Comparing classification performance on high vs. reduced dimensional data with dimensionality reduction methods

Wesley N. Galvão
Federal University of São Carlos
wesleygalvao@estudante.ufscar.br

Mohammad S. Joshan
Federal University of São Carlos
mohammad@estudante.ufscar.br

https://github.com/galvaowesley/dimensionality_reduction

Abstract

This work investigates the impact of dimensionality reduction and machine learning, leveraging Python libraries. Comparative analysis unveils nuanced method performances, with Kernel PCA excelling in supervised classification. Challenges in unsupervised clustering are addressed, and the visualization of 2D embeddings not only provides insights into their functionality but also aligns with theoretical expectations and obtained results. The study emphasizes the importance of selecting methods in harmony with dataset characteristics.

1. Introduction

In the vast landscape of data analysis, the pursuit of discerning meaningful patterns from intricate datasets stands as a formidable challenge[1]. This endeavor seamlessly marries the precision of engineering techniques with the analytical power of mathematical methods to unveil latent structures within data autonomously. Positioned prominently at this intersection is Machine Learning, a transformative tool that empowers systems to glean patterns from data, enabling them to make predictions or decisions without explicit programming.

This article embarks on a nuanced exploration of the pivotal role played by Dimensionality Reduction (DR) methods within . As datasets burgeon in complexity, dimensionality reduction emerges as a crucial preprocessing step, particularly when grappling with high-dimensional data. The overarching objective revolves around distilling essential features while mitigating the curse of dimensionality, thereby amplifying the efficiency and interpretability of subsequent classification models.

The Significance of Dimensionality Reduction: The astronomical growth in data generation across various domains, from biomedical research to financial analytics, has ushered in an era where datasets often exhibit high dimensionality. In this context, the curse of dimensionality rears its head, introducing challenges such as increased computational complexity, heightened risk of overfitting, and diminished model interpretability. Dimensionality reduction methods offer a remedy by transforming the data into a more manageable form while preserving its essential characteristics.

Navigating the Landscape of Dimensionality Reduction Techniques: A plethora of dimensionality reduction methods has emerged, each with its unique strengths and applications. Principal Component Analysis (PCA), a stalwart in the field, employs mathematical techniques such as eigenvalue decomposition to uncover the most salient features in the data. Kernel PCA extends this methodology by leveraging the kernel trick, facilitating the extraction of nonlinear patterns.

ISOMAP, on the other hand, tackles the challenge of high-dimensionality by capturing the intrinsic geometry of the data through geodesic distances. Locally Linear Embedding (LLE) focuses on preserving local relationships in the data, while Laplacian Eigenmaps delve into graph theory, constructing a low-dimensional representation by minimizing the Laplacian eigenvalue.

In the realm of pattern recognition, t-Distributed Stochastic Neighbor Embedding (t-SNE) stands out for its prowess in visualizing high-dimensional data in two or three dimensions. These dimensionality reduction techniques serve as indispensable tools in the data scientist's arsenal, each offering a unique perspective on how to navigate the intricate landscape of high-dimensional datasets.

Comparative Analysis: Unraveling the Impact on Classification Performance: The crux of our exploration lies in dissecting the impact of dimensionality reduction on classification performance. High-dimensional datasets, brimming with intricate patterns, often pose computational challenges and hinder the interpretability of models. By subjecting these datasets to dimensionality reduction, we seek to unravel whether the sacrifice of some information re-

sults in enhanced classification accuracy and efficiency. Our comparative analysis spans a spectrum of classification methods, from traditional algorithms such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) to modern techniques like Multilayer Perceptron (MLP) and Gaussian Mixture Models (GMM). We aim to discern how the employment of dimensionality reduction methods influences the intricate dance between precision and efficiency in the realm of classification.

Real-world Applications and Practical Implications: To ground our discussion, we delve into practical case studies and experiments. These real-world applications showcase the tangible benefits and potential pitfalls associated with employing dimensionality reduction methods in tandem with classification tasks. By drawing insights from diverse domains, we paint a comprehensive picture of the practical implications and limitations of these techniques. Through this comprehensive journey, we aim to provide practitioners and researchers with valuable insights, equipping them to make informed decisions when navigating the intricate interplay of dimensionality reduction and classification performance in their data analysis endeavors.

