Clustering

Clustering of data is a method of unsupervised learning that enables the discovery of hidden patterns in data without predefined labels or classes. It organizes data into clusters, where instances within a group are as similar as possible, while different clusters are as distinct as possible. This allows for a better understanding of the data structure, its organization, and dimensionality reduction, which facilitates further analysis.

Clustering methods are often used in user analysis, where companies identify different customer segments based on their purchasing habits. In biomedicine, they enable the discovery of disease subtypes or the classification of genetic data, while in computer vision, they are used for image segmentation and object recognition. They are also key in natural language processing, enabling automatic classification of documents by topic, which is commonly used in search engines and text analysis.

In addition to revealing hidden structures, clustering also helps detect anomalies, as isolated points in the data may indicate fraudulent transactions, measurement errors, or rare events. It is used in geographic analysis to identify dense regions, in network analysis to detect connected communities, and in recommendation systems to tailor content offerings to users. As one of the fundamental tools for analyzing unlabeled data, clustering is indispensable in numerous disciplines.

Attempt at a Formal Definition of the Clustering Objective

Clustering is an unsupervised learning method whose goal is to find an optimal partition of data into k clusters such that the data within each cluster are as similar as possible, while the data across different clusters are as dissimilar as possible. Formally, clustering can be defined as the process of dividing a set of data instances $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ into k disjoint clusters $C_1, C_2, ..., C_k$, where the following holds:

1. Each data instance belongs to exactly one cluster:

$$C_1 \cup C_2 \cup ... \cup C_k = X, \quad C_i \cap C_j = \emptyset, \quad \text{for } i \neq j.$$

2. Within-cluster similarity is maximized:

$$\arg\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i} d(\mathbf{x}, \mu_i),$$

where μ_i is the center of cluster C_i, and $d(\mathbf{x}, \mu_i)$ is a distance function between instance $\mathbf{x}$ and the cluster center (e.g., Euclidean distance).

3. Between-cluster dissimilarity is maximized:

$$\arg\max_{C_1,\ldots,C_k} \sum_{i\neq j} d(\mu_i, \mu_j),$$

where $d(\mu_i, \mu_j)$ measures the distance between the centers of different clusters to ensure they are well-separated.

Practical clustering methods optimize different objectives. For example, the k-means technique minimizes the sum of squared within-cluster distances, hierarchical clustering builds a hierarchy of clusters based on inter-group distances, and DBSCAN identifies dense regions without requiring the number of clusters to be specified in advance.

Types of Clustering

Clustering methods differ in how they form clusters and the criteria they use to group data instances. Different algorithms are suited to different types of data and analytical goals. Broadly, they can be divided into several main categories, as illustrate in the table below.

Clustering Type	Description	Example Algorithms
Partitional	Divides data into a fixed number k of clusters	K-means, K-medoids, Fuzzy C-means
Hierarchical	Builds a tree structure of clusters	Single-linkage, Ward, Complete-linkage
Density-based	Clusters are regions of high data density	DBSCAN, OPTICS, Mean-Shift
Model-based	Clusters are based on statistical models	GMM, Bayesian approaches

Clustering Type	Description	Example Algorithms
Graph-based	Clusters are based on graph theory	Spectral Clustering, Louvain

Different approaches to clustering offer various advantages and disadvantages, depending on the characteristics of the data and the desired outcome of the analysis. Partitional methods are efficient and simple but require a predefined number of clusters. Hierarchical methods allow for multi-level analysis but are computationally intensive. Density-based clustering is suitable for data with non-linear structures, model-based methods allow flexible assignment of instances to multiple clusters, and graph-based clustering is useful for capturing complex relationships between data. Soft clustering is particularly valuable when group boundaries are unclear. The choice of the optimal method depends on the nature of the data and the analytical goals.

