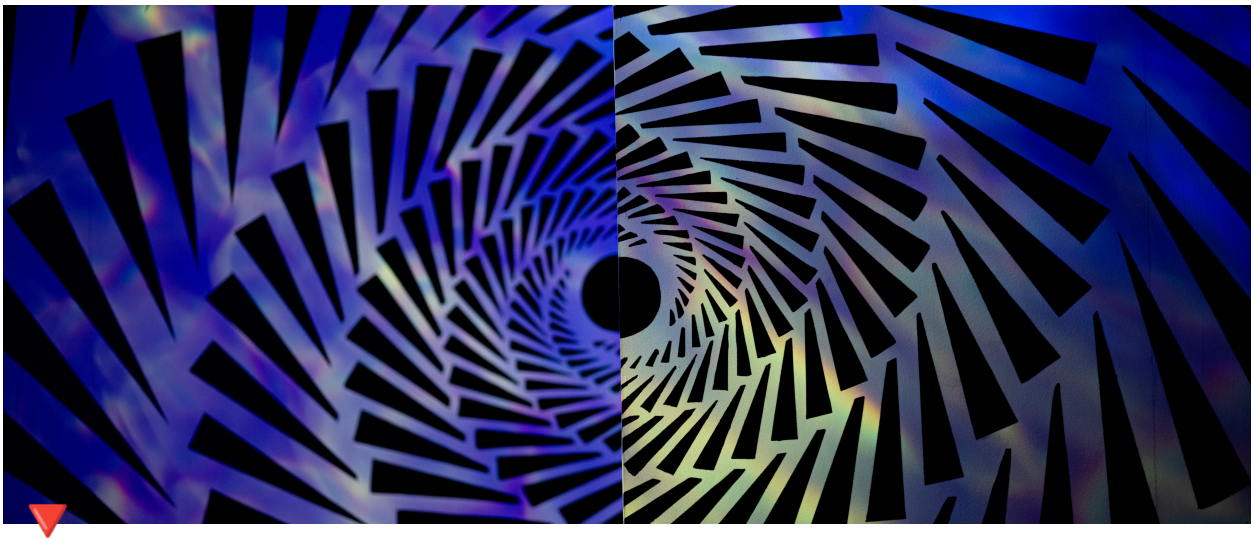


Revealing Dimensional Reduction in Data Mining

Type of CW	Take Home Assignments (THA)
Deadline	@April 28, 2024
Lecturer	ProgrammerRDAI
Week	8

Glossary Terms

Multicollinearity

- Multicollinearity is a statistical phenomenon when a regression model strongly correlates with two or more independent variables (Bhandari, 2024).

Manifold learning (ML)

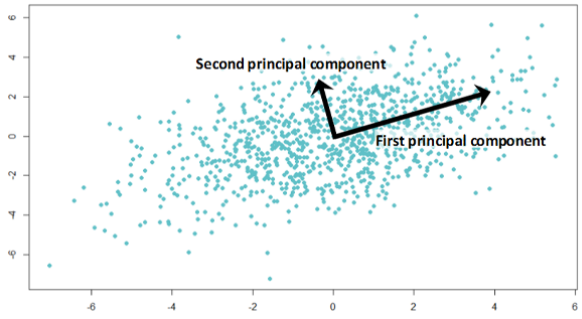
- Manifold learning (ML) is a group of techniques for determining the low-dimensional structure of data. It is called manifold learning (ML), sometimes called non-linear dimension reduction (Meilă and Zhang, 2023).

Intrinsic Dimensionality Estimation

- Intrinsic Dimensionality Estimation: Based on this hypothesis, global estimators of intrinsic dimension (ID) return a single estimated dimension for the entire dataset. The term intrinsic dimension (ID) intuitively refers to the minimum number of features required to represent a dataset with little information loss (scikit-dimension: intrinsic dimension estimation in Python; scikit-dimension 0.3.3 documentation, no date).

Orthogonal

- Orthogonal: The image, which is based on simulated data with two predictors, shows the directions of the components, which are orthogonal as expected and suggest that there is no correlation between the two components if they are uncorrelated (Avcontentteam, 2024).



Linearly Separable

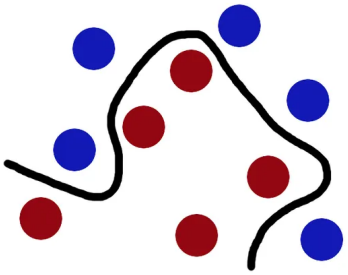
- Linearly Separable: In binary classification problems, we have two classes: positive and negative. We say the classes are separable if there is a classifier whose decision boundary divides the positive objects from the negative ones. This idea applies to binary classification problems (Simic and Simic, 2024).

Support Vector Machines

- Support Vector Machines: Establish boundaries between data points using outputs, labels, or predefined classes as a guide (Kanade, 2022a).

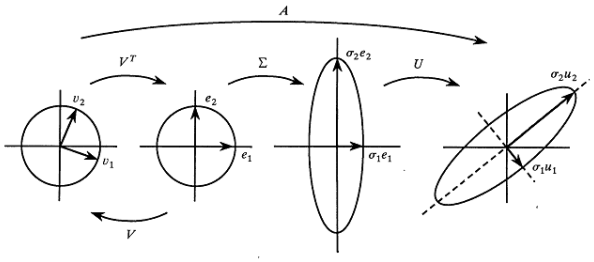
Kernel trick

- Kernel trick: The term "kernel" typically refers to the kernel trick, which is a technique for applying a linear classifier to the solution of a non-linear problem (Afonja, 2018).



Singular Value Decomposition (SVD)

Singular value decomposition is one of the most important matrix factorizations, providing a numerically stable matrix decomposition that is used for a variety of purposes. SVD decomposes high-dimensional data into its most statistically descriptive factors, in turn reducing residual data while retaining the most important data. SVD advantages: allows for numerically stable computation that produces a hierarchical representation of the data and helps make sense of data with a lot of noise (by identifying a relevant subspace of all the dimensions created with noisy data). SVD is a powerful tool in the space of data; it goes beyond the dimensionality reduction space (Admin, 2021; GeeksforGeeks, 2023g; Vincent, 2022; Khadka and Khadka, 2023)



Basic Visualization of SVD in a High-Level Mathematical Perspective (Wicklin, 2019).

Introduction

Escalating Challenges of High-Dimensional Datasets

In the era of big data, the curse of dimensionality is reoccurring due to the volume and density of datasets in modern times, which raises the data's complexity without actually adding more valuable information. This influx of dimensions causes challenges such as data sparsity, computational scalability, visualization challenges, and challenges for machine learning applications. As dataset dimensions grow, traditional approaches to using data have become ineffective over time, and it calls for the adoption of advanced techniques such as dimension reduction to extract insights from high-dimensional datasets. (Awan, 2023)

The Significance of Dimensional Reduction

Dimensional reduction is used as the go-to solution for high-dimensional datasets. It transforms high-dimensional datasets into lower dimensions, ensuring the accuracy and effectiveness of the pre-processing. The production of low dimensions, such as 2D or 3D, allows for easier visualizations while enabling more effective storage and speeding calculations, especially in machine learning dimensional reduction provides fewer computation times, removes multi-

collinearity-enhancing machine learning model’s parameter interpretation, and results in higher model performance due to high-dimensional datasets showing poor generalization (Kanade, 2022; Simplilearn, 2023; Abhigyan, 2021; What is Dimensionality Reduction? | IBM, no date)

Understanding Dimensionality Reduction

Dimensionality reduction techniques play a crucial role in pruning dimensionality in datasets while keeping essential information. In this section, we discuss core concepts such as manifold learning and intrinsic density estimation and their implications for data analysis.

Definition

Dimensionality reduction is where millions of features in data, analyzed using numerous sources and calculations, are transformed into a lower-dimensional dataset without changing the essential characteristics. This is done by combining multiple characteristics that have a high correlation with each other. (Kanade, 2022b).

Objectives

The main objective of dimension reduction is to reduce dimension in a dataset, allowing for enhanced data visualization, attaining computational effectiveness, and mitigating the curse of dimensionality. By reducing the number of features with dimensionality reduction techniques, they retain the majority of the pertinent data information, which solves the challenges of high-dimensional datasets. (Abhigyan, 2021b; Shrivastava, 2022)

Types of Dimensionality Reduction

Feature Selection

- Following Occam’s Razor’s Law of Parsimony, which states the solution that makes the fewest assumptions possible is the best explanation for a given solution, feature selection has employed this and strengthened data analysis by identifying and keeping the most crucial characteristics, in turn reducing unnecessary assumptions and computation costs while producing more accurate and effective outcomes. (Gupta, 2023; Heavy.AI, no date).

Feature Extraction

- Feature extraction is the process of converting unprocessed data (such as text or images) into numerical features that can be handled while retaining information from the original datasets. It minimizes the amount of resources required for processing without losing crucial data. The information that is extracted from the raw data is then retained in new features. In essence, feature extraction allows for streamlined data processing, transforming high-dimensional data into new features. (Feature extraction explained, no date; DeepAI, 2020; What is feature extraction?, Feature Extraction Techniques Explained, no date)

Differences between Feature Selection and Extraction

- (GeeksforGeeks, 2023a; Sinha, 2023b; How does feature selection benefit machine learning tasks?, no date)

Feature Selection	Feature Extraction
Choosing a subset of the original pool of features.	Getting useful features from existing data and creating new features
Used when original features are interpretable and meaningful	Used when original features are noisy, redundant or irrelevant
A small difference between the number of features and	There is a big difference between the number of

Feature Selection	Feature Extraction
the sample	features and the sample
Features are independent and have low correlations with each other	Feature have complex non-linear relationships

Challenges

Data Sparsity

Data sparsity occurs in large, high-dimensional datasets where a significant amount of the data is missing or set to zero. This causes challenges such as information loss, the need for specialized algorithms to handle sparse data, and problems when clustering similar features. (R, 2022; Kanade, 2022d; Dremio, 2024)

Month/ year	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable
Jan-21	0	0	0	0	0	0
Feb-21	0	6	1	6	0	0
Mar-21	0	0	0	0	0	0
Apr-21	25	0	5	0	0	0
May-21	0	6	0	0	0	0
Jun-21	0	0	0	0	0	0
Jul-21	0	0	0	0	0	0
Aug-21	0	0	0	0	0	0
Sep-21	0	0	0	0	0	0
Oct-21	45	5	4	8	9	0
Nov-21	6	1	2	0	0	0
Dec-21	0	0	0	0	0	0

(Science and Science, 2024)