2. Materials and methods

2.1. Dimensionality reduction methods

The DR methods are used to reduce the number of features in a dataset while retaining the most important information. This is particularly useful in high-dimensional datasets, where the number of features is much larger than the number of samples. There are several dimensionality reduction methods available, including linear methods and non-linear. The choice of dimensionality reduction method depends on the specific problem and the characteristics of the dataset, such as the number of samples, the number of features, and the type of data.

2.1.1 Principal Component Analysis

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique that extracts features and patterns from data by creating new uncorrelated variables, called principal components, which successively maximize the variance. PCA is an adaptive data analysis technique that does not rely on a priori assumptions about the data, but rather finds the principal components based on the dataset at hand.

Mathematically, PCA depends on the eigen-decomposition of positive semi-definite matrices and the singular value decomposition (SVD) of rectangular matrices.

The process of PCA involves finding the eigenvectors and eigenvalues of the covariance matrix or the correlation matrix of the data, and then selecting the eigenvectors corresponding to the largest eigenvalues to represent the principal components.

In summary, PCA is a mathematical objective function-based technique that extracts features and patterns from data by finding new orthogonal variables that maximize variance, and it is widely used in various domains for dimensionality reduction and pattern recognition.

2.1.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a popular technique for dimensionality reduction and feature extraction in machine learning and pattern classification applications. LDA finds a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier or for dimensionality reduction before later classification. LDA is also closely related to principal component analysis (PCA) and factor analysis, but it explicitly models the difference between the classes of data. LDA has been extended to solve problems such as Small Sample Size (SSS) and non-linearity, and novel feature extraction methods such as Robust Sparse Linear Discriminant Analysis (RSLDA) have been proposed to address the interpretability, sensitivity to noise, and selection of projection directions issues of classical LDA. LDA depends on the eigen-decomposition of positive semi-definite matrices and is usually computed using class-dependent or class-independent methods. The effectiveness of LDA can be evaluated using cross-validation techniques.

2.1.3 Kernel PCA

Kernel PCA (KPCA) is a variant of Principal Component Analysis (PCA) that uses a kernel function to map the input data into a high-dimensional feature space, where PCA is then performed. The kernel function is used to implicitly compute the dot product between the high-dimensional feature vectors, avoiding the explicit computation of the feature vectors themselves. This allows for the efficient computation of PCA in high-dimensional feature spaces, where the number of features may be much larger than the number of samples. Typical examples of kernel functions include the Gaussian RBF kernel and the Laplace kernel. Kernel PCA has been applied in various domains, such as chemical process monitoring and latent space exploration using generative models.

Principal Component Analysis only allows linear dimensionality reduction. However, if the data has more complicated structures which are nonlinear functions of the original features, standard PCA will fail in capturing meaningful information. Fortunately, kernel PCA allows us to generalize standard PCA to nonlinear dimensionality reduction 66.

The Vapnik-Chervonenkis theory shows that under certain circumstances mappings which take us into a higher dimensional space than the dimension of the input space often provide us with greater classification power 82. We consider a nonlinear mapping from the original m-dimensional input space to a M-dimensional feature space, where M ¿ m. However, high-dimensional mappings can seriously increase the computational cost.

Fortunately, we can make use of the kernel trick: given any algorithm that can be expressed solely in terms of dot products, this trick allows us to construct different nonlinear versions of it 76. With the kernel trick we can compute the inner product in a higher dimensional space, without the need of projecting the data itself. That the key idea behind kernel PCA. It allows us to extract up to n (number of samples) nonlinear principal components without expensive computations.

2.1.4 ISOMAP

ISOMAP (Isometric Feature Mapping) is a nonlinear dimensionality reduction technique that preserves the local geometry of the data. It is based on the idea of finding a low-dimensional representation of the data while maintaining the pairwise distances between data points. ISOMAP works by constructing a neighborhood graph, computing the geodesic distances between points, and then performing a multi-dimensional scaling (MDS) algorithm to find the low-dimensional representation. ISOMAP has been applied in various domains, such as image processing, bioinformatics, and fault diagnosis in electrical systems. The search results indicate that ISOMAP has been combined with other techniques to address specific challenges, such as noise reduction and feature extraction in big data problems.

