Work 4

Carlos Jiménez, Sheena Lang, Zachary Parent, and Kacper Poniatowski

December 23, 2024

1 Abstract

This report presents an investigation into dimensionality reduction techniques and their impact on clustering performance, with a particular focus on Principal Component Analysis (PCA). We implement our own PCA algorithm and compare it against scikit-learn's implementation, evaluating both accuracy and computational efficiency.

The study utilizes two distinct datasets (Mushroom and Vowel) to assess the effectiveness of dimensionality reduction in conjunction with clustering algorithms, specifically Global K-Means and OPTICS. We explore various visualization techniques, including PCA and UMAP, to represent high-dimensional data in lower-dimensional spaces.

Our analysis demonstrates the trade-offs between dimensionality reduction and clustering performance, providing insights into optimal configurations for different data characteristics. The results show that our PCA implementation achieves comparable performance to scikit-learn's version, and that carefully selecting parameters for each stage of the pipeline can lead to better visualizations.

2 Background and Related Work

3 Methods

Dimensionality Reduction In this study, we implement our own version of Principal Component Analysis (PCA) as the primary dimensionality reduction technique. Our implementation follows the standard PCA algorithm, computing eigenvalues and eigenvectors of the covariance matrix to identify principal components. For validation and comparison purposes, we maintain a parallel implementation using scikit-learn's PCA [4], which serves as our baseline. This approach builds upon our previous work with clustering algorithms [3].

Clustering We employ two distinct clustering algorithms: Global K-Means [2] and OPTICS [1]. For Global K-Means, we use dataset-specific configurations: the Mushroom dataset uses `n_clusters=2`, `max_iterations=100`, and `tolerance=1e-3`, while the Vowel dataset uses `n_clusters=11`, `max_iterations=100`, and `tolerance=1e-4`. The OPTICS algorithm is similarly tuned with dataset-specific parameters: for Mushroom, we use `min_samples=10`, `min_cluster_size=5`, and `xi=0.1` with euclidean metric, while Vowel uses `min_samples=20`, `min_cluster_size=10`, and `xi=0.1` with manhattan metric.

Metrics To evaluate the effectiveness of our dimensionality reduction and clustering approaches, we employ several metrics. For clustering quality assessment, we use both internal metrics (Davies-Bouldin Index, Calinski-Harabasz Index) and external metrics (Adjusted Rand Index, F-Measure). Additionally, we measure the computational efficiency and accuracy of our PCA implementation against the scikit-learn baseline.

Visualization Our visualization strategy employs two main techniques: PCA and UMAP. While PCA serves both as a dimensionality reduction method and visualization tool, we specifically use it to project high-dimensional data onto 2D and 3D spaces for visual analysis. UMAP complements

this by providing an alternative visualization approach, particularly useful for preserving local structure in the data. Both techniques are applied to visualize the original data distributions and the resulting cluster assignments.

4 Results and Analysis

Full results, in tabular and graphical form, are available in the appendix (Section 6).

First we plot the original datasets to visualize the data distribution. The original Vowel dataset is shown in Figure 1. The original Mushroom dataset cannot give us meaningful visual information at this time since all its features are categorical and one-hot encoded.

