# Data quality improvement through data cleaning and augmentation methods

## How do different tabular imputation techniques compare when addressing missing values in 6G datasets?

**Kenneth Chan**
**Supervisor(s): Rihan Hai, Yuandou Wang**
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Kenneth Chan Final project course: CSE3000 Research Project
Thesis committee: Rihan Hai, Yuandou Wang, Julian Urbano

An electronic version of this thesis is available at
https://github.com/KenChan000/ImputationFor6GDatasets

**Abstract**

Sixth-generation (6G) wireless systems depend on data-hungry machine-learning pipelines, yet datasets collected from heterogeneous sources frequently contain missing values that bias models and degrade simulation reliability. Tabular imputation has been studied extensively—from statistical baselines (mean, kNN) through model-based methods (MICE, SoftImpute) to recent deep approaches (HyperImpute, GRAPE, DiffPuter)—but no prior work systematically compares this range on 6G data under realistic missingness. We benchmark seven methods on DeepSense 6G datasets across four mechanisms and three missingness rates, evaluating reconstruction accuracy, statistical fidelity, and downstream beam-prediction performance. Our benchmarks show that no single imputation method consistently dominates; performance depends on the missingness mechanism. Under cell-wise missingness, deep methods such as HyperImpute achieve the highest reconstruction fidelity, though downstream beam prediction remains robust to these localised corruptions. In contrast, row-wise missingness degrades all learned and deep approaches by breaking cross-feature dependencies. Here, kNN is the only method that consistently preserves the downstream label signal. Overall, our results provide guidance for 6G pipeline defaults and highlight the limitations of applying purely tabular imputation to temporal wireless data.

# 1 Introduction

Sixth-generation (6G) wireless systems promise ultra-reliable, low-latency communication to support increasingly data-intensive applications [15]. These systems rely heavily on machine-learning pipelines that demand large volumes of high-quality data [9]. However, in practice, 6G datasets collected from heterogeneous sources such as end-user devices and network infrastructure often contain missing or incomplete values due to communication constraints and system heterogeneity. Poor data quality introduces bias into model predictions and undermines simulation reliability [5, 6]; as a result, the handling of missing data becomes a critical prerequisite for 6G machine-learning applications. Although 6G data is inherently temporal, it is commonly transformed into tabular feature representations for downstream learning, where missing values are handled using standard imputation techniques [2].

In these tabular representations, data imputation is the process of estimating and filling in missing values. It has been studied extensively in the broader machine-learning and data-science literature. Classical approaches include simple baselines such as mean and kNN imputation [18], statistical models such as MICE [19], and low-rank methods like SoftImpute [11]. More recent approaches leverage machine learning and deep learning, including HyperImpute [7], GRAPE [22], and DiffPuter [23]. However, these methods have not been systematically evaluated on 6G datasets under realistic and varied missingness conditions.

This work addresses that gap by answering the central research question: *How do different tabular imputation techniques compare when addressing missing values in 6G datasets?* We decompose it into three sub-questions:

**RQ1.** How do imputation methods compare in reconstruction error across missingness rates (10%, 30%, 50%) and mechanisms (MCAR, MAR, MNAR, and a structured row-wise variant) in 6G datasets?

**RQ2.** How well do imputation methods preserve the statistical properties (mean, variance, distribution) of the original 6G data?

**RQ3.** To what extent does the choice of imputation method affect the performance of a downstream machine-learning task on 6G data?

The main contribution is a structured benchmark of seven imputation methods—mean, kNN, MICE, SoftImpute, HyperImpute, GRAPE, and DiffPuter—conducted under controlled and reproducible experimental conditions.

# 2 Background and related work

This section establishes what distinguishes 6G data and why it carries missingness; it then discusses the families of tabular imputation methods and positions our study against existing research.

## 2.1 6G domain and data quality

A 6G dataset is a collection of measurements from (or in simulation of) a sixth-generation wireless system, intended to train and evaluate machine-learning models that operate inside the network stack. Two properties distinguish a 6G dataset from a generic wireless dataset. First, the measurements reflect 6G physical-layer features such as millimetre-wave and sub-terahertz bands. Second, the data is typically multi-modal and user-centric, combining heterogeneous communication measurements (e.g. RSSI, channel-state information, beam indices) with co-registered sensing data such as GPS, camera, LiDAR, and radar.

These properties also shape how data goes missing, which is the central concern of this work. Blockage events and fading produce bursty, structured gaps in the beam measurements: an obstacle can remove an entire beam sweep at once [4]. Sensing modalities such as GPS, meanwhile, fail under conditions (urban canyons, poor geometry) that are themselves recorded, making the missingness informative rather than random. A 6G imputation benchmark must therefore consider not only how much data is missing but how it goes missing, since different mechanisms violate different assumptions made by imputation algorithms (Section 3.3).

## 2.2 Families of imputation methods

Imputation completes an incomplete data matrix $X$ by producing $\hat{X}_{ij}$ for every masked entry $(i, j) \in \mathcal{M}$. Existing methods group into five broad families that differ in the statistical assumptions they make about how observed and missing entries relate (Table 1); the specific methods evaluated in this work are introduced in Section 3.4.

Table 1: Families of tabular imputation methods, with representative algorithms from each.

| Family | Key idea | Representative methods |
|---|---|---|
| Simple | Marginal or local statistics | Mean, kNN [5, 18] |
| Iterative chained | Feature-wise regression with iterative refinement | MICE [19], HyperImpute [7] |
| Matrix completion | Low-rank reconstruction of the data matrix | SoftImpute [11] |
| Graph-based deep | Message passing over a feature–sample graph | GRAPE [22] |
| Generative deep | Learned joint distribution over all features | DiffPuter [23] |

## 2.3 Related Work

**Benchmarks of tabular imputation.** Systematic comparisons exist in the general machine-learning literature: Jadhav et al. [5] compare classical statistical imputers on numeric data, and Jäger et al. [6]

benchmark statistical and machine-learning methods more broadly, notably showing that reconstruction accuracy and downstream utility frequently disagree. Method papers such as HyperImpute [7] and DiffPuter [23] add their own evaluations against strong baselines. These studies use almost exclusively generic numeric or mixed-type benchmarks (e.g. UCI tables), whose structure differs substantially from wireless measurements: beam-power vectors are strongly correlated across adjacent antenna indices, and channel-state matrices exhibit a known low-rank structure that generic tables do not reproduce. Whether method rankings established on generic data carry over to wireless data is therefore an open question—reinforced by Rubachev et al. [13], who show that under realistic evaluation elaborate deep models frequently fail to outperform simple baselines such as multilayer perceptrons.

**Tabular versus temporal treatment.** 6G measurements are collected as sequences with spatial and temporal dependence, and a body of work imputes them with time-series models. Ignoring this structure carries risks: Rubachev et al. [13] show that splitting temporally collected data randomly at the row level leaks information between adjacent samples, inflating measured performance and altering method rankings. We deliberately adopt a tabular treatment, recovering each sample's missing values from cross-feature structure alone, independent of its position in the sequence. This is the object of study rather than a mere simplification: by withholding the sequential information that time-series imputers exploit, we test whether general-purpose tabular methods can substitute for explicitly temporal modelling on inherently sequential 6G data.

**Reconstruction of missing wireless measurements.** The wireless-communications literature has independently developed signal-specific recovery methods: exploiting the sparse, low-rank structure of millimetre-wave channel matrices via matrix completion or compressed sensing [8], or treating the channel response as an image reconstructed with deep super-resolution and denoising networks [16]. These reconstructors are powerful but assume a particular acquisition structure (pilot grids, channel matrices) and are evaluated against their own objective on sensing-dictated missingness, rather than the controlled MCAR/MAR/MNAR mechanisms used to benchmark general imputers. They therefore say little about how a general-purpose tabular imputer behaves on 6G data, and vice versa.

**The gap.** We are not aware of any prior study that systematically evaluates the full range of tabular imputation families on 6G data under varied, realistic missingness. This work addresses that gap on two DeepSense 6G beam-prediction scenarios under MCAR, MAR, MNAR, and structured row-wise missingness.

# 3 Methodology

This section describes the experimental methodology: the overall pipeline (Section 3.1), the construction of complete reference datasets (Section 3.2), the generation of missing-data masks (Section 3.3), the imputation methods (Section 3.4), and the evaluation metrics (Section 3.5). Dataset and implementation details are presented in Section 4.

## 3.1 General pipeline

We consider a complete tabular dataset $X \in \mathbb{R}^{n \times d}$, where each of the $n$ rows is a sample and each of the $d$ columns a feature. The columns are partitioned into target columns $T \subseteq \{1, \ldots, d\}$, into which missingness is injected, and disjoint auxiliary columns $A = \{1, \ldots, d\} \setminus T$ that remain fully observed. The auxiliary columns serve two roles: they drive the MAR mechanism (Section 3.3) and they provide the inputs to the downstream predictor (RQ3, Section 3.5). The $T$/$A$ assignment is a dataset-specific modelling choice (Section 4).

We synthesise an incomplete observation by applying a binary mask $M \in \{0,1\}^{n\times d}$, where $M_{ij} = 1$ renders $X_{ij}$ missing and $M_{ij} = 0$ leaves it observed, with $M_{ij} = 1$ permitted only for $j \in T$; the observed matrix $X^{\text{obs}}$ equals $X$ where $M_{ij} = 0$ and is NaN otherwise. An imputation method is a function $g$ producing $\hat{X} = g(X^{\text{obs}})$. Each completed matrix is evaluated along three axes: reconstruction, distributional fidelity, and downstream task. Figure 4 (Appendix) summarises the pipeline.

## 3.2 Constructing the complete ground-truth matrix

Reconstruction and fidelity can only be measured against ground truth, so the pipeline starts from a complete matrix $X$ into which missingness is later injected. From the raw data we keep only rows whose target and auxiliary entries are all genuinely observed. Values that are degraded but still valid (e.g. an attenuated signal under blockage) are kept as legitimate low-magnitude measurements, since removing them would strip out the realistic variation the benchmark is meant to stress. Because target columns are standardised before masking, mean imputation—which predicts each column's mean—incurs a per-cell error equal to the standardised true value, so its RMSE equals the standard deviation of the held-out entries: RMSE $\approx 1$ under value-independent masking, and any method scoring above 1 is worse than this baseline. The dataset-specific outcome of this procedure is reported in Section 4.

