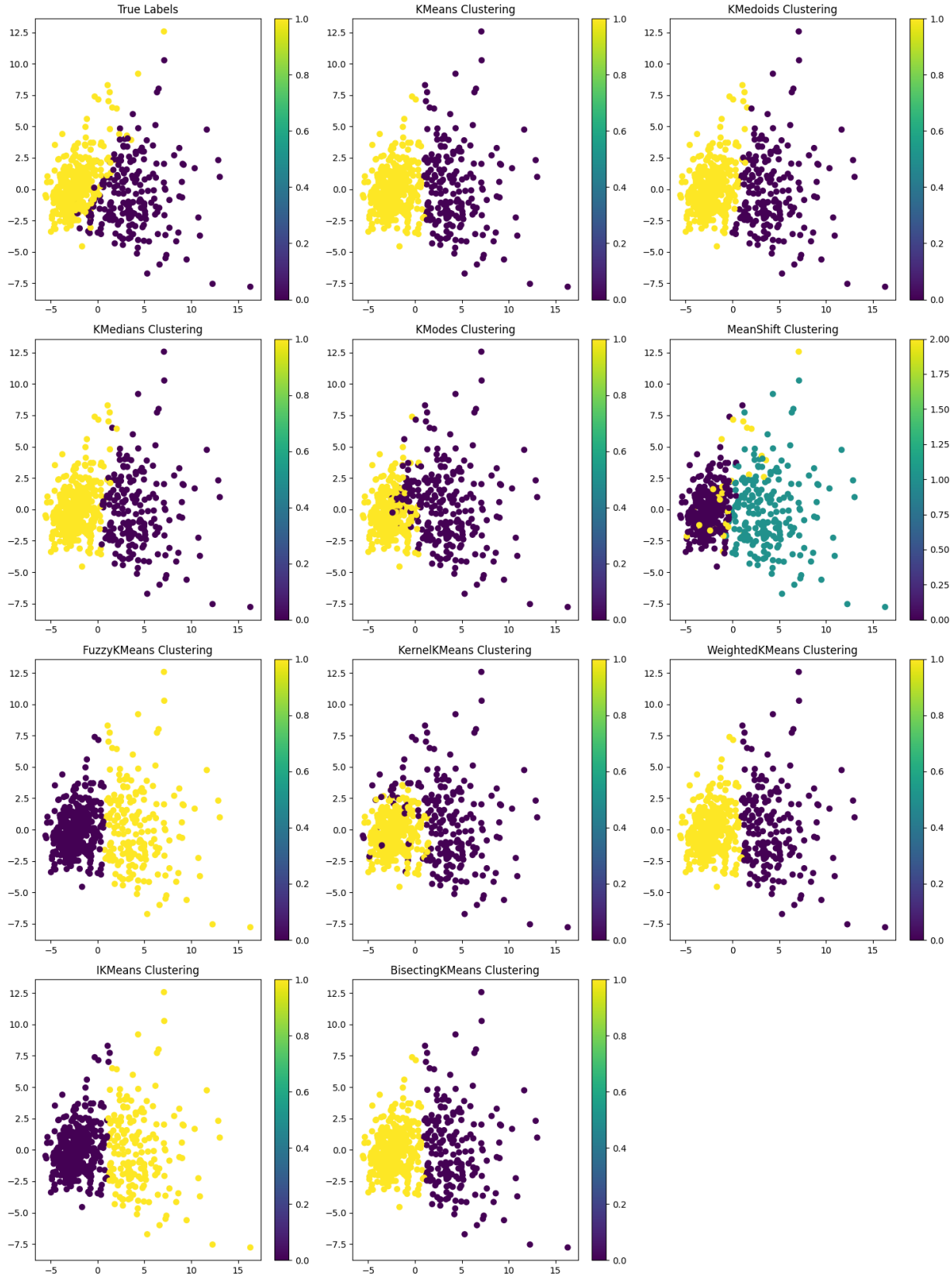

Figure 11: Clustering visualization of Breast Cancer Datasets

