TABLE OF CONTENTS

1a. Abstract

Clustering is a major research area in data science, enabling data grouping based on similarity metrics to enhance human interpretability. Given the self-augmenting knowledge demands of organizations in e-commerce, finance, healthcare, logistics, and other fields, the application of clustering algorithms to high-dimensional datasets is a subject with particularly notable research and engineering attention. Many clustering algorithms are in use today—with considerable differences in parameters, metrics, scalability, and geometry—so some, of course, are better suited to certain types of data than others. It is sometimes a challenge, then, to determine which method of clustering is best for a given need. This project presents a comparative study of the effectiveness of clustering algorithms—including DBSCAN, BIRCH, KMeans, and spectral clustering—on both low- (spatial coordinates) and high- (MNIST text data) dimensional datasets. To reduce the dimensionality of high-dimensional vectorized text data, principal component analysis (PCA) is used. The performance of these algorithms is evaluated both visually (by plotting the labeled clusters on a two-dimensional chart), and with the algorithmic methods of silhouette score, Calinski-Harabasz index (CHI), and Davies-Bouldin index (DBI) to determine the quality and efficiency of the clustering algorithms on the range of tested datasets. This study's program code also features automatic adjustment for certain clustering parameters based on maximizing silhouette score (e.g. number of clusters in k-means, or all three parameters for BIRCH), but manual inputs can still be used.

1b. Introduction

Clustering plays an important role in modern data science, offering a way to group similar data points without requiring labeled examples. It is an unsupervised learning technique which is particularly valuable in contexts where human interpretation of patterns is paramount, such as customer segmentation, anomaly detection, and data summarization. As organizations increasingly rely on data-driven strategies, the need for clustering methods individualized to various data types and scales continues to grow. The great amount of available clustering algorithms, each with unique assumptions and parameterizations, presents both an opportunity and a challenge. Algorithms such as DBSCAN excel at identifying arbitrarily shaped clusters, while KMeans is widely used for its simplicity and efficiency. BIRCH is designed for scalability with large datasets, and spectral clustering is notable for its ability to handle complex, non-convex clusters. Crucially, no single method universally outperforms others; the choice of algorithm often depends on the dataset's structure, dimensionality, and application requirements.

This project aims to address these challenges by systematically evaluating the performance of several popular clustering algorithms across datasets of differing dimensionalities. By incorporating dimensionality reduction techniques such as principal component analysis (PCA), the study also explores how preprocessing impacts clustering outcomes. Both qualitative (visual) and quantitative (unsupervised benchmark) metrics are employed to compare things, highlighting the strengths and limitations of each method. The limitations of parameter optimization based on silhouette scores also underscores the importance of fine-tuning in achieving meaningful results.

2a. Problem Statement

To evaluate the performance of commonly used clustering algorithms on datasets with distinct dimensional structures to understand their applicability, limitations, and clustering efficacy in high- and low-dimensional data contexts.

2b. Goal & Objectives

Data pre-processing:

- The real-world datasets used are already cleaned and normalized, so there is no need to map the text data onto word embedding vectors. However, this project still utilizes PCA on the text data to vastly reduce its dimensions.
- Reading of datasets from local files
- Python libraries used: HDF5, NumPy, pandas

Algorithm implementation:

- Implementation, optimization, and application of clustering algorithms
- Python libraries used: scikit-learn, kneed, matplotlib, seaborn, NumPy, pandas

Visualization and interpretation:

- Representation of clustering via 2D projections
- Python libraries used: matplotlib, seaborn, NumPy

Performance evaluation:

- Evaluation of clustering quality and interpretability using synthetic benchmark scores (silhouette score, CHI, DBI)
- Python libraries used: scikit-learn, kneed

