Journal of Computational Science 78 (2024) 102269

Contents lists available at ScienceDirect

Journal of Computational Science

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

Siamese Autoencoder Architecture for the Imputation of Data Missing Not at Random

Ricardo Cardoso Pereira [a,*], Pedro Henriques Abreu [a], Pedro Pereira Rodrigues [b]

[a] Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, University of Coimbra, Coimbra, 3030-290, Portugal
[b] Center for Health Technology and Services Research, Faculty of Medicine (MEDCIDS), University of Porto, Porto, 4200-319, Portugal

ARTICLE INFO

Keywords:
Missing data
Imputation
Siamese autoencoder
Missing Not at Random

ABSTRACT

Missing data is an issue that can negatively impact any task performed with the available data and it is often found in real-world domains such as healthcare. One of the most common strategies to address this issue is to perform imputation, where the missing values are replaced by estimates. Several approaches based on statistics and machine learning techniques have been proposed for this purpose, including deep learning architectures such as generative adversarial networks and autoencoders. In this work, we propose a novel siamese neural network suitable for missing data imputation, which we call Siamese Autoencoder-based Approach for Imputation (SAEI). Besides having a deep autoencoder architecture, SAEI also has a custom loss function and triplet mining strategy that are tailored for the missing data issue. The proposed SAEI approach is compared to seven state-of-the-art imputation methods in an experimental setup that comprises 14 heterogeneous datasets of the healthcare domain injected with Missing Not At Random values at a rate between 10% and 60%. The results show that SAEI significantly outperforms all the remaining imputation methods for all experimented settings, achieving an average improvement of 35%. This work is an extension of the article *Siamese Autoencoder-Based Approach for Missing Data Imputation* [1] presented at the International Conference on Computational Science 2023. It includes new experiments focused on runtime, generalization capabilities, and the impact of the imputation in classification tasks, where the results show that SAEI is the imputation method that induces the best classification results, improving the F1 scores for 50% of the used datasets.

1. Introduction

Missing data can be described by the absence of values in the instances of a dataset. It is one of the most common data issues, affecting most real-world domains. The existence of missing values has a deep impact on the conclusions that can be drawn from the data [2]. Within machine learning, the majority of the classification and regression models cannot cope with missing data, or suffer a decrease in their performance. In domains such as healthcare, missing values can compromise the results to the point where the study becomes unfeasible [3]. However, different types of missing values may have different impacts. Missing data can be classified according to three different mechanisms which are related to the missingness causes [4,5]:

- Missing Completely At Random (MCAR), which is described by Eq. (1), where $P(x)$ is the probability function, R is the binary matrix of the missing values in Y, Y is a combination of the observed values Y_{obs} and the missing values Y_{mis}, and ψ are the parameters of the missing data model. In this mechanism, the missingness causes are purely random, and therefore only depend on ψ. An example would be if someone simply forgot to fill in a field in a questionnaire.

$$P(R = 0|Y_{obs}, Y_{mis}, \psi) = P(R = 0|\psi) \quad (1)$$

- Missing At Random (MAR), which is described by Eq. (2) and the missingness nature is related to observed data in one or more features, therefore depending on both Y_{obs} and ψ. For example, people may not have results for a specific medical exam because they are too young to be doing such an exam.

$$P(R = 0|Y_{obs}, Y_{mis}, \psi) = P(R = 0|Y_{obs}, \psi) \quad (2)$$

- Missing Not At Random (MNAR), where the missingness causes are unknown since the missing values are related to themselves or to external data not available. As a consequence, the missingness depends on Y_{obs}, Y_{mis} and ψ. For example, people who drink a lot of alcohol may be apprehensive about answering how many drinks they have per day.

* Corresponding author.
E-mail addresses: rdpereira@dei.uc.pt (R.C. Pereira), pha@dei.uc.pt (P.H. Abreu), pprodrigues@med.up.pt (P.P. Rodrigues).

https://doi.org/10.1016/j.jocs.2024.102269
Received 5 November 2023; Received in revised form 29 February 2024; Accepted 14 March 2024
Available online 24 March 2024

A solution often used to address the missing data issue is imputation, which consists of replacing the missing values with estimates [6]. There are several statistical and machine learning-based methods to perform imputation, and each one may be more suitable for a specific missing mechanism [2]. However, most approaches are tailored for MCAR and MAR, while MNAR is still the less addressed mechanism with few solutions, mostly because of its relation with unknown data. The proposal of new approaches that perform well under the MNAR mechanism is an important open challenge when considering that most real-world contexts suffer from this mechanism, including critical domains such as healthcare [7].

Recently, several deep learning architectures have been explored to perform imputation assuming the different mechanisms. The state-of-the-art architectures in this scope are generative adversarial networks (GAN) [8,9], denoising autoencoders (DAE) [10,11], and variational autoencoders (VAE) [7,12]. In this work, we explore the use of siamese networks [13] to perform missing data imputation. We propose a new model called Siamese Autoencoder-based Approach for Imputation (SAEI), which extends and adapts the vanilla siamese network for the imputation task by adding a deep autoencoder architecture, a custom loss function, and a custom triplet mining strategy. To the best of our knowledge, this is the first time siamese networks have been used for this purpose. We compared our SAEI model with a baseline of seven state-of-the-art imputation methods in an experimental setup encompassing 14 datasets of the healthcare domain injected with MNAR values at different missingness levels (10% to 60% missing rates). The results proved that our SAEI model significantly outperformed the remaining baseline methods in all experimented settings, achieving an average improvement of 35%. We also provide a runtime benchmark comparing all the imputation methods, and additional experiments to assess the generalization capabilities of the SAEI method for larger datasets and different data domains. Finally, we also present a study focused on the impact of the imputation methods on classification tasks. The results showed that SAEI is the imputation method that induces the best classification results, followed by the SoftImpute method. SAEI improves the F1 scores for 50% of the used datasets, followed by MICE which only improves the results for less than 30% of the datasets.

The remainder of the paper is organized in the following way: Section 2 presents related work on the missing data field; Section 3 describes with detail the proposed SAEI model; Section 4 presents the experimental setup used to validate the effectiveness of our SAEI model; Section 5 analyzes and discusses the results obtained from the experiments; Section 6 presents the runtime benchmark results; Section 7 presents the study focused on the impact of classification tasks; Section 8 discusses the generalization capabilities of the SAEI method; and Section 9 presents our conclusions and the future work.

2. Related work

As previously stated, the missing data problem cannot be neglected, otherwise it will have a major impact on any tasks performed with the data. An approach often used to deal with this issue is to remove the missing values, which is called case deletion. Two different deletion approaches can be applied [2]: listwise deletion, where all instances containing at least one missing value are eliminated, which is a very simple procedure but can lead to the loss of a considerable amount of information, being for that reason only recommended in big data scenarios or when the instances with missing values are less than 5% of the data; and pairwise deletion, where only the instances containing missing values for the features of interest are deleted, which suffers from the same problems of listwise deletion but reduces the loss of information. To apply any deletion strategies, the missing mechanism must be considered since deleting data may remove relations between instances and features, and, for that reason, this approach should only be applied with missing values under MCAR [2].

When deleting instances is not a suitable option, imputation tends to be the preferred strategy [6]. The key idea is to generate plausible new estimates to replace the missing values. Among the existent methods, the statistical-based ones are frequently used [5]. A common approach is to use the mean or mode of the feature containing missing values for numeric and nominal variables, respectively [2]. Another strategy is to use a regression to model one or more dependent variables (the ones containing missing values) using as independent variables the remaining features of the dataset [2]. The fitting process must only consider the instances that are complete for the independent features, and the type of regression (e.g., linear or non-linear) must be chosen taking in consideration the nature of the data. Both latter approaches are said to be single imputation strategies since only one value is used for the imputation of each missing value. For this reason, the imputation results may be biased since the uncertainty of the generated values is not accounted for. To address this issue multiple imputation strategies can be applied. The key idea is to perform the imputation M times, generating different but complete results each time (different imputation methods may be used). The M complete datasets are then analyzed and combined, being the result of this combination the final dataset [14]. The state-of-the-art method that uses this approach is the Multiple Imputation by Chained Equations (MICE) [15]. It creates a series of regressions where each one is modeled for a different variable with missing values, meaning that each feature is modeled conditionally upon the other features. The process is repeated throughout several iterations until the parameters of the regressions converge to stable values [16].

Other approaches based on statistical and algebra concepts often used are matrix completion methods, such as the SoftImpute. The goal is to perform the imputation by finding a low-rank matrix that is an approximation of the original one. The process usually relies on matrix factorization, where a matrix with (m, n) dimensions is decomposed into two matrices with dimensions (m, k) and (k, n), where k is the rank and must be smaller than m and n. The idea is to find the latent features that better describe the available values. The low-rank approximation matrix can finally be obtained by multiplying the resulting matrices from the decomposition process [17].

