Journal of Big Data and Artificial Intelligence

WWW.JBDAI.ORG

ISSN: 2692-7977

JBDAI Vol. 4, No. 1, 2026 DOI: 10.54116/jbdai.v4i1.64

SudokuImputer: A Best-in-Class Graph Framework for MNAR and MAR Missing Data Imputation

Vivek Nandhan Kanpa
University College Dublin
vivek.kanpa@icahn.mssm.edu

Riddhishrree Badhan
Washington University
in St. Louis
b.riddhishrree@wustl.edu

Vineeth Kanpa
Northeastern University
kanpa.vi@northeastern.edu

ABSTRACT

Missing data are a ubiquitous challenge when training unbiased, high confidence machine learning (ML) models on real-world data. To help address this issue, we propose SudokuImputer v1.0.0, a novel graph-based framework, to estimate missing values in an iterative, uncertainty aware process. The behavior of SudokuImputer is assessed across numerous design hyperparameters, including multiple network centrality methods, feature prioritization modes, statistical associations and pairwise availability ratios for edge weight assignment, and partner node proportions. We benchmark SudokuImputer against point-value imputations, MICE, kNN, matrix factorization, and SoftImpute. SudokuImputer achieves best-in-class RMSE on MAR (mean rank = 1.7) and MNAR (mean rank = 1.8) missing data across most missingness proportions from ten percent to fifty percent in three experimental benchmarks. SudokuImputer is sensitive to dataset dimensionality, and optimizing algorithmic runtime remains an unresolved challenge. Future work should evaluate SudokuImputer on mixed-data benchmarks and seek to iterate on the foundational graph framework laid here.

Keywords *imputation*, *constraint satisfaction*, *data preprocessing*, *graph network*, *MNAR*, *MAR*, *benchmarking*.

1. Introduction

Large, high-quality datasets play an instrumental role in training effective machine learning (ML) models and artificial intelligence (AI) systems. Specifically, achieving state-of-the-art (SOTA) performance with deep learning networks is dependent on extensive, heterogeneous training data to match the learning capacity of large models

(Hestness et al. 2017; Kaplan et al. 2020; Hoffmann et al. 2026). Aggregating and cleaning datasets is a non-trivial and ubiquitous challenge across AI/ML domains, due to the widespread presence of missing feature data in real-world datasets (Kang 2013; Zhang 2016; Schouten and Vink 2018; Misztal 2019). Missing data can introduce bias, reduce model performance, and compromise the generalizability of predictions if not properly handled.

It has been shown that low data quality, including incomplete data, can lead to revenue losses from 8% to 12% for service organizations (Prasad 2024). One example of its repercussions can be seen in the context of public health. Vaccination databases often have incomplete entries for various reasons such as data entry errors and difference protocols between clinics. During outbreaks of diseases, missing vaccination fields cause underestimated risk zones, delaying appropriate responses (Brown 2025). In another instance, cameras and sensors on roads often drop frames. These missing readings can break traffic-flow prediction models and cause reduced accuracy in navigation systems. Imputation algorithms mitigate these challenges by replacing missing data with numerical estimates (Sun et al. 2022).

2. Literature Review

2.1 Classes of Data Missingness

Rubin establishes three classes of missing data mechanisms, delineated by the implicit biases driving their missingness (Rubin 1976). Missing Completely at Random (MCAR) describes cases where the probability of an absent data point is independent of both observed and unobserved data. Missing at Random (MAR) describes data whose probability of being missing is dependent only on observed data and is independent of absent data. Finally, Missing Not at Random (MNAR) describes data whose missingness is dependent on unobserved data, but is independent of observed data (Table 1) (Heymans and Twisk 2022). Discarding entries with missing values can severely reduce a dataset's size and introduce bias, especially in MAR or MNAR scenarios. Therefore, understanding the causal probabilities that drive data missingness is a crucial consideration when identifying and designing unbiased approaches to approximate missing data.

2.2 A Brief Overview of Commonly Used Imputation Methods

Imputation is the process of inferring synthetic estimates for missing values (Table 2) (Heymans and Twisk 2022). Some of the simplest imputation strategies include case substitution and point-value imputation, where missing values are filled with either a randomly selected feature value or the mean, median, or mode of the feature data

Table 1: Table characterizing and comparing the three classes of data missingness: MCAR, MNAR, and MAR.

Missingness Type	Definition	Key Assumptions	Bias Implications	Handling Strategies
MCAR (Missing Completely at Random)	Missingness is independent of both observed and unobserved data.	No systematic relationship between missingness and data values.	No bias introduced; loss of efficiency only (smaller sample size).	Complete-case analysis, simple imputation (mean/median), advanced methods optional.
MAR (Missing at Random)	Missingness depends only on observed data (not on the missing values).	Missingness can be explained fully by variables that are present.	Can bias results if not modeled correctly; unbiased if the missingness mechanism is accounted for.	Multiple imputation, maximum likelihood, model-based imputation.
MNAR (Missing Not at Random)	Missingness depends only on unobserved values; independent of observed values.	Missingness mechanism not captured by observed covariates.	Highest risk of bias; standard methods may fail.	Sensitivity analysis, explicit modeling of missingness process, specialized algorithms.

Table 2: Characterization and comparison of various numerical imputation strategies.

Method	Mechanism	Assumptions	Advantages	Limitations
Point Value Imputation	Replace missing values with feature mean, median, or mode	Missingness is random; central tendency adequately represents data	Extremely fast; preserves dataset size; easy to implement	Distorts variance and correlations; underestimates uncertainty
SoftImpute (SVD-based)	Low-rank matrix completion via iterative soft-thresholded SVD	Data lies near a low-rank structure	Captures global feature relationships	Sensitive to rank choice; struggles with nonlinear interactions
Matrix Factorization (e.g., NMF, PMF)	Factorize data matrix into latent factors, then reconstruct missing entries	Data approximates low-dimensional latent structure	Good for collaborative filtering and sparse data	May ignore local feature correlations; requires tuning latent dimension
kNN Imputation	Replace missing values with weighted average of neighbors	Similar samples have similar values	Captures local structure; non-parametric; easy to explain	Bias in sparse/ high-dimensional data
MICE (Multiple Imputation by Chained Equations)	Iteratively imputes each variable using regression on other variables, generating multiple datasets	Relationships can be modeled linearly (extensions exist for non-linear)	Accounts for uncertainty; flexible; widely used in biomedical research	Slow on large/ high-dimensional data; requires careful model choice per variable
MissForest	Iterative imputation using Random Forests	Tree ensembles approximate relationships well	Nonlinear + mixed data support; often strong empirical performance	Computationally intensive; can be overfit with small samples
Variational Autoencoder (VAE) Imputation	Learns latent representation of data, imputes by reconstructing missing values	Data distribution is well-approximated by latent space	Captures complex relationships; scalable; probabilistic imputations	Requires tuning architecture; sensitive to training instability
GAIN (Generative Adversarial Imputation Networks)	Adversarial training; generator imputes and discriminator distinguishes real vs imputed	Missingness patterns can be modeled through adversarial training	Performs well on complex data; captures uncertainty; flexible	Training instability; sensitive to hyperparameters; computationally heavy