Computational Scalability

If the dimensionality of datasets increases, the computational complexity required for dimensionality reduction methods also increases. In turn, scalable, efficient computational resources are crucial and a challenge when dealing with large-scale datasets. (*Big data and computational scalability*, no date; Awan, 2023b)

Visualization Challenges

With high-dimensional data, it has become difficult to visualize, and exploratory data analysis is more challenging. Data must be reduced to forms such as 1D, 2D, or 3D with the use of dimensionality reduction to make visualization understandable for humans. (Probst and Reymond, 2020; Awan, 2023b)

Challenges For Machine Learning Applications

- High Training Resources: High-dimensional datasets require a significant amount of computing power to store, process and analyze, which can be costly due to the frequently sluggish learning process connected to a complex hierarchy of learning data abstractions (Mori, 2023; High-Dimensional Data: Challenges and Strategies for Analysis, no date)
- Low Model Performance: Model performance is affected in machine learning applications with high-dimensional datasets. It makes it more challenging to understand relationships and patterns in the data, making the model less effective, and the results from the model can be less robust and reliable. (High-Dimensional Data: Challenges and Strategies for Analysis, no date)
- Overfitting: With high-dimensional datasets used in machine learning training, models suffer from poor generalization of new data due to overfitting on the training set. At the same time, the model grows too complex and only learns the underlying noise of the training data. This issue is extremely common in high-dimensional datasets, where the dataset contains a lot of features (GeeksforGeeks, 2024; Vaj, 2023; High-Dimensional Data: Challenges and Strategies for Analysis, no date)

Real-world Impact

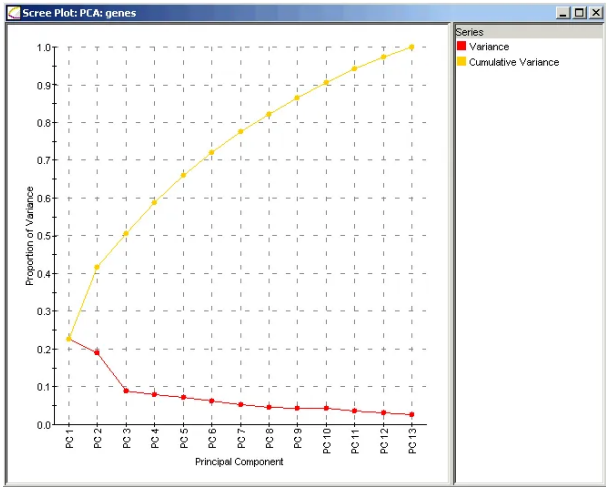
- Dimensionality reduction impacts real-world case studies across many domains, such as image recognition and natural language processing. By allowing for more efficient data analysis

pipelines to enhance model performance, density reduction allows businesses to extract valuable insight from a variety of applications (Kanade, 2022b).

Comparative Analysis of Dimensionality Reduction Methods

Principal Component Analysis (PCA)

Principal component analysis (PCA) is a powerful linear feature extraction method in data analysis in the domains of statistics and data science that is used in data analysis to reduce dimensionality in data while preserving variance as much as possible. The PCA process involves multiple mathematical steps where mathematical principles such as converging matrixes, eigenvectors, and eigenvalues are used. The aim is to maximize variance so that features are as dependent on each other as possible. A situation where PCA might not be beneficial is where multicollinearity exists within variables. The produced principle components are independent of each other since they are orthogonal to each other. (StatQuest with Josh Starmer, 2018; StatQuest with Josh Starmer, 2017; Nair, 2022; Brems, 2022b).



(*Creating a scree plot*, no date)

A scree plot is used to determine how many PCs should be kept according to the variance that needs to be kept in the produced PCA (Brems, 2022b).

Incremental PCA

Incremental PCA is a variation on conventional principal component analysis; it is used in circumstances where the data set is too large to fit into memory, it partially fits mini-batches and it updates the principal components. It is mostly used in similar applications listed: streamed data processing, large-scale machine learning tasks and real-time analytics. Incremental PCA causes longer computational times as well as the possibility of providing less precise outcomes than conventional PCA (GeeksforGeeks, 2023d; N, 2024).

Kernel PCA

Kernel PCA is a type of PCA that is made to handle non-linear data. It uses a kernel trick (kernel methods) to transform the data into a dimensional space where the data is easily linearly separable. A variety of kernel methods can be used, such as linear, polynomial, and Gaussian, to project non-linear data into a higher-dimensional space and then linearly separate them. The concept is similar to that of support vector machines. KPCA is useful, especially in tasks such as image or speech recognition where nonlinear relationships are

$$C = \frac{1}{\ell}\sum_{j=1}^{\ell}\mathbf{x}_j\mathbf{x}_j^\top.$$

$$\bar{C} = \frac{1}{\ell}\sum_{j=1}^{\ell}\Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^\top,$$

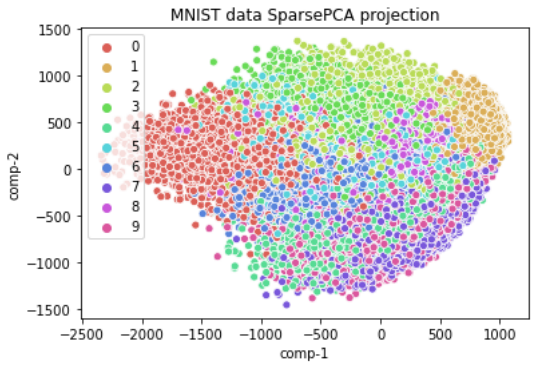
From a mathematical perspective, the first image attached is the basic transformation, which is used to determine an estimate of the covariance matrix of the data. The second image

in the input features. The problem with using KPAC is the difficulty and time consumed in finding the correct kernel function/method, and at the same time, KPAC is extremely computationally expensive. Overall, KPCA is an extremely useful dimension reduction method because it solves some of the core issues with the original PCA (GeeksforGeeks, 2023d; GeeksforGeeks, 2023e; Zvornicanin and Zvornicanin, 2024b).

shows the process but with a kernel function added into the mix when calculating the covariance matrix of the data. This is the biggest mathematical difference between kernel PCA and PCA. After the covariance matrix is produced, the next steps in calculating kernel PCA are identical to PCA. (Zvornicanin and Zvornicanin, 2024b; Afonja, 2018)

Sparse PCA

Sparse PCA is a feature selection method and a variation of PCA that encourages finding a lower-dimensional representation of the data with fewer non-zero components with the use of a sparsity constraint to PCA (essentially constraining the number of features that are used in PCA with the use of a sparsity constraint) by setting all other loadings to zero and identifying a select few that are the most important original variables, which are controlled using a hyperparameter (such as `alpha`). In a simple explanation, PCA is performed in a specific sparse set (a subset of variables with non-zero loading) that is produced. Sparse PCA produces a more interpretable set of principle components while reducing noise in the data and producing more accurate and robust results. As well as Sparse PCA's pros, it has its cons, such as It requires a lot of fine-tuning to produce the best overall result, in turn taking a lot of resources and time, and sparse PCA is a complex optimization process, in turn requiring more computational complexity (N, 2024c; GeeksforGeeks, 2023d; IQmates, 2020)

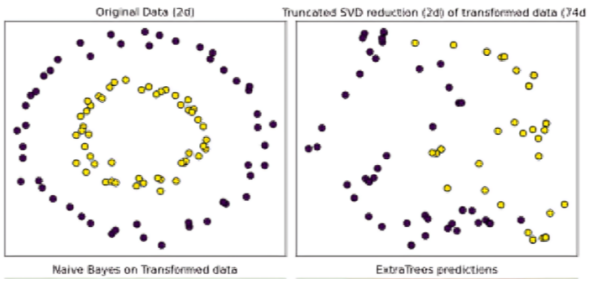


(*SparsePCA Projection example in Python*, 2021)

The above is a simple visualization of the MNIST dataset after being processed with SparsePCA.

Truncated SVD

Truncated Singular Value Decomposition (SVD) is a "naive" algorithm that is used for the linear dimension reduction technique. TruncatedSVD is extremely similar to PCA, but its differences come from the fact that TruncatedSVD doesn't centre the data before single value decomposition and SVD directory operators on the data matrix instead of a covariance matrix. Truncated SVD is also fairly different from the original SVD, where Truncated SVD chooses the top k singular values, a reduced yet informatively rich representation of the

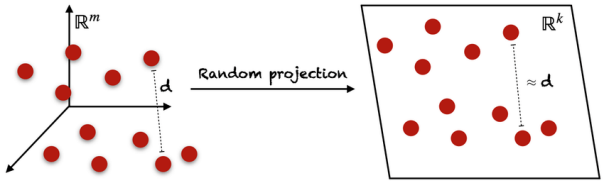


original data. (For example, in an n × n matrix, SVD would produce several columns but truncated SVD will produce matrices with a specific number of numbers.). Truncated SVD plays a major role in data analysis and machine learning applications; it helps reduce the impact of noise or redundancy in data and offers a blend of computational efficiency and information retention. A case where truncated SVD won't work efficiently would be when the relationships between features are too complex. (Khadka and Khadka, 2023; sklearn. decomposition). (TruncatedSVD, no date; Garreta *et al.*, no date)

Comparison between original data and TruncatedSVD transformed data. (learn. decomposition.TruncatedSVD, no date)

Gaussian Random Projection

The Gaussian Random Projection is a specific method of random projections, an alternative, less computationally expensive computational tool for dimensionality reduction that follows the Johnson-Lindenstrauss lemma (which states that high-dimensional data points can be projected into lower-dimensional space while approximately preserving pairwise distances or, in simple words, preserving the original data structure as much as possible in the dimensionality reduction process). Gaussian random projection is used when methods such as principal component analysis cannot provide adequate results for the given data. It is one of the most widely used random project techniques while being easy to implement at the same time. The difference between a normal random project and this is that the random project matrix is made using a Gaussian distribution. Other similar random projection methods are Sparse Random projection and Johnson-Lindenstrauss Random Project. Overall, the Gaussian Random Project is a general and efficient dimension reduction method while following the Johnson-Lindenstrauss lemma (Yufeng, 2022: *How to implement Random Projection using Python Scikit-learn?;* no date; Riswanto, 2023; M. Gupta, 2023; R3d_Robot, 2022)



M = Number of Samples

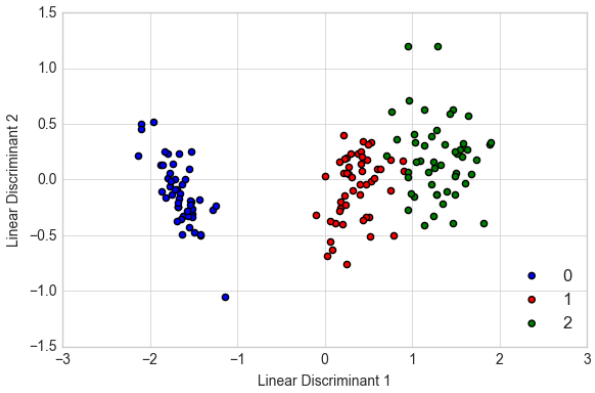
N = Number of Feautres (Original)

D = Number of Features (Reduced)

First is the input data matrix `K` with the size of `MxN`. Then the random project matrix `R` with the size of `NxD` is initialized using a normal distribution with a mean of 0 and a standard deviation of `1/sqrt(d)`. This is done to ensure that `R`'s columns have unit length after normalization. Finally, the projection operation occurs `J=KXR` with a final output dimension of `MXD` (M. Gupta, 2023).

