



Extended MetaLIRS: Meta-learning for Imputation and Regression Selection Model with Explainability for Different Missing Data Mechanisms

Işıl Baysal Erez¹ · Jan Flokstra¹ · Mannes Poel¹ · Maurice van Keulen¹

Received: 27 November 2024 / Accepted: 5 May 2025 / Published online: 29 May 2025
© The Author(s) 2025

Abstract

Missing data can cause problems for model decision-making procedures, which is a ubiquitous problem in data science. Imputation is a popular and agile method to handle missing data that enables predictive models to work smoothly. Given the plethora of available imputers and regressors, selecting the appropriate pair for a specific problem can be challenging. Many search-based approaches, such as AutoML tools, require significant processing power and focus on developing the predictive model itself, often bypassing the imputation processes with only basic steps. Models based on a meta-learning approach, combining data characteristics and algorithm performance, can help in imputer and method selection for missing data. Nevertheless, enhancing confidence in resulting models' decisions and ensuring transparency through explainability is crucial. In this paper, we present a novel meta-learning-based framework for imputer/regressor selection; this framework is called MetaLIRS. With MetaLIRS, we built three recommendation models for MCAR, MAR, and MNAR data mechanisms, respectively, achieving an accuracy ranging from 63% to 67%. These models are designed to select the best-fitting imputer/regressor pair in terms of RMSE values of regressors. The models were learned using 26 datasets with six imputation methods, six regression methods, six missing levels (1%, 5%, 10%, 30%, 50%, and 70%), and 33 meta-features derived from missing data. A limitation of the approach is that due to a group of imputer/regressor pairs with very low occurrence, the so-called "other" class, the recommender occasionally predicts the unhelpful "other". The main scientific contribution is the MetaLIRS framework which provides interpretability for its recommendations, is resource-efficient for recommendation model outputs, and enables rapid selection of the best imputer/regressor pair for a specific dataset without exhaustive brute-force searching.

Keywords Missing data · Imputation · Meta-learning · Explainability

1 Introduction

Machine learning models have emerged as indispensable instruments for analyzing various datasets across multiple fields, such as business, biology [1], chemistry, engineer-

ing [2], health [3], and transportation [4], where missing data are frequently present. In machine learning applications, regression-based supervised learning algorithms are frequently used as the bedrock of predictive modeling. However, most regression models cannot handle missing data directly, and this situation can affect model performance.

Several solutions are proposed, such as missing data deletion [5], data imputation [6] and the use of machine learning (ML) method adaptations for addressing missing data [7]. Deletion is an ad hoc solution, which involves removing the rows or columns that contain missing values from the dataset. Data imputation fills in the missing values with estimated ones, which can be based on statistical methods such as mean, mode, and median imputation [2, 8], machine learning-based methods such as k -Nearest Neighbors (k -NN) [9], and deep learning-based methods such as Variational Autoencoders (VAEs) [10], Generative Adversarial Imputation Networks

✉ Işıl Baysal Erez
i.baysalerez@utwente.nl
Jan Flokstra
jan.flokstra@utwente.nl
Mannes Poel
m.poel@utwente.nl
Maurice van Keulen
m.vankeulen@utwente.nl

¹ Data Management and Biometrics, University of Twente, Drienerlolaan 5, 7500 AE Enschede, Overijssel, The Netherlands

(GAIN) [11] models, and some ML method adaptations, such as tree-based methods, handle missing data naturally. Some tree-based algorithms can work directly with missing data through different splitting mechanisms [7], such as probabilistic splits in C4.5 [12], block propagation in LightGBM [13], and Missing Incorporated in Attributes (MIA) in XGBoost [14].

Nowadays, the impact of various imputation methods on regression models and the challenge of finding universally consistent models in the presence of missing data have become central topics [7, 15, 16]. According to the No Free Lunch theorem [17], no one solution is ideal for every problem. One of the most common approaches is trial and error, often implemented through grid search, where data analysts test various imputation methods and regression models. This approach, however, is time-consuming and resource-intensive because it requires testing each option separately. To streamline this process, Automated Machine Learning (AutoML) tools such as TPOT (Tree-based Pipeline Optimization Tool) [18], and Auto-Sklearn [19] have been developed to automate and optimize the search for the best-fitting machine imputation method, learning method, and accompanying configurations of hyperparameters [19]. Additionally, commercial AutoML systems like Google AutoML [20], JADBio [21], and Autogluon [22] offer similar automation but with limitations. Despite their advantages, AutoML systems tend to have a limited range of imputation methods [23], and their search-based approach is rather computationally expensive [24].

Meta-learning approaches can offer a solution [25, 26] for algorithm selection and recommendation. These approaches learn the relationship between meta-features and the performance of previously trained models via a meta-database. Meta models are then trained using these datasets by extracting the meta-features (characteristics of datasets) [27–29]. Using a recommendation model eliminates the need for a search, which involves the cumbersome process of trying every combination, as it relies on a pre-trained meta-database. Recommendation models can be used for some tasks, such as single/multi-classification and single/multi-regression tasks [28].

To the best of the authors' knowledge, there has been no work using meta-learning approaches to select the combination of both an imputer (or no imputer) and a regressor.

Besides the challenges of finding appropriate imputer and regressor algorithms for the missing dataset, explaining the recommendations provided by these models is crucial for data analysts. They need to interpret these explanations to ensure that the models are making informed decisions, maintain transparency, and enhance trust in the outcomes. Meta models are generally built as black-box models, which possess numerous parameters. This complexity makes it challenging to comprehend how decisions are made and

how these models work internally. To address this issue and thereby increase data scientists' trust in the models, the field of Explainable AI (XAI) proposes a set of methods that enable data analysts to understand the reasons behind the recommendations made by the models [30].

The explainability of models can be evaluated on different aspects. Nauta et al. [31] defined 12 desirable characteristics for the explainability of black-box models, combining different perspectives. We interpret the recommendation models we offer based on the confidence property mentioned in this study [31], using post hoc analysis to help determine how much trust users can place in our model and its predictions. Regardless of the internal structure of the model, two separate approaches for post hoc analysis are employed: global and local explanations. Global methods provide an overview of the highly relevant meta-features used by the model across the entire dataset, while local explanation methods produce explanations for individual recommendations [32].

We aim to address the following three research questions:

- 1) How can we design a framework for the recommendation to select a (near) optimal imputer/regressor pair?
- 2) How can the performance of the recommendation (meta) model change based on different missing data mechanisms?
- 3) How can global and local explanations be obtained for selecting an imputer/regressor pair?

This work builds upon our previous study [33], expanding and improving the results by exploring new dimensions of the research. The recommendation model described in [33] is based on less datasets, uses fewer meta-features and takes only the MCAR mechanism into consideration. Explainability analysis was performed using only a model-specific approach. The main contributions of this paper can be summarized as follows:

-We propose a framework to construct a recommendation model for recommending imputer+regressor pairs. A framework has been built by generating missing data at various levels (1%, 5%, 10%, 30%, 50%, 70%), extracting 33 meta-features from the missing data at each missing level, and using six imputation methods along with six regression models.

-The proposed recommendation model is based on Random Forest and gives imputer+regressor pair recommendations according to MCAR, MAR, and MNAR data mechanisms [6].

-We used global and local explanations for the recommendation model using the SHAP method and aggregated decision rules. This approach involves explaining the overall recommendations of the recommendation model (global) and the reasoning behind individual recommendations (local).

2 Motivation

We developed a framework that enables an explainable recommendation model for selecting the optimal or near-optimal pair of imputer and regression methods, thereby reducing the computational cost and time associated with brute-force methods. The objective of this approach is to facilitate faster and more effective method selection. Our first motivation for this study was that although many algorithm selection studies have been conducted using the meta-learning approach, there is a notable absence of research on selecting imputer+regressor pairs in the context of missing data. According to a recent study [34], there have been only seven studies since 2005 that aim to select the imputation method based on meta-learning. Furthermore, no meta-learning approach was found for regression model selection in case of missing data. In addition, the imputation methods used are similar to those in the studies reviewed in this paper [34], and no advanced imputation methods have been encountered. These studies are detailed in the related work section. This gap in the literature has increased our motivation for this study.

Another motivation is that time complexity is high in search-based algorithms. We present the training time using different imputation and regression methods per missing level and data mechanisms. Table 1 shows the processing time (seconds) of each model. TPOT and Auto-sklearn do not select imputer automatically, so we used a default imputer, such as the median for TPOT and mean imputer for Auto-sklearn. It appears that the search for the regression method is costly, even when using a single imputer. Additionally, finding the right imputer and regressor through the trial-and-error (TT) method can be very time-consuming. The method consists of evaluating the current imputer and regressor pairings until an optimal result is obtained. However, thanks to the trained model presented in this study, selecting the appropriate imputer+regressor pair for the dataset takes only a few seconds.

3 Related work

3.1 Using imputation+regression methods presence of missing data

When dealing with missing data using imputation and regression methods, there are two strategies for making prediction analysis: 1) impute and then regress, and 2) regression without imputation. In this section, we discuss the most relevant work for both applications.

The most common imputation method based on statistics is mean imputation [8]. In some studies, mean imputation has been found to perform better compared to more com-

Table 1 Comparison processing time (seconds) of Methods on CASP dataset with 30% missing level for one. Lower time is better. (*imp.* refers to imputers, and *reg.* refers to regressors. TT_imp. refers to the trial and error method for imputers. TT_reg. refers to the trial and error method for regressors.)

Methods	MCAR	MAR	MNAR
TT_imp	1022.00	436.55	985.79
TT_reg	911.89	934.30	932.43
Auto-sklearn_imp	N/A	N/A	N/A
Auto-sklearn_reg	597.02	607.65	603.12
TPOT_imp	N/A	N/A	N/A
TPOT_reg	613.18	640.47	704.51
Our model	0.09	0.04	0.08

Bold values represent the lowest processing time.

plex models like iterative imputation and nearest neighbor imputers [35].

Bertimas et al. [15] investigated the effects of imputers on the performance of regression models, theoretically and empirically, using synthetic, semi-real, and real missing data. They explored various missingness mechanisms, including MCAR, MAR, MNAR, and adversarially missing (AM) mechanisms, across missing levels ranging from 10% to 80%. Their key findings were as follows: 1) In datasets with naturally occurring missing values, authors found that simpler imputers, such as mean or mode, performed better with regression methods than more complex imputers like MICE. 2) The performance of the imputer+regressor pair varies depending on the missingness pattern. 3) The complex interactions between imputation and regression models are not yet fully understood and require further investigation.

Morven et al. [36] also employed the "impute then regress" approach, emphasizing that even with optimal imputation, model performance may still be affected by the complexity of learning in the regressors. To address this issue, they presented a neural network-based approach called NeuMiss, which uses the Neumann series to perform the learning process for both the imputer and regression methods simultaneously.

3.2 Meta-learning approaches for algorithm selection and recommendation

Meta-learning approaches, also known as "learning to learn", are based on early studies [37] and are commonly used for algorithm recommendation [38]. These approaches use machine learning algorithms to build relationships between the performance of algorithms and the characteristics of the data [25] to recommend the best algorithms for new datasets [25]. Moreover, many meta-learning approaches are utilized separately for a single-task recommendation, such as regression or imputation [28, 39]. There are some works to

select imputation methods using meta-learning approaches in the literature [34, 94]. As part of an earlier study, Zou et al. used an Extension of Learning from Examples Modules Version 2 (ELEM2) algorithm as a meta-learner to select different imputers such as Deletion, Median/Mode Imputation (MMI), Group/Class MMI, EM Imputation, etc. and one machine learning-based method (k -NN) for handling missing data 30 tabular datasets and missing level ranging from 10% to 90%. They utilized simple meta-features such as the number of features and classes, trained the ELEM2 meta-model for some new test datasets and compared classification error rates according to ignore, k -NN, and C4.5 (without imputation). Their meta-model obtained the best performance, especially with the low levels of missing data. Similarly, Zagatti et al. [23] developed a recommendation system to automatically select data preprocessing methods, including data normalization, standardization, imputation, and class balancing. They extracted 73 meta-features from 80 tabular datasets based on statistical and information theory methods using 11 data preprocessing techniques and obtained 180 possible pipelines. For the meta-labeling strategy, the five top-ranked pipelines were determined based on the accuracy of classification models. They used the k -NN algorithm as a meta-model to describe the most similar datasets according to their meta-features. They compared their meta-model with some AutoML tools, such as Auto-sklearn and Learn2Clean, to evaluate their performances. The results were competitive with AutoML tools in terms of performance, and it achieved results much faster. Another study [40], aiming to be fast and effective in algorithm selection performance, proposed a novel approach called A Provenance Meta-Learning Framework. The framework was established by considering nine influential factors based on classification method, missing data mechanism, missing data pattern, missing level, number of labels, numerical features, and categorical features as inputs for meta-learning. Meta labels are determined by imputation accuracy. The Framework reaches 77.72% accuracy.