(Alam et al. 2023). Hot and cold deck imputation strategies are like point-value imputation, except they are preceded by partitioning the available data into clusters and then associating missing data with a certain cluster (Monard 2002). While straightforward and computationally efficient, these methods can distort data distributions and fail to preserve relationships between features, particularly in complex, high-dimensional datasets (Alam et al. 2023). ML-based imputation strategies often preserve the data's variance from the ground truth distribution better than point-based methods. For instance, the lazy learning algorithm k-Nearest Neighbors (kNN) predicts missing values by referencing the non-missing values of its closest existing neighbors across the full feature space (Halder et al. 2024). Regression imputation iterates across each feature as a linear function of all other features, allowing each column's missing values to be predicted using known values in the rest of the feature set (Zhang 2016). MissForest, an abstraction of the Random Forest algorithm, bootstraps decision trees for non-linear data imputation using pre-determined data partitioning and stop criteria (Schonlau and Zou 2020). Multivariate imputation by chained equations (MICE) begins by imputing missing values with point-value estimates in all but one feature.

Then, once that feature is imputed using a regression model, the model is refitted on newly imputed data, and another feature can be imputed using the updated model. This process repeats across all features and is done in multiple iterations until the imputed values stabilize (Mera-Gaona et al. 2021). Matrix factorization-based methods instead approximate the observed dataset as the product of two lower-rank matrices, leveraging latent structure to estimate missing values via reconstruction. SoftImpute employs iterative singular value decomposition (SVD) to a convex optimization problem by minimizing the nuclear norm (the sum of singular values) of the completed matrix. SoftImpute is reportedly less prone to overfitting than matrix factorization methods, better preserves global correlation structure, and is widely regarded as a gold-standard approach for SVD-based imputation.

2.3 Existing Methods Show Only Moderate Success on Benchmarks

Previous investigations sought to characterize the performance of numerical imputation strategies on benchmark datasets to identify the strongest general use approach. Poulos and Valle find that kNN imputation—compared to random forest imputation, random replacement, point-value imputation, and support vector machine-based methods—most strongly reduced the downstream validation error performance of supervised classification models (Poulos and Valle 2018). Jadhav, Pramod, and Ramanathan similarly find that kNN imputation achieves the lowest mean RMSE compared with predictive mean matching, Bayesian Linear Regression, non-Bayesian regression, and random sampling for three experimental datasets perturbed with ten percent to fifty percent data missingness (Jadhav et al. 2019). Jäger, Allhorn, and Bießmann benchmarked mean value, kNN, Random Forest, Variational Autoencoder (VAE), and Generative Adversarial Imputation Networks (GAIN)-based imputation strategies across 69 labelled datasets across the three data missingness mechanisms. They find that Random Forest-based imputation consistently outperforms all others, regardless of whether the imputation methods are provided complete training data for model fitting (Jäger et al. 2021). They also report that VAE and GAIN perform poorly relative to other strategies regardless of missingness type or missingness proportion. However, while they report the average relative rank of each algorithm on the imputation experiments, they do not report the average RMSE of imputations, making it unclear the degree to which one strategy is preferable over another. Furthermore, no experiments benchmarking imputation assess the runtime of their models, a crucial consideration that affects the scalability for big data and deep learning applications.

2.4 Technical Challenges of ML-Based Imputation Strategies

While ML-based imputers repeatedly achieve best-in-class performance, there exist demonstrable limitations and areas for algorithmic improvement. Eager learners, such as linear regression models and random forest regressors that drive MICE and MissForest, hold out one feature as a pseudo-target and leverage remaining feature columns as a pseudo-feature set. These algorithms train a model on rows where there are no missing data in either the pseudo-feature set or the pseudo-target, then predict the pseudo-target for rows where it is absent (and where the pseudo-feature set is complete). This creates two technical challenges.

The first challenge is how these algorithms handle missing values in the pseudo-feature set, since a complete feature set is necessary to train models and predict missing pseudo-targets. This is particularly noteworthy in MAR and MNAR-type missing data. In MAR contexts, data missing in a pseudo-target may be associated with missing values among pseudo-features, meaning these eager learners often fit models on feature subsets with fewer training instances which in turn increases the bias of the imputation estimate. In MNAR contexts, where missing values in a pseudo-target may be associated with values observed in that same pseudo-target, the distribution of pseudo-target values in the training subset may not be representative of the imputation subset, creating poorly generalized estimates.

The second challenge is with how these algorithms select the sequence of features to impute, which is an overlooked but non-trivial design. In active learning and Bayesian optimization, this challenge is resolved with acquisition functions, which model uncertainty associated with instances of unlabeled data to determine the sequence of instances the model should predict to minimize future uncertainty (Aral and Van Alstyne 2010; Larson 2017; Di Fiore et al. 2024). ML-based imputation inherently lends itself to consider an active learning-inspired framework, as imputing values in each pseudo-target will influence associations between that label and the feature set, propagating changes to future features' imputations.

Here, we present SudokuImputer, a graph-based imputation framework that leverages network analysis and ML modeling to reduce the error of estimates by selecting an optimal sequence of features whose imputation would strengthen feature-wise associations for future iterations of imputation. The roadmap for this paper is as follows: first, we discuss the theory and logic underlying the framework, then evaluate its performance across an exhaustive

design hyperparameter space, and finally benchmark its performance on three datasets induced with MCAR, MNAR, and MAR missingness at a ten percent to fifty percent proportion of missingness.

3. Methods

3.1 SudokuImputer Algorithm Design

The values missing from a feature set can be analogized to a sudoku puzzle; sudoku requires the values 0 through 9 be used only once in each row, column, or three-by-three cluster, and the player's objective is to fill in empty cells using information provided by the values of surrounding cells. From a game theory perspective, sudoku is a constraint satisfaction problem where the optimal strategy minimizes the risk of a misassignment at each step and maximizes information gain for downstream steps. As the player makes assignments, it influences their future assignments. Therefore, there exists an optimal path to sequentially assign values to cells in order to solve the puzzle with 0 backtracks or corrections.

This was the inspiration behind our novel imputation technique called SudokuImputer. SudokuImputer estimates missing values through an iterative three-stage process of (1) weighted graph construction, (2) network-based prioritization, and (3) regression-based prediction. This three-step iterative algorithm mimics uncertainty-based principles leveraged by graph Bayesian optimization, entropy minimization, and acquisition functions in active learning (Gärtner et al. 2003; Frazier 2018; Garnett 2023; Xie et al. 2025). First, the algorithm represents the data matrix as an undirected weighted graph whose nodes correspond to features and whose edges quantify pairwise associations. Edge weights can be defined (1) by measures of correlation (e.g., Spearman's ρ, Pearson's r, or coefficient of determination R^2), (2) by the proportion of rows jointly observed for each feature pair, or (3) by their product. This weighting scheme ensures that both statistical association and data availability influence the topology of the graph. Users may also impose an edge-weight threshold, thereby restricting the graph to only the strongest or most data-supported feature connections; high thresholds increase graph sparsity, which could enhance detection of strong subgraphs in multicollinear datasets or isolate islands in uncorrelated data.