2.1.5 Locally Linear Embedding

Locally Linear Embedding (LLE) is a nonlinear dimensionality reduction technique that preserves the local geometry of the data. LLE is an unsupervised learning algorithm that computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. Unlike clustering methods for local dimensionality reduction, LLE maps its inputs into a single global coordinate system of lower dimensionality, and its optimizations do not involve local minima. By exploiting the local symmetries of linear reconstructions, LLE is able to learn the global structure of nonlinear manifolds, such as those generated by images of faces or documents of text. LLE has been applied in various domains, such as image processing, bioinformatics, and fault diagnosis in electrical systems. The search results indicate that LLE has been combined with other techniques to address specific challenges, such as unsupervised feature selection and seed germination rate detection in maize.

2.1.6 Laplacian Eigenmaps

Laplacian Eigenmaps (LE) is a nonlinear dimensionality reduction technique that preserves the local geometry of the data. It is based on the graph Laplacian, which is a discrete approximation of the Laplace-Beltrami operator on a manifold. The algorithm constructs a representation for data sampled from a low-dimensional manifold embedded in a higher-dimensional space. Laplacian Eigenmaps has locality-preserving properties and a natural connection to clustering. Laplacian Eigenmaps has been applied in various domains, such as image processing, bioinformatics, and graph classification. The technique has been extended to contrastive learning, as seen in the Contrastive Laplacian Eigenmaps (COLES) paper, which combines Laplacian Eigenmaps with contrastive learning to preserve intrinsic and structural properties of a graph.

Additionally, Laplacian Eigenmaps has been used to enhance the loss function for graph classification, as seen in the Enhanced Loss Function based on Laplacian Eigenmaps for Graph Classification paper. In summary, Laplacian Eigenmaps is a geometrically motivated algorithm for representing high-dimensional data that lies on a low-dimensional manifold. It has locality-preserving properties and a natural connection to clustering, and it has been applied in various domains and extended to contrastive learning and graph classification.

2.1.7 t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a nonlinear dimensionality reduction technique commonly used for the visualization of high-dimensional data in scatter plots. It is particularly useful for visualizing complex, nonlinear relationships between data points. t-SNE works by modeling the pairwise similarities between data points in high-dimensional space and then mapping these similarities to a lower-dimensional space. The mapping is performed using a probabilistic approach that minimizes the divergence between the pairwise similarities in the high-dimensional space and those in the lower-dimensional space. In summary, t-SNE is a nonlinear dimensionality reduction technique commonly used for the visualization of high-dimensional data in scatter plots. Recent developments in t-SNE include techniques for improving the visualization, accelerating the computation, and representing the embeddings as decision tree structures.

2.2. Classification methods

Classification methods are techniques used to categorize or label objects based on certain criteria. These methods are widely implemented in various fields such as customer segmentation, fraud detection, computer vision, speech recognition, and medical diagnosis [27]. They involve analyzing

data or images to determine the category or class to which an object belongs. Some popular classification methods include the k-nearest neighbor algorithm, regression models, Bayesian networks, artificial neural networks, and decision trees [33]. These methods use different approaches to classify objects based on their features or characteristics. For example, one method involves processing sensor data to determine an element within a scene and classifying it using a machine learning model. Another method involves extracting features from an image and using a pre-trained classification model to determine the type of the object [30]. Overall, classification methods play a crucial role in various applications by enabling the automated categorization of objects based on their attributes or characteristics.

2.2.1 Perceptron

2.2.2 Multilayer Perceptron

Multilayer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected nodes, where each node is a simple processing unit that receives input from other nodes and produces an output. MLPs are commonly used for supervised learning tasks such as classification and regression. The search results provide information on various aspects of MLPs, including their applications, extensions, and optimization algorithms.

In summary, MLPs are a type of artificial neural network commonly used for supervised learning tasks such as classification and regression. The search results provide information on various aspects of MLPs, including their applications, extensions, and optimization algorithms.

2.2.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a simple and popular machine learning algorithm used for classification and regression tasks. The algorithm works by finding the K nearest data points to a given query point and assigning the query point to the class that is most common among its K nearest neighbors. The search results provide information on various aspects of KNN, including its applications, extensions, and optimization algorithms. In summary, K-Nearest Neighbors is a simple and popular machine learning algorithm used for classification and regression tasks. The search results provide information on various aspects of KNN, including its applications, extensions, and optimization algorithms.