Partitional Clustering

Partitional methods divide the data set into a fixed number k of clusters, with each data point belonging to exactly one cluster. The goal is to minimize within-cluster distances and optimize the assignment of data instances to clusters. The most well-known representative is K-means, which iteratively searches for cluster centers and reassigns points. A similar method is K-medoids, which uses actual data instances as cluster centers instead of the mean, making it less sensitive to outliers. For cases where a data point can belong to multiple clusters at once, Fuzzy C-means is used, assigning each point a probabilistic membership to multiple clusters. Partitional methods are computationally efficient and easy to implement, but require the number of clusters to be specified in advance and are sensitive to the choice of initial centers.

Hierarchical Clustering

Hierarchical methods build a dendrogram, a tree structure of clusters, allowing data analysis at different levels of granularity. The process can be agglomerative (bottom-up), where individual points are gradually merged into larger clusters, or divisive (top-down), where the entire set is gradually split into smaller clusters. Different methods for measuring inter-cluster distances, such as single-linkage (minimum distance between points), complete-linkage

(maximum distance between points), and Ward's method (minimizing variance), influence the cluster shapes. A key advantage of hierarchical clustering is that it does not require predefining the number of clusters and allows for visual interpretation of results, but it is computationally intensive and less suitable for large datasets.

Density-Based Clustering

Density-based clustering methods do not require a predetermined number of clusters but instead identify clusters as densely populated regions in the data space, separated by low-density areas. The most well-known representative is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which defines clusters based on the number of points within a given radius and allows for the identification of outliers. Its extension, OPTICS, detects clusters of varying densities. Mean-Shift is another approach that iteratively shifts cluster centers toward dense areas. These algorithms are useful for finding irregularly shaped clusters and detecting anomalies but struggle with data of uneven density and are computationally demanding.

Model-Based Clustering

Model-based methods rely on statistical models and assume that the data are generated from a combination of several hidden distributions, typically Gaussian. Gaussian Mixture Models (GMM) represent clusters as a mixture of Gaussian distributions and use the Expectation-Maximization (EM) algorithm to find optimal parameters. More advanced approaches, such as Bayesian methods, allow determining the number of clusters based on posterior probabilities. Model-based methods support soft clustering, where a single instance can belong to multiple clusters, and are particularly useful when clusters vary in shape and size. Their drawbacks include complex parameter tuning and high computational cost.

Graph-Based Clustering

Graph-based approaches treat data as a graph, where data instances are nodes and the connections between them are based on similarity. Spectral Clustering uses the eigenvalues of the Laplacian matrix to find clusters and is suitable for complex data structures that are not necessarily convex. The Louvain Method is often used to detect communities in large networks, optimizing the modularity of connections. Graph-based methods are especially useful for analyzing social networks, molecular structures, and geometrically complex data

spaces, but can be computationally expensive for large datasets.

Some Selected Approaches

Among the approaches above, let us take a closer look at a few selected ones that are perhaps the most commonly used.

Hierarchical Clustering

Hierarchical clustering arranges data into a tree structure of clusters (*dendrogram*), where each data point is initially treated as a separate cluster, and then gradually merged into larger groups until only a single cluster remains or a certain level of merging is reached. There are two approaches: agglomerative clustering (bottom-up), which starts with individual points and gradually merges them into larger clusters, and divisive clustering (top-down), which starts with a single cluster and splits it into smaller subgroups. Agglomerative clustering is more commonly used, possibly due to the simplicity of the algorithm (see below). The advantage of hierarchical clustering is that it allows analysis at different levels of granularity, while its drawbacks include high computational or memory complexity ($O(n^2)$ or more) and sensitivity to noise.

Agglomerative clustering can be carried out using the following algorithm:

1. Treat each data point as a separate cluster.
2. Compute the distance matrix between all pairs of clusters.
3. Merge the two most similar clusters into a new, larger cluster.
4. Update the distance matrix (according to the chosen similarity measure).
5. Repeat steps 3 and 4 until only one cluster remains or the desired number of clusters is reached.

