Do we really need imputation in AutoML predictive modeling?

GEORGE PATERAKIS, Computer Science Department, University of Crete, Greece
STEFANOS FAFALIOS, Sphynx Technology Solutions, Greece
PAULOS CHARONYKTAKIS, JADBio Gnosis DA S.A., Greece
VASSILIS CHRISTOPHIDES, ENSEA, ETIS, France
IOANNIS TSAMARDINOS, Computer Science Department, University of Crete, Greece

Numerous real-world data contain missing values, while in contrast, most Machine Learning (ML) algorithms assume complete datasets. For this reason, several imputation algorithms have been proposed to predict and fill in the missing values. Given the advances in predictive modeling algorithms tuned in an AutoML setting, a question that naturally arises is to what extent sophisticated imputation algorithms (e.g., Neural Network based) are really needed, or we can obtain a descent performance using simple methods like Mean/Mode (MM). In this paper, we experimentally compare 6 state-of-the-art representatives of different imputation algorithmic families from an AutoML predictive modeling perspective, including a feature selection step and combined algorithm and hyper-parameter selection. Experiments ran on 25 real-world incomplete datasets with missing values and 10 complete datasets in which synthetic missing values are introduced according to different missingness mechanisms, at varying missing frequencies. The main conclusion drawn from our experiments is that the best method on average is the Denoise AutoEncoder (DAE) on real-world datasets and the MissForest (MF) in simulated datasets, followed closely by Mean/Mode. In addition, binary indicator (BI) variables encoding missingness patterns actually improve predictive performance, on average. Last but not least, although there are cases where Neural-Network-based imputation significantly improves predictive performance, this comes at a great computational cost and requires measuring all feature values to impute new samples.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**.

Additional Key Words and Phrases: missing values, imputation, automl, machine learning, optimization

ACM Reference Format:
George Paterakis, Stefanos Fafalios, Paulos Charonyktakis, Vassilis Christophides, and Ioannis Tsamardinos. 2018. Do we really need imputation in AutoML predictive modeling?. In . ACM, New York, NY, USA, 41 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Real-world data often contain missing values, stemming from faulty sensors, non-responders in questionnaires, incomplete data entry, or other reasons. For example, in the openml portal, as of March 2022, 364 out of the 3487 active datasets contain missing values. Unfortunately, most ML algorithms demand complete datasets to operate on[1]. To address this problem, a plethora of imputation algorithms, ranging from simple to very advanced, have been developed to predict the missing values and allow the remaining algorithms in the analysis pipeline to complete.

[1]Although some ML algorithms such as k-NN and Naive Bayes are robust to missing values, their implementations in popular platforms like sklearn does not currently support the presence of missing values.

The problem of imputation is under study for decades [22, 39, 52]. Initially, it was studied in the context of estimating the coefficients of linear models, call it *estimation perspective*. In contrast, we study imputation from a *predictive modeling perspective* where the goal is to create an accurate model to predict a specific outcome of interest (target variable) in new samples. There are important differences in approaching the subject, under these two perspectives. Under the estimation perspective, (a) some methods [66] would impute the missing values in the training data, but would not create an *imputation model* that is able to impute test data. Hence, these methods cannot be applied to predictive modeling. In addition, (b) standard guidelines [56] suggest using the outcome in imputing feature values, e.g., to differentiate imputation values in cases vs. controls. This technique is not applicable in predictive modeling where the outcome is unknown in test samples. Finally, (c) a useful metric of imputation efficacy under the estimation perspective is the *imputation accuracy* [23, 28] i.e., the accuracy of predicting the missing values. Imputation accuracy is important for estimation purposes but may not be indicative of the impact of imputation on predictive performance.

Under the predictive modeling perspective, several interesting questions arise:

- Are advanced predictive modeling algorithms in need of imputation beyond the simple MM technique? A non-linear algorithm could potentially learn a rule of the sort "if a feature value equals its mean (i.e., it is missing), then do not use it but instead rely on other observed features values for prediction". Hence, it is questionable whether imputation would provide an advantage to such an algorithm.
- Is the need for sophisticated imputation further reduced in Automated Machine Learning context (AutoML) whereby the most appropriate combination of algorithm and hyper-parameter values (CASH optimization) [57] is taking place?
- Do Binary Indicator (**BI**) variables (1 if the value of a feature is missing and 0, otherwise) encoding the missingness patterns provide additional information to a classifier to learn a predictive model?
- How does the feature selection step interact with imputation? Feature selection aims to reduce the number of features that enter the model without sacrificing predictive performance and leads to more interpretable models by providing insights regarding the underlying data generation. It remains open how the benefits of feature selection are impacted when we impute the missing values.
- What is the trade-off between the computational overhead of imputation and the improvement in predictive performance? Imputation algorithms impute all the missing values, independently of whether or not they contribute to the predictions of the model. In other words, imputation is unsupervised and not guided by the outcome to predict. Hence, they potentially perform a significant amount of unnecessary computations.
- If imputation algorithms indeed improve performance, are there any characteristics of the datasets (called meta-features) that allow us to predict the value of imputation prior to their analysis and decide whether imputation is worth the computational overhead?

To the best of our knowledge, this is the first empirical study that answers all the above research questions via an experimental evaluation over 25 real-world datasets, as well as 10 complete datasets in which synthetic missing values are introduced according to different missingness mechanisms, at varying missing frequencies. The Mean/Mode (MM) imputation is used as a baseline and is compared against state-of-the-art representatives of different imputation algorithmic families, namely *Discriminative*, such as Miss-Forest [55] and *Generative*, such as SoftImpute [36] and PPCA [59] exploiting matrix-factorization, or GAIN, [72] and DAE [16] based on Neural Networks. The imputation algorithms are integrated into the JADBio AutoML platform [62] which performs combined algorithm and hyper-parameter selection (CASH) and it includes a feature selection step.

In summary, the results show that the single best-performing algorithm is DAE and MissForest for the real and the simulated datasets, respectively. For 5 out of the 6 imputation algorithms studied, the inclusion of BI variables is beneficial, on average. MM, when BI variables are included and Combined Algorithm and Hyper-parameter Selection (CASH) is taking place, is a close competitor and places as the second-best algorithm. Advanced imputation methods do offer a significant advantage but only in a few datasets. In contrast, they require the measurements of all feature values to impute new samples, which in some way invalidates the feature selection step and leads to models of high dimensionality. In addition, they require orders of magnitude more computational time. Meta-level analysis has indicated that only one feature is correlated with the relative performance of the algorithms; unfortunately, the correlation is not statistically significant when corrected for multiple testing. More datasets and new meta-features are needed to extract patterns of when sophisticated imputation should be used over the simple MM.

Overall, in an AutoML setting where optimization is taking place and BI variables are included, MM is a reasonable option; other algorithms should be used only if feature selection is not required and computational time is of little importance relative to improving predictive performance.

The paper is organized as follows. Section 2 introduces missing data mechanisms and a taxonomy of imputation families. In Section 3, we present the experimental environment, the selected datasets for evaluation, and the metrics and hyper-parameters tuned. Section 4 describes the missing data generation procedure. The experimental results for real-world data with missing data and simulated missing data are presented in Sections 5 and 6, respectively. In Section 7, we discuss the results of the meta-level analysis on real-world datasets. Related work is discussed in Section 8, followed by the contributions and lessons learned in Section 9. Finally, Section 10 presents the conclusions and limitations of the study. The detailed information about the datasets, missing value simulation setup, and experimental results are provided in Appendices A, B, and C, respectively.

2 BACKGROUND AND CONTEXT

2.1 Missingness Mechanisms

The concept of a *missing mechanism* [52] formalizes the generation process of missing data. In this respect, the Binary Indicators (BI) are modeled as random variables and assigned a distribution. There are three types of underlying mechanisms that generate missing data, namely, *missing completely at random* (MCAR), *missing at random* (MAR), and *missing not at random* (MNAR). For formal definitions of these mechanisms, readers are referred to [33]. Intuitively, MCAR implies that the probability of a value missing is independent of the actual value, the other observed quantities, and any latent variables. MAR implies that the missingness only depends on the observed data (so it can be predicted). MNAR refers to the case that the missing values are related to both the observed and unobserved variables, including the missing value itself. When missingness is MNAR, it is in principle and in general not possible to impute the missing values in a way that follows the unknown underlying data distribution [2].

2.2 An Imputation Family Taxonomy

There are numerous imputation algorithms and approaches in the literature and we do not attempt a full review. Imputation approaches can be partitioned into various distinct families/groups of methods. A taxonomy is attempted in Figure 1. First, imputed values can be decided based on only the feature with the missing value (*Univariate* imputation), or several features (*multivariate* imputation). The former methods include *Mean/Median/Max* imputation for continuous

[2]Recent work [38] shows that when the causal graph of the distribution is known there are cases where MNAR data can be imputed.

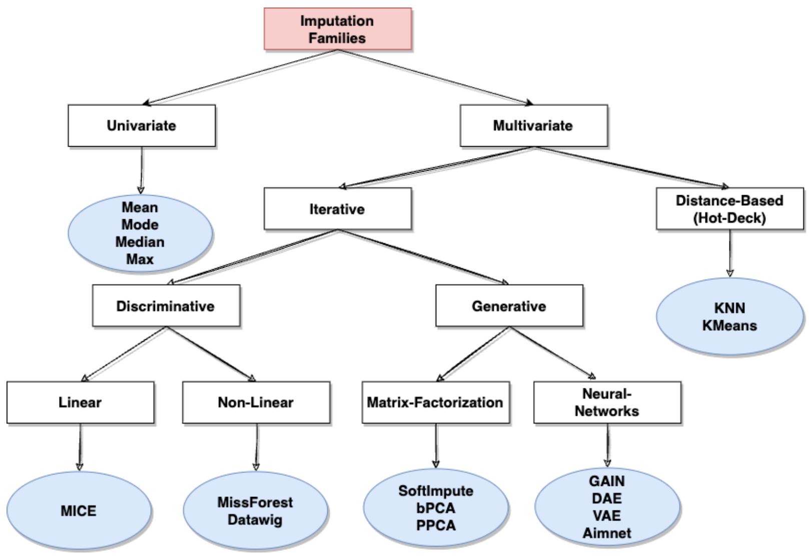


Fig. 1. The taxonomy of imputation families. Rectangular nodes represent algorithmic families, while oval nodes represent algorithms of that family.

data and *Mode* imputation for categorical data. *Multivariate* methods can be partitioned into *Iterative* and *Distance-Based*, also known as Hot-Deck methods [4]. Distance-based methods employ a distance or a similarity metric for samples to find neighbors or cluster them. A commonly used algorithm in this category is the *K-nearest neighbors (KNN)* imputation [60], which imputes values based on the neighbors of the sample with missing values. *k-means* based methods [30] cluster the samples before imputation.

Iterative methods, start with a simple initial guess (e.g., using MM imputation) and, in each iteration, try to improve the imputed values. We further split iterative methods into *Discriminative* and *Generative*. *Discriminative* methods, build a predictive model per feature with missing values, given the other features in the dataset. This model is used to predict the missing values of the corresponding feature, in each iteration. The Discriminative family can either utilize a (generalized) linear model or a non-linear model. Linear discriminative methods include *Multivariate Imputation by Chained Equations (MICE)[65]*. Non-linear discriminative methods include the *MissForest* algorithm [55] employing Random Forests, and *Datawig* [7] that can impute continuous, categorical and text data by employing different loss functions according to the missing features' datatype.

Generative methods try to model the joint distribution of the data and use the generative model to impute values. They can be split into two categories, methods that employ matrix factorization and methods that use neural networks. The matrix-Factorization family includes low-rank matrix decomposition methods: first missing values are imputed with an initial guess, and the matrix is decomposed (factorized) and used to predict the missing values. Imputation is improved in each cycle via expectation-maximization steps. Examples of this family include the probabilistic principal component analysis (*PPCA*) [59], SVDImpute [60], bPCA [8] and SoftImpute [36]. Such algorithms scale better w.r.t. to the number of features than MICE or MissForest that train a different model for each feature with missing values, in each iteration. Recently, neural networks have also been tried as generative models. These algorithms are essentially non-linear alternatives to matrix factorization. These methods start with an initial guess and then train a neural network that learns

Table 1. This table presents a comparison of the imputation methods and their characteristics. Abbreviations: n is the number of samples, m is the number of features, $k$is the number of iterations, $\#trees$ stands for the number of trees in the forest(hp), $\#comp$ stands for the number of principal components employed in the matrix factorization, $\#sing$ for the number of singular values of the SVD.

Algorithm	Model Family	Base Model	Learning Procedure	Categorical Handling	Approx. Complexity
MM	Univariate	-	No Iterative	Native	O($n \cdot m$)
SOFT	Generative	SVD	Iterative	One-hot-encoding	O($k \cdot n \cdot m \cdot \#sing$)
PPCA	Generative	PCA	Iterative	One-hot-encoding	O($k \cdot n \cdot m \cdot \#comp$)
MF	Discriminative	RF	Iterative	Native	O($k \cdot m^2 \cdot n \cdot \log(n) \cdot$ #trees)
GAIN	Generative	GAN	Iterative	Ordinal-Encoding	-
DAE	Generative	AE	Iterative	One-hot-encoding	-

the joint distribution. This family includes methods based on AutoEncoders (AE), such as Denoise AutoEncoder (*DAE)* [16] and Variational Autoencoder (*VAE)* [37]. Also, it includes an adaptation of generative adversarial networks named *GAIN* [72]. Finally, HoloClean, a data cleaning tool, implements an attention-based neural network for imputation, named *Aimnet* [71]. A detailed comparison of imputation methods is detailed in Section 8. In the next subsection, we will explain the rationale for our choice to include in our empirical study a subset of the aforementioned imputation methods.

2.3 Description of the selected imputation methods

In this section, we present the main characteristics of the imputation methods given in Table 1 that we included in our testbed. In the analysis of their computational complexity n denotes the number of samples, m the number of features, $\#comp$ the number of principal components, $\#sing$ the number of singular values and $\#trees$ for the number of trees.

2.3.1 Mean/Mode. Mean/Mode (MM) is the most common imputation method in AutoML tools and is included as the baseline methodology. It is an instance of the univariate imputation family. In the MM algorithm, missing values are imputed with the mean in the training data of the corresponding feature if it is continuous, and the mode (most frequent value), if it is discrete. MM is the most computationally efficient method as it needs only $O(n \cdot m)$ to impute the whole dataset.

2.3.2 MissForest. MissForest (MF) is a discriminative iterative method based on Random Forests [55]. First, the missing values are imputed by Mean/Mode. Subsequently, for each feature with missing values serving as the outcome, the algorithm trains a random forest on the rest of the features and uses it to predict the outcome's missing values. After imputing all missing values, the algorithm uses the (now) complete dataset to warm-start the new iteration until a stopping criterion is met. MF is one of the slowest methods as it requires building a forest per feature for a number of iterations. The approximate worst case is $O(k \cdot m^2 \cdot n \cdot \log(n) \cdot \#\text{trees})$. MF encounters scalability issues in datasets with more than 50 features. In addition, MF needs to store a forest for each feature, which creates model-storing issues. To avoid such issues, in our experiments we limit the maximum allowed depth of the Random Forests.

2.3.3 Probabilistic PCA. Probabilistic Principal component analysis (PPCA) is a statistical iterative method [32]. In each iteration, a principal component analysis (PCA) is performed, which is improved in the next step using maximum likelihood estimation [49] (MLE) and assuming a multivariate Gaussian distribution of the data. To impute a new sample, the optimal set of principal components found from training is used to identify the missing values that maximize the joint probability of the sample. Categorical features are one-hot-encoded before applying PPCA and then are inverse transformed after PPCA returns the imputed data. PPCA is one of the fastest methods, scaling linearly to the number of samples, features, and the number of principal components computed. The approximate complexity for PPCA is $O(k \cdot n \cdot m \cdot \#\text{comp})$.

2.3.4 SoftImpute. SoftImpute (SOFT) is a statistical iterative method [36]. It starts with the initialization of missing values with the mean. Then, it iteratively solves the optimization problem on the complete matrix using a soft-thresholded SVD and proceeds iteratively until a stopping criterion is met. Categorical features are one-hot-encoded before applying SOFT and then are inverse transformed after SOFT returns the imputed data. SOFT like PPCA is very very fast, utilizing an EM approach. The approximate time complexity is $O(k \cdot n \cdot m \cdot \#\text{sing})$.

2.3.5 Denoise Autoencoder. Denoise Autoencoder (DAE) is a deep learning algorithm based on autoencoders [16]. The Denoise Autoencoder is based on an overcomplete implementation and a dropout layer. DAE projects the input data to a higher dimensional subspace where the missing data are recovered by the decoder. The categorical data are one-hot-encoded before DAE is applied. Then the one-hot-encoded data are transformed back to the original representation. The complexity of the DAE is mostly measured by the number of epochs needed for the algorithm to impute the dataset accurately and the hidden layers' size and depth. For further information see section 3.8.

2.3.6 Generative Adversarial Imputation Nets. Generative Adversarial Imputation Nets (GAIN) is an adaptation of the GAN framework [72]. A generator is used to impute missing data based on the observed data. The discriminator tries to determine which data are observed and which are imputed. The goal of the generator is to provide an accurate imputation whereas the goal of the discriminator is to distinguish between the observed and missing data. The two neural networks are trained in an adversarial process. Categorical data are turned into ordinal features and normalized between 0 and 1. After applying the GANs we revert the categorical data to the original representation by doing the inverse procedure. The complexity of GAIN is mostly bottlenecked by the number of iterations needed to train the GANs. See 3.8 for more details.

2.3.7 Binary Indicators. Binary Indicator (BI) is not an imputation method but a feature construction method. Specifically, for each feature F_j with missing values, we construct a new feature, call it I_{jk}, which indicates whether the value at the kth sample of feature j is missing or not. The idea is to encode in I_j the missingness pattern. BIs may help the classifier and allow it to learn whether to trust value F_{jk} for prediction, or not. BI can complement any imputation method. BI's complexity is $O(n \cdot m)$, however, we should note that it increases the complexity of subsequent stages of the ML pipeline by increasing the dataset's dimensionality by a maximum factor of 2.

2.4 Rationale of the Selection of Algorithms

MM imputation is selected as a baseline and one of the most commonly used methods. **MissForest** is selected as a representative of a multivariate iterative imputation method over MICE, based on the results on imputation accuracy presented in [68]. Encoding missingness information using **BI** is also experimentally evaluated as it performed better than other methods in [35]. Distance-based methods are excluded for various reasons. First, they need to memorize the full dataset in order to produce imputation as they don't learn a model. K-means and KNN imputation were not included in our testbed as according to other empirical studies are outperformed by MissForest [24, 34, 55]. As representatives of matrix completion-based methods, we chose **PPCA** that have shown the best performance according to previous empirical studies[19, 25, 49]. **SoftImpute** was also selected based on the experimental results presented in [74]. **GAIN** and **DAE** were also included as representatives of neural network-based methods as they excel in several studies[9, 45].

The former is based on using a generative adversarial network to learn the probability distribution to impute, and the latter on autoencoder. **VAE** and **Aimnet** were not included based on the inferior or comparable results to GAIN and MM respectively [9, 31]. Finally, **Datawig** [7] is not included as according to the results reported in [7] it is outperformed by MissForest for both continuous and categorical data while it comes with a high cost to fit one neural network per feature with missing values.

