Department of Computer Engineering

Bu-Ali Sina University

Machine Learning Course

Clustering Analysis

Machine Learning 2nd Assignment

By:

Alireza Maleki

Course Professor:

Professor Muharram Mansoorizadeh

Winter 2023-24

TABLE OF CONTENTS

1 INTRODUCTION

Clustering analysis is an important unsupervised learning technique for discovering patterns and structure within data. This report evaluates and compares three major clustering algorithms - K-Means, DBSCAN, and HDBSCAN - on both synthetic and real-world datasets. The algorithms are assessed based on their ability to correctly reconstruct embedded clusters in the data as well as runtime performance.

The report first provides an overview of each algorithm, describing its methodology and key parameters that need to be set. K-Means is a simple, efficient partitional clustering method that requires the number of clusters to be pre-specified. DBSCAN and HDBSCAN are more advanced density-based techniques that can identify arbitrary cluster shapes and sizes without fixing the cluster count a priori. However, they require more care in setting other tuning parameters.

Both synthetic benchmark data as well as a real-world California housing dataset are utilized to test the clustering approaches. The artificial data provides a controlled setting where the true clusters are known, enabling quantitative evaluation of accuracy. The housing dataset demonstrates applicability to real-world problems with non-uniform and nonlinear relationships. Different validity metrics are introduced, including Silhouette score, Calinski-Harabasz index, and completeness, which assesscluster cohesion, separation, and correspondence to ground truth labels.

The experiments analyze the ability of K-Means, DBSCAN, and HDBSCAN to reconstruct the embedded clusters across the two datasets. Additionally, their parameter sensitivity, computational efficiency, and inherent biases are investigated. The report provides guidance on selecting appropriate clustering algorithms based on dataset characteristics and demonstrates the importance of validation metrics for optimizing performance.

2 METHODOLOGY

DBSCAN HDBSCAN Algorithm overview Selecting parameters

2.1 K-Means Clustering Algorithm

2.1.1 Algorithm overview

K-Means is one of the simplest and most popular unsupervised machine learning algorithms for clustering. The objective of K-Means is to group data points into k clusters, where each data point belongs to the cluster with the nearest mean. The algorithm works as follows:

1. Randomly initialize k cluster centers or means (centroids).
2. Assign each data point to the nearest cluster center based on the Euclidean distance between the data point and the cluster center. This forms k clusters.
3. Calculate the new mean for each cluster.
4. Repeat steps 2 and 3 until the cluster assignments stop changing or the maximum number of iterations is reached.
5. The algorithm tries to minimize the sum of squared distances between each data point and its assigned cluster center. The final clusters capture the inherent structure of the data.

K-Means is popular due to its simplicity and efficiency in clustering large datasets. However, it assumes spherical and well-separated clusters which is not always the case in real-world data. The algorithm is also sensitive to initialization and may converge to local optima based on the starting centroids.

2.1.2 Selecting parameters

The key parameters when applying K-Means clustering are:

- **k - Number of clusters:** This is usually pre-determined based on domain knowledge or the elbow method. Choosing k is critical as it impacts the distribution of points across clusters. Since we visualized the dataset, we decided to set the number of clusters to 5 for the fake dataset, and 5 for the California house price either. All these decisions were taken based on the elbow method graph.
- **Number of iterations:** The algorithm is run for a maximum number of iterations for convergence. More iterations may improve accuracy but can increase computation time. we set this parameter to 100 for both datasets.
- **Initialization method:** Different methods like random initialization or $k-means++$ improve clustering accuracy by avoiding poor starting centroids.
- **Seeds:** Running the algorithm multiple times with different random seeds can avoid bad initializations. The best clustering is chosen out of the runs. Setting this parameter remains as default.

Choosing the right parameters is crucial for K-Means to discover meaningful clusters in the data. The number of clusters k has the greatest impact on results. Parameters like distance metric and initialization also significantly affect the formation of clusters.

Table 1: K-Means parameters for both datasets

K-Means	n_clusters	init	max_iter	n_init
Fake dataset	5	k-means++	100	10
California dataset	5	k-means++	100	10

2.2 DBSCAN Clustering Algorithm

2.2.1 Algorithm overview

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups closely packed points into clusters and marks outliers as noise. The key idea is that clusters are dense regions separated by low-density regions.

The algorithm requires two parameters - eps and $min\_samples$. It starts with an arbitrary starting point that has not been visited. The point's eps-neighborhood is retrieved, and if it contains sufficient points ($>= min\_samples$), a cluster is started. Otherwise, the point is marked as noise.

This process continues recursively for all points in the eps-neighborhood. Eps defines the radius of the neighborhood, while min_samples defines the minimum number of points required to form a dense cluster. As DBSCAN visits each point just once, it is efficient for large datasets.

DBSCAN can discover clusters of arbitrary shapes and sizes. It does not require knowing the number of clusters a prior, unlike K-Means. However, it struggles with clusters of varying density and is sensitive to the choice of eps and min_samples. Overall, it performs well when clusters are sufficiently dense.