The imputation task can also be performed by using machine learning models. In theory, any algorithm that can be trained to predict new values is suitable for imputation [5]. Generating new estimates for numerical features may be seen as a regression problem or a classification problem for nominal features.

An algorithm often used for this purpose is the k-nearest neighbors (KNN) [18,19]. It tries to find the k most similar instances to the one that contains missing values using only the features that are complete. To calculate this similarity a distance function is used, and it must be chosen taking into consideration the data type: the Euclidean distance is suitable for numeric data but not for nominal. For this latter case the data can be transformed through one-hot encoding or a different distance function must be used (e.g., hamming distance works with nominal data) [20]. When $k > 1$ the values of the neighbors' instances must be combined to produce the new value. For numerical data, a common approach used is the simple or weighted mean, and for categorical values, a vote of majority may be applied.

Neural networks are also often used for imputation, particularly autoencoders and generative adversarial networks (GAN). Autoencoders are a type of neural network that learns a representation of the data from the input layer and tries to reproduce it at the output layer. Among its variants, the denoising autoencoder (DAE) is the one more often used for missing data imputation because it is designed to recover noisy data, which can exist due to data corruption via some additive mechanism or by the introduction of missing values [11]. However, the variational autoencoder (VAE) has recently been used for the same purpose. This architecture learns the multidimensional parameters of a Gaussian distribution and generates the estimates to replace the missing values by sampling from that distribution [7]. Regarding GANs,

these are trained through an adversarial process and are based on a generative model that learns the data distribution and a discriminative model that outputs the probability of a sample being from the training data rather than the generative model [21]. GANs are directly applied to the missing data imputation task through the generative adversarial imputation nets (GAIN) [8], where the generator performs the imputation and the discriminator tries to distinguish between the original and the imputed data.

A type of neural network that has not yet been used for missing data imputation is the siamese network. This architecture is based on two or more sub-networks that share the same architecture and parameters. The goal is to output embedding representations in a way that similar concepts would have a similar latent space embedding [13,22]. In this work, we propose an extension to the siamese network tailored for missing data imputation, which is, to the best of our knowledge, the first time that such architecture is being used for this purpose. Since this network uses a triplet loss function, which is based on distances between anchor, positive, and negative samples, it can learn from reduced quantities of data (especially when compared to other neural network-based approaches). This quality makes this type of network feasible to be used with both lower and higher missing rates. Furthermore, the number of complete samples (i.e., without containing missing values) needed to train the model can be rather small when compared to other network-based imputation models. We leverage these characteristics of the siamese networks and propose an adapted method for missing data imputation with a novel deep autoencoder architecture, custom loss function, and custom triplet mining strategy.

3. Siamese autoencoder-based approach for imputation

The Siamese Autoencoder-based Approach for Imputation (SAEI) is an extension of a vanilla siamese network, and it is tailored for missing data imputation by comprising three adaptations:

- A deep autoencoder architecture that allows the network to reproduce the input data at the output layer in an unsupervised fashion;
- A custom loss function that includes both the distance-based triplet loss and the reconstruction error of the autoencoder component of the network;
- A custom triplet mining strategy that was designed specifically for the missing data issue by creating hard triplets based on the existing missing values.

These three adaptations are independently described in the next subsections. Besides incorporating the common capabilities of any autoencoder, such as the ability to learn non-linear complex patterns in an unsupervised way, the SAEI method adds to these advantages the capabilities of a siamese architecture. In particular, this type of network can capture pair-wise relationships between samples, which can help establish the relation between records with and without missing values. Additionally, it also creates the concept of imputation through similarity, which leverages the information from similar instances to aid the generation of estimates.

3.1. Deep autoencoder architecture

The architecture of our SAEI model is roughly inspired by the well-known ZFNet [23], although it presents several changes motivated by it being aimed at tabular data. Fig. 1 and Fig. 2 depict graphical descriptions of the encoder and decoder networks, respectively. The encoder network is composed of two one-dimensional convolutional layers with 16 filters, ReLU as the activation function, and kernel sizes of five and three. Such layers are followed by max-pooling layers with two strides, which are then followed by two regularization layers that perform batch normalization and dropout at a rate of 25%. Moreover, a residual connection is also used to skip the second convolutional layer. Finally, the output from the last pooling layer is flattened and passed through a hyperbolic tangent activation function, and the latent output is obtained from a final dense layer with 128 units and the latter activation function. The decoder network is symmetric to the encoder, presenting the same architecture but in reserve order (without the residual connection). To perform the deconvolution operation, the one-dimensional convolutional and the max-pooling layers are replaced by one-dimensional transposed convolutional layers with two strides. The output dense layer uses the sigmoid activation function so that the data can be normalized within $[0, 1]$.

Convolutional layers operate based on the spatial positioning of the data. In other words, the position where each value is placed is relevant for the feature extraction process. Tabular data does not present this behavior because the features' positions are irrelevant. To surpass this limitation, the SAEI model feeds the input data to a dense layer with 1024 units and without an activation function. The purpose of this layer is to learn an abstract representation of the input data where the spatial relation between the new abstract features is meaningful. Therefore, our SAEI model delegates the task of learning the spatial structure of the features to the network. The first convolutional layer is then fed with the output of the mentioned dense layer.

3.2. Custom loss function

One of the original loss functions proposed to train a siamese network was the triplet loss. It tries to minimize the distance between an anchor and a positive sample that represent the same concept, while maximizing the distance between that same anchor and a negative sample of a different concept. The formulation is presented in Eq. (3), where $f(x)$ is a function of the embedding representation, N is the number of triples composed by an anchor (x_i^a), a positive sample (x_i^p) and a negative one (x_i^n) [22]. Furthermore, α is a margin used to ensure a minimum distance between positive and negative samples.

$$TL_i = \sum_i^N \left[\left\| f(x_i^a) - f(x_i^p) \right\|_2^2 - \left\| f(x_i^a) - f(x_i^n) \right\|_2^2 + \alpha \right] \tag{3}$$

Using the triplet loss function is ideal for a vanilla siamese network since the goal is for the model to distinguish between positive and negative concepts in comparison to the anchor, while outputting embedding representations that incorporate such differences. However, our SAEI model relies on an autoencoder architecture that outputs a reconstruction of the input data based on the anchor latent representation. Therefore, this reconstruction component must also be reflected in the loss function of the model. Eq. (4) presents our custom loss function, where the triplet loss (TL_i) from Eq. (3) is added to the mean squared error between the anchor and positive sample, similarly to what would happen in a denoising autoencoder (i.e., the anchor is the corrupted version of the ground truth represented by the positive sample).

$$TL_i + \sum_i^N (\hat{x_i^a} - x_i^p)^2 \tag{4}$$

3.3. Custom triplet mining

The triplet selection is a key step for successfully training a siamese network. To ensure that the network is able to learn how to distinguish between positive and negative concepts, it is imperative to select triplets that the network is unable to differentiate before being trained. These triplets are composed of the so-called hard positives and hard negatives, which are samples that violate the constraint of the triplet loss, presented in Eq. (5) [22]. If the network is trained with easy triples where the triplet loss constraint is already satisfied before training, it would not gain the capacity to distinguish between positive and negative concepts.

$$\| f(x_i^a) - f(x_i^p) \|_2^2 + \alpha < \| f(x_i^a) - f(x_i^n) \|_2^2 \tag{5}$$

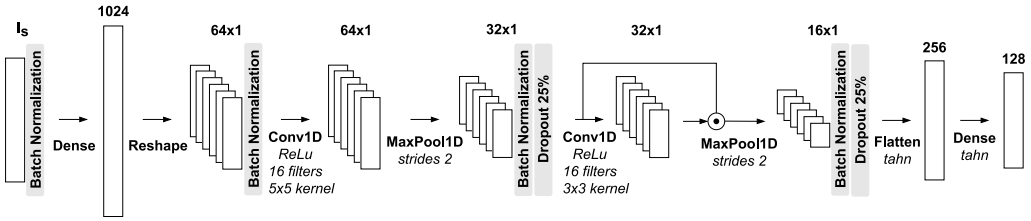

Fig. 1. SAEI encoder architecture. I_s represents the input shape. The encoder network is composed of two one-dimensional convolutional layers with 16 filters followed by max-pooling layers with two strides. Regularization is performed through batch normalization and dropout at a rate of 25%. A residual connection is also used to skip the second convolutional layer. The latent output is obtained from a dense layer with 128 units.

