Information Fusion 54 (2020) 44–60

Contents lists available at ScienceDirect

Information Fusion

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

Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines

Francisco J. Pulgar*, Francisco Charte, Antonio J. Rivera, María J. del Jesus

Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), Computer Science Dpt., University of Jaén, Jaén 23071, Spain

ARTICLE INFO

Keywords:
Classification
Deep learning
Autoencoders
Dimensionality reduction
Feature fusion

ABSTRACT

Classifying data patterns is one of the most recurrent applications in machine learning. The number of input features influences the predictive performance of many classification models. Most classifiers work with high-dimensional spaces. Therefore, there is a great interest in facing the task of reducing the input space. Manifold learning has been shown to perform better than classical dimensionality reduction approaches, such as Principal Component Analysis and Linear Discriminant Analysis. In this sense, Autoencoders (AEs) provide an automated way of performing feature fusion, finding the best manifold to reconstruct the data. There are several models and architectures of AEs. For this reason, in this study an exhaustive analysis of the predictive performance of different AEs models with a large number of datasets is proposed, aiming to provide a set of useful guidelines. These will allow users to choose the appropriate AE model for each case, depending on data traits and the classifier to be used. A thorough empirical analysis is conducted including four AE models, four classification paradigms and a group of datasets with a variety of traits. A convenient set of rules to follow is obtained as a result.

1. Introduction

Machine learning is one of the most widely studied fields of artificial intelligence, due to its extensive application in solving real problems. The generalization of behaviors from a series of training instances is the main objective of the methods developed in this field [1]. Some of the main applications of these algorithms are classification, regression and clustering [2,3]. In particular, classification is one of the most well-known and developed tasks. The main purpose of a classifier is to establish a prediction for new patterns, based on the information provided by training instances. To meet this challenge, different proposals have emerged over time that can be categorized into different methodologies according to their structure and function [4].

There are very diverse approaches among the existing methodologies that handle the classification task. Some of the most commonly used that offer the best results are: Instance-based learning (IBL), which bases the prediction on the information provided by the training data without carrying out a training process [5]; Support Vector Machines (SVMs), which generate a distribution of the examples in the space with the objective of establishing groupings of related instances [6]; Artificial Neural Networks (ANNs), which present an architecture based on the human brain, and whose objective is to establish relationships between the data through an internal training process [7]; And Decision trees (DTs), which are tree-based models, whose architecture allows different features of the data to be evaluated in order to obtain the greatest separability between classes [8].

The approaches included in the previous methodologies have difficulties when working with data that has certain characteristics. In general, these methods are created to work with real data. Therefore, they must take the traits of the input data features into account in order to give the best possible response. One of these characteristics is the high dimensionality of the data. At present, the information captured and stored is growing due to the increase in the generation, reception and storage mechanisms. In this context, the data sets generated have a growing number of features, so the classification algorithms must adapt to this fact. Traditional methodologies decrease their predictive performance when working with data that has a large number of features. This is mainly due to the curse of dimensionality [9,10].

In this context, different proposals have emerged for handling the reduction of dimensionality. The fundamental objective of these methods is to mitigate the effects of high dimensionality on the predictive performance obtained with different classifiers [11,12]. Initially, the process consisted of manual evaluation by an expert who selected the most relevant features. This task was automated in subsequent years, along with the very first methods of feature selection [13]. Some of the best-known algorithms are: Linear Discriminant Analysis (LDA) [14], Principal Component Analysis (PCA) [15], Isometric feature mapping (ISOMAP) [16] and Locally Linear Embedding (LLE) [17]. Over time,

* Corresponding author.
E-mail address: fpulgar@ujaen.es (F.J. Pulgar).

https://doi.org/10.1016/j.inffus.2019.07.004
Received 31 January 2019; Received in revised form 17 July 2019; Accepted 17 July 2019
Available online 18 July 2019

new approaches have emerged to reduce input space dimensionality that improve on aspects of previous tasks or use different methodologies to deal with the problem [18–21].

Recently, the term Feature Fusion has emerged, due to the need to work with multimedia data using machine learning algorithms. The objective of this type of method is to combine features with the aim of eliminating non-relevant information [22–24]. In order to carry out this task, proposals based on deep learning (DL) have emerged that offer effective performance [25]. These good results in certain fields have led to a rise in the use of DL methods [26,27]. Specifically, one of the most suitable models for the feature fusion task is Autoencoders (AEs), due to its operation and architecture [28–34]. In addition to the improvement in predictive performance, the use of dimensionality reduction models allows a considerable reduction in the computation time of the classification algorithms. The reason for this is that the original high-dimensional data leads to a greater cost in computing time; when the input space is reduced, a smaller amount of data is provided to the classification algorithm and the associated time is reduced [30].

The architecture of AEs allows them to learn an internal representation of the input data during the training process. This phase consists of reproducing the input in the output of the network through a series of hidden layers. When the task of reducing the input space dimensionality is tackled, the internal coding must be of smaller dimensionality than the original data. Therefore, the hidden layer that supplies the information must be of smaller dimensionality than the input layer [28,30,35]. This is, however, only an overview of the operation of the AES. There are different models that include variations in different aspects, for example, introducing noise in the input data or changing the loss function. There a numerous different variants of AEs and it is not possible to incorporate all of them in this study. Therefore, four of the most widely used AE models have been considered in this paper. These models are: basic AE [36], denoising AE [37], contractive AE [38] and robust AE [39]. The objective of this study is to carry out an exhaustive study of the performance of the different models of AEs in coping with dimensional reduction. Similarly, the study will not focus on a single classification methodology but rather will aim to evaluate the behavior of classifiers belonging to different methodologies according to the type of AEs used. In this way the reader is provided with a broad set of tests, as well as associated conclusions that allow decisions to be made when facing the task of dimensionality reduction with mediated AEs.

In summary, the objective of this paper is to guide the choice of the proper autoencoder for feature fusion according to the classifier and the data complexity. In this sense, the main contributions are: (1) a parametric analysis of the four models of AEs included in the study that allow users to select the best performing configuration, (2) an experimentation of the four classification algorithms including a comparison of their results with the four AE models and with the original data, (3) an experimental demonstration of the AE model that offers the best performance for each classification methodology, (4) a comparison between AE models and other classical methods, such as PCA, LDA, ISOMAP and LLE, and (5) a series of guidelines that allow the reader to decide which AE model to use according to the characteristics of the input data.

In conclusion, the experimentation presented in this paper shows the great performance of the AEs when facing the dimensionality reduction task. Specifically, the predictive performance of the four classifiers considered after applying the most sophisticated models of AEs clearly improves with respect to the basic AE model. In general, the results generated through any model of AE are better than those that use the raw data. In addition, experimentation shows that AEs behave better than other classic models of dimensionality reduction, such as PCA, LDA, ISOMAP and LLE.

This paper is organized as follows: Section 2 includes the main theoretical concepts that are used throughout the paper: in Section 2.1 the different classification methodologies and the algorithms used in the experimentation are presented; the theoretical foundations of the AEs are described, as well as each of the AE models involved in the experimentation, in Sections 2.3 and 2.4. In Section 3, the experimental framework is defined. The selection of the best architecture of AEs is undertaken in Section 4. Section 5 presents the results obtained after applying different classification algorithms on the data generated by the AE models. In Section 6, a comparison between AEs and classic models of dimensionality reduction is carried out. Section 7 presents a series of guidelines established according to the experience provided by previous experimentation, the objective of which is to facilitate decision-making when faced with the problem of dimensionality reduction. Finally, Section 8 includes the main conclusions reached in this study.

2. Preliminaries

The main objective of this study is to analyze the performance of different models of AEs in tackling the task of dimensionality reduction and how these new representations affect classification methods corresponding to different paradigms.

For this reason, the use of different classification algorithms is considered. In Section 2.1, the main methodologies for classification tasks are presented, as well as the main algorithms used in the subsequent experimentation. In addition, it is necessary to introduce the methods used to perform dimensionality reduction. For this, the concept of AE (2.3) and the main models used (2.4) are presented.

2.1. Classifier paradigms and algorithms

Since the 20th century, different methods for tackling classification tasks have been developed. The proposals can be grouped into several methodologies according to their architecture and operation. The existing paradigms are very diverse, which opens up a large number of possibilities when opting for a specific classification method. Some of these methodologies are: instance-based learning [5], artificial neural networks [7], Bayesian networks [40], support vector machines [6], decision trees [8], rule-based systems [41], genetic algorithms [42] and inductive logic programming [43], among others. This wide variety of options requires experts to know the characteristics of each algorithm, as well as to adapt their choice to the data used in each case. The four paradigms used in this paper are described in more detail bellow, all of which are among the most well-known paradigms:

- Instance-based learning (IBL): This type of algorithm is lazy, that is to say there is no learning process where a model is built. To make the prediction, the information provided by the training instances is used directly in the inference phase [5].
- Artificial Neural Networks (ANNs): These models are inspired by the structure of the human brain. The fundamental elements used in the construction of this type of algorithm are neurons and the connections between them. ANNs use their own experience to identify relationships between the data [7].
- Support Vector Machines (SVMs): These methods relate training instances with points in space. The process, called *kernel trick*, consists of projecting the data patterns into a space of greater dimensionality. Thus, the model manages to separate linearly instances that, initially, were not separable [6].
- Decision trees (DTs): The performance and structure of these algorithms is based on trees. The main elements of these models are the branches, where the attribute that provides more information to make a division in the samples is evaluated, and the leaves, which contain the values of the target class [8].

A specific algorithm of each of these classification methodologies has been chosen aiming to include a representative of each of the most widespread paradigms in the experimentation:

- Within the IBL methodology, one of the best known methods is k-nearest neighbors (kNN). It is a non-parametric algorithm used for classification and regression tasks [44,45]. In classification, kNN

does not build a model to undertake the prediction task. The method does not do any work until it is not necessary to predict a new instance, therefore, it is called the *lazy* approach [46]. Once the new example arrives, kNN predicts the class using the information provided by the *k* nearest examples, assigning the class that is the most common among the neighbors.

- Multi-layer perceptron (MLP) is the proposal used to evaluate the performance of dimensionality reduction in ANNs. MLP is a traditional model within ANNs [47]. The algorithm can model nonlinear functions and is trained to learn and generalize knowledge from new data. MLP is formed by a series of interconnected elements, known as neurons or nodes. The neurons are organized in different layers and the connections between them are weighted. During the training process, the network uses the back-propagation algorithm to transmit the error throughout the network and adjust the weights to minimize it [48]. The fundamental objective of the model is to map the input data through the layers of the network, generating the output vector [49].
- SVMs are a type of learning algorithm commonly used for classification. This algorithm was originally introduced in 1992 [50] and has been widely used and extended due to its robust performance. This type of model separates the training data provided within a hyperplane that maximizes the distances between them. In this way, elements of similar categories will be closer than examples of different classes. If there is no possible line separation, then the algorithm employs techniques to perform non-linear mapping to a feature space [6].
- There are different proposals for DTs for tackling the task of classification. In this study C4.5 has been selected because it is a classic model within this family and because it is widely used to deal with this type of problem. This algorithm represents the features of the input data as branches and the different values of the target class as leaves of the DT. Finally, the classification rules are obtained from the tree [8,51].

The methodologies and algorithms proposed for dealing with the classification task must take the characteristics of the data used into account. These data are often extracted from different sources. At present, due to the large number of devices and sensors, one of the most frequent characteristics is the high dimensionality of the data. This factor negatively affects the predictive performance of most traditional classifiers. This phenomenon is known as the curse of dimensionality [9,10]. Another consequence associated with high-dimensional data is the need to increase the number of training instances in order to maintain an acceptable level of performance, which is known as the Hughes phenomenon [52].

This factor affects the paradigms specified in this study differently. The IBL algorithms base their operation on the information provided by the nearest instances; therefore, distance is a fundamental element. In spaces of high dimensionality, distances tend to equalize, that is they become less significant, which implies less relevant results. Something similar happens with SVM, since it tries to maximize the distances between data samples of different classes that are less significant with high dimensional data. Similarly, ANNs are affected by the existence of features that do not provide information or are redundant, so that a reduction in dimensionality where more significant features are provided would produce better results. Finally, in some case, decision trees would produce enormous structures based on non-significant features when processing data with a large number of characteristics. The problem of high dimensionality has been widely studied, and in Subsection 2.2 some of the methods proposed to deal with it are detailed.

2.2. Dimensionality reduction methods

The classification methods described in 2.1 are used for prediction tasks based on real data. These data often have high dimensionality, a factor that affects their predictive performance. Therefore, different proposals have emerged with the aim of mitigating its effects [11,12,53]. The first solutions were based on the manual work of the experts who selected the most important characteristics. However, this process was soon automated and different methods of feature selection arose. Some of the most widely used traditional algorithms are: LDA [54], PCA [15,55], ISOMAP [16] and LLE [17]. In recent years, new models have appeared for tackling this task. Advances in technology and the large amount of data available have allowed DL algorithms to be developed. In particular, AEs have shown good performance due to their structure and operation [28–30,56].

One of the first methodologies that tackles the task of dimensionality reduction is feature selection [13]. The process consists of selecting the best subset of input variables. Moreover, feature selection treats each input attribute independently. However, it is well known that the information provided by the variables treated together can be more useful than if they are used independently. In order to solve this factor, other methodologies arose, such as feature extraction and feature fusion [57].