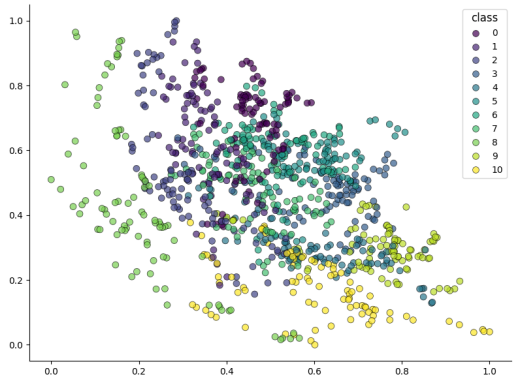


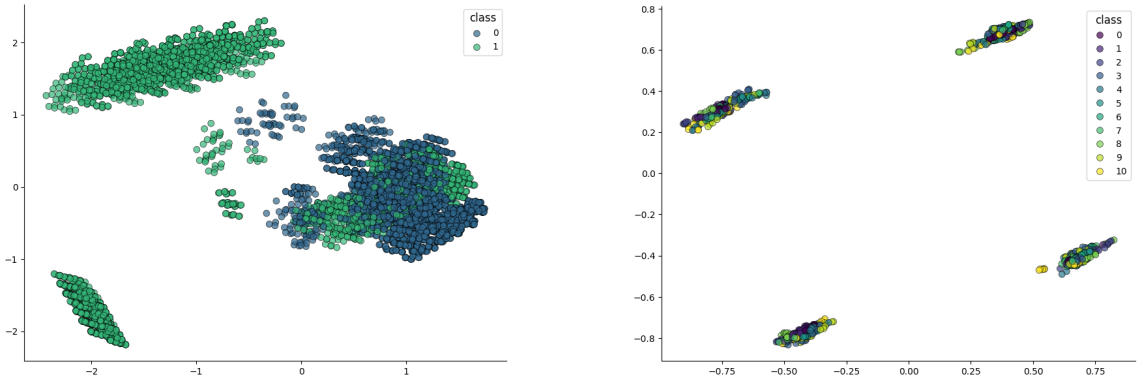
Figure 1: Visualization of the original Vowel dataset

We developed our own implementation of PCA and compared it with several scikit-learn based PCA implementations. The results of the reduced datasets using our PCA implementation are shown in Figure 2.

Table 1 highlights the comparison of average reduction runtime and F-measure across different reduction methods and clustering models. It is easy to note that `SklearnPCA` achieves the fastest average reduction runtime (0.003899 seconds), followed closely by `Our PCA` (0.011565 seconds). However, `IncrementalPCA` is significantly slower (0.074605 seconds). In terms of clustering performance, measured by the F-measure, `Global Kmeans` consistently outperforms `OPTICS` for all reduction methods. Notably, both `Our PCA` and `SklearnPCA` paired with `Global Kmeans` achieve the highest average F-measure (0.410158), while `IncrementalPCA` combined with `Global Kmeans` achieves a slightly lower score of 0.381697.

4.1 Our PCA vs sklearn.decomposition.PCA

The visualization compares the reduced datasets obtained using our custom PCA implementation (Figure 2) with those derived from the Scikit-learn PCA implementation (Figure 3). Both methods produce qualitatively similar results, preserving the overall structure and class separability of the datasets. In the Mushroom dataset (subplots 3a), the two main clusters corresponding to the two



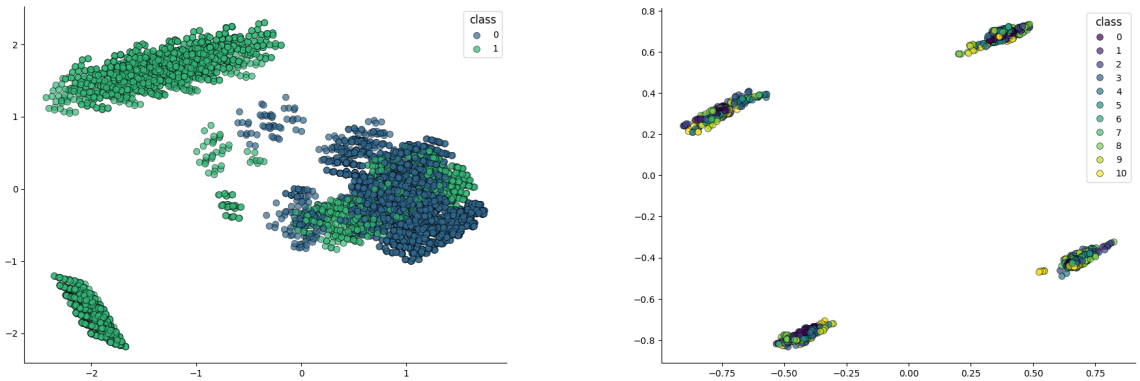
(a) Reduced Mushroom dataset

(b) Reduced Vowel dataset

Figure 2: Visualization of the reduced datasets using our PCA implementation.