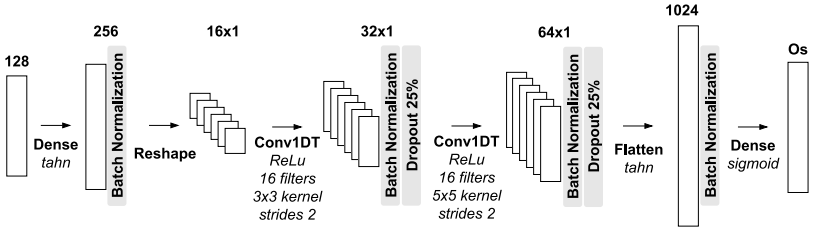

Fig. 2. SAEI decoder architecture. O_s represents the output shape. The decoder network is symmetric to the encoder. Therefore, it has the same layers and regularization but in reserve order. The residual connection is not applied by the decoder.

When extending this type of network and its respective training procedure to address missing data imputation, the definitions of anchor, negative, and positive samples must be redefined within the missing data scope. Our SAEI model proposes the following definitions:

- Anchors are the samples containing missing values, which are pre-imputed with the mean of the available values in each feature (or the mode in categorical features). This pre-imputation step is required since neural networks are unable to be trained with data containing missing values.
- Positives are the original samples without containing missing values (i.e., the ground truth of the anchor samples). This implies that we must have a portion of complete data to train the SAEI model, but such an assumption is common to all deep learning-based imputation methods. Furthermore, the anchor samples are created by artificially generating missing values in the positive samples, according to a pre-established missing data mechanism.
- Negatives are the same as anchor samples but with the missing values being replaced by Gaussian noise sampled according to Eq. (6), where D represents the entire dataset and λ represents the variance of the noise domain.

$$\mathcal{N}\left(\frac{\max(D)-\min(D)}{2}, \lambda\right) \tag{6}$$

The rationale behind the mean value is to use the midpoint within the domain of the data, therefore keeping the Gaussian noise centered on that domain. Such a strategy works especially well when the data is normalized within a specific range (e.g., $[0, 1]$). Moreover, the λ parameter acts as a control variable to define how hard or easy should the triplets be: a low λ value leads to Gaussian noise mostly or completely contained within the data domain, therefore leading to harder triplets since the negative sample is likely to be partially overlapped with the positive one; on the other hand, a high λ value increases the noise domain and creates a negative sample that is more dissimilar to the positive, therefore creating easier triplets where the negative and positive samples are immediately distinguishable by the model. As a consequence, this parameter should be defined aiming to generate hard triplets while considering the data domain.

4. Experimental setup

The quality of the imputation results achieved by the SAEI model was assessed by comparing it with a baseline of seven state-of-the-art imputation methods introduced in Section 2: kNN with $k = 5$, Mean/Mode, MICE with 100 iterations, SoftImpute with 100 iterations, DAE, VAE, and GAIN. The DAE and VAE were defined according to the following architecture and hyperparameters: a hidden layer with half of the input dimension and ReLU as the activation function (although VAE has two additional layers for the Gaussian distribution parameters), Adam as the optimizer with a learning rate of 0.001, Sigmoid as the activation function of the output layer (forcing the data to be normalized within $[0, 1]$), batches of 64 samples, 200 training epochs, a dropout rate of 10% for regularization, Mean Squared Error as the reconstruction loss, and early stop and learning rate reduction by 80% if the validation loss does not improve over 100 epochs. The VAE loss function also includes the Kullback–Leibler divergence for regularization. Furthermore, both DAE and VAE require pre-imputation, which is performed with the mean/mode of the features. The GAIN method was used with the hyperparameters proposed by its authors. Our SAEI model follows the architecture described in Section 3.1 while all the remaining hyperparameters are the same used by the DAE and VAE. Moreover, the Gaussian noise of the negative instances was sampled from $\mathcal{N}(0.5, 0.05^{0.5})$, and the margin of the triplet loss function was set to $\alpha = 0.2$ as the original authors proposed. Except for GAIN, the described hyperparameters of all methods were defined through a grid search process. This is a standard procedure that aims to achieve hyperparameters that conform to common use cases [24]. Moreover, the DAE and VAE were implemented with the Keras library, the GAIN implementation was obtained from its original authors,[1] and the remaining methods were obtained and used from the Scikit-learn library. Our SAEI model was also coded with the Keras library and is available on GitHub.[2]

The experiments were conducted over 14 datasets of the healthcare domain which cover different types of clinical data that was collected for several pathologies. We chose to cover this medical domain since it often suffers from the missing data issue, particularly missing values under the MNAR mechanism, which creates deep challenges to any

[1] https://github.com/jsyoon0823/GAIN
[2] https://github.com/ricardodcpereira/SAEI

Table 1
Datasets characteristics.

Dataset	# Instances	# Features	
		Categorical	Continuous
diabetic-retinopathy	1151	4	16
ecoli	336	1	7
ctg-2c	2126	1	21
new-thyroid-N-vs-HH	215	1	5
kala-azar	68	2	5
immunotherapy	90	3	5
saheart	462	2	8
bc-coimbra	116	1	9
cleveland-0-vs-4	173	1	13
newthyroid-v1	185	1	5
biomed	194	1	5
cryotherapy	90	3	4
thyroid-3-vs-2	703	1	21
pima	768	1	8

subsequent analysis or task performed with the data [3]. In this health context, each instance usually represents a group of values collected for a patient, and each value belongs to a feature that could be a measurement, an exam result, or any other medical input. The missing values may appear at any feature and/or instance. All the 14 datasets are public and available at the UC Irvine Machine Learning[3] and Kaggle[4] repositories, and they present heterogeneous characteristics as seen in Table 1.

The datasets were normalized within $[0, 1]$ and split into train and test sets with 70% and 30% of the instances, respectively. The scaler learns the minimum and maximum values from the train set and transforms both sets. This strategy ensures the test data is not biased with information from the training data. However, in the presence of high missing rates (usually above 50%), the test set may contain values unseen by the scaler, which leads to the normalization boundaries being slightly extended from the expected $[0, 1]$ domain. Also, the autoencoder-based methods use 20% of the train set as the validation set. Furthermore, the categorical nominal features were transformed through the one-hot encoding procedure so that they can be supported by all imputation methods. Finally, in order to be able to calculate the imputation error of the experimented methods, the datasets must be complete and the test set must be injected with artificially generated missing values so that we can compare the estimated values with the original ones (i.e., ground truth). We chose to generate MNAR values since it is the hardest mechanism to address and the one more often found in real-world contexts such as healthcare [3,7]. Our generation strategy is based on removing the smaller values of several features at once (upon a certain missing rate) in a multivariate procedure. Therefore, the missing rate is defined for the entire dataset and the imputation is performed on all features simultaneously. The obtained results were evaluated with the Mean Absolute Error (MAE) between the estimates and the ground truth. Regarding the missing rate, we considered four levels of missingness: 10%, 20%, 40%, and 60%.

To avoid the impact of stochastic behaviors in the results, the experiment was executed 30 independent times, with the train/test split being randomly performed in every run. The MAE results of the experiment are the average of the 30 runs. Moreover, each run was executed in a computer with the following specifications: Ubuntu, CPU AMD Ryzen 5600X, 32 GB RAM, and GPU NVIDIA GeForce RTX 4060 8 GB.

[3] https://archive.ics.uci.edu/ml
[4] https://www.kaggle.com/datasets

5. Imputation results

The imputation results obtained from the experimental setup are displayed with detail in Table 2. The MAE results (average and standard deviation of the 30 independent runs) are individually displayed for each dataset and missing rate. Furthermore, the overall results for each imputation method and per missing rate (i.e., considering all datasets) are displayed in Fig. 3.

In an overall analysis, our SAEI model clearly outperforms all the remaining imputation methods, achieving smaller MAE values for every dataset and missing rate considered, as seen in Table 2. In fact, the average improvement of the SAEI model in comparison to the remaining baseline methods is 35%, peaking at the 20% missing rate with an improvement of 42%. For the lower missing rate (10%) the improvement is 36%. For the higher missing rates (40% and 60%), the improvement rates are 39% and 24%, respectively. Such results show the consistency of our SAEI model with different levels of missingness and with data presenting different characteristics.

In order to validate if the obtained results were statistically significant, we applied the Three-Way ANOVA on ranks test with a significance level of 5%. We considered as factors the dataset, the missing rate, and the imputation method, while the MAE was set as the dependent variable. The normality assumptions were not met, so the data was transformed into rankings using the Ordered Quantile normalization [25]. Such assumptions were also ensured for the data subgroups. The obtained p-values show that the results are statistically significant for all factors, with $p < 0.001$. Additionally, to validate if our SAEI model outperformed the remaining methods with a statistical significance of 5%, the post-hoc Tukey's HSD test was applied to this factor. The obtained p-values show that the SAEI model significantly outperformed all the baseline of imputation methods, with $p < 0.001$ in all scenarios.