2.5 How is imputation treated in AutoML platforms

AutoML platforms employ imputation methods, as well as modeling algorithms that directly treat missing values as a separate category. The current versions of JADBio (version v1.4) and AutoSklearn [12] employ MM by default, while DataRobot[3] may also include BI variables. AutoSklearn allows the user to specify additional imputation methods to optimize over, as part of the pipeline. TPOT employs median imputation for all missing features [29]. Auger.AI[4] does iterative regression or mean imputation for numerical features depending on the dataset size and creates a new category for the categorical features. BigML[5] by default doesn't impute the missing value; the missing values are handled internally by their predictive models, which are based on trees only. Autoprognosis optimizes the ML pipeline over a variety of missing data imputation algorithms. Specifically, it employs MICE, MissForest, Bootstrapped Expectation-Maximization imputation, Soft-Impute and MM [2]. DriverlessAI by H2O creates a new value to express missingness when the XGBoost, LightGBM and RuleFit algorithms are used. For generalized linear models, it performs MM imputation, while for tensorflow models missing values are treated as outliers [17]. GAMA [15] does not impute missing values by default. Autogluon [11] uses median imputation for continuous features and introduces a new "Unknown" category for categorical features.

3 EXPERIMENTAL SETUP

We now present the design choices for the experimental setup and the comparative evaluation.

3.1 Datasets

Incomplete Real Datasets. There are currently 364 datasets with missing values in the OpenML repository [67], we restricted our selection to binary classification datasets. We selected 25 binary classification datasets in an effort to cover a range of various dataset characteristics. The datasets contain both continuous and discrete features. The number of features ranges from 7 to 69, the sample size ranges from 155 to 31406, the prevalence of the minority class ranges from 0.06 to 0.48, the number of features with at least 1% missing values ranges from 1 to 32, and finally, the percentage of missing values ranges from 1.11% to 71.64%. Table 6 in appendix, presents the characteristics of the datasets, along with their OpenML id.

Complete Datasets: We selected 10 complete datasets from OpenML, where we introduce and simulate missingness. The number of features ranges from 9 to 135, the sample size ranges from 101 to 5473, and the prevalence of the minority class ranges from 10% to 49%. Table 7 in the appendix contains these values for each dataset, along with their OpenML id.

[3]https://www.datarobot.com/
[4]https://auger.ai/
[5]https://bigml.com/

3.2 Evaluation Task and Metric

We note that the evaluation concerns only *binary classification*. The metric of predictive performance is the Area Under the ROC curve (**AUC**). The datasets are split to 50% training and 50% hold-out test set used only for performance evaluation.

3.3 AutoML Environment

To experiment with different imputation algorithms when CASH optimization is taking place, we employed the JADBio (Just Add Data Bio) AutoML platform [62]. JADBio is a commercial product (a version of JADBio with basic functionality is freely available) but was offered to us for research purposes. JADBio includes feature selection as part of the ML pipeline and thus, it can be used to study the effect of feature selection on imputation.

A quick description of JADBio's architecture now follows. For each dataset to analyze, an internal knowledge base system, called Algorithm and Hyper-Parameter Space selection (AHPS) in [62], selects the feature construction, preprocessing, feature selection, and modeling algorithms to try, along with a set of values for their hyper-parameters. The AHPS also selects the configuration evaluation protocol, e.g., 10-fold cross-validation, repeated cross-validation, or hold-out to estimate the performance of each configuration and select the winning one. The knowledge in AHPS is engineered by experienced analysts but also induced by meta-level learning algorithms.

The choices of the AHPS are based on the meta-features of the dataset (e.g., sample size, number of features), as well as the user preferences. For example, an algorithm that does not scale to the number of samples in the current dataset, will not be selected by AHPS. The choice of the evaluation protocol also depends on the meta-features: for a typical-sized dataset, JADBio may run a 10-fold cross-validation, for a large balanced dataset a hold-out, while for a small sample or an imbalanced dataset, it may run a repeated cross-validation protocol.

Subsequently, JADBio executes all configurations effectively performing a grid search for CASH optimization. However, JADBio includes pruning heuristics that may drop a configuration in the early folds of cross-validation if it is not deemed promising, departing from a pure grid search strategy [63]. Once configurations execute, the final model is built on all available data using the winning configuration.

The final performance of the model producing with the winning configuration is the cross-validated AUC *adjusted for the bias incurred due to multiple tries* (called "winner's curse" in statistics). This adjustment is conceptually equivalent to adjusting p-values in multiple hypotheses testing. JADBio uses the BBC-CV algorithm for the performance estimate adjustment [63]. In [62], experiments on 360 omics datasets of small sample size show that this estimation protocol returns slightly conservative out-of-sample AUC performances of the returned model. Nevertheless, for the purposes of this paper, JADBio's performance estimation was not used; instead, the performances on the 50% held-out set are reported.

Regarding the settings of JADBio employed in this set of experiments, we note the following. One of the user preferences indirectly controls the execution time and the number of configurations to try and has the settings *Preliminary, Typical, Extensive*, with Extensive trying more configurations and performing a more thorough optimization. *All subsequent experiments were run using the Preliminary setting* to make the computational requirements manageable. The number of configurations may vary between datasets depending on their meta-features, but in our experiments, it ranges from 900 to over 1000. The training protocol of JADBio depends on the sample size, the class imbalance, and other factors. For typical-size datasets, JADBio uses a repeated 10-fold cross-validation with #repeats from 1 to 20. A heuristic procedure stops repetitions of cross-validation if no progress is detected. Overall, JADBio uses estimation protocols that execute each configuration between 10 to 200 times per dataset to choose the winning configuration and produce a model.

JADBio optimizes over the following set of algorithms. For feature selection, JADBio uses the Lasso [58] and a variant of the SES algorithm [64] with an upper bound on the number of conditional independence tests to perform. For classification, it optimizes over Ridge Logistic Regression, Decision Tree, Random Forests, and Support Vector Machines with polynomial, linear, and radial basis kernels.

To evaluate imputation algorithms, we embedded them into the JADBio configurations as the second step, after the standardization of continuous features and before feature selection, using the API provided. It is important to note that *configurations are cross-validated as an atom*, and hence, *learning to impute is based only on the training data*. This is necessary to avoid overestimating the performances of configurations and correspondingly, the imputation methods. Each imputation method returns an imputation model that is used to impute the test data before modeling is applied. It is worth noting that even if the feature selection step selects a small subset S of features when some values of S are missing in the test set, the imputation model may impute them based on other features. Hence, even if the predictive model requires just the features in S, the predictive pipeline may require more features. Specifically, all multivariate algorithms selected in the paper *require all features to impute*. Hence, the predictive pipeline always requires all features when these algorithms are employed, even with feature selection.

3.4 Imputation algorithms implementations

We used the JadBio version 1.4.0 for our experiments. MM and BI methods were already implemented by the developing team of the tool used. For PPCA and SoftImpute, we relied on third-party implementations in R from the PCA methods 1.64.0 [54] and 'softImpute' package version 1.4.1 respectively. We implemented MissForest in python 3.8.4 using the iterativeImputer and RandomForest models from sci-kit learn 1.0.1 [44]. Pytorch 1.7.1 version [43] was utilized for the implementation of GAIN and DAE. We adapted the DAE implementation found at https://github.com/Harry24k/MIDA-pytorch to closely follow the description of DAE by the original authors in [16]. We employed the GAIN from https://github.com/dhanajitb/GAIN-Pytorch.

3.5 Machine Specifications

The predictive performance experiments of the paper were conducted on a fedora-powered VM using 8-core AMD Threadripper 3970x at 3.7Ghz with 12GB RAM. The neural networks were trained using CPUs. The execution time results reported, were measured on an 8-core AMD Ryzen-3600x at 4.6Ghz with 16GB ram and Windows 11 OS.

3.6 Computational Resources Employed

During the experiments, more than 41 days of CPU time have been spent training more than 80.000 configurations to conduct the experiments mentioned in the paper.

3.7 Availability of code

The code is available on the Github repository: https://github.com/mensxmachina/Imputation_in_AutoML. The code in the repository consists of scripts for the plots, the datasets, the meta-level analysis as well as the basic implementation of each imputation algorithm.

3.8 Exploring the Hyper-parameter Space of Imputation Algorithms

In the experiments, 24 hyper-parameter (hp) value sets were tried for the imputation algorithms: MM (1 hp set), MissForest (2 hp sets), SoftImpute (3 hp sets), PPCA (3 hp sets), DAE (9 hp sets), and GAIN (6 hp sets). The values tried for each hyper-parameter are shown in Table 2. These choices were based on the algorithm's authors' defaults and suggestions. These 24 hp sets were coupled with all other choices of JADBio multiplying by 24 the number of configurations normally tried. In subsequent experiments, each of these 24 hp sets is run on the original dataset, as well as, the dataset with the inclusion of the BI features, leading to 48 different combinations. MM has no parameters and therefore doesn't need tuning. For MissForest, we train RF models with 250 trees, which offers higher imputation accuracy according to [55]. However, we restrict the maximum depth of the tree and maximum leaf nodes because the trained model had storing memory issues (see Section 2.3.2). SoftImpute and PPCA require selecting the number of principal components to use, as a hyper-parameter. The majority of papers in the literature fails to report the tuning of the aforementioned methods which led us to develop the following heuristic: we select as many components required to explain $x\%$ of the data variance. The values of x are shown in Table 2 as the values of "variance-explained". The default hyper-parameters are used for DAE with the exception of the dropout layer and the hidden layers' dimensions. The range of the dropout layer is based on [53], while the theta value is tuned within a neighborhood of the author's suggested default value. In the current implementation, we have 3 hidden layers for the encoder and the decoder. For each successive layer in the encoder, θ hidden layer nodes are added and hyperbolic tangent is used as the activation function as it produces better results for small and medium-sized datasets [16]. The model is trained using Stochastic Gradient Descent with an adaptive learning rate with a time decay factor of 0.99 and Nesterov's accelerated gradient. GAIN architecture consists of 3 hidden layers for the discriminator and the generator while using Rectified Linear Unit as the activation function. For GAIN, we tune two hyper-parameters; alpha and hint rate. These hyper-parameters are considered the most important for GAIN. Alpha balances the loss between the discriminator and the generator, while the hint rate is responsible for the training of the discriminator. Both DAE and GAIN are trained at the specific epochs as suggested by authors and use the sigmoid activation function for the output layer.

4 SIMULATING MISSING DATA

To experiment with a ranging percentage of missing values, as well as different missing mechanisms, we simulated the presence of missing values in the complete datasets presented in Appendix A.2.

4.1 Simulating Missing Completely At Random Data

Under MCAR, missing values are missing with a given probability (percentage) independently of any other factors such as the value itself or the values of other features. To simulate missing values at a realistic missingness percentage we sampled 64 real-world datasets from the OpenML repository with varying characteristics (see Section B.1). We then computed the 25%, 50%, and 75% quantiles of missingness percentages. Features with less than 1% of missing values were excluded from the calculation, as they probably point to features that missing values from typos or non-systematic sources. The quantile values turn out to be about 10%, 25%, and 50% of missingness. The quantile values are then used to vary the missingness percentages in both MCAR and MAR simulation experiments. We then introduced missing values with the given percentages at the 10 complete datasets described in Section 3.1. Even though it is trivial to introduce MCAR missing values, for consistency reasons, we employed the code available by [40], which is also used for the MAR simulations below.

Table 2. The set of values tried for each hyper-parameter tuned. For each algorithm, all combinations of values for its hyper-parameters shown were tried and combined with all other choices for feature selection and modeling by JADBio. There are 48 combinations of imputation algorithms and hyper-parameter values.

Algorithm	Hyper-parameter	Value
Mean/Mode	-	-
MissForest	n-trees	250
	maxDepth	20,30
	maxLeafNodes	30
SoftImpute	variance-explained	50%, 70%,90%
PPCA	variance-explained	50%, 70%,90%
DAE	dropout	0.25,0.4,0.5
	batch-size	64
	θ	5,7,10
	epochs	500
GAIN	alpha	0.1, 1, 10
	hint-rate	0.5, 0.9
	batch-size	64
	epochs	10.000

4.2 Simulating Missing At Random Data

Under MAR, missing values are missing with a probability (percentage) that depends (is conditional) on other observed features, i.e., $P(I_j = 1|F_{k_1}, \ldots, F_{k_m})$. To realistically simulate data under MAR, one needs to decide (a) the number of features upon which the probability depends, (b) the functional form of the conditional probability function, and (c) the set $\{F_{k_1}, \ldots, F_{k_m}\}$. To answer (a) we needed a realistic estimate of the number m of features in the conditional probability. To that end, in the corpus of the 10 real-world binary datasets of Table 7, we randomly selected one feature with missing values as the target feature and then performed predictive modeling using JADBio including feature selection[6]. These experimental results suggest that, on average, a feature with missing values is dependent on 12 features, so we set $m = 12$. Subsequently, for each F_j we randomly selected a set of m other features with uniform probability. Finally, for the functional form of P, we used a logistic regression model: $P(I_j = 1|F_{k_1} = f_1, \ldots, F_{k_m} = f_m) = \frac{1}{1+e^{\langle -w, f \rangle}}$, where w is a set of randomly chosen coefficients from a normal Gaussian distribution, f is the vector of values of the features F_{k_l}, and $\langle \cdot, \cdot \rangle$ denotes the inner product. For the simulation, the software [40] was also used. The software allows the simulation of MAR missing data, as described above, with prespecified missingess percentages. The same percentages as in the MCAR case were used.

5 COMPARATIVE EVALUATION ON REAL-WORLD DATASETS WITH MISSING VALUES

The 25 real-world datasets with missing values were analyzed with JADBio, optimizing over configurations that include the imputation algorithms selected and their hyper-parameter values.

5.1 Binary Indicators improve the predictive performance

First, we partition results achieved when optimizing over any single imputation algorithm. Specifically, for each imputation algorithm, on a given dataset, the best AUC was selected over all configurations that include the specific

[6]Table 9 contains the dataset name, the missing feature, and the number of features selected for the missing feature as the outcome.

algorithm. *We will refer to this best AUC simply as the AUC of a given imputation algorithm, in all subsequently reported results.* Figure 2 shows the average difference in imputation performance when Binary Indicators are used versus when excluded of the AUC. As we can see, MM and GAIN increase by 0.0074 AUC and 0.0056 AUC respectively. MF when extended with BI shows an average AUC increase of 0.0046, while PPCA shows an increase of 0.0038 AUC. The lowest average improvement is achieved by SOFT, which improves by 0.00022 AUC. Contrary to the above observations, DAE is the only method that doesn't benefit from the addition of BIs with a decrease of 0.0004 AUC when BI's are included.

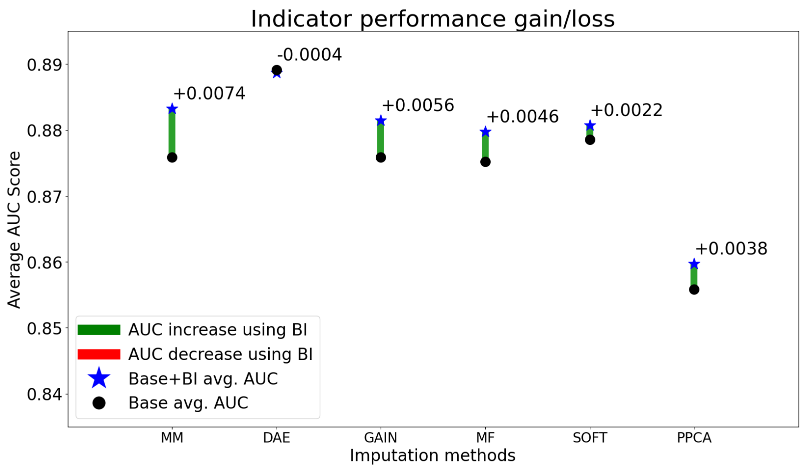


Fig. 2. AUC performance gain for each imputation method when BIs are included. BI indicators increase the predictive performance on average for all but one imputation method.

Table 3. The table shows the p-values of the matched t-test and the q-values after FDR correction (sorted). Only MF has p-value < 0.05. GAIN and PPCA have p-value < 0.1. Setting the q-value threshold to 0.25 leads to accepting the hypothesis that BI's are beneficial for 4 algorithms (MF, GAIN, PPCA, MM), expecting a 25% (1 out of 4) of these discoveries to be false on average.

Base Imp.Method	p-value	q-value
MF	0.036	0.182
GAIN	0.091	0.182
PPCA	0.064	0.182
MM	0.148	0.222
SOFT	0.31	0.375
DAE	0.552	0.552

To determine the statistical significance of the results, we performed a paired matched t-test for each algorithm with the null hypothesis H0 being that the BI+base has worse performance than the base method. The resulting p-values were converted to q-values with the Benjamini/Hochberg [5] method, to control for multiple testing. Table 3 shows the results. Using a q-value threshold of 0.25 there are 4 statistically significant results, resulting in accepting the alternative hypotheses that MF, GAIN, PPCA, and MM improve their performance when BIs are present. At the level of $q = 0.25 = \frac{1}{4}$ this implies that, in the worse case, we expect one of these four discoveries to be false. While the inclusion of BIs may, in the worse case, double the dimensionality of the dataset, based on the above results, we would recommend their inclusion when the above imputation algorithms are employed.

5.2 BI+DAE is the best imputation method in real-world data

Figure 3 shows the average ranking achieved by each algorithm using the Autorank tool [20] (lower ranking is better). To avoid clutter, and based on the results of Section 5.1, we only show results when BIs are included. The horizontal black bars in the graph connect tools with non-statistically different ranks, according to a non-parametric Friedman test and post-hoc Nemenyi test.

Results show that BI+DAE is the highest ranking algorithm with an average rank of 2.84, followed by BI+MM with a 2.94 average ranking, BI+MF with 3.5, and BI+GAIN with 3.56, although their rank difference is not statistically significant at the 0.05 level. The two lowest-ranked methods are BI+SOFT and BI+PPCA, with 3.74 and 4.42 average rankings respectively. BI+DAE's rank is statistically significantly lower compared to BI+PPCA. Detailed results are shown in Table 10 in the appendix.

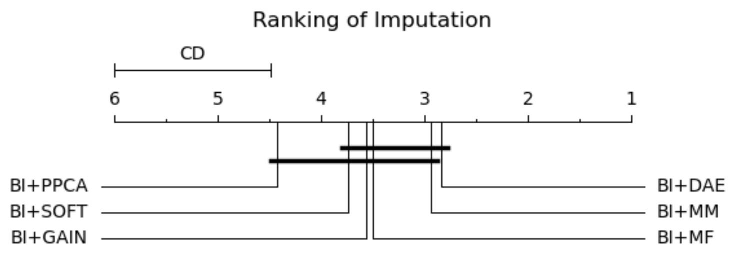


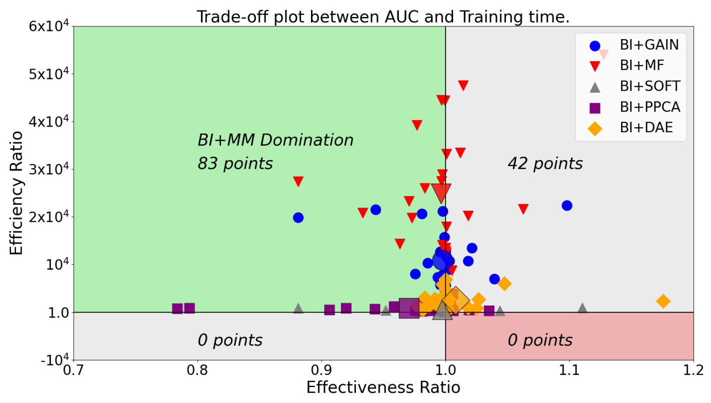
Fig. 3. The figure illustrates the average rank of each imputation method when binary indicators are used. BI+DAE has the lowest average ranking. Rank differences are not statistically significant except for the average rank between BI+DAE and BI+PPCA.

