LEARNING WHEN STRUCTURE-BASED SEMANTICS HELP TABULAR IMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained language models provide rich semantic priors for tabular imputation, but their utility is inherently conditional: general semantic associations may conflict with dataset-specific dependencies or become redundant when the observed values already provide sufficient evidence. Existing language-enhanced imputers therefore face a fundamental mismatch between general-purpose semantic priors and the instance-specific evidence required for each missing entry. To address this mismatch, we propose **SemImputer**, a structure-based semantic representation imputation framework that learns when semantic priors provide complementary evidence for each missing-value prediction. Specifically, SemImputer first aligns semantic representations with tabular structures through structure-semantic representation alignment, then encodes the missingness context for each target through context-aware missing pattern encoder, and finally learns whether semantic information improves each prediction through counterfactual utility-guided fusion for selective semantic injection. Across 10 real-world datasets, SemImputer achieves state-of-the-art performance, reducing RMSE by 5.4% and improving R^2 by 16.1% on average over the strongest baseline, while remaining robust across heterogeneous missingness mechanisms without mechanism-specific retraining.

1 INTRODUCTION

Tabular imputation estimates missing entries from the observed portion of an incomplete table Hollmann et al. (2025); Yang et al. (2025a); Luo et al. (2026). Missing values reduce the evidence available for downstream analysis and prediction Gorla et al. (2025); Wang et al. (2025); Zahmatkesh & Zech (2026). The task becomes particularly challenging when missingness removes the fields most informative about a target, leaving the remaining observations consistent with multiple plausible values. Imputation therefore depends on learning dataset-specific dependencies and identifying complementary evidence. Field names, descriptions, and observed values provide semantic information that pretrained language models can encode as priors for imputation Mei et al. (2021); Gorla et al. (2025); Wang et al. (2025). Combining these priors with dependencies learned from the table offers a way to improve predictions when observed evidence is sparse.

Existing methods model tabular dependencies through predictive, reconstruction-based, or generative approaches, with language-enhanced methods adding semantic priors to improve missing-value prediction accuracy. MICE and KPI predict missing fields from other variables, while ReMasker reconstructs masked entries Van Buuren & Groothuis-Oudshoorn (2011); Wang et al. (2026); Du et al. (2024). GAIN and Meta-GAIN learn completions through adversarial objectives, and DiffPuter uses diffusion Yoon et al. (2018); Tong et al. (2026); Zhang et al. (2025a). Among language-enhanced methods, CACTI integrates column context into masked reconstruction under empirical missingness patterns Gorla et al. (2025), whereas UnIMP fuses language features with high-order dependencies aggregated over cell-oriented hypergraphs Wang et al. (2025).

However, these language-enhanced methods optimize the accuracy of fused predictions without directly supervising the incremental gain from semantic injection for each target Gorla et al. (2025); Wang et al. (2025). Task-specific adaptation can improve the relevance of language representations for tabular prediction Arazi et al. (2026). A semantic prior may agree with the table's dependencies yet provide little additional predictive value given the observed fields. Figure 1 illustrates how semantic priors can be redundant, complementary, or irrelevant as the target and observed evidence

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

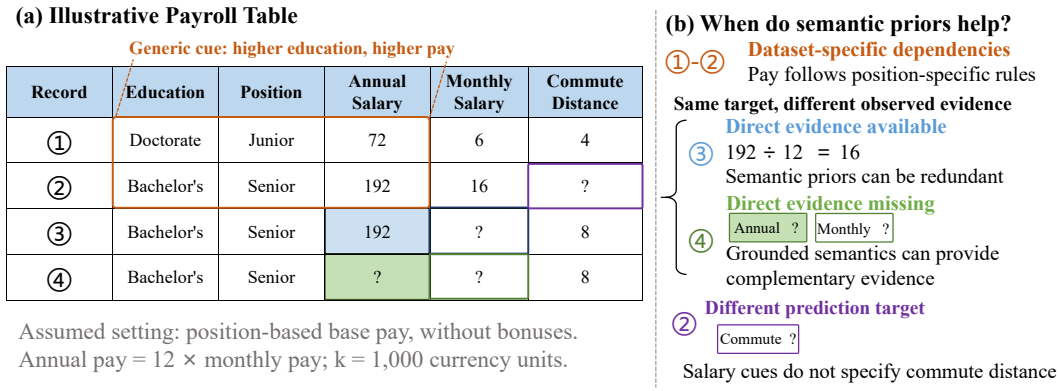


Figure 1: **Motivation for SemImputer**. The illustrative payroll example highlights the need to ground semantic priors in dataset-specific dependencies and learn their additional predictive value for each target and observation context.

change. Yet existing training objectives do not explicitly distinguish among these target-dependent cases. This limitation arises because the reconstruction objective typically used to train the fused predictor evaluates only the final prediction, without comparing the same target with and without semantic injection under the same observed evidence. Consequently, the incremental predictive value of semantics remains implicit. This motivates grounding semantic priors in table structure while separately supervising their target-wise predictive utility.

To address this limitation, we propose **SemImputer** (Figure 1), which learns when semantic priors improve a missing-value prediction given the observed fields. Since general-purpose semantic associations may diverge from the dependencies of a particular table, we first propose a **Structure-Semantic Representation Alignment** module to align semantic and structural representations at both the field and row levels. Because the evidence available for predicting a target varies with the current missingness pattern, we then design a **Context-Aware Missing Pattern Encoder** that encodes the row mask and observed-value summaries to construct target-conditioned context for utility estimation and prediction. Even after structural alignment, semantic information may provide little additional predictive value beyond the observed fields. We therefore further propose a **Counterfactual Utility-Guided Fusion** mechanism, which compares matched predictions for the same missing target with and without semantic injection to derive target-wise utility supervision and selectively control semantic contribution. For robust utility learning across diverse observation contexts, we design an **Interleaved Missingness-View Training** strategy that exposes the model to varying missingness views across epochs. A curriculum further shifts training from representation alignment toward reconstruction and utility learning. In summary, our main contributions are as follows:

- We propose SemImputer, a structure-aware semantic imputation framework that grounds semantic priors in dataset-specific dependencies, models the observation context, and selectively injects them according to predictive utility for each missing value.
- We propose Counterfactual Utility-Guided Fusion mechanism, which explicitly measures whether semantic information improves a target prediction by comparing matched predictions with and without semantic injection, and uses this target-wise utility signal to control when and how much semantic information should contribute.
- We develop Interleaved Missingness-View Training, which exposes the model to diverse missingness mechanisms and rates during training, enabling a single model to generalize across heterogeneous missingness settings without mechanism- or rate-specific retraining.
- Extensive experiments on 10 real-world datasets demonstrate that SemImputer achieves consistent state-of-the-art performance across heterogeneous missingness settings, validating the effectiveness of the proposed framework.

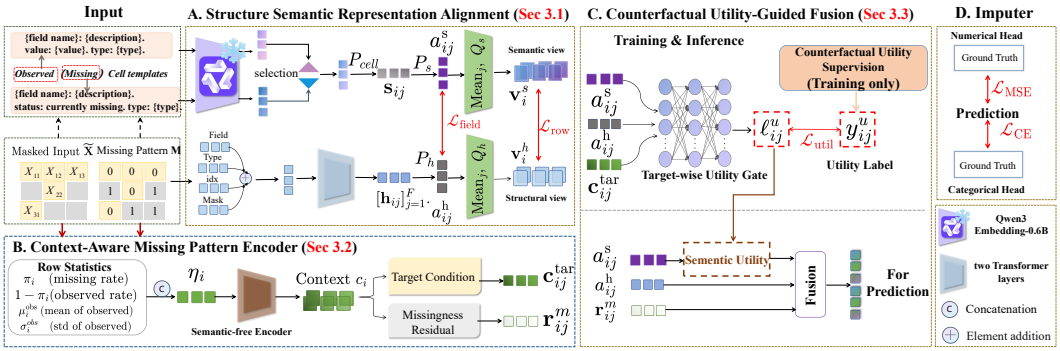


Figure 2: **Overview of SemImputer.** SemImputer separates structural compatibility from predictive utility when incorporating semantic priors for tabular imputation.