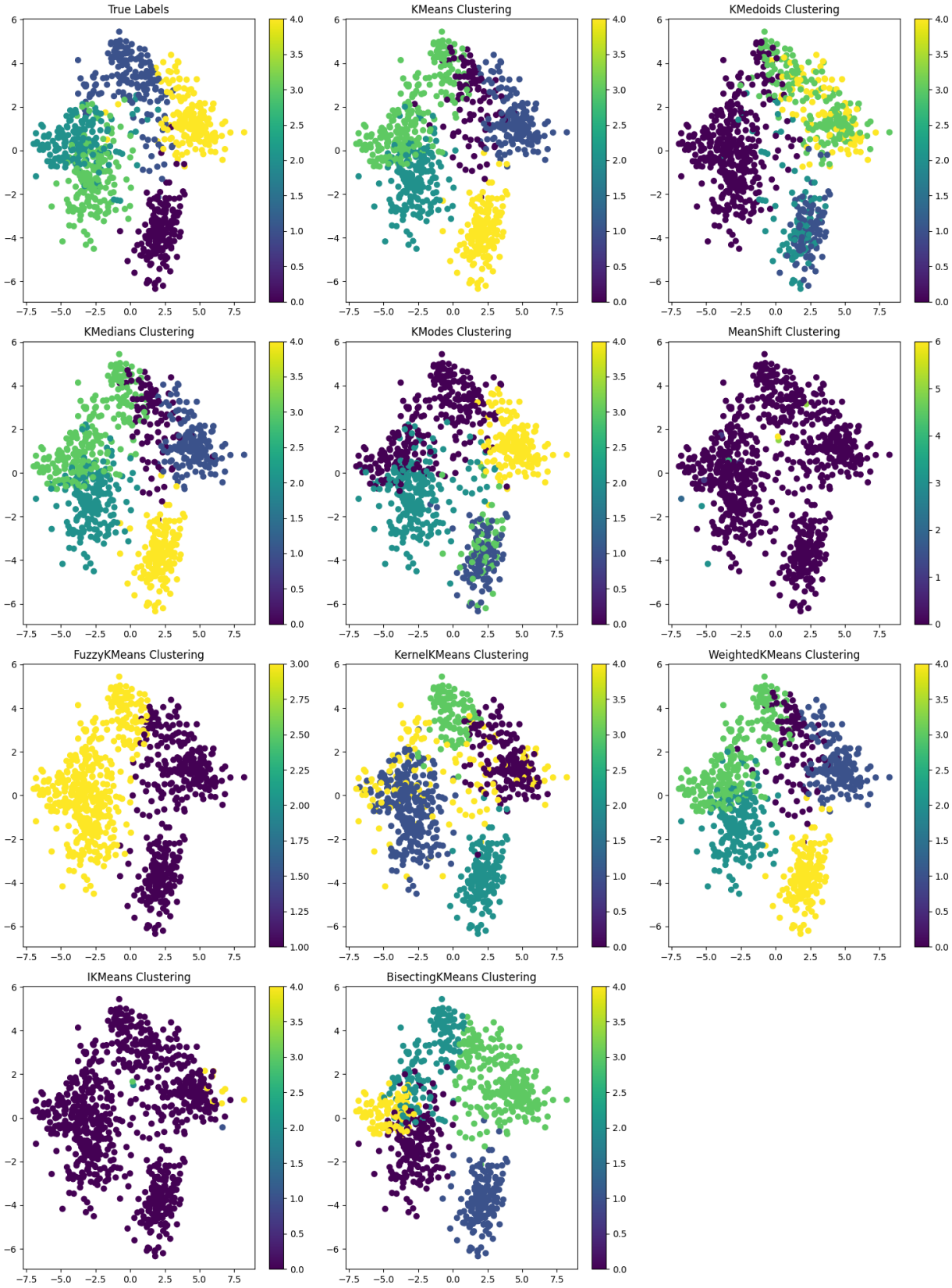

Figure 12: Clustering visualization of Digits Datasets