5.3 BI+MM is the best method when considering the efficiency-effectiveness trade-off.

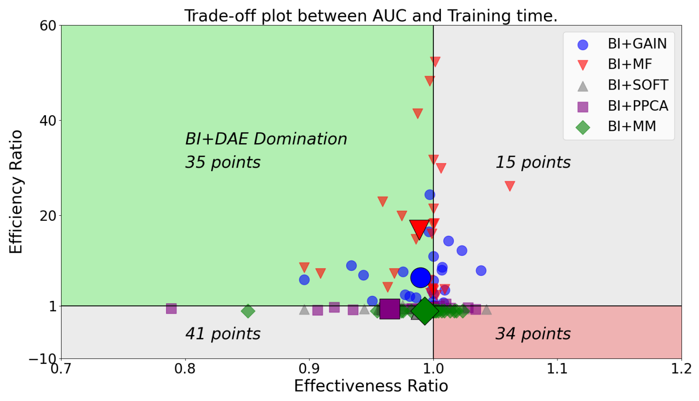
We now study the performance effectiveness vs the computational efficiency trade-off of the algorithms. In Figure 4a we use MM as the baseline. A point (execution run) corresponds to AutoML predictive modeling on a dataset with a given imputation algorithm. This results in $5 \times 25 = 125$ points. The x-axis shows the effectiveness ratio defined as the ratio of the AUC corresponding to the point divided by the corresponding performance of BI+MM. Similarly, the y-axis shows the efficiency ratio defined as the training time of the point divided by the corresponding time of BI+MM. Hence, points in the first/fourth quadrant (top-left/bottom-right) correspond to runs where BI+MM dominates/is-dominated by other algorithms on the same datasets in both time and AUC. Notice that the scale of the y-axis is logarithmic. Larger points correspond to the mean value of an imputation method over all datasets.

In total BI+MM is inferior in terms of predictive performance in 42 cases (16 out of the 25 datasets) and, unsurprisingly, never gets dominated in terms of AUC performance and efficiency at the same time. The computational time of the other algorithms is orders of magnitude slower than BI+MM. However, in 83 out of 125 points, BI+MM is both more efficient and effective than the compared method. Only BI+DAE scores on average higher than the AUC score. All the other imputation methods are on average slower to train and worse in terms of predictive performance.

Figure 4b shows the same exact results with BI+DAE as the baseline. In contrast to BI+MM above, BI+DAE dominates the other imputation methods on predictive performance and training time, in only 35 out of 125 combinations. In 41 out of 125 cases, it provides better predictive performance but at a higher computation cost. In 15 points, BI+DAE is faster but has lower predictive performance than the compared imputation method. Finally, 34 times it is dominated in both metrics. In conclusion, when *if a single imputation algorithm is to be used, BI+MM arguably provides the best trade-off between computational time and predictive performance.*



(a) The figure illustrates the relative efficiency for each imputation method against BI+MM versus the relative effectiveness in terms of AUC. Larger points indicate the mean values for a given algorithm. 83 out of 125 points fall in the top-left quadrant where BI+MM dominates other algorithms in both efficiency and effectiveness. Only BI+DAE outperforms BI+MM in effectiveness by 0.8% on average, however, it is 2500 times slower than BI+MM.



(b) The same results as above when using BI+DAE as the reference algorithm. 34 out of 125 points fall in the bottom-right quadrant where BI+DAE is dominated in both metrics.

Fig. 4. Figure (a) denotes the efficiency-effectiveness trade-off by using BI+MM as reference algorithm. Figure (b) illustrates the efficiency-effectiveness trade-off when BI+DAE is used as reference.

5.4 Best Imputation subset for maximizing AUC performance is $\{BI + MM, BI + DAE\}$

In this section, we examine the results from a different perspective, trying to answer the question: what is the minimal-size subset of algorithms to try to achieve close-to-maximum AUC performance? To answer this question, we have implemented a simple greedy algorithm, where we assume the analyst starts with the subset {BI+MM} as an efficient baseline and adds algorithms to consider. In each iteration, the algorithm that leads to the largest AUC improvement of the subset when added is selected for inclusion.

The results are shown in Figure 5 and quantitatively in Table 11. The x-axis shows the imputation algorithms, in order of addition to the subset. For each algorithm, several hyper-parameter combinations are tried and combined with all other feature selection and modeling choices by AutoML. Hence the total number of configurations tried is multiplied by this factor. At each tick, the multiplication factor for the whole set is depicted in the parenthesis next to the name of the algorithm added to the set in that step. For example, BI+MM has no hyper-parameters (1×), while BI+DAE has 9, so the multiplicative factor of the set $\{BI + MM, BI + DAE\}$ is 10×. The y-axis is the average (over all datasets) relative AUC achieved when performance is optimized over all algorithms and their hyper-parameters in the corresponding subset.

BI+MM, by itself, accounts for 98.69% of the maximum AUC. When BI+DAE is added to the mix, relative performance reaches 99.69%. The next best algorithm to add is BI+MF. 100% of AUC is reached when invoking all imputation algorithms *In summary, the addition of BI+MF, BI+PPCA, BI+GAIN and BI+SOFT, provide only marginal gains to the set* $\{$*BI+MM, BI+DAE*$\}$.

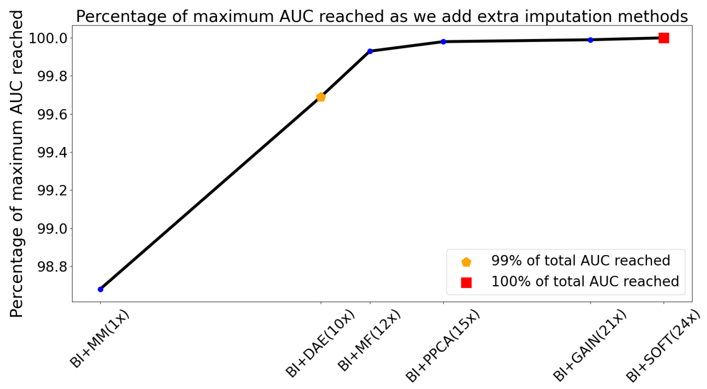


Fig. 5. The percentage of maximum AUC achieved by each subset of imputation methods. Each tick in the x-axis shows the algorithm to add to the previous subset. The multiplier next to the name of an algorithm shows the factor by which the total number of configurations tried is multiplied, due to the different combinations of hyper-parameter values of the imputation algorithms. BI+DAE and BI+MM allow us to recover 99.69% of the maximum AUC while increasing the configuration space by 10 times.

5.5 The interplay between Feature Selection and Imputation

Feature selection algorithms try to reduce the number of features that enter the model without sacrificing predictive performance. Feature selection is often the *primary task* in analysis, while the predictive model may be just a side-benefit. For example, a medical doctor may be more interested in the quantities that determine the risk of disease and may reveal new medical knowledge, rather than the risk model itself. Feature selection leads to more interpretable models that provide intuition into the domain. In fact, the solution to the feature selection problem is directly linked to the causal model that underlies data generation [61]. In other circumstances, it is important to reduce the cost of measuring the features to provide predictions. The cost may be measured in monetary units, the computational cost to compute the features or risk to a patient from medical procedures that measure these features.

Figure 6 shows the impact of feature selection for each imputation algorithm on the real dataset. The drop in AUC performance when feature selection is enforced vs not enforced (i.e., optimizing over all configurations) in the final configuration is shown. For each algorithm is about 2-3 AUC points. In other extensive experiments with hundreds of complete (no missing values), small-sample, high dimensional omics datasets, JADBio has been shown to reduce the number of features by a factor of 4000 without a noticeable drop in AUC performance [62]. The results provide evidence that feature selection may be more challenging in the presence of missing values.

In any case, the problem of including both imputation and a feature selection step in the ML pipeline is that *imputation invalidates feature selection*, in some sense. Let us explain this statement with an example. Let us assume the pipeline that produces the final model consists of MF imputation, Lasso feature selection, and RF predictive modeling. Let us assume that Lasso selects the features $\{A, B, C\}$. If any of these values (say the value of A) is missing on a new sample, the MF imputation model will impute them using a Random Forest for A using some other subsets of features. If any of those are also missing, MF will invoke its Random Forests for each value that is missing, and so on, recursively. Hence, *if there are missing values on the test samples, one may need to measure an arbitrarily large feature subset, not just the selected features.* The storage required to apply the ML pipeline includes both the RF as well as the MF model, which in turn includes a Random Forest for every feature that may need imputation.

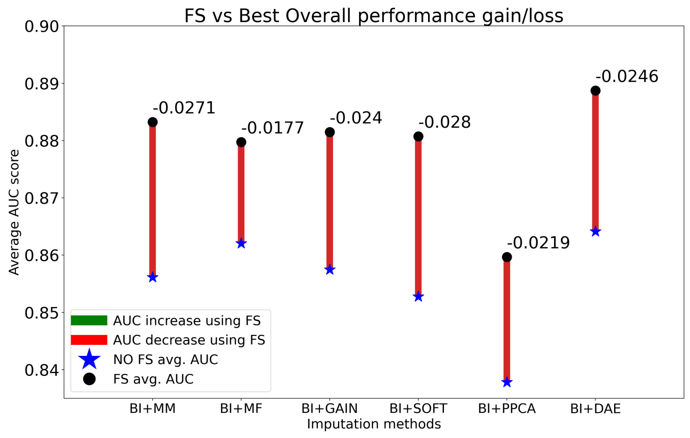


Fig. 6. The figure denotes the loss in predictive performance when enforcing feature selection in the pipeline for the real-world data. For all imputation methods, enforcing feature selection leads to a drop in predictive performance (Red lines) by less than 3 AUC points. While feature selection can reduce the required features to measure for an acceptable loss of performance in some applications, it is invalidated by the imputation models that need to measure all features.

6 COMPARATIVE EVALUATION ON DATASETS WITH SIMULATED MISSING VALUES

This section focuses on comparing imputation methods in datasets with generated missing values. To that end, we compare the predictive performance of each imputation method under various missingness mechanisms and percentages. Additionally, we study the effect of feature selection when the missingness increases. The figures below illustrate the more general MAR case. The results for MCAR results are included in the appendix C.2 and are qualitatively similar.

6.1 BI+MF is the best imputation method in MCAR and MAR simulated missing data

Figure 7 presents the AUC performance results (see Figure 10b for MCAR results). First, we note that the figure illustrates that as the missingness percentage increases the average predictive performance for every imputation method used decreases, as expected. As we can see, increasing the missingness from 25% to 50% leads to a sizable performance drop for all imputation methods. Specifically, methods based on linear dimensionality reduction, namely BI+PPCA and BI+SOFT are the most affected by this increase in missing values.

Figures 7 and 10b illustrate that in both MAR and MCAR data respectively, *MissForest combined with Binary Indicators is, on average, the best-performing method*. Additionally, we note that PPCA and SOFT are the two worst imputation methods, especially as the missingness percentage increases. Table 12a in appendix C.2 contains the quantitative results in detail and a detailed discussion on the ranking of the algorithms.

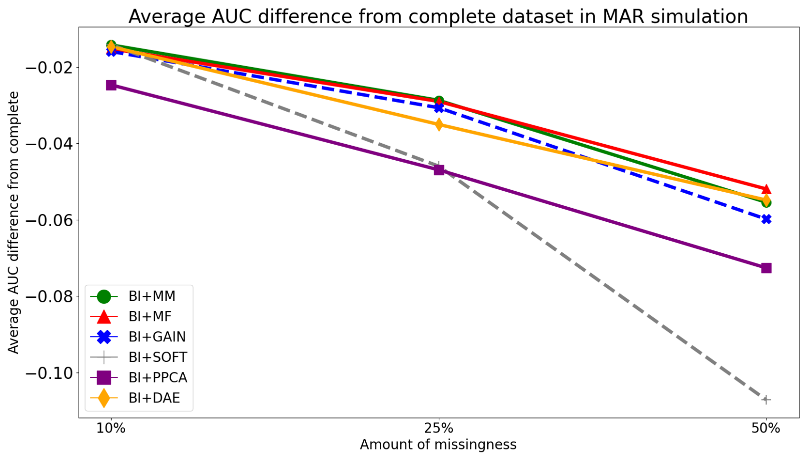


Fig. 7. MAR data: Average AUC difference of each imputation method from the complete dataset. BI+MM is the best at 10% missingness. BI+MF and BI+MM are the top methods, tied at 25% missingness. BI+MF has the lowest avg. loss at 50%.

6.2 The Best Imputation subset for maximizing AUC performance is {BI+MM, BI+MF}

We now identify the minimal-size algorithm subset with close-to-optimal performance for simulated missing data. We use again the simple greedy algorithm introduced in 5.4 and apply it to the MCAR and MAR simulated data results. The results for MAR are in Figure 8, which is similar to Figure 5. The quantitative results are shown in Table 13b in the appendix. As shown in the figure, *the* $\{BI + MM, BI + MF\}$ *subset can score over 99% of the total max AUC for MAR data and would be the suggested set of algorithms to run in such problems.* The results for MCAR are in C.2.5 in the appendix. They are qualitatively similar. The results for simulated missing values are somewhat different than the ones in the real datasets, namely BI+MF scores better than BI+DAE, which is now placed in third place. Possible reasons why are discussed in Section 9.

6.3 BI+MM provides the best trade-off between Effectiveness and Efficiency.

Figure 9, and Figure 10a in the appendix, show the effectiveness vs. efficiency trade-off of the algorithms. The figures are similar to Figure 4a above for the real datasets. We repeat the explanation of the figure: The reference (baseline)

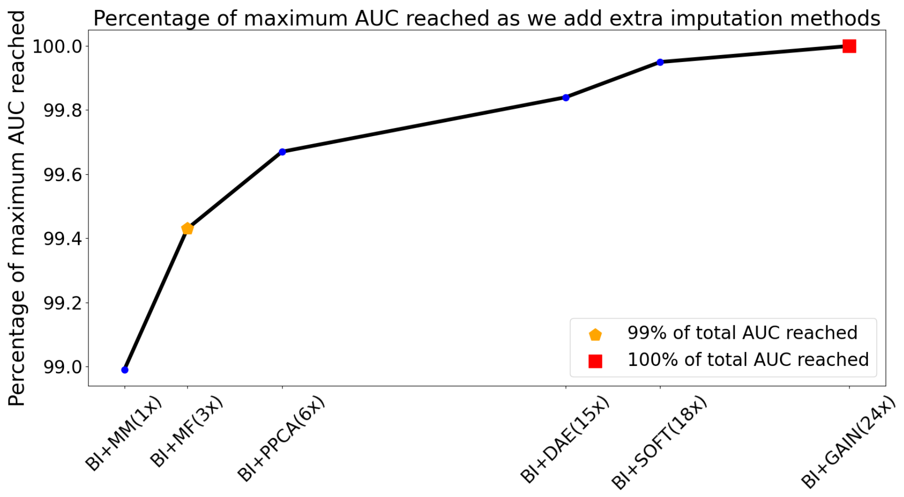


Fig. 8. The percentage of maximum AUC achieved by each subset of imputation methods. Each tick in the x-axis shows the algorithm to add to the previous subset. The multiplier next to the name of an algorithm shows the factor by which the total number of configurations tried is multiplied, due to the different combinations of hyper-parameter values of the imputation algorithms. The set containing BI+MF and BI+MM recovers 99.43% of the maximum AUC for MAR data. The complexity of the configuration space increases by 3 times when including both MM and MF.

algorithm is BI+MM. The x-axis shows the effectiveness ratio defined as the ratio of the AUC corresponding to the point divided by the corresponding performance of BI+MM. Similarly, the y-axis shows the efficiency ratio defined as the training time of the point divided by the corresponding time of BI+MM. Hence, points in the first/fourth quadrant (top-left/bottom-right) correspond to runs where BI+MM dominates/is-dominated-by other algorithms on the same datasets in both time and AUC. Notice that the scale of the y-axis is logarithmic. Larger points correspond to the mean value of an imputation method over all datasets. There are five imputation methods to compare against MM for 10 datasets over 3 percentages of missing values. This will naturally result in 150 points. However, MissForest did not run in three of the datasets (image dataset variations) due to its dimensionality, see Section 2.3.2 for details . The resulting plot will consist of 147 points.

For MAR data, BI+MM is never dominated in both metrics as it is by far the most efficient method. In 104 out of 147 cases, it dominates the opposing imputation methods in terms of both effectiveness and efficiency. However, 43 times it is dominated in efficiency. Only BI+MF has on average better predictive performance than BI+MM. However, it is 23.000 times slower to train on average. All the other imputation methods are worse than BI+MM on average while also taking more time to train. The results for MCAR data are qualitatively similar (see Figure 10a and discussion in Appendix C.2).

In total, BI+MM is again found to provide the best, arguably, trade off between efficiency and effectiveness. The results in the simulated data are further validating the results in the real-world data, verifying that BI+MM is indeed a decent imputation method all around. *BI+MM, on average, is on par with more sophisticated methods such as BI+MF, BI+GAIN, and BI+DAE, while being thousands of times faster to train.*

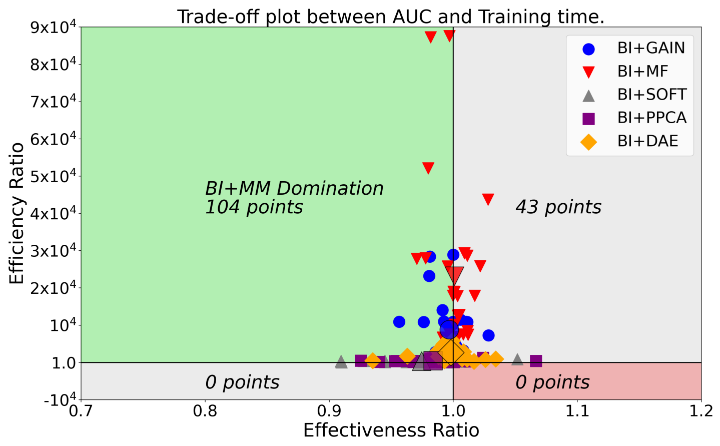


Fig. 9. The Figure illustrates the relative efficiency for each imputation method against BI+MM versus the relative effectiveness in terms of AUC. Larger points indicate the mean values for a given algorithm. 104 out of 147 pairs are won by BI+MM in both efficiency and effectiveness. Only MF dominates MM on average in terms of effectiveness. However, MF is 23.000x slower than MM.

7 META-LEVEL ANALYSIS OF REAL-WORLD RESULTS

In Machine Learning it is always invariably the case that there is no single better algorithm for all datasets, a one-size-fits-all type of algorithm. Hence, one needs to optimize over several choices for the dataset at hand. This school of thought is what gave rise to AutoML systems. The field of Meta-Level Learning [13] studies how to predict the most promising algorithm or algorithms to run on a given dataset based on its characteristics. These characteristics are called *meta-level features or meta-features* of the dataset, and include the sample size, the number of features, the type of features, the percentage of missing values, and others [51].

In this section, we try to identify meta-features that correlate with the performance of the imputation algorithms. Such correlations could help predict which algorithms to run on a given dataset. They could also shed light on the dataset properties that enable an algorithm to perform better and lead to the design of better algorithms. Hence, we defined and computed the meta-features in Table 4. The selected meta-features can be split into three categories: (1) General meta-features, which report general characteristics of the dataset such as the number of samples or features. (2) Missing value-related meta-features, which provide insight into the dataset's missing patterns, such as missing value percentage of features. (3) Cluster-based meta-features. One such type of metric is the silhouette coefficient, computed with the k-means algorithm with $k = 2, 3, 4$, as was proposed in [1]. It shows the tendency of the data to cluster. Another type of such metric is the number of PCA components that explain $\%x$ of the data. It shows whether the data are limited to a lower-dimensional subspace and the extent of cross-correlations between features. General meta-features were extracted using the pymfe package [3]. We implemented the missing and clustering-based meta-feature extraction using sklearn [44] and numpy [18]. To apply clustering or PCA the data are first imputed with MM.