The objective of the feature extraction methodology is to generate the best representation of the input data [21]. This representation will depend on the features of this data and the characteristics of the machine learning algorithm that will be used. To generate the new representation several techniques are used: basic transformations in the data, normalization, discretization and scaling. These types of methods can be classified as supervised (LDA) [54] or non-supervised (PCA) [15,55]. In a similar manner, the new feature space can be obtained through linear combinations of input data, such as in PCA or LDA, or through non-linear combinations, such as ISOMAP [16] or LLE [17]. In the second case, the methods that apply nonlinear dimensionality reduction techniques are known as *manifold learning* [58,59]. These methods have been shown to be very effective in generating new feature spaces, but they have an extremely high computational cost.

As a result, the new term feature fusion has recently emerged [22–24,32–34], due to the need to process multimedia data, especially text, images and sound. The main objective of feature fusion algorithms is to generate new attributes by combining original variables. This way, less relevant and redundant information is eliminated, making the task of automatic learning algorithms more effective [22]. In this context, the use of different DL models has experienced an important rise, specifically regarding AEs. This type of model has shown great performance in discovering *manifolds* between the data automatically, with a much lower computational cost than traditional methods. Therefore, this study focuses on analyzing the behavior of different AE models when tackling the task of dimensionality reduction. In Subsection 2.3 the AE concept and the different models used are presented.

2.3. Autoencoder foundations

AEs are ANNs whose structure is symmetrical. The main objective of this type of model is to reconstruct the input into the output. The network learns using only the input attributes to carry out this process, without the need for any labeling or prior processing; therefore, it is about unsupervised learning. The network can simply copy the input to the output. To avoid this, certain restrictions must be verified [28,30]. Nowadays, the most widely used term used to refer to these structures is autoencoder, but they have also been referred to as diabolo networks [60], autoassociative neural networks [61] and replicator neural networks [62].

The structure of the AEs allows a coded representation of the input information to be obtained in the middle layer. From this coding, the network is able to reconstruct the original input. In cases in which this middle layer is smaller than the input, a representation of lower dimensionality can be obtained [29,56,63]. This fact means that AEs are being widely used to reduce the size of the feature space. Specifically, these models combine variables to remove redundant and irrelevant information, which is known as feature fusion.

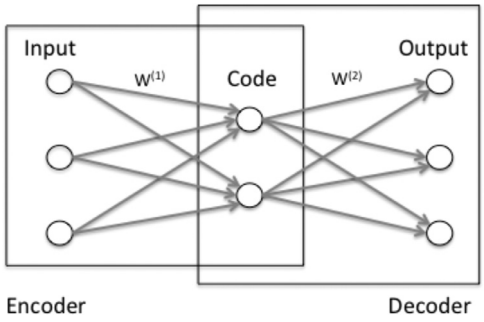


Fig. 1. Architecture of autoencoder with one hidden layer.

In general terms, the basic structure of an AE is a feed-forward neural network [47] without cycles, where information always flows in the same direction. This structure is very similar to the multilayer perceptron. The architecture of the AE is composed of a sequence of layers: an input layer, a series of hidden layers and an output layer. The input and output layers must have the same dimension in order to reproduce the input into the output through the network [64]. The middle layer must be smaller than the input when the objective is to perform feature fusion. This architecture is presented in Fig. 1.

In Fig. 1, the AE has 3 layers. The architecture is divided into two parts. In the first part, the encoder is made up of the first two layers, including the middle encoding one. In the second part the decoder starts in the middle layer and reaches the output layer. Each neuron connects with all of the previous layers, and these connections form weight matrices denoted by W.

AEs can have as many layers as necessary, often located symmetrically as seen in Fig. 1. However, this representation corresponds to the most basic architecture; there are proposals that include modifications. Some of these will be seen in Section 2.4.

In this context, the output layer in the AE can be useful depending on the task in hand. On the one hand, there are studies, such as this one, where this layer is not the central objective, since the purpose is to extract information from the hidden layers. On the other hand, the output layer acquires great importance, for example, in cleaning the noise of the input data.

According to the objectives set out in this paper the importance lies in the information located in the hidden layers. Therefore the restrictions included in the network are fundamental, allowing the model to learn an internal representation of the input data. This new coding corresponds to feature fusion with a higher level of representation. As a result, the process eliminates redundant or unnecessary information. The benefit of the model is that this code can be extracted and used in an independent process [65]. According to the size of the middle layer, there are two types of AEs:

- Undercomplete: The size of the middle layer is smaller than the input. This type of architecture forces the network to learn a coded and compressed representation of the input space. This model is used to carry out feature fusion, where new higher level variables are generated.
- Overcomplete: The size of the middle layer is larger than the input and output layer. This model can be limited by its structure to copying the entry through the network without learning anything useful. So, it is necessary to include restrictions to avoid this. Therefore, the architecture allows for a sparse representation of the input data.

This study focuses on undercomplete AEs, since the aim is to reduce the dimensionality of the original data. This architecture is a very suitable tool for performing feature fusion, obtaining new variables of a higher level and lower dimensionality.

There are several AE models that can be used to tackle the task of dimensionality reduction. However, there are no data in the literature that facilitate the task of selecting the most appropriate method depending on the type of data or the type of classification algorithm to be used. Therefore, this paper aims to provide exhaustive experimentation to facilitate this task. With this aim, the performance of different AE models is analyzed. In Section 2.4 the approaches used in this experimentation are presented.

2.4. Autoencoder models

The feature fusion task using AEs can be achieved using different approaches. In this study we focus on undercomplete models where the coding produced in the intermediate layer is of lower dimensionality. Therefore, overcomplete models are not taken into account, due to their structure. However, these models are widely used to deal with other problems. For example, sparse AEs are suitable for speech [66] and image recognition [67].

As already indicated, the objective of AEs is to find a codification of the data by learning non-linear combinations of their features. In this section, different approaches that lead to a lower-dimensional space are discussed. Specifically, four of the best-known AE models are described: basic, contractive, denoising and robust. This study focuses on these four proposals with the aim of establishing a baseline study on the use of AEs to reduce dimensionality. However, there are many other variants in the literature. In fact, new proposals for AEs arise continuously. Some of these proposals are: Correspondence AEs [68], AE Node Saliency [69] and AE With Invertible Functions [70].

2.4.1. Basic autoencoder

The main objective of the AE models analyzed in this section is to tackle the task of dimensionality reduction on a wide set of datasets with disparate characteristics. Hence, the model obtained will be able to map new examples onto the latent feature space. All AEs start from a basic model, called basic AE (BAE) [36].

In the previous section the basic AE structure has been presented. This consists of feed-forward ANN with symmetrical layer architecture. The symmetry does not necessarily have to be reflected in the weights and activation functions.

The most basic AE, when there is only one hidden layer, is composed of two weight matrices, W and W', and two bias vectors, b and b'. Therefore, where x is the input vector the functions of AE can be expressed as follows:

$$z = f(x) = \gamma_1(Wx + b) \tag{1}$$

$$x' = g(y) = \gamma_2(W'z + b') \tag{2}$$

Eq. (1) corresponds to the compression function, from which an encoded input representation is obtained. Eq. (2) corresponds to the decoder part, where the AE reconstructs the input from the information contained in the hidden layer. Here, γ_1 and γ_2 are two activation functions, which are usually nonlinear.

The objective function for AEs generally corresponds to a per-instance loss function. For example, a widely used metric is the mean square error (MSE). Similarly, the algorithms used to optimize the weights and biases in these models are stochastic gradient descent (SGD) [71] and variants, such as RMSProp [72] or AdaGrad [73].

The gradient descent technique consists of modifying the parameters in order to minimize the objective function [74]. So, by applying the back-propagation algorithm, the necessary gradients are computed [48]. Back-propagation starts by calculating terms of the last layers and transmits the value through the network.

In many cases, a regularization term is added to prevent the overfitting of the model to the training data. The following models incorporate restrictions and mechanisms on basic AE in order to increase its performance.

2.4.2. Contractive autoencoder

AEs are very sensitive to variations in the input data, and small perturbations could generate very different encodings. This is a drawback and motivates the appearance of the model known as contractive AE (CAE) [38]. This type of AE achieves local invariance to changes in the training instances and so it is able to identify lower-dimensional manifold structures more easily.

CAEs include a regularization term that allows them to stabilize the encodings in spite of disturbances in the input data. This new model can generate instances from the learning performed, adding noise at different points and computing its coding [38].

2.4.3. Denoising autoencoder

Denoising AE (DAE) [37] introduces modifications to the basic model in order to achieve greater robustness, that is, allowing the model to reconstruct the input by eliminating any noise that is present.

AEs have previously been applied to the noise elimination task [75]. However, this model has a broader objective, since it takes advantage of noise to build a new feature space that is more resistant to corrupt entries. As such, the field of application is wider and the performance of the AE increases.

The architecture of the DAE model is identical to that of the basic model. The fundamental modification is in the corruption of the entrance during the training phase. An example of this process is given in [37], where a number of input variables are randomly chosen and set to 0. However, the reconstruction is compared to the original unmodified values. Therefore, the AE training process will be adjusted in order to detect the missing values.

DAE can have several hidden layers. Similarly, the training technique can be adapted to other ways of corrupting the input data [76], for example, additive Gaussian noise or salt-and-pepper noise.

2.4.4. Robust autoencoder

Robust AEs (RAE) are networks trained to tolerate possible noise present in the training data [39], just like DAE. However, this task is handled in a different way. The alternative used is to modify the loss function used when training the network. At the time of minimizing the reconstruction error, changes are introduced that reduce the effects of noise in the calculations performed.

Robust stacked AEs incorporate this idea with the aim of being more tolerant to input noise than basic models. To accomplish this, the proposal in [77] uses an error function based on correntropy.

Correntropy allows us to measure the probability density that two events are similar. The outliers affect this measure to a lesser extent than the MSE. Therefore, RAE attempts to maximize this measurement, implying greater resilience to noise.

3. Experimental study

In order to analyze the improvements of tackling the task of feature fusion using AEs, an exhaustive experimental study has been carried out. In addition, this analysis compares different AE models with the aim of determining which of them offers better performance. Thus the experimentation performed follows the steps detailed bellow:

- Firstly, the different models of AEs are used to generate a new feature space from that of the input data. For this, different AE architectures are used in each case. In this way several subsets with different degrees of reduction are obtained from the original datasets. The models of AEs used are: basic, contractive, denoising and robust.
- Secondly, the classification with the different methods is carried out. Each classification algorithm works with the different data sets generated in the previous phase. In this way classifications are performed for the different subsets of the same dataset. This process allows us to establish comparisons between the AE models and architectures used when generating reduced subsets of the input data. The classifiers used are: kNN, MLP, SVM and C4.5.

In the following sections, the analysis of the experimentation is presented. Fundamentally, the objectives pursued are:

- Determine which structure of AEs offers better predictive performance. For this purpose, classification results for the different configurations are compared in Section 4. Due to the large number of results generated, this step is necessary in order to focus the subsequent analysis on just one architecture.
- Perform a comparison of the classification results obtained by the different classifiers in Section 5. The comparative data will be those corresponding to the classification of the subsets obtained with the four models of AEs with the configuration selected in Section 3 and the classification of the original data without reduction of dimensionality. The purposes of Section 5 are:
  - Present, in Section 5.1, the experimental framework of the classification algorithms used in this study.
  - Analyze, by type of classifier, the performance of the different models of AEs in Sections 5.2–5.5.
  - Establish a general analysis of the results supported by statistical tests in Section 5.6.
  - Study the execution time of the classification algorithms considering the different AE architectures in Section 5.7.
- Carry out a comparison of the AEs' performance when tackling the task of dimensionality reduction compared to the traditional methods outlined in Section 6.
- Propose guidelines to facilitate the task of dimensionality reduction using AEs in Section 7.

In addition, Section 3.1 presents the framework used in the experimentation.

3.1. Experimental framework

This study aims to analyze the performance of different models of AEs when dealing with the reduction of dimensionality. Similarly, the experimentation aims to determine the behavior of different classifiers depending on the type of AE. For this, a wide range of datasets with very varied characteristics has been used. Their traits are presented in Table 1, which also shows the origin of the dataset in the Ref column. When performing the executions, a 2×5 fold cross validation scheme was applied.

The datasets described in Table 1 are those used by all models of AEs to establish a comparison based on a large data set with different characteristics. In addition, establishing a form of evaluation is necessary in order to assess the predictive performance of different methods and models. In this case the area under the ROC curve (AUC) has been used. This metric offers a robust view of the predictive performance of the models evaluated. That is to say, AUC provides a reliable overview of the results, something that has not been obtained with other metrics such as Precision or Accuracy, which give a more partial view. AUC is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. AUC is given by the Eq. (3):

$$AUC = \int_{\infty}^{-\infty} TPR(T)FPR(T)dT \tag{3}$$

where *TPR* stands for the true positive rate and *FPR* is false positive rate.

In this study, the *pROC* package for R, which contains a set of tools for displaying and analyzing ROC curves, is used to calculate AUC for both binary and multi-class datasets [87]. This package obtains multiclass AUC as defined by Hand and Till [88].

Finally, two statistical tests are used in this study to verify the significance of the results obtained. This will allow us to establish the conclusions associated with the study. The tests performed are:

- The Friedman test [89] is used to rank the different configurations. In this way, the selection of the best architecture and model can be carried out.

Table 1
Characteristics of the datasets used in the experimentation.