In the literature, studies on the selection of regression methods using meta-learning have not discussed any missing data problems and have focused directly on the selection of the prediction method [28, 39]. Lorena et al. [39] adapted data complexity meta-features such as feature correlation measures, linearity measures, smoothness measures, geometry, topology, and density measures for synthetic and real regression tasks. They used the SVR (Support Vector Regression) method as the meta-learner to recommend 14 different regression methods designed to set the parameter values of the support vector regressors and estimate the type of regression function of some synthetic data sets. In recent years, Aguiar et al. [28] extracted 58 meta-features based on statistical model performance metrics related to linear regression, data smoothness, and dataset distribution to find the most

suitable multi-target regression models. They compare baselines (majority, random) and some meta-learners (RF, SVM (linear, radial, and polynomial kernels), XGB) performance. RF meta-model obtained the best performance. Based on previous studies, it should be noted that the impact of advanced imputation methods on algorithm selection in cases of missing data has not been studied comprehensively enough.

3.3 Explainability of meta-model for recommendation

Pimentel et al. proposed meta-features based on correlation and dissimilarity for the selection of clustering algorithms. They compared the feature importance values of the models obtained with meta-features in different groups [41] using model-specific methods such as Random Forest. For Random Forest model-dependent feature importance method is based on the mean decrease in impurity [42–46]. According to their results, the percentage (%) of values in the interval [0.4,0.5] was the most important, and variance and standard deviation had a lower importance. Woźnica and Biecek [47] created a recommendation model using the gradient boosting tree algorithm and incorporated landmark meta-features to propose the selection of the most important hyperparameters for this model. For model explanation, they calculated the importance of meta-features using model-agnostic methods. The hierarchical clustering method was used to understand the effect of highly correlated meta-features on model performance. Ceteris Paribus (CP)/ Individual Conditional Expectation (ICE) profile approaches were used to show the impact of a change in the value of a variable on model outputs.

Kostovska et al. [48] developed an approach to select Multi-Label classification algorithms using five meta-feature groups, such as dimensionality, label distribution, label imbalance, label relationship, and attribute metrics in different application domains. They calculated the importance of the features of the model using explainability techniques and the SHAP algorithm. Shao et al. [49] propose an interpretable meta-learning framework (EFFECT) to explain recommendation results and algorithm performance using black (SVM, Random Forest, XGBoost, Neural network model meta-learner, CNN, ResNet, FT-Transformer)-and-white box (KNN and decision tree) models. Authors employ intervention-based and counterfactual-based causation to provide contextual explanations for the explainability of recommendation models.

We stated that this study is an extension of our previous work [33]. In our study, only the important features (from 12 meta-features) of the recommendation model prepared for the MCAR data mechanism were calculated. To the best of the author's knowledge, no other research has been published about selecting imputer+regressor pairs based on a

meta-learning approach for MCAR, MAR, and MNAR data mechanisms. Moreover, we realized that the use of explainability methods in recommendation models created with the meta-learning approach has not yet been much explored. We provided an approach for a model of recommendations.

4 Methodology

Our methodology is visualized in Figure 1 and consists of three main steps: **4.1: Creating a meta-database (including steps 1.1-1.9 in Figure 1)**, which involves a set of 26 complete datasets collected from various domains, such as health, stock market, social, and physical [50, 51]. We also utilized the OpenML-CTR23 benchmark to select some numerical datasets [52], and selected datasets are given in Table 9 in Appendix C.1.

The datasets were split into train and test sets in a 75/25 ratio and generated missing data synthetically according to the MCAR, MAR, and MNAR data mechanisms with missing levels of 1%, 5%, 10%, 30%, 50%, and 70%. MCAR data mechanism, the probability of missing data is independent of non-missing and missing values. **MCAR data simulation:** Since we generate missing values synthetically, we denoted the full data matrix as X and applied a binary mask $\forall \omega_{ij} \in \Omega_{MCAR}$, $\omega_{ij} \in \{0, 1\}^{n \times d}$, via using random variables drawn from a Bernoulli distribution $\mathcal{B}(p)$ on X ($\omega_{ij} \sim \mathcal{B}(p)$).

The missingness of the MAR data mechanism depends on the values of the observed variables. **MAR data simulation:** This generation includes two steps. a) A subset of data to be observed is determined using (p_{obs}). b) The logistic model $\omega_{ij} \sim \frac{1}{1+e^{-(w \cdot X)}}$ created with random coefficients was applied to the remaining data set to create missing values. Missing data are generated by keeping a certain proportion of observed data points fixed (p_{obs}), selects a subset of fully observed variables and creates missing values in the remaining variables using a logistic model. We set (p_{obs})=0.5. The missingness is conditional on the unobserved data in the MNAR data mechanism. **MNAR data simulation:** It is similar to the MAR data mechanism simulation, a logistic model was used to generate missing data. This model computes the probability of missing data using all variables, both observed and unobserved, as inputs. Then, the input variables are randomly masked according to the MCAR data mechanism. For all missing data generation, we reproduced these similar settings [53].

Meta-features: Different types of meta-features, such as general, statistical, and complexity-based, were extracted from missing datasets (detailed information is given in Table 3).

We investigated studies on the selection of relevant meta-features, from early works [54] to recent ones [25, 29, 39].

Since this study investigated the choice of imputer+regressor, we could not access meta-features directly related to these tasks. We used three different types of meta-feature families. These meta-features are categorized as follows: a) General meta-features: These meta-features represent simple characteristics of datasets such as the number of features, number of instances, and missing ratio. b) Statistical meta-features: These represent meta-features obtained by using statistical functions. c) Data complexity meta-features: Lorena et al. [39] proposed complexity-based meta-features for regression task. In regression problems, complexity determines how complex the relationships in the dataset are, how well the model can learn these relationships, and how to choose the appropriate model. This includes linearity measures, smoothness, geometry, topology, and density measures [showed MF21-26 in Table 3]. After determining the meta-feature families that will be useful in the study, our meta-feature analysis consists of 3 components using the meta-feature development framework suggested by the [55] study. 1) Object: Defines the data on which the process will be performed. A feature-based selection is made, and columns containing missing data are replaced by 0 according to the meta-function's capacity. 2) Meta-function: a function that analyzes the relevant features from missing data and converts it into output. A meta-function consists of an input layer (defined by specification and arity) and an output layer (characterized by datatype and arity). 3) Post-processing: The post-processing function processes the output from the meta-function and converts it into a meta-feature.

Imputers: We used seven imputation methods to estimate values for the missing data and complete the missing datasets based on statistical, machine learning, and deep learning methods. We selected the imputation methods utilized in the survey [2] and benchmark study [56], ensuring a comprehensive evaluation of their effectiveness.

1. Mean imputation: This is a simple method that replaces missing data with the average of observable features (column-wise) for a continuous variable [8].
2. k -Nearest Neighbors (k -NN) imputation: This method estimates missing values using k -NN algorithms and utilizes the information available from those neighbors regarding distance [9].
3. Imputation by Chained Equations (ICE): This method is one of the most commonly used imputers based on iterative cycling in a round-robin fashion using linear models conditioned on observed data. This approach is known as fully conditional specification in statistical literature [6, 57]. We used Ridge regression models as an estimator. We used the scikit-learn library for this implementation. This implementation [58] was inspired by the R software MICE package (Multivariate Imputation by Chained Equations [57]).

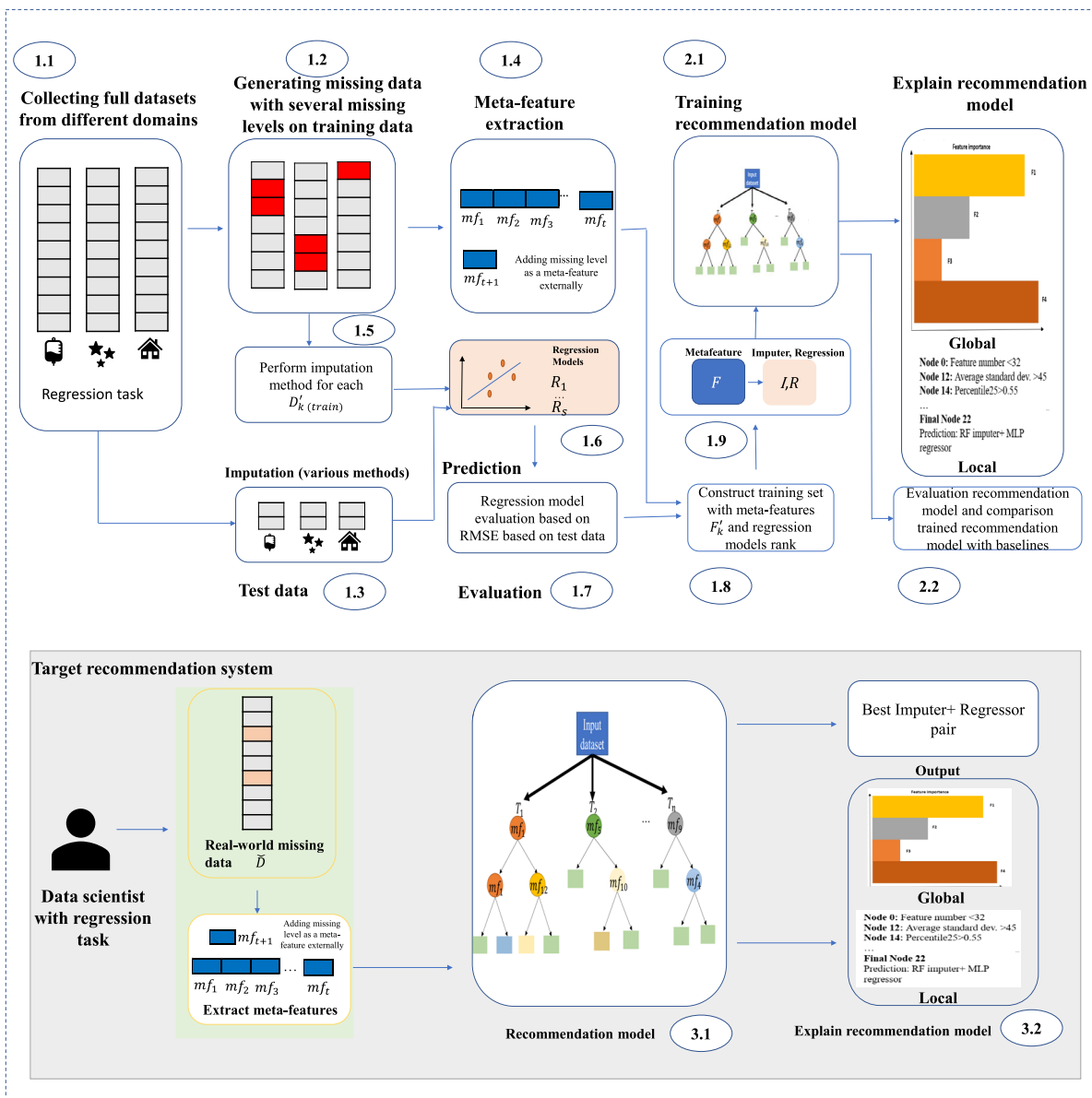


Fig. 1 An overview of the MetaLIRS framework. We divided the data into train and test split before imputation to prevent data information. We consider three mechanisms for generating missing data: MCAR, MAR and MNAR with missing levels of 1%, 5%, 10%, 30%, 50%, and 70%. To create a meta-database, we applied six imputation methods: Mean imputer, ICE, k -NN, MLP, GAIN and VAE, to the missing

training dataset. 33 meta-features extracted from missing datasets. Six regression models were used: XGBoost, LightGBM, AdaBoost, SVR, Ridge Regression and MLP. At the same time, XGBoost and LightGBM algorithms have also been trained directly with missing data. The performance of these trained models on test data was evaluated. Improved version of the methodology presented in [33]

Other iterative imputation methods based on machine/deep learning:

- MLP: It has an iterative process like ICE, and column-based prediction is performed using the MLP regressor [59].
- Generative Adversarial Imputation Networks (GAIN): This imputation is based on generative adversarial networks. We used the formal implementation of GAIN [11], but for the MAR data mechanism, we faced some prob-

lems in obtaining results, so we used this paper settings [56]. It has two important components: a generator and a discriminator. While the generator tries to produce data that is suitable for the distribution of the data, the discriminator tries to understand whether the data produced by the generator is real or fake with the help of the hint matrix.

- Variational Autoencoders (VAE): This imputation method comes from a generative model class similar to GAIN. In

Table 2 Results of a framework for systematic development of meta-features. GE: General meta-features. ST: Statistical meta-features. CO: Data complexity-based meta-features. DS: Descriptive Statistic. F: Features excluding the target, T: Target value.

Family	Meta-feature	Meta-function	Arity (Input)	Post-processing
GE	MF1	Missing ratio	F	Non-aggregated
GE	MF2	Target Cardinality	T	Non-aggregated
GE	MF3	Number of instance	F	Non-aggregated
GE	MF4, MF25	Dimensionality		Non-aggregated
GE	MF7	Feature count	F	Non-aggregated
ST	MF5, MF6	Standard deviation	F+T	DS (Mean)
ST	MF8, MF9	Mean	F+T	DS (Mean)
ST	MF10, MF11	Min, max	F	Non-aggregated
ST	MF12, MF 13	Variance	F+T	DS (Mean)
ST	MF26, MF29	Kurtosis	F+T	DS (Mean)
ST	MF27, MF28	Skewness	F+T	DS (Mean)
ST	MF30, MF31, MF32, MF33	First and last quartile	F+T	DS (Mean)
ST	MF14, MF15, MF16, MF17, MF18	Correlation	F+T	DS (Mean, sd, min., max.)
ST	MF19	Covariance	F	DS (Mean)
CO	MF20, MF21	Linear regression	F+T	DS (Mean)
CO	MF22	Euclidean norm	F+T	DS (Mean)
CO	MF23, MF24	Nearest neighbor regressor	F+T	DS (Mean)

this method, the missing data are replaced with random noise and then reconstructed by the model, which trains on the data, learns the latent representations, and uses these representations to fill in the missing values [10].

