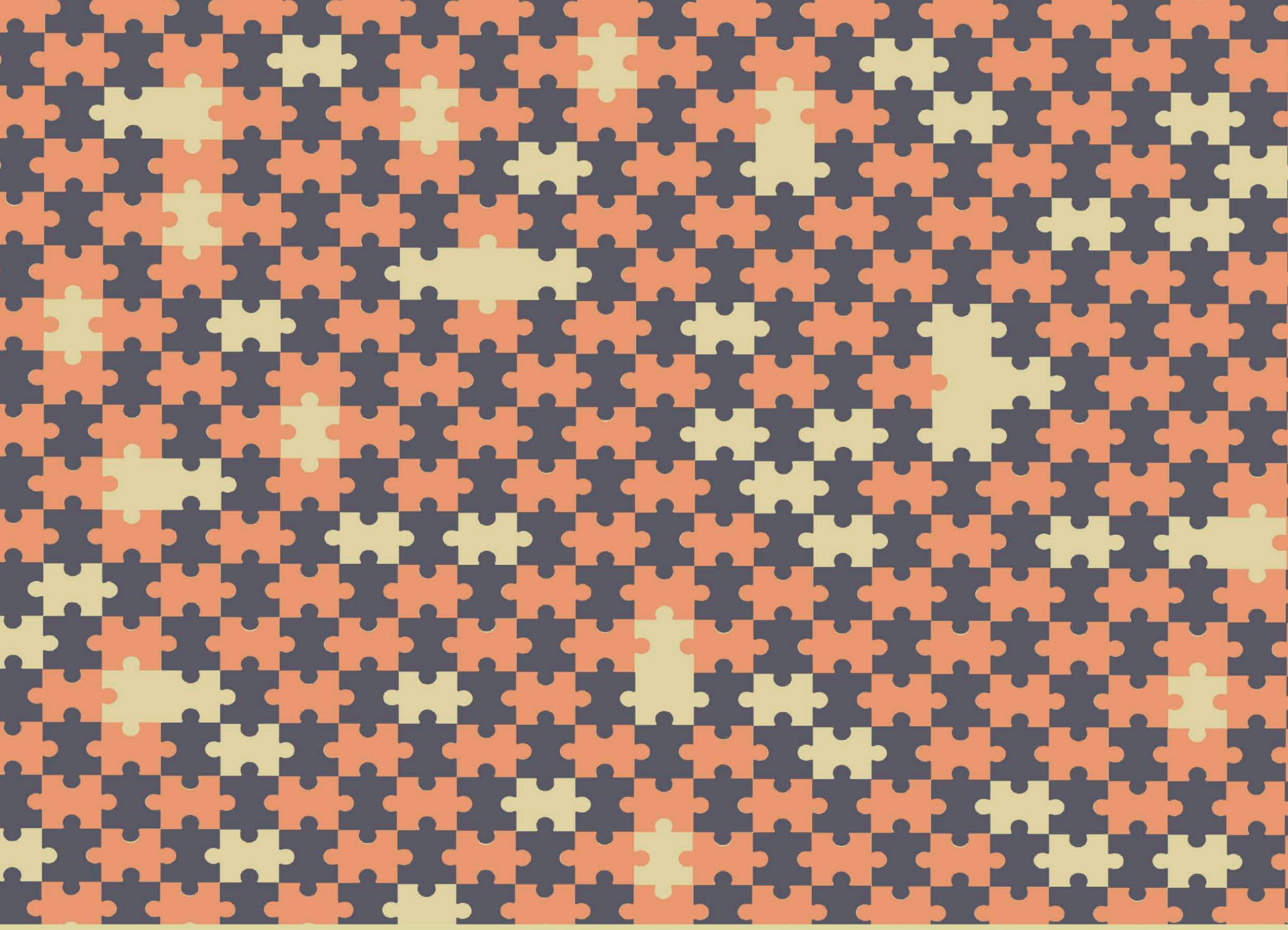


Adriana Fonseca Costa

A study on missing data: handling missingness using Denoising Autoencoders

Thesis submitted to the University of Coimbra
for the degree of Master in Biomedical Engineering

2018

· U C ·

UNIVERSIDADE DE COIMBRA

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Adriana Fonseca Costa

A study on missing data: handling missingness using Denoising Autoencoders

Thesis submitted to the
University of Coimbra for the degree of
Master in Biomedical Engineering

Supervisors:
Pedro Henriques Abreu (PhD)
Miriam Seoane Santos (MSc)

Coimbra, 2018

Este trabalho foi desenvolvido em colaboração com:
This work was developped in collaboration with:

Centre for Informatics and Systems of the University of Coimbra

Agradecimentos

If I have seen further than others, it is by standing upon the shoulders of giants.
Isaac Newton

Agradeço ao Professor Doutor Pedro Henriques Abreu, pelo incentivo constante e inestimável e pela confiança demonstrada. Graças às suas reconhecidas competências científicas e académicas, colocou-me perante desafios vários e estimulou o meu interesse pelas temáticas abordadas, oferecendo-me uma oportunidade ímpar de enriquecimento académico e pessoal.

Exprimo a minha gratidão à Miriam Seoane Santos, aluna de Doutoramento, pelo aconselhamento desinteressado e pela disponibilidade manifestada, afirmando-se como uma verdadeira âncora no amadurecimento e construção deste projeto.

Deixo um agradecimento à minha família, especialmente aos meus pais, por terem percorrido, sempre, mesmo sempre, esta caminhada a meu lado, abraçando um sonho que, apesar de meu, tomaram como seu.

Finalmente, agradeço aos meus colegas que me mostraram, na senda de Antoine de Saint-Exupéry, que "aqueles que passam por nós não vão sós, não nos deixam sós". Ao invés, "deixam um pouco de si, levam um pouco de nós".

Resumo

Com a evolução tecnológica, verificou-se um aumento exponencial da quantidade de dados recolhidos e armazenados. Assim, surgiu a necessidade de criar mecanismos automáticos para extrair conhecimento dos referidos dados. Estes mecanismos automáticos, conhecidos por modelos de aprendizagem automática, foram, na sua maioria, desenvolvidos para dados completos, requisito que nem sempre é possível cumprir. Neste contexto, a imputação dos dados (substituição dos valores em falta por estimativas plausíveis) surge como uma possível solução, garantindo a qualidade dos dados para posterior análise.

Nos últimos anos, vários estudos têm proposto novas técnicas de imputação, de entre as quais se destaca a utilização de *Stacked Denoising Autoencoders.* Dada a sua extraordinária capacidade de recuperar dados corrompidos, os *Stacked Denoising Autoencoders* mostram-se promissores na área da imputação de dados, tendo despertado um interesse crescente por parte da comunidade científica.

No entanto, sendo um tópico recente, a sua aplicação ainda não se encontra suficientemente bem estudada, apresentando diversos aspetos por explorar; em particular, a sua adequação a diferentes mecanismos de dados em falta (*Missing Completely At Random*, *Missing At Random* e *Missing Not At Random*).

Esta tese apresenta um estudo aprofundado da imputação de dados via *Stacked Denoising Autoencoders*, considerando diferentes mecanismos e percentagens de dados em falta. Em comparação com métodos de imputação do estado da arte, os *Stacked Denoising Autoencoders* mostraram ser abordagens robustas para a imputação de elevadas percentagens de dados em falta, especialmente quando o mecanismo subjacente à sua geração é *Missing Not At Random.*

Palavras-Chave: mecanismos de dados em falta, preenchimento de dados em falta, *denoising autoencoders*

Abstract

The evolution of technology led to an exponential increase in the amount of data being collected and stored, thus creating the need to develop automatic mechanisms to extract knowledge from data. These automatic mechanisms, known as Machine Learning techniques, were mostly designed for complete data, a requirement that is not always fulfilled. In this context, data imputation (replacement of missing values by plausible estimates) arises as a possible solution, ensuring the quality of data for later analysis.

Over the years, several studies presented alternative imputation strategies, among which Stacked Denoising Autoencoders stand out. Given their ability to recover corrupted data, Stacked Denoising Autoencoders are promising in the area of data imputation, generating great interest in the scientific community. However, their application is an understudied topic, still presenting challenging aspects for research; namely, their suitability for different missing data mechanisms (Missing Completely At Random, Missing At Random and Missing Not At Random).

This thesis presents a thorough study of data imputation via Stacked Denoising Autoencoders, considering different missing data mechanisms and missing rates. In comparison to state-of-the-art imputation methods, Stacked Denoising Autoencoders proved to be robust for imputing high missing rates, especially, when the mechanism underlying their generation is Missing Not At Random.

Keywords: missing data mechanisms, missing data imputation, denoising autoencoders

List of Figures

List of Tables

List of Algorithms

Contents

Acronyms

AE Autoencoder. xi, 26, 27, 31, 41

ANN Artificial Neural Networks. 31, 32

ARIMA Autoregressive Integrated Moving-Average. 32

BP Backpropagation. 32

DAE Denoising Autoencoder. xi, 26, 27

DNA Deoxyribonucleic Acid. 3

EM Expectation-Maximization. 39–41, 50, 51

GBWkNN a weighted k-Nearest Neighbours data filling algorithm based on Grey correlation analysis. 32

HCC Hepatocellular Carcinoma. 37, 75

kNN k-Nearest Neighbours. 31–33

kNNimp imputation with k-Nearest Neighbours. 39–41, 50, 51

LLN Law of Large Numbers. 10

MAE Mean Absolute Error. 31, 34

MAR Missing At Random. 1, 4, 6, 7, 11, 16, 20, 25, 31, 39, 40, 48–52, 60

MCAR Missing Completely At Random. xi, 1, 4, 6–10, 16–18, 29–32, 39, 40, 48–52, 58, 60

MD Missing Data. 10–25, 38–40, 42, 47, 54, 58

Meanimp Mean Imputation. 39–41, 50, 51, 60

MICE Multiple Imputation by Chained Equations. xiv, 29–31, 39–41, 51, 52, 54, 57, 58, 88

MIV Missingness depending on Its Value itself. 23

MkNNI Mutual k-Nearest Neighbours Imputation. 32

MNAR Missing Not At Random. 1, 4, 6, 7, 14, 16, 22, 23, 25, 29–31, 39, 48–52, 60

MR Missing Rate. 4, 9–25, 31, 38–40, 47, 50, 52, 54, 57, 58, 60, 74

MRE Mean Relative Error. 31

MSE Mean Squared Error. 30, 34

MuOV Missingness depending on unobserved Variables. 23

ReLu Rectified Linear Units. 30

RF Random Forest. 31

RMSE Root Mean Squared Error. xiii, 31, 32, 34, 35, 39, 44, 51, 52, 54, 57, 58, 60, 62

RMSE$_{sum}$ Sum of Root Mean Squared Error. 30

SDAE Stacked Denoising Autoencoders. xi, 29–36, 38, 41–43, 47, 48, 50–52, 54, 56–58, 60, 62

SGD Stochastic Gradient Descent. 42, 43

SVD Singular Value Decomposition. 31

SVM Support Vector Machines. 33

SVMimp imputation with Support Vector Machines. xiv, 39–41, 49–51, 54, 56–58, 60, 62, 88

Tanh Hyperbolic Tangent. 30, 42, 43

1

Introduction

Over the years, the evolution of information technology and the improvement in terms of computational power led to an increase in the amount of available data – as datasets became larger and more complex, it was no longer possible to interpret them manually. This created a need to extract knowledge from datasets using automated techniques.

In the late 80's, Gregory Piatetsky-Shapiro proposed the term Knowledge Discovery in Databases (KDD) for the title of a workshop held at the *Internacional Joint Conference on Artificial Intelligence* [1]. Some years later, Fayyad et al. defined KDD as the "overall process of discovering useful knowledge from data" [2]. This interactive and iterative process is composed by 5 main steps (Figure 1.1): Selection, Preprocessing, Transformation, Data Mining and Interpretation/Evaluation.

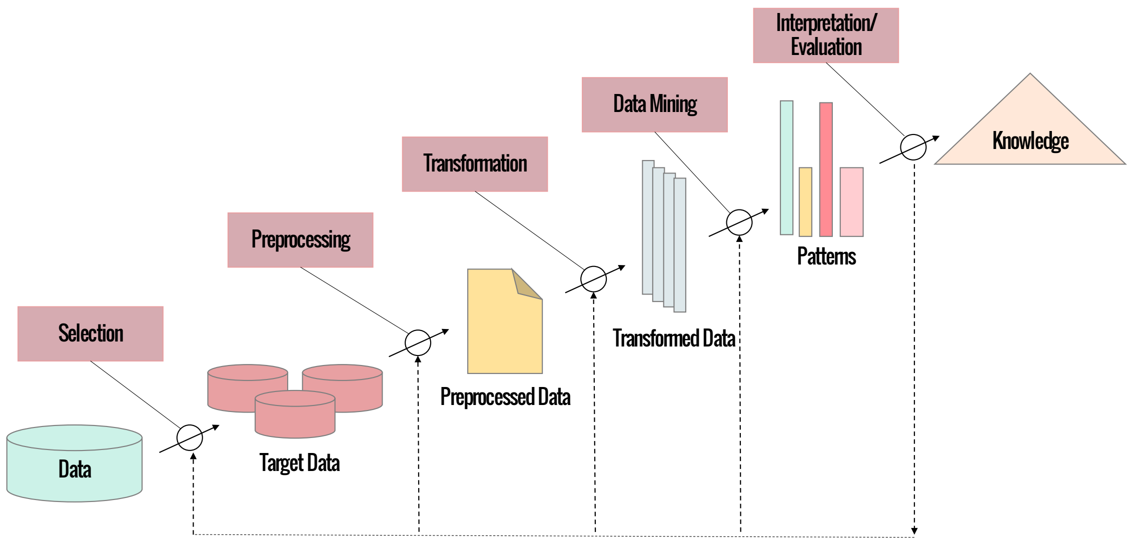


Figure 1.1: Knowledge Discovery in Databases process. Adapted from Fayyad et al. [2].

The Selection step encompasses the collection of data from which the knowledge will be extracted. The next step is the Preprocessing which consists essentially of data cleaning operations - removal of noise or outliers, handling missing data, among others. In the Transformation step, dimensionality reduction techniques or transformation methods are applied in order to obtain useful representations of the data. In the Data Mining step, the aim is to search for patterns in data, selecting methods according to the type of problem to be solved (e.g., classification, clustering, regression). The last step consists on interpreting and validating the model obtained from the previous step.

The work presented in this thesis focuses on the preprocessing stage, where several issues may arise. In particular, we focus on the problem of missing data, the lack of information in one or several features in a dataset.

1.1 Context

Missing Data (MD) is a common problem that appears in real-world datasets, and may compromise the performance of learning models [3, 4]. In the research community, three missing mechanisms are recognised: Missing Completely At Random (MCAR), Missing At Random (MAR) and Missing Not At Random (MNAR) (a detailed description is provided in Chapter 2). Furthermore, an incomplete dataset may have MD only in one feature – univariate MD (here denoted as *univa*) – or in several features – multivariate MD (here denoted as *unifo*) .

In the literature, there are several ways of handle missing data, as shown in Figure 1.2: Case deletion, Imputation Methods, Maximum Likelihood and Machine Learning without MD estimation.

Each of these approaches presents their advantages and limitations, however, the one most used in the literature is data imputation [6]. Imputation methods aim to find plausible values to replace the missing ones, and are mainly divided into statistical-based and machine learning-based methods [5]. Statistical methods consist in replacing the missing observations with the most similar ones among the training data, without the need of constructing a predictive model to evaluate their "similarity" (e.g. Mean/Mode imputation). Machine learning-based techniques, construct a predictive model with the available data to estimate values for replacing those that are missing (e.g. k-Nearest Neighbours imputation).

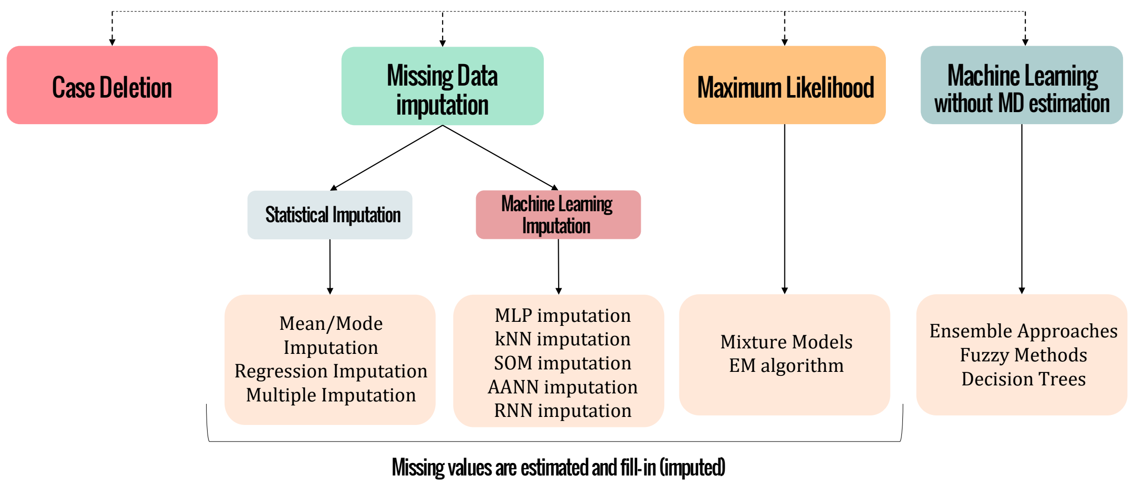


Figure 1.2: Methods to handle missing values. Adapted from García-Laencina et. al. [5].

Deep learning techniques are currently a hot topic in Machine Learning since they have proved to find elegant solutions for several classic problems [7].

Stacked Denoising Autoencoders (SDAE) are a special type of deep neural networks, developed to recover a clean output from a corrupted input. Since the presence of missing values in a dataset constitutes a type of corruption, it seems that a natural extension is the use of SDAE in the imputation task. Although the subject of imputation has been previously discussed in the literature [19, 5, 23], the application of SDAE–based approaches for imputation purposes remains an understudied topic.

1.2 Goals

The main goal of this work is to study the performance of SDAE-based approaches for imputation purposes, analysing whether this technique constitutes a good imputation approach, when compared to well-established approaches (Mean Imputation, kNN imputation, Support Vector Machines imputation, Multiple Imputation by Chained Equations and Expectation-Maximization).

To achieve this goal we performed three main experiments in order to answer the three following questions (a full description of such experiments is presented in Chapter 4):

1. How do the SDAE perform when there are enough complete data to train the model (missing values only in the test set)?

    - We selected 20 complete datasets, from open source repositories, and performed synthetic MD generation using 9 *univa* implementations, under 5 and 20% of Missing Rate (MR). Then, we compared the performance of our proposed SDAE approach with 7 state-of-the-art imputation methods.

2. How do the SDAE perform when training data corruption follows an underlying missing mechanism?

    - We selected 33 complete datasets and performed synthetic MD generation using 3 *univa* implementations and 3 *unifo* implementations, under 5, 10, 15, 20 and 40% of MR. Then, we compared the performance of this approach for two different SDAE with 7 state-of-the-art imputation methods.

3. Does the performance of SDAE increase for larger datasets, with higher sample sizes?

    - We investigated the usefulness of SDAE when handling larger datasets. For that, 5 complete datasets with higher sample sizes were selected and the simulation setup performed for question 2 was repeated.

1.3 Research Contributions

The work developed during this thesis resulted in the following contributions:

- Adriana F. Costa, Miriam S. Santos, Jastin Soares, Pedro H. Abreu. "Missing Data Imputation Using Deep Denoising Autoencoders: data recoverability in different missing scenarios". IJCAI-ECAI 2018, 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (Submitted on 31th January 2018).

- Adriana F. Costa, Miriam S. Santos, Jastin Soares, Pedro H. Abreu. "Missing Data Imputation Using Denoising Autoencoders: the untold story". IDA 2018 - 17th International Symposium on Intelligent Data Analysis (Accepted as full paper).

- Miriam S. Santos, Adriana F. Costa, Jastin Soares, Pedro H. Abreu, "Synthetic Missing Data Generation: The Nuts and Bolts" (in preparation, to be submitted to Pattern Recognition).
- Adriana F. Costa, Miriam S. Santos, Inês Domingues, Pedro H. Abreu, "Stacked Denoising Autoencoders for Missing Data Imputation: exploring the effects of sample size, missing mechanisms and rates on performance classification" (in preparation, to be submitted to Expert Systems with Applications).

1.4 Document Structure

The remainder of this document is organised as follows: in Chapter 2 we provide some useful background knowledge which will be the basis of this work. Chapter 3 presents several research works from the missing data field that use deep learning approaches for imputation purposes. Then, Chapter 4 describes the different stages of the experimental setup and Chapter 5 discusses the obtained results. Finally, Chapter 6 concludes the thesis and presents some possibilities for future work.

2

Background Knowledge

Missing data corresponds to the lack of information in a dataset, so it occurs when no data is available for an observation [14]. This is a common shortcoming that arises in several real-world domains. For example, in UCI Machine Learning Repository – which provides a collection of datasets widely used for the empirical analysis of machine learning algorithms – about 45% of datasets have missing data [15]. Handling missing data is an important issue since, in addition of being a very frequent problem, the presence of missing values affects the results obtained on classification.

There are a lot of plausible explanations for the occurrence of missing values, such as equipment errors, erroneous data registration, non-response in surveys, among others [16]. For example, in the field of medical diagnosis, some values may not be saved due to a deficient manual data registration or to the unavailability of certain medical equipments. In genetic research, there are also a lot of incomplete data when handling Deoxyribonucleic Acid (DNA) microarrays, where this data may be missing due to several reasons including sample contamination. In control based-applications, missing data can also be a result of equipment errors and incorrect measures - a faulty equipment may not be able to record certain observations because of their fault or because of the value of the observation itself.

The occurrence of missing values is more easily explained using as example the process of responding to a survey. In this case, respondents can accidentally skip questions or can skip a question intentionally, due to its content. More, questions can be skipped depending on the age of the respondent, in other words, a younger person may skip a question to which an older one would respond easily. For example, an adult may find it easier to answer a question of political nature than a teenager. These three examples follow different missing mechanisms, which will be explained in what follows.

2.1 Basic Notations

The missing data theory was introduced by Rubin [17] in the late 70s. To provide a clear explanation about this theory it is important to establish some basic notation. Let **X** denote a dataset with $m \in \mathbb{N}$ observations and $n \in \mathbb{N}$ features. Each element of the dataset is defined by X_{ij}, where $i = \{1,2,...,m\}$ and $j = \{1,2,...,n\}$.

A complete dataset, X_{com}, consists of the data with no missing values. However, most real-world datasets have missing values. Suppose that X is divided into two parts such that $X = \{X_{obs}, X_{miss}\}$ where X_{obs} will be the set of observed values of X and X_{miss} will be the set with missing values of X. Each feature of X will be denoted as x_{miss} if it is composed by some observations of the missing part of the dataset and x_{obs} if it is a part of the observed values in the dataset. For example, the dataset in Table 2.1 is composed by two features, Age and Number of Cigarettes: Age can also be denoted as x_{obs}, since their observations are all observed in the dataset, while Number of Cigarettes can be called x_{miss} since it contains missing values. Furthermore, observed values of both Age and Number of Cigarettes constitute X_{obs} whereas, in this case, X_{miss} consists only on the missing values of one of the features, Age. Both parts, X_{obs} and X_{miss} can consist of more than one feature, depending on the number of features in the dataset.

Let R be the missing data indicator matrix that has the same dimensions as X:

$$R_{ij} = \begin{cases} 1 \text{ if } X_{ij} \text{ is missing} \\ 0 \text{ if } X_{ij} \text{ is obseved} \end{cases} \tag{2.1}$$

To illustrate this notation, reconsider the small dataset in Table 2.1. The missing data indicator matrix has the same dimensions as the simulated dataset and takes value 1 when data is missing and value 0 when data is observed.

A dataset X could have different percentages of missing data which are referred to Missing Rates (MRs).

2.2 Missing Data Mechanisms

There are three mechanisms under which missing data can occur: Missing Completely At Random (MCAR), Missing At Random (MAR) and Missing Not At Random (MNAR). The mechanism under the missing data can be characterized by

Table 2.1: A study in adolescent tobacco use containing missing values.

(a) Simulated dataset with missing values.

Age	Number of Cigarettes
15	2
15	-
15	-
16	2
16	2
16	4
16	3
17	-
17	6
17	-
17	5
17	5
18	-
18	6
18	-
19	3
19	-
19	-
20	9
20	2

(b) Missing data indicator matrix for this dataset.

Age	Number of Cigarettes
0	0
0	1
0	1
0	0
0	0
0	0
0	0
0	1
0	0
0	1
0	0
0	0
0	1
0	0
0	1
0	0
0	1
0	1
0	0
0	0

the conditional distribution of R given X:

$$p(R|X,\xi) = p(R|X_{obs}, X_{miss}, \xi) \tag{2.2}$$

where p is the probability distribution, R is the missing data indicator matrix, X_{obs} and X_{miss} are the sets of observed and missing data, respectively, and ξ designates the set of parameters that describes the relation between R and the dataset, X. The set of parameters, ξ is normally unknown [18]. However, this is not a problem since it is the existence or absence of a relationship between X and R that defines the missing mechanisms.

To illustrate the three missing mechanisms, consider the small dataset in Table 2.2 which corresponds to a simulated dataset of a study in adolescent tobacco, with 20 participants. It is assumed that this data was collected via a survey made to students of high school level. In this study, several students have missing values on their daily number of cigarettes.

Table 2.2: Simulated dataset of a study in adolescent tobacco use, with N = 20 participants. The daily average of cigarettes is presented with MCAR, MAR and MNAR missing values.

Age	Number of Cigarettes			
	Complete	MCAR	MAR	MNAR
15	2	2	-	2
15	3	-	-	3
15	4	-	-	4
16	2	2	-	2
16	2	2	-	2
16	4	4	-	4
16	3	3	-	3
17	9	-	9	-
17	6	6	6	-
17	4	-	4	4
17	5	5	5	5
17	5	5	5	5
18	7	-	7	-
18	6	6	6	-
18	7	-	7	-
19	3	3	3	3
19	8	-	8	-
19	3	-	3	3
20	9	9	9	-
20	2	2	2	2

Missing Completely At Random

The Missing Completely At Random mechanism (MCAR) occurs when the mechanism under the missingness is unrelated to any observed or unobserved value from the dataset, so:

$$p(R = 1|X_{obs}, X_{miss}, \xi) = p(R = 1|\xi) \tag{2.3}$$

Equation 2.3 shows that the probability that R takes 1 as value (i.e., there is a missing value) is conditioned by some parameter (or set of parameters) ξ, but it is not conditioned by any data, whether it may be X_{miss} or X_{obs}.

In the MCAR column, the missing values for the daily number of cigarettes are unrelated with their own values and also with the Age, so missingness is unrelated to the data. One possible explanation for MCAR values is that students did not go to school to participate in the survey, because of some personal event (e.g. a funeral, a car accident, an illness) which is not part of the dataset.

Missing At Random

In this case, the cause of the missing data is related with observed values from the dataset. This mechanism can be expressed by the relation:

$$p(R = 1|X_{obs}, X_{miss}, \xi) = p(R = 1|X_{obs}, \xi) \tag{2.4}$$

Equation 2.4 describes the probability of a missing value occurring under MAR mechanism. This probability is conditioned by the observed data, X_{obs}, through some parameter (or set of parameters), ξ.

For the MAR example, the number of daily cigarettes is missing for younger students (aged between 15 and 16). For example, younger students are less likely to fill in their number of smoked cigarettes in a day because they do not want to admit that they are regular smokers. However, this lack of values has nothing to do with the daily number of cigarettes reported by a student – it is only related to their age.

Missing Not At Random

Finally, data is Missing Not At Random (MNAR) when the probability of a value to be missing is related with the missing data itself. This mechanism can be expressed by the expression:

$$p(R = 1|X_{obs}, X_{miss}, \xi) \tag{2.5}$$

Equation 2.5 shows that the probability of missing a value depends on the set of missing data, X_{miss}, but also on the set of observed data, X_{obs}. So, this probability varies in a way that is unknown to us, which makes this mechanism more complex.

Considering again the dataset in Table 2.2, we can see that MNAR values are directly related with the missing values. If students smoke a lot they are more likely to hide their number of daily cigarettes. So, the probability of a missing value depends on the value itself - the probability of a missing value is higher for students who smoke frequently and may also be related to their age.

2.2.1 Synthetic Missing Data Generation Methods

The three mechanisms can be synthetically generated in various ways, and consequently different implementations can be found in the literature. Tipically, the studies on missing data imputation follow a specific pipeline:

1. Selection of complete datasets, that will constitute the ground truth; these datasets can have different characteristics (e.g., dimensionality, sample size, types of features)
2. Synthetic generation of missing values, that can encompass the different missing mechanisms and univariate (*univa*) and multivariate (*unifo*) approaches;
3. Data imputation using several strategies, either statistic-based, or machine learning-based techniques;
4. Evaluation of imputation methods in terms of imputation quality and/or classification performance.

In this subsection, we focus on step 2 and will present several *univa* and *unifo* Missing Data (MD) generation approaches (based on related works) and the respective pseudo-codes. These pseudo-codes use several functions as building blocks which are described in Table A.1. The nomenclature we use represents the missing values by "NaN" values. Table 2.3 shows all the missing data generation approaches that were implemented and a brief description. We implemented all the synthetic generation approaches using *Python 3*.

In addition to the pseudo-codes, we also present illustrative schemes for some approaches. In all of them, we use grey observations to represent the location of the missing values. Moreover, we represent the observed values that are relevant in a particular approach using different shades of green: darker shades are used to represent higher values while lighter shades represent lower values.

2.2.1.1 Univariate implementations

For the univariate implementations, the feature that will have MD, x_{miss}, will always be the one most correlated with the class labels. This correlation is accessed through the calculation of all the correlation coefficients between each feature and the class label. It is important to emphasize that we have decide to choose x_{miss} in this way but this approach may not match the one used on related works.

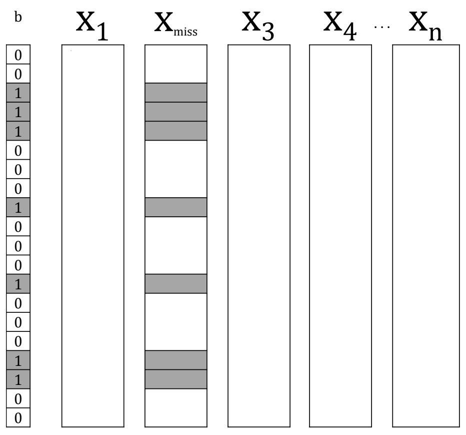
Table 2.3: Summary of the different missing data configurations found in the related work.

Univariate			**Multivariate**		
Designation	**Authors**	**Description**	**Designation**	**Authors**	**Description**
$MCAR1_{univa}$	Twala et al. [19]	Based on the values of a Bernoulli distribution.	$MCAR1_{unifo}$	Twala et al. [19]	Based on the values of a Bernoulli distribution.
$MCAR2_{univa}$	Rieger et al. [20]; Xia et al. [21]	Random observations of x_{miss} are set to be missing.	$MCAR2_{unifo}$	Nanni et al. [22]	For each observation, a random percentage of features is set to be missing.
			$MCAR3_{unifo}$	Garciarena et al. [23]; Ali et al. [24]; Zhu et al. [25]	Random observations are set to be missing, considering all the dataset.
$MAR1_{univa}$	Twala et al. [19]	Observations bellow the MR percentile of x_{obs} are set to be missing on x_{miss}.	$MAR1_{unifo}$	Twala et al. [19]	Observations bellow the MR percentile of x_{obs} are set to be missing on x_{miss}.
$MAR2_{univa}$		Based on ranks of x_{obs}.	$MAR2_{unifo}$	Zhu et al. [26]; Ali et al. [24]	Based on the median of x_{obs}.
$MAR3_{univa}$	Rieger et al. [20]	Based on the median of x_{obs}.	$MAR3_{unifo}$	Garciarena et al. [23]	There is only one observed feature. Each x_{miss} will have missing values for the lowest observations of x_{obs}.
$MAR4_{univa}$		x_{miss} will have missing values depending on the highest values of x_{obs}.			
$MAR5_{univa}$		x_{miss} will have missing values depending on the lowest and the highest values of x_{obs}.			
$MNAR1_{univa}$	Twala et al. [19]	Lowest values of x_{miss} are deleted.	$MNAR1_{unifo}$	Zhu et al [26].; Ali et al. [24]	Based on the median of x_{miss}.
$MNAR2_{univa}$	Xia et al. [21]	Highest values of x_{miss} are deleted.	$MNAR2_{unifo}$	Garciarena et al. [23]	All the features will have missing values for the same observations.
			$MNAR3_{unifo}$	Garciarena et al. [23]	Lower values of each x_{miss} are set to be missing.
			$MNAR4_{unifo}$	Twala et al. [19]	Features are separated into pairs. Each has one x_{miss}. Lower values of each x_{miss} are set to be missing.

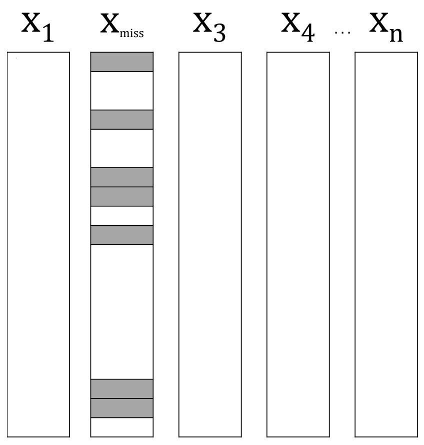
Choosing x_{miss} to be the most correlated feature with the class labels ensures a generic pattern among all the datasets used. On the other hand, it is also safeguarded that the values that are set to be missing are relevant information. However, other approaches can be found in the literature such as choosing x_{miss} randomly as in Xia et al. [21] and Rieger et al. [20].

Univariate MCAR implementations

For MCAR mechanism, we consider two univariate implementations, $MCAR1_{univa}$ (Figure 2.1a and Algorithm 1) and $MCAR2_{univa}$ (Figure 2.1b and Algorithm 2), based on three related works (Twala et al. [19], Rieger et al. [20], Xia et al. [21]).



(a) Missing data pattern for $MCAR1_{univa}$ implementation. b represents the Bernoulli distribution.



(b) Missing data pattern for $MCAR2_{univa}$ implementation.

Figure 2.1: Schemes describing missing data patterns for each MCAR implementation. In (a), the missingness is defined by a Bernoulli distribution.

Twala et al. [19] performed MCAR generation by choosing the locations of x_{miss} using a Bernoulli distribution – it is referred to as $MCAR1_{univa}$. The Bernoulli distribution is a discrete distribution that has outcome $k = 1$ with a probability p and outcome $k = 0$ with a probability of $1 - p$, as shown in Equation 2.6. The probability p represents the expected missing rate and, as in any Bernoulli trial, each value of x_{miss} has a probability p of being deleted.

$$f(k,p) = \begin{cases} 1 - p \text{ for } k = 0 \\ p \text{ for } k = 1 \end{cases} \tag{2.6}$$

This implementation is not the most accurate as it may not return the expected missing rate, especially when it comes to small datasets. Of course, in any of the implementations, we may have to round the number of Missing Values (MVs) that we want to generate. For example, consider a dataset with 30 observations and 5 features in which we want to generate 15% of MVs that will be equally distributed by all features – each feature should have 4.5 ($30 \times 15\%$) missing values but in reality, it will have 5, which corresponds to a MR of, approximately, 16.7%. However, there are some implementations that, by their nature, can generate a very different number of MVs that was intended and hence they are limiting.

According to the Law of Large Numbers (LLN), the expected probability is more easily achieved for datasets with many samples. The increase in the number of observations makes the number of outcomes equal to 1 ($k = 1$) converge to the expected probability.

Algorithm 1: Implementation of $MCAR1_{univa}$.

```
Input   :
          data: Complete dataset
          MR: MD percentage
          $j_{miss}$: Missing feature index
Output:
          Dataset with MR% generated MD
begin
    x = numObservations(data)
    observations = bernoulli(MR, size = x)
    data[ (observations == 1), $j_{miss}$ ] = “$NaN$”
return (data)
```

Rieger et al. [20] and Xia et al. [21] proposed a simple method for generating MCAR which chooses random locations in x_{miss} to be missing. We choose this random locations with a random number generator function. This implementation method is the most immediate for a random mechanism and is herein referred to as $MCAR2_{univa}$.