Hierarchical clustering uses different methods to measure distances between clusters. The choice of linkage method affects the structure and interpretation of clustering results. Let C_i and C_j be clusters, and $d(\mathbf{x}_a, \mathbf{x}_b)$ the distance between points $\mathbf{x}_a \in C_i$ and $\mathbf{x}_b \in C_j$. Mathematically, the linkage methods are defined as follows:

1. **Single linkage**. The distance between clusters is the minimum distance between any two points from different clusters. This method can detect irregularly shaped clusters but is sensitive to the chaining effect.

$$d(C_i, C_j) = \min_{\mathbf{x}_a \in C_i, \mathbf{x}_b \in C_j} d(\mathbf{x}_a, \mathbf{x}_b)$$

2. **Complete linkage**. The distance between clusters is the maximum distance between any two points. It forms more compact clusters but is less robust to outliers.

$$d(C_i, C_j) = \max_{\mathbf{x}_a \in C_i, \mathbf{x}_b \in C_j} d(\mathbf{x}_a, \mathbf{x}_b)$$

   - Merges clusters based on the greatest distance between any two points.
   - Produces compact, spherical clusters, but is sensitive to outliers.
3. **Average linkage**. The average distance between all pairs of points from different clusters. It provides balanced merging and is less sensitive to outliers.

$$d(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_j} d(\mathbf{x}_a, \mathbf{x}_b)$$

   - Merges clusters based on the average distance between all pairs of points.
   - A balanced method that yields more stable results.
4. **Ward's method (minimum increase in variance)**. Minimizes the increase in variance when merging clusters, resulting in more evenly sized clusters. Commonly used when clusters are roughly spherical.

$$d(C_i, C_j) = \sum_{\mathbf{x} \in C_{ij}} \|\mathbf{x} - \mu_{ij}\|^2 - \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mu_i\|^2 - \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mu_j\|^2$$

   - Merges clusters by minimizing the increase in within-cluster variance.
   - Suitable for data where clusters are approximately spherical.

Each method has its own strengths and weaknesses, and the choice depends on the data structure and the application. Typically, for real-world data, such as business data, we would use either Euclidean or cosine distance in combination with Ward's method.

The main advantage of hierarchical clustering is the graphical representation of results via the dendrogram, which can be effectively combined with other data visualizations, such as heatmaps. However, there are several drawbacks: dendrograms are suitable only for presenting small datasets (typically up to a few hundred instances), their representation is not unique (the same clustering can yield exponentially many dendrograms), and the decision on how many clusters are present in the data is left to the user.

The K-means Method

Clustering using the K-means method is one of the most commonly used partitional clustering techniques. The algorithm partitions a dataset into k clusters by minimizing the sum of squared distances between data points and their respective cluster centers (*centroids*). Each data point is assigned to the nearest center, and then the centers are updated as the mean of all points in each cluster. This process is repeated until the centers converge or the assignments stop changing. Due to its simplicity and efficiency, the method is widely used across various domains, such as user analysis, image classification, and biomedical data processing.

Mathematically, the goal of the method is to minimize the within-cluster error given by:

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

where C_i is the set of data points in the i-th cluster, and μ_i is the cluster center, computed as the mean of all points in the cluster. The algorithm operates in two key steps: (1) Assignment – each point is assigned to the nearest cluster center based on Euclidean distance, and (2) Update – the centers are moved to the mean of the points in their cluster. These two steps are repeated until the centers stabilize.

Although K-means is efficient and converges quickly, it has some drawbacks. The number of clusters k must be specified in advance, and the algorithm is sensitive to the initial choice of centers, which can lead to local minima. It also performs poorly on data with irregularly shaped clusters and is sensitive to outliers. To improve stability, multiple initializations (K-means++) are often used to select better starting centers, or alternative algorithms like K-medoids, which use actual data points as cluster centers. The method is particularly sensitive to outliers, as they can significantly affect the cluster centers, so in noisy data, clustering with medoids or density-based methods like DBSCAN is often more appropriate.