Regressors: We select various types of regressors, ranging from complex black-box models (MLP, Boosting Trees, SVM) to simple interpretable models (Ridge Regression).

1. MLP neural network (NN) consists of input, hidden, and output layers. It is an artificial neural network that progresses from data input to output (feedforward), then updates the weights by propagating the error signal of the network backwards with the back propagation algorithm, so the error decreases and the network learns (backpropagation) [59].
2. SVM was proposed by Vapnik in 1995 [60]. Support Vector Machines eliminate local minima problems and reach a single global optimum by using a convex quadratic optimization problem to find the best-separating hyperplane [61].
3. AdaBoost is known as the first boosting algorithm and was introduced into the literature by Freund and Schapire [62]. It aims to create a strong model by bringing weak models together. Weak models are added one after the other and given weight in a try to reduce errors.
4. XGBoost and LightGBM are also members of Gradient Boosted Decision Tree families [63]. While XGBoost

creates deeper trees using a level-based growth strategy, LightGBM uses three approaches: Gradient-based One-Sided Sampling (GOSS), Exclusive Feature Bundling (EFB), and histogram-based, and produces faster (GPU support) and lighter models using a leaf-based growth strategy [64]. XGBoost utilizes the 'Missing Incorporated in Attribute' (MIA) approach to handle missing data directly, while LightGBM employs a block propagation strategy for the same purpose. For more details, see [7].

Ranking and Mapping: This part is the last process for the construction of a meta-database. The performance of the algorithms obtained in the previous steps is collected in a database, and the Top₁ imputer+regressor pair is determined according to the RMSE values of the regressors. In more detail, the rank distribution was defined as follows: The rank is assigned as 1 for the first place, 2 for the second place, and so on. The meta-features extracted from the missing data are then mapped to the performance of the imputer+regressor pairs. The outcomes of this stage are integrated into a single meta-dataset by these specifications, and a recommendation model is ultimately generated as a result of the meta-learning process.

4.2: Training recommendation model (steps 2.1, 2.2 in Figure 1): We selected the Random Forest classifier as the recommendation model, originally proposed in [42]. Meta-feature vectors were used as inputs and top-performer imputer+regressor pairs were used as labels of the rec-

Table 3 Overview of meta-features used in this paper. Features: Variables excluding the target. Target: The variable that is the prediction target of the regression

Name	Description
MF1	The missing ratio.
MF2	The number of unique values in the target value.
MF3	The number of trained instances in the dataset.
MF4	The ratio of features to instances.
MF5	The average standard deviation of the features.
MF6	The average standard deviation of the target values.
MF7	The number of features.
MF8	The mean over the feature values.
MF9	The mean over the target values.
MF10	The minimum value of the features.
MF11	The maximum value of the features.
MF12	The average variance of the features.
MF13	The average variance of the target values.
MF14	The absolute mean of the correlation between features.
MF15	The average standard deviation of the absolute correlations among the features.
MF16	The mean of the absolute correlations between features and target values.
MF17	The maximum value of the absolute correlation between features and target values.
MF18	The minimum value of the absolute correlation between features and target values.
MF19	The average value of the absolute covariance between features.
MF20	The average of the squared residuals from the linear regression (l2).
MF21	The nonlinearity of a linear regressor (l3) measure.
MF22	Input distribution measure (s2).
MF23	The error of nearest neighbor regressor (s3) measure.
MF24	The nonlinearity of a nearest neighbor regressor (s4) measure.
MF25	The ratio of instances to attributes (t2) measure.
MF26	The average kurtosis values of features.
MF27	The average skewness values of features.
MF28	The average skewness values of target values.
MF29	The average kurtosis values of target values.
MF30	The average of last quartile values of the features.
MF31	The average of last quartile values of target values.
MF32	The average of first quartile values of target values.
MF33	The average of first quartile values of the features.

ommendation model. Appendix C.1 shows the class label distributions. Other classes have been created for recommendation models according to MCAR, MAR, and MNAR data mechanisms with performance values below a certain threshold (fewer than 7 occurrences). This results in a total of six classes. The Leave-One-Out Cross-Validation (LOOCV) strategy evaluated the model's overall performance.

Comparison baselines: We used some baselines (search and Large Language Model (LLM) based models) to compare our recommendation models for MCAR, MAR, and MNAR data mechanisms performances. Moreover, we calculated the difference between the outputs generated by the recommendation model and the optimal values to evaluate the model's performance. The datasets used for testing were

randomly selected from the framework. In the results section, we provided detailed information.

Explainability and interpretability of recommendation model: We selected two different explanation types: feature importance calculation and decision rules. We prefer to use model-agnostic methods to calculate feature importance values. Model-agnostic methods are flexible strategies that aim to interpret black-box model predictions of black-box models without considering the model's internal structure [32]. The most popular model-agnostic method is SHAP (SHapley Additive exPlanations), which calculates Shapley values and provides explanations of the importance of features globally and locally [65, 66].

Decision rules: These rules represent a set of requirements for tree-branching architectures. For prediction algorithms, split points are determined according to a certain threshold value between features, separate the data into relevant labels and create rules [32]. A comprehensive rule list is created by aggregating different rules according to the majority class for a given instance.

4.3: Recommendation (including steps 3.1, 3.2 in Figure 1): The data analyst uses the trained models we provide, leveraging the meta-features extracted from the test data, to identify the most suitable imputer/regressor pair for their dataset through the recommendation model. When the recommendation model recommends an additional (other) class, the imputer/regressor pair with the highest frequency in that group is selected.

For more detail, we also explained our approach by means of three algorithms in Appendix B.

Baselines (including steps 2.2 in Figure 1): In this study, we build our work on algorithm selection problem [38, 67]. Current algorithm selection methods include Bayesian optimization-based approaches, such as Auto-Sklearn, and evaluation-based algorithms, such as TPOT. Additionally, we experimented with the recent popular natural language processing (NLP) model ChatGPT as baselines for our framework. We compare the recommendation models we built for each missing data mechanism to the following three baselines:

1. Auto-Sklearn: This framework automatically produces machine learning pipelines that use Bayesian optimization to find optimal pipelines [19].
2. Tree-Based Pipeline Optimization Tool (TPOT): This tool uses a genetic programming-based optimizer to find optimal results [68].
3. ChatGPT-4: This is an artificial intelligence language model based (LLM) on a Generative Pre-trained Transformer (GPT) developed by OpenAI [69]. GPT models are designed to generate sentences, texts or paragraphs. Improved versions are released to the public every year and have a lot of parameters. GPT has 175 billion parameters [70]. It has a maximum response capacity of 4,096 tokens and a context window of 128,000 tokens. It was last updated on April 9, 2024 [71].

5 Results

5.1 Interaction between imputers and regressors

Table 4a, 4b, and 4c shows the performance of imputer/regressor pairs in terms of the number of times the pair achieves the best rank under the MCAR, MAR, and MNAR data mechanisms, respectively.

It can be seen in the *Top5* lists that for all missing data mechanisms, LightGBM without imputation is the best combination for most data sets. The combination achieves best rank for 28%, 24%, and 27% of the data sets for MCAR, MAR, and MNAR, respectively. Ridge is present as a regressor in every *Top5*, but it only reaches the best rank in combination with different imputers. Furthermore, it can be observed that imputers Mean, GAIN and VAE and regressors SVM, AdaBoost, and MLP are in none of the three *Top5* lists in any combination.

Regarding imputer/regressor selection, the results for MCAR and MNAR resemble each other quite well while MAR paints a rather different picture. For the MAR mechanism, XGBoost with no imputation achieves the best rank almost as often as LightGBM with no imputation. Moreover, a combination with KNN appears less advisable instead favoring ICE or MLP. It is an open question why the same imputer/regressor pairs function well for MCAR and MNAR mechanisms while apparently different choices are best for MAR.

Overall, it can be concluded that there is no single optimal imputer/regressor combination. The performance of the various combinations varies significantly depending on the characteristics of the dataset and the missing data mechanisms involved. Although LightGBM combined with no imputation came out on top, it is only the best choice in 27% of the cases overall.

5.2 Comparison of MetaLIRS to other algorithm selection tools

Table 5 shows performance comparison results of MetaLIRS with these tools: (1) TPOT framework, (2) Auto-sklearn, and (3) ChatGPT-4. We compare the tools in terms of prediction time and RMSE values of the imputer+regressor recommendation pairs.

The “Time factor” column in Table 5 shows speed factors ranging from 290 to 200043 and indicates how much faster the MetaLIRS model is compared to the best-performing model or, in cases where MetaLIRS has the best performance, how much faster it is compared to the runner-up model. MetaLIRS is several orders of magnitude faster than search-based or LLM-based recommendation approaches, even though Auto-sklearn and TPOT only consider simple imputers. The “Differences” column shows that a trained model using the MetaLIRS recommendation has comparable performance to a model trained with the recommendation of the best recommendation system, highlighting the percentage differences in performance based on RMSE. As the amount of missing data increases, the prediction task becomes harder. Hence, the performance of resulting regression models becomes lower in terms of higher RMSE. There is also more diversity in

Table 4 Performance of imputer/regressor pairs under three missing data mechanisms in terms of the number of times the pair achieves the best rank. Data sets were generated synthetically for the three missing data mechanisms for six levels of missingness (1%, 5%, 10%, 30%,

50%, 70%). We assigned a score of 1 to the method with the lowest RMSE. The best ranks are calculated as the average of the results from five running experiments with different seeds. No Imp. refers to No Imputation

Imputation Methods	Regression models						Total
	SVM	XGBoost	LightGBM	AdaBoost	MLP	Ridge	
(a) Synthetic generation of 156 datasets by applying the MCAR data mechanism to 26 original datasets.							
No Imputation	-	-	44	-	-	-	44
	-	6	-	-	-	-	6
KNN	-	1	19	1	2	14	37
Mean	-	1	1	-	1	-	3
ICE	1	1	13	1	-	5	21
MLP	2	1	19	3	-	6	31
GAIN	-		4	-	-	2	6
VAE	-	3	3	-	-	2	8
Total	3	13	103	5	3	29	156
Top-5 MCAR							
1	LightGBM + No Imp.						
2	LightGBM + KNN						
	LightGBM + MLP						
4	Ridge + KNN						
5	LightGBM + ICE						
No Imputation	-	-	22	-	-	-	22
	-	21	-	-	-	-	21
KNN			3	-	-	2	5
Mean	-	-	5	-	-	-	5
ICE	1	-	5	4	-	8	18
MLP	-	-	7	1	-	7	15
GAIN				1			1
VAE	-	-	-	1	-	2	3
Total	1	21	42	7	-	19	90
Top-5 MAR							
1	LightGBM + No Imp.						
2	XGBoost + No Imp.						
3	Ridge + ICE						
4	Ridge + MLP						
	LightGBM + MLP						
No Imputation	-	-	42	-	-	-	42
	-	2	-	-	-	-	2
KNN	-	2	22			8	32
Mean	-	-	4	1	1	3	9
ICE	2	1	8	2	1	9	23
MLP	-	2	16	1	-	13	32
GAIN	1	2	3	-	-	2	8
VAE	-	3	2	1	-	2	8
Total	3	12	97	6	2	37	156

Table 4 continued

Imputation Methods	Regression models						
	SVM	XGBoost	LightGBM	AdaBoost	MLP	Ridge	Total
Top-5 MNAR							
1	LightGBM + No Imp.						
2	LightGBM + KNN						
3	LightGBM + MLP						
4	Ridge + MLP						
5	Ridge + ICE						

Table 5 Performance in terms of RMSE of a trained model using the imputer/regressor as recommended by our MetaLIRS, TPOT, Auto-sklearn, and ChatGPT. The best results are shown in bold; the second-best performers with an absolute percentage difference within 4% of the Top-RMSE are shown in italics. The differences (%) represent the relative performance variation between the top-performing tool and MetaLIRS. Time factor shows how much faster MetaLIRS is compared to the best-performing model when MetaLIRS is not the top performer. Conversely, when MetaLIRS is the best performer, it shows how much faster it is compared to the runner-up model.

Data mechanism	Datasets	MetaLIRS	TPOT	Auto-sklearn	Chat- GPT	Differences (%)	Time factor
MCAR	Wine-red _{%1} [72]	<i>0.61</i>	0.59	0.59	0.64	-3.28	1319
	Wine-red _{%70}	0.69	0.70	0.69	0.71	0	150303
	Concrete _{%70}	10.23	10.45	9.27	10.23	-9.38	99312
	Diamonds _{%70} [50]	465.40	452.88	431.67	508.28	-7.25	20179
	Parkinson _{%1} [73]	<i>1.38</i>	1.35	1.48	12.30	-2.17	30768
	Parkinson _{%70}	<i>6.66</i>	7.10	6.3	7.12	-5.41	20023
MAR	Wine-red _{%1}	<i>0.60</i>	0.59	0.59	<i>0.60</i>	-1.67	290
	Wine-red _{%70}	<i>0.65</i>	0.74	0.66	0.61	-6.15	200043
	Concrete _{%70}	4.97	10.45	9.59	<i>5.06</i>	1.81	N/A
	Diamonds _{%70}	318.77	452.88	249.63	302.58	-21.69	8398
	Parkinson _{%1}	<i>1.34</i>	1.35	1.23	2.32	-8.21	119638
	Parkinson _{%70}	1.39	7.10	6.42	<i>2.14</i>	53.96	119188
MNAR	Wine-red _{%1}	0.59	0.59	0.59	0.60	0	3150
	Wine-red _{%70}	<i>0.71</i>	0.73	0.67	0.76	-5.63	3868
	Concrete _{%70}	5.93	10.45	9.43	8.32	40.30	2838
	Diamonds _{%70}	387.26	417.62	369.54	605.05	-4.58	866
	Parkinson _{%1}	1.35	1.56	1.15	1.44	-14.81	100083
	Parkinson _{%70}	<i>7.10</i>	6.86	7.54	6.93	-3.38	5853

The lowest RMSE values and the percentage differences in cases where MetaLIRS achieves the lowest RMSE are shown in bold

which imputer/regressor pair is best under these circumstances, which also shows with the recommenders.