6. Runtime results

The imputation results presented in Section 5 assess the quality of the estimated values produced by the new SAEI method and seven other state-of-the-art techniques used as a baseline for comparison. However, the eight methods have very different natures, which can potentially lead to different runtimes (i.e., they can take different amounts of time to train and be applied). This aspect may also impact the decision on what method to use since different domains and applications may have different time restrictions. Therefore, we measured the time needed to train and apply each method within the experimental setup described in Section 4. The average runtime per imputation method is presented in Table 3, with the time displayed in the format *minutes:seconds*. Based on these results, we can conclude that most methods take 12 seconds or less to run, except for autoencoder-based methods, which take more time: the DAE and VAE take between 30 and 40 seconds, while the new SAEI takes approximately two and a half minutes. The difference in the runtime for the SAEI method is justified by the convolutional layers used in the network. Nevertheless, considering that this method is based on convolutional neural networks, the runtime is faster than usual and feasible for most domains.

7. Impact on classification tasks

Motivated by the positive imputation results achieved by the SAEI method, we decided to extend the experiments to evaluate the impact of the imputation in classification tasks. To do so, we designed a new experimental setup that adds a final step to the setup from the main experiments. This step consists of training three different state-of-the-art classifiers using the imputed data and evaluating the classification performance. Many missing data studies have concluded that the imputation methods that deliver the best imputation quality do not necessarily induce the highest improvements in classification

Table 2
Experimental results per dataset and missing rate (average ± standard deviation of MAE). The best results are bolded and highlighted.

Dataset	MR (%)	kNN	Mean/Mode	MICE	SoftImpute	DAE	VAE	GAIN	SAEI
diabetic-retinopathy	10	0.133 ± 0.02	0.194 ± 0.01	0.119 ± 0.02	0.165 ± 0.02	0.152 ± 0.01	0.178 ± 0.01	0.192 ± 0.02	**0.109 ± 0.01**
	20	0.157 ± 0.02	0.213 ± 0.02	0.128 ± 0.02	0.185 ± 0.03	0.164 ± 0.02	0.175 ± 0.01	0.216 ± 0.03	**0.100 ± 0.01**
	40	0.207 ± 0.03	0.252 ± 0.03	0.179 ± 0.03	0.236 ± 0.03	0.199 ± 0.02	0.163 ± 0.01	0.264 ± 0.04	**0.131 ± 0.01**
	60	0.324 ± 0.05	0.358 ± 0.05	0.292 ± 0.05	0.354 ± 0.04	0.298 ± 0.05	0.242 ± 0.04	0.367 ± 0.06	**0.231 ± 0.04**
ecoli	10	0.213 ± 0.03	0.304 ± 0.04	0.229 ± 0.03	0.286 ± 0.04	0.265 ± 0.03	0.328 ± 0.03	0.337 ± 0.06	**0.173 ± 0.03**
	20	0.224 ± 0.04	0.309 ± 0.04	0.236 ± 0.04	0.297 ± 0.04	0.257 ± 0.03	0.306 ± 0.03	0.327 ± 0.06	**0.151 ± 0.03**
	40	0.363 ± 0.08	0.413 ± 0.07	0.363 ± 0.08	0.367 ± 0.05	0.347 ± 0.07	0.352 ± 0.05	0.456 ± 0.08	**0.229 ± 0.06**
	60	0.750 ± 0.21	0.759 ± 0.20	0.730 ± 0.20	0.703 ± 0.18	0.674 ± 0.21	0.672 ± 0.20	0.800 ± 0.22	**0.589 ± 0.20**
ctg-2c	10	0.151 ± 0.02	0.288 ± 0.03	0.137 ± 0.02	0.155 ± 0.02	0.188 ± 0.02	0.255 ± 0.02	0.261 ± 0.02	**0.121 ± 0.02**
	20	0.191 ± 0.02	0.296 ± 0.02	0.161 ± 0.01	0.167 ± 0.01	0.201 ± 0.02	0.227 ± 0.01	0.271 ± 0.02	**0.116 ± 0.01**
	40	0.318 ± 0.03	0.362 ± 0.03	0.286 ± 0.04	0.233 ± 0.02	0.293 ± 0.04	0.205 ± 0.02	0.347 ± 0.03	**0.191 ± 0.03**
	60	0.498 ± 0.05	0.526 ± 0.05	0.488 ± 0.05	0.399 ± 0.05	0.508 ± 0.11	0.356 ± 0.06	0.511 ± 0.06	**0.354 ± 0.06**
new-thyroid-N-vs-HH	10	0.264 ± 0.07	0.270 ± 0.05	0.266 ± 0.08	0.258 ± 0.05	0.272 ± 0.06	0.413 ± 0.04	0.361 ± 0.08	**0.124 ± 0.04**
	20	0.273 ± 0.04	0.264 ± 0.04	0.256 ± 0.05	0.227 ± 0.04	0.238 ± 0.04	0.408 ± 0.04	0.369 ± 0.08	**0.143 ± 0.03**
	40	0.269 ± 0.04	0.296 ± 0.04	0.269 ± 0.04	0.204 ± 0.03	0.247 ± 0.04	0.427 ± 0.04	0.392 ± 0.09	**0.156 ± 0.03**
	60	0.355 ± 0.10	0.432 ± 0.09	0.391 ± 0.09	0.257 ± 0.07	0.374 ± 0.11	0.502 ± 0.07	0.532 ± 0.11	**0.239 ± 0.12**
kala-azar	10	0.317 ± 0.07	0.358 ± 0.07	0.288 ± 0.06	0.300 ± 0.07	0.356 ± 0.06	0.488 ± 0.09	0.398 ± 0.08	**0.217 ± 0.06**
	20	0.409 ± 0.05	0.401 ± 0.05	0.369 ± 0.04	0.357 ± 0.05	0.404 ± 0.07	0.508 ± 0.05	0.455 ± 0.07	**0.232 ± 0.05**
	40	0.468 ± 0.07	0.472 ± 0.07	0.468 ± 0.08	0.408 ± 0.04	0.441 ± 0.09	0.589 ± 0.07	0.540 ± 0.12	**0.277 ± 0.05**
	60	0.951 ± 0.35	0.952 ± 0.35	0.953 ± 0.35	0.805 ± 0.31	0.908 ± 0.35	1.025 ± 0.34	0.983 ± 0.35	**0.796 ± 0.36**
immunotherapy	10	0.346 ± 0.08	0.384 ± 0.08	0.341 ± 0.09	0.362 ± 0.08	0.348 ± 0.08	0.433 ± 0.09	0.388 ± 0.09	**0.285 ± 0.08**
	20	0.370 ± 0.07	0.372 ± 0.07	0.349 ± 0.07	0.384 ± 0.06	0.337 ± 0.06	0.447 ± 0.06	0.386 ± 0.07	**0.268 ± 0.06**
	40	0.463 ± 0.08	0.463 ± 0.08	0.460 ± 0.09	0.469 ± 0.07	0.396 ± 0.08	0.524 ± 0.07	0.498 ± 0.08	**0.345 ± 0.08**
	60	1.026 ± 0.38	1.028 ± 0.38	1.015 ± 0.36	0.958 ± 0.36	0.929 ± 0.36	1.084 ± 0.36	1.027 ± 0.38	**0.871 ± 0.36**
saheart	10	0.289 ± 0.03	0.367 ± 0.03	0.295 ± 0.03	0.245 ± 0.03	0.315 ± 0.03	0.344 ± 0.03	0.387 ± 0.05	**0.231 ± 0.03**
	20	0.335 ± 0.04	0.389 ± 0.04	0.336 ± 0.03	0.268 ± 0.03	0.336 ± 0.04	0.332 ± 0.02	0.410 ± 0.05	**0.216 ± 0.03**
	40	0.464 ± 0.07	0.486 ± 0.06	0.464 ± 0.07	0.366 ± 0.05	0.442 ± 0.07	0.361 ± 0.04	0.531 ± 0.07	**0.296 ± 0.06**
	60	0.876 ± 0.28	0.884 ± 0.28	0.869 ± 0.28	0.732 ± 0.27	0.854 ± 0.29	0.729 ± 0.29	0.934 ± 0.29	**0.677 ± 0.26**
bc-coimbra	10	0.232 ± 0.04	0.300 ± 0.05	0.232 ± 0.04	0.209 ± 0.03	0.328 ± 0.04	0.464 ± 0.04	0.326 ± 0.06	**0.177 ± 0.05**
	20	0.277 ± 0.03	0.317 ± 0.03	0.269 ± 0.03	0.226 ± 0.02	0.319 ± 0.06	0.482 ± 0.02	0.357 ± 0.05	**0.175 ± 0.03**
	40	0.391 ± 0.06	0.411 ± 0.06	0.390 ± 0.07	0.293 ± 0.05	0.363 ± 0.08	0.559 ± 0.05	0.427 ± 0.08	**0.221 ± 0.05**
	60	0.661 ± 0.15	0.691 ± 0.16	0.656 ± 0.14	0.499 ± 0.13	0.675 ± 0.14	0.799 ± 0.14	0.697 ± 0.17	**0.459 ± 0.15**
cleveland-0-vs-4	10	0.340 ± 0.04	0.381 ± 0.04	0.346 ± 0.04	0.273 ± 0.04	0.332 ± 0.04	0.406 ± 0.04	0.347 ± 0.05	**0.255 ± 0.05**
	20	0.358 ± 0.04	0.401 ± 0.04	0.364 ± 0.03	0.280 ± 0.03	0.305 ± 0.03	0.399 ± 0.02	0.383 ± 0.04	**0.224 ± 0.03**
	40	0.526 ± 0.08	0.550 ± 0.08	0.526 ± 0.08	0.412 ± 0.08	0.434 ± 0.11	0.499 ± 0.09	0.524 ± 0.09	**0.303 ± 0.09**
	60	0.917 ± 0.17	0.931 ± 0.16	0.919 ± 0.17	0.761 ± 0.17	0.884 ± 0.18	0.922 ± 0.19	0.920 ± 0.17	**0.733 ± 0.22**
newthyroid-v1	10	0.250 ± 0.05	0.303 ± 0.05	0.249 ± 0.06	0.271 ± 0.06	0.295 ± 0.06	0.408 ± 0.04	0.368 ± 0.09	**0.157 ± 0.04**
	20	0.268 ± 0.05	0.300 ± 0.04	0.257 ± 0.05	0.302 ± 0.05	0.254 ± 0.05	0.409 ± 0.05	0.396 ± 0.11	**0.155 ± 0.03**
	40	0.299 ± 0.05	0.327 ± 0.05	0.311 ± 0.05	0.335 ± 0.05	0.240 ± 0.04	0.434 ± 0.04	0.449 ± 0.13	**0.161 ± 0.03**
	60	0.444 ± 0.15	0.494 ± 0.15	0.468 ± 0.15	0.437 ± 0.10	0.351 ± 0.12	0.554 ± 0.12	0.571 ± 0.19	**0.290 ± 0.12**
biomed	10	0.210 ± 0.05	0.288 ± 0.04	0.208 ± 0.05	0.277 ± 0.07	0.232 ± 0.04	0.409 ± 0.04	0.379 ± 0.10	**0.159 ± 0.03**
	20	0.228 ± 0.04	0.299 ± 0.03	0.211 ± 0.04	0.255 ± 0.04	0.217 ± 0.04	0.418 ± 0.03	0.381 ± 0.10	**0.149 ± 0.02**
	40	0.283 ± 0.04	0.343 ± 0.04	0.261 ± 0.04	0.274 ± 0.04	0.257 ± 0.05	0.448 ± 0.04	0.424 ± 0.08	**0.179 ± 0.03**
	60	0.474 ± 0.11	0.513 ± 0.11	0.448 ± 0.11	0.383 ± 0.10	0.399 ± 0.14	0.573 ± 0.12	0.572 ± 0.14	**0.335 ± 0.14**
cryotherapy	10	0.302 ± 0.08	0.371 ± 0.10	0.317 ± 0.09	0.295 ± 0.09	0.354 ± 0.10	0.453 ± 0.08	0.412 ± 0.11	**0.267 ± 0.09**
	20	0.347 ± 0.09	0.415 ± 0.07	0.339 ± 0.08	0.361 ± 0.07	0.401 ± 0.07	0.465 ± 0.08	0.425 ± 0.10	**0.284 ± 0.06**
	40	0.497 ± 0.10	0.521 ± 0.10	0.477 ± 0.10	0.455 ± 0.09	0.464 ± 0.09	0.573 ± 0.10	0.547 ± 0.11	**0.390 ± 0.10**
	60	1.373 ± 1.05	1.396 ± 1.05	1.371 ± 1.05	1.252 ± 1.02	1.352 ± 1.07	1.419 ± 1.06	1.392 ± 1.06	**1.218 ± 1.03**
thyroid-3-vs-2	10	0.094 ± 0.01	0.099 ± 0.01	0.085 ± 0.01	0.059 ± 0.01	0.068 ± 0.01	0.095 ± 0.01	0.111 ± 0.03	**0.045 ± 0.01**
	20	0.106 ± 0.02	0.109 ± 0.02	0.099 ± 0.02	0.062 ± 0.01	0.072 ± 0.02	0.092 ± 0.01	0.119 ± 0.03	**0.041 ± 0.01**
	40	0.137 ± 0.02	0.135 ± 0.02	0.138 ± 0.02	0.069 ± 0.01	0.093 ± 0.02	0.086 ± 0.01	0.145 ± 0.03	**0.051 ± 0.01**
	60	0.226 ± 0.04	0.224 ± 0.04	0.222 ± 0.04	0.112 ± 0.04	0.170 ± 0.06	0.119 ± 0.04	0.234 ± 0.05	**0.110 ± 0.04**
pima	10	0.266 ± 0.03	0.338 ± 0.03	0.273 ± 0.02	0.239 ± 0.02	0.297 ± 0.03	0.306 ± 0.03	0.384 ± 0.07	**0.207 ± 0.03**
	20	0.275 ± 0.02	0.336 ± 0.03	0.279 ± 0.02	0.234 ± 0.02	0.290 ± 0.03	0.269 ± 0.02	0.368 ± 0.06	**0.168 ± 0.02**
	40	0.360 ± 0.03	0.391 ± 0.04	0.354 ± 0.04	0.281 ± 0.03	0.341 ± 0.04	0.261 ± 0.03	0.452 ± 0.08	**0.199 ± 0.03**
	60	0.569 ± 0.12	0.575 ± 0.12	0.558 ± 0.12	0.435 ± 0.10	0.526 ± 0.13	0.406 ± 0.09	0.628 ± 0.14	**0.392 ± 0.13**