Linear Discriminant Analysis

Linear discriminant analysis (LDA), also known as normal discriminant analysis (NDA) or discriminant function analysis (DFA), is one of the most popular techniques for linear supervised dimensionality reduction. It is a classification dimensionality reduction method, in turn requiring at least 2 classes in the data. Linear discriminant analysis (LDA) aims to create a new axis so that it maximizes the distance between the means of the classes while minimizing the variation within each class and allowing for easy transformations from 2D and 3D into a 1-dimensional plane. For example, if we have 3 classes and 18 features, LDA will produce a 2-feature output from the entire 18 features (c-1, c being the number of classes). Linear discriminant analysis (LDA) is similar to PCA. Still, in comparison, PCA only aims to produce principle components without any sort of context or account of the data. In contrast, LDA takes into consideration the data's classes, in turn producing a more efficient and better dimension reduction for classification. Linear Discriminant Analysis is based on Fisher's linear discriminant statistical method, originally developed by Sir Ronald Fisher in the 1930s and later simplified by C.R. Rao as a multi-class version. Fisher originally used the discriminant function to classify between two plant species, Iris setosa and Iris versicolor. Furthermore, LDA aims to reduce resources and dimensionality costs while increasing computational efficiency. There are many versions of discriminate analysis, such as quadratic discriminant analysis (QDA), flexible discriminant analysis (FDA), and linear discriminant *analysis (LDA) in machine learning (JavatPoint*, no date; V, 2024; *What is linear discriminant analysis? | IBM*, no date; Zhou, 2023; Meigarom, 2022).



(Raschka, no date)

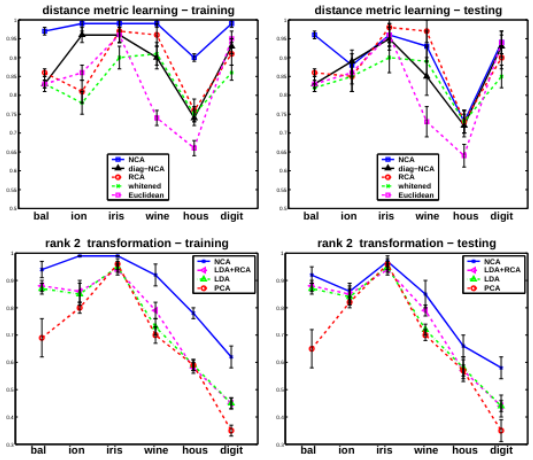
LDA finds the optimal linear transformation that maximizes the separation between classes by computing scatter matrices, eigenvalues, and eigenvectors and finally projecting the data into a new feature space. (V, 2024b)

Neighborhood Components Analysis

Neighborhood Components Analysis (NCA) is a linear supervised non-parametric dimensionality reduction that is also used as a classification model. It doesn't consider any assumptions about the shape of the class distribution or the boundaries between them while allowing for the data to maximize the k nearest neighbours' performance. It allows for reduced storage sizes and search times for kNN algorithms; this is achieved by directly maximizing a stochastic

Comparison of NCA with PCA and LDA in terms of visualization (Goldberger et al., n.d.)

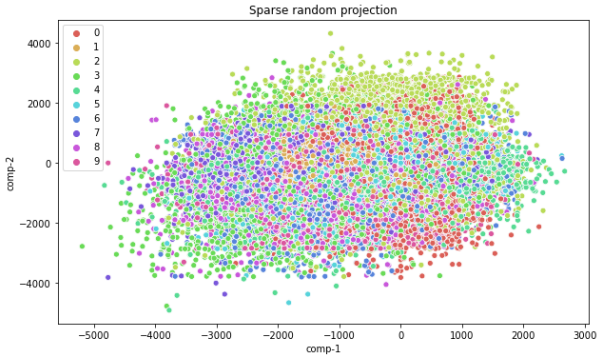
variant of the leave-one-out kNN score on the training set. The main goal of NCA is to maximize the prediction accuracy of regression and classification algorithms. With labelled data, specifically, NCCA performs better in classification problems because it learns the low-dimensional linear embedding of the labelled data, which is utilized for data visualization and fast classification. ('kNN classification using neighbourhood Components Analysis,' 2020; *Neighborhood Component Analysis (NCA) feature selection: MATLAB & Simulink*, no date; *neighbourhood Components Analysis, state wiki*, no date; Goldberger et al., n.d.)



Sparse Random Projection

Sparse Random Projection (SRP) is a dimensionality reduction method, an algorithm used to estimate distances between pairs of points in high-dimensional vector spaces, that is used for stable scenarios where computing all pairwise distances between data points is computationally expensive due to the large dimensionality of the data (data matrix `A`). The issue is solved by multiplying a data matrix `A` of size `nxD` by a random matrix `R` of size `Dxk`, where `k` is smaller than `D`. This method has been used in many fields, including machine learning, VLSI layout, latent semantic indexing, and bioinformatics. SRP is seen as an alternative to the dense random projection matrix, which produces similar embedding quality, but SRP is appealing due to its simplicity, effectiveness, and efficiency (memory), which in turn has made it a popular choice throughout many real-world applications, especially in applications that require image or text processing. (Zou, Li and Dai, 2013; *Random Projection in Python, by Yufeng, in Towards Data Science, Freedium*, no date; *Dimensionality Reduction with Sparse, Gaussian Random Projection and PCA in Python*, 2020; *sklearn.random_projection.SparseRandomProjection*, no date)

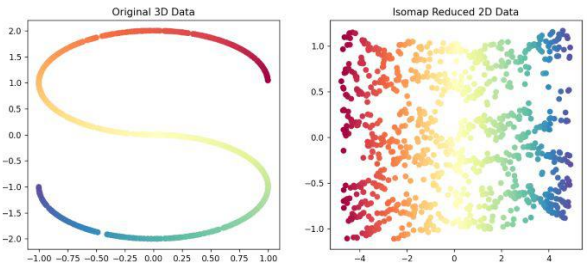
SRP, when compared to Gaussian random projection, requires fewer computational resources due to 2 main reasons: one is that it only requires integer arithmetic; the other is that a sparse projection matrix contains fewer non-zeroes.



(*Dimensionality Reduction with Sparse, Gaussian Random Projection and PCA in Python*, 2020)

Iso Map

Iso Map, also known as Isometric Mapping, is a type of manifold learning that utilizes a non-linear dimensionality reduction technique in machine learning and data analysis. Developed as an alternative to methods such as principal component analysis to preserve the intrinsic geometry of the original high-dimensional data, which is done by preserving the pairwise distances between data points, in turn producing a low-dimensional representation of the data, usually in two- or three-dimensional space. The iso map highlights the fundamental relationships found in the data before density reduction while taking into consideration niche possibilities such as the underlying structure being broken or folded. Iso Map is used for applications such as speech recognition, image analysis, biological systems, and natural language processing. Its advantages are its ability to handle non-linear structures, preserve local and global structures, and versatility across applications. Its cons are that it is computationally expensive and sensitive to noise and outlines in the data. (GeeksforGeeks, 2024a; *Sklearn. manifold.iso map*, no date)



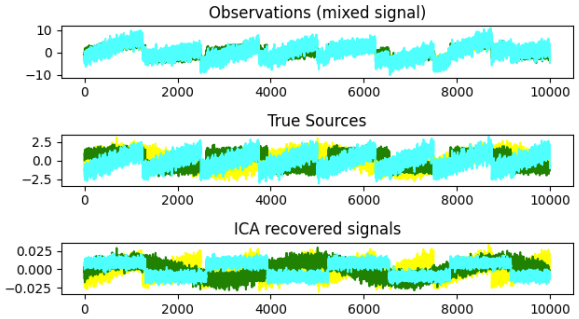
(GeeksforGeeks, 2024a)

Mini-Batch Dictionary Learning

Mini Batch Dictionary Learning is a variation of the dictionary learning algorithm that helps learn signals of interest in data forms such as images, videos, time series simulations, and experiment data used to extract meaningful features from high-dimensional data. It allows for the data to be split into smaller subsets, making it more effective on large datasets. The subsets are iteratively provided to the algorithm and the algorithm is updated in every increment. Furthermore, Mini Batch Dictionary Learning allows for sped-up computation and efficiently learns a compact and informative presentation of the data. (Archibald and Tran, 2022; Deelaka, 2024; *sklearn. decomposition.MiniBatchDictionaryLearning*, no date)

Fast ICA

Fact ICA is a fast algorithm that is a popular variation of Independent Covariance Analysis, which searches for mutually independent non-gaussian latent variables when components of the multivariate data are assumed to be linear combinations of them (transform multivariate datasets so that the produced components are as reliable as possible). Many real-life events use Fact ICA, such as audio source separation, medical tests, and communication applications, where individual signals are separated from a

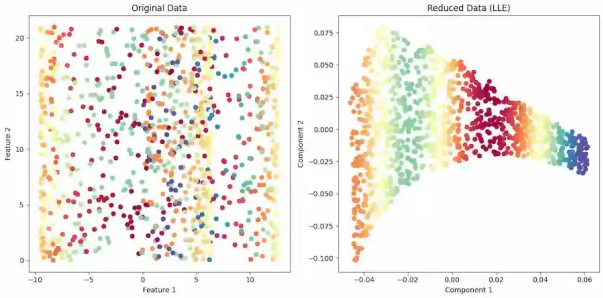


mixed signal. There are mainly two types of FactICA, which are: deflation-based FastICA, where the components are found sequentially; and symmetric FastICA, where the components are found simultaneously. (scikit-learn.org, n.d.; GeeksforGeeks, 2023; Miettinen, Nordhausen and Taskinen, 2019; Hyvärinen, 1999)