ChatGPT shows overall quite acceptable recommendations. This is quite surprising since it wasn't engineered for this purpose at all.

Auto-Sklearn had the lowest RMSE in the two cases. Our proposed model performed the best for the Concrete dataset_{%70} and Parkinson_{%70}, while the Wine-red_{%1}, and Wine-red_{%70} datasets showed similar performance to the optimum result.

When we look at the results for the MNAR mechanism, it is observed that MetaLIRS performed better in more cases than the MCAR and MAR mechanisms.

It is worth noting that Auto-sklearn and TPOT tools do not have the capability to select both an imputer and a regressor automatically. Even though they have simple imputers like mean and median imputation, the prediction time takes significantly longer compared to our model.

Interestingly, it has been observed that ChatGPT-4 shows similar results to other tools except for Parkinson_{%1} and Diamonds_{%70} data sets.

For the MAR data mechanism model, it can be observed that Auto-sklearn generally achieves the best results. Although there are fewer meta-instances in our meta-database than for the MCAR and MNAR mechanisms, MetaLIRS has consistently produced nearly optimal recommendations, while Auto-sklearn performs well in most cases.

Table 6 Evaluation metrics results for the recommendation model with MCAR, MAR, and MNAR data mechanisms are presented. Higher values indicate better performance for all metrics. The results are shown as the mean and standard deviation over 30 different random initializations using the Leave One Out split method. The Accuracy of random selection is determined as 17% since the recommendation models have six classes. “Mecha.” means “missing data mechanism”.

Mecha.	Accuracy	Precision	Recall	F1-score
MCAR	0.63±0.02	0.64±0.02	0.63±0.02	0.63±0.02
MAR	0.67±0.03	0.69±0.03	0.67±0.03	0.67±0.03
MNAR	0.64±0.02	0.63±0.02	0.64±0.02	0.63±0.02

For the MNAR data mechanism, while MetaLIRS shows the best performance three times, TPOT shows the best performance two times. This is especially true for wine-quality datasets, as all data mechanism results are similar.

To sum up these three analyses, it can be said that MetaLIRS reached advanced AutoML tools most of the time. No model consistently produced the best results across the datasets in a dominant way, even while each model outperformed the others on specific datasets. Moreover, as shown in Table 5, MetaLIRS provides great advantages in terms of recommendation time.

5.3 Performance of recommendation model

Table 6 shows the performance of the recommendation model based on accuracy, precision, recall, and F1-score.

To evaluate the impact of a non-optimal recommendation by MetaLIRS, we compared the performance of a model trained using the optimal imputer+regressor pair with the recommended pair. The results are presented in Table 7. For the selected datasets, two show a difference of less than 5% between the optimal performance and the performance reached according to the recommendation. This means that even though the recommender has an accuracy of 63%–67% (see Table 6), following the recommendation almost always leads to a model with near-optimal performance. We also analyze the evaluation metrics of recommendations, considering cases where the model’s predictions, with a deviation of less than 5% from the optimal value, are accepted as correct. Results are shown in Table 18 in the Appendix C.3.

5.4 Explainability of recommendations

We focus in this paper on model-agnostic methods for explainability. Model-agnostic methods are strategies that aim to interpret predictions of black-box models without considering the model’s internal structure. One such method, SHAP (Shapley Additive exPlanations) calculates Shapley values to measure each feature’s contribution to the model’s output. By doing so, SHAP provides data analysts with

insights into the importance of features in a model’s predictions.

Figure 2 and 3 contain summary plots that illustrate feature effects and feature importance (based on Shapley values) for the MLP/LightGBM class and the No imputation/LightGBM class, respectively, across three different data mechanisms. The *Top*₁ meta-feature varies among the missing data mechanisms. For example, the *Top*₁ feature ranking frequently shows variability in the prediction of class MLP/LightGBM, depending on the missing data mechanisms. The number of features (MF7) meta-feature is the most important, while the average of first quartile values of the dataset meta-feature (MF33) is the most important for the MAR data mechanism for MLP/LightGBM class prediction. The maximum value of the features (MF11) is the most important meta-feature for the MNAR data mechanism. This meta-feature is also ranked second in importance for the MCAR data mechanism. Low values of the maximum value of the features (MF11) meta-feature negatively affect MLP/LightGBM predictions for MCAR and MNAR data mechanisms. The average of the squared residuals from the linear regression (MF20) is a meta-feature that is among the *Top*₅ most important meta-features for MCAR and MAR data mechanisms. However, the high value of the average of the squared residuals from the linear regression (MF20) meta-feature has a negative effect on model output for the models of MCAR and MAR data mechanisms. There are some interesting results for MLP/LightGBM class prediction, while the meta-feature derived from the average quartile values of features within the same family is highly important for the model’s output, the same meta-feature for target values has minimal impact. This unexpected result suggests that the quartile characteristics of the target variable contribute little to the model’s decisions, unlike those of input features, challenging the assumption that similar statistical metrics hold comparable relevance for both surprisingly. In predicting the no imputation/LightGBM class, the number of features (MF7) meta-feature and the number of unique values in the target variable (MF2) meta-feature have important impacts under the MCAR and MNAR data mechanisms, respectively, similar to in the previous class prediction. Furthermore, the average of the last quartile values of the dataset meta-feature (MF31) is the most important for the no imputation/LightGBM class under the MAR data mechanism scenario. The average of the squared residuals from the linear regression (MF20) is the only meta-feature that is among the *Top*₅ most important meta-features for predicting both classes for the MAR data mechanism. Interestingly, a low value of this meta-feature negatively impacts the model’s output for class no imputation/LightGBM, while positively contributing to the prediction of class MLP/LightGBM. This shows that the meta-feature important for predicting these classes is related to the effect of the imputation method. The

Table 7 Difference between the performance of a trained model using the recommended vs. the best imputer/regressor pair in terms of RMSE. Experiments run one time. The lowest percentage values are shown in bold

Dataset	Mechanism	$RMSE_{rec.}$	Recommendation	$RMSE_{best}$	Best pair	Percentage (%)
Geo.001	MCAR	16.02	LightGBM (No Imp.)	16.02	LightGBM (No Imp.)	0.00
Qsar.005	MCAR	0.88	k -NN+LightGBM	0.84	SVM+MLP	4.55
Space.005	MCAR	0.11	k -NN+LightGBM	0.11	k -NN+LightGBM	0.00
Wisc.001	MAR	34.46	XGBoost (No imp.)	33.24	GAIN+ADABOOST	3.54
Bupa001	MAR	3.11	Other	2.88	GAIN+ADABOOST	7.40
Kin8030	MAR	0.02	LightGBM (No Imp.)	0.02	LightGBM (No Imp.)	0.00
Qsar030	MAR	0.06	XGBoost (No Imp.)	0.00	VAE+Ridge	100.00
Cars005	MNAR	2389.72	Other	2306.27	k -NN+LightGBM	3.49
CASP070	MNAR	4.80	LightGBM (No Imp.)	4.79	Mean+LightGBM	0.21
CPU070	MNAR	4.69	Other	4.69	ICE+LightGBM	0.00
Delta030	MNAR	0.00	Other	0.00	GAIN+Ridge	0.00

positive contribution of a low average of the squared residuals from the linear regression (MF20) value to the model can be explained by the fact that MLP models effectively handle data with low linearity.

Additionally, meta-features that are not very significant for MLP/LightGBM class prediction show fluctuation based on the missing data mechanisms. Under the MCAR data mechanism, for instance, the nonlinearity of linear regression (MF21) meta-feature has the lowest contribution in class MLP/LightGBM prediction; under the MAR mechanism, the average of first quartile values of target values (MF32) is the least significant; and under the MNAR mechanism, the average of last quartile values of the dataset (MF30) has the smallest influence. For prediction of no imputation/LightGBM class, while the average of last quartile values of target values meta-feature (MF31) has not so much effect on model output for MCAR data mechanism, the same meta-feature has the most important factor according to MAR data mechanism. In summary, the impact of meta-features differs for both classes and missing data mechanisms.

The impact of meta-features on prediction, represented by mean SHAP values aggregated importance of the meta-features in each missing data mechanism, is shown in Figure 4. We draw lines between the Top_{20} meta-features of the three recommendation models.

If the meta-features are unimportant (i.e., not among the Top_{20}), the line is marked in gray. According to this line figure, the number of unique values in the target values (MF2), the average standard deviation of the absolute correlations among the features (MF15), the average of first quartile values of the dataset (MF33) meta-features are among the Top_5 most important features for the MCAR and MNAR data mechanisms. Interestingly, the meta-features important for the MAR mechanism, except for the average of first quartile values of the dataset (MF33) meta-

feature, are considered less important for the MCAR and MNAR data mechanisms. The reason for this could be the label distribution in each model. In the recommendation model for the MAR data mechanism, classes no imputation/LightGBM and MLP/LightGBM are overlapping, whereas in the MCAR and MNAR models, there are more overlapping outputs in the meta-databases, which allows them to provide recommendation outputs for no imputation+LightGBM, KNN+LightGBM, and MLP+LightGBM.

Figure C.1 shows the aggregated rules that were taken from the Random Forest model for sample 35th according to MCAR and MAR data mechanisms. These rules present decision rules based on certain thresholds for meta-features (MFs) under various scenarios. The levels at which a meta-feature becomes significant in impacting the result for this specific sample are shown in the 'Higher' and 'Smaller' columns. This analysis was conducted by selecting the 35th sample, which shows differences in the output of the recommendation model. ICE/LightGBM class was predicted for the MCAR data mechanism, and the MLP/LightGBM class was predicted for the MNAR data mechanism, respectively, based on the 35th sample. The average standard deviation of the target values (MF6), the average variance of the target values (MF13), and the average of the last quartile values of target values (MF31) were not observed in the rules for MCAR data mechanism, while the missing ratio (MF1), the mean over the target values (MF9), the minimum value of the features (MF10), the average standard deviation of the absolute correlations among the features (MF15), and the average of first quartile values of target values (MF32) did not appeared in the rules. The absolute mean of the correlation between (MF14) does not have any effect on the model's output for MCAR and MNAR data mechanisms. The mean over the feature values (MF8) was not observed in the rules for MCAR and MNAR data mechanisms. The mean over

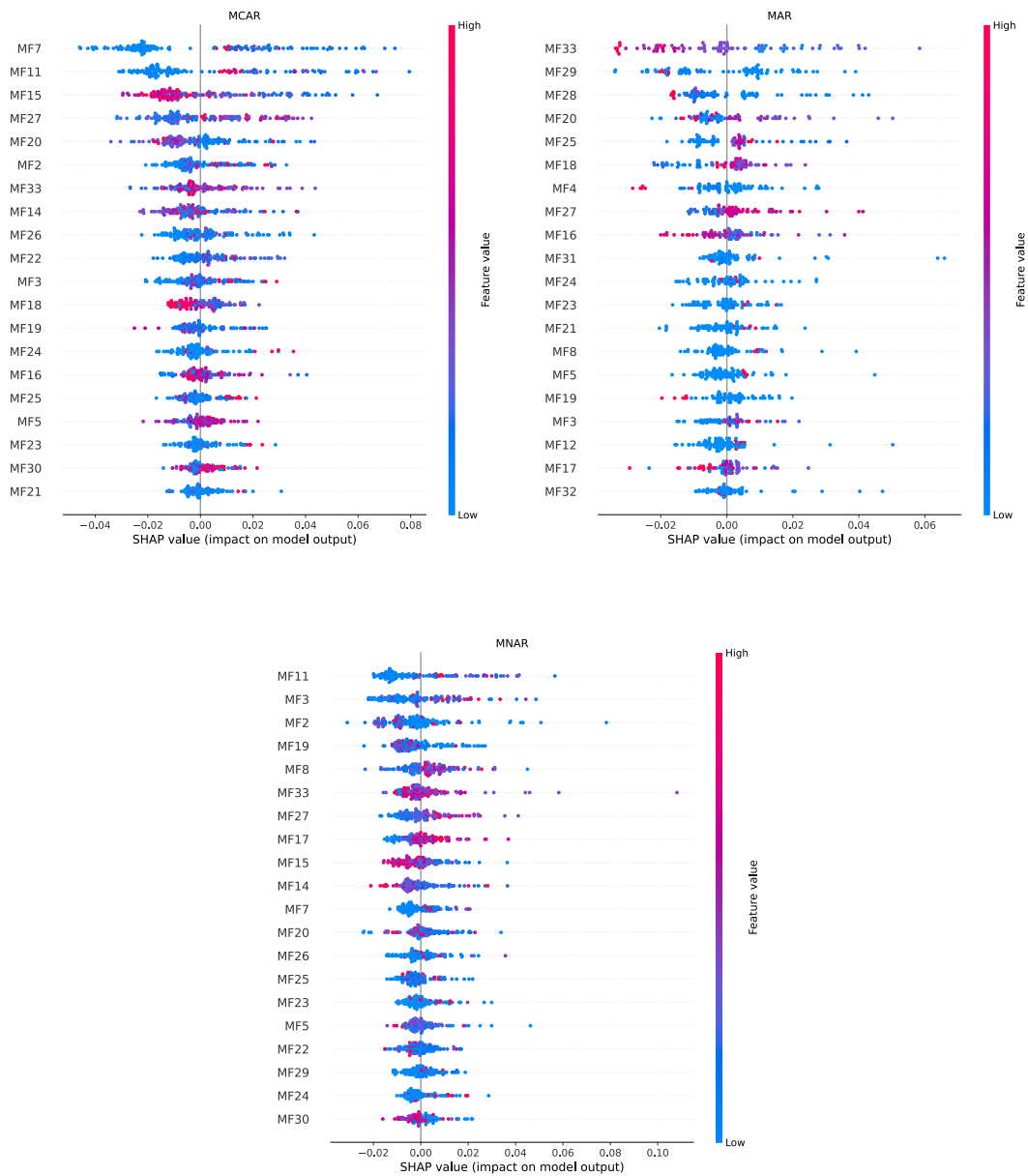


Fig. 2 Beeswarm summary plot of the *Top*₂₀ meta-features using SHAP values of input meta-features on the predictive performance of the Random Forest recommendation model for the MLP/LightGBM class according to the three missing data mechanisms. The x-axis denotes the