Algorithm 2: Implementation of $MCAR2_{univa}$.

```
Input   :
          data: Complete dataset
          MR: MD percentage
          $j_{miss}$: Missing feature index
Output:
          Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    observations = random(x, size = numMV)
    data[ observations, $j_{miss}$ ] = “$NaN$”
return(data)
```

Univariate MAR implementations

For MAR mechanism we consider five different implementations: $MAR1_{univa}$ (Figure 2.2 and Algorithm 3), $MAR2_{univa}$ (Algorithm 4), $MAR3_{univa}$ (Algorithm 5), $MAR4_{univa}$ (Figure 2.3 and Algorithm 6) and $MAR5_{univa}$ (Figure 2.4 and Algorithm 7). All MAR generation methods make use of an observed feature x_{obs} to define the missing locations in x_{miss}: again, the missing feature x_{miss} will be the one most correlated with the class labels and the observed feature, x_{obs}, is the one most correlated with the missing feature, x_{miss}. As mentioned previously, this choice of features was made in order to coherently compare the obtained results on all the used datasets. Once again, this approach may not coincide with those used in related works.

Twala et al. [19] implemented a MAR generation algorithm, referred to as $MAR1_{univa}$, where x_{miss} will be missing for the observations that are below the MR percentile in the observed feature x_{obs}. This means that the lowest observations of x_{obs} will be deleted on x_{miss}.

Algorithm 3: Implementation of $MAR1_{univa}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $x_{obs}$: Observed feature
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    observations = sort($x_{obs}$, reverse = False)
    observations = observations[ 0 : numMV ]
    data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

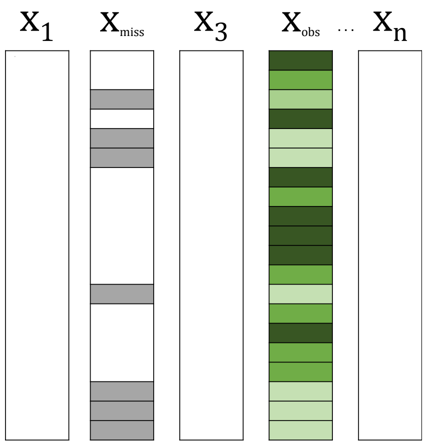


Figure 2.2: Missing data pattern for $MAR1_{univa}$ implementation.

In the work of Rieger et al. [20], 4 different MAR generations are suggested, which we have implemented as well in this work ($MAR2_{univa}$, $MAR3_{univa}$, $MAR4_{univa}$ and $MAR5_{univa}$):

$MAR2_{univa}$ is based on ranks of x_{obs} (r_{obs}): the probability of a pattern $x_{i,miss}$ to be missing is computed by dividing the rank of $x_{i,obs}$ by the sum of all ranks for x_{obs}. Then, the patterns to have missing values are sampled according to such probability, until the desired MR is reached (this method is also used by Xia et al. [21]);

$$p_{miss} = \frac{r_{i,obs}}{\sum_{i=1}^{m} r_{i,obs}} \tag{2.7}$$

Algorithm 4: Implementation of $MAR2_{univa}$.

```
Input  :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $x_{obs}$: Observed feature
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    ranks = rank($x_{obs}$)
    probs = ranks / sum(ranks)
    observations = random(x, size = numMV, p = probs)
    data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

In $MAR3_{univa}$, the patterns are divided into two groups according to the median of the observed feature x_{obs}, so that the probability of missingness is different among groups: patterns with observations greater than (or equal to) the median of x_{obs} will belong to Group 1, otherwise the observations belong to Group 2. In Algorithm 5, *groups* is an array of size x that contains the group of each observation in x_{obs}. The observations are then sampled according to an established probability of missingness that will be $\frac{0.9}{nG_1}$ for Group 1 and $\frac{0.1}{nG_2}$ for Group 2 (nG_1 and nG_2 are the number of observations in Group 1 and Group 2, respectively);

$$p_{miss} = \begin{cases} \frac{0.9}{nG_1} \text{ if } x_{i,obs} >= median(x_{obs}) \\ \frac{0.1}{nG_2} \text{ if } x_{i,obs} < median(x_{obs}) \end{cases} \tag{2.8}$$

In $MAR4_{univa}$, the locations of x_{miss} where x_{obs} assumes its highest values are set to be missing.

$MAR5_{univa}$ considers both the highest and lowest values of x_{obs}: given the necessary number of observations to have missing values for the specified missing rate, call it $numMV$, $MAR5_{univa}$ sets $numMV/2$ observations to have missing values according

Algorithm 5: Implementation of $MAR3_{univa}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $groups$: array with the groups of each observation of $x_{obs}$
        $nG1$ and $nG2$: number of observations in Group 1 and Group 2, respectively
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    probs = []
    for $g$ in $groups$ do
        if $g$ == 1 then
            probs.append(0.9/$nG_1$)
        if $g$ == 2 then
            probs.append(0.1/$nG_2$)
    observations = random(x, size = numMV, p = probs)
    data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

Algorithm 6: Implementation of $MAR4_{univa}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $x_{obs}$: Observed feature
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    observations = sort($x_{obs}$, reverse = True)
    observations = observations[ 0 : numMV ]
    data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

to the highest values of x_{obs}, and $numMV/2$ according to the lowest.

Univariate MNAR implementations

In the MNAR context, there is no dependency on any observed feature, so for this mechanism there is only one feature of interest, the missing feature, x_{miss}.

For this mechanism, two methods were implemented, $MNAR1_{univa}$ (Figure 2.5a, Figure 2.5b and Algorithm 8) and $MNAR2_{univa}$ (Figure 2.6a, Figure 2.6b and Algorithm 9). $MNAR1_{univa}$ was proposed by Twala et al. [19]: this method is similar to the approach used in $MAR1_{univa}$, but the feature x_{miss} itself is used as observed

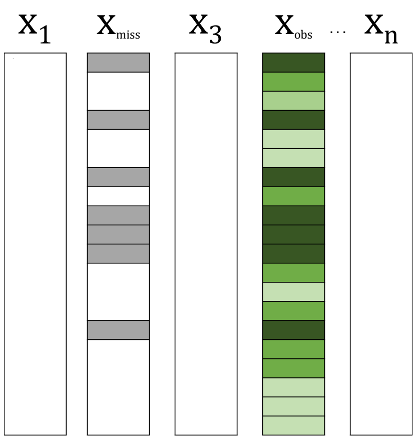


Figure 2.3: Missing data pattern for $MAR4_{univa}$ implementation.

Algorithm 7: Implementation of $MAR5_{univa}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $x_{obs}$: Observed feature
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    highest_observations = sort($x_{obs}$, reverse = True)
    lowest_observations = sort($x_{obs}$, reverse = False)
    highest_observations = highest_observations[ 0 : numMV/2 ]
    lowest_observations = lowest_observations[ 0 : numMV/2 ]
    data[ highest_observations, $j_{miss}$ ] = “$NaN$”
    data[ lowest_observations, $j_{miss}$ ] = “$NaN$”
return (data)
```

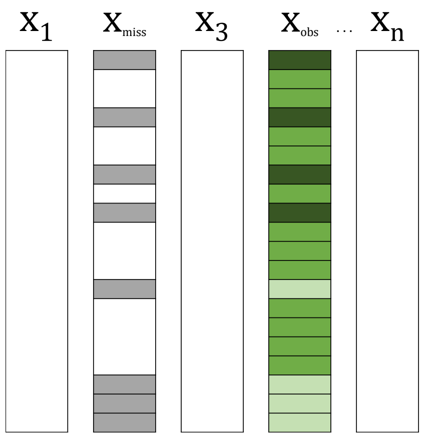
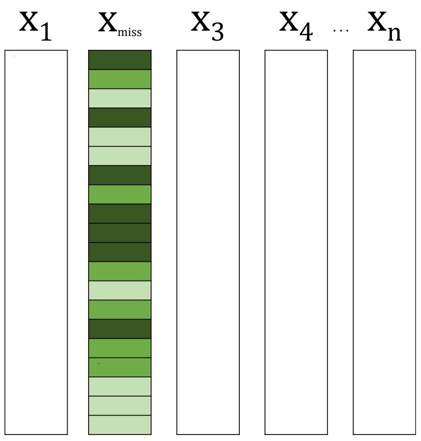


Figure 2.4: Missing data pattern for $MAR5_{univa}$ implementation.

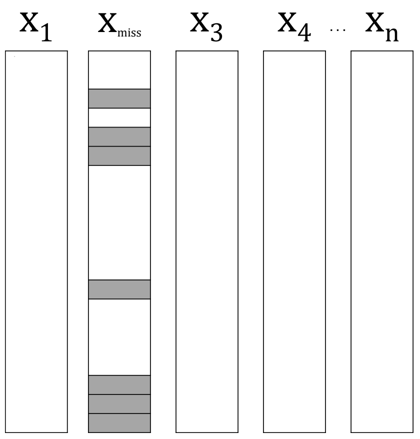
feature, i.e, the MR percentile of x_{miss} is determined and values of x_{miss} lower than the cut-off value are removed.

Algorithm 8: Implementation of $MNAR1_{univa}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $x_{miss}$: Missing feature
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    observations = sort($x_{miss}$, reverse = False)
    observations = observations[ 0 : numMV ]
    data[observations, $j_{miss}$] = "$NaN$"
return (data)
```



(a) Complete dataset.



(b) Dataset with synthetic missing values.

Figure 2.5: Missing data pattern for $MNAR1_{univa}$ implementation.

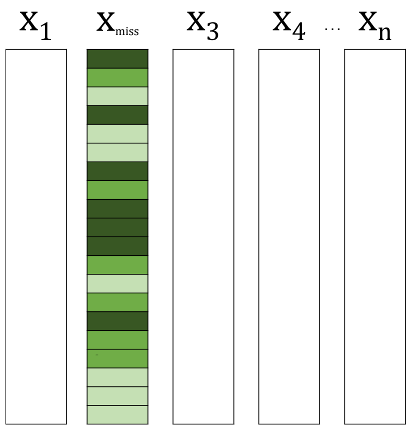
$MNAR2_{univa}$ is created according to Xia et al. [21]: larger values of x_{miss} are removed until the MR is reached.

2.2.1.2 Multivariate implementations

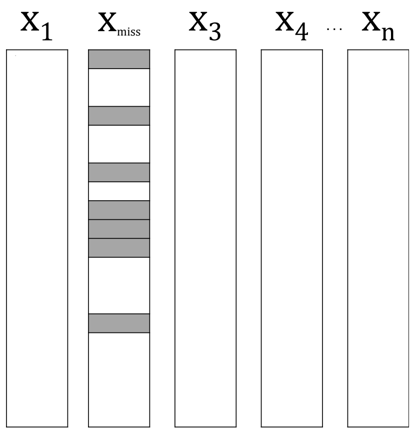
Regarding multivariate implementations, there are several alternatives to choose the missing values positions. For MCAR and MNAR, usually the missing values are generated in all the features of the dataset (except for $MNAR2_{unifo}$ and $MNAR4_{unifo}$, as will be explained bellow). However, for MAR, the process will be different. In this

Algorithm 9: Implementation of $MNAR2_{univa}$.

```
Input  :
        data: Complete dataset
        MR: MD percentage
        $j_{miss}$: Missing feature index
        $x_{miss}$: Missing feature
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    numMV = round($x \times MR \div 100$)
    observations = sort($x_{miss}$, reverse = True)
    observations = observations[ 0 : numMV ]
    data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```



(a) Complete dataset.



(b) Dataset with synthetic missing values.

Figure 2.6: Missing data pattern for $MNAR2_{univa}$ implementation.

case, it is necessary to have at least one observed feature and it is also common to create pairs of correlated features which are composed of an observed and a missing feature, (x_{obs}, x_{miss}), where x_{miss} has the missing values while x_{obs} will be complete.

Multivariate MCAR implementations

We consider three different multivariate implementations of the missing completely at random (MCAR) mechanism, which are presented on five related works (Twala et al. [19], Nanni et al. [22], Garciarena et al. [23], Ali et al. [24] and Zhu et al. [26]).

In Twala et al. [19], the three multivariate generations are performed similar to its univariate implementations. MCAR is generated similar to $MCAR1_{univa}$, but in this case random locations are chosen for each feature, also using a Bernoulli distribution. It would be expected that all features have the same amount of missingness which may not happen because of the Bernoulli distribution. This implementation will be denoted as $MCAR1_{unifo}$ (Figure 2.7 and Algorithm 10).

Algorithm 10: Implementation of $MCAR1_{unifo}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    for $j_{miss}$ in range(0, y) do
        observations = bernoulli(MR, size = x)
        data[(observations == 1), $j_{miss}$] = "$NaN$"
return (data)
```

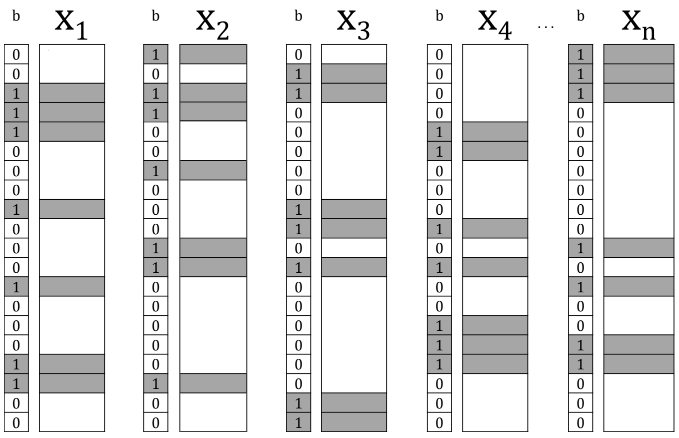


Figure 2.7: Missing data pattern for $MCAR1_{unifo}$ implementation. b represents the Bernoulli distribution for each feature.

Nanni et al. [22] generate MCAR in a slightly different way: instead of generating missing values by feature, they are generated by observation. Here, the desired number of missing values is equally distributed by all the observations and the choice of missing features is randomly performed. So, in this case, different features may be missing for different observations of the dataset. This implementation is not completely accurate since the desired MR may not be attained since, by rounding the number of missing values for each observation, the desired number of missing values may not be reached. Nanni et al. [22] implementation is herein referred to as $MCAR2_{unifo}$ (Figure 2.8 and Algorithm 11).

Algorithm 11: Implementation of $MCAR2_{unifo}$.

```
Input  :
        data: Complete dataset
        MR: MD percentage
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    numMV = round($y \times MR \div 100$)
    for $i$ in range(0,x) do
        featureMV = random(y, size = numMV)
        data[i, featureMV] = "$NaN$"
return (data)
```

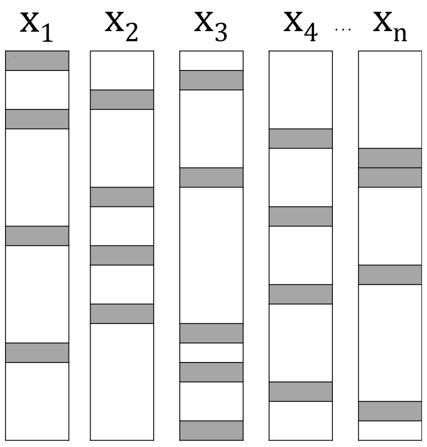


Figure 2.8: Missing data pattern for $MCAR2_{unifo}$ implementation.

In Garciarena et al. [23], Ali et al. [24] and Zhu et al. [26] is proposed a simple method for MCAR generation which will be denoted as $MCAR3_{unifo}$ (Figure 2.9 and Algorithm 12). This method chooses random locations in the dataset to be missing until the desired missing rate is reached.

Multivariate MAR implementations

We consider three multivariate implementations for MAR mechanism, which are based on four different works (Twala et al. [19], Zhu et al. [26], Ali et al. [24] and Garciarena et al. [23]).

The following two MAR implementations use the same approach for choosing the observed and missing features. Twala et al. [19], Zhu et al. [26] and Ali et al. [24] define pairs of features which include an observed and a missing feature (x_{obs}, x_{miss}). This pair selection of features is based on high correlations among all the features of the dataset. In the case of having an odd number of features, the unpaired feature

Algorithm 12: Implementation of $MCAR3_{unifo}$.

```
Input   :
          data: Complete dataset
          MR: MD percentage
Output:
          Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    numMV = round($x \times y \times MR \div 100$)
    countMV = count(data)
    while countMV != numMV do
        observationMV = random(x, size = 1)
        featureMV = random(y, size = 1)
        data[observationMV, featureMV] = “$NaN$”
        countMV = count(data)
return (data)
```

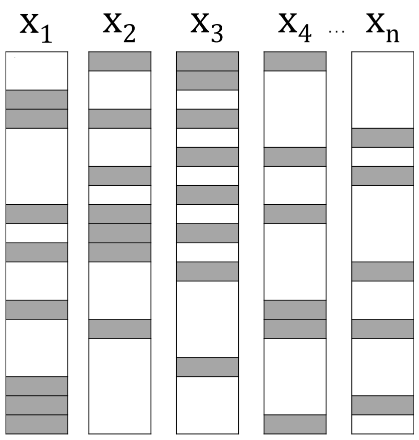


Figure 2.9: Missing data pattern for $MCAR3_{unifo}$ implementation.

may be added to the pair which contains its most correlated feature.

Twala et al. [19] proposed a multivariate implementation which is similar to its univariate version ($MAR1_{univa}$) – herein referred to as $MAR1_{unifo}$ (Figure 2.10 and Algorithm 13). For each pair of correlated features, the missing feature will be the one most correlated with the class labels. In the case of having a triple of correlated features, there will be two missing features which will also be those most correlated with the labels. $MAR1_{unifo}$ sets the observations of x_{miss} to be missing when they correspond to values of x_{obs} bellow $2 \times MR$ quantil ($\frac{3}{2} \times MR$ quantil for triples).

Zhu et al. [26] and Ali et al. [24] also created pairs of correlated features, however, it is not stated how observed and missing features are chosen. So, we adapt the implementation proposed in this work using the following approach: for each pair

Algorithm 13: Implementation of $MAR1_{unifo}$. This pseudo-code assumes that there is an even number of features. *pairs* is an array with the paired indices of features.

```
Input :
        data: Complete dataset
        MR: MD percentage
        pairs: array containing the pairs of correlated features
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    for pair in pairs do
        $j_{obs}$, $j_{miss}$ = select_by_correlation(pair, data)
        $x_{obs}$ = data[ :, $j_{obs}$ ]
        quantil = $2 \times x \times MR \div 100$
        observations = sort($x_{obs}$, reverse = False)
        observations = observations[ 0 : quantil ]
        data[ observations, $j_{miss}$ ] = “$NaN$”
return (data)
```

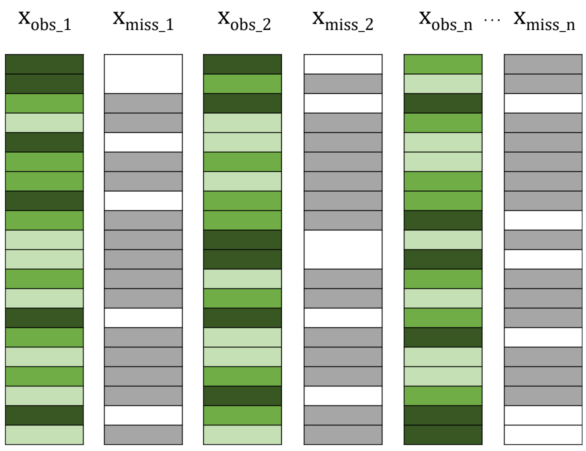


Figure 2.10: Missing data pattern for $MAR1_{unifo}$ implementation.

of correlated features, the missing feature, x_{miss}, will be randomly chosen and the remaining one will be the observed feature, x_{obs}. We use this approach in order to present an alternative to the choice of x_{miss} and x_{obs}. For each pair or triple of features, x_{miss} is divided in two groups according to the median of x_{obs}: the values of x_{miss} will be assigned to a group (or another) according to whether the respective observations of x_{obs} have lower (or greater) values than its median. Zhu et al. [26] proposed a more complete version of this implementation: for categorical features, two equally-sized groups are created by randomly dividing the observations. After splitting the observations into two groups, one group is randomly selected to have missing values with a probability of $4 \times MR$ ($3 \times MR$ for triples). This implementation is herein referred to as $MAR2_{unifo}$ (Algorithm 14).

Algorithm 14: Implementation of $MAR2_{unifo}$. This pseudo-code assumes that there is an even number of features.

```
Input  :
         data: Complete dataset
         MR: MD percentage
         pairs: array containing the pairs of correlated features
Output:
         Dataset with MR% generated MD
begin
    x = numObservations(data)
    for pair in pairs do
        $j_{obs}$, $j_{miss}$ = random_select(pair)
        $x_{obs}$ = data[ :, $j_{obs}$ ]
        g1, g2 = group($x_{obs}$, threshold = median($x_{obs}$))
        g = random_select(g1, g2)
        observations = random(g, size = $4 \times MR \times size(g) \div 100$)
        data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

Garciarena et al. [23] performed MAR generation setting the lowest observations of x_{obs} to be missing in the missing features. In this case, there is only an observed feature x_{obs}, which is randomly chosen. Here, there are nF missing features that are also randomly chosen. This implementation of Garciarena et al. will be referred to as $MAR3_{unifo}$ (Figure 2.11 and Algorithm 15).

Algorithm 15: Implementation of $MAR3_{unifo}$.

```
Input  :
         data: Complete dataset
         MR: MD percentage
         nF: number of missing features
Output:
         Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    numMV = round($x \times y \times MR \div 100 \div nF$)
    $j_{obs}$ = random(y, size = 1)
    $x_{obs}$ = data[ : , $j_{obs}$ ]
    features = remove([1 : y], $j_{obs}$)
    featuresMV = random(features, size = nF)
    observations = sort($x_{obs}$, reverse = False)
    observations = observations[ 0 : numMV ]
    for $j_{miss}$ in featuresMV do
        data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

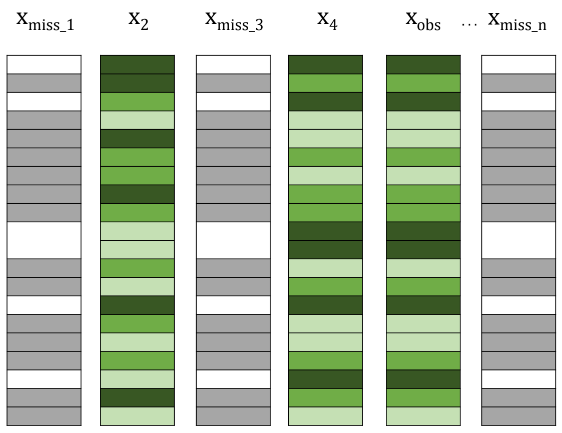


Figure 2.11: Missing data pattern for $MAR3_{unifo}$ implementation.

Multivariate MNAR implementations

For MNAR mechanism we implement four multivariate MNAR versions, which are presented in four different works (Zhu et al. [26], Ali et al. [24], Garciarena et al. [23] and Twala et al. [19]).

Zhu et al. [26] and Ali et al. [24] generate MNAR by dividing each missing feature (i.e., each feature of the dataset) into two groups: one group will have the observations bellow the median of x_{miss} and the other will have observations above the median. Zhu et al. [26] proposed a more complete version of this implementation as already explained for $MAR2_{unifo}$. After splitting observations, one group is randomly chosen and their values are set to be missing with a probability of $2 \times MR$. The approach chosen to be implemented was the one of Zhu et al. [26] which will be denoted as $MNAR1_{unifo}$ (Algorithm 16).

Algorithm 16: Implementation of $MNAR1_{unifo}$.

```
Input  :
        data: Complete dataset
        MR: MD percentage
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    for $j_{miss}$ in range (0,y) do
        $x_{miss}$ = data[ : , $j_{miss}$ ]
        g1, g2 = group($x_{miss}$, threshold = median($x_{obs}$))
        g = random_select(g1, g2)
        observations = random(g, size = $2 \times MR \times size(g) \div 100$)
        data[ observations, $j_{miss}$ ] = "$NaN$"
return (data)
```

Garciarena et al. [23] proposed two different implementations for MNAR multivariate generation: $MNAR2_{unifo}$ (Figure 2.12 and Algorithm 17) and $MNAR3_{unifo}$ (Figures 2.13a and 2.13b and Algorithm 18). $MNAR2_{unifo}$ is also called *Missingness depending on unobserved Variables (MuOV)* since each feature of the dataset will have the same number of missing values for the same observations. The missing observations and the nF missing features are randomly chosen. Here, it is not intended to generate MD in all the features from the dataset because in that case we would be deleting complete samples. However, the authors do not refer to any restriction on the value of nF. *Missingness depending on Its Value itself (MIV)* is the definition chosen by Garciarena et al. [23] for $MNAR3_{unifo}$. This implementation chooses the lower values of each feature to be missing so, in this case, an observation can be missing depending on its value itself. Also, for this implementation, all the features will have the same number of missing values.

Algorithm 17: Implementation of $MNAR2_{unifo}$.

```
Input  :
            data: Complete dataset
            MR: MD percentage
            nF: number of missing features
Output:
            Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    numMV = round($x \times y \times MR \div 100 \div nF$)
    observations = random(x, size = numMV)
    featuresMV = random(y, size = nF)
    for $j_{miss}$ in featuresMV do
        data[ observations, $j_{miss}$ ] = “$NaN$”
return (data)
```

Twala et al. [19] proposed a MNAR multivariate implementation similar to their MAR implementation ($MAR1_{unifo}$). This implementation is called $MNAR4_{unifo}$ (Figures 2.14a and 2.14b and Algorithm 19) and, unlike the other approaches used for MNAR generation, there is also the creation of pairs of correlated features. As in $MAR1_{unifo}$, x_{miss} will be the one most correlated with the class labels and, in this case, there will be no x_{obs}. Furthermore, for an odd number of features, there will be a triple of correlated features with two missing features. For each x_{miss}, observations bellow $2 \times MR$ are set to be missing ($\frac{3}{2} \times MR$ in case of a triple).

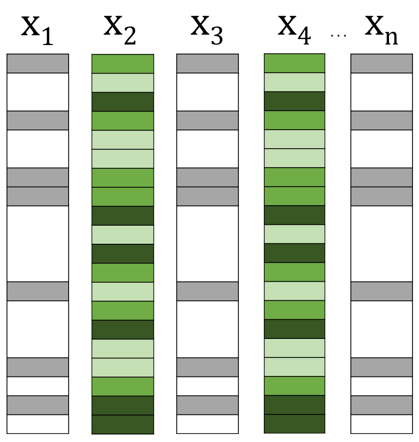
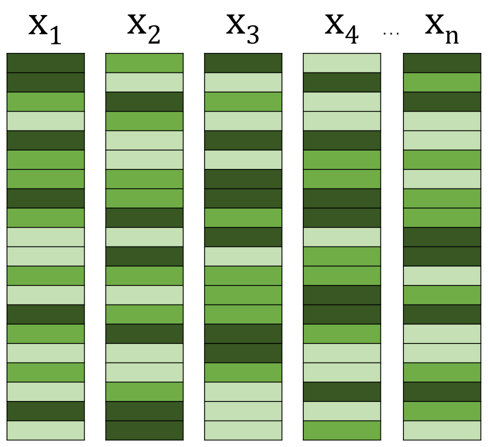


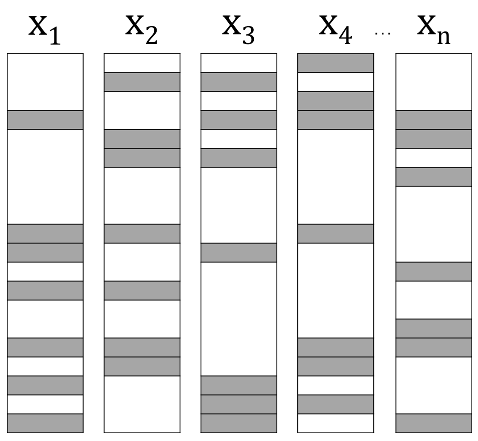
Figure 2.12: Missing data pattern for $MNAR2_{unifo}$ implementation.

Algorithm 18: Implementation of $MNAR3_{unifo}$.

```
Input :
        data: Complete dataset
        MR: MD percentage
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    y = numFeatures(data)
    numMV = round($x \times MR \div 100$)
    for $j_{miss}$ in range(0, y) do
        observations = sort($j_{miss}$, reverse = False)
        observations = observations[ 0 : numMV ]
        data[ observations, $j_{miss}$ ] = “$NaN$”
return (data)
```



(a) Complete dataset.

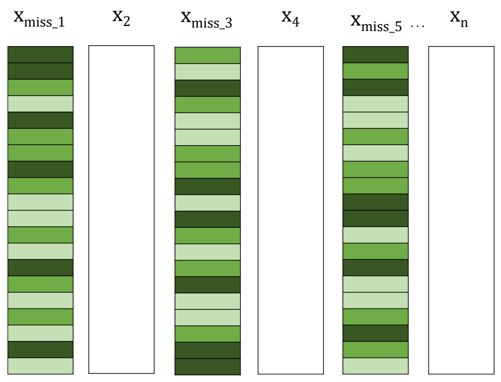


(b) Dataset with synthetic missing values.

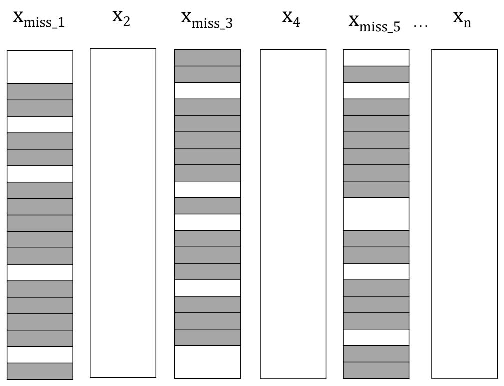
Figure 2.13: Missing data pattern for $MNAR3_{unifo}$ implementation.

Algorithm 19: Implementation of $MNAR4_{unifo}$.