Reduction Method	**Clustering Model**	**Avg. Reduction Runtime**	**Avg. F-Measure**
IncrementalPCA	Global Kmeans	0.074605	0.381697
IncrementalPCA	OPTICS	0.074605	0.024717
SklearnPCA	**Global Kmeans**	**0.003899**	**0.410158**
SklearnPCA	OPTICS	0.003899	0.021456
Our PCA	Global Kmeans	0.011565	0.410158
Our PCA	OPTICS	0.011565	0.021457

Table 1: Aggregated Results for PCA Approaches



(a) Reduced Mushroom dataset

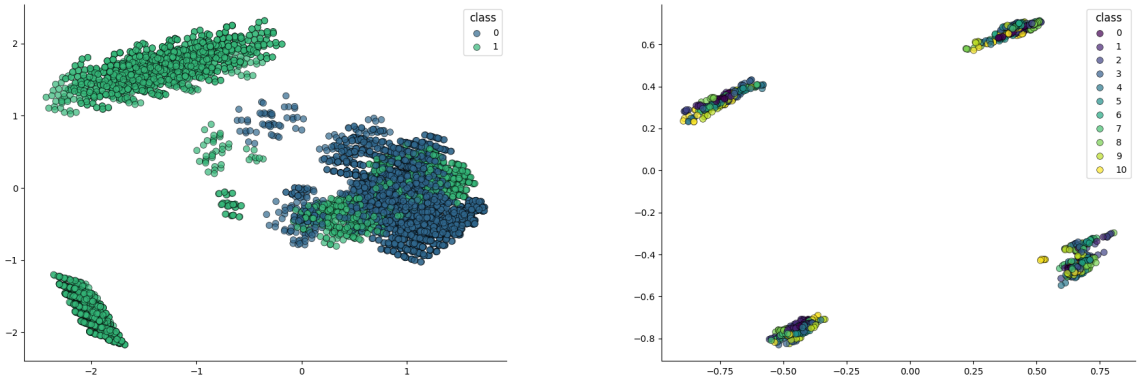
(b) Reduced Vowel dataset

Figure 3: Visualization of the reduced datasets using Scikit-learn basic PCA.

classes are clearly distinguishable in both implementations, indicating consistency in dimensionality reduction. For the Vowel dataset (subplots 3b), the distribution of the reduced data points aligns closely between the two approaches, with clusters for different classes exhibiting comparable

orientations and separations.

4.2 Our PCA vs sklearn.decomposition.IncrementalPCA



(a) Reduced Mushroom dataset

(b) Reduced Vowel dataset

Figure 4: Visualization of the reduced datasets using Incremental PCA.

The visualization compares the reduced datasets obtained using our custom PCA implementation (Figure 2) with those derived from Incremental PCA (Figure]4). Both methods effectively capture the structure of the data, preserving the class separability in the reduced dimensionality space. In the Mushroom dataset (subplots 4a), the clustering of the two classes remains consistent between the two approaches, with both implementations producing similar geometric distributions. For the Vowel dataset (subplots 4b), Incremental PCA closely aligns with the results from our PCA implementation, maintaining the orientation and separation of class clusters. This indicates that both methods achieve comparable results despite differences in computation techniques, with Incremental PCA being particularly advantageous for large datasets where memory efficiency is critical. The similarity in the visual outcomes demonstrates the reliability of our PCA implementation and its ability to replicate the performance of advanced techniques like Incremental PCA.