Table 3
Runtime results per imputation method (average ± standard deviation time). The time is displayed in the format *minutes:seconds*.

Method	Runtime
kNN	< 00:01 ± 00:00
Mean/Mode	< 00:01 ± 00:00
MICE	00:12 ± 00:30
SoftImpute	00:01 ± 00:01
DAE	00:32 ± 00:18
VAE	00:36 ± 00:18
GAIN	00:12 ± 00:00
SAEI	02:27 ± 01:13

and regression tasks [26]. Therefore, it is relevant to understand if our SAEI model can also have a positive impact on classification tasks while achieving top performance in imputation quality.

The chosen classification models were the k-Nearest Neighbors (kNN), the Random Forest (RF), and the eXtreme Gradient Boosting (XGBoost) since these are the state-of-the-art classifiers more frequently used for tabular data and the ones that provide the best results [27]. The implementations of the kNN and RF models were obtained from the Scikit-learn library, and for XGBoost the official implementation was used. Regarding the models' configuration, the default parameters were considered: for the kNN, a $k = 5$ was applied with the metric being the Euclidean distance; for the RF and XGBoost, 100 trees were considered without restrictions for their depth. The used datasets are the same as in the main experimental setup, and all of them contain a binary classification target. Each dataset was randomly split into train and test sets with 70% and 30% of the instances, respectively.

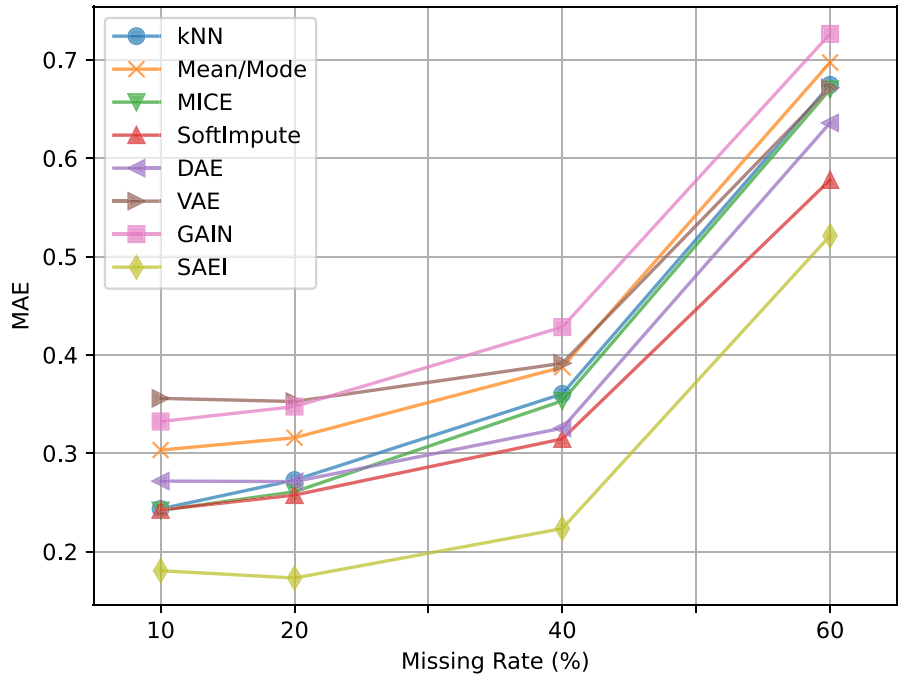

Fig. 3. Overall Mean Absolute Error of all imputation methods per missing rate. Our SAEI model significantly outperforms the remaining methods in all settings.