2 Related Works

Predictive and Reconstruction-Based Tabular Imputation. Predictive methods estimate missing fields from observed variables. MICE uses chained conditional models and MissForest uses random forests Van Buuren & Groothuis-Oudshoorn (2011); Stekhoven & Bühlmann (2012). HyperImpute automates predictive model selection Jarrett et al. (2022), while KPI adapts the model form for each field Wang et al. (2026). ReMasker learns cross-field dependencies through masked reconstruction Du et al. (2024), and PMAE matches masking proportions to empirical missingness statistics Kim et al. (2025). These methods derive predictive evidence from observed table values.

Generative Tabular Imputation. Generative methods model **joint** data distributions to infer missing values **conditioned on observed entries**. GAIN uses adversarial learning Yoon et al. (2018), while Meta-GAIN adds regularization and meta-optimization to improve imputation diversity and generalization Tong et al. (2026). NewImp uses negative-entropy regularization to suppress unintended imputation diversity Chen et al. (2024). DiffPuter combines diffusion with expectation–maximization Zhang et al. (2025a), whereas RefiDiff couples predictive refinement with diffusion Ahamed et al. (2026). CtrTab addresses high-dimensional, small-sample data synthesis through controlled diffusion Li et al. (2026), providing an adjacent generative baseline.

Language-Enhanced Tabular Imputation. Language models provide complementary evidence by linking sparse observations to plausible values and encoding relations conveyed by field names and descriptions. ConTextTab integrates semantic representations into tabular in-context prediction Spinaci et al. (2026). For imputation, IPM predicts or ranks candidate missing values with fine-tuned language models Mei et al. (2021), while Jellyfish instruction-tunes LLMs for tabular preprocessing tasks including imputation Zhang et al. (2024). CACTI uses column context and copy masking for reconstruction Gorla et al. (2025), whereas UnIMP fuses language representations with high-order hypergraph message passing Wang et al. (2025). CACTI and UnIMP optimize imputation accuracy without separately supervising target-wise semantic gains. In contrast, SemImputer combines explicit semantic–structural alignment with target-wise utility supervision to learn when semantic priors provide additional predictive value under the current observation context.

3 SemImputer

We next describe SemImputer, with its overall architecture illustrated in Figure 2. We begin by formalizing the imputation problem and the missingness settings considered in this work.

Problem formulation. For a batch of B rows and F fields, let $\widetilde{\mathbf{X}} \in \mathbb{R}^{B\times D}$ denote the encoded masked input obtained by concatenating preprocessed numerical values and one-hot categorical vectors, where D is the encoded width. Let $\mathbf{M} = [m_{ij}] \in \{0,1\}^{B\times F}$ be the field mask, with $m_{ij} = 1$ indicating a masked field; for categorical fields, the same mask applies to all one-hot coordinates.

We use $t_j \in \{\text{num}, \text{cat}\}$ to denote the field type and x_{ij} for its numerical value or categorical label. The masked targets are $\Omega = \{(i,j) : m_{ij} = 1\}$, whose ground-truth values are retained for supervision during training. Given $\widetilde{\mathbf{X}}$, $\mathbf{M}$, and semantic priors derived from field metadata and observed values, SemImputer predicts $\hat{x}_{ij}$ for each $(i,j) \in \Omega$, using scalar outputs for numerical fields and class probabilities $\widehat{\mathbf{p}}_{ij}$ for categorical fields.

Missingness mechanisms. We consider four missingness settings: MCAR, MAR, MNAR, and MIXED. Following Rubin's taxonomy Rubin (1976); Little & Rubin (2019), MCAR generates missingness independently of the data, MAR allows missingness to depend on observed values but not on unobserved values given the observed data, and MNAR allows missingness to depend on the unobserved values themselves. To model coexisting missingness processes, MIXED follows the multi-mechanism setting of Shades-of-Null Khan et al. (2025): for each target missing rate, we generate MCAR, MAR, and MNAR masks, partition the fields into three approximately equal groups, and assign each group its corresponding mechanism-specific mask.

Semantic input construction. For each cell, we construct a text template containing the field name, description, and type, together with its original value when observed or a missing-status marker when masked. A frozen Qwen3-Embedding-0.6B encoder (Zhang et al., 2025b) embeds these templates offline. The current mask selects the corresponding cached embedding, which a trainable projection P_{cell} maps to a semantic token $\mathbf{s}_{ij} \in \mathbb{R}^d$, where d is the model width. Appendix B.1 provides the templates, an example, and embedding dimensions.

3.1 Structure-Semantic Representation Alignment

Structure-Semantic Representation Alignment (Figure 2) matches semantic and structural representations at field and row levels. Value projections are combined with field identity, type, and observation-state embeddings. A Transformer applies self-attention across fields within each masked row to produce structural tokens $\mathbf{h}_{ij} \in \mathbb{R}^d$, one per original field.

Learned linear projections give $\mathbf{a}^s_{ij} = P_s(\mathbf{s}_{ij})$ and $\mathbf{a}^h_{ij} = P_h(\mathbf{h}_{ij})$. Field alignment averages cosine distances over observed and masked fields:

$$\mathcal{L}_{\text{field}} = \frac{1}{BF} \sum_{i=1}^{B} \sum_{j=1}^{F} [1 - \cos(\mathbf{a}^s_{ij}, \mathbf{a}^h_{ij})], \tag{1}$$

where cos denotes cosine similarity.

For row alignment, learned projections of mean-pooled field representations give $\mathbf{v}^s_i = Q_s(F^{-1} \sum_j \mathbf{a}^s_{ij})$ and $\mathbf{v}^h_i = Q_h(F^{-1} \sum_j \mathbf{a}^h_{ij})$. Symmetric InfoNCE (Oord et al., 2018) uses same-row pairs as positives and other batch rows as negatives:

$$\mathcal{L}_{\text{row}} = -\frac{1}{2B} \sum_{i=1}^{B} \left[\log \frac{\exp(sim_{ii})}{\sum_{k=1}^{B} \exp(sim_{ik})} + \log \frac{\exp(sim_{ii})}{\sum_{k=1}^{B} \exp(sim_{ki})} \right], \tag{2}$$

where $sim_{ik} = \cos(\mathbf{v}^s_i, \mathbf{v}^h_k)/\tau$ and $\tau > 0$ is the temperature. The aligned field tokens feed Counterfactual Utility-Guided Fusion; row projections provide training supervision. Layer configurations and hyperparameters are detailed in Appendix B.3.

3.2 Context-Aware Missing Pattern Encoder

Semantic utility varies with the available observations. The Context-Aware Missing Pattern Encoder computes:

$$\mathbf{c}_i = C_m(\mathbf{m}_i, \boldsymbol{\eta}_i), \qquad \mathbf{c}^{\text{tar}}_{ij} = P_c([P_g(\mathbf{c}_i); m_{ij}; \text{onehot}(t_j)]), \tag{3}$$

where $\mathbf{m}_i$ is the row mask, $\boldsymbol{\eta}_i$ contains missing and observed fractions and the within-row mean and standard deviation of observed field scalars, C_m is an MLP, and P_g, P_c are learned mappings; brackets denote concatenation. The context $\mathbf{c}_i \in \mathbb{R}^d$ conditions utility estimation through $\mathbf{c}^{\text{tar}}_{ij}$ and supplies a projected fusion residual $\mathbf{r}^m_{ij}$, with a gate conditioned on $\mathbf{h}_{ij}, \mathbf{c}_i, m_{ij}$ (Appendix B.4).

3.3 COUNTERFACTUAL UTILITY-GUIDED FUSION

Even aligned semantics may add little beyond observed evidence. Counterfactual Utility-Guided Fusion explicitly supervises this predictive gain, which reconstruction alone leaves implicit.

