Computer Science Review 40 (2021) 100378

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)

Farzana Anowar [a,b,*], Samira Sadaoui [a], Bassant Selim [b]

[a] University of Regina, Department of Computer Science, Canada
[b] Ericsson, Montreal, Canada

ARTICLE INFO

Article history:
Received 16 October 2020
Received in revised form 8 January 2021
Accepted 1 February 2021
Available online 21 February 2021

Keywords:
Dimension reduction
Optimal set of features
Data quality
High-dimensional datasets
Correlation metrics
Classification accuracy
Run-time

ABSTRACT

Feature Extraction Algorithms (FEAs) aim to address the curse of dimensionality that makes machine learning algorithms incompetent. Our study conceptually and empirically explores the most representative FEAs. First, we review the theoretical background of many FEAs from different categories (linear vs. nonlinear, supervised vs. unsupervised, random projection-based vs. manifold-based), present their algorithms, and conduct a conceptual comparison of these methods. Secondly, for three challenging binary and multi-class datasets, we determine the optimal sets of new features and assess the quality of the various transformed feature spaces in terms of statistical significance and power analysis, and the FEA efficacy in terms of classification accuracy and speed.

Contents

1. Introduction ... 2
2. Categorization and selection ... 2
   2.1. Categorization ... 2
   2.2. Selection ... 3
   2.3. Notations ... 3
3. Principal component analysis ... 3
4. Linear discriminant analysis ... 4
5. Multi-dimensional scaling ... 4
6. Singular value decomposition ... 5
7. Locally linear embedding ... 5
8. Isometric mapping ... 5
9. Laplacian Elgenmap ... 6
10. Independent component analysis ... 6
11. t-Distributed Stochastic Neighbor Embedding ... 6
12. Summary and comparison ... 7
13. First case study ... 7
   13.1. Feature scaling ... 7
   13.2. Optimal number of PCs ... 7
   13.3. Quality of transformed data ... 9
   13.4. Classification performance ... 9
   13.5. Data plotting ... 10
14. Second case study ... 10
   14.1. Classification performance ... 10
   14.2. Data plotting ... 10
15. Third case study ... 10

* Corresponding author at: University of Regina, Department of Computer Science, Canada.
E-mail address: fad469@uregina.ca (F. Anowar).

https://doi.org/10.1016/j.cosrev.2021.100378
1574-0137/

15.1. Classification performance 11
15.2. Data plotting 11
16. FEA comparison for all three case studies 11
17. Conclusion and future work 12
Declaration of competing interest 12
References 12

1. Introduction

Nowadays, a tremendous amount of data is collected daily. Hence, many measurements are obtained, but only a portion is useful for decision making tasks. Although Machine Learning Algorithms (MLAs) can process big data [1,2], their performance degrades as the dimensionality increases [3,4]. When the number of attributes escalates, the number of observations proportionally increases, and as a result the learned model becomes more complex [5,6]. A model trained on many features becomes strongly reliant on the data and therefore leading to overfitting and low performances on unseen data [7]. The models' accuracy decreases due to the involvement of insignificant and similar features [5]. Dimensionality reduction algorithms aim to solve the curse of dimensionality, with the goal of improving data quality by reducing data complexity. They are mainly categorized into feature selection and feature extraction. The former looks for the most informative features and eliminate less informative ones. The latter combines the features into fewer new features through algebraic transformations [8]. In our study, we focus on Feature Extraction Algorithms (FEAs) because they can better handle issues in real-life datasets, such as noise, complexity, and sparsity [9]. That is why FEAs attracted much more attention from the research community [10].

Generally speaking, FE is a procedure for bringing the number of columns down or converting a sphere to a circle in a two-dimensional space. FEAs linearly or non-linearly combine correlated features into artificial features by preserving the intrinsic properties or structure of the original features i.e., with a minimal loss of information. FEAs look for a representation of a manifold to project the input data and then determine a lower-dimensional space that is embedded in the input space [11]. FEAs comes with several benefits, including, [12]:

- Improvement of MLAs' performance through less misleading and redundant features.
- Avoidance of overfitting through fewer features, and therefore lesser assumptions by the model, and simpler the model.
- Less computing time and much less storage is required with lower data dimensions.
- More ease of data visualization and interpretation.

We examine various FEAs from different categories because they are no clear winners as FEA depend on the data characteristics, quality and size. For instance, nonlinear FEAs perform better for data possessing *"well-sampled smooth manifolds"* [4], and linear methods function the best when data lie on a smooth and linear surface [13]. The authors in [4] explored empirically multiple nonlinear FEAs and one linear FEA, the Principal Component Analysis (PCA), on real and artificial datasets. They conclude that nonlinear FEAs did better on artificial tasks, but not necessarily on real-life tasks even though nonlinear methods were proposed to overcome the drawback of linear methods. Also, nonlinear FEAs do not always outperform PCA because they may be sensitive to the dimensionality curse. Another recent work [10] reviewed several supervised FEAs, and showed that their accuracy might be significantly affected by the datasets' sample size.

Some studies reviewed the complicated theories behind FEAs, each one using different concepts and backgrounds. Our goal is to facilitate the description of the behavior of a good number of FEAs from various categories and provide simpler algorithms to increase their understandability. Additionally, we present two tables summarizing the taxonomy and several key concepts of FEAs. Moreover, we conduct an extensive empirical investigation of FEAs based on three real-world datasets with different dimensionalities and classification settings using several quality metrics. There is no prior research that provided a detailed theoretical analysis and extensive experimental investigation in one place to the best of our knowledge.

In our study, we first classify FEAs into three main groups: linear vs. nonlinear, supervised vs. unsupervised, random projection -based vs. manifold-based. Then, we provide a rationale for selecting certain FEAs and rejecting some others according to past studies' findings. As a result, we examine nine noticeable FEAs theoretically, present their algorithms, and compare them conceptually using several factors, such as data characteristics, transformation method, computational complexity, weaknesses, and tuning parameters. Lastly, we assess the performance of the examined FEAs on three challenging real-life datasets. In the first two 2-class datasets, the number of instances is relatively small compared to the data dimensionality; the first one has 96 features and 200 data points, and the second one 10,000 features and 200 data points. The third multi-class dataset has 561 features, 10,299 instances and 6 classes. As mentioned in [6], for a robust learning, the number of data should grow proportionally with the number of features. For example, the first dataset necessitates at least 10^{96} data points. We assess the quality of the various reduced feature spaces based on correlation metrics, such as the statistical significance and power analysis, and data visualization. We also devise a methodology to determine the optimal set of new features for each FEA, and then evaluate the performance of the optimal data spaces via classification to check whether MLAs learn better from lower dimensional spaces.

2. Categorization and selection

2.1. Categorization

Feature extractors can be classified into different categories as shown below [4,14–16]:

- **Linear** FEAs linearly map a high-dimensional space into a lower space [17], i.e., the lower dimensions are a linear combination of the original dimensions [18]. Examples of this group are Singular Value Decomposition (SVD), Principal Component Analysis (PCA), Wavelet Transformation (WT), Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Factor Analysis (FA), and Non-Negative Matrix Factor (NMF).
- **Nonlinear** FEAs non-linearly map a high-dimensional space into a lower space. This group is in turn divided into:
  - **Global** FEAs provide a representation of data points' global structure [19]. Examples are Multi-Dimensional Scaling (MDS), Kernel PCA (KPCA), and Isometric Mapping (ISOMAP).

- **Local** FEAs produce a better performance on manifolds where the "local geometry is close to Euclidean", but the "global geometry is probably not" [19]. Examples are AutoEncoder, Laplacian Eigen maps (LE), Kernel Fisher Discriminant Analysis (KFDA), a linear extension of LDA, Locally-Linear-Embedding (LLE), and t-Distributed Stochastic Neighbor Embedding (t-SNE).

* **Unsupervised** FEAs focus on data itself with no pre-existing class labels [20]. Most FEAs are unsupervised. Examples are MDS, SVD, PCA, LLE, ISOMAP, t-SNE, and LE.

* **Supervised** FEAs take into account the information of class labels while learning from data [20]. Examples are LDA and ICA.

\+ **Random Projection-based** FEAs efficiently project the original data by utilizing a random matrix in which each column contains a unit length where the root square of the sum of the squares of each element of the column is 1, while preserving the actual relative distances in a Euclidean space between the data. [21,22]. Examples are PCA, LDA, and SVD.

\+ **Manifold-based** Most of the manifold-based FEAs are nonlinear, unsupervised [23] and neighborhood graph-based [4]. They mainly focus on nonlinear mapping, and assume that in low dimensional space, data lies on a densely sampled manifold to be unrolled [18]. Examples are ISOMAP, LLE, MDS, KPCA, t-SNE, and LE.