2.2.4 Nearest Mean

Nearest Mean (NM) is a simple classification algorithm that works by computing the mean of each class and assigning a new data point to the class with the closest mean. The search results provide information on various aspects of NM, including its applications, extensions, and comparison with other classification algorithms. In summary, Nearest Mean is a simple classification algorithm that works by computing the mean of each class and assigning a new data point to the class with the closest mean. The search results provide information on various aspects of NM, including its applications, extensions, and comparison with other classification algorithms such as KNN and SVM. The extensions of NM, such as WNM and RNM, improve the classification performance and robustness of the algorithm.

2.2.5 Bayesian

Bayesian methods are a class of statistical techniques that are based on the concept of Bayes' theorem, which allows for the updating of probabilities based on new information. Bayesian methods are widely used in various fields, including machine learning, data analysis, and decision-making. These examples demonstrate the versatility and applicability of Bayesian methods in various domains, including business, engineering, and computer science. Bayesian methods are particularly useful when dealing with uncertainty and when prior knowledge is available, as they allow for the updating of probabilities based on new information.

2.2.6 SVM

Support Vector Machines (SVM) is a popular machine learning algorithm used for classification and regression tasks. The search results provide information on various aspects of SVM, including its applications, optimization algorithms, and comparison with other classification algorithms. In summary, SVM is a popular machine learning algorithm used for classification and regression tasks. The search results provide information on various aspects of SVM, including its applications, optimization algorithms, and comparison with other classification algorithms such as ANN. SVMs are particularly well-suited for large-scale learning tasks and have been applied in various domains, such as cancer genomics and stock forecasting.

2.2.7 K-Means

Performance Evaluation of Simple K-Means and Parallel K-Means Clustering Algorithms: The study evaluates the performance of Simple K-Means and Parallel K-Means clustering algorithms using different datasets. It highlights that the outcomes of the Simple K-Means clustering algorithms vary throughout multiple runs or iterations, while the outcomes of the Parallel K-Means clustering algorithms are consistent, leading to improved cluster quality, fewer executions, and faster execution times.

Cluster Analysis with K-Means versus K-Medoid in Financial Performance Evaluation: This research compares the results of K-Means and K-Medoid clustering methods in the context of financial performance evaluation. The study demonstrates that the choice of clustering method can influence the assessment of financial performance, as the two methods yield different results.

Analysis of Simple K-Means and Parallel K-Means Clustering for Software Products and Organizational Performance Using Education Sector Dataset: The study focuses on the application of Simple K-Means and Parallel K-Means clustering techniques in the educational sector to assess factors contributing to improving student academic performance. It emphasizes that the Parallel K-Means algorithms overcome the limitations of the Simple K-Means algorithm, leading to consistent outcomes and improved cluster quality, number of iterations, and elapsed time.

These findings underscore the importance of evaluating different clustering algorithms, such as K-Means and its variants, in specific application domains to understand their performance characteristics and suitability for the given tasks. The studies also highlight the potential of parallel clustering algorithms to address the limitations of traditional clustering methods.

2.2.8 GMM

GMM stands for Gaussian Mixture Model. It is a probabilistic model used in various applications such as seismic hazard assessment, deep-learning classifiers, production function estimation, and codebook construction for wireless communication systems. In seismic hazard assessment, a regional GMM is developed using local data to correct the misfit of global models and improve ground-motion predictions [36]. In deep-learning classifiers, GMM is used as part of a two-stage architecture to represent each class and enable incremental learning of new classes with a small set of annotated images. In production function estimation, a single-step GMM approach is proposed to estimate a production function with multiple inputs, productivity, and inefficiency. In wireless communication systems, GMM is used for codebook construction and feedback encoding, improving performance in configurations with reduced pilot overhead.

2.3. Datasets

2.3.1 Digits

The Optical Recognition of Handwritten Digits dataset [1] is widely used in research and experiments in the fields of machine learning and pattern recognition. This dataset consists of images of handwritten digits with ten classes, ranging from 0 to 9. Each image has a resolution of 8×8 pixels in grayscale, resulting in 64 features for each sample. The intensity of a pixel is represented by an integer in the range from 0 to 16. The Figure 1 illustrates a sample of images for each class.