Counterfactual utility supervision. For each known masked target $(i, j) \in \Omega$, matched predictions P_0, P_1 reuse one upstream encoding and the same fusion and prediction parameters. During paired evaluation, dropout is disabled in fusion and prediction heads, so the pair differs only in semantic scales: 0 versus $m_{ij}\lambda_{t_j}$ in Eq. equation 6. The learned cap $\lambda_{t_j} \in (0, 1)$ sets the maximum injection for each field type. The zero-injection prediction retains the parameters learned through Structure-Semantic Representation Alignment. Their errors $e_{ij}^{(b)}$, $b \in \{0, 1\}$, compare predictions with x_{ij} using Huber loss ($\delta_H = 1$) Huber (1992) for preprocessed numerical targets and cross-entropy for categorical fields. Positive utility requires both relative and absolute error reductions:

$$y_{ij}^{u} = \mathbb{I}\Big[\Delta_{ij}^{\mathrm{rel}} > \delta_{\mathrm{rel}} \wedge \Delta_{ij}^{\mathrm{abs}} > \delta_{t_j}^{\mathrm{abs}}\Big], \quad (i, j) \in \Omega, \tag{4}$$

where $\mathbb{I}$ is the indicator, $\Delta_{ij}^{\mathrm{abs}} = e_{ij}^{(0)} - e_{ij}^{(1)}$, and $\Delta_{ij}^{\mathrm{rel}} = \Delta_{ij}^{\mathrm{abs}}/(e_{ij}^{(0)} + e_{ij}^{(1)} + \epsilon_u)$ with $\epsilon_u = 10^{-6}$. Fixed margins $\delta_{\mathrm{rel}} = 0.05$ and $\delta_{\mathrm{num}}^{\mathrm{abs}} = \delta_{\mathrm{cat}}^{\mathrm{abs}} = 0.02$ exclude negligible gains. Endpoints and labels are recomputed without gradients.

Utility-gated residual fusion. The gate predicts the utility label from aligned tokens and observation context:

$$\ell_{ij}^{u} = G_\phi([\mathrm{sg}(\mathbf{a}_{ij}^{s}); \mathrm{sg}(\mathbf{a}_{ij}^{h}); \mathbf{c}_{ij}^{\mathrm{tar}}]), \qquad \rho_{ij} = \sigma(\ell_{ij}^{u}), \tag{5}$$

where G_ϕ is an MLP, σ is sigmoid, and sg stops gradients. Binary cross-entropy $\mathcal{L}_{\mathrm{util}}$ trains ℓ_{ij}^{u} against y_{ij}^{u}. The score ρ_{ij} controls injection:

$$\begin{aligned} \gamma_{ij} &= m_{ij}\,\mathrm{sg}(\rho_{ij})\lambda_{t_j}, \\ \widetilde{\mathbf{h}}_{ij} &= \mathrm{LN}\big(\mathbf{a}_{ij}^{h} + \mathbf{r}_{ij}^{m} + \gamma_{ij}\,\mathrm{LN}_0(P_f(\mathbf{a}_{ij}^{s}))\big)\,, \end{aligned} \tag{6}$$

where P_f transforms semantic tokens, LN_0 is affine-free LayerNorm, and LN has learned affine parameters. Normalization bounds the injected semantic residual's root-mean-square magnitude by γ_{ij} before the outer LayerNorm and prediction heads. Utility gradients update G_ϕ, P_g, P_c, C_m, with aligned tokens detached. Reconstruction updates the tabular encoder, $P_{\mathrm{cell}}, P_s, P_h, P_f$, type caps, prediction heads, and the context-residual path. Detaching ρ_{ij} blocks reconstruction gradients to the gate; C_m receives gradients from both losses. Field-wise heads produce numerical predictions or categorical probabilities. Inference requires one gated prediction. Appendix B.5 gives initialization, loss reduction, and the bound derivation.

3.4 INTERLEAVED MISSINGNESS-VIEW TRAINING

Because semantic utility depends on the available observed evidence, the same target may benefit differently under different missingness patterns. Interleaved Missingness-View Training therefore exposes each row to diverse observation contexts during training. We precompute 16 mask views per dataset, covering four missingness mechanisms at rates of 10%, 20%, 30%, and 40%. At each epoch, every row independently samples one view uniformly and contributes once, with assignments resampled in the next epoch and batches mixing different views. No view identifiers are provided to the model. Validation reconstruction loss under fixed masks selects one checkpoint per dataset, which is reused unchanged at test rates of 30%, 40%, and 50%; the 50% setting is unseen during training. Algorithm 1 (Appendix B.7) provides the pseudocode.

3.5 LOSS FUNCTION

At epoch e, scheduled weights balance reconstruction, alignment, and utility losses:

$$\begin{aligned} \mathcal{L}(e) = {} & w_n(e)\mathcal{L}_{\mathrm{num}} + w_c(e)\mathcal{L}_{\mathrm{cat}} + w_m(e)\mathcal{L}_{\mathrm{mmm}} \\ & + w_f(e)\mathcal{L}_{\mathrm{field}} + w_r(e)\mathcal{L}_{\mathrm{row}} + w_u(e)\mathcal{L}_{\mathrm{util}}, \end{aligned} \tag{7}$$

$\mathcal{L}_{\mathrm{num}}$ averages MSE over masked numerical targets. $\mathcal{L}_{\mathrm{cat}}$ averages cross-entropy over masked targets within each categorical field, then equally across supervised fields. The marginal moment-matching loss $\mathcal{L}_{\mathrm{mmm}}$ matches field-wised prediction and target means and standard deviations over

masked numerical entries within each batch, discouraging marginal shrinkage. Reconstruction remains active throughout training. The schedule later decreases alignment weights while increasing reconstruction weights and gradually activating utility supervision (Details in Appendices B.6–B.7).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and evaluation settings. We evaluate on ten real-world benchmark datasets (Gorla et al., 2025; Wang et al., 2025); dataset statistics and preprocessing appear in Appendix A. Each dataset is split 80/20 into training and test sets, with 15% of the training portion reserved for validation. We evaluate the four missingness mechanisms defined above at test rates of 30%, 40%, and 50%. The 30% and 50% rates follow CACTI (Gorla et al., 2025), while 40% provides an intermediate setting. A single checkpoint per dataset is evaluated across all test conditions, with performance computed only on masked fields; the 50% setting evaluates generalization to an unseen missing rate.

Baselines and metrics. We compare SemImputer with seven recent baselines spanning the main imputation paradigms: predictive/reconstruction-based KPI and ReMasker, generative DiffPuter, RefiDiff, and CtrTab, and language-enhanced UnIMP and CACTI (Wang et al., 2026; Du et al., 2024; Zhang et al., 2025a; Ahamed et al., 2026; Li et al., 2026; Wang et al., 2025; Gorla et al., 2025). This selection covers both representative non-semantic imputers and the most directly related semantic baselines. All methods use common data splits and test masks while retaining their native estimation procedures (Appendix B.8). We evaluate numerical imputation using RMSE, MAE, R^2, and field-averaged Wasserstein distance (WD), and categorical imputation using field-level accuracy. All metrics are computed on masked entries and macro-averaged across datasets. Full metric definitions are provided in Appendix B.9.

Implementation details. All experiments are conducted on a single NVIDIA RTX 3090 GPU (24 GB). We train SemImputer with AdamW for up to 200 epochs using batch size 128, learning rate 3×10^{-4}, weight decay 2×10^{-3}, and gradient clipping at 1. Unless otherwise stated, we use seed 42. A 10-epoch warm-up is followed by cosine learning-rate decay. The curriculum assigns 30/20/50% of training to alignment, transition, and reconstruction, respectively. Checkpoints are selected by the minimum validation $\mathcal{L}_{\text{num}} + \mathcal{L}_{\text{cat}}$ after utility warm-up, with early-stopping patience 50 during reconstruction. The full training schedule is provided in Appendix B.7.