K-means++ is an improved initialization method for K-means that reduces the risk of poor convergence to local minima caused by poorly chosen initial centers. Classical K-means selects initial centers randomly, which can result in unbalanced clusters or slow convergence. K-means++ improves this by initializing centers based on data spread, allowing for faster and more stable clustering. Its main advantage is that it provides a better distribution of initial centers, increasing accuracy and reducing the number of iterations needed for convergence.

K-means++ initialization procedure:

1. Randomly select the first center μ_1 from the data points.
2. For each point $\mathbf{x}$, compute its distance to the nearest already selected center $D(\mathbf{x}) = \min_i d(\mathbf{x}, \mu_i)$.
3. Select a new center μ_j randomly, where the probability of selecting point $\mathbf{x}$ is proportional to $D(\mathbf{x})^2$ (points farther away are more likely to be selected).
4. Repeat until all k centers are selected.
5. Once the initial centers are chosen, continue with the standard K-means algorithm (assignment and update steps).

This method ensures that the initial centers are better spread across the data space, increasing accuracy and reducing the algorithm's sensitivity to initial selection. Experiments have shown that K-means++ generally requires fewer iterations and yields solutions closer to the global optimum compared to the classical K-means algorithm.

The Medoid Method

The medoid clustering method (K-medoids) is a variant of the K-means method, where the cluster centers (*medoids*) are actual data points instead of means. The algorithm iteratively updates the medoids to minimize the sum of distances between data points and their corresponding medoids, making it less sensitive to outliers compared to K-means. In addition, it allows the use of arbitrary metric distances, not just Euclidean, and is more suitable for non-spherical clusters, as it does not assume uniform cluster size or shape. Its main drawback is higher computational complexity ($O(n^2)$) compared to K-means ($O(nk)$), which limits its scalability for very large datasets.

The DBSCAN Method

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering method that does not require the number of clusters to be specified in advance. Instead, it identifies clusters as regions with high density of data points, separated by areas of low density. This enables the detection of clusters with irregular shapes, which is an advantage over methods like K-means that assume spherical clusters. In addition, DBSCAN can identify outliers (points that do not belong to any cluster), making it useful for anomaly detection, pattern discovery in geolocation data, and processing large unstructured datasets.

The DBSCAN algorithm is based on a density criterion, requiring that each cluster contains at least minPts points within a radius ε (*epsilon neighborhood*). The algorithm begins by randomly selecting a data point and checking how many points lie within its ε radius. If there are at least minPts, the point is marked as a core point and cluster expansion begins—all points within its reach are added to the same cluster, and the process is recursively repeated for new core points. Points with fewer than minPts neighbors may be border points (belonging to a cluster but not core points) or outliers (not belonging to any cluster). The algorithm continues until all points are either assigned to clusters or marked as outliers.

DBSCAN has several advantages: it does not require the number of clusters to be specified in advance, it detects irregularly shaped clusters, it identifies outliers, and it performs well on large datasets. Its drawbacks include sensitivity to the parameters ε and minPts, which need to be set appropriately for different datasets. DBSCAN also struggles with datasets of varying density, as sparse clusters may be missed or several dense regions may be merged into a single cluster.

Optimal values for the parameters ε (epsilon) and minPts in DBSCAN are determined empirically or using analytical methods, as they strongly affect the resulting cluster structure. The parameter minPts, which defines the minimum number of points in the neighborhood, should be at least $d + 1$, where d is the data dimensionality, to ensure that a cluster is not defined by a single point. In practice, values between 4 and 10 are often used, with higher values helping reduce the impact of noise and outliers. Too small a value can lead to excessive fragmentation into small clusters, while too large a value may merge distinct clusters or miss smaller ones altogether.

Choosing the optimal value of ε is more challenging, as it depends on the data distribution. One of the most common methods is the k-distance graph analysis, where for each point the distance to its k-th nearest neighbor is computed (usually $k = \mathrm{minPts}$), the distances are sorted, and a plot is drawn. The *elbow point*, where the distances begin to rise sharply, indicates a good choice for ε. Alternatively, different values of ε can be tested experimentally, observing how clusters form. A value that is too small results in many small clusters and many outliers, while a value that is too large may merge distinct clusters into one.