3. Related Work

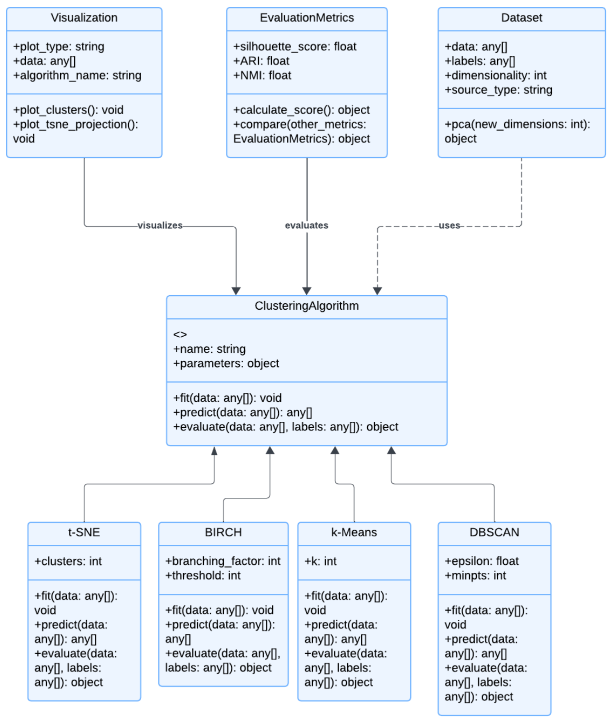
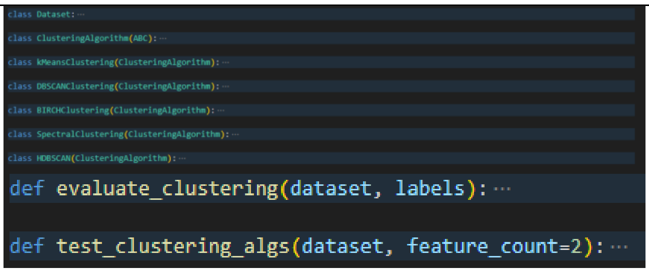
Clustering is a well-explored field of data mining research. To start, an older article by Blei (2012) discusses probabilistic models for topic classification, starting with the simplest form of Latent Dirichlet Allocation. The author also proposes that a future avenue of research should be to "develop evaluation methods that match how the algorithms are used," which he calls a "model checking" problem, i.e., given some new dataset, what model should one use? This long-term interest in evaluating clustering quality helped motivate this study. One particularly recent study by Yin et al. (2024) provides a "rapid review" of different clustering algorithms, aimed at providing an overview of them, their characteristics, and suitable tasks. This study was used as a basis for much of this project, particularly under the category of “unsupervised learning.” It also includes a review of certain internal metrics of clustering quality, which, unlike comparison to precomputed clusters, revolve around “factors like the compactness (cohesion) of data samples within a cluster and the distinctiveness (separation) between clusters.” They list silhouette score and Davies-Bouldin Index as two measures from this category, which were both used in this study.

Beyond evaluation of existing models, other studies have, of course, proposed their own novel methods. A paper by Sharma et al. (2023) discusses some of these algorithms developed in recent years, especially focusing on the synthesis of deep learning models with clustering algorithms used in applications like anomaly detection. The paper describes the form in which these approaches follow as firstly defining some fixed input data, then undergoing a process of unsupervised learning (from the data alone) to understand relationships, before finally assigning each datum to a cluster. They conclude that “AI-primarily based clustering algorithms are [] effective at identifying outliers and outliers in massive datasets with many variables.” An example of a recent novel clustering algorithm—which has yet to receive widespread adoption—is K-DBSCAN, proposed in an article by Gholizadeh et al. (2020). They aim to address the limitations and needs of big data, acknowledging that, while DBSCAN is common and very effective, it is also rather slow for truly huge datasets. They describe big data in data mining as “datasets that cannot be managed and processed by conventional software tools on a single machine in an acceptable time” (which is largely outside the scope of this project assignment, but still of interest because it underscores the recent advancements in the field). To address this prohibitive time cost, they leveraged the K-means++ algorithm for initial grouping, with DBSCAN then being used to perform the final clustering of separate groups. This resulted in a massive reduction in computational complexity, boasting, in the best case, a 98% reduction compared to DBSCAN alone. They propose that future research investigate the parallelization of clustering algorithms to run on multiple machines, which could enable another jump in algorithmic capability.