```
Input  :
        data: Complete dataset
        MR: MD percentage
        pairs: array containing the pairs of correlated features
Output:
        Dataset with MR% generated MD
begin
    x = numObservations(data)
    for pair in pairs do
        $j_{obs}$, $j_{miss}$ = select_by_correlation(pair, data)
        $x_{miss}$ = data[ : , $j_{miss}$ ]
        quantil = $2 \times x \times MR \div 100$
        observations = sort($x_{miss}$, reverse = False)
        observations = observations[ 0 : quantil ]
        data[ observations, $j_{miss}$ ] = “$NaN$”
return (data)
```



(a) Complete dataset.



(b) Dataset with synthetic missing values.

Figure 2.14: Missing data pattern for $MNAR4_{unifo}$ implementation.

Conclusions

In this section, we have provided a detailed description of several implementations for the three missing mechanisms. We implemented all the above approaches, however, we did not use them at all the stages of this work. We considered that the most appropriate approaches are those that guarantee the generation of the desired number of missing values. Therefore, we performed a selection of these implementations in order to have one per each mechanism and type (*univa* and *unifo*). More details will be provided in Chapter 4.

2.3 Missing Data Imputation

Imputation methods aim to find plausible values to replace the ones that are missing and are mainly divided into statistical-based or machine learning-based methods [5]. Statistical methods consist in replacing the missing observations with the most similar ones among the training data, without the need of constructing a predictive model to evaluate their "similarity" (e.g. Mean imputation). Machine learning-based techniques, construct a predictive model with the available data to estimate values for replacing those that are missing (e.g. k-Nearest Neighbours (kNN) imputation, Support Vector Machines (SVM) imputation, Stacked Denoising Autoencoders (SDAE) imputation).

2.3.1 Mean/Mode Imputation

The simplest method to impute missing values is Mean/Mode imputation. The missing entries of each feature, x_{miss}, are replaced by the mean of its observed values or by the mode, in case of categorical values. Class-conditional mean imputation is a variant approach where each MV is replaced with the mean of observed values belonging to its class [5]. Although this method is quite simple it has several limitations since it can produced biased estimates and it does not preserve the relationships between features [27]. Mean/Mode imputation is herein referred to as Meanimp and was applied using the implementation of *Scikit Learn's*.

2.3.2 Imputation with k-Nearest Neighbours

The kNN algorithm is quite popular in the missing data imputation field [28]. Basically, given a missing value in a pattern, a plausible substitute value can be estimated using the values of patterns that are close to it, considering observed values from other features than x_{miss}. This estimation may be the mean of nearest neighbours in case of a continuous feature and also may be the mode, for categorical features. This technique requires a selection of a distance metric and a definition of the number of neighbours, k: we defined the nearest neighbours using Euclidean Distance, as it is done in several related works [29]; we used three different values for k – 1, 3 and 5. Furthermore, we refer to this method as kNNimp and to apply it we use *fancyimpute* [30] implementation in *Python*.

2.3.3 Imputation with Support Vector Machines

SVM are a powerful classification technique widely used in the pattern recognition field. This approach can be used in both classification and regression problems. Originally, SVM was a linear classifier but it can be improved by modifying kernel functions [31]. One of the most typical kernel functions is the Gaussian Radial Basis Function (RBF) kernel. In this work, we used a SVM regressor with a RBF kernel to imputation purposes. This method requires a tuning process where the optimal parameters C and γ are calculated, for each complete dataset. To fill in the missing values, each missing feature, x_{miss} is used as target while the remaining features are used to train the model. This imputation approach, here referred to as SVMimp, was implemented by us using *scikit-learn*.

2.3.4 Multiple Imputation by Chained Equations

Multiple Imputation by Chained Equations (MICE) is a widely used multiple imputation approach [32, 33] that creates several regression models so that each x_{miss} is conditionally modeled by the remaining features. This method is composed of 4 main steps [34]:

1. All the MVs are replaced by the average of the observed values – pre-imputation step using mean imputation;
2. For each x_{miss}: the MV are set back to be missing;
3. A regression of x_{miss} predicted by the remaining features of the dataset is performed, using the observed values of x_{miss};
4. The regression equation obtained in the previous step is used to predict the MV of x_{miss}.

This iterative process through the missing features is repeated until the convergence of the imputation parameters (e.g., coefficients of the regression model) [34]. At the end, MICE has replaced the MV using several regressions that preserved the relationship between the data. MICE was applied using *fancyimpute* [30] implementation in *Python*, with default settings.

2.3.5 Expectation-Maximization

The Expectation-Maximization (EM) [35] is a maximum likelihood method that estimates the MV considering their relation with the unknown parameters of the data model [36]. The main idea of this algorithm is to iteratively adjust the MV while preserving the covariance structure of the data. In brief, the EM can be summarized in the following steps:

1. Initialize the unknown parameters of the data model (randomly);
2. Compute the probability distribution over the possible values, using the current parameters. Fill-in the missing values with the estimated values – Expectation step;
3. Estimate another set of parameters for the current data – Maximization step.

The iteration between steps 2 and 3 continues until the estimates converge. We used EM's implementation from library *impyute* [37] in *Python*.

2.3.6 Denoising Autoencoders

Neural network-based methods have been increasingly used for missing data imputation [38]; however, deep learning architectures especially designed for missing data imputation remain an understudied topic.

Denoising Autoencoders (DAEs) [39] are designed to recover noisy data ($\tilde{\mathbf{x}}$), which can exist due to data corruption via some additive mechanism or by missing data [40]. DAEs are a variant of Autoencoders (AEs) (Figure 2.15) which are a type of artificial neural networks that are trained to reproduce its input at the output layer. Each autoencoder is composed by three layers (input, hidden and output layer) which can be divided into two parts: encoder (from the input layer to the output of the hidden layer) and decoder (from the hidden layer to the output of the output layer).

The DAE is similar to a basic AE: the main difference is the application of a stochastic corruption to the inputs of the DAE during the training phase. One of the possible corruptions techniques consists in setting to 0 a fixed amount of features for each observation (Figure 2.17). There are other possible corruption processes, such as adding Gaussian noise or salt-and-pepper noise [41].

The encoder part of a DAE maps an input vector $\tilde{x}$ to a hidden representation $\mathbf{y}$, through a nonlinear transformation $f_\theta(\tilde{x}) = s(\tilde{x}\mathbf{W}^T + \mathbf{b})$ where θ represents

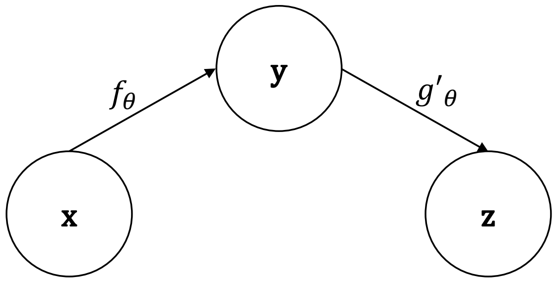


Figure 2.15: Simplified structure of an Autoencoder. f represents the encoder and g represents the decoder.

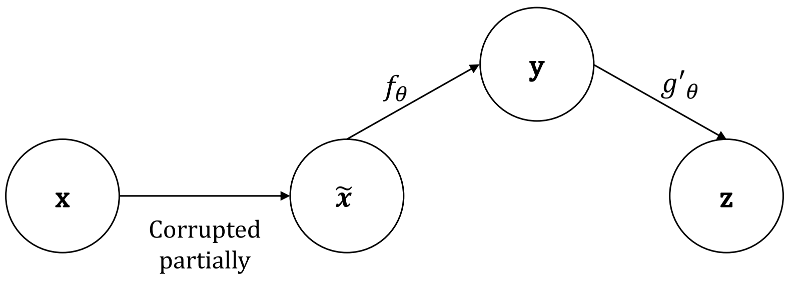


Figure 2.16: Simplified structure of an Denoising Autoencoder.

the weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$. The resulting $\mathbf{y}$ representation is then mapped back to a vector $\mathbf{z}$ which has the same shape of $\tilde{x}$, where $\mathbf{z}$ is equal to $g'_\theta(\mathbf{y}) = s(\mathbf{W'y} + \mathbf{b'})$. The training of an DAE consists in optimising the model parameters ($\mathbf{W}$, $\mathbf{W'}$, $\mathbf{b}$ and $\mathbf{b'}$) to minimise the reconstruction error between $\mathbf{x}$ (the uncorrupted input) and $\mathbf{z}$, using the squared error loss:

$$L_2(\mathbf{x},\mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^2 \tag{2.9}$$

The difference for the train of an AE is that $\mathbf{z}$ is a deterministic function of $\tilde{x}$ rather than $\mathbf{x}$. Defining the joint distribution:

$$q^0(X, \tilde{X}, Y) = q^0(X) q_D(\tilde{X}|X) \delta_{f_\theta(\tilde{X})}(Y) \tag{2.10}$$

where $\delta_u(v)$ equals 0 when $u \neq v$, Y is a deterministic function of $\tilde{X}$. We use upper-case letters to represent random variables. The objective function minimized by Stochastic Gradient Descent (SGD) is:

$$\arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X,\tilde{X})} L_2(X, g_{\theta'}(f_\theta(\tilde{X}))) \tag{2.11}$$

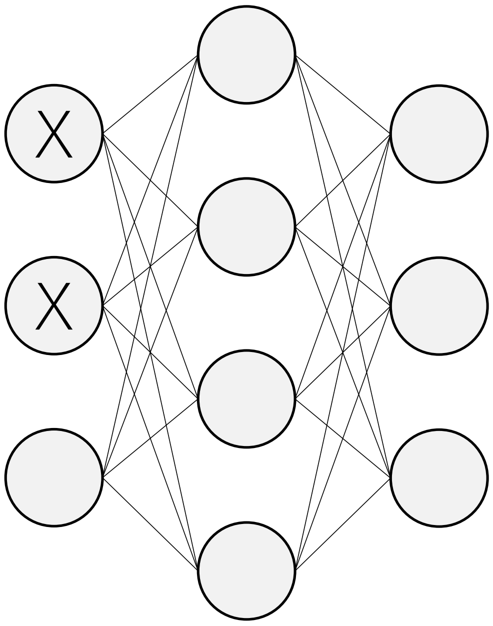


Figure 2.17: Representation of the training phase of DAE. Some of the input nodes are randomly setted to 0. However, the reconstruction error compares the original data ($\mathbf{x}$) with the reconstructed data ($\mathbf{z}$).

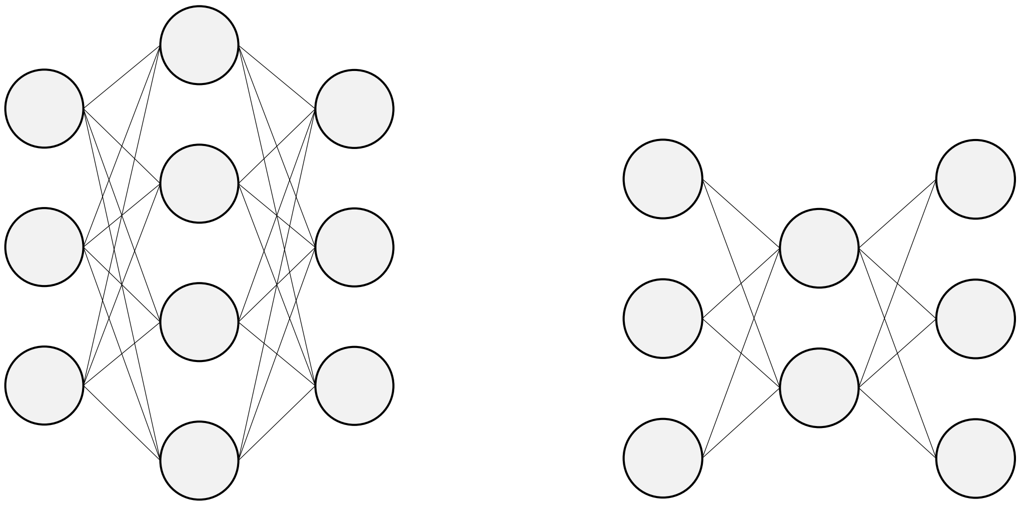
where $\mathbb{E}$ represents the Expectation and $q^0(X,\tilde{X})$ corresponds to the stochastic mapping which performs the partial destruction of X.

Network Structure

There are two types of representations for a DAE [40]: overcomplete (Figure 2.18a), when the hidden layer has more nodes than the input layer, and undercomplete (Figure 2.18b), when the hidden layer is smaller than the input layer.

In the case of an AE, an overcomplete architecture only learns the identity function and, therefore, copies the input to the output. To avoid this behavior, the objective function can be modified to include a regularization term. The DAE can be overcomplete without the need of any regularization since it compares the original impute to its corrupted version [40].

When an undercomplete architecture is used, the DAE is forced to learn a more concise representation of the input data.



(a) Representation of an overcomplete architecture.

(b) Representation of an undercomplete architecture.

Figure 2.18: Differences in network architectures.

Stacked Denoising Autoencoders

Vincent et al. [41] proposed a strategy to build deep networks by stacking layers of Denoising Autoencoders. The results have shown that stacking DAEs improves the performance over the standard DAE. Furthermore, deeper architectures tend to be a better solution in terms of generalization performance, using fewer nodes per layer and, consequently, fewer parameters. On the other hand, the optimization of these architectures is more complex [42].

– *Layer-wise Pretraining and Fine Tuning*

The training process of SDAE consists of a layer-wise unsupervised pre-training – the representation of the k-th layer is the input for (k+1)-th layer which is trained after the k-th layer has been trained. When the k layer is trained it will have as input the uncorrupted output from the previous layers. After the training of a few layers, the fine tuning will be performed – the current network parameters will initialize a network that will be optimized under a supervised training criterion.

Regularizations

Adding regularizers to the objective function forces the model to have different properties. There are several regularization terms that can be added to the objective function to prevent the overfitting of the training data. For example, L2 regularization is also known as “weight decay” because it forces the weights to decay towards zero (but not exactly zero). The L2 term consists of the sum of the squared values of the weights so, larger weights lead to a larger error which causes the training algorithm to favour and generate smaller weights. Applying dropout is also regularization [43] since it prevents the overfitting to the training data by “ignoring” some nodes of the network during the training phase.

Activation Functions

Each node of any layer uses an activation function to compute the weighted sum of the inputs and to define its output. The most popular activations functions are Rectified Linear Units (ReLu), Hyperbolic Tangent (Tanh) and logistic function [40].

Optimization Algorithms

The most usual optimization algorithm is Stochastic Gradient Descent (SGD) [44] which has several variants, such as AdaGrad [45], Adadelta [46], RMSProp [47] and Adam [48]. These algorithms are based in the gradient descent method [44]. Gradient descent is an iterative optimization algorithm that aims at minimizing a given function. More precisely, Gradient descent uses backpropagation to calculate the gradient of the objective function and allows the optimization algorithm to adjust the parameters (**W**, **W’**, **b** and **b’**) in order to find a minimum of the function.

2.4 Classification and Evaluation Metrics

The main purpose of data imputation is to replace the MVs with estimates that are closest to the original values. On the other hand, the imputation process precedes the classification task and, therefore, the imputation method must be properly chosen in order to not affect the performance of the classifier. These are two key points of the MD problem that must be evaluated: imputation quality and classification performance.

Classification

In machine learning, there are a lot of supervised learning algorithms to perform classification: SVM is known to belong to the best performers [49]. SVMs are based on the construction of a hyperplane that defines a decision boundary. In other words, for a set of labeled data, the SVM returns an optimal hyperplane that maximizes the margin of separation and that will be capable of categorizing new samples (data from the test set). As already mentioned in Section 2.3.3, SVM is a linear discriminant algorithm but when no linear separation is possible, the implementation uses a kernel which maps the training samples to a higher dimensional space and learns to separate the samples in that space. The use of this kernel trick makes SVM so popular and accurate since it has a large capacity for generalization regardless of the distribution of the patterns [50].

Every machine learning model must be evaluated in terms of generalization capability - also known as model evaluation. The Holdout method is a popular strategy used to perform this evaluation. This method randomly separates the dataset into 2 subsets, called training and test set. The model is fitted to the training set while the error estimation is computed for the test set [50].

In this work, classification performance was assessed with F-measure which considers both Precision and Recall [51]:

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.12}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.13}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.14}$$

In Equations 2.13 and 2.14, TP correspond to the true positives (i.e. the number of positive patterns correctly predicted), FP denotes the false positives (i.e. the number of patterns wrongly predicted as positive) and FN are the false negatives (i.e. the number of patterns wrongly predicted as negative).

Imputation Quality

Besides the evaluation of classification performance, it is also important to assess the imputation quality, in other words, measure how close the imputed values are to the original ones [5, 52].

The coefficient of determination, R^2, is equivalent to the square of Pearson correlation coefficient. This metric measures the correlation between 2 features which, in the context of imputation, corresponds to measure between the original feature (before the synthetic generation of MV) and the imputed feature. R^2 varies between 0 and 1.

$$R^2 = \left(\frac{\sum_{i=1}^{n} (\tilde{x}_i - \bar{\tilde{x}}_i)(x_i - \bar{x}_i)}{\sqrt{\sum_{i=1}^{n} (\tilde{x}_i - \bar{\tilde{x}}_i)^2 \sum_{i=1}^{n} (x_i - \bar{x}_i)^2}} \right)^2 \tag{2.15}$$

Root Mean Squared Error (RMSE) is a quadratic metric used to measure the differences between 2 features of interest. In other words, RMSE is the square root of the average of the squared differences between the imputed feature and the original one. This metric can range from 0 to ∞.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2} \tag{2.16}$$

In Equations 2.15 and 2.16, $\tilde{x}$ are the imputed values of a feature, $\bar{\tilde{x}}$ is the mean of the imputed values, x are the corresponding original values, $\bar{x}$ is the mean of the original values and n is the number of missing values.

2.5 Statistical Tests

Statistical inference is normally used for comparing classifiers over several datasets [53]. Following this idea, statistical tests can also be performed in order to compare imputation methods over multiple datasets.

The Friedman test [54] is a non-parametric statistical test which is similar to the non-parametric repeated measures ANOVA [53, 55]. Given N datasets and k algorithms, Friedman test ranks the algorithms for each dataset – ${r_i}^j$ will be the rank for the j^{th} algorithm on dataset i – and compares the mean of the ranks for each algorithm, $R_j = \frac{1}{N}\sum_i {r_i}^j$. Under the null hypothesis of the test, there are no differences between the algorithms and the Friedman statistic is:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{2.17}$$

This statistic has a equivalent distribution to χ_F^2 with k-1 degrees of freedom as N and k become large. Iman et al. [56] derived an approximation for the test statistic which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{2.18}$$

The F statistic value must be compared with the value of F distribution, which will give the $p - value$ for the test.

The t-Student test for paired samples is a parametric test that compares population means for a pair of random samples [57]. The null hypothesis of the test states there is no statistical difference between the mean of the two populations.

The use of parametric tests requires the verification of two assumptions: i) normal distribution ii) homogeneity of variance. In order to test if a sample follows a normal distribution may be performed a Kolmogorov-Smirnov [58] test or a Shapiro-Wilk test [59]. To determine if two samples have homogeneous variance is used the Levene's test [60].

The Wilcoxon signed rank test [61] is a non-parametric test that can be used for paired data or for a single set of observations – it is a non-parametric alternative to the t-Student test for paired samples. When used for paired data, the null hypothesis of the test states that the median difference between the pairs of observations is zero [62]. The test calculates the differences between pairs and ranks them ignoring the sign; the test statistic is the sum of the ranks for either positive and negative differences.

3

Literature Review

In this chapter, we provide a detailed description of several imputation approaches based on Stacked Denoising Autoencoders (SDAE), since the main goal of this work is the study of these techniques for imputation purposes. For each studied work, we perform a detailed characterization regarding the characteristics of the used data, generated missing mechanism (and the respective missing rates) and also refer to each proposed architecture regarding its hyperparameters.

In two recent works, Gondara et al. studied the appropriateness of SDAE for multiple imputation [8] and their application to imputation in clinical health records [9]. In these works, the proposed algorithm is compared with Multiple Imputation by Chained Equations (MICE) using the Predictive Mean Matching method.

In the first work, Gondara et al. [8] proposed a multiple imputation approach based on overcomplete SDAE. Two different scenarios of missingness were considered: all the features were set to have missing values (uniform synthetic generation) and only half of the features were set to be missing (random synthetic generation). These two scenarios were created for Missing Completely At Random (MCAR) and Missing Not At Random (MNAR) mechanisms using 15 real-world and publicly-available datasets. The proposed model includes a pre-imputation step which imputes the incomplete datasets with the well-known method Mean/Mode – continuous features are imputed with its mean while categorical ones are imputed using its mode. The training phase starts with a stochastic corruption (dropout) of 50% (i.e., for each training batch, half of the inputs are randomly set to zero). The proposed architecture is composed of 5 hidden layers - each successive hidden layer has $\theta = 7$ more nodes than the previous one. So, this is an overcomplete architecture and θ quantifies the increase in the dimensionality of successive layers - different values for θ were tested in order to choose the best one. At the end, $\theta = 7$ was chosen as increment since it showed better results on several datasets. The model was trained using 500 epochs, an adaptative learning rate with a time decay factor of 0.99 and a Nesterov's accelerated gradient. In order to ensure a faster convergence, the input was standardized between 0 and 1. Furthermore, the authors have chosen Hyperbolic Tangent (Tanh) as activation function rather than Rectified Linear Units (ReLu) since they found that the first performed better for small datasets. They also used an early stopping rule which ensures that the training process is finished when a Mean Squared Error (MSE) of 1×10^{-6} is achieved or when there is no improvement in a moving average (length 5) of the error deviance. Each training process uses 70% of the data while the remaining 30% is used as test set. The imputation results of both mechanisms (MCAR and MNAR) are compared using Sum of Root Mean Squared Error (RMSE_{sum}) and this value is relative to the test set - they show that the SDAE-based approach outperforms MICE for all the uniform scenarios and in 7 seven cases for the random scenario (this can be seen for 2 datasets under MCAR and for 5 datasets under MNAR). Additionally, MNAR mechanism is also evaluated in terms of classification error, using a Random Forest (RF) classifier. This analysis also proved that data imputed with the SDAE model has a higher classification (in average) than data imputed with MICE.

In the second work [9], the authors propose a SDAE-based model to fill in loss to follow-up information, using 10 simulated and 4 real-world datasets, under MCAR and MNAR mechanisms. Loss to follow-up information occurs often in clinical research when a patient who was participating in a clinical trial leaves it before it is completed. The proposed model is composed of 4 fully connected hidden layers and each successive hidden layer has an increment of $\theta = 5$ in the number of nodes (in the encoder). Contrary to the work presented above, a dropout of 20% and batch normalization are applied to each layer of the proposed model. These two techniques are used in order to avoid a possible overfitting issue that may arise in overcomplete architectures, when the number of units is greater than the dimensionality of the input. So, the authors proposed an alternative version of the SDAE architecture since they applied dropout at all the hidden layers and not just at the input layer. Contrary to what was expected, the authors did not mention any pre-imputation phase. Furthermore, ReLu was used as activation function, MSE was used as loss function, no early stopping rule was used and the models were trained using 1000 epochs, although convergence occurs in less than 200 epochs for most cases. Besides the use of RMSE_{sum} for measuring the imputation performance of continuous time, the authors also evaluate the quality of imputation for binary outcomes using a normalized metric (which allows to compare the results of the different datasets). The analysis proved that the SDAE-based model performs better in terms of imputation quality than MICE – SDAE outperforms MICE with a minimum and maximum difference of 2.4% and 64.9%, respectively (for real-world datasets).

In both works, although the authors prove the advantages of SDAE for imputation, a complete study under all missing mechanisms is not provided, since in both cases, Missing At Random (MAR) generation is completely disregarded. Furthermore, they only compare two imputation methods (MICE and SDAE) and the classification performance is only evaluated for one mechanism (MNAR).

Beaulieu et al. [63] used Autoencoders (AEs) to impute data for electronic health records. This approach is compared with 5 other imputation strategies: Iterative Singular Value Decomposition (SVD), k-Nearest Neighbours (kNN) imputation, SoftImpute, Mean imputation and Median imputation [63]. The imputation performance was evaluated, in terms of Root Mean Squared Error (RMSE), for two missing mechanisms, MCAR and MNAR, and the disease progression prediction was also evaluated, comparing the performance of a RF regressor for the different imputation strategies. Regarding the AE used for the evaluation, it is composed of 2 hidden layers of 500 nodes each (undercomplete architecture) and a dropout of 20% was applied between each layer. The training process starts with normalization of the inputs in order to have values between 0 and 1. Furthermore, AEs were trained using a modified binary cross entropy cost function [64] that takes into consideration the values that are missing, so pre-imputation is not required. The results have shown that the AE-based approach is the one with best results under MCAR mechanism - with a minimum and maximum difference from the second best method (SoftImpute) of 0.005 (Missing Rate (MR) of 50%) and 0.1 (MR of 30%), respectively. For MNAR mechanism, the authors proved that AE had the best results but these are very similar to those obtained by kNN imputation, Softimpute and SVD - AEs differ from the second best method by a minimum of 0.0045 (for kNN imputation under a MR of 20%) and a maximum of 0.0125 (for SoftImpute under a MR of 40%).

Duan et al. [10, 11] used SDAE for traffic data imputation and evaluated the imputation quality using RMSE, Mean Absolute Error (MAE) and Mean Relative Error (MRE). The type of data used in these works does not require a pre-imputation step since the missing values are represented by 0.

In the first work [10], Duan et al. compared the proposed approach with another that uses Artificial Neural Networks (ANN) with the same set of layers and nodes as the ones used in SDAE. Contrary to several works that deal with missing data separately from the observed data, the authors used observed data along with missing data for the process of imputation. The SDAE architecture is undercomplete and is composed of 3 hidden layers - input and output layer was designed to contain 288 nodes and the 3 hidden layers were made of 144, 72 and 144 nodes, respectively. The data was divided into training and test sets using a ratio of 3:2. The results are analysed for a random mechanism of missingness (MCAR) performed for several missing rates ranging from 10% to 90%. Regarding the RMSE metric, it can be observed that its values vary between 16.9 and 20.3, for the SDAE approach, and between 17 and 21, for the ANN - SDAE proved to be a better imputation method than ANN in most cases.