2.2.2 Selecting parameters

The DBSCAN algorithm depends heavily on the two parameters eps and min_samples.

Eps controls the local neighborhood radius around each point. A small eps leads to many small clusters, while large eps values result in fewer large clusters. Eps can be chosen based on domain knowledge or optimized using techniques like the k-dist graph.

Min_samples sets the minimum number of samples in an eps-neighborhood for clustering. A low min_samples value is prone to noise, while a high value may miss smaller clusters. Typical values range from 5-10 for small datasets and 10-15 for large datasets.

DBSCAN is also affected by the distance metric used to measure the distance between points. Common options are Euclidean, Manhattan, or Cosine distance. The optimal metric depends on the data characteristics.

Table 2: DBSCAN parameters for both datasets.

DBSCAN	eps	min_samples
Fake dataset	0.5	10
California dataset	0.29	15

Overall, eps primarily controls cluster sizes, while min_samples helps reduce noise. Tuning these parameters appropriately is important for DBSCAN's success. Domain knowledge should be used to narrow suitable parameter ranges.

2.3 HDBSCAN Clustering Algorithm

2.3.1 Algorithm overview

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) builds on the original DBSCAN algorithm to address some of its limitations. It converts the estimated density cluster tree into a condensed tree to simplify clustering.

The algorithm first transforms the data into a hierarchical clustering tree based on density reachability. Branches with high density become clusters while branches with low density turn into noise/outliers.

The condensed tree collapses subclusters and child clusters into a single cluster. This simplifies the variable density clusters found in real-world data. The condensed tree allows automated extraction of flat clusters based on the stability of clusters - how persistent clusters are across different eps values.

HDBSCAN automatically selects suitable eps values based on the clustered tree structure. This removes the difficulty in manually selecting eps and $min\_samples$ as in DBSCAN. The final clusters capture points strongly connected by density reachability.

2.3.2 Selecting parameters

A key advantage of HDBSCAN is the automatic selection of eps and $min\_samples$ based on the condensed cluster tree. This simplifies the tuning process significantly. The $min\_cluster\_size$ parameter dictates the minimum size of clusters. Smaller clusters are marked as noise. This parameter replaces $min\_samples$ in DBSCAN and is easier to select based on domain knowledge. The $min\_samples$ parameter in HDBSCAN refers to the number of samples in a leaf node of the tree. It

Table 3: HDBSCAN parameters for both datasets.

HDBSCAN	min_cluster_size	min_samples
Fake dataset	50	10
California dataset	400	150

affects the granularity of the clustered tree generation and is typically set to 1.

The distance metric works similarly to DBSCAN and influences density estimation. HDBSCAN runs faster for condensed trees than DBSCAN but has higher memory requirements due to the tree data structure. Overall, HDBSCAN simplifies the parameter selection through its hierarchical tree analysis and condensed tree representation. $min\_cluster\_size$ is the most important tuning parameter in practice.

2.4 Datasets

2.4.1 Fake Dataset

Synthetic datasets provide a valuable and convenient means of evaluating the performance of machine learning algorithms in a controlled setting. The $make\_blobs()$ function in the scikit-learn Python library enables the straightforward creation of multivariate Gaussian blob datasets with a variety of customizable parameters. By specifying the desired number of samples, centers, cluster standard deviations, and bounds, one can produce artificial datasets exhibiting different types of clustering characteristics. $make\_blobs()$ allows controlling the randomness via an initial seed state for reproducible results across trials. In our analysis, we utilized a simulated dataset containing 5 clusters with 1000 total data points, where 2 features characterized each data point. Using such simulated data with known embedded patterns is recommended for the initial validation of clustering algorithms before application to real-world datasets where ground truth labels are not available. This provides a principled framework for assessing algorithmic stability and accuracy by comparing the model's reconstructed clusters to the simulated data's underlying blob structure. Figure**??** represents the data distribution for the simulated dataset.

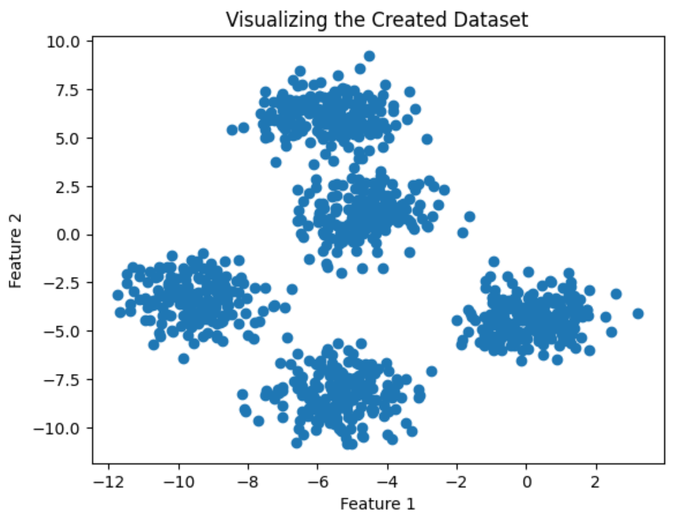

Figure 1: Data distribution on the fake dataset.