Returning to this project’s own scope, now, another relevant paper is one by Hu et al. (2024). This paper focuses on the interpretability of different clustering algorithms, with an added emphasis of highly scalable architectures for big data. It does not provide a single definition of interpretability, but all its characterizations implicate the machine-human interface to make the black box of computation, and internal or emergent relations among data points, legible and meaningful to humans. It, like the paper by Sharma et al. (2023), also provides an overview of "the current state of explainable clustering algorithms, identifying key criteria to distinguish between various methods" to inform researchers and developers about which clustering algorithms may be best suited to their needs. None of these papers heretofore mentioned, though, numerically compare the performance of clustering algorithms to each other with various datasets. Probably this is in part because a one-size-fits-all approach of evaluation would be misleading to at least some people and their application needs, but it could still be helpful to see actual, low-level evaluations of how they perform on both synthetic and real-world datasets, with both high- and low- dimensionality. This project aims to fill some of this lacuna. Although the unsupervised performance metrics used in this study (silhouette score, CHI, DBI) are shown to only play a part in the greater role of performance evaluation, they are still instructive in demonstrating relative differences between clustering sets—and, indeed, in revealing their own limitations, when compared to visual graphs. One other slight influence is an older paper by van der Maaten and Hinton (2008), which proposed a technique called t-distributed stochastic neighbor embedding (t-SNE), now widely used for visualizing high-dimensional on two-dimensional spaces. Although this project did not end up implementing this algorithm for visualization purposes, the MNIST dataset they used to test it was also used here.

4. Design Challenges & Development

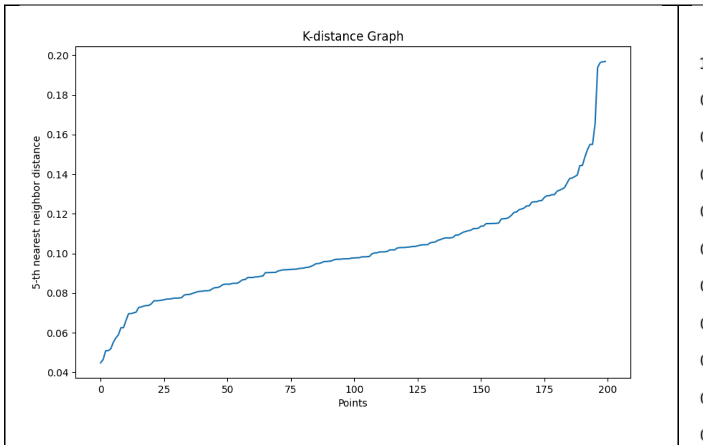
The basic design objectives of this project and its structure were laid out in the preliminary report. For the most part, the object-oriented manner of this proposal was followed by the implementation, though for EvaluationMetrics, everything was done outside of a class, as can be seen below. All the clustering algorithms, though, were implemented via polymorphism extending a base ClusteringAlgorithm abstract class, with overridden constructors to pass in different parameters as desired. The dataset class stores the dataset and dataset metadata and has methods for performing PCA and finding epsilon values. This class structure helps to cleanly separate the necessary code for each algorithm and adhere to the dependency inversion principle of programming.

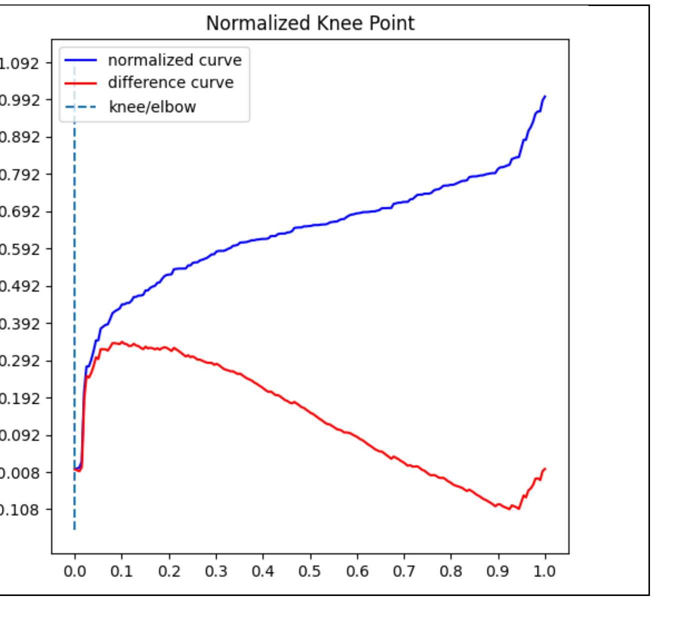
Proposed UML class design	Actual implementation
	

```python
class Dataset: ···
class ClusteringAlgorithm(ABC): ···
class kMeansClustering(ClusteringAlgorithm): ···
class DBSCANClustering(ClusteringAlgorithm): ···
class BIRCHClustering(ClusteringAlgorithm): ···
class SpectralClustering(ClusteringAlgorithm): ···
class HDBSCAN(ClusteringAlgorithm): ···
def evaluate_clustering(dataset, labels): ···

def test_clustering_algs(dataset, feature_count=2): ···
```