In the second work [11], other 3 imputation methods were used for comparison: Autoregressive Integrated Moving-Average (ARIMA), history model and ANN. The authors studied the influence of spatial and temporal factors on the imputation process. For this reason, they evaluate the difference between data from different scenarios: data collected at one or multiple stations (spatial factors) and data collected on weekdays and non-weekdays (temporal factors). Here, they perform a more complete study than in [10] where only data from weekdays is used. The proposed architecture based on SDAE used sigmoid function as activation function and is composed of 3 hidden layers with 144, 72 and 144 nodes (undercomplete architecture). The determination of this hyperparameters was a result of a grid search procedure. At the end, the results were analysed for missing rates ranging from 5% to 50% and the proposed SDAE model has proved to outperform the remaining models, followed by ANN - in terms of RMSE, the values range from 13.5 to 14.9 for the SDAE approach, while for ANN the range is between 15.2 and 17.5.

Ning et al. [12] proposed an algorithm based on SDAE for dealing with big data of quality inspection. The proposed approach is compared with 2 other imputation algorithms – a weighted k-Nearest Neighbours data filling algorithm based on Grey correlation analysis (GBWkNN) [65] and Mutual k-Nearest Neighbours Imputation (MkNNI) [66] – that are both based on the kNN algorithm. Contrary to most of the works, the authors do not describe the architecture of the SDAE and they do not refer any pre-imputation step. The final results are evaluated through d_2 (suitability between the imputed value and the actual value) and RMSE under several missing rates (1, 5, 10, 15, 20, 25 and 30%). The authors conclude that the proposed SDAE-based approach surpasses the comparative imputation strategies. Moreover, the SDAE is followed by GBWkNN and the RMSE metric ranges from 13.4 to 14.5 and 15.2 to 17.5 for these two approaches, respectively.

Sánchez-Morales et al. [13] proposed an imputation method that uses a SDAE. The main goal of their work was to understand how the proposed approach can improve the results obtained in the pre-imputation step. Only MCAR generation was consider under three different missing rates: 10, 20 and 30%. Besides this synthetic generation of missing values, the authors deleted some input values of the pre-imputed dataset, in order to generate some noise. So, these deleted values correspond to some known values that are used as targets which allows a more accurate prediction of the real missing values – this technique could be considered as a "kind of dropout" applied over the inputs and not over the hidden nodes (traditional approach). The proposed model has 3 hidden layers with 25%-75% of expansion and for each imputation there were made 50 training runs. Moreover, each dataset is split into training set (80%) and test set (20%). They used three state-of-the-art methods for pre-imputation: Zero Imputation, kNN Imputation and Support Vector Machines (SVM) Imputation. In this study, 3 datasets from UCI were used to report the results. These results showed that SDAE is capable of improving the final imputed values from a pre-imputed dataset - the quality of imputation improves from 17% to 96%, regarding all the studied scenarios.

Table 3.1: Summary of reviewed works.

Authors	**Missing Mechanisms**	**Missing Rates**	**Dataset(s)**	**Comparative strategies**	**Metrics**
Gondara et al. [8]	MCAR MNAR	20%	Standard open source datasets (15): Boston Housing; Breast Cancer; DNA; Glass; House Votes; Ionosphere; Ozone; Satellite; Servo; Shuttle; Sonar; Soybean; Vehicle; Vowel; Zoo.	MICE	RMSE
Gondara et al. [9]	MCAR MNAR	60 and 80%	Simulated data (1); Real life datasets (4): Grace; EORTC; RH and HDD.	MICE	RMSE
Beaulieu et al. [63]	MCAR MNAR	10, 20, 30, 40 and 50%	Pooled Resource Open-Access ALS Clinical Trials Database (PRO-ACT)	IterativeSVD kNN imputation SoftImpute Mean imputation Median imputation	RMSE
Duan et al. [10]	MCAR	1, 10, 20, 30, 40, 50, 60, 70, 80 and 90%	Traffic flow data - Caltrans Performance Measurement System (PeMS)	ANN	RMSE MAE MRE
Duan et al. [11]	MCAR	5, 10, 15, 20, 25, 30, 35, 40, 45 and 50%	Traffic flow data - Caltrans Performance Measurement System (PeMS)	ARIMA history model ANN	RMSE MAE MRE
Ning et al. [12]	MCAR	1, 5, 10, 15, 20, 25 and 30%	Big Data of quality inspection	GBWkNN MkNNI	d2 RMSE
Sánchez-Morales et al. [13]	MCAR	10, 20 and 30%	Standard datasets from UCI (3): Cloud dataset; Blood transfusion and Boston housing	Used for pre-imputation: Zero Imputation k-Nearest Neighbour Imputation Support Vector Machine Imputation.	MSE

To summarise, most of related works do not address all three missing data mechanisms, and mostly evaluate the results in terms of quality of imputation (e.g., RMSE, MSE, MAE) rather than evaluating the usefulness of an imputation method to generate quality data for classification (e.g. evaluation of final classification metrics). Furthermore, none of the reviewed works studied the effects of different missing data mechanisms and imputation techniques including SDAE for several missing rates. In this thesis, we propose a more complete study on the use of SDAE for data imputation, accessing the effects of several missing mechanisms, missing rates and comparing them with other well-established imputation methods.

4

Proposed Architecture

Our experimental setup comprised five main stages: (1) Data Collection, (2) Missing Data Generation, (3) Missing Data Imputation, (4) Classification and (5) Evaluation (Figures 4.1 and 4.2).

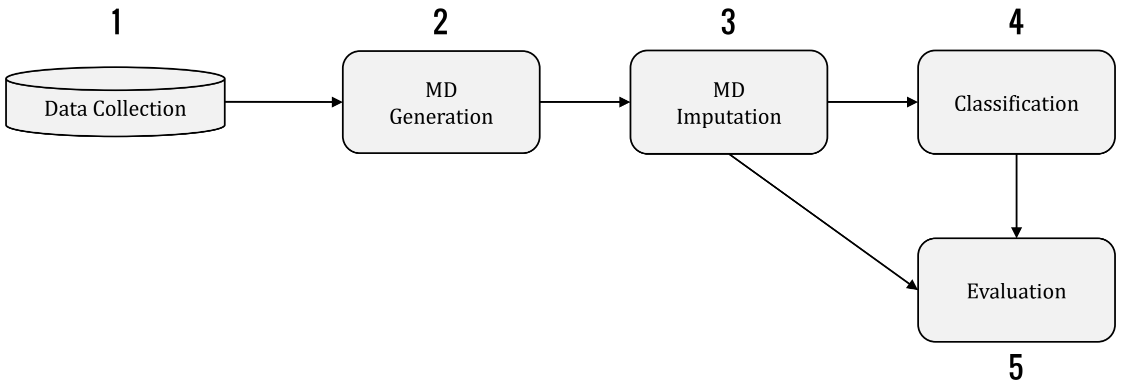


Figure 4.2: Summary of the experimental setup.

This work can be divided into three distinct experiments regarding the simulations that were carried out:

1st Experiment: We selected 20 complete datasets from different open source repositories and simulated the missing mechanisms using 9 univariate implementations (*univa*). Then, 7 well-known imputation techniques and other 2 imputation approaches based on SDAE are evaluated in terms of F-measure and R^2.

2nd Experiment: We increased the benchmark of datasets used in the first one, selecting another 13 complete datasets. In this case, we simulated each one of the three missing mechanisms using two different configurations (*univa* and *unifo*), which gives a total of 6 different implementations. After that, we impute the missing values using the same 7 widely-known imputation techniques used in the 1st experiment and we propose another 2 SDAE-based approaches for imputation. We evaluate the results with three different metrics: F-measure, R^2 and RMSE.

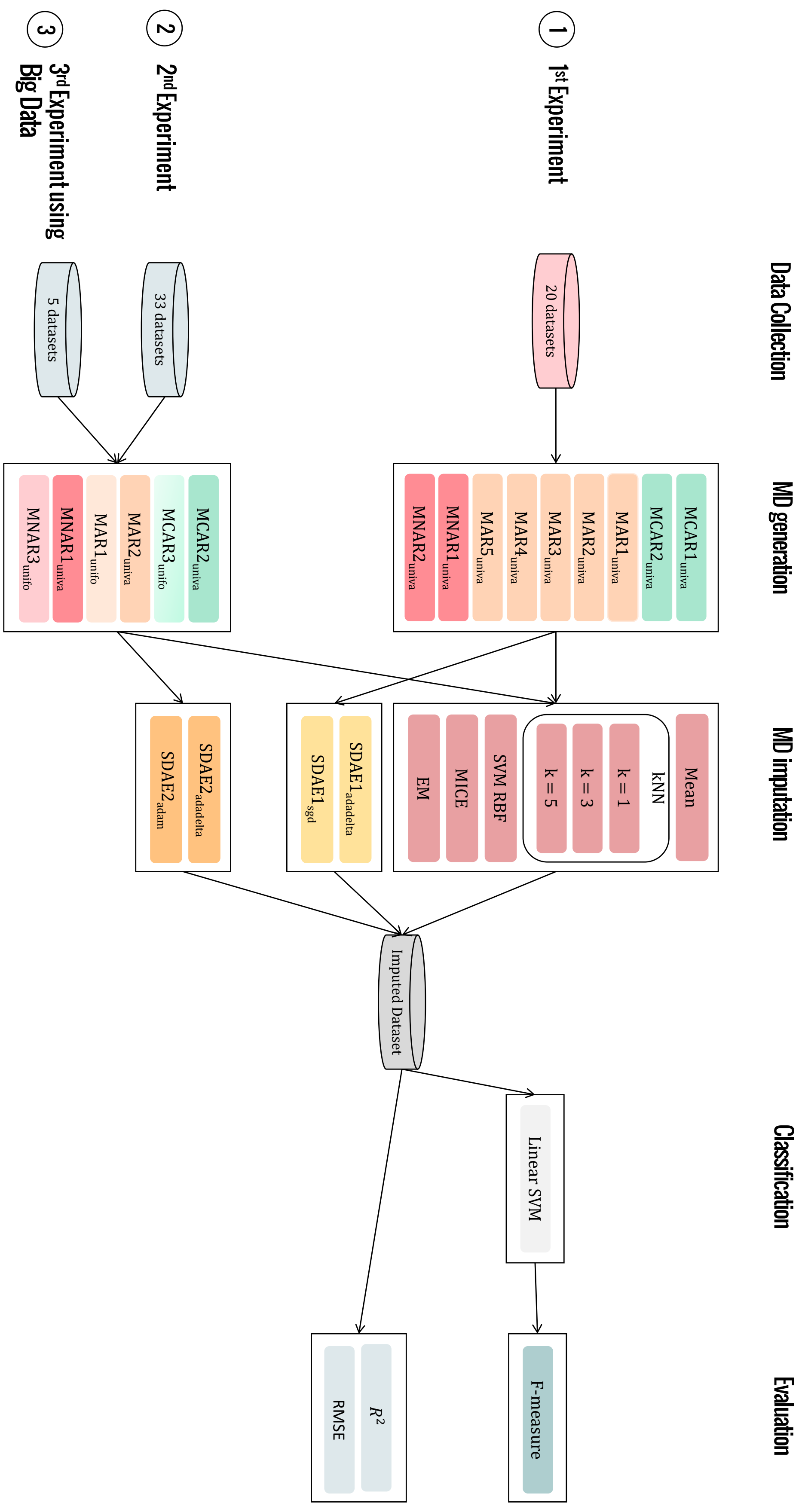


Figure 4.1: Detailed experimental setup.

3rd Experiment: This experiment is quite similar to the 2nd, where we explore datasets with different characteristics, namely, a higher number of samples.

4.1 Data Collection

To analyse the effect of different implementations of missing mechanisms on imputation methods, we selected several datasets attending to different contexts, sample sizes, number of features and types of features.

All datasets were publicly available and were obtained from the following repositories:

- UCI Machine Learning Repository – `http://archive.ics.uci.edu/ml`
- Knowledge Extraction based on Evolutionary Learning (KEEL) – `http://sci2s.ugr.es/keel/datasets.php`
- StatLib – `http://lib.stat.cmu.edu/datasets/`
- Kaggle – `http://www.kaggle.com/datasets/`
- Pattern Recognition and Neural Networks book [67] – `https://www.stats.ox.ac.uk/pub/PRNN/`

Since imputation experiments required complete data for evaluation, some of the original datasets were preprocessed in order to remove observations with missing values. In the case of multiclass datasets, they were modified in order to become binary [68], since we focus solely on binary classification problems. After preprocessing, all datasets are complete and binary. Their basic characteristics are presented in Table 4.1. Additional information about the datasets, such as their source and positive/negative classes is shown in Table B.1.

For the 1st and 2nd experiments, we chose standard datasets (low dimensionality and low sample size) so that we had a reasonable computational time. **lung-cancer** has the lowest number of observations (27) and **bankote** has the highest (1376). Regarding the number of features, **toy** has the lowest (2) while **lung-cancer** has the highest (56). In the 2nd experiment, we decided to remove **ecoli** dataset from the study since we found that it has a feature that takes the same value (0.5) for all observations, which makes the dataset a bit incoherent and may also create some conflicts during the synthetic generation of missing values – in particular, for *unifo* configurations.

For the 3rd experiment we selected 5 datasets with higher sample sizes. **thyroid1** and **thyroid2** have the lowest number of observations (7034) while **nursery2** has the highest (8586). Regarding the number of features, **nursery1** and **nursery2** have the lowest number (8) while the remaining datasets have 21.

4.2 Missing Data Generation

In this section, we will describe how we generated the synthetic missing values. Furthermore, we will introduce some intermediate results that we consider fundamental to establish some directions followed in this work. These results are related to a preliminary experiment that was performed in order to select the generation approaches that should be used in the remaining experiments.

As previously mentioned, several approaches to the synthetic generation of missing values can be found in the literature. We used *Python 3* to implement all the synthetic generation methods mentioned in Section 2.2.1. In Table 4.2 we present a summary of the characteristics of the generation configurations used in the different experiments.

Table 4.2: Configurations used for the synthetic missing data generation on the different experiments.

Experiment	**# Generation Methods**		**# Datasets**	**# Runs**	**MR**	**# Incomplete Datasets**
	Univariate	**Multivariate**				
1st	9	0	20	10	5 and 20%	3600
2nd	3	3	33	5	5, 10, 15, 20 and 40%.	4950
3rd			5			750

Synthetic generation of Missing Data in the 1st Experiment

In the 1st experiment, we study only univariate implementations (*univa*) for 2 different missing rates, 5% and 20%. The process of artificially generating missing values was performed 10 times (10 runs) for each dataset and for each implementation. As already mentioned in Section 2.2.1, the missing feature, x_{miss}, was the one most correlated with the class labels and the observed feature, x_{obs}, was the one most correlated with x_{miss}. The correlation between features is assessed using different coefficients depending on the type of features (Table 4.3). For example, if we want to measure the correlation between a nominal feature and the class labels (binary) we would use Cramer's V.

Table 4.1: Description of the datasets used in this study. The last columns show the number of features according to their type: C (continuous), O (ordinal), N (nominal) and B (binary).

	Dataset	**Context**	**Sample Size**	**No. Of Features**	**C**	**O**	**N**	**B**
1st and 2nd Experiments	australian	Credit card applications.	690	14	6	0	4	4
	banknote	Data extracted from banknote images.	1372	4	4	0	0	0
	biomed	Blood measurements database.	194	5	5	0	0	0
	breast-ljub	Breast cancer data.	277	9	0	5	1	3
	breast-tissue	Impedance measurements of tissue from the breast.	106	9	9	0	0	0
	cleveland	Heart disease database.	297	13	6	1	3	3
	crabs	Morphological features of crabs.	200	6	5	0	0	1
	dermatology1	Clinical features of erythema and scaling.	96	34	1	32	0	1
	ecoli	Measurements about the cell to predict the location site of proteins.	220	7	7	0	0	0
	glass1	Inflammation about 6 types of glass.	214	9	9	0	0	0
	heart-statlog	Heart disease database.	270	13	6	1	3	3
	iris	Iris plant database.	150	4	4	0	0	0
	kidney	Chronic kidney disease database.	158	24	11	0	3	10
	lung-cancer	Lung cancer database.	27	56	0	0	43	13
	lymphography	Lymphoma detection.	142	18	3	0	6	9
	postoperative	Patient features used to determine whether a patient should be moved from an area to another.	86	8	0	4	2	2
	saheart	South African heart database.	462	9	8	0	0	1
	urinary	Acute inflammation of urinary bladder database.	120	6	1	0	0	5
	wine1	Chemical analysis of wines.	130	13	13	0	0	0
	wpbc	Follow-up data for breast cancer cases (prognostic).	198	32	32	0	0	0
2nd Experiment	balancescale	Balance scale weight and distance database.	576	4	4	0	0	0
	bankrupcy	Qualitative parameters to predict bankrupcy.	250	6	0	0	6	6
	cmc	Contraceptive method choice database.	844	9	2	0	4	3
	dermatology2	Clinical features of erythema and scaling.	182	34	1	32	0	1
	edu-data1	Students' academic performance database.	269	16	4	0	6	6
	edu-data2		353	16	4	0	6	6
	glass2	Information about 6 types of glass.	214	9	9	0	0	0
	hcc-data-mortality	Clinical features of real patients diagnosed with Hepatocellular Carcinoma (HCC).	165	5	1	1	0	3
	hcc-data-survival		165	5	1	1	0	3
	hepato	Hepatobiliary disorders database.	302	9	9	0	0	0
	new-thyroid	Thyroid disease database.	65	5	5	0	0	0
	toy	Synthetic dataset composed of five Gaussian components.	1250	2	2	0	0	0
	wine2	Chemical analysis of wines.	107	13	13	0	0	0
	wine3		119	13	13	0	0	0
3rd Experiment	mushrooms	Mushrooms description regarding of physical characteristics.	8124	21	0	0	17	4
	nursery1	Ranking applications for nursery schools.	8364	8	0	0	7	1
	nursery2		8586	8	0	0	7	1
	thyroid1	Thyroid disease database - data from 10 different databases.	7034	21	6	0	0	15
	thyroid2		7034	21	6	0	0	15

So, for each dataset, x_{miss} and x_{obs} were the same regardless of the implementation used – Table C.1 shows the index of these features for each dataset. For this experiment we generate Missing Data (MD) using the following implementations: $MCAR1_{univa}$, $MCAR2_{univa}$, $MAR1_{univa}$, $MAR2_{univa}$, $MAR3_{univa}$, $MAR4_{univa}$, $MAR5_{univa}$, $MNAR1_{univa}$ and $MNAR2_{univa}$.

Table 4.3: Different correlation coefficients depending on the type of features (or labels). Adapted from [69].

	Continuous	**Ordinal**	**Nominal**	**Binary**
Continuous	Pearson R	Spearman rho	ETA	Point-Biserial
Ordinal	–	Spearman rho	ETA (with ranks in ordinal)	Rank Biserial
Nominal	–	–	Cramer's V	Cramer's V
Binary	–	–	–	Phi Coefficient

After the 1st experiment, we moved to the 2nd experiment, increasing the number of datasets as well as using 2 multiple imputation approaches based on SDAE. These changes would lead to an increase on the required computational time, so we changed the configuration of the synthetic generation as follows: we used only 6 different generation approaches, one for each mechanism (MCAR, MAR and MNAR) and for each type of implementation, *univa* and *unifo*; we only perform 5 runs for each dataset and implementation; furthermore, we use a larger set of MRs (5, 10, 15, 20 and 40%) in order to generalize the obtained results.

Synthetic generation of Missing Data in the 2nd and 3rd Experiments

In order to select the 6 generation approaches that would be used both in the 2nd and 3rd experiments, we performed a preliminary analysis of the effect of the different implementations in the well-known imputation techniques. For this, we generated MD using all the implementations described in Section 2.2.1: this MD generation was performed 10 times (10 runs) for all 33 datasets and for 5 different MRs (5, 10, 15, 20 and 40%). Then, the incomplete datasets were imputed using 7 well-known techniques: Mean Imputation (Meanimp), imputation with k-Nearest Neighbours (kNNimp) for k=1, 3 and 5, imputation with Support Vector Machines (SVMimp), MICE and Expectation-Maximization (EM). The results of this analysis correspond to the mean results for all the datasets and are shown both in terms of quality of imputation (R^2 and RMSE) and classification performance (F-measure): the values obtained for each evaluation metric and the ranks of the different imputation methods per generation approach are presented in the Appendices of this document – Tables E.1 and E.2 for F-measure under *univa* generation approaches; Tables E.3 and E.4 for R^2 and RMSE under *univa* generation approaches; Tables E.5 and E.6

for F-measure under *unifo* generation approaches; Tables E.7 and E.8 for R^2 and RMSE under *unifo* generation approaches. Next, we discuss the best approach to implement each configuration and missing mechanism.

- Univariate Implementations, *univa* (Tables E.1 and E.3)

  For MCAR mechanism, we chose $MCAR2_{univa}$ although the results in terms of imputation quality and classification performance are in most cases superior for $MCAR1_{univa}$. However, we believe it would not be correct to use $MCAR1_{univa}$ due to its limitation in the generation of the desired MR because of the use of Bernoulli distribution.

  To generate MD under MAR mechanism we use $MAR2_{univa}$ since this seems the best generation method regarding the quality of imputation. Although this approach does not guarantee the best results in terms of classification performance, it was chosen because of its simple implementation which also ensures the generation of the desired number of missing values.

  $MNAR2_{univa}$ was chosen since in most scenarios it was superior to $MNAR1_{univa}$, both in terms of quality of imputation and classification performance.

- Multivariate Implementations, *unifo* (Tables E.5 and E.7)

  We choose to use $MCAR3_{unifo}$ because besides being the MCAR implementation with the best results for most of the studied scenarios, it is also the most common implementation for this mechanism. Furthermore, the remaining MCAR implementations have the limitation of not being able to generate the desired MR.

  For MAR mechanism, we discarded $MAR2_{unifo}$ since it has the limitation of only being able to generate MR up to 25%. $MAR3_{unifo}$ is the one that achieves the best results for most studied scenarios, however, this implementation chooses only one observed feature, x_{obs}, to influence the MD generation process in the entire dataset. Since we do not consider this last approach the most exemplary of MAR $unifo$ implementations, we chose $MAR1_{unifo}$.

  We chose $MNAR3_{unifo}$ even though $MNAR2_{unifo}$ was the implementation that shows superiority in terms of imputation quality. This implementation was not chosen since the pattern it generates is somehow limited: if we define the number of missing features (nF) as the total number of features, this implementation will delete entire samples.

4.3 Missing Data Imputation

After the MD generation step, we move on to the imputation of the incomplete datasets. In the three experiments, we compared the performance of SDAE with 7 well-known imputation methods, namely: Meanimp, kNNimp for k=1, 3 and 5, SVMimp, MICE and EM (please refer to Section 2.3).

Next, we will refer to the details of the implementations used for the imputation methods, especially to the configurations of the SDAE – most of the methods were applied using open-source implementations.

Meanimp and kNNimp were implemented using *fancyimpute* [30] implementation in *Python*. For kNNimp we considered the euclidean distance and a set of closest neighbours $\{1, 3, 5\}$.

SVMimp was implemented using *scikit-learn*. The search for optimal parameters C and γ of the kernel was performed through a grid search for each dataset. Different ranges were tested: 10^{-2} to 10^{10} for C and 10^{-9} to 10^{3} for γ, both ranges increasing by a factor of 10 – these are considered suitable ranges, according to the documentation of *scikit-learn*. We performed 5 repetitions of a Holdout validation, for each combination of C and γ, using 80% of the data to train and 20% to test. Table D.1 shows the optimal values of C, γ and the average accuracy over the 5 repetitions, for all the datasets used in this study.

For MICE we used *fancyimpute* [30] implementation in *Python* with default settings. We perform 100 iterations of the method which means that the complete dataset was a result of 100 imputed datasets (multiple imputation). Finally, EM was implemented using the *Python* library *impyute* [37].

SDAE-based approaches

In this work we propose four different SDAE-based approaches for the complete reconstruction of missing data: two of them are used and evaluated in the 1st experiment while the remaining two are evaluated in the 2nd and 3rd experiments. All the models were implemented using Keras library with a Theano backend [70, 71]. We chose to use overcomplete representations since Gondara et al. [9] proved that overcomplete architectures provided better results. Furthermore, for all the proposed approaches we defined that our target would be the complete data, i.e. the data before the synthetic generation of Missing Value (MV).

In the 1st experiment, we explored 2 different approaches based on SDAE: $SDAE1_{adadelta}$ and $SDAE1_{sgd}$. Here, we use only complete samples to train the model while the incomplete ones are used as test set. The MVs on the test set are replaced by zero values, since SDAE does not accept "NaN" values. We also apply z-score (Equation 4.1) standardisation to the input data in order to have a faster convergence. The z-score of a value, x, is obtained by the following equation:

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

where μ and σ are the mean and standard deviation of the feature that we want to scale, respectively. After applying z-score standardisation, each dataset will have zero mean and unitary standard deviation.

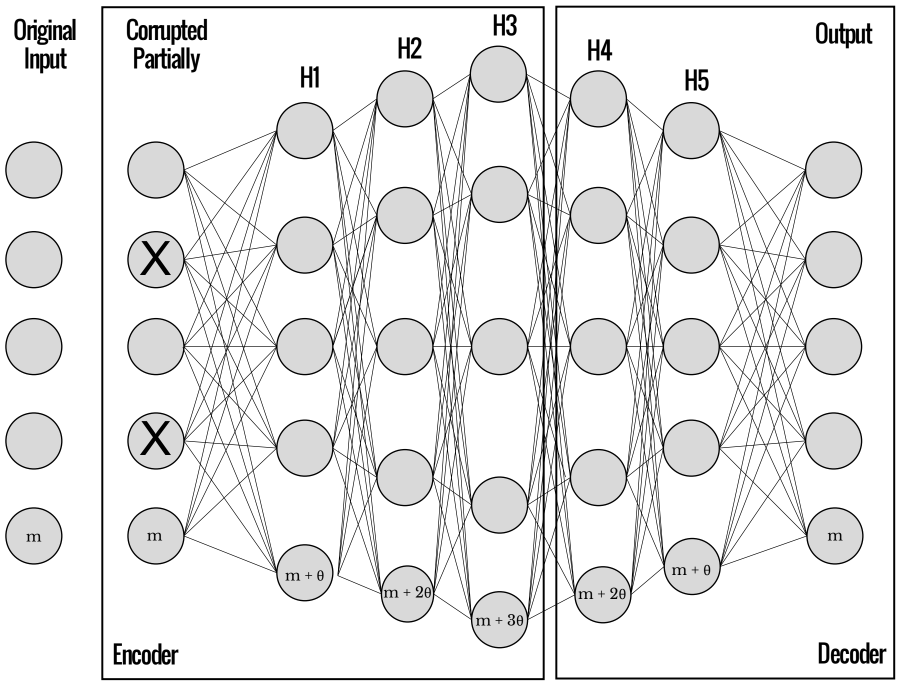


Figure 4.3: Our overcomplete SDAE general architecture composed of 5 hidden layers. Each successive hidden layer from the encoder has an increment in the number of nodes of $\theta = 7$. In the decoder, the number of nodes decreases symmetrically up to the original dimensions.

The architectures of $SDAE1_{adadelta}$ and $SDAE1_{sgd}$ are similar to the architecture proposed by Gondara et al. [9]. The models are composed by an input layer, 5 hidden layers and an output layer which form the encoder and the decoder (both constructed using regular densely-connected neural network layers) – Figure 4.3. The increment in the number of nodes for each successive hidden layer was set to 7, as it has proven to obtain good results in previous works [9]. The two models have an input dropout ratio of 30%. For both versions, $SDAE1_{adadelta}$ and $SDAE1_{sgd}$, we use the activation function Tanh for encoding and decoding layers. First, the SDAE are fed with the complete patterns to train, and after the training phase, the model is used to impute the incomplete patterns (from the test set).

The difference between $SDAE1_{adadelta}$ and $SDAE1_{sgd}$ is in the configuration of the training phase. $SDAE1_{sgd}$ is adapted from Gondara et al. [9], while for $SDAE1_{adadelta}$, we have decided to study a different optimisation function – Adadelta optimisation algorithm – since it avoids the difficulties of defining a proper learning rate [46]. Therefore, $SDAE1_{adadelta}$ is trained with 500 epochs using Adadelta optimisation algorithm [46] and mean squared error as loss function, while $SDAE1_{sgd}$ uses the Stochastic Gradient Descent (SGD) [44] with a time decay factor of 0.99 and Nesterov's accelerated gradient [72].

The major limitation of SDAE-based approaches found in the related works is that they report the results only for the test set. In practice, this means that we are only imputing the MD from the test set. So, it is assumed that the training group has no missing values and the only existent corruption is related to the training process of the Denoising Autoencoder (DAE). In the two approaches we have described earlier, we created the training sets only with complete data. Of course this option has a great limitation, since it is influenced by the percentage of missing values in the dataset. Moreover, $SDAE1_{adadelta}$ and $SDAE1_{sgd}$ would hardly be used for *unifo* implementations because in these cases there may not be any complete pattern to form the training set for the models.

For the 2nd and 3rd experiment we propose another two approaches based on SDAE: $SDAE2_{adadelta}$ and $SDAE2_{adam}$. With these two approaches, we seek to explore the effect of an SDAE architecture that uses a training set that, similar to the test set, is corrupted according to the missing mechanisms. For this, we propose a multiple imputation method based on SDAE that ensures that the entire dataset is imputed. These approaches include a pre-imputation step for which we use the well-known Mean/Mode imputation method as was done in Gondara et al. [9]. We also apply z-score (Equation 4.1) standardisation to the input data in order to have a faster convergence. Once again, these two architectures are similar to the one proposed by Gondara et al. [9] (Figure 4.3). For the encoding layers we chose Tanh as activation function due to its greater gradients [40]. ReLu was used as activation function in the decoding layers since Charte et al. [40] affirm that it is the best choice when combined with the mean squared error. The difference between $SDAE2_{adadelta}$ and $SDAE2_{adam}$ is the optimization algorithm - Adadelta optimisation algorithm [46] is used for $SDAE2_{adadelta}$ while Adam optimisation algorithm [48] is used $SDAE2_{adam}$. We chose Adam optimizer since Charte et al. [40] compared it with other SGD-based approaches (such as AdaGrad [45] and RMSProp [47]) and showed that Adam obtained the best results in terms of speed of convergence. Both models are trained with 100 epochs using mean squared error as loss function. Our models have an input dropout ratio of 50%, which means that half of the network's inputs are set to zero in each training batch. To prevent the training data from overfitting we add a regularization function named $L2$ [40].

Our imputation approaches based on $SDAE2_{adadelta}$ and $SDAE2_{adam}$ consider the creation of three different models (for three different training sets), for which three runs will be performed (multiple imputation). This approach is illustrated in Algorithm 20 and in Figure 4.4 and works as follows: (1) the instances of each dataset are divided into three equal-size sets (p = 3); (2) each set is used as test set, while the remaining two are used to feed the SDAE in the training phase; (3) 3 multiple runs (l = 3) will be performed for each one of these models; (4) the output mean of the three models is used to impute the unknown values of the test set.

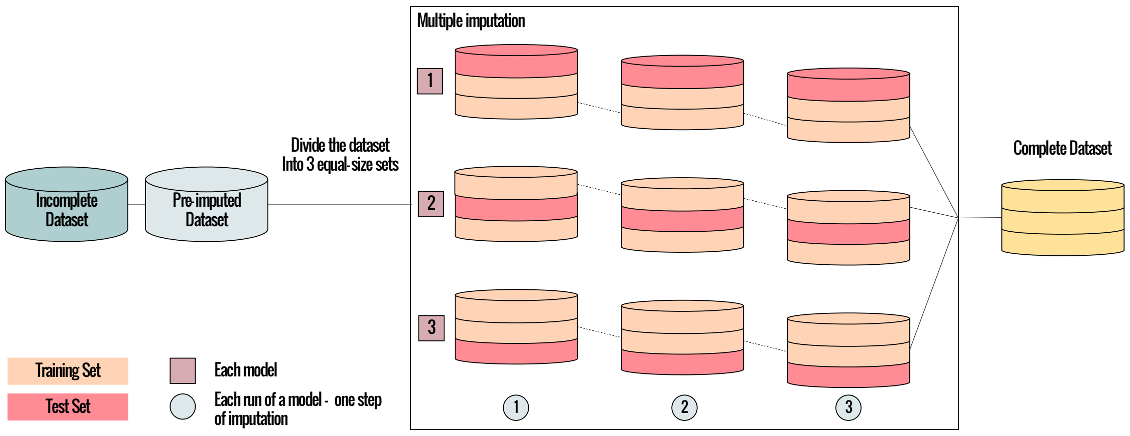


Figure 4.4: Multiple imputation using SDAE - data division for each model.

4.4 Classification and Evaluation Metrics

After the imputation step is concluded, we move towards the classification stage. The step of classification is common to all experiments. We perform classification with a SVM with linear kernel and considering a value of C = 1, as explained in Section 2.4.

Algorithm 20: Multiple imputation using SDAE – $SDAE2_{adadelta}$ and $SDAE2_{adam}$

