Machine Learning
https://doi.org/10.1007/s10994-024-06584-1

Autoreplicative random forests with applications to missing value imputation

Ekaterina Antonenko[1,2,3,4] · Ander Carreño[5] · Jesse Read[1]

Received: 6 December 2023 / Revised: 4 June 2024 / Accepted: 12 June 2024

Abstract
Missing values are a common problem in data science and machine learning. Removing instances with missing values is a straightforward workaround, but this can significantly hinder subsequent data analysis, particularly when features outnumber instances. There are a variety of methodologies proposed in the literature for imputing missing values. Denoising Autoencoders, for example, have been leveraged efficiently for imputation. However, neural network approaches have been relatively less effective on smaller datasets. In this work, we propose Autoreplicative Random Forests (ARF) as a multi-output learning approach, which we introduce in the context of a framework that may impute via either an iterative or procedural process. Experiments on several low- and high-dimensional datasets show that ARF is computationally efficient and exhibits better imputation performance than its competitors, including neural network approaches. In order to provide statistical analysis and mathematical background to the proposed missing value imputation framework, we also propose probabilistic ARFs, where the confidence values are provided over different imputation hypotheses, therefore maximizing the utility of such a framework in a machine-learning pipeline targeting predictive performance.

Keywords Multi-label classification · Multi-output modeling · Missing value imputation · Probabilistic inference

Editor: Myra Spiliopoulou.

✉ Ekaterina Antonenko
ekaterina.antonenko@minesparis.psl.eu

[1] LIX, Ecole Polytechnique, IP Paris, 91120 Palaiseau, France

[2] Mines Paris, CBIO-Centre for Computational Biology, PSL Research University, 75006 Paris, France

[3] Institut Curie, PSL Research University, 75005 Paris, France

[4] INSERM, U900, 75005 Paris, France

[5] Quant AI Lab, 28043 Madrid, Spain

1 Introduction

Missing values are a common problem and an important issue in the domain of data science and machine learning. Most off-the-shelf statistical and machine learning methods cannot learn from data containing missing values, and so prior to analysis or learning, either all instances with missing values must be removed or missing value imputation (MVI) must be performed. When many values are missing, the first approach of considering only complete instances (no missing values) can lead to a significant loss of information or even an empty dataset, and thus MVI becomes important.

Indeed, missing values can occur in many or most training samples, especially when there are sufficiently more features (p) than samples (N), i.e. when $p \gg N$. Examples of this scenario include medical and bioinformatics arrays, classification problems in astronomy, tool development for finance data, and weather prediction (Johnstone and Titterington 2009).

Missing value patterns are traditionally classified into three types (Santos et al. 2019). Where values are Missing Completely at Random (MCAR), the presence or absence of missing values does not depend on observed or unobserved data. In the Missing at Random (MAR) case, the missingness is dependent on observed variables but not on the missing values themselves; and values Missing Not at Random (MNAR) are missing with probability dependent on the values that are missing. Like the methods we will refer to, we will assume that data follows the MCAR pattern.

In this paper we propose a novel framework specifically distinguishing between two types of approaches, which we call the 1) *procedural* approach, where each value is imputed only once, versus 2) *iterative* approach in which values are successively re-imputed until convergence. We first propose a unifying framework for MVI within which to set these two strategies.

Within this framework, we propose a novel MVI method, Autoreplicative Random Forests, that does not require such a vast number of instances to obtain accurate results and also leverages from the statistical dependence information of the surrounding features to predict the target missing values. The proposed approach can be carried out in either a procedural or iterative fashion. To the best of our knowledge, using multi-label models in such an autoreplicative fashion without explicit encoding has not been studied in the literature.

We empirically demonstrate the advantages of this method in terms of marginal accuracy, joint accuracy, and likelihood. Furthermore, we show the computational efficiency of this method when dealing with a small number of instances.

We also notice that many MVI methods do not explicitly consider the underlying distribution, in particular the joint distribution. That is to say, there is little work that explicitly considers *joint imputation*. Rather, existing approaches simply do MVI in the view that each imputation will be treated independently (of other imputations) and identically (to existing/non-imputed values). Furthermore, they do not explicitly model the associated uncertainty of such imputation. To approach this task, we further propose in our general framework a probabilistic imputation method, *distributional iterative* Autoreplicative Random Forest (ditARF), that takes uncertainty into account during MVI iterations and provides a corresponding estimate of uncertainty (or, inversely speaking, confidence) associated with final imputations, both value-wise (marginal) and instance-wise (jointly).

We consider Autoreplicative Random Forests for MVI as a multi-label predictive method, which allows us not only to exploit target interdependencies but also to sufficiently alleviate the time complexity when compared to leave-one-out schemes such as MICE. As we show in the empirical evaluation part, itARF consistently outperforms its iterative competitors in terms of computation time though maintains high imputation quality.

In this work, we focus on categorical features as a multi-label multi-output classification problem. The proposed framework does not fundamentally show any obstruction to work with continuous features and we believe that it would be easily adapted to work with such data. In the meantime, the categorical approach can be applied to continuous data via feature discretization. Data discretization is known to be an effective approach to regression in some contexts (Dougherty et al. 1995), particularly where interpretation is required.

To sum up, we contribute to the state-of-the-art with the following:

- We propose a general MVI framework, incorporating both procedural and iterative imputation strategies;
- In this framework, we identify weaknesses of existing methods and propose Autoreplicative Random Forests (ARF), represented by a variant in both procedural (pARF) and iterative (itARF) strategies;
- We propose distributional iterative ARF (ditARF), a probabilistic variant that provides confidence over imputation hypotheses, both under the assumptions of marginal (individual) and joint (combinatorial) imputations;
- We demonstrate the effectiveness of our proposed methods when compared to a range of competing methods on both standard-dimensional and high-dimensional ($p \gg N$) data.

The rest of the paper is organized as follows. Together with summarizing the background and related work, we present an imputing framework unifying different methods in Sect. 2. We expand this framework with a group of new methods, pARF, itARF, and ditARF, in Sect. 3. The results and their discussion as well as complexity analysis are described in Sect. 4. In Sect. 5, we draw conclusions and describe future work.

2 A general framework for missing value imputation

In this section, we describe a general framework unifying different approaches for MVI.

First, in Sect. 2.1, we formalize the problem and set out our notation. In Sect. 2.2 we classify the existing strategies as procedural and iterative; thus laying all the foundation in which to consider related work, which we do in detail in Sect. 2.3; and then we propose our novel methodology – in Sect. 3.

2.1 Features, missing and imputed values

We define a dataset $\mathcal{D} = \{X \cup \tilde{X}\}$, consisting of N rows (instances) and p columns (features), with observed and missing values denoted as X and $\tilde{X}$, respectively. For example ($N = 5$, $p = 3$),

$$\mathcal{D} = \begin{bmatrix} \tilde{x}_{1,1} & \tilde{x}_{1,2} & x_{1,3} \\ x_{2,1} & \tilde{x}_{2,2} & x_{2,3} \\ \tilde{x}_{3,1} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \\ x_{5,1} & x_{5,2} & x_{5,3} \end{bmatrix} \quad \text{where} \tilde{X} = \{\tilde{x}_{1,1}, \tilde{x}_{1,2}, \tilde{x}_{2,2}, \tilde{x}_{3,1}\},$$

where $x_{i,j}$ stands for an existing value in the dataset, and $\tilde{x}_{i,j}$ implies that such a value is not yet known/realized (i.e. it is missing).

We will further denote $\dot{x}_{i,j}^{[t]}$ for the imputation of missing values on the iteration t. Besides, $\boldsymbol{x}_i$ corresponds to the i-th instance of the dataset $\mathcal{D}$ and $\dot{\boldsymbol{x}}_i^{[t]}$ is the i-th instance after imputation t.

A model h (e.g. Autoencoder, Random Forest, ...) is parametrized by $\boldsymbol{\theta}$, and $p_t(\dot{\boldsymbol{x}}_i^{[t]} \mid \dot{\boldsymbol{x}}_i^{[t-1]}, \boldsymbol{\theta})$ is the probability that random vector $\tilde{\boldsymbol{x}}_i$ takes value $\dot{\boldsymbol{x}}_i^{[t]}$ at iteration t.

Formally, each of the missing value types can be formalized as follows. Let $\mathcal{D} = \{X \cup \tilde{X}\}$ be a dataset, where X and $\tilde{X}$ represent observed and missing data, respectively, and $X_{i,j}$ is the observed value of the observation i for the variable j.

Let $\mathcal{R}$ represent the indicator matrix where $R_{i,j} = 1$ if $X_{i,j}$ is observed and $R_{i,j} = 0$ if $X_{i,j}$ is missing.

Further, let $P(\mathcal{R}_{i,j} = 1)$ be read as *the probability that the j-th feature of the i-th row be missing*. We consider the MCAR (Missing Completely At Random) framework, where this probability P is a Bernoulli distribution of unknown parameter π: $\mathcal{R}_{i,j} \sim P_\pi(\cdot)$; unknown but assumed to be independent of all other missingness. On the other hand, MAR (Missing At Random) is the case where $\mathcal{R}_{i,j} \sim P_\pi(\cdot \mid \boldsymbol{x}_i)$ (i.e., missingness depends on other observed features). Finally, the MNAR (Missing Not At Random) scenario is when $\mathcal{R}_{i,j} \sim P_\pi(\cdot \mid \tilde{\boldsymbol{x}}_i)$, i.e. depends on the missing values themselves.