An additional way to evaluate clustering quality is by using metric scores such as the Silhouette Coefficient or the Davies-Bouldin Index, which enable quantitative comparison of results under different parameter settings. Proper selection of ε and minPts is particularly important for data with varying density, where different parts of the data space may require

different ε values—this is a limitation of the DBSCAN method. In such cases, an extension like OPTICS can be used, which allows the density criterion to adapt within the dataset.

The Louvain Method

The Louvain method is one of the most established techniques for clustering in networks, where nodes are grouped into communities based on modularity optimization—a measure that assesses the quality of the network's division into communities. The algorithm is fast and efficient, using hierarchical clustering, which enables it to process large networks with millions of nodes. It was developed in 2008 at the Université catholique de Louvain in Belgium. The Louvain method is used across various fields, such as social network analysis, biology (gene and protein networks), economics, and transportation system optimization.

The Louvain method operates in two main phases, which are repeated iteratively until the network structure stabilizes. In the first phase, each node initially belongs to its own community. Then, each node is individually moved to the community of one of its neighbors if the move increases the network's modularity—meaning that the connections within the community become denser than expected under a random distribution of edges. This process continues until no further improvement in modularity is possible. In the second phase, the current communities are treated as new nodes, and the graph is compressed so that the connections between the original communities become edges between the new, aggregated nodes. The algorithm is then rerun on this reduced version of the network and continues iteratively until modularity reaches a maximum or no more changes occur.

Network modularity is a measure that evaluates the quality of a network's division into communities. It reflects the difference between the actual number of edges within communities and the expected number of such edges if the connections were randomly distributed across the network. Mathematically, modularity is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where A_{ij} is the adjacency matrix, k_i and k_j are the degrees of nodes i and j, m is the total number of edges, c_i is the community label of node i, and $\delta(c_i, c_j)$ is an indicator function that is 1 if both nodes are in the same community, and 0 otherwise. Modularity measures how much better the clustering is than a random partitioning and typically ranges from 0 to 1, with higher values indicating more well-defined communities.

The main advantages of the Louvain method are its computational efficiency and its ability to automatically determine the number of communities without requiring preset parameters. Because it builds a hierarchical structure, it enables multi-level community analysis, which is useful in complex networks. A drawback of the method is that the results are not always stable, as small changes in the data can affect the final community structure. Additionally, the method is optimized for modularity, which is not always the best measure for community detection in some types of networks, such as very sparsely connected graphs or networks with dynamic changes.

Gaussian Mixture Models

Gaussian Mixture Models (GMM) represent a probabilistic approach to clustering, where the data are assumed to be generated from multiple Gaussian distributions, with each distribution corresponding to one cluster. Instead of assigning each data point to exactly one cluster, GMM estimates the probability of belonging to each cluster, enabling *soft clustering*.

Mathematically, the dataset is modeled as a combination of k normal distributions:

$$p(\mathbf{x}) = \sum_{i=1}^{k} \pi_i \cdot \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i)$$

where π_i is the weight of the i-th component, representing the prior probability of a data point belonging to cluster i (with $\sum \pi_i = 1$), and $\mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i)$ is the Gaussian distribution with mean μ_i and covariance matrix Σ_i.

The GMM algorithm is based on the Expectation-Maximization (EM) procedure:

1. **Expectation step (E-step):** For each data point, compute the probability of belonging to each cluster based on the current parameter estimates.
2. **Maximization step (M-step):** Update the parameters (means μ_i, covariance matrices Σ_i , and weights π_i) to maximize the likelihood of the observed data.
3. Repeat until the parameters converge.

GMM is more flexible than the K-means algorithm, as it allows clusters of different shapes and sizes by using arbitrary covariance matrices. It also supports soft assignment of instances to clusters, which is useful when cluster boundaries are unclear. The main drawback of the method is that it requires the number of components k to be specified,

which can be determined using criteria such as the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC). Additionally, GMM is more computationally intensive than K-means, due to its probabilistic calculations and handling of full covariance matrices.