Several Python libraries were used in this project, some more niche than others. For instance, h5py was learned to extract data from very large hierarchical data files. One very important library used by this project is kneed, which is a library providing methods to help estimate the knee or elbow point of a k-nearest neighbors distance graph. This is used in the optimization of DBSCAN (for epsilon) and KMeans (for k). One challenge with this was that it was sometimes too sensitive to rough data and put the knee far too high. The first thing attempted to remedy this was applying a Gaussian filter, but no values for this produced satisfactory results. Instead, the default sensitivity parameter in the knee locator method was tweaked from one to two, and this produced values usually matching those of what a human would see. This is a graph with knee at around 0.15, and normalized graph using kneed:

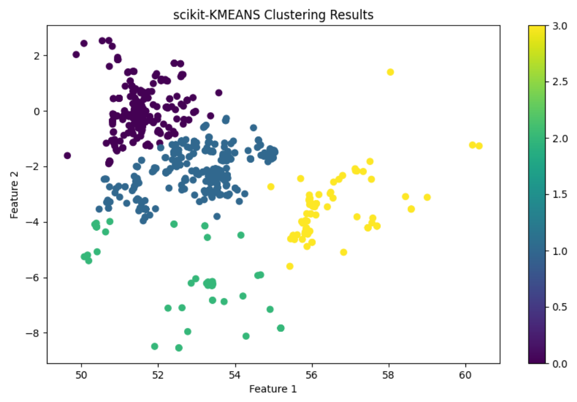
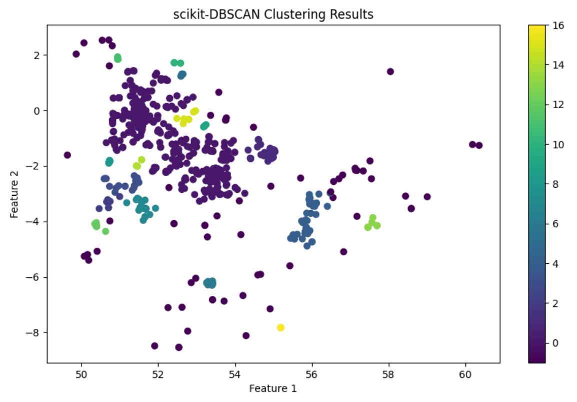