We then correlated (Spearman correlation) these meta-features with the AUC performance of an algorithm relative to the performance of BI+MM as the baseline. A positive (negative) correlation indicates that when the meta-feature increases, the performance of the algorithm increases (decreases), relative to BI+MM. There are 5 algorithms (except BI+MM which is used as a baseline) and 16 meta-features, leading to 80 correlations over datasets. Only one correlation

Table 4. This table reports the meta-features used in the meta-level analysis. The first column contains the name of the meta-feature, the second column denotes the category of the meta-feature, and the third column provides a brief explanation of the meta-feature.

Name	Category	Description
inst_to_attr	General	Samples to features ratio
Minority Class %	General	% of minority class
nr_attr	General	Number of features
nr_inst	General	Number of samples
n_num	General	Number of numerical features
n_cat	General	Number of categorical features
% NA	Missing	% of missing values in data
% samples /w NA	Missing	% of samples with missing values
% features /w NA	Missing	% of features with missing values
% NA/Feat. /w (NA 1+%)	Missing	Mean % of missing values per feature with more than 1% missing
# components 50%	Clustering	Number of components that explain 50% of data variance
# components 70%	Clustering	Number of components that explain 70% of data variance
# components 90%	Clustering	Number of components that explain 90% of data variance
Slh(k=2)	Clustering	Mean Silhouette Coefficient of all samples when using 2 clusters
Slh (k=3)	Clustering	Mean Silhouette Coefficient of all samples when using 3 clusters
Slh (k=4)	Clustering	Mean Silhouette Coefficient of all samples when using 4 clusters

was found to be significant at the level significance 0.1 (p-value = 0.059). Specifically, *BI+PPCA relative performance is positively correlated (correlation = 0.383) with the number of categorical variables in a dataset.* This means that as the number of categorical variables increases we expect BI+PPCA to perform better relative to BI+MM. However, when correcting the p-values for multiple testing using the FDR control technique of Benjamini-Hochberg [5], we see that the q-value is 0.991 which means that detecting one such correlation is expected even if all meta-features are uncorrelated with the relative performance. BI+PPCA doesn't handle categorical features natively, which further makes us believe that the result is probably a false positive. Statistically significant correlations couldn't be found using meta-learning analysis. Further experiments containing more datasets and meta-features need to be conducted.

8 RELATED WORK

In this section, we discuss related work on missing values imputation and position our contributions. We focus on empirical studies that compare different imputation methods based on the performance of the predictive models build on imputed datasets rather than the original values of complete datasets [23], [69], [47], [10].

Current literature can be split into two categories: AutoML and Adhoc ML modeling. The first category extends a specific AutoML tool by adding imputation methods, while the latter creates a predictive modeling pipeline that may contain a subset of a modern AutoML tool's pipeline, such as hyper-parameter optimization, model selection, and pre-processing. AutoML in general is able to optimize the performance over various stages in a pipeline. As we optimize the whole pipeline, we expect the effect of each stage to become less significant, as other stages may compensate. AutoML tools allow us to get more insights on which features are more important for the task (feature selection), optimize the hyper-parameters for each stage of the pipeline (hyper-parameter tuning), and select the best predictive model for each imputation method (model selection). Consequently, imputation methods can be evaluated fairly under this optimization framework.

Table 5. An overview of related work on predictive modeling. Most benchmarks either use datasets with simulated missing values or with native, but not both. Abbreviations: # symbol means number, "—" denotes that the paper doesn't mention any details about the topic, On column data B, M, R denotes binary, multiclass, and regression datasets respectively., On the column mechanism, values Nat, MC, MR, and MN denote Native, MCAR, MAR, and MNAR. On column #Imp. methods, N, C, and M denotes numerical, categorical, and mixed imputation method respectively. The NNs column denotes that Neural Network imputation methods were included., BI means that methods were extended with BIs, FS means feature selection was included in the pipeline, and Tuning represents whether the study tuned the imputation methods (Imp.), the predictive models(pred.), or both (Imp+Pred). Meta presents whether a study has conducted a meta-level analysis.
*One of the features was made missing. **missing values were only generated on the train data.

Study	#Datasets	Mechanism	% Missing values	#Imp. methods	NNs	BI	System	FS	Tuning	#Models	Meta
[31]	6B	Nat	7%-84%	3N, 2C, 2M.	No	No	Adhoc	No	Imp+Pred	7	No
[70]	13B	Nat	0.6%-33.6%	2N,1C,4M	No	No	Adhoc	No	No	5	No
[48]	2B	MC,MR	0%-40% **	7C	No	No	Adhoc	No	No	3	No
[6]	5B,5R	MC-MN	10% -50%	8M	No	No	Adhoc	No	Imp.	4	No
[24]	31B, 21R, 17M	MC,MR,MN	1% -50%*	6M	Yes	No	Adhoc	No	Imp.	2	No
[46]	10B, 3R	Nat	-	7M	No	Yes	Adhoc	No	Pred.	3	No
[14]	23B,M	MC	7%	6M	No	No	AutoML	Yes	-	10	No
[41]	5B	Nat	-	4N, 2C, 1M	No	No	AutoML	-	Imp+Pred	Ensemble	No
Ours	35B	N,MC,MR	1%-72%	6M	Yes	Yes	AutoML	Yes	Imp+Pred	4	Yes

As shown in table 5, the majority of related work either uses datasets with native or simulated missing values. Literature mainly focuses on the binary classification task (included in all previous works). Out of the 8 previous works, 2 papers include binary+regression [46] [6] and 1 work binary and multi-class data [14]. [24] is the only study that includes all three types of outcomes. The most prominent missingness mechanism is MCAR found in all works that simulate missing values. [24] is the only work that includes deep learning-based imputation methods. Binary Indicators are very prominent in AutoML tools however only [46] has studied their effect when extending imputation methods. Finally, [41] is the only work that included ensemble models for the prediction phase while [14] is the only work that includes feature selection as part of the pipeline. As shown in the table 5, none of the related work has included every step mentioned in the table's columns.

Summarizing the related work section, the majority of the literature uses datasets with native missing values or generates them through a simulation based on various missingness mechanisms and missingness proportions. However, none of the mentioned studies benchmarks imputation methods on both native and generated missing value datasets. The studies on real-world datasets in general conclude that simple imputation methods such as MM are on par with other more complex methods. Research on datasets with simulated missing values concludes that more complex methods can indeed improve predictive performance on average. However, there is no universal best method proposed by any of the aforementioned benchmarks. Literature mainly focuses on the binary classification task. Specific predictive models could benefit from native handling of missing values compared to simple imputation, for instance, Gradient Boosted Trees. However, not all classifiers support missing value handling, making imputation still an essential part of the pre-processing step of ML pipelines. In general, hyper-parameter tuning, model selection, and feature selection are given less importance in previous literature. Most works skip one or more of the previous steps or fail to mention information about the specific stage. For example, only one predictive model is tuned or imputation methods are used with default parameters specified by the authors of the methods or the package implementations.

The research closer to ours is [41]. In the aforementioned paper, Autosklearn was extended to include the data cleaning process, the emerging tool named AutoClean. Part of the extension was imputation. The study compares mean , median, mode, KNN [60], and Iterative imputation [66] for continuous features. For categorical features, constant, KNN, and mode imputation were selected. The study used 5 binary classification datasets with 891 to 10.500 observations and 9 to 39 features that include missing values at low percentages. AutoClean optimized the pipeline by Bayesian hyper-parameter optimization. In autosklearn the predictive model is an ensemble of methods. The study concluded that KNN is a valuable addition to the simpler imputation methods. However, in most cases, simple imputation methods are selected more frequently than KNN for both continuous and categorical data. Contrary to the aforementioned literature, we included feature selection in our experimental setup. Also, we conducted comparisons on both datasets with native and simulated missing values. In general, our evaluation was conducted on more datasets, with a higher range in terms of samples and features. Finally, we included neural network imputation methods and extended imputation with binary indicators.

In [14], TPOT AutoML tool was extended with imputation methods, specifically mean, median, mode, max, MICE [66], and EM [21]. The median and mode were found to be the best imputation methods based on a restricted simulation study on 23 datasets at 7% MCAR missingness. Compared to the mentioned work, we simulated missing values with other mechanisms and missing proportions as well as used datasets with native missing values in our experiments. Also, we included recent state-of-the-art methods based on NNs such as DAE and GAIN. We also implemented and measured the effect of binary indicators when coupled with MM and complex methods.

Similarly, missing data imputation has been also researched as part of the data cleaning systems. [31] compared deletion, mean, median, mode, new-category, and HoloClean [50] on 6 datasets with native missing values. For the predictive task 7 models were considered: Logistic Regression, k-Nearest Neighbors (KNN), Decision Tree, Random Forest, AdaBoost, Naive Bayes, and XGBoost. They concluded that simple imputation methods yield competitive performance to more complex methods such as Holoclean. Contrary to the aforementioned work, we included neural networks, the binary indicator method, and feature selection in the experimental setup. We conducted experiments on more datasets that, generated missing values but also had real-world missing values.

The benchmark study [70] was conducted on 13 real-world datasets from OpenML and concluded that mean/mode is comparable to more complex imputation methods such as random, SOFT [36], MF [55], KNN [60], Hot-Deck [27], and MICE [66]. Specifically, while measuring the F1 score MM had the highest average ranking. In contrast, for the AUC score, KNN is found to be the best-performing method. However, both KNN and MM are among the three best methods in both metrics. Hyper-parameter tuning was not considered in this paper, the imputation and predictive methods used default parameters. In our work, we tune all the steps of the pipeline for a fair comparison. We also included deep learning methods that are the current state-of-the-art for imputation as well as binary indicators.

The largest benchmark study was conducted by [24] on 69 real-world datasets with simulated missing values. The missing mechanisms were MCAR, MAR, and MNAR. The generation of missing values was set at 1%, 10%, 30%, and 50% missingness. They compared MM, KNN [60] , MF [55] , custom DL-based imputation inspired by [7], GAIN [72] and variational autoencoders [26] for the imputation problem. They concluded that MissForest is the best imputation method. However, they used a single classifier for the prediction phase. We argue that different imputation methods work better with different classifiers, which should be tuned as well. For example, on the cylinder-band dataset GAIN imputation method performs best with the Ridge Logistic Regression, whereas the DAE imputation method performs best with the RandomForest classifier on the same dataset. Additionally, missing values for the downstream task were generated for a randomly sampled feature in the dataset. We also uniformly simulated missing values which is a harder problem to solve for the imputation methods as less observed data exist. In the mentioned work, GAIN had a convergence problem in 33% of the cases resulting in the worst ranking among the mentioned methods. In our work, GAIN does indeed converge due to different hyperparameter tuning. Finally, we include Binary Indicators as well as the DAE imputation method which is the best method on average in real-world data with missing values. Simulated missingness results in our work are on par with the results of the aforementioned work as MissForest is the best method in both works. However, deep learning methods in our work are among the best methods and not the worst as in the literature mentioned.

Another study [48], compared the predictive performance on two datasets with imputed and incomplete data. Missing values were simulated on categorical features on the train data. They generated MCAR and MNAR missing values from 10% to 40% missingness in categorical features. For the imputation of categorical features, they used six imputation models: mode, random, k-NN [60], iterative imputation based on logistic regression, random forest [55], and SVM. For the classification, they used three predictive models, ANNs, decision trees, and random forests. The authors optimized the hyper-parameters for the ANNs only. The imputation models and the other classifiers were not tuned. They didn't conclude that an imputation method or a classifier is better than others and heavily depends on the nature and proportion of the missing data. However, results indicated that imputation is better than simply creating a new category in the data. In our work, for fair evaluation, we tune both imputation and predictive models. We include binary indicator methods and neural network imputation. Finally, we simulated missingness on both numeric and categorical features.

[46] compared mean, median, KNN [60], Iterative Imputer, Iterative Imputer /w Bagging (Multiple Imputation) [66], MIA (The native handling of missing values by Gradient Boosted Trees), and MIA /w bag. All of the previous methods, except MIA, were also extended with binary indicators. The study was conducted on 13 real-world datasets from 4 databases with native missing values. The predictive models were set to Gradient-boosted trees and linear models. They concluded that MIA is a better alternative to imputation. Also, the indicator method helps improve the performance of the predictive task which is on par with the results of our work. They conclude that simple imputation using mean or median is on par with KNN and iterative imputation with linear models. In our work, we included deep learning imputation in our set of imputation methods. Also, we tuned the imputation methods to fairly evaluate the performance of each method, as tuning is important in the performance of some imputation methods.

Last but not least, [6] introduced a group of three methods named OptImpute, focusing on optimizing KNN and iterative imputation based on SVMs and decision trees. They compared the group against five other imputation methods: mean/mode, K-nearest neighbors [60], iterative known [73], Bayesian PCA [42], and predictive-mean matching [66]. They compared the introduced method across 84 datasets with simulated missing values measuring imputation accuracy. They additionally measured the group's performance on learning algorithm performance on 10 datasets. The missing values are generated by the MCAR mechanism with a range from 10% to 50%. The classifiers used for regression tasks are LASSO and SVR while for the classification tasks SVM and Optimal Trees. These data sets range in size, having 150 to 5,875 observations and 4 to 16 features. Their group of methods improved the predictive performance of the models. Their method scored 86.1% average accuracy and average R-Squared of 0.339 compared to 84.4% and 0.315 R-Squared for the classification and regression data respectively. However, no neural networks were used and the methods introduced haven't been compared individually. Additionally, tuning was applied only to the group of proposed methods, and not to the other imputation methods that were used and the predictive models. We tune all of the imputation methods and models. We also extend methods with binary indicators and include NNs imputation methods in our test bed, as well as MissForest. Finally, we have a wider range of datasets, both with native and simulated missing values.

8.1 Synopsis of Contributions Relative to the Related Work

Compared to the related work, we contribute in various ways. Our work can be directly compared to 2 other works that are conducted in an AutoML tool [41] [14]. Compared to the mentioned works, we include more datasets, more missingness mechanisms, neural network methods, and binary indicators in the experimental setup. For the first time, deep learning methods are compared to simple imputation methods in an AutoML predictive setting. One of the deep learning methods, BI+DAE has the best average performance on real-world data with native missing values. Additionally, for the first time, the effect of the imputation methods on predictive performance is measured on datasets with generated missing values and native missing values. Till now, comparisons were conducted on only one of the two settings, specifically, half of the papers use real-world datasets with missing values in them while the other half use complete datasets with generated missing values. Contrary to the majority of the literature, we tune both imputation and predictive methods to fairly evaluate them. Only 2 out of the 8 related mention tuning both imputation and predictive modeling methods [41], [31]. We also conducted experiments on more datasets compared to the majority of the literature, while unlike [24], our simulation setting is applied to all features in the datasets and not only one. Also, only 1 out of the 8 aforementioned works includes feature selection as part of the ML pipeline. Finally, meta-learning, for the first time, is used to identify useful data characteristics that could give insights into the choice of a simple vs a sophisticated imputation method. In general, as shown in table 5, our testbed is the most complete overall in terms of dataset selection, missingness selection, imputation method selection, and pipeline steps. This allows us to fairly evaluate imputation methods in a state-of-the-art AutoML environment.

9 LESSONS LEARNED AND CONTRIBUTIONS

The main insights that are drawn from our experimental results are the following:

- *Including Binary Indicators (BI) in the dataset improves the predictive performance of the machine learning pipeline, for most algorithms* (see Section 5.1). The inclusion of BIs does increase the dimensionality – and difficulty – of the machine learning task. On the other hand, it does encode the information about which missing values are missing; this, allows a classifier to learn which values to trust or not. Results indicate that encoding this information turns out to be more beneficial than harmful.
- *BI+DAE is found to be the single best imputation method in real-world data with native missing values followed by BI+MM, which is the standard in AutoML tools.* As seen in Section 5.2 and Table 10 both methods have the same number of wins across datasets with BI+DAE having higher mean AUC. The worst performance is exhibited by matrix-factorization (linear dimensionality reduction) methods such as PPCA. These methods do scale with the number of features and may be more suitable for high-dimensional, low-sample datasets.
- *BI+MM exhibits the best trade-off between efficiency and effectiveness.* As expected (see Section 5.3, 6.3, and C.2.4) BI+MM is the fastest method to train and also is more effective in the majority of the comparisons. MF due to its iterative nature is the slowest among all closely followed by GAIN. GAIN's main bottleneck is the number of epochs required to train the network. The authors' suggestion was 10.000 epochs which is 20 times more than the 500 epochs suggested by the authors of the DAE method.
- Based on the results of Section 5.4, we would suggest practitioners to optimize their models over the BI+MM and BI+DAE algorithms. *BI+MM and BI+DAE score over 99% of the maximum AUC in real-world data as shown in Section 5.4.* Specifically, BI+MM scores 98.68% of the maximum AUC. Adding BI+DAE to the pipeline leads to 99.69% of the maximum AUC. However, this comes at the cost of increasing the configuration space by

10x, as DAE has 9 tuning configurations compared to 1 of BI+MM. Also, in order to reach 100% of the optimal performance, we have to train 24-times more configurations than by simply using BI+MM.

- *BI+MF is the best method in datasets with simulated missing values.* As shown in section 6.1, in both MCAR and MAR simulations, BI+MF is on average the best. In contrast, BI+MF is the third best with real-world data, falling behind BI+DAE. Despite our best efforts to realistically simulate missing values, there may still be differences between real-world missing-data generative mechanisms and our simulations. First, we simulated MCAR and MAR missing values. Real-world missing values may be NMAR. Second, the missingness probability for MAR data is determined by a generalized linear model (logistic regression model). Real-world missing values may follow non-linear models. The majority of the literature employs similar simulations for comparing imputation algorithms. However, as indicated by this study, results with simulated missingness may not generalize to real-world datasets. New simulation methodologies need to be proposed to this end.
- *Missingness increase leads to a deterioration in predictive performance.* As shown in section 6.1, increasing missingness, causes a drop in the AutoML tool's capability of predicting the outcome. Missingness at 10% leads to a 0.025 AUC drop compared to the complete dataset. Similarly, 25% missingness leads to 0.048 AUC drop, while at 50% we can inspect up to 0.143 drop average as seen in tables 12a, 12b.
- *The set containing BI+MM and BI+MF reaches 99% of maximum AUC for simulated data as shown in section 6.2.* BI+MM scores the 98.7% of the maximum AUC for MCAR data and 98.99% for MAR data. In order to surpass 99% of the maximum AUC, the addition of BI+MF is needed. This addition, allows the tool to reach 99.62% and 99.43% on MCAR and MAR data respectively. However, BI+MF has to be tuned, leading to a total 3x increase in pipeline complexity.
- A Meta-Learning methodology to correlate meta-features with performance is presented in Section 7. It could allow scientists to select the appropriate sophisticated methods based on meta-features, saving training time and improving overall performance. In addition, it could provide insight into the design choice of an algorithm that leads to better or worse performance on a given dataset. Unfortunately, no statistically significant results were found. This means that either there are no correlations present with the selected meta-features, or these correlations are not strong enough to be found significant with the given sample size of 25 datasets.