Dataset	Number of			Type	Field	Ref
	Samples	Features	Classes			
arcene	900	10,000	2	Real	Medical	[78]
batch	13,910	128	6	Real	Chemical	[79]
coil2000	9822	85	2	Integer	Social	[80]
dota	102,944	116	2	Real	Game	[81]
drive	58,509	48	11	Real	Motor	[81]
facial	2964	301	2	Real	Image	[81]
fashionmnist	70,000	784	10	Integer	Image	[82]
gisette	13,500	5000	2	Integer	Image	[78]
hapt	10,929	561	12	Real	Activity	[83]
image	2310	19	7	Real	Image	[81]
isolet	7797	617	26	Real	Image	[84]
letter	20,000	16	26	Integer	Image	[81]
madelon	2000	500	2	Real	Artificial	[85]
mfeat	2000	649	10	Real	Image	[81]
microv1	360	1300	10	Real	Biology	[81]
microv2	571	1300	20	Real	Biology	[81]
mnist	70,000	784	10	Integer	Image	[86]
musk	6598	168	2	Integer	Physical	[81]
nomao	1970	118	2	Real	Technology	[81]
semeion	1593	256	10	Integer	Image	[81]

- The Li post-hoc tests [90] for Friedman test are applied in order to compare all the configurations and allow us to verify whether there are significant differences between these results. The Li test is a non-parametric method appropriate when the number of samples is not very large and it is a good alternative for finding significant differences [91].

The equipment used to develop this set of executions is a cluster made up of 18 computers, with 2 CPUs (2.33 GHz) of 7 GB RAM each. The cluster is mounted using Rocks 6.1.1, a cluster Linux distribution. Rocks 6.1.1 is based upon CentOS 6.5. Therefore, CentOS 6.5 is the operating system included in each of the nodes of the cluster. The different models of AEs were coded in Python language, using the Keras library [92]. The classification methods used were programmed using R language [93] and are detailed in Subsection 5.1. Finally, it is important to note that during the executions of this study the cluster has been dedicated exclusively to them, without any other type of work executed in the system. In addition, two parallel tasks have been executed in each node, as each one has 2 CPUs.

4. Autoencoders architectures analysis

The objective of this section is to study which structure of AEs offers a better predictive performance. In this way, the subsequent analysis will focus on the best architecture obtained. Section 4.1 describes the different architectures used and Subsection 4.2 presents the results.

4.1. Autoencoders architectures framework

In this subsection, the architectures used with the different AE models to perform the feature fusion are presented. One of the fundamental characteristics of these models is that they have symmetrical architecture where the number of neurons in the input layer is equal to that of the output layer. In this work, AEs are used to reduce dimensionality, therefore, the intermediate layer must have a lower dimensionality than the input and output layers. In Table 2, the different configurations are shown.

To evaluate the different models, the same selection of several architectures were used for all models of AEs. The architectures proposed in Table 2 make a reduction in the number of original features of 75% (ARQ_75), 50% (ARQ_50) and 25% (ARQ_25), respectively. Thus, AEs will essentially have the following layers:

Table 2
Autoencoder architectures used in the experimentation.

	Number of layers	Number of neurons (% of total)		
		Input	Hidden	Output
ARQ_25	3	100	25	100
ARQ_50	3	100	50	100
ARQ_75	3	100	75	100

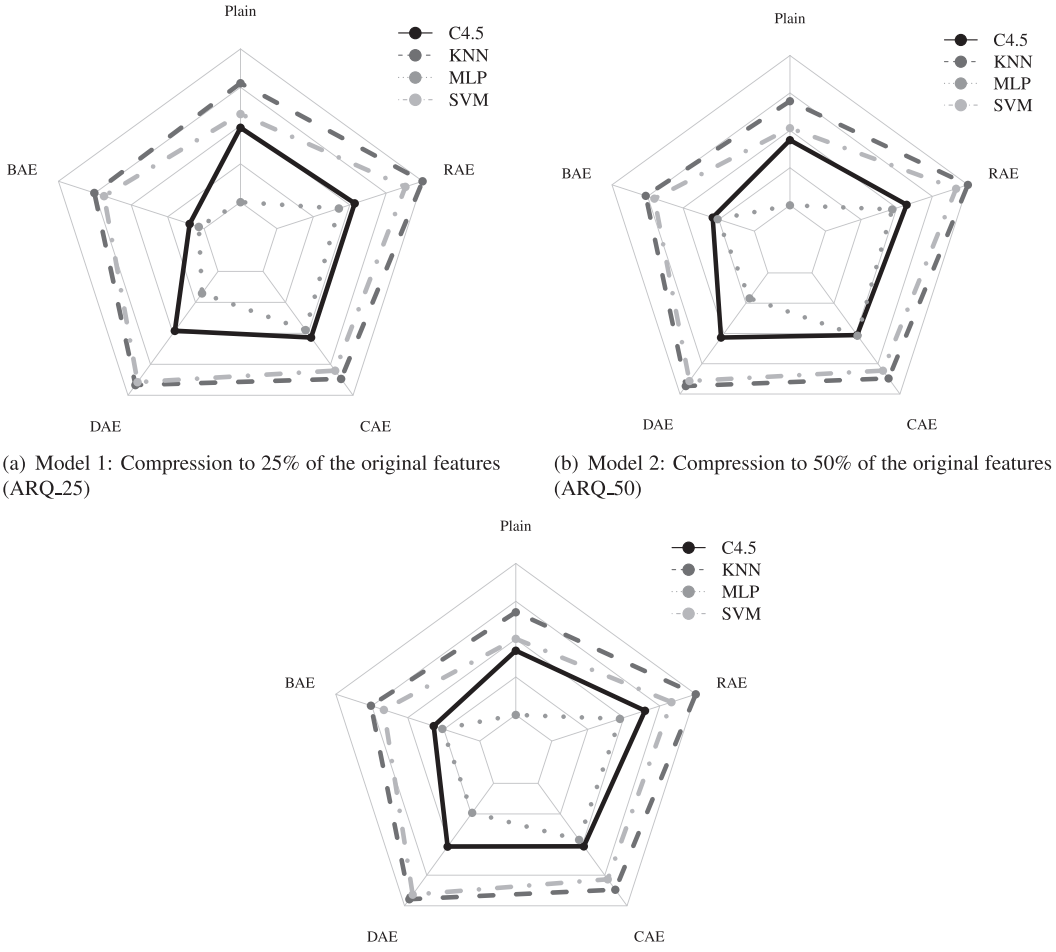
- Input layer: This part has as many neurons as the original dataset has features.
- Hidden layer: The number of units is variable in each configuration according to the reduction carried out.
- Output layer: Due to the symmetric architecture of AEs, this layer has the same number of elements as the input.

As indicated above, the architectures used have a single hidden layer. The objective is to make the evaluation of basic architectures to establish a baseline study. However, AEs can be expanded with more layers while maintaining their symmetrical structure. In this experimentation, the architectures considered have a single hidden layer. The main reason is, as has been shown in previous experiments [30], that including more hidden layers in the form of stacked AEs does not imply an improvement in predictive performance.

4.2. Autoencoders architectures analysis

The main objective of this Section is to study which AE configuration provides a better predictive performance with the different classifiers. Due to the large number of results generated, the objective is to establish one of the configurations in order to later perform a more detailed analysis of classifiers.

The visualization using plots of the results generated can provide a first approximation of their quality and of which models work better. Therefore, the first step is to graphically represent the results obtained for the different configurations. Fig. 2(a)–(c) present the results obtained by compressing 25% (ARQ_25), 50% (ARQ_50) and 75% (ARQ_75), respectively. The three plots show the aggregated results by configuration, AE model and classifier. The classifiers are represented by the different strokes, while the vertices of the plot correspond to the different AE models.



(a) Model 1: Compression to 25% of the original features (ARQ_25)

(b) Model 2: Compression to 50% of the original features (ARQ_50)

(c) Model 3: Compression to 75% of the original features (ARQ_75)

Fig. 2. Predictive performance (AUC) of the different autoencoder models and classification algorithms for each architecture.

Table 3
Average rankings of the different autoencoder architectures by classification method.

kNN		SVM		MLP		C4.5	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
ARQ_75	1.850	ARQ_75	2.037	ARQ_75	1.850	ARQ_75	1.825
ARQ_50	2.075	ARQ_50	2.050	ARQ_50	2.150	ARQ_50	2.300
ARQ_25	2.750	ARQ_25	2.662	ARQ_25	2.725	Plain	2.837
Plain	3.325	Plain	3.250	Plain	3.275	ARQ_25	3.037

Plots 2(a)–(c) show the results for the different configurations proposed in this study. Each figure is a radar chart that represents the results obtained with the 4 models of AE and the model without performing compression. Similarly, there is a line for each classification algorithm and so, a global vision of the performance of the different models is generated. The representation is an aggregation of all the datasets used.

In general, RAE works better in all cases, since it can be observed that for all classification methods it tends to the maximum value. On the contrary, the worst results are obtained with BAE and with the model without compression, which is expected since they are the less sophisticated models and contain original data without processing, respectively.

Regarding the different classification methods, some patterns can be observed. On the one hand, for the kNN, SVM and MLP algorithms, the different models of AE improve in the three configurations on execution without compression. On the other hand, C4.5 performs the worst, since in some cases the execution without compression improves some of the models of AE used.

Once a global vision of the results obtained has been presented, the objective is to select the configuration with the best predictive performance. Due to the large number of executions carried out, it is necessary to perform a more detailed subsequent analysis.

To determine the best configuration, it is necessary to verify whether there are significant differences between them. To do so, first the Friedman test [89] is applied. Average ranks obtained by applying this test for AUC are shown in Table 3. These rankings have been generated by adding data from all datasets and AE models. In this case, the objective is to select the best architecture, therefore it is not necessary to separate the results by AE model. This more detailed analysis will be shown in Section 5.

As can be seen in Table 3, the configuration that provides the best predictive performance is that which reduces the number of features

Table 4
Li post-hoc Friedman test for classification algorithms by autoencoder architecture.

		ARQ_25	ARQ_50	ARQ_75	Plain
kNN	ARQ_25	-	-	-	**6.601E-03**
	ARQ_50	**1.292E-03**	-	-	**1.253E-09**
	ARQ_75	**1.423E-05**	**1.703E-01**	-	**6.817E-13**
	Plain	-	-	-	-
SVM	ARQ_25	-	-	-	**7.571E-02**
	ARQ_50	**5.229E-02**	-	-	**8.464E-08**
	ARQ_75	**4.311E-02**	9.512E-01	-	**5.837E-08**
	Plain	-	-	-	-
MLP	ARQ_25	-	-	-	**8.147E-03**
	ARQ_50	**5.617E-03**	-	-	**4.148E-08**
	ARQ_75	**2.114E-05**	**1.416E-01**	-	**3.413E-12**
	Plain	-	-	-	-
C4.5	ARQ_25	-	-	-	-
	ARQ_50	**4.497E-04**	-	-	**1.242E-02**
	ARQ_75	**4.236E-09**	**2.882E-02**	-	**1.047E-06**
	Plain	3.272E-01	-	-	-

to 75% (ARQ_75) of the total. In addition, the worst results are obtained with the execution without compression, except for C4.5 where the worst results are obtained with the configuration ARQ_25 (25% of the total).

To select a configuration based on solid results, it is necessary to verify whether there are statistically significant differences. Therefore, Li post-hoc tests [90] for Friedman test are applied in order to compare all the configurations. Table 4 shows the different *p-values* obtained by the Li test. In this table, the p-values are represented when the architecture of the row has a better ranking than the architecture of the column.

As can be seen in Table 4, there are significant differences between the different configurations proposed in this study if we set the *p-value* threshold to the usual range [0.05, 0.1]. The only cases where there are no clearly significant differences is for SVM, between the models that reduce to 50% (ARQ_50) and 75% (ARQ_75) of the total, and with C4.5, between the models that reduce to 25% and the model without reduction. However, the results presented in this section allow us to establish a solid base for selecting the architecture that reduces 75% of the total as the best in terms of predictive performance.

5. Classification performance analysis

Once the configuration that generates a better predictive performance in classification has been selected in Section 4, the next objective is to perform a more detailed analysis of the different AE models and how they affect the classification methods proposed in this study. For this, Section 5.1 presents the framework used for the different classification algorithms and Sections 5.2–5.5 present the results obtained by kNN, SVM, MLP and C4.5, respectively. In these subsections the results are presented in greater detail, showing the values of AUC for each one of the datasets and for each AE model. The different tables present the performance of the four classifiers when using the original dataset without reduction and the subsets that were generated after carrying out fusion features with the AE models. In all of the tables, the best result for each dataset is highlighted in bold. In addition, Section 5.6 establishes a general analysis where statistical tests are used to support the results and Section 5.7 studies the execution time of the different architectures considered.

5.1. Classification algorithm framework

The main objective of this paper is to carry out an exhaustive study that analyzes the performance of AE models in handling the task of dimensionality reduction. This experimentation allows us to offer useful information when using these models according to the characteristics of the data and the classification algorithms that are used. In Section 3.1, data sets with different traits have been described. In this Subsection, the classification algorithms used, their origin and their main parameters are presented.