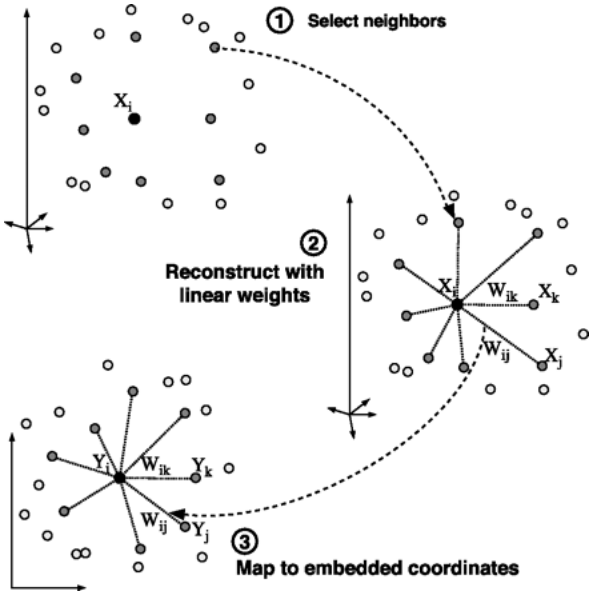
Example of Fast ICA (Geeks for Geeks, 2023)

Locally Linear Embedding

Locally Linear Embedding (LLE) is an unsupervised method used for dimension reduction introduced by Sam T. Roweis and Lawrence K. Saul in 2000. It retains essential geometric characteristics of the underlying nonlinear feature structure, in turn surpassing traditional density modelling techniques such as PCA. The high-level steps in LEE are: Constructing a nearest neighbour graph to capture the local relationships, Optimizes weight values for each data point (aiming to minimise recognition error); this weight reflects the strength of the connections between points, Computes lower-dimensional representation by finding eigenvectors for a matrix which is derived from the weighted matrix. The advantages provided by using locally linear embedding are Preservation of the local structures, non-linearity and computational efficiency; the disadvantages, on the other hand, are: Memory and computation requirements, Outliers and noisy data may reduce performance. (Roweis, 2000; GeeksforGeeks, 2023)



(Roweis, 2000)



(Mihir, 2024)

Understanding the Provided Python Code

Data Loading and Preprocessing

Imports

- First, the required packages are imported. The following are the general packages that are important: numpy (`np`), and matplotlib.pyplot (`plt`), pandas (`pd`), sklearn.datasets, sklearn.model_selection.train_test_split
- The following packages are dimensionality reduction-related modules imported: `PCA, IncrementalPCA, KernelPCA, TruncatedSVD, FastICA, MiniBatchDictionaryLearning, SparsePCA, Isomap, LocallyLinearEmbedding, LinearDiscriminantAnalysis, GuassianRandomProjection, SparseRandomProjection, NeighbourhoodComponentAnalysis`, and `KNeighborhoodClassifier` are used as the models that the density-reduced data is tested out on to see their effectiveness; `make_pipeline` is used to create a pipeline with the dimensionality reduction modules so that they can be used after one another; `StandardScalar` is used as a basic data preprocessing step throughout the project.

Loading Dataset

- The MNIST dataset is loaded using `datasets.load_digits()`, then split in `X` (input image data) and `y` (targets/labels)
- Parameters Initialization
  - Dimensionality (`dim`) is taken by getting the length of the first row of `X`
  - Then the number of classes (`n_classes`) is calculated by getting the length of the number of unique `y` elements.
  - The `n_neighbours` variable is set, which is used later on in the notebook for the KNN model initialization
  - `random_state` is used so that the produced outcomes are easily reproducible
  - `X_train, X_test, y_train, and y_test` are produced by passing `X` and `y` into the `train_test_split` function with the parameters `test_size`, which is the test size split of the entire dataset; `random_state` is also passed as a parameter to ensure reproducibility.

```
# Load Digits dataset
digits = datasets.load_digits()
X, y = digits.data, digits.target
# Parameters
dim = len(X[0])
n_classes = len(np.unique(y))
n_neighbors = 3
random_state = 0
# Split into train/test
X_train, X_test, y_train, y_test = \
train_test_split(X, y, test_size=0.5, stratify=y,
random_state=random_state)
```

Preprocessing for Dimensionality Reduction

- Discussion on the importance of data preprocessing for ensuring effective dimensionality reduction.
- To ensure effective dimensionality reduction, the notebook has used the

```
pipeline = make_pipeline(StandardScaler(), "DIMENSIONALITY_REDUCTION_METHOD")
```

`StandardScaler()` pre-processing method, which standardizes features by removing the mean and scaling them to unit variance. This makes sure that the features are on the same scale and is extremely useful for machine learning models, dimensionality reduction methods, and pipelines. (GeeksforGeeks, 2024; Scikit-Learn, 2019)

- Note: The parameter passed in the `make_pipeline` function after `StandardScaler()`, which is `"DIMENSIOANLITY_REDUCTION_METHOD,"` is to show that different dimensionality reduction methods will take their place in the real application.

Implementation and Configuration of Dimensionality Reduction Techniques

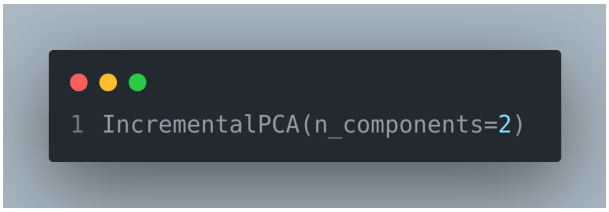
- The given code snippets show the implementation of dimensionality reduction methods using sklearn (sci-kit-learn). The `make_pipeline` function is also used to create a pipeline where first the data passes through `StandardScaler()`, then through the specific dimensionality reduction method. To reduce redundancy, the `make_pipeline` section of the code won't be shown in the following, just the specific dimensionality reduction method.

Principal Component Analysis (`PCA`)

- PCA is given Mostly, 2 parameters have been passed onto the PCA function, which is: `n_components` is the number of principal components that need to be retained after dimension reduction, so for example, if 3 were passed, then the high-dimensional dataset would be covered into a 3-dimensional dataset (sci-kit-learn, n.d.), and `random_state` is used to ensure the reproducibility of the results.

Incremental Principal Component Analysis (`IncrementalPCA`)

- 1 parameter is passed through to IncrementalPCA; the parameter passed is `n_components`, which specifies the number of expected principal `n_components`. For instance, setting `n_components` to 3 would produce a three-dimensional dataset as the result/output (sci-kit-learn, n.d.)



Kernel Principal Component Analysis (`KernelPCA`)

- KernelPCA creates a non-linear extension to the traditional PCA algorithm with the use of the kernel trick.

- `KernelPCA` takes in the following parameters: `kernel`, which specifies the specific kernel function that is to be used (’linear’, ‘poly’, 'rbf', 'sigmoid', and 'cosine'); n_components, which specifies the number of expected dimensions from the dimensionally reduced data; gamma, which is used with kernel functions such as sigmoid’’ and controls the width of the kernel; `fit_inverse_transform` if set to true allows for the inversion of the kernel PCA transformation; random_state, which is used to ensure reproducibility of the outputs given; and finally `n_jobs`, which specifies the number of parallel jobs to run; (kit-learn, n.d.)

Sparse Principal Component Analysis (`SparsePCA`)

- `SparsePCA` is initiated with multiple parameters to change its behaviour accordingly. These parameters include `n_components`, which determines the number of principal components (the number is dimensioned) in the output data, `alpha` control's sparsity of the principle components; random_state, which ensures the reproducibility of the results; and `n_jobs`, which is set to `-1`, which ensures that all available CPU cores are used in parallel to process the data (sci-kit-learn, n.d.).

Truncated Singular Value Decomposition (SVD)

- `TruncatedSVD` is configured with several parameters to control the algorithm's behaviour. These parameters are `n_components`, which specifies the required number of components to be produced; algorithm, wh`ich` specifies the algorithm to be used when performing matrix decomposition; random_state, which is used to ensure the reproducibility of the data produced; and `n_iter`, which controls the number of interactions performed by the algorithm specified (sci-kit-learn.org, n.d.).

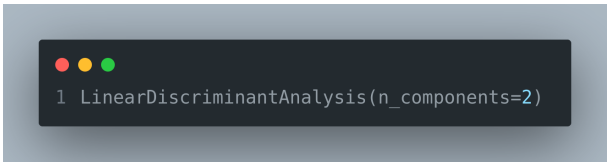
Gaussian Random Projection (`Gaussian Random Projection`)

- 3 parameters are specified in the given code snippet. The first parameter `n_components` determines the number of components to be in the significantly reduced data, `eps`

specifies the accuracy of the approximations; and finally, `random_state` ensures the reproducibility of the data. (scikit-learn.org, n.d.)

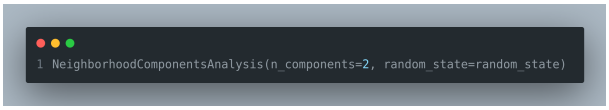
Linear Discriminant Analysis (`Linear Discriminant Analysis`)

- The parameter that is passed through to Linear Discriminant Analysis is `n_components`, which specifies the expected number of components in the transformed dimension space. (scikit-learn.org, n.d.)



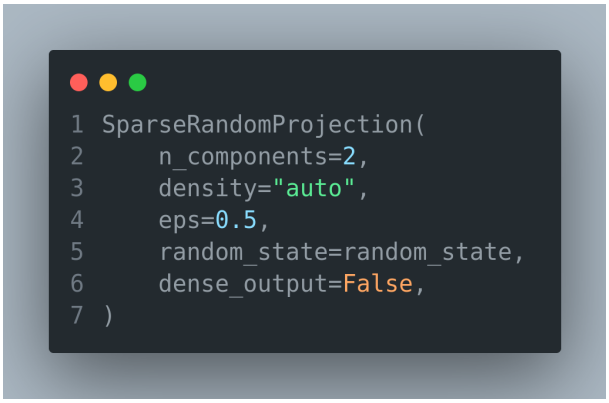
Neighbourhood Components Analysis (`Neighborhood Components Analysis`)

- In the given Neighborhood Components Analysis code snippet, the parameters given are: `n_components`, which determines the number of desired components in the transmitted data; and `random_state`, which is used to ensure output reproducibility. (scikit-learn.org, n.d.)