4.2 OVERALL PERFORMANCE

Comparison with existing methods. Tables 1 and 2 compare all methods across three missing rates and four mechanisms. SemImputer achieves the best dataset-averaged performance across all 48 numerical metric–setting combinations and all 12 categorical settings. Compared with the strongest baseline in each setting, SemImputer reduces RMSE, MAE, and WD by 5.4%, 3.8%, and 7.4% on average, respectively, while improving R^2 and categorical accuracy by 16.1% and 1.6%. Appendix C.1 reports dataset-level paired comparisons. Appendices C.2 and C.3 evaluate generalization to unseen missing rates of 60% and 70% and held-out mechanisms, respectively.

4.3 CAN SEMIMPUTER IDENTIFY WHEN SEMANTICS HELP?

We assess the incremental value of semantic injection using paired predictions for each masked test target: P_0 disables the semantic residual, whereas P_1 applies it at the learned type-specific cap. An error reduction gives $\Delta^{\text{rel}} > 0$, while $y^u = 1$ denotes the stricter margin-qualified benefit in Eq. equation 4. Figure 3 shows that semantic injection improves 38% of pooled target evaluations, with dataset-level benefit rates of 33%–51%, demonstrating that semantic utility varies across targets and observation contexts. To test whether Counterfactual Utility-Guided Fusion can identify such cases, we group targets into equal-count deciles by the predicted utility score ρ. The predicted scores closely track the observed frequency of $y^u = 1$, yielding an ECE of 0.004. This indicates that Counterfactual Utility-Guided Fusion provides a calibrated target-wise estimate of semantic benefit for controlling semantic injection.

Table 1: **Numerical imputation performance of eight methods is compared across missingness settings.** Each entry is an arithmetic mean over 10 datasets at the indicated nominal missing rate. Bold and underlined values indicate the best and second-best distinct results. Rel. gain is the percentage improvement over the strongest baseline in that column.

Missing rate	Method	MCAR				MAR				MNAR				MIXED			
		RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓
30%	KPI	1.007	0.587	0.110	0.316	0.809	0.461	0.365	0.260	1.000	0.611	0.098	0.332	0.936	0.527	0.214	0.284
	ReMasker	0.879	0.533	0.249	0.359	0.898	0.557	0.232	0.398	0.859	0.533	0.257	0.356	0.888	0.541	0.263	0.374
	DiffPuter	0.852	0.454	0.322	0.228	0.808	0.427	0.387	0.216	0.845	0.453	0.335	0.232	0.844	0.432	0.350	0.218
	UnIMP	0.798	0.426	0.358	0.259	0.767	0.416	0.407	0.256	0.766	0.424	0.389	0.258	0.794	0.417	0.383	0.254
	CACTI	0.809	0.454	0.344	0.300	0.779	0.440	0.395	0.294	0.789	0.456	0.358	0.304	0.807	0.443	0.371	0.292
	RefiDiff	0.894	0.480	0.192	0.233	0.865	0.471	0.213	0.232	0.866	0.479	0.202	0.236	0.883	0.466	0.234	0.229
	CtrTab	0.933	0.451	0.271	0.256	0.842	0.410	0.370	0.233	0.924	0.456	0.271	0.256	0.937	0.449	0.283	0.247
	Ours	**0.758**	**0.409**	**0.416**	**0.210**	**0.686**	**0.380**	**0.515**	**0.190**	**0.730**	**0.409**	**0.442**	**0.210**	**0.747**	**0.394**	**0.449**	**0.202**
	Rel. gain (%)	*5.01*	*3.99*	*16.20*	*7.89*	*10.56*	*7.32*	*26.54*	*12.04*	*4.70*	*3.54*	*13.62*	*9.48*	*5.92*	*5.52*	*17.23*	*7.34*
40%	KPI	1.022	0.629	0.056	0.348	0.845	0.462	0.325	0.236	1.058	0.656	0.033	0.366	0.989	0.574	0.123	0.302
	ReMasker	0.869	0.529	0.240	0.348	0.881	0.536	0.284	0.367	0.875	0.532	0.219	0.353	0.884	0.525	0.230	0.352
	DiffPuter	0.891	0.476	0.286	0.247	0.856	0.430	0.398	0.233	0.900	0.476	0.272	0.250	0.881	0.456	0.309	0.240
	UnIMP	0.799	0.441	0.348	0.259	0.749	0.410	0.441	0.256	0.802	0.440	0.333	0.269	0.801	0.424	0.350	0.266
	CACTI	0.817	0.473	0.323	0.314	0.775	0.422	0.408	0.284	0.819	0.471	0.308	0.315	0.817	0.453	0.328	0.304
	CtrTab	0.975	0.479	0.224	0.252	0.905	0.413	0.345	0.237	0.978	0.480	0.202	0.254	0.962	0.452	0.237	0.244
	RefiDiff	0.897	0.498	0.166	0.243	0.806	0.450	0.353	0.224	0.890	0.496	0.147	0.244	0.897	0.478	0.170	0.238
	Ours	**0.770**	**0.432**	**0.391**	**0.226**	**0.680**	**0.369**	**0.525**	**0.191**	**0.775**	**0.435**	**0.373**	**0.232**	**0.756**	**0.412**	**0.415**	**0.216**
	Rel. gain (%)	*3.63*	*2.04*	*12.36*	*7.00*	*9.21*	*10.00*	*19.05*	*14.73*	*3.37*	*1.14*	*12.01*	*4.92*	*5.62*	*2.83*	*18.57*	*9.24*
50%	KPI	1.063	0.658	0.024	0.373	0.855	0.530	0.288	0.280	1.085	0.677	0.016	0.380	0.979	0.558	0.164	0.268
	ReMasker	0.903	0.542	0.205	0.363	0.873	0.583	0.267	0.390	0.899	0.536	0.209	0.361	0.898	0.534	0.235	0.359
	DiffPuter	0.909	0.509	0.262	0.276	0.860	0.488	0.366	0.242	0.950	0.511	0.225	0.277	0.914	0.481	0.283	0.255
	UnIMP	0.843	0.463	0.303	0.276	0.764	0.469	0.402	0.288	0.849	0.462	0.292	0.275	0.834	0.446	0.331	0.264
	CACTI	0.864	0.503	0.269	0.339	0.781	0.482	0.387	0.326	0.866	0.500	0.265	0.337	0.849	0.478	0.310	0.326
	RefiDiff	0.941	0.529	0.120	0.266	0.824	0.507	0.303	0.236	0.937	0.524	0.125	0.265	0.910	0.498	0.192	0.252
	CtrTab	1.018	0.517	0.184	0.265	0.882	0.504	0.301	0.271	1.048	0.509	0.167	0.262	0.956	0.478	0.222	0.252
	Ours	**0.813**	**0.462**	**0.346**	**0.258**	**0.713**	**0.441**	**0.472**	**0.226**	**0.826**	**0.459**	**0.324**	**0.254**	**0.798**	**0.435**	**0.382**	**0.235**
	Rel. gain (%)	*3.56*	*0.22*	*14.19*	*2.64*	*6.68*	*5.97*	*17.41*	*4.24*	*2.71*	*0.65*	*10.96*	*3.05*	*4.32*	*2.47*	*15.41*	*6.75*

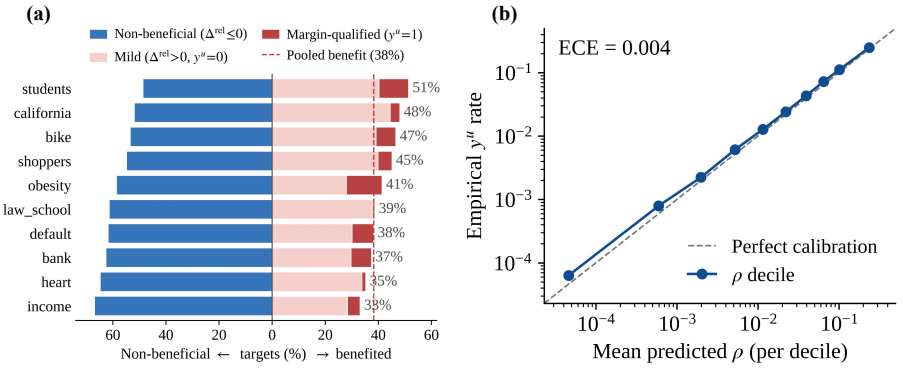


Figure 3: **SemImputer identifies when semantic injection is beneficial.** Across 10 datasets and three seeds, (a) semantic injection improves 38% of 9.2 million pooled target evaluations, with dataset-level benefit rates of 33%–51%; the dashed line marks the pooled rate. (b) Predicted utility scores ρ are calibrated to observed margin-qualified benefit ($y^u = 1$; ECE=0.004).