The classification algorithms are implemented in R [93], using in each case a specific package of this platform. Another important aspect is the parameterization of the classification algorithms. Bellow, the main characteristics of the different algorithms are shown:

- For the kNN algorithm, the *kknn* package has been used [94]. Similarly, the value of parameter *k* is 5, since it is the one recommended in the literature [30].
- The SVM algorithm has been obtained from the *e1071* package [95]. The parameters used are those established by default in the package documentation including radial kernel.
- To perform the classification with the MLP algorithm, *caret* and *RSNNS* packages were used [96,97]. The structure of the network and the number of iterations are determined by the default parameters of the package.
- The C4.5 algorithm was provided by the *RWeka* package [98]. The default parameters were used in the implementation of C4.5.

The parameters used in four algorithms are those provided by default for the different algorithms, since the objective is to assess the predictive performance obtained by reducing with AEs without adjusting the methods in any other way.

5.2. kNN

Table 5 shows the results obtained using the kNN algorithm. These data show that in most cases the best results are obtained with data sets reduced by AEs. Specifically, in 10 out of 20 cases the best result is obtained with RAE, in 4 out of 20 datasets DAE generates the best predictive performance and in 5 out of 20 the best data is obtained by CAE. However, there are 2 cases where the best results are obtained with the original dataset.

These results show the improvement of predictive performance occurs when reducing dimensionality using AEs. The justification for this can be found in the functional scheme of the IBL algorithms, specifically, kNN. The behavior of this type of method is affected by the high dimensionality of the input space, because in this scenario the distances between the instances tend to equalize. Therefore, the loss of significance of this factor leads to a fall in predictive performance [99]. In this context, AEs are incorporated. The objective is to reduce the dimensionality of the input data using several AE models, aiming to obtain distances between the examples that are more significant. AEs carry out the fusion features, where new spaces are generated by adding the most relevant information from the original data.

Another aspect to note is that for 18 datasets the best results were obtained with sophisticated AE models (CAE, DAE, RAE), which is obvious since they incorporate improvements over BAE. Therefore, the generated feature fusion contains more and more relevant information and is more useful when carrying out the classification.

5.3. SVM

In Table 6 the classification results of the SVM algorithm are presented. The trend of the previous algorithm is maintained according to the data shown. The best results are obtained by classifying subsets generated by AEs in most cases. In more detail, the RAE model shows the best performance in 8 out of 20 cases, DAE works best in 5 out of 20 datasets and CAE generates better results in 5 out of 20 cases. Finally, there are no improvements for the other 3 datasets, where the best performance is obtained with the plain data.

The behavior of the SVM algorithm when decreasing the input space by means of AEs clearly improves with respect to the results with the

Table 5
kNN results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.693 ± 0.003	0.591 ± 0.011	0.605 ± 0.015	0.672 ± 0.005	**0.743** ± 0.008
batch	0.870 ± 0.005	0.877 ± 0.003	0.900 ± 0.010	0.990 ± 0.001	**0.992** ± 0.001
coil2000	0.546 ± 0.003	0.543 ± 0.002	**0.554** ± 0.004	**0.554** ± 0.004	0.545 ± 0.003
dota	0.416 ± 0.021	0.515 ± 0.002	0.519 ± 0.000	**0.522** ± 0.003	0.516 ± 0.002
drive	0.800 ± 0.009	0.819 ± 0.011	**0.877** ± 0.005	0.846 ± 0.006	0.873 ± 0.004
facial	**0.795** ± 0.011	0.668 ± 0.005	0.690 ± 0.006	0.685 ± 0.003	0.697 ± 0.002
fashionmnist	0.906 ± 0.004	0.914 ± 0.003	0.912 ± 0.001	0.920 ± 0.003	**0.922** ± 0.001
gisette	0.824 ± 0.005	0.879 ± 0.009	0.897 ± 0.004	**0.923** ± 0.002	0.900 ± 0.004
hapt	0.903 ± 0.005	0.918 ± 0.004	0.876 ± 0.005	0.931 ± 0.003	**0.939** ± 0.002
image	0.929 ± 0.005	0.952 ± 0.000	0.952 ± 0.002	0.950 ± 0.001	**0.958** ± 0.003
isolet	0.943 ± 0.003	0.942 ± 0.004	0.974 ± 0.001	0.968 ± 0.002	**0.976** ± 0.002
letter	**0.973** ± 0.015	0.818 ± 0.007	0.925 ± 0.012	0.923 ± 0.006	0.947 ± 0.005
madelon	0.526 ± 0.004	0.550 ± 0.003	0.566 ± 0.006	0.591 ± 0.003	**0.618** ± 0.005
mfeat	0.703 ± 0.004	0.982 ± 0.002	**0.986** ± 0.002	0.984 ± 0.001	0.984 ± 0.001
microv1	0.920 ± 0.005	0.933 ± 0.002	0.922 ± 0.003	**0.939** ± 0.005	0.929 ± 0.004
microv2	0.883 ± 0.006	0.896 ± 0.003	**0.955** ± 0.002	0.932 ± 0.005	0.954 ± 0.003
mnist	0.965 ± 0.003	0.958 ± 0.001	0.969 ± 0.000	0.975 ± 0.001	**0.979** ± 0.000
musk	0.829 ± 0.005	0.914 ± 0.004	0.940 ± 0.002	0.941 ± 0.001	**0.947** ± 0.002
nomao	0.891 ± 0.001	0.891 ± 0.003	**0.914** ± 0.005	0.904 ± 0.002	0.905 ± 0.003
semeion	0.927 ± 0.003	0.929 ± 0.002	0.907 ± 0.004	0.940 ± 0.001	**0.942** ± 0.000

Table 6
SVM results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	**0.632** ± 0.005	0.500 ± 0.000
batch	0.923 ± 0.003	0.933 ± 0.005	0.924 ± 0.004	0.980 ± 0.002	**0.985** ± 0.002
coil2000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000
dota	0.509 ± 0.002	0.500 ± 0.000	**0.553** ± 0.001	0.550 ± 0.002	0.534 ± 0.005
drive	0.885 ± 0.005	0.934 ± 0.007	**0.985** ± 0.003	0.941 ± 0.002	0.863 ± 0.004
facial	**0.802** ± 0.003	0.697 ± 0.006	0.701 ± 0.004	0.713 ± 0.002	0.714 ± 0.005
fashionmnist	0.932 ± 0.002	0.933 ± 0.005	0.938 ± 0.003	0.940 ± 0.000	**0.942** ± 0.001
gisette	0.803 ± 0.008	0.821 ± 0.007	0.816 ± 0.001	**0.969** ± 0.003	0.955 ± 0.006
hapt	**0.900** ± 0.001	0.895 ± 0.001	0.871 ± 0.003	0.889 ± 0.001	0.886 ± 0.004
image	0.846 ± 0.004	0.853 ± 0.003	0.886 ± 0.001	0.877 ± 0.005	**0.910** ± 0.002
isolet	0.954 ± 0.003	0.968 ± 0.002	0.971 ± 0.002	0.975 ± 0.001	**0.976** ± 0.001
letter	**0.966** ± 0.003	0.790 ± 0.006	0.841 ± 0.004	0.828 ± 0.004	0.879 ± 0.005
madelon	0.569 ± 0.004	0.569 ± 0.004	0.580 ± 0.002	**0.591** ± 0.001	0.587 ± 0.000
mfeat	0.970 ± 0.002	0.982 ± 0.001	**0.985** ± 0.002	**0.985** ± 0.001	**0.985** ± 0.001
microv1	0.500 ± 0.000	0.774 ± 0.004	0.796 ± 0.003	**0.918** ± 0.005	0.786 ± 0.002
microv2	0.500 ± 0.000	0.849 ± 0.002	**0.916** ± 0.003	0.871 ± 0.001	0.881 ± 0.001
mnist	0.981 ± 0.002	0.984 ± 0.001	0.984 ± 0.001	0.985 ± 0.001	**0.987** ± 0.000
musk	0.838 ± 0.004	0.893 ± 0.004	0.939 ± 0.005	0.948 ± 0.006	**0.964** ± 0.003
nomao	0.914 ± 0.001	0.915 ± 0.001	**0.935** ± 0.000	0.924 ± 0.002	0.932 ± 0.001
semeion	0.891 ± 0.000	0.913 ± 0.002	0.957 ± 0.003	0.957 ± 0.001	**0.962** ± 0.001

plain data. The philosophy of the SVM algorithm is based on the maximization of distances between data samples of different classes [6]. In this sense, the distance between the instances tends to equalize in spaces of high dimensionality. This leads to a considerable loss of predictive performance and justifies the use of methods to mitigate its effects [10].

In this study, the use of AEs allows the generation of a new space of features where the distances are more significant. During the process, different AE models are trained to discard irrelevant or redundant information. In this new scenario the SVM algorithm improves the distribution of instances, generating better predictive results.

The results for SVM confirm that more complex AE models offer better results in 17 datasets. These results provide a solid basis for considering the use of AEs to pre-process the data when classification is performed using SVM-based algorithms. This is especially recommended when the data has high-dimensional input space.

5.4. MLP

The classification results generated by the MLP algorithm are shown in Table 7. These data reflect that the predictive performance is improved by reducing the dimensionality using the different models of AEs. This continues to confirm the trend marked by the previous algorithms. In this case, all datasets show improvements when applying AEs. Specifically, RAE generates the best results in 10 out of 20 cases, DAE obtains the best predictive performance in 2 out of 20 datasets, CAE in 6 out of 20 sets and BAE is the best model in the 2 remaining cases.

The results shown in Table 7 confirm that the predictive performance of a basic neural network (MLP) improves by reducing the input space using different models of AEs. The MLP algorithm corresponds to a basic neural network whose objective is to learn to predict the class of a new instance from a series of input instances. During the training process, the network adjusts its parameters in order to generate the smallest possible error in the training data [47]. The information of each of the classes that the network generalizes is worse if the dimensionality of the data increases, so it is necessary to use different methods that improve this.

The AEs used in this experimentation are also ANNs. Therefore, the feature fusion produced generates information that follows a ”natural” process when used as input to another ANN. In fact, this process is very similar to classification schemes based on DL, which do not use any external models [65]. ANN convention allows us to extract higher-level information that improves the predictive performance of MLP.

Following the trend marked by previous algorithms, MLP results indicate that complex AE models offer the best performance in 18 datasets.

Table 7
MLP results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.522 ± 0.001	0.492 ± 0.003	0.465 ± 0.005	0.500 ± 0.000	**0.523** ± 0.001
batch	0.914 ± 0.003	0.920 ± 0.005	**0.929** ± 0.002	0.920 ± 0.003	0.920 ± 0.002
coil2000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000
dota	0.500 ± 0.000	0.504 ± 0.001	0.529 ± 0.003	0.500 ± 0.000	**0.547** ± 0.005
drive	0.698 ± 0.008	0.740 ± 0.011	**0.964** ± 0.004	0.680 ± 0.010	0.812 ± 0.006
facial	0.520 ± 0.009	0.699 ± 0.005	0.706 ± 0.003	0.701 ± 0.004	**0.718** ± 0.003
fashionmnist	0.901 ± 0.001	0.908 ± 0.004	0.915 ± 0.001	0.911 ± 0.004	**0.917** ± 0.002
gisette	0.605 ± 0.004	**0.755** ± 0.003	0.591 ± 0.002	0.627 ± 0.003	0.614 ± 0.003
hapt	0.839 ± 0.005	**0.855** ± 0.002	0.838 ± 0.002	0.850 ± 0.001	0.841 ± 0.003
image	0.703 ± 0.005	0.823 ± 0.004	0.884 ± 0.003	0.872 ± 0.003	**0.898** ± 0.001
isolet	0.827 ± 0.008	0.786 ± 0.005	0.850 ± 0.006	0.842 ± 0.007	**0.880** ± 0.006
letter	0.734 ± 0.003	0.701 ± 0.005	**0.749** ± 0.003	0.740 ± 0.002	0.742 ± 0.005
madelon	0.500 ± 0.000	0.556 ± 0.002	0.590 ± 0.001	**0.637** ± 0.004	0.631 ± 0.002
mfeat	0.500 ± 0.000	0.932 ± 0.003	0.938 ± 0.003	0.968 ± 0.001	**0.976** ± 0.000
microv1	0.678 ± 0.005	0.692 ± 0.006	0.713 ± 0.004	0.690 ± 0.005	**0.714** ± 0.003
microv2	0.708 ± 0.006	0.500 ± 0.000	**0.747** ± 0.002	0.725 ± 0.004	0.742 ± 0.003
mnist	0.872 ± 0.003	0.891 ± 0.004	0.896 ± 0.000	0.905 ± 0.002	**0.909** ± 0.001
musk	0.850 ± 0.003	0.953 ± 0.004	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000
nomao	0.901 ± 0.002	0.904 ± 0.002	**0.923** ± 0.003	0.530 ± 0.011	0.913 ± 0.003
semeion	0.827 ± 0.003	0.817 ± 0.008	**0.860** ± 0.005	0.756 ± 0.002	0.793 ± 0.006