```
Input :
        Pre-imputed dataset $X$, $p$ data partitions, $l$ multiple imputations
Output:
        Complete dataset
begin
    for $i$ in range(1, $p$) do
        Consider all partitions (except partition $i$) as training set
        Consider partition $i$ as test set
        for $j$ in range(1, $l$) do
            Perform dropout (50%) in training set
            Initialise the SDAE with random weights
            Fit the imputation model to the training set
            Apply the trained model to test set $i$ and save its imputed version $j$
return Complete dataset $X$
```

We evaluate two key performance requirements for imputation techniques: their efficiency on retrieving the true values in data (quality of imputation) [73] and their ability of providing quality data for classification [5]. The quality of imputation was assessed calculating two metrics: square of Pearson's Correlation Coefficient (Equation 2.15 which is used in all experiments), R^2, and RMSE (Equation 2.16 used for 2nd and 3rd experiment). The higher the value of R^2, the better the performance of the imputation method. For the RMSE, the closer to 0 the value is, the better the imputation quality.

The classification performance was assessed using F-measure which considers a harmonic mean of precision and recall [51]. The higher the value of F-measure, the better the classification performance. In order to evaluate the classification performance, we perform Holdout validation [50] which simply divides the dataset into two different sets, train and test (70% and 30%), and calculates the desired metric for the test set. This method has some limitations since the training set may not be representative of the whole data which can cause biased results [51]. Despite this shortcoming, we chose this method because any other would greatly increase the computational time of the simulations.

5

Experimental Results

In this chapter, we will report the results obtained from the three experiments described in Chapter 4. We intend to analyse the effect of the different MD implementations on several imputation methods and under various MRs. Furthermore, our main goal is to study the performance of imputation approaches based on SDAE, compared to other imputation methods from the state of the art, divided into three experiments:

- 1st Experiment: Assess the performance of SDAE-based methods for *univa* configurations;
- 2nd Experiment: Assess the performance of SDAE-based methods for both *univa* and *unifo* configurations;
- 3rd Experiment: Reassess the 2nd Experiment for datasets with higher sample sizes.

5.1 1st Experiment

The main goals of the 1st experiment are the following:

- Propose 2 imputation methods based on SDAE capable of reconstructing the incomplete data;
- Compare the performance of these methods with other approaches from the state of the art, in terms of classification performance and imputation quality;
- Study the effect of several *univa* generation methods on different imputation approaches.

Missing values were inserted at two MRs (5 and 20%) following 9 different scenarios ($MCAR1_{univa}$, $MCAR2_{univa}$, $MAR1_{univa}$, $MAR2_{univa}$, $MAR3_{univa}$, $MAR4_{univa}$,

$MAR5_{univa}$, $MNAR1_{univa}$ and $MNAR2_{univa}$) that are described in Section 2.2.1. Ten runs were performed for each missing generation, per dataset and missing rate (a total of 3600 incomplete datasets). Regarding the MRs, we chose to use 2 extreme values representing both scenarios, with a small amount of missing data (5%) and a large amount (20%). Furthermore, we use another 7 imputation techniques often explored in the literature, in order to compare their performance with our proposed models. Finally, we selected 20 complete datasets from different open source repositories (Table 4.1). This first experiment allows us to distinguish the effects of different *univa* implementations (for the same mechanism) on different imputation methods. Furthermore, this is an initial experiment used to explore basic aspects of SDAE under several generation methods.

Table 5.1: Results obtained from the 1st experiment: average results are shown regarding each missing data mechanism, implementation, metric (R^2 and F-measure) and missing rate. For each mechanism is presented the average of its implementations and the respective rank. The best results for each configuration are in boldface.

			Imputation Methods																	
	MR	**Mechanism**	$SDAE1_{adadelta}$		$SDAE1_{sgd}$		**Meanimp**		**kNNimp1**		**kNNimp3**		**kNNimp5**		**SVMimp**		**MICE**		**EM**	
		MCAR1	0.9091	5	0.8962	8	0.8986	7	0.9062	6	0.9182	4	**0.9213**	1	0.9210	2	0.8623	9	0.9205	3
		MCAR2	0.9446	5	0.9374	8	0.9383	7	0.9391	6	0.9511	3	0.9536	2	0.9510	4	0.9042	9	**0.9541**	1
		Mean	0.9276	5	0.9182	8	0.9193	7	0.9239	6	0.9356	4	0.9382	2	0.9363	3	0.8844	9	0.9384	1
		MAR1	0.9128	5	0.9014	7	0.9008	8	0.9051	6	0.9180	4	0.9217	2	**0.9223**	1	0.8669	9	0.9215	3
		MAR2	0.8814	6	0.8553	7	0.8495	8	0.8993	4	0.9100	2	**0.9107**	1	0.8843	5	0.8225	9	0.9043	3
		MAR3	0.8966	6	0.8802	7	0.8700	8	0.8996	5	0.9119	3	0.9146	2	0.9074	4	0.8403	9	**0.9156**	1
	5%	MAR4	0.9103	5	0.8937	7	0.8846	8	0.9023	6	0.9128	4	0.9171	3	**0.9221**	1	0.8569	9	0.9189	2
		MAR5	0.9107	5	0.9022	7	0.9006	8	0.9052	6	0.9188	4	0.9210	2	0.9205	3	0.8676	9	**0.9219**	1
		Mean	0.9038	5	0.8887	7	0.8844	8	0.9037	6	0.9155	3	0.9181	1	0.9133	4	0.8532	9	0.9170	2
		MNAR1	0.8637	6	0.8514	7	0.8371	8	0.8856	5	0.8876	4	0.8879	3	**0.9028**	1	0.8109	9	0.8902	2
		MNAR2	0.8078	6	0.7731	7	0.7554	8	0.8104	5	0.8309	2	0.8306	3	0.8284	4	0.7281	9	**0.8437**	1
		Mean	0.8358	6	0.8122	7	0.7962	8	0.8480	5	0.8593	3	0.8592	4	0.8656	2	0.7695	9	0.8670	1
R^2		MCAR1	0.8004	5	0.7658	7	0.7623	8	0.7936	6	0.8340	4	0.8416	2	**0.8439**	1	0.6540	9	0.8372	3
		MCAR2	0.8327	5	0.8012	7	0.8003	8	0.8147	6	0.8574	4	0.8649	2	0.8593	3	0.6644	9	**0.8677**	1
		Mean	0.8134	5	0.7784	7	0.7782	8	0.7990	6	0.8423	4	0.8501	1	0.8472	3	0.6565	9	0.8496	2
		MAR1	0.7913	5	0.7499	7	0.7459	8	0.7868	6	0.8266	4	0.8323	2	**0.8329**	1	0.6309	9	0.8287	3
		MAR2	0.7516	5	0.7126	7	0.6584	8	0.7172	6	0.7772	4	0.7865	3	**0.8055**	1	0.5766	9	0.7934	2
		MAR3	0.7485	6	0.6707	7	0.6604	8	0.7529	5	0.8071	4	0.8146	3	0.8160	2	0.5586	9	**0.8189**	1
	20%	MAR4	0.8115	4	0.7513	7	0.7052	8	0.7795	6	0.8188	3	0.8262	2	0.8098	5	0.6175	9	**0.8288**	1
		MAR5	0.7981	5	0.7560	7	0.7513	8	0.7809	6	0.8235	4	0.8325	3	**0.8417**	1	0.6351	9	0.8331	2
		Mean	0.7832	5	0.7333	7	0.7127	8	0.7669	6	0.8136	4	0.8214	3	0.8239	1	0.6092	9	0.8228	2
		MNAR1	**0.7407**	1	0.6927	6	0.5690	8	0.6722	7	0.7019	5	0.7098	4	0.7253	3	0.5283	9	0.7405	2
		MNAR2	**0.6830**	1	0.6128	6	0.4749	8	0.6145	5	0.6322	4	0.6339	3	0.6035	7	0.4555	9	0.6598	2
		Mean	0.7119	1	0.6528	6	0.5220	8	0.6433	7	0.6670	4	0.6718	3	0.6644	5	0.4919	9	0.7002	2
		MCAR1	0.7372	6	0.7362	8	0.7359	9	0.7373	5	0.7379	4	0.7380	3	**0.7407**	1	0.7370	7	0.7396	2
		MCAR2	0.7372	7	0.7375	5	0.7370	8	0.7389	2	0.7383	3	0.7382	4	**0.7396**	1	0.7365	9	0.7374	6
		Mean	0.7368	6	0.7368	7	0.7366	8	0.7373	5	0.7374	4	0.7377	3	0.7393	1	0.7363	9	0.7383	2
		MAR1	0.7344	6	0.7327	9	0.7339	7	0.7363	5	0.7375	3	0.7384	2	**0.7390**	1	0.7329	8	0.7368	4
		MAR2	0.7374	4	0.7349	7	0.7324	9	0.7369	5	0.7382	3	0.7365	6	0.7413	2	0.7338	8	**0.7415**	1
		MAR3	0.7367	8	0.7369	7	0.7373	6	0.7377	5	0.7382	2.5	0.7382	2.5	**0.7385**	1	0.7345	9	0.7380	4
	5%	MAR4	0.7383	2	0.7383	5	**0.7385**	1	0.7379	7	0.7383	3.5	0.7383	3.5	0.7381	6	0.7373	9	0.7377	8
		MAR5	0.7366	6	0.7362	8	0.7366	7	0.7382	4	0.7383	3	0.7384	2	**0.7396**	1	0.7344	9	0.7366	5
		Mean	0.7365	6	0.7356	7	0.7355	8	0.7373	5	0.7379	3	0.7377	4	0.7392	1	0.7344	9	0.7380	2
		MNAR1	0.7311	8	0.7343	5	0.7316	7	0.7385	2	0.7367	3	0.7326	6	**0.7393**	1	0.7299	9	0.7362	4
F-measure		MNAR2	0.7295	6	0.7279	8	0.7272	9	0.7346	5	0.7359	2	**0.7362**	1	0.7350	4	0.7286	7	0.7356	3
		Mean	0.7303	7	0.7311	6	0.7294	8	0.7366	2	0.7363	3	0.7344	5	0.7372	1	0.7292	9	0.7359	4
		MCAR1	0.7267	8	0.7263	9	0.7274	6	0.7293	5	0.7315	3	0.7328	2	**0.7411**	1	0.7268	7	0.7308	4
		MCAR2	0.7321	8	0.7334	5	0.7333	6	0.7358	3	0.7349	5	0.7355	4	**0.7388**	1	0.7299	9	0.7329	7
		Mean	0.7307	8	0.7316	6	0.7315	7	0.7335	4	0.7343	3	0.7352	2	0.7395	1	0.7279	9	0.7334	5
		MAR1	0.7247	6	0.7247	7	0.7223	9	0.7307	3	0.7311	2	0.7305	4	**0.7396**	1	0.7223	8	0.7295	5
		MAR2	0.7355	6	0.7332	7	0.7366	4	0.7313	8	0.7408	2	**0.7409**	1	0.7391	3	0.7270	9	0.7365	5
		MAR3	0.7316	6	0.7304	7	0.7331	5	0.7301	8	0.7344	4	0.7380	2	**0.7384**	1	0.7262	9	0.7366	3
	20%	MAR4	0.7297	6	0.7280	7	0.7267	9	0.7341	4	0.7350	3	**0.7357**	1	0.7355	2	0.7278	8	0.7306	5
		MAR5	0.7281	6	0.7269	7	0.7256	8	0.7316	5	0.7343	3	0.7352	2	**0.7413**	1	0.7229	9	0.7321	4
		Mean	0.7293	6	0.7278	8	0.7281	7	0.7312	5	0.7346	3	0.7355	2	0.7391	1	0.7242	9	0.7322	4
		MNAR1	**0.7298**	1	0.7237	5	0.7205	9	0.7225	7	0.7253	3	0.7246	4	0.7255	2	0.7222	8	0.7230	6
		MNAR2	**0.7275**	1	0.7249	3	0.7170	8	0.7160	9	0.7195	7	0.7215	5	0.7260	2	0.7200	6	0.7229	4
		Mean	0.7287	1	0.7243	3	0.7188	9	0.7192	8	0.7224	6	0.7231	4	0.7257	2	0.7211	7	0.7230	5

Table 5.1 and Figures 5.1 and 5.2 present the average results obtained for each dataset. The results are divided by metric (F-measure and R^2), missing mechanism (MCAR, MAR and MNAR) and missing rate (5 and 20%). Table 5.1 further distinguishes between different implementations of the missing mechanisms.

Overall, the increase of the missing rate leads to a decrease in the performance of classifiers (F-measure) and in the quality of imputation (R^2). It is also noticeable that the values of both metrics decrease from MCAR to MAR and from MAR to MNAR for most imputation scenarios. We now thoroughly analyse the results obtained regarding the quality of imputation and impact on classification results individually.

Quality of Imputation ($\mathbf{R^2}$):

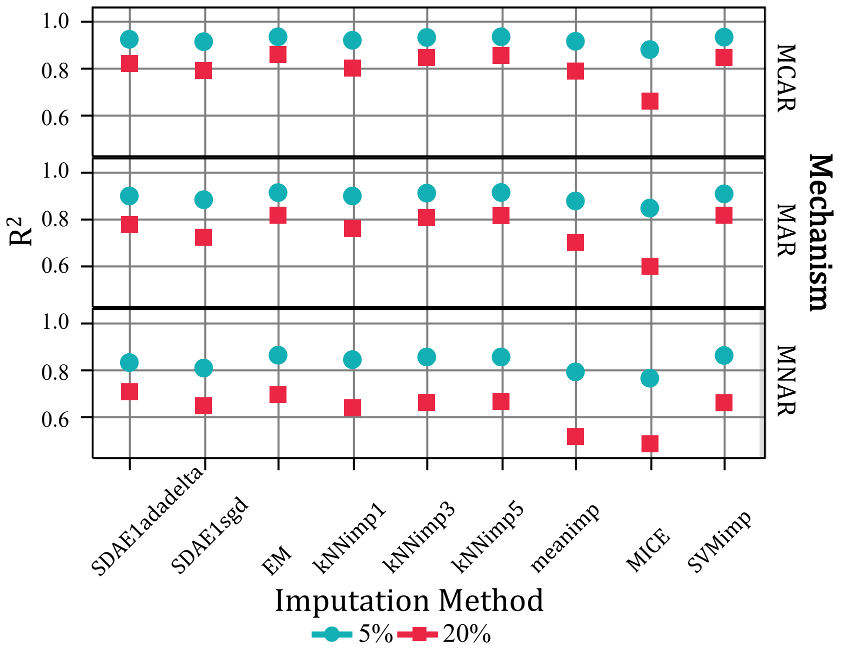


Figure 5.1: Average R^2 values for all datasets, considering the different imputation methods under the three missing mechanisms.

For MCAR mechanisms and considering a 5% missing rate, kNNimp5 and EM were the best approaches for $MCAR1_{univa}$ and $MCAR2_{univa}$, respectively; while for a 20% missing rate, SVMimp and EM proved to be the best imputation techniques for $MCAR1_{univa}$ and $MCAR2_{univa}$, respectively. For both MRs, $SDAE1_{adadelta}$ is the 5th best approach, with differences from the best method of 0.0108 (MR 5%) and 0.0367 (MR 20%). $SDAE1_{sgd}$ is the 8th (MR 5%) and 7th (MR 20%) with differences from the best method of 0.0202 and 0.0717, respectively.

For MAR mechanism there are three winner approaches when missing rate is 5%: SVMimp (for $MAR1_{univa}$ and $MAR4_{univa}$), EM (for $MAR3_{univa}$ and $MAR5_{univa}$) and kNNimp5 (for $MAR2_{univa}$). For a missing rate of 20%, two winner approaches arise: SVMimp (for $MAR1_{univa}$, $MAR2_{univa}$ and $MAR5_{univa}$) and EM ($MAR3_{univa}$ and $MAR4_{univa}$). Regardless of the MR, $SDAE1_{adadelta}$ and $SDAE1_{sgd}$ are the 5th and 7th best imputation approaches – $SDAE1_{adadelta}$ differs from the best method of 0.0144 (MR of 5%) and 0.0407 (MR of 20%); $SDAE1_{sgd}$ 0.0295 (MR of 5%) and 0.0906 (MR of 20%) from the 1st approach.

Regarding MNAR mechanism, SVMimp and EM show the best results for $MNAR1_{univa}$ and $MNAR2_{univa}$, respectively, considering a 5% missing rate. $SDAE1_{adadelta}$ (0.0312) and $SDAE1_{sgd}$ (0.0547) are ranked with 5 and 7, respectively, and the values in parentheses indicate the difference from the best approach. For a higher MR (20%), $SDAE1_{adadelta}$ seems to be the best imputation approach – $SDAE1_{adadelta}$ reaches an average value of 0.71187 followed by EM that achieves 0.70016.

Impact on classification (F-measure):

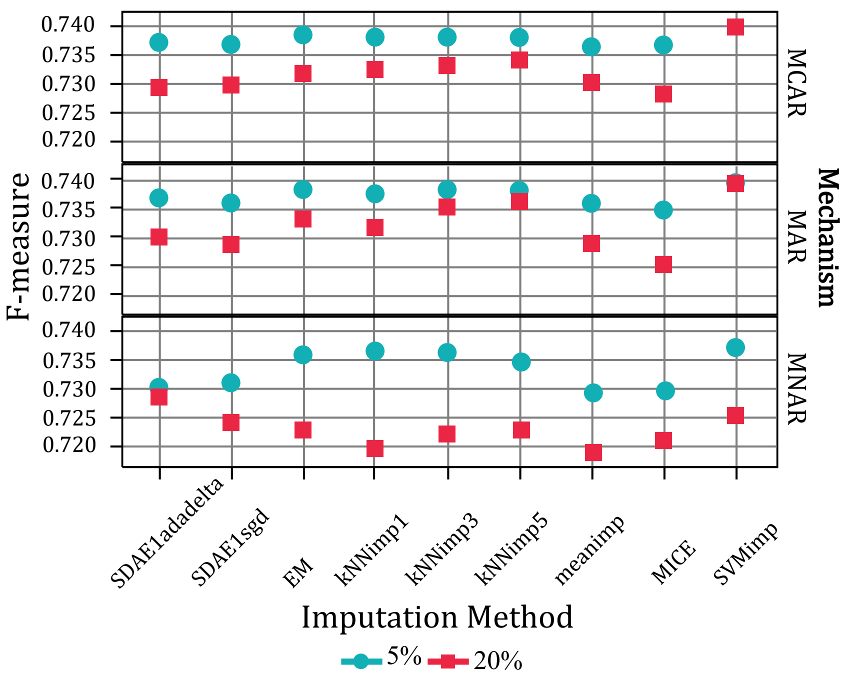


Figure 5.2: Average F-measure values for all datasets, considering the different imputation methods under the three missing mechanisms.

For MCAR mechanisms, SVMimp seems to be the best imputation method in terms of classification performance (F-measure), regardless of the missing rate. $SDAE1_{adadelta}$ is ranked with 6 (5%) and 8 (20%), differing from the best method of 0.00254 and 0.00880, respectively. $SDAE1_{sgd}$ is the 7th and the 6th best method under the 2 MRs. Here, the differences between the imputation methods are smaller than for R^2 – the differences between the SDAE-based approach and the best method for each scenario are 0.00255 and 0.00786, respectively.

Considering MAR mechanism, there are three winner approaches for 5% of MR: Meanimp (for $MAR4_{univa}$), SVMimp (for $MAR1_{univa}$, $MAR3_{univa}$ and $MAR5_{univa}$) and EM (for $MAR2_{univa}$). For a higher MR (20%), SVMimp (for $MAR1_{univa}$, $MAR3_{univa}$ and $MAR5_{univa}$) and kNNimp5 ($MAR2_{univa}$ and $MAR4_{univa}$) proved to be the best approaches. $SDAE1_{adadelta}$ (5% – 0.0028; 20% – 0.0098) is ranked as 6 best method for both MRs. $SDAE1_{sgd}$ is the 7th (MR 5%) and 8th (MR 5%) best method differing from the 1st of 0.0079 and 0.0114, respectively.

For MNAR mechanism and 5% of missingness there are two winner approaches: SVMimp (for $MNAR1_{univa}$) and kNN5imp (for $MNAR2_{univa}$). For the same MR, $SDAE1_{adadelta}$ (0.0069) and $SDAE1_{sgd}$ (0.0061) are ranked with 7 and 6, respectively. When the missing rate increases (20%), $SDAE1_{adadelta}$ and $SDAE1_{sgd}$ are among the top 3 approaches: $SDAE1_{adadelta}$ surpasses all the remaining methods, while $SDAE1_{sgd}$ is the third best imputation method, falling just behind kNNimp5.

Conclusions

In this 1st experiment, we propose 2 SDAE-based approaches capable of imputing data that are missing. The results show that our proposed approaches surpass other imputation methods from the state of the art, for some combinations of missing mechanisms and MRs. Despite this, it is important to note that our models have some limitations since we assume that there is enough complete data to be used in the training phase: in other words, our training set is composed of complete observations while the incomplete ones belong to the test set.

These results show that our model ($SDAE1_{adadelta}$) which uses Adadelta as optimization algorithm surpasses the results obtained by $SDAE1_{sgd}$ proposed in Gondara et al. [9]. Furthermore, the results obtained for MNAR mechanisms under a high MR, show the superiority of SDAE over the remaining state-of-the-art methods, indicating that autoencoders might be a feasible approach for this scenario, which is a relevant insight for the research community, since MNAR is a non-ignorable mechanism and the most problematic to impute in an unbiased way.

Regarding both metrics (R^2 and F-measure), the top 3 imputation methods per mechanism are always the same except for MNAR with 20% of missing values. EM, kNNimp5 and SVMimp are the approaches that achieve the best results for MAR and MCAR mechanisms, and MNAR seems to be the only mechanism which influences the choice of imputation method.

In summary, the conclusions of the 1st experiment are the following:

- Our experiments show that under MCAR and MAR mechanisms there are three imputation methods that perform well: EM, kNNimp5 and SVMimp. For these scenarios, $SDAE1_{adadelta}$ and $SDAE2_{adam}$ are ranked between 5 and 8 position – $SDAE1_{adadelta}$ seems to be superior to $SDAE1_{sgd}$;
- For MNAR mechanism, the imputation technique based on overcomplete SDAE – $SDAE1_{adadelta}$ – outperforms well-known imputation methods for high rates of missing data (20%), both in terms of classification performance (F-measure) and quality of imputation (R^2);

5.2 2nd Experiment

The goals of this experiment are:

- Propose another 2 SDAE architectures to recover missing data;
- Propose a multiple imputation algorithm capable of imputing missing data regardless of the MR, using the 2 SDAE architectures mentioned above;
- Compare the performance of our proposed models with imputation methods from the state of the art, under several missing rates;
- Study the effect of the different generation methods on several imputation methods.

Before we started a new experiment, it was necessary to select the synthetic generation methods that would be used from then on. If we apply all the generation methods to the datasets, we would need a lot of computational time. So, in order to reduce the computational time, we selected a set of generation methods. The aim of this preliminary study was to investigate the generation methods that led to a better performance of the imputation methods. In addition to the performance of the imputation methods, we also took into consideration some limitations of the generation methods that were already known. At the end, we chose to use a generation method for each mechanism (MCAR, MAR and MNAR) and for each type of implementation (*univa* and *unifo*) – $MCAR2_{univa}$, $MCAR3_{unifo}$, $MAR2_{univa}$, $MAR1_{unifo}$, $MNAR1_{univa}$ and $MNAR3_{unifo}$. This analysis can be found in Section 4.2.

In brief, we performed a comparative study on the effect of the various generation methods described in Section 2.2.1 on 7 well-known imputation methods from the state of the art (Meanimp, kNNimp for k=1,3 and 5, SVMimp, MICE and EM).

This 2nd experiment also requires a missing value generation phase (where 5 runs were performed) followed by imputation and classification. Then, we evaluated both the imputation quality and its impact on classification. The results are divided by metric (F-measure, R^2 and RMSE), missing mechanism (MCAR, MAR and MNAR), type of configuration (*univa* and *unifo*) and missing rate (5, 10, 15, 20 and 40%). We evaluate 9 different imputation techniques including 2 SDAE-based approaches for a total of 4950 incomplete datasets.

Table 5.2 presents the average results obtained for all the 33 complete datasets used in this study (Table 4.1). As in the 1st experiment, the increase in the missing rate leads to a decrease in the classification performance (F-measure) and the quality of imputation (R^2 and RMSE). Moreover, we evaluate 2 important perspectives from the missing data problem – imputation quality and classification performance – the results obtained for these 2 perspectives do not match. Next, we provide a detailed analysis of the results obtained considering imputation quality and classification performance.

Quality of Imputation ($\mathbf{R^2}$ and RMSE):

For *univa* configurations, $SDAE2_{adam}$ proved to be the best approach for all missing mechanisms, regardless of the missing rate. $SDAE2_{adadelta}$ was ranked as the 6th or 7th best method, in terms of R^2, and regarding RMSE metric was considered to be 5th or 6th best method.

For the *unifo* configurations, MICE is the best imputation method for MCAR missing mechanisms for all the MRs while the SDAE-based approaches are ranked between 2nd and 6th positions. Considering MAR missing mechanism, MICE is also the best method in most of the scenarios: for a higher MR (40%), $SDAE2_{adam}$ seems to be the best approach in terms of RMSE. In the case of MNAR mechanism and for lower MRs (5% and 10%), MICE is also the best approach. However, for higher MRs, the SDAE-based approaches guarantee a better imputation quality – $SDAE2_{adadelta}$ is the best method under 15% and 40% MRs while $SDAE2_{adam}$ seems to be superior for 20%.

Table 5.2: Results obtained from the 2nd experiment by imputation method: average F-measure, R^2 and RMSE are shown regarding each missing data mechanism, configuration, metric and missing rate. The best results for each configuration are marked in bold. Pink and blue values indicate whether $SDAE2_{adadelta}$ or $SDAE2_{adam}$ belong to the top 3 of the best imputation approaches, respectively.

		Classification Performance												Imputation Quality																							
		F-measure												**R^2**												**RMSE**											
		Univa						**Unifo**						**Univa**						**Unifo**						**Univa**						**Unifo**					
MR	**Methods**	**MCAR2**		**MAR2**		**MNAR1**		**MCAR3**		**MAR1**		**MNAR3**		**MCAR2**		**MAR2**		**MNAR1**		**MCAR3**		**MAR1**		**MNAR3**		**MCAR2**		**MAR2**		**MNAR1**		**MCAR3**		**MAR1**		**MNAR3**	
	meanimp	0.7626	8	0.7648	8	0.7628	8	0.7593	7	0.7675	3	0.7759	6	0.9518	7	0.9480	7	0.8702	8	0.9415	6	0.9248	8	0.8675	5	0.2202	7	0.2291	7	0.3691	7	0.2206	6	0.2613	7	0.3621	5
	kNNimp1	0.7630	5	0.7671	4	0.7673	3	0.7587	8	0.7630	9	0.7535	9	0.9541	5	0.9538	5	0.9043	6	0.9370	7	0.9360	4	0.8592	8	0.2180	5	0.2187	5	0.3169	5	0.2339	7	0.2391	4	0.3748	7
	kNNimp3	0.7642	4	0.7680	2	0.7679	2	0.7646	4	0.7672	5	0.7765	2	0.9679	4	0.9655	4	0.9180	4	0.9510	3	0.9492	3	0.8733	3	0.1802	4	0.1872	4	0.2932	4	0.1986	3	0.2050	3	0.3518	3
	kNNimp5	0.7645	3	0.7672	3	0.7654	4	0.7671	2	0.7674	4	0.7763	5	0.9701	3	0.9676	3	0.9215	3	0.9535	2	0.9516	2	0.8772	2	0.1736	3	0.1813	3	0.2869	3	0.1917	2	0.1985	2	0.3451	2
5%	SVMimp	**0.7676**	1	**0.7693**	1	**0.7715**	1	**0.7686**	1	0.7676	2	**0.7871**	1	0.9468	8	0.9428	8	0.9069	5	0.9301	8	0.9296	5	0.8635	7	0.4918	9	0.5857	9	0.5622	9	0.5094	9	0.5285	9	0.6300	9
	EM	0.7591	9	0.7634	9	0.7618	9	0.7460	9	0.7648	8	0.7654	8	0.9140	9	0.9132	9	0.8431	9	0.9055	9	0.8954	9	0.8368	9	0.2979	8	0.2998	8	0.4064	8	0.2947	8	0.3173	8	0.4016	8
	MICE	0.7656	2	0.7665	5	0.7644	5	0.7659	3	**0.7692**	1	0.7751	7	0.9713	2	0.9706	2	0.9249	2	**0.9578**	1	**0.9556**	1	**0.8976**	1	0.1701	2	0.1736	2	0.2805	2	**0.1806**	1	**0.1887**	1	**0.3143**	1
	$SDAE2_{adadelta}$	0.7627	7	0.7651	6	0.7630	6	0.7595	6	0.7666	6	0.7763	4	0.9523	6	0.9500	6	0.8710	7	0.9416	4	0.9279	6	0.8686	4	0.2191	6	0.2245	6	0.3679	6	0.2202	4	0.2547	5	0.3605	4
	$SDAE2_{adam}$	0.7629	6	0.7651	7	0.7628	7	0.7626	5	0.7665	7	0.7764	3	**0.9835**	1	**0.9830**	1	**0.9699**	1	0.9415	5	0.9278	7	0.8665	6	**0.0785**	1	**0.0816**	1	**0.1439**	1	0.2203	5	0.2549	6	0.3634	6
	meanimp	0.7618	8	0.7631	8	0.7592	5	0.7490	6	0.7676	4	0.7682	4	0.9064	8	0.8958	8	0.7741	8	0.8915	6	0.8761	8	0.7781	4	0.3070	6	0.3255	7	0.4997	7	0.3144	5	0.3530	6	0.4891	4
	kNNimp1	0.7646	5	0.7656	4	0.7612	4	0.7494	5	0.7626	8	0.7614	6	0.9091	6	0.9065	6	0.8270	6	0.8796	8	0.8832	7	0.7476	8	0.3104	7	0.3155	5	0.4362	5	0.3420	7	0.3536	7	0.5268	7
	kNNimp3	**0.7677**	1	0.7664	3	0.7615	3	0.7538	4	0.7685	2	0.7563	8	0.9355	4	0.9263	4	0.8570	4	0.9076	3	0.9127	3	0.7741	6	0.2565	4	0.2760	4	0.3960	4	0.2897	3	0.2859	3	0.4949	6
	kNNimp5	0.7677	2	0.7674	2	**0.7622**	1	0.7558	3	0.7668	5	0.7602	7	0.9397	3	0.9310	3	0.8601	3	0.9130	2	0.9181	2	0.7792	3	0.2474	3	0.2664	3	0.3919	3	0.2795	2	0.2749	2	0.4878	3
10%	SVMimp	0.7668	3	**0.7702**	1	0.7620	2	**0.7655**	1	0.7679	3	**0.7801**	1	0.9182	5	0.9133	5	0.8456	5	0.8880	7	0.8906	4	0.7693	7	0.6793	9	0.6784	9	0.6194	9	0.7005	9	0.5958	9	0.8301	9
	EM	0.7592	9	0.7592	9	0.7589	7	0.7333	9	0.7557	9	0.7483	9	0.8355	9	0.8251	9	0.7345	9	0.8230	9	0.8257	9	0.7276	9	0.4165	8	0.4299	8	0.5411	8	0.4187	8	0.4285	8	0.5384	8
	MICE	0.7661	4	0.7652	5	0.7583	9	0.7586	2	**0.7690**	1	0.7693	3	0.9418	2	0.9400	2	0.8692	2	**0.9238**	1	**0.9247**	1	**0.8172**	1	0.2435	2	0.2477	2	0.3786	2	**0.2599**	1	**0.2631**	1	**0.4467**	1
	$SDAE2_{adadelta}$	0.7625	6	0.7646	6	0.7589	6	0.7485	7	0.7654	7	0.7676	5	0.9072	7	0.9002	7	0.7956	7	0.8915	5	0.8886	5	0.7745	5	0.3057	5	0.3178	6	0.4755	6	0.3144	6	0.3314	5	0.4933	5
	$SDAE2_{adam}$	0.7625	7	0.7639	7	0.7587	8	0.7473	8	0.7659	6	0.7699	2	**0.9779**	1	**0.9770**	1	**0.9593**	1	0.8916	4	0.8886	6	0.7950	2	**0.1089**	1	**0.1132**	1	**0.1833**	1	0.3142	4	0.3314	4	0.4697	2
	meanimp	0.7589	6	0.7581	8	0.7597	7	0.7381	6	0.7514	6	0.7451	5	0.8464	8	0.8490	8	0.6840	8	0.8404	4	0.8227	8	0.7025	5	0.3934	7	0.3937	7	0.6075	7	0.3879	5	0.4381	7	0.5849	4
	kNNimp1	0.7624	5	0.7635	4	0.7612	3	0.7335	8	0.7651	2	0.7205	9	0.8631	6	0.8658	6	0.7609	6	0.8131	8	0.8374	7	0.6442	9	0.3843	5	0.3822	6	0.5259	5	0.4365	7	0.4195	6	0.6534	8
	kNNimp3	0.7655	2	0.7647	2	0.7604	4	0.7451	4	0.7531	5	0.7324	7	0.8962	4	0.9024	4	0.7998	4	0.8567	3	0.8634	3	0.6781	7	0.3268	4	0.3168	4	0.4802	4	0.3698	3	0.3721	3	0.6162	6
	kNNimp5	0.7647	3	0.7643	3	0.7593	8	0.7503	3	0.7509	7	0.7374	6	0.9038	3	0.9076	3	0.8081	3	0.8663	2	0.8700	2	0.6840	6	0.3136	3	0.3074	3	0.4707	3	0.3551	2	0.3624	2	0.6088	5
15%	SVMimp	**0.7691**	1	**0.7678**	1	**0.7658**	1	**0.7672**	1	**0.7789**	1	**0.7872**	1	0.8870	5	0.8900	5	0.7908	5	0.8396	7	0.8389	6	0.7062	4	0.9742	9	0.7884	9	0.8418	9	0.8628	9	0.8873	9	1.0848	9
	EM	0.7548	9	0.7539	9	0.7582	9	0.7085	9	0.7332	9	0.7230	8	0.7480	9	0.7535	9	0.6260	9	0.7416	9	0.7576	9	0.6452	8	0.5189	8	0.5154	8	0.6584	8	0.5157	8	0.5229	8	0.6311	7
	MICE	0.7630	4	0.7607	5	0.7612	2	0.7528	2	0.7504	8	0.7611	3	0.9090	2	0.9107	2	0.8249	2	**0.8866**	1	**0.8828**	1	**0.7439**	3	0.3054	2	0.3021	2	0.4502	2	**0.3262**	1	**0.3494**	1	0.5404	3
	$SDAE2_{adadelta}$	0.7585	8	0.7583	7	0.7602	5	0.7378	7	0.7561	3	0.7633	2	0.8483	7	0.8579	7	0.7445	7	0.8399	6	0.8507	4	0.7556	1	0.3907	6	0.3802	5	0.5482	6	0.3886	6	0.3950	4	**0.5305**	1
	$SDAE2_{adam}$	0.7587	7	0.7597	6	0.7598	6	0.7383	5	0.7550	4	0.7609	4	**0.9700**	1	**0.9708**	1	**0.9519**	1	0.8403	5	0.8506	5	0.7509	2	**0.1407**	1	**0.1383**	1	**0.2105**	1	0.3879	4	0.3951	5	0.5355	2
	meanimp	0.7587	6	0.7573	8	0.7553	7	0.7219	6	0.7335	8	0.7317	5	0.8009	8	0.7960	8	0.5962	8	0.7907	6	0.7742	8	0.6315	5	0.4479	7	0.4595	7	0.7067	7	0.4479	5	0.5094	7	0.6700	4
	kNNimp1	0.7554	8	0.7583	5	0.7564	5	0.7185	8	0.7408	4	0.7062	8	0.8208	6	0.8256	6	0.7067	7	0.7534	8	0.7834	7	0.5546	9	0.4438	5	0.4386	5	0.5979	5	0.5094	7	0.5077	6	0.7541	8
	kNNimp3	0.7609	3	0.7607	4	0.7570	4	0.7381	3	0.7414	3	0.7228	7	0.8660	4	0.8637	4	0.7363	4	0.8069	3	0.8225	3	0.5881	7	0.3717	4	0.3759	4	0.5686	4	0.4346	3	0.4359	3	0.7185	7
	kNNimp5	0.7613	2	0.7620	3	0.7556	6	0.7380	4	0.7395	5	0.7296	6	0.8747	3	0.8728	3	0.7446	3	0.8187	2	0.8310	2	0.5948	6	0.3581	3	0.3618	3	0.5613	3	0.4189	2	0.4243	2	0.7114	6
20%	SVMimp	**0.7661**	1	**0.7714**	1	**0.7627**	1	**0.7643**	1	**0.7582**	1	**0.7896**	1	0.8576	5	0.8587	5	0.7314	5	0.7922	4	0.7959	6	0.6418	4	1.1028	9	0.9814	9	0.9409	9	1.0050	9	1.5731	9	1.2533	9
	EM	0.7547	9	0.7533	9	0.7536	8	0.6861	9	0.7146	9	0.6976	9	0.6729	9	0.6759	9	0.5514	9	0.6672	9	0.7030	9	0.5731	8	0.5966	8	0.5974	8	0.7395	8	0.5938	8	0.5950	8	0.7101	5
	MICE	0.7599	4	0.7632	2	0.7515	9	0.7431	2	0.7425	2	0.7499	4	0.8806	2	0.8799	2	0.7671	2	**0.8497**	1	**0.8496**	1	0.6712	3	0.3484	2	0.3511	2	0.5329	2	**0.3806**	1	**0.4009**	1	0.6326	3
	$SDAE2_{adadelta}$	0.7589	5	0.7577	6	0.7591	2	0.7210	7	0.7363	7	0.7515	3	0.8034	7	0.8090	7	0.7224	6	0.7903	7	0.8178	5	0.6977	2	0.4448	6	0.4416	6	0.6045	6	0.4484	6	0.4435	5	0.6101	2
	$SDAE2_{adam}$	0.7587	7	0.7575	7	0.7586	3	0.7219	5	0.7373	6	0.7574	2	**0.9637**	1	**0.9644**	1	**0.9489**	1	0.7907	5	0.8181	4	**0.7240**	1	**0.1616**	1	**0.1605**	1	**0.2301**	1	0.4476	4	0.4431	4	**0.5843**	1
	meanimp	0.7478	6	0.7431	8	0.7448	4	0.6676	4	0.6801	6	0.6710	4	0.5961	8	0.5844	8	0.3363	9	0.5899	4	0.6085	8	0.4146	3	0.6387	7	0.6682	7	1.0326	8	0.6357	3	0.7553	6	0.9590	4
	kNNimp1	0.7469	8	0.7441	7	0.7415	8	0.6387	8	0.6855	5	0.5867	8	0.6575	6	0.6549	6	0.4532	7	0.4875	8	0.6111	7	0.2882	9	0.6364	6	0.6502	6	0.9124	6	0.7798	7	0.7817	7	1.0844	8
	kNNimp3	0.7512	4	0.7464	4	0.7440	5	0.6621	6	0.6973	4	0.5836	9	0.7261	5	0.7241	5	0.4891	5	0.5714	7	0.6386	5	0.3297	8	0.5404	4	0.5492	4	0.8883	4	0.6701	6	0.7163	5	1.0379	7
	kNNimp5	0.7541	2	0.7480	3	0.7428	6	0.6581	7	0.6988	3	0.5948	7	0.7397	3	0.7387	3	0.4856	6	0.5907	3	0.6474	4	0.3394	7	0.5229	3	0.5302	3	0.8951	5	0.6497	5	0.6987	4	1.0274	6
40%	SVMimp	**0.7649**	1	**0.7695**	1	0.7532	3	**0.7618**	1	**0.7670**	1	**0.7385**	1	0.7369	4	0.7338	4	0.5103	4	0.6205	2	0.6304	6	0.4120	4	1.1565	9	1.4194	9	1.8521	9	1.3355	9	3.7205	9	1.4128	9
	EM	0.7407	9	0.7392	9	0.7416	7	0.6105	9	0.6585	9	0.5954	6	0.3962	9	0.4093	9	0.3396	8	0.3993	9	0.5416	9	0.3784	6	0.8460	8	0.8408	8	0.9938	7	0.8355	8	0.8270	8	0.9483	3
	MICE	0.7526	3	0.7491	2	0.7393	9	0.6988	2	0.7025	2	0.6344	5	0.7601	2	0.7542	2	0.5691	3	**0.6821**	1	**0.6719**	1	0.3975	5	0.5027	2	0.5094	2	0.8396	3	**0.5648**	1	0.6975	3	0.9758	5
	$SDAE2_{adadelta}$	0.7479	5	0.7453	5	0.7552	2	0.6652	5	0.6781	8	0.7330	2	0.5994	7	0.6180	7	0.7417	2	0.5881	6	0.6659	3	**0.7387**	1	0.6347	5	0.6256	5	0.7078	2	0.6366	4	0.5890	2	**0.6687**	1
	$SDAE2_{adam}$	0.7471	7	0.7449	6	**0.7552**	1	0.6685	3	0.6781	7	0.7321	3	**0.9380**	1	**0.9394**	1	**0.9528**	1	0.5892	5	0.6661	2	0.7327	2	**0.2280**	1	**0.2264**	1	**0.2641**	1	0.6351	2	**0.5878**	1	0.6761	2