## 3.3 Injecting missingness

We evaluate four missingness conditions (Figure 1): the three Rubin [14] mechanisms (MCAR, MAR, MNAR) and a structured row-wise variant of MCAR. Each is calibrated to the same target marginal rate $p \in \{0.1, 0.3, 0.5\}$ over the target columns $T$, leaving the auxiliary columns $A$ observed, so that differences reflect the mechanism rather than a different denominator. **MCAR** masks each value independently of all data, $P(M \mid X) = P(M)$, modelling random packet loss or sensor dropout. **MAR** lets missingness depend only on observed variables, $P(M \mid X) = P(M \mid X^{\text{obs}})$; in our pipeline it is a function of the auxiliary columns, the assumption on which MICE [19] and HyperImpute [7] are built. **MNAR**, the hardest case, lets missingness depend on the unobserved value itself, $P(M \mid X) \neq P(M \mid X^{\text{obs}})$—plausible in 6G when channel-state information drops out precisely when signal quality is poorest—and cannot be fully recovered without modelling the missingness process. **Structured row-wise MCAR** is value-independent but block-shaped: a fraction $p$ of rows is selected uniformly at random and fully masked, stressing methods that rely on within-row co-observation. The exact masking procedures are given in Section 4.2.

## 3.4 Imputation methods

We evaluate seven methods spanning the families of Section 2.2; all follow the protocols and architectures of their original publications, and the selection rationale is given in Appendix A.1.

**Mean imputation** replaces each missing entry with its column mean (assumption-free baseline).

**$k$-nearest neighbours imputation (kNN)** [18] replaces each missing entry with the average over the $k$ nearest complete samples in observed-feature space.

**Multiple Imputation by Chained Equations (MICE)** [19] regresses each feature on the others, cycling to convergence (fully conditional specification).

**SoftImpute** [11] completes the matrix by iterative soft-thresholded SVD, enforcing low rank.

**HyperImpute** [7] is an AutoML iterative imputer that selects and tunes a per-feature estimator automatically.

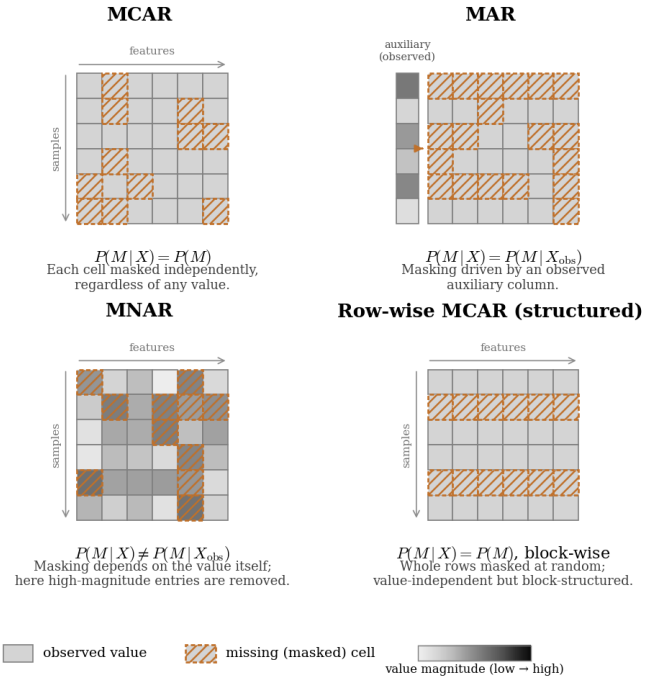


Figure 1: The four missingness mechanisms on a samples $\times$ features matrix. Solid cells are observed, dashed cells masked. MCAR masks cells independently; MAR masks more in rows where the observed auxiliary column (left) is high; MNAR preferentially masks high-magnitude (darker) entries; row-wise MCAR masks whole rows at random. Grey shading encodes value magnitude where the mechanism uses it.

**GRAPE** [22] casts imputation as edge prediction on a bipartite sample feature graph solved with a GNN.

**DiffPuter** [23] learns the joint distribution with a diffusion model and imputes by conditional denoising within an EM loop; internally it Gaussianises each column and rescales inputs by a factor of 0.5.

## 3.5 Evaluation metrics

Imputation quality is judged along three axes, and a method strong on one is not guaranteed to be strong on the others [6].

**Reconstruction error** measures how closely $\hat{X}_{ij}$ approximates $X_{ij}$ at the masked positions $\mathcal{M} = \{(i,j) : M_{ij} = 1\}$:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} (\hat{X}_{ij} - X_{ij})^2}, \qquad \text{MAE} = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} |\hat{X}_{ij} - X_{ij}|. \tag{1}$$

RMSE penalises large deviations quadratically and is sensitive to outlier errors, whereas MAE treats deviations linearly and is more robust [3]. Both are computed only over held-out positions, measuring how well the imputer recovers values it was never shown. Because every feature

is standardised to unit variance before masking (Section 4.1), the reported RMSE and MAE are dimensionless—each error is expressed in units of the feature's standard deviation, i.e. a standard-deviation–normalised RMSE—which is why the mean-imputation baseline falls at $\mathrm{RMSE} \approx 1$ (Section 3.2).

**Statistical preservation.** Reconstruction metrics are blind to distributional distortion: an imputer that regresses every entry toward its column mean can achieve low RMSE while collapsing marginal variance. Let $\mathbf{x}_j$ and $\hat{\mathbf{x}}_j$ denote the held-out (masked) entries of the $j$-th column in the reference and imputed matrices, taken as empirical distributions; restricting to the masked positions isolates the imputer's effect, since the observed entries are identical in both matrices. We report the per-column Wasserstein-1 distance $W_1(\mathbf{x}_j, \hat{\mathbf{x}}_j)$, the mean shift $\Delta\mu_j = \bar{\hat{x}}_j - \bar{x}_j$, and the variance-retention ratio $\rho_j^\sigma = \widehat{\mathrm{Var}}(\hat{\mathbf{x}}_j)/\widehat{\mathrm{Var}}(\mathbf{x}_j)$ (where $\rho_j^\sigma = 1$ indicates perfect preservation, $< 1$ shrinkage, and $> 1$ over-dispersion).

To capture joint feature dependencies, we also report the Frobenius distance between the empirical Pearson correlation matrices, $C$ and $\hat{C}$, of the original and imputed data respectively, computed over all rows:

$$d_F = \|C - \hat{C}\|_F = \sqrt{\sum_{j=1}^{d}\sum_{k=1}^{d}(C_{jk} - \hat{C}_{jk})^2}. \tag{2}$$

This measures joint-structure recovery, where $d_F = 0$ indicates perfectly recovered inter-feature dependency.

**Downstream-task performance.** The ultimate question is whether models trained on imputed data perform as well as on clean data [6, 7]. Crucially, the imputer touches only the *training* portion of the data: the training labels are derived from the imputed target matrix, while the predictor's input features and the test labels are read from the clean, fully-observed matrix. The downstream view thus isolates how much the corruption introduced by each imputer degrades the label signal a model learns from, holding both inputs and evaluation target at ground truth.

The task is position-only beam prediction [12]: given a codebook $\mathcal{F} = \{f_m\}_{m=1}^M$, the optimal beam for sample $k$ is $f^* = \arg\max_{f\in\mathcal{F}} |h^T f|^2$, and a predictor outputs a ranked candidate list. Top-$K$ accuracy counts a prediction correct at $K$ if $f^*$ lies in the model's $K$ highest-scoring beams. We report $K = 1, \ldots, 5$ and bound all scores by a *Clean* ceiling, the same predictor on uncorrupted data.

# 4 Experiments

This section gives the concrete instantiation of the pipeline of Section 3: datasets, missingness and experiment design, compute environment, tuning protocol, downstream protocol, and reproducibility steps.

## 4.1 Datasets

We use two vehicle-to-infrastructure (V2I) beam-prediction scenarios from the DeepSense 6G dataset [2], a large-scale real-world multimodal sensing-and-communication dataset. Each sample contains a 64-element millimetre-wave received-power vector, GPS position information, GPS-quality indicators, a sequence index, a timestamp, and the optimal beam label. Although the dataset also provides RGB imagery and, for Scenario 33, LiDAR and radar measurements, we focus exclusively on the GPS-to-beam modality.

*Scenario 5* is an urban V2I deployment. After the complete-matrix construction described in Section 3.2, it contributes $n = 2{,}300$ samples across 29 trajectory sequences, with a 64-element beam-power vector and six auxiliary GPS-related features: latitude, longitude, vehicle direction, satellite count, PDOP, and HDOP.

*Scenario 33* is a separate urban deployment on a two-way city street. It contributes $n = 3{,}644$ samples across 21 sequences and contains the same 64-element beam-power vector and GPS position information. The auxiliary features differ slightly, providing satellite count, PDOP, HDOP, and VDOP, but no vehicle-direction field.

**Preprocessing.** For each dataset, the retained matrix is standardised column-wise to zero mean and unit variance (`StandardScaler`); all missingness, imputation, and metrics operate in this standardised space. Two methods apply an additional internal transform that is reversed before evaluation: DiffPuter Gaussianises each column with a quantile transform (and rescales inputs by a factor of $0.5$, see 3.4), and GRAPE min–max scales to $[0, 1]$. For the downstream task only, the two GPS coordinate columns used as predictors are min–max scaled to $[0, 1]$, which Morais et al. [12] found best; the target beam columns are left in native scale because the label is the scale-invariant arg max.

## 4.2 Missingness generation

Missingness is injected only into the $64$ beam-power target columns of both datasets, leaving the GPS/position driver columns fully observed. Given a target marginal rate $p \in \{0.10, 0.30, 0.50\}$, the boolean mask $M$ realises the four mechanisms of Section 3.3 as follows.

- **MCAR.** Each target cell is masked independently: $M_{ij} \sim \text{Bernoulli}(p)$.
- **MAR.** A per-row score $s_i$ is formed by standardising the driver column (GPS HDOP, for both scenarios) and mapped through a logistic, $p_i = \sigma(s_i + b)$, with intercept $b$ calibrated by bisection so that $\frac{1}{n}\sum_i p_i = p$. Every target cell in row $i$ is masked with probability $p_i$; because $p_i$ is shared across a row's target columns, missingness concentrates in high-scoring rows and, at large $p$, approaches whole-row removal.
- **MNAR.** Each target cell is masked with probability $\sigma(x_{ij} + b)$, where $x_{ij}$ is the standardised value and $b$ is calibrated by the same bisection; the positive slope makes larger values more likely to be removed.
- **Structured row-wise MCAR.** Exactly $k = \text{round}(pn)$ rows are drawn uniformly at random without replacement and fully masked; all other rows remain observed.

Masks are regenerated per seed from `numpy.random.default_rng(seed)` with seeds $0, \ldots, N_{\text{seed}} - 1$, so that all methods see identical masks within a seed.