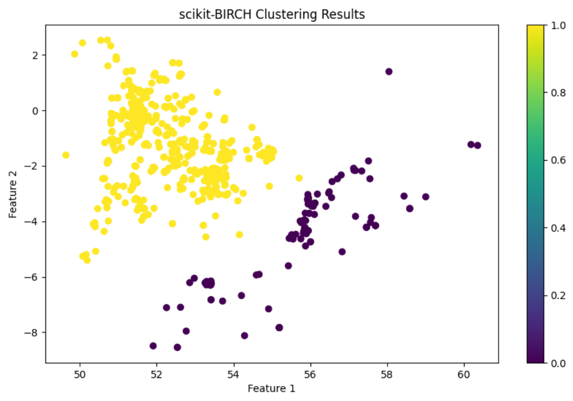


5. Implementation & Results

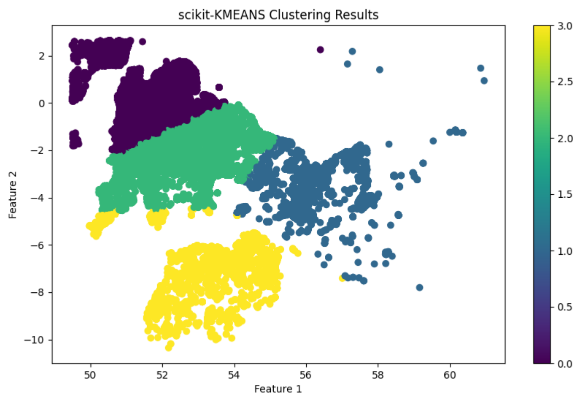
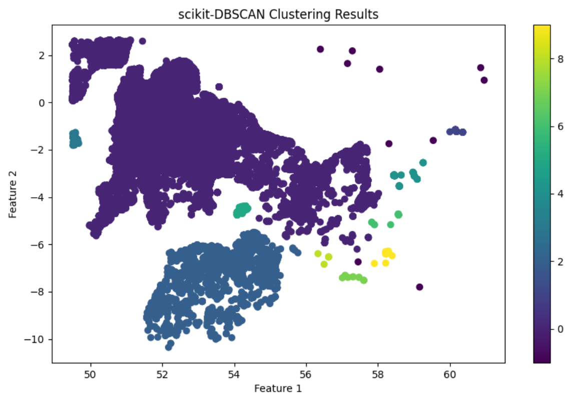
The performance of the algorithms varies widely across the datasets, depending on their shape and number of dimensions. Along with visual inspection, three unsupervised benchmarks were used in evaluating clustering performance: Silhouette Score (higher is better), Calinski-Harabasz Index (higher is better), and Davies-Bouldin Index (lower is better).

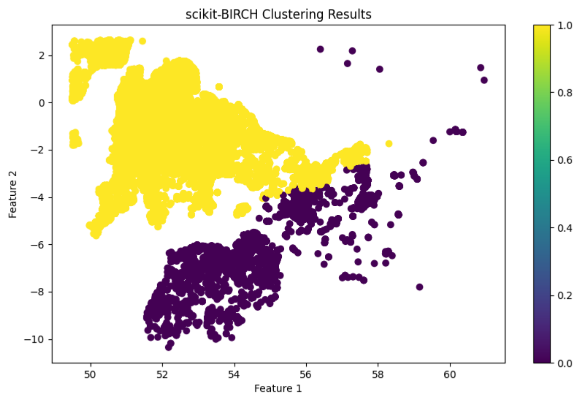
Clustering performance for truncated coordinate dataset (1,000 elements):

Algorithm	Visualization	Silhouette Score	CHI	DBI
KMeans		0.505	1028.706	0.715
DBSCAN		-0.076	78.931	1.319

BIRCH		0.556	739.621	0.747

Clustering performance for truncated coordinate dataset (100,000 elements):

Algorithm	Visualization	Silhouette Score	CHI	DBI
KMeans		0.484	103531.406	0.682
DBSCAN		-0.002	3977.864	0.991

BIRCH		0.523	55704.777	0.762

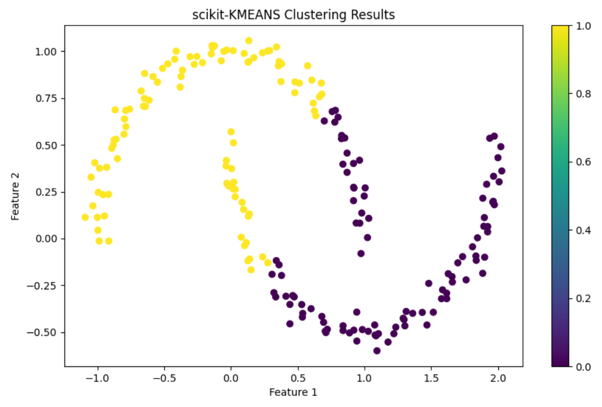
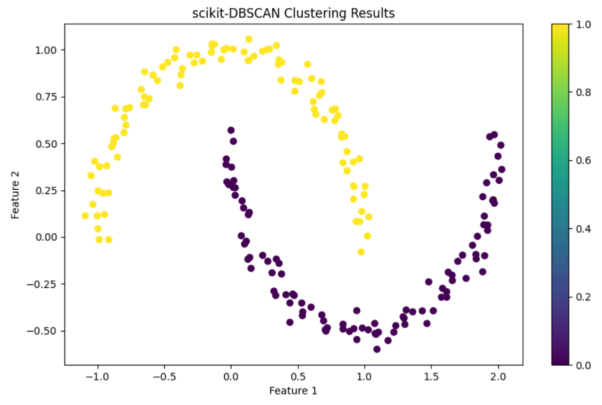
Looking at the clusters of the 1000-entry clustering, BIRCH produces the most discrete clusters, while KMeans makes more clusters which are evenly spaced. DBSCAN is much less clean, and also produces many outliers, or noise points. The performance of DBSCAN on this dataset of varying entry count is very interesting. Although it produces noise points at this scale, it may still be useful clustering by being smaller, but as the entry count increases a bit into the thousands, even this advantage falls away and DBSCAN produces a mess of noise. However, when the coordinate count is really high (e.g. 100k), the points finally form clearly distinct clusters, and DBSCAN pulls ahead again in finding these, though, the performance metrics still rate it poorly.