SHAP value, with each row corresponding to a meta-feature. Each dot represents the Shapley value of the meta-feature for one instance where red dots show higher feature values and blue dots show lower ones

the target values (MF9), the average kurtosis values of features (MF26), and the average of first quartile values of target values (MF32) were not observed in the rule list for this sample according to the MCAR data mechanism. Analyzing the common features of these meta-features, it is observed that they are related to the target value or the label column. The average of the squared residuals from the linear regression (MF20), the nonlinearity of a linear regressor measure

(MF21), and the average of the last quartile values of target values (MF31) were not listed among the important features of the MNAR data mechanism. These results align with the findings shown in the feature importance figures in Figure 4. As seen in Figure 4, these meta-features do not appear in the *Top*₂₀ most important feature lists, except for the absolute mean of the correlation between (MF14).

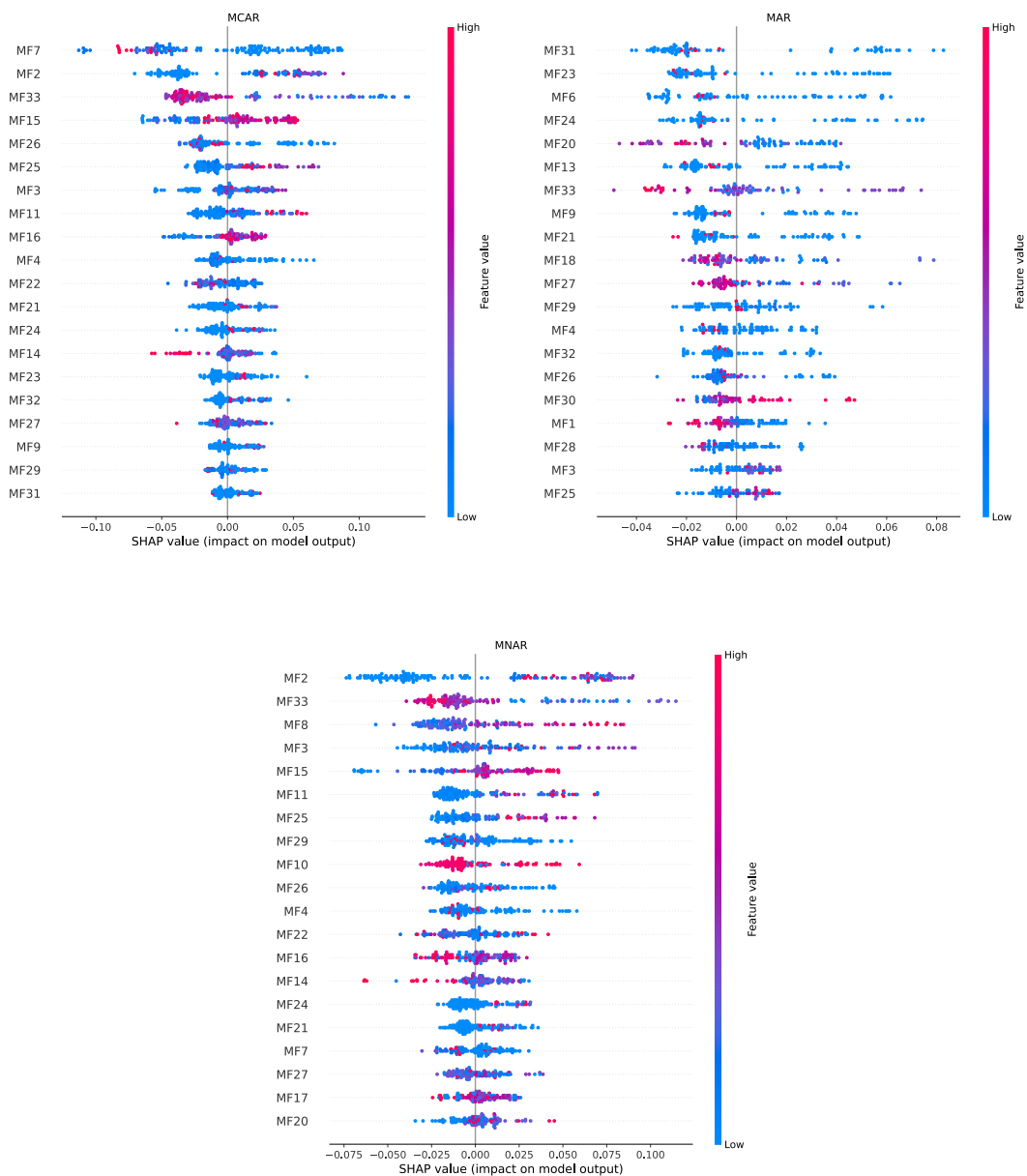


Fig. 3 Beeswarm summary plot of the *Top*₂₀ meta-features using SHAP values of input meta-features on the predictive performance of the Random Forest recommendation model for the No imputation/LightGBM class according to the three missing data mechanisms. The x-axis

denotes the SHAP value, with each row corresponding to a meta-feature. Each dot represents the Shapley value of the meta-feature for one instance where red dots show higher feature values and blue dots show lower ones

6 Discussion

In this section, we discuss the extended MetaLIRS framework for selecting the imputer+regressor pair and develop by means of a meta-features-based recommendation model. In the following, we present a few of the most significant findings.

6.1 Interaction between imputer and regressor

We investigated the interaction between imputer and regressor based on RMSE values of regressors to build a recommendation model using meta-learning approaches (see Tables 4a, 4b and 4c). The list of *Top*₅ imputer/regressor pairs varied across different missing data mechanisms. How-

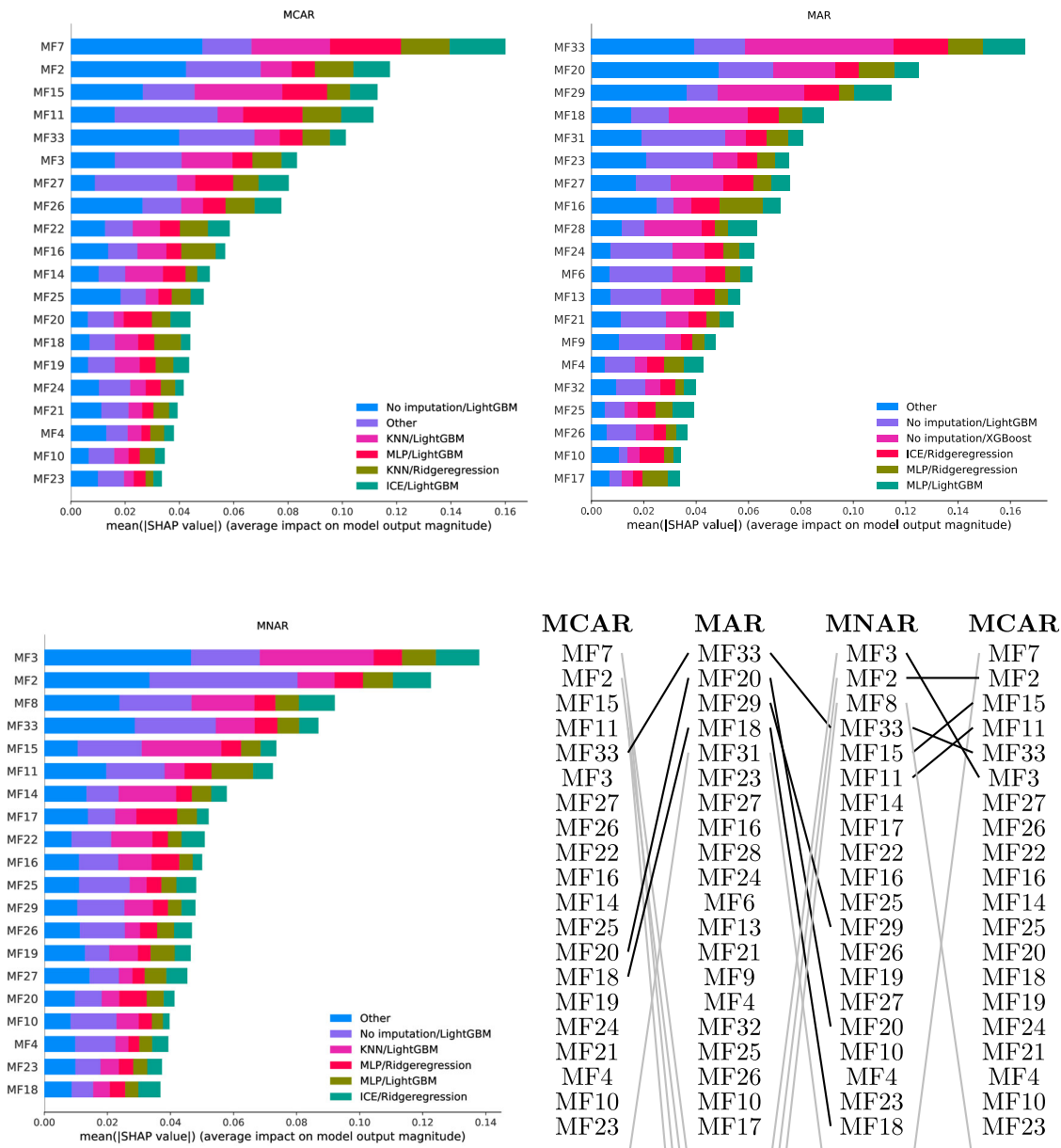


Fig. 4 Meta-feature importance evaluated using the mean SHAP values aggregated over all classes. On the bottom right a pairwise comparison between the rankings (MCAR ranking repeated on the right to show all

three pairs). Black lines relate the positions of the same meta-feature in the pair; gray lines denote a rank below the *Top*₂₀

ever, using LightGBM without any imputation method was the common method that gave the best results for all three missing data mechanisms. This is due to the LightGBM algorithm's ability to use the block propagation approach to handle missing data. Thanks to this approach, a splitting is made only on the non-missing values; missing values are directed to one side as a block, and this direction is performed in the direction of the branch that minimizes the error [74].

6.2 Performance and explainability of recommendation models

Random Forest has been chosen for the recommendation model, which is expected to perform well and can be explained well in this framework.

The performance of the models we present in Table 5 is compared with various AutoML tools and the GPT-4

Table 8 Aggregated decision rules for 35th sample based on different missing data mechanisms: missing completely at random (MCAR) and missing at random (MAR) represented from left to right in the figures. Outputs of predictions of 35th sample: For MCAR: ICE/LightGBM, For MAR: MLP/LightGBM

MF	Higher	Smaller	MF	Higher	Smaller
1	>0.20	–	1	–	–
2	>20.10	≤ 906.10	2	–	≤ 4236.5
3	>1981.5	≤	3	>949.5	–
4	–	≤ 0.07	4	>0.00	≤ 0.01
5	–	≤ 0.99	5	–	≤ 1.01
6	–	–	6	–	≤ 0.85
7	>10.00	–	7	–	≤ 7.5
8	–	≤ 0.00	8	–	≤ 0.00
9	>7.31	≤ 104.40	9	–	–
10	>	≤ –2.38	10	–	–
11	>20.79	≤ 31.56	11	–	≤ 6.41
12	–	≤ 0.98	12	–	–
13	–	–	13	–	≤ 0.72
14	–	–	14	–	–
15	>0.26	≤ 0.30	15	–	–
16	>0.30	–	16	>0.21	–
17	>0.37	–	17	>0.30	–
18	–	≤ –0.31	18	>–0.39	–
19	–	≤ 0.27	19	>0.48	–
20	>0.02	≤ 0.07	20	–	≤ 0.01
21	>8.55	–	21	–	≤ 0.03
22	–	≤ 3.09	22	–	≤ 2.73
23	–	≤ 54716468.0	23	>4.48	≤ 0.05
24	>73.81	≤ 2598702.67	24	>0.00	≤ 0.04
25	>200.79	≤ 1065.45	25	>103.14	≤ 788.92
26	–	≤ 64.92	26	–	≤ 0.56
27	>1.28	–	27	>–0.06	≤ 0.44
28	–	≤ 0.19	28	–	≤ –0.49
29	>–1.28	≤ 16.63	29	>0.38	≤ 43.33
30	>0.37	≤ 0.37	30	–	≤ 0.32
31	–	–	31	–	≤ 2.45
32	>177210.0	≤ 177210.0	32	–	–
33	>–0.60	≤ –0.44	33	>–0.66	≤ –0.62

model. GPT-4, while not performing as well as other tools in most cases, interestingly demonstrates good performance in certain situations. However, since ChatGPT is a large language-based black-box model, its decision-making process is often unclear. It cannot be said that it showed a good performance in terms of transparency and performance. In addition, when we asked questions similar to ChatGPT in the same time period, it was observed that it made similar predictions even for different datasets. Another disadvantage of ChatGPT-4 is that a different recommendation can be expected when making the same recommendation request.