4.4 Ablation Studies

Contributions of the main components. Figure 4 evaluates the main components and semantic contribution. Removing Counterfactual Utility-Guided Fusion or Interleaved Missingness-View Training increases RMSE by 6.9% and 8.4%, respectively. Replacing semantic embeddings with random embeddings degrades performance, confirming that the gains rely on semantic content rather than additional representations. For Structure-Semantic Representation Alignment, row-level alignment contributes more than field-level alignment, while combining both yields the best performance.

What makes the utility gate informative? Table 3 examines the roles of counterfactual supervision and observation context. For the variant without counterfactual supervision, we remove the utility loss while retaining reconstruction gradients through the utility gate (Appendix D.1). AUROC decreases from 0.910 to 0.543 and RMSE increases from 0.7201 to 0.7296. Removing observation

Table 2: **Categorical imputation accuracy of eight methods is compared across missingness settings.** Values are percentages averaged over 10 datasets. Bold and underlined values mark the best and second-best results.

Missing rate	Method	Cat. Acc. (%)↑			
		MCAR	MAR	MNAR	MIXED
30%	KPI	64.181	73.796	61.388	69.902
	ReMasker	71.756	76.108	71.791	75.091
	DiffPuter	69.126	74.885	69.495	73.807
	UnIMP	<u>72.124</u>	<u>77.641</u>	<u>71.811</u>	<u>76.482</u>
	CACTI	70.160	76.387	69.967	73.737
	RefiDiff	66.363	72.182	66.130	71.148
	CtrTab	65.491	71.999	65.286	70.400
	Ours	**74.092**	**78.561**	**73.905**	**76.640**
	Rel. gain (%)	*2.729*	*1.185*	*2.916*	*0.207*
40%	KPI	62.135	64.049	59.477	61.498
	ReMasker	70.780	<u>68.449</u>	70.455	69.887
	DiffPuter	67.830	66.238	67.686	67.850
	UnIMP	<u>71.205</u>	67.883	<u>71.169</u>	<u>70.920</u>
	CACTI	67.952	67.138	67.945	68.067
	RefiDiff	65.237	63.044	64.951	64.685
	CtrTab	64.424	62.305	64.323	63.760
	Ours	**72.598**	**70.747**	**72.254**	**71.574**
	Rel. gain (%)	*1.956*	*3.357*	*1.525*	*0.922*
50%	KPI	61.248	68.138	59.461	62.171
	ReMasker	69.349	71.954	69.795	69.803
	DiffPuter	66.903	69.674	67.001	67.439
	UnIMP	<u>70.677</u>	<u>72.280</u>	<u>70.400</u>	<u>70.705</u>
	CACTI	66.077	70.121	66.183	66.969
	RefiDiff	64.051	66.260	64.083	64.779
	CtrTab	63.336	65.216	63.447	63.452
	Ours	**71.151**	**74.110**	**70.875**	**71.209**
	Rel. gain (%)	*0.671*	*2.532*	*0.675*	*0.713*

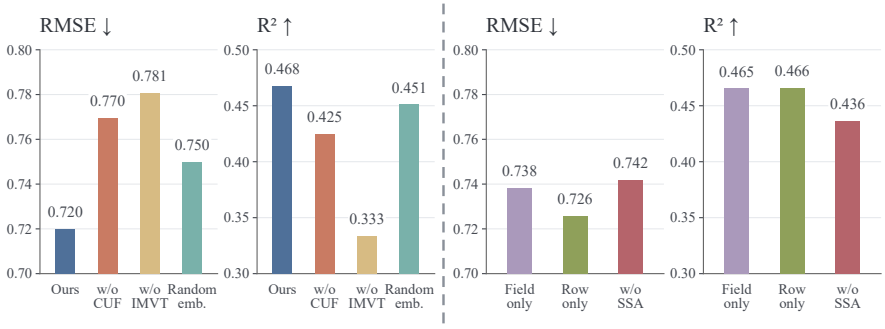


Figure 4: **Ablation of the main components and semantic information.** Results are averaged over Bike, Default, Students, and Heart across 12 test settings and three seeds. Left: CUF, IMVT, and random-embedding controls; right: field- and row-level SSA variants.

context has a smaller but consistent effect, reducing AUPRC from 0.383 to 0.337 and increasing RMSE to 0.7255.

4.5 Sensitivity and Efficiency

Sensitivity to utility supervision. Figure 5 varies the relative utility margin δ_{rel} and post-warm-up utility-loss weight w_u on the four ablation datasets, retraining each configuration with other settings fixed. RMSE varies by less than 0.6%, with similarly stable MAE. Increasing δ_{rel} from 0.02 to 0.10

Table 3: **Ablation of utility supervision and observation context.** Experiments are conducted on four datasets across three test rates, four missingness mechanisms, and three seeds.

Variant	RMSE ↓	AUROC ↑	AUPRC ↑
Full CUF	**0.7201**	**0.910**	**0.383**
w/o counterfactual supervision	0.7296	0.543	0.174
w/o observation context	0.7255	0.900	0.337

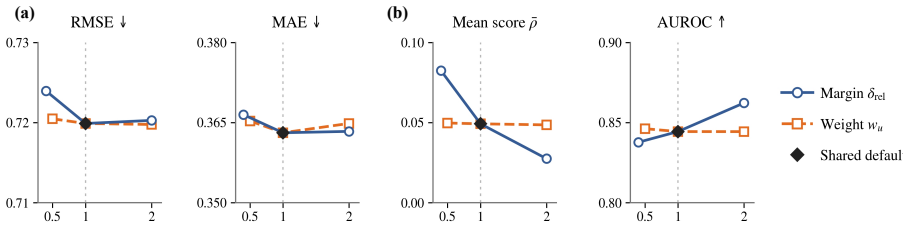


Figure 5: **Sensitivity to utility-supervision hyperparameters.** (a) report test performance, while (b) evaluate utility prediction.

lowers $\bar{\rho}$ from 0.082 to 0.027 while improving AUROC, whereas varying w_u has little effect on imputation or utility prediction (AUROC 0.844–0.846).

Inference efficiency. On Default (6,001 MCAR test rows), SemImputer has the second-lowest measured latency and FLOPs per row (0.02 G), with 1.21 million imputer parameters and peak GPU memory below 100 MB (Fig. 6). These measurements cover online imputation with cached semantics; the frozen language encoder and cache construction are excluded.

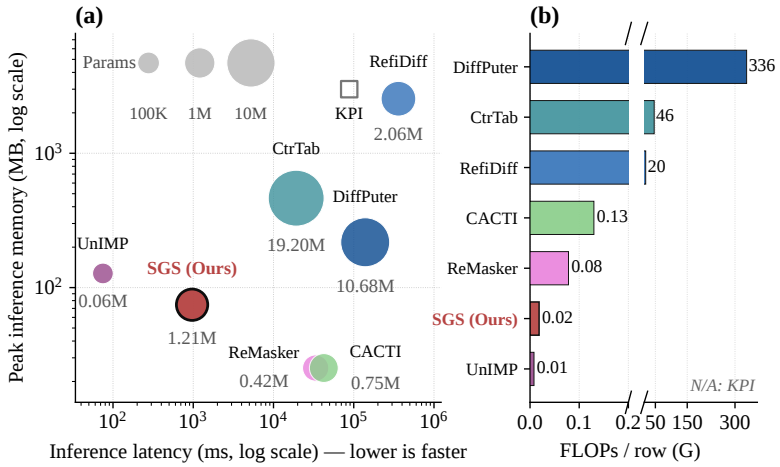


Figure 6: **Inference cost comparison.** (a) Latency versus peak GPU memory, with bubble size indicating parameter count. (b) FLOPs per imputed row.