The aforementioned scenarios can be frequently encountered in real-world situations. For instance, MCAR is commonly found in biological data, and in particular, Single Nucleotide Polymorphisms (SNP) data used in experiments in this paper. Often, some of the numerous features obtained from the genome are not valid due to a failure of the tests, the machines that carry on the analyses, or the mistake of the practitioner. These faults can not be directly associated with any specific cause and they are considered random. At the same time, if the source can be recognized, for instance, because there is a faulty machine, the MNAR scenario may be found, e.g. if a specific machine is prone to output samples as positive, rather than negative due to faulty behavior. Finally, the MAR can be described as when the machine is not able to analyze a specific type of individual. Hence, the dataset would result in the missingness of an entire observation due to its nature.

2.2 Missing value imputation: procedural vs iterative

In this work, we particularly distinguish between the ways how the missing values can be imputed, i.e. procedurally or iteratively. **Procedural** methods impute values only once, based on the observed values. **Iterative** methods first impute values randomly and then update these imputations until some convergence criterion is met. We note, that a method belonging to one of these families, might be easily adaptable to another one.

A general schema for the procedural methods is given in Algorithm 1. The following example below illustrates one-shot row-wise procedural imputation (blue represents training samples):

$$\begin{bmatrix} \tilde{x}_{1,1} & \tilde{x}_{1,2} & x_{1,3} \\ x_{2,1} & \tilde{x}_{2,2} & x_{2,3} \\ \tilde{x}_{3,1} & x_{3,2} & x_{3,3} \\ \color{blue}{x_{4,1}} & \color{blue}{x_{4,2}} & \color{blue}{x_{4,3}} \\ \color{blue}{x_{5,1}} & \color{blue}{x_{5,2}} & \color{blue}{x_{5,3}} \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{x}_{1,1}^{[1]} & \dot{x}_{1,2}^{[1]} & x_{1,3} \\ x_{2,1} & \dot{x}_{2,2}^{[1]} & x_{2,3} \\ \dot{x}_{3,1}^{[1]} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \\ x_{5,1} & x_{5,2} & x_{5,3} \end{bmatrix}$$

Algorithm 1 General framework for procedural imputation

```
1: procedure PROCEDURAL IMPUTATION($\mathcal{D} = \{X \cup \tilde{X}\}$)
2:     $h \leftarrow \mathsf{Train}(\{\boldsymbol{x}\})$                ▷ Train the model with complete data
3:     $\{\dot{\boldsymbol{x}}\} \leftarrow h(\{\tilde{\boldsymbol{x}}\})$                ▷ Predict the missing values with $h$
4: end procedure
```

Algorithm 2 General framework for iterative imputation

```
1: procedure ITERATIVE IMPUTATION($\mathcal{D} = \{X \cup \tilde{X}\}, \varepsilon$)
2:     $\{\dot{\boldsymbol{x}}\}^{t=0} \leftarrow \mathcal{R}(\tilde{x})$                ▷ Random imputation of missing values
3:     while $\Delta_{imp} > \varepsilon$ do
4:         $h^t \leftarrow \mathsf{Train}(\{\boldsymbol{x}\} \cup \{\dot{\boldsymbol{x}}\}^{t-1})$
5:         $\{\dot{\boldsymbol{x}}^t\} \leftarrow h^t(\{\dot{\boldsymbol{x}}\}^{t-1})$
6:         $\Delta_{imp} \leftarrow \mathsf{convergence}(\{\dot{\boldsymbol{x}}\}^t, \{\dot{\boldsymbol{x}}\}^{t-1}, h^t)$     ▷ Compute convergence criteria
7:     end while
8: end procedure
```

Algorithm 2 summarizes the general schema for iterative imputation and the following example illustrates an approach (all samples are used for training):

$$\begin{bmatrix} \tilde{x}_{1,1} & \tilde{x}_{1,2} & x_{1,3} \\ x_{2,1} & \tilde{x}_{2,2} & x_{2,3} \\ \tilde{x}_{3,1} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \\ x_{5,1} & x_{5,2} & x_{5,3} \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{x}_{1,1}^{[0]} & \dot{x}_{1,2}^{[0]} & x_{1,3} \\ x_{2,1} & \dot{x}_{2,2}^{[0]} & x_{2,3} \\ \dot{x}_{3,1}^{[0]} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \\ x_{5,1} & x_{5,2} & x_{5,3} \end{bmatrix} \Rightarrow \dots \Rightarrow \begin{bmatrix} \dot{x}_{1,1}^{[t]} & \dot{x}_{1,2}^{[t]} & x_{1,3} \\ x_{2,1} & \dot{x}_{2,2}^{[t]} & x_{2,3} \\ \dot{x}_{3,1}^{[t]} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \\ x_{5,1} & x_{5,2} & x_{5,3} \end{bmatrix} \Rightarrow \dots$$

and thus so for $t + 1, t + 2, \dots$ until some convergence is established.

2.3 Missing value imputation: related work

We will now review existing work from the literature on MVI, roughly categorized into the above-mentioned approaches, procedural and iterative.

2.3.1 Basic statistical imputation

The procedural methods range from rather simple ones such as replacement with the column-wise mean, mode, or median statistics (Little and Rubin 2019) to techniques such as *k*-Nearest Neighbours (kNN) (Schwender 2012) and Cascade Imputation (CIM) (Montiel et al. 2018), leveraging machine learning methods. While the kNN method processes the data row-wise, i.e. extracting information from the *k* instances that are most similar to the one whose missing values need to be replaced, CIM first rearranges data, so that missing values may be imputed block by block.

2.3.2 Denoising autoencoders (DAE)

Using Denoising Autoencoders (DAE) for MVI (Vincent et al. 2008) can be considered a state-of-the-art; wherein missing values are treated as 'noise'. They may be trained either procedurally on complete data (ignoring missing values) or iteratively, with missing values randomly imputed first (i.e., as noisy values), and then iteratively re-trained on updated re-imputed data until convergence, as in, e.g., (Seo et al. 2022; Wright 2015; McCoy et al. 2018). Principal Component Analysis (PCA), which indeed can be seen as a special (linear) autoencoder, has been used in a similar way (Dray and Josse 2015); often done via singular value decomposition (SVD), e.g., (Troyanskaya et al. 2001).

Classical Autoencoders implemented within neural networks architecture consist of encoder and decoder structures as illustrated in Fig. 1a. A classical Autoencoder learns to embed the input data into a hidden representation aiming at a low reconstruction error when decoded. Denoising Autoencoders try to eliminate noise from the data by first manually corrupting the input data, embedding such input into a hidden representation, and then performing reconstruction (see Fig. 1b). The reconstruction error is computed against the clean input data and hence, the Autoencoder learns to clean noisy data or, in other words, impute missing values. While the inner structure of hidden layers of these approaches can be very different, the typical common property is having at least one narrow middle layer H (so-called bottleneck) to restrict the model to learning only important information from the data.

2.3.3 Multiple imputation with chained equations (MICE)

Another well-known approach for MVI is Multiple Imputation with Chained Equations (MICE) (Buuren and Groothuis-Oudshoorn 2011). MICE, like iterative DAE approaches, also initially imputes the missing values randomly, but proceeds in a column-wise leave-one-out scheme using an off-the-shelf method (base learner). The procedure is further independently repeated with the goal to obtain multiple candidate values for imputation, i.e., an ensemble-like approach. The well-known MissForest imputer (Stekhoven and Buhlmann 2011) can be seen as a special case of MICE with Random Forest chosen as a base learner.

We need to mention also the idea of applying Gaussian Processes for MVI proposed in Jafrasteh et al. (2023). Gaussian Processes are non-parametric models and output predictive distributions for target variables, which can be also considered as uncertainty estimates. Sparse Gaussian Processes have been applied for MVI leveraging the idea of the MICE method where the features are processed one by one in a cascaded fashion. While obtaining promising results, the proposed method has a much higher computational cost both for

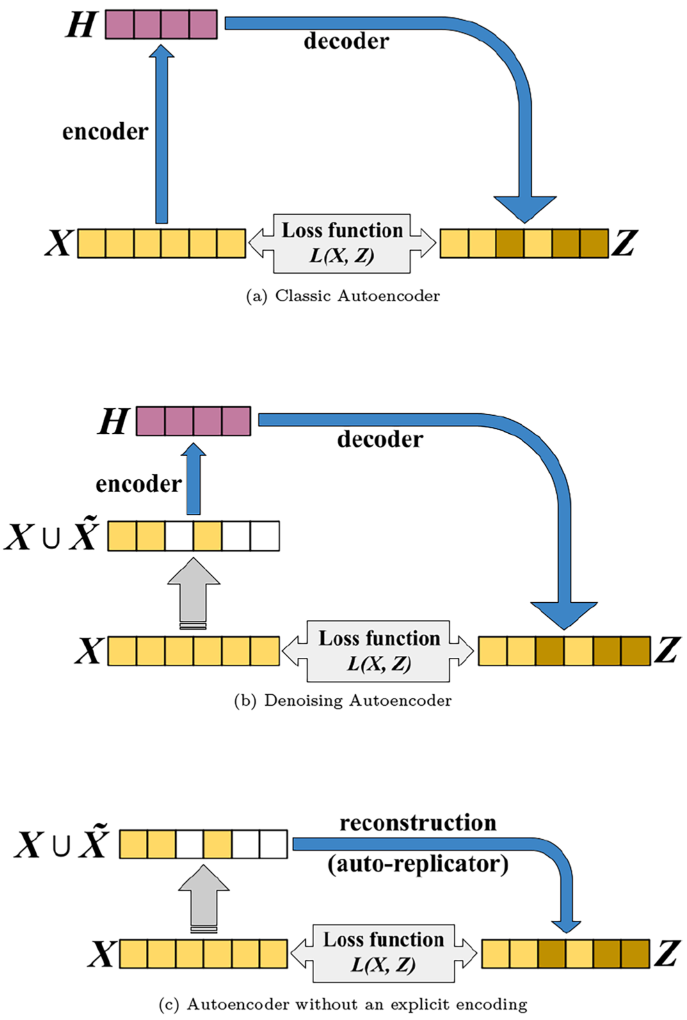

Fig. 1 **a** Classic Autoencoder embeds the input data into a hidden representation H, often referred to as bottleneck, **b** Denoising Autoencoder, where input is corrupted with noise or missing values as $X \cup \tilde{X}$ before encoding, and **c** Autoreplicator, or an Autoencoder without an explicit encoding, where we propose to bypass hidden representation and directly reconstruct input from its corruption $X \cup \tilde{X}$. In all cases, the goal is to minimize the difference between input X and its reconstruction Z

training and prediction than other baselines including already computationally expensive MICE. Also, in Jafrasteh et al. (2023), Gaussian Processes are applied for continuous variables and assume a normal distribution for each, which may not always be the case.