A distinctive aspect of SudokuImputer is its flexibility in how the order of imputation is determined. In centrality prioritization, the sequence is governed by graph-theoretic centrality measures such as degree, betweenness, or eigenvector centrality. This strategy allows the algorithm to begin with either highly connected "hub" features (descending centrality) or with more peripheral features (ascending centrality). In contrast, utility-based prioritization directly ranks features by the proportion of observed data available for them (they possess more completeness, and thus "utility," to fit imputation models for partner features), in either ascending or descending rank order. Based on uncertainty sampling in active learning, the ascending direction is recommended for centrality-prioritized imputation. While centrality prioritization emphasizes structural importance in the correlation network, utility prioritization emphasizes practical completeness, offering an alternative that may be advantageous in settings with uneven missingness patterns.

Once centrality-based or utility-based prioritization selects the top-ranking target feature, it is imputed using its neighbors in the graph that exceed the chosen edge threshold. These neighbors serve as predictors in a multivariate regression model, which may be instantiated by linear regression, tree-based methods, or other user-specified learners. The model is trained on the subset of rows where both the target and its partner features are observed and then used to predict missing entries in the target. If some values remain missing after this process, which may be the case if selected partner nodes are also null at a given row, then they are filled using a simple median fallback, ensuring that no feature column remains incomplete.

After each column is imputed, the data matrix is updated, and the weighted graph is reconstructed to reflect the new joint availability and correlations. This cycle of graph construction, prioritization, and imputation repeats until all features have been filled. By continuously adapting the graph to incorporate newly imputed data, SudokuImputer leverages both global feature relationships and evolving data availability, yielding a dynamic and context-aware imputation process (Figure 1).

We performed two phases of experiments: (1) SudokuImputer framework optimization and hyperparameter exploration and (2) benchmarking evaluation of SudokuImputer against competing numerical imputation strategies.

3.2 Evaluating SudokuImputater Design Construction Parameters

We evaluate six design hyperparameters on SudokuImputer: centrality method (degree, betweenness, or eigenvector centrality), feature prioritization mode (centrality-based, utility-based), partner node proportion (eight percent, fifteen percent, or twenty-five percent), an eight-five percent edge threshold, and various combinations of alpha and

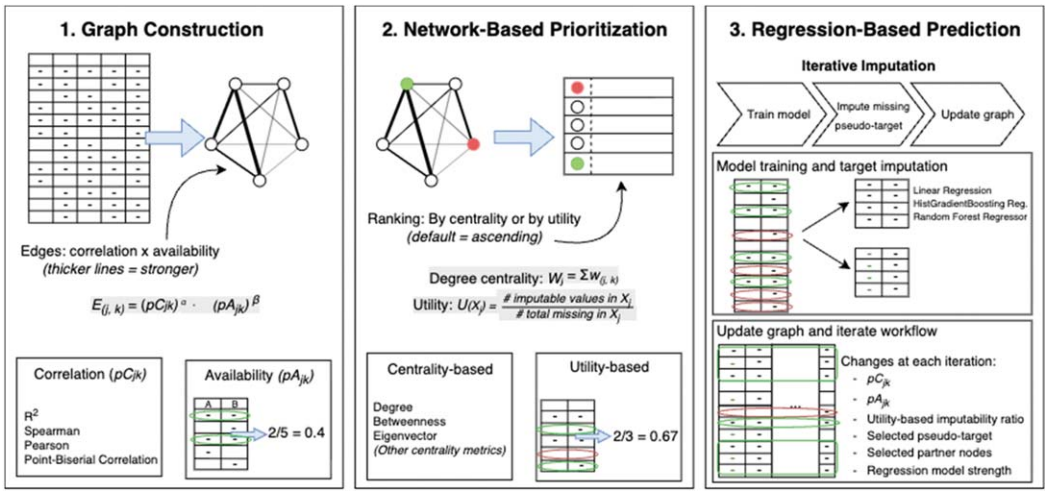


Figure 1: Flowchart depicting the SudokuImputer framework. The input dataset is iteratively processed through three steps: (1) graph construction, (2) network-based node prioritization, and (3) regression-based prediction. The framework provides a non-arbitrary method for identifying which values in each iteration will propagate better estimates for future imputations.

beta values ([1.5, 0.5], [0.5, 1.5], and [1.0, 1.0]). Alpha and beta coefficients tune the attribution of statistical association (alpha) and pairwise availability (beta) to edge weight construction; larger alpha increases the effect of feature correlation strength on edge weighting, while larger beta increases the effect of joint data availability. The electroencephalogram (EEG) eye state dataset induced with MCAR, MAR, and MNAR missingness at a ten percent proportion is used to perform a grid search across this hyperparameter space.

3.3 Benchmarking Sudoku Imputation Against SOTA Algorithms

Three classification benchmarking datasets were sourced and downloaded from the University of California Irvine Machine Learning Repository. The QSAR biodegradation dataset contains 1055 chemical compounds (rows) characterized by 41 molecular descriptors (columns) (Mansouri et al. 2013). The EEG eye state dataset contains 14 EEG measurements for 14,980 instances of eye state measurements (Roesler 2013). Finally, the Arcene mass spectrometry dataset characterizes the protein expression for 10,000 species in blood serum across 900 patients (Guyon et al. 2004). These three benchmarks were selected to evaluate datasets with varying dimensionality, size, and inductive biases. For instance, the Arcene dataset has a high feature space with few instances and given the nature of tumor biology there may be coregulation of plasma proteomic features, while the EEG dataset has many instances with few features that capture rather distinct EEG traits. We induce data missingness at varying proportions from 0.1 to 0.5 (by increments of ten percent) for each data missingness class (MCAR, MAR, and MNAR) across each dataset. A complete characterization of experimental benchmark datasets is available in Supplemental Table 1.

SudokuImputer performance was benchmarked against point-value imputation (with mean and median), MICE, kNN, matrix factorization, and SoftImpute (SVD). We evaluated model performance primarily using algorithm runtime and the RMSE between imputed and ground truth values for each column; we normalize column metrics and compute the mean across columns to aggregate a model-by-dataset performance value. R^2, Pearson correlation, and mean absolute error are also computed for each experimental dataset and available in the supplemental data tables.