## 4.3 Experiment design

The study is a full factorial over four axes: imputation method, missingness mechanism, missingness proportion, and random seed. We evaluate the seven methods of Section 3.4 (Mean, kNN, MICE, SoftImpute, HyperImpute, GRAPE, DiffPuter) under the three classical mechanisms (MCAR, MAR, MNAR) at $p \in \{0.10, 0.30, 0.50\}$, each replicated over three seeds (0, 1, 2), yielding $7 \times 3 \times 3 \times 3 = 189$ reconstruction runs per dataset. The structured row-wise MCAR condition adds a further 63 runs.

Two design choices keep the comparison controlled: every method uses the single hyper-parameter configuration selected once on MCAR $\times$ 30% (Section 4.5), held fixed across all conditions; and for a given (mechanism, proportion, seed) the same mask is presented to all seven methods (Section 4.2). Differences therefore reflect the condition rather than re-tuning or different missing cells.

The same frozen pipeline feeds two complementary views: a reconstruction view (RQ1: RMSE/MAE; RQ2: distributional fidelity) and a downstream view (RQ3: beam-prediction utility, Section 4.6). A clean baseline on the fully observed matrix is computed once and broadcast as the downstream ceiling.

## 4.4 Compute environment

All experiments were run on a single workstation with an NVIDIA GeForce RTX 4070 Ti SUPER GPU (16 GB GDDR6X), an Intel Core i9-12900K CPU, and 32 GB of 3200 MHz RAM, running Windows 10, with Python 3.12.3, PyTorch 2.4.1, and CUDA 12.1. The deep imputers dominate runtime—a single DiffPuter fit takes on the order of ten minutes on this GPU—motivating the caching and reduced-fidelity tuning below; the classical imputers and HyperImpute run in seconds to a few minutes on CPU.

## 4.5 Hyperparameter tuning

To keep the comparison fair and the cost bounded, every method is tuned once, on MCAR at 30% missingness, and the resulting configuration is fixed across all mechanisms and proportions. The selection objective is reconstruction RMSE on the artificially masked entries, computed with the same masking procedure used for evaluation (Section 3.5); cheap methods are tuned over three seeds (0, 1, 2) with the RMSE averaged, while the two deep methods are tuned on a single seed (0), as each fit is too expensive to repeat.

Three methods are recorded as untuned: Mean is parameter-free, MICE exposes only an iteration budget (`max_iter`, left at the library default of 10), and HyperImpute is itself an AutoML procedure so hand-tuning its per-column model selection would only undercut it. The remaining methods are tuned on their influential parameters. By grid search: kNN's `n_neighbors` over $\{3, 5, 10, 20, 50\}$ and SoftImpute's shrinkage $\lambda$ over a data-derived grid based on the singular-value spectrum. By Optuna [1]: DiffPuter's `lr` and `hid_dim` (15 TPE trials, after a sensitivity scan fixes non-influential parameters and an EM probe sets `n_em_iterations`=3); and GRAPE's `epochs` $\in \{2000, 5000, 10000, 20000\}$, tied `node_edge_dim` $\in \{32, 64, 128\}$, and `lr` (log-uniform over $[3 \times 10^{-4}, 3 \times 10^{-3}]$, 20 TPE trials). Both deep searches use a small ($10^{-3}$) capacity penalty as a tie-breaker. Table 3 gives the selected configurations for Scenario 5 (MCAR $\times$ 30%); Scenario 33 is in Table 4.

## 4.6 Downstream setup

We adopt the position-aided beam-prediction task of Morais et al. [12], in which the receiver's GPS coordinates predict the optimal millimetre-wave beam, sidestepping an exhaustive beam sweep. The predictor is a multilayer perceptron following Morais et al. [12] (Table 5).

**Task and labels.** The input is the 2-D GPS position $(g_{\text{lat}}, g_{\text{lon}})$, min-max scaled to $[0, 1]$; the label is the optimal beam (arg max over the 64 beam-power columns). Two properties localise where imputation enters. First, missingness is injected only into the beam columns, which serve solely to derive the label; the GPS inputs are never masked, so imputation affects only the training labels, not

the model inputs. Second, imputation is applied to the training rows alone: for each split we mask and impute the training block, derive its labels from the imputed beams, and read the test labels from the clean beams.

**Evaluation splits.** All splits are grouped by capture sequence so that no sequence is shared between training and test, preventing temporal adjacency from leaking across the split. For both datasets, methods are evaluated with 3-fold sequence-grouped cross-validation (3 seeds $\times$ 3 folds = 9 evaluations per cell); the heavier methods (MICE, HyperImpute, DiffPuter, and GRAPE) are evaluated on a single fold (3 seeds $\times$ 1 fold = 3 evaluations) due to their substantially higher per-fit cost. Within each training fold the predictor reserves a further 25% of its rows as an internal validation set, and the epoch with the highest validation accuracy is retained, with all other optimiser settings as in Table 5. Each fold imputes its own training rows independently.

**Evaluation metric.** Performance is reported as Top-$K$ accuracy for $K \in \{1, 2, 3, 4, 5\}$: correct at $K$ if the true optimal beam lies within the model's $K$ highest-scoring beams. Top-$K$ accuracy is natural for beam prediction because a base station can probe a small candidate set, so recovering the optimum within the top few beams is operationally as useful as ranking it first.

### 4.7 Reproducibility

All experiments are driven by three notebooks (tuning $\rightarrow$ imputation $\rightarrow$ downstream) with parameters handed off via a JSON file. Random seeds are fixed for mask generation, model initialisation, and stochastic imputers (seeds $0, \ldots, N_{\text{seed}} - 1$). The full code — preprocessing, missingness mechanisms, and method wrappers — is available at `https://github.com/KenChan000/ImputationFor6GDatasets`.

## 5 Results

This section reports observed outcomes only; interpretation is deferred to Section 7. All values are averaged over three seeds (0, 1, 2) and reported with standard deviation. Results are provided for two datasets: Scenario 5 ($n$ = 2,300, 29 sequences) and Scenario 33 ($n$ = 3,644, 21 sequences). Scenario 33 is used to evaluate whether method ordering observed in Scenario 5 transfers across datasets. All metrics are computed in the standardised space (Section 4); the reported RMSE is therefore standard-deviation–normalised (formula in Section 3.5), and the mean-imputation baseline sits at $\mathrm{RMSE} \approx 1$ under value-independent masking.

### 5.1 Reconstruction error (RQ1)

Under MCAR and MAR, all learned methods sit far below the mean baseline in both scenarios (Figure 2; full per-mechanism RMSE/MAE in Table 6). In Scenario 5, HyperImpute leads at MCAR 30% ($0.216 \pm 0.009$), with SoftImpute, MICE, and DiffPuter close behind; the only exception is at 10%, where MICE matches HyperImpute on RMSE. In Scenario 33 the top four methods at MCAR 30% — GRAPE, DiffPuter, kNN, and HyperImpute — are closely competitive (0.220–0.239), and no single method dominates.

The two harder mechanisms diverge. Under MNAR, mean imputation degrades sharply in Scenario 5 (RMSE $2.5 \rightarrow 1.4$ from 10% to 50%) while GRAPE and HyperImpute stay most robust (0.63–0.75); MNAR is less severe in Scenario 33, where GRAPE is again strongest and only

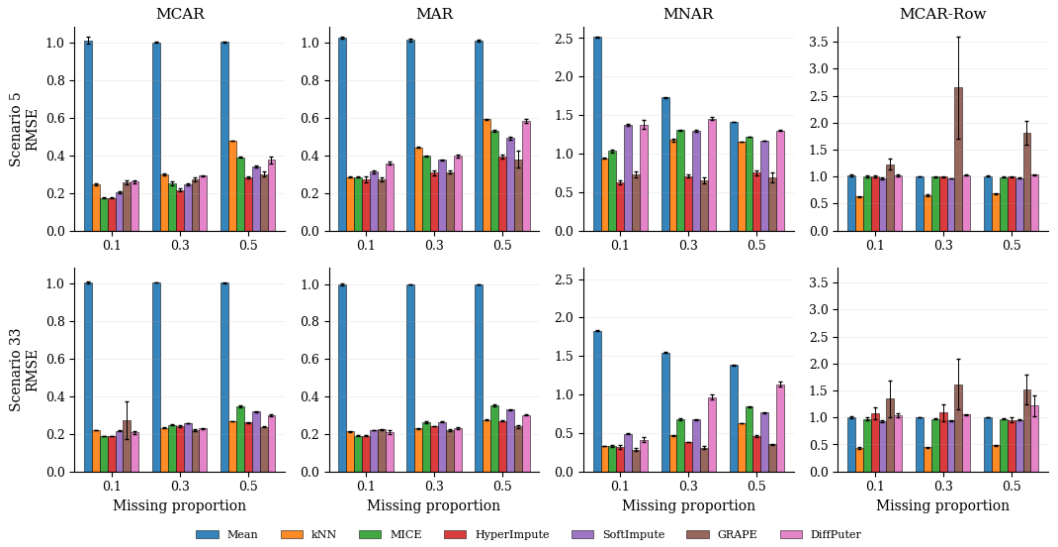


Figure 2: RMSE reconstruction error (lower is better) for seven imputation methods across both scenarios. Bars show means over three seeds with $\pm 1$ SD. Errors are lowest under MCAR/MAR and increase under MNAR and row-wise MCAR.

DiffPuter degrades at 50%. Under structured row-wise MCAR, almost all methods collapse to RMSE $\approx 1$, with kNN the sole exception ($\approx 0.65$ in Scenario 5, 0.43–0.49 in Scenario 33); GRAPE is uniquely unstable, reaching $2.6 \pm 0.9$ at 30% in Scenario 5 and $\approx 1.5$–$1.6$ in Scenario 33.

**Consistency between RMSE and MAE.** Across all scenarios, mechanisms, and methods, MAE closely tracks RMSE with only minor rank swaps among closely performing methods (Table 6). We therefore report RMSE in the main text and include MAE for completeness.

## 5.2 Distributional fidelity (RQ2)

Table 2 reports the four fidelity metrics at MCAR 30% as a representative slice; the complete sweep over all mechanisms and missing rates is deferred to the Appendix, one table per metric (correlation distortion, Table 7; Wasserstein-1, Table 8; variance retention, Table 9; mean shift, Table 10). The ranking does not match reconstruction. In Scenario 5, HyperImpute is strongest on three of four metrics—Wasserstein-1 ($0.031 \pm 0.001$), variance retention ($0.945 \pm 0.008$), and correlation-matrix distance ($0.336 \pm 0.017$)—while SoftImpute attains the smallest mean shift ($0.006 \pm 0.001$). Mean imputation retains zero column variance and has by far the largest correlation distortion ($5.819 \pm 0.016$). DiffPuter is the only method to over-disperse rather than shrink (variance-retention ratio $1.070 \pm 0.006$).