Evaluating Clustering Quality

Clustering methods are unsupervised, meaning that in practice we usually do not have known ground truth labels for the data. Therefore, it is crucial to assess the quality of the resulting clusters and verify whether the identified groups are meaningful and consistent with the data structure. Without appropriate metrics, algorithms may return clusters that are computationally optimal but lack real-world relevance. A good clustering evaluation allows comparison of different methods, selection of the optimal number of clusters, and identification of potential model weaknesses, such as overfragmented or poorly merged clusters.

Different approaches are used for clustering evaluation. **Internal metrics** assess clusters based solely on data characteristics, measuring within-cluster density and between-cluster separation. Typical internal metrics include the silhouette coefficient, Dunn index, and the ratio of within-cluster to between-cluster variance (Davies-Bouldin index). **External metrics**, such as the Rand index and F-measure (micro-averaged), require comparison with known labels and are used when a reference classification is available. These metrics essentially measure the agreement between two clusterings.

There are also **stability metrics**, which assess whether the obtained clusters remain consistent across different data subsets.

Silhouette Method

Among internal metrics, perhaps surprisingly due to its simplicity, the silhouette method is one of the most commonly used and intuitive approaches for evaluating clustering quality without requiring external labels. The silhouette coefficient measures how well a point fits within its cluster compared to other clusters. For each data point i, we first compute the average distance to all other points in the same cluster ($a(i)$, internal cohesion). Then, we compute the average distance to all points in the nearest neighboring cluster ($b(i)$, external separation). The silhouette coefficient for point i is then defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

The value of $s(i)$ ranges from -1 to 1, where values close to 1 indicate that the point is well separated from other clusters and clearly belongs to its own cluster; values near 0 indicate that the point lies between clusters; and values below 0 suggest that the point has been misclassified. The average silhouette score across the entire dataset is a good indicator of clustering quality—higher values indicate better clustering. This method is frequently used to determine the optimal number of clusters, by comparing the average silhouette score for different values of k and selecting the one with the most distinct clustering. The silhouette method is especially useful for evaluating methods like K-means and Gaussian Mixture Models but is less suitable for methods not based on distances, such as some graph-based clustering techniques.

Rand Index

The Rand Index (RI) and the Adjusted Rand Index (ARI) are external metrics for evaluating clustering quality, meaning they are used when reference labels (true groupings) are available to compare with the obtained clusters. Both measure how well the clustering matches the actual data partitioning.

The Rand Index is based on the number of pairs of data points that are either correctly assigned to the same cluster or correctly assigned to different clusters. If we have n data points, there are $\binom{n}{2}$ possible pairs that can be compared. The Rand Index is calculated as:

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP (True Positives): Pairs of points that are in the same cluster in both the true and predicted clusterings.
- TN (True Negatives): Pairs of points that are in different clusters in both the true and predicted clusterings.
- FP (False Positives): Pairs of points that are in the same cluster in the predicted clustering but in different clusters in the true clustering.
- FN (False Negatives): Pairs of points that are in different clusters in the predicted clustering but in the same cluster in the true clustering.

The Rand Index ranges from 0 to 1, where 1 indicates perfect agreement between the clustering and the ground truth, and 0 indicates complete disagreement.

A known limitation of the Rand Index is that it can return high values for random clusterings due to coincidental agreement. To correct for this, the **Adjusted Rand Index (ARI)** is used, which normalizes the Rand Index by accounting for the expected value of RI under random labeling. ARI is defined as:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

where $E[RI]$ is the expected value of the Rand Index under a random clustering. The Adjusted Rand Index ranges from -1 to 1, where:

- 1 indicates perfect agreement,
- 0 indicates random labeling,
- negative values indicate clustering worse than random.