2.4.2 Real-World Dataset

The California Housing Prices dataset from the StatLib repository contains information on 20,640 census blocks in California from the 1990 US census. The goal is to predict the median house value for each block based on 8 features: median income, median house age, total rooms, total bedrooms, population, households, latitude, and longitude. House value distribution is right-skewed with values ranging from \$15,000 to \$500,000, indicating multiple outlying high-value areas. Median income varies from \$0 to over \$15,000 per month, while the housing age spans over 50 years. Geographical location features longitude and latitude which may correlate with population density. Overall the mix of census, housing, and location attributes make this a challenging dataset for predictive modeling and provide insight into the nuances of California real estate. Preprocessing steps like log-transforming skewed variables are recommended to improve model accuracy. The distribution of the data in this dataset based on the longitude and latitude is presented in Figure **??**.

2.5 Evaluation Criteria

$$\text{Silhouette score: } s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)} \tag{1}$$

The silhouette score is a metric used to evaluate the goodness of a clustering algorithm. It quantifies how well each data point lies within its cluster compared to other clusters.
Here $a(i)$ is the average distance between point i and all other points in its own cluster, while $b(i)$ is the minimum average distance from point i to points in any other cluster. The score ranges from -1 to 1. A value near 1 indicates the point is appropriately clustered. A negative value means it is

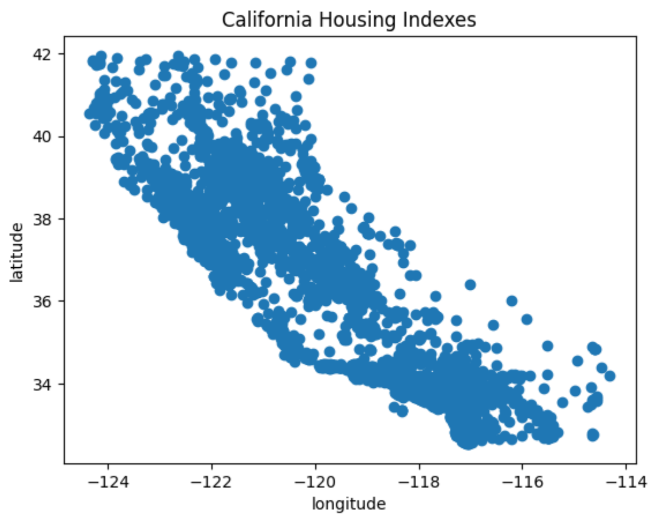

Figure 2: Data distribution on the California house price dataset.

assigned to the wrong cluster. The overall silhouette score for the clustering is the average of $s(i)$ over all data points. A high silhouette score implies good clustering.

The adjusted Rand index is calculated based on the following formula:

$$\text{Adjusted Rand score: RI} = \frac{\text{TP + TN}}{\text{TP + FP + FN + TN}} \tag{2}$$

Where:

1. **TP (True Positive):** Number of pairs of data points that are in the same cluster in both clustering
2. **TN (True Negative):** Number of pairs of data points in different clusters in both clusterings
3. **FP (False Positive):** Number of pairs in same cluster in 1st, but different clusters in 2nd clustering
4. **FN (False Negative):** Number of pairs in different clusters in 1st, but same cluster in 2nd clustering

The Rand index (RI) measures the percentage of decisions that are common between the two clusterings. To account for chance, the expected similarity E is calculated using the individual assignments in each clustering.

The adjusted Rand index is then calculated as: $AdjustedRandIndex = (RI - E)/(1 - E)$ This normalization ensures the adjusted Rand index lies between -1 and 1, with 1 indicating identical clustering assignments and 0 indicating random clustering.

$$\text{Calinski-Harabasz score: CH} = \frac{\text{BGSS}}{\text{WGSS}} \tag{3}$$

The Calinski-Harabasz score also called the variance ratio criterion, is a metric used to evaluate clustering models. It measures the ratio of between-cluster dispersion to within-cluster dispersion.

1. **B = Between-cluster dispersion**
2. **W = Within-cluster dispersion**
3. **k = Number of clusters**
4. **n = Number of data points**

The between-cluster dispersion B is calculated as the sum of squared distances between cluster centers and the global mean. The within-cluster dispersion W is calculated as the sum of squared distances between each point and its cluster center.

A higher CH score indicates clusters are dense and well-separated. The score is higher for models with more distinct clusters. It can be used to optimize the number of clusters k by selecting k that maximizes CH score.