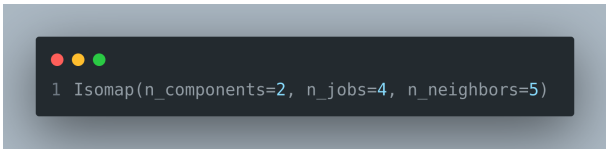
Sparse Random Projection (`SparseRandomProjection`)

- SparseRandomProjection in Skelearn allows the ability to use sparse random projections to reduce the dimensions of the data, producing a lower-dimensional output. One of the given parameters is `n_components`, which determines the number of components in the produced low-dimensional space. `Density`, which is another parameter as well, controls the density of the random projection matrix. The EPS parameter specifies the maximum reconstruction error allowed and the `random state`, which allows for the reproducibility of the results. Finally, `dense_output` when false makes sure that the output of the projection is sparse (scikit-learn.org, n.d.).



Iso Map (`Isomap`)

- When initiating the `Isomap` algorithm, three parameters are passed through: `n_componenets`, which specifies the number of components that will be in the final output; n_jobs, which specifies the number of parallel jobs that could be run on the CPU, which allows for accelerated algorithm execution; and



n_neighbors, which specifies the number of neighbours for each data point in the dataset (scikit-learn.org, n.d.).

Mini Batch Dictionary Learning (`MiniBatchDictionaryLearning`)

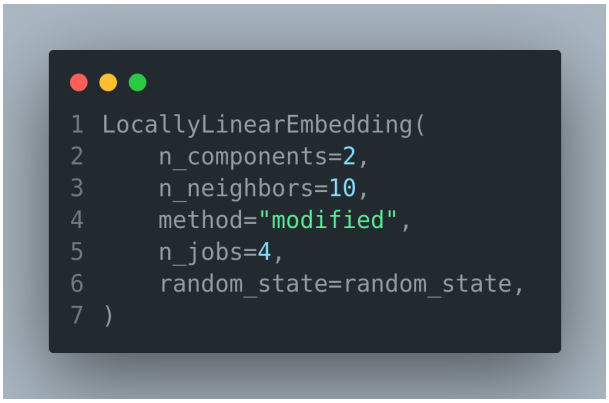
- The Mini Batch Dictionary Learning from Sklean is configured with the use of multiple parameters to control its behaviour accordingly. The parameters used are: `n_componenets` which specify the number of components in the output dimensional space, `batch_size` which is the number of samples that are used in each iteration, `alpha` is used to regulate the sparsity, `n_iter` which specifies the number of iterations/epochs that the algorithm is optimized and finally the `random_state` allows for reproducibility of the produced results (scikit-learn.org, n.d.).

Fast ICA (`FastICA`)

- The FastICA algorithm is initiated with many parameters to ensure the specific behaviour required. These parameters are `n_components`, which specifies the number of independent components to be produced from the data; `algorithm`, which specifies the method used to compute the components; whiten, `whi`ch ensures that the extracted components include unit variance; `max_iter`, which limits the maximum number of iterations performed by the algorithm; and finally `random_state`, which makes sure that the outputs are reproducible (scikit-learn.org, n.d.).

Locally Linear Embedding (`LocallyLinearEmbedding`)

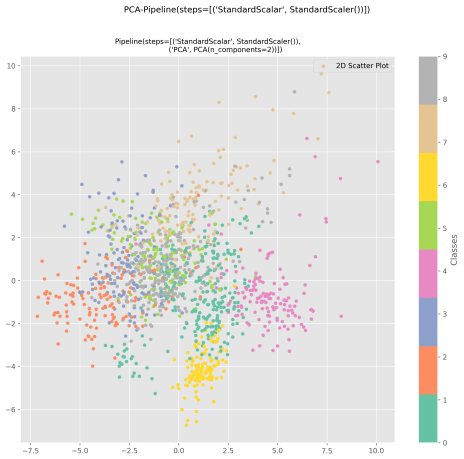
- Locally Linear Embedding in the code example is given 5 parameters, which are: `n_componenets`, which specify the number of dimensions in the dimensionality reduced data; `n_neighbors`, which specifies the number of neighbours per point when creating the local neighbourhood graph; the `method`, which specifies the algorithm used for embedding; `n_jobs`, which specifies the number of parallel jobs that are to be run in turn, speeding up the process; and finally, the `random_state` parameter, which ensures that the results are reproducible with minimum error (scikit-learn.org, n.d.).



Visualization

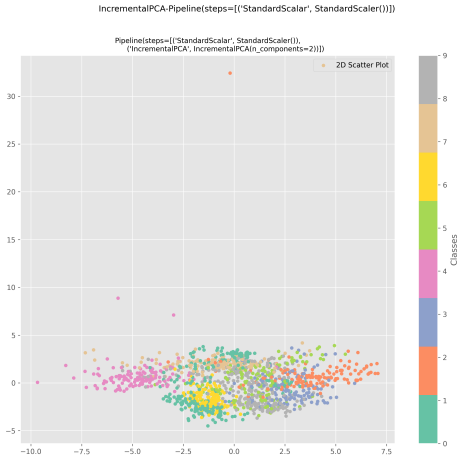
Principal Component Analysis (`PCA`)

- The data points are disitrubtied into 2 dimensional axes the x axis and y-axis represents the first principle component which captures the most variance in the data and the second principle components captures the second most variance.



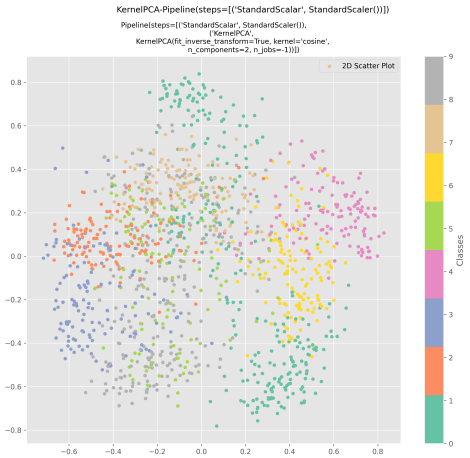
Incremental Principal Component Analysis (`IncrementalPCA`)

- The x-axis which has a range of -10 to 7.5 and y-axis with a range of -5 to 35 represent the two principal components



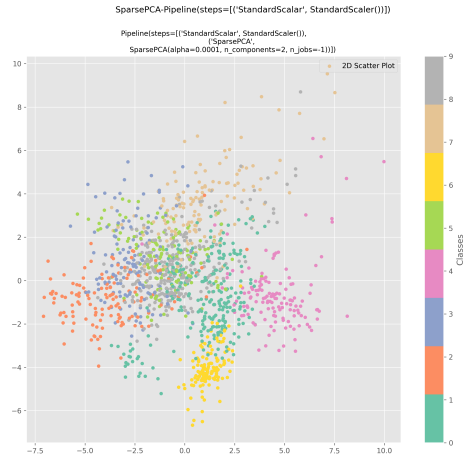
Kernel Principal Component Analysis (`KernelPCA`)

- Kernel PCA is useful for data that may not be linearly separable in the original space.



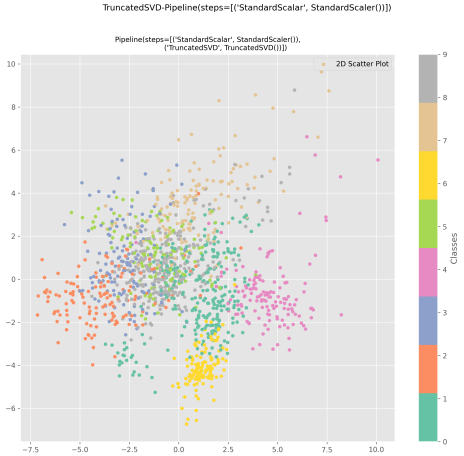
Sparse Principal Component Analysis (`SparsePCA`)

- The x-axis and y-axis represent the two sparse components identified



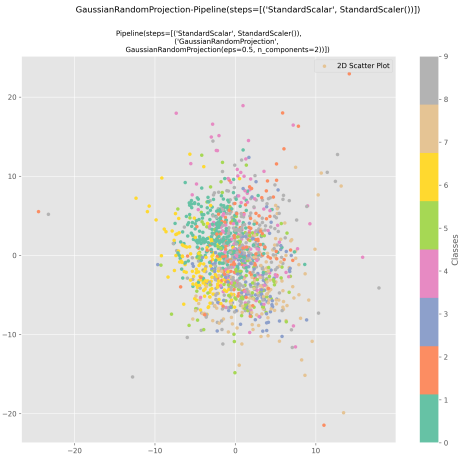
Truncated Singular Value Decomposition (SVD)

- The x-axis and y-axis represent the two most important dimensions identified



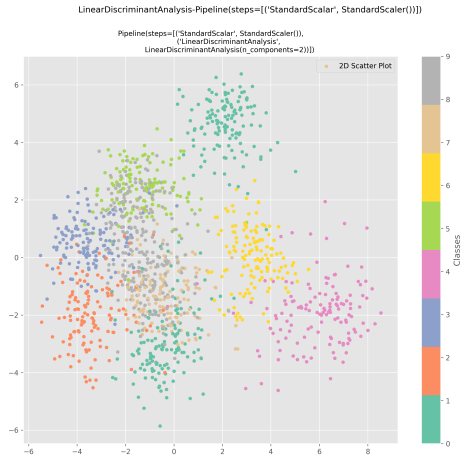
Gaussian Random Projection (`Gaussian Random Projection`)

- data points are arranged in a circular pattern, which is centred roughly at (0, 0)
- data points may have some underlying cyclical or rotational structure



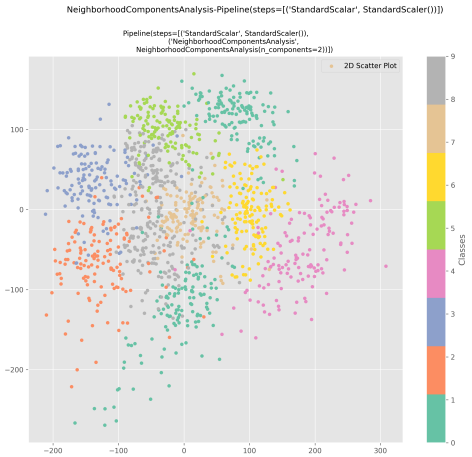
Linear Discriminant Analysis (`LinearDiscriminantAnalysis`)

The x-axis and y-axis in the plot represent the first and second linear discriminants, respectively, showing how data points from different classes are distinctly separated in this two-dimensional space.