The pattern shifts in Scenario 33. HyperImpute retains its lead on Wasserstein-1 ($0.030 \pm 0.001$) and correlation distance ($0.308 \pm 0.008$), but kNN closes the gap considerably ($0.033 \pm 0.001$). DiffPuter's over-dispersion resolves almost entirely (variance retention $0.979 \pm 0.003$, best in this scenario), while SoftImpute deteriorates: its correlation Frobenius distance rises from 0.556 to $1.238 \pm 0.034$, indicating its low-rank assumption fits Scenario 33 poorly. Mean imputation worsens on every metric, with correlation distortion worsening from 5.819 to $10.814 \pm 0.026$.

Table 2: Fidelity metrics under MCAR missingness at 30%. Values are reported as mean ± standard deviation over three random seeds.

| Dataset | Method | Wasserstein ↓ | Mean shift ↓ | Var. retention (→ 1) | Corr. Frob. ↓ |
|---|---|---|---|---|---|
| Scenario 5 | Mean | 0.630 ± 0.001 | 0.036 ± 0.002 | 0.000 ± 0.000 | 5.819 ± 0.016 |
| | kNN | 0.053 ± 0.000 | 0.030 ± 0.001 | 0.814 ± 0.003 | 0.512 ± 0.029 |
| | MICE | 0.048 ± 0.001 | 0.008 ± 0.001 | 0.896 ± 0.009 | 0.355 ± 0.016 |
| | HyperImpute | **0.031 ± 0.001** | 0.008 ± 0.001 | **0.945 ± 0.008** | **0.336 ± 0.017** |
| | SoftImpute | 0.077 ± 0.001 | **0.006 ± 0.001** | 0.769 ± 0.003 | 0.556 ± 0.049 |
| | GRAPE | 0.087 ± 0.022 | 0.053 ± 0.030 | 0.900 ± 0.013 | 0.892 ± 0.047 |
| | DiffPuter | 0.054 ± 0.005 | 0.028 ± 0.009 | 1.070 ± 0.006 | 0.592 ± 0.020 |
| Scenario 33 | Mean | 0.789 ± 0.001 | 0.027 ± 0.002 | 0.000 ± 0.000 | 10.814 ± 0.026 |
| | kNN | 0.033 ± 0.001 | 0.019 ± 0.002 | 0.942 ± 0.003 | 0.550 ± 0.034 |
| | MICE | 0.050 ± 0.001 | 0.007 ± 0.000 | 0.908 ± 0.002 | 0.370 ± 0.016 |
| | HyperImpute | **0.030 ± 0.001** | 0.007 ± 0.000 | 0.969 ± 0.005 | **0.308 ± 0.008** |
| | SoftImpute | 0.099 ± 0.001 | **0.005 ± 0.000** | 0.783 ± 0.001 | 1.238 ± 0.034 |
| | GRAPE | 0.041 ± 0.006 | 0.019 ± 0.006 | 0.956 ± 0.037 | 0.751 ± 0.180 |
| | DiffPuter | 0.037 ± 0.003 | 0.023 ± 0.003 | **0.979 ± 0.003** | 0.470 ± 0.046 |

## 5.3 Downstream beam prediction (RQ3)

Figure 3 reports Top-1 beam-prediction accuracy against the clean-data ceiling for Scenario 5 (top row) and Scenario 33 (bottom row). The clean ceiling itself differs between datasets—0.401 for Scenario 5 and 0.478 for Scenario 33—reflecting how informative GPS position is for the optimal beam in each layout; all method accuracies are read relative to the corresponding ceiling. The exact Top-1 and Top-5 accuracies for every mechanism, rate, and method are listed in Table 11 (Appendix).

**Cell-wise mechanisms (MCAR, MAR, MNAR).** Under all three cell-wise mechanisms the seven methods form a narrow band close to the clean ceiling in both datasets, with between-method differences typically within the $\pm 1$ SD error bars. For Scenario 5 the Top-1 spread across all seven methods stays within roughly 0.02 at 10% and widens only to about 0.06 at 50% (MNAR 50%: 0.324–0.394 against the 0.401 ceiling). Scenario 33 shows the same pattern, with a band that at the lowest rates sits at or slightly above its 0.478 ceiling and widens modestly at 50% (MCAR 50%: 0.392–0.500).

**Structured row-wise MCAR.** The row-wise mechanism is the only setting where the methods diverge strongly, and this pattern is consistent across both datasets. As the row-wise rate increases, methods relying on marginal statistics deteriorate, while neighbour- and structure-based methods remain close to the ceiling. At 50% row-wise missingness in Scenario 5, kNN performs best (0.349), followed by DiffPuter (0.258) and HyperImpute (0.119), whereas GRAPE (0.053), MICE (0.048), mean (0.041), and SoftImpute (0.039) fall close to chance relative to the 0.401 ceiling. Scenario 33 shows the same ordering: kNN remains near its ceiling (0.477 vs. 0.478), followed by DiffPuter (0.391), HyperImpute (0.289), MICE (0.276), SoftImpute (0.213), with mean and GRAPE lowest (both 0.174). kNN is also remarkably stable across row-wise rates in Scenario 33 ($0.481 \rightarrow 0.478 \rightarrow 0.477$), indicating that whole-row masking barely affects its downstream label recovery.

**Behaviour across $K$.** The Top-1 ordering does not carry over to Top-5 (Table 11). Under the cell-wise mechanisms every method, mean included, lies within a few points of the clean Top-5 ceiling, so the already-narrow Top-1 band all but closes. Under row-wise missingness the ranking reshuffles: kNN keeps the lead ($0.349$ at Top-1, $0.831$ at Top-5 for 50% in Scenario 5, against the $0.883$ ceiling), but several methods near chance at Top-1 recover sharply by Top-5—most notably MICE, rising from $0.048$ to $0.813$—while mean imputation stays lowest at Top-5 in both scenarios. Scenario 33 shows the same reshuffle with a tighter Top-5 spread.

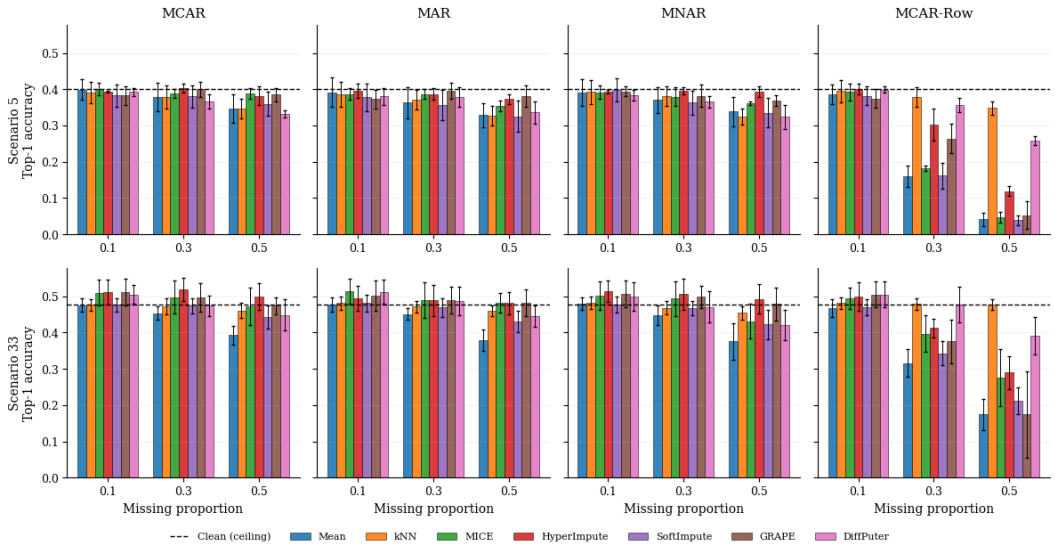


Figure 3: Top-1 beam-prediction accuracy (higher is better) across four missingness mechanisms and three rates for Scenario 5 (top) and Scenario 33 (bottom). Bars show mean grouped cross-validation results with $\pm 1$ standard deviation. The dashed line indicates the clean-data ceiling (model trained on fully observed data).

# 6 Responsible Research

This section reflects on the ethical dimensions and reproducibility of the work, following the FAIR principles of Findable, Accessible, Interoperable, and Reusable research [20].

**Codes of conduct.** This work follows the Netherlands Code of Conduct for Research Integrity and the TU Delft Code of Conduct, observing their principles of honesty, transparency, and verifiability throughout the experimental design, reporting, and release of code and configurations.

**Reproducibility.** The full pipeline, fixed seeds, and hardware/runtime details are documented in Section 4.7 and released at the repository linked on the title page. The DeepSense 6G data is used under its non-commercial academic licence with the citation its authors require. We note two honest limits to exact reproducibility: GPU floating-point operations are not fully deterministic even under fixed seeds, so the deep methods (GRAPE, DiffPuter) may vary slightly between runs; and MICE, HyperImpute, DiffPuter and GRAPE are evaluated on a single sequence-grouped fold rather than

three, giving their downstream estimates a wider interval — both reflected in the reported standard deviations.

**Data, licensing, and privacy.** All experiments use the publicly released DeepSense 6G dataset, accessed under its non-commercial academic licence and cited as its authors require; no data is redistributed in our repository. The data originates from a dedicated experimental testbed — a roadside base station observing a single instrumented transmitter vehicle — rather than from members of the public. The only privacy-relevant attribute we use is the transmitter's GPS position, and we deliberately restrict the study to the GPS-to-beam modality; the co-registered camera, LiDAR, and radar streams, which could incidentally capture bystanders, are excluded entirely. The missingness analysed in this paper is synthetic and injected under controlled mechanisms, so no genuine measurements are withheld or fabricated.

**Bias.** Two design choices are potential sources of bias in the comparison, both flagged as limitations in Section 7. The missingness is synthetic and injected under predefined MCAR, MAR, MNAR, and row-wise mechanisms, which may not reflect the mixed patterns of genuine 6G loss and could shift the relative ordering of methods. Hyper-parameters are also tuned once on MCAR at 30% and reused across all conditions, which may favour methods whose optimal configuration is stable rather than condition-dependent. We do not correct for these in the results but report them transparently so the rankings are read with the appropriate scope.