2.3.4 Expectation-maximization and coordinate-ascent methods

The approach of iterative imputation takes the general schema of coordinate ascent, with special cases including expectation maximization (EM) and classification maximization (CM, i.e. 'hard EM') (MacKay 2003; Dempster et al. 1977) where the expectation step E is replaced by a hard classification step (an actual value is imputed), and the maximization step M refers to training. Inspired by this idea, many MVI algorithms start from a random or data-driven (mode, mean, median, ...) initial imputation so that any classifier can be trained on the entire dataset. Next, the missing values are predicted and the model is re-trained after each imputation. This process is repeated until convergence is reached.

2.3.5 Other approaches

We would also like to mention the work (Van Wolputte and Blockeel 2020) which uses a Random-Forest-based predictor to perform imputation in the prediction phase assuming complete data in the training phase which corresponds to a different problem setting. While we think that with some extra effort, one could adapt this method to the setup described in our work and thus incorporate it into the framework, we leave this question for future research.

2.4 Summary, and framework parameters

Most of all iterative methods listed above (DAE, SVD, PCA) may be considered multi-output predicting models, as they impute all missing values simultaneously. Oppositely, Multivariate Imputation by Chained Equations (MICE) is also iterative but single-output,

Table 1 Some example methods as specific parametrizations of a general framework. Imputation types CW = column-wise, RW = row-wise, BW = block-wise. Strategies p = procedural, it = iterative. Method families SO = single-output, MO = multi-output. Taking uncertainty into account SI = single(standard)-imputation, MI = multiple imputation where '-' indicates that it could be implemented, but we are not aware of any reference doing so; with Ensemble (MICE) or via a predictive posterior Distribution ($\tilde{x} \sim p(\cdot \mid \dot{x})$) being options

Method	Strategy	SO/MO	Type	MI	References
Mode	p	SO	CW	No	Little and Rubin (2019)
kNN	p	MO	RW	No	Schwender (2012)
MICE	it	SO	CW	Ens	Buuren and Groothuis-Oudshoorn (2011)
CIM	p	SO	BW	–	Montiel et al. (2018)
DAE	p	MO	RW	–	Vincent et al. (2008)
DAE	it	MO	RW	–	Seo et al. (2022)
PCA/SVD	it	MO	RW	–	Dray and Josse (2015); Troyanskaya et al. (2001)
ARF	p	MO	RW	–	This work
ARF	it	MO	RW	–	This work
ditARF	it	MO	RW	Dist	This work

as it processes features consequently in a leave-one-out manner. The MICE method is very flexible with regard to the base model, i.e. any per-feature estimator is possible. The MICE method is commonly used for different types of data and, in particular, clinical data, and can be considered state-of-the-art for MVI, but we have not found substantial evidence of using the MICE method for high-dimensional datasets, as the computational cost drastically increases in this setting. The CIM method mentioned above may be considered a procedural version of MICE. Table 1 summarizes the characteristics of the discussed methods.

Obviously, multiple variations of the methodologies shown above can be discussed. Although we believe that these are out of the scope of this work, we think that is worth sharing some insights. In procedural imputation, the main characteristic is that the imputation is only done once for each missing value. Following this idea, we could introduce the *row-wise imputation* in which we increasingly impute the missing values through time and we relearn the imputation model after each procedure. By using this procedure, we would add more true values to the training dataset after every imputation, but, at the same time, we might supply incorrect imputations to the model as ground truth. Similar to the previous approach, there is the *column-wise* imputation that matches the MICE imputation strategy.

2.4.1 Estimator

For the MICE method, any single-output estimator can be used. As a default parameter, we use Random Forests (of 20 trees each) as they proved to be a robust and stable method, though any other classifier may be provided manually to the framework. Among multi-output methods, we propose including Autoencoders and PCA as a standard choice and Autoreplicative Random Forests as a novelty (see Sect. 3).

2.4.2 Initial imputation

For iterative methods, an initial pre-starting imputation is needed. There are several possibilities for that: imputing all missing values with a constant, imputing with modes of the values for each feature, or imputing randomly with a uniform or simulated distribution over the observed values. We use random imputation with uniform distribution over the observed values as a default value.

2.4.3 Number of iterations

For iterative methods, the re-predictive process stops when the convergence is reached. To measure convergence, we calculate the fraction of the number of labels changed after the last imputation and the number of all missing labels. If this fraction is lower than a provided parameter ε, set as default to 0.005, we stop and recover the last imputed dataset as the final estimation. However, to keep the overall complexity feasible and to avoid infinite loops, we provide a maximum number of iterations parameter that we set to 10 as default.

3 Autoreplicative random forests

Following the description of the general imputation framework, we first introduce a new imputation approach, Autoreplicative Random Forests. Secondly, we propose its distributional extension.

3.1 Autoreplicative random forests (ARF)

Although apparently largely overlooked in the literature, we have noticed that any other model designed for multi-label prediction can be used instead of a neural network as an Autoreplicator for data denoising. One such example is a combination of Decision Trees (İrsoy and Alpaydin 2016) where the first Decision Tree is used as an encoder, and the second one is used in a vice versa manner as a decoder. Meanwhile, this idea can be simplified even more: in our approach, we will use a multi-output Random Forest as an estimator.

In contrast to Denoising Autoencoders, we use Random Forests as Autoreplicators without an explicit encoding/representation, as shown in Fig. 1c, implicitly as an off-the-shelf multi-label model. Not having an explicit latent representation in matrix form does not concern us, as for MVI we aim to directly reconstruct the input from its corruption without modeling hidden structure.

Random Forests have been selected since they naturally are multi-label and multi-class classifiers and they proved to be competitive and robust classifiers in several works (Wood et al. 2023). Such an approach can facilitate the optimization process for the model on data containing a small number of samples, and at the same time, tree-based models are both efficient and simple to understand and interpret. To the best of our knowledge, this simple but efficient idea has not been well studied in the literature. We argue, that however it deserves attention and can be further investigated. Applying this idea, we suggest further Autoreplicative Random Forests.

It is worth noting, that while we choose Random Forests as a well-known and stable multi-label method with good performance, this idea may be developed by using other multi-label methods, such as e.g. Classifier Chains (Read et al. 2011), Multilabel k Nearest Neighbours (Zhang and Zhou 2007), Random k-Labelsets (Tsoumakas and Vlahavas 2007), Conditional Dependency Networks (Guo and Gu 2011). However, a survey of base learners is not the main objective of this paper.

In the procedural approach, further referred as procedural Autoreplicative Random Forest (pARF), we first select complete instances X of the entire dataset $\mathcal{D} = \{X \cup \tilde{X}\}$, corrupt them manually to $\tilde{X}'$ with induced missing values (uniformly distributed, following the missing value ratio in the original dataset), and train an Autoreplicative Random Forest to reproduce $Z \sim X$, i.e. fill missing values in $\tilde{X}'$ by minimizing loss function between Z and X. In other words, a multi-label Random Forest is trained to predict p outputs corresponding to X from p features corresponding to corrupted $\tilde{X}'$. Then the fitted model is used to impute actual missing values in the instances $\tilde{X}$. In the usage of iterative Autoreplicative Random Forests (itARF), values should be first imputed randomly, then a Random Forest is re-trained in an iterative manner, on iteration t receiving $\mathcal{D} = \{X \cup \tilde{X}\}$ as an input, learning to reproduce $Z \sim \dot{X}^{[t-1]}$ as output and storing a prediction $\dot{X}^{[t]}$ as a new imputation.

3.2 Distributional iterative ARF (ditARF)

A known issue of using MVI in a machine-learning pipeline is the imputation of imperfect values. Imputation is inherently imperfect, but furthermore, masks the information that values were imputed as well as any associated uncertainty about such values.

To address this issue, methods such as MICE propose using the technique of 'multiple imputation', that is repeating the imputation several times independently (essentially, bootstrapping) in order to obtain multiple plausible values and run further analysis on these datasets.

Here we propose a distributional variant of ARF (ditARF) which provides a probability distribution associated with imputations, i.e., encapsulating and expressing the uncertainty associated with any imputation.