Table 8
C4.5 results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.655 ± 0.005	0.679 ± 0.009	0.565 ± 0.006	0.678 ± 0.003	**0.730** ± 0.004
batch	0.921 ± 0.003	0.941 ± 0.002	0.911 ± 0.003	0.982 ± 0.000	**0.983** ± 0.001
coil2000	0.503 ± 0.002	0.509 ± 0.003	**0.521** ± 0.001	0.516 ± 0.002	0.513 ± 0.004
dota	0.462 ± 0.004	0.498 ± 0.006	**0.548** ± 0.003	0.539 ± 0.002	0.521 ± 0.005
drive	0.869 ± 0.003	0.862 ± 0.006	**0.944** ± 0.003	0.850 ± 0.004	0.867 ± 0.005
facial	**0.789** ± 0.011	0.629 ± 0.009	0.660 ± 0.005	0.655 ± 0.013	0.683 ± 0.008
fashionmnist	0.877 ± 0.002	0.875 ± 0.000	0.883 ± 0.003	0.888 ± 0.001	**0.890** ± 0.000
gisette	0.657 ± 0.005	0.758 ± 0.008	0.714 ± 0.003	0.743 ± 0.005	**0.790** ± 0.006
hapt	0.798 ± 0.001	0.791 ± 0.002	**0.800** ± 0.001	0.799 ± 0.000	0.799 ± 0.001
image	0.832 ± 0.002	0.753 ± 0.001	0.862 ± 0.003	0.859 ± 0.003	**0.927** ± 0.001
isolet	0.870 ± 0.004	0.853 ± 0.002	**0.888** ± 0.003	0.874 ± 0.003	0.867 ± 0.004
letter	**0.922** ± 0.005	0.742 ± 0.008	0.813 ± 0.012	0.811 ± 0.008	0.820 ± 0.004
madelon	0.520 ± 0.004	0.515 ± 0.002	0.557 ± 0.006	0.524 ± 0.003	**0.565** ± 0.002
mfeat	0.911 ± 0.003	0.922 ± 0.003	0.930 ± 0.001	0.913 ± 0.004	**0.941** ± 0.002
microv1	0.838 ± 0.004	0.851 ± 0.002	0.871 ± 0.002	0.849 ± 0.001	**0.937** ± 0.003
microv2	0.736 ± 0.005	0.746 ± 0.003	0.867 ± 0.006	0.845 ± 0.002	**0.873** ± 0.003
mnist	0.965 ± 0.005	0.958 ± 0.004	0.969 ± 0.002	0.975 ± 0.001	**0.979** ± 0.002
musk	0.800 ± 0.002	0.805 ± 0.000	0.880 ± 0.002	0.883 ± 0.001	**0.885** ± 0.000
nomao	**0.886** ± 0.002	0.877 ± 0.001	0.885 ± 0.001	0.877 ± 0.003	0.882 ± 0.002
semeion	0.716 ± 0.003	0.606 ± 0.005	0.727 ± 0.002	0.739 ± 0.005	**0.759** ± 0.001

It is important to note that in all cases the best results are obtained with the use of AEs.

5.5. C4.5

In this section, the classification results obtained for the C4.5 method are presented in Table 8. The C4.5 algorithm behaves similarly to the previous algorithms while using the new input space produced by AE models. On the one hand, the best results are obtained with the data generated by the AEs in most cases. Specifically, RAE generates the best performance in 12 out of 20 cases and CAE in 5 out of 20 datasets. On the other hand, in 3 cases the results obtained with the original data are the best. Unlike the previous algorithms, in this case the DAE model never obtains the best result. However, the performance of DAE is close to that obtained with RAE in many cases. This fact is best seen in Fig. 2(c).

The predictive performance of the C4.5 algorithm is improved when AEs are used. Higher-level feature generation positively affects tree-based algorithms. The methodology of the C4.5 algorithm is based on the analysis of the attributes that provide more information to separate the instances in classes [8]. In a high dimensionality space, the high number of features prevents this type of algorithms from selecting attributes that provide little information. For this reason, the use of dimensionality reduction methods is recommended.

The process followed in this study allows information to be merged from different features using AEs and generating new attributes by combining the originals. In this new scenario, the C4.5 algorithm has new features that contain more relevant information; therefore, the divisions performed are better and the predictive performance improves.

For C4.5, the results follow the line of the previous algorithms. The most sophisticated AE models generate the best predictive performance. These results justify the use of AEs to reduce dimensionality as a previous phase to that using tree-based algorithms, such as C4.5. The fusion of features allows for the generation of more relevant attributes that provide more useful information.

5.6. Results analysis

Once the results for each of the classification algorithms with the different AE models have been presented, the objective is to perform statistical tests that support the conclusions reached. This step is essential in order to confirm the results obtained and the differences between the different models.

Table 9
Average rankings of the different autoencoder models by classification method.

kNN		SVM		MLP		C4.5	
AE model	Ranking	AE model	Ranking	AE model	Ranking	AE model	Ranking
RAE	1.750	RAE	2.025	RAE	1.750	RAE	1.700
DAE	2.400	DAE	2.200	CAE	2.400	CAE	2.450
CAE	2.700	CAE	2.850	DAE	3.100	DAE	3.000
BAE	4.000	BAE	3.850	BAE	3.500	Plain	3.800
Plain	4.150	Plain	4.075	Plain	4.250	BAE	4.050

Table 10
Li post-hoc Friedman test for classification algorithm by autoencoder model.

		BAE	CAE	DAE	RAE	Plain
kNN	BAE	-	-	-	-	7.642E-01
	CAE	**3.803E-02**	-	-	-	**1.558E-02**
	DAE	**5.794E-03**	6.993E-01	-	-	**1.969E-03**
	RAE	**2.881E-05**	**1.958E-01**	4.508E-01	-	**6.728E-06**
	Plain	-	-	-	-	-
SVM	BAE	-	-	-	-	7.046E-01
	CAE	**1.426E-01**	-	-	-	**4.961E-02**
	DAE	**3.521E-03**	4.143E-01	-	-	**6.457E-04**
	RAE	**9.573E-04**	2.655E-01	7.263E-01	-	**1.509E-04**
	Plain	-	-	-	-	-
MLP	BAE	-	-	-	-	**1.882E-01**
	CAE	**4.603E-02**	-	**1.891E-01**	-	**3.739E-04**
	DAE	4.237E-01	-	-	-	**3.588E-02**
	RAE	**8.067E-04**	2.514E-01	**1.189E-02**	-	**9.948E-07**
	Plain	-	-	-	-	-
C4.5	BAE	-	-	-	-	-
	CAE	**3.576E-03**	-	4.147E-01	-	**1.779E-02**
	DAE	**8.534E-02**	-	-	-	2.225E-01
	RAE	**6.794E-06**	2.586E-01	**2.376E-02**	-	**6.969E-05**
	Plain	6.171E-01	-	-	-	-

To support the results obtained, it is necessary to check whether there are significant differences between the models. The fundamental objective is to check the differences between the most sophisticated models and the plain data, since the previous results determined that these cases were those that significantly improved predictive performance. To do this, first average ranks are obtained by applying the Friedman test [89]. The rankings for each classification algorithm are shown in Table 9.

Table 9 shows that for the four classification algorithms the best predictive performance is obtained with RAE. The DAE and CAE models are in second and third place, alternating their position according to the classification method. In addition, the worst results are obtained with the original data, except for C4.5, which is obtained with the BAE model. These rankings confirm the conclusions drawn in the previous subsections where it was indicated that the most sophisticated models offered better performance in most cases.

However, the previous rankings must be confirmed by statistical tests that verify that there are significant differences. This is necessary in order to be able to draw conclusions based on solid results. For this reason, the Li post-hoc [90] for Friedman test is carried out, comparing the different models with each other. Table 10 presents the different *p-values* obtained with the Li test for the four classification algorithms.

Table 10 shows that there are significant differences between most of the comparisons between the different models used in this experiment, setting the *p-value* threshold to the usual range [0.05, 0.1]. In general, the most sophisticated models present clear differences with respect to the BAE model and the model with plain data. The RAE, DAE and CAE models do not always have significant differences, since each model presents better results for some of the datasets. Similarly, there are no significant differences between the plain models and the most basic, since they are generally surpassed by more sophisticated models.

These results confirm that the more sophisticated AE models improve on the predictive performance obtained by the plain model using the four selected classification algorithms. There are no significant differences among the sophisticated models, although the rankings indicate that the RAE model works better.

5.7. Running time analysis

Once each of the classification algorithms has been analyzed in detail from a predictive performance perspective, an analysis of the execution time of the different configurations for each algorithm is carried out. The objective of this subsection is to verify the time improvements in the configurations as the size of the output space is reduced by AEs.

First, it is necessary to clarify the execution time studied in this Subsection. Hence, the following analysis only considers the time of the different classifiers. The training and compression time of the AEs models is not taken into account, since this process is performed only once for each model and architecture, generating a subset that can be used by any classifier in later stages. The reason for this approach is to highlight the performance improvement in terms of execution time obtained when classifying data of lower dimensionality. Thus the analysis is focused on studying the times grouped by the different architectures and classification algorithms.

Table 11 shows a series of rankings obtained after applying the Friedman test [89] for execution time. The data of all the datasets and AE models have been grouped, since the main interest in this section is to verify the time differences of each configuration. In addition, it is important to indicate that the times considered are averages for all the executions performed considering the 2x5 fold cross validation scheme. These averages have been used to develop the rankings presented.

As can be seen in Table 11, the best performance with respect to execution time is obtained with the configuration that reduces the number

Table 11
Average rankings considering execution time of the different autoencoder architecture by classification method.

kNN		SVM		MLP		C4.5	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
ARQ_25	1.031	ARQ_25	1.025	ARQ_25	1.012	ARQ_25	1.037
ARQ_50	2.056	ARQ_50	2.075	ARQ_50	2.000	ARQ_50	2.068
ARQ_75	2.912	ARQ_75	3.031	ARQ_75	3.012	ARQ_75	2.981
Plain	4.000	Plain	3.868	Plain	3.975	Plain	3.912

Table 12
Li post-hoc Friedman test for running time results by autoencoder architecture.

		ARQ_25	ARQ_50	ARQ_75	Plain
kNN	ARQ_25	-	5.128E-07	0.000E + 00	0.000E + 00
	ARQ_50	-	-	2.732E-05	0.000E + 00
	ARQ_75	-	-	-	9.949E-08
	Plain	-	-	-	-
SVM	ARQ_25	-	2.691E-07	0.000E + 00	0.000E + 00
	ARQ_50	-	-	2.805E-06	0.000E + 00
	ARQ_75	-	-	-	4.080E-05
	Plain	-	-	-	-
MLP	ARQ_25	-	1.313E-06	0.000E + 00	0.000E + 00
	ARQ_50	-	-	7.041E-06	0.000E + 00
	ARQ_75	-	-	-	2.414E-06
	Plain	-	-	-	-
C4.5	ARQ_25	-	4.370E-07	0.000E + 00	0.000E + 00
	ARQ_50	-	-	7.810E-06	0.000E + 00
	ARQ_75	-	-	-	5.063E-06
	Plain	-	-	-	-

of features to 25% of the total for all the algorithms. In a similar manner, the execution time increases as compression decreases. In this way, the worst results are obtained with the plain model, that is the model that does not use AEs to reduce dimensionality.

Finally, it is necessary to determine whether there are statistically significant differences in order to establish solid and well-founded results. To this end, Li post-hoc tests [90] for the Friedman test are applied. Table 12 presents the different *p-values* generated using the Li test.

The results obtained from the Li test, shown in Table 12, show that there are significant differences between the configurations considered in this study, setting the *p-value* threshold to the usual range [0.05, 0.1]. These differences are given in all cases compared.

Therefore, the fact that the execution time decreases as the compression of the data increases is confirmed by solid results. In conclusion, the use of AEs to reduce dimensionality entails a considerable reduction in terms of execution time. Thus, the selection of the architecture may depend on the relevance of the execution time in the problem in question. If it is important to reduce the time, an architecture with fewer units in the hidden layer can be selected, although this implies lower predictive performance.

6. Autoencoders vs classical feature extraction techniques

Finally, the objective of this section is to assess the competitiveness of AE models as compared against traditional dimensionality reduction methods. Specifically, four of the most well-known and widely-used algorithms have been selected, PCA, LDA, ISOMAP and LLE. To do so, a comparison has been made between the most sophisticated models of AE (DAE, CAE and RAE) and the results obtained with PCA, LDA, ISOMAP and LLE on the same datasets. The configuration selected in all cases corresponds to that established in Section 4.2 and provides the best performance. It is important for the number of features obtained through the different methods to be the same in order to make a reliable comparison. Similarly, this contrast has been made considering the four classification algorithms used in the experimentation.

Tables 13–16 present the classification results of the algorithms kNN, SVM, MLP and C4.5, respectively. Each table contains the data obtained after applying the 7 methods of dimensionality reduction: CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE. In each case, the best result is highlighted in bold.

Observing the results presented in Tables 13–16, it can be seen that the best results are obtained with the AE models for most of the cases. Specifically, for kNN there are only three datasets (*arcene, microv1, musk*) where the best result is obtained with traditional algorithms and another dataset (*mfeat*) where CAE and ISOMAP have the same result. The SVM and MLP algorithms show similar behavior, there are two datasets (*arcene, microv1*) where the best results are obtained with traditional algorithms and another dataset (*coil2000*) where all the results obtained are similar. Finally, considering the C4.5 algorithm, it can be observed that there are three datasets (*arcene, drive, madelon*) where the best predictive performance is obtained from the data generated by traditional dimensionality reduction algorithms. Therefore, a general trend in all classification algorithms can be observed that indicates that AE-based methods work better than LDA, PCA, ISOMAP and LLE.

However, it is necessary to confirm that these differences are statistically significant. To do so, the Friedman test is applied first [89]. The average rankings obtained are presented in Table 17. Then, a series of performance classifications of the different dimensionality reduction methods are generated.

The different rankings presented in 17 show that the best models are those based on AEs, while the traditional algorithms generate the worst results from an overall point of view. This trend is maintained in all the classification algorithms used in the experiment.