The V-measure is a clustering evaluation metric that measures the homogeneity and completeness of clusters. It is based on conditional entropy and is computed as the harmonic mean of homogeneity and completeness.

$$\text{V-measure score: VM} = 2 \cdot \frac{\text{Homogeneity} \cdot \text{Completeness}}{\text{Homogeneity} + \text{Completeness}} \tag{4}$$

Where:

Homogeneity = 1 - Conditional entropy of the cluster distribution given the class distribution

Completeness = 1 - Conditional entropy of the class distribution given the cluster distribution

Homogeneity measures how well clusters contain a single class. Completeness measures how well classes are assigned to a single cluster.

The V-measure ranges from 0 to 1, with 1 indicating perfectly homogeneous and complete clustering. A higher V-measure indicates a model that better captures the ground-truth classes within the clusters.

It balances both objectives of homogeneity and completeness in evaluating cluster quality. The harmonic mean reduces the impact of extremes. V-measure is useful when the number of clusters is unknown beforehand.

$$\text{Completeness score: CP} = \frac{1}{N} \sum_k \max_j |w_k \cap c_j| \tag{5}$$

Completeness ranges from 0 to 1. A high completeness score indicates each cluster contains data points primarily from a single class. The clusters properly capture the class distribution. A low score means classes are scattered across clusters.

Maximizing completeness ensures clusters map well to the ground truth categories in the data. It is commonly used along with homogeneity to evaluate clustering quality.

3 EXPERIMENTAL RESULTS

3.1 K-Means Results

In this part, the evaluation criteria and visualizing the results are presented. We knew that K-Means is working well in the following datasets: The data is numerical and has a spherical or circular distribution. K-means clustering is most effective when the data is distributed in a way that allows for clearly defined clusters. This is typically the case with data that is normally distributed or has a circular pattern.

Also, the number of clusters is known or can be estimated. The main challenge in K-means clustering is determining the optimal number of clusters (k). This is often done by trial and error or using techniques such as the elbow method or silhouette analysis. The data is not too sparse or too large. K-means clustering can be applied to large datasets, but it can become computationally expensive for large or sparse datasets. In these cases, other clustering algorithms may be more suitable. As obvious, K-Means clustered the data appropriately when our dataset is distributed spherical or even circular. The Fake data we built has these distributions. Figure 1 illustrates the acceptable clustering results.

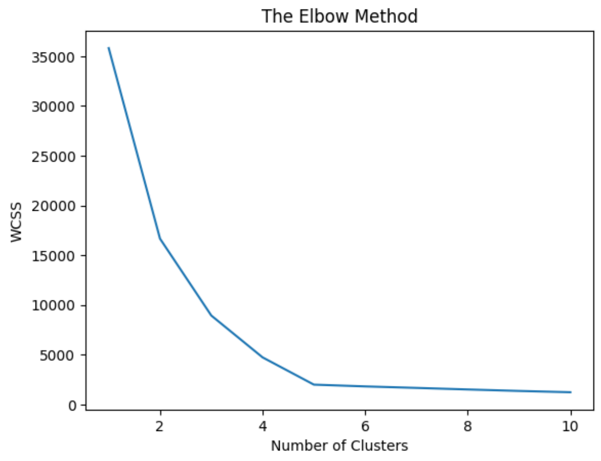

Figure 3: Elbow method for determining the number of optimum clusters.

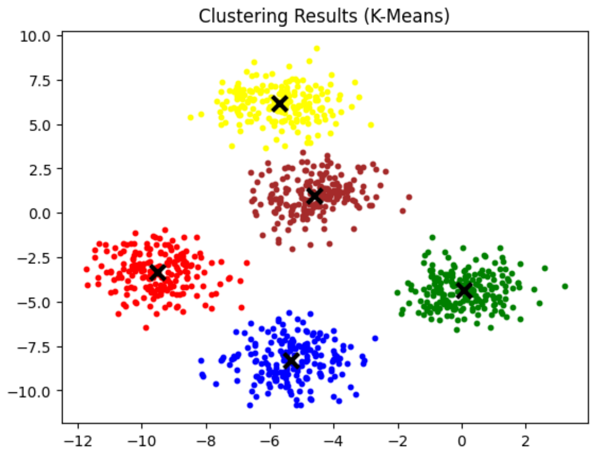

Figure 4: K-Means performance on the Fake dataset.

Since finding the best and most appropriate number of clusters is a crucial challenge, We are trying to use the WCSS algorithm to understand which number of clusters is suitable for our dataset. This method is called the Elbow method which helps us to figure the best number of clusters right on the elbow place. For this fake dataset, we represent the elbow diagram in Figure.

The California housing prices scatter plot exhibits a non-uniform distribution, challenging the suitability of k-means. To enhance clustering efficacy, data standardization, and appropriate distance metrics are crucial. Evaluating clustering results using metrics like silhouette score and Calinski-Harabasz score is essential, either. In the Figures below, a complete representation is illustrated to show how K-Means performs in California housing prices.