And in particular, we take into account a novel consideration not embraced by other methods; namely a model of the joint distribution for a given instance with missing values. Whereas imputation from a marginal distribution (j-th feature) can be expressed as

$$\dot{x}_j = \underset{\tilde{x}_j}{\operatorname{argmax}}\, p(\tilde{x}_j \mid \boldsymbol{x}), \tag{1}$$

the imputation (full row/vector) from a joint distribution is expressed as

$$\dot{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}}}{\operatorname{argmax}}\, p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}). \tag{2}$$

There is an issue with a naive implementation of multi-output Random Forests as formally this model produces an empirical distribution that can be interpreted (with some generalization) as

$$p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \prod_{j=1}^{p} p(\tilde{x}_j \mid \boldsymbol{x}) \tag{3}$$

which assumes that each imputed feature is conditionally independent of the others for a given instance. This may not be the case in real-world data, and ignoring feature

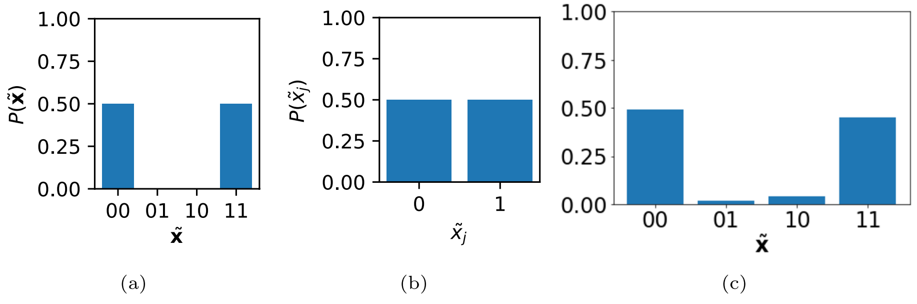

Fig. 2 Illustrating the difference between the joint and marginal distributions of two binary missing-value variables $\tilde{\boldsymbol{x}} = \{\tilde{x}_1, \tilde{x}_2\}$ of an instance. The marginal distribution (the same distribution covers both variables, having been marginalized from the joint) indicates that all combinations of values 0 and 1 are equally likely **b**, even though only two such combinations would occur **a**. A multi-output Random Forest may impute values from which such impossible combinations appear **c**. This indicates the potential importance of joint modeling, which would produce $\dot{\boldsymbol{x}} \in \{00, 11\}$ according to Eq. (2)

dependencies can hinder the accuracy of imputation. Indeed, in certain application domains such as medicine, it may be a critical mistake to make this assumption (Gerych et al. 2021).

Consider the illustration of Fig. 2, where the joint distribution $p(\tilde{\boldsymbol{x}})$ only gives non-zero probability to two values ($x_1x_2 = 00$ and $x_1x_2 = 11$), yet the marginal probability (as would be estimated by Random Forest under Eq. (3)) indicates the equal probability for all combinations (00, 01, 10, 11). This means that even though the dataset does not contain values for two of the possible combinations, a Random Forest would produce them as predictions. As an example, a Random Forest predicting gender and type of cancer may assign a male gender and the presence of ovarian cancer, which do not co-exist in reality.

The proposed ditARF variant is similar to itARF (introduced in Subsection 3.1) and learns to predict missing values iteratively. However, at every iteration, the instances are weighted by the output joint probability.

The Label Powerset (LP) method (Tsoumakas and Katakis 2007), for example, transforms each combination of output values into a unique class and thus naturally models the labels jointly. However, such an approach could not be applied in the iterative setting as initial imputation creates value combinations that may not exist in the data while closing the opportunity to learn other possible combinations in future iterations.

We can imagine adapting, for example, a less strict and more generalized version of the LP approach, the Random *k*-Labelsets method (Tsoumakas and Vlahavas 2007), to tackle this issue, as well as inducing some randomness at each iteration. However, these possible solutions are out of the scope of this work and we leave them for future research.

The proposed solution is closely related to other well-known iterative methods such as the Expectation Maximization (EM) algorithm (Dempster et al. 1977) and to more general coordinate-ascent methods (Wright 2015). Such methods find the maximum likelihood parameters for the corresponding model based on data. In the case of the EM, it can be used to fit a mixture of Gaussian distribution models while the coordinate-ascent method just performs a linear optimization in the log-likelihood function by iteratively learning and predicting data. DitARF also maximizes the log-likelihood after each iteration, which is computed as

$$\log \mathcal{L}(\boldsymbol{\theta} \mid \mathcal{D}) = \sum_{i=1}^{N} \max_{j=1}^{p} \log(p(\tilde{x}_{i,j} \mid \boldsymbol{x}_i)).$$

Similar to the EM algorithm, ditARF considers a set of weights $\mathbf{w}$, one per instance,

$$w_i = \prod_j \max p(\tilde{x}_{i,j} \mid \boldsymbol{x}_i).$$

These weights are used when training a Random Forest classifier as weights for each instance, thus giving higher weights for instances where the model is more confident about the imputation. Following the strategy of the iterative version of ARFs, a Random Forest is iteratively re-trained until it reaches convergence in terms of likelihood. When this occurs, an estimate of the joint posterior distribution $\tilde{\boldsymbol{x}}^{[t]} \sim P$ is obtained and hence, we provide $p(\tilde{x}_{i,j} \mid \boldsymbol{x}_i)$ as a measure of uncertainty along with the imputed missing value $\dot{x}_{i,j}$.

Table 2 Datasets used in experiments, p features, N samples. Discretized continuous datasets are marked with ^{d}. High-dimensional datasets ($p > N$) are marked with *. Datasets with a large number of samples are marked with ♦

Name	p	N	References
Mushroom	22	8124	Dua and Graff (2017)
Soybean	35	307	Dua and Graff (2017)
Primary Tumor	17	339	Dua and Graff (2017)
Lymphography	18	148	Dua and Graff (2017)
Congressional Voting Records	16	435	Dua and Graff (2017)
Financial Survey	212	6394	CFPB (2017)
Nursery	8	12,960	Dua and Graff (2017)
Splice	60	3190	Dua and Graff (2017)
Land Use	50	632	Karimi (2023)
SNP Maize*	1000	247	Negro et al. (2019)
SNP Eucalyptus*	1000	970	Grattapaglia (2019)
SNP Wheat*	1000	388	Reif (2020)
Yeast^d	8	1484	Dua and Graff (2017)
Metro^d♦	17	1,516,948	Dua and Graff (2017)
Energy^d♦	7	2,049,279	Dua and Graff (2017)

4 Experimental study

In order to compare the performance of the proposed solution, we perform several experiments on real-world datasets obtained from the UCI repository (Dua and Graff 2017) as well as on three high-dimensional ($p > N$) Single Nucleotide Polymorphism (SNP) datasets which we have truncated to 1000 features in order to bound the memory consumption. Most of these datasets contain categorical multinomial variables, while three of them are described by continuous variables which we uniformly discretize to b bins (Yeast, $b = 3$; Metro, $b = 3$; Energy, $b = 2$). We include Metro and Energy to demonstrate the methods' performance in a setting with a big number of samples. The datasets used in the experiments are summarized in Table 2. So as to properly simulate missing values in real-world situations, we followed the MCAR strategy by corrupting a percentage of the data values. These percentages range from 1% to 30%. We refer to this parameter as the Missing Value Ratio (MVR) throughout the text.

For the purpose of evaluating the proposed solution, we consider marginal accuracy, which is also known as Hamming Score, among the imputed values; and joint accuracy also referred to as Exact Match in the literature. Formally, marginal accuracy can be defined as

$$\frac{1}{N_m}\frac{1}{p_m^i}\sum_{i=1}^{N_m}\sum_{j=1}^{p_m^i}\mathbb{1}(\dot{x}_{i,j}, x_{i,j}), \tag{4}$$

where N_m and p_m^i refer to the number of instances and the number of features per instance with missing values, respectively. Similarly, joint accuracy can be defined as

$$\frac{1}{N_m}\sum_{i=1}^{N_m}\mathbb{1}(\dot{\boldsymbol{x}}_i,\boldsymbol{x}_i). \tag{5}$$

Finally, since MVI is usually a preprocessing step for further classification tasks, we compare the classification accuracy obtained with a Random Forest classifier trained on full data, and on imputed data. The experiments have been run 5 times and the average of the scores of all runs is used.

We compare our method against a variety of well-known approaches from the literature. Autoencoder and PCA methods are implemented using the scikit-learn (Pedregosa et al. 2011) package. We tested the performance of both procedural and iterative Autoencoders in three variants: with one hidden layer of $0.1p$ neurons, one hidden layer of $0.2p$ neurons, or three hidden layers of $0.2p$, $0.1p$, and $0.2p$ neurons respectively, where p is the number of features. The model with one hidden layer of $0.1p$ neurons has shown slightly better performance, although the difference was not significant. The results of this model are further presented. The PCA method was also realized as a neural network with one hidden layer of $0.1p$ neurons but with an identity activation function. The kNN method is presented with the number of neighbors $k = 2$ selected during inner validation where it consistently outperformed other k values.

In order to select the best-performing parameters, we have internally run a grid search over the parameters of Autoreplicative Random Forests. As a result, we opted to use 20 trees (base classifiers) per forest (no significant difference compared to other values), each tree trained on all p provided features (better performance than with default $\sqrt{p}$ parameter), a minimal number of samples per split equal to 5. Criterion (gini/entropy) has not shown an influence on the method's performance. Other hyperparameters of the competitors are used with default values shown in their original papers and implemented in the ScikitLearn python library.

The main aim of this experimental study is to answer the following research questions:

(a) Analyze the imputation performance of the proposed solution and its competitors in terms of marginal (entry-wise) and joint (row-wise) accuracy.
(b) Study the performance of the methods under the curse of dimensionality (when $p > N$).
(c) Evaluate the impact of the number of features (p) and the number of samples (N) for each of the imputation methods w.r.t. the time taken to finish their imputation as well as predictive performance.