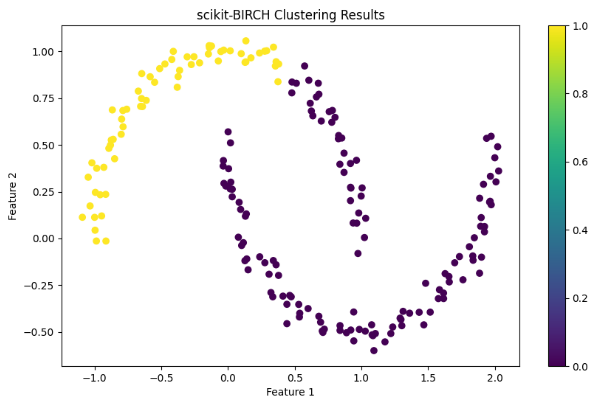
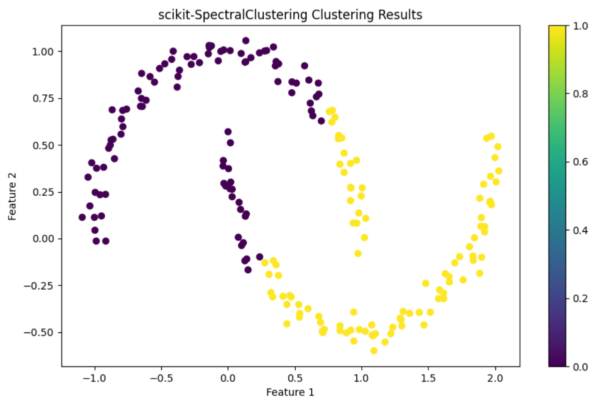
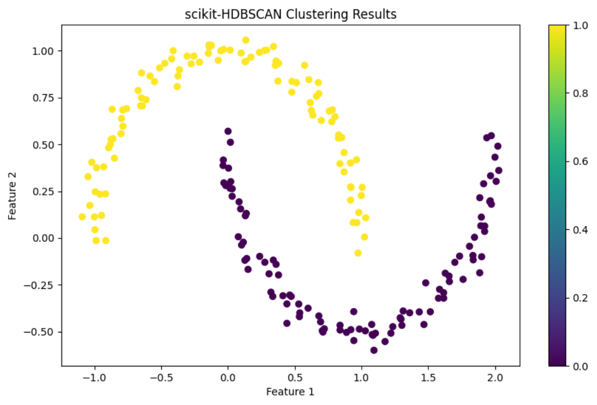
Performance metrics using “ground truths” like Rand index (RI) and Normalized Mutual Information (NMI) were not used in this project. Instead, only evaluations using the models themselves were performed, using silhouette score, Calinski-Harabasz index (CHI), and Davies-Bouldin index (DBI).