The second step when verifying the differences between the models is to verify whether there are significant differences or not. In order to do so, Li post-hoc tests [90] for the Friedman test are used to compare the different models. Table 18 presents the *p-values* obtained with the Li test.

Table 18 shows that there are significant differences between most AE models with traditional models in the cases considered, setting the *p-value* threshold to the usual range [0.05, 0.1]. However, there are some AE models that do not show significant differences with LDA, ISOMAP and LLE in some classification algorithms. Nevertheless, the RAE model shows significant differences with LDA, PCA, ISOMAP and LLE in the four classification algorithms considered. This confirms the trend that indicates that the RAE model produces the best results from an overall point of view.

In summary, the data presented in this section show that dimensionality reduction models based on AEs generate a predictive performance superior to traditional models such as LDA, PCA ISOMAP and LLE. This is mainly due to the fact that models based on AEs generate more relevant features and provide the classification algorithms with more useful information. This allows us to consider this type of model as an option to take into account when tackling this task.

7. General guidelines on the use of autoencoder models

In the previous sections an exhaustive experimentation on the performance of several classifiers with data reduced using AE models and the different conclusions reached in each case have been presented. This section aims to analyze in more detail the relationship between the characteristics of the datasets and the results obtained. As a result, a series of guidelines are presented that allow the most appropriate AE model to be selected according to the classification algorithm and the traits of the data specific to each problem. To this end, the most relevant recommendation for each algorithm is presented in the following list. In addition, Table 19 presents a summary of the main ideas.

KNN:

- For a dataset with a very high dimensionality, the best results are obtained with DAE and RAE models. Therefore, we recommend using them for datasets with more than 1000 features.
- The most recommended model when using datasets with a number of features between 500 and 1000 is RAE. However, CAE model also offers good results in certain cases.
- If datasets have between 100 and 500 features, RAE model generally performs well. Similarly, DAE and CAE models obtain good results with binary datasets.
- The models that generate the best predictive performance for datasets with less than 100 features are RAE and CAE.

SVM:

Table 13
kNN classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.605 ± 0.015	0.672 ± 0.005	0.743 ± 0.008	0.661 ± 0.006	0.654 ± 0.009	0.738 ± 0.003	**0.747** ± 0.005
batch	0.900 ± 0.010	0.990 ± 0.001	**0.992** ± 0.001	0.861 ± 0.005	0.852 ± 0.007	0.786 ± 0.004	0.862 ± 0.006
coil2000	**0.554** ± 0.004	**0.554** ± 0.004	0.545 ± 0.003	0.523 ± 0.006	0.525 ± 0.003	0.525 ± 0.003	0.502 ± 0.002
dota	0.519 ± 0.000	**0.522** ± 0.003	0.516 ± 0.002	0.513 ± 0.003	0.517 ± 0.001	0.516 ± 0.002	0.516 ± 0.002
drive	**0.877** ± 0.005	0.846 ± 0.006	0.873 ± 0.004	0.683 ± 0.011	0.782 ± 0.006	0.693 ± 0.003	0.726 ± 0.008
facial	0.690 ± 0.006	0.685 ± 0.003	**0.697** ± 0.002	0.631 ± 0.005	0.648 ± 0.004	0.601 ± 0.004	0.602 ± 0.003
fashionmnist	0.912 ± 0.001	0.920 ± 0.003	**0.922** ± 0.001	0.831 ± 0.004	0.911 ± 0.002	0.842 ± 0.003	0.896 ± 0.001
gisette	0.897 ± 0.004	**0.923** ± 0.002	0.900 ± 0.004	0.855 ± 0.003	0.883 ± 0.006	0.877 ± 0.003	0.878 ± 0.005
hapt	0.876 ± 0.005	0.931 ± 0.003	**0.939** ± 0.002	0.553 ± 0.012	0.903 ± 0.004	0.750 ± 0.003	0.781 ± 0.006
image	0.952 ± 0.002	0.950 ± 0.001	**0.958** ± 0.003	0.734 ± 0.011	0.779 ± 0.004	0.882 ± 0.007	0.874 ± 0.002
isolet	0.974 ± 0.001	0.968 ± 0.002	**0.976** ± 0.002	0.659 ± 0.006	0.971 ± 0.002	0.962 ± 0.003	0.927 ± 0.001
letter	0.925 ± 0.012	0.923 ± 0.006	**0.947** ± 0.005	0.882 ± 0.005	0.893 ± 0.003	0.894 ± 0.003	0.883 ± 0.004
madelon	0.566 ± 0.006	0.591 ± 0.003	**0.618** ± 0.005	0.523 ± 0.002	0.505 ± 0.005	0.506 ± 0.003	0.501 ± 0.001
mfeat	**0.986** ± 0.002	0.984 ± 0.001	0.984 ± 0.001	0.963 ± 0.003	0.972 ± 0.002	0.985 ± 0.001	**0.986** ± 0.002
microv1	0.922 ± 0.003	0.939 ± 0.005	0.929 ± 0.004	0.532 ± 0.006	0.897 ± 0.002	**0.947** ± 0.000	0.946 ± 0.001
microv2	**0.955** ± 0.002	0.932 ± 0.005	0.954 ± 0.003	0.632 ± 0.005	0.891 ± 0.003	0.948 ± 0.002	0.951 ± 0.002
mnist	0.969 ± 0.000	0.975 ± 0.001	**0.979** ± 0.000	0.828 ± 0.002	0.966 ± 0.003	0.923 ± 0.001	0.944 ± 0.002
musk	0.940 ± 0.002	0.941 ± 0.001	0.947 ± 0.002	0.912 ± 0.004	**0.964** ± 0.001	0.938 ± 0.003	0.901 ± 0.005
nomao	**0.914** ± 0.005	0.904 ± 0.002	0.905 ± 0.003	0.797 ± 0.004	0.817 ± 0.006	0.804 ± 0.003	0.849 ± 0.005
semeion	0.907 ± 0.004	0.940 ± 0.001	**0.942** ± 0.000	0.732 ± 0.003	0.926 ± 0.005	0.894 ± 0.005	0.884 ± 0.003

Table 14
SVM classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.500 ± 0.000	0.632 ± 0.005	0.500 ± 0.000	0.531 ± 0.003	0.545 ± 0.004	0.721 ± 0.002	**0.722** ± 0.003
batch	0.924 ± 0.004	0.980 ± 0.002	**0.985** ± 0.002	0.983 ± 0.001	0.980 ± 0.002	0.983 ± 0.001	0.892 ± 0.003
coil2000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	0.493 ± 0.002	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000
dota	**0.553** ± 0.001	0.550 ± 0.002	0.534 ± 0.005	0.517 ± 0.005	0.529 ± 0.003	0.547 ± 0.002	0.548 ± 0.000
drive	**0.985** ± 0.003	0.941 ± 0.002	0.863 ± 0.004	0.978 ± 0.001	0.979 ± 0.002	0.981 ± 0.002	0.982 ± 0.001
facial	0.701 ± 0.004	0.713 ± 0.002	**0.714** ± 0.005	0.672 ± 0.005	0.703 ± 0.003	0.697 ± 0.005	0.501 ± 0.007
fashionmnist	0.938 ± 0.003	0.940 ± 0.000	**0.942 ± 0.001**	0.921 ± 0.003	0.935 ± 0.001	0.929 ± 0.001	0.934 ± 0.002
gisette	0.816 ± 0.001	**0.969** ± 0.003	0.955 ± 0.006	0.925 ± 0.002	0.941 ± 0.004	0.962 ± 0.001	0.963 ± 0.003
hapt	0.871 ± 0.003	**0.889** ± 0.001	0.886 ± 0.004	0.788 ± 0.005	0.875 ± 0.003	0.859 ± 0.003	0.869 ± 0.001
image	0.886 ± 0.001	0.877 ± 0.005	**0.910** ± 0.002	0.850 ± 0.004	0.862 ± 0.001	0.784 ± 0.005	0.620 ± 0.003
isolet	0.971 ± 0.002	0.975 ± 0.001	**0.976** ± 0.001	0.958 ± 0.003	0.965 ± 0.001	0.963 ± 0.001	0.966 ± 0.003
letter	0.841 ± 0.004	0.828 ± 0.004	**0.879** ± 0.005	0.812 ± 0.003	0.839 ± 0.005	0.842 ± 0.004	0.852 ± 0.002
madelon	0.580 ± 0.002	**0.591** ± 0.001	0.587 ± 0.000	0.557 ± 0.003	0.585 ± 0.002	0.547 ± 0.004	0.523 ± 0.004
mfeat	**0.985** ± 0.002	**0.985** ± 0.001	**0.985** ± 0.001	0.926 ± 0.003	0.962 ± 0.003	0.980 ± 0.002	0.981 ± 0.000
microv1	0.796 ± 0.003	0.918 ± 0.005	0.786 ± 0.002	0.731 ± 0.005	0.876 ± 0.004	**0.953** ± 0.002	**0.953** ± 0.002
microv2	**0.916** ± 0.003	0.871 ± 0.001	0.881 ± 0.001	0.652 ± 0.010	0.791 ± 0.008	0.832 ± 0.009	0.835 ± 0.011
mnist	0.984 ± 0.001	0.985 ± 0.001	**0.987** ± 0.000	0.971 ± 0.002	0.979 ± 0.002	0.829 ± 0.005	0.976 ± 0.001
musk	0.939 ± 0.005	0.948 ± 0.006	**0.964** ± 0.003	0.955 ± 0.003	0.958 ± 0.001	0.959 ± 0.001	0.954 ± 0.002
nomao	**0.935** ± 0.000	0.924 ± 0.002	0.932 ± 0.001	0.929 ± 0.000	0.930 ± 0.001	0.933 ± 0.001	0.925 ± 0.003
semeion	0.957 ± 0.003	0.957 ± 0.001	**0.962** ± 0.001	0.936 ± 0.005	0.957 ± 0.002	0.929 ± 0.003	0.949 ± 0.003

Table 15
MLP classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.465 ± 0.005	0.500 ± 0.000	0.523 ± 0.001	0.521 ± 0.004	0.512 ± 0.003	**0.691** ± 0.006	0.673 ± 0.004
batch	**0.929** ± 0.002	0.920 ± 0.003	0.920 ± 0.002	0.915 ± 0.001	0.512 ± 0.003	0.919 ± 0.004	0.566 ± 0.001
coil2000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000
dota	0.529 ± 0.003	0.500 ± 0.000	**0.547** ± 0.005	0.524 ± 0.002	0.525 ± 0.004	0.527 ± 0.002	0.525 ± 0.001
drive	**0.964** ± 0.004	0.680 ± 0.010	0.812 ± 0.006	0.941 ± 0.003	0.883 ± 0.004	0.802 ± 0.005	0.809 ± 0.004
facial	0.706 ± 0.003	0.701 ± 0.004	**0.718** ± 0.003	0.692 ± 0.002	0.704 ± 0.002	0.707 ± 0.001	0.697 ± 0.004
fashionmnist	0.915 ± 0.001	0.911 ± 0.004	**0.917** ± 0.002	0.902 ± 0.003	0.909 ± 0.001	0.898 ± 0.004	0.905 ± 0.004
gisette	0.591 ± 0.002	**0.627** ± 0.003	0.614 ± 0.003	0.602 ± 0.001	0.612 ± 0.003	0.609 ± 0.004	0.605 ± 0.002
hapt	0.838 ± 0.002	**0.850** ± 0.001	0.841 ± 0.003	0.834 ± 0.005	0.839 ± 0.003	0.840 ± 0.004	0.820 ± 0.005
image	0.884 ± 0.003	0.872 ± 0.003	**0.898** ± 0.001	0.881 ± 0.003	0.552 ± 0.010	0.672 ± 0.008	0.645 ± 0.009
isolet	0.850 ± 0.006	0.842 ± 0.007	**0.880** ± 0.006	0.849 ± 0.005	0.819 ± 0.006	0.791 ± 0.009	0.805 ± 0.005
letter	**0.749** ± 0.003	0.740 ± 0.002	0.742 ± 0.005	0.713 ± 0.005	0.732 ± 0.002	0.739 ± 0.006	0.722 ± 0.002
madelon	0.590 ± 0.001	**0.637** ± 0.004	0.631 ± 0.002	0.577 ± 0.008	0.582 ± 0.005	0.556 ± 0.005	0.512 ± 0.007
mfeat	0.938 ± 0.003	0.968 ± 0.001	**0.976** ± 0.000	0.921 ± 0.003	0.953 ± 0.004	0.974 ± 0.001	0.964 ± 0.002
microv1	0.713 ± 0.004	0.690 ± 0.005	0.714 ± 0.003	0.694 ± 0.003	0.701 ± 0.005	0.829 ± 0.006	**0.868** ± 0.004
microv2	**0.747** ± 0.002	0.725 ± 0.004	0.742 ± 0.003	0.731 ± 0.002	0.720 ± 0.002	0.740 ± 0.001	0.733 ± 0.003
mnist	0.896 ± 0.000	0.905 ± 0.002	**0.909** ± 0.001	0.902 ± 0.002	0.906 ± 0.001	0.893 ± 0.000	0.896 ± 0.003
musk	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	0.992 ± 0.002	0.991 ± 0.002	0.994 ± 0.001	0.952 ± 0.001
nomao	**0.923** ± 0.003	0.530 ± 0.011	0.913 ± 0.003	0.913 ± 0.002	0.915 ± 0.003	0.911 ± 0.005	0.914 ± 0.003
semeion	**0.860** ± 0.005	0.756 ± 0.002	0.793 ± 0.006	0.820 ± 0.004	0.849 ± 0.002	0.746 ± 0.005	0.822 ± 0.003