There are, of course, several **limitations** of the study that we would like to point to. The results and conclusions stem from computational experiments with binary classification tasks within a range of a number of features, sample size, imbalance of the classes, and missingness percentage. Despite the significant computational effort involved – optimizing over thousands of ML pipelines for each dataset – results stem from only 25 real-world datasets with native missing values, and 60 complete datasets where missing values were introduced (10 original datasets times 2 missingness mechanisms (MCAR, MAR) times 3 missingness percentages). This fact limits the statistical power of our statistical tests. While JADBio is an effective AutoML tool, results should also be obtained from other AutoML tools to further generalize the conclusions.

10 CONCLUSIONS

In this paper, we conducted experiments on real-world datasets with native missing values and simulated missing values. We compared six imputation methods extended by binary indicators, on a state-of-the-art AutoML tool. BI+DAE is the best method on real-world datasets with native missing values. However, BI+MM is comparable, if not better than the more sophisticated imputation methods in terms of predictive performance and efficiency on real-world data.

Increasing missingness leads to predictive performance deterioration. Additionally, simulation data lead to contradicting results compared to real-world datasets. BI+DAE and BI+MM are the best methods on real-world data, however, when simulated data are considered BI+MF is the best method on average followed by BI+MM. Finally, meta-learning was employed but could not successfully find any patterns to predict whether a sophisticated imputation method can be used instead of the simple BI+MM to improve the downstream performance.

The results make us question whether advanced, multivariate imputation algorithms are really necessary for predictive modeling with AutoML. The simple BI+MM imputation is surprisingly effective and computationally efficient when the ML pipeline is properly tuned within an AutoML setting. BI features allow advanced classifiers to learn when to trust a value or not. Multivariate Imputation algorithms try to learn the full joint distribution of the dataset, a task that is quite challenging with low sample, imbalanced, or high-dimensional data, and prone to error. It is also a very computationally demanding task. Imputing values for features that are redundant or irrelevant to the final model is a waste of computations. When imputing using multivariate imputation, one needs to store not only the final model (e.g. RF, SVM, or a NN) but also the imputation model to impute test samples. For some imputation models (Deep Neural Networks, or one RF for each feature as in MF) the additional storage may be non-negligible. In addition, the imputation model requires measuring all features and invalidates the efforts of feature selection. Arguably, the research effort that goes into novel and better-perfoming imputation methods would be more productive to be spent on novel and better-performing ways to natively handle missing values in our classification and feature selection algorithms.

ACKNOWLEDGMENTS

The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant" (Project Number: 1941).

REFERENCES

[1] 2023. Under-Submission.

[2] Ahmed Alaa and Mihaela van der Schaar. 2018. AutoPrognosis: Automated Clinical Prognostic Modeling via Bayesian Optimization with Structured Kernel Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 139–148. https://proceedings.mlr.press/v80/alaa18b.html

[3] Edesio Alcobaça, Felipe Siqueira, Adriano Rivolli, Luís P. F. Garcia, Jefferson T. Oliva, and André C. P. L. F. de Carvalho. 2020. MFE: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research* 21, 111 (2020), 1–5. http://jmlr.org/papers/v21/19-348.html

[4] Rebecca R Andridge and Roderick J A Little. 2010. A Review of Hot Deck Imputation for Survey Non-response. *Int Stat Rev* 78, 1 (April 2010), 40–64.

[5] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)* 57, 1 (1995), 289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x arXiv:https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1995.tb02031.x

[6] Dimitris Bertsimas, Colin Pawlowski, and Ying Daisy Zhuo. 2018. From Predictive Methods to Missing Data Imputation: An Optimization Approach. *Journal of Machine Learning Research* 18, 196 (2018), 1–39. http://jmlr.org/papers/v18/17-073.html

[7] Felix Biessmann, Tammo Rukat, Phillipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing Value Imputation for Tables. *Journal of Machine Learning Research* 20, 175 (2019), 1–6. http://jmlr.org/papers/v20/18-753.html

[8] Christopher Bishop. 1998. Bayesian PCA. In *Advances in Neural Information Processing Systems*, M. Kearns, S. Solla, and D. Cohn (Eds.), Vol. 11. MIT Press. https://proceedings.neurips.cc/paper/1998/file/c88d8d0a6097754525e02c2246d8d27f-Paper.pdf

[9] Ramiro Daniel Camino, Christian A. Hammerschmidt, and Radu State. 2019. Improving Missing Data Imputation with Deep Generative Models. *CoRR* abs/1902.10666 (2019). arXiv:1902.10666 http://arxiv.org/abs/1902.10666

[10] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. 2021. A survey on missing data in machine learning. *Journal of Big Data* 8, 1 (27 Oct 2021), 140. https://doi.org/10.1186/s40537-021-00516-9

[11] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. https://doi.org/10.48550/ARXIV.2003.06505

[12] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2020. Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning. (2020).

[13] João Gama and Pavel Brazdil. 1995. Characterization of Classification Algorithms. 189–200. https://doi.org/10.1007/3-540-60428-6_16

[14] Unai Garciarena, Roberto Santana, and Alexander Mendiburu. 2017. Evolving imputation strategies for missing data in classification problems with TPOT. *CoRR* abs/1706.01120 (2017). arXiv:1706.01120 http://arxiv.org/abs/1706.01120

[15] Pieter Gijsbers and Joaquin Vanschoren. 2020. GAMA: a General Automated Machine learning Assistant. *CoRR* abs/2007.04911 (2020). arXiv:2007.04911 https://arxiv.org/abs/2007.04911

[16] Lovedeep Gondara and Ke Wang. 2018. *MIDA: Multiple Imputation Using Denoising Autoencoders*. 260–272. https://doi.org/10.1007/978-3-319-93040-4_21

[17] H2O.ai. 2022. *DriverlessAI*. https://www.h2o.ai/products/h2o-driverless-ai/

[18] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[19] Harshad Hegde, Neel Shimpi, Aloksagar Panny, Ingrid Glurich, Pamela Christie, and Amit Acharya. 2019. MICE vs PPCA: Missing data imputation in healthcare. *Informatics in Medicine Unlocked* 17 (2019), 100275. https://doi.org/10.1016/j.imu.2019.100275

[20] Steffen Herbold. 2020. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software* 5, 48 (2020), 2173. https://doi.org/10.21105/joss.02173

[21] James Honaker, Gary King, and Matthew Blackwell. 2011. Amelia II: A Program for Missing Data. *Journal of Statistical Software* 45, 7 (2011), 1–47. https://doi.org/10.18637/jss.v045.i07

[22] Md Hamidul Huque, John B. Carlin, Julie A. Simpson, and Katherine J. Lee. 2018. A comparison of multiple imputation methods for missing data in longitudinal studies. *BMC Medical Research Methodology* 18, 1 (12 Dec 2018), 168. https://doi.org/10.1186/s12874-018-0615-6

[23] Anil Jadhav, Dhanya Pramod, and Krishnan Ramanathan. 2019. Comparison of Performance of Data Imputation Methods for Numeric Dataset. *Applied Artificial Intelligence* 33 (07 2019), 1–21. https://doi.org/10.1080/08839514.2019.1637138

[24] Sebastian Jäger, Arndt Allhorn, and Felix Bießmann. 2021. A Benchmark for Data Imputation Methods. *Frontiers in Big Data* 4 (2021). https://doi.org/10.3389/fdata.2021.693674

[25] Jintao Ke, Shuaichao Zhang, Hai Yang, and Xiqun (Michael) Chen. 2019. PCA-based missing information imputation for real-time crash likelihood prediction under imbalanced data. *Transportmetrica A: Transport Science* 15, 2 (2019), 872–895. https://doi.org/10.1080/23249935.2018.1542414 arXiv:https://doi.org/10.1080/23249935.2018.1542414

[26] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. https://doi.org/10.48550/ARXIV.1312.6114

[27] Alexander Kowarik and Matthias Templ. 2016. Imputation with the R Package VIM. *Journal of Statistical Software* 74, 7 (2016), 1–16. https://doi.org/10.18637/jss.v074.i07

[28] Gayaneh Kyureghian, Oral Capps, and Rodolfo M. Nayga. 2011. *A missing variable imputation methodology with an empirical application*. 313–337. https://doi.org/10.1108/S0731-9053(2011)000027A015

[29] Trang T Le, Weixuan Fu, and Jason H Moore. 2020. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics* 36, 1 (2020), 250–256.

[30] Dan Li, Jitender Deogun, William Spaulding, and Bill Shuart. 2004. Towards Missing Data Imputation: A Study of Fuzzy K-means Clustering Method. In *Rough Sets and Current Trends in Computing*, Shusaku Tsumoto, Roman Słowiński, Jan Komorowski, and Jerzy W. Grzymała-Busse (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 573–579.

[31] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2019. CleanML: A Benchmark for Joint Data Cleaning and Machine Learning [Experiments and Analysis]. *CoRR* abs/1904.09483 (2019). arXiv:1904.09483 http://arxiv.org/abs/1904.09483

[32] Yuebiao Li, Zhiheng Li, and Li Li. 2014. Missing traffic data: Comparison of imputation methods. *Intelligent Transport Systems, IET* 8 (02 2014), 51–57. https://doi.org/10.1049/iet-its.2013.0052

[33] R.J.A. Little and D.B. Rubin. 2002. *Statistical analysis with missing data*. Wiley. http://books.google.com/books?id=aYPwAAAAMAAJ

[34] Ms. R. Malarvizhi. 2012. K-NN Classifier Performs Better Than K-Means Clustering in Missing Value Imputation. *IOSR Journal of Computer Engineering* 6 (2012), 12–15.

[35] Behrooz Mamandipoor, Mahshid Majd, Monica Moz, and Venet Osmani. 2019. Blood lactate concentration prediction in critical care patients: handling missing values. *CoRR* abs/1910.01473 (2019). arXiv:1910.01473 http://arxiv.org/abs/1910.01473

[36] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research* 11, 80 (2010), 2287–2322. http://jmlr.org/papers/v11/mazumder10a.html

[37] John McCoy, Steve Kroon, and Lidia Auret. 2018. Variational Autoencoders for Missing Data Imputation with Application to a Simulated Milling Circuit. *IFAC-PapersOnLine* 51 (01 2018), 141–146. https://doi.org/10.1016/j.ifacol.2018.09.406

[38] Karthika Mohan and Judea Pearl. 2021. Graphical Models for Processing Missing Data. *J. Amer. Statist. Assoc.* 116, 534 (2021), 1023–1037. https://doi.org/10.1080/01621459.2021.1874961 arXiv:https://doi.org/10.1080/01621459.2021.1874961

[39] Carol M. Musil, Camille B. Warner, Piyanee Klainin Yobas, and Susan L. Jones. 2002. A Comparison of Imputation Techniques for Handling Missing Data. *Western Journal of Nursing Research* 24, 7 (2002), 815–829. https://doi.org/10.1177/019394502762477004

arXiv:https://doi.org/10.1177/019394502762477004 PMID: 12428897.

[40] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. 2020. Missing Data Imputation using Optimal Transport. arXiv:2002.03860 [stat.ML]

[41] Felix Neutatz, Binger Chen, Yazan Alkhatib, Jingwen Ye, and Ziawasch Abedjan. 2022. Data Cleaning and AutoML: Would an Optimizer Choose to Clean? *Datenbank-Spektrum* 22, 2 (01 Jul 2022), 121–130. https://doi.org/10.1007/s13222-022-00413-2

[42] Shigeyuki Oba, Masa-aki Sato, and Shin Ishii. 2003. Variational Bayes method for Mixture of Principal Component Analyzers. *Systems and Computers in Japan* 34, 11 (2003), 55–66. https://doi.org/10.1002/scj.10394 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/scj.10394

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

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

[45] Ricardo Cardoso Pereira, Miriam Seoane Santos, Pedro Pereira Rodrigues, and Pedro Henriques Abreu. 2020. Reviewing Autoencoders for Missing Data Imputation: Technical Trends, Applications and Outcomes. *J. Artif. Intell. Res.* 69 (2020), 1255–1285.

[46] Alexandre Perez-Lebel, Gael Varoquaux, Marine Le Morvan, Julie Josse, and Jean-Baptiste Poline. 2022. Benchmarking missing-values approaches for predictive models on health databases. *GigaScience* 11 (04 2022). https://doi.org/10.1093/gigascience/giac013

[47] Ben Omega Petrazzini, Hugo Naya, Fernando Lopez-Bello, Gustavo Vazquez, and Lucía Spangenberg. 2021. Evaluation of different approaches for missing data imputation on features associated to genomic data. *BioData Mining* 14, 1 (03 Sep 2021), 44. https://doi.org/10.1186/s13040-021-00274-7

[48] Jason Poulos and Rafael Valle. 2018. Missing Data Imputation for Supervised Learning. *Applied Artificial Intelligence* 32, 2 (2018), 186–196. https://doi.org/10.1080/08839514.2018.1448143 arXiv:https://doi.org/10.1080/08839514.2018.1448143

[49] Li Qu, Li Li, Yi Zhang, and Jianming Hu. 2009. PPCA-Based Missing Data Imputation for Traffic Flow Volume: A Systematical Approach. *IEEE Transactions on Intelligent Transportation Systems* 10, 3 (2009), 512–522. https://doi.org/10.1109/TITS.2009.2026312

[50] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (aug 2017), 1190–1201. https://doi.org/10.14778/3137628.3137631

[51] Adriano Rivolli, Luís P.F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C.P.L.F. de Carvalho. 2022. Meta-features for meta-learning. *Knowledge-Based Systems* 240 (2022), 108101. https://doi.org/10.1016/j.knosys.2021.108101

[52] Donald B. Rubin. 1976. Inference and Missing Data. *Biometrika* 63, 3 (1976), 581–592. http://www.jstor.org/stable/2335739

[53] Seunghyoung Ryu, Minsoo Kim, and Hongseok Kim. 2020. Denoising Autoencoder-Based Missing Value Imputation for Smart Meters. *IEEE Access* PP (02 2020), 1–1. https://doi.org/10.1109/ACCESS.2020.2976500

[54] Wolfram Stacklies, Henning Redestig, Matthias Scholz, Dirk Walther, and Joachim Selbig. 2007. pcaMethods a bioconductor package providing PCA methods for incomplete data. *Bioinformatics* 23, 9 (03 2007), 1164–1167. https://doi.org/10.1093/bioinformatics/btm069 arXiv:https://academic.oup.com/bioinformatics/article-pdf/23/9/1164/761822/btm069.pdf

[55] Daniel J. Stekhoven and Peter Bühlmann. 2011. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (10 2011), 112–118. https://doi.org/10.1093/bioinformatics/btr597 arXiv:https://academic.oup.com/bioinformatics/article-pdf/28/1/112/583703/btr597.pdf

[56] Jonathan A C Sterne, Ian R White, John B Carlin, Michael Spratt, Patrick Royston, Michael G Kenward, Angela M Wood, and James R Carpenter. 2009. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ* 338 (2009). https://doi.org/10.1136/bmj.b2393 arXiv:https://www.bmj.com/content

[57] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2012. Auto-WEKA: Automated Selection and Hyper-Parameter Optimization of Classification Algorithms. *CoRR* abs/1208.3719 (2012). arXiv:1208.3719 http://arxiv.org/abs/1208.3719

[58] Robert Tibshirani. 1996. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x arXiv:https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1996.tb02080.x

[59] Michael E. Tipping and Christopher M. Bishop. 1999. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 61, 3 (1999), 611–622. http://www.jstor.org/stable/2680726

[60] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. 2001. Missing value estimation methods for DNA microarrays . *Bioinformatics* 17, 6 (06 2001), 520–525. https://doi.org/10.1093/bioinformatics/17.6.520 arXiv:https://academic.oup.com/bioinformatics/article-pdf/17/6/520/760366/170520.pdf

[61] Ioannis Tsamardinos and Constantin F. Aliferis. 2003. Towards Principled Feature Selection: Relevancy, Filters and Wrappers. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. R4)*, Christopher M. Bishop and Brendan J. Frey (Eds.). PMLR, 300–307. https://proceedings.mlr.press/r4/tsamardinos03a.html Reissued by PMLR on 01 April 2021..

[62] Ioannis Tsamardinos, Paulos Charonyktakis, Georgios Papoutsoglou, Giorgos Borboudakis, Kleanthi Lakiotaki, Jean Claude Zenklusen, Hartmut Juhl, Ekaterini Chatzaki, and Vincenzo Lagani. 2022. Just Add Data: automated predictive modeling for knowledge discovery and feature selection. *npj Precision Oncology* 6, 1 (16 Jun 2022), 38. https://doi.org/10.1038/s41698-022-00274-8

[63] Ioannis Tsamardinos, Elissavet Greasidou, Michalis Tsagris, and Giorgos Borboudakis. 2017. Bootstrapping the Out-of-sample Predictions for Efficient and Accurate Cross-Validation. *CoRR* abs/1708.07180 (2017). arXiv:1708.07180 http://arxiv.org/abs/1708.07180

[64] I Tsamardinos, V Lagani, and D Pappas. 2012. Discovering multiple, equivalent biomarker signatures. In *7th Conference of the Hellenic Society for Computational Biology and Bioinformatics (HSCBB12). Heraklion.*

[65] S. van Buuren and C. G. M. Groothuis-Oudshoorn. 1999. *Flexible Multivariate Imputation by MICE.* Vol. (PG/VGZ/99.054). TNO Prevention and Health, Leiden.

[66] Stef van Buuren and Karin Groothuis-Oudshoorn. 2011. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software* 45, 3 (2011), 1–67. https://doi.org/10.18637/jss.v045.i03

[67] J. Vanschoren, J.N. van Rijn, B. Bischl, and L. Torgo. 2013. OpenML : networked science in machine learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[68] Akbar K Waljee, Ashin Mukherjee, Amit G Singal, Yiwei Zhang, Jeffrey Warren, Ulysses Balis, Jorge Marrero, Ji Zhu, and Peter DR Higgins. 2013. Comparison of imputation methods for missing laboratory data in medicine. *BMJ Open* 3, 8 (2013). https://doi.org/10.1136/bmjopen-2013-002847 arXiv:https://bmjopen.bmj.com/content/3/8/e002847.full.pdf

[69] Akbar K Waljee, Ashin Mukherjee, Amit G Singal, Yiwei Zhang, Jeffrey Warren, Ulysses Balis, Jorge Marrero, Ji Zhu, and Peter DR Higgins. 2013. Comparison of imputation methods for missing laboratory data in medicine. *BMJ Open* 3, 8 (2013). https://doi.org/10.1136/bmjopen-2013-002847 arXiv:https://bmjopen.bmj.com/content/3/8/e002847.full.pdf

[70] Katarzyna Woźnica and Przemysław Biecek. 2020. Does imputation matter? Benchmark for predictive models. arXiv:2007.02837 [stat.ML]

[71] Richard Wu, Aoqian Zhang, Ihab Ilyas, and Theodoros Rekatsinas. 2020. Attention-based Learning for Missing Data Imputation in HoloClean. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. 307–325. https://proceedings.mlsys.org/paper/2020/file/202cb962ac59075b964b07152d234b70-Paper.pdf

[72] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. *ArXiv* abs/1806.02920 (2018).

[73] Shichao Zhang. 2012. Nearest neighbor selection for iteratively kNN imputation. *Journal of Systems and Software* 85, 11 (2012), 2541–2552. https://doi.org/10.1016/j.jss.2012.05.073

[74] Xinmeng Zhang, Chao Yan, Cheng Gao, Bradley A. Malin, and You Chen. 2020. Predicting Missing Values in Medical Data Via XGBoost Regression. *Journal of Healthcare Informatics Research* 4, 4 (01 Dec 2020), 383–394. https://doi.org/10.1007/s41666-020-00077-1

A DATASETS APPENDIX

A.1 Real-World Datasets with native missing values

This Section presents the 25 real-world binary classification datasets with native missing values. See Table 6 for more details.