Impact on classification (F-measure):

The results show that SVMimp seems to be the best imputation method in terms of classification performance and for the three highest MRs (15, 20 and 40%), regardless of the missing mechanism and configurations considered. For the lower MR (5 and 10%) there is no standard, suggesting that small amounts of missing values have little influence on the quality of the dataset for classification purposes – there is an exception for the *univa* configurations under 5% of MR: in this case, SVMimp is the winner.

Regarding the performance of the SDAE-based approaches, the 2 SDAE are ranked between the 5th and 7th position (considering all the MRs), for $MCAR2_{univa}$ and $MAR2_{univa}$. Regarding $MNAR1_{univa}$, the SDAE belong to the top 3 best imputation approaches for higher MRs (20% and 40%) while for lower MRs they are ranked between 5th and 8th positions. The SDAE are placed between 2nd and 5th positions for $MCAR3_{unifo}$ and $MAR1_{unifo}$, while for $MNAR3_{unifo}$ they are placed between 2nd and 5th positions, for all the MRs.

We also compared all the imputation methods using the Friedman rank test, for k=8 imputation methods and N=33 datasets. Table 5.3 shows the mean ranks for all the datasets and the value of the Friedman statistic, F_F, for each scenario. The null hypothesis of this test, H_0, is the equivalence between all the methods. Following the work of Demšar [53], we compared the values of F_F to the F distribution $F(0.05)_{8.256} = 1.9747$ (8 degrees of freedom). For the boldface values at Table 5.3 there is weak evidence against the null hypothesis, so conclusions about the statistical differences between the imputation methods cannot be drawn. On the other hand, for all the F_F values that are not in boldface, there is strong evidence against the null hypothesis, so we reject it - meaning that there are significant differences between the imputation methods. In brief, for all the cases where we reject the null hypothesis, we can affirm that all methods under comparison are different.

We performed another statistical test for those cases in which we could not conclude if there were statistical differences between all the methods (using the Friedman test). We use the Wilcoxon Signed Rank test to verify if there are statistical differences between some pairs of methods: for each scenario, we compared the SDAE with the imputation method that performed better. Under the null hypothesis, the median difference between pairs of observations of the 2 imputation methods is zero – when the null hypothesis is rejected, the 2 methods are considered to be different.

Table 5.3: Average ranks for the datasets used in 2nd experiment: Friedman test. Boldface values indicate weak evidence against the null hypothesis.

		Classification Performance						Imputation Quality											
		F measure						***R^2***						**RMSE**					
		Univa			**Unifo**			**Univa**			**Unifo**			**Univa**			**Unifo**		
MR	**Methods**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**
	meanimp	5.0606	5.3333	5.2879	5.3788	4.8182	4.8939	6.3939	6.2727	6.4545	5.2727	6.3939	4.8182	6.4242	6.3030	6.5152	5.2424	6.3939	5.0000
	kNNimp1	5.2879	5.0606	4.8030	6.0455	5.3333	5.5606	6.1818	6.3030	5.1515	6.5152	5.6970	6.4848	6.2121	6.3636	5.1515	6.5758	5.7273	6.4545
	kNNimp3	4.8030	4.7273	4.6515	4.6970	5.2727	5.1515	4.3333	4.5152	4.2727	4.5455	3.6364	5.0909	4.3939	4.6061	4.3030	4.6061	3.5758	5.0606
	kNNimp5	4.7879	4.8333	4.8182	4.3182	5.0000	5.0000	3.5455	3.6364	4.1515	3.1212	2.5758	4.2121	3.5758	3.6970	4.2121	3.0909	2.5455	4.0303
	SVMimp	3.9697	4.0152	3.9848	3.2121	4.2879	3.6818	3.6970	3.8788	3.6970	3.2727	3.9697	3.6970	3.7273	3.9394	3.7879	3.2424	3.9394	3.5455
5%	EM	6.3485	5.6970	5.8485	6.9545	5.9545	6.2576	8.7273	8.6061	8.1818	8.9394	8.8485	8.3333	8.7576	8.6061	8.2121	8.9394	8.8182	8.2121
	MICE	4.8182	4.6667	5.1818	4.1212	4.9091	5.0455	3.8788	3.5758	4.0303	2.0303	2.3939	2.5152	3.9091	3.6061	4.1212	2.0606	2.5152	2.5152
	$SDAE2_{adadelta}$	4.9848	5.2273	5.1970	5.0152	4.6364	4.6970	6.2727	6.1212	6.6970	5.4848	5.5152	4.3030	6.3030	6.1515	6.7576	5.5152	5.5152	4.4848
	$SDAE2_{adam}$	4.9394	5.4394	5.2273	5.2576	4.7879	4.7121	1.9697	2.0909	2.3636	5.8182	5.9697	5.5455	1.6970	1.7273	1.9394	5.7273	5.9697	5.6970
	F_F	**1.7358**	**1.1197**	**1.1894**	6.1321	**1.0023**	2.1907	40.5899	35.1833	24.4371	42.8025	41.9454	19.3831	46.1462	40.5043	28.8380	42.9527	40.7516	19.0893
	meanimp	5.5758	4.9545	5.1667	5.2121	4.5758	4.5152	6.0303	6.3636	7.0000	5.0000	6.6061	4.8485	6.0303	6.4242	7.0909	4.9697	6.6061	4.9697
	kNNimp1	4.7879	5.2273	4.9091	5.8636	5.2576	5.2273	6.3030	6.2727	5.5758	6.5758	6.0303	7.1818	6.4545	6.3333	5.5152	6.6970	6.1212	7.1818
	kNNimp3	4.1515	4.8636	4.8788	4.8333	4.6364	5.8182	4.4242	4.6364	4.2121	4.7273	3.8485	5.5758	4.4545	4.6364	4.1818	4.8182	3.8182	5.5758
	kNNimp5	4.4545	4.7576	4.7121	4.3636	4.8333	5.2576	3.7576	3.6364	3.8788	3.1818	2.9697	4.6667	3.7879	3.6364	3.9091	3.2121	2.8788	4.5758
	SVMimp	4.5758	4.2273	4.6667	3.0303	4.5455	3.8030	3.8788	4.0303	4.1515	3.3636	4.0000	3.7273	3.8788	4.0606	4.1515	3.3939	4.0909	3.6061
10%	EM	6.1970	6.3333	5.1212	7.2273	6.6364	7.1818	8.7576	8.6970	8.0606	8.9394	8.7879	8.0000	8.7576	8.7273	8.1212	8.9091	8.7879	7.9091
	MICE	4.7727	5.0455	5.5152	4.0303	4.7727	4.1212	3.6970	3.2727	4.1212	1.8788	2.2424	2.7576	3.7273	3.3333	4.1515	1.8788	2.2424	2.7879
	$SDAE2_{adadelta}$	5.1970	4.5455	4.9394	5.1818	4.9545	4.7273	6.2727	6.0909	5.9697	5.6061	5.0303	5.1818	6.2424	6.0909	6.0606	5.5152	5.0000	5.2424
	$SDAE2_{adam}$	5.2879	5.0455	5.0909	5.2576	4.7879	4.3485	1.8788	2.0000	2.0303	5.7273	5.4848	3.0606	1.6667	1.7576	1.8182	5.6061	5.4545	3.1515
	F_F	**1.7909**	**1.5176**	**0.2888**	7.2668	**1.9159**	5.2719	39.2978	39.7825	25.8806	43.8830	37.2574	22.3424	43.5758	44.9668	29.7909	42.8400	38.3763	21.4042
	meanimp	5.1970	5.1364	5.1667	4.8788	4.6970	5.3333	6.2121	6.3636	7.1212	4.6364	6.4242	5.1818	6.2727	6.3636	7.2121	4.6364	6.4545	5.3030
	kNNimp1	5.0606	5.1667	4.8182	6.2879	4.8636	6.3030	6.3030	6.3030	5.7879	6.9697	6.0909	7.7576	6.3636	6.4242	5.5758	7.0303	6.0303	7.6667
	kNNimp3	4.4242	4.6515	4.8939	5.2424	5.0455	5.8636	4.4848	4.3636	4.4242	4.9091	4.3333	6.0909	4.5152	4.4545	4.4242	4.9091	4.3636	6.0909
	kNNimp5	4.8182	4.5606	5.1667	4.7424	4.8333	5.4394	3.6061	3.6061	3.7576	3.6061	3.2424	5.3030	3.6667	3.6364	3.9394	3.6667	3.1212	5.3030
	SVMimp	3.6212	3.9394	4.4545	2.5758	3.8182	2.8182	3.7879	3.8182	4.3030	3.3030	4.1818	4.0000	3.8182	3.8485	4.3030	3.3030	4.3333	3.8788
15%	EM	6.2273	6.5606	5.2727	7.9242	7.5455	6.9545	8.7576	8.7879	8.3636	8.8788	8.8788	7.4242	8.7576	8.8182	8.3636	8.8788	8.6970	7.3030
	MICE	4.5000	4.9091	5.1818	3.8485	5.0303	4.4545	3.7576	3.8485	4.0303	1.8182	2.2424	3.4242	3.7576	3.8485	4.0303	1.8485	2.4545	3.4848
	$SDAE2_{adadelta}$	5.6212	5.2727	4.9091	4.7727	4.5303	3.8788	6.1818	6.0606	5.4242	5.3939	4.6970	2.3939	6.1515	5.9697	5.5152	5.3030	4.6667	2.4848
	$SDAE2_{adam}$	5.5303	4.8030	5.1364	4.7273	4.6364	3.9545	1.9091	1.8485	1.7879	5.4848	4.9091	3.4242	1.6970	1.6364	1.6364	5.4242	4.8788	3.4848
	F_F	2.7394	2.3119	**0.2861**	13.3671	5.1877	9.7175	40.3388	41.9402	32.5015	43.0065	32.3665	27.1747	44.1538	46.4629	34.2090	42.3292	27.6815	24.4096
	meanimp	4.9848	5.1667	5.1515	5.0000	5.2424	5.5000	6.2121	6.5152	7.3636	4.6061	6.3636	5.1818	6.1212	6.5758	7.4848	4.4848	6.6061	5.2727
	kNNimp1	5.9848	5.6364	4.9545	6.5000	5.3485	6.1061	6.3333	6.1818	5.4848	6.9091	6.5455	7.6970	6.4848	6.2727	5.3636	7.3333	6.4545	7.6364
	kNNimp3	4.9394	4.8636	4.6364	4.6667	4.8939	5.7576	4.3636	4.6061	4.6364	4.7879	4.3636	6.3030	4.5152	4.6364	4.5152	4.9697	4.2121	6.3939
	kNNimp5	4.7727	4.5303	4.8939	4.4242	5.2576	5.4545	3.4545	3.6364	4.2121	3.5758	3.1212	5.4848	3.4848	3.6970	4.3333	3.5758	3.0303	5.5455
	SVMimp	3.9697	3.3030	4.7424	2.6061	3.2879	2.7879	4.0606	3.7879	4.5455	3.5455	4.8788	4.2121	4.0909	3.7879	4.5758	3.8182	5.0909	4.2424
20%	EM	5.9697	6.6212	5.9242	7.8182	6.5455	7.1818	8.8182	8.7576	8.0909	8.8788	8.7879	7.3636	8.8182	8.6970	7.9697	8.8182	8.6667	7.0303
	MICE	4.5909	4.6364	5.5758	4.0455	4.3939	4.2121	3.6667	3.6970	4.0303	1.8485	2.2424	3.7273	3.6667	3.6364	3.9697	1.8485	2.4545	3.7273
	$SDAE2_{adadelta}$	4.8939	5.1667	4.5000	5.0152	4.9545	4.3939	6.2727	5.9697	5.0303	5.2121	4.2121	3.1515	6.1515	5.9697	5.2121	4.8485	4.0303	3.2424
	$SDAE2_{adam}$	4.8939	5.0758	4.6212	4.9242	5.0758	3.6061	1.8182	1.8485	1.6061	5.6364	4.4848	1.8788	1.6667	1.7273	1.5758	5.3030	4.4545	1.9091
	F_F	**1.8130**	3.7847	**0.9977**	12.9468	3.5244	10.5144	44.8706	41.4673	29.5290	40.2238	34.2888	32.4856	46.7288	43.5321	29.5725	41.1742	32.7893	28.4231
	meanimp	4.7424	5.5758	5.1818	4.4697	5.1667	4.6667	6.0909	6.5758	7.4848	4.0606	6.0909	4.7576	6.1515	6.5455	7.7727	4.1515	6.0909	5.3030
	kNNimp1	5.6061	5.8030	5.4697	6.9091	5.7576	6.5606	6.3030	6.1818	6.1667	7.5455	6.7576	8.4545	6.3939	6.5152	5.6818	8.0303	7.0909	8.1515
	kNNimp3	5.2727	4.5758	5.3788	5.2424	4.8030	6.3485	4.6061	4.6970	5.2121	5.6970	5.1515	7.0606	4.6061	4.6667	4.9091	5.9394	5.3636	7.0000
	kNNimp5	4.7424	4.6818	5.3485	5.4545	4.8636	6.1818	3.7273	3.8485	5.3333	4.3636	4.1212	6.1818	3.6667	3.8182	5.5758	4.4848	4.4545	6.2121
	SVMimp	3.3939	3.2879	4.3485	2.0606	3.0455	3.0758	4.0000	3.8788	5.1061	3.0909	4.7576	4.8182	4.0606	3.9697	5.1364	3.6061	5.7576	4.9394
40%	EM	6.5303	6.6364	4.9697	8.2273	6.3485	6.1212	8.8182	8.7273	6.7576	8.7576	8.8485	5.4848	8.7879	8.6364	6.5909	8.6364	7.9394	4.8182
	MICE	4.3788	4.3333	5.3939	3.4091	4.6970	5.6212	3.5152	3.6364	4.6364	1.6061	3.5758	5.2424	3.5455	3.6061	4.8182	1.6667	3.8485	5.5758
	$SDAE2_{adadelta}$	5.0152	4.9848	4.4242	4.8788	5.1818	3.0909	6.2727	5.8485	3.0303	5.0303	2.7879	1.0606	6.1515	5.6364	3.3333	4.2121	2.0909	1.0303
	$SDAE2_{adam}$	5.3182	5.1212	4.4848	4.3485	5.1364	3.3333	1.6667	1.6061	1.2727	4.8485	2.9091	1.9394	1.6364	1.6061	1.1818	4.2727	2.3636	1.9697
	F_F	3.5444	4.4506	**0.9309**	24.8328	3.8485	13.2082	45.3483	43.4286	29.3350	53.5586	34.7544	79.9435	45.3827	42.8346	28.8203	57.2090	36.4368	68.0134

We define the level of significance as $\alpha = 0.05$ and the $p - values$ of the comparison tests are shown in Table 5.4.

Table 5.4: $P - values$ for the Wilcoxon Signed Rank test. We use this test to ascertain whether there are significant differences between the indicated pairs of imputation methods. Boldface methods are the superior ones (in terms of classification performance) from each pair. Boldface *p-values* indicate strong evidence against the null hypothesis.

F-measure					
	MR	Mechanism	Methods		p-value
univa	5%	MCAR2	**SVMimp**	$SDAE2_{adadelta}$	0.1570
				$SDAE2_{adam}$	0.1701
		MAR2	**SVMimp**	$SDAE2_{adadelta}$	**0.0174**
				$SDAE2_{adam}$	**0.0113**
		MNAR1	**SVMimp**	$SDAE2_{adadelta}$	**0.0131**
				$SDAE2_{adam}$	**0.0131**
	10%	MCAR2	**kNNimp3**	$SDAE2_{adadelta}$	**0.0400**
				$SDAE2_{adam}$	**0.0442**
		MAR2	**SVMimp**	$SDAE2_{adadelta}$	**0.0386**
				$SDAE2_{adam}$	**0.0158**
		MNAR1	**kNNimp5**	$SDAE2_{adadelta}$	0.2721
				$SDAE2_{adam}$	0.1823
	15%	MNAR1	**SVMimp**	$SDAE2_{adadelta}$	0.1728
				$SDAE2_{adam}$	0.1556
	20%	MCAR2	**SVMimp**	$SDAE2_{adadelta}$	**0.0333**
				$SDAE2_{adam}$	**0.0366**
		MNAR1	**SVMimp**	$SDAE2_{adadelta}$	0.6092
				$SDAE2_{adam}$	0.5321
	40%	MNAR1	SVMimp	$SDAE2_{adadelta}$	0.9176
				$\boldsymbol{SDAE2_{adam}}$	0.9588
unifo	5%	MAR1	**MICE**	$SDAE2_{adadelta}$	0.1771
				$SDAE2_{adam}$	0.1240
	10%	MAR1	**MICE**	$SDAE2_{adadelta}$	0.9826
				$SDAE2_{adam}$	0.4348

We reject the null hypothesis of the Wilcoxon Signed Rank test when the $p-value$ is bellow the level of significance ($\alpha = 0.05$). For example, according to the information in Table 5.4, for 5% of MR and $MAR2_{univa}$, SVMimp shows to be significantly different from SDAE-based approaches. On the other hand, although $SDAE2_{adam}$ has superiority for 40% of MR and $MNAR1_{univa}$, it does not prove to be significantly different from SVMimp.

According to Table 5.4, there seems to be no superiority of the SDAE compared to the remaining methods. For some scenarios, we can conclude that there are significant differences between the 2 imputation methods in comparison – MR of

5%: $MAR2_{univa}$ and $MNAR1_{univa}$; MR of 10%: $MCAR2_{univa}$ and $MAR2_{univa}$; MR of 20%: $MCAR2_{univa}$. However, in these cases the SDAE are not superior. For the remaining scenarios, we can not conclude about the superiority of any of the compared methods. Therefore, although the SDAE are not superior in terms of F-measure, they show no statistically significant differences regarding the best method.

We continue this analysis by referring to the results obtained by Gondara et al. [9], which used a similar benchmarking of datasets (although smaller, with only 15 datasets) and a SDAE approach for imputation ($DAE1_{sgd}$). Gondara et al. [9] proposed a SDAE based model for imputation but only compare the results with one other imputation method, MICE. Furthermore, in the work of Gondara et al., only 2 missing mechanisms were generated – MCAR and MNAR – for a fixed MR of 20%. Their results have shown that the SDAE-based approach outperforms MICE for most studied scenarios. Therefore, we also perform this comparison, only for *unifo* configuration since Gondara et al. [9] did not use any *univa* methods for the synthetic generation of MD. These results are presented in Figures 5.3 and 5.4.

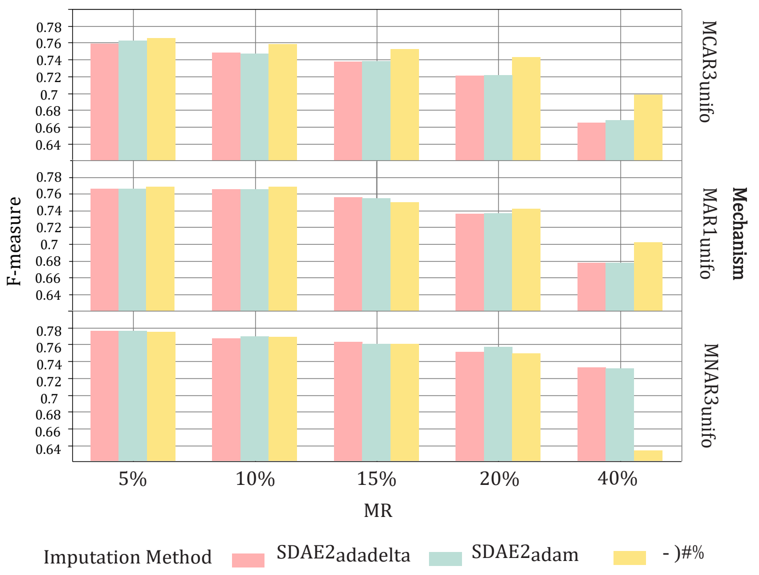


Figure 5.3: Comparison between the results obtained with the SDAE based approaches and the well-known method MICE (*unifo* configuration), in terms of classification performance (F-measure).

In terms of classification performance (F-measure), the results for $MCAR3_{unifo}$ show that MICE is superior under all the MRs. Furthermore, for $MAR1_{unifo}$, MICE remains superior, except for the MR of 15%, in which both SDAE approaches show a better performance. For $MNAR3_{unifo}$, the results are somewhat different: for higher MRs (15, 20 and 40%), the 2 SDAE-based approaches are superior to MICE.

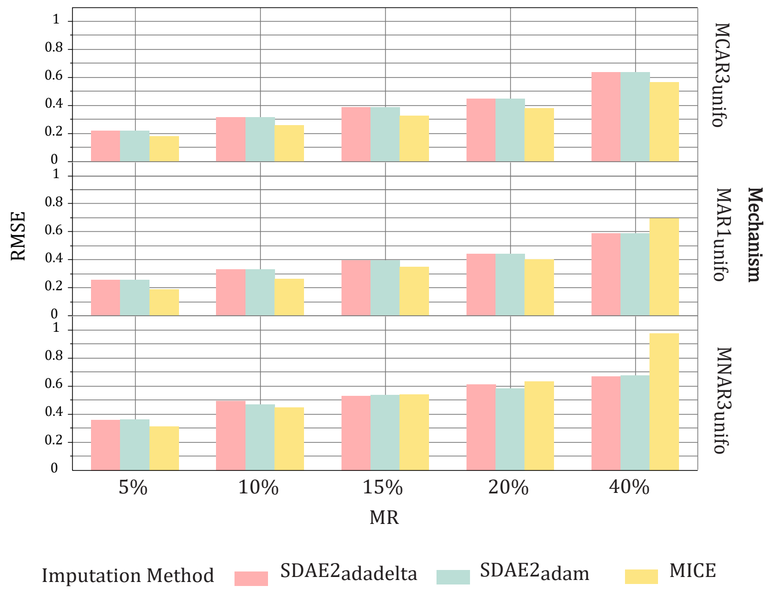


Figure 5.4: Comparison between the results obtained with the SDAE based approaches and the well-known method MICE (*unifo* configuration), in terms of imputation quality (RMSE).

Regarding imputation quality, measured with RMSE, the results show that for $MCAR3_{univo}$ there is no advantage in using our SDAE-based approaches compared to MICE. For $MAR1_{unifo}$, SDAE seem to be superior to MICE for a higher MR (40%). Finally, $MNAR3_{unifo}$ benefits from the use of SDAE, for higher MRs (15, 20 and 40%) while under lower MRs (5 and 10%) MICE seems to be superior, despite the small difference compared to the SDAE.

Conclusions

In this 2$^{\text{nd}}$ experiment, we proposed a multiple imputation approach based on SDAE where the training data is corrupted in a way that mimics a particular missing mechanism. Furthermore, we study the performance of several imputation approaches for different generation methods (*univa* and *unifo* configurations). Overall, the following conclusions are derived:

- In terms of imputation quality (RMSE), $SDAE2_{adam}$ shows the best results for *univa* configurations. Considering *unifo* configurations and MCAR and MAR mechanisms, MICE seems to be the best method. For $MNAR3_{unifo}$ and higher MR, there is a clear advantage in using SDAE-based approaches;

- In terms of imputation quality and considering *univa* implementations, Adam optimizer seems to achieve better results than the Adadelta optimizer;

- Regarding classification performance and for several of the studied scenarios there are not statistical differences between all the methods. Therefore, even if some methods obtained better results than SDAE-based approaches does not mean that they are superior, since the differences are not statistically significant;

- For higher MRs, SVMimp seems to be the best imputation method, in terms of classification performance. In this case, the SDAE also have a good performance. For lower MRs, the results are not homogeneous.
- Although the $SDAE2_{adam}$ ensures the best imputation quality for *univa* configurations, the same does not occur in terms of classification performance;

Although the results of the SDAE-based approaches show its advantage for some of the studied scenarios, these results are not sufficiently high to counteract the greatest limitation of the use of SDAE: the high computational time and space/memory required.

5.3 3rd Experiment

In the 2nd experiment, we used a benchmark of standard datasets to study the imputation performance of SDAE-based approaches. Our SDAE-based approaches have demonstrated to be superior for many of the studied scenarios, although there was no agreement on the best methods regarding the classification performance and imputation quality.

Therefore, in this 3rd experiment, we reproduce the setup proposed for the 2nd experiment (Section 5.2) but, in this case, we selected 5 datasets that have higher sample sizes than the ones used in the previous experiment, since deep learning techniques have a “good reputation” in contexts where more data is available [7]. With this 3rd experiment, we want to determine if there is any advantage in using the proposed SDAE approaches when the amount of data increases and if this advantage exists for in both evaluated perspectives: classification and imputation quality.

Thus being, the main goal of this experiment is:

- Evaluate the performance of 2 SDAE-based approaches for datasets with higher sample sizes.

As for the 2nd experiment, we synthetically generated MVs for 5 runs, using 6 different implementations (*univa* and *unifo*), for 5 MRs (5, 10, 15, 20 and 40%) – a total of 750 incomplete datasets. Once again, it can be observed that, the classification performance (F-measure) and imputation quality (R^2 and RMSE) decrease as the MRs increase. We can also verify that the results regarding both classification and imputation performance are superior for *univa* configurations. Next, we analyse in detail the results of this 3rd Experiment, regarding both imputation quality and classification performance, shown in Table 5.5.

Quality of Imputation ($\mathbf{R^2}$ and RMSE):

For *univa* configurations, $SDAE2_{adadelta}$ and $SDAE2_{adam}$ proved to be the first and the second best imputation approaches, respectively, for all missing mechanisms, regardless of the MR.

Regarding *unifo* configurations, the results are not so homogeneous. MICE proved to be the best imputation method under MCAR configurations while $SDAE2_{adadelta}$ and $SDAE2_{adam}$ are ranked as 4th and 5th best approaches.

For MAR configuration and R^2 metric, SVMimp is the top winner. However, RMSE do not coincide with R^2 for all scenarios: RMSE shows that MICE is the best imputation approach under 5% of MR while $SDAE2_{adadelta}$ is the top winner for 40% of MR. In summary, both SDAE-based methods are ranked between the 1st and 4th positions. Both metrics show that $SDAE2_{adadelta}$ is the best imputation approach for MNAR configurations, for most scenarios: the only exception occurs in terms of RMSE and under 5% of MR, where MICE is superior to the SDAE-based approaches.

Considering all the MRs, we can also verify that at least one of our proposed methods based on SDAE approaches is included in the top 3 best imputation approaches, for MAR and MNAR configurations. In the case of MNAR, the SDAE approaches are in the 1st and 2nd place. For MAR configurations, they present a minimum and maximum difference from the best method of 0.0081 (MR of 5%) and 0.0182 (20%), respectively. Furthermore, Adadelta seems to be the optimizer that guarantees the best imputation results.

Table 5.5: Results obtained from the 3rd experiment by imputation method: average F-measure, R^2 and RMSE are shown regarding each missing data mechanism, configuration, metric and missing rate. The best results for each configuration are marked in bold. Pink and blue values indicate whether $SDAE2_{adadelta}$ or $SDAE2_{adam}$ belong to the top 3 of the best imputation approaches, respectively.

		Classification Performance						Imputation Quality											
		F-measure						R^2						**RMSE**					
		Univa			**Unifo**			**Univa**			**Unifo**			**Univa**			**Unifo**		
MR	**Methods**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**
5%	meanimp	0.9486 4	0.9486 4	**0.9619** 3	0.9460 4	0.9365 2	**0.9648** 2	0.9476 5	0.9606 5	0.8827 8	0.9406 3	0.9363 6	0.8889 3	0.2289 5	0.1987 5	0.3515 8	0.2234 3	0.2344 6	0.3338 4
	kNNimp1	0.9358 8	0.9387 8	0.9523 8	0.9345 8	0.9212 8	0.9467 8	0.9114 8	0.9309 8	0.9203 6	0.9099 8	0.9152 8	0.8386 8	0.3015 8	0.2667 8	0.2909 6	0.2874 8	0.2812 8	0.4027 8
	kNNimp3	0.9393 7	0.9432 7	0.9562 6	0.9373 7	0.9252 7	0.9547 6	0.9276 7	0.9438 7	0.9273 5	0.9309 7	0.9344 7	0.8579 6	0.2706 7	0.2389 7	0.2779 5	0.2449 7	0.2387 7	0.3745 6
	kNNimp5	0.9396 6	0.9435 6	0.9558 7	0.9387 6	0.9276 6	0.9543 7	0.9296 6	0.9446 6	0.9275 4	0.9347 6	0.9370 5	0.8624 5	0.2668 6	0.2371 6	0.2776 4	0.2367 6	0.2330 5	0.3678 5
	SVMimp	**0.9640** 1	**0.9632** 1	**0.9619** 3	**0.9647** 1	**0.9650** 1	0.9632 4	0.9723 3	0.9840 3	0.9585 3	0.9446 2	**0.9454** 1	0.8020 9	0.1668 3	0.1270 3	0.2109 3	0.2160 2	0.2143 2	0.5845 9
	EM	0.9268 9	0.9285 9	0.9471 9	0.9182 9	0.9048 9	0.9413 9	0.9094 9	0.9240 9	0.8509 9	0.9031 9	0.9027 9	0.8523 7	0.3037 9	0.2785 9	0.3955 9	0.2983 9	0.3022 9	0.3802 7
	MICE	0.9470 5	0.9485 5	0.9612 5	0.9456 5	0.9315 5	0.9608 5	0.9547 4	0.9681 4	0.9192 7	**0.9493** 1	0.9453 2	0.8886 4	0.2128 4	0.1785 4	0.2927 7	**0.2031** 1	**0.2137** 1	**0.3272** 1
	$SDAE2_{adadelta}$	0.9493 3	0.9489 3	**0.9619** 3	0.9463 2	0.9350 4	**0.9648** 2	**0.9864** 1	**0.9869** 1	**0.9811** 1	0.9404 4	0.9373 3	**0.8910** 1	**0.0638** 1	**0.0597** 1	**0.0992** 1	0.2238 4	0.2321 3	0.3300 2
	$SDAE2_{adam}$	0.9496 2	0.9497 2	**0.9619** 3	0.9462 3	0.9350 3	**0.9648** 2	0.9864 2	0.9869 2	0.9809 2	0.9403 5	0.9372 4	0.8894 2	0.0639 2	0.0598 2	0.1003 2	0.2241 5	0.2324 4	0.3326 3
10%	meanimp	0.9311 4	0.9405 4	**0.9618** 3	0.9269 4	0.9072 4	0.9635 3	0.8971 5	0.9210 5	0.7769 4	0.8917 3	0.8881 5	0.8126 3	0.3208 5	0.2814 5	0.4977 4	0.3143 3	0.3246 5	0.4449 3
	kNNimp1	0.9121 8	0.9210 8	0.9309 8	0.9105 8	0.8777 8	0.9132 9	0.8294 8	0.8706 8	0.7178 9	0.8360 8	0.8440 8	0.6753 8	0.4247 8	0.3700 8	0.5609 9	0.4038 8	0.4043 8	0.7525 8
	kNNimp3	0.9169 7	0.9297 7	0.9470 7	0.9165 7	0.8889 7	0.9330 7	0.8612 7	0.8914 7	0.7212 8	0.8764 7	0.8764 7	0.7112 7	0.3773 7	0.3346 7	0.5580 8	0.3401 7	0.3482 7	0.6349 7
	kNNimp5	0.9192 6	0.9301 6	0.9487 6	0.9191 6	0.8905 6	0.9398 6	0.8644 6	0.8933 6	0.7255 7	0.8849 6	0.8821 6	0.7193 6	0.3727 6	0.3317 6	0.5543 7	0.3262 6	0.3386 6	0.6128 6
	SVMimp	**0.9645** 1	**0.9642** 1	0.9607 4	**0.9642** 1	**0.9624** 1	0.9572 5	0.9435 3	0.9630 3	0.7823 3	0.8996 2	**0.9070** 1	0.6642 9	0.2393 3	0.1948 3	0.4970 3	0.3052 2	**0.2928** 1	0.9015 9
	EM	0.8895 9	0.8969 9	0.9279 9	0.8787 9	0.8441 9	0.9222 8	0.8254 9	0.8466 9	0.7299 6	0.8202 9	0.8286 9	0.7586 5	0.4254 9	0.4001 9	0.5452 6	0.4202 9	0.4186 9	0.5032 5
	MICE	0.9299 5	0.9397 5	0.9572 5	0.9245 5	0.9032 5	0.9593 4	0.9111 4	0.9355 4	0.7730 5	**0.9071** 1	0.9015 2	0.7997 4	0.2982 4	0.2540 4	0.5016 5	**0.2889** 1	0.3005 2	0.4615 4
	$SDAE2_{adadelta}$	0.9326 3	0.9408 3	0.9619 2	0.9290 2	0.9098 2	**0.9647** 1	**0.9825** 1	**0.9834** 1	**0.9738** 1	0.8913 4	0.8919 3	**0.8313** 1	**0.0896** 1	**0.0841** 1	**0.1364** 1	0.3150 4	0.3172 3	**0.4211** 1
	$SDAE2_{adam}$	0.9329 2	0.9409 2	**0.9619** 1	0.9286 3	0.9098 3	0.9647 2	0.9824 2	0.9834 2	0.9734 2	0.8911 5	0.8918 4	0.8282 2	0.0897 2	0.0843 2	0.1379 2	0.3153 5	0.3175 4	0.4252 2
15%	meanimp	0.9182 4	0.9277 4	**0.9619** 2	0.9146 4	0.8907 2	**0.9600** 2	0.8541 5	0.8791 5	0.6870 4	0.8375 3	0.8407 5	0.7444 3	0.3820 5	0.3482 5	0.6055 4	0.3912 3	0.3954 5	0.5353 3
	kNNimp1	0.8812 8	0.8987 8	0.9119 8	0.8872 8	0.8326 8	0.8852 9	0.7690 8	0.8082 8	0.5962 9	0.7602 8	0.7823 8	0.5615 7	0.4988 8	0.4570 8	0.6937 9	0.4986 8	0.4926 8	1.0575 7
	kNNimp3	0.8933 7	0.9107 7	0.9362 7	0.8958 7	0.8544 7	0.9100 7	0.8036 7	0.8370 7	0.6173 8	0.8147 7	0.8249 7	0.6039 6	0.4523 7	0.4136 7	0.6757 8	0.4243 7	0.4198 7	0.8922 6
	kNNimp5	0.8965 6	0.9130 6	0.9370 6	0.8983 6	0.8608 6	0.9122 6	0.8074 6	0.8392 6	0.6235 7	0.8262 6	0.8342 6	0.6125 5	0.4476 6	0.4105 6	0.6713 7	0.4088 6	0.4050 6	0.8621 5
	SVMimp	**0.9626** 1	**0.9652** 1	0.9607 4	**0.9641** 1	**0.9650** 1	0.9581 4	0.9216 3	0.9442 3	0.7296 3	0.8498 2	**0.8657** 1	0.5530 9	0.2828 3	0.2398 3	0.5703 3	0.3827 2	**0.3605** 1	1.2867 9
	EM	0.8577 9	0.8666 9	0.9097 9	0.8342 9	0.8052 9	0.9067 8	0.7513 9	0.7721 9	0.6385 6	0.7363 9	0.7616 9	0.6787 4	0.5129 9	0.4930 9	0.6454 6	0.5180 9	0.5076 9	0.5966 4
	MICE	0.9149 5	0.9244 5	0.9562 5	0.9110 5	0.8806 5	0.9497 5	0.8734 4	0.9002 4	0.6752 5	**0.8583** 1	0.8553 2	0.5550 8	0.3559 4	0.3160 4	0.6181 5	**0.3657** 1	0.3775 2	1.1655 8
	$SDAE2_{adadelta}$	0.9191 3	0.9286 3	**0.9619** 2	0.9155 3	0.8897 3	**0.9600** 2	**0.9789** 1	**0.9798** 1	**0.9684** 1	0.8370 4	0.8480 3	**0.7916** 1	**0.1078** 1	**0.1034** 1	**0.1621** 1	0.3919 4	0.3824 3	**0.4827** 1
	$SDAE2_{adam}$	0.9192 2	0.9290 2	**0.9619** 2	0.9157 2	0.8894 4	**0.9600** 2	0.9788 2	0.9798 2	0.9678 2	0.8367 5	0.8478 4	0.7873 2	0.1080 2	0.1036 2	0.1640 2	0.3923 5	0.3828 4	0.4877 2
20%	meanimp	0.9081 4	0.9165 3	**0.9619** 2	0.8943 4	0.8591 4	0.9597 4	0.8021 5	0.8430 5	0.6038 4	0.7924 3	0.7931 5	0.6766 3	0.4449 5	0.3973 5	0.7002 4	0.4451 3	0.4590 6	0.6209 3
	kNNimp1	0.8556 8	0.8740 8	0.8900 8	0.8396 8	0.7929 8	0.8566 9	0.6951 8	0.7550 8	0.5150 9	0.6960 8	0.7355 8	0.4647 9	0.5803 8	0.5235 8	0.7807 9	0.5722 8	0.5533 8	1.4089 8
	kNNimp3	0.8760 7	0.8876 7	0.9226 7	0.8597 7	0.8076 7	0.8994 7	0.7384 7	0.7905 7	0.5510 8	0.7641 7	0.7809 7	0.5108 6	0.5256 7	0.4731 7	0.7501 8	0.4840 7	0.4737 7	1.1896 6
	kNNimp5	0.8802 6	0.8931 6	0.9240 6	0.8633 6	0.8089 6	0.9115 6	0.7443 6	0.7940 6	0.5585 7	0.7785 6	0.7911 6	0.5205 5	0.5192 6	0.4687 6	0.7452 7	0.4662 6	0.4580 5	1.1539 5
	SVMimp	**0.9650** 1	**0.9659** 1	0.9607 4	**0.9654** 1	**0.9679** 1	**0.9624** 1	0.8944 3	0.9235 3	0.6539 3	0.8114 2	**0.8239** 1	0.4766 7	0.3278 3	0.2846 3	0.6837 3	0.4300 2	**0.4238** 1	1.4969 9
	EM	0.8189 9	0.8345 9	0.8870 9	0.8061 9	0.7738 9	0.8762 8	0.6716 9	0.7051 9	0.5613 6	0.6629 9	0.7015 9	0.6057 4	0.5947 9	0.5679 9	0.7280 6	0.5939 9	0.5828 9	0.6786 4
	MICE	0.9009 5	0.9126 5	0.9514 5	0.8927 5	0.8468 5	0.9467 5	0.8298 4	0.8709 4	0.6033 5	**0.8193** 1	0.8093 2	0.4733 8	0.4125 4	0.3595 4	0.7002 5	**0.4143** 1	0.4409 4	1.3386 7
	$SDAE2_{adadelta}$	0.9104 2	0.9166 2	**0.9619** 2	0.8957 2	0.8618 3	0.9616 3	**0.9748** 1	**0.9764** 1	**0.9652** 1	0.7917 4	0.8057 3	**0.7653** 1	**0.1254** 1	**0.1188** 1	**0.1809** 1	0.4461 4	0.4363 2	**0.5311** 1
	$SDAE2_{adam}$	0.9103 3	0.9162 4	**0.9619** 2	0.8957 3	0.8619 2	0.9616 3	0.9747 2	0.9764 2	0.9645 2	0.7913 5	0.8054 4	0.7596 2	0.1256 2	0.1192 2	0.1829 2	0.4466 5	0.4368 3	0.5372 2
40%	meanimp	0.8471 4	0.8729 3	0.9606 3	0.8162 4	0.7379 4	0.9509 4	0.5932 5	0.6762 5	0.4403 5	0.5909 3	0.6048 5	0.4814 3	0.6379 5	0.5729 5	0.9305 5	0.6322 3	0.6646 5	0.9013 3
	kNNimp1	0.7383 9	0.7872 8	0.7908 8	0.7147 8	0.6770 8	0.6161 9	0.4438 8	0.5764 8	0.3511 9	0.4133 8	0.5646 8	0.2143 9	0.8225 8	0.7216 8	1.0001 9	0.8519 9	0.8052 9	1.4815 7
	kNNimp3	0.7645 7	0.8090 7	0.8006 7	0.7382 7	0.6824 7	0.6463 8	0.5048 7	0.6233 7	0.3783 8	0.5315 7	0.5869 7	0.2728 6	0.7440 7	0.6526 7	0.9797 8	0.6955 7	0.7156 7	1.1914 6
	kNNimp5	0.7792 6	0.8186 6	0.8051 6	0.7549 6	0.6875 6	0.6919 7	0.5147 6	0.6303 6	0.3881 6	0.5643 6	0.5937 6	0.2889 5	0.7339 6	0.6444 6	0.9754 7	0.6619 6	0.6946 6	1.1410 5
	SVMimp	**0.9737** 1	**0.9704** 1	0.9598 4	**0.9727** 1	**0.9818** 1	0.9553 3	0.7744 3	0.8395 3	0.5486 3	0.6243 2	**0.6457** 1	0.2287 8	0.4836 3	0.4189 3	0.8527 3	0.6270 2	0.6391 3	2.3626 9
	EM	0.7465 8	0.7699 9	0.7809 9	0.7087 9	0.6746 9	0.7549 6	0.3939 9	0.4528 9	0.3874 7	0.3935 9	0.5287 9	0.4095 4	0.8403 9	0.8135 9	0.9511 6	0.8343 8	0.7991 8	0.9250 4
	MICE	0.8357 5	0.8612 5	0.9126 5	0.8117 5	0.7000 5	0.8701 5	0.6472 4	0.7254 4	0.4532 4	**0.6337** 1	0.6166 4	0.2448 7	0.5941 4	0.5250 4	0.9006 4	**0.5978** 1	0.6572 4	1.7857 8
	$SDAE2_{adadelta}$	0.8484 3	0.8726 4	0.9620 2	0.8184 3	0.7400 3	**0.9564** 1	**0.9586** 1	**0.9623** 1	**0.9709** 1	0.5900 4	0.6187 2	**0.7841** 1	**0.1787** 1	**0.1686** 1	**0.1990** 1	0.6334 4	**0.6128** 1	**0.6192** 1