**Integrity of the evaluation.** Because the benchmark compares competing methods, we took care that none is implicitly favoured. A single hyper-parameter configuration per method is fixed before the main experiments and reused across all mechanisms and rates, masks are shared within a seed so that every method sees identical missing cells, and results are reported with standard deviations rather than as single runs. Where a method's optimal configuration may vary across conditions, or where three seeds are insufficient to separate closely matched methods, we state this explicitly as a limitation rather than report a falsely sharp ranking (Section 7).

**Beyond the project.** The study is a comparative benchmark of existing, published imputation methods and introduces no new capability with a direct pathway to misuse. The main risk is misapplication of the findings: because only two DeepSense 6G V2I scenarios were evaluated, the practical guidance (Section 8)—in particular the robustness of kNN under row-wise loss—should be treated as a scenario-informed recommendation rather than a universal default, and revalidated before transfer to other 6G settings or modalities.

**Use of AI tools.** AI-based assistants were used in a supporting role. AI was used to help debug code and to generate a PowerShell script that creates the virtual environment and installs dependencies. It was further used to assist with the grammar, spelling, and structural editing of this paper, and as an interactive aid for clarifying technical questions. All experimental design, implementation, analysis, and conclusions are the author's own.

# 7 Discussion

We interpret the results along the three research questions, relating method rankings to the assumptions each method makes. All reconstruction metrics are in standardised space (mean baseline $\mathrm{RMSE} \approx 1$).

**Which method wins, and why.** The clearest finding across both scenarios is that no method dominates universally; performance is governed by the missingness mechanism. Under the cell-wise mechanisms (MCAR and MAR), all learned methods substantially outperform the mean baseline, with HyperImpute [7] the most reliable reconstructor: it leads Scenario 5 at higher rates and remains among the best in the more competitive Scenario 33. This matches its design as a fully conditional iterative imputer with per-column AutoML model selection, which exploits the strong correlations between neighbouring antenna indices in the beam-power vectors. The fidelity tables confirm this advantage is not confined to point error: HyperImpute attains the lowest correlation distortion under almost every MCAR and MAR setting in both scenarios (Table 7)—with kNN and GRAPE edging it out only in Scenario 5 MAR at 10%—and is consistently among the smallest in marginal Wasserstein-1 distance (Table 8).

More challenging mechanisms expose where assumptions break down. In MNAR, where high-valued entries are removed more frequently, mean imputation is biased downward and performs poorly (RMSE $2.51 \rightarrow 1.42$ in Scenario 5), while GRAPE [22] and HyperImpute remain robust in both scenarios, suggesting that modelling conditional relationships partially mitigates selection bias. This robustness is mechanism-wide rather than metric-specific: under MNAR, HyperImpute and GRAPE consistently incur the lowest mean shift and retain the most column variance (Tables 10, 9), and in Scenario 33 GRAPE is in fact the strongest method on all four fidelity metrics. DiffPuter [23] instead deteriorates at high rates, rising above the $\mathrm{RMSE} \approx 1$ reference baseline in Scenario 5 (RMSE $1.46$ at 30%, $1.30$ at 50%): because it learns the joint distribution of the observed data, which is a biased sample under MNAR, its reconstructions are pulled towards the biased observations, with correlation distortion rising to the largest of any learned method and variance retention collapsing toward $0.1$ (Tables 7, 9). The over-dispersion noted in Section 5.2 (Scenario 5 variance ratio $1.07$) largely disappears in the larger Scenario 33 ($0.98$), indicating the generative model benefits from more data.

Structured row-wise MCAR provides the clearest link between assumptions and robustness. When an entire row of target columns is masked, methods that rely on within-row observations lose their conditioning information. Consequently, MICE [19], HyperImpute, SoftImpute [11], and DiffPuter all deteriorate toward the mean baseline ($\mathrm{RMSE} \approx 1$), and the fidelity tables show why: they retain essentially zero column variance under row-wise masking (Table 9), falling back to a near-constant fill for fully masked rows. kNN is the only exception in both scenarios, as it imputes from neighbours identified in the observed auxiliary-feature space and can therefore recover complete target vectors from similar rows; correspondingly it preserves both the marginal distribution (lowest Wasserstein-1 under row-wise missingness, Table 8) and a large fraction of the variance (Table 9).

GRAPE performs worst under this mechanism, showing high error and variance ($2.6 \pm 0.9$ at 30% in Scenario 5). A likely explanation is that row-wise masking removes all edges between affected samples and target-feature nodes in GRAPE's bipartite graph, leaving little neighbourhood information for message passing. The fidelity tables show this is a genuine breakdown rather than mere noise: GRAPE's correlation distortion explodes under row-wise missingness, it over-disperses the variance, and its mean shift and Wasserstein-1 distance are an order of magnitude larger than any other method (Tables 7–10).

**Reconstruction does not predict fidelity or downstream utility.** A central message, anticipated by Jäger [6], is reproduced here: the three axes disagree. The best reconstructor is not always best at preserving distributional structure—SoftImpute attains the smallest mean shift under MCAR in both scenarios (Table 10) despite only mid-range RMSE. And downstream Top-1 accuracy is almost flat across methods under the cell-wise mechanisms (Section 5.3): even mean imputation, weakest

on every reconstruction and fidelity metric, stays within the $\pm 1$ SD band, because the arg-max beam label tolerates the distributional damage the fidelity metrics penalise. The only exception is row-wise MCAR, where preserving within-row structure translates directly into higher downstream accuracy—and where kNN, the sole imputer that recovers whole masked rows, stays closest to the clean ceiling.

**Threats to validity.** Several factors limit these conclusions. *Dataset specificity:* only two DeepSense 6G scenarios were evaluated; findings such as SoftImpute's weaker Scenario 33 result and DiffPuter's Scenario 5 variance overestimation may be dataset-specific. *Synthetic missingness:* the predefined MCAR, MAR, MNAR, and row-wise mechanisms may not reflect the mixed patterns of real 6G loss, which could shift the rankings. *Limited seeds:* three seeds suffice to identify large effects but not to reliably separate closely matched methods. *Hyper-parameter selection:* tuning once on MCAR-30% and reusing it for all conditions may disadvantage methods whose optimum is condition-dependent. Finally, *downstream evaluation:* only one beam-prediction model was used, so the Top-1 robustness under cell-wise missingness may not generalise to other models or tasks that depend on the full beam-power distribution.

# 8 Conclusions and Future Work

This work compares tabular imputation methods for reconstructing missing values in 6G datasets. The main finding is that no method consistently dominates: performance is driven primarily by the missingness mechanism. While the deep methods lead under specific settings, none is reliable across all of them; kNN is the only method that never collapses, making it the most robust default for this wireless setting.

Answering the sub-questions directly:

**(RQ1)** Under the cell-wise mechanisms (MCAR, MAR) every learned method far outperforms the mean baseline, with HyperImpute [7] the most consistent reconstructor. MNAR is the hardest cell-wise case: here GRAPE [22] and HyperImpute are the most robust, while mean imputation and DiffPuter [23] degrade. Structured row-wise masking is the sharpest separator—every method except kNN collapses toward the baseline, because row-wise masking removes the within-row co-observation the others depend on.

**(RQ2)** Distributional fidelity does not track reconstruction error. Under the cell-wise mechanisms HyperImpute best preserves both marginal and joint structure and mean imputation destroys both, while method-specific artefacts emerge: DiffPuter over-disperses the variance in the smaller scenario, and SoftImpute's low-rank fit degrades on Scenario 33.

**(RQ3)** The imputation method barely affects downstream beam prediction under cell-wise missingness—even mean imputation stays near the clean-data ceiling, because the arg-max beam label tolerates per-cell corruption. It matters decisively only under whole-row missingness, where kNN (and, to a lesser extent, DiffPuter) preserve the label signal while the others fall toward chance. This row-wise ordering is consistent across datasets, indicating the behaviour is driven by the mechanism rather than either scenario.

**Future work.** First, the tabular treatment discards sequential structure—precisely what fails under row-wise loss—so a natural next step is a head-to-head against a parallel time-series study under the same row-wise missingness, quantifying how much within-row co-observation can instead be recovered from temporal adjacency. Second, the benchmark should extend beyond two DeepSense 6G V2I scenarios and GPS-to-beam data toward additional 6G settings, multimodal inputs, and realistic missingness: empirical blockage and sensor-dropout patterns could be extracted from raw traces and re-injected onto complete data, preserving the ground truth that reconstruction and fidelity metrics require, while testing whether the row-wise findings hold under genuine loss. Third, evaluation should move beyond arg-max beam labels to continuous or higher-resolution targets more tightly linked to reconstruction quality, to determine whether the observed insensitivity reflects imputation generally or this specific task.

# A Extra information

## A.1 Choice of imputation methods.

The methods in Table 1 are selected so that each family is represented at least once, covering each distinct assumption about how observed and missing entries relate. Evaluating them under identical datasets and missingness conditions isolates the effect of the modelling assumption, and spanning the range from an assumption-free baseline to recent generative models lets us test the questions raised in Section 2.2: whether method rankings from generic tabular data transfer to wireless data, and whether the added complexity of deep approaches is warranted here.

Within each family we favour established or recent state-of-the-art methods with maintained, runnable implementations, and prioritise breadth across families over exhaustive coverage within them: adding a further method to a family already represented increases computational cost, which is substantial for the deep models, without introducing a new modelling assumption. Several well-known methods were therefore considered but excluded as redundant. MissForest [17], a random-forest iterative imputer, falls within the iterative-chained family already covered by MICE and HyperImpute, whose automated model selection subsumes tree-based per-feature regressors. GAN-based methods such as GAIN [21] and VAE-based methods such as MIWAE [10] belong to the generative-deep family represented by DiffPuter, which is more recent and avoids the training instability inherent to adversarial objectives. The goal is a controlled comparison across distinct modelling assumptions, not an exhaustive benchmark of every method within each family.