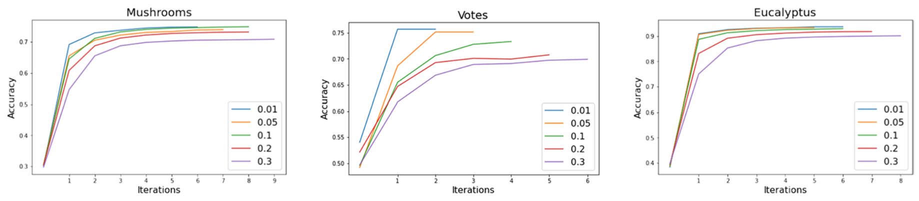

Fig. 3 Convergence of the itARF method (accuracy vs number of iterations)

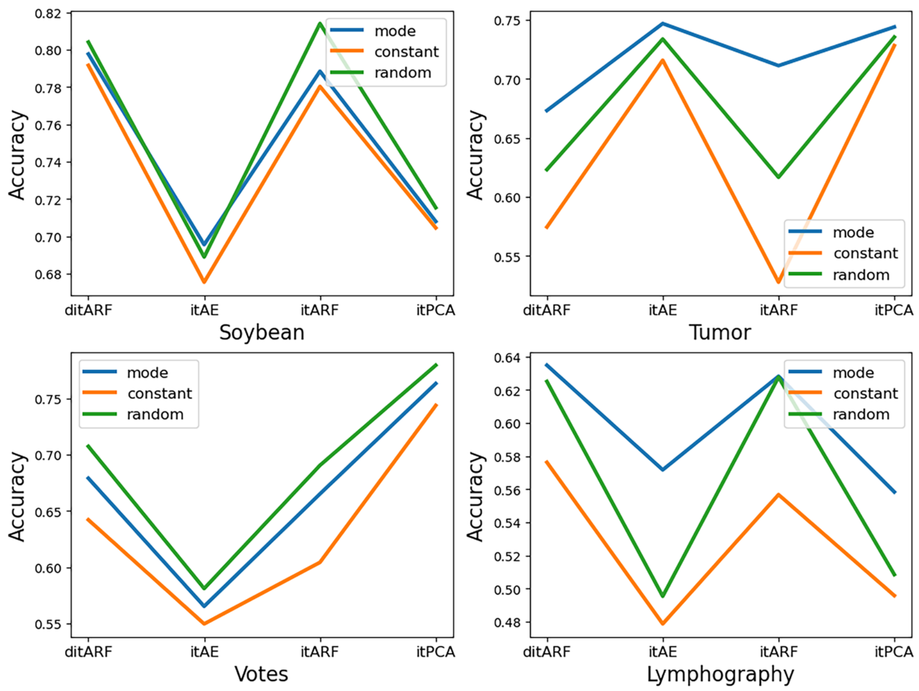

Fig. 4 Comparison of initial imputation strategies for iterative methods on four datasets. The results are averaged across 5 different missing value ratios and 5 independent runnings

4.1 Results and discussion

4.1.1 Imputation performance

First, we empirically evaluate the convergence of the proposed itARF method and demonstrate the results for three datasets and different missing value ratios in Fig. 3. We observe that in all cases accuracy monotonously increases and reaches its maximum after several iterations. The number of iterations is shown to be small enough to maintain a feasible computation time of itARF imputation.

Further, we evaluate three different initial imputation strategies for iterative methods, namely imputing with a constant (0), with a mode of each feature, and with a randomly selected value from the set of the observed ones for the corresponding feature. The results illustrated in Fig. 4 suggest that imputing with a constant consistently provides the worst imputation accuracy for all iterative methods, while imputing randomly and with modes are competitive, and the best choice may depend on the dataset. In all further experiments, a random initial imputation is used.

Then, we evaluate marginal and joint accuracies for the imputed missing values. Table 3 summarizes the performance of all methods measured by the marginal accuracy, i.e. percentage of correctly imputed values out of the missing ones. Table 4 shows joint accuracy, i.e. percentage of the instances where *all* values were imputed correctly. MICE results are not shown for the datasets with a large number of features because of excessive computation time. The ditARF method was not evaluated on the datasets with

Table 3 Marginal accuracy. The best accuracy per column is in **bold**. The second best accuracy is underlined. All results are rounded to 3 dp. For [it]erative (includes MICE) and [p]rocedural versions of methods

MVR	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3
	Mushroom					Soybean					Tumor				
Complete cases	80.1%	32.3%	10.1%	0.7%	0.04%	69.7%	13.7%	1.0%	0%	0%	83.8%	38.9%	15.0%	1.2%	0%
MICE	0.649	0.698	0.730	**0.753**	**0.767**	**0.867**	**0.873**	**0.875**	**0.838**	**0.823**	**0.775**	**0.749**	**0.778**	**0.749**	0.725
ditARF	0.748	**0.763**	0.747	0.727	0.699	0.830	0.845	0.818	0.768	0.755	0.639	0.660	0.679	0.653	0.658
itARF	0.741	0.742	0.746	0.727	0.684	0.809	0.844	0.829	0.776	0.777	0.604	0.653	0.665	0.643	0.645
pARF	**0.767**	**0.763**	**0.764**	0.684	0.514	0.774	0.780	0.656	–	–	0.572	0.654	0.705	0.687	0.698
itAE	0.605	0.587	0.595	0.563	0.566	0.667	0.718	0.699	0.682	0.673	0.702	0.708	0.742	0.726	**0.727**
pAE	0.574	0.517	0.500	0.530	0.525	0.667	0.725	0.642	–	–	0.702	0.701	0.743	0.683	0.606
itPCA	0.613	0.611	0.612	0.607	0.596	0.710	0.742	0.721	0.688	0.692	0.702	0.712	0.739	0.727	**0.727**
pPCA	0.609	0.585	0.571	0.532	0.490	0.686	0.729	0.662	–	–	0.702	0.610	0.717	0.588	0.522
kNN	0.642	0.659	0.678	0.670	0.569	0.731	0.774	0.768	0.729	0.697	0.526	0.549	0.594	0.559	0.507
	Votes					Lymphography					Financial Survey				
Complete cases	85.3%	42.2%	18.5%	1.3%	0%	81.8%	40.5%	14.9%	2.7%	0%	11.8%	0%	0%	0%	0%
MICE	**0.888**	**0.774**	**0.772**	**0.758**	**0.768**	0.562	**0.677**	**0.621**	**0.633**	**0.643**	–	–	–	–	–
ditARF	0.712	0.708	0.697	0.684	0.703	0.669	0.556	0.608	0.590	0.610	**0.687**	**0.674**	0.670	0.663	0.655
itARF	0.765	0.703	0.696	0.682	0.689	0.677	0.546	0.609	0.583	0.606	0.676	0.670	**0.673**	**0.664**	**0.656**
pARF	0.653	0.722	0.720	0.712	–	**0.708**	0.586	0.610	0.513	–	0.668	–	–	–	–
itAE	0.612	0.634	0.591	0.531	0.553	0.462	0.480	0.538	0.461	0.465	0.622	0.618	0.616	0.606	0.589
pAE	0.594	0.616	0.537	0.596	–	0.462	0.466	0.553	0.465	–	0.518	–	–	–	–
itPCA	0.676	0.634	0.620	0.603	0.556	0.431	0.517	0.535	0.466	0.462	0.649	0.647	0.645	0.636	0.623
pPCA	0.629	0.528	0.537	0.548	–	0.446	0.493	0.550	0.472	–	0.625	–	–	–	–
kNN	0.824	0.615	0.667	0.625	0.641	0.346	0.406	0.436	0.425	0.447	0.490	0.489	0.491	0.490	0.487

Table 3 (continued)

	Nursery					Splice					Land Use				
Complete cases	92.3%	66.4%	43.3%	16.8%	5.6%	53.8%	4.0%	0.1%	0%	0%	59.7%	8.4%	0.9%	0%	0%
MICE	0.021	0.112	0.196	0.265	0.296	**0.431**	**0.438**	**0.433**	**0.415**	**0.390**	**0.394**	**0.349**	**0.369**	**0.371**	**0.359**
ditARF	0.299	0.291	0.297	0.310	0.314	0.399	0.373	0.350	0.321	0.306	0.338	0.296	0.302	0.298	0.300
itARF	0.297	0.289	0.295	0.308	**0.318**	0.384	0.379	0.368	0.341	0.325	0.316	0.312	0.301	0.311	0.295
pARF	0.298	0.290	0.295	0.311	0.314	0.349	0.287	0.275	–	–	0.322	0.292	0.255	–	–
itAE	0.312	0.306	0.306	0.317	0.317	0.236	0.243	0.245	0.243	0.233	0.173	0.134	0.122	0.117	0.118
pAE	0.305	0.303	**0.308**	**0.320**	**0.318**	0.244	0.243	0.260	–	–	0.174	0.151	0.163	–	–
itPCA	0.312	0.306	0.306	0.318	0.317	0.274	0.276	0.278	0.267	0.260	0.271	0.213	0.210	0.192	0.181
pPCA	**0.313**	**0.307**	**0.308**	0.317	0.317	0.268	0.243	0.268	–	–	0.218	0.166	0.176	–	–
kNN	0.181	0.145	0.182	0.200	0.211	0.231	0.237	0.236	0.226	0.210	0.155	0.168	0.169	0.182	0.173
	SNP Maize*					SNP Eucalyptus*					SNP Wheat*				
Complete cases	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
MICE	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ditARF	0.837	0.819	0.798	0.755	0.694	**0.936**	0.918	0.908	0.848	0.782	0.923	0.931	0.920	0.904	0.920
itARF	**0.857**	**0.846**	**0.835**	**0.825**	**0.817**	0.935	**0.933**	**0.929**	**0.915**	**0.901**	**0.942**	**0.940**	**0.937**	**0.933**	**0.934**
pARF	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
itAE	0.724	0.724	0.717	0.717	0.715	0.715	0.723	0.720	0.715	0.706	0.931	0.929	0.931	0.931	0.931
pAE	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
itPCA	0.725	0.694	0.672	0.645	0.624	0.832	0.850	0.854	0.849	0.831	0.895	0.890	0.883	0.876	0.856
pPCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
kNN	0.758	0.753	0.749	0.746	0.736	0.912	0.912	0.909	0.903	0.897	0.914	0.914	0.913	0.912	0.911