Considering all the above categories, as examples, PCA is linear, unsupervised, and random projection-based, while ISOMAP is nonlinear (global), unsupervised and manifold-based. ICA can adopt random projection by preserving the quality of data as a pre-processing task [24]. Moreover, FEAs can be classified into two more categories based on their intrinsic nature: (1) convex methods, such as PCA, LLE, LE, Isomap and (2) non-convex methods, such as Autoencoder. A convex method optimizes the objective function in a way that it does not hold local optima, as opposed to non-convex methods [11].

2.2. Selection

Our strategy is to select FEAs from the three main categories by choosing the most representative and successful FEAs according to the literature. [25] found that manifold-based methods, such as LLE, ISOMAP, LE, KPCA, and MDS, are highly efficient because they can better adjust to the local data structure and handle data non-linearity more efficiently. PCA and its variants are still the most preferred methods due to their simplicity. Consequently, we examine the following unsupervised methods: PCA, KPCA, MDS, SVD, LLE, LE, and ISOMAP. In addition, we select two supervised methods, LDA and ICA because LDA performs better in multi-class settings and ICA is useful when we want to represent data as independent sub-elements.

On the other hand, we did not consider the following FEAs for various reasons: (1) FA as it is related to PCA — its use has sharply decreased [25]; (2) NMF is used for speech recognition and computer vision [10] and cannot handle negative values, where the first dataset that we experiment with possesses 40% of negatives values; (3) Autoencoders as they are also sensitive to negative values; and (4) WT as it is mainly used for signal processing.

2.3. Notations

Table 1 presents the notations used in the FEA description.

Table 1
Notations and Definitions.

Notation	Definition
PC	Principal component
X	Input dataset in high dimension
Y	Output dataset in low dimension
d	Original high dimension
k	New low dimension
x_a	ath input data in d dimension
x_b	bth input data in d dimension
y_a	ath output data in k dimension
y_b	bth output data in k dimension
K	Kernel function
c	Nearest neighbors for a data point
n	Number of data points
i	Number of iterations

3. Principal component analysis

Principal Component Analysis (PCA) is an unsupervised, linear transformation algorithm that produces the new features, called Principal Components (PCs), by determining the maximum variance of the data [26]. PCA projects the highly-dimensional dataset to a new subspace where the orthogonal axes, or PCs, are considered as the directions of the maximum data variance [26]. During the transformation, the first PC has the highest variance, and the subsequent PCs have decreasing variances.

We summarize the steps of PCA in Algorithm 1. PCA builds a dxk-dimensional transformation matrix W that maps the original d-dimensional space X to a new k-dimensional space Y (k $\leqslant$ d) [27]. The linear Eigen decomposition method is applied to the covariance matrix ($X.X^T$) to produce the Eigen vectors (PCs) and Eigen values. The Eigen vectors show the data directions and Eigen values the data magnitude. The Eigen values are used to order the columns in the matrix W, where each column is an Eigen vector. The Eigen decomposition method can be defined as follows where first the covariance matrix is decomposed to three other matrices [28]:

$$X.X^T \rightarrow B.D.B^T \tag{1}$$

Here, B is a square matrix ($d \times d$) that contains the Eigen vectors; D is diagonal matrix ($d \times d$) where all the elements, except the main diagonal elements, are zero and the diagonal elements are the respective Eigen values; B^T is the transposed matrix of B.

Algorithm 1: PCA

Input: X $\in R^{n \times d}$
Output: Y $\in R^{n \times k}$

1: Construct the covariance matrix ($X.X^T$)
2: Apply linear Eigen decomposition to $X.X^T$ to obtain Eigen values and vectors
3. Sort Eigen values in decreasing order to sort Eigen vectors
4: Build matrix W ($d \times k$) with k top Eigen vectors
5. Transform X using W to obtain the new subspace $Y = X.W$

In Fig. 1, we apply PCA to the ECG200 dataset investigated in Section 10. We visualize the original feature space with 96 features (light blue) and the new space with 3 features (orange). The figure shows that the original data points are scattered, and in contrast the new data points are more compact. Hence, there are fewer chances of having noises and redundancies.

PCA has several benefits [29]: it is non-iterative and thus less time consuming, reduces over-fitting well, and can also be utilized as a denoising and data compression techniques. However, PCA comes with some disadvantages [30]: it is limited to linear

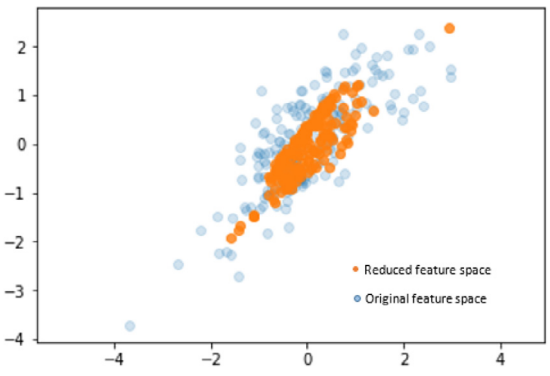

Fig. 1. Original vs. reduced feature spaces.

projections, and thus cannot handle nonlinear data well; data standardization is compulsory before applying PCA, otherwise it will not be able to find the optimal PCs since the directions are significantly sensitive to the feature scales; and if PCs are not selected carefully, the information loss can happen.

PCA does not perform well on nonlinear data, as the produced subspace is not optimal [26]. Hence, Kernel PCA (KPCA) becomes handy. It uses the 'kernel trick' K shown in Equation (2) to project data to a higher feature space so that data can be linearly separated [31].

$$K(x_a, x_b) = \phi(x_a)^T.\phi(x_b) \tag{2}$$

Here, x_a and x_b are two arbitrary data points. As exposed in Algorithm 2, instead of using the input X, KPCA first maps X to a higher feature space $\phi(X)$ by utilizing polar coordinates. The kernel function avoids the explicit mapping of data with the function ϕ, which makes the whole process computationally cheaper [32]. The commonly used kernels in KPCA are: (1) Linear is generally used when data is linearly separable, where $k(x_a, x_b) = {x_a}^T x_b + constant$; (2) Polynomial shows the similarity among data in a feature space over the polynomials of actual variables, where $k(x_a, x_b) = (\alpha {x_a}^T x_b + constant)^{degree}$; (3) RBF, a.k.a. Gaussian kernel, is used when data is non-linearly separable, where $k(x_a, x_b) = \exp(-\frac{|x_a - x_b|^2}{2\sigma^2})$; (4) Sigmoid, mainly used in neural networks, where $k(x_a, x_b) = tanh({x_a}^T x_b + constant)$ [33]. After applying the kernel trick, PCA is then utilized on the new linearly separable data to reduce the dimensionality.

Algorithm 2: KPCA

Input: X $\in R^{n \times d}$, such that X is nonlinear
Output: Y $\in R^{n \times k}$

1: Obtain linear data X' with kernel mapping function K to X
2: Apply PCA to X' to obtain reduced space Y (see Algorithm 1)

4. Linear discriminant analysis

Most of the time, Linear Discriminant Analysis (LDA) is defined as a linear, supervised FEA. However, some authors state that LDA also performs as a linear classifier [23]. LDA identifies a new feature space to project data with the goal of maximizing the classes' separability. From the d independent features of a dataset, it extracts k new independent features that separate the classes (dependent features) the most. Hence, the number of produced components is smaller than the number of classes − 1. First, as shown in Equations (3) and (4) [34], LDA builds two scatter matrices: (1) an in-between-class (SM_b) matrix that computes the distance between the mean of each class, and (2) a within-class (SM_w) matrix that returns the distance between the mean of each class and the data within that class. The remaining process is similar to PCA, as presented in Algorithm 3.

$$SM_b = \sum_{k=1}^{m} N_k(\mu_k - \mu)(\mu_k - \mu)^T \tag{3}$$

$$SM_w = \sum_{k=1}^{m} \sum_{x=1}^{n} (x - \mu'_k)(x - \mu'_k)^T \tag{4}$$

Here, m is the number of classes, μ is the overall mean, μ_k and N_k are the mean and sizes of the respective classes respectively, and μ'_k is the mean vector of a class.

Algorithm 3: LDA

Input: X $\in R^{n \times d}$
Output: Y $\in R^{n \times k}$

1: Build two scatter matrices of X: in-between-class and within-class
2. Calculate Eigen values and respective Eigen vectors of scatter matrices
3: Rank Eigen vectors by their values in descending order
4. Build matrix W $(d \times k)$ with k top Eigen vectors
5. Transform X using W to obtain the new subspace $Y = X.W$

One major disadvantage of using LDA is that for binary classification problems, we will have only one new feature regardless of how many features were present in the original dataset.