Table 16
C4.5 classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.565 ± 0.006	0.678 ± 0.003	0.730 ± 0.004	0.654 ± 0.005	0.662 ± 0.004	**0.766** ± 0.003	0.695 ± 0.004
batch	0.911 ± 0.003	0.982 ± 0.000	**0.983** ± 0.001	0.971 ± 0.003	0.973 ± 0.002	0.980 ± 0.001	0.975 ± 0.002
coil2000	**0.521** ± 0.001	0.516 ± 0.002	0.513 ± 0.004	0.509 ± 0.002	0.504 ± 0.002	0.507 ± 0.001	0.500 ± 0.000
dota	**0.548** ± 0.003	0.539 ± 0.002	0.521 ± 0.005	0.527 ± 0.003	0.531 ± 0.005	0.503 ± 0.005	0.515 ± 0.003
drive	0.944 ± 0.003	0.850 ± 0.004	0.867 ± 0.005	0.936 ± 0.003	0.938 ± 0.003	**0.954** ± 0.001	0.947 ± 0.002
facial	0.660 ± 0.005	0.655 ± 0.013	**0.683** ± 0.008	0.626 ± 0.003	0.651 ± 0.005	0.635 ± 0.007	0.632 ± 0.004
fashionmnist	0.883 ± 0.003	0.888 ± 0.001	**0.890** ± 0.000	0.872 ± 0.002	0.883 ± 0.000	0.879 ± 0.001	0.881 ± 0.001
gisette	0.714 ± 0.003	0.743 ± 0.005	**0.790** ± 0.006	0.682 ± 0.003	0.705 ± 0.005	0.692 ± 0.003	0.709 ± 0.004
hapt	**0.800** ± 0.001	0.799 ± 0.000	0.799 ± 0.001	0.797 ± 0.002	0.791 ± 0.003	0.792 ± 0.002	0.788 ± 0.004
image	0.862 ± 0.003	0.859 ± 0.003	**0.927** ± 0.001	0.913 ± 0.002	0.922 ± 0.002	0.774 ± 0.004	0.838 ± 0.003
isolet	**0.888** ± 0.003	0.874 ± 0.003	0.867 ± 0.004	0.840 ± 0.004	0.859 ± 0.004	0.851 ± 0.005	0.862 ± 0.002
letter	0.813 ± 0.012	0.811 ± 0.008	**0.820** ± 0.004	0.783 ± 0.010	0.811 ± 0.004	0.793 ± 0.005	0.805 ± 0.003
madelon	0.557 ± 0.006	0.524 ± 0.003	0.565 ± 0.002	0.626 ± 0.007	**0.677** ± 0.003	0.652 ± 0.005	0.631 ± 0.005
mfeat	0.930 ± 0.001	0.913 ± 0.004	**0.941** ± 0.002	0.889 ± 0.002	0.898 ± 0.003	0.892 ± 0.001	0.895 ± 0.002
microv1	0.871 ± 0.002	0.849 ± 0.001	**0.937** ± 0.003	0.731 ± 0.010	0.806 ± 0.003	0.685 ± 0.005	0.868 ± 0.002
microv2	0.867 ± 0.006	0.845 ± 0.002	**0.873** ± 0.003	0.835 ± 0.005	0.837 ± 0.005	0.815 ± 0.006	0.838 ± 0.003
mnist	0.969 ± 0.002	0.975 ± 0.001	**0.979** ± 0.002	0.891 ± 0.003	0.927 ± 0.002	0.897 ± 0.003	0.915 ± 0.001
musk	0.880 ± 0.002	0.883 ± 0.001	**0.885** ± 0.000	0.869 ± 0.004	0.872 ± 0.002	0.876 ± 0.004	0.871 ± 0.003
nomao	**0.885** ± 0.001	0.877 ± 0.003	0.882 ± 0.002	0.881 ± 0.001	0.862 ± 0.000	0.767 ± 0.003	0.872 ± 0.003
semeion	0.727 ± 0.002	0.739 ± 0.005	**0.759** ± 0.001	0.729 ± 0.000	0.739 ± 0.003	0.744 ± 0.003	0.731 ± 0.002

Table 17
Average rankings considering CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE by classification method.

kNN		SVM		MLP		C4.5	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
RAE	1.925	RAE	2.600	RAE	2.250	RAE	2.025
DAE	2.700	DAE	3.150	CAE	3.025	CAE	3.075
CAE	2.750	CAE	3.550	DAE	4.125	DAE	3.275
LDA	4.475	LDA	4.200	ISOMAP	4.350	LDA	4.275
LLE	4.775	ISOMAP	4.275	LDA	4.425	LLE	4.750
ISOMAP	4.925	LLE	4.300	LLE	4.800	ISOMAP	4.950
PCA	6.450	PCA	5.925	PCA	5.025	PCA	5.650

Table 18
Li post-hoc Friedman test for dimensionality reduction methods by classification algorithm.

		CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
kNN	CAE	-	-	-	**4.260E-07**	**1.862E-02**	**3.810E-03**	**6.360E-03**
	DAE	9.417E-01	-	-	**4.234E-07**	**1.634E-02**	**3.373E-03**	**5.558E-03**
	RAE	2.870E-01	3.067E-01	-	**7.346E-10**	**6.626E-04**	**5.908E-05**	**1.268E-04**
	PCA	-	-	-		-	-	-
	LDA	-	-	-	**7.316E-03**	-	5.650E-01	6.970E-01
	ISOMAP	-	-	-	**3.564E-02**	-	-	-
	LLE	-	-	-	**2.124E-02**	-	8.408E-01	
SVM	CAE	-	-	-	**3.549E-03**	4.219E-01	3.792E-01	3.792E-01
	DAE	6.144E-01	-	-	**5.103E-04**	1.972E-01	**1.840E-01**	**1.840E-01**
	RAE	2.517E-01	4.906E-01	-	**2.376E-05**	**5.924E-02**	**5.924E-02**	**5.924E-02**
	PCA	-	-	-	-	-	-	-
	LDA	-	-	-	**5.924E-02**	-	9.226E-01	9.072E-01
	ISOMAP	-	-	-	**5.924E-02**	-	-	9.708E-01
	LLE	-	-	-	**5.924E-02**	-	-	-
MLP	CAE	-	2.122E-01	-	**1.426E-02**	**1.027E-01**	**1.181E-01**	**2.784E-02**
	DAE	-	-	-	3.276E-01	7.165E-01	7.762E-01	4.676E-01
	RAE	4.048E-01	**2.104E-02**	-	**1.020E-03**	**1.013E-02**	**1.104E-02**	**1.986E-03**
	PCA	-	-	-	-	-	-	-
	LDA	-	-	-	4.876E-01	-	-	6.606E-01
	ISOMAP	-	-	-	4.676E-01	9.126E-01	-	6.080E-01
	LLE	-	-	-	7.762E-01	-	-	-
C4.5	CAE	-	7.860E-01	-	**8.588E-04**	**1.245E-01**	**1.806E-02**	**3.687E-02**
	DAE	-	-	-	**2.131E-03**	**1.946E-01**	**3.687E-02**	**6.366E-02**
	RAE	**1.805E-01**	**1.147E-01**	-	**2.347E-06**	**3.457E-03**	**1.947E-04**	**4.644E-04**
	PCA	-	-	-	-	-	-	-
	LDA	-	-	-	**8.257E-02**	-	3.657E-01	5.217E-01
	ISOMAP	-	-	-	**3.626E-01**	-	-	-
	LLE	-	-	-	2.388E-01	-	7.860E-01	

Table 19
Autoencoder recommended considering number of feature and classifier.

Number of features	Classifiers			
	kNN	SVM	MLP	C4.5
> 1000	RAE DAE	DAE	RAE CAE (large number of classes)	RAE
> 500 - < 1000	RAE CAE	RAE (non-binary) DAE (binary)	RAE (non-binary) DAE (binary)	RAE CAE (more 10 classes)
> 100 - < 500	RAE CAE-DAE (binary)	RAE CAE (binary)	CAE RAE (binary)	RAE CAE (binary)
< 100	RAE CAE	RAE CAE	RAE CAE	RAE CAE

- When working with a dataset with very high dimensionality we recommend using DAE model. This model has obtained the best results for datasets with more than 1000 features.
- The model that works best when starting from a binary dataset with a number of features between 500 and 1000 is DAE. However, if it is a non-binary dataset the best model is RAE.
- RAE model offers good results for datasets with between 100 and 500 features. Similarly, CAE model demonstrates good predictive performance with binary datasets.
- CAE and RAE are the most recommended models if datasets with less than 100 features are used.

MLP:

- The model with the best performance is RAE when working with a dataset with a number greater than 1000 features. However, CAE model offers good results when working with datasets with a high number of classes.
- The best results are obtained with RAE model for a non-binary dataset with a number of features between 500 and 1000. However, the best model is DAE when using a binary dataset.
- The model that generates the best predictive performance when datasets have between 100 and 500 features is CAE. However, the best model when using several binary datasets is RAE.
- We recommend using CAE and RAE models when using datasets with less than 100 features.

C4.5:

- Based on this study, RAE is recommended for a dataset with a very high dimensionality (with more than 1000 features).
- When working with datasets that has between 500 and 1000 characteristics, it is advisable to use RAE and CAE models when the number of classes is greater than 10. However, RAE is the best model when using a binary dataset.
- The model that generates the best predictive performance is RAE if the number of features of the datasets is between 100 and 500. Similarly, CAE model works well with some binary datasets.
- According to the generated results, we recommend using CAE and RAE models when using datasets with less than 100 features.

Time:

- The results presented in this study indicate that, in general, the highest predictive performance is obtained with configurations that compress the data less. However, sometimes it is important to reduce execution times. In these cases a configuration that compresses more data may be more useful, despite a loss in predictive performance.

In summary, the AE model used to carry out the reduction of dimensionality must be adapted to the data traits, as well as to the methodology of the classification algorithm that is used. Therefore, the above advice has been presented according to the experience provided by this study.

8. Concluding remarks

In this paper an exhaustive study has been carried out on the performance of AEs when tackling the task of dimensionality reduction. This task is one of the challenges faced by machine learning, due to the effects of high dimensionality of the data in many current processes. Specifically, this study focuses on the task of classification. In this context there are several different classification methodologies. Therefore, experimentation is necessary to incorporate algorithms that cover several of the most widely used methodologies in order to provide a general vision of the performance of AEs. For this reason the most well known algorithms with the best results from the most important methodologies have been selected, specifically kNN, SVM, MLP and C4.5.

In order to undertake the task of dimensionality reduction, AEs have been used. The main reason is the good results that they have obtained when generating high level features, which is fundamental to improving the subsequent predictive performance. In this case, in order to give a broader view of the performance of AEs four different models were used, starting from the most basic model and generating three more sophisticated models, which give a global vision of the purchase of AEs.

To assess the performance of the different AE models a thorough experimentation on 20 datasets was designed combining the four AE models with the four classification algorithms. New feature spaces are generated for each dataset using the different AE models, which are used as input for the different classification algorithms. In particular, in the first part of the experimentation we determined which AE architecture generated the best performance. In the second part, the results generated by the best architecture have been compared with the results obtained from the original data.

The results obtained after performing the experiments show that predictive performance improves when using AE models to reduce the dimensionality of the input space. In particular, the most sophisticated AE models significantly improve the results obtained with the most basic model and with the original data. In a similar way, the execution time is significantly improved when using the different AE models by reducing the input space and the computational load of the different classification algorithms. The conclusions reached have been supported by a series of statistical tests that confirm the existence of significant differences between the different models.

In addition, AE models have been compared with traditional dimensionality reduction methods, with the aim of verifying their performance in relation to other models that have previously performed well in this task. For this reason, AE models have been compared with LDA, PCA, ISOMAP and LLE. The conclusions obtained indicate that sophisticated models based on AEs have a better predictive performance than the traditional models analyzed. This may indicate that the features generated by methods based on AEs are more significant and provide more relevant information to the classification algorithms.

Finally, the study leads to a series of guidelines with the aim of facilitating the choice of the most appropriate AE models according to the classification algorithm and the characteristics of the input data. It is

important to consider these factors in order to maximize the predictive performance of the model used.