A.2 Complete datasets for missing data simulation

This Section presents the 10 complete binary classification datasets used for the simulated missing value experiments. Table 7 contains the dataset names and their characteristics.

B MISSING VALUE SIMULATION SETUP APPENDIX

B.1 Datasets selected to determine the percentage of missing values per feature.

Realistic simulation of missing values requires selecting the missingness percentage for each feature, see Section 4.1. We sampled 64 real-world datasets with missing values from OpenML repository. Table 8 describes the datasets' characteristics.

B.2 Determining the average number of features, a missing feature depends on.

In this Section, we present the quantitative results for the experiments regarding the simulation of MAR mechanism presented in 4. Table 9, presents the dataset name, the randomly selected feature with missing values (target) and the result of the feature selection (# features selected). On average a missing feature is dependent on 12 features.

Table 6. Binary Classification Real-World datasets used in the comparative evaluation. The table below contains the Dataset name, id, number of samples, number of features, number of categorical and numeric features, Missingness percentage in the whole dataset, Minority Class with missing values over 1finally the outcome type of each dataset.

Dataset	ID	Samples	Features	#Numerical	#Categorical	Missing %	Imbalance ratio	#Feat with miss>1%	%Missing/Feature
analcatdata_reviewer	1008	379	7	0	7	51.56	0.43	7	51.56
audiology	999	226	69	0	69	2.03	0.25	6	23.23
anneal	989	898	38	16	22	64.98	0.24	29	85.15
autoHorse	840	205	25	17	8	1.11	0.40	4	6.46
braziltourism	957	412	8	7	1	2.91	0.23	2	10.68
bridges	328	107	11	4	7	6.03	0.41	7	9.35
cjs	1024	2796	34	32	2	71.64	0.24	28	86.97
colic	27	368	22	7	15	23.80	0.37	19	27.50
colleges_aaup	897	1161	15	13	2	1.47	0.30	6	3.68
cylinder-bands	6332	540	39	24	15	4.74	0.42	23	7.93
dresses-sales	23381	500	12	1	11	13.92	0.42	5	33.04
eucalyptus	990	736	19	14	5	3.21	0.29	6	9.95
hepatitis	55	155	19	6	13	5.67	0.21	11	9.56
hungarian	231	294	13	12	1	20.46	0.36	5	52.93
kdd_el_nino-small	839	782	8	8	0	7.45	0.35	4	14.90
mushroom	24	8124	22	0	22	1.39	0.48	1	30.53
pbcseq	802	1945	17	13	4	3.43	0.50	6	9.71
primary-tumor	1003	339	17	0	17	3.90	0.25	2	32.74
profb	470	672	9	5	4	19.84	0.33	2	89.29
schizo	466	340	14	12	2	17.52	0.48	11	22.30
sick	38	3772	29	7	22	5.54	0.06	7	22.96
soybean	1023	683	35	0	35	9.78	0.13	32	10.68
stress	42167	199	12	8	4	8.29	0.20	7	14.22
vote	56	435	16	0	16	5.63	0.39	16	5.63
water-treatment	940	527	36	36	0	2.86	0.15	22	4.53

Table 7. Binary complete datasets in which we inject missing values. The table reports the dataset name, ID, the number of samples, number of features, the imbalance ratio, the number of numerical and categorical variables.

Dataset	ID	#Samples	#Features	#Numerical	#Categorical	Minority Class %
Australian	40981	690	14	8	6	0.44
boston	853	506	13	12	1	0.41
churn	40701	5000	20	16	4	0.14
compas-two-years	42193	5278	13	7	6	0.47
image	40592	2000	135	135	0	0.21
page-blocks	1021	5473	10	10	0	0.1
parkinsons	1488	195	22	22	0	0.25
segment	958	2310	19	19	0	0.14
stock	841	950	9	9	0	0.49
zoo	965	101	16	0	16	0.41

C EXPERIMENTAL RESULTS APPENDIX

C.1 Real-World Results

C.1.1 BI+DAE is the highest ranking method, followed by BI+MM. Table 10 shows the quantitative results of the real-world experiments. Specifically, it depicts the number of wins (ties included) for each imputation method, as well as the average difference in AUC from the winning imputation method for each dataset. The table also reports the average AUC and average ranking per method.

Table 8. Datasets used for missing value simulation experimental setup. The table below contains the Dataset name, id, number of samples, number of features, number of categorical and numeric features, Missingness percentage in the whole dataset, Minority Class %, the number of features with missing values over 1%, the missingness percentage over features with missing values and finally the outcome type of each dataset.

Dataset	ID	Samples	Features	#Numerical	#Categorical	Missing %	Minority Class %	#Feat miss>1%	%Missing/Feature	Type
adult	179	48842	14	6	8	0.95	0.24	3	4.41	Binary
albert	41147	425240	78	78	0	13.64	0.50	43	24.73	Binary
analcatdata_reviewer	1008	379	7	0	7	51.56	0.43	7	51.56	Binary
anneal	989	898	38	16	22	64.98	0.24	29	85.15	Binary
aps_failure	41138	76000	170	170	0	8.35	0.02	160	8.83	Binary
ASP-POTASSCO-classification	41705	1294	142	139	3	9.94	0.02	138	10.23	MultiClass
ASP-POTASSCO-regression	41704	14234	142	138	4	9.94	0.00	138	10.23	Regression
audiology	999	226	69	0	69	2.03	0.25	6	23.23	Binary
autoHorse	840	205	25	17	8	1.11	0.40	4	6.46	Binary
braziltourism	957	412	8	7	1	2.91	0.23	2	10.68	Binary
bridges	328	107	11	4	7	6.03	0.41	7	9.35	Binary
Census-Income-KDD	42750	199523	41	13	28	5.08	0.06	7	29.72	Binary
cjs	1024	2796	34	32	2	71.64	0.24	28	86.97	Binary
Code_Smells_Data_Class	43079	86467	66	66	0	49.99	0.00	62	53.20	Regression
colic	27	368	22	7	15	23.80	0.37	19	27.50	Binary
colleges	42727	7063	47	31	16	31.42	0.00	30	49.19	Regression
colleges_aaup	897	1161	15	13	2	1.47	0.30	6	3.68	Binary
colleges_usnews	930	1302	33	32	1	18.22	0.47	25	23.96	Binary
cylinder-bands	6332	540	39	24	15	4.74	0.42	23	7.93	Binary
Domainome	41533	1623	9838	9838	0	82.17	0.35	9688	83.44	Binary
dresses-sales	23381	500	12	1	11	13.92	0.42	5	33.04	Binary
echoMonths	222	130	9	7	2	8.29	0.00	6	12.31	Regression
eucalyptus	990	736	19	14	5	3.21	0.29	6	9.95	Binary
fishcatch	232	158	7	7	0	7.87	0.00	1	55.06	Regression
fps-in-video-games	42737	425833	44	33	11	6.94	0.00	12	25.44	Regression
hepatitis	55	155	19	6	13	5.67	0.21	11	9.56	Binary
house_prices_nominal	42563	1460	79	36	43	6.04	0.00	16	29.74	Regression
hungarian	231	294	13	12	1	20.46	0.36	5	52.93	Binary
ipums_la_97-small	993	7019	60	34	26	11.42	0.04	18	38.06	MultiClass
ipums_la_98-small	381	7485	60	34	26	11.59	0.01	17	40.91	MultiClass
ipums_la_99-small	378	8844	60	34	26	9.71	0.02	18	32.36	MultiClass
jungle_chess_2pcs_endgame_rat_panther	41002	5880	46	18	28	1.30	0.23	6	10.00	MultiClass
KDD98	42343	82318	477	358	119	11.30	0.12	87	61.98	Binary
KDDCup09-Upselling	1112	50000	15000	13391	1609	3.35	0.07	608	82.59	Binary
KDDCup09_churn	42759	50000	230	192	38	69.78	0.07	205	78.28	Binary
kdd_coil_1	567	316	11	8	3	1.61	0.00	3	4.85	Regression
kdd_el_nino-small	839	782	8	8	0	7.45	0.35	4	14.90	Binary
kick	41162	72983	32	17	15	6.39	0.12	5	40.51	Binary
lymphoma_2classes	1101	45	4026	4026	0	3.28	0.49	2116	6.25	Binary
meta	566	528	21	18	3	4.55	0.00	3	31.82	Regression
MiceProtein	40966	1080	81	77	4	1.60	0.10	8	14.69	MultiClass
Midwest_Survey_nominal	42532	2778	27	1	26	1.95	0.03	5	10.51	MultiClass
mlr_ranger_rng	42458	278863	14	8	6	3.56	0.00	1	49.69	Regression
mlr_svm_rng	42456	540576	13	7	6	9.38	0.00	2	60.95	Regression
Moneyball	41021	1232	14	11	3	20.87	0.00	4	73.05	Regression
mushroom	24	8124	22	0	22	1.39	0.48	1	30.53	Binary
NewFuelCar	41506	36203	17	17	0	1.46	0.00	1	24.78	Regression
okcupid-stem	42734	50789	19	3	16	15.97	0.10	12	25.28	MultiClass
pbc	524	418	18	17	1	16.47	0.00	12	24.66	Regression
pbcseq	802	1945	17	13	4	3.43	0.50	6	9.71	Binary
porto-seguro	42206	595212	37	25	12	3.84	0.04	5	28.21	Binary
primary-tumor	1003	339	17	0	17	3.90	0.25	2	32.74	Binary
profb	470	672	9	5	4	19.84	0.33	2	89.29	Binary
rl	41160	31406	22	22	0	10.45	0.10	8	28.71	Binary
road-safety	42803	363243	66	61	5	9.10	0.05	41	14.62	MultiClass
SAT11-HAND-runtime-regression	41980	4440	116	113	3	5.27	0.00	10	61.15	Regression
schizo	466	340	14	12	2	17.52	0.48	11	22.30	Binary
sick	38	3772	29	7	22	5.54	0.06	7	22.96	Binary
soybean	1023	683	35	0	35	9.78	0.13	32	10.68	Binary
speeddating	40536	8378	122	61	61	2.87	0.00	109	3.17	Binary
stress	42167	199	12	8	4	8.29	0.20	7	14.22	Binary
us_crime	315	1994	127	126	1	15.48	0.00	24	81.91	Regression
vote	56	435	16	0	16	5.63	0.39	16	5.63	Binary
water-treatment	940	527	36	36	0	2.86	0.15	22	4.53	Binary

According to the total number of wins, BI+MM and BI+DAE share first place by winning in 9 out of 25 datasets. However, BI+DAE has the least AUC difference from the leading imputation method, followed by BI+MM with a 0.012 AUC difference. BI+MF has 6 wins, while BI+PPCA and BI+SOFT have 4 wins each. Finally, BI+GAIN only has 3 wins.

In summary, BI+MM and BI+DAE have the most wins, but BI+DAE has the highest mean AUC and the lowest average ranking. Additionally, BI+DAE has the least average difference from the leading imputation method, on average.

Table 9. The table summarizes the feature selection experiments for MAR simulation. On average a missing feature depends on 12 other features.

Dataset	Missing Feature (Target)	#Selected features
aps_failure	cn_006	25
colleges_aaup	Average_salary-full_professors	5
colleges_usnews	Out-of-state_tuition	18
dresses-sales	V3	7
eucalyptus	PMCno	6
hepatitis	ALBUMIN	6
hungarian	thalach	3
mushroom	stalk-root	16
pbcseq	presence_of_asictes	7
speeddating	attractive	24

Table 10. Number of Wins and average difference from winner for each imputation method on real datasets. BI+MM and BI+DAE are tied at 9 wins each. BI+DAE scores on average the highest AUC and has the lowest average ranking.

Method	#Wins with ties	Avg. Difference from leader	Mean AUC	Average Ranking
BI+MM	9	0.012	0.883	2.94
BI+DAE	9	0.006	0.889	2.84
BI+MF	6	0.015	0.880	3.50
BI+PPCA	4	0.035	0.860	4.42
BI+SOFT	4	0.014	0.881	3.74
BI+GAIN	3	0.014	0.881	3.56

C.1.2 *BI+MM and BI+DAE score 99.69% of maximum AUC.*

As shown in Table 11, BI+MM scores 99.68% of the maximum AUC. Adding BI+DAE to the imputation set that already contains BI+MM allows us to score over 99% of the maximum AUC. However, the complexity increases by 10 times. Finally, in order to reach 100% of the maximum AUC, all imputation methods need to be included. This results in an increase in the original complexity by a factor of 24.

Table 11. Table denotes the methods that are added to the imputation set, the % of the maximum AUC reached by each set, and the configuration complexity increase for each set. The set containing BI+MM and BI+DAE accounts for 99.69% of the maximum AUC.

Added Method	% of Max AUC	Mult. Config.
BI+MM	98.68	1
BI+DAE	99.69	10
BI+MF	99.93	12
BI+PPCA	99.98	15
BI+GAIN	99.99	21
BI+SOFT	100	24

C.2 Simulation Results

C.2.1 *Increasing missingness leads to predictive performance drop.*

In the Section, we present the mean performance drop in terms of AUC compared to the complete dataset. Table 12 provides an overview of the results for MCAR data (see Table 12a) and MAR data (see Table 12b) at varying missigness percentages. For MCAR data, we inspect a drop of up to 0.024 AUC points at 10% missingness. This drop further increases to 0.048 and 0.143 for 25% and 50% missingness, respectively.

Table 12. The tables present the average AUC loss from the complete datasets by each imputation method when missing data are MCAR (left) and MAR(right).

(a) Average AUC loss for MCAR data.

Method	10%	25%	50%
BI+MM	-0.011	-0.024	-0.078
BI+MF	-0.006	-0.02	-0.064
BI+GAIN	-0.014	-0.026	-0.073
BI+SOFT	-0.019	-0.045	-0.143
BI+PPCA	-0.024	-0.048	-0.113
BI+DAE	-0.011	-0.022	-0.080

(b) Average AUC loss for MAR data.

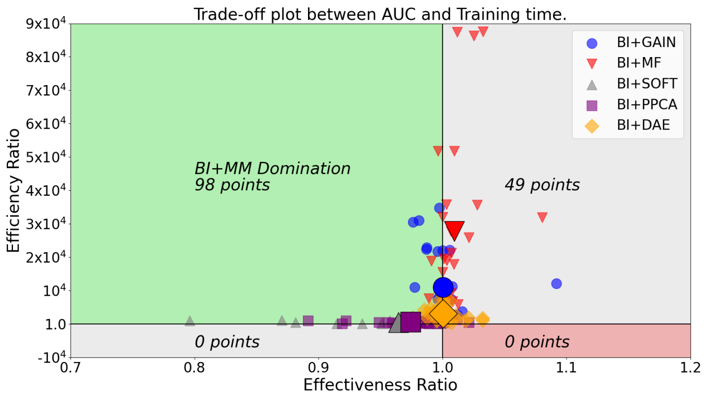
Method	10%	25%	50%
BI+MM	-0.014	-0.029	-0.055
BI+MF	-0.015	-0.029	-0.052
BI+GAIN	-0.016	-0.031	-0.06
BI+SOFT	-0.014	-0.046	-0.107
BI+PPCA	-0.025	-0.047	-0.073
BI+DAE	-0.015	-0.035	-0.055

C.2.2 BI+MF is the best method for MCAR data. Thank you for bringing that to my attention. Here's the corrected text:

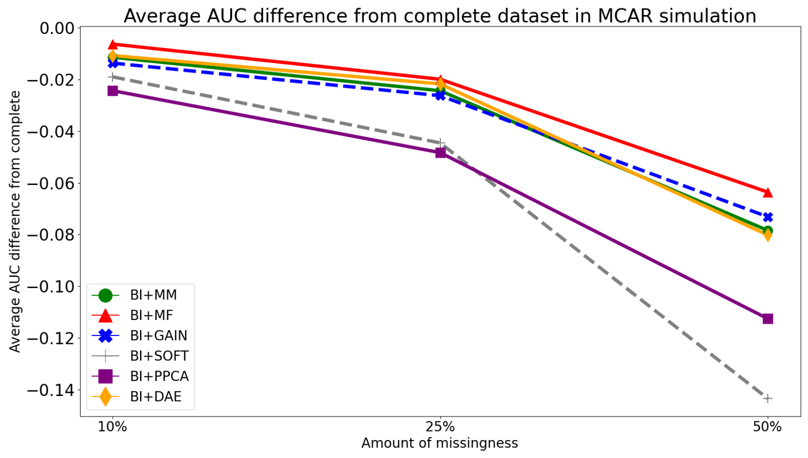
Table 12a summarizes the average AUC loss compared to the complete dataset for each imputation method, at the specified missingness percentage for the MCAR simulation. Starting at 10% missingness, BI+MF is the best method with -0.006 mean AUC loss followed by BI+DAE and BI+MM tied at -0.011. BI+GAIN follows with a -0.014 AUC loss. Finally, BI+PPCA and BI+SOFT have the highest mean AUC loss at -0.024 and -0.019 respectively. When missingness is 25%, BI+MF again performs better than the other methods with -0.020 mean AUC loss. BI+DAE is second with -0.022 and BI+MM third with -0.024. BI+GAIN closely follows at -0.026 mean AUC loss. BI+PPCA and BI+SOFT exhibit a big average AUC drop. Specifically, they have an average AUC loss of -0.048 and -0.045 respectively. At 50% missingness, BI+MF is again first followed by BI+GAIN, BI+MM, and BI+DAE in the order mentioned. BI+PPCA and BI+SOFT again lose a lot of AUC, on average. The difference between the AUC loss of BI+MF and BI+MM, BI+DAE increases at 50% missingness. Additionally, BI+GAIN at 50

C.2.3 BI+MF is the best method for MAR data. Table 12b provides an overview of the average AUC loss for each imputation method at the specified levels of missingness in the MAR simulation. At 10% missingness, the best methods are BI+MM and BI+SOFT, with a mean AUC loss of -0.014 each. BI+MF and BI+DAE follow closely with -0.015 mean AUC loss, while BI+GAIN and BI+PPCA exhibit a slightly higher mean AUC loss of -0.016 and -0.025, respectively. When missingness is increased to 25%, BI+MM and BI+MF are again the best methods on average, with a mean AUC loss of -0.029. BI+GAIN is the third-best method with a mean AUC loss of -0.031, followed by BI+DAE at -0.035. BI+SOFT and BI+PPCA have the highest mean AUC loss, with -0.046 and -0.047, respectively. At 50% missingness, BI+MF achieves the lowest average AUC loss of -0.052, followed by BI+DAE and BI+MM with -0.055. BI+GAIN exhibits a mean AUC loss of -0.06, while BI+PPCA and BI+SOFT have the highest mean AUC loss, with -0.073 and -0.0107, respectively.

C.2.4 MCAR efficiency vs effectiveness trade-off. Regarding MCAR data, it is worth noting that BI+MM dominates in 98 out of 147 pairs in terms of both effectiveness and efficiency. BI+MM is only dominated in terms of predictive performance in the rest of the cases (49 points), while still dominating in efficiency. BI+MM is never dominated in both metrics or inefficiency, which is expected. Although BI+GAIN, BI+MF, and BI+DAE have a higher mean AUC than



(a) MCAR: Illustrates the efficiency for each imputation method against BI+MM versus the effectiveness in terms of AUC. 98 out of 147 points are in favor of BI+MM in both efficiency and effectiveness.



(b) MCAR data: Average AUC difference of each imputation method from the complete dataset. BI+MF is on average the best method for MCAR data followed by BI+MM and BI+DAE at every percentage.

Fig. 10. Figure (a) denotes the trade-off for mcar data. Figure (b) illustrates the average AUC for each imputation method at various missingness percentages.