5. Multi-dimensional scaling

Multi-Dimensional Scaling (MDS) is a nonlinear, unsupervised FEA that focuses on the relationships (similarities or dissimilarities) between data in a multi-dimensional space [35]. MDS has two main approaches [36]: (1) Metric MDS (classic) and Non-metric MDS. In this work, we cover the classic MDS, also known as Principal Coordinate Analysis [26], which is widely employed. As shown in Algorithm 4, MDS is not concerned with the data in general; rather, it is more attentive to the pairwise dissimilarities between data pairs.

MDS locates data points in lower dimensions by calculating the dissimilarity/distance matrix such that similar data are together and less similar data are far apart [37]. Based on the distance matrix d^X, MDS finds the output Y that maximizes the similarity between d^X and d^Y, where $d^X = \sqrt{x_a - x_b}$ and $d^Y = \sqrt{y_a - y_b}$. Here, d^X and d^Y represent the distances between any two points a and b. MDS converts the distance matrix d^X to a kernel matrix K that can be formulated as [36]:

$$K = H.d^X.H \tag{5}$$

where $H = I\text{-}\ \frac{1}{n}.(ee^T)$ is a centering matrix, I is an identity matrix and e is a column vector of 1s. Then, Eigen decomposition is applied to K, $K \rightarrow B.D.B^T$. Here, D is a diagonal matrix containing the Eigen values of K and B contains the corresponding eigen vectors. After selecting the top k Eigen values, the rest of the values and corresponding vectors are removed, and therefore D and B are renamed $\hat{D}$ and B_1, where $\hat{D} \in R^{k \times k}$ is diagonal matrix of top k Eigen values and $B_1 \in R^{n \times k}$ contains the top k Eigen vectors. The data is then mapped to low dimension as follows [26]:

$$Y = \hat{D}^{1/2}.B_1^T \tag{6}$$

Since MDS is frequently used for data visualization, the lower dimension is usually two or three. However, MDS is known to be computationally expensive.

Algorithm 4: MDS

Input: X $\in R^{n\times d}$
Output: Y $\in R^{n\times k}$

1: Compute the dissimilarity matrix d^X of X
2. Compute K by utilizing the centering matrix H
3: Apply Eigen decomposition to K to have top k Eigen values and corresponding vectors
4. Obtain Y using diagonal matrix K and top k Eigen vectors

6. Singular value decomposition

Singular Value Decomposition (SVD) provides an exact illustration of a dataset represented as a matrix with any number of dimensions. Nevertheless, the less the number of dimensions (components) we select, the less precise will be the illustration of the SVD [38]. With SVD, the top k largest singular values are chosen according to the following theorem [39,40]:

$$X \rightarrow N.S.Z^T \tag{7}$$

Here,

- X is the original matrix, which is decomposed into three matrices.
- N is an $(n \times k)$ matrix with orthogonal unit vector columns, i.e. the dot product of two columns is equal to 0 (orthogonal), which means $N.N^T = N^T.N = I$. N is also known as the Left Singular Vector. The columns are ordered by importance.
- S is a $(k \times k)$ diagonal matrix, where the entries that are not on the main diagonals are 0. The diagonal entries of S are known as singular values.
- Z is an orthogonal matrix ($Z^T.Z = Z.Z^T = I$) of size $k \times d$, known as Right Singular Vector.

The SVD theorem rebuilds the input matrix X in a k-low dimension/rank as exposed in Equation (8), where N_k, S_k, and Z_k^T are the truncated versions of N, S, Z^T. Here, only the top k singular values are retained in Y.

$$Y = N_k.S_k.Z_k^T \tag{8}$$

The singular values of S are positive and arranged in descending order. Scikit-learn employs the 'TruncatedSVD' estimator to linearly extract the features by means of truncated SVD by assigning 0 to all the singular values, except the first k largest singular values (the number of desired components) and using only the first k columns of N and Z. One advantage of SVD is that it can handle sparse matrices quite efficiently. SVD can also be used as a preferred FEA, especially for prediction problems [39].

Algorithm 5: SVD

Input: X $\in R^{n\times d}$
Output: Y $\in R^{n\times k}$

1: Decompose X into 3 matrices: N, S and Z
2: Build Y by selecting k top singular values from S

7. Locally linear embedding

Locally Linear Embedding (LLE) is a nonlinear, unsupervised extraction process that produces low-dimensional global coordinates by assuming that data lies on a smooth nonlinear manifold embedded in a high feature space [41]. While reducing the features, LLE preserves the geometric features (local properties) of the original dataset [4]. The local properties of a data point (x_a) are preserved by considering the linear combination of W_{ab} (reconstruction weights) with its c-nearest neighbors (x_b). To conduct the dimensionality reduction, first LLE finds the c-nearest neighbors of each data using the Euclidean distance. Second, it calculates the local weights for each data point (X_a) that optimally represents data (X'_a) as a linear combination of the nearest neighbors by reducing the reconstruction error. Third, a new vector space (low-dimensional embedding of points) Y is defined by utilizing the Eigen vector-based optimization in a way that minimizes the cost for Y by computing the weights that best rebuild the vectors (data points) from their nearest neighbors. In Equation (9), we show the reconstruction error [41].

$$Error(W) = \sum_a |\vec{x}_a - \sum_b W_{ab}.\vec{x}_b|^2 \tag{9}$$

Here, W_{ab} shows the contribution of X_b while rebuilding the data point X_a. The error is minimized under the following two conditions:

- Every data (X_a) will be rebuilt only from its nearest neighbors, which enforces W_{ab} = 0 when X_b and X_a are not neighbors.
- The sum of every row of the weight matrix is 1 i.e., $\sum_b W_{ab}$= 1.

Every data X_a from d-dimension is mapped to Y_a in k-dimension by minimizing the cost, as shown below [42]:

$$\phi(Y) = \sum_a |\vec{y}_a - \sum_b W_{ab}.\vec{y}_b|^2 \tag{10}$$

Algorithm 6: LLE

Input: X $\in R^{n\times d}$
Output: Y $\in R^{n\times k}$

1: Find c-nearest neighbors for each data point of X
2: Calculate local weights W that linearly best rebuild data (X') from its neighbors
3. Map X' to Y on k-dimensions using same weights from step#2 by minimizing the cost

The strength of LLE lies in its ability to handle nonlinear data, and sparse matrices efficiently. As a result, it may require lower computational time and space than other FE techniques.

8. Isometric mapping

Classical scaling is a well recognized method for calculating data dissimilarities that aims to retain the pairwise distances (Euclidean) of data points, but does not consider the distribution of neighboring data [4]. This implies that classical scaling does not capture possible nonlinear patterns in a dataset. For instance, if data points lie on nonlinear manifold, and after applying any classical scaling, like PCA and MDS, the data points on the different sides of the manifold will mistakenly fall next to each other, since PCA/MDS cannot handle the nonlinear structure in a manifold [43]. Hence, a new method is required to consider the pairwise distances of data points in the manifold. Here comes ISOMAP, known as Isometric Mapping, that solves this issue by maintaining the 'pairwise geodesic distances' between data in lower dimension space [44]. ISOMAP, a nonlinear, unsupervised FEA, is one of the earliest manifold learning approaches. It can also be considered as the extension of MDS [23].

ISOMAP first generates a neighborhood graph by determining the geodesic distances where data x_a is connected to its

nearest neighbors. After that, the shortest path of all pairs of data in the neighborhood graph is computed to approximate the geodesic distance between them, which can quickly be done by using Floyd's or Dijkstra's shortest-path algorithm [44]. A pairwise geodesic distance matrix (d^G) is then built after computing the geodesic distances between all data points. H is the centering matrix, as shown in Equation (5). Following this, MDS with kernel matrix is applied on (d^G) as follows [36]:

$$K = H.d^G.H \tag{11}$$

ISOMAP then finds the low-dimension embedding of a data points from Equation (6) with $\hat{D}$ and B being the respective Eigen values and Eigen vectors for Equation (11).

Algorithm 7: ISOMAP

Input: X $\in R^{n\times d}$
Output: Y $\in R^{n\times k}$

1: Develop the neighborhood graph of X
2: Calculate the geodesic distance and build d_G
3. Apply MDS to d_G to obtain new space Y (see Algorithm 4)

It is worth mentioning that ISOMAP has some severe flaws compared to the other FEAs, including the fact that it suffers from topological instability as it may build wrong connections in the neighborhood graph, which may adversely affect the performance [45]. Also, it may fail when the manifold is non-convex [46] or in the presence of a 'hole' in the manifold [47]. Despite having these issues, ISOMAP is widely used in different scientific domains, like wood inspection and head pose estimation [48].