Acknowledgments

This study by F. Pulgar was supported by the Spanish Ministry of Education under the FPU National Program (Ref. FPU16/00324). This work was partially supported by the Spanish Ministry of Economy and Competitiveness under project TIN2015-68454-R.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley, New York, 1973.
[2] R. Kohavi, F. Provost, Glossary of terms, Mach. Learn. 30 (2–3) (1998) 271–274, doi:10.1023/A:1017181826899.
[3] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99, doi:10.1023/A:1022602019183.
[4] S.B. Kotsiantis, Supervised machine learning: a review of classification techniques, Informatica 31 (2007) 249–268.
[5] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66, doi:10.1023/A:1022689900470.
[6] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.
[7] R.J. Schalkoff, Artificial Neural Networks, vol. 1, McGraw-Hill, New York, 1997.
[8] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[9] R. Bellman, Dynamic Programming, Princeton University Press, 1957.
[10] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.
[11] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) 42 (4) (2012) 463–484.
[12] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsletter 6 (1) (2004) 20–29.
[13] M. Dash, H. Liu, Feature selection for classification, Intell. Data Anal. 1 (3) (1997) 131–156, doi:10.3233/IDA-1997-1302.
[14] H. Yu, J. Yang, A direct lda algorithm for high-dimensional data-with application to face recognition, Pattern Recogn. 34 (10) (2001) 2067–2070, doi:10.1016/S0031-3203(00)00162-X.
[15] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, Lond. Edinb. Dublin Philos.Mag. J. Sci. 2 (11) (1901) 559–572, doi:10.1080/14786440109462720.
[16] J.B. Tenenbaum, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323, doi:10.1126/science.290.5500.2319.
[17] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326, doi:10.1126/science.290.5500.2323.
[18] P. Domingos, A few useful things to know about machine learning, Commun. ACM 55 (10) (2012) 78–87, doi:10.1145/2347736.2347755.
[19] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, Pattern Anal. Mach. Intell. IEEE Trans. 35 (8) (2013) 1798–1828, doi:10.1109/TPAMI.2013.50.
[20] S. Garca, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer, 2015, doi:10.1007/978-3-319-10247-4.
[21] I. Guyon, A. Elisseeff, An Introduction to Feature Extraction, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–25, doi:10.1007/978-3-540-35488-8_1.
[22] U.G. Mangai, S. Samanta, S. Das, P.R. Chowdhury, A survey of decision fusion and feature fusion strategies for pattern classification, IETE Tech. Rev. 27 (4) (2010) 293–307, doi:10.4103/0256-4602.64604.
[23] G. Lin, H. Zhu, X. Kang, C. Fan, E. Zhang, Feature structure fusion and its application, Inf. Fusion 20 (2014) 146–154, doi:10.1016/j.inffus.2014.01.002.
[24] Y. Du, W. Song, Q. He, D. Huang, A. Liotta, C. Su, Deep learning with multi-scale feature fusion in remote sensing for automatic oceanic eddy detection, Inf. Fusion 49 (2019) 89–99, doi:10.1016/j.inffus.2018.09.006.
[25] Q. Zhang, L.T. Yang, Z. Chen, P. Li, A survey on deep learning for big data, Inf. Fusion 42 (2018) 146–157, doi:10.1016/j.inffus.2017.10.006.
[26] L. Deng, Deep learning: methods and applications, Found. Trend. Signal Process. 7 (3–4) (2014) 197–387, doi:10.1561/2000000039.
[27] Y. Bengio, Deep learning of representations: looking forward, in: International Conference on Statistical Language and Speech Processing, 2013, pp. 1–37, doi:10.1007/978-3-642-39593-2_1.
[28] D. Charte, F. Charte, S. Garca, M.J. del Jesus, F. Herrera, A practical tutorial on autoencoders for nonlinear feature fusion: taxonomy, models, software and guidelines, Inf. Fusion 44 (2018) 78–96, doi:10.1016/j.inffus.2017.12.007.
[29] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, doi:10.1126/science.1127647.
[30] F.J. Pulgar, F. Charte, A.J. Rivera, M.J.D. Jesus, Aeknn: an autoencoder knn-based classifier with built-in dimensionality reduction, Int. J. Comput. Intell. Syst. 12 (11) (2018) 436–452, doi:10.2991/ijcis.2018.125905686.
[31] H.F. Nweke, Y.W. Teh, G. Mujtaba, M.A. Al-garadi, Data fusion and multiple classifier systems for human activity detection and health monitoring: review and open research directions, Inf. Fusion 46 (2019) 147–170, doi:10.1016/j.inffus.2018.06.002.
[32] Z. Chen, W. Li, Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network, IEEE Trans. Instrument. Measur. 66 (7) (2017) 1693–1702, doi:10.1109/tim.2017.2669947.
[33] V. Singh, N.K. Verma, Z.U. Islam, Y. Cui, Feature learning using stacked autoencoder for shared and multimodal fusion of medical images, in: Computational Intelligence: Theories, Applications and Future Directions - Volume I, Springer Singapore, Singapore, 2019, pp. 53–66, doi:10.1007/978-981-13-1132-1_5.
[34] S. Maurya, V. Singh, S. Dixit, N.K. Verma, A. Salour, J. Liu, Fusion of low-level features with stacked autoencoder for condition based monitoring of machines, in: IEEE International Conference on Prognostics and Health Management (ICPHM), 2018, pp. 1–8, doi:10.1109/ICPHM.2018.8448969.
[35] J. López, A.S. Garea, D.B. Heras, F. Argüello, Stacked autoencoders for multiclass change detection in hyperspectral images, in: IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2018, pp. 1906–1909, doi:10.1109/IGARSS.2018.8518338.
[36] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybern. 59 (4–5) (1988) 291–294, doi:10.1007/BF00332918.
[37] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 1096–1103, doi:10.1145/1390156.1390294.
[38] S. Rifai, Y. Bengio, Y. Dauphin, P. Vincent, A generative process for sampling contractive auto-encoders, in: Proceedings of the 29th International Conference on Machine Learning, ICML, 2012. Vol. 2, 2012
[39] Y. Qi, Y. Wang, X. Zheng, Z. Wu, Robust feature learning by stacked autoencoder with maximum correntropy criterion, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6716–6720, doi:10.1109/ICASSP.2014.6854900.
[40] D. Heckerman, D. Geiger, D.M. Chickering, Learning bayesian networks: the combination of knowledge and statistical data, Mach. Learn. 20 (3) (1995) 197–243, doi:10.1007/BF00994016.
[41] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Syst. Man Cybern. SMC- 3 (1) (1973) 28–44, doi:10.1109/TSMC.1973.5408575.
[42] L. Davis, Handbook of genetic algorithms, CUMINCAD (1991).
[43] S. Muggleton, Inductive logic programming, New Generat. Comput. 8 (4) (1991) 295–318, doi:10.1007/BF03037089.
[44] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. 46 (3) (1992) 175–185, doi:10.1080/00031305.1992.10475879.
[45] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27, doi:10.1109/TIT.1967.1053964.
[46] C.G. Atkeson, A.W. Moorey, S. Schaalz, A.W. Moore, S. Schaal, Locally weighted learning, Artif. Intell. 11 (1997) 11–73, doi:10.1023/A:1006559212014.
[47] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[48] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533, doi:10.1038/323533a0.
[49] M.W. Gardner, S. Dorling, Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences, Atmos. Environ. 32 (14–15) (1998) 2627–2636.
[50] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, ACM, 1992, pp. 144–152.
[51] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011. ISBN 0123748569.
[52] G.F. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Trans. Inf. Theory 14 (1) (1968) 55–63, doi:10.1109/TIT.1968.1054102.
[53] W. Liu, X. Yang, D. Tao, J. Cheng, Y. Tang, Multiview dimension reduction via hessian multiset canonical correlations, Inf. Fusion 41 (2018) 119–128, doi:10.1016/j.inffus.2017.09.001.
[54] R.A. Fisher, The statistical utilization of multiple measurements, Annal. Eugen. 8 (4) (1938) 376–386, doi:10.1111/j.1469-1809.1938.tb02189.x.
[55] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 24 (6) (1933) 417–441, doi:10.1037/h0071325.
[56] W. Wang, Y. Huang, Y. Wang, L. Wang, Generalized autoencoder: a neural network framework for dimensionality reduction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 496–503, doi:10.1109/CVPRW.2014.79.
[57] H. Liu, H. Motoda, Feature extraction, construction and selection, A Data Mining Perspective, Vol. 453, Springer Science & Business Media, 1998, doi:10.1007/978-1-4615-5725-8.
[58] L. Cayton, Algorithms for manifold learning, University of California at San Diego Tech. Rep2005, 12, 1–17, 1.
[59] J.A. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer Science & Business Media, 2007.
[60] H. Schwenk, Y. Bengio, Training methods for adaptive boosting of neural networks, in: Advances in Neural Information Processing Systems, 1998, pp. 647–653, doi:10.1162/089976600300015178.
[61] M.A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE J. 37 (2) (1991) 233–243, doi:10.1002/aic.690370209.
[62] R. Hecht-Nielsen, Replicator neural networks for universal optimal source coding, Science 269 (5232) (1995) 1860–1863, doi:10.1126/science.269.5232.1860.
[63] S. Rifai, X. Muller, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, Vol. 85, 2011, pp. 833–840.
[64] Y. Bengio, Learning deep architectures for AI, Found. Trend. Mach. Learn. 2 (1) (2009) 1–127, doi:10.1561/2200000006.
[65] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.

[66] J. Deng, Z. Zhang, E. Marchi, B. Schuller, Sparse autoencoder-based feature transfer learning for speech emotion recognition, in: 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII), Vol. 00, 2014, pp. 511–516, doi:10.1109/ACII.2013.90.
[67] B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by v1? Vision Res. 37 (23) (1997) 3311–3325, doi:10.1016/S0042-6989(97)00169-7.
[68] F. Feng, X. Wang, R. Li, Cross-modal retrieval with correspondence autoencoder, in: Proceedings of the 22nd ACM International Conference on Multimedia, MM '14, ACM, New York, NY, USA, 2014, pp. 7–16, doi:10.1145/2647868.2654902.
[69] Y.J. Fan, Autoencoder node saliency: selecting relevant latent representations, Pattern Recognition 88 (2019) 643–653, doi:10.1016/j.patcog.2018.12.015.
[70] Y. Yang, Q.M.J. Wu, Y. Wang, Autoencoder with invertible functions for dimension reduction and image reconstruction, IEEE Trans. Syst. Man Cybern. 48 (7) (2018) 1065–1079, doi:10.1109/TSMC.2016.2637279.
[71] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Statist. 22 (3) (1951) 400–407, doi:10.1214/aoms/1177729586.
[72] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude, COURSERA: Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.
[73] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[74] A. Cauchy, Méthode générale pour la résolution des systemes déquations simultanées, Comp. Rend. Sci. Paris 25 (1847) (1847) 536–538.
[75] L. Yann, Modeles connexionnistes de lapprentissage, ph.d. thesis, These de Doctorat, volume 6, Universite Paris, 1987.
[76] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach, Learn. Res. 11 (2010) 3371–3408. Dec.
[77] W. Liu, P.P. Pokharel, J.C. Principe, Correntropy: a localized similarity measure, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, 2006, pp. 4919–4924, doi:10.1109/IJCNN.2006.247192.
[78] 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.
[79] A. Vergara, S. Vembu, T. Ayhan, M.A. Ryan, M.L. Homer, R. Huerta, Chemical gas sensor drift compensation using classifier ensembles, Sensor. Actuat. 166 (2012) 320–329, doi:10.1016/j.snb.2012.01.074.
[80] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318, doi:10.1007/s00500-008-0323-y.
[81] D. Dua, C. Graff, UCI machine learning repository, 2019, http://archive.ics.uci.edu/ml.
[82] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, http://arxiv.org/abs/cs.LG/1708.07747.
[83] J.L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, D. Anguita, Transition-aware human activity recognition using smartphones, Neurocomputing (171) (2016) 754–767, doi:10.1016/j.neucom.2015.07.085.
[84] R. Cole, M. Fanty, Spoken letter recognition, in: Proceedings of the Workshop on Speech and Natural Language, 1990, pp. 385–390, doi:10.3115/116580.116725.
[85] I. Guyon, S. Gunn, A. Ben-Hur, G. Dror, Result analysis of the NIPS 2003 feature selection challenge, in: Proceedings of Neural Information Processing Systems, Vol. 4, 2004, pp. 545–552.
[86] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, volume 86, 1998, pp. 2278–2324, doi:10.1109/5.726791.
[87] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.C. Sanchez, M. Müller, Proc: an open-source package for r and s+ to analyze and compare roc curves, BMC Bioinf. 12 (1) (2011) 77, doi:10.1186/1471-2105-12-77.
[88] D.J. Hand, R.J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Mach. Learn. 45 (2) (2001) 171–186, doi:10.1023/A:1010920819831.
[89] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701, doi:10.1080/01621459.1937.10503522.
[90] J.D. Li, A two-step rejection procedure for testing multiple hypotheses, J. Stat. Plan. Inference 138 (6) (2008) 1521–1527, doi:10.1016/j.jspi.2007.04.032.
[91] S. Garca, A. Fernndez, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064. Special Issue on Intelligent Distributed Information Systems. doi:10.1016/j.ins.2009.12.010.
[92] F. Chollet, et al., Keras, 2015, https://keras.io.
[93] R.C. Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, 2016. https://www.R-project.org/.
[94] K. Schliep, K. Hechenbichler, kknn: weighted k-nearest neighbors, r package version 1.3.1, 2016, https://CRAN.R-project.org/package=kknn.
[95] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, e1071: misc functions of the department of statistics, Probability Theory Group (Formerly: E1071), TU Wien, r Package Version 1.6–7, 2015. https://CRAN.R-project.org/package=e1071.
[96] M.K. C., J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, R.C. Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, caret: classification and regression training, r package version 6.0–68, 2016, https://CRAN.R-project.org/package=caret.
[97] C. Bergmeir, J.M. Benítez, Neural networks in r using the stuttgart neural network simulator: RSNNS, J. Stat. Softw. 46 (7) (2012) 1–26.
[98] K. Hornik, C. Buchta, A. Zeileis, Open-source machine learning: R meets weka, Comput. Stat. 24 (2) (2009) 225–232, doi:10.1007/s00180-008-0119-7.
[99] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful? in: International Conference on Database Theory, 1999, pp. 217–235, doi:10.1007/3-540-49257-7_15.