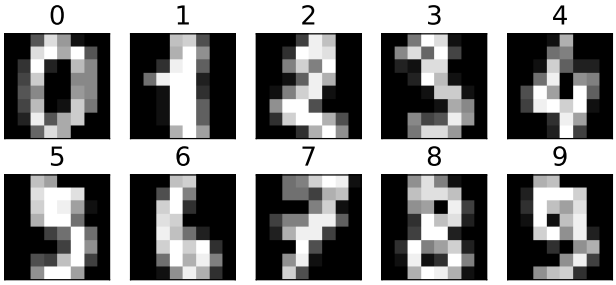

Figure 1. Sample from Digits dataset.

For this experiment, the test set was loaded from Scikit-Learn Python API [2]. The loaded set consists of 1797 samples, 10 classes, and approximately 180 samples per class. To handle the features, the images were loaded as feature vectors with a length of 64.

2.3.2 Mfeat-Pixels

Mfeat-Pixels dataset is available on an OpenML repository [3] and via Scikit-Learn API [2]. This dataset is part of a family of sets created from handwritten digits that were obtained from nine maps from a Dutch public utility [8]. The images consist of 2000 images in an 8-bit grayscale, with ten classes of numbers, from 0 to 9, as is shown in Figure 2, and with 200 samples per class. The set was preprocessed with sharpening and threshold, and normalized to fit into a dimension 30×48 pixels.

Figure 2. Samples of the pixel feature set. Extracted from [8].

Especially, the pixel feature set was made by dividing the image into 240 parts of 2×3 pixels, resulting in a tabular dataset, with 240 features. The values of features are integers and vary between 0 and 6.

2.4. Computing platform

The experiments were conducted on a computational system equipped with an AMD Ryzen 7 5700x processor and 64 GB of RAM. The choice of this hardware configuration aimed to provide sufficient computational power for the tasks involved in the study, since the datasets are made by thousands of samples, and there are a considerable number

of runs that combine each dimensionality reduction method with a series of classification algorithms.

The experiments were implemented using Jupyter Notebooks and the core functionality was implemented in Python scripts, encapsulating the methods for the experiments. All the Python scripts and notebooks are available in this public repository [4].

3. Experiments and results

For this experiment, we used Python libraries such as Scikit-Learn, to load the datasets, the methods for dimensionality reduction, the machine learning models, and the evaluation metrics. We also used Pandas for handling data in tabular form, NumPy for numerical operations, and Tqdm for displaying progress bars during training iterations.

The experiment consists of comparing the performance of different machine learning models for classification tasks over reduced and raw data. The reduced data is obtained by applying the already mentioned dimensionality reduction techniques over the raw data in order to obtain a reduced dataset with 20 dimensions. After, machine learning models are trained and evaluated using these embeddings. Each model is trained and evaluated 20 times, so we can obtain an average accuracy for supervised models and an average rand index for unsupervised models.

Specially for LDA and t-SNE, the maximum number of components couldn't be set to 20, because the number of components for the LDA is limited by the number of classes in dataset minus 1, so for this experiment, the maximum number of dimensions for LDA is 9, once the both datasets have 10 classes. On the other hand, the maximum number of components for t-SNE is 3.

The features data for LDA must be pertubed with a constant and small value, in order to make the matrix invertible. This is done by adding a small value to the diagonal of the matrix.

To evaluate the results, we summarize them by computing average accuracy or average rand index, grouped by the dimensionality reduction method. So that, it is possible to compare the performance obtained relative to the embedding method used. Finally, to visualize the embeddings, the data is reduced to 2 dimensions for each method and plotted using a scatter plot with different colors for each class.

To automate the whole process of reducing the dimensionality of the data, training multiple models, and evaluating their performance, we developed the Dimensionality Reduction class [4]. This class is designed to work with both supervised and unsupervised machine learning models, and it provides flexibility in choosing dimensionality reduction methods and run multiple training iterations for different models. The class also supports splitting data into training and testing sets, and optional feature scaling using Standard Scaler. All the transformations over the data, such as standardization and dimensionality reduction are done after splitting the data into training and testing sets, to avoid data leakage.

Besides, the class includes a method for training multiple machine learning models and evaluating their performance over multiple runs. For this, the method responsible for run multiple training takes as input a dictionary containing the models to be trained, the type of model (supervised or unsupervised), the dimensionality reduction method to be applied, if it's desired, and the number of training runs to perform for each model. The method returns a data frame containing the results of the training iterations, including the model name, average accuracy or average rand index, depending on the model type, and standard deviation.