9. Laplacian Elgenmap

Laplacian Eigenmap (LE) is another unsupervised-nonlinear FEA that looks for a low-dimension data by maintaining the 'local properties' of a manifold, which is similar to LLE. LE first generates a neighborhood graph G', where each data x_a is linked to the nearest neighbors. All x_a and x_b in G' are linked by an edge and the 'Gaussian kernel function' is used where the weights of the edges are evaluated as [26]:

$$Weight_{ab} = e^{-|x_a - x_b|^2/2\sigma^2} \tag{12}$$

where σ specifies the variance of the Gaussian leading to an adjacency matrix (W). To reduce the dimensionality from X to Y, LE minimizes the cost function defined in [49] as:

$$\phi(Y) = \sum_{ab} |y_a - y_b|^2.Weight_{ab} \tag{13}$$

Optimizing the cost function implies a smaller distance between x_a and x_b in the high-dimension, hence, a smaller distance is also obtained between y_a and y_b in the lower dimension space. Besides, the calculation of diagonal matrix D and Laplacian matrix L of G' leads to the reformulation of the cost function in Equation (13) as an Eigen decomposition problem given as [4]:

$$\sum_{ab} |y_a - y_b|^2.Weight_{ab} = 2.Y.L.Y^T \tag{14}$$

where L can be defined as $L = W\text{-}D$; here D is a diagonal matrix where the elements are the row sums of W, such as $D_{aa} = \sum_b W_{ab}$, and the entries of W indicate if a pair of vertices are adjacent or not; W's entries are either 1 or 0, where the diagonal entries are 0. Hence, the goal is to minimize $\phi(Y)$ subject to $Y.L.Y^T$ [25].

The Scikit-learn tool implements LE using the spectral decomposition, which is the Eigen decomposition for the LE. The generated graph is considered as a 'discrete approximation' of the lower dimensional manifold in the high dimension. Optimizing the cost function confirms that data points are similar on the manifold, and are mapped closer in the lower dimension by maintaining local distances [23].

Algorithm 8: LE

Input: X $\in R^{n\times d}$
Output: Y $\in R^{n\times k}$

1: Develop a neighborhood graph $G^{'}$ of X using adjacency matrix W
2: Compute weights for the edges of $G^{'}$
3. Obtain new space Y by optimizing the cost function subject to its Eigen decomposition

10. Independent component analysis

Independent Component Analysis (ICA) is a linear, supervised FEA that generates new features that are statistically independent by reducing the higher order and second order dependencies in a given dataset [50]. What differentiates ICA from any other FEAs is that ICA searches for the feature that are non-Gaussian and statistically independent. For instance, PCA looks for the directions that vest represent the data, but ICA looks for such directions that are independent (most) from each other. First, ICA decomposes the data X as follows [25]:

$$X \rightarrow A.S \tag{15}$$

where A is a mixing matrix and S is the basis coefficient where the features are as much independent as possible. To obtain k dimensions from a dataset, ICA produces the data Y by selecting the top k independent components:

$$Y = A_k.S_k \tag{16}$$

The components can be recovered in different orders and with different scales. One constraint of ICA is that the components are commonly supposed to be non Gaussian, because the linear combination of two Gaussian variables is also Gaussian, which makes the separation an ill-posed problem [25].

ICA is a special circumstance of the "blind source separation" problem known in the signal processing field [51] where ICA deals with the separation of the original signals from a mixed data with little or no information about the source signals or the mixing process. Here, it is worth mentioning that the Scikit-learn tool uses FastICA to make ICA computationally and memory efficient.

Algorithm 9: ICA

Input: X $\in R^{n\times d}$
Output: Y $\in R^{n\times k}$

1: Decompose X to A and S.
2: Select top k independent components
3: Build Y by using k components

11. t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear, unsupervised and manifold-based FE method in which high dimension data is mapped to low dimension (typically 2 or 3 dimensions) while preserving the significant structure of the original data [52]. Primarily, t-SNE is used for data exploration and visualization. In simpler words, t-SNE provides the

intuition of how data is arranged in the high dimensional space. Despite having strong performance, FEAs are not often successful in visualizing high-dimensional data, and most of them are not capable of preserving both the local and global structure of data [52]. In this scenario, t-SNE becomes handy for visualizing high-dimensional data by retaining the data's significant structure.

t-SNE first applies Stochastic Neighbor Embedding (SNE) to the dataset, which converts the high dimensional Euclidean distances into conditional probabilities representing similarities for every pair of data [53]. The similarity of data x_a to data x_b is represented by the conditional probability $p_{a|b}$, defined in the equation below [52]:

$$p_{a|b} = \frac{exp\frac{-\|x_b - x_a\|^2}{2\sigma^2}}{\sum_{a \neq k} \frac{-\|x_k - x_a\|^2}{2\sigma^2}} \tag{17}$$

Equation (17) measures how close a data point x_a is from another data point x_b considering a Gaussian distribution around x_b with a given variance σ^2. This variance differs for each data and is chosen in a way that data in dense area have smaller variance than data in sparse area [52].

Afterward, instead of utilizing the Gaussian distribution, a "Student t-distribution" with one degree of freedom, similar to Cauchy distribution, is used to obtain the second set of probabilities ($Q_{a|b}$) in the low dimension space [54]. If the high dimension data x_a and x_b are correctly mapped to low dimension data y_a and y_b, then the similarity between $P_{a|b}$ and $Q_{a|b}$ becomes equal. Therefore, t-SNE minimizes the difference between these two probabilities from the low dimension to high dimensional spaces. This difference is measured by optimizing the cost function (ϕ) of the sum of Kullback–Leibler divergence as shown below [54]:

$$\phi = \sum_a \sum_b P_{a|b} log \frac{P_{a|b}}{Q_{a|b}} \tag{18}$$

Algorithm 10: t-SNE

Input: $X \in R^{n \times d}$
Output: $Y \in R^{n \times k}$

1: Apply SNE to X to calculate the conditional probabilities $P_{a|b}$ and $Q_{a|b}$
2: Map X to Y by minimizing the difference between $P_{a|b}$ and $Q_{a|b}$ based on the cost function ϕ

12. Summary and comparison

Tables 2 and 3 summarizes the key concepts of FEAs. For instance, SVD is unsupervised, linear, iterative, random projection-based, uses the SVD theorem to conduct the feature transformation by minimizing the reconstruction error, and has two hyper-parameters: the number of PCs and the number of iterations. ISOMAP is unsupervised, nonlinear, non-iterative, manifold-based, uses the neighboring graph to compute the pairwise geodesic distances, and has two hyper-parameters: the number of PCs and the number of nearest neighbors.

For the methods examined in our paper, we observe that the manifold-based are nonlinear and the random projection-based are linear. Iterative methods are more time consuming as demonstrated by the time complexity column. LDA is the only FEA that does not possess tuning parameters.

13. First case study

We assess the FEAs' performance using the ECG200 dataset that comes with 96 real-valued features. The dataset was formatted in [55] and made public in the MIT-BIH Arrhythmia Database CD-ROM and its affiliated online repository PhysioNet [55]. It was analyzed by domain experts and annotated into Normal or Abnormal (Myocardial Infarction) heartbeat. However, it contains only 200 observations in which 133 are Normal and 67 are Abnormal. Since our dataset is relatively small, FEAs are needed to improve the MLA's performance. To conduct the experimental analysis, we employ the Scikit-learn toolkit from the Anaconda distribution, such that we can take advantages of the existing FEA libraries.

13.1. Feature scaling

In Fig. 2, we plot the histogram of some features chosen randomly to visualize their distribution. We observe that the features have different value distributions. Feature #C55 has a skewed left histogram where the mean is smaller than the median. In this case, the presence of relatively smaller values brings the mean down for this feature. On the other hand, if a histogram is skewed at the right, like Feature #C92, the mean is larger than the median. This situation occurs when skewed-right data have some high values that push the mean upward. Symmetric data, like Feature #C20, have almost the same shape on both sides. Lastly, Feature #C43 shows a multi-modal histogram, which implies that this feature has two or more peaks (local maxima). Therefore, this dataset has a good combination of features that come with different distributions.

Before applying FEAs to the ECG200 dataset, we first standardize the features since they have different distributions: the maximum value is 4.1991449 and the minimum value is -3.0144511. Some FEAs, like PCA, KPCA, and most of the MLAs are sensitive to data scaling. We therefore apply the StandardScaler of Scikit-learn to scale all the features such that their standard deviation is one and mean value is zero, with a range of $[-1, 1]$.

13.2. Optimal number of PCs

An important question that arises is how to determine the optimal number of PCs that will lead to the highest accuracy. Regarding PCA and KPCA, we believe the higher the variance, the better the data representation since PCA seeks to maximize the variance [27] in order to acquire unique information from the data. From Fig. 3, we can see that 100% "Cumulative Explained Variance" was obtained with 65+ features. This metric denotes the summation of variances of the new features, and gives the percentage of the variance of the first 65 PCs. Since 65 is the first number that produced 100% cumulative variance, hence we choose it as the optimal number of PCs.