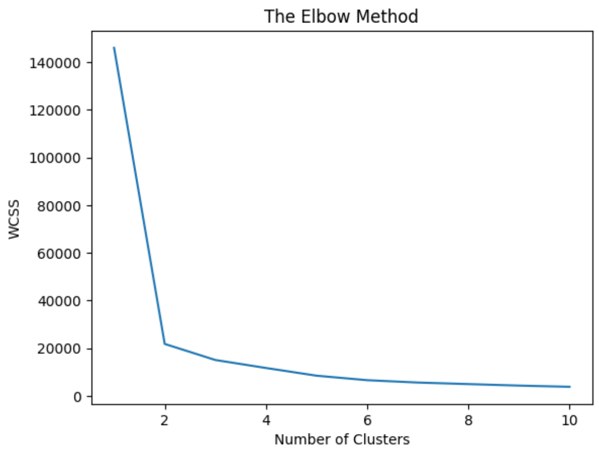

Figure 5: Elbow method for determining the number of optimum clusters.

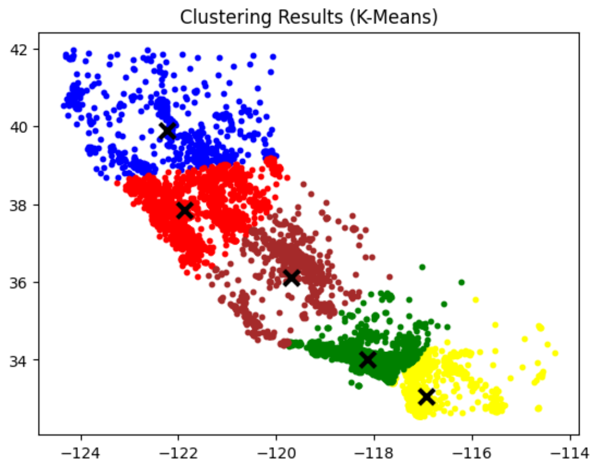

Figure 6: K-Means performance on the California house price dataset.

3.2 DBSCAN Results

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that is well-suited for identifying clusters of arbitrary shape and size. Based on this information, In the fake dataset, we see no significant difference between K-Means and DBSCAN while the algorithm and the way they work are completely different. On the other hand, DBSCAN can detect a different class of instances which are called noises with label -1.

Since DBSCAN is a density-based clustering algorithm that groups data points into clusters based on their density. It does this by finding regions of high density and assigning all data points in a region to the same cluster.

DBSCAN is a better choice for clustering the California housing prices scatter plot than k-means as shown in Figure8, as it can detect clusters of arbitrary shape. However, it is important to choose the correct parameters for DBSCAN, such as the minimum number of points in a cluster and the epsilon parameter, which defines the neighborhood of a data point.

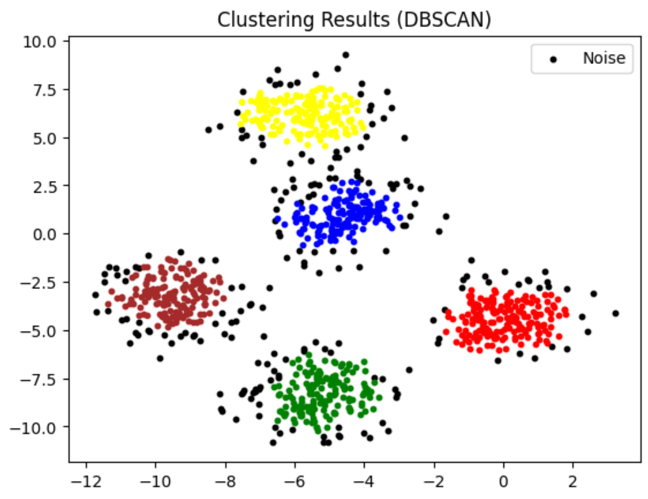

Figure 7: DBSCAN performance on the fake dataset.

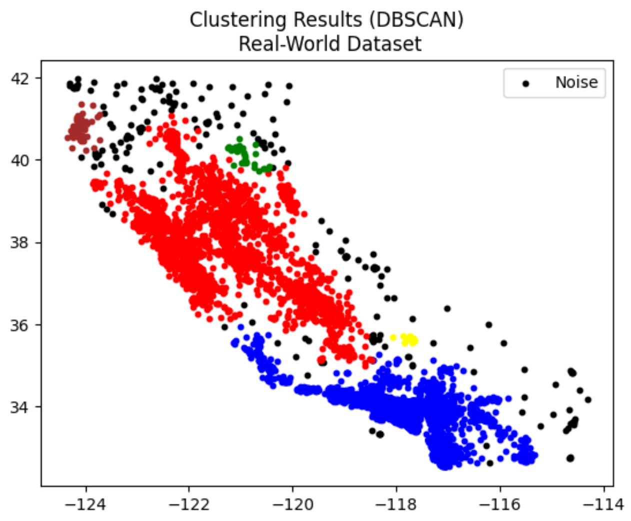

Figure 8: DBSCAN performance on the California housing price dataset.