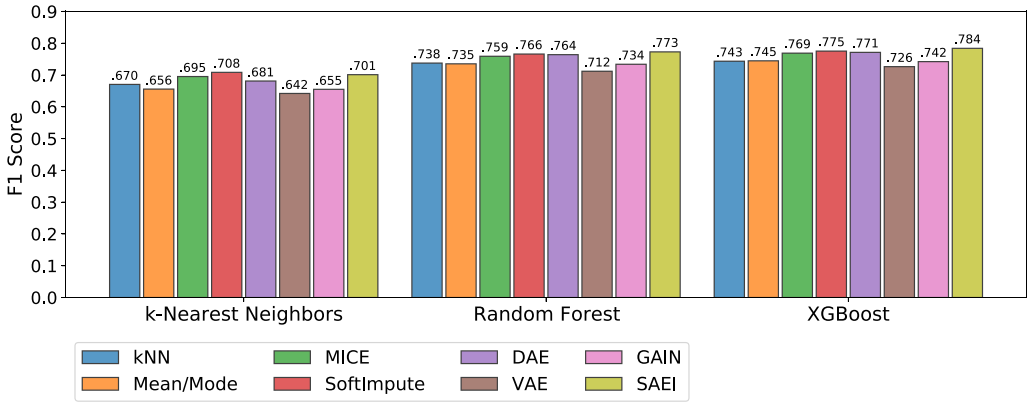

Fig. 4. Overall F1 score of all imputation methods per classification model. Our SAEI imputation method induces the best classification results for the Random Forest and XGBoost, and for the k-Nearest Neighbors it is only surpassed by the SoftImpute method.

The same pre-processing and setup details from the main experiments were applied here, with a single difference: the artificially generated missing values were injected in both the train and test sets, and each one was independently imputed using the same seven state-of-the-art imputation methods plus SAEI. Considering that most of the datasets present a moderate imbalance ratio (the average for all datasets is 40%), the SMOTE [28] over-sampling method was applied to the training set in order to improve the classification results for all models. Finally, the classification performance was evaluated through the F1 score since it accounts for both precision and recall and works better with imbalanced classes.

This new experiment was executed 30 independent times to mitigate the bias caused by stochastic behaviors, and the final results are the average of these 30 runs. The F1 scores per dataset and classifier are available in Table 4, and the overall results per classification model are presented in Fig. 4.

In an overall analysis, the results show that the SAEI is the imputation method that induces the best classification results with an average F1 score of 0.753, followed by SoftImpute (0.749) and MICE (0.741). When considering the results per classification model, SAEI induces the best results for XGBoost (0.784) followed by SoftImpute (0.775), and also for the Random Forest (0.773) followed again by SoftImpute (0.766). For the k-Nearest Neighbors classifier, the SAEI score (0.701) is only surpassed by the SoftImpute (0.708).

When performing an independent analysis per dataset, we can conclude that SAEI induces the best results for seven datasets (50%), which are *cleveland-0-vs-4*, *cryotherapy*, *ctg-2c*, *ecoli*, *immunotherapy*, *newthyroid-v1*, and *thyroid-3-vs-2*; followed by MICE with four datasets (≈30%), which are *biomed*, *kala-azar*, *new-thyroid-N-vs-HH*, and *saheart*; and SoftImpute with three datasets (≈20%), which are *bc-coimbra*, *diabetic-retinopathy*, and *pima*. These results show again that SAEI is the method with the best generalization for data with different characteristics.

In conclusion, besides showing the best imputation quality, the SAEI method leads to the overall best classification results when considering state-of-the-art classifiers for tabular data, such as the XGBoost. Moreover, it provides the best generalization capabilities by improving the results for the highest number of heterogeneous datasets.

Table 4
Experimental results per dataset and classifier (average ± standard deviation F1 score). The best results are bolded and highlighted. The classifiers legend reads as follows: k-Nearest Neighbors (kNN), Random Forest (RF), and XGBoost (XGB).