Table 3 (continued)

	Yeast^d					Metro^{d}♦					Energy^{d}♦				
Complete cases	92.4%	66.0%	43.6%	16.8%	5.8%	86.0%	46.3%	20.6%	3.5%	0.5%	93.2%	69.8%	47.8%	20.9%	8.2%
MICE	**0.761**	0.788	**0.795**	0.785	0.770	**0.959**	**0.959**	**0.958**	**0.956**	**0.953**	**0.924**	**0.924**	**0.924**	**0.923**	**0.923**
ditARF	0.690	0.791	0.776	**0.788**	**0.783**	–	–	–	–	–	–	–	–	–	–
itARF	0.697	0.793	0.781	0.787	**0.783**	0.871	0.871	0.872	0.872	0.871	0.919	0.919	0.920	0.920	0.920
pARF	0.644	0.708	0.708	0.725	0.710	0.958	0.957	0.956	0.948	0.938	0.923	0.923	0.923	**0.923**	0.922
itAE	0.754	**0.794**	0.788	0.786	**0.783**	0.934	0.931	0.932	0.930	0.922	0.920	0.921	0.921	0.920	0.921
pAE	0.754	**0.794**	0.788	0.786	**0.783**	0.949	0.949	0.947	0.949	0.946	0.923	0.923	0.923	**0.923**	**0.923**
itPCA	0.751	0.783	0.779	0.781	0.779	0.942	0.935	0.927	0.929	0.921	0.920	0.920	0.920	0.920	0.920
pPCA	0.747	0.781	0.785	0.787	0.736	0.933	0.946	0.945	0.945	0.941	0.923	0.923	0.923	**0.923**	0.922
kNN	0.576	0.604	0.607	0.604	0.574	–	–	–	–	–	–	–	–	–	–

Table 4 Joint accuracy. The best accuracy per column is in **bold**. The second best accuracy is underlined. All results are rounded to 3 dp. For [it]erative (includes MICE) and [p]rocedural versions of methods

MVR	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3
	Mushroom					Soybean					Tumor				
Complete cases	80.1%	32.3%	10.1%	0.7%	0.04%	69.7%	13.7%	1.0%	0%	0%	83.8%	38.9%	15.0%	1.2%	0%
MICE	0.622	0.563	0.480	0.307	**0.191**	**0.845**	**0.765**	**0.622**	**0.339**	**0.177**	**0.763**	**0.663**	**0.606**	**0.436**	**0.264**
ditARF	.7310	0.666	0.541	**0.321**	0.154	0.802	0.714	0.545	0.266	0.106	0.619	0.584	0.501	0.289	0.180
itARF	0.725	0.643	0.534	0.306	0.130	0.777	0.719	0.552	0.251	0.113	0.581	0.579	0.488	0.271	0.170
pARF	**0.752**	**0.669**	**0.565**	0.280	0.039	0.742	0.614	0.290	–	–	0.548	0.556	0.531	0.305	0.208
itAE	0.582	0.462	0.349	0.136	0.046	0.613	0.538	0.378	0.141	0.048	0.685	0.622	0.557	0.370	0.261
pAE	0.553	0.390	0.253	0.114	0.038	0.613	0.543	0.264	–	–	0.685	0.612	0.563	0.313	0.111
itPCA	0.591	0.492	0.374	0.179	0.069	0.662	0.559	0.396	0.149	0.084	0.685	0.628	0.546	0.370	0.261
pPCA	0.587	0.463	0.328	0.114	0.031	0.637	0.550	0.289	–	–	0.685	0.508	0.525	0.211	0.087
kNN	0.620	0.527	0.420	0.240	0.100	0.700	0.594	0.427	0.188	0.068	0.500	0.434	0.378	0.205	0.098
	Votes					Lymphography					Financial Survey				
Complete cases	85.3%	42.2%	18.5%	1.3%	0%	81.8%	40.5%	14.9%	2.7%	0%	11.8%	0%	0%	0%	0%
MICE	**0.885**	**0.701**	**0.631**	**0.470**	**0.376**	0.527	**0.560**	**0.424**	**0.262**	**0.105**	–	–	–	–	–
ditARF	0.703	0.636	0.551	0.363	0.272	0.655	0.453	**0.424**	0.201	0.086	**0.454**	**0.038**	0.001	0.000	0.000
itARF	0.770	0.633	0.554	0.343	0.229	0.645	0.437	0.416	0.215	0.088	0.442	0.035	**0.002**	0.000	0.000
pARF	0.642	0.646	0.573	0.400	–	**0.682**	0.473	0.421	0.168	–	0.432	–	–	–	–
itAE	0.600	0.538	0.403	0.230	0.096	0.455	0.389	0.336	0.105	0.019	0.384	0.021	0.001	0.000	0.000
pAE	0.582	0.535	0.388	0.248	–	0.455	0.374	0.336	0.125	–	0.270	–	–	–	–
itPCA	0.667	0.551	0.457	0.287	0.136	0.418	0.431	0.339	0.110	0.022	0.412	0.029	0.001	0.000	0.000
pPCA	0.618	0.428	0.355	0.213	–	0.455	0.409	0.336	0.137	–	0.383	–	–	–	–
kNN	0.818	0.528	0.495	0.269	0.237	0.318	0.297	0.248	0.103	0.034	0.259	0.006	0.000	0.000	0.000

Table 4 (continued)

	Nursery					Splice					Land Use				
Complete cases	92.3%	66.4%	43.3%	16.8%	5.6%	53.8%	4.0%	0.1%	0%	0%	59.7%	8.4%	0.9%	0%	0%
MICE	0.010	0.053	0.096	0.104	0.081	**0.382**	**0.213**	**0.128**	**0.065**	**0.024**	**0.327**	**0.128**	**0.041**	**0.002**	**0.002**
ditARF	0.285	0.249	0.212	0.160	0.104	0.349	0.164	0.075	0.015	0.001	0.287	0.089	0.023	0.000	0.000
itARF	0.285	0.246	0.211	0.159	0.108	0.341	0.173	0.083	0.023	0.003	0.270	0.106	0.028	0.001	0.002
pARF	0.286	0.247	0.212	0.161	0.106	0.308	0.078	0.011	–	–	0.276	0.094	0.017	–	–
itAE	**0.303**	0.266	0.230	0.173	**0.109**	0.199	0.060	0.008	0.000	0.000	0.141	0.035	0.003	0.000	0.000
pAE	0.297	0.266	**0.232**	**0.174**	**0.109**	0.206	0.060	0.011	–	–	0.141	0.047	0.003	–	–
itPCA	0.301	0.267	0.230	0.173	**0.109**	0.233	0.073	0.013	0.000	0.000	0.223	0.065	0.014	0.000	0.000
pPCA	0.302	**0.268**	**0.232**	0.171	**0.109**	0.230	0.060	0.011	–	–	0.180	0.044	0.004	–	–
kNN	0.175	0.109	0.130	0.092	0.055	0.201	0.114	0.071	0.027	0.008	0.137	0.057	0.013	0.002	0.000
	Yeast^{d}					Metro^{d}♦					Energy^{d}♦				
Complete cases	92.4%	66.0%	43.6%	16.8%	5.8%	86.0%	46.3%	20.6%	3.5%	0.5%	93.2%	69.8%	47.8%	20.9%	8.2%
MICE	**0.761**	0.758	**0.729**	0.636	0.529	**0.957**	**0.944**	**0.925**	**0.877**	**0.821**	**0.921**	**0.913**	**0.900**	**0.870**	**0.836**
ditARF	0.690	**0.762**	0.701	0.638	0.553	–	–	–	–	–	–	–	–	–	–
itARF	0.697	**0.762**	0.705	0.638	**0.555**	0.865	0.838	0.803	0.727	0.659	0.917	0.908	0.895	0.865	0.830
pARF	0.644	0.672	0.626	0.561	0.444	0.956	0.942	0.921	0.858	0.780	**0.921**	0.912	**0.900**	**0.870**	0.835
itAE	0.754	0.760	0.714	0.638	**0.555**	0.929	0.907	0.881	0.813	0.733	0.918	0.909	0.897	0.866	0.831
pAE	0.754	0.760	0.714	0.638	**0.555**	0.946	0.931	0.905	0.862	0.803	**0.921**	0.912	**0.900**	**0.870**	0.835
itPCA	0.751	0.749	0.704	0.631	0.547	0.938	0.911	0.874	0.818	0.726	0.918	0.909	0.896	0.866	0.831
pPCA	0.747	0.747	0.711	**0.639**	0.482	0.928	0.926	0.902	0.850	0.787	**0.921**	0.911	0.899	**0.870**	0.835
kNN	0.576	0.547	0.510	0.390	0.261	–	–	–	–	–	–	–	–	–	–

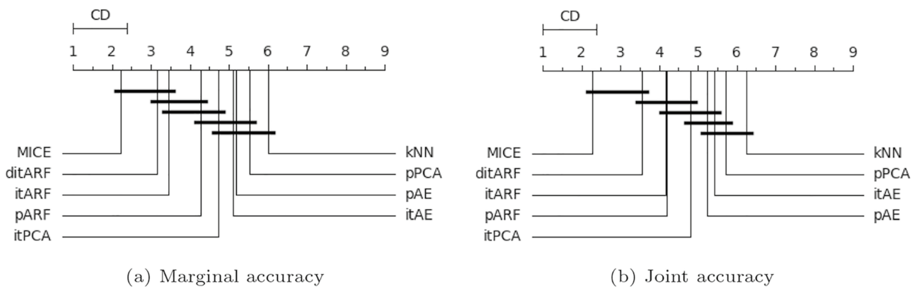

Fig. 5 Friedman–Nemenyi diagrams comparing the ranking of the experimentally tested methods. A lower rank is better, statistically indistinguishable methods are connected by a horizontal line