These recommendation models represent the classification algorithms used to determine the best imputer/regressor pair based on meta-features. Therefore, we evaluated the performance of these models by using the metrics of the classification algorithm. Performances also change based on missing data mechanisms (see Table 6). Although the simulations of MCAR and MNAR data mechanisms are very different from each other, it can be observed that this difference does not influence the selection of imputer/regressor much since real-world patterns are probably more intricate and complex than those produced by simulation. Interest-

ingly, the distribution of feature importance changes based on missing data mechanisms. For the MNAR (Missing Not At Random) data mechanism, it has been observed that important features are often those that are unaffected by missing data generation, such as the number of unique values in the target (MF2) and the number of trained instances in the dataset (MF3). Based on these findings, meta-features with basic information have a greater impact on the recommendation model decisions than meta-features with values that are more prone to change.

7 Limitations

We propose a recommendation model for datasets with important missing data mechanisms (MCAR, MAR, and MNAR) and datasets with various missing data levels to ensure robustness. Despite the achievements, this study has some limitations:

- Each dataset is mapped to a single imputer + regressor pair. Some pairs have shown success in only a very small number of cases. Pairs below a certain threshold are aggregated as “other”. When the recommendation model recommends the “other” option, the optimal imputer + regressor pair cannot be reached directly. The performance of the suboptimal imputer + regressor pair, which shows the best performance for more than one under “other”, is calculated. For this reason, it is not guaranteed that the optimal imputer + regressor pair will be obtained when the recommendation model recommends the “other” option.
- Our recommendation model is designed for and limited to tabular datasets. Datasets with image, text or time series data are beyond the scope of this study.
- The recommendation model chooses the best imputer-regressor pair based on a single performance measure. In particular, in creating recommendation models, priority is given to prediction accuracy metrics such as RMSE, ignoring the execution time of different regressors and the accuracy of imputation methods. While this approach aims to maximize prediction performance, it does not take into account situations where users may prefer faster but potentially less accurate regressors. Therefore, the model may not be suitable for applications where computational efficiency or imputation quality is more important than prediction accuracy. Advanced weighted performance metrics can be developed for future work where critical metrics such as imputation performance, regressor performance, and execution time are important.

8 Conclusion

We answer our three main research questions in this section. The findings (see Figure 7) related to RQ1 (How can we design a framework for the recommendation to select a (near) optimal imputer/regressor pair?) demonstrate that a proof-of-concept Random Forest-based recommendation model constructed with MetaLIRS yields promising results for data scientist. These findings are based on extensive experimental training data from various meta-features, datasets, imputation and regression methods, and missingness scenarios. Although TPOT and Auto-sklearn have more regressors and perform hyperparameter optimization, our MetaLIRS-based recommendation model achieves similar performance orders of magnitude faster and uses fewer resources. Furthermore, it has been shown that non-optimal recommendations for imputer+regressor lead to only marginal loss in the performance of the model that is trained according to the suboptimal recommendation. In most cases, it also outperforms large language models such as ChatGPT (see Table 5). In the future, performance comparisons can be made by investing in exhaustive experiments with additional regressors and imputers, thus enriching the meta-database of the framework. Furthermore, comparisons with other big language models can be an interesting study.

To answer RQ2 (How can the performance of the recommendation (meta) model change based on different missing data mechanisms?) We create three recommendation models based on MCAR, MAR, and MNAR data mechanisms. Although MCAR and MNAR have similar performance, with 63–64% accuracy, respectively, the model of the MAR data mechanism has shown better performance with 67% accuracy. This model showed results that were very close to the best-performing pair, even in the case of high missing data (see Table 18). Interesting future work would be to apply feature selection to the recommendation model for the purpose of an even smaller, faster, more accurate, and better explainable recommendation model.

To address RQ3 (How can global and local explanations be obtained for selecting an imputer/regressor pair?) To obtain a more explainable recommendation model, we applied the SHAP method for global explanations and extracted decision rules for local explanations. According to the SHAP method, no common meta-features were observed when looking at the Top5 important features (except the average of the first quartile values of the features MF33). Although we emulated MCAR and MNAR data mechanisms, the recommendation model has similar results, and explainability can change based on missing data mechanisms.

The explainability support in our MetaLIRS can help data analysts investigate the important meta-features of the recommendations and understand how meta-feature importance is different for the different missing data mechanisms.

In this paper, the MetaLIRS framework was extended with an explainability component. In this way, data analysts can determine which meta-features are important for their models and reach local explanations through decision rules. These rules increase explainability, allowing users to understand better the model's decision-making processes, how it functions in specific circumstances and the reasoning behind its conclusions, while also providing insight into the conditions under which specific regression or imputation methods work well or poorly. Data analysts can create more trustworthy and transparent models in this way.

Appendix A Table of Abbreviations

Appendix B Pseudocode of the methodology

Algorithm 1: Create meta-database

Output: META_DB: datasets $\times$ missing data levels $\times$ missing data mechanisms $\times$ imputers $\times$ regressors

```

1 function SETMETADB (datasets, missingDataLevels, imputers,
  regressors) :
2   meta_db = [ ];
3   foreach dataset ds in datasets do
4     (train_ds, test_ds) = Split (ds, 75%, 25%);
5     foreach missing data level ml in missingDataLevels do
6       foreach missing mechanism mm in {'MCAR', 'MAR',
        'MNAR'} do
7         perf = [ ]; mf_ds = [ ];
8         missing_ds = GenerateMissingData
          (train_ds, ml, mm);
9         mf_ds = ComputeMetafeatures
          (missing_ds);
10        foreach imputer imp in imputers do
11          imputed_ds = ImputeMissingData (imp,
            missing_ds);
12          foreach regressor reg in regressors do
13            model = TrainModel (reg, imputed_ds);
14            rmse = EvaluateModel (model,
              test_ds);
15            perf[imp, reg] = rmse;
16        sort perf by rmse;
17        foreach imputer imp in imputers do
18          foreach regressor reg in regressors do
19            rank = index of (imp, reg) in perf;
20            append ds, ml, mm, imp, reg, mf_ds,
              perf[imp, reg], rank to meta_db;
21 return meta_db;

```

Abbreviation	Description
MCAR	Missing Completely-At-Random
MAR	Missing-At-Random
MNAR	Missing Not-At-Random
ML	Machine Learning
k-NN	k-Nearest Neighbors
VAEs	Variational Autoencoders
GAIN	Generative Adversarial Imputation Networks
MIA	Missing Incorporated in Attributes
XGBoost	eXtreme Gradient Boosting
LightGBM	Light Gradient Boosting Machine
AutoML	Automated Machine Learning
TPOT	Tree-based Pipeline Optimization Tool
JADBio	Just Add Data Bio
XAI	Explainable AI
ELEM2	Examples Modules Version 2
imp.	Imputers
reg.	Regressors
TT imp.	The trial and error method for imputers
TT reg.	The trial and error method for regressors
Mecha.	missing data mechanism
AM	Adversarially Missing
RF	Random Forest
SVM	Support Vector Machine
SVR	Support Vector Regression
MMI	Mean/Mode Imputation
CP	Ceteris Paribus
ICE	Individual Conditional Expectation
CNN	Convolutional Neural Network
ResNet	Residual Neural Network
FT-Transformer	Feature Tokenizer + Transformer
GE	General Meta-Features
ST	Statistical Meta-Features
CO	Data Complexity-Based Meta-Features
DS	Descriptive Statistic
ICE imputation	Imputation by Chained Equations
MICE	Multivariate Imputation by Chained Equation
NN	Neural Network
GOSS	Gradient-based One-Sided Sampling
EFB	Exclusive Feature Bundling
LOOCV	Leave-One-Out Cross-Validation
LLM	Large Language Model based models
SHAP	SHapley Additive exPlanations
NLP	Natural Language Processing
GPT	Generative Pre-trained Transformer
MLP	Multilayer Perceptron

Appendix C Data characteristics

We rescaled the columns to the same range using standard scaling. Basic data characteristics are shown in Table 9. [GO], [NA], [MH], [H8], [PU], [CPU], [BN], [SC], [PP], [Q], and [LO] were removed for the MAR data mechanism model.

C.1 Label distributions

In Table 10, 12, and 14 the label setting for the recommendation models according to MCAR, MAR, and MNAR data mechanisms are shown. Since the classes did not have a balanced distribution, the classes with a frequency below seven

Algorithm 2: Construct training set for metalearning

Output: Dataset for training the recommendation model

```

1 function CONSTRUCTRECOMMENDATIONDATASET (mm, meta_db) :
2   recommendation_ds = [ ];
3   foreach row in meta_db do
4     label = DetermineLabel (row.imp, row.reg);
5     if row.rank == 1 and row.mm == mm then
6       append row.mf_ds, label to recommendation_ds;
7   return recommendation_ds

```

Algorithm 3: Train recommendation model per missing mechanism

Output: Model predicting best imputer/regressor pair given meta-features of a dataset

```

1 function PERFORMMETALEARNING (mm, meta_db) :
2   recommendation_ds =
3     CONSTRUCTRECOMMENDATIONDATASET (mm, meta_db);
4   return TrainModel ('RandomForest', recommendation_ds);
5   trained_model_MCAR = PERFORMMETALEARNING ('MCAR', meta_db);
6   trained_model_MAR = PERFORMMETALEARNING ('MAR', meta_db);
7   trained_model_MNAR = PERFORMMETALEARNING ('MNAR', meta_db);

```

Algorithm 4: Evaluate recommendation model

Output: Estimated accuracy

```

1 function EVALUATEPERFORMANCE (mm, meta_db) :
2   recommendation_ds =
3     CONSTRUCTRECOMMENDATIONDATASET (mm, meta_db);
4   nr_correct_predictions = 0;
5   foreach row in recommendation_ds do
6     train_ds = recommendation_ds except row;
7     m = TrainModel ('RandomForest', train_ds);
8     if ApplyModel (m, row.mf_ds) == row.label then
9       nr_correct_predictions++;
10  return nr_correct_predictions / | recommendation_ds |;

```

were grouped before training the recommendation model, and the model was trained with six classes (Table 11, 13, and 15).

C.2 Experimental Set Up

For mean, k -NN, ICE, and MLP imputer, we utilized the Scikit-learn library. ICE and MLP require a simple initial imputer for the first iteration, for which we use mean imputation.

For k -NN imputer, the Euclidean distance determines the neighbors. We used original paper approaches for GAIN [11] and VAE [10] imputation. To obtain reproducible results, we used the experimental setting of Jager et al. [56] for the

Table 9 Dataset characteristics

Repository	Name	Instances	Features	Source
Open ML ID:191	Wisconsin [Wisc.]	194	32	[75]
Open ML ID:8	Liver Disorder [LD]	345	5	[76]
Open ML ID:44994	Cars [CA]	804	17	[77]
UCI	Physicochemical Properties [CASP]	45730	9	[78]
Open ML ID:209	Quake [Q]	2178	3	[79]
Open ML ID:218	House8L [H8]	22784	8	[80]
Open ML ID:23515	Sulfur [SU]	10081	6	[81]
Open ML ID:42636	Long [LO]	4477	19	[50]
Open ML ID:673	Chscase _{census2} [CH]	400	7	[82]
Open ML ID:40691	Wine quality red [WQR]	1599	11	[72]
Open ML ID:40498	Wine quality white [WQW]	4898	11	[72]
Open ML ID:189	Kin8nm [KIN8]	8192	8	[83]
Open ML ID:225	Puma [PU]	8192	8	[80]
Open ML ID:198	Delta Elevators [Delta]	9517	6	[80]
Open ML ID:42545	Stock _{fardamento02} [ST]	6277	6	[50]
Open ML ID:44957	Airfoil _{selfnoise} [AS]	1503	5	[84]
Open ML ID:1200	BNG (stock) [BN]	59049	9	[50]
Open ML ID:4353	Concrete [CO]	1030	8	[50]
Open ML ID:44978	CPU _{Activity} [CPU]	8192	21	[50]
Open ML ID:44965	Geographical _{origin} of music [GO]	1059	116	[85]
Open ML ID:537	California _{Housing} [CP]	20640	8	[86]
Open ML ID:43093	Miami _{Housing} [MH]	13932	16	[50]
Open ML ID:44969	Naval [NA]	11934	14	[87]
Open ML ID:44969	Qsar fish [Qsar]	987	6	[88]
Open ML ID:507	Space-ga [Space.]	3107	6	[86]
UCI	Super Conductivity [SC]	21263	81	[89]

training phase: a batch size of 64 and a maximum of 50 epochs.