As mentioned earlier, for the LDA method, the number of PCs should be less than the number of classes-1. Since we have only two classes in our dataset, the number of PCs should be equal to 1. For the remaining FEAs, as shown in Algorithm 11, we compute the predictive accuracy to decide on the best number of PCs. By varying the number of PC from 1 to the maximum dimensions (here 96), we look for the number that offers the best performance. We select the SVM algorithm to evaluate the FEA performance due to wide spread usage and high performance [56, 57]. We train SVM on the 96 transformed datasets using the stratified 10-fold Cross-Validation, where, the final accuracy of the model is the average accuracy over ten runs.

Table 2
Conceptual Comparison of FEAs.

	Goal	Supervision	Linearity	Data Charac.	Iteration	Topology
PCA	Maximize variance	Unsuper.	Linear	Handle skewed data	No	Random Projection
LDA	Maximize class separation	Super.	Linear	Perform better with uniformly distributed data	No	Random Projection
MDS	Preserve Euclidean pairwise distances	Unsuper.	Nonlinear	Perform better with relational data	Yes	Manifold
KPCA	Linearly separate data	Unsuper.	Nonlinear	Perform better with nonlinear data	No	Manifold
SVD	Minimize reconstruction error	Unsuper.	Linear	Handle sparse data	Yes	Random Projection
LLE	Preserve local properties	Unsuper.	Nonlinear	Handle sparse and nonlinear data	Yes	Manifold
ISOMAP	Preserve geodesic pairwise distances	Unsuper.	Nonlinear	Handle sparse and noisy data	No	Manifold
LE	Preserve local distances	Unsuper.	Nonlinear	Handle noisy data	No	Manifold
ICA	Maximize statistical independence	Super.	Linear	Handle higher order statistics	Yes	Random Projection
t-SNE	Preserve local structure	Unsuper.	Nonlinear	Perform better with numeric data	Yes	Manifold

Table 3
Conceptual Comparison of FEAs (cont')

	Tuning of Parameters	Computational Complexity	Weakness	Transformation
PCA	No. of components	$O(d^2n + n^3)$	Limited to linear projection	Eigen decomposition
LDA	None	$O(d^2n)$, n>d; $O(d^3)$, d>n	Suffer from class singularity issue	Class separability by scatter matrices
MDS	No. of components and iterations	$O(n^3)$	Require high computation and memory to calculate dissimilarity matrix	Dissimilarity matrix by Euclidean distance
KPCA	No. of components, Kernel	$O(n^3)$	Require high training time	Kernel mapping
SVD	No of components and iterations	$O(d^2n + n^3)$	Not suitable for nonlinear data	SVD theorem
LLE	No. of components, iterations and nearest neighbors	$O(dlog(c)nlog(n))$ $+O(dnc^3)+$ $O(kn^2)$	Consume memory	Neighborhood graph, reconstruction weights
ISOMAP	No. of components and nearest neighbors	$O[dlog(c)nlog(n)]+$ $O[n^2(c + log(n))]+$ $O(kn^2)$	Suffer from topological instability	Neighborhood graph, geodesic distance
LE	No. of components and nearest neighbors	$O(dlog(c)nlog(n))$ $+O(dnc^3)+$ $O(kn^2)$	May produce disconnected neighborhood graph	Neighborhood graph, weight update of edges
ICA	No. of components and iterations, Tolerance	$O[2di(d+1)n]$	Require high training time	Non-Gaussian distribution
t-SNE	No. of components and iterations	$O(n^2)$	Provide only 2 or 3 features	Conditional probabilities

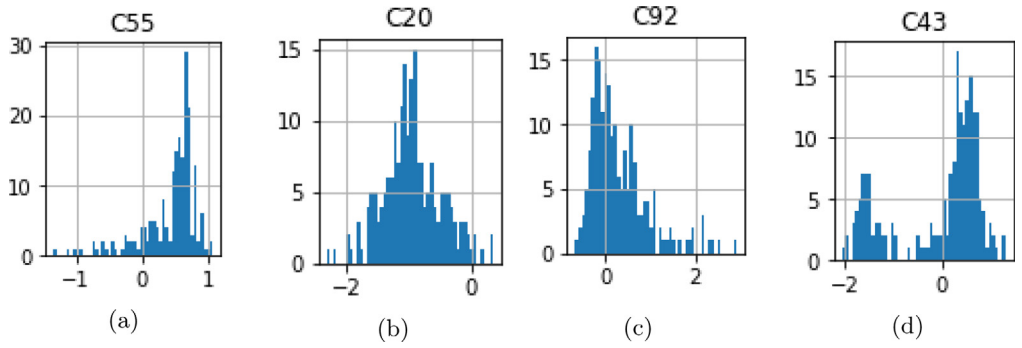

Fig. 2. (a) Skewed-left histogram, (b) Symmetric histogram, (c) Skewed-right histogram, and (d) Multimodal histogram.

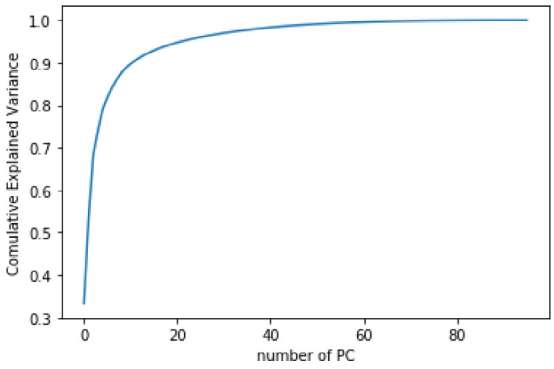

Fig. 3. Optimal PC Number for PCA and KPCA.

Algorithm 11: Searching Optimal Classification Accuracy

Input: Training Reduced Dataset
Output: Optimal Number of PCs
1: $accuracy_0$ = 0
2: **for** $j = 1$ *to* d **do**
  2.1: $model_d$ = pipeline (FEA, Classifier)
  2.2: $accuracy_j$ = averageAccuracy($model_j$, Stratified 10-Fold CV, 10 runs)
  2.3: **if** $Accuracy_j > Accuracy_{d-1}$ **then**
    2.3.1: optimalPC = j
end

13.3. Quality of transformed data

To evaluate the quality of the produced datasets, we compare the correlation, in terms of p-value and power analysis between the class column and the first feature of the original dataset, and the class column and the first component of the new dataset. We still experiment with LDA and t-SNE, but we exclude them when comparing FEAs because LDA reduces the input space to one dimension only, and t-SNE to two dimensions for this dataset (the performance is better with two features than with three). The p-value quantifies the statistical significance of a test and is assessed with a fixed significance level. If the obtained p-value is less than this fixed level, the outcome is statistically significant [58]. In our experiment, a low p-value means a higher chance of having better data [58]. Power is the likelihood of taking a precise decision to invalidate the null hypothesis if it is false in a test. Hence, the higher the power value, the more robust is the data [59].

Most of nonlinear methods have tuning hyper-parameters. We optimize the number of iterations for MDS to 300, LLE to 100, and SVD to 5. RBF kernel offered better accuracy than Linear kernel for KPCA. For ICA, we tune the number of iteration to 500 and set the tolerance to 0.5. In Table 4, we optimize the number of nearest neighbors (c). We choose the small range of [2, 5] since the ECG200 dataset has only 200 data points. The best c value is 4, 3 and 4 for LLE, ISOMAP and LE respectively. However, for LE, we got the same p-value and power for c equals to 4 and 5. We choose 4 over 5 since it requires less computation. Table 5 presents the p-value and power analysis of the original and transformed datasets. After applying FEAs, we obtained a lower p-value and a higher power analysis than the values produced with the original dataset. Thus, the new reduced datasets are of better quality than the original dataset. For both p-value and power, we got the best results with MDS and KPCA, followed by LLE. There is a huge gap between the p-value of the original dataset (0.003) and the p-value of the datasets transformed by KPCA and MDS ($4.2e^{-22}$). We may note that LE shows the worst performance for this dataset.

13.4. Classification performance

We present the training performance of the original and the nine transformed datasets in Table 6. We train the SVM classifier with RBF kernel on the new feature spaces. Table 6 exposes the F1-score and run-time in milliseconds (ms) before and after applying the FEAs. On the original data with 96 features, SVM returned an F1-score of 89.66% and a speed of 200 ms. It is clear that the classifier performed better on the new reduced datasets. For the comparison, we do not consider the LDA's accuracy as it returns only one feature for this dataset. The highest performance is attained with KPCA followed by ICA and PCA/SVD. There is a gap of 3.43% between the original and the best reduced feature space (with KPCA), which is significant in the medical diagnosis field. However, ISOMAP provided the lowest accuracy, which is expected since it has the highest number of components (81) and classifiers are sensitive to the large number of features. We can also see that with PCA, the classification is the fastest with only 3.6 ms, followed by SVD with 7 ms. The classification with MDS takes the maximum time of 122.8 ms.