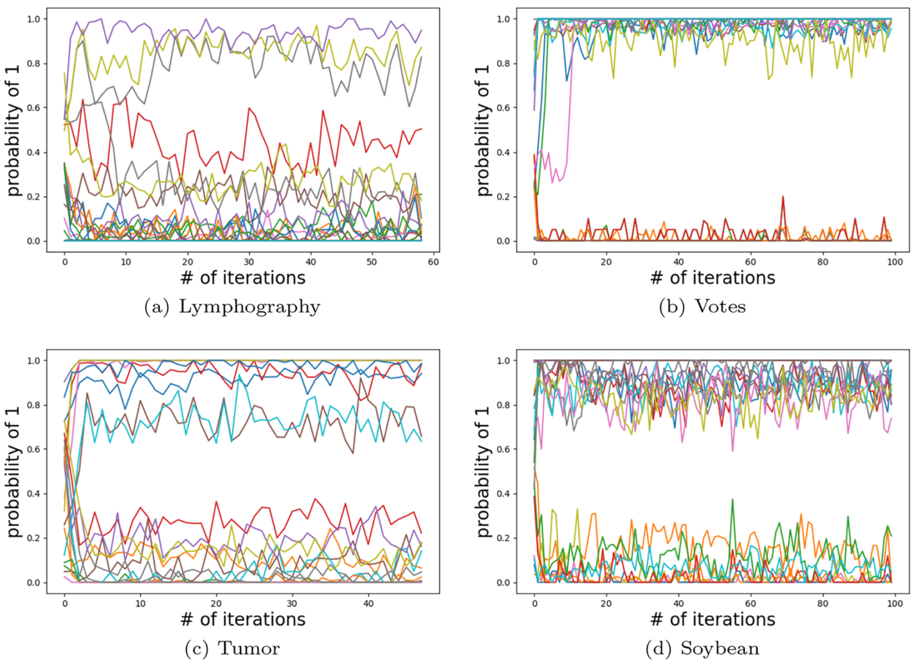

Fig. 6 Representation of the stability of the ditARF method over a set of binary missing values. Each line represents the changes in the probability of a missing value imputation throughout the different iterations. In this case, we opted to plot the $p(\tilde{\boldsymbol{x}} = 1 \mid \boldsymbol{x})$. Each plot shows the stability of the method in a different dataset

a very large number of samples for computational sake. For the same datasets, the kNN results are not included, since kNN computational time is quadratic w.r.t. the number of samples (Troyanskaya et al. 2001) and thus difficultly accessible when the number of samples is large.

When evaluated, i.e. in low-dimensional datasets, the MICE method remains very competitive. Its time consumption is significantly higher than for the ARF-based methods but stays feasible when the number of features is relatively small. The procedural and iterative ARFs show competitive performance. For the Mushroom dataset, pARF shows the

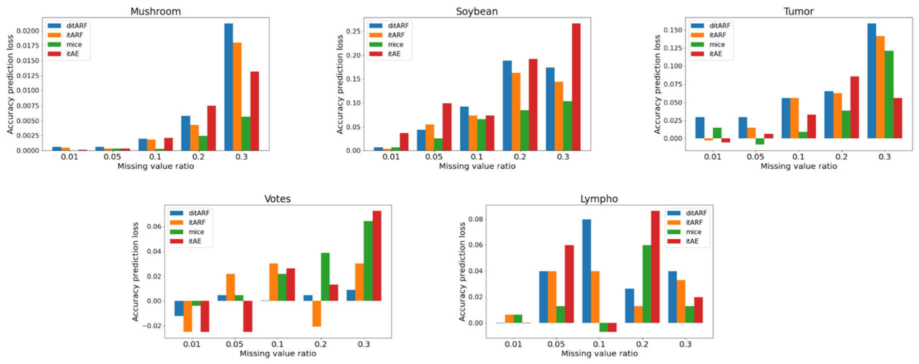

Fig. 7 Classification accuracy gain/loss when compared to a complete dataset (smaller value = better)

best results when the missing value ratio is small but fails when this ratio is big and thus there is not enough data to train a reliable model. In most cases, the itARF method along with its ditARF modification runs second best. We also observe that on the Nursery dataset, MICE fails to predict relevant values, while other methods demonstrate more optimistic results. The Friedman–Nemenyi diagrams demonstrate the statistical significance of the methods' performance difference in Fig. 5, confirming that three ARF-based methods lie in the high spectrum of methods ranking along with the MICE method.

The ditARF method computes the probabilities of imputed values $p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$ on every iteration and uses these to provide a measure of confidence per instance as sample weights of the model on the next iteration. To understand better its behavior, we illustrate probabilities of having a '1' class changing through iterations on Fig. 6. We observe that after several iterations each probability 'converges' to a certain level and continues oscillating around it. From this evidence, we conclude that the model is not overfitting (otherwise we would expect converging to 0 or 1) and indeed can provide a distribution for possible values for imputation.

Figure 7 shows the difference in classification accuracy of a Random Forest classifier learned on ground-truth complete data, and a Random Forest learned on datasets imputed by different methods. The analysis is performed for the datasets possessing a label to predict. First, we observe that imputation quality and further classification quality do not strictly correlate. This poses the question if the best strategy would be to do imputation and classification simultaneously to optimize the performance of both. Second, in some cases, the proposed ARF method facilitates classification compared to the MICE method even when imputation accuracy is lower. Third, we see in the Votes dataset that the ditARF method in some cases provides significantly better accuracy even when compared to the itARF method, which supports the further need of considering prediction confidence during the imputation step.

4.1.2 Imputation performance in high-dimensional datasets

We study the performance of the proposed methods and their competitors in high-dimensional settings, when $p > N$. The marginal imputation accuracy values are given in Table 3, where the datasets of interest are marked with an asterisk (*). Results for the MICE

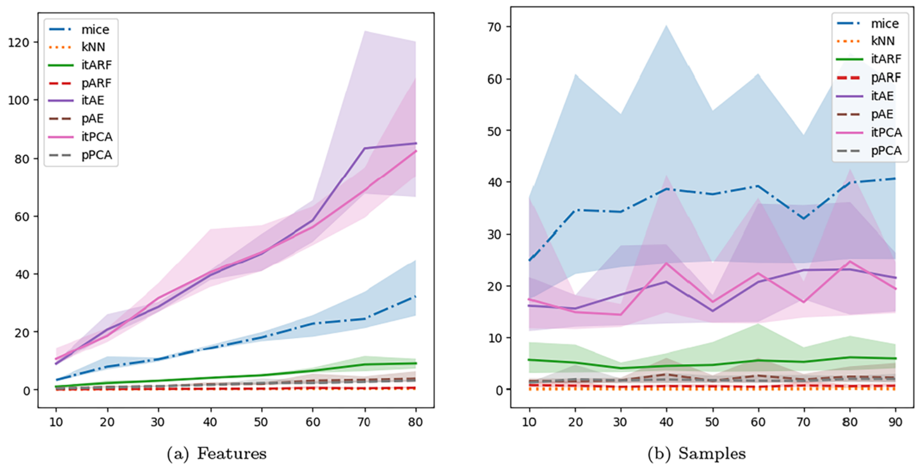

Fig. 8 Empirical results on time complexity (in seconds) for imputation methods. For each method, its average running time (across 5 launches) is shown (by line) as well as its minimum and maximum (borders of color interval). In **a** the number of features varies from 10 to 80 while the number of samples is constant, in **b** the number of samples varies from 10 to 90 while the number of features is constant. Here, ditARF is not specifically included since it is already covered by iterative ARF (itARF)

method could not be computed due to its excessive computation time. Similarly, procedural methods cannot be used as all instances are affected by missing values. We observe that the itARF and ditARF methods systematically outperform other iterative methods such as itAE and itPCA, as well as the kNN method, and thus prove to be a competitive and powerful alternative family of approaches.

Table 5 Computational time (in seconds) for the experiments with missing value ratio 0.01. Median times for 5 independent runnings are shown. All times are rounded to 3 dp

	mice	ditARF	itARF	pARF	itAE	pAE	itPCA	pPCA	kNN
Mushroom	60.040	20.554	15.237	1.228	185.009	10.682	279.053	10.466	0.878
Soybean	14.237	2.142	0.724	0.180	1.877	0.580	5.997	0.566	0.018
Tumor	6.438	1.303	0.425	0.100	1.140	0.523	1.188	0.613	0.011
Votes	5.346	0.397	0.255	0.072	1.243	0.251	4.192	0.265	0.007
Lympho	6.442	1.062	0.445	0.077	0.740	0.274	3.639	0.263	0.008
Survey	–	2755.776	2559.408	29.400	1011.730	16.980	1420.271	20.842	10.857
Nursery	24.176	18.928	7.033	0.813	113.971	13.104	124.335	18.703	0.901
Splice	386.047	71.791	70.706	3.661	27.653	6.814	140.628	6.767	0.558
Land Use	71.623	12.857	11.709	0.707	4.527	1.594	23.079	1.608	0.027
SNP Maize*	–	524.257	377.264	–	219.756	–	188.778	–	0.399
SNP Euc.*	–	1207.453	459.983	–	318.493	–	798.399	–	2.465
Yeast^{d}	1.656	1.078	0.608	0.091	1.159	1.045	11.868	0.980	0.011
Metro^{d}♦	852.686	–	219.643	166.201	2600.627	572.480	1368.618	564.626	–
Energy^{d}♦	216.515	–	91.264	74.137	1566.590	316.695	4364.438	251.719	–

4.1.3 Time complexity analysis

The complexity of one Decision Tree with binary features is $\mathcal{O}(pN\log N)$ with regard to the number of features p and the number of instances N. If all the trees in an Autoreplicative Random Forest are trained on all features, the total complexity of the forest remains the same. In the MICE method, a separate model is trained per feature, thus for one iteration, the complexity of the MICE method with Random Forest base estimator becomes quadratic $\mathcal{O}\left(p^2N\log N\right)$.