3.0.1 Supervised classification

The reduction and training pipeline was applied to the the both datasets using supervised machine learning models. The results for digits dataset are shown in the Table 1, with the average accuracy obtained for each model when applied to the embeddings generated by the dimensionality reduction methods. Even with the raw data, the models achieved the high accuracy, in average. In this case, using raw data was the best option.

The best reduction method was Kernel PCA, followed by LDA and PCA, but the difference between them was not impactful. The manifold methods, only LLE and Isomap achieved a good performance, while t-SNE and LE were the worst methods.

In the case of Mfeat-Pixels, the supervised models got lower performance when trained with raw data, on average. This is explained by the test accuracy obtained by the Bayesian model. On the other hand, the Kernel PCA and PCA were the best methods, with close average accuracy. For the manifold methods, only Isomap and LLE performed well, but LE and t-SNE did not achieve good results, similar to the Digits dataset. The results are shown in the Table 2.

3.0.2 Unsupervised classification

The unsupervised models were trained and evaluated in a similar way as the supervised models. However, the evaluation metric used was the Adjusted Rand Index (ARI) [5].

In this case, the results for both datasets are quite similar. The best method was LDA, but differently from the previous experiment, the embeddings obtained by t-SNE contributed to increase the performance using unsupervised models. The average ARI for raw data suggests that the unsupervised had difficulty clustering the ten classes and high dimmension. The results are shown in the Table 3 for the Digits dataset and in the Table 4 for the Mfeat-Pixels dataset.

Model	Kernel PCA	LDA	PCA	LLE	Isomap	t-SNE	LE	Raw Data
Perceptron	0.907	0.903	0.884	0.864	0.874	0.134	0.048	0.912
Logistic Regression	0.932	0.930	0.931	0.940	0.943	0.144	0.098	0.958
MLP	0.953	0.938	0.954	0.956	0.944	0.087	0.077	0.964
KNN	0.953	0.945	0.953	0.949	0.930	0.120	0.192	0.957
Nearest Centroid	0.869	0.937	0.869	0.937	0.907	0.111	0.071	0.875
QDA	0.951	0.943	0.951	0.907	0.919	0.074	0.194	0.878
Linear SVM	0.960	0.937	0.959	0.922	0.941	0.111	0.098	0.976
RBF SVM	0.962	0.951	0.963	0.947	0.943	0.111	0.087	0.973
Average	0.936	0.935	0.933	0.928	0.925	0.112	0.108	0.937

Table 1. Results of average accuracy obtained by applying supervised machine learning models over raw data and embeddings obtained from dimensionality reduction methods on the Digits dataset.

Model	Kernel PCA	PCA	LLE	Isomap	LDA	LE	t-SNE	Raw Data
Perceptron	0.917	0.906	0.828	0.833	0.903	0.210	0.218	0.923
Logistic Regression	0.960	0.960	0.961	0.953	0.939	0.089	0.326	0.972
MLP	0.970	0.969	0.972	0.960	0.938	0.352	0.278	0.974
KNN	0.972	0.972	0.970	0.967	0.942	0.397	0.359	0.976
Nearest Centroid	0.933	0.933	0.966	0.962	0.940	0.425	0.390	0.935
QDA	0.977	0.977	0.950	0.956	0.947	0.102	0.288	0.241
Linear SVM	0.964	0.964	0.962	0.963	0.930	0.089	0.352	0.977
RBF SVM	0.980	0.980	0.974	0.969	0.947	0.369	0.318	0.980
Average	0.959	0.958	0.948	0.945	0.936	0.254	0.316	0.872

Table 2. Results of average accuracy obtained by applying supervised machine learning models over raw data and embeddings obtained from dimensionality reduction methods on Mfeat-Pixels dataset.

Model	LDA	t-SNE	Isomap	PCA	Kernel PCA	LE	LLE	Raw Data
KMeans	0.918	0.897	0.804	0.667	0.668	0.574	0.454	0.668
GMM	0.892	0.814	0.783	0.724	0.709	0.672	0.704	0.643
Average	0.905	0.856	0.793	0.695	0.688	0.623	0.579	0.656

Table 3. Results of average ARI obtained by applying unsupervised machine learning models over raw data and embeddings obtained from dimensionality reduction methods on Digits dataset.