## A.2 Figures and Tables

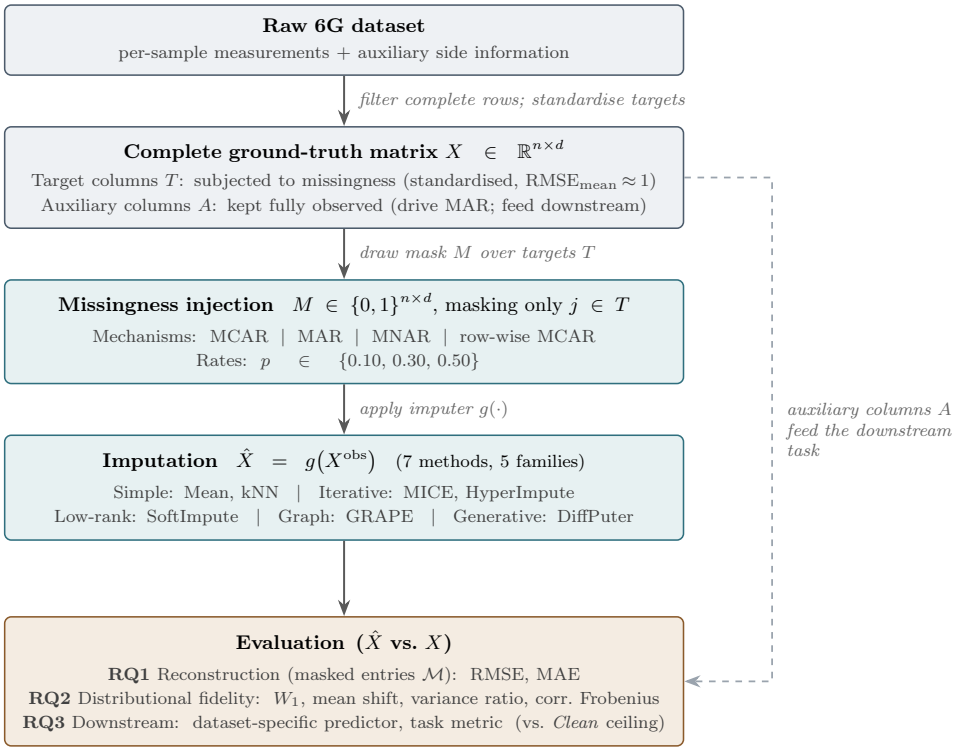


Figure 4: Methodological pipeline illustrating data preparation, missingness masking mechanisms, imputation families, and multi-angle evaluation frameworks.

Table 3: Hyper-parameter Configurations for Scenario 5 trained on (MCAR $\times$ 30%)

| Parameters | Values |
|---|---|
| **kNN** | |
| knn_n_neighbors | 5 |
| **SoftImpute** | |
| softimpute_shrinkage | null |
| **GRAPE** | |
| grape_epochs | 20,000 |
| grape_node_edge_dim | 128 |
| grape_lr | $9.044 \times 10^{-4}$ |
| **DiffPuter** | |
| lr | $1.031 \times 10^{-4}$ |
| hid_dim | 512 |
| gaussianize | True |
| n_em_iterations | 3 |
| n_train_epochs | 5,000 |
| n_sampling_trials | 10 |
| early_stopping_patience | 300 |

Table 4: Hyper-parameter Configurations for Scenario 33 trained on (MCAR $\times$ 30%)

| Parameters | Values |
|---|---|
| **kNN** | |
| knn_n_neighbors | 5 |
| **SoftImpute** | |
| softimpute_shrinkage | 3.5984 |
| **GRAPE** | |
| grape_epochs | 20,000 |
| grape_node_edge_dim | 128 |
| grape_lr | $1.442 \times 10^{-3}$ |
| **DiffPuter** | |
| lr | $1.902 \times 10^{-4}$ |
| hid_dim | 256 |
| gaussianize | True |
| n_em_iterations | 3 |
| n_train_epochs | 5,000 |
| n_sampling_trials | 10 |
| early_stopping_patience | 300 |

Table 5: Downstream model configuration [12]

| **Parameters** | **Values** |
|---|---|
| **Input size** | 2 ($g_{lat}$ and $g_{long}$) |
| **Hidden Layers** | 3 layers, 256 nodes each |
| **Output size** | 64 |
| **Activation** | ReLU on hidden layers |
| **Training Batch Size** | 32 |
| **Learning Rate** | $1 \times 10^{-2}$ |
| **Learning Rate Decay** | epochs 20 and 40 |
| **Learning Rate Reduction Factor** | 0.2 |
| **Total Training Epochs** | 60 |

Table 6: Reconstruction error over masked entries, reported as RMSE / MAE (lower is better). The standardised mean baseline gives $\mathrm{RMSE} \approx 1$. Bold marks the lowest RMSE in each row.

| | % | Mean | kNN | MICE | SoftImpute | HyperImpute | DiffPuter | GRAPE |
|---|---|---|---|---|---|---|---|---|
| *Scenario 5* | | | | | | | | |
| | 10 | 1.01/0.63 | 0.25/0.13 | **0.17**/0.09 | 0.20/0.11 | 0.17/0.09 | 0.26/0.13 | 0.26/0.16 |
| MCAR | 30 | 1.00/0.63 | 0.30/0.15 | 0.25/0.14 | 0.25/0.13 | **0.22**/0.12 | 0.29/0.14 | 0.27/0.18 |
| | 50 | 1.00/0.63 | 0.48/0.23 | 0.39/0.22 | 0.34/0.18 | **0.28**/0.15 | 0.38/0.18 | 0.30/0.19 |
| | 10 | 1.03/0.65 | 0.28/0.15 | 0.29/0.13 | 0.31/0.15 | 0.27/0.13 | 0.36/0.15 | **0.27**/0.17 |
| MAR | 30 | 1.02/0.63 | 0.44/0.23 | 0.40/0.20 | 0.38/0.19 | **0.31**/0.15 | 0.40/0.18 | 0.31/0.19 |
| | 50 | 1.01/0.63 | 0.59/0.30 | 0.53/0.28 | 0.49/0.26 | 0.40/0.19 | 0.59/0.26 | **0.38**/0.21 |
| | 10 | 2.51/1.82 | 0.94/0.53 | 1.04/0.51 | 1.37/0.79 | **0.63**/0.31 | 1.38/0.80 | 0.73/0.44 |
| MNAR | 30 | 1.73/1.04 | 1.17/0.59 | 1.31/0.65 | 1.30/0.66 | 0.71/0.33 | 1.46/0.72 | **0.65**/0.34 |
| | 50 | 1.41/0.77 | 1.15/0.56 | 1.22/0.60 | 1.16/0.57 | 0.75/0.34 | 1.30/0.61 | **0.69**/0.33 |
| | 10 | 1.01/0.63 | **0.63**/0.33 | 1.00/0.62 | 0.96/0.59 | 1.00/0.62 | 1.02/0.49 | 1.23/0.79 |
| MCAR-Row | 30 | 1.01/0.63 | **0.65**/0.35 | 1.00/0.62 | 0.97/0.59 | 1.00/0.62 | 1.03/0.49 | 2.65/2.13 |
| | 50 | 1.01/0.63 | **0.68**/0.36 | 0.99/0.62 | 0.97/0.60 | 1.00/0.62 | 1.04/0.50 | 1.81/1.30 |
| *Scenario 33* | | | | | | | | |
| | 10 | 1.00/0.79 | 0.22/0.13 | 0.19/0.12 | 0.22/0.14 | **0.19**/0.11 | 0.21/0.13 | 0.27/0.19 |
| MCAR | 30 | 1.00/0.79 | 0.23/0.14 | 0.25/0.16 | 0.26/0.17 | 0.24/0.15 | 0.23/0.14 | **0.22**/0.14 |
| | 50 | 1.00/0.79 | 0.27/0.16 | 0.35/0.23 | 0.32/0.21 | 0.26/0.16 | 0.30/0.18 | **0.24**/0.16 |
| | 10 | 1.00/0.79 | 0.21/0.13 | 0.19/0.12 | 0.22/0.14 | **0.19**/0.12 | 0.21/0.13 | 0.22/0.15 |
| MAR | 30 | 1.00/0.79 | 0.23/0.14 | 0.26/0.17 | 0.26/0.17 | 0.24/0.15 | 0.23/0.14 | **0.22**/0.15 |
| | 50 | 1.00/0.79 | 0.27/0.17 | 0.35/0.23 | 0.33/0.22 | 0.27/0.17 | 0.30/0.18 | **0.24**/0.16 |
| | 10 | 1.83/1.50 | 0.33/0.20 | 0.33/0.20 | 0.49/0.33 | 0.31/0.19 | 0.41/0.25 | **0.28**/0.19 |
| MNAR | 30 | 1.55/1.18 | 0.47/0.28 | 0.68/0.41 | 0.68/0.43 | 0.38/0.24 | 0.96/0.57 | **0.31**/0.20 |
| | 50 | 1.38/0.99 | 0.63/0.37 | 0.84/0.53 | 0.76/0.47 | 0.46/0.29 | 1.13/0.70 | **0.35**/0.22 |
| | 10 | 1.01/0.80 | **0.43**/0.27 | 0.98/0.77 | 0.93/0.73 | 1.08/0.79 | 1.04/0.70 | 1.35/1.04 |
| MCAR-Row | 30 | 1.01/0.79 | **0.45**/0.28 | 0.97/0.77 | 0.94/0.74 | 1.09/0.78 | 1.05/0.69 | 1.62/1.26 |
| | 50 | 1.00/0.78 | **0.49**/0.31 | 0.97/0.76 | 0.95/0.75 | 0.96/0.72 | 1.22/0.82 | 1.51/1.19 |

Table 7: Correlation distortion $\|C - \hat{C}\|_F$ (lower is better).

| | % | Mean | kNN | MICE | SoftImpute | HyperImpute | DiffPuter | GRAPE |
|---|---|---|---|---|---|---|---|---|
| *Scenario 5* | | | | | | | | |
| | 10 | 2.02 | 0.18 | 0.14 | 0.18 | **0.14** | 0.19 | 0.33 |
| MCAR | 30 | 5.82 | 0.51 | 0.35 | 0.56 | **0.34** | 0.59 | 0.89 |
| | 50 | 9.71 | 1.74 | 0.77 | 1.22 | **0.60** | 1.63 | 1.75 |
| | 10 | 2.14 | **0.24** | 0.41 | 0.51 | 0.36 | 0.57 | 0.34 |
| MAR | 30 | 5.33 | 1.13 | 0.88 | 1.04 | **0.60** | 1.41 | 1.03 |
| | 50 | 8.77 | 2.78 | 1.42 | 1.57 | **0.93** | 2.82 | 2.26 |
| | 10 | 7.93 | 1.45 | 1.33 | 2.39 | **0.92** | 2.85 | 1.13 |
| MNAR | 30 | 11.90 | 4.60 | 4.33 | 5.37 | **2.22** | 6.85 | 2.25 |
| | 50 | 14.28 | 7.35 | 6.40 | 7.92 | **3.47** | 8.87 | 4.01 |
| | 10 | 0.44 | 0.46 | 0.41 | **0.35** | 0.41 | 0.73 | 1.64 |
| MCAR-Row | 30 | 1.00 | 1.43 | 0.90 | **0.68** | 0.93 | 2.12 | 25.70 |
| | 50 | 1.85 | 2.76 | 1.67 | **1.28** | 1.72 | 3.58 | 16.19 |
| *Scenario 33* | | | | | | | | |
| | 10 | 3.64 | 0.20 | 0.16 | 0.35 | **0.16** | 0.18 | 0.39 |
| MCAR | 30 | 10.81 | 0.55 | 0.37 | 1.24 | **0.31** | 0.47 | 0.75 |
| | 50 | 17.99 | 1.20 | 0.99 | 2.69 | **0.70** | 1.54 | 1.34 |
| | 10 | 3.45 | 0.18 | 0.16 | 0.36 | **0.15** | 0.17 | 0.28 |
| MAR | 30 | 10.15 | 0.57 | 0.35 | 1.31 | **0.34** | 0.48 | 0.69 |
| | 50 | 16.92 | 1.31 | 0.84 | 2.76 | **0.75** | 1.64 | 1.15 |
| | 10 | 9.12 | 0.38 | 0.43 | 1.28 | 0.38 | 0.68 | **0.32** |
| MNAR | 30 | 18.17 | 1.50 | 2.24 | 4.31 | 1.01 | 6.83 | **0.85** |
| | 50 | 24.08 | 3.30 | 4.04 | 7.81 | 1.89 | 13.30 | **1.51** |
| | 10 | 0.48 | **0.32** | 0.46 | 0.37 | 1.10 | 0.76 | 2.90 |
| MCAR-Row | 30 | 1.33 | **0.98** | 1.30 | 1.01 | 2.97 | 1.94 | 8.23 |
| | 50 | 2.28 | 1.97 | 2.27 | **1.74** | 5.07 | 7.44 | 12.32 |