One of the things this revealed is that these scores alone are clearly not enough to evaluate the performance and quality of the datasets. For instance, in the “moons” sample dataset, the labels of DBSCAN (and HDBSCAN, which was identical) had a lower Silhouette Score Calinski-Harabasz index than both KMeans and BIRCH, even though it was the only one to correctly identify the two distinct clusters (even with KMeans and BIRCH having the number of clusters manually set to two, which produced a lower silhouette score). This is because these metrics tend to prefer more densely populated clusters, while the moon shapes in the test dataset were greatly elongated. This underscores the continued relevance of visual validation to determine certain aspects of clustering quality.

In terms of scalability, all the algorithms are reasonably quick to run on 100,000 entries or under, with the exception of BIRCH when finding automatic parameters. This is because, unlike the way KMeans or DBSCAN were used for this project, BIRCH can take advantage of three parameters—threshold, branching factor, and a precomputed number of clusters—which requires testing each combination of parameters (within some arbitrary range and increment limitations). For example, training BIRCH on only 1,000 elements of the coordinate dataset took about 16 seconds on a Ryzen 7 9700X system, while everything else (clustering, performance evaluation, and other parameter optimization) took only about three seconds. The performance evaluation of these algorithms generally took much more time than the clustering algorithms themselves, though this study does not benchmark their exact values.

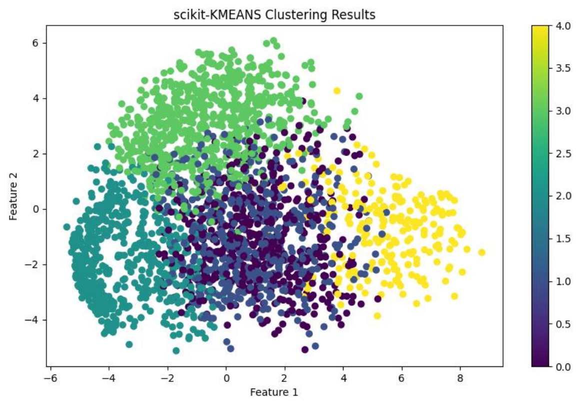
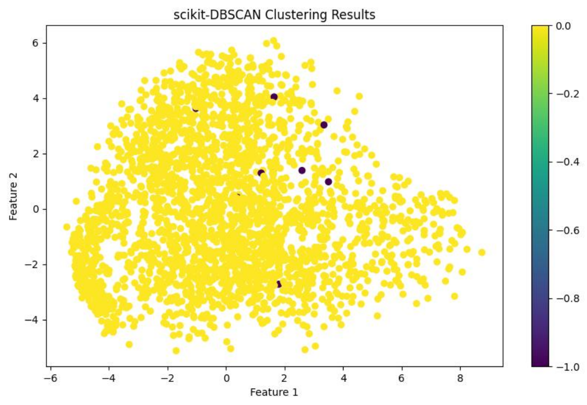
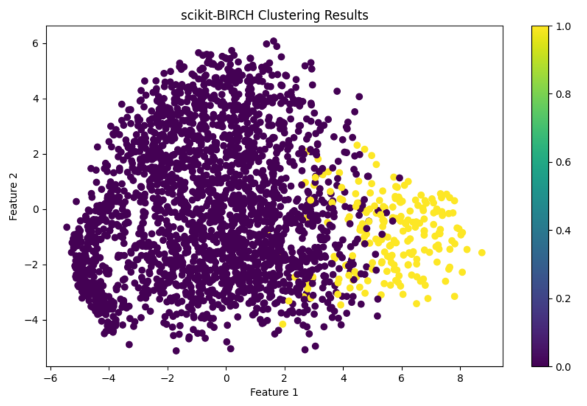
Clustering performance for synthetic moons dataset (200 elements):

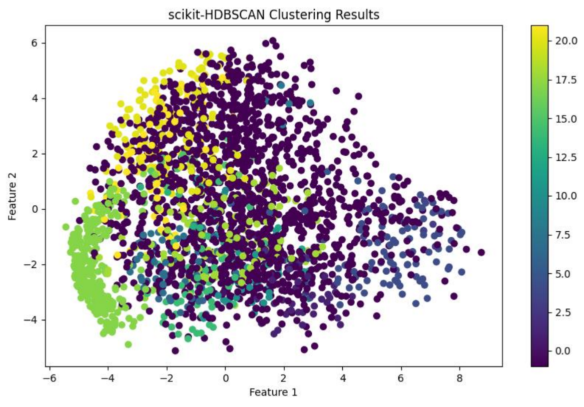
Algorithm	Visualization	Silhouette Score	CHI	DBI
KMeans (k set to 2)		0.487	293.118	0.780
DBSCAN(epsilon sensitivity set to 1)		0.328	130.727	1.157