4. Results

4.1 Behavior of SudokuImputer across Broad Design Parameter Space

Exploration of the hyperparameter space using Gini entropy-based feature importance demonstrated that runtime was almost entirely determined by prioritization mode (Figure 2(A)). Across MCAR, MNAR, and MAR missingness, prioritization mode accounted for 99.38 percent, 97.68 percent, and 98.87 percent of runtime variance, respectively. Centrality-based prioritization produced average runtimes of 3.48, 0.84, and 4.30 seconds across MCAR, MNAR, and MAR, while utility-based prioritization produced longer runtimes of 46.57, 11.42, and 54.32 seconds

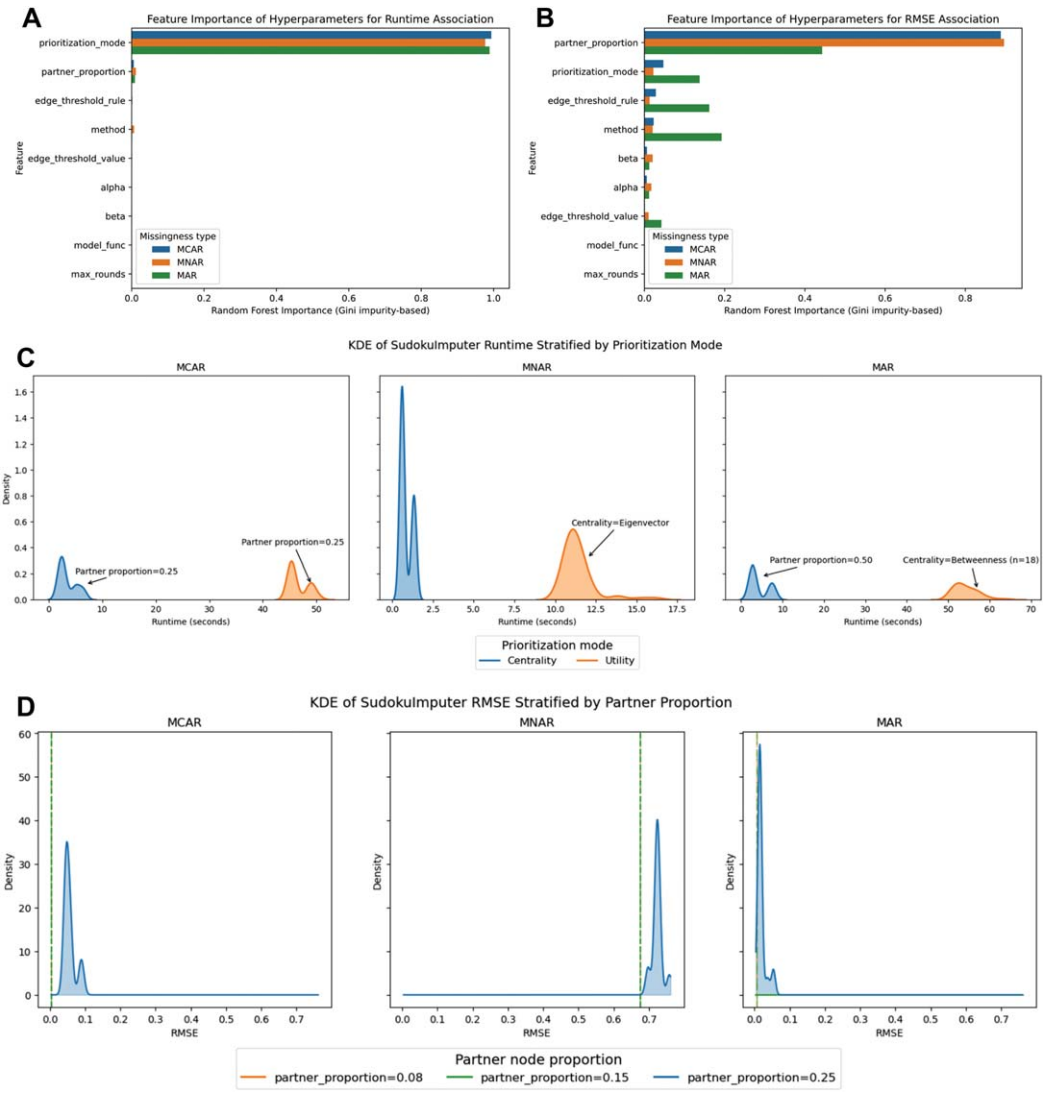


Figure 2: Gini entropy-based associations between SudokuImputer (A) runtime and (B) RMSE against algorithm hyperparameters. (C) Runtime distributions for 108 hyperparameter combinations in MCAR (left), MNAR (middle), and MAR (right) data missingness grouped by prioritization mode, the hyperparameter most strongly associated with runtime by Gini impurity. (D) RMSE distributions for 108 hyperparameter combinations grouped by imputation partner node proportion.

(Figure 2(C)). RMSE was strongly affected by numerous hyperparameters. Partner proportion showed the strongest Gini entropy association with RMSE (88.84 percent, 89.65 percent, and 44.30 percent across MCAR, MNAR, and MAR), while prioritization mode, edge threshold rule, edge threshold value, and centrality method each contributed more than two percent feature importance to RMSE in at least one missingness class (Figure 2(B)). A partner proportion of 0.25 consistently resulted in higher average RMSE (Figure 2(D)).

UMAP projections of hyperparameter combinations reveals that distinct clusters of parameters are associated with lower runtimes, while disparate coordinates in the hyperparameter space are associated with lower RMSE (Supplemental Figure 1). Furthermore, association analyses between UMAP embeddings and corresponding hyperparameter values demonstrates that the prioritization mode was a dominant contributor to model grouping (Supplemental Figure 1(C)) across MCAR and MAR missingness classes. Interestingly, UMAP embeddings for MNAR missingness are most strongly associated with the edge threshold rule (a design criteria for graph sparsity), which is either an 85th percentile cutoff or no rule (full graph). The raw RMSE and runtime of hyperparameter grid search experiments are available in Supplemental Tables 3, 4, and 5.

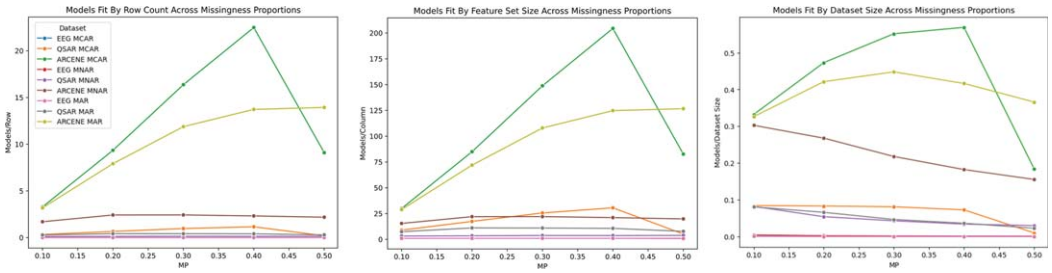


Figure 3: Model fitting behavior of SudokuImputer relative to total dataset volume, column count, and row count across missingness proportions. All other model properties (imputability ratio, fallback imputations per dataset, etc.) were trivial or insignificant; a full table is defined and available in the log files published to GitHub (see Supplemental Materials).

The number of models trained varied substantially across experimental datasets. No clear relationship was observed between the number of models trained and the missingness proportion (Supplemental Figure 2). Instead, dataset dimensions (row count, column count, and their product) were correlated with the total number of models trained (Figure 3). Dataset volume was therefore a more consistent driver of model training burden than missingness level. Other logged properties, including imputability ratio and fallback imputations, contributed minimally and were not retained for further analysis. Overall, these findings indicate that SudokuImputer's scalability is primarily influenced by dataset size and structure rather than the proportion of missing values.