5 Conclusion

In this work, we propose SemImputer around a simple premise: language-derived semantics are not uniformly beneficial for tabular imputation, but depend on the target and available evidence. Experiments on 10 real-world datasets consistently validate this premise and demonstrate strong performance across heterogeneous missingness conditions without test-time adaptation. More broadly, our results suggest that semantic priors should be treated as conditional evidence whose value must be assessed for each prediction, rather than information to be fused indiscriminately.

REFERENCES

Md Atik Ahamed, Qiang Ye, and Qiang Cheng. Refidiff: Progressive refinement diffusion for efficient missing data imputation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19551–19559, 2026.

Alan Arazi, Eilam Shapira, and Roi Reichart. Tabstar: A tabular foundation model for tabular data with text fields. *Advances in Neural Information Processing Systems*, 38:172108–172161, 2026.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Zhichao Chen, Haoxuan Li, Fangyikang Wang, Odin Zhang, Hu Xu, Xiaoyu Jiang, Zhihuan Song, and Hao Wang. Rethinking the diffusion models for missing data imputation: A gradient flow perspective. *Advances in Neural Information Processing Systems*, 37:112050–112103, 2024.

Tianyu Du, Luca Melis, and Ting Wang. Remasker: Imputing tabular data with masked autoencoding. In *International Conference on Learning Representations*, volume 2024, pp. 9002–9024, 2024.

Aditya Gorla, Ryan Wang, Zhengtong Liu, Ulzee An, and Sriram Sankararaman. Cacti: Leveraging copy masking and contextual information to improve tabular data imputation. In *International Conference on Machine Learning*, pp. 20187–20225. PMLR, 2025.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Daniel Jarrett, Bogdan C Cebere, Tennison Liu, Alicia Curth, and Mihaela van der Schaar. Hyperimpute: Generalized iterative imputation with automatic model selection. In *International Conference on Machine Learning*, pp. 9916–9937. PMLR, 2022.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository, 2023.

Falaah Arif Khan, Nazar Protsiv, Denys Herasymuk, and Julia Stoyanovich. Still more shades of Null: An evaluation suite for responsible missing value imputation. *Proceedings of the VLDB Endowment*, 18(9):2899–2913, 2025.

Jungkyu Kim, Kibok Lee, and Taeyoung Park. To predict or not to predict? proportionally masked autoencoders for tabular data imputation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17886–17894, 2025.

Sijie Li, Chen Chen, and Jungong Han. Simmlm: A simple framework for multi-modal learning with missing modality. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 24068–24077, 2025.

Zuqing Li, Junhao Gan, and Jianzhong Qi. Towards synthesizing high-dimensional tabular data with limited samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15207–15215, 2026.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2019.

Feng Luo, Hai Lan, Hui Luo, Zhifeng Bao, J Shane Culpepper, Shazia Sadiq, and Xiaoli Wang. Missing value imputation in tabular data lakes unleashed: A hybrid approach: F. luo et al. *The VLDB Journal*, 35(2):11, 2026.

DongAo Ma, Jiaxuan Pang, Michael B Gotway, and Jianming Liang. A fully open ai foundation model applied to chest radiography. *Nature*, 643(8071):488–498, 2025.

Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. Capturing semantics for imputation with pre-trained language models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 61–72. IEEE, 2021.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. Contexttab: A semantics-aware tabular in-context learner. *Advances in Neural Information Processing Systems*, 38:147005–147040, 2026.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Tao Tong, Xiaofeng Zhu, and Jiangzhang Gan. Meta-gain for missing data imputation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25975–25982, 2026.

Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011.

Hao Wang, Zhichao Chen, Xu Chen, Shuting He, Guangyi Liu, Haoxuan Li, Zhouchen Lin, et al. Iterative missing data imputation with model form adaptation and non-missing feature supervision. *Advances in Neural Information Processing Systems*, 38:20973–20997, 2026.

Jianwei Wang, Kai Wang, Ying Zhang, Wenjie Zhang, Xiwei Xu, and Xuemin Lin. On llm-enhanced mixed-type data imputation with high-order message passing. *arXiv preprint arXiv:2501.02191*, 2025.

Chenyu Yang, Yuyu Luo, Chuanxuan Cui, Ju Fan, Chengliang Chai, and Nan Tang. Data imputation with limited data redundancy using data lakes. *Proceedings of the VLDB Endowment*, 18(10): 3354–3367, 2025a.

Zequn Yang, Hongfa Wang, and Di Hu. Efficient quantification of multimodal interaction at sample level. *arXiv preprint arXiv:2506.17248*, 2025b.

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

Samira Zahmatkesh and Philipp Zech. Spatio-temporal missing data imputation: A systematic literature review with a focus on statistical and machine learning-based approaches. *ACM Computing Surveys*, 58(10):1–41, 2026.

Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. Jellyfish: Instruction-tuning local large language models for data preprocessing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8754–8782, 2024.

Hengrui Zhang, Liancheng Fang, Qitian Wu, and Philip Yu. Diffputer: Empowering diffusion models for missing data imputation. In *International Conference on Learning Representations*, volume 2025, pp. 63164–63185, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

A Datasets and Missingness Protocol

A.1 Datasets

Table 4 summarizes the ten benchmark datasets used in our experiments, including their sources, domains, field composition, encoded dimensionality, and exact train/validation/test splits. Most datasets originate from UCI (Kelly et al., 2023); the remaining ones are California Housing from scikit-learn, the Kaggle Cardiovascular Disease benchmark, and the LSAC National Longitudinal Bar Passage Study.

Numerical fields are standardized using training-set statistics, and categorical fields are one-hot encoded. Integer-coded nominal variables are treated as categorical, and the continuous California Housing target is discretized into five quantile bins following the benchmark protocol. Field descriptions and human-readable category labels are obtained from the source documentation before constructing semantic templates.

Dataset	Source	Domain	Rows	Num.	Cat.	Enc. dim.	Split (train / val / test)
Bank	UCI	Finance	40,004	6	8	44	27,202 / 4,801 / 8,001
Bike	UCI	Transportation	8,760	10	3	18	5,956 / 1,052 / 1,752
California	sklearn	Housing	20,640	8	1	13	14,035 / 2,477 / 4,128
Default	UCI	Finance	30,000	14	10	93	20,399 / 3,600 / 6,001
Heart	Kaggle CVD	Healthcare	70,000	5	7	21	47,613 / 8,403 / 13,984
Income	UCI	Demographics	47,621	6	9	110	32,380 / 5,715 / 9,526
Law School	LSAC	Education	20,798	6	6	25	14,141 / 2,496 / 4,161
Obesity	UCI	Healthcare	2,111	8	9	38	1,436 / 254 / 421
Shoppers	UCI	E-commerce	12,330	10	8	77	8,368 / 1,477 / 2,485
Students	UCI	Education	4,424	19	18	258	3,008 / 531 / 885

Table 4: **Characteristics of ten benchmark datasets are summarized.** Counts reflect full-table preprocessing. "Num." and "Cat." count original fields, including the conventional prediction target, and "Enc. dim." is the dimension after one-hot encoding.

A.2 Missingness Protocol

Complete rows are split 80/20 into training and test sets, with 15% of the training portion reserved for validation. Masks are generated for original fields and broadcast over categorical one-hot coordinates. MCAR masks cells independently. MAR retains a randomly selected half of the fields as observed covariates and uses a logistic model to mask the remainder. Its specified rate applies to these remaining fields, so the expected table-wide missing rate is approximately half as large. MNAR uses half of the fields to parameterize logistic missingness for the remainder while independently masking the conditioning fields. MIXED uses a fixed-seed random partition into three approximately equal field groups, each taking the corresponding columns from an MCAR, MAR, or MNAR component mask. Assignments are fixed across rows. Realized missing rates are recorded for each setting.