4.3 Clustering with reduction vs without

This section analyses the performance of the clustering algorithms with and without reductions of the feature space. As mentioned previously in the report, the reductions of the feature space are done by applying different implementations of PCA. The PCA implementations are the following:

- PCA via SKLearn
- Our own implementation of PCA
- Kernel PCA via SKLearn

Figures 5 display the results of the clustering algorithms with and without PCA against the two datasets used, 'Mushroom' and 'Vowel'. Mean of the F-measure is used to track the performance of the algorithms.

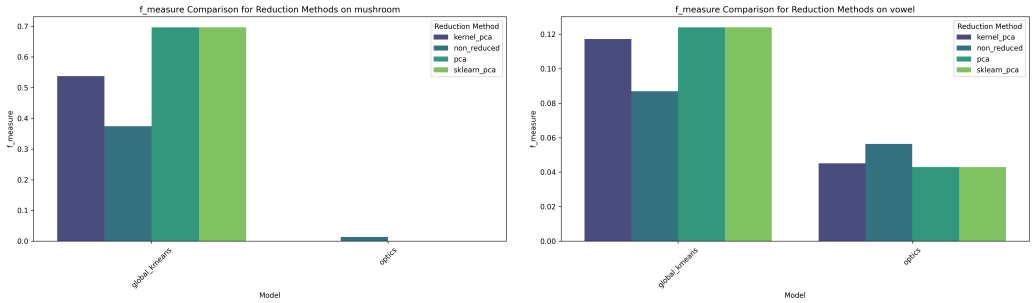


Figure 5: Mean F-measure score with and without PCA for Mushroom (left) and Vowel (right) datasets.

From analysing the results, it is evident that the PCA implementations generally have an impact on the clustering performance.

For the 'Global K-Means' algorithm, each of the three PCA implementations have a positive impact on the clustering performance for both datasets. On the 'Mushroom' dataset, 'Global-KMeans' achieves an F-Measure score of approximately 0.37 without PCA. Applying Kernel PCA via SKLearn, the F-Measure score increases to approximately 0.53. Our own implementation of PCA, as well as PCA vs SKLearn, achieves an F-Measure score of approximately 0.70, which is the highest score achieved. A similar situation is observed for the 'Vowel' dataset. Without PCA, 'Global-KMeans' achieves an F-Measure score of approximately 0.082. Applying Kernel PCA via SKLearn, the F-Measure score increases to approximately 0.11. Our own implementation of PCA, as well as PCA viaLearn, once again achieve the highest score of approximately 0.12.

Upon analysing the results for the 'OPTICS' algorithm, it is evident that the PCA implementations have an opposite effect on the clustering performance. For the 'Mushroom' dataset, 'OPTICS' achieves an F-Measure score of approximately 0.02 without PCA. Applying any of the PCA implementations results in a decrease in the F-Measure score, resulting in an F-Measure score of approximately 0.0 for all three PCA implementations. The same situation is observed for the 'Vowel' dataset. Without PCA, 'OPTICS' achieves an F-Measure score of approximately 0.058. Applying any of the PCA implementations results in a decrease in the F-Measure score. When applying Kernel PCA via SKLearn, the F-Measure score decreases slightly to approximately 0.043. Once again, our own implementation of PCA and PCA via SKLearn result in the same F-Measure score of approximately 0.04.

PCA improves 'Global K-Means' performance by reducing dimensionality and putting an emphasis on the most important features. This is evident in the results, where the F-Measure score is higher when PCA is applied. However, the same cannot be said for 'OPTICS'. PCA reducing the dimensionality of the feature space may have a negative impact on the performance of 'OPTICS' because it's density-based approach is sensitive to transformations that distort local density structures, which are critical for detecting clusters[5].

Table 2 highlights the best-performing models by dataset. For both datasets, the best performing model is 'Global K-Means', as opposed to 'OPTICS'. This isn't surprising as we have seen previously

that 'OPTICS' has a lower F-Measure score.