4.2 SudokuImputer Is Best-in-Class against Six Numerical Imputation Techniques

Across 45 experimental datasets (three datasets induced with three missingness types over five missingness proportions), SudokuImputer records best-in-class performance by RMSE in 22 datasets. This effect is most noteworthy at higher proportions of data missingness, where relative to other strategies SudokuImputer often records a substantially lower average RMSE (Figure 4; Supplemental Figure 3). The performance of SudokuImputer is more robust on MNAR and MAR data missingness than comparable ML-based techniques like MICE and SoftImpute. Statistical ML-based imputation strategies outperform point-value imputation across all benchmarks. When each method was ranked in each experimental dataset and averaged together by data missingness class, SudokuImputer was consistently the top-ranking algorithm by missingness type (Figure 5). Other algorithms such as MICE and kNN inconsistently beat SudokuImputer and demonstrate a worse performance distribution than SudokuImputer. In a comparison of the mean and median rank of MICE, kNN, and SudokuImputer grouped by experimental dataset and missingness type, SudokuImputer records the highest or second highest rank in seven out of nine groups, while kNN and MICE combined record the highest or second highest mean/median rank in eight out of nine groups (Figure 6). Mean RMSE scores of all models in each dataset are available in Supplemental Table 1.

SudokuImputer outperforms all imputation algorithms (records the lowest RMSE) in four out of 15 MCAR experiments, nine out of 15 MNAR experiments, and nine out of 15 MAR experiments (Figure 7). When SudokuImputer achieves the lowest RMSE, it is often by a stable margin; average normalized RMSE margins between SudokuImputer and the second-best method are 0.0057, 0.0067, and 0.012 for MCAR, MNAR, and MAR-type missingness. When SudokuImputer does not achieve best-in-class performance in MCAR and MNAR experiments, it is by an equivalently stable margin (average normalized RMSE margins: 0.0053, 0.0038). However, in MNAR SudokuImputer underachieves by considerably larger margins (Figure 8). SudokuImputer demonstrated stable performance relative to other imputation techniques; it had fewer monotonicity violations than kNN and MICE on EEG and QSAR datasets (Supplemental Figure 4) and noteworthy performance jumps (greater than a mean RMSE of 0.05) between subsequent missingness proportions were only observed between forty percent and fifty percent missingness (Supplemental Figure 5).

While SudokuImputer records the lowest average RMSE across most experimental datasets, it also records the highest runtime across datasets compared with six numerical imputation strategies (337.96, 138.60, and 5212.77 seconds for the QSAR, EEG, and Arcene datasets) (Figure 8). Given that the QSAR, EEG, and Arcene datasets each contain 40, 12, and 100 columns of data, it appears that SudokuImputer and MICE runtimes scale proportionally to the feature set size of the dataset while kNN, matrix factorization, and SoftImpute are more robust against dataset size effects. Across all non-point value imputation algorithms, runtime is slightly proportional to data missingness proportion, with longer runtimes documented for larger missing data volumes. The raw runtime of each model in each experimental dataset is available in Supplemental Table 2.

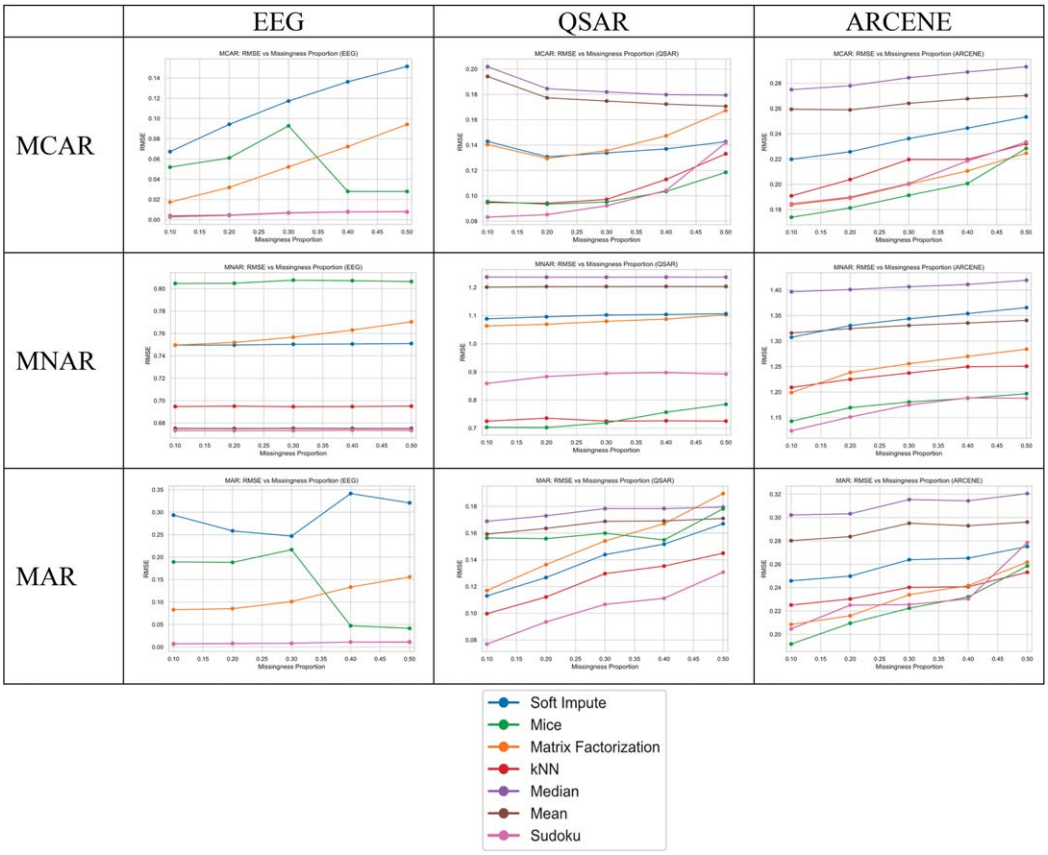


Figure 4: Comparative line plots of mean RMSE over data missingness proportions for SudokuImputer against median, mean, kNN, MICE, SoftImpute, and Matrix Factorization imputation. Results are grouped by data missingness type and experimental dataset.