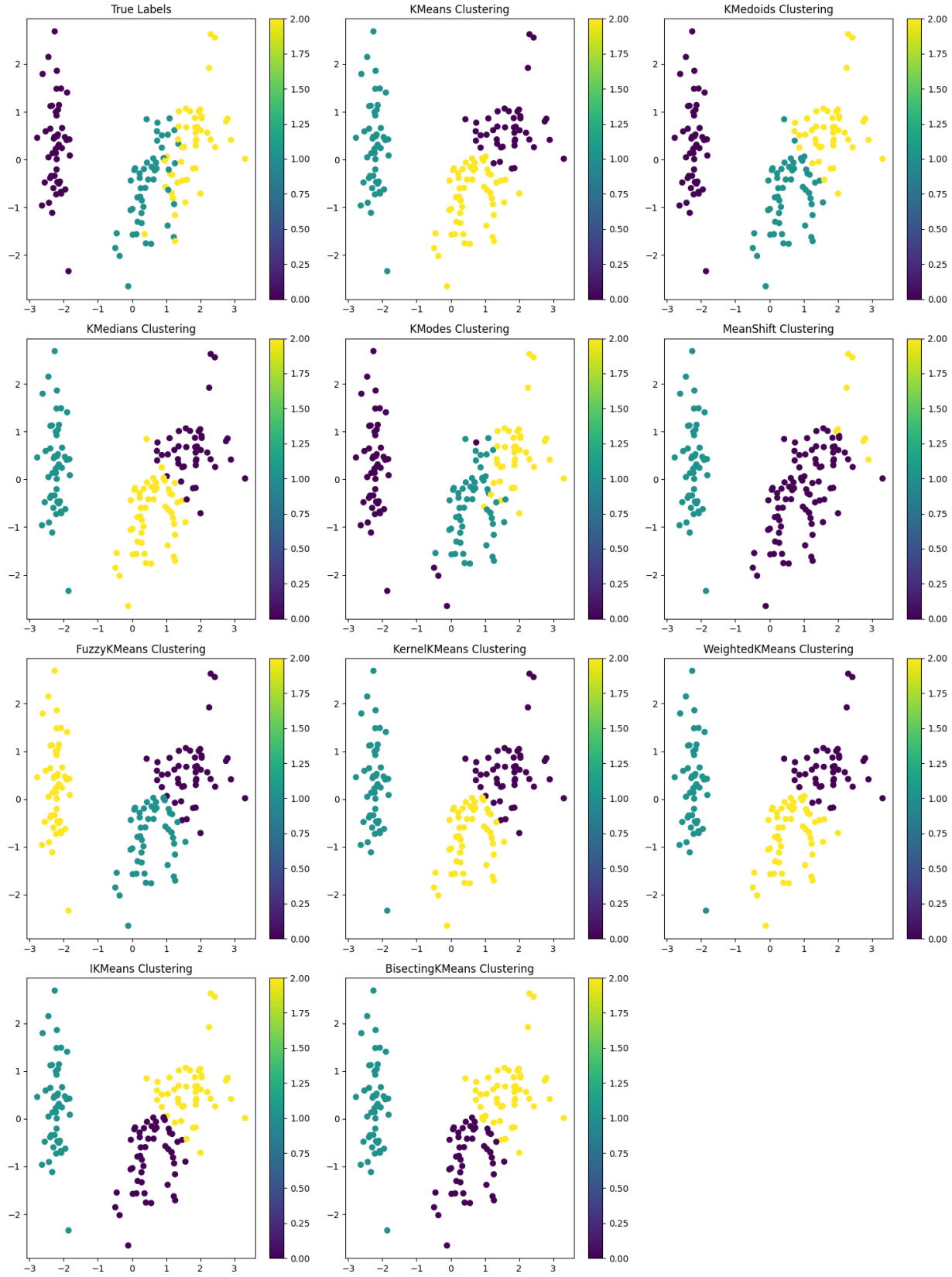

Figure 13: Clustering visualization of Iris Datasets