BI+MM, their training time is on average a thousand times slower. On the other hand, BI+SOFT and BI+PPCA have slower training time and lower AUC compared to BI+MM, on average. For more details, please refer to Figure 10a.

C.2.5 MCAR Minimal-Size Subset. The minimal-size subset of algorithms with close-to-optimal performance for MCAR missing data is $BI + MM, BI + MF$. We used the simple greedy algorithm introduced in Section 5.4. As illustrated in Figure 11, this subset achieves over 99% of the maximum achievable AUC for MCAR data. Detailed quantitative results are presented in Table 13a.

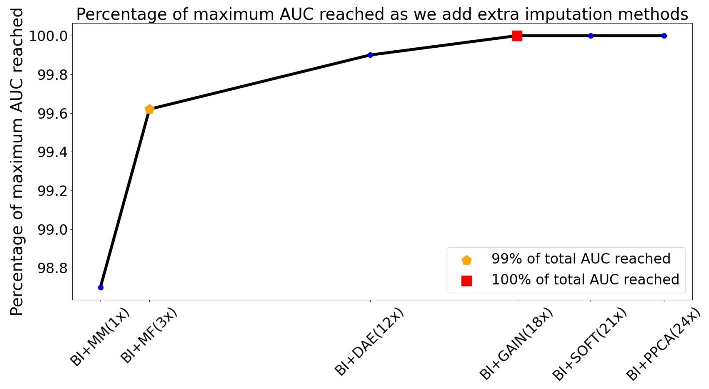


Fig. 11. Illustrates the percentage of maximum AUC achieved by each set of imputation methods versus the increased complexity of the pipelines.On x-axis we denote the added method as well as the additional complexity to the configuration space. The addition of BI+MF to the set increases the complexity by 3x compared to the original complexity. BI+MF and BI+MM allow us to recover 99.62% of the maximum AUC for MCAR data.

(a) MCAR: Table denotes the methods that are added to the imputation set, the % of the maximum AUC reached by each set and the configuration complexity increase for each set.

Added Method	% of Max AUC	Mult. Config.
BI+MM	98.7	1
BI+MF	99.62	3
BI+DAE	99.9	12
BI+GAIN	100	18
BI+SOFT	100	21
BI+PPCA	100	24

(b) MAR: Table denotes the methods that are added to the imputation set, the % of the maximum AUC reached by each set and the configuration complexity increase for each set.

Added Method	% of Max AUC	Mult. Config.
BI+MM	98.99	1
BI+MF	99.43	3
BI+DAE	99.67	12
BI+GAIN	99.84	18
BI+SOFT	99.95	21
BI+PPCA	100	24

C.3 Real-World: Tables of Results

In this section we include the quantitative results of the real-world experiments. Table 14 contains the results of feature selection enforced pipelines. Table 15 the results of the pipelines where there is no feature selection step. Finally, table 16 contains the results when optimizing with and without feature selection.

C.4 MCAR: Tables of Results

This section presents the results for missing completely at random (MCAR) data under varying levels of missingness. Tables 17, 18, and 19 show the results for 10% missingness, while Tables 20, 21, and 22 display the results for 25% missingness. Finally, Tables 23, 24, and 25 present the results for 50% missingness. For each level of missingness, the tables are divided based on the pipeline used, which includes enforced feature selections, excluding feature selection, and optimizing overall configurations (with and without feature selection).

Table 14. Real-World results when feature selection is enforced in the pipeline

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
analcatdata_reviewer	0.652	0.585	0.585	0.5	0.597	0.561	0.595	0.602	0.602	0.558	0.558	0.585	0.585
anneal	0.986	0.883	0.988	0.761	0.97	0.916	0.973	0.987	0.967	0.995	0.986	0.975	0.968
audiology	0.98	0.986	0.986	0.98	0.98	0.98	0.974	0.98	0.98	0.98	0.98	0.98	0.98
autoHorse	0.966	0.967	0.967	0.966	0.966	0.966	0.966	0.966	0.966	0.901	0.996	0.966	0.966
braziltourism	0.643	0.634	0.634	0.64	0.64	0.643	0.634	0.64	0.64	0.725	0.725	0.643	0.643
bridges	0.892	0.844	0.844	0.853	0.857	0.891	0.849	0.891	0.891	0.892	0.885	0.88	0.847
cjs	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.987	0.997	1.0	1.0
colic	0.829	0.829	0.829	0.837	0.855	0.845	0.839	0.836	0.836	0.839	0.839	0.83	0.83
colleges_aaup	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.996	0.996	0.999	0.999
cylinder-bands	0.834	0.808	0.785	0.81	0.788	0.808	0.797	0.808	0.798	0.723	0.723	0.832	0.826
dresses-sales	0.56	0.562	0.562	0.564	0.561	0.56	0.552	0.565	0.565	0.5	0.549	0.562	0.562
eucalyptus	0.834	0.833	0.833	0.821	0.82	0.823	0.835	0.842	0.817	0.75	0.816	0.807	0.834
hepatitis	0.826	0.748	0.748	0.834	0.83	0.803	0.807	0.673	0.673	0.799	0.799	0.826	0.826
hungarian	0.896	0.899	0.899	0.883	0.884	0.893	0.867	0.871	0.871	0.897	0.897	0.875	0.875
kdd_el_nino-small	0.989	0.983	0.983	0.98	0.981	0.988	0.987	0.983	0.981	0.95	0.926	0.983	0.986
mushroom	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pbcseq	0.849	0.849	0.849	0.851	0.852	0.84	0.849	0.836	0.831	0.849	0.849	0.846	0.843
primary-tumor	0.882	0.875	0.887	0.855	0.875	0.846	0.874	0.829	0.882	0.829	0.864	0.786	0.887
profb	0.642	0.642	0.642	0.642	0.642	0.642	0.642	0.642	0.642	0.631	0.631	0.642	0.642
schizo	0.555	0.526	0.526	0.556	0.557	0.534	0.515	0.556	0.556	0.543	0.543	0.623	0.623
sick	0.992	0.984	0.984	0.992	0.993	0.993	0.992	0.977	0.969	0.991	0.991	0.992	0.992
soybean	0.991	0.981	0.983	0.983	0.992	0.991	0.981	0.985	0.985	0.973	0.973	0.991	0.991
stress	0.932	0.916	0.916	0.932	0.932	0.932	0.932	0.932	0.932	0.933	0.933	0.932	0.932
vote	0.991	0.983	0.985	0.992	0.995	0.986	0.99	0.991	0.991	0.989	0.991	0.978	0.986
water-treatment	0.988	0.916	0.988	0.986	0.987	0.958	0.988	0.943	0.943	0.5	0.5	0.962	0.979

Table 15. Real-World results when feature selection is excluded from the pipeline

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
analcatdata_reviewer	0.59	0.661	0.668	0.599	0.646	0.602	0.61	0.597	0.659	0.606	0.656	0.661	0.668
anneal	0.996	0.881	0.996	0.938	0.97	0.931	0.983	0.972	0.991	0.996	0.996	0.982	0.982
audiology	0.993	0.998	0.998	0.992	0.993	0.994	0.991	0.992	0.998	0.995	0.989	0.993	0.995
autoHorse	0.988	0.99	0.989	0.983	0.988	0.982	0.988	0.981	0.99	0.989	0.993	0.976	0.976
braziltourism	0.669	0.616	0.727	0.716	0.725	0.721	0.709	0.709	0.725	0.669	0.668	0.731	0.715
bridges	0.912	0.909	0.911	0.902	0.901	0.882	0.916	0.902	0.889	0.901	0.916	0.915	0.912
cjs	1.0	1.0	1.0	0.994	1.0	0.985	0.998	0.996	0.991	0.987	0.99	0.999	0.999
colic	0.856	0.839	0.838	0.853	0.863	0.845	0.872	0.842	0.865	0.848	0.858	0.852	0.856
colleges_aaup	0.998	0.999	0.999	0.997	0.999	0.997	0.997	0.998	0.997	0.998	0.998	0.998	0.997
cylinder-bands	0.848	0.82	0.819	0.826	0.832	0.826	0.828	0.815	0.821	0.807	0.824	0.848	0.858
dresses-sales	0.631	0.619	0.605	0.6	0.597	0.569	0.583	0.597	0.601	0.545	0.539	0.631	0.62
eucalyptus	0.849	0.777	0.777	0.778	0.778	0.778	0.777	0.779	0.844	0.82	0.824	0.849	0.855
hepatitis	0.848	0.826	0.848	0.866	0.866	0.844	0.85	0.869	0.864	0.876	0.866	0.852	0.869
hungarian	0.918	0.918	0.915	0.901	0.901	0.917	0.915	0.881	0.895	0.895	0.898	0.914	0.912
kdd_el_nino-small	0.987	0.987	0.989	0.984	0.985	0.988	0.988	0.985	0.988	0.98	0.985	0.986	0.987
mushroom	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pbcseq	0.849	0.849	0.85	0.857	0.844	0.856	0.848	0.846	0.845	0.841	0.838	0.848	0.842
primary-tumor	0.892	0.88	0.892	0.875	0.875	0.871	0.88	0.877	0.889	0.861	0.87	0.88	0.892
profb	0.698	0.695	0.696	0.691	0.693	0.695	0.692	0.695	0.696	0.579	0.58	0.693	0.695
schizo	0.684	0.719	0.684	0.764	0.765	0.717	0.74	0.76	0.76	0.56	0.559	0.794	0.805
sick	0.994	0.99	0.989	0.994	0.994	0.992	0.995	0.986	0.988	0.992	0.989	0.993	0.992
soybean	0.993	0.991	0.994	0.989	0.986	0.989	0.992	0.987	0.991	0.991	0.985	0.989	0.993
stress	0.909	0.902	0.904	0.899	0.903	0.906	0.904	0.902	0.901	0.948	0.946	0.909	0.909
vote	0.995	0.992	0.991	0.994	0.991	0.994	0.99	0.995	0.991	0.995	0.992	0.992	0.992
water-treatment	0.988	0.988	0.988	0.988	0.987	0.988	0.988	0.954	0.986	0.788	0.774	0.98	0.981

C.5 MAR: Tables of Results

This section presents the results for missing at random (MAR) data under varying levels of missingness. Tables 26, 27, and 28 show the results for 10% missingness, while Tables 29, 30, and 31 display the results for 25% missingness. Finally, Tables 32, 33, and 34 present the results for 50% missingness. These tables are divided based on the pipeline

Table 16. Real-World results overall

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
analcatdata_reviewer	0.607	0.661	0.668	0.605	0.643	0.607	0.63	0.597	0.659	0.606	0.656	0.661	0.668
anneal	0.996	0.883	0.996	0.943	0.969	0.896	0.991	0.972	0.991	0.996	0.996	0.975	0.982
audiology	0.993	0.998	0.998	0.98	0.981	0.993	0.992	0.992	0.98	0.995	0.989	0.993	0.98
autoHorse	0.988	0.99	0.989	0.966	0.966	0.987	0.988	0.981	0.99	0.989	0.993	0.966	0.966
braziltourism	0.643	0.616	0.727	0.643	0.64	0.632	0.64	0.709	0.64	0.669	0.668	0.731	0.715
bridges	0.912	0.909	0.911	0.902	0.909	0.905	0.909	0.902	0.889	0.901	0.916	0.915	0.912
cjs	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.987	0.997	1.0	1.0
colic	0.86	0.829	0.829	0.849	0.881	0.846	0.862	0.836	0.865	0.839	0.858	0.83	0.83
colleges_aaup	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.998	0.998	0.999	0.999
cylinder-bands	0.848	0.82	0.819	0.832	0.831	0.813	0.816	0.815	0.821	0.807	0.824	0.848	0.858
dresses-sales	0.571	0.619	0.562	0.564	0.561	0.567	0.552	0.565	0.565	0.545	0.539	0.631	0.562
eucalyptus	0.849	0.833	0.833	0.778	0.778	0.832	0.836	0.842	0.817	0.82	0.824	0.849	0.855
hepatitis	0.848	0.826	0.848	0.867	0.858	0.858	0.866	0.869	0.864	0.799	0.799	0.852	0.869
hungarian	0.918	0.918	0.915	0.887	0.888	0.913	0.918	0.871	0.871	0.895	0.897	0.914	0.912
kdd_el_nino-small	0.987	0.987	0.989	0.982	0.986	0.988	0.987	0.985	0.988	0.98	0.985	0.986	0.987
mushroom	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pbcseq	0.849	0.849	0.85	0.851	0.85	0.851	0.849	0.846	0.845	0.841	0.838	0.848	0.842
primary-tumor	0.892	0.88	0.892	0.867	0.892	0.886	0.87	0.877	0.889	0.861	0.87	0.88	0.892
profb	0.689	0.695	0.696	0.692	0.694	0.692	0.686	0.695	0.696	0.631	0.631	0.693	0.695
schizo	0.684	0.719	0.684	0.781	0.772	0.736	0.751	0.76	0.76	0.543	0.543	0.794	0.805
sick	0.992	0.99	0.989	0.994	0.994	0.992	0.993	0.986	0.988	0.992	0.989	0.993	0.992
soybean	0.993	0.981	0.994	0.985	0.991	0.99	0.991	0.987	0.991	0.991	0.985	0.989	0.993
stress	0.932	0.916	0.916	0.932	0.932	0.932	0.932	0.932	0.932	0.933	0.933	0.909	0.932
vote	0.991	0.992	0.991	0.992	0.991	0.995	0.992	0.991	0.991	0.989	0.991	0.992	0.992
water-treatment	0.988	0.988	0.988	0.987	0.987	0.988	0.988	0.954	0.986	0.788	0.774	0.98	0.981

Table 17. MCAR 10% Results when feature selection is enforced

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.906	0.902	0.905	0.922	0.914	0.913	0.919	0.885	0.915	0.89	0.897	0.906	0.906
boston	0.933	0.933	0.933	0.926	0.926	0.924	0.924	0.912	0.917	0.89	0.89	0.926	0.926
churn	0.923	0.914	0.914	0.92	0.916	0.907	0.916	0.887	0.883	0.884	0.885	0.906	0.906
compas-two-years	0.697	0.704	0.704	0.698	0.692	0.699	0.705	0.693	0.693	0.688	0.688	0.697	0.697
image	0.878	0.87	0.845	-	-	0.883	0.875	0.85	0.85	0.845	0.849	0.877	0.878
page-blocks	0.989	0.99	0.989	0.99	0.989	0.988	0.99	0.983	0.985	0.969	0.969	0.988	0.988
parkinsons	0.849	0.85	0.846	0.871	0.871	0.849	0.849	0.85	0.845	0.848	0.861	0.868	0.866
segment	0.998	0.999	0.999	1.0	1.0	0.999	0.999	0.996	0.997	0.978	0.977	0.999	0.999
stock	0.993	0.99	0.99	0.993	0.993	0.984	0.987	0.975	0.975	0.954	0.954	0.99	0.99
zoo	0.898	0.993	0.993	0.895	0.895	0.929	0.929	0.986	0.986	0.898	0.895	0.995	0.995

Table 18. MCAR 10% Results when feature selection is excluded

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.903	0.916	0.911	0.917	0.915	0.911	0.904	0.886	0.896	0.892	0.896	0.915	0.91
boston	0.936	0.931	0.928	0.94	0.935	0.927	0.931	0.933	0.908	0.913	0.921	0.928	0.926
churn	0.906	0.909	0.912	0.914	0.916	0.91	0.916	0.904	0.91	0.886	0.886	0.906	0.909
compas-two-years	0.692	0.702	0.698	0.703	0.701	0.701	0.698	0.702	0.702	0.703	0.699	0.692	0.692
image	0.892	0.885	0.884	-	-	0.881	0.878	0.877	0.877	0.863	0.859	0.892	0.887
page-blocks	0.99	0.988	0.988	0.99	0.99	0.989	0.988	0.983	0.982	0.973	0.974	0.987	0.987
parkinsons	0.896	0.896	0.896	0.935	0.923	0.896	0.895	0.898	0.892	0.899	0.907	0.919	0.914
segment	1.0	0.999	1.0	1.0	1.0	0.999	1.0	0.998	0.999	0.985	0.986	1.0	0.999
stock	0.993	0.989	0.99	0.992	0.993	0.99	0.989	0.979	0.98	0.969	0.97	0.991	0.991
zoo	1.0	0.979	0.979	0.992	0.998	1.0	1.0	0.973	0.989	0.897	0.994	0.903	1.0

used, which includes enforced feature selections, excluding feature selection, and optimizing overall configurations (with and without feature selection).

Table 19. MCAR 10% Results Optimizing Overall (with and without feature selection)

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.906	0.902	0.911	0.922	0.914	0.909	0.908	0.885	0.896	0.892	0.896	0.906	0.906
boston	0.93	0.933	0.933	0.938	0.941	0.922	0.921	0.912	0.908	0.913	0.921	0.926	0.926
churn	0.911	0.909	0.912	0.917	0.916	0.908	0.919	0.904	0.91	0.884	0.886	0.906	0.909
compas-two-years	0.692	0.702	0.704	0.703	0.696	0.7	0.699	0.693	0.702	0.688	0.688	0.692	0.697
image	0.892	0.885	0.884	-	-	0.882	0.883	0.877	0.877	0.863	0.859	0.892	0.887
page-blocks	0.989	0.988	0.989	0.99	0.99	0.987	0.989	0.983	0.982	0.973	0.974	0.988	0.987
parkinsons	0.896	0.896	0.896	0.932	0.925	0.907	0.894	0.898	0.892	0.899	0.907	0.919	0.914
segment	1.0	0.999	1.0	1.0	1.0	1.0	1.0	0.998	0.999	0.985	0.986	1.0	0.999
stock	0.993	0.99	0.99	0.993	0.993	0.986	0.986	0.979	0.98	0.969	0.97	0.99	0.99
zoo	0.995	0.979	0.993	0.994	1.0	0.929	0.989	0.986	0.989	0.897	0.994	0.903	1.0

Table 20. MCAR 25% Results when feature selection is enforced

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.877	0.876	0.877	0.896	0.882	0.872	0.88	0.845	0.864	0.831	0.832	0.887	0.887
boston	0.912	0.925	0.917	0.916	0.913	0.912	0.917	0.878	0.878	0.911	0.911	0.907	0.907
churn	0.868	0.866	0.87	0.874	0.871	0.866	0.874	0.834	0.819	0.85	0.849	0.862	0.86
compas-two-years	0.67	0.683	0.676	0.69	0.691	0.687	0.656	0.642	0.645	0.678	0.678	0.67	0.675
image	0.878	0.801	0.801	-	-	0.839	0.845	0.817	0.817	0.851	0.845	0.878	0.878
page-blocks	0.985	0.983	0.982	0.986	0.985	0.982	0.981	0.966	0.964	0.912	0.912	0.983	0.979
parkinsons	0.933	0.793	0.832	0.847	0.85	0.806	0.838	0.78	0.808	0.833	0.833	0.849	0.933
segment	1.0	0.999	0.999	0.999	1.0	0.999	0.999	0.978	0.985	0.94	0.941	0.998	0.998
stock	0.992	0.976	0.976	0.988	0.99	0.979	0.98	0.937	0.937	0.93	0.93	0.983	0.981
zoo	1.0	0.973	0.974	1.0	1.0	0.973	0.997	0.998	0.998	1.0	1.0	0.99	0.994