Regarding regressors, We used the scikit-learn library [58] and its default settings for implementing the MLP [59], AdaBoost [62], SVR [60], and Ridge regressor [90]; for XGBoost [14] and LightGBM [91] regressors, we chose the original paper references. Used hyperparameters are shown in Table 16.

In relation to baseline recommendation model settings, for Auto-sklearn, we used the 0.15.0 version. Each model's training and evaluation time are limited to 30 seconds, and a dataset's overall processing time is limited to around 600 seconds. For the TPOT framework, we used 0.12.2 version.

For the ChatGPT model, we used the GPT-4 (gpt-4o-2024-08-06) version. We have some key steps to utilize the ChatGPT model:

Data preparation: Synthetic missing datasets were created by introducing missing data into a complete dataset at two levels, 10% and 70%, using three mechanisms (MCAR, MAR, and MNAR).

Methods recommendation: We asked the following questions in the prompt:

-I have the following imputers: ["KNN", "MEAN", "ICE", "MLP", "GAIN", "VAE", "no imputation with LightGBM", "no imputation with XGBoost"], and the following regressors: ["Multi-Layer Perceptron regressor", "Support Vector Regressor", "AdaBoostRegressor", "XGBoost", "LightGBM", "RidgeRegression"]. Can you suggest the best imputer-regressor combination for this dataset?

Gradient-boosted regression trees (GBRTs) are frequently used in tabular regression problems [92]. The most well-known examples are the XGBoost and LightGBM algorithms. We divided the dataset into a 75% training subset and a 25% testing subset. We used imputation methods after train and test splitting to prevent data leakage. All experiments were performed independently five times using different random seeds.

Algorithm 5: Actual use of the recommendation model: getting a recommendation for an Imputer+Regressor pair

Output: Recommended imputer+regressor pair

```

1 function GetRecommendation(ds, mm):
    // To obtain a recommendation,
    // meta-features need to be extracted
    // from the data set, which are then
    // given to the recommendation model
2 mf_ds = ComputeMetafeatures(ds);
3 if mm == 'MCAR' then
    // Use the trained model for MCAR
    // (Missing Completely at Random)
4 recommendation =
    ApplyModel(trained_model_MCAR, mf_ds);
5 else if mm == 'MAR' then
    // Use the trained model for MAR
    // (Missing at Random)
6 recommendation = ApplyModel(trained_model_MAR,
    mf_ds);
7 else if mm == 'MNAR' then
    // Use the trained model for MNAR
    // (Missing Not at Random)
8 recommendation =
    ApplyModel(trained_model_MNAR, mf_ds);
9 return recommendation;
```

Algorithm 6: Actual use of the recommendation model: getting global and local explanations

Output: Global and local explanations

```

1 function GlobalExplanation(model, datasets):
    // Aggregate SHAP values over all data
    // sets
2 SHAPlist = [ ];
3 foreach dataset ds in datasets do
4 SHAPds = average over CalculateSHAPValues
    (model, ds) based on leave-one-out approach;
5 append SHAPds to SHAPlist;
6 return average over SHAPlist;
7 function LocalExplanation(model, ds):
    // DecisionRules is a mapping of
    // meta-features to boundaries
8 DecisionRules = [ ];
9 foreach tree t in random forest model do
10 path = decision path of ApplyModel(t, ds);
11 foreach node n in path do
12 mf = meta-feature of node node;
13 update boundaries of DecisionRules[mf] with
    boundaries of mf in node node;
14 return DecisionRules;
```

Table 10 Distribution of classes used in the recommendation model according to MCAR data mechanism. A ranking was performed based on the performance of the regression models of each data set, which was characterized according to missing levels, and the imputer+regressor pair with the best rank value was selected

Regressor+Imputer	Frequency
LightGBM+No imputation	44
LightGBM+KNN	19
LightGBM+MLP	19
RidgeRegression+KNN	14
LightGBM+ICE	13
RidgeRegression+MLP	6
XGBoost+No imputation	6
RidgeRegression+ICE	5
LightGBM+GAIN	4
LightGBM+VAE	3
XGBoost+VAE	3
AdaBoostRegressor+MLP	3
RidgeRegression+GAIN	2
MLPRegressor+KNN	2
RidgeRegression+VAE	2
SVRegression+MLP	2
XGBoost+ICE	1
AdaBoostRegressor+KNN	1
AdaBoostRegressor+ICE	1
XGBoost+MLP	1
XGBoost+MEAN	1
LightGBM+MEAN	1
MLPRegressor+MEAN	1
XGBoost+KNN	1
SVRegression+ICE	1

Table 11 Distribution of combined classes used in the classification model

Regressor+Imputer	Frequency
LightGBM+No imputation	44
LightGBM+KNN	19
LightGBM+MLP	19
Ridge Regression+KNN	14
LightGBM+ICE	13
Other	47

Table 12 Distribution of classes used in the recommendation model according to MAR data mechanism. A ranking was performed based on the performance of the regression models of each data set, which was characterized according to missing levels, and the imputer+regressor pair with the best rank value was selected

Regressor+Imputer	Frequency
LightGBM+No imputation	22
XGBoost+No imputation	21
RidgeRegression+ICE	8
LightGBM+MLP	7
RidgeRegression+MLP	7
LightGBM+MEAN	5
LightGBM+ICE	5
AdaBoostRegressor+ICE	4
LightGBM+KNN	3
RidgeRegression+KNN	2
RidgeRegression+VAE	2
AdaBoostRegressor+MLP	1
AdaBoostRegressor+GAIN	1
SVRegression+ICE	1
AdaBoostRegressor+VAE	1

Table 13 Distribution of combined classes used in the recommendation model

Regressor+Imputer	Frequency
LightGBM+No imputation	22
XGBoost+No imputation	21
RidgeRegression+ICE	8
LightGBM+MLP	7
RidgeRegression+MLP	7
Other	25

C.3 Recommendation model

We used the Random Forest classifier model for a recommendation and applied hyperparameter optimization using Hyperopt-Sklearn library [93], which employs Bayesian optimization for different missing data mechanisms. The hyperparameter search was run for 100 evaluations per meta-database of missing data mechanisms. We used accuracy, precision, recall, and F1 score metrics to evaluate this model. Table 17 presents the hyperparameter settings used for the recommendation models built according to the MCAR, MAR, and MNAR mechanisms.

Table 14 Distribution of classes used in the recommendation model according to MNAR data mechanism. A ranking was performed based on the performance of the regression models of each data set, which was characterized according to missing levels, and the imputer+regressor pair with the best rank value was selected.

Regressor+Imputer	Frequency
LightGBM+No imputation	42
LightGBM+KNN	22
LightGBM+MLP	16
RidgeRegression+MLP	13
RidgeRegression+ICE	9
LightGBM+ICE	8
RidgeRegression+KNN	8
LightGBM+MEAN	4
RidgeRegression+MEAN	3
XGBoost+VAE	3
LightGBM+GAIN4	3
RidgeRegression+VAE	2
RidgeRegression+GAIN	2
AdaBoostRegressor+ICE	2
XGBoost+KNN	2
SVR+ICE	2
LightGBM+VAE	2
XGBoost+No imputation	2
XGBoost+MLP	2
XGBoost+GAIN	2
MLPRegressor+ICE	1
MLPRegressor+MEAN	1
AdaBoostRegressor+MEAN	1
AdaBoostRegressor+VAE	1
AdaBoostRegressor+MLP	1
XGBoost+ICE	1
SVR+GAIN	1

Table 15 Distribution of combined classes used in the classification model.

Regressor+Imputer	Frequency
LightGBM+No imputation	42
LightGBM+KNN	22
LightGBM+MLP	16
Ridge Regression+MLP	13
Ridge Regression+ICE	9
Other	54

Table 16 Used hyperparameters for regression models

Regression model	Hyperparameters
Adaboost	Number of estimators:100.
LightGBM	Number of leaves:31.
XGBoost	Maximum depth: 3, Learning rate: 0.3.
SVR	Default.
MLP	Maximum iterations: 1000, Learning rate: 0.001
Ridge Regression	Default.

Table 17 Used hyperparameters for recommendation models

Missing Data Mechanism	Hyperparameters
MCAR	No of estimators:18, maximum depth: 18 balanced class weight, minimum samples leaf: 0.010.
MAR	No of estimators:17, maximum depth: 36 balanced class weight, min samples leaf: 0.01.
MNAR	No of estimators:29, maximum depth: 11 balanced class weight, minimum samples leaf: 0.0101

Table 18 Evaluation metrics of recommendations considering cases where the model's predictions, with a deviation of less than 5% from the optimal value, are accepted as correct. The results are shown as the mean and standard deviation over 30 different random initializations using the Leave One Out split method. "Mecha." means "missing data mechanism"

Mecha.	Accuracy	Precision	Recall	F1-score
MCAR	0.70±0.02	0.70±0.02	0.70±0.02	0.69±0.02
MAR	0.80±0.02	0.81±0.02	0.80±0.02	0.80±0.02
MNAR	0.66±0.02	0.66±0.02	0.66±0.02	0.65±0.02

Acknowledgements We would like to express our sincere gratitude to Jorn-Jan van de Beld for his valuable insights and feedback regarding the SHAP method and to Tugce Arican for her feedback on the preliminary version of the work.

Author Contributions I.B.E was involved in formal analysis, methodology, visualization, and writing (original draft). J.F. helped with data curation and software. M.P. provided input for conceptualization, methodology, supervision, and writing (review & editing), M.v.K. contributed to supervision and writing (review & editing).

Funding This work is partially supported by the Ministry of National Education of the Republic of Turkey under the scope of the 1416 Higher Education Law.

Data Availability All data used in this study were obtained from publicly accessible sources. More detailed information is provided in the manuscript.

Code Availability Github: <https://github.com/isilay123/Extended-MetaLIRS/tree/main>

Declarations

Competing interests The authors declare that they have no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Song, M., et al.: A review of integrative imputation for multi-omics datasets. *Frontiers in genetic* **11**, 570255 (2020). <https://doi.org/10.3389/fgene.2020.570255>
2. Emmanuel, T., et al.: A survey on missing data in machine learning. *Journal of Big data* **8**, 1–37 (2021). <https://doi.org/10.1186/s40537-021-00516-9>
3. Liu, M., et al.: Handling missing values in healthcare data: A systematic review of deep learning-based imputation techniques. *Artificial intelligence in medicine* **142**, 102587 (2023). <https://doi.org/10.1186/s40537-021-00516-9>
4. Sun, T., Zhu, S., Hao, R., Sun, B., Xie, J.: Traffic missing data imputation: A selective overview of temporal theories and algorithms. *Mathematics* **10**, 2544 (2022). <https://doi.org/10.3390/math10142544>
5. Schafer, J.L.: Analysis of incomplete multivariate data. Chapman & Hall/CRC (1997). <https://doi.org/10.1201/9780367803025>
6. Van Buuren, S. *Flexible imputation of missing data* (Second Edition) (2nd ed.). Chapman and Hall/CRC, (2018). <https://doi.org/10.1201/9780429492259>
7. Josse, J., Prost, N., Scornet, E. & Varoquaux, G. On the consistency of supervised learning with missing values. *arXiv preprint arXiv:1902.06931* (2019)
8. Little, R. J. & Rubin, D. B. *Statistical analysis with missing data* Vol. 793 (John Wiley & Sons, 2019)
9. Troyanskaya, O., et al.: Missing value estimation methods for dna microarrays. *Bioinformatics* **17**, 520–525 (2001). <https://doi.org/10.1093/bioinformatics/17.6.520>
10. Camino, R. D., Hammerschmidt, C. & State, R. Working with deep generative models and tabular data imputation (2020). <https://api.semanticscholar.org/CorpusID:221659123>
11. Yoon, J., Jordon, J. & van der Schaar, M. Dy, J. & Krause, A. (eds) *GAIN: Missing data imputation using generative adversarial nets*. (eds Dy, J. & Krause, A.) *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of*