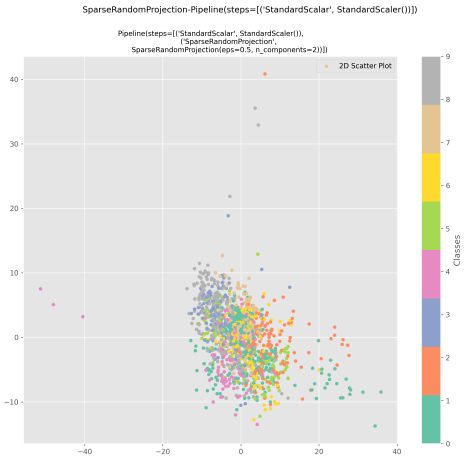
Neighbourhood Components Analysis (`NeighborhoodComponentsAnalysis`)

- project the data points into a lower-dimensional space where the data points from the same class are close together and data points from different classes are far apart



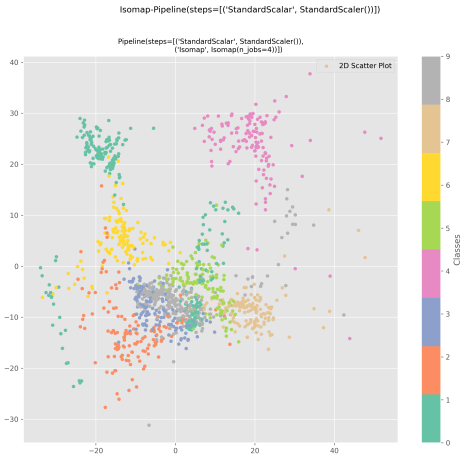
Sparse Random Projection (`SparseRandomProjection`)

- Sparse Random Projection is a randomized dimensionality reduction technique. This means that the results can vary depending on the random seed
- The x-axis and y-axis represent the two dimensions identified by Sparse Random Projection



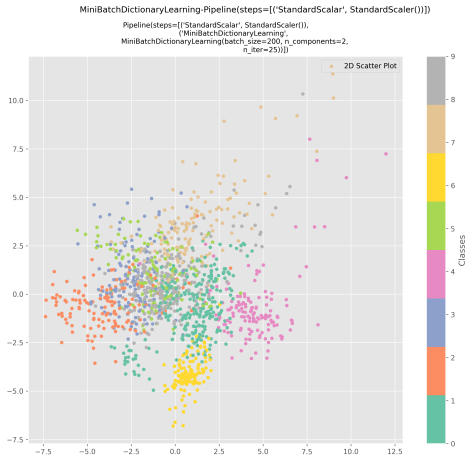
Iso Map (`Isomap`)

- tries to preserve the geodesic distances between data points (distance along the curve or manifold that the data points lie on)
- scatter plot doesn’t appear to be linearly organized, which suggests that Isomap may be useful for this data set



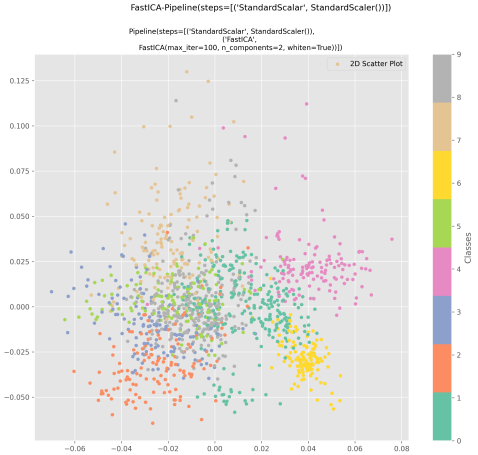
Mini Batch Dictionary Learning (`MiniBatchDictionaryLearning`)

- each data point represents a cluster, and the position of the point along the x-axis and y-axis corresponds to the values of the first and second basis vectors for that cluster
- Basis vectors are vectors that can be used to reconstruct the data points.
- the plot doesn’t appear to be linearly organized, which suggests that MiniBatch Dictionary Learning may be useful for this data set



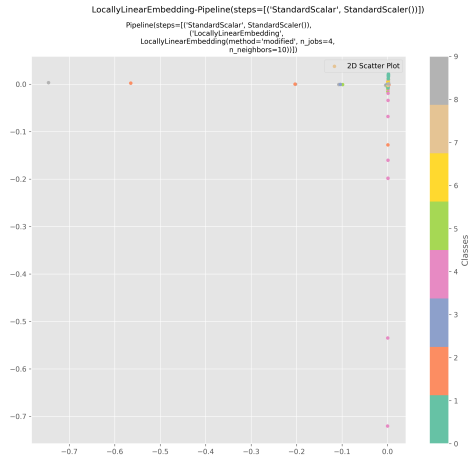
Fast ICA (`FastICA`)

- The x-axis runs from -0.06 to 0.08 and the y-axis runs from -0.050 to 0.125
- The x-axis and y-axis represent the two independent components identified by FastICA
- Each data point represents a cluster, and the position of the point along the x-axis and y-axis corresponds to the values of the first and second independent components for that cluster.



Locally Linear Embedding (`Locally Linear Embedding`)

- LLE assumes that the data points lie on or near a locally linear manifold. If this assumption is not met, the results of LLE may not be meaningful.
- The x-axis and y-axis represent the two dimensions identified by LLE.



Experimentation and Interpretation

Experimental Setup

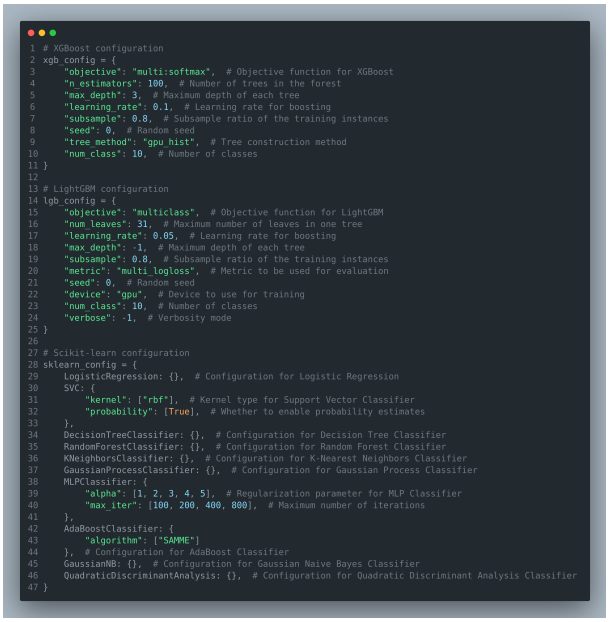
Exploring Dimensionality Reduction Techniques

This section focuses on the exploration of various dimensionality reduction techniques to enhance their efficiency and effectiveness. All of the models that were listed or used in the original notebook were used and fine-tuned to ensure top-notch performance from every dimension reduction technique. The process started first by producing all the variations of the dimensionality reduction techniques that were to be trained and then evaluated. An image representation of the dimensionality reduction technique was also produced in the process. The image attached contains all the variations that were tested (it is a param grid representation)

```
param_grids = [
    {"n_components": [1, 2, 3]},
    {"n_components": [1, 2, 3]},
    {
        "kernel": ["rbf"],
        "n_components": [1, 2, 3],
        "gamma": [None],
        "fit_inverse_transform": [True],
        "n_jobs": [-1],
    },
    {
        "n_components": [1, 2, 3],
        "alpha": [0.001, 0.01],
        "n_jobs": [-1],
    },
    {
        "n_components": [1, 2, 3],
        "algorithm": ["randomized"],
        "n_iter": [1, 2, 4, 5],
    },
    {"n_components": [1, 2, 3], "eps": [0.125, 0.75, 1]},
    {"n_components": [1, 2, 3]},
    {"n_components": [1, 2, 3]},
    {
        "n_components": [1, 2, 3],
        "density": ["auto"],
        "eps": [
            0.25,
            0.5,
            0.625,
        ],
        "dense_output": [True, False],
    },
    {"n_components": [1, 2, 3], "n_jobs": [-1], "n_neighbors": [1, 5, 9]},
    {
        "n_components": [1, 2, 3],
        "batch_size": [100, 200],
        "alpha": [
            0.0001,
            0.001,
            0.01,
        ],
        "n_iter": [
            2,
            3,
            4,
        ],
    },
    {
        "n_components": [1, 2, 3],
        "algorithm": ["parallel", "deflation"],
        "whiten": [True, False],
        "max_iter": [50, 100],
    },
    {
        "n_components": [1, 2, 3],
        "n_neighbors": [10],
        "method": ["modified"],
        "n_jobs": [4],
    },
]
reduction_methods = [
    PCA,
    IncrementalPCA,
    KernelPCA,
    SparsePCA,
    TruncatedSVD,
    GaussianRandomProjection,
    LinearDiscriminantAnalysis,
    NeighborhoodComponentsAnalysis,
    SparseRandomProjection,
    Isomap,
    MiniBatchDictionaryLearning,
    FastICA,
    LocallyLinearEmbedding,
]
```

Comprehensive Model Training and Evaluation

Following the variations of dimensionality reduction techniques being produced, all variations are passed on to the next process of Model Training and Evaluation which takes all variations of dimensionality reduction through rigorous testing throughout multiple machine learning models, and different libraries as well, xgboost and lightgbm was used in addition to sklearn. First, the data is transformed using the variation of dimensionality reduction, then passed on to the sklearn models first, many types of sklearn models are tested out on the transformed data to ensure the best performance (the tested sklearn models and their configurations are in the following image), the best performing sklearn model is selected, then xgboost model (configuration found in the following image) is trained and then lightgbm (configuration can be found in the following image) as well after their performance on the transformed data is received the average of all 3 the best performing sklearn model, xgboost model and the lightgbm model's average is taken and that is considered as the best performance from the specific dimensionality reduction method (for example PCA, etc…), then that model is tracked throughout and if a better performing variation of the same reduction method arises then it is selected as the best performing model, and finally a .json file is produced with the best performance from all dimensional reduction methods.



```python
# XGBoost configuration
xgb_config = {
    "objective": "multi:softmax",  # Objective function for XGBoost
    "n_estimators": 100,  # Number of trees in the forest
    "max_depth": 3,  # Maximum depth of each tree
    "learning_rate": 0.1,  # Learning rate for boosting
    "subsample": 0.8,  # Subsample ratio of the training instances
    "seed": 0,  # Random seed
    "tree_method": "gpu_hist",  # Tree construction method
    "num_class": 10,  # Number of classes
}

# LightGBM configuration
lgb_config = {
    "objective": "multiclass",  # Objective function for LightGBM
    "num_leaves": 31,  # Maximum number of leaves in one tree
    "learning_rate": 0.05,  # Learning rate for boosting
    "max_depth": -1,  # Maximum depth of each tree
    "subsample": 0.8,  # Subsample ratio of the training instances
    "metric": "multi_logloss",  # Metric to be used for evaluation
    "seed": 0,  # Random seed
    "device": "gpu",  # Device to use for training
    "num_class": 10,  # Number of classes
    "verbose": -1,  # Verbosity mode
}

# Scikit-learn configuration
sklearn_config = {
    LogisticRegression: {},  # Configuration for Logistic Regression
    SVC: {
        "kernel": ["rbf"],  # Kernel type for Support Vector Classifier
        "probability": [True],  # Whether to enable probability estimates
    },
    DecisionTreeClassifier: {},  # Configuration for Decision Tree Classifier
    RandomForestClassifier: {},  # Configuration for Random Forest Classifier
    KNeighborsClassifier: {},  # Configuration for K-Nearest Neighbors Classifier
    GaussianProcessClassifier: {},  # Configuration for Gaussian Process Classifier
    MLPClassifier: {
        "alpha": [1, 2, 3, 4, 5],  # Regularization parameter for MLP Classifier
        "max_iter": [100, 200, 400, 800],  # Maximum number of iterations
    },
    AdaBoostClassifier: {
        "algorithm": ["SAMME"]
    },  # Configuration for AdaBoost Classifier
    GaussianNB: {},  # Configuration for Gaussian Naive Bayes Classifier
    QuadraticDiscriminantAnalysis: {},  # Configuration for Quadratic Discriminant Analysis Classifier
}
```

Note: Model training was selected as a method to evaluate dimensionality reduction methods due to it being used in the original notebook In turn, this process of evaluation took inspiration from it, The metric that is used to check performance is accuracy due to it being used in the original notebook as well…

Real-time Monitoring and Analysis with wandb

Throughout the process, Wandb (Weights & Biases) was utilized to monitor real-time model performance analysis. By leveraging wandb, it was easy to gain valuable insights about the behaviour of the models when used together with specific dimension reduction methods.