3.3 HDBSCAN Results

Based on the dataset distribution and the HDBSCAN definition, we conclude that HDBSCAN is a good choice for clustering the California housing prices scatter plot, as it can detect clusters of arbitrary shape and does not require the user to specify the number of clusters.

Overall, we would recommend using DBSCAN or HDBSCAN to cluster the California housing prices scatter plot. These algorithms are able to detect clusters of arbitrary shape and do not require the user to specify the number of clusters. According to the illustration based on the DBSCAN and HDBSCAN in Figure 11 and Figure 12, no significant difference is witnessed.

The HDBSCAN algorithm is designed to perform well on both uniform and non-uniform datasets.

On uniform datasets, where the density of points is relatively consistent throughout, HDBSCAN can effectively identify clusters by connecting points within the same high-density region. The key advantage of HDBSCAN for uniform data is its ability to find clusters of varying shapes and sizes, unlike algorithms that enforce circular or globular clusters. The algorithm builds clusters by expanding outwards from core high-density points, allowing it to adapt to the natural shape of uniform clusters.

For non-uniform datasets, where density can vary significantly in different areas, HDBSCAN handles this by identifying cluster groups based on areas of higher local density. Points in sparser areas are treated as noise or outliers unless they extend the reach of a cluster originating from a dense area. This density-based notion allows HDBSCAN to discover meaningful clusters in non-uniform data. The main parameters allow adjusting the density threshold to suit the level of variability in the dataset. Overall, HDBSCAN provides a robust clustering approach effective for both uniform and highly non-uniform distributions compared to methods relying solely on distance metrics or fixed-shape clusters.

The HDBSCAN algorithm utilizes a condensed tree which is shown in Figure 14 and Figure 10 to hierarchically represent clusters of varying densities. It builds the tree by initially treating each point as a separate cluster, then iteratively combining the two closest clusters according to a nearest neighbor distance metric. The result is a hierarchical tree that reflects the variable density landscape of the dataset. Nodes higher in the tree represent clusters of lower density while leaf nodes form the highest density clusters. The tree provides a visualization of the density-based relationships between points. A key advantage is that it allows identifying stable cluster branches that persist across multiple distance thresholds, as opposed to unstable branches sensitive to small parameter changes. Overall, the condensed tree gives HDBSCAN its hierarchical clustering capabilities and provides valuable insights into the density structure of the dataset when visualized. It enables robustly extracting significant clusters from noisy data based on the hierarchical aggregation of points in areas of locally maximal density.

3.4 Comparison of Results

K-means is a partitioning clustering algorithm that divides data into a predefined number of clusters. It works by iteratively assigning data points to the nearest cluster centroid. The algorithm is simple and efficient, but it is not well-suited for data with non-uniform distributions or outliers. DBSCAN is a density-based clustering algorithm that identifies clusters based on the density of data points. It is more robust to outliers and non-uniform data distributions than k-means, but it can be sensitive to the choice of parameters, such as the eps (epsilon) and $minPts$ parameters. HDBSCAN is a hierarchical density-based clustering algorithm that combines the strengths of k-means and DBSCAN. It can identify clusters of arbitrary shape and size, and it is also robust to outliers and noise.