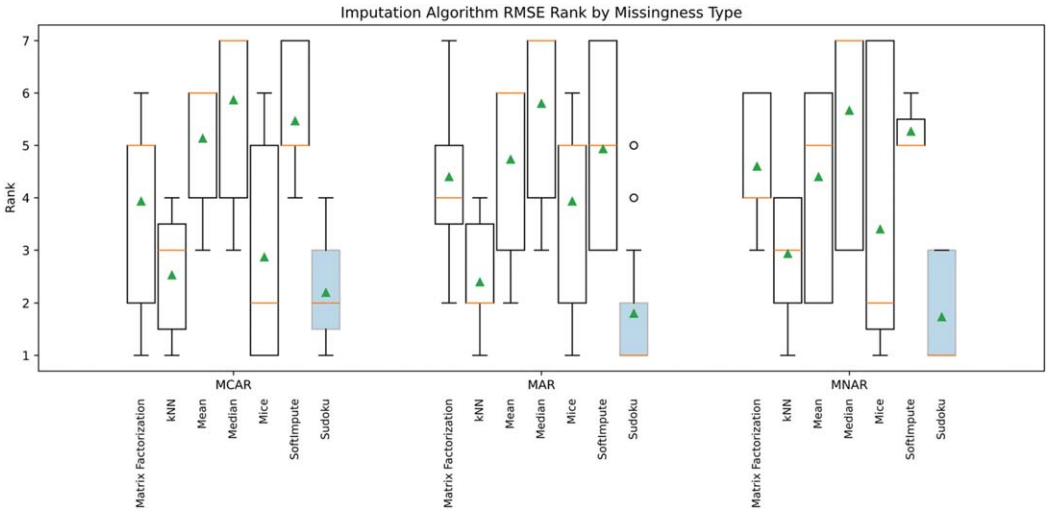


Figure 5: Median and mean ranking of SudokuImputer against six benchmarking imputation methods by average imputation RMSE. Ranking is scored among the seven algorithms within each missingness proportion for a given experimental dataset (i.e. within "EEG 10% missingness"). Each box-and-whisker plot represents the distribution of 15 rankings for that algorithm on the grouped missingness type. The median rank is denoted with the orange line, and the average rank with the green triangle. The highest overall ranking method for each missingness type is highlighted in blue.

	SudokuImputer			*MICE*			*kNN*		
	EEG	QSAR	Arcene	EEG	QSAR	Arcene	EEG	QSAR	Arcene
MCAR	*1.8 (2.0)*	*1.6 (1.0)*	*3.2* (3.0)*	*5.6 (6.0)*	*1.8 (2.0)*	*1.2 (1.0)*	*1.2 (1.0)*	*2.6 (3.0)*	*3.8 (4.0)*
MNAR	*1.0 (1.0)*	*3.0 (3.0)*	*1.2 (1.2)*	*7.0 (7.0)*	*1.4 (1.0)*	*1.8 (2.0)*	*4.0 (4.0)*	*1.6 (2.0)*	*3.2 (3.0)*
MAR	*1.8 (1.0)*	*1.0 (1.0)*	*2.6 (2.0)*	*5.6 (6.0)*	*4.8 (5.0)*	*1.4 (1.0)*	*2.0 (2.0)*	*2.0 (2.0)*	*3.2 (4.0)*

Figure 6: Mean and median (in parenthesis) RMSE rankings for three ML-based imputation algorithms: SudokuImputer (left), MICE (middle), and kNN (right). Green and purple values indicate the highest and second highest rank among all seven evaluated imputation strategies. * Matrix factorization scored the second highest rank in the MCAR Arcene dataset (mean rank = 1.8, median rank = 2.0).

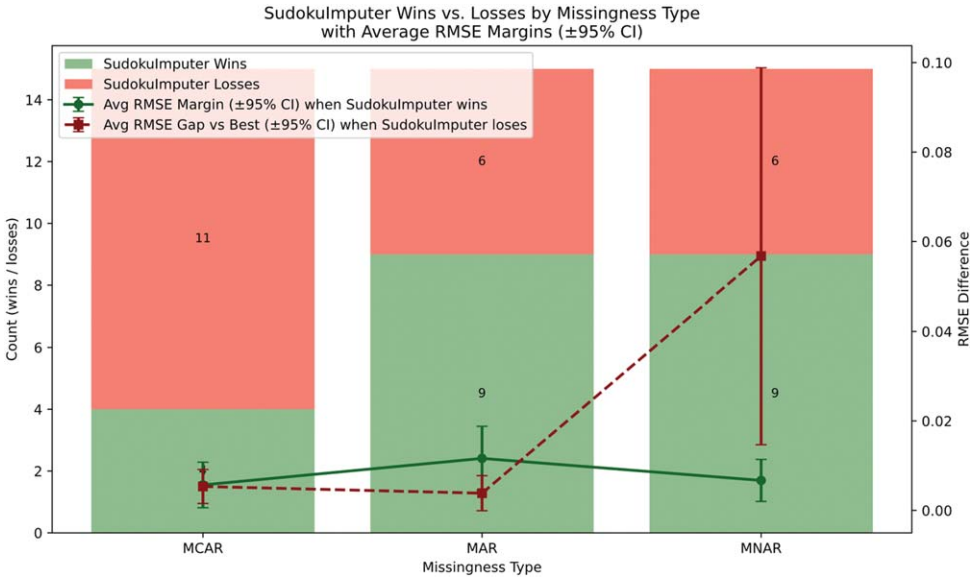


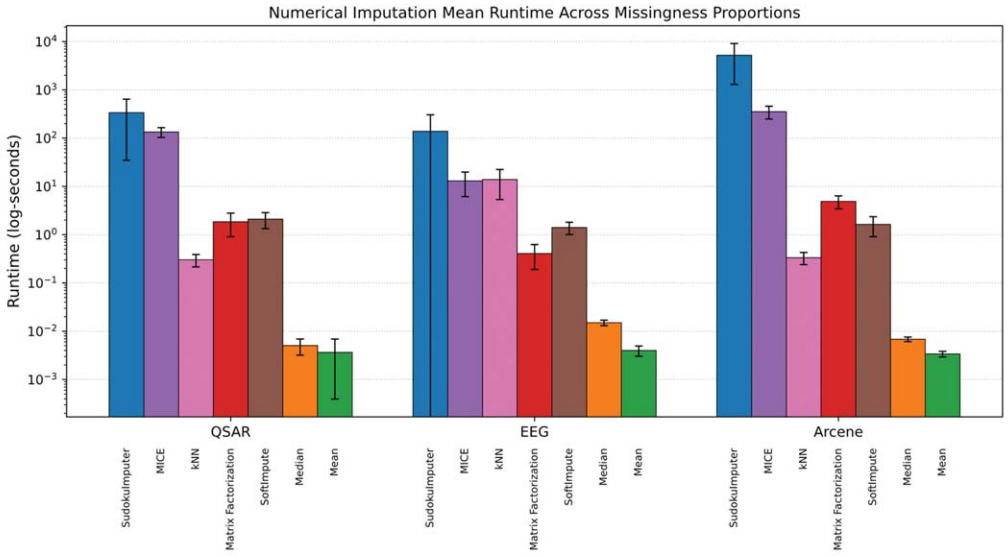
Figure 7: Number of SudokuImputer best-in-class performances relative to six imputation methods across 15 experimental datasets, as measured by average RMSE. Among SudokuImputer wins (lowest RMSE), the average RMSE margin against the runner-up method is overlayed in green. Among SudokuImputer losses, the average RMSE margin against the best performing imputation method is overlayed in red. Error bars indicate a ninety-five percent confidence interval.

5. Discussion

In this study, we introduce SudokuImputer, a graph-based imputation framework inspired by constraint satisfaction in Sudoku puzzles, and we evaluate its performance on three benchmark datasets with MCAR, MAR, and MNAR missingness across a range of missingness proportions. Our results demonstrate that SudokuImputer frequently achieves SOTA RMSE performance, particularly under MAR and MNAR conditions, but at the cost of higher computational runtime.