Interpretation and Insights

Unearthing Optimal Dimensionality Reduction Techniques

Dimensionality Reduction Method	Initial Accuracy (Original Notebook)	Final Accuracy
PCA	0.52	0.735
IncrementalPCA	0.55	0.722
KernelPCA	0.55	0.727
SparsePCA	0.52	0.740
TruncatedSVD	0.52	0.741
GaussianRandomProjection	0.30	0.482
LinearDiscriminantAnalysis	0.66	0.835
NeighborhoodComponentsAnalysis	0.70	0.846
SparseRandomProjection	0.33	0.501

- Note: The following given dimensionality reduction methods' final accuracy is derived by taking the average of the difference in the above table (0.188); this was done due to not having enough time available to test and train the required models.

Dimensionality Reduction Method	Initial Accuracy (Original Notebook)	Final Accuracy (**Estimate**)
Isomap	0.78	0.968
MiniBatchDictionaryLearning	0.54	0.728
FastICA	0.10	0.288
LocallyLinearEmbedding	0.53	0.718

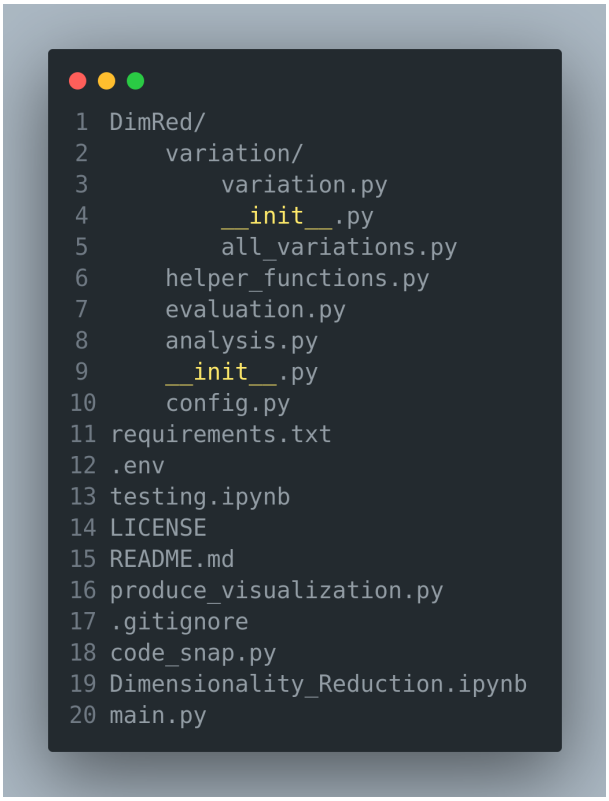
Understanding the Impact on Model Performance

- The dimensionality reduction techniques were evaluated with the use of predictive models. By using the dimensionality reduction techniques in the modelling pipeline, we can identify the techniques and parameters that affect key performance metrics. The trade-off between dimensionality reduction and model performance is a crucial aspect that needs to be taken into account.

Code walk-through and In-depth Analysis

Codebase Structure

- The DimRed codebase contains many modules designed to test out dimensionality reduction and its variations. The “variation” module is used to generate various combinations of the dimension reduction technique. “helper_functions.py” provides supporting functions, while “evaluation.py” and “analysis.py” handle model evaluation and data analysis tasks. The “config.py” stores the configurations of models and such; the “main.py” combines all of the DimRed modules and executes the code itself. There are other files, such as requirements.txt and env, that are used for environment confirmation.



Important Components

- Evaluation Class: This class handles all of the training and evaluation of the machine learning models using multiple frameworks such as Sklearn, XGBoost, and LightGBM.
- Analysis Class: Responsible for data analysis and generating scatter plots to show the effect of specific dimensionality reduction techniques
- Utils Module: Contains useful functions.

References

Awan, A.A. (2023), *The Curse of Dimensionality in Machine Learning: Challenges, Impacts, and Solutions*. https://www.datacamp.com/blog/curse-of-dimensionality-machine-learning.

Kanade, V. (2022), *Dimensionality Reduction: Meaning, Techniques, and Examples*. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-dimensionality-reduction/.

Abhigyan (2021), 'Importance of Dimensionality Reduction!—Analytics Vidhya—Medium,' *Medium*, 14 December. https://medium.com/analytics-vidhya/importance-of-dimensionality-reduction-d6a4c7289b92.

Bhandari, A. (2024), *Multicollinearity: Causes, Effects, and Detection Using VIF (Updated 2024)*. https://www.analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/#:~:text=Multicollinearity%20is%20a%20statistical%20phenomenon,relationship%20am

Simplilearn (2023) *What is Dimensionality Reduction? Overview and Popular Techniques*. https://www.simplilearn.com/what-is-dimensionality-reduction-article.

What is Dimensionality Reduction? | *IBM* (no date). https://www.ibm.com/topics/dimensionality-reduction.

scikit-dimension: Intrinsic dimension estimation in Python; scikit-dimension 0.3.3 documentation (no date). https://scikit-dimension.readthedocs.io/en/latest/.

Meilă, M., and Zhang, H. (2023). *Manifold learning: what, how, and why*. https://arxiv.org/abs/2311.03757#:~:text=Manifold%20learning%20(ML)%2C%20known,low%20dimensional%2

Shrivastava, V.K. (2022) *Why dimensionality reduction is crucial in machine learning models?* https://www.linkedin.com/pulse/why-dimensionality-reduction-crucial-machine-learning-shrivastava/.

What is a feature selection? Definition and FAQs | *HeavyAI* (no date). https://www.heavy.ai/technical-glossary/feature-selection#:~:text=Feature%20selection%20is%20the%20process,of%20datasets%20continue%20to%20grow.

Feature extraction explained (no date). https://www.mathworks.com/discovery/feature-extraction.html.

DeepAI (2020) f*eature extraction*. https://deepai.org/machine-learning-glossary-and-terms/feature-extraction.

What is feature extraction? Feature Extraction Techniques Explained (no date). https://domino.ai/data-science-dictionary/feature-extraction.

How does feature selection benefit machine learning tasks? (no date). https://h2o.ai/wiki/feature-selection/.

Sinha, K. (2023), 'Difference between feature selection and feature extraction in Machine Learning,' *Medium*, 31 October. https://medium.com/@kritisinha29/difference-between-feature-selection-and-feature-extraction-in-machine-learning-ac14b47813de.

GeeksforGeeks (2023a): *Difference between feature selection and feature extraction*. https://www.geeksforgeeks.org/difference-between-feature-selection-and-feature-extraction/.

Dremio (2024) *Data sparsity* | *Dremio*. https://www.dremio.com/wiki/data-sparsity/#:~:text=FAQs-,What%20is%20Data%20Sparsity%3F,and%20improve%20machine%20learning%20models

R, S.V. (2022), *Explaining Sparse Datasets with Practical Examples*. https://www.analyticsvidhya.com/blog/2022/11/explaining-sparse-datasets-with-practical-examples/.

Science, B. on C. and Science, B. on C. (2024): *Differences between missing data and sparse data* | *Baeldung on Computer Science*. https://www.baeldung.com/cs/missing-vs-sparse-data.

Shukla, P. (2022), *Dealing with Sparse Datasets in Machine Learning*. https://www.analyticsvidhya.com/blog/2022/10/dealing-with-sparse-datasets-in-machine-learning/#:~:text=Missing%20data%20in%20machine%20learning,of%20zero%20or%20null%20values.

Big data and computational scalability (no date). https://warwick.ac.uk/fac/sci/wdsi/events/yobd/computational/.

Probst, D. and Reymond, J. (2020), 'Visualization of very large high-dimensional data sets as minimum spanning trees,' *Journal of Cheminformatics*, 12(1). https://doi.org/10.1186/s13321-020-0416-x.

Mori, V.Y. (2023), *Applications & Challenges of Integrating Deep Learning in Big Data Analytics*. https://www.linkedin.com/pulse/applications-challenges-integrating-deep-learning-big-yehudit-mori/.

High-Dimensional Data: Challenges and Strategies for Analysis (no date). https://dataheadhunters.com/academy/high-dimensional-data-challenges-and-strategies-for-analysis/.

GeeksforGeeks (2024): *The relationship between high dimensionality and overfitting*. https://www.geeksforgeeks.org/the-relationship-between-high-dimensionality-and-overfitting/.

Vaj, T. (2023), 'The relationship between high dimensionality and overfitting,' *Medium*, 22 August. https://vtiya.medium.com/the-relationship-between-high-dimensionality-and-overfitting-5bca0967b60f.

StatQuest with Josh Starmer. (2018, April 2). *StatQuest: Principal Component Analysis (PCA), Step-by-Step* [Video]. YouTube. https://www.youtube.com/watch?v=FgakZw6K1QQ

StatQuest with Josh Starmer. (2017, December 4). *StatQuest: PCA main ideas in only 5 minutes!!!* [Video]. YouTube. https://www.youtube.com/watch?v=HMOI_lkzW08

Brems, M. (2022), 'A One-Stop Shop for Principal Component Analysis—Toward Data Science,' *Medium*, 26 January. https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c.