In Table 7, we rank both the data quality (correlation) and classification performance. For both cases, KPCA is the top FEA. Besides, the linear and random projection-based methods, like SVD and PCA, ranked fourth for data quality, and third for classification. On the other hand, the supervised-linear method ICA obtained the seventh position for data quality; however, it was ranked second for classification performance. The gap of F-score between KPCA and ICA/PCA/SVD are 0.32% and 0.98% respectively. Moreover, there is a significant gap between KPCA and ICA/PCA/SVD in terms of data quality. We can make three main observations: (1) the nonlinear FEAs performed better than the linear FEAs; (2), manifold-based FEAs do better than random projection-based methods; (3), unsupervised FEAs behaved better than supervised FEA.

Table 4
P-value and power optimization.

No. of Neighbors	LLE		ISOMAP		LE	
	P-val	Power	P-val	Power	P-val	Power
5	$8.74e^{-07}$	0.9987808	0.005974	0.788329	0.001318	0.897834
4	$2.35e^{-13}$	**1**	0.00002	0.997842	**0.001318**	**0.897834**
3	$3.63e^{-11}$	0.999999	$7.829e^{-08}$	**0.999751**	0.025211	0.612
2	0.02300	0.62569	0.0115	0.359821	0.238	0.218

Table 5
Best *P*-value and power analysis of reduced datasets.

Dataset	Dimension	P-value	Power Analysis
Original	96	0.003	0.829
PCA	65	$1.61e^{-11}$	1
LDA	1	$7.7e^{-76}$	1
MDS	**70**	$4.2e^{-22}$	**1**
KPCA	**65**	$4.2e^{-22}$	**1**
SVD	55	$6.9e^{-12}$	1
LLE	70	$2.35e^{-13}$	1
ISOMAP	81	$7.829e^{-08}$	0.999751
LE	71	0.001318	0.8978
ICA	26	0.000008	0.994581
t-SNE	2	$5.45e^{-20}$	1

Table 6
Classification performance of original and reduced datasets.

Dataset	Dimension	Kernel SVM	
		F1-Score	Run-time (ms)
Original	96	0.8966	200.1
PCA	**65**	0.9205	**3.6**
LDA	1	0.9885	0.01
MDS	70	0.9168	122.8
KPCA	**65**	**0.9303**	46.3
SVD	55	0.9205	7
LLE	70	0.9163	42.1
ISOMAP	81	0.9022	49.7
LE	71	0.9181	53.0
ICA	26	0.9271	13.0
t-SNE	2	0.8928	156.5

Table 7
Ranking of FEAs.

FE	NonLin/Lin	M/RP	S/U	Rank for Data Quality	Rank for Classification Performance
PCA	Lin	RP	U	4	3
MDS	NonLin	M	U	1	5
KPCA	NonLin	M	U	1	1
SVD	Lin	RP	U	3	3
LLE	NonLin	M	U	2	6
Isomap	NonLin	M	U	5	7
LE	NonLin	M	U	7	4
ICA	Lin	RP	S	6	2

13.5. Data plotting

In the following Fig. 4, we show the scatter plot of the original and the reduced dataset that we obtain using KPCA for better visualization and understanding before using it for the classification purpose. Here, it is evident that the original dataset (a) is a nonlinear dataset and after applying KPCA, the reduced dataset (b) is quite well separated into two classes.

14. Second case study

The benchmark Arcene dataset was developed by merging three mass-spectrometry datasets. At a given mass value, the features determines the profusion of proteins in human sera. Based on these features, cancer patients can be distinguished from healthy ones. In addition to the 7,000 original features, the dataset's creator added 3,000 probe features (do not have any predictive power) to make the dataset ideal for feature selection/extraction tasks [60]. This dataset, made public in the UCI repository in 2008, has in total 10,000 features but only 200 samples, and is almost equally balanced. We normalize all the features to the range of $[-1, 1]$ as done previously for the ECG200 dataset.

14.1. Classification performance

To search for the optimal components number for each FEA, we follow the same methodology presented in Section 13.2 and Algorithm 11. For instance, PCA and KPCA reduce the dimensionality from 10,000 to 200 (the maximum reduced feature number among all FEAs), which implies that 100% "Cumulative Explained Variance" was obtained with the 200th PC. Table 8 reports the optimal PC number for each FEA. ISOMAP reduces the dimensionality from 10,000 to 57 (the minimum reduced number). In all cases, FEAs lowered the number of features remarkably. The new feature spaces will solve the issue of data storage and visualization, and will increase the accuracy of classifiers.

We train a RBF+SVM classifier on the various reduced datasets, and report the performance in Table 8. LE (unsupervised, nonlinear and manifold-based) and SVD (unsupervised, linear and random projection-based) provided the best F1-score of 84.82% with 96 new features and 84.81% with 76 new features, respectively. SVD and LE's performance are similar, but SVD is almost twice faster than LE. In contrast, the least performing method is again ISOMAP (nonlinear, unsupervised and manifold-based) which returned an F1-score of 67.65%. Regarding the time-efficiency, KPCA is the fastest method with only 191.1 ms, and MDS required the maximum time of 1778.7 ms. Although LE offered the best accuracy, it is however ranked fourth in terms of speed (606.4 ms). The original dataset generated a low accuracy of 63%, and a very high run-time of 4856.2 ms. Here, the accuracy gap between the original and the best-reduced dataset is 21.82%, which means that the accuracy greatly increased after adopting FEA. Besides, with KPCA, the classifier is almost 25 times faster than SVM without FEA. Again, t-SNE provided better F1-score with two dimensions than with three.

14.2. Data plotting

Moreover, we plot the original Arcene dataset, and the optimally reduced dataset in Fig. 5. Fig. 5(a) shows that all the data are inclined to the vertical and horizontal axes and scattered. In Fig. 5(b), all the data are deviated from the axes and well distributed in the feature space after applying LE.

15. Third case study

This time we deal with a multi-class dataset called 'Human Activity Recognition (HAR) Using Smartphones', made public in 2012 in the UCI repository [61]. The dataset contains six classes

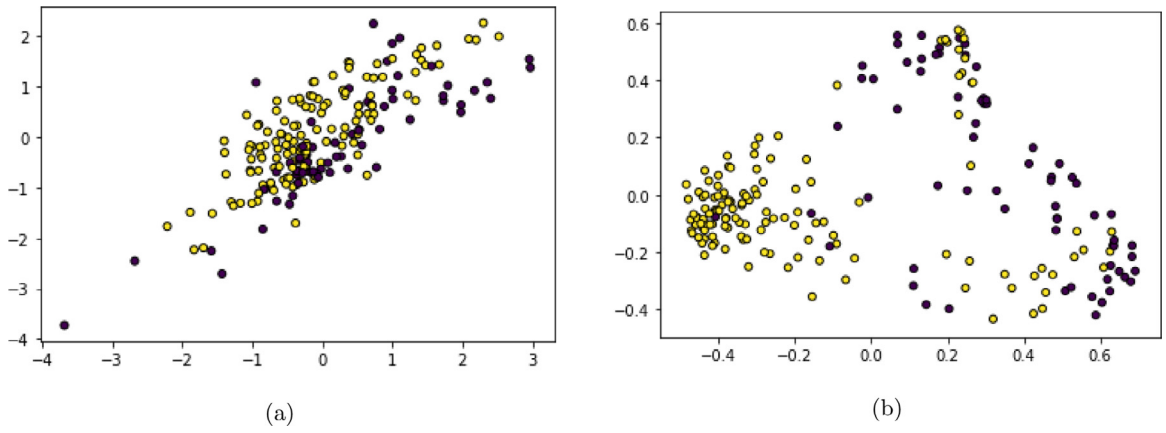

Fig. 4. Scatter Plot of (a) Original ECG200 Dataset and (b) Reduced ECG200 Dataset Using KPCA.