3.0.3 2D Embeddings

In favor of visualize the embeddings generated by the DR methods, both datasets were reduced to 2D and plotted using a scatter plots to facilitate the visualization of the cloud of points for each class. In the Figures 3 and 4 are shown the scatter plots for the Digits and Mfeat-Pixels datasets, respectively.

It's possible to note that the behavior of the classes in the reduced space is different for each method but equivalent for both datasets. For example, PCA and KPC have a sparse and similar scattering for the classes, but with overlapping points. Nonetheless, LDA demonstrated a capacity to maximize the separation inter-classes [7], forming clouds of points more defined.

LLE created a massive colinear cloud of points, and it is more evident in the embeddings for Digits. This is coherent with the main assumption, that in densely sampled regions, vectors and their neighbors form linear patches [6], belonging to an Euclidean subspace. This allows LLE to characterize local geometry through linear coefficients, aiding in the preservation of local relationships and capturing intrinsic manifold structures in high-dimensional data.

The t-SNE and Isomap embeddings emphasize the separation of the classes, but maintain the points of the same class close to each other. This property was well exploited by the unsupervised models, which achieved better performance using these embeddings than supervised models.

Model	LDA	t-SNE	Isomap	LLE	Kernel PCA	PCA	LE	Raw Data
KMeans	0.953	0.903	0.854	0.723	0.646	0.609	0.636	0.643
GMM	0.910	0.872	0.777	0.795	0.689	0.651	0.614	0.600
Average	0.931	0.887	0.816	0.759	0.668	0.630	0.625	0.622

Table 4. Results of average ARI obtained by applying unsupervised machine learning models over raw data and embeddings obtained from dimensionality reduction methods on Mfeat-Pixels dataset.

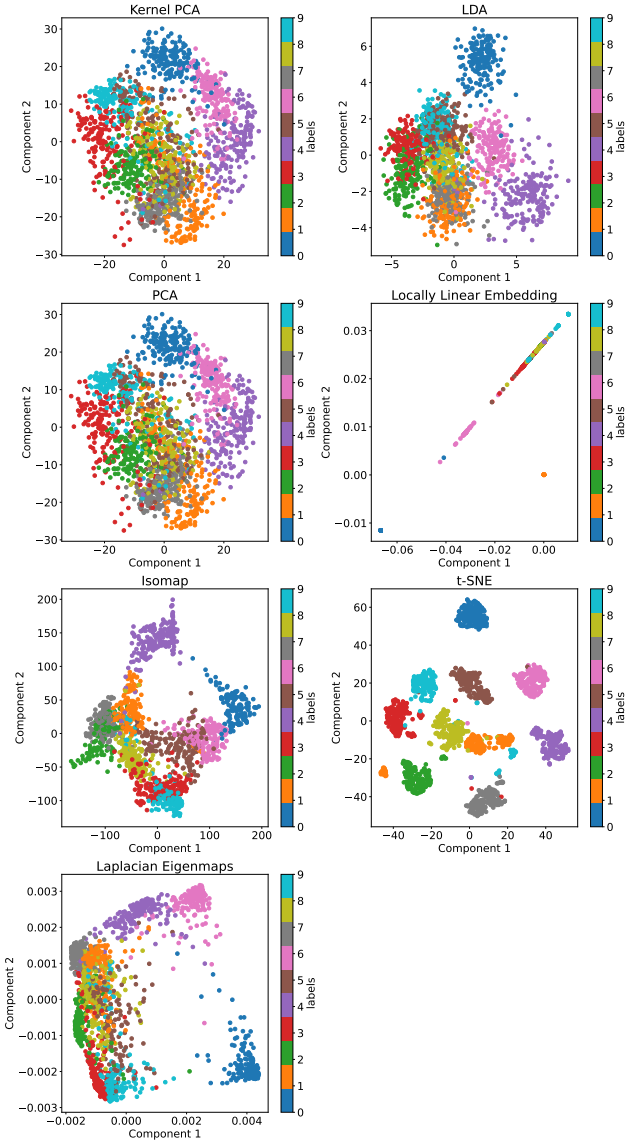

Figure 3. 2D Scatter plots of embeddings extracted by dimensionality reduction methods applied over Digits Dataset