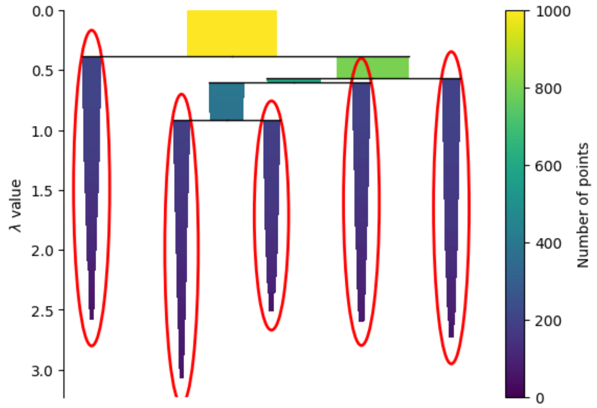

Figure 9: Condensed tree for the fake dataset

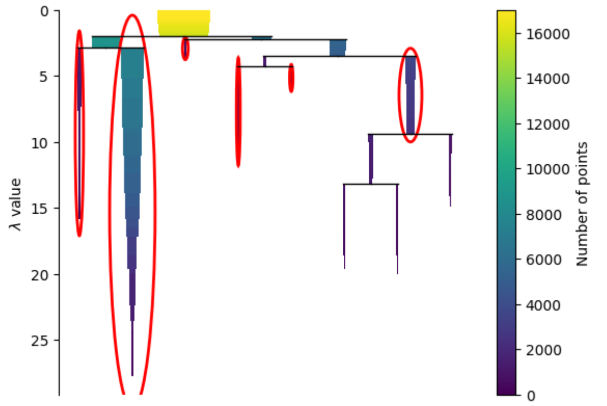

Figure 10: Condensed tree for the California price house price dataset

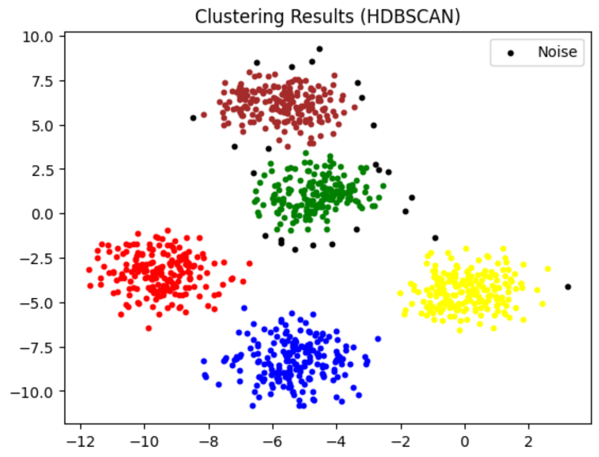

Figure 11: HDBSCAN performance on the fake dataset.

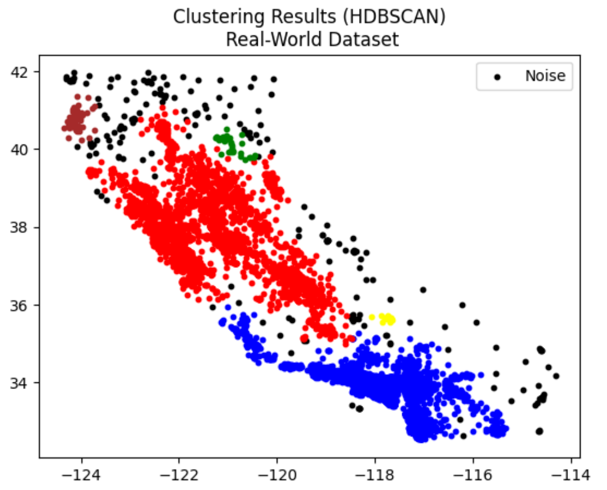

Figure 12: HDBSCAN performance on the California house price dataset.

However, it can be slower than k-means and DBSCAN.

From a better point of view, choosing the appropriate algorithm, based on the objective will be applied. for example, consider the below nutshell from our experiments:

1. **The number of clusters :** K-means requires the user to specify the number of clusters, while DBSCAN and HDBSCAN do not.
2. **The presence of outliers:** If the data contains outliers, DBSCAN and HDBSCAN are more robust than k-means.
3. **The dimensionality of the data:** DBSCAN and HDBSCAN can handle high-dimensional data, while k-means is more limited.

A complete comparison for each dataset has been done to see how much each algorithm per-

Table 4: Evaluation metrics for 3 clustering algorithms

	Silhouette score	Adjusted Rand score	Calinski-Harabasz score	Completeness
K-Means	0.68	0.99	4209.47	0.99
DBSCAN	0.46	0.67	615.35	0.71
HDBSCAN	0.46	0.67	615.35	0.71

forms better in uniform and non-uniform datasets. Previous evidence and research are illustrated in Figure13 for the fake dataset and in Figure14 for the California house pricing dataset. Also, we gathered all evaluation metrics tested for 3 algorithms in Table4.

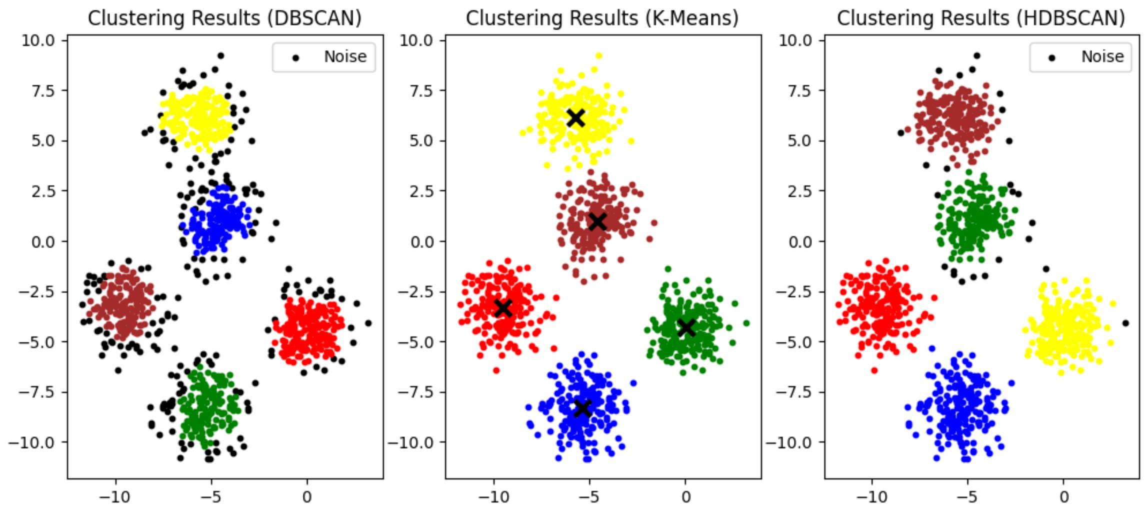

Figure 13: Comparing 3 implemented algorithms on the fake dataset.