To compute ARI, the expected value $E[RI]$ is crucial for normalization, accounting for what the RI would be if clustering were random. Let:

- n be the number of data points,
- n_{ij} the number of shared points between cluster i in the ground truth and cluster j in the predicted clustering,
- a_i the number of points in cluster i in the ground truth,
- b_j the number of points in cluster j in the predicted clustering.

The expected value of the Rand Index is computed using combinatorics on point pairs, considering how true positives (TP) and true negatives (TN) are distributed under random clustering:

$$E[RI] = \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}$$

where:

- $\binom{a_i}{2} = \frac{a_i(a_i-1)}{2}$ is the number of pairs within cluster i in the true clustering,
- $\binom{b_j}{2} = \frac{b_j(b_j-1)}{2}$ is the number of pairs within cluster j in the predicted clustering,
- $\binom{n}{2} = \frac{n(n-1)}{2}$ is the total number of possible pairs in the dataset.

With this normalization, the Adjusted Rand Index is defined as:

$$ARI = \frac{\sum_i \sum_j \binom{n_{ij}}{2} - E[RI]}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - E[RI]}$$

This adjustment eliminates the Rand Index's bias and provides a fairer evaluation of clustering quality, yielding a value of 0 for random clustering and 1 for perfect agreement with the reference partitioning.

The key difference between RI and ARI is that RI is not normalized and may produce high values for random results, while ARI corrects this bias by considering the expected distribution of random outcomes. For this reason, ARI is more commonly used in practice, as it provides a more accurate picture of clustering quality.

Interpreting Clustering Results

Clustering data enables the discovery of hidden patterns, but without proper interpretation of the resulting groups, their usefulness is limited. Understanding what each cluster represents is essential for decision-making, strategy development, and further data analysis. For example, in medicine, clusters may correspond to different disease subtypes; in marketing, to customer segments; and in genomics, to functionally related genes. Without interpretation, there is a high risk of misusing results, as clusters may be formed based on irrelevant features or statistical artifacts. Therefore, it is important to apply methods that provide insight into the characteristics and structure of the resulting groups. Below, we outline some common approaches.

Mean Values and Representative Examples. One of the simplest methods for interpreting clusters is reviewing the average feature values within each cluster, which helps quickly understand key differences between groups. This technique is most effective when combined with feature ranking methods that highlight features most distinct across clusters. Alternatively, one can identify the most representative data points in a cluster, such as medoids in K-medoids or data points closest to the cluster centroid.

Data Visualization. For high-dimensional data, it is useful to reduce dimensionality and visualize the clusters. Major techniques include Principal Component Analysis (PCA), which projects data into 2D or 3D space; t-SNE (t-distributed Stochastic Neighbor Embedding),

which emphasizes local data patterns; and UMAP (Uniform Manifold Approximation and Projection), which is similar to t-SNE but better preserves global data structure. These methods will be discussed in detail in the next chapter.

Identifying Features Important for Clustering. Each feature's contribution to cluster formation can be assessed. This can be done using statistical tests (e.g., ANOVA for numerical data, chi-squared tests for categorical data) or feature attribution techniques such as SHAP values, which estimate the impact of individual variables on clustering.

Semantic Interpretation of Clusters. In certain domains, such as natural language processing and biomedicine, clusters can be interpreted using ontologies and semantic models. For instance, clusters in text data can be analyzed using TF-IDF to extract keywords, while in biological data, Gene Ontology can be used to link gene clusters to functional categories.

Rules and Decision Trees for Cluster Explanation. To facilitate the explanation of clusters, one can construct decision trees or rules based on features that best distinguish between clusters. This approach generates simple rules such as "*If age > 50 and blood pressure > 140, then the probability of belonging to cluster X is 85%*". Such rules help in understanding complex results and make them applicable in real-world scenarios.

Interpreting clustering results is essential for meaningful application, as without it, the resulting groups remain mathematical entities with no clear significance. By employing various methods for visualization, feature analysis, semantic interpretation, and rule generation, we can gain deeper insights into the data structure and enhance the utility of clustering models across diverse applications.