Dataset	Clf.	kNN	Mean/Mode	MICE	SoftImpute	DAE	VAE	GAIN	SAEI
diabetic-retinopathy	kNN	0.625 ± 0.04	0.586 ± 0.03	0.615 ± 0.04	0.631 ± 0.06	0.611 ± 0.04	0.589 ± 0.03	**0.632 ± 0.06**	0.616 ± 0.04
	RF	0.695 ± 0.05	0.658 ± 0.03	0.715 ± 0.05	**0.756 ± 0.06**	0.726 ± 0.06	0.644 ± 0.03	0.723 ± 0.06	0.726 ± 0.06
	XGB	0.703 ± 0.06	0.655 ± 0.03	0.732 ± 0.06	**0.763 ± 0.06**	0.731 ± 0.06	0.649 ± 0.03	0.728 ± 0.06	0.734 ± 0.05
ecoli	kNN	0.915 ± 0.03	0.922 ± 0.02	0.922 ± 0.03	0.925 ± 0.02	0.927 ± 0.02	0.918 ± 0.02	0.913 ± 0.03	**0.930 ± 0.02**
	RF	0.944 ± 0.02	0.949 ± 0.02	0.945 ± 0.02	0.944 ± 0.02	**0.951 ± 0.02**	0.945 ± 0.02	0.947 ± 0.02	0.950 ± 0.02
	XGB	0.945 ± 0.02	0.952 ± 0.02	0.947 ± 0.02	0.947 ± 0.02	**0.953 ± 0.02**	0.943 ± 0.02	0.946 ± 0.02	**0.953 ± 0.02**
ctg-2c	kNN	0.725 ± 0.08	0.734 ± 0.05	0.774 ± 0.05	0.779 ± 0.04	0.759 ± 0.04	0.739 ± 0.04	0.754 ± 0.05	**0.786 ± 0.04**
	RF	0.834 ± 0.08	0.876 ± 0.03	0.881 ± 0.04	0.887 ± 0.05	0.888 ± 0.04	0.864 ± 0.04	0.858 ± 0.05	**0.892 ± 0.04**
	XGB	0.831 ± 0.08	0.874 ± 0.03	0.880 ± 0.04	0.885 ± 0.04	0.882 ± 0.03	0.872 ± 0.03	0.863 ± 0.04	**0.892 ± 0.04**
new-thyroid-N-vs-HH	kNN	0.859 ± 0.08	0.842 ± 0.07	**0.900 ± 0.07**	**0.900 ± 0.06**	0.879 ± 0.07	0.786 ± 0.11	0.827 ± 0.08	0.870 ± 0.05
	RF	0.876 ± 0.08	0.883 ± 0.06	0.906 ± 0.06	0.901 ± 0.06	**0.916 ± 0.06**	0.822 ± 0.10	0.866 ± 0.07	0.896 ± 0.06
	XGB	0.876 ± 0.07	0.866 ± 0.06	**0.908 ± 0.06**	0.902 ± 0.06	0.904 ± 0.06	0.815 ± 0.11	0.872 ± 0.07	0.876 ± 0.06
kala-azar	kNN	**0.295 ± 0.17**	0.236 ± 0.16	0.264 ± 0.19	0.277 ± 0.17	0.251 ± 0.17	0.223 ± 0.15	0.230 ± 0.16	0.271 ± 0.18
	RF	0.319 ± 0.27	0.236 ± 0.25	**0.332 ± 0.28**	0.268 ± 0.27	0.312 ± 0.29	0.252 ± 0.24	0.278 ± 0.27	0.287 ± 0.28
	XGB	0.346 ± 0.25	0.255 ± 0.25	0.378 ± 0.29	0.306 ± 0.26	0.355 ± 0.29	0.309 ± 0.24	0.311 ± 0.27	**0.406 ± 0.29**
immunotherapy	kNN	0.780 ± 0.08	0.781 ± 0.07	0.789 ± 0.08	0.795 ± 0.07	0.788 ± 0.07	0.762 ± 0.08	0.769 ± 0.08	**0.802 ± 0.07**
	RF	0.859 ± 0.06	0.858 ± 0.06	0.869 ± 0.06	0.861 ± 0.06	0.869 ± 0.05	0.860 ± 0.05	0.868 ± 0.05	**0.879 ± 0.06**
	XGB	0.852 ± 0.06	0.858 ± 0.05	0.860 ± 0.06	**0.864 ± 0.06**	0.862 ± 0.05	0.857 ± 0.06	0.851 ± 0.05	0.862 ± 0.05
saheart	kNN	0.533 ± 0.07	0.501 ± 0.06	0.557 ± 0.07	**0.558 ± 0.07**	0.522 ± 0.06	0.492 ± 0.06	0.512 ± 0.07	0.526 ± 0.06
	RF	0.523 ± 0.08	0.504 ± 0.06	0.566 ± 0.09	0.571 ± 0.10	**0.579 ± 0.08**	0.512 ± 0.06	0.530 ± 0.07	0.546 ± 0.07
	XGB	0.536 ± 0.08	0.533 ± 0.06	0.591 ± 0.08	0.577 ± 0.09	**0.594 ± 0.08**	0.521 ± 0.06	0.544 ± 0.08	0.568 ± 0.07
bc-coimbra	kNN	0.685 ± 0.11	0.630 ± 0.11	**0.700 ± 0.10**	0.688 ± 0.10	0.650 ± 0.10	0.640 ± 0.10	0.653 ± 0.10	0.646 ± 0.09
	RF	0.716 ± 0.11	0.675 ± 0.09	0.742 ± 0.09	**0.757 ± 0.07**	0.724 ± 0.09	0.631 ± 0.09	0.705 ± 0.09	0.733 ± 0.08
	XGB	0.729 ± 0.10	0.671 ± 0.08	0.759 ± 0.09	**0.762 ± 0.08**	0.724 ± 0.09	0.659 ± 0.09	0.697 ± 0.10	0.750 ± 0.07
cleveland-0-vs-4	kNN	0.498 ± 0.18	0.496 ± 0.20	0.531 ± 0.16	0.594 ± 0.17	0.528 ± 0.19	0.492 ± 0.17	0.516 ± 0.18	**0.597 ± 0.17**
	RF	0.558 ± 0.26	0.531 ± 0.26	0.490 ± 0.27	0.543 ± 0.25	0.528 ± 0.26	0.493 ± 0.26	0.527 ± 0.26	**0.632 ± 0.26**
	XGB	0.619 ± 0.22	0.608 ± 0.22	0.560 ± 0.24	0.636 ± 0.22	0.595 ± 0.22	0.579 ± 0.22	0.615 ± 0.22	**0.695 ± 0.22**
newthyroid-v1	kNN	0.949 ± 0.05	0.955 ± 0.03	0.972 ± 0.03	0.978 ± 0.02	0.958 ± 0.03	0.911 ± 0.05	0.933 ± 0.04	**0.979 ± 0.01**
	RF	0.961 ± 0.03	0.978 ± 0.02	0.976 ± 0.02	0.980 ± 0.02	0.976 ± 0.02	0.947 ± 0.03	0.962 ± 0.02	**0.984 ± 0.01**
	XGB	0.957 ± 0.03	0.972 ± 0.02	0.971 ± 0.02	0.979 ± 0.02	0.969 ± 0.02	0.939 ± 0.03	0.956 ± 0.03	**0.981 ± 0.01**
biomed	kNN	0.855 ± 0.06	0.848 ± 0.04	**0.889 ± 0.04**	0.872 ± 0.04	0.868 ± 0.05	0.811 ± 0.06	0.835 ± 0.05	0.875 ± 0.03
	RF	0.883 ± 0.05	0.895 ± 0.04	**0.909 ± 0.03**	0.883 ± 0.04	0.906 ± 0.04	0.866 ± 0.04	0.880 ± 0.04	0.904 ± 0.03
	XGB	0.870 ± 0.05	0.891 ± 0.04	0.899 ± 0.04	0.876 ± 0.04	**0.908 ± 0.04**	0.851 ± 0.04	0.877 ± 0.04	0.889 ± 0.04
cryotherapy	kNN	0.803 ± 0.10	0.796 ± 0.09	0.817 ± 0.08	0.810 ± 0.09	0.810 ± 0.08	0.770 ± 0.10	0.777 ± 0.09	**0.821 ± 0.09**
	RF	0.834 ± 0.09	0.857 ± 0.07	**0.865 ± 0.09**	**0.865 ± 0.09**	0.838 ± 0.08	0.799 ± 0.11	0.847 ± 0.08	0.864 ± 0.07
	XGB	0.817 ± 0.09	0.830 ± 0.08	0.827 ± 0.09	0.840 ± 0.08	0.822 ± 0.08	0.787 ± 0.10	0.812 ± 0.08	**0.842 ± 0.08**
thyroid-3-vs-2	kNN	0.265 ± 0.12	0.275 ± 0.13	0.359 ± 0.12	0.464 ± 0.15	0.377 ± 0.15	0.274 ± 0.12	0.232 ± 0.10	**0.475 ± 0.15**
	RF	0.716 ± 0.24	0.781 ± 0.14	0.766 ± 0.18	0.842 ± 0.09	0.798 ± 0.17	0.732 ± 0.21	0.632 ± 0.24	**0.851 ± 0.09**
	XGB	0.715 ± 0.20	0.835 ± 0.08	0.783 ± 0.15	0.844 ± 0.08	0.814 ± 0.14	0.773 ± 0.16	0.670 ± 0.19	**0.849 ± 0.08**
pima	kNN	0.599 ± 0.06	0.578 ± 0.05	0.643 ± 0.06	**0.646 ± 0.06**	0.607 ± 0.05	0.580 ± 0.05	0.587 ± 0.06	0.621 ± 0.04
	RF	0.609 ± 0.07	0.613 ± 0.05	0.665 ± 0.06	0.666 ± 0.06	**0.686 ± 0.07**	0.599 ± 0.05	0.651 ± 0.06	0.674 ± 0.06
	XGB	0.611 ± 0.06	0.626 ± 0.04	0.667 ± 0.06	0.672 ± 0.06	**0.682 ± 0.07**	0.616 ± 0.04	0.648 ± 0.07	0.679 ± 0.06

8. Generalization capabilities

As discussed in Section 5, the proposed SAEI method achieved significantly better results than the baseline used for comparison. However, since the experiments were entirely focused on healthcare datasets of limited size, we decided to extend these experiments to account for larger datasets and different data domains. The goal is to understand how well SAEI generalizes with these data characteristics. We decided to rerun the experimental setup from Section 4 with two larger datasets, one from the medical domain (Key Indicators of Heart Disease[5]) and another one from the banking industry (Bank Churn Data for Classification[6]). The characteristics of these datasets are described in Table 5, while their respective results are presented in Figs. 5 and 6. Based on Fig. 5, we can see that SAEI stays the best imputation method for a larger dataset within the healthcare domain, since it outperforms the baseline methods in all missing rates. However, based on Fig. 6, we see that in a different domain, the SAEI method is still the best one for smaller missing rates (10% and 20%), but it is surpassed by some of its competitors in higher missing rates (40% and 60%). Overall, SAEI generalizes well for larger datasets, but it loses some capabilities when dealing with higher missing rates in data domains that are not healthcare-related.

[5] https://bit.ly/4bU3opo
[6] https://bit.ly/3uIIcSB

9. Conclusions

In this work, we propose a new model for missing data imputation called Siamese Autoencoder-based Approach for Imputation (SAEI), which is an extension of the vanilla siamese networks adapted for the imputation task. The model incorporates three main adaptations: a deep autoencoder architecture that is tailored for missing data reconstruction, a custom loss function that encompasses both the triplet and reconstruction losses, and a triplet mining strategy tailored for the missing data issue that is capable of generating hard triplets that are meaningful for the training procedure. To the best of our knowledge, this is the first time a siamese architecture is being used and adapted for missing data imputation. We compared our SAEI model with seven state-of-the-art imputation methods from both statistical and machine learning backgrounds, in an experimental setup that used 14 datasets of the healthcare domain injected with MNAR values in a multivariate fashion at 10%, 20%, 40%, and 60% missing rates. Our SAEI model significantly outperformed all the baseline methods used for comparison in all the datasets and missing rates, with a statistical significance of 5%, achieving an average improvement of 35%. We also presented a runtime benchmark for all imputation methods and an experimental study to assess the generalization capabilities of the SAEI method, which showed that it generalizes well for larger datasets, although it loses some qualities when dealing with higher missing rates in data domains not related to healthcare. Furthermore, a study to

Table 5
Larger datasets characteristics.