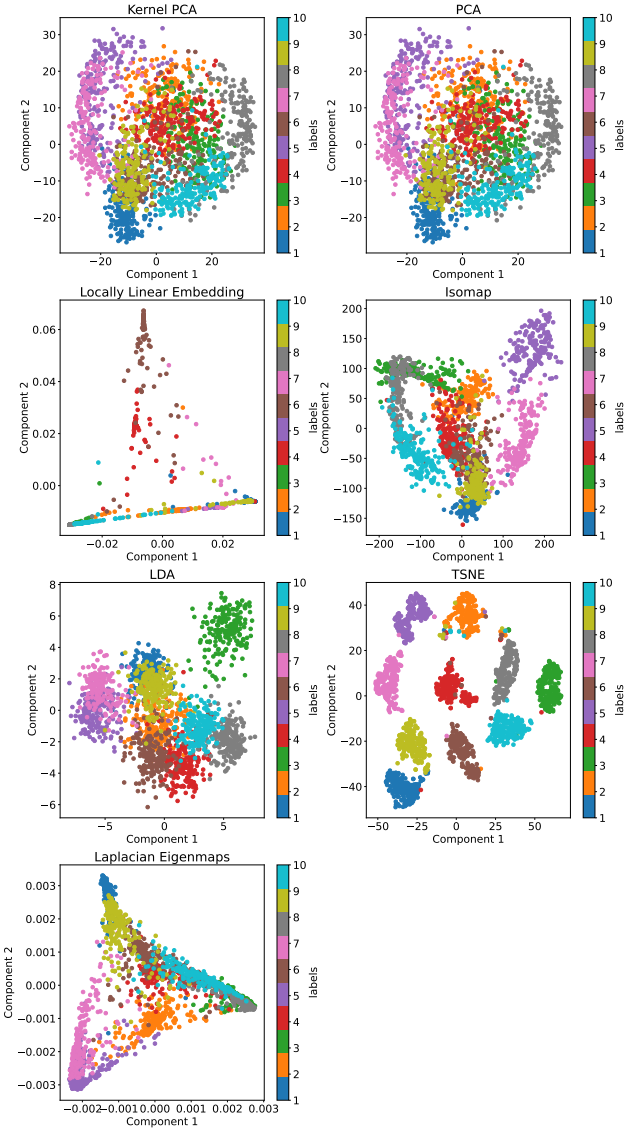

Figure 4. 2D Scatter plots of embeddings extracted by dimensionality reduction methods applied over Mfeat Pixel Dataset

4. Conclusions

Dimensionality reduction methods can be combined with machine learning algorithms to improve the performance of the models, and they are good tools for extracting features from high-dimensional data and visualizing it in a lower-dimensional space. To make the process easier, it was necessary to develop a specialized code that automates the whole process, aiming to make it easier to compare the performance of different models and dimensionality reduction

methods.

The results obtained in this experiment showed that the performance of the models was influenced by the dimensionality reduction method used. In other words, the embeddings generated by the DR methods

KPCA and PCA behaved similarly, but KPCA outperformed the rest when its embeddings were used to train supervised models, while LE and t-SNE were the worst. However, for unsupervised models,LDA, t-SNE and Isomap got the best results, and this is coherent with the visual representation of the embedding, which showed a good separation of the classes, contributing to the clustering of the data.

One of the most challenging tasks was tuning the parameters of the DR methods and machine learning models, in order to obtain the best accuracy or ARI. In terms of computation time, t-SNE was the most expensive method, spending 4 minutes per model, in average, while PCA was the fastest method, taking less than 10 seconds per model.

Finally, it was evidenced that DR techniques are powerful and can be used in a wide range of applications and data, and this research field is still relevant for improvements and new discoveries.

References

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

[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[3] Robert Duin. Multiple Features. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5HC70.

[4] Wesley N. Galvão. Dimensionality reduction experiments repository. `https://github.com/galvaowesley/dimensionality_reduction/tree/main`.

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

[6] Alexandre Levada. A review on dimensionality reduction for unsupervised metric learning: From linear spaces to manifolds. 2023.

[7] Alexandre Luis Magalhães Levada. Introdução ao reconhecimento de padrões.

[8] Martijn van Breukelen, Robert PW Duin, David MJ Tax, and JE Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.