	$SDAE2_{adam}$	0.8487 2	0.8729 2	**0.9620** 1	0.8187 2	0.7407 2	0.9562 2	0.9586 2	0.9622 2	0.9702 2	0.5895 5	0.6185 3	0.7780 2	0.1789 2	0.1691 2	0.2019 2	0.6341 5	0.6134 2	0.6271 2

Impact on classification (F-measure):

SVMimp seems to be the method which ensures the best classification perfomance for MCAR and MAR mechanisms, regardless of the missing rate and type of configuration (*univa* and *unifo*).

For MNAR *univa* configurations, $SDAE2_{adam}$ proved to be the best imputation method under 2 different MRs (10 and 40%). For 5% of MR, there is a tie between 4 methods: $SDAE2_{adam}$, $SDAE2_{adadelta}$, SVMimp and Meanimp. For the remaining MR (15 and 20%), there are 3 tied methods in the top of best imputation approaches, including $SDAE2_{adam}$, $SDAE2_{adadelta}$ and Meanimp. For *unifo* configurations, there is no standard: there are 3 tied methods, including the 2 SDAE and Meanimp, when the MR is low (5 and 15%); $SDAE2_{adadelta}$ seems to be the best method for MRs of 10 and 40%; SVMimp seems to be the best method for a MR of 20%.

Regarding the performance of SDAE, at least one of the used approaches is included in the top 3 best imputation approaches for all the studied scenarios, with a minimum and maximum difference from the best method of 0.0135 ($MAR2_{univa}$ and MR of 5%) and 0.2411 ($MAR1_{unifo}$ and MR of 40%), respectively. Moreover, Adam optimizer seems to achieve better results than Adadelta.

For this experiment, we also made a comparison between all the imputation methods using the Friedman rank test [53]. Table 5.6 shows the mean ranks for all the datasets and the value of the Friedman statistic, F_F, for each scenario. Once again, we followed the work of Demšar [53] and compared the values of F_F to the F distribution $F(0.05)_{8.32} = 2.2444$ (8 degrees of freedom). The boldface values in Table 5.6 indicate weak evidence against the null hypothesis, so in this case it is not possible to conclude whether there are significant differences between the imputation methods.

We reject the null hypothesis of Friedman's test for most of the scenarios, regardless of the metric, which means that the methods are not equivalent. Regarding of classification performance (F-measure), for $MNAR1_{univa}$ there is a tie between 4 imputation methods (Meanimp, SVMimp, $SDAE2_{adadelta}$ and $SDAE2_{adam}$). In addiction, there is weak evidence against the null hypothesis so, in this case, we do not conclude if there are statistical differences between the methods.

For all the cases where SDAE belongs to the top 3 best imputation approaches, the results proved that all methods are significantly different. Therefore, we can conclude that the SDAE are effectively superior to the remaining methods.

Table 5.6: Average ranks for the datasets used in 3rd experiment: Friedman test. Boldface values indicate weak evidence against the null hypothesis.

		Classification Performance						Imputation Quality											
		F measure						***R*²**						**RMSE**					
		Univa			**Unifo**			**Univa**			**Unifo**			**Univa**			**Unifo**		
MR	**Methods**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**	**MCAR2**	**MAR2**	**MNAR1**	**MCAR3**	**MAR1**	**MNAR3**
	meanimp	3.8	4.6	4.0	4.4	3.4	2.8	5.4	5.2	6.2	3.4	5.6	3.8	5.8	5.2	6.2	3.4	5.6	3.8
	kNNimp1	7.5	7.3	6.3	7.5	7.4	8.1	7.8	7.8	6.4	7.8	7.8	8.2	8.2	8.2	6.4	7.8	7.8	7.8
	kNNimp3	6.5	6.3	4.9	6.1	4.2	6.7	6.6	6.4	5.0	6.2	5.2	6.6	7.0	6.8	5.0	6.2	5.2	6.6
	kNNimp5	6.1	5.9	4.9	5.1	4.6	6.1	5.4	5.2	4.4	5.2	3.8	5.2	5.8	5.6	4.4	5.2	3.8	5.2
	SVMimp	1.5	1.3	4.0	1.5	1.6	3.6	2.6	3.4	4.4	2.6	2.4	5.0	3.0	3.8	4.4	2.6	2.4	5.0
5%	EM	7.8	7.8	7.8	8.4	8.8	8.2	8.2	7.8	8.0	8.6	8.6	6.6	8.2	7.8	8.0	8.6	8.6	6.6
	MICE	4.1	4.2	5.1	3.4	6.0	3.9	3.6	4.2	6.0	1.4	2.2	3.6	4.0	4.6	6.0	1.4	2.2	3.2
	$SDAE2_{adadelta}$	3.7	3.7	4.0	4.3	4.5	2.8	2.2	2.2	1.8	4.4	4.2	2.0	1.0	1.2	1.8	4.4	4.2	2.4
	$SDAE2_{adam}$	4.0	3.9	4.0	4.3	4.5	2.8	3.2	2.8	2.8	5.4	5.2	4.0	2.0	1.8	2.8	5.4	5.2	4.4
	F_F	5.5012	4.8955	**1.1680**	5.6696	6.2652	9.0719	7.8577	5.0909	3.7922	11.0754	6.8696	3.6923	52.6038	16.0000	3.7922	11.0754	6.8696	2.8182
	meanimp	4.8	3.8	4.1	5.2	3.1	2.5	5.8	5.4	4.0	3.4	4.8	3.6	5.8	5.4	4.0	3.4	4.8	3.6
	kNNimp1	7.1	7.6	6.7	7.6	8.2	8.6	8.2	7.8	7.4	7.8	7.6	8.6	8.2	7.8	7.4	7.8	7.6	8.8
	kNNimp3	6.5	6.6	5.7	6.1	6.2	7.2	6.8	6.0	7.4	6.0	6.2	7.6	6.8	6.4	7.4	6.0	6.2	7.6
	kNNimp5	5.9	6.2	5.1	5.3	6.6	6.0	6.0	4.8	6.4	5.0	5.0	6.6	6.0	5.2	6.4	5.0	5.0	6.6
	SVMimp	1.3	1.6	3.3	1.0	1.4	3.5	2.6	3.6	5.6	2.6	2.2	5.8	3.2	4.0	5.6	2.6	2.2	5.6
10%	EM	7.0	7.8	8.2	8.6	8.0	8.0	7.8	7.8	6.8	8.6	8.6	5.6	7.8	7.8	6.8	8.6	8.6	5.4
	MICE	4.6	5.2	5.5	3.8	5.0	4.2	4.4	5.4	4.4	1.8	3.2	3.4	4.2	5.4	4.4	1.8	3.2	3.6
	$SDAE2_{adadelta}$	4.1	3.0	3.5	3.3	3.0	2.4	1.2	1.8	1.0	4.4	3.4	1.4	1.0	1.2	1.0	4.4	3.4	1.4
	$SDAE2_{adam}$	3.7	3.2	2.9	4.1	3.5	2.6	2.2	2.4	2.0	5.4	4.0	2.4	2.0	1.8	2.0	5.4	4.0	2.4
	F_F	3.4673	7.3636	2.7114	9.4831	13.2166	18.7704	23.7778	6.0000	10.4928	8.5523	5.9338	15.2308	25.4118	10.7059	10.4928	8.5523	5.9338	15.8675
	meanimp	3.3	4.6	3.4	4.4	2.6	2.3	6.0	5.4	4.6	3.4	5.0	3.6	5.8	5.4	4.6	3.4	4.8	3.2
	kNNimp1	7.7	7.4	7.0	7.6	8.4	7.6	8.2	7.8	8.0	7.8	7.6	8.2	8.2	7.8	8.2	8.0	7.6	8.2
	kNNimp3	6.5	6.4	6.4	5.8	6.1	7.2	6.6	6.4	7.0	6.0	6.2	7.2	6.6	6.4	6.8	6.0	6.0	7.2
	kNNimp5	6.5	6.0	6.0	5.6	6.1	7.0	6.0	5.2	6.2	5.0	5.0	6.2	6.0	5.2	6.2	5.0	5.0	6.2
	SVMimp	1.7	1.4	3.2	1.0	1.6	3.1	2.8	3.6	5.2	2.4	2.2	5.4	3.4	4.0	5.2	2.4	2.4	5.8
15%	EM	7.8	7.8	7.4	8.6	8.2	8.2	7.8	7.8	6.6	8.6	8.6	5.2	7.8	7.8	6.6	8.4	8.6	5.2
	MICE	3.7	3.8	4.8	4.4	3.8	5.0	4.2	5.4	4.4	2.0	3.2	6.2	4.2	5.4	4.4	2.0	3.4	6.2
	$SDAE2_{adadelta}$	4.0	4.4	3.4	3.6	4.0	2.3	1.2	1.4	1.0	4.4	3.2	1.0	1.0	1.2	1.0	4.4	3.2	1.0
	$SDAE2_{adam}$	3.8	3.2	3.4	4.0	4.2	2.3	2.2	2.0	2.0	5.4	4.0	2.0	2.0	1.8	2.0	5.4	4.0	2.0
	F_F	6.6857	5.4937	2.6079	8.2449	11.6047	23.6498	21.0000	9.2743	9.7615	8.3967	6.2041	12.1290	21.2101	10.7059	10.1509	8.2449	5.3458	14.2927
	meanimp	3.6	3.3	3.0	4.6	2.9	3.2	6.0	5.2	4.8	3.4	5.2	3.6	5.8	5.2	5.0	3.4	5.2	3.4
	kNNimp1	7.4	7.7	7.0	7.2	8.4	8.6	8.2	7.8	8.2	7.8	7.8	8.2	8.2	7.8	8.2	8.2	8.0	8.6
	kNNimp3	6.6	6.9	7.0	6.4	6.5	7.2	7.0	6.6	6.4	6.2	6.2	7.2	7.2	6.6	6.4	6.2	6.0	7.2
	kNNimp5	5.8	6.1	6.6	5.2	6.6	6.2	5.8	5.2	5.8	5.0	5.0	6.2	5.8	5.2	5.8	4.8	5.0	6.2
	SVMimp	1.4	1.7	3.0	1.0	2.4	2.4	3.0	3.6	5.6	2.2	2.2	5.8	3.0	4.0	5.6	2.6	2.4	5.8
20%	EM	7.8	7.8	7.4	8.8	7.4	8.0	7.8	7.8	6.6	8.6	8.8	5.0	7.8	7.8	6.4	8.2	8.6	4.8
	MICE	5.2	4.3	5.0	4.8	3.8	5.0	4.2	5.4	4.6	2.0	3.2	6.0	4.2	5.4	4.6	1.8	3.2	6.0
	$SDAE2_{adadelta}$	3.2	3.2	3.0	3.5	3.6	2.2	1.0	1.4	1.0	4.4	3.0	1.0	1.0	1.2	1.0	4.4	3.0	1.0
	$SDAE2_{adam}$	4.0	4.0	3.0	3.5	3.4	2.2	2.0	2.0	2.0	5.4	3.6	2.0	2.0	1.8	2.0	5.4	3.6	2.0
	F_F	6.0000	7.1421	4.6705	9.3185	7.5718	35.4737	32.1446	9.6364	8.6050	9.4529	8.5000	12.1290	34.9610	11.1515	8.1951	8.6582	7.4068	16.8333
	meanimp	3.1	3.2	3.4	3.8	3.2	2.9	5.6	5.2	4.6	3.4	5.0	4.0	5.6	5.4	4.6	3.0	4.8	3.8
	kNNimp1	8.6	7.7	7.7	8.8	8.0	8.4	8.2	7.8	8.0	8.2	7.6	7.6	8.2	7.8	8.0	8.8	8.0	7.8
	kNNimp3	7.1	6.9	6.9	6.2	7.0	8.0	7.0	6.4	7.4	6.2	6.2	7.2	7.2	6.4	7.4	6.2	6.6	7.2
	kNNimp5	6.3	6.1	6.4	5.2	6.4	6.6	6.0	5.2	6.6	5.0	5.4	6.6	6.2	5.0	6.6	5.0	5.4	6.6
40%	SVMimp	1.3	1.7	3.4	1.0	1.0	2.7	3.0	4.0	5.4	2.4	2.0	6.4	3.2	4.0	5.4	3.0	2.8	6.8
	EM	7.6	7.8	7.0	8.2	8.0	7.0	7.8	7.8	5.6	8.2	8.4	3.8	7.8	7.8	5.6	8.2	8.4	3.4
	MICE	4.8	4.9	5.0	4.8	5.6	4.8	4.4	5.6	4.4	1.8	3.8	6.4	3.8	5.6	4.4	1.6	4.0	6.4
	$SDAE2_{adadelta}$	3.0	3.6	2.7	3.4	3.3	2.1	1.0	1.2	1.0	4.4	3.0	1.0	1.0	1.2	1.0	4.0	2.0	1.0
	$SDAE2_{adam}$	3.2	3.1	2.5	3.6	2.5	2.5	2.0	1.8	2.0	5.4	3.6	2.0	2.0	1.8	2.0	5.2	3.0	2.0
	F_F	21.0000	7.7994	5.0634	16.2703	28.8767	24.3019	29.3333	10.7783	10.4231	9.3333	6.1351	12.5746	38.8571	10.8515	10.4231	14.0723	9.4529	18.7273

Conclusions

In this 3rd experiment we study the performance of 2 approaches based on SDAE compared to other imputation methods, for datasets with high sample sizes. Moreover, we analyse the effect of different generation methods (*univa* and *unifo*) on several imputation approaches, both in terms of imputation quality (R^2 and RMSE) and classification performance (F-measure). The results show that there seems to be an advantage in using SDAE imputation methods for larger datasets. In general, the SDAE are ranked between 1st and 5th best method which is not the case for the 2nd experiment. In particular:

- SDAE belong to the top 3 best imputation approaches regarding classification performance, for almost all scenarios;
- Adadelta optimizer has shown to be the best optimizer in terms of imputation quality while Adam proved to be superior for classification performance;
- SVMimp has shown to be the best imputation approach for classification, in the MCAR and MAR scenarios; $SDAE2_{adam}$ and $SDAE2_{adadelta}$ also take a prominent position regardless the MR and the type of configuration (*univa* and *unifo*);
- Regarding imputation quality, $SDAE2_{adadelta}$ is the best approach for *univa* configurations, followed by $SDAE2_{adam}$; SVMimp (MCAR), MICE (MAR) and $SDAE2_{adadelta}$ (MNAR) are the best approaches under *unifo* configurations;
- In comparison to the results from the 2nd Experiment, there seems to be an advantage in using our SDAE-based approaches when dealing with MD for larger datasets.

In the context of this thesis, we were not able to explore a larger number of datasets due to the computational time required for the simulations. However, the results obtained are promising; with the aim of performing a more rigorous interpretation and generalization of these results, the realization of new experiments would be useful.

6

Conclusions

Data quality is a fundamental requirement to ensure a good performance of Data Mining models. In this regard, missing data arises as a common problem that affects the quality of the data. The scientific community has been studying several ways to handle the MD problem, however, the use of SDAE as imputation technique remains a underdeveloped topic.

In this work, we study the performance of SDAE-based approaches in order to provide some insights regarding three main research questions:

1. How do the SDAE perform when there are enough complete data to train the model?
2. How do the SDAE perform when training data corruption follows an underlying missing mechanism?
3. Does the performance of SDAE increase for larger datasets, with higher sample sizes?

We conducted three main experiments designed to answer these research questions. According to the results obtained from the 1^{st}, 2^{nd} and 3^{rd} experiments, three main conclusions may be derived for each question:

1. $DAE1_{adadelta}$ shows to be superior to the remaining methods, for MNAR mechanism and high missing rates. However, the SDAE-based approaches explored in this experiment are limited: it is assumed that there are enough complete samples to train each model.
2. The SDAE show to be superior in many of the studied scenarios. However, this superiority may not be sufficient to counteract the limitations of SDAE-based approaches, especially, when they are used in a multiple imputation context, since they require a high computational effort and memory space.

3. For datasets with larger sample sizes, there seems to be a great advantage in using SDAE for imputation purpose.

Considering the results obtained in this work, one of the possible directions for future work is to investigate the usefulness of SDAE for a larger benchmark of datasets. Also, as the advantage of SDAE seems to be more clear for higher missing rates (40%), a smoother step of missing rates (between 20% and 40%) could bring new insights. Other future direction is hyperparameter optimization of the SDAE's architecture (e.g., learning rate, #hidden layers, #nodes per hidden layer etc.) to ease their computational cost.

Bibliography

[1] G. Piatetsky-Shapiro, “Knowledge discovery in real databases: A report on the IJCAI-89 Workshop”, *AI magazine*, vol. 11, no. 4, p. 68, 1990.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases”, *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[3] P. H. Abreu, H. Amaro, D. C. Silva, P. Machado, and M. H. Abreu, “Personalizing breast cancer patients with heterogeneous data”, in *The International Conference on Health Informatics*, pp. 39–42, Springer, 2014.

[4] P. H. Abreu, H. Amaro, D. C. Silva, P. Machado, M. H. Abreu, N. Afonso, and A. Dourado, “Overall survival prediction for women breast cancer using ensemble methods and incomplete clinical data”, in *XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013*, pp. 1366–1369, Springer, 2014.

[5] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, “Pattern classification with missing data: a review”, *Neural Computing and Applications*, vol. 19, pp. 263–282, 2009.

[6] P. J. García-Laencina, P. H. Abreu, M. H. Abreu, and N. Afonoso, “Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values”, *Computers in biology and medicine*, vol. 59, pp. 125–133, 2015.

[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, *Nature*, vol. 521, no. 7553, p. 436, 2015.

[8] L. Gondara and K. Wang, “Recovering Loss to Followup Information Using Denoising Autoencoders”, Simon Fraser University, 2017.

[9] L. Gondara and K. Wang, “Multiple Imputation Using Deep Denoising Autoencoders”, Department of Computer Science, Simon Fraser University, 2018.

[10] Y. Duan, Y. Lv, W. Kang, and Y. Zhao, "A deep learning based approach for traffic data imputation", in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014.

[11] Y. Duan, Y. Lv, Y. Liu, and F. Wang, "An efficient realization of deep learning for traffic data imputation", *Transportation research part C: emerging technologies*, vol. 72, pp. 168–181, 2016.

[12] X. Ning, Y. Xu, X. Gao, and Y. Li, "Missing data of quality inspection imputation algorithm base on stacked denoising auto-encoder", in *IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pp. 84–88, 2017.

[13] A. Sánchez-Morales, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Values Deletion to Improve Deep Imputation Processes", *Biomedical Applications Based on Natural and Artificial Computing (IWINAC). Lecture Notes in Computer Science*, vol. 10338, pp. 240–246, 2017.

[14] H. Kang, "The prevention and handling of the missing data", *Korean journal of anesthesiology*, vol. 64, no. 5, pp. 402–406, 2013.

[15] P. J. García-Laencina, J. Sancho-Gomez, and A. R. Figueiras-Vidal, "Pattern classification with missing values using multitask learning", in *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 3594–3601, 2006.

[16] R. Pan, T. Yang, J. Cao, K. Lu, and Z. Zhang, "Missing data imputation by K nearest neighbours based on grey relational structure and mutual information", *Applied Intelligence*, vol. 43, no. 3, pp. 614–632, 2015.

[17] D. B. Rubin, "Inference and missing data", *Biometrika*, vol. 63, pp. 581–592, 1976.

[18] C. K. Enders, *Applied missing data analysis.* Guilford Press, 2010.

[19] B. Twala, "An Empirical Comparison of Techniques for Handling Incomplete Data Using Decision Trees", *Applied Artificial Intelligence*, vol. 23, pp. 373–405, 2009.

[20] A. Rieger, T. Hothorn, and C. Strobl, "Random Forests with Missing Values in the Covariates", Department of Statistics, University of Munich, 2010.

[21] J. Xia, S. Zhang, G. Cai, L. Li, Q. Pan, J. Yan, and G. Ning, "Adjusted weight voting algorithm for random forests in handling missing values", *Pattern Recognition*, vol. 69, pp. 52–60, 2017.

[22] L. Nanni, A. Lumini, and S. Brahnam, "A classifier ensemble approach for the missing feature problem", *Artificial intelligence in medicine*, vol. 55, no. 1, pp. 37–50, 2012.

[23] U. Garciarena and R. Santana, "An extensive analysis of the interaction between missing data types, imputation methods, and supervised classifiers", *Expert Systems with Applications*, vol. 89, pp. 52–65, 2017.

[24] N. A. Ali and Z. M. Omer, "Improving accuracy of missing data imputation in data mining", *Kurdistan Journal of Applied Research*, vol. 2, no. 3, pp. 66–73, 2017.

[25] J. Josse, M. E. Timmerman, and H. A. Kiers, "Missing values in multi-level simultaneous component analysis", *Chemometrics and Intelligent Laboratory Systems*, vol. 129, pp. 21–32, 2013.

[26] B. Zhu, C. He, and P. Liatsis, "A robust missing value imputation method for noisy data", *Applied Intelligence*, vol. 36, no. 1, pp. 61–74, 2012.

[27] S. Van Buuren, *Flexible imputation of missing data.* CRC press, 2012.

[28] M. S. Santos, P. H. Abreu, P. J. García-Laencina, A. Simão, and A. Carvalho, "A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients", *Journal of biomedical informatics*, vol. 58, pp. 49–59, 2015.

[29] G. Batista and 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", *University of São Paulo*, vol. 34, 2003.

[30] A. Rubinsteyn and S. Feldman, "fancyimpute", March 2016.

[31] S. Amari and S. Wu, "Improving Support Vector Machine classifiers by modifying kernel functions", *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[32] T. E. Raghunathan, J. M. Lepkowski, J. Van Hoewyk, and P. Solenberger, "A multivariate technique for multiply imputing missing values using a sequence of regression models", *Survey methodology*, vol. 27, no. 1, pp. 85–96, 2001.

[33] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in R", *Journal of statistical software*, pp. 1–68, 2010.

[34] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, "Multiple Imputation by Chained Equations: What is it and how does it work?", *International Journal of Methods in Psychiatric Research*, vol. 20, pp. 40–49, 2011.

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

[36] J. W. Graham, "Missing data analysis: Making it work in the real world", *Annual review of psychology*, vol. 60, pp. 549–576, 2009.

[37] E. Law, "impyute", May 2017.

[38] F. V. Nelwamondo, S. Mohamed, and T. Marwala, "Missing data: A comparison of neural network and expectation maximization techniques", *Current Science*, pp. 1514–1521, 2007.

[39] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders", in *International Conference on Machine Learning proceedings*, 2008.

[40] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines", *Elsevier*, vol. 44, pp. 78–96, 2018.

[41] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[42] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[44] H. Robbins and S. Monro, "A stochastic approximation method", *The annals of mathematical statistics*, pp. 400–407, 1951.

[45] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[46] M. D. Zeiler, “ADADELTA: an adaptive learning rate method”, *arXiv preprint arXiv:1212.5701*, 2012.

[47] T. Tieleman and G. Hinton, “Lecture 6.5-RMSProp, COURSERA: Neural networks for machine learning”, *University of Toronto, Technical Report*, 2012.

[48] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.

[49] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, “An up-to-date comparison of state-of-the-art classification algorithms”, *Expert Systems with Applications*, vol. 82, pp. 128–150, 2017.

[50] J. M. De Sa, *Pattern recognition: concepts, methods and applications.* Springer Science & Business Media, 2012.

[51] P. H. Abreu, M. S. Santos, M. H. Abreu, B. Andrade, and D. C. Silva, “Predicting breast cancer recurrence using machine learning techniques: a systematic review”, *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 52, 2016.

[52] R. Chambers, “Evaluation criteria for editing and imputation in EUREDIT”, *Statistical Data Editing*, vol. 3, 2006.

[53] J. Demšar, “Statistical comparisons of classifiers over multiple data sets”, *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[54] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”, *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[55] S. García, A. Fernández, J. Luengo, and F. Herrera, “A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability”, *Soft Computing*, vol. 13, no. 10, p. 959, 2009.

[56] R. L. Iman and J. M. Davenport, “Approximations of the critical region of the fbietkan statistic”, *Communications in Statistics-Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980.

[57] J. Marôco, *Análise Estatística com o SPSS Statistics.: 7^a edição.* ReportNumber, Lda, 2018.

[58] A. Kolmogorov, “On the empirical determination of a distribution function”, in *Breakthroughs in statistics*, pp. 106–113, Springer, 1992.

[59] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)", *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[60] H. Levene *et al.*, "On a matching problem arising in genetics", *The annals of mathematical statistics*, vol. 20, no. 1, pp. 91–94, 1949.

[61] F. Wilcoxon, "Individual comparisons by ranking methods", *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[62] R. Woolson, "Wilcoxon signed-rank test", *Wiley encyclopedia of clinical trials*, pp. 1–3, 2007.

[63] B. K. Beaulieu-Jones and J. H. Moore, "Missing Data Imputation in the Electronic Health Record Using Deeply Learned Autoencoders", in *Pacific Symposium on Biocomputing (PSB)*, pp. 207–218, 2017.

[64] B. K. Beaulieu-Jones, C. S. Greene, *et al.*, "Semi-supervised learning of the electronic health record for phenotype stratification", *Journal of biomedical informatics*, vol. 64, pp. 168–178, 2016.

[65] G. Sang, K. Shi, Z. Liu, and L. Gao, "Missing Data Imputation Based on Grey System Theory", *International Journal of Hybrid Information Technology*, vol. 27, no. 2, pp. 347–355, 2014.

[66] Z. Man-long, "MkNNI: New Missing Value Imputation Method Using Mutual Nearest Neighbor", *Modern Computer*, vol. 31, p. 1, 2012.

[67] B. D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[68] K. Napierala and J. Stefanowski, "Types of minority class examples and their influence on learning classifiers from imbalanced data", *Journal of Intelligent Information Systems*, vol. 46, no. 3, pp. 563–597, 2016.

[69] M. S. Grosof and H. Sardy, *A research primer for the social and behavioral sciences.* Academic Press, 2014.

[70] F. Chollet, "Keras." `https://github.com/keras-team/keras`, 2018.

[71] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions", *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[72] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O (1/k2)", in *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[73] M. S. Santos, J. P. Soares, P. H. Abreu, H. Araújo, and J. Santos, "Influence of Data Distribution in Missing Data Imputation", in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 285–294, Springer, 2017.

Appendices

A

Synthetic generation of MD

Table A.1: Functions that are used as building blocks of the pseudo-codes.

Function	**Description**
bernoulli(mr, size = int)	Returns a Bernoulli distribution of a specified size and a probability equal to MR.
range(start, stop)	Generates an array of natural numbers starting from *start* to *stop* with an increment of 1. The output will not include *stop*.
count(data)	Returns the total number of missing values in *data*, a matrix that represents the dataset.
group(array, threshold = float)	Takes an *array* as input and divides it into two groups that will contain the values below and above the *threshold*.
median(array)	Returns the median of an *array* of integers.
numFeatures(data)	Returns the number of features in *data*.
numObservations(data)	Returns the number of observations in *data*.
random(b, size = int, p = array)	Takes two or three input parameters depending on how the random choice of values will be made. If the first input parameter is a number, the values will be randomly chosen from the range between zero and that number. If the first input parameter is an array, a random choice of its values will be made. The second input parameter is mandatory and represents the number of values that will be selected. Finally, *p* is not mandatory and can only be used when the first input is an array: this parameter is an *array* which contains the probability for each element of the first input to be selected.
random_select(array)	Takes one *array* or two arrays as input. If the input is an array, this function will return one or more elements of it, depending on the number of outputs defined by the user. If the input consists of two arrays, the function will return one of them.
rank(array)	Returns an *array* with the computed numerical ranks (1 through *n*, being *n* the total number of elements in *array*) of the input *array*.
round(float)	Returns the floating point value number rounded to zero digits after the decimal point.
remove(array, value)	This function will remove *value* from *array*.
select_by_correlation(array, data)	Given an *array* containing indexes of two features and the dataset matrix *data*, returns ordered indexes in terms of correlation with the class labels: the first index will correspond to the feature with lowest correlation with the class labels and the second one will correspond to the most correlated feature with the class labels.
size(array)	Returns the number of elements in an array.
sort(array, reverse = boolean)	Sorts the elements of the input *array* in a specific order: ascending or descending. If reverse is equal to False, descending order will be performed (ascending otherwise). Returns an *array* with the sorted values of the input *array*.

B

Data Collection

Table B.1: Additional information on the datasets used in this study.

	Dataset	**Source**	**Context**	**Positive Class**	**Negative Class**
1st and 2nd Experiment	australian	UCI	Credit card applications.	1	0
	banknote	UCI	Data extracted from banknote images.	1	0
	biomed	STATLIB	Blood measurements database.	carrier	normal
	breast-ljub	UCI	Breast cancer data.	recurrence-events	no-recurrence-events
	breast-tissue	UCI	Impedance measurements of tissue from the breast.	positive	negative
	cleveland	KEEL	Heart disease database.	presence	absence
	crabs	PRNN	Morphological features of crabs.	B	O
	dermatology1	UCI	Clinical features of erythema and scaling.	4	5
	ecoli	KEEL	Measurements about the cell to predict the location site of proteins.	positive	negative
	glass1	KEEL	Information about 6 types of glass.	non-window	window
	heart-statlog	UCI	Heart disease database.	2	1
	iris	KEEL	Iris plant database.	positive	negative
	kidney	UCI	Chronic kidney disease database.	ckd	notckd
	lung-cancer	UCI	Lung cancer database.	2	1
	lymphography	KEEL	Lymphoma detection.	malign_lymph	metastases
	postoperative	UCI	Patient features used to determine whether a patient should be moved from na area to another.	S	A
	saheart	KEEL	South African heart database.	1	0
	urinary	UCI	Acute inflammation of urinary bladder database.	yes	no
	wine1	UCI	Chemical analysis of wines.	1	2
	wpbc	UCI	Follow-up data for breast cancer cases (prognostic).	R	N
2nd Experiment	balancescale	UCI	Balance scale weight and distance database	L	R
	bankrupcy	UCI	Qualitative parameters to predict the bankrupcy.	B	NB
	cmc	UCI	Contraceptive method choice database.	L	S
	dermatology2	UCI	Clinical features of erythema and scaling.	1	3
	edu-data1	KAGGLE	Students' academic performance database.	H	L
	edu-data2	KAGGLE		H	M
	glass2	KEEL	Information about 6 types of glass.	non-float processed	remaining
	hcc-data-mortality	CISUC	Clinical features of real patients diagnosed with HCC.	dies	lives
	hcc-data-survival	CISUC		lives	dies
	hepato	ISICAL	Hepatobiliary disorders database.	PH	LC
	new-thyroid	KEEL	Thyroid disease database.	1	2
	toy	PRNN	Synthetic dataset composed of five Gaussian components.	1	0
	wine2	UCI	Chemical analysis of wines.	3	1
	wine3	UCI		3	2
3rd Experiment	mushrooms	UCI	Mushrooms description regarding of physical characteristics.	p	e
	nursery1	UCI	Ranking applications for nursery schools.	spec_prior	not_recom
	nursery2	UCI		priority	not_recom
	thyroid1	KEEL	Thyroid disease database - data from 10 different databases.	368	6666
	thyroid2	KEEL		6666	368

C

Correlation Results

Table C.1: Missing and observed features for each dataset - univariate missing data generation (*univa*).

	Dataset	x_{miss}	**Correlation Measure**	**Value**	x_{obs}
1st and 2nd Experiment	australian	8	Phi-Coefficient	0.7204	9
	banknote	1	Point-Biserial	0.7248	3
	biomed	5	Point-Biserial	0.6066	4
	breast-ljub	5	Phi-Coefficient	0.2890	4
	breast-tissue	2	Point-Biserial	0.7251	3
	cleveland	11	Cramer's V	0.5269	10
	crabs	2	Point-Biserial	0.4380	6