BIRCH (threshold set to 0.2, branching factor set to 8, num clusters set to 2)		0.444	219.965	0.778
Spectral (num clusters set to 2)		0.488	292.238	0.780
HDBSCAN		0.328	130.727	1.157

Clustering performance for MNIST dataset (2500 dimensions, reduced to 20 via PCA):

Algorithm	Visualization	Silhouette Score	CHI	DBI

KMeans		0.138	261.497	2.125
DBSCAN		0.042	3.089	3.207
BIRCH (threshold set to 1, branching factor set to 30, num clusters set to 2)		0.163	205.191	1.791

HDBSCAN		-0.111	38.611	1.732

In the case of the high-dimensional dataset, it’s difficult to tell, visually, which clustering algorithms work best, since the number of reduced dimensions is 20, while the plot is only two-dimensional. DBSCAN proves insufficient for clustering here, while HDBSCAN is able to adjust its parameters and find distinct clusters, even though it scores very poorly. KMeans and BIRCH receive the best scores here. Spectral Clustering took prohibitively long in this test and was excluded.

6. Conclusion

In this project, we learnt about the implementation, application, and evaluation of several different clustering algorithms—KMeans, DBSCAN, BIRCH, Spectral Clustering, and HDBSCAN. The project explored use-case scenarios of these clustering algorithms, both well-established and some more novel, and compared how they performed in different contexts. It also showed how to use scores to rate the quality and interpretability of computed clusters numerically, but the detachment of some scores from the visible quality of the clustering, for instance on the moons dataset, shows that visual inspection and manual adjustments are still extremely important.

The scope of this project limited us to only using unsupervised metrics for performance evaluation—i.e., clustering quality was evaluated on its own basis, without comparing one calculated clustering set to some pre-known "ground truth" dataset. An obvious avenue for expansion of this project, then, would be to implement these performance metrics which require ground truths, such as Rand index (RI), Adjusted Rand index (ARI), and Normalized Mutual Information (NMI). Another limitation of this project was the method used to measure the distance between points. Euclidean distance was used as the default for all calculations, which usually performs well in 2D space, but other metrics may be better in higher dimensions. A future expansion to this project could therefore explore other metrics for geometry and compare their relative effectiveness, particularly with high-dimensional datasets. One last avenue for expansion is to more formally benchmark the scalability and "speed" of the clustering algorithms, as well as the metrics used for performance evaluation (for instance, CHI is generally the fastest to compute), which becomes more important as dataset size grows as the time complexity scales linearly with the number of dataset entries.

7. References

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77. https://doi.org/10.1145/2133806.2133826

Gholizadeh, N., Saadatfar, H., & Hanafi, N. (2020). K-DBSCAN: An improved DBSCAN algorithm for big data. *The Journal of Supercomputing*, *77*(6), 6214–6235. https://doi.org/10.1007/s11227-020-03524-3

Hu, L., Jiang, M., Dong, J., Liu, X., & He, Z. (2024). *Interpretable clustering: A survey*. ArXiv.org. https://arxiv.org/abs/2409.00743

Malzer, C., & Baum, M. (2020). A Hybrid Approach to Hierarchical Density-based Cluster Selection. *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 223–228. https://doi.org/10.1109/MFI49285.2020.9235263

Sharma, S., Singh, V., Jeyalaxmi M, Chaudhari, P. B., Das, N., & Garg, A. (2023). Advances in clustering algorithms for large-scale data processing with AI. *International Conference on Smart Generation Computing, Communication and Networking*, 1–7. https://doi.org/10.1109/smartgencon60755.2023.10441990

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

Yin, H., Aryani, A., Petrie, S., Nambissan, A., Astudillo, A., & Cao, S. (2024, January 14). *A rapid review of clustering algorithms*. ArXiv.org. https://doi.org/10.48550/arXiv.2401.07389