Table 21. MCAR 25% Results when feature selection is excluded

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.885	0.879	0.884	0.898	0.879	0.881	0.895	0.865	0.875	0.842	0.824	0.882	0.881
boston	0.926	0.919	0.906	0.927	0.917	0.916	0.907	0.888	0.887	0.919	0.909	0.906	0.903
churn	0.866	0.864	0.87	0.867	0.865	0.868	0.869	0.847	0.851	0.856	0.859	0.866	0.866
compas-two-years	0.67	0.685	0.685	0.69	0.684	0.68	0.673	0.664	0.663	0.682	0.681	0.677	0.674
image	0.88	0.868	0.866	-	-	0.862	0.864	0.837	0.844	0.864	0.856	0.88	0.882
page-blocks	0.983	0.982	0.983	0.983	0.985	0.983	0.983	0.966	0.968	0.961	0.966	0.984	0.982
parkinsons	0.918	0.893	0.904	0.915	0.919	0.891	0.899	0.86	0.873	0.837	0.887	0.918	0.93
segment	1.0	0.999	0.999	1.0	1.0	0.999	0.998	0.994	0.99	0.962	0.967	0.998	0.999
stock	0.992	0.983	0.981	0.99	0.992	0.981	0.982	0.958	0.959	0.938	0.939	0.983	0.981
zoo	0.997	0.998	1.0	1.0	1.0	0.99	1.0	1.0	0.989	0.938	1.0	1.0	0.995

Table 22. MCAR 25% Results Optimizing Overall (with and without feature selection)

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.877	0.876	0.877	0.891	0.883	0.893	0.884	0.865	0.864	0.842	0.832	0.887	0.887
boston	0.912	0.925	0.917	0.917	0.914	0.926	0.923	0.878	0.878	0.919	0.909	0.907	0.907
churn	0.879	0.866	0.87	0.872	0.876	0.862	0.867	0.847	0.851	0.856	0.859	0.866	0.866
compas-two-years	0.668	0.685	0.685	0.69	0.69	0.674	0.678	0.664	0.663	0.682	0.681	0.67	0.675
image	0.88	0.868	0.866	-	-	0.862	0.866	0.837	0.844	0.864	0.856	0.88	0.882
page-blocks	0.985	0.983	0.982	0.983	0.986	0.983	0.983	0.966	0.968	0.961	0.966	0.984	0.982
parkinsons	0.918	0.893	0.904	0.919	0.914	0.901	0.882	0.86	0.873	0.833	0.833	0.918	0.933
segment	1.0	0.999	0.999	1.0	1.0	0.999	0.999	0.994	0.99	0.962	0.967	0.998	0.999
stock	0.991	0.983	0.981	0.991	0.99	0.977	0.98	0.958	0.959	0.938	0.939	0.983	0.981
zoo	1.0	0.973	1.0	1.0	1.0	0.998	1.0	0.998	0.989	1.0	1.0	1.0	0.995

Table 23. MCAR 50% Results when feature selection is enforced

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.864	0.864	0.864	0.845	0.855	0.846	0.856	0.806	0.825	0.856	0.827	0.865	0.868
boston	0.829	0.863	0.863	0.893	0.868	0.839	0.845	0.754	0.754	0.862	0.836	0.854	0.851
churn	0.776	0.785	0.785	0.777	0.787	0.76	0.79	0.733	0.739	0.768	0.762	0.777	0.777
compas-two-years	0.646	0.626	0.626	0.647	0.639	0.635	0.642	0.594	0.604	0.632	0.632	0.633	0.633
image	0.823	0.778	0.754	-	-	0.781	0.754	0.681	0.671	0.797	0.817	0.823	0.823
page-blocks	0.965	0.943	0.934	0.963	0.963	0.952	0.956	0.896	0.879	0.791	0.791	0.955	0.934
parkinsons	0.795	0.731	0.704	0.817	0.842	0.747	0.693	0.7	0.652	0.812	0.795	0.816	0.809
segment	0.994	0.99	0.992	0.997	0.986	0.986	0.992	0.864	0.909	0.852	0.86	0.992	0.991
stock	0.954	0.903	0.903	0.972	0.949	0.949	0.918	0.784	0.784	0.8	0.8	0.935	0.923
zoo	0.818	0.965	0.918	0.925	0.81	0.86	0.901	0.921	0.821	0.833	0.818	0.894	0.699

Table 24. MCAR 50% Results when feature selection is excluded

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.859	0.843	0.859	0.851	0.858	0.852	0.836	0.792	0.821	0.853	0.838	0.86	0.866
boston	0.86	0.853	0.831	0.885	0.884	0.848	0.851	0.765	0.761	0.86	0.808	0.839	0.842
churn	0.78	0.789	0.79	0.785	0.789	0.787	0.787	0.772	0.781	0.776	0.784	0.784	0.785
compas-two-years	0.65	0.648	0.646	0.642	0.639	0.646	0.639	0.611	0.614	0.638	0.639	0.642	0.643
image	0.843	0.826	0.82	-	-	0.841	0.824	0.735	0.75	0.845	0.838	0.843	0.847
page-blocks	0.965	0.96	0.956	0.969	0.967	0.96	0.958	0.899	0.91	0.906	0.917	0.957	0.958
parkinsons	0.819	0.842	0.819	0.839	0.869	0.84	0.843	0.753	0.677	0.821	0.814	0.847	0.822
segment	0.996	0.995	0.995	0.995	0.998	0.992	0.994	0.923	0.93	0.912	0.914	0.993	0.992
stock	0.97	0.948	0.944	0.965	0.971	0.92	0.938	0.809	0.822	0.858	0.842	0.935	0.95
zoo	0.916	0.97	0.848	0.938	0.946	0.902	0.902	0.841	0.844	0.821	0.819	0.884	0.829

Table 25. MCAR 50% Results Optimizing Overall (with and without feature selection)

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.859	0.843	0.859	0.852	0.852	0.854	0.865	0.806	0.825	0.853	0.838	0.865	0.868
boston	0.833	0.853	0.863	0.884	0.881	0.853	0.844	0.754	0.761	0.86	0.836	0.854	0.842
churn	0.78	0.789	0.79	0.79	0.789	0.789	0.799	0.772	0.781	0.776	0.762	0.784	0.785
compas-two-years	0.632	0.626	0.646	0.651	0.647	0.649	0.646	0.594	0.614	0.632	0.639	0.633	0.643
image	0.843	0.826	0.82	-	-	0.82	0.833	0.735	0.75	0.845	0.838	0.843	0.847
page-blocks	0.966	0.96	0.956	0.967	0.968	0.959	0.955	0.899	0.91	0.906	0.917	0.957	0.958
parkinsons	0.795	0.842	0.819	0.82	0.84	0.837	0.803	0.753	0.652	0.812	0.795	0.847	0.809
segment	0.997	0.995	0.995	0.997	0.997	0.994	0.991	0.923	0.93	0.912	0.914	0.993	0.992
stock	0.967	0.948	0.944	0.965	0.971	0.933	0.932	0.809	0.822	0.858	0.842	0.935	0.95
zoo	0.916	0.97	0.848	0.93	0.916	0.854	0.925	0.841	0.844	0.833	0.818	0.884	0.829

Table 26. MAR 10% Results when feature selection is enforced

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.911	0.899	0.907	0.893	0.895	0.905	0.907	0.88	0.886	0.873	0.873	0.895	0.911
boston	0.935	0.94	0.94	0.926	0.926	0.928	0.918	0.918	0.918	0.926	0.926	0.933	0.94
churn	0.896	0.893	0.897	0.899	0.896	0.896	0.897	0.89	0.881	0.871	0.873	0.896	0.896
compas-two-years	0.694	0.704	0.704	0.694	0.71	0.707	0.702	0.687	0.689	0.705	0.702	0.706	0.693
image	0.872	0.855	0.858	-	-	0.868	0.857	0.873	0.873	0.82	0.82	0.872	0.868
page-blocks	0.99	0.987	0.988	0.989	0.988	0.988	0.988	0.985	0.986	0.976	0.954	0.987	0.987
parkinsons	0.87	0.873	0.873	0.87	0.871	0.875	0.876	0.874	0.874	0.889	0.801	0.87	0.87
segment	0.998	0.999	0.999	1.0	1.0	0.999	1.0	0.999	0.998	0.96	0.954	0.998	0.998
stock	0.995	0.993	0.985	0.995	0.995	0.991	0.992	0.991	0.987	0.965	0.965	0.991	0.991
zoo	0.993	1.0	1.0	1.0	1.0	0.824	1.0	1.0	1.0	1.0	1.0	0.987	0.993

Table 27. MAR 10% Results when feature selection is excluded

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.907	0.921	0.912	0.91	0.915	0.901	0.918	0.912	0.904	0.886	0.89	0.919	0.899
boston	0.946	0.936	0.932	0.947	0.943	0.938	0.939	0.936	0.929	0.93	0.923	0.94	0.932
churn	0.899	0.894	0.896	0.899	0.902	0.898	0.902	0.891	0.894	0.873	0.875	0.902	0.898
compas-two-years	0.712	0.707	0.705	0.711	0.706	0.703	0.7	0.699	0.703	0.703	0.697	0.709	0.702
image	0.885	0.878	0.879	-	-	0.879	0.888	0.875	0.875	0.862	0.865	0.882	0.885
page-blocks	0.99	0.988	0.988	0.989	0.99	0.988	0.988	0.985	0.985	0.982	0.983	0.987	0.988
parkinsons	0.93	0.908	0.911	0.923	0.926	0.925	0.912	0.918	0.918	0.93	0.894	0.917	0.912
segment	0.999	0.999	1.0	1.0	1.0	1.0	1.0	0.999	1.0	0.968	0.97	1.0	0.999
stock	0.995	0.993	0.994	0.995	0.995	0.993	0.995	0.991	0.989	0.98	0.981	0.994	0.994
zoo	0.998	0.992	1.0	1.0	1.0	1.0	0.997	1.0	0.997	0.99	1.0	0.983	1.0

Table 28. MAR 10% Results Optimizing Overall (with and without feature selection)

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.907	0.899	0.907	0.907	0.915	0.918	0.912	0.912	0.904	0.886	0.89	0.895	0.911
boston	0.942	0.94	0.94	0.926	0.921	0.942	0.922	0.918	0.918	0.93	0.923	0.933	0.94
churn	0.896	0.893	0.896	0.899	0.897	0.898	0.9	0.891	0.894	0.873	0.875	0.896	0.896
compas-two-years	0.709	0.704	0.704	0.711	0.71	0.7	0.699	0.699	0.703	0.703	0.697	0.706	0.702
image	0.885	0.878	0.879	-	-	0.875	0.879	0.875	0.875	0.862	0.865	0.882	0.885
page-blocks	0.988	0.988	0.988	0.988	0.988	0.987	0.987	0.985	0.985	0.982	0.983	0.987	0.988
parkinsons	0.87	0.873	0.873	0.936	0.87	0.905	0.873	0.918	0.918	0.93	0.894	0.87	0.87
segment	0.998	0.999	1.0	1.0	1.0	0.999	1.0	0.999	1.0	0.968	0.97	0.998	0.998
stock	0.995	0.993	0.994	0.995	0.994	0.993	0.994	0.991	0.989	0.98	0.981	0.994	0.994
zoo	1.0	0.992	1.0	1.0	1.0	0.997	1.0	1.0	0.997	0.99	1.0	0.983	0.993

Table 29. MAR 25% Results when feature selection is enforced

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.905	0.901	0.905	0.888	0.886	0.883	0.89	0.864	0.87	0.877	0.869	0.909	0.909
boston	0.908	0.907	0.907	0.903	0.901	0.904	0.908	0.841	0.844	0.887	0.887	0.891	0.891
churn	0.837	0.841	0.841	0.847	0.841	0.844	0.839	0.804	0.811	0.808	0.816	0.837	0.837
compas-two-years	0.694	0.701	0.697	0.688	0.701	0.688	0.696	0.685	0.7	0.685	0.681	0.682	0.697
image	0.871	0.806	0.806	-	-	0.798	0.84	0.835	0.839	0.826	0.823	0.871	0.871
page-blocks	0.982	0.976	0.979	0.982	0.984	0.977	0.978	0.95	0.951	0.932	0.94	0.979	0.98
parkinsons	0.87	0.889	0.876	0.872	0.87	0.889	0.864	0.803	0.803	0.844	0.829	0.849	0.881
segment	1.0	0.994	0.997	1.0	1.0	0.998	0.997	0.974	0.961	0.941	0.963	0.998	0.999
stock	0.981	0.979	0.968	0.98	0.98	0.975	0.977	0.882	0.908	0.886	0.903	0.974	0.974
zoo	0.99	0.991	0.984	1.0	0.99	0.986	0.967	0.973	0.989	0.96	0.99	0.923	0.925

Table 30. MAR 25% Results when feature selection is excluded

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.898	0.903	0.898	0.887	0.888	0.887	0.896	0.884	0.888	0.87	0.875	0.908	0.902
boston	0.903	0.894	0.891	0.903	0.905	0.9	0.905	0.858	0.852	0.895	0.887	0.901	0.89
churn	0.843	0.833	0.836	0.846	0.842	0.843	0.831	0.84	0.827	0.817	0.814	0.832	0.843
compas-two-years	0.692	0.701	0.695	0.693	0.695	0.69	0.69	0.673	0.692	0.692	0.697	0.694	0.691
image	0.878	0.871	0.875	-	-	0.871	0.865	0.86	0.862	0.869	0.86	0.878	0.884
page-blocks	0.982	0.976	0.98	0.982	0.984	0.975	0.978	0.95	0.956	0.866	0.947	0.978	0.978
parkinsons	0.886	0.904	0.892	0.909	0.918	0.898	0.892	0.883	0.871	0.886	0.883	0.905	0.905
segment	1.0	0.999	0.999	1.0	1.0	0.997	0.998	0.989	0.995	0.972	0.975	0.999	0.999
stock	0.983	0.979	0.978	0.981	0.986	0.973	0.977	0.887	0.933	0.931	0.942	0.976	0.975
zoo	1.0	0.99	0.989	0.973	1.0	0.998	1.0	0.997	0.998	0.986	0.997	0.952	0.99

Table 31. MAR 25% Results Optimizing Overall (with and without feature selection)

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.905	0.903	0.905	0.886	0.884	0.88	0.898	0.864	0.87	0.877	0.869	0.909	0.909
boston	0.91	0.894	0.891	0.901	0.91	0.914	0.9	0.858	0.852	0.895	0.887	0.901	0.89
churn	0.837	0.833	0.836	0.845	0.846	0.847	0.841	0.84	0.827	0.808	0.814	0.837	0.843
compas-two-years	0.7	0.701	0.695	0.695	0.693	0.69	0.685	0.673	0.7	0.685	0.681	0.682	0.691
image	0.878	0.871	0.875	-	-	0.879	0.869	0.86	0.862	0.869	0.86	0.878	0.884
page-blocks	0.982	0.976	0.979	0.984	0.982	0.979	0.976	0.95	0.956	0.932	0.947	0.978	0.978
parkinsons	0.886	0.904	0.892	0.872	0.876	0.891	0.875	0.883	0.871	0.886	0.883	0.905	0.881
segment	1.0	0.999	0.999	1.0	1.0	0.999	0.998	0.989	0.995	0.972	0.975	0.999	0.999
stock	0.982	0.979	0.978	0.982	0.981	0.975	0.978	0.887	0.933	0.931	0.942	0.976	0.974
zoo	0.997	0.99	0.989	0.965	1.0	1.0	1.0	0.973	0.998	0.96	0.997	0.952	0.925

Table 32. MAR 50% Results when feature selection is enforced

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.868	0.855	0.887	0.867	0.878	0.852	0.84	0.785	0.838	0.821	0.821	0.863	0.884
boston	0.884	0.867	0.897	0.884	0.884	0.868	0.877	0.814	0.786	0.869	0.856	0.853	0.863
churn	0.766	0.758	0.773	0.764	0.774	0.754	0.747	0.704	0.711	0.765	0.776	0.766	0.757
compas-two-years	0.659	0.662	0.668	0.665	0.674	0.642	0.664	0.643	0.658	0.663	0.671	0.659	0.666
image	0.812	0.771	0.76	-	-	0.742	0.779	0.721	0.735	0.812	0.812	0.805	0.81
page-blocks	0.959	0.963	0.964	0.952	0.959	0.963	0.962	0.89	0.902	0.878	0.87	0.961	0.957
parkinsons	0.865	0.809	0.855	0.861	0.88	0.839	0.813	0.8	0.801	0.865	0.884	0.803	0.85
segment	0.988	0.995	0.993	0.991	0.991	0.995	0.991	0.877	0.909	0.877	0.895	0.991	0.993
stock	0.961	0.962	0.935	0.966	0.973	0.957	0.95	0.876	0.869	0.713	0.803	0.954	0.939
zoo	0.943	0.96	0.905	0.96	0.956	0.949	0.99	0.946	0.948	0.896	0.863	0.967	0.884

Table 33. MAR 50% Results when feature selection is excluded

Dataset	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.877	0.862	0.873	0.879	0.858	0.851	0.861	0.772	0.821	0.822	0.829	0.86	0.874
boston	0.871	0.887	0.897	0.894	0.899	0.885	0.875	0.818	0.832	0.866	0.864	0.849	0.867
churn	0.778	0.755	0.766	0.775	0.793	0.756	0.752	0.72	0.737	0.773	0.764	0.759	0.76
compas-two-years	0.665	0.666	0.682	0.663	0.674	0.658	0.678	0.637	0.662	0.666	0.682	0.665	0.679
image	0.857	0.824	0.824	-	-	0.836	0.824	0.731	0.749	0.841	0.828	0.857	0.852
page-blocks	0.964	0.963	0.962	0.959	0.96	0.96	0.966	0.896	0.911	0.898	0.918	0.961	0.958
parkinsons	0.887	0.875	0.847	0.883	0.876	0.855	0.865	0.778	0.77	0.908	0.903	0.887	0.869
segment	0.996	0.995	0.993	0.992	0.994	0.992	0.994	0.886	0.903	0.907	0.934	0.993	0.993
stock	0.972	0.962	0.959	0.972	0.971	0.956	0.962	0.871	0.897	0.925	0.928	0.956	0.956
zoo	0.96	0.954	0.952	0.973	0.981	0.908	0.924	0.948	0.952	0.943	0.965	0.954	0.968

Table 34. MAR 50% Results Optimizing Overall (with and without feature selection)

Datasets	Best	MM	BI+MM	MF	BI+MF	GAIN	BI+GAIN	SOFT	BI+SOFT	PPCA	BI+PPCA	DAE	BI+DAE
Australian	0.878	0.855	0.887	0.882	0.861	0.843	0.849	0.785	0.838	0.822	0.821	0.863	0.884
boston	0.893	0.867	0.897	0.895	0.893	0.871	0.875	0.818	0.832	0.866	0.856	0.853	0.863
churn	0.766	0.755	0.766	0.775	0.775	0.761	0.762	0.72	0.737	0.765	0.764	0.766	0.757
compas-two-years	0.665	0.666	0.682	0.668	0.676	0.647	0.678	0.643	0.662	0.666	0.682	0.665	0.679
image	0.857	0.824	0.824	-	-	0.828	0.83	0.731	0.749	0.841	0.828	0.857	0.852
page-blocks	0.962	0.963	0.964	0.958	0.961	0.959	0.964	0.896	0.911	0.898	0.918	0.961	0.957
parkinsons	0.887	0.875	0.847	0.876	0.87	0.855	0.84	0.8	0.77	0.908	0.903	0.887	0.869
segment	0.996	0.995	0.993	0.994	0.997	0.985	0.995	0.886	0.903	0.907	0.934	0.991	0.993
stock	0.972	0.962	0.959	0.969	0.975	0.953	0.954	0.871	0.897	0.925	0.928	0.954	0.956
zoo	0.96	0.954	0.952	0.954	0.957	0.916	0.979	0.948	0.952	0.943	0.965	0.954	0.968