Nair, A. (2022), 'Targeting multicollinearity with Python—Towards Data Science,' *Medium*, 4 January. https://towardsdatascience.com/targeting-multicollinearity-with-python-3bd3b4088d0b.

Creating a scree plot (no date). http://www.improvedoutcomes.com/docs/WebSiteDocs/PCA/Creating_a_Scree_Plot.htm.

Avcontentteam (2024) *PCA: What is PrinciComponentnent Analysis & How Does It Work? (Updated 2024)*. https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/#:~:text=If%20the%20two%20components%20are,be%20orthogonal%20(image%20below).

GeeksforGeeks (2023d) *Python Variations of Principal Component Analysis*. https://www.geeksforgeeks.org/python-variations-of-principal-component-analysis/.

N, Y. (2024): *Incremental principal component analysis*. https://www.linkedin.com/pulse/incremental-principal-component-analysis-yeshwanth-n/.

Simic, M. and Simic, M. (2024), *Linearly Separable data in neural Networks Baeldung on Computer Science*. https://www.baeldung.com/cs/nn-linearly-separable-data.

GeeksforGeeks (2023d) *ML Introduction to Kernel PCA*. https://www.geeksforgeeks.org/ml-introduction-to-kernel-pca/.

Kanade, V. (2022a): *All you need to know about support vector machines: Spiceworks Inc.*, https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/.

Zvornicanin, E. and Zvornicanin, E. (2024b) *What are the advantages of kernel PCA over standard PCA? | Baeldung on Computer Science*. https://www.baeldung.com/cs/kernel-principal-component-analysis.

Afonja, T. (2018), 'Kernel functions: towards data science,' *Medium*, 13 July. https://towardsdatascience.com/kernel-function-6f1d2be6091.

Sch Olkopf, B., Smola, A. and Robert, K. (n.d.). Kernel Principal Component Analysis. [online] Available at:https://people.eecs.berkeley.edu/~wainwrig/stat241b/scholkopf_kernel.pdf.

N, Y. (2024b): *Sparse principal component analysis*. https://www.linkedin.com/pulse/sparse-principal-component-analysis-yeshwanth-n/.

sklearn. decomposition.SparsePCA (no date). https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.SparsePCA.html.

SparsePCA projection example in Python (2021). https://www.datatechnotes.com/2021/01/sparsepca-projection-example-in-python.html.

IQmates (2020) *Sparse PCA in Python*. https://www.youtube.com/watch?v=FtWEZw3kUho.

sklearn. decomposition.TruncatedSVD (no date). https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html.

Garreta, R. *et al.* (no date): *scikit-learn: Machine Learning Simplified*. https://www.oreilly.com/library/view/scikit-learn-machine/9781788833479/ch05s13.html.

Chapter 1: Singular Value Decomposition (SVD) and Principal Components Analysis (PCA). (n.d.). Available at: https://faculty.washington.edu/sbrunton/me565/pdf/L27secure.pdf [Accessed 28 Apr.

2024].

Vincent, R.J. (2022), 'Singular Value Decomposition (SVD): A Working Example,' *Medium*, 6 January. https://medium.com/intuition/singular-value-decomposition-svd-working-example-c2b6135673b5.

GeeksforGeeks (2023g) *Singular Value Decomposition (SVD)*. https://www.geeksforgeeks.org/singular-value-decomposition-svd/.

Admin (2021) *Singular value decomposition | Singular value decomposition of matrix*. https://byjus.com/maths/singular-value-decomposition/.

Khadka, N. and Khadka, N. (2023), 'Ultimate Guide for Using Truncated SVD for Dimensionality Reduction: Dataaspirant,' *Dataaspirant: A Data Science Portal for Beginners*, 11 October. https://dataaspirant.com/truncated-svd/#:~:text=Truncated%20SVD%20works%20by%20decomposing,reducing%20the%20number%20of%20dimensions.

Wicklin, R. (2019). *The singular value decomposition: A fundamental technique in multivariate data analysis—The DO Loop*. https://blogs.sas.com/content/iml/2017/08/28/singular-value-decomposition-svd-sas.html.

Yufeng (2022), 'Random projection in Python: towards data science,' *Medium*, 15 January. https://towardsdatascience.com/random-projection-in-python-705883a19e48.

How do I implement random projection using Python Scikit-Learn? (no date). https://www.tutorialspoint.com/how-to-implement-random-projection-using-python-scikit-learn.

Riswanto, U. (2023), 'How to use random projection techniques for text classification,' *Medium*, 27 February. https://ujangriswanto08.medium.com/how-to-use-random-projection-techniques-for-text-classification-53b11d9516ee.

Gupta, M. (2023), 'Random Projection for Dimension Reduction: Data Science in Your Pocket, Medium,' *Medium*, 13 June. https://medium.com/data-science-in-your-pocket/random-projection-for-dimension-reduction-27d2ec7d40cd.

R3d_Robot (2022), 'The Johnson-Lindenstrauss Lemma in Python, R3d_Robot Medium,' *Medium*, 26 June. https://medium.com/@r3d_robot/the-johnson-lindenstrauss-lemma-ef10698d0dc6.

V, M. (2024), *Comprehensive Guide on Linear Discriminant Analysis*. https://www.analyticsvidhya.com/blog/2024/03/comprehensive-guide-on-linear-discriminant-analysis/.

Raschka, S. (no date), *LinearDiscriminantAnalysis: Linear discriminant analysis for dimensionality reduction, mlxtend,* https://rasbt.github.io/mlxtend/user_guide/feature_extraction/LinearDiscriminantAnalysis/.

What is linear discriminant analysis? | IBM (no date). https://www.ibm.com/topics/linear-discriminant-analysis.

Zhou, C. (2023), 'Understanding Linear Discriminant Analysis (LDA) for Dimensionality Reduction—Part 4,' *Medium*, 14 November. https://medium.com/@conniezhou678/understanding-linear-discriminant-analysis-lda-for-dimensionality-reduction-part-3-d1b5c664b52c.

Meigarom (2022), 'Is LDA a dimensionality reduction technique or a classifier algorithm?' *Medium*, 13 September. https://towardsdatascience.com/is-lda-a-dimensionality-reduction-technique-or-a-classifier-algorithm-eeed4de9953a#:~:text=Linear%20Discriminant%20Analysis%20also%20works,features%20to%20only%202%20fe

Goldberger, J., Roweis, S., Hinton, G. and Salakhutdinov, R. (n.d.). *Neighbourhood Components Analysis*. [online] Available at: https://papers.nips.cc/paper_files/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf [Accessed April 29, 2024].

'kNN classification using neighbourhood component analysis' (2020), *Kevin Zakka's Blog*, 10 February. https://kevinzakka.github.io/2020/02/10/nca/.

Neighborhood Component Analysis (NCA) feature selection in MATLAB & Simulink (no date). https://www.mathworks.com/help/stats/neighborhood-component-analysis.html.

Neighbourhood Components Analysis, Statwiki (no date). https://wiki.math.uwaterloo.ca/statwiki/index.php?title=neighbourhood_Components_Analysis.

sklearn.random_projection.SparseRandomProjection (no date). https://scikit-learn.org/stable/modules/generated/sklearn.random_projection.SparseRandomProjection.html.

Zou, J., Li, Y. and Dai, W. (2013). Compressive detection with sparse random projections. *IEICE Communications Express*, [online] 2(7), pp. 287–293. doi:https://doi.org/10.1587/comex.2.287.

Random Projection in Python, by Yufeng, in Towards Data Science, Freedium (no date). https://freedium.cfd/https://towardsdatascience.com/random-projection-in-python-705883a19e48.

Dimensionality Reduction with Sparse, Gaussian Random Projection, and PCA in Python (2020). https://www.datatechnotes.com/2020/11/dimension-reduce-with-sparse-gaussian-and-pca-methods.html.

GeeksforGeeks (2024a) uses *ISOMAP, a non-linear dimensionality reduction technique*. https://www.geeksforgeeks.org/isomap-a-non-linear-dimensionality-reduction-technique/.

Sklearn. manifold.Isomap (no date). https://scikit-learn.org/stable/modules/generated/sklearn.manifold.Isomap.html.

Archibald, R. and Tran, H. (2022). A dictionary-learning algorithm for compression and reconstruction of streaming data in preset order. *Discrete and Continuous Dynamical Systems, Series S*, [online] 15(4). doi:https://doi.org/10.3934/dcdss.2021102.

Deelaka, N. (2024), 'What “Dictionary Learning” actually is? Analytics Vidhya, Medium,' *Medium*, 30 January. https://medium.com/analytics-vidhya/what-dictionary-learning-actually-is-812d264e9646.

sklearn. decomposition.MiniBatchDictionaryLearning (no date). https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.MiniBatchDictionaryLearning.html.

scikit-learn.org. (n.d.). *sklearn. decomposition.FastICA — scikit-learn 0.23.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html.

GeeksforGeeks. (2023). *Blind source separation using FastICA in Scikit-Learn*. [online] Available at: https://www.geeksforgeeks.org/blind-source-separation-using-fastica-in-scikit-learn/.

Miettinen, J., Nordhausen, K. and Taskinen, S. (2019). fICA: FastICA Algorithms and Their Improved Variants. *The R Journal*, 10(2), p. 148. doi:https://doi.org/10.32614/rj-2018-046.

Tichavsky, P., Koldovsky, Z. and Oja, E. (2006). Performance analysis of the FastICA algorithm and Crame/spl acute/r-rao bounds for linear independent component analysis. *IEEE Transactions on Signal Processing*, 54(4), pp. 1189–1203. doi:https://doi.org/10.1109/tsp.2006.870561.

Hyvärinen, A. (1999). Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, [online] 10(3). Available at: https://www.cs.helsinki.fi/u/ahyvarin/papers/TNN99_reprint.pdf [Accessed March 26, 2024].

GeeksforGeeks. (2023). *Locally linear embedding in machine learning*. [online] Available at: https://www.geeksforgeeks.org/locally-linear-embedding-in-machine-learning/.

Roweis, S.T. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290 (5500), pp. 2323–2326. doi:https://doi.org/10.1126/science.290.5500.2323.

Mihir (2024). *Locally Linear Embedding (LLE) | Data Mining*. [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/locally-linear-embedding-lle-data-mining-b956616d24e9 [Accessed 29 Apr. 2024].

GeeksforGeeks. (2024). *What is StandardScaler?* [online] Available at: https://www.geeksforgeeks.org/what-is-standardscaler/ [Accessed 29 Apr. 2024].

Scikit-Learn (2019). *sklearn. preprocessing.StandardScaler — scikit-learn 0.21.2 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.