	dermatology1	34	Point-Biserial	0.0478	6
	ecoli	4	Point-Biserial	0.9043	1
	glass1	3	Point-Biserial	0.7583	8
	heart-statlog	11	Cramer's V	0.5255	10
	iris	3	Point-Biserial	0.9227	4
	kidney	4	Cramer's V	0.9840	19
	lung-cancer	20	Cramer's V	0.6814	1
	lymphography	13	Cramer's V	0.6107	15
	postoperative	6	Cramer's V	0.1885	2
	saheart	9	Point-Biserial	0.3730	4
	urinary	4	Phi-Coefficient	0.6954	6
	wine1	13	Point-Biserial	0.8453	10
	wpbc	1	Point-Biserial	0.3513	4
2nd Experiment	balancescale	2	Point-Biserial	0.4283	3
	bankrupcy	5	Cramer's V	0.9692	3
	cmc	1	Point-Biserial	0.2726	4
	dermatology2	11	Phi-Coefficient	0.3624	31
	edu-data1	11	Spearman rho	0.7980	10
	edu-data2	10	Spearman rho	0.3863	12
	glass2	2	Point-Biserial	0.2739	8
	hcc-data-mortality	4	Point-Biserial	0.1461	1
	hcc-data-survival	4	Point-Biserial	0.1461	1
	hepato	1	Point-Biserial	0.2090	2
	new-thyroid	2	Point-Biserial	0.9092	3
	toy	2	Point-Biserial	0.7559	1
	wine2	7	Point-Biserial	0.9521	6
	wine3	10	Point-Biserial	0.7944	11
3rd Experiment	mushrooms	5	Cramer's V	0.9710	15
	nursery1	8	Cramer's V	1.0000	2
	nursery2	8	Cramer's V	1.0000	2
	thyroid1	17	Point-Biserial	0.4167	21
	thyroid2	17	Point-Biserial	0.4167	21

D

Parameter Optimization

Table D.1: Optimal parameters C and γ for the different datasets as well as the average accuracy. C and γ were obtained through grid search. Average accuracy is computed over a 5-fold cross-validation procedure, for each combination of these 2 parameters.

	Dataset	**C**	γ	**Average Accuracy**
1st and 2nd Experiment	australian	1	1×10^{-3}	0.8812
	banknote	1	1	1.0000
	biomed	1×10^{4}	1×10^{-3}	0.9128
	breast-ljub	1×10^{3}	1×10^{-3}	0.7536
	breast-tissue	1×10^{1}	1×10^{-2}	0.9636
	cleveland	1×10^{1}	1×10^{-2}	0.8433
	crabs	1×10^{1}	1×10^{-1}	1.0000
	dermatology1	1×10^{-3}	1×10^{-5}	1.0000
	ecoli	1×10^{2}	1×10^{-2}	0.9864
	glass1	1	1×10^{-1}	0.9302
	heart-statlog	1	1×10^{-2}	0.8185
	iris	1×10^{-1}	1×10^{-1}	1.0000
	kidney	1×10^{-1}	1×10^{-1}	1.0000
	lung-cancer	1×10^{1}	1×10^{-3}	0.8667
	lymphography	1×10^{2}	1×10^{-3}	0.8759
	postoperative	1×10^{-3}	1×10^{-5}	0.7222
	saheart	1×10^{7}	1×10^{-5}	0.7355
	urinary	1×10^{-1}	1×10^{-1}	1.0000
	wine1	1×10^{2}	1×10^{-2}	0.9923
	wpbc	1×10^{1}	1×10^{-1}	0.8150
2nd Experiment	balancescale	1×10^{3}	1×10^{-2}	1.0000
	bankrupcy	1×10^{-1}	1×10^{-1}	0.9960
	cmc	1×10^{6}	1×10^{-4}	0.6840
	dermatology2	1×10^{-1}	1×10^{-2}	1.0000
	edu-data1	1×10^{-1}	1×10^{-2}	0.9704
	edu-data2	1×10^{4}	1×10^{-5}	0.8310
	glass2	1	1	0.8000
	hcc-data-mortality	1×10^{4}	1×10^{-2}	0.7030
	hcc-data-survival	1×10^{1}	1×10^{-3}	0.6545
	hepato	1×10^{2}	1	0.8557
	new-thyroid	1×10^{-1}	1×10^{-1}	1.0000
	toy	1×10^{6}	1×10^{-1}	0.9040
	wine2	1×10^{-1}	1×10^{-2}	1.0000
	wine3	1	1×10^{-2}	0.9917
3rd Experiment	mushrooms	1	1×10^{-1}	1.0000
	nursery1	1×10^{-2}	1×10^{-1}	1.0000
	nursery2	1×10^{-1}	1×10^{-2}	1.0000
	thyroid1	1×10^{4}	1×10^{-2}	0.9908
	thyroid2	1×10^{4}	1×10^{-3}	0.9912

E

Experimental Results

Table E.1: Effect of different univariate (*univa*) configurations in the classification performance (F-measure) using imputation methods from the state of the art.

		F-measure − *univa*								
MR	**Methods**	**MCAR1**	**MCAR2**	**MAR1**	**MAR2**	**MAR3**	**MAR4**	**MAR5**	**MNAR1**	**MNAR2**
	Meanimp	0.7731	0.7693	0.7764	0.7714	0.7735	0.7698	0.7716	0.7690	0.7662
	kNNimp1	0.7735	0.7700	0.7721	0.7725	0.7720	0.7729	0.7720	0.7734	0.7696
	kNNimp3	0.7735	0.7713	0.7728	0.7729	0.7734	0.7737	0.7734	0.7739	0.7700
	kNNimp5	0.7737	0.7717	0.7731	0.7728	0.7730	0.7732	0.7729	0.7715	0.7707
5%	SVMimp	0.7747	0.7741	0.7721	0.7749	0.7752	0.7758	0.7725	0.7775	0.7717
	EM	0.7696	0.7684	0.7727	0.7704	0.7718	0.7712	0.7703	0.7688	0.7663
	MICE	0.7740	0.7713	0.7747	0.7732	0.7738	0.7749	0.7739	0.7699	0.7714
	Meanimp	0.7695	0.7699	0.7745	0.7683	0.7699	0.7688	0.7738	0.7655	0.7618
	kNNimp1	0.7726	0.7707	0.7723	0.7713	0.7691	0.7671	0.7718	0.7674	0.7631
	kNNimp3	0.7721	0.7727	0.7712	0.7723	0.7718	0.7714	0.7726	0.7677	0.7641
	kNNimp5	0.7725	0.7729	0.7732	0.7729	0.7711	0.7738	0.7741	0.7684	0.7626
10%	SVMimp	0.7760	0.7750	0.7713	0.7758	0.7746	0.7762	0.7742	0.7683	0.7649
	EM	0.7674	0.7670	0.7696	0.7667	0.7665	0.7658	0.7683	0.7639	0.7626
	MICE	0.7716	0.7717	0.7733	0.7714	0.7714	0.7702	0.7759	0.7652	0.7662
	Meanimp	0.7673	0.7666	0.7741	0.7659	0.7680	0.7717	0.7706	0.7659	0.7530
	kNNimp1	0.7705	0.7673	0.7695	0.7690	0.7681	0.7678	0.7726	0.7674	0.7578
	kNNimp3	0.7709	0.7683	0.7720	0.7706	0.7709	0.7737	0.7708	0.7666	0.7627
	kNNimp5	0.7711	0.7688	0.7736	0.7701	0.7698	0.7745	0.7709	0.7656	0.7634
15%	SVMimp	0.7747	0.7739	0.7705	0.7743	0.7768	0.7745	0.7773	0.7713	0.7631
	EM	0.7640	0.7593	0.7712	0.7624	0.7630	0.7657	0.7673	0.7658	0.7553
	MICE	0.7698	0.7685	0.7728	0.7683	0.7706	0.7726	0.7738	0.7671	0.7624
	Meanimp	0.7674	0.7642	0.7688	0.7639	0.7640	0.7709	0.7676	0.7617	0.7561
	kNNimp1	0.7666	0.7648	0.7693	0.7656	0.7633	0.7659	0.7645	0.7627	0.7528
	kNNimp3	0.7699	0.7684	0.7693	0.7677	0.7655	0.7726	0.7661	0.7633	0.7581
	kNNimp5	0.7707	0.7685	0.7704	0.7696	0.7646	0.7726	0.7691	0.7620	0.7580
20%	SVMimp	0.7764	0.7742	0.7701	0.7782	0.7748	0.7727	0.7715	0.7689	0.7649
	EM	0.7625	0.7576	0.7668	0.7602	0.7582	0.7646	0.7634	0.7632	0.7567
	MICE	0.7705	0.7660	0.7701	0.7703	0.7657	0.7698	0.7714	0.7586	0.7633
	Meanimp	0.7586	0.7542	0.7652	0.7522	0.7526	0.7553	0.7652	0.7515	0.7480
	kNNimp1	0.7578	0.7517	0.7641	0.7517	0.7509	0.7566	0.7647	0.7482	0.7508
	kNNimp3	0.7617	0.7559	0.7673	0.7562	0.7564	0.7569	0.7680	0.7507	0.7549
	kNNimp5	0.7617	0.7595	0.7641	0.7570	0.7573	0.7602	0.7711	0.7496	0.7511
40%	SVMimp	0.7756	0.7712	0.7687	0.7759	0.7758	0.7677	0.7674	0.7594	0.7601
	EM	0.7507	0.7472	0.7587	0.7487	0.7483	0.7525	0.7581	0.7480	0.7452
	MICE	0.7612	0.7603	0.7693	0.7578	0.7581	0.7603	0.7618	0.7464	0.7508

Table E.2: Ranks of the effect of different univariate (*univa*) configurations in the classification performance (F-measure) using imputation methods from the state of the art.

		F-measure − ***univa***								
MR	**Methods**	**MCAR1**	**MCAR2**	**MAR1**	**MAR2**	**MAR3**	**MAR4**	**MAR5**	**MNAR1**	**MNAR2**
5%	Meanimp	3	7	1	5	2	6	4	8	9
	kNNimp1	1	8	5	4	7	3	6	2	9
	kNNimp3	3	8	7	6	5	2	4	1	9
	kNNimp5	1	7	3	6	4	2	5	8	9
	SVMimp	5	6	8	4	3	2	7	1	9
	EM	6	8	1	4	2	3	5	7	9
	MICE	3	8	2	6	5	1	4	9	7
	MEAN	**3.143**	7.429	3.857	5.000	4.000	**2.714**	5.000	**5.143**	8.714
10%	Meanimp	5	4	1	7	3	6	2	8	9
	kNNimp1	1	5	2	4	6	8	3	7	9
	kNNimp3	4	1	7	3	5	6	2	8	9
	kNNimp5	6	5	3	4	7	2	1	8	9
	SVMimp	2	4	7	3	5	1	6	8	9
	EM	3	4	1	5	6	7	2	8	9
	MICE	4	3	2	6	5	7	1	9	8
	MEAN	**3.571**	3.714	3.286	4.571	5.286	5.286	**2.429**	**8.000**	8.857
15%	Meanimp	5	6	1	8	4	2	3	7	9
	kNNimp1	2	8	3	4	5	6	1	7	9
	kNNimp3	4	7	2	6	3	1	5	8	9
	kNNimp5	3	7	2	5	6	1	4	8	9
	SVMimp	3	6	8	5	2	4	1	7	9
	EM	5	8	1	7	6	4	2	3	9
	MICE	5	6	2	7	4	3	1	8	9
	MEAN	**3.857**	6.857	2.714	6.000	4.286	3.000	**2.429**	**6.857**	9.000
20%	Meanimp	4	5	2	7	6	1	3	8	9
	kNNimp1	2	5	1	4	7	3	6	8	9
	kNNimp3	2	4	3	5	7	1	6	8	9
	kNNimp5	2	6	3	4	7	1	5	8	9
	SVMimp	2	4	7	1	3	5	6	8	9
	EM	5	8	1	6	7	2	3	4	9
	MICE	2	6	4	3	7	5	1	9	8
	MEAN	**2.714**	5.429	3.000	4.286	6.286	**2.571**	4.286	**7.571**	8.857
40%	Meanimp	3	5	2	7	6	4	1	8	9
	kNNimp1	3	6	2	5	7	4	1	9	8
	kNNimp3	3	7	2	6	5	4	1	9	8
	kNNimp5	3	5	2	7	6	4	1	9	8
	SVMimp	3	4	5	1	2	6	7	9	8
	EM	4	8	1	5	6	3	2	7	9
	MICE	3	5	1	7	6	4	2	9	8
	MEAN	**3.143**	5.714	**2.143**	5.429	5.429	4.143	**2.143**	**8.571**	8.286

Table E.3: Effect of different univariate (*univa*) configurations in imputation quality (R^2 and RMSE) using imputation methods from the state of the art.

		univa																	
		RMSE									**R^2**								
MR	**Methods**	**MCAR1**	**MCAR2**	**MAR1**	**MAR2**	**MAR3**	**MAR4**	**MAR5**	**MNAR1**	**MNAR2**	**MCAR1**	**MCAR2**	**MAR1**	**MAR2**	**MAR3**	**MAR4**	**MAR5**	**MNAR1**	**MNAR2**
5%	Meanimp	0.2271	0.2194	0.2646	0.2278	0.2247	0.3379	0.3201	0.3677	0.4683	0.9344	0.9379	0.9181	0.9343	0.9358	0.8749	0.8838	0.8569	0.7764
	kNNimp1	0.2160	0.2162	0.2258	0.2192	0.2222	0.2709	0.2658	0.3157	0.3957	0.9407	0.9406	0.9365	0.9393	0.9381	0.9150	0.9190	0.8907	0.8383
	kNNimp3	0.1854	0.1814	0.2014	0.1870	0.1874	0.2511	0.2435	0.2923	0.3722	0.9519	0.9532	0.9460	0.9513	0.9511	0.9244	0.9289	0.9042	0.8551
	kNNimp5	0.1809	0.1747	0.1890	0.1796	0.1810	0.2506	0.2396	0.2863	0.3716	0.9535	0.9555	0.9506	0.9539	0.9534	0.9248	0.9303	0.9076	0.8558
	SVMimp	0.5127	0.4520	0.5228	0.5424	0.4814	0.5377	0.8098	0.5566	0.7711	0.9296	0.9337	0.9255	0.9307	0.9322	0.8733	0.8894	0.8927	0.8283
	EM	0.3044	0.2991	0.3218	0.3061	0.3005	0.3850	0.3736	0.4117	0.4952	0.8962	0.8992	0.8865	0.8952	0.8983	0.8431	0.8494	0.8252	0.7526
	MICE	0.1741	0.1714	0.1860	0.1726	0.1801	0.2370	0.2105	0.2810	0.3438	0.9561	0.9567	0.9519	0.9565	0.9537	0.9392	0.9452	0.9103	0.8785
10%	Meanimp	0.3163	0.3086	0.3806	0.3212	0.3227	0.4404	0.4195	0.4972	0.6054	0.8866	0.8912	0.8501	0.8843	0.8834	0.8030	0.8113	0.7621	0.6519
	kNNimp1	0.3151	0.3091	0.3145	0.3213	0.3144	0.4054	0.3508	0.4334	0.5013	0.8923	0.8954	0.8921	0.8888	0.8924	0.8398	0.8688	0.8148	0.7599
	kNNimp3	0.2644	0.2559	0.2686	0.2723	0.2647	0.3534	0.3214	0.3944	0.4835	0.9174	0.9215	0.9159	0.9137	0.9173	0.8706	0.8852	0.8439	0.7753
	kNNimp5	0.2563	0.2461	0.2603	0.2626	0.2577	0.3386	0.3126	0.3903	0.4816	0.9213	0.9260	0.9199	0.9184	0.9207	0.8791	0.8903	0.8470	0.7771
	SVMimp	0.6993	0.6504	0.6324	0.7285	0.7318	0.5349	1.6003	0.6140	1.1444	0.8998	0.9060	0.8874	0.8995	0.8979	0.8573	0.8384	0.8318	0.7386
	EM	0.4199	0.4153	0.4633	0.4121	0.4250	0.5089	0.4985	0.5445	0.6308	0.8189	0.8219	0.7873	0.8244	0.8147	0.7446	0.7464	0.7172	0.6237
	MICE	0.2502	0.2454	0.2562	0.2533	0.2450	0.3007	0.3021	0.3798	0.4514	0.9244	0.9265	0.9230	0.9231	0.9274	0.9078	0.9053	0.8541	0.8068
15%	Meanimp	0.3869	0.3859	0.4699	0.3937	0.3982	0.5235	0.4952	0.6044	0.7057	0.8374	0.8379	0.7864	0.8345	0.8316	0.7351	0.7438	0.6730	0.5544
	kNNimp1	0.3787	0.3814	0.3822	0.3812	0.3799	0.4937	0.4273	0.5252	0.5854	0.8520	0.8508	0.8486	0.8514	0.8513	0.7807	0.8156	0.7472	0.6908
	kNNimp3	0.3242	0.3205	0.3332	0.3188	0.3266	0.4053	0.3795	0.4796	0.5748	0.8832	0.8856	0.8794	0.8867	0.8818	0.8337	0.8480	0.7861	0.7027
	kNNimp5	0.3121	0.3084	0.3260	0.3095	0.3164	0.3957	0.3707	0.4703	0.5692	0.8902	0.8925	0.8839	0.8919	0.8878	0.8402	0.8542	0.7942	0.7079
	SVMimp	0.8036	0.8455	1.1006	0.8017	0.7873	0.8561	0.8797	0.8310	0.9695	0.8746	0.8754	0.8511	0.8734	0.8662	0.7903	0.8255	0.7809	0.6635
	EM	0.5098	0.5121	0.5576	0.5164	0.5207	0.5945	0.5873	0.6458	0.7185	0.7433	0.7407	0.7076	0.7379	0.7351	0.6653	0.6588	0.6255	0.5361
	MICE	0.2997	0.3011	0.3179	0.3052	0.3066	0.3804	0.3541	0.4553	0.5374	0.8978	0.8967	0.8911	0.8948	0.8939	0.8637	0.8751	0.8070	0.7428
20%	Meanimp	0.4421	0.4417	0.5412	0.4605	0.4636	0.5839	0.5554	0.7027	0.7889	0.7918	0.7922	0.7303	0.7804	0.7785	0.6801	0.6821	0.5864	0.4968
	kNNimp1	0.4313	0.4429	0.4659	0.4396	0.4404	0.5322	0.5088	0.5982	0.6621	0.8155	0.8081	0.7960	0.8094	0.8081	0.7367	0.7564	0.6924	0.6263
	kNNimp3	0.3676	0.3687	0.3904	0.3759	0.3796	0.4550	0.4433	0.5677	0.6456	0.8545	0.8541	0.8418	0.8490	0.8468	0.7938	0.8014	0.7227	0.6475
	kNNimp5	0.3547	0.3545	0.3718	0.3630	0.3699	0.4469	0.4262	0.5605	0.6469	0.8627	0.8631	0.8550	0.8574	0.8531	0.8016	0.8137	0.7310	0.6449
	SVMimp	0.8546	0.9983	1.0076	0.9954	0.8850	1.3102	0.7093	0.9316	1.6054	0.8481	0.8463	0.8078	0.8413	0.8374	0.7749	0.7846	0.7185	0.6049
	EM	0.5894	0.5910	0.6364	0.5977	0.5975	0.6701	0.6575	0.7297	0.7942	0.6674	0.6658	0.6348	0.6608	0.6611	0.5892	0.5804	0.5478	0.4640
	MICE	0.3502	0.3460	0.3809	0.3549	0.3533	0.4897	0.3948	0.5356	0.6133	0.8659	0.8684	0.8548	0.8629	0.8652	0.8147	0.8445	0.7506	0.6851
40%	Meanimp	0.6322	0.6355	0.7893	0.6667	0.6793	0.8045	0.7262	1.0263	1.0630	0.5915	0.5863	0.5004	0.5730	0.5651	0.4643	0.4737	0.3295	0.2809
	kNNimp1	0.6298	0.6317	0.7169	0.6488	0.6436	0.7349	0.7138	0.9097	0.9341	0.6544	0.6481	0.5728	0.6379	0.6363	0.5424	0.5630	0.4425	0.3828
	kNNimp3	0.5362	0.5381	0.6382	0.5505	0.5531	0.6597	0.6034	0.8850	0.9102	0.7170	0.7130	0.6438	0.7061	0.7010	0.6048	0.6452	0.4785	0.4254
	kNNimp5	0.5192	0.5207	0.6347	0.5317	0.5365	0.6486	0.5820	0.8914	0.9106	0.7299	0.7267	0.6501	0.7210	0.7150	0.6174	0.6660	0.4756	0.4255
	SVMimp	1.3388	1.2047	2.5167	1.3036	1.3260	4.1266	2.0344	1.8302	2.5581	0.7233	0.7223	0.6081	0.7158	0.7039	0.5832	0.6303	0.4988	0.3623
	EM	0.8304	0.8265	0.8769	0.8429	0.8417	0.8841	0.8641	0.9837	1.0059	0.4024	0.4038	0.3871	0.3954	0.3974	0.3578	0.3289	0.3399	0.2829
	MICE	0.5081	0.5114	0.6111	0.5212	0.5308	0.7149	0.6293	0.8433	0.8715	0.7409	0.7382	0.7100	0.7326	0.7261	0.6545	0.6790	0.5528	0.4867

Table E.4: Ranks of the effect of different univariate (*univa*) configurations in imputation quality (R^2 and RMSE) using imputation methods from the state of the art.

		univa																	
		RMSE									R^2								
MR	**Methods**	**MCAR1**	**MCAR2**	**MAR1**	**MAR2**	**MAR3**	**MAR4**	**MAR5**	**MNAR1**	**MNAR2**	**MCAR1**	**MCAR2**	**MAR1**	**MAR2**	**MAR3**	**MAR4**	**MAR5**	**MNAR1**	**MNAR2**
5%	Meanimp	3	1	5	4	2	7	6	8	9	3	1	5	4	2	7	6	8	9
	kNNimp1	1	2	5	3	4	7	6	8	9	1	2	5	3	4	7	6	8	9
	kNNimp3	2	1	5	3	4	7	6	8	9	2	1	5	3	4	7	6	8	9
	kNNimp5	3	1	5	2	4	7	6	8	9	3	1	5	2	4	7	6	8	9
	SVMimp	3	1	4	6	2	5	9	7	8	4	1	5	3	2	8	7	6	9
	EM	3	1	5	4	2	7	6	8	9	3	1	5	4	2	7	6	8	9
	MICE	3	1	5	2	4	7	6	8	9	3	1	5	2	4	7	6	8	9
	MEAN	2.571	**1.143**	4.857	3.429	**3.143**	6.714	6.429	**7.857**	8.857	2.714	**1.143**	5.000	3.000	**3.143**	7.143	6.143	**7.714**	9.000
10%	Meanimp	2	1	5	3	4	7	6	8	9	2	1	5	3	4	7	6	8	9
	kNNimp1	4	1	3	5	2	7	6	8	9	3	1	4	5	2	7	6	8	9
	kNNimp3	2	1	4	5	3	7	6	8	9	2	1	4	5	3	7	6	8	9
	kNNimp5	2	1	4	5	3	7	6	8	9	2	1	4	5	3	7	6	8	9
	SVMimp	5	4	3	6	7	1	9	2	8	2	1	5	3	4	6	7	8	9
	EM	3	2	5	1	4	7	6	8	9	3	2	5	1	4	7	6	8	9
	MICE	3	2	5	4	1	6	7	8	9	3	2	5	4	1	6	7	8	9
	MEAN	3.000	**1.714**	4.143	4.143	**3.429**	6.000	6.571	**7.143**	8.857	2.429	**1.286**	4.571	3.714	**3.000**	6.714	6.286	**8.000**	9.000
15%	Meanimp	2	1	5	3	4	7	6	8	9	2	1	5	3	4	7	6	8	9
	kNNimp1	1	4	5	3	2	7	6	8	9	1	4	5	2	3	7	6	8	9
	kNNimp3	3	2	5	1	4	7	6	8	9	3	2	5	1	4	7	6	8	9
	kNNimp5	3	1	5	2	4	7	6	8	9	3	1	5	2	4	7	6	8	9
	SVMimp	3	5	9	2	1	6	7	4	8	2	1	5	3	4	7	6	8	9
	EM	1	2	5	3	4	7	6	8	9	1	2	5	3	4	6	7	8	9
	MICE	1	2	5	3	4	7	6	8	9	1	2	5	3	4	7	6	8	9
	MEAN	**2.000**	2.429	5.571	**2.429**	3.286	6.857	6.143	**7.429**	8.857	**1.857**	**1.857**	5.000	**2.429**	3.857	6.857	6.143	**8.000**	9.000
20%	Meanimp	2	1	5	3	4	7	6	8	9	2	1	5	3	4	7	6	8	9
	kNNimp1	1	4	5	2	3	7	6	8	9	1	4	5	2	3	7	6	8	9
	kNNimp3	1	2	5	3	4	7	6	8	9	1	2	5	3	4	7	6	8	9
	kNNimp5	2	1	5	3	4	7	6	8	9	2	1	4	3	5	7	6	8	9
	SVMimp	2	6	7	5	3	8	1	4	9	1	2	5	3	4	7	6	8	9
	EM	1	2	5	4	3	7	6	8	9	1	2	5	4	3	6	7	8	9
	MICE	2	1	5	4	3	7	6	8	9	2	1	5	4	3	7	6	8	9
	MEAN	**1.571**	2.429	5.286	**3.429**	**3.429**	7.143	5.286	**7.429**	9.000	**1.429**	1.857	4.857	**3.143**	3.714	6.857	6.143	**8.000**	9.000
40%	Meanimp	1	2	6	3	4	7	5	8	9	1	2	5	3	4	7	6	8	9
	kNNimp1	1	2	6	4	3	7	5	8	9	1	2	5	3	4	7	6	8	9
	kNNimp3	1	2	6	3	4	7	5	8	9	1	2	6	3	4	7	5	8	9
	kNNimp5	1	2	6	3	4	7	5	8	9	1	2	6	3	4	7	5	8	9
	SVMimp	4	1	7	2	3	9	6	5	8	1	2	6	3	4	7	5	8	9
	EM	2	1	6	4	3	7	5	8	9	2	1	5	4	3	6	8	7	9
	MICE	1	2	5	3	4	7	6	8	9	1	2	5	3	4	7	6	8	9
	MEAN	**1.571**	1.714	6.000	**3.143**	3.571	7.286	5.286	**7.571**	8.857	**1.143**	1.857	5.429	**3.143**	3.857	6.857	5.857	**7.857**	9.000

Table E.5: Effect of different multivariate (*unifo*) configurations in the classification performance (F-measure) using imputation methods from the state of the art.

		unifo									
MR	**Methods**	**MCAR1**	**MCAR2**	**MCAR3**	**MAR1**	**MAR2**	**MAR3**	**MNAR1**	**MNAR2**	**MNAR3**	**MNAR4**
5%	Meanimp	0.7544	0.7563	0.7578	0.7675	0.7575	0.7610	0.7558	0.7479	0.7759	0.7545
	kNNimp1	0.7582	0.7592	0.7592	0.7630	0.7575	0.7554	0.7592	0.7478	0.7535	0.7597
	kNNimp3	0.7609	0.7607	0.7628	0.7672	0.7546	0.7566	0.7614	0.7478	0.7765	0.7588
	kNNimp5	0.7606	0.7598	0.7647	0.7674	0.7561	0.7574	0.7629	0.7478	0.7763	0.7596
	SVMimp	0.7645	0.7652	0.7683	0.7695	0.7642	0.7717	0.7698	0.7685	0.7826	0.7634
	EM	0.7491	0.7472	0.7461	0.7627	0.7509	0.7538	0.7486	0.7469	0.7619	0.7492
	MICE	0.7591	0.7617	0.7611	0.7689	0.7555	0.7600	0.7652	0.7458	0.7751	0.7582
10%	Meanimp	0.7387	0.7404	0.7466	0.7676	0.7347	0.7501	0.7366	0.7245	0.7682	0.7409
	kNNimp1	0.7438	0.7498	0.7484	0.7626	0.7538	0.7379	0.7385	0.7248	0.7614	0.7487
	kNNimp3	0.7486	0.7539	0.7518	0.7685	0.7459	0.7393	0.7478	0.7248	0.7563	0.751
	kNNimp5	0.7507	0.7570	0.7521	0.7668	0.7447	0.7401	0.7472	0.7248	0.7602	0.7532
	SVMimp	0.7602	0.7654	0.7599	0.7650	0.7659	0.7762	0.7667	0.7625	0.7819	0.7637
	EM	0.7240	0.7270	0.7249	0.7500	0.7316	0.7318	0.7231	0.7164	0.7501	0.73
	MICE	0.7501	0.7528	0.7536	0.7683	0.7557	0.7438	0.7492	0.7179	0.7695	0.749
15%	Meanimp	0.7234	0.7355	0.7297	0.7514	0.7351	0.7400	0.7279	0.6976	0.7451	0.7385
	kNNimp1	0.7321	0.7467	0.7313	0.7651	0.7393	0.7346	0.7265	0.6986	0.7205	0.7417
	kNNimp3	0.7374	0.7564	0.7410	0.7531	0.7446	0.7301	0.7341	0.6986	0.7324	0.7443
	kNNimp5	0.7381	0.7560	0.7431	0.7509	0.7440	0.7298	0.7349	0.6986	0.7374	0.7456
	SVMimp	0.7602	0.7607	0.7624	0.7780	0.7622	0.7852	0.7596	0.7589	0.7857	0.7735
	EM	0.6992	0.7156	0.7061	0.7352	0.7180	0.7245	0.7090	0.6890	0.7261	0.7247
	MICE	0.7407	0.7550	0.7482	0.7522	0.7494	0.7357	0.7419	0.6918	0.7609	0.7512
20%	Meanimp	0.7165	0.7087	0.7202	0.7335	0.7428	0.7318	0.7080	0.6778	0.7317	0.7299
	kNNimp1	0.7200	0.7143	0.7201	0.7408	0.7109	0.7337	0.7072	0.6778	0.7062	0.729
	kNNimp3	0.7355	0.7295	0.7346	0.7414	0.7185	0.7273	0.7132	0.6778	0.7228	0.7409
	kNNimp5	0.7394	0.7296	0.7339	0.7395	0.7168	0.7251	0.7155	0.6778	0.7296	0.7404
	SVMimp	0.7661	0.7599	0.7594	0.7557	0.7645	0.7786	0.7556	0.7594	0.7876	0.7749
	EM	0.6885	0.6763	0.6911	0.7165	0.7016	0.7034	0.6897	0.6571	0.7036	0.709
	MICE	0.7397	0.7378	0.7406	0.7425	0.7445	0.7285	0.7322	0.6652	0.7516	0.7395
40%	Meanimp	0.6693	0.6629	0.6675	0.6801	-	0.6566	0.6181	0.5761	0.6710	0.6664
	kNNimp1	0.6392	0.6276	0.6396	0.6855	-	0.6565	0.5943	0.5809	0.5867	0.6443
	kNNimp3	0.6661	0.6585	0.6633	0.6973	-	0.6602	0.6196	0.5809	0.5836	0.6608
	kNNimp5	0.6671	0.6600	0.6626	0.6988	-	0.6529	0.6273	0.5809	0.5948	0.6566
	SVMimp	0.7671	0.7705	0.7567	0.7641	-	0.7844	0.7134	0.7575	0.7356	0.7166
	EM	0.6162	0.5961	0.6073	0.6565	-	0.6291	0.6076	0.5551	0.6129	0.6433
	MICE	0.6941	0.6969	0.6903	0.7022	-	0.6571	0.6537	0.5640	0.6361	0.6617

Table E.6: Ranks of the effect of different multivariate (*unifo*) configurations in the classification performance (F-measure) using imputation methods from the state of the art.

		F-measure − *unifo*									
MR	**Methods**	**MCAR1**	**MCAR2**	**MCAR3**	**MAR1**	**MAR2**	**MAR3**	**MNAR1**	**MNAR2**	**MNAR3**	**MNAR4**
5%	Meanimp	9	6	4	2	5	3	7	10	1	8
	kNNimp1	6	3	5	1	7	8	4	10	9	2
	kNNimp3	5	6	3	2	9	8	4	10	1	7
	kNNimp5	5	6	3	2	9	8	4	10	1	7
	SVMimp	8	7	6	4	9	2	3	5	1	10
	EM	6	8	10	1	4	3	7	9	2	5
	MICE	7	4	5	2	9	6	3	10	1	8
	MEAN	6.571	5.714	**5.143**	**2.000**	7.429	5.429	4.571	9.143	2.286	6.714
10%	Meanimp	7	6	4	2	9	3	8	10	1	5
	kNNimp1	7	4	6	1	3	9	8	10	2	5
	kNNimp3	6	3	4	1	8	9	7	10	2	5
	kNNimp5	6	3	5	1	8	9	7	10	2	4
	SVMimp	9	5	10	6	4	2	3	8	1	7
	EM	8	6	7	2	4	3	9	10	1	5
	MICE	6	5	4	2	3	9	7	10	1	8
	MEAN	7.000	**4.571**	5.714	**2.143**	5.571	6.286	7.000	9.714	**1.429**	5.571
15%	Meanimp	9	5	7	1	6	3	8	10	2	4
	kNNimp1	6	2	7	1	4	5	8	10	9	3
	kNNimp3	6	1	5	2	3	9	7	10	8	4
	kNNimp5	6	1	5	2	4	9	8	10	7	3
	SVMimp	8	7	5	3	6	2	9	10	1	4
	EM	9	6	8	1	5	4	7	10	2	3
	MICE	8	2	6	3	5	9	7	10	1	4
	MEAN	7.429	**3.429**	6.143	**1.857**	4.714	5.857	7.714	10.000	4.286	**3.571**
20%	Meanimp	7	8	6	2	1	3	9	10	4	5
	kNNimp1	5	6	4	1	7	2	8	10	9	3
	kNNimp3	3	5	4	1	8	6	9	10	7	2
	kNNimp5	3	5	4	2	8	7	9	10	6	1
	SVMimp	4	6	8	9	5	2	10	7	1	3
	EM	8	9	6	1	5	4	7	10	3	2
	MICE	5	7	4	3	2	9	8	10	1	6
	MEAN	**5.000**	6.571	5.143	**2.714**	5.143	4.714	8.571	9.571	4.429	**3.143**
40%	Meanimp	3	6	4	1	-	7	8	9	2	5
	kNNimp1	5	6	4	1	-	2	7	9	8	3
	kNNimp3	2	6	3	1	-	5	7	9	8	4
	kNNimp5	2	4	3	1	-	6	7	9	8	5
	SVMimp	3	2	6	4	-	1	9	5	7	8
	EM	4	8	7	1	-	3	6	9	5	2
	MICE	3	2	4	1	-	6	7	9	8	5
	MEAN	**3.143**	4.857	4.429	**1.429**	-	4.286	7.286	8.429	6.571	**4.571**

Table E.7: Effect of different multivariate (*unifo*) configurations in imputation quality (R^2 and RMSE) using imputation methods from the state of the art.

		unifo																			
		RMSE										**R^2**									
MR	**Methods**	**MCAR1**	**MCAR2**	**MCAR3**	**MAR1**	**MAR2**	**MAR3**	**MNAR1**	**MNAR2**	**MNAR3**	**MNAR4**	**MCAR1**	**MCAR2**	**MCAR3**	**MAR1**	**MAR2**	**MAR3**	**MNAR1**	**MNAR2**	**MNAR3**	**MNAR4**
5%	Meanimp	0.2277	0.2296	0.2245	0.3613	0.3434	0.2770	0.2470	0.2197	0.3621	0.3098	0.9383	0.9375	0.9399	0.8736	0.8839	0.9150	0.9305	0.9417	0.8675	0.8944
	kNNimp1	0.2399	0.2366	0.2389	0.3313	0.3143	0.3160	0.2509	0.2183	0.3748	0.3258	0.9343	0.9360	0.9350	0.8947	0.9022	0.8983	0.9293	0.9423	0.8592	0.8885
	kNNimp3	0.2048	0.2017	0.2022	0.2840	0.2876	0.2749	0.2190	0.2183	0.3518	0.2782	0.9487	0.9498	0.9497	0.9200	0.9143	0.9172	0.9431	0.9423	0.8733	0.9133
	kNNimp5	0.1982	0.1943	0.1950	0.2747	0.2808	0.2650	0.2132	0.2183	0.3451	0.2686	0.9512	0.9526	0.9523	0.9248	0.9180	0.9220	0.9455	0.9423	0.8772	0.9181
	SVMimp	0.5048	0.5076	0.4990	0.5851	0.5199	0.4366	0.5351	0.3595	0.5519	0.6823	0.9271	0.9272	0.9286	0.8865	0.9049	0.8968	0.9178	0.9241	0.8621	0.8940
	EM	0.3015	0.3055	0.3005	0.4432	0.3767	0.3447	0.3128	0.2962	0.4001	0.4108	0.9018	0.8998	0.9026	0.8140	0.8604	0.8757	0.8958	0.9047	0.8379	0.8287
	MICE	0.1861	0.1912	0.1838	0.2616	0.2645	0.2584	0.2020	0.2327	0.3172	0.2571	0.9560	0.9554	0.9569	0.9323	0.9262	0.9248	0.9514	0.9365	0.8964	0.9256
10%	Meanimp	0.3156	0.3189	0.3161	0.4884	0.4679	0.3846	0.3605	0.3152	0.4891	0.4476	0.8908	0.8890	0.8905	0.7806	0.8099	0.8488	0.8669	0.8910	0.7782	0.7952
	kNNimp1	0.3457	0.3387	0.3453	0.4891	0.4196	0.4585	0.3825	0.3133	0.5268	0.4653	0.8770	0.8812	0.8773	0.7940	0.8426	0.8132	0.8540	0.8921	0.7476	0.7922
	kNNimp3	0.2917	0.2877	0.2931	0.3959	0.3899	0.3897	0.3302	0.3133	0.4949	0.3975	0.9064	0.9088	0.9058	0.8504	0.8620	0.8498	0.8859	0.8921	0.7741	0.8370
	kNNimp5	0.2817	0.2777	0.2830	0.3801	0.3852	0.3727	0.3202	0.3133	0.4878	0.3844	0.9117	0.9140	0.9111	0.8609	0.8655	0.8597	0.8920	0.8921	0.7792	0.8458
	SVMimp	0.6961	0.7272	0.6743	0.8831	0.6727	0.6409	0.7088	0.5061	0.8465	0.9516	0.8869	0.8859	0.8868	0.8180	0.8535	0.8305	0.8575	0.8699	0.7682	0.8195
	EM	0.4196	0.4268	0.4200	0.5924	0.4952	0.4787	0.4455	0.4204	0.5366	0.5817	0.8225	0.8173	0.8221	0.6848	0.7836	0.7766	0.8051	0.8216	0.7293	0.6809
	MICE	0.2597	0.2670	0.2634	0.3651	0.3668	0.3610	0.2990	0.3293	0.4469	0.3692	0.9238	0.9214	0.9229	0.8728	0.8779	0.8655	0.9067	0.8824	0.8164	0.8579
15%	Meanimp	0.3894	0.3982	0.3876	0.6057	0.5720	0.4759	0.4524	0.3876	0.5849	0.5544	0.8393	0.8329	0.8406	0.6788	0.7510	0.7776	0.8024	0.8406	0.7032	0.7004
	kNNimp1	0.4354	0.4254	0.4336	0.5797	0.5207	0.5477	0.4755	0.3849	0.6534	0.5735	0.8146	0.8224	0.8153	0.7069	0.7873	0.7476	0.7852	0.8423	0.6442	0.7029
	kNNimp3	0.3701	0.3610	0.3679	0.5152	0.4896	0.4826	0.4177	0.3849	0.6162	0.4936	0.8569	0.8630	0.8581	0.7560	0.8077	0.7822	0.8272	0.8423	0.6781	0.7599
	kNNimp5	0.3574	0.3481	0.3540	0.5020	0.4846	0.4651	0.4067	0.3849	0.6088	0.4778	0.8650	0.8711	0.8671	0.7684	0.8115	0.7925	0.8355	0.8423	0.6840	0.7721
	SVMimp	0.8461	0.8920	0.8810	1.2580	0.9334	1.0113	0.9172	0.6201	1.0979	1.1520	0.8390	0.8344	0.8401	0.7052	0.8020	0.7496	0.7945	0.8203	0.7037	0.7501
	EM	0.5180	0.5279	0.5137	0.7217	0.5862	0.5846	0.5487	0.5153	0.6316	0.7145	0.7397	0.7312	0.7433	0.5562	0.7251	0.6802	0.7185	0.7417	0.6451	0.5466
	MICE	0.3273	0.3347	0.3253	0.4861	0.4679	0.4443	0.3804	0.4041	0.5420	0.4571	0.8865	0.8839	0.8873	0.7919	0.8266	0.8043	0.8592	0.8283	0.7421	0.7931
20%	Meanimp	0.4482	0.4756	0.4481	0.7051	0.6662	0.5378	0.5422	0.4457	0.6700	0.6581	0.7904	0.7659	0.7905	0.5858	0.6989	0.7231	0.7305	0.7926	0.6319	0.5933
	kNNimp1	0.5141	0.5274	0.5107	0.7001	0.6146	0.6208	0.5751	0.4425	0.7541	0.6776	0.7497	0.7401	0.7524	0.6048	0.7342	0.6807	0.7013	0.7949	0.5545	0.6055
	kNNimp3	0.4388	0.4473	0.4364	0.6037	0.5874	0.5414	0.5101	0.4425	0.7185	0.5837	0.8039	0.7969	0.8059	0.6778	0.7539	0.7269	0.7546	0.7949	0.5881	0.6730
	kNNimp5	0.4235	0.4310	0.4209	0.5878	0.5819	0.5260	0.4975	0.4425	0.7114	0.5664	0.8152	0.8089	0.8174	0.6938	0.7588	0.7374	0.7657	0.7949	0.5948	0.6871
	SVMimp	0.9825	1.0023	0.9782	2.2203	0.9844	4.5539	1.0995	0.6999	1.2071	1.3422	0.7949	0.7745	0.7950	0.6269	0.7447	0.6984	0.7281	0.7713	0.6436	0.6615
	EM	0.5936	0.6305	0.5943	0.8251	0.6595	0.6631	0.6390	0.5932	0.7109	0.8217	0.6671	0.6314	0.6664	0.4484	0.6832	0.6020	0.6335	0.6671	0.5729	0.4258
	MICE	0.3827	0.4035	0.3810	0.5565	0.5544	0.5107	0.4599	0.4647	0.6320	0.5474	0.8485	0.8325	0.8498	0.7292	0.7826	0.7465	0.8050	0.7764	0.6715	0.7110
40%	Meanimp	0.6351	0.6585	0.6328	1.0467	-	0.7585	0.9192	0.6355	0.9590	1.0746	0.5906	0.5635	0.5935	0.2681	-	0.5002	0.3842	0.5900	0.4150	0.5808
	kNNimp1	0.7799	0.8160	0.7805	1.0868	-	0.9245	0.9684	0.6302	1.0844	1.0417	0.4873	0.4593	0.4892	0.2719	-	0.4342	0.3229	0.5941	0.2881	0.5636
	kNNimp3	0.6709	0.7033	0.6688	0.9956	-	0.8178	0.8893	0.6302	1.0379	1.0103	0.5713	0.5418	0.5743	0.3241	-	0.4899	0.3977	0.5941	0.3297	0.5871
	kNNimp5	0.6501	0.6811	0.6484	0.9704	-	0.7782	0.8772	0.6302	1.0274	1.0086	0.5909	0.5625	0.5932	0.3412	-	0.5088	0.4130	0.5941	0.3394	0.5895
	SVMimp	1.3565	1.4252	1.3720	4.2964	-	9.2494	1.6060	0.9620	1.4485	1.5480	0.6202	0.5979	0.6228	0.3122	-	0.4898	0.4010	0.5770	0.4085	0.5865
	EM	0.8326	0.8668	0.8326	1.1386	-	0.9146	0.9211	0.8327	0.9496	1.0094	0.4029	0.3691	0.4039	0.1449	-	0.3328	0.3456	0.4014	0.3771	0.5491
	MICE	0.5627	0.5898	0.5628	0.9647	-	0.7871	0.8265	0.6666	0.9775	0.9780	0.6837	0.6575	0.6848	0.3891	-	0.5103	0.4832	0.5562	0.3967	0.6179

Table E.8: Ranks of the effect of different multivariate (*unifo*) configurations in imputation quality (R^2 and RMSE) using imputation methods from the state of the art.

		unifo																			
		RMSE										**R^2**									
MR	**Methods**	**MCAR1**	**MCAR2**	**MCAR3**	**MAR1**	**MAR2**	**MAR3**	**MNAR1**	**MNAR2**	**MNAR3**	**MNAR4**	**MCAR1**	**MCAR2**	**MCAR3**	**MAR1**	**MAR2**	**MAR3**	**MNAR1**	**MNAR2**	**MNAR3**	**MNAR4**
	Meanimp	3	4	2	9	8	6	5	1	10	7	3	4	2	9	8	6	5	1	10	7
	kNNimp1	4	2	3	9	6	7	5	1	10	8	4	2	3	8	6	7	5	1	10	9
	kNNimp3	3	1	2	8	9	6	5	4	10	7	3	1	2	6	8	7	4	5	10	9
	kNNimp5	3	1	2	8	9	6	4	5	10	7	3	1	2	6	9	7	4	5	10	8
5%	SVMimp	4	5	3	9	6	2	7	1	8	10	3	2	1	9	6	7	5	4	10	8
	EM	3	4	2	10	7	6	5	1	8	9	3	4	2	10	7	6	5	1	8	9
	MICE	2	3	1	8	9	7	4	5	10	6	2	3	1	6	7	9	4	5	10	8
	MEAN	3.143	2.857	**2.143**	8.714	7.714	5.714	**5.000**	**2.571**	9.429	7.714	3.000	2.429	**1.857**	7.714	7.286	**7.000**	4.571	**3.143**	9.714	8.286
	Meanimp	2	4	3	9	8	6	5	1	10	7	2	4	3	9	7	6	5	1	10	8
	kNNimp1	4	2	3	9	6	7	5	1	10	8	4	2	3	8	6	7	5	1	10	9
	kNNimp3	2	1	3	8	7	6	5	4	10	9	2	1	3	7	6	8	5	4	10	9
	kNNimp5	2	1	3	7	9	6	5	4	10	8	2	1	3	7	6	8	5	4	10	9
10%	SVMimp	5	7	4	9	3	2	6	1	8	10	1	3	2	9	6	7	5	4	10	8
	EM	1	4	2	10	7	6	5	3	8	9	1	4	2	9	6	7	5	3	8	10
	MICE	1	3	2	7	8	6	4	5	10	9	1	3	2	7	6	8	4	5	10	9
	MEAN	**2.429**	3.143	2.857	8.429	6.857	**5.571**	5.000	**2.714**	9.429	8.571	**1.857**	2.571	2.571	8.000	**6.143**	7.286	4.857	**3.143**	9.714	8.857
	Meanimp	3	4	2	10	8	6	5	1	9	7	3	4	1	10	7	6	5	2	8	9
	kNNimp1	4	2	3	9	6	7	5	1	10	8	4	2	3	8	5	7	6	1	10	9
	kNNimp3	3	1	2	9	7	6	5	4	10	8	3	1	2	9	6	7	5	4	10	8
	kNNimp5	3	1	2	9	8	6	5	4	10	7	3	1	2	9	6	7	5	4	10	8
15%	SVMimp	2	4	3	10	6	7	5	1	8	9	2	3	1	9	5	8	6	4	10	7
	EM	3	4	1	10	7	6	5	2	8	9	3	4	1	9	5	7	6	2	8	10
	MICE	2	3	1	9	8	6	4	5	10	7	2	3	1	9	6	7	4	5	10	8
	MEAN	2.857	2.714	**2.000**	9.429	7.143	**6.286**	4.857	**2.571**	9.286	7.857	2.857	2.571	**1.571**	9.000	5.714	7.000	**5.286**	**3.143**	9.429	8.429
	Meanimp	3	4	2	10	8	5	6	1	9	7	3	4	2	10	7	6	5	1	8	9
	kNNimp1	3	4	2	9	6	7	5	1	10	8	3	4	2	9	5	7	6	1	10	8
	kNNimp3	2	4	1	9	8	6	5	3	10	7	2	3	1	8	6	7	5	4	10	9
	kNNimp5	2	3	1	9	8	6	5	4	10	7	2	3	1	8	6	7	5	4	10	9
20%	SVMimp	3	5	2	9	4	10	6	1	7	8	2	3	1	10	5	7	6	4	9	8
	EM	2	4	3	10	6	7	5	1	8	9	3	6	4	9	1	7	5	2	8	10
	MICE	2	3	1	9	8	6	4	5	10	7	2	3	1	8	5	7	4	6	10	9
	MEAN	2.429	3.857	**1.714**	9.286	6.857	**6.714**	5.143	**2.286**	9.143	7.571	2.429	3.714	**1.714**	8.857	**5.000**	6.857	5.143	**3.143**	9.286	8.857
	Meanimp	2	4	1	8	-	5	6	3	7	9	2	5	1	9	-	6	8	3	7	4
	kNNimp1	2	4	3	9	-	5	6	1	8	7	4	5	3	9	-	6	7	1	8	2
	kNNimp3	3	4	2	7	-	5	6	1	9	8	4	5	3	9	-	6	7	1	8	2
	kNNimp5	3	4	2	7	-	5	6	1	9	8	3	5	2	8	-	6	7	1	9	4
40%	SVMimp	2	4	3	8	-	9	7	1	5	6	2	3	1	9	-	6	8	5	7	4
	EM	1	4	2	9	-	5	6	3	7	8	3	6	2	9	-	8	7	4	5	1
	MICE	1	3	2	7	-	5	6	4	8	9	2	3	1	9	-	6	7	5	8	4
	MEAN	**2.000**	3.857	2.143	7.857	-	**5.571**	6.143	**2.000**	7.571	7.857	2.857	4.571	**1.857**	8.857	-	**6.286**	7.286	**2.857**	7.429	3.000