5.1 Hyperparameter Behavior and Design Implications

Analysis of the hyperparameter space revealed that runtime is overwhelmingly determined by prioritization mode, with centrality-based feature imputation sequencing resulting in order-of-magnitude reductions in execution time



	Sudoku Imputer	MICE	kNN	Matr. Fact.	Soft Impute	Median	Mean
QSAR	337.956	134.293	0.302	1.852	2.098	0.005	0.004
EEG	138.596	12.983	13.889	0.407	1.410	0.015	0.004
Arcene	5212.768	353.707	0.334	4.897	1.635	0.007	0.003

Figure 8: Runtime comparison between seven imputation strategies grouped by experimental dataset in log-seconds. Error bars reflect the standard deviation of runtime across the five missingness proportions for a given dataset. Values in the data table reflect runtime in non-logarithmic seconds.

compared with utility-based feature imputation sequencing. This finding underscores the importance of feature ordering in iterative imputation, a design choice that has received little attention in prior work. While kNN, MICE, and MissForest are known to achieve robust imputations through neighborhood or ensemble learning, they do not adaptively prioritize features during the imputation process. SudokuImputer's reliance on network centrality thus provides a novel mechanism to improve scalability.

In contrast, RMSE was shaped by multiple interacting hyperparameters, most notably partner proportions. Strikingly, excessively lenient partner inclusion (e.g., twenty-five percent of features) degraded accuracy, suggesting that excess connectivity introduces redundancy that weakens predictive signal. Accuracy was primarily influenced by hyperparameters such as partner proportion, which can introduce excess noise when a large amount of weakly informative features are included. This aligns with theories from network science and recommendation systems that excess connections dilute the relevance of strong associations (Larson 2017; Kirsch et al. 2019). For SudokuImputer, tuning the partner proportion hyperparameter represents a critical balance between information sufficiency and over-saturation.

5.2 Scalability and Model Training Burden

The number of models trained was associated primarily with dataset size rather than the proportion of missing values. This indicates that SudokuImputer's computational burden is more sensitive to dataset dimensionality (rows $\times$ columns) than to the extent of missingness. Practically, this means SudokuImputer is more efficient for wide but shallow datasets than for large-scale, high-volume cohorts. All benchmarked imputation algorithms

demonstrated marginally weaker performance on the Arcene benchmark than the EEG or QSAR datasets, indicating a slight association between larger feature set size and larger estimation errors. By contrast, simpler strategies such as point-value imputation or kNN scale more gracefully with dataset size, though at the expense of accuracy.

5.3 Benchmarking against Existing Methods

SudokuImputer achieved best-in-class RMSE in nearly half of the 45 experimental datasets, with especially strong performance under MAR and MNAR conditions (best-in-class in nine out of 15 datasets from each condition). This is notable because MAR and MNAR mechanisms pose greater challenges to traditional imputation methods, which often assume randomness in missingness (MCAR). The robustness of SudokuImputer in these settings suggests that its graph-based prioritization and iterative refitting capture structural dependencies that are overlooked by eager learners like MICE or matrix factorization methods.

Nevertheless, SudokuImputer's advantage was not universal. In MCAR experiments, methods such as kNN and MICE occasionally performed best among the seven evaluated methods, highlighting that simple similarity-based strategies remain competitive when missingness is random. This pattern reflects broader benchmarking studies reporting kNN and Random Forest imputers as consistently strong general-purpose methods. SudokuImputer therefore complements rather than supplants existing approaches, excelling in structured missingness regimes where others falter.

5.4 Runtime Tradeoffs and Practical Feasibility

Despite its accuracy, SudokuImputer recorded the longest runtimes of all evaluated methods. This tradeoff mirrors prior observations that ensemble- and regression-based imputers (e.g., MissForest, MICE) sacrifice speed for accuracy, while matrix factorization and deep generative models achieve faster runtimes but poorer imputations. This is due to the sheer volume of models fit by SudokuImputer, which is proportional to the size of the dataset (namely, the feature count). SudokuImputer's scaling behavior—runtime increasing proportionally with feature set size—makes it less suitable for extremely high-dimensional datasets unless paired with optimization strategies such as parallelization, dimensionality reduction, or sparsity constraints.

The critical question, therefore, is whether the observed RMSE gains justify the computational cost. For small to medium-sized datasets, particularly in biomedical domains where accuracy outweighs runtime, SudokuImputer may provide a favorable tradeoff. However, for very large-scale datasets such as electronic health records or biobank-scale omics data, runtime optimization will be essential before SudokuImputer can be deployed in practice.

5.5 Limitations and Future Directions

We identified numerous opportunities to evaluate and optimize the SudokuImputer v1.0.0 framework. First, evaluations were restricted to three benchmark datasets, which, although diverse, may not capture the heterogeneity of real-world data. Future investigations of benchmarking imputation strategies should evaluate a wide missingness proportion over all three classes of data missingness, across real-world datasets of varying dimensions. Second, we focused exclusively on numerical imputation, excluding mixed-type datasets where categorical features are prevalent. While the beta release of SudokuImputer v1.0.0 supports mixed-type and categorical imputation, these functions were not the subject of our benchmarking experiments and require further rigorous validation and optimization. Third, although SudokuImputer was benchmarked against widely used methods, emerging deep learning–based imputers such as GAIN and diffusion models were not fully explored.

Next releases of SudokuImputer will focus on three directions. Methodologically, optimizing SudokuImputer's runtime via parallel graph construction, adaptive stopping criteria, or hybrid partner selection could substantially reduce computational burden. Additionally, iterative model fitting was not thoroughly explored; a MICE-like or MissForest-like process, where temporary placeholder values are added to increase the imputability ratio may result in stronger estimates on high density missing data. Finally, integrating SudokuImputer into downstream predictive pipelines will determine whether its gains in RMSE translate into improved classification or regression performance, which is the end goal of imputation.

6. Conclusion

SudokuImputer offers a novel graph-based approach to numerical imputation that consistently achieves low RMSE across diverse missingness settings, especially MAR and MNAR, but incurs substantially higher runtime compared with SOTA methods. These findings highlight both the promise and current limitations of graph-based imputation,

motivating future efforts to balance accuracy with scalability. Imputation strategies optimized for both estimation accuracy and dataset scalability allow for efficient generation of reliable training datasets for ML models and AI systems. Many diverse training datasets are critical for these deep learning networks to achieve SOTA performance.

7. Data Availability

Experimental datasets, benchmark imputations, summary tables, figures, and all experimental scripts, analytical workflows, and visualization notebooks will be made available on the SudokuImputer GitHub repository: https://github.com/vkanpa01/SudokuImputer

8. Supplemental Material

Supplemental material can be accessed using the following link: https://github.com/vkanpa01/SudokuImputer.