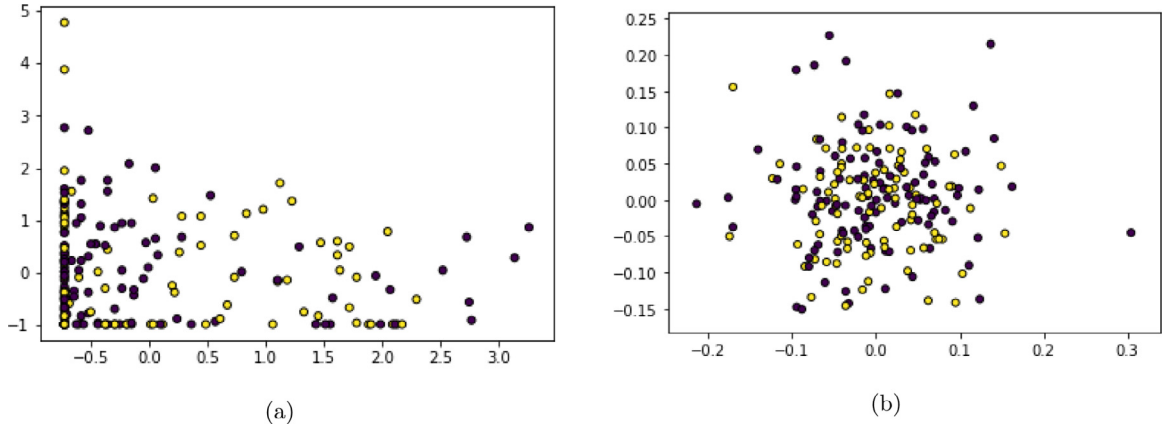

Fig. 5. Scatter Plot of (a) the Original Arcene Dataset and (b) the Reduced Arcene Dataset Using LE.

Table 8
Classification performance of original and reduced datasets.

Dataset	Dimension	F1-score	Run-time (ms)
Original	10,000	0.630	4856.2
PCA	200	0.8019	343.7
LDA	1	0.8181	889.5
MDS	186	0.8118	1778.7
KPCA	**200**	0.7162	**191.1**
SVD	**76**	**0.8481**	368.7
LLE	102	0.7409	1115.7
Isomap	57	0.6765	996.0
LE	**96**	**0.8482**	606.4
ICA	70	0.8036	319.9
t-SNE	2	0.7997	1723.7

to indicate six activities (sitting, walking, standing, walking_upstairs, walking_downstairs, and laying), 10299 samples and 561 features. The 6 classes have almost the same distributions. The features were already scaled to the range of $[-1, 1]$. We apply the selected FEAs and evaluate their effectiveness in the multi-class context. To obtain the optimal feature space for this dataset, we follow the same strategy described earlier in Section 12.

15.1. Classification performance

From Table 9, we notice that the SVM classifier without any dimensionality reduction provided a high F-score of 95.05% due to the high data quality. The best F-score (98.48%) obtained after applying FEAs is with LDA (supervised, linear and random-projection-based) with a reduced feature space of only five components. Thanks to LDA, the new dataset necessitates a lot less storage and will significantly facilitate the data plotting and analysis. The second-best accuracy of 96.79% is obtained by the unsupervised, nonlinear and manifold-based KPCA with 300 PCs, and the third top classifier is ICA (supervised, random projection-based and linear) that returned 95.75% with 226 components. The least performing classifier is MDS with 92.5% accuracy using 478 new features. On the other hand, SVM without FEA required the high processing time of 156,090 ms, unlike LDA that needed only 1,244.8 ms. MDS took the maximum time of 126,865.8 ms (102 times slower than LDA). We can also see that t-SNE and MDS were ineffective for the 6-class data setting as they provided an accuracy lower than with the original data.

15.2. Data plotting

Lastly, we plot the original and the optimal reduced datasets in Fig. 6. It is clear that the original dataset is nonlinear and the new dataset obtained by LA facilitates the visualization and separation of the six classes.

16. FEA comparison for all three case studies

Table 10 reports the top three FEAs for the three investigated datasets. As observed, SVD which is unsupervised, linear, and random projection-based, performed quite well on the ECG200 and

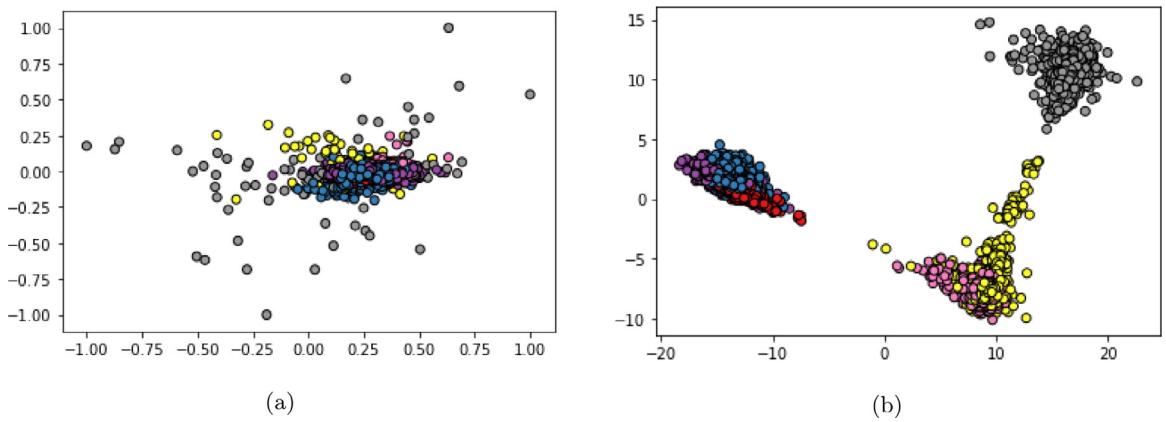

Fig. 6. Scatter Plot of (a) the Original and (b) Reduced HAR Using Smartphones Dataset Using LDA.

Table 9
Classification performance of original and reduced datasets.

Dataset	Dimensions	F-score	Run-time (ms)
Original	561	0.9505	156090.0
PCA	300	0.9557	63712.2
LDA	**5**	**0.9848**	**1244.8**
MDS	478	0.9250	126865.8
KPCA	300	0.9679	55614.8
SVD	243	0.9562	54674.9
LLE	530	0.9527	143393.8
Isomap	480	0.9505	100911.9
LE	200	0.9536	115238.2
ICA	226	0.9575	122646.6
t-SNE	2	0.6786	110284.7

Table 10
Dataset comparison with three top FEAs.

ECG200 Dataset		Arcene Dataset		HAR Dataset	
Top FEA	F1-score	Top FEA	F1-score	Top FEA	F1-score
KPCA	0.9303	LE	0.8482	LDA	0.9848
ICA	0.9271	**SVD**	0.8481	**KPCA**	0.9679
PCA, **SVD**	0.9205	MDS	0.8118	**ICA**	0.9575

Arcene datasets. Also, KPCA has done very well on ECG200 and HAR datasets. Moreover, there is a significant gap between the performances of the three datasets, as FEAs depend on the nature and quality of the data. According to the case studies, unsupervised, nonlinear and manifold-based FEAs outperform linear, supervised and random projection-based FEAs on two datasets, while the supervised, linear and random projection-based FEAs work best on the last dataset. Hence, there is no clear-cut winner, and empirical analysis is essential to determine the optimal FEA for a dataset.

17. Conclusion and future work

Training datasets should be high-quality and do not possess insignificant and redundant features; otherwise, the predictions will be unreliable. In this article, we first theoretically described and compared multiple FEAs belonging to different categories. We then empirically evaluated and compared the performance of FEAs on different datasets in both binary and multi-class settings. We assessed the various transformed feature spaces' quality using correlation, classification, and run-time metrics. Based on the experimental results, we can make two main observations: (1) the data quality and classification accuracy increased after applying FEAs, where in some cases, the increase is remarkable; (2) in most cases, nonlinear FEAs performed better than linear FEAs and manifold-based outperformed random projection-based FEAs. Additionally, in the multi-class context, supervised FEAs behaved better than unsupervised FEAs, where one unsupervised FEA was ineffective for this setting. For future research directions, we will explore the performance of deep learning with FEAs on high dimensional datasets. Additionally, we will explore FEAs on more complex datasets, such as multi-dimensional time-series and multi-label data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] F. Anowar, S. Sadaoui, Incremental neural-network learning for big fraud data, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2020, pp. 3551–3557.
[2] F. Anowar, S. Sadaoui, Incremental learning framework for real-world fraud detection environment, Comput. Intell. (2021) 1–22, http://dx.doi.org/10.1111/coin.12434.
[3] V. Spruyt, The curse of dimensionality in classification, Comput. Vision Dummies 21 (3) (2014) 35–40.
[4] L. Van Der Maaten, E. Postma, J. Van den Herik, Dimensionality reduction: A comparative review, J. Mach. Learn. Res. 10 (66–71) (2009) 13.
[5] P. Jindal, D. Kumar, A review on dimensionality reduction techniques, Int. J. Comput. Appl. 173 (2) (2017) 42–46.
[6] M. Verleysen, D. François, The curse of dimensionality in data mining and time series prediction, in: International Work-Conference on Artificial Neural Networks, Springer, 2005, pp. 758–770.
[7] D.M. Hawkins, The problem of overfitting, J. Chem. Inf. Comput. Sci. 44 (1) (2004) 1–12.
[8] S. Abe, Feature selection and extraction, in: Support Vector Machines for Pattern Classification, in: Advances in Pattern Recognition, Springer, 2010, pp. 331–341.
[9] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, Z. Chen, Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing, IEEE Trans. Knowl. Data Eng. 18 (3) (2006) 320–333.
[10] G. Chao, Y. Luo, W. Ding, Recent advances in supervised dimension reduction: A survey, Mach. Learn. Knowl. Extr. 1 (1) (2019) 341–358.
[11] A. Gracia, S. González, V. Robles, E. Menasalvas, A methodology to compare dimensionality reduction algorithms in terms of loss of quality, Inform. Sci. 270 (2014) 1–27.
[12] S. Khalid, T. Khalil, S. Nasreen, A survey of feature selection and feature extraction techniques in machine learning, in: 2014 Science and Information Conference, IEEE, 2014, pp. 372–378.
[13] P. Joshi, What is manifold learning? 2014, https://prateekvjoshi.com/2014/06/21/what-is-manifold-learning/.
[14] D. Garrett, D.A. Peterson, C.W. Anderson, M.H. Thaut, Comparison of linear, nonlinear, and feature selection methods for EEG signal classification, IEEE Trans. Neural Syst. Rehabil. Eng. 11 (2) (2003) 141–144.