Table 8: Marginal Wasserstein-1 distance, averaged over columns (lower is better)

| | % | Mean | kNN | MICE | SoftImpute | HyperImpute | DiffPuter | GRAPE |
|---|---|---|---|---|---|---|---|---|
| *Scenario 5* | | | | | | | | |
| | 10 | 0.63 | 0.05 | 0.04 | 0.07 | **0.03** | 0.06 | 0.07 |
| MCAR | 30 | 0.63 | 0.05 | 0.05 | 0.08 | **0.03** | 0.05 | 0.09 |
| | 50 | 0.63 | 0.09 | 0.11 | 0.12 | **0.04** | 0.08 | 0.07 |
| | 10 | 0.65 | 0.06 | 0.07 | 0.10 | **0.06** | 0.08 | 0.08 |
| MAR | 30 | 0.63 | 0.09 | 0.11 | 0.13 | **0.06** | 0.08 | 0.06 |
| | 50 | 0.63 | 0.13 | 0.18 | 0.19 | 0.07 | 0.15 | **0.07** |
| | 10 | 1.82 | 0.44 | 0.43 | 0.76 | **0.22** | 0.67 | 0.34 |
| MNAR | 30 | 1.04 | 0.52 | 0.56 | 0.63 | **0.24** | 0.64 | 0.26 |
| | 50 | 0.77 | 0.48 | 0.51 | 0.52 | **0.25** | 0.56 | 0.25 |
| | 10 | 0.63 | **0.13** | 0.61 | 0.56 | 0.60 | 0.37 | 0.49 |
| MCAR-Row | 30 | 0.63 | **0.12** | 0.60 | 0.57 | 0.61 | 0.38 | 1.96 |
| | 50 | 0.63 | **0.13** | 0.60 | 0.58 | 0.61 | 0.40 | 1.00 |
| *Scenario 33* | | | | | | | | |
| | 10 | 0.79 | 0.04 | 0.03 | 0.08 | **0.03** | 0.04 | 0.09 |
| MCAR | 30 | 0.79 | 0.03 | 0.05 | 0.10 | **0.03** | 0.04 | 0.04 |
| | 50 | 0.79 | 0.04 | 0.11 | 0.13 | **0.04** | 0.06 | 0.05 |
| | 10 | 0.79 | 0.04 | 0.03 | 0.08 | **0.03** | 0.04 | 0.05 |
| MAR | 30 | 0.79 | 0.03 | 0.05 | 0.10 | **0.03** | 0.04 | 0.06 |
| | 50 | 0.79 | 0.05 | 0.11 | 0.14 | **0.04** | 0.06 | 0.05 |
| | 10 | 1.49 | 0.11 | 0.10 | 0.28 | 0.09 | 0.14 | **0.09** |
| MNAR | 30 | 1.18 | 0.19 | 0.31 | 0.37 | 0.12 | 0.49 | **0.11** |
| | 50 | 0.99 | 0.27 | 0.43 | 0.41 | 0.16 | 0.63 | **0.11** |
| | 10 | 0.80 | **0.09** | 0.71 | 0.68 | 0.20 | 0.49 | 0.81 |
| MCAR-Row | 30 | 0.79 | **0.08** | 0.70 | 0.70 | 0.15 | 0.47 | 1.01 |
| | 50 | 0.78 | **0.09** | 0.70 | 0.71 | 0.39 | 0.52 | 1.01 |

Table 9: Variance retention, ratio to ground-truth variance (1.0 ideal; values may fall either side).

| | % | Mean | kNN | MICE | SoftImpute | HyperImpute | DiffPuter | GRAPE |
|---|---|---|---|---|---|---|---|---|
| *Scenario 5* | | | | | | | | |
| | 10 | 0.00 | 0.86 | **0.97** | 0.82 | 0.97 | 1.06 | 0.93 |
| MCAR | 30 | 0.00 | 0.81 | 0.90 | 0.77 | **0.95** | 1.07 | 0.90 |
| | 50 | 0.00 | 0.66 | 0.74 | 0.66 | 0.91 | **1.04** | 0.88 |
| | 10 | 0.00 | 0.86 | 0.93 | 0.77 | 0.93 | **1.01** | 0.91 |
| MAR | 30 | 0.00 | 0.71 | 0.79 | 0.67 | 0.86 | **0.97** | 0.90 |
| | 50 | 0.00 | 0.64 | 0.62 | 0.56 | 0.79 | 0.73 | **0.89** |
| | 10 | 0.00 | 0.58 | 0.60 | 0.33 | **0.76** | 0.42 | 0.66 |
| MNAR | 30 | 0.00 | 0.23 | 0.21 | 0.13 | 0.54 | 0.15 | **0.55** |
| | 50 | 0.00 | 0.11 | 0.10 | 0.08 | 0.40 | 0.07 | **0.40** |
| | 10 | 0.00 | **0.69** | 0.00 | 0.02 | 0.01 | 0.08 | 0.57 |
| MCAR-Row | 30 | 0.00 | **0.65** | 0.00 | 0.01 | 0.00 | 0.08 | 1.75 |
| | 50 | 0.00 | 0.63 | 0.00 | 0.01 | 0.00 | 0.08 | **1.10** |
| *Scenario 33* | | | | | | | | |
| | 10 | 0.00 | 0.95 | 0.97 | 0.83 | 0.97 | **0.99** | 0.90 |
| MCAR | 30 | 0.00 | 0.94 | 0.91 | 0.78 | 0.97 | **0.98** | 0.96 |
| | 50 | 0.00 | 0.91 | 0.78 | 0.71 | 0.93 | 0.93 | **0.94** |
| | 10 | 0.00 | 0.96 | 0.97 | 0.82 | 0.97 | **0.99** | 0.95 |
| MAR | 30 | 0.00 | 0.94 | 0.92 | 0.77 | 0.97 | **0.97** | 0.95 |
| | 50 | 0.00 | 0.91 | 0.78 | 0.70 | **0.93** | 0.92 | 0.92 |
| | 10 | 0.00 | 0.92 | 0.92 | 0.67 | 0.93 | 0.89 | **0.93** |
| MNAR | 30 | 0.00 | 0.74 | 0.57 | 0.45 | 0.84 | 0.41 | **0.89** |
| | 50 | 0.00 | 0.54 | 0.33 | 0.31 | 0.69 | 0.17 | **0.83** |
| | 10 | 0.00 | **0.83** | 0.02 | 0.03 | 0.79 | 0.16 | 0.44 |
| MCAR-Row | 30 | 0.00 | 0.83 | 0.02 | 0.02 | **0.85** | 0.18 | 0.34 |
| | 50 | 0.00 | **0.81** | 0.02 | 0.01 | 0.37 | 0.57 | 0.38 |

Table 10: Mean shift, absolute displacement of per-column means (lower is better).

| | % | Mean | kNN | MICE | SoftImpute | HyperImpute | DiffPuter | GRAPE |
|---|---|---|---|---|---|---|---|---|
| *Scenario 5* | | | | | | | | |
| | 10 | 0.05 | 0.02 | 0.01 | 0.01 | **0.01** | 0.03 | 0.04 |
| MCAR | 30 | 0.04 | 0.03 | 0.01 | **0.01** | 0.01 | 0.03 | 0.05 |
| | 50 | 0.04 | 0.05 | 0.01 | **0.01** | 0.01 | 0.05 | 0.03 |
| | 10 | 0.12 | **0.03** | 0.03 | 0.04 | 0.03 | 0.05 | 0.03 |
| MAR | 30 | 0.09 | 0.03 | 0.03 | 0.03 | **0.02** | 0.06 | 0.03 |
| | 50 | 0.07 | 0.04 | 0.03 | **0.03** | 0.03 | 0.12 | 0.05 |
| | 10 | 1.65 | 0.43 | 0.42 | 0.73 | **0.21** | 0.66 | 0.30 |
| MNAR | 30 | 0.86 | 0.50 | 0.55 | 0.55 | 0.23 | 0.64 | **0.22** |
| | 50 | 0.60 | 0.46 | 0.47 | 0.42 | 0.24 | 0.55 | **0.22** |
| | 10 | 0.06 | **0.04** | 0.06 | 0.06 | 0.06 | 0.30 | 0.37 |
| MCAR-Row | 30 | 0.03 | **0.02** | 0.03 | 0.02 | 0.03 | 0.32 | 1.92 |
| | 50 | 0.04 | 0.03 | 0.04 | **0.02** | 0.04 | 0.32 | 0.88 |
| *Scenario 33* | | | | | | | | |
| | 10 | 0.04 | 0.02 | 0.01 | 0.01 | **0.01** | 0.02 | 0.07 |
| MCAR | 30 | 0.03 | 0.02 | 0.01 | **0.01** | 0.01 | 0.02 | 0.02 |
| | 50 | 0.03 | 0.03 | 0.01 | **0.01** | 0.01 | 0.04 | 0.03 |
| | 10 | 0.05 | 0.02 | 0.01 | 0.01 | **0.01** | 0.02 | 0.02 |
| MAR | 30 | 0.03 | 0.02 | 0.01 | **0.01** | 0.01 | 0.02 | 0.05 |
| | 50 | 0.03 | 0.03 | 0.01 | **0.01** | 0.01 | 0.04 | 0.04 |
| | 10 | 1.29 | 0.11 | 0.09 | 0.25 | 0.08 | 0.13 | **0.06** |
| MNAR | 30 | 0.97 | 0.19 | 0.30 | 0.34 | 0.11 | 0.49 | **0.09** |
| | 50 | 0.78 | 0.27 | 0.40 | 0.35 | 0.16 | 0.63 | **0.08** |
| | 10 | 0.04 | **0.02** | 0.04 | 0.03 | 0.04 | 0.37 | 0.65 |
| MCAR-Row | 30 | 0.02 | **0.01** | 0.02 | 0.01 | 0.02 | 0.36 | 0.81 |
| | 50 | 0.03 | 0.02 | 0.03 | **0.02** | 0.03 | 0.46 | 0.87 |