All methods share data splits and fixed test masks and are scored only on masked fields. SemImputer trains on 16 precomputed views combining the four mechanisms with specified rates of 10%, 20%, 30%, and 40%. Each row independently samples one view per epoch; validation assignments remain fixed. SemImputer receives no mechanism or rate identifiers and applies no additional online masking. One selected checkpoint is reused without parameter updates at test rates of 30%, 40%, and 50%; the last rate is unseen during training. Higher-rate and held-out-mechanism experiments are reported in Appendices C.2 and C.3.

B Implementation and Reproducibility

B.1 Semantic Input Construction Details

Metadata construction. For each field, we store a concise description and, when integer codes represent categories, a code-to-label dictionary. These entries are specified in a dataset-level meta-

data registry and may be overridden by the processed dataset metadata. If no description is available, we use a deterministic fallback formed from the cleaned field name and its numerical or categorical type. This procedure changes only the text presented to the language encoder; the tabular values and their numerical or one-hot representations remain unchanged.

Text templates. Let n_j, d_j, and t_j denote the name, description, and type of field j, respectively, and let r_{ij} denote its original value before numerical preprocessing or categorical encoding. An observed cell is serialized as `field` n_j`:` d_j`. value:` r_{ij}`. type:` t_j`.`; a masked target uses `field` n_j`:` d_j`. status: currently missing. type:` t_j`.` instead. The type is written as `numerical` or `categorical`. Numerical values are formatted with four significant digits, and a known categorical code is rendered as its label followed by the original code.

Example. For the *Default of Credit Card Clients* data, the field `PAY_0` is described as "repayment status in September 2005," and code 2 denotes a two-month payment delay. Its observed and masked templates are therefore:

```
field PAY_0: repayment status in September 2005. value:
payment delayed by two months (coded as 2). type:
categorical.
field PAY_0: repayment status in September 2005. status:
currently missing. type: categorical.
```

The masked template replaces the value clause with a missing-status marker while keeping the field name, description, and type fixed.

Frozen embeddings and trainable projection. Let $\mathcal{T}_{ij}^{\mathrm{obs}}$ and $\mathcal{T}_{j}^{\mathrm{tar}}$ denote the observed-cell and masked-target templates. The frozen Qwen3-Embedding-0.6B encoder E_{LM} (Zhang et al., 2025b) encodes each template independently:

$$\mathbf{q}_{ij}^{\mathrm{obs}} = E_{\mathrm{LM}}(\mathcal{T}_{ij}^{\mathrm{obs}}), \qquad \mathbf{q}_{j}^{\mathrm{tar}} = E_{\mathrm{LM}}(\mathcal{T}_{j}^{\mathrm{tar}}), \tag{8}$$

where both embeddings belong to $\mathbb{R}^{d_{\mathrm{ext}}}$ and $d_{\mathrm{ext}} = 1024$. They are computed offline and stored in caches of shape $N \times F \times 1024$ and $F \times 1024$, respectively, where N is the number of rows and F the number of fields. The target descriptor is shared across rows for each field. For the example above, every masked `PAY_0` cell selects the same descriptor embedding.

For a batch of B rows, the current mask selects the cached vector before the trainable projection:

$$\mathbf{q}_{ij} = (1 - m_{ij})\mathbf{q}_{ij}^{\mathrm{obs}} + m_{ij}\mathbf{q}_{j}^{\mathrm{tar}}, \qquad \mathbf{s}_{ij} = P_{\mathrm{cell}}(\mathbf{q}_{ij}), \tag{9}$$

where P_{cell} is a linear–GELU–dropout–LayerNorm map from $\mathbb{R}^{1024}$ to $\mathbb{R}^{128}$. Thus, the selected embeddings have shape $B \times F \times 1024$, and the semantic tokens have shape $B \times F \times 128$, with model width $d = 128$. The encoder remains frozen while P_{cell} is optimized jointly with the imputer.

B.2 SemImputer Architecture and Dimensions

Table 5 traces the tensors used by the implementation. The symbols N, B, F, and D denote the number of rows, batch size, number of original fields, and encoded tabular width. Categorical one-hot groups are projected as single field tokens, so all field-wise modules operate on sequences of length F.

B.3 Structure-Semantic Representation Alignment Details

The two alignment objectives in Eqs. equation 1 and equation 2 act on the same semantic and structural representations. This section specifies their encoders, projection heads, and numerical settings.

For a batch of B rows and F fields, the encoded table is $\mathbf{X} \in \mathbb{R}^{B \times D}$, where D includes all categorical one-hot coordinates. Its masked input is $\widetilde{\mathbf{X}} = \mathbf{X} \odot (1 - \overline{\mathbf{M}})$, where $\odot$ is element-wise multiplication and $\overline{\mathbf{M}}$ broadcasts the field mask over those coordinates. The semantic tokens $\mathbf{s}_{ij} \in \mathbb{R}^{128}$

Component	Input	Output	Main setting
Semantic cache	cell/target templates	$N \times F \times 1024$ and $F \times 1024$	frozen Qwen3-Embedding-0.6B
Semantic projection	selected cache entries	field $B \times F \times 128$	linear–GELU–norm
Structural encoder	$\widetilde{\mathbf{X}} \in \mathbb{R}^{B \times D}$, mask $B \times F$	$\mathbf{H} \in \mathbb{R}^{B \times F \times 128}$	2 layers, 4 heads
Structure-Semantic Representation Alignment	semantic and structural field tokens	field $B \times F \times 128$; row $B \times 128$	cosine + symmetric InfoNCE
Context-Aware Missing Pattern Encoder	field mask, observed field summaries	$\mathbf{c} \in \mathbb{R}^{B \times 128}$	2-layer, width-256 MLP
Counterfactual Utility-Guided Fusion	aligned views, $\mathbf{c}$, field state/type	$\rho, \gamma \in \mathbb{R}^{B \times F \times 1}$	MLP $384 \to 128 \to 64 \to 1$
Fusion and heads	structure, context, semantic direction and scale	numerical scalar or K_j categorical logits	field-wise width-128 heads

Table 5: **Interfaces and dimensions of SemImputer modules are summarized.** Qwen embeddings are cached offline; ρ is the target-wise utility score and γ the bounded semantic scale used by the imputation path.

are constructed by Eq. equation 9; template construction, caching, and projection details are given in Appendix B.1.

The structural encoder operates on one token per original field:

$$\begin{aligned} \mathbf{u}_{ij} &= E_j(\widetilde{\mathbf{v}}_{ij}) + \mathbf{p}_j + \mathbf{e}^t_{t_j} + \mathbf{e}^m_{m_{ij}}, \\ \mathbf{H}_i &= T_{\mathrm{tab}}([\mathbf{u}_{ij}]_{j=1}^F) = [\mathbf{h}_{ij}]_{j=1}^F, \end{aligned} \tag{10}$$

where $\widetilde{\mathbf{v}}_{ij}$ is a masked numerical scalar or categorical one-hot vector, and E_j is a field-specific affine projection to $\mathbb{R}^d$. The learned vectors $\mathbf{p}_j, \mathbf{e}^t_{t_j}, \mathbf{e}^m_{m_{ij}}$ encode field identity, type, and observation state. The input token $\mathbf{u}_{ij}$ enters the two-layer, four-head Transformer T_{tab}, followed by LayerNorm, to produce $\mathbf{h}_{ij} \in \mathbb{R}^d$. The matrix $\mathbf{H}_i$ collects the contextualized field tokens. Missing fields retain tokens and participate in attention, with their missingness indicated by the state embedding.