[15] M.C. Yeh, I.H. Lee, G. Wu, Y. Wu, E.Y. Chang, Manifold learning, a promised land or work in progress? in: 2005 IEEE International Conference on Multimedia and Expo, IEEE, 2005, pp. 1154–1157.
[16] H. Xie, J. Li, Q. Zhang, Y. Wang, Comparison among dimensionality reduction techniques based on Random Projection for cancer classification, Comput. Biol. Chem. 65 (2016) 165–172.
[17] J.P. Cunningham, Z. Ghahramani, Linear dimensionality reduction: Survey, insights, and generalizations, J. Mach. Learn. Res. 16 (1) (2015) 2859–2900.
[18] M. Sedlmair, M. Brehmer, S. Ingram, T. Munzner, Dimensionality Reduction in the Wild: Gaps and Guidance, Tech. Rep. TR-2012-03, Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada, 2012, pp. 1–10.
[19] V.D. Silva, J.B. Tenenbaum, Global versus local methods in nonlinear dimensionality reduction, in: Advances in Neural Information Processing Systems, 2003, pp. 721–728.
[20] V. Nasteski, An overview of the supervised machine learning methods, HORIZONS. B 4 (2017) 51–62.
[21] S. Shams, Random projection in dimensionality reduction, 2020, https://machinelearningmedium.com/2017/07/28/random-projection-in-dimensionality-reduction/.
[22] M. Nabil, Random projection and its applications, 2017, pp. 1–6, arXiv preprint arXiv:1710.03163.
[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[24] A. Ansari, A.B. Shafie, A.B. Said, Independent component analysis using random projection for data pre-processing, Int. J. Comput. Sci. Issues 9 (3) (2012) 200.
[25] C.O.S. Sorzano, J. Vargas, A.P. Montano, A survey of dimensionality reduction techniques, 2014, arXiv preprint arXiv:1403.2877.
[26] B. Ghojogh, M.N. Samad, S.A. Mashhadi, T. Kapoor, W. Ali, F. Karray, M. Crowley, Feature selection and feature extraction in pattern analysis: A literature review, 2019, arXiv preprint arXiv:1905.02845.
[27] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscipl. Rev.: Comput. Statist. 2 (4) (2010) 433–459.
[28] H. Abdi, The eigen-decomposition: Eigenvalues and eigenvectors, in: Encyclopedia of Measurement and Statistics, Sage Thousand Oaks, CA, 2007, pp. 304–308.
[29] S. Karamizadeh, S.M. Abdullah, A.A. Manaf, M. Zamani, A. Hooman, An overview of principal component analysis, J. Signal Inf. Process. 4 (3B) (2013) 173.
[30] I.T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments, Phil. Trans. R. Soc. A 374 (2065) (2016) 20150202.
[31] H. Hoffmann, Kernel PCA for novelty detection, Pattern Recognit. 40 (3) (2007) 863–874.
[32] N. Kwak, Nonlinear projection trick in kernel methods: An alternative to the kernel trick, IEEE Trans. Neural Netw. Learn. Syst. 24 (12) (2013) 2113–2119.
[33] G. Baudat, F. Anouar, Kernel-based methods and function approximation, in: IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222), Vol. 2, IEEE, 2001, pp. 1244–1249.
[34] S. Raschka, Linear discriminant analysis, 2014, https://sebastianraschka.com/Articles/2014_python_lda.html.
[35] D. Granato, G. Ares, Mathematical and Statistical Methods in Food Science and Technology, John Wiley & Sons, 2014.
[36] M.A. Cox, T.F. Cox, Multidimensional scaling, in: Handbook of Data Visualization, Springer, 2008, pp. 315–347.
[37] S.T. Muller, Distance, similarity, and multidimensional scaling, 2019, https://pages.mtu.edu/~shanem/psy5220/daily/Day16/MDS.html.
[38] J. Leskovec, A. Rajaraman, J. Ullman, Dimensionality reduction, in: Mining of Massive Datasets, 2014, pp. 415–447.
[39] A.G. Akritas, G.I. Malaschonok, Applications of singular-value decomposition (SVD), Math. Comput. Simul. 67 (1–2) (2004) 15–31.
[40] H. Abdi, Singular value decomposition (SVD) and generalized singular value decomposition, in: Encyclopedia of Measurement and Statistics, 2007, pp. 907–912.
[41] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[42] L.K. Saul, S.T. Roweis, An introduction to locally linear embedding, 2000, unpublished. Available at: http://www.cs.toronto.edu/~roweis/lle/publications.html.
[43] E. Krivov, M. Belyaev, Dimensionality reduction with isomap algorithm for EEG covariance matrices, in: 2016 4th International Winter Conference on Brain-Computer Interface, BCI, IEEE, 2016, pp. 1–4.
[44] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[45] M. Balasubramanian, E.L. Schwartz, J.B. Tenenbaum, V. de Silva, J.C. Langford, The isomap algorithm and topological stability, Science 295 (5552) (2002) 7.
[46] J.B. Tenenbaum, Mapping a manifold of perceptual observations, in: Advances in Neural Information Processing Systems, 1998, pp. 682–688.
[47] J.A. Lee, M. Verleysen, Nonlinear dimensionality reduction of data manifolds with essential loops, Neurocomputing 67 (2005) 29–53.
[48] M. Niskanen, O. Silvén, Comparison of dimensionality reduction methods for wood surface inspection, in: Sixth International Conference on Quality Control By Artificial Vision, Vol. 5132, International Society for Optics and Photonics, 2003, pp. 178–188.
[49] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[50] L. Cao, K.S. Chua, W. Chong, H. Lee, Q. Gu, A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine, Neurocomputing 55 (1–2) (2003) 321–336.
[51] A. Tharwat, Independent component analysis: An introduction, Appl. Comput. Inform. (2018) 1–15.
[52] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (1) (2008) 2579–2605.
[53] B.M. Devassy, S. George, Dimensionality reduction and visualisation of hyperspectral ink data using t-SNE, Forensic Sci. Int. 1 (1) (2020) 1–9.
[54] F.H.M. Oliveira, A.R. Machado, A.O. Andrade, On the use of t-distributed stochastic neighbor embedding for data visualization and classification of individuals with Parkinson's disease, Comput. Math. Methods Med. 1 (1) (2018) 1–18.
[55] R. Olszewski, Generalized feature extraction for structural pattern recognition in time-series data, 2001, http://www.timeseriesclassification.com/description.php?Dataset=ECG200&fbclid=IwAR0h4KYdTifENafVDvXojXt37avCw2QvHLsMh1Xq2QOnkhtDzHbfOsi-ELwm.
[56] F. Anowar, S. Sadaoui, Detection of auction fraud in commercial sites, J. Theor. Appl. Electron. Commer. Res. 15 (1) (2020) 81–98.
[57] F. Anowar, S. Sadaoui, M. Mouhoub, Auction fraud classification based on clustering and sampling techniques, in: The 17th IEEE International Conference on Machine Learning and Applications, ICMLA, IEEE, 2018, pp. 366–371.
[58] B. Vidgen, T. Yasseri, P-values: Misunderstood and misused, 2016, https://www.frontiersin.org/articles/10.3389/fphy.2016.00006/full#:~:text=The%20p%2Dvalue%20quantifies%20the,result%20is%20considered%20statistically%20significant.
[59] L. Theme, What is power? 2017, https://www.statisticsteacher.org/2017/09/15/what-is-power/.
[60] I. Guyon, S. Gunn, A. Ben-Hur, G. Dror, Result analysis of the NIPS 2003 feature selection challenge, in: Advances in Neural Information Processing Systems, 2005, pp. 545–552.
[61] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Vol. 3, 2013, pp. 437–442.