Acknowledgements: *We would like to thank William Patterson University and the organizers of the New Jersey Big Data Association Annual Symposium, who provided us the opportunity to receive feedback and recommendations on SudokuImputer version 1.0.0 from experienced academics and data practitioners. Furthermore, we would like to thank Dr. Vadim Zhernovkov, Assistant Professor at University College Dublin and Director of the AI in Medicine & Medical Research program, for his feedback and insights into benchmarking methods and analytic techniques to assess the performance of SudokuImputer version 1.0.0.*

References

Alam, S., M. S. Ayub, S. Arora, and M. A. Khan. 2023. "An Investigation of the Imputation Techniques for Missing Values in Ordinal Data Enhancing Clustering and Classification Analysis Validity." *Decision Analytics Journal* **9**: 100341. https://doi.org/10.1016/j.dajour.2023.100341

Aral, S., and M. W. Van Alstyne. 2010. "The Diversity-Bandwidth Tradeoff." *American Journal of Sociology* (forthcoming). https://doi.org/10.2139/ssrn.958158

Brown, S., O. Kudia, K. Kleine, B. Kidd, R. Wines, and N. Meckes. 2025. "Comparing Multiple Imputation Methods to Address Missing Patient Demographics in Immunization Information Systems: Retrospective Cohort Study." *Archives of Computational Methods in Engineering* **31**: 2985–3013. https://doi.org/10.2196/preprints.73916

Di Fiore F., M. Nardelli, and L. Mainini. 2024. "Active Learning and Bayesian Optimization: A Unified Perspective to Learn with a Goal." *Archives of Computational Methods in Engineering* (forthcoming). https://arxiv.org/abs/2303.01560

Frazier, P. I. 2018. "A Tutorial on Bayesian Optimization." Preprint, submitted July 8.

Garnett, R. 2023. *Bayesian Optimization Book*. Available at https://bayesoptbook.com/

Gärtner, T., P. Flach, S. Wrobel. 2003. "On Graph Kernels: Hardness Results and Efficient Alternatives." In *Learning Theory and Kernel Machines. Lecture Notes in Computer Science*, B. Schölkopf, and M. K. Warmuth, editors, volume 2777. Berlin and Heidelberg, Germany: Springer. https://doi.org/10.1007/978-3-540-45167-9_11

Guyon, I., S. Gunn, A. Ben-Hur, and G. Dror. 2004. "Arcene." [Dataset]. *UCI Machine Learning Repository*. https://doi.org/10.24432/C58P55

Halder, R. K., M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat. 2024. "Enhancing K-Nearest Neighbor Algorithm: A Comprehensive Review and Performance Analysis of Modifications." *Journal of Big Data* **11**, no. 1. https://doi.org/10.1186/s40537-024-00973-y

Hestness, J., S. Narang, N. Ardalani, et al. 2017. "Deep Learning Scaling is Predictable, Empirically." Preprint, submitted December 1.

Heymans, M. W., and J. W. R. Twisk. 2022. "Handling Missing Data in Clinical Research." *Journal of Clinical Epidemiology* **151**: 185–188. https://doi.org/10.1016/j.jclinepi.2022.08.016

Hoffmann, J., S. Borgeaud, A. Mensch, et al. 2022. "Training Compute-Optimal Large Language Models." Preprint, submitted March 29.

Jadhav, A., D. Pramod, and K. Ramanathan. 2019. "Comparison of Performance of Data Imputation Methods for Numeric Dataset." *Applied Artificial Intelligence* **33**, no. 10: 913–933. https://doi.org/10.1080/08839514.2019.1637138

Jäger, S., A. Allhorn, and F. Bießmann. 2021. "A Benchmark for Data Imputation Methods." *Frontiers in Big Data* **4**: 693674. https://doi.org/10.3389/fdata.2021.693674

Kang, H. 2013. "The Prevention and Handling of the Missing Data." *Korean Journal of Anesthesiology* **64**, no. 5: 402–406. https://doi.org/10.4097/kjae.2013.64.5.402

Kaplan, J., S. McCandlish, T. J. Henighan, et al. 2020. "Scaling Laws for Neural Language Models." Preprint, submitted January 23.

Kirsch, A., J. van Amersfoort, and Y. Gal. 2019. "BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning [Review of BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning]." *NeurIPS* **33**: 1–12. https://doi.org/10.48550/arXiv.1906.08158

Larson, J. M. 2017. "The Weakness of Weak Ties for Novel Information Diffusion." *Applied Network Science* **2**, no. 1. https://doi.org/10.1007/s41109-017-0034-3

Mansouri, K., T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni. 2013. "*QSAR Biodegradation."* [Dataset]. *UCI Machine Learning Repository*. https://doi.org/10.24432/C5H60M

Mera-Gaona, M., U. Neumann, R. Vargas-Canas, and D. M. López. 2021. "Evaluating the Impact of Multivariate Imputation by MICE in Feature Selection." *PLoS One* **16**, no. 7: e0254720. https://doi.org/10.1371/journal.pone.0254720

Misztal, M. A. 2019. "Comparison of Selected Multiple Imputation Methods for Continuous Variables – Preliminary Simulation Study Results." *Acta Universitatis Lodziensis. Folia Oeconomica* **6**, no. 339: 73–98. https://doi.org/10.18778/0208-6018.339.05

Monard, M.-C. 2002. "A Study of K-Nearest Neighbour As an Imputation Method [Review of A Study of K-Nearest Neighbour As an Imputation Method]." *Soft Computing Systems: Design, Management and Applications* 1–10. https://www.researchgate.net/publication/220981745_A_Study_of_K-Nearest_Neighbour_as_an_Imputation_Method

Poulos, J., and R. Valle. 2018. "Missing Data Imputation for Supervised Learning." *Applied Artificial Intelligence* **32**, no. 2: 186–196. https://doi.org/10.1080/08839514.2018.1448143

Prasad, A. 2024. "Impact of Poor Data Quality on Business Performance: Challenges, Costs, and Solutions." *SSRN Electronic Journal*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4843991

Roesler, O. 2013. "*EEG Eye State."* [Dataset]. *UCI Machine Learning Repository*. https://doi.org/10.24432/C57G7J

Rubin, D. B. 1976. "Inference and Missing Data." *Biometrika* **63**, no. 3: 581–592. https://doi.org/10.2307/2335739

Schonlau, M., and R. Y. Zou. 2020. "The Random Forest Algorithm for Statistical Learning." *The Stata Journal* **20**, no. 1: 3–29. https://doi.org/10.1177/1536867X20909688

Schouten, R. M., and G. Vink. 2018. "The Dance of the Mechanisms: How Observed Information Influences the Validity of Missingness Assumptions." *Sociological Methods & Research* **50**, no. 3: 1243–1258. https://doi.org/10.1177/0049124118799376

Sun, T., S. Zhu, R. Hao, B. Sun, and J. Xie. 2022. "Traffic Missing Data Imputation: A Selective Overview of Temporal Theories and Algorithms." *Mathematics* **10**, no. 14: 2544. https://doi.org/10.3390/math10142544

Xie, Y., S. Zhang, J. Qing, R. Misener, and C. Tsay. 2025. "Global Optimization of Graph Acquisition Functions for Neural Architecture Search." Preprint, submitted May 29.

Zhang, Z.. 2016. "Missing Data Imputation: Focusing on Single Imputation." *Annals of Translational Medicine* **4**, no. 1: 9. https://doi.org/10.3978/j.issn.2305-5839.2015.12.38