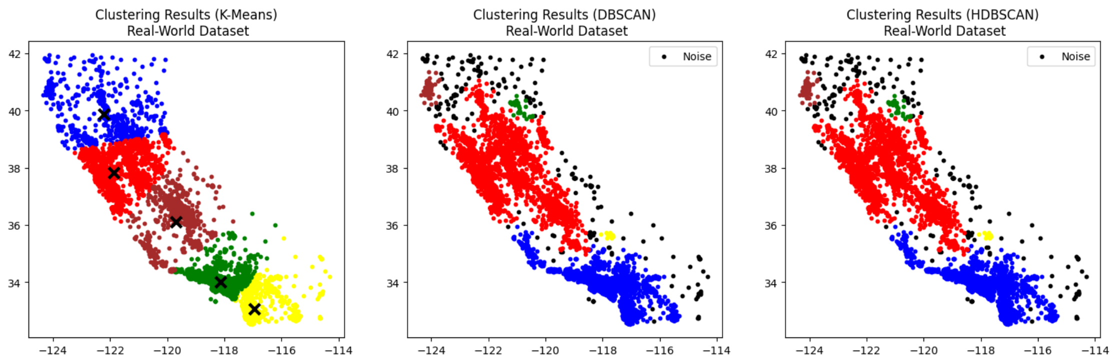

Figure 14: Comparing 3 implemented algorithms on the California house pricing dataset.

3.5 Analysis of Strengths/Weaknesses

K-Means is popular for its simplicity, efficiency, and empirical success. It scales well to large datasets and converges quickly to a partitioned clustering solution. However, k-means assumes spherical cluster shapes with similar density, making it unsuitable for non-uniform and non-globular distributions. It is also sensitive to outlier points that distort cluster centers. Choosing the number of clusters k can be challenging without domain expertise.

Table 5: Clustering algorithms strengths and weaknesses.

Algorithm	Type	Strengths	Weaknesses
K-Means	Partitional	Simple, efficient	Not well-suited for non-uniform data or outliers
DBSCAN	Density-based	Robust to outliers and non-uniform data	Sensitive to parameter settings
HDBSCAN	Hierarchical, density-based	Can identify clusters of arbitrary shape and size	Slower than k-means and DBSCAN

DBSCAN and HDBSCAN overcome many of k-means' weaknesses by adopting a density-based clustering approach. They can identify arbitrarily shaped clusters and handle varying-density datasets. DBSCAN suffers from difficulty setting input parameters like epsilon radius and minimum points. HDBSCAN improves on this by automatically computing these values. However, both algorithms have higher time complexity than k-means, with DBSCAN in particular struggling on large datasets. They also cannot directly specify the number of clusters upfront like k-means.

Table 6: Clustering algorithms based on the Application and Data type.

Algorithm	Data Type	Example
K-Means	Regularly distributed data	Customer segmentation based on age, gender, and income
DBSCAN	Non-uniformly distributed data	Fraud detection in credit card transactions
HDBSCAN	Data with outliers	Identifying customer segments in a retail dataset with a small number of outliers

In summary, k-means is fast and simple but inflexible, while density-based techniques are more robust but slower and reliant on parameter tuning. K-means may be preferred for uniform spherical data or when cluster number is known. DBSCAN and HDBSCAN offer advantages for non-linear relationships by identifying areas of high density, though with added computational costs. The optimal choice depends on the underlying cluster characteristics and goals of the analysis. A brief summary of the applications and the strengths and weaknesses of these three algorithms is given in Tables 5 and 6.

4 CONCLUSION

This report presented a comparative evaluation of three widely used clustering algorithms on synthetic and real-world data. The experiments demonstrated the superior performance of K-Means on datasets exhibiting globular cluster patterns with uniform density, where it achieved higher validity scores compared to the density-based techniques. However, DBSCAN and HDB-SCAN proved more robust in identifying arbitrary clusters for the nonlinear housing dataset.

Overall, the analysis highlighted the importance of selecting clustering algorithms based on underlying data characteristics. K-Means is suitable for spherical distributions but struggles with non-uniform densities and outliers. Density-based methods are more flexible at the cost of higher complexity and parameter sensitivity. The report also emphasized proper cluster validation to optimize algorithm hyperparameters and assess model quality.

In summary, this study provided a practical methodology for applied clustering analysis. The combination of artificial and real datasets, choice of appropriate evaluation metrics, and systematic comparison of multiple algorithms can serve as a template for developing unsupervised learning solutions tailored to specific data domains. The insights from this report can guide the selection and tuning of clustering techniques for real-world applications.