Table 2: Best Performing Models by Dataset (Based on F-Measure)

Dataset	Clustering Model	Reduction Method	F Measure	Ari	Chi	Dbi	Clustering Runtim
mushroom	global_kmeans	sklearn_pca	0.8933	0.6244	2723.1162	1.6517	14.
vowel	global_kmeans	kernel_pca	0.1678	0.0333	10282.9355	0.6666	13.

When comparing the reduction methods, Table 3 demonstrates the impact of different PCA implementations. Once again, the results are consistent with the previous analysis: 'Global K-Means' outperforms 'OPTICS' in every method.

Table 3: Best Performing Models by Reduction Method (Based on F-Measure)

Dataset	Clustering Model	Reduction Method	F Measure	Ari	Chi	Dbi	Clustering Runtim
mushroom	global_kmeans	kernel_pca	0.7036	0.2127	7596.6534	0.5166	8.
mushroom	global_kmeans	non_reduced	0.3742	-0.0011	207.4272	1.4914	8.
mushroom	global_kmeans	pca	0.8933	0.6244	2723.1162	1.6517	14.
mushroom	global_kmeans	sklearn_pca	0.8933	0.6244	2723.1162	1.6517	14.

4.4 Best Visualization Configurations

The best visualization of the clustering for the mushroom dataset demonstrates two well-separated and compact clusters using kernel PCA reduction, global k-means clustering, and PCA visualization. This result, achieved with the configuration `pca_n_components=3`, `kernel=rbf`, `gamma=0.1`, `n_clusters=2`, `max_iterations=100`, `tolerance=0.001`, and `random_state=4`, effectively highlights the separability of the dataset's two classes in the transformed space.

PCA visualization, especially with Kernel PCA, effectively captures the non-linear relationships in the mushroom dataset by mapping the data into a higher-dimensional space and reducing it to compact, separable clusters. This is well-suited for the mushroom dataset because it has only two distinct classes and exclusively categorical features, making it easier to uncover clear separability in the transformed space.

The best visualization of the clustering for the vowel dataset is achieved using kernel PCA reduction combined with global k-means clustering and UMAP visualization. This result, obtained with the configuration `umap_n_components=11`, `kernel=rbf`, `gamma=1`, `n_clusters=11`, `max_iterations=100`, `tolerance=0.0001`, and `random_state=4`, effectively captures the dataset's complex structure.

Although the clusters are not as compact as those in the mushroom dataset and often appear as lines rather than blobs, UMAP visualization provides the most interpretable representation for this dataset. The inherent difficulty of finding clear and well-separated clusters is heightened by the presence of 11 natural clusters corresponding to the dataset's diverse vowel classes.

UMAP visualization preserves both local and global structures, which is essential for the vowel dataset due to its high dimensionality and the presence of 11 intricate clusters. Unlike PCA, UMAP can effectively handle the dataset's non-linear relationships and overlapping clusters, revealing patterns that are more interpretable in this complex dataset.

Figure 6: Clustering visualization for the mushroom dataset using kernel PCA and global k-means clustering.

5 Conclusion

5.1 Key Findings

5.2 Practical Implications

5.3 Limitations and Future Work

5.4 Final Remarks

Figure 7: Clustering visualization for the vowel dataset using UMAP and global k-means clustering.

References

[1] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, June 1999.

[2] Jakob J. Verbeek Artistidis Likas, Nikos Vlassis. The global k-means clustering algorithm. 12, 2001.

[3] Carlos Jiménez, Sheena Lang, Zachary Parent, and Kacper Poniatowski. Implementation, evaluation, and comparison of k-means and other clustering algorithms. 2024.

[4] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Philipp Sepin, Jana Kemnitz, Safoura Rezapour Lakani, and Daniel Schall. Comparison of clustering algorithms for statistical features of vibration data sets, 2023.

6 Appendix