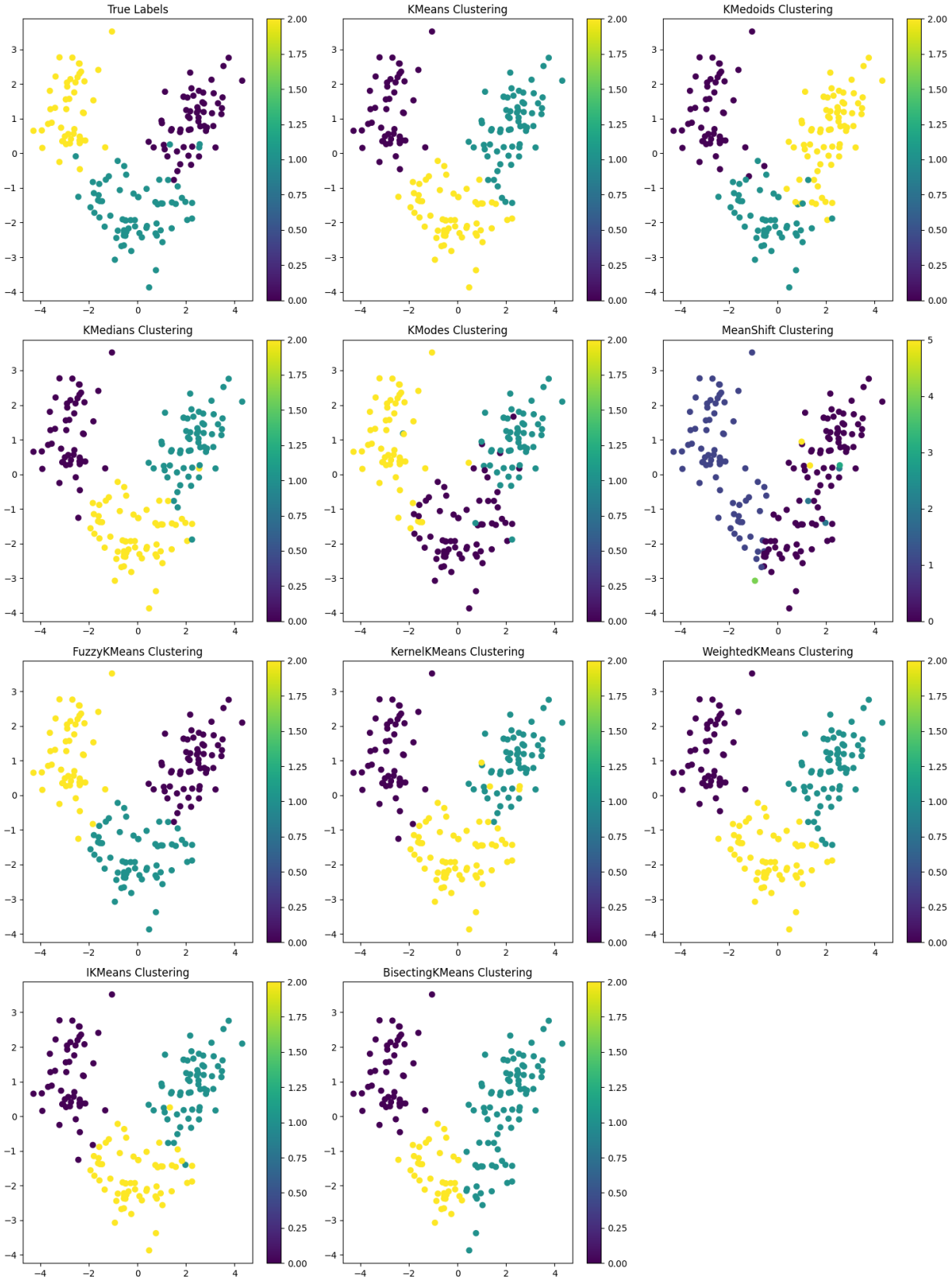

Figure 14: Clustering visualization of Wine Datasets