- Machine Learning Research*, 5689–5698 (PMLR, 2018). <https://proceedings.mlr.press/v80/yooni18a.html>
12. Quinlan, J. R. *C4. 5: programs for machine learning* (Elsevier, 2014)
 13. Ke, G., et al.: LightGBM: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* **30**, 3149–3157 (2017)
 14. Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System KDD '16 (Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2939672.2939785>
 15. Bertsimas, D., Delarue, A. & Pauphilet, J. Simple imputation rules for prediction with missing data: Theoretical guarantees vs. empirical performance. *Transactions on Machine Learning Research* (2024). <https://openreview.net/forum?id=IKH5ziX9dk>
 16. Le Morvan, M., Josse, J., Scornet, E., Varoquaux, G.: What's a good imputation to predict with missing values? *Advances in Neural Information Processing Systems* **34**, 11530–11540 (2021)
 17. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* **1**, 67–82 (1997)
 18. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a tree-based pipeline optimization tool for automating data science. *Proceedings of the Genetic and Evolutionary Computation Conference* **2016**, 485–492 (2016)
 19. Feurer, M. *et al.* Efficient and robust automated machine learning. *Advances in neural information processing systems* **28** (2015)
 20. Cloud, G. Cloud automl custom machine learning models. <https://cloud.google.com/automl>
 21. Tsamardinos, I., et al.: Just add data: automated predictive modeling for knowledge discovery and feature selection. *NPJ precision oncology* **6**, 38 (2022). <https://doi.org/10.1038/s41698-022-00274-8>
 22. Erickson, N. *et al.* Autoglun-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* (2020)
 23. Zagatti, F. R. *et al.* MetaPrep: Data preparation pipelines recommendation via meta-learning In 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA) pp. 1197–1202. IEEE. (2021)
 24. Garouani, M., et al.: Towards big industrial data mining through explainable automated machine learning. *The International Journal of Advanced Manufacturing Technology* **120**, 1169–1188 (2022). <https://doi.org/10.1007/s00170-022-08761-9>
 25. Brazdil, P., van Rijn, J.N., Soares, C., Vanschoren, J. Metalearning., for hyperparameter optimization.: In *Metalearning: Applications to Automated Machine Learning and Data Mining*, pp. 103–122. Springer International Publishing, Cham (2022)
 26. Bahrpeyma, F., Ngo, V. M., Roantree, M. & McCarren, A. A meta-learner approach to multistep-ahead time series prediction. *International Journal of Data Science and Analytics* 1–13 (2024). <https://doi.org/10.1007/s41060-024-00599-6>
 27. Rossi, A. L. D., de Leon Ferreira, A. C. P., Soares, C., De Souza, B. F. *et al.* Metastream: A meta-learning based method for periodic algorithm selection in time-changing data. *Neurocomputing* **127**, 52–64 (2014). <https://doi.org/10.1016/j.neucom.2013.05.048>
 28. Aguiar, G. J., Santana, E. J., de Carvalho, A. C. & Junior, S. B. Using meta-learning for multi-target regression. *Information Sciences* **584**, 665–684 (2022). <https://doi.org/10.1016/j.ins.2021.11.003>
 29. Rivolli, A., Garcia, L.P., Soares, C., Vanschoren, J., de Carvalho, A.C.: Meta-features for meta-learning. *Knowledge-Based Systems* **240**, 108101 (2022)
 30. Marconi, L., Matamoros Aragon, R., Epifania, F. *et al.* Discovering the unknown suggestion: a short review on explainability for recommender systems. *CEUR WORKSHOP PROCEEDINGS* **3463**, 1–15 (2023)
 31. Nauta, M., et al.: From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Computing Surveys* **55**, 1–42 (2023). <https://doi.org/10.1145/3583558>
 32. Molnar, C. *Interpretable machine learning* (Lulu. com, 2020)
 33. Baysal Erez, I., Flokstra, J., Poel, M. & van Keulen, M. MetaLIRS: Meta-learning for Imputation and Regression Selection. Paper presented at the 25th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Valencia, Spain, pp. 155–166 (2024). https://doi.org/10.1007/978-3-031-77731-8_15
 34. Pio, P. B., Rivolli, A., Carvalho, A. C. d. & Garcia, L. P. A review on preprocessing algorithm selection with meta-learning. *Knowledge and Information Systems* **66**, 1–28 (2024). <https://doi.org/10.1007/s10115-023-01970-y>
 35. Lenz, O. U., Peralta, D. & Cornelis, C. No imputation without representation. *arXiv preprint arXiv:2206.14254* (2022)
 36. Le Morvan, M., Josse, J., Moreau, T., Scornet, E., Varoquaux, G.: Neumiss networks: differentiable programming for supervised learning with missing values. *Advances in Neural Information Processing Systems* **33**, 5980–5990 (2020)
 37. Rice, J. R. The algorithm selection problem (1976)
 38. Dagan, I., Vainshtein, R., Katz, G., Rokach, L.: Automated algorithm selection using meta-learning and pre-trained deep convolution neural networks. *Information Fusion* **105**, 102210 (2024). <https://doi.org/10.1016/j.inffus.2023.102210>
 39. Lorena, A.C., Maciel, A.I., de Miranda, P.B., Costa, I.G., Prudêncio, R.B.: Data complexity meta-features for regression problems. *Machine Learning* **107**, 209–246 (2018). <https://doi.org/10.1007/s10994-017-5681-1>
 40. Liu, Q. & Hauswirth, M. A provenance meta learning framework for missing data handling methods selection. 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 0349–0358, (2020)
 41. Pimentel, B. A. & De Carvalho, A. C. A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences* **477**, 203–219 (2019). <http://dx.doi.org/10.1016/j.ins.2018.10.043>
 42. Breiman, L.: Random forests. *Machine learning* **45**, 5–32 (2001). <https://doi.org/10.1023/A:1010933404324>
 43. Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., Zeileis, A.: Conditional variable importance for random forests. *BMC bioinformatics* **9**, 1–11 (2008). <https://doi.org/10.1186/1471-2105-9-307>
 44. Strobl, C., Boulesteix, A.-L., Zeileis, A., Hothorn, T.: Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics* **8**, 1–21 (2007). <https://doi.org/10.1186/1471-2105-9-307>
 45. Wei, P., Lu, Z. & Song, J. Variable importance analysis: A comprehensive review. *Reliability Engineering & System Safety* **142**, 399–432 (2015). <http://dx.doi.org/10.1016/j.res.2015.05.018>
 46. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888* (2018)
 47. Woźnica, K. & Biecek, P. Towards explainable meta-learning. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* 505–520 (2021)
 48. Kostovska, A. *et al.* Explainable model-specific algorithm selection for multi-label classification (2022)
 49. Shao, X., et al.: Effect: Explainable framework for meta-learning in automatic classification algorithm selection. *Information Sciences* **622**, 211–234 (2023). <https://doi.org/10.1016/j.ins.2022.11.144>
 50. Vanschoren, J., van Rijn, J. N., Bischl, B. & Torgo, L. Openml: Networked science in machine learning. *SIGKDD Explorations* **15**, 49–60 (2013). <http://doi.acm.org/10.1145/2641190.2641198>

51. Dua, D. & Graff, C. UCI machine learning repository (2017). <http://archive.ics.uci.edu/ml>, Accessed: 2024-08-13
52. Fischer, S. F., Feurer, M. & Bischl, B. Openml-ctr23—a curated tabular regression benchmarking suite. *AutoML Conference 2023 (Workshop)* (2023)
53. Muzellec, B., Josse, J., Boyer, C. & Cuturi, M. III, H. D. & Singh, A. (eds) *Missing data imputation using optimal transport*. (eds III, H. D. & Singh, A.) *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119 of *Proceedings of Machine Learning Research*, 7130–7140 (PMLR, 2020). <https://proceedings.mlr.press/v119/muzellec20a.html>
54. Michie, D., Spiegelhalter, D. J., Taylor, C. C. & Campbell, J. *Machine learning, neural and statistical classification* (Ellis Horwood, 1995)
55. Pinto, F., Soares, C. & Mendes-Moreira, J. Towards automatic generation of metafeatures. *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19–22, Proceedings, Part I* 20, pp. 215–226 (2016). https://doi.org/10.1007/978-3-319-31753-3_18
56. Jäger, S., Allhorn, A., Bießmann, F.: A benchmark for data imputation methods. *Frontiers in big data* **4**, 693674 (2021). <https://doi.org/10.3389/fdata.2021.693674>
57. Van Buuren, S. & Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software* **45**, 1–67 (<https://doi.org/10.18637/jss.v045.i03>)
58. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
59. Hinton, G.E.: Connectionist learning procedures (1990). [https://doi.org/10.1016/0004-3702\(89\)90049-0](https://doi.org/10.1016/0004-3702(89)90049-0)
60. Cortes, C., Vapnik, V.: Support-vector networks. *Machine learning* **20**, 273–297 (1995). <https://doi.org/10.1007/BF00994018>
61. Shin, K.-S., Lee, T.S., Kim, H.-J.: An application of support vector machines in bankruptcy prediction model. *Expert systems with applications* **28**, 127–135 (2005). <https://doi.org/10.1016/j.eswa.2004.08.009>
62. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* **55**, 119–139 (1997). <https://doi.org/10.1006/jcss.1997.1504>
63. Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics* 1189–1232 (2001). <http://dx.doi.org/10.1214/aos/1013203451>
64. Alshboul, O., Almasabha, G., Shehadeh, A., Al-Shboul, K.: A comparative study of lightGBM, XGBoost, and GEP models in shear strength management of sfrc-sbws **61**, 106009 (2024). <https://doi.org/10.1016/j.istruc.2024.106009>
65. Hakkoum, H., Idri, A., Abnane, I.: Global and local interpretability techniques of supervised machine learning black box models for numerical medical data. *Engineering Applications of Artificial Intelligence* **131**, 107829 (2024). <https://doi.org/10.1016/j.engappai.2023.107829>
66. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems* **30** (2017)
67. Gijsbers, P., et al.: Amlb: an automl benchmark. *Journal of Machine Learning Research* **25**, 1–65 (2024)
68. Le, T.T., Fu, W., Moore, J.H.: Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics* **36**, 250–256 (2020). <https://doi.org/10.1093/bioinformatics/btz470>
69. Achiam, J. et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023)
70. Brown, T. B. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020)
71. OpenAI. Openai models (2024). <https://platform.openai.com/docs/models/o1>. Accessed: 2024-10-31
72. Cortez, P., Cerdeira, A., Almeida, F., Matos, T. & Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems* **47**, 547–553 (2009). DOI: <https://doi.org/10.24432/C56S3T>
73. Little, M. Parkinsons (2007). DOI: <https://doi.org/10.24432/C59C74>
74. Josse, J., Chen, J.M., Prost, N., Varoquaux, G., Scornet, E.: On the consistency of supervised learning with missing values. *Statistical Papers* **65**, 5447–5479 (2024). <https://doi.org/10.1007/s00362-024-01550-4>
75. Street, W.N., Mangasarian, O.L., Wolberg, W.H.: An inductive learning approach to prognostic prediction (1995). <https://doi.org/10.1016/B978-1-55860-377-6.50071-2>
76. Liver Disorders. UCI Machine Learning Repository (1990). <https://doi.org/10.24432/C54G67>
77. Kuiper, S. Introduction to multiple regression: How much is your car worth? *Journal of Statistics Education* **16** (2008). <https://doi.org/10.1080/10691898.2008.11889579>
78. Rana, P. Physicochemical Properties of Protein Tertiary Structure. UCI Machine Learning Repository (2013). DOI: <https://doi.org/10.24432/C5QW3H>
79. Simonoff, J. S. *Smoothing Methods in Statistics* (Springer-Verlag, New York, 1996). <ftp://stat.cmu.edu/datasets>
80. Torgo, L. Collection of regression datasets. Online. <http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html>. DELVE repository of data
81. Fortuna, L., Graziani, S., Rizzo, A., Xibilia, M. G. et al. *Soft sensors for monitoring and control of industrial processes* Vol. 22 (Springer, 2007). <http://dx.doi.org/10.1007/978-1-84628-480-9>
82. Chatterjee, S., Handcock, M.S., Simonoff, J.S.: A casebook for a first course in statistics and data analysis. Tech. Rep, Wiley, New York (1995)
83. Ghahramani, Z. The kin datasets (1996). <https://www.cs.toronto.edu/~delve/data/kin/desc.html>
84. Brooks, T.F., Pope, D.S., Marcolini, M.A.: Airfoil self-noise and prediction. Tech. Rep (1989)
85. Pace, R.K., Barry, R.: Quick computation of spatial autoregressive estimators. *Geographical analysis* **29**, 232–247 (1997). [https://doi.org/10.1016/S0167-7152\(96\)00140-X](https://doi.org/10.1016/S0167-7152(96)00140-X)
86. Pace, R.K., Barry, R.: Sparse spatial autoregressions. *Statistics & Probability Letters* **33**, 291–297 (1997)
87. Coraddu, A., et al.: Machine learning approaches for improving condition-based maintenance of naval propulsion plants (2016). <https://doi.org/10.1177/1475090214540874>
88. Cassotti, M., Ballabio, D., Todeschini, R., Consonni, V.: A similarity-based qsar model for predicting acute toxicity towards the fathead minnow (*pimephales promelas*). *SAR and QSAR in Environmental Research* **26**, 217–243 (2015). <https://doi.org/10.1080/1062936x.2015.1018938>
89. Hamidieh, K.: A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science* **154**, 346–354 (2018). <https://doi.org/10.1016/j.commatsci.2018.07.052>
90. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* **12**, 55–67 (1970). <https://doi.org/10.1080/00401706.1970.10488634>
91. Microsoft. LightGBM: A fast, distributed, high-performance gradient boosting (gbdt, gbdt, gbm or mart) framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks. <https://github.com/microsoft/LightGBM> (2022)
92. Januschowski, T., et al.: Forecasting with trees. *International Journal of Forecasting* **38**, 1473–1481 (2022). <https://doi.org/10.1016/j.ijforecast.2021.10.004>

93. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery* **8**, 014008 (2015). <https://doi.org/10.1088/1749-4699/8/1/014008>
94. Zou Y, An A, Huang X: Evaluation and automatic selection of methods for handling missing data. 2005 IEEE International Conference on Granular Computing 2,728–733

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.