Table 11: Downstream beam-prediction accuracy over masked entries, reported as Top-1 / Top-5 (higher is better). Bold marks the best imputer in each row at each $K$. *Clean* ceiling (fully observed): $0.401/0.883$ for Scenario 5 and $0.478/0.925$ for Scenario 33.

| | % | Mean | kNN | MICE | SoftImpute | HyperImpute | DiffPuter | GRAPE |
|---|---|---|---|---|---|---|---|---|
| *Scenario 5* | | | | | | | | |
| MCAR | 10 | 0.399/0.883 | 0.391/0.878 | **0.400**/0.894 | 0.382/0.877 | 0.393/**0.907** | 0.392/0.884 | 0.383/0.896 |
| | 30 | 0.378/0.870 | 0.379/0.871 | 0.388/0.896 | 0.380/0.872 | **0.402**/0.893 | 0.367/0.890 | 0.400/**0.906** |
| | 50 | 0.347/0.849 | 0.347/0.840 | **0.388/0.897** | 0.360/0.858 | 0.382/0.887 | 0.333/0.842 | 0.386/0.882 |
| MAR | 10 | 0.392/0.878 | 0.387/0.880 | 0.387/0.899 | 0.378/0.881 | **0.396**/0.901 | 0.380/**0.904** | 0.373/0.892 |
| | 30 | 0.363/0.868 | 0.371/0.869 | 0.387/**0.900** | 0.357/0.870 | 0.387/0.892 | 0.379/0.887 | **0.396**/0.892 |
| | 50 | 0.329/0.828 | 0.328/0.831 | 0.355/**0.882** | 0.325/0.836 | 0.373/0.870 | 0.336/0.833 | **0.381**/0.877 |
| MNAR | 10 | 0.391/0.882 | 0.393/0.883 | 0.392/0.891 | **0.398**/0.881 | 0.393/0.884 | 0.383/0.893 | 0.394/**0.899** |
| | 30 | 0.371/0.868 | 0.382/0.870 | 0.379/0.902 | 0.363/0.864 | **0.396/0.911** | 0.365/0.871 | 0.382/0.909 |
| | 50 | 0.338/0.848 | 0.325/0.842 | 0.362/0.875 | 0.336/0.853 | **0.394/0.891** | 0.324/0.833 | 0.369/0.887 |
| MCAR-Row | 10 | 0.386/0.851 | 0.395/0.880 | 0.393/0.870 | 0.382/0.849 | **0.400**/0.881 | 0.399/**0.892** | 0.375/0.880 |
| | 30 | 0.159/0.810 | **0.379/0.863** | 0.182/0.845 | 0.162/0.799 | 0.302/0.811 | 0.356/0.853 | 0.264/0.796 |
| | 50 | 0.041/0.560 | **0.349/0.831** | 0.048/0.813 | 0.039/0.565 | 0.119/0.650 | 0.258/0.692 | 0.053/0.728 |
| *Scenario 33* | | | | | | | | |
| MCAR | 10 | 0.476/0.906 | 0.477/0.928 | 0.510/0.933 | 0.477/0.926 | 0.511/**0.937** | 0.504/0.936 | **0.511/0.937** |
| | 30 | 0.453/0.884 | 0.472/0.923 | 0.498/0.934 | 0.474/0.919 | **0.518/0.941** | 0.474/0.924 | 0.497/0.936 |
| | 50 | 0.392/0.872 | 0.461/0.919 | 0.472/0.924 | 0.442/0.901 | **0.500/0.939** | 0.448/0.899 | 0.474/0.933 |
| MAR | 10 | 0.478/0.904 | 0.481/0.929 | **0.514**/0.934 | 0.481/0.928 | 0.495/0.935 | 0.512/**0.939** | 0.501/0.934 |
| | 30 | 0.451/0.879 | 0.471/0.923 | 0.489/0.934 | 0.469/0.918 | 0.488/0.931 | 0.487/0.929 | **0.490/0.938** |
| | 50 | 0.380/0.866 | 0.460/0.918 | 0.482/0.926 | 0.430/0.897 | 0.482/**0.942** | 0.446/0.891 | **0.483**/0.929 |
| MNAR | 10 | 0.479/0.908 | 0.483/0.926 | 0.502/0.935 | 0.477/0.925 | **0.515**/0.938 | 0.499/0.933 | 0.507/**0.941** |
| | 30 | 0.447/0.882 | 0.468/0.922 | 0.494/0.933 | 0.467/0.919 | **0.506**/0.932 | 0.471/0.922 | 0.498/**0.934** |
| | 50 | 0.376/0.869 | 0.454/0.919 | 0.430/0.919 | 0.422/0.903 | **0.493**/0.925 | 0.421/0.904 | 0.478/**0.928** |
| MCAR-Row | 10 | 0.468/0.899 | 0.481/**0.927** | 0.495/0.909 | 0.469/0.904 | 0.499/0.913 | **0.505**/0.927 | 0.505/0.918 |
| | 30 | 0.316/0.866 | **0.478/0.925** | 0.397/0.882 | 0.343/0.864 | 0.412/0.880 | 0.476/0.888 | 0.375/0.875 |
| | 50 | 0.174/0.748 | **0.477/0.917** | 0.276/0.825 | 0.213/0.830 | 0.289/0.838 | 0.391/0.818 | 0.174/0.841 |

# References


[1] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG]. URL: https://arxiv.org/abs/1907.10902.

[2] Ahmed Alkhateeb et al. "DeepSense 6G: A Large-Scale Real-World Multi-Modal Sensing and Communication Dataset". In: *IEEE Communications Magazine* 61.9 (2023), pp. 122–128. DOI: 10.1109/MCOM.006.2200730.

[3] Tianfeng Chai, Roland R Draxler, et al. "Root mean square error (RMSE) or mean absolute error (MAE)". In: *Geoscientific model development discussions* 7.1 (2014), pp. 1525–1534.

[4] Gouranga Charan, Muhammad Alrabeiah, and Ahmed Alkhateeb. "Vision-Aided 6G Wireless Communications: Blockage Prediction and Proactive Handoff". In: *IEEE Transactions on Vehicular Technology* 70.10 (2021), pp. 10193–10208. DOI: 10.1109/TVT.2021.3104219.

[5] Anil Jadhav, Dhanya Pramod, and Krishnan Ramanathan. "Comparison of performance of data imputation methods for numeric dataset". In: *Applied Artificial Intelligence* 33.10 (2019), pp. 913–933.

[6] Sebastian Jäger, Arndt Allhorn, and Felix Bießmann. "A benchmark for data imputation methods". In: *Frontiers in big Data* 4 (2021), p. 693674.

[7] Daniel Jarrett et al. *HyperImpute: Generalized Iterative Imputation with Automatic Model Selection*. 2022. arXiv: 2206.07769 [stat.ML]. URL: https://arxiv.org/abs/2206.07769.

[8] Tianyu Jiang et al. *Fast Time-Varying mmWave MIMO Channel Estimation and Reconstruction: An Efficient Rank-Aware Matrix Completion Method*. 2025. arXiv: 2511.05902 [eess.SP]. URL: https://arxiv.org/abs/2511.05902.

[9] Khaled B. Letaief et al. "The Roadmap to 6G: AI Empowered Wireless Networks". In: *IEEE Communications Magazine* 57.8 (2019), pp. 84–90. DOI: 10.1109/MCOM.2019.1900271.

[10] Pierre-Alexandre Mattei and Jes Frellsen. "MIWAE: Deep generative modelling and imputation of incomplete data sets". In: *International conference on machine learning*. PMLR. 2019, pp. 4413–4423.

[11] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. "Spectral regularization algorithms for learning large incomplete matrices". In: *The Journal of Machine Learning Research* 11 (2010), pp. 2287–2322.

[12] João Morais et al. *Position Aided Beam Prediction in the Real World: How Useful GPS Locations Actually Are?* 2022. arXiv: 2205.09054 [eess.SP]. URL: https://arxiv.org/abs/2205.09054.

[13] Ivan Rubachev et al. "TabReD: Analyzing Pitfalls and Filling the Gaps in Tabular Deep Learning Benchmarks". In: *International Conference on Learning Representations*. Ed. by Y. Yue et al. Vol. 2025. 2025, pp. 35166–35202. URL: https://proceedings.iclr.cc/paper_files/paper/2025/file/571799482291411607c54984153190b0-Paper-Conference.pdf.

[14] Donald B Rubin. "Inference and missing data". In: *Biometrika* 63.3 (1976), pp. 581–592.

[15] Walid Saad, Mehdi Bennis, and Mingzhe Chen. "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems". In: *IEEE Network* 34.3 (2020), pp. 134–142. DOI: 10.1109/MNET.001.1900287.

[16] Mehran Soltani et al. *Deep Learning-Based Channel Estimation*. 2019. arXiv: 1810.05893 [cs.IT]. URL: https://arxiv.org/abs/1810.05893.

[17] Daniel J Stekhoven and Peter Bühlmann. "MissForest-non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1 (2012), pp. 112–118.

[18] Olga Troyanskaya et al. "Missing value estimation methods for DNA microarrays". In: *Bioinformatics* 17.6 (2001), pp. 520–525.

[19] Stef Van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate imputation by chained equations in R". In: *Journal of statistical software* 45 (2011), pp. 1–67.

[20] Mark D Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific data* 3.1 (2016), pp. 1–9.

[21] Jinsung Yoon, James Jordon, and Mihaela Schaar. "Gain: Missing data imputation using generative adversarial nets". In: *International conference on machine learning*. PMLR. 2018, pp. 5689–5698.

[22] Jiaxuan You et al. "Handling missing data with graph representation learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19075–19087.

[23] Hengrui Zhang et al. "DiffPuter: An EM-Driven Diffusion Model for Missing Data Imputation". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=3fl1SENSYO.