Dataset	# Instances	# Features	
		Categorical	Continuous
Key Indicators of Heart Disease	319 795	14	4
Bank Churn Data for Classification	175 028	9	16

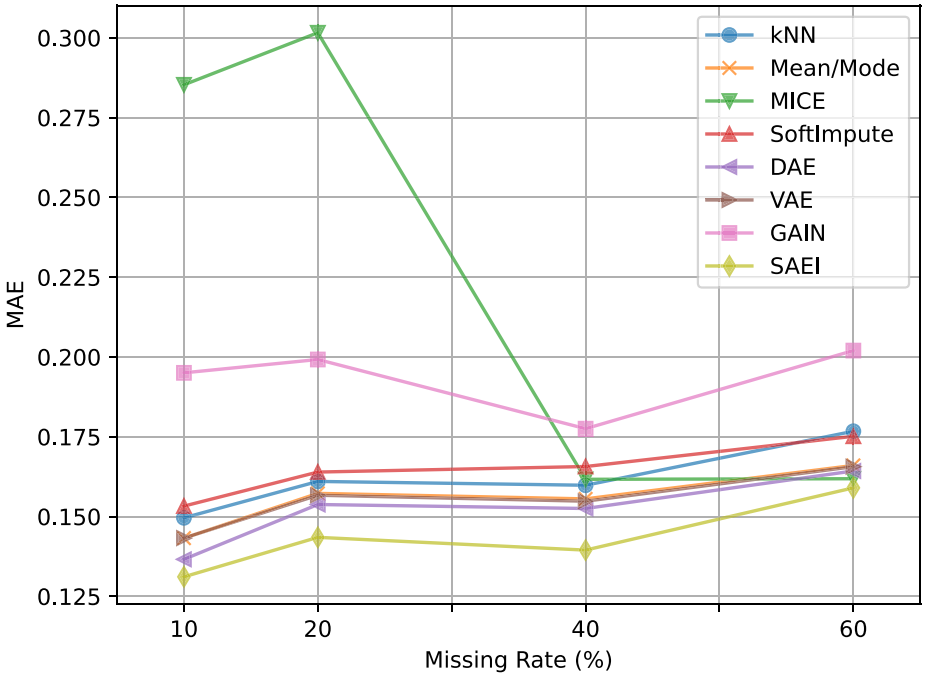

Fig. 5. Mean Absolute Error, per missing rate, for the Key Indicators of Heart Disease dataset. Our SAEI model outperforms the remaining methods in all settings.

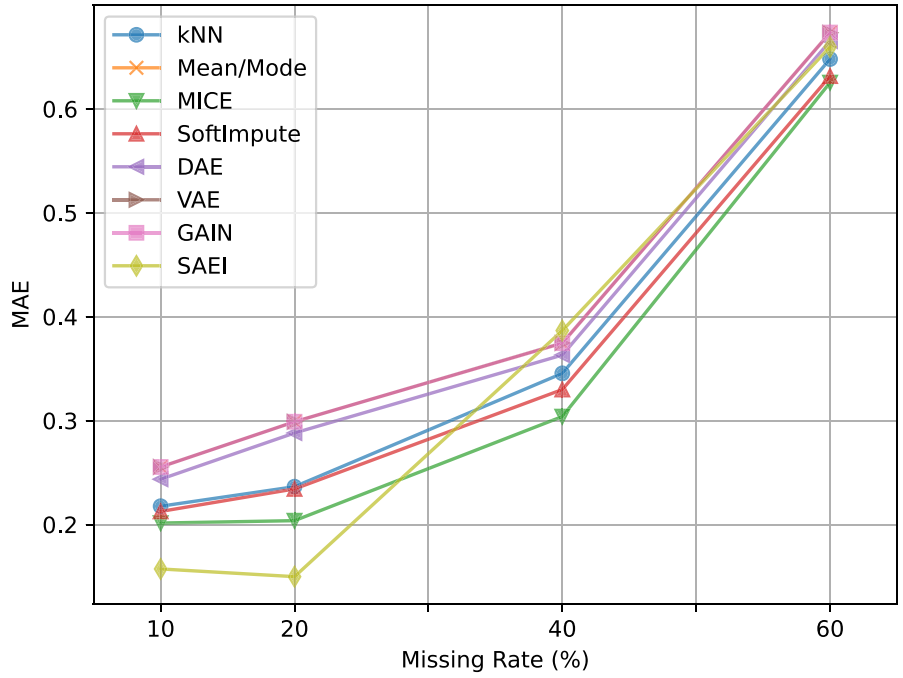

Fig. 6. Mean Absolute Error, per missing rate, for the Bank Churn Data for Classification dataset. Our SAEI model outperforms the remaining methods for the 10% and 20% missing rates, but it achieves worse results for the 40% and 60% rates.

assess the impact of the imputation in classification tasks was also conducted. SAEI was the imputation method that induced the best overall classification results while improving the F1 scores for 50% of the datasets.

In the future, we want to extend this work to test other missing data mechanisms, and we want to explore automatic approaches to evolve our deep autoencoder architecture so it can be even further optimized.

CRediT authorship contribution statement

Ricardo Cardoso Pereira: Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **Pedro Henriques Abreu:** Investigation, Supervision, Validation, Writing – review & editing. **Pedro Pereira Rodrigues:** Investigation, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

This work is supported in part by the FCT - Foundation for Science and Technology, I.P., Research Grant SFRH/BD/149018/2019. This work is also funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020, and by the Portuguese Recovery and Resilience Plan (PRR) through project C645008882-00000055 (Center for Responsible AI).

References

[1] R.C. Pereira, P.H. Abreu, P.P. Rodrigues, Siamese autoencoder-based approach for missing data imputation, in: J. Mikyška, C. de Mulatier, M. Paszynski, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M. Sloot (Eds.), Computational Science – ICCS 2023, Springer Nature Switzerland, Cham, 2023, pp. 33–46.
[2] R.J. Little, D.B. Rubin, Statistical Analysis with Missing Data, vol. 793, John Wiley & Sons, 2019.
[3] N. Peek, P.P. Rodrigues, Three controversies in health data science, Int. J. Data Sci. Anal. 6 (3) (2018) 261–269.
[4] D.B. Rubin, Inference and missing data, Biometrika 63 (3) (1976) 581–592.
[5] P.J. García-Laencina, J.-L. Sancho-Gómez, A.R. Figueiras-Vidal, Pattern classification with missing data: a review, Neural Comput. Appl. 19 (2) (2010) 263–282.
[6] S. Van Buuren, Flexible Imputation of Missing Data, Chapman and Hall/CRC, 2018.
[7] R.C. Pereira, P.H. Abreu, P.P. Rodrigues, Partial multiple imputation with variational autoencoders: Tackling not at randomness in healthcare data, IEEE J. Biomed. Health Inf. 26 (8) (2022) 4218–4227.
[8] J. Yoon, J. Jordon, M. Schaar, Gain: Missing data imputation using generative adversarial nets, in: International Conference on Machine Learning, PMLR, 2018, pp. 5689–5698.
[9] D.T. Neves, J. Alves, M.G. Naik, A.J. Proença, F. Prasser, From missing data imputation to data generation, J. Comput. Sci. 61 (2022) 101640.
[10] 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, 2008, pp. 1096–1103.
[11] D. Charte, F. Charte, S. García, 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.
[12] J.T. McCoy, S. Kroon, L. Auret, Variational autoencoders for missing data imputation with application to a simulated milling circuit, IFAC-PapersOnLine 51 (21) (2018) 141–146.
[13] D. Chicco, Siamese neural networks: An overview, Artif. Neural Netw. (2021) 73–94.
[14] D.B. Rubin, Multiple Imputation for Nonresponse in Surveys, vol. 81, John Wiley & Sons, 2004.
[15] S.v. Buuren, K. Groothuis-Oudshoorn, Mice: Multivariate imputation by chained equations in R, J. Stat. Softw. (2010) 1–68.
[16] M.J. Azur, E.A. Stuart, C. Frangakis, P.J. Leaf, Multiple imputation by chained equations: what is it and how does it work? Int. J. Methods Psychiatr. Res. 20 (1) (2011) 40–49.
[17] M. Udell, C. Horn, R. Zadeh, S. Boyd, et al., Generalized low rank models, Found. Trends Mach. Learn. 9 (1) (2016) 1–118.
[18] G. Batista, M. Monard, Experimental Comparison of K-Nearest Neighbor and Mean or Mode Imputation Methods with the Internal Strategies Used by C4. 5 and CN2 to Treat Missing Data, 34, University of Sao Paulo, 2003.
[19] P.J. García-Laencina, P.H. Abreu, M.H. Abreu, N. Afonoso, Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values, Comput. Biol. Med. 59 (2015) 125–133.
[20] G.E. Batista, M.C. Monard, et al., A study of K-nearest neighbour as an imputation method, HIS 87 (251–260) (2002) 48.
[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[22] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.
[23] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, Springer, 2014, pp. 818–833.
[24] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012) 281–305.
[25] R.A. Peterson, J.E. Cavanaugh, Ordered quantile normalization: a semiparametric transformation built for the cross-validation era, J. Appl. Stat. (2019) 1–16.
[26] M.S. Santos, P.H. Abreu, A. Fernández, J. Luengo, J. Santos, The impact of heterogeneous distance functions on missing data imputation and classification performance, Eng. Appl. Artif. Intell. 111 (2022) 104791.
[27] A. Burkov, The Hundred-Page Machine Learning Book, vol. 1, Andriy Burkov Quebec City, QC, Canada, 2019.
[28] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.