At the same time, with a multi-label Random Forest, the total complexity remains linear. Thus, both the methods itARF and pARF provide linear complexity with regard to the number of features, as the complexity of one forest is only multiplied by the number of iterations which typically is low as convergence is reached soon.

The complexity of both single- and multi-output Random Forests remains similar with regard to the number n of samples, i.e. $N\log N$.

These theoretic estimations are well supported in the simulation study, see Fig. 8. We empirically compare the time complexity of the imputation methods on subsets of the Eucalyptus dataset under the MCAR scenario with 10% missing values. The subsets are selected as (a) the first p_s features of the original dataset, $10 \le p_s \le 80$, and (b) the first N_s samples, $10 \le N_s \le 90$.

Further, we access actual computation times for all methods and present the results for the missing value ratio of 0.01 in Table 5. In all datasets, the procedural methods and kNN are very fast, but we have seen above that they do not always produce adequate results or simply can not be used if there are not enough complete instances for training. Also, across all datasets, the MICE method works several times slower and is not applicable when the number of features increases. At the same time, we observe that itAE and itPCA methods often also require significant time expense while producing not necessary high imputation accuracy.

5 Conclusions and future work

In this work, we propose a general framework for missing value imputation and we deeply analyze the literature on missing value imputation schemes. We identify that while there exist multi-output missing value imputation methods such as Autoencoders, this idea may be further applied to any multi-output machine learning methods but is yet not presented in the literature.

Developing this idea, we propose multi-output Autoreplicative Random Forests (ARFs) for accurate missing value imputation, in three different variants. First, we propose procedural ARF (pARF) that leverages the idea of Denoising Autoencoders for missing value imputation that only impute once the missing values. Second, we propose iterative ARF (itARF). The proposed itARF approach works as a deterministic iterative imputation method that not only obtains competitive results to the state-of-the-art methods but also drastically outperforms them in terms of computational time. We have shown that these approaches can provide significant improvements especially when there is a lack of complete instances in the case of high-dimensional data. Moreover, we focused on the necessity of providing a measure of uncertainty with respect to the imputed missing values, and we proposed the distributional itARF (ditARF) which works similarly to the EM algorithm and estimates the posterior distribution. With the probabilistic versions of the proposed framework, we provide not only imputation for the missing values but also a measure of uncertainty which we believe could be beneficial in numerous applications. Note that missing value imputation is commonly used in the preprocessing steps of broader machine-learning tasks. Hence, wrongly imputed values could significantly impact the forthcoming learning tasks. This could be avoided by only considering confident imputations.

To evaluate the proposed solution, we have performed an extensive evaluation of the proposed and previously existing methods on low- and high-dimensional datasets in which we included a variety of datasets from the UCI repository and three SNP datasets. As can be seen, the proposed solutions drastically outperform existing literature approaches when $p \gg N$. Finally, we have also tested the difference between training a Random Forest classifier for an imputed dataset and ground-truth data. The results show that the obtained accuracy with the classifier learned in ARF methods are good estimates since they obtain similar results to the classifier learned with ground-truth data.

Author Contributions E.A. developed the general framework including the proposed new ARF approach and conducted the experiments. A.C. developed the probabilistic version of the ARF method. All authors actively participated in the discussions and in writing the manuscript.

Funding Non Applicable.

Data availability The data that support the findings of this study were derived from the following resources available in the public domain: • UCI datasets: https://archive.ics.uci.edu/• Land Use dataset: https://doi.org/10.17632/GK3Z8GP7CP.2• Financial Survey dataset: https://www.consumerfinance.gov/data-research/financial-well-being-survey-data• SNP Maize: https://doi.org/10.15454/AEC4BN• SNP Eucalyptus: https://doi.org/10.6084/m9.figshare.8044370• SNP Wheat: https://doi.org/10.5447/ipk/2020/12

Code availablility All code for general missing value imputation framework, including newly proposed methods based on Autoreplicative Random Forests, is available at https://github.com/ekaantonenko/ARF.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Buuren, S., & Groothuis-Oudshoorn, K. (2011). MICE: Multivariate imputation by chained equations in r. *Journal of Statistical Software, 45*(3), 1–7.

CFPB. (2017). Financial well-being survey data. https://www.consumerfinance.gov/data-research/financial-well-being-survey-data/

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological), 39*(1), 1–38.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Elsevier*. https://doi.org/10.1016/b978-1-55860-377-6.50032-3

Dray, S., & Josse, J. (2015). Principal component analysis with missing values: A comparative survey of methods. *Plant Ecology, 216*, 657–667.

Dua, D., & Graff, C. (2017). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

Gerych, W., Hartvigsen, T., Buquicchio, L., Agu, E., & Rundensteiner, E. A. (2021). Recurrent Bayesian classifier chains for exact multi-label classification. *Advances in Neural Information Processing Systems, 34*, 15981–15992.

Grattapaglia, D. (2019). Quantitative genetic parameters for growth and wood properties in Eucalyptus urograndis - SNP marker data. figshare.

Guo, Y., & Gu, S. (2011). Multi-label classification using conditional dependency networks. *IJCAI Proceedings-International Joint Conference on Artificial Intelligence, 22*, 1300.

İrsoy, O., & Alpaydin, E. (2016). Autoencoder trees. Proceedings of Machine Learning ResearchIn G. Holmes & T.-Y. Liu (Eds.), *Asian Conference on Machine Learning* (Vol. 45, pp. 378–390). Hong Kong: PMLR.

Jafrasteh, B., Hernández-Lobato, D., Lubián-López, S. P., & Benavente-Fernández, I. (2023). Gaussian processes for missing value imputation. *Knowledge-Based Systems, 273*, 110603. https://doi.org/10.1016/j.knosys.2023.110603

Johnstone, I. M., & Titterington, D. M. (2009). Statistical challenges of high-dimensional data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367*(1906), 4237–4253.

Karimi, Arsalan. (2023). *Questionnaire data on land use change of Industrial Heritage: Insights from Decision-Makers in Shiraz*. Iran: Mendeley. https://doi.org/10.17632/GK3Z8GP7CP.2

Little, R. J., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data* (Vol. 793). Hoboken, USA: John Wiley & Sons.

MacKay, D. J. (2003). *Information Theory. Inference and Learning Algorithms.* Cambridge, Great Britain: Cambridge University Press.

McCoy, J. T., Kroon, S., & Auret, L. (2018). Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine, 51*(21), 141–146.

Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scalable model-based cascaded imputation of missing data. In: PAKDD 2018: 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 64–76.

Negro, S. S., Millet, E. J., Madur, D., Bauland, C., Combes, V., Welcker, C., Tardieu, F., Charcosset, A., & Nicolas, S. D. (2019). Genotyping-by-sequencing and SNP-arrays are complementary for detecting quantitative trait loci by tagging different haplotypes in association studies. *BMC Plant Biology, 19*(1), 1–22.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning, 85*(3), 333–359. https://doi.org/10.1007/s10994-011-5256-5

Reif, J. (2020). Genotyping information for diverse european bread wheat genotypes based on the ZUCHTWERT project. e!DAL - Plant Genomics and Phenomics Research Data Repository (PGP), IPK Gatersleben, Seeland OT Gatersleben, CorrensstraSSe 3, 06466, Germany.

Santos, M. S., Pereira, R. C., Costa, A. F., Soares, J. P., Santos, J., & Abreu, P. H. (2019). Generating synthetic missing data: A review by missing mechanism. *IEEE Access, 7*, 11651–11667.

Schwender, H. (2012). Imputing missing genotypes with Weighted k Nearest neighbors. *Journal of Toxicology and Environmental Health, Part A, 75*(8–10), 438–446.

Seo, B., Shin, J., Kim, T., & Youn, B. D. (2022). Missing data imputation using an iterative denoising autoencoder (IDAE) for dissolved gas analysis. *Electric Power Systems Research, 212*, 108642. https://doi.org/10.1016/j.epsr.2022.108642

Stekhoven, D. J., & Buhlmann, P. (2011). MissForest-non-parametric missing value imputation for mixed-type data. *Bioinformatics, 28*(1), 112–118.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., & Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics, 17*(6), 520–525.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification. *International Journal of Data Warehousing and Mining, 3*(3), 1–13. https://doi.org/10.4018/jdwm.2007070101

Tsoumakas, G., & Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. *Machine Learning: ECML, 2007*, 406–417.

Van Wolputte, E., & Blockeel, H. (2020). Missing value imputation with mercs: a faster alternative to missforest. In: Discovery Science: 23rd International Conference, DS 2020, Thessaloniki, Greece, October 19–21, 2020, Proceedings, pp. 502–516.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning - ICML ’08. https://doi.org/10.1145/1390156.1390294

Wood, D., Mu, T., Webb, A., Reeve, H., Lujan, M., & Brown, G. (2023). A unified theory of diversity in ensemble learning. *Journal of Machine Learning Research, 24*(359), 1–49.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming, 151*(1), 3–34. https://doi.org/10.1007/s10107-015-0892-3

Zhang, M.-L., & Zhou, Z.-H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition, 40*(7), 2038–2048.