Separate field projections give

$$\mathbf{a}^s_{ij} = P_s(\mathbf{s}_{ij}), \qquad \mathbf{a}^h_{ij} = P_h(\mathbf{h}_{ij}), \tag{11}$$

where P_s, P_h are linear maps from $\mathbb{R}^d$ to $\mathbb{R}^d$, and $\mathbf{a}^s_{ij}, \mathbf{a}^h_{ij}$ are the aligned semantic and structural tokens. Equation equation 1 averages their cosine penalty over all BF cells, including observed and hidden fields. Each norm in the cosine calculation is lower-bounded by 10^{-12}.

The row projections act on the mean of these aligned tokens:

$$\mathbf{v}^s_i = Q_s\left(F^{-1}\sum_j \mathbf{a}^s_{ij}\right), \qquad \mathbf{v}^h_i = Q_h\left(F^{-1}\sum_j \mathbf{a}^h_{ij}\right), \tag{12}$$

where Q_s, Q_h are separate linear–GELU–LayerNorm maps, and $\mathbf{v}^s_i, \mathbf{v}^h_i \in \mathbb{R}^d$ are the projected row summaries. Both sides pool aligned field representations, including P_h on the structural side. Equation equation 2 uses temperature $\tau = 0.1$ and averages the two contrastive directions equally. For $B = 1$, the row loss is zero. Row projections provide training supervision and are omitted from deployed prediction.

B.4 Context-Aware Missing Pattern Encoder Details

The Context-Aware Missing Pattern Encoder constructs a shared row context from the field mask and observed-value summaries. Its two branches provide the utility condition and the fusion residual. The summary vector is

$$\boldsymbol{\eta}_i = [\pi_i, 1 - \pi_i, \mu^{\mathrm{obs}}_i, \sigma^{\mathrm{obs}}_i], \tag{13}$$

where $\mathbf{m}_i = [m_{i1}, \ldots, m_{iF}] \in \{0,1\}^F$ is the row mask, $\pi_i = F^{-1}\sum_j m_{ij}$ is its missing rate, and $\mu_i^{\text{obs}}, \sigma_i^{\text{obs}}$ are the mean and population standard deviation of the observed field scalars. Numerical fields retain their preprocessed values. For a categorical field with $K_j > 1$ categories, the scalar is $(\kappa_{ij} - 1)/(K_j - 1)$, where $\kappa_{ij} \in \{1, \ldots, K_j\}$ is its category index. A one-category field retains its encoded scalar. Fully missing rows use zero mean and standard deviation. The four-dimensional vector $\boldsymbol{\eta}_i$ and full mask are projected separately to width 256 and added, processed by a two-layer residual MLP, and projected to $\mathbf{c}_i \in \mathbb{R}^d$, with $d = 128$.

For the utility condition in Eq. equation 3, a linear projection P_g maps the shared context to $\mathbb{R}^d$. Its output is concatenated with the target state m_{ij} and the two-coordinate type encoding $\text{onehot}(t_j)$. A linear–GELU map $P_c : \mathbb{R}^{d+3} \to \mathbb{R}^d$ produces $\mathbf{c}_{ij}^{\text{tar}}$. The batch tensors for the shared and conditioned contexts have shapes $B \times d$ and $B \times F \times d$. This condition depends on row context, target type, and observation state. Counterfactual Utility-Guided Fusion combines it with the aligned field tokens $\mathbf{a}_{ij}^s, \mathbf{a}_{ij}^h$ to estimate field-specific utility.

The residual is

$$g_{ij}^m = g_{\max}\sigma(G_m([\mathbf{h}_{ij}; \mathbf{c}_i; m_{ij}])), \qquad \mathbf{r}_{ij}^m = g_{ij}^m R_m(\mathbf{c}_i), \tag{14}$$

where G_m reads the structural token $\mathbf{h}_{ij}$, shared context $\mathbf{c}_i$, and observation state m_{ij} to produce a scalar logit. The sigmoid σ and cap $g_{\max} = 0.5$ bound the gate g_{ij}^m. Both G_m and R_m use linear–GELU–dropout–linear maps, with widths $(2d+1) \to d \to 1$ and $d \to d \to d$, respectively. Dropout is 0.1, and the final bias of G_m is initialized to -2. The gate scales the projected context $R_m(\mathbf{c}_i)$ to produce $\mathbf{r}_{ij}^m \in \mathbb{R}^d$, which enters the fusion path.

B.5 Counterfactual Utility-Guided Fusion Details

Endpoint implementation. The matched comparison builds on sample-wise interaction analysis and fusion under missing inputs (Yang et al., 2025b; Li et al., 2025). Fusion and prediction heads operate field-wise, so switching the scales of all masked fields together yields a matched comparison for each target. Each field type has a learned scalar θ_t, giving $\lambda_t = \sigma(\theta_t)$. Initial caps are 0.45 for categorical fields and 0.15 for numerical fields.

Endpoint errors. For endpoint $b \in \{0,1\}$, let $\hat{x}_{ij}^{(b)}$ denote a numerical prediction and $\widehat{\mathbf{p}}_{ij}^{(b)}$ a categorical probability vector. The error functions used in Eq. equation 4 are

$$e_{ij}^{(b)} = \begin{cases} \text{Huber}_{\delta_H}(\hat{x}_{ij}^{(b)}, x_{ij}), & t_j = \text{num}, \\ -\log \widehat{p}_{ij,\kappa_{ij}}^{(b)}, & t_j = \text{cat}, \end{cases} \tag{15}$$

where κ_{ij} indexes the category label x_{ij}. During utility training, endpoint errors and labels are recomputed from the current model for each batch.

Loss reduction and initialization. The utility loss averages within each field type before balancing the types:

$$\mathcal{L}_{\text{util}} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \frac{1}{|\Omega_t|} \sum_{(i,j)\in\Omega_t} \text{BCE}(\ell_{ij}^u, y_{ij}^u), \tag{16}$$

where BCE is binary cross-entropy with logits, $\Omega_t = \{(i,j) \in \Omega : t_j = t\}$ contains supervised targets of type t, and $\mathcal{T} = \{t : |\Omega_t| > 0\}$ contains the supervised types. Positive and negative utility labels receive no additional reweighting. The final layer of G_ϕ is initialized to zero, giving $\rho_{ij} = 0.5$ before utility supervision starts.

Residual-bound derivation. The transformation P_f uses a linear–GELU–dropout–linear map of width $d \to d \to d$, with dropout 0.1. Let v_{ij} be the population variance across the d coordinates of $P_f(\mathbf{a}_{ij}^s)$ and $\epsilon_{\text{LN}} > 0$ the normalization stabilizer. For $\bar{\mathbf{s}}_{ij} = \text{LN}_0(P_f(\mathbf{a}_{ij}^s))$, we obtain

$$\begin{aligned} \frac{\|\gamma_{ij}\bar{\mathbf{s}}_{ij}\|_2}{\sqrt{d}} &= \gamma_{ij}\sqrt{\frac{v_{ij}}{v_{ij}+\epsilon_{\text{LN}}}} \\ &\le \gamma_{ij} \le m_{ij}\lambda_{t_j}, \end{aligned} \tag{17}$$

where γ_{ij} is defined in Eq. equation 6.

B.6 Reconstruction and Moment-Matching Losses

The reconstruction losses in Eq. equation 7 are:

$$\mathcal{L}_{\text{num}} = \frac{1}{|\Omega_{\text{num}}|} \sum_{(i,j)\in\Omega_{\text{num}}} (\hat{x}_{ij} - x_{ij})^2,$$
$$\mathcal{L}_{\text{cat}} = \frac{1}{|\mathcal{J}_{\text{cat}}|} \sum_{j\in\mathcal{J}_{\text{cat}}} \frac{1}{|\Omega_j|} \sum_{i\in\Omega_j} -\log \widehat{p}_{ij,\kappa_{ij}}, \tag{18}$$

where $\Omega_{\text{num}} = \{(i,j) \in \Omega : t_j = \text{num}\}$ is the set of masked numerical targets, $\Omega_j = \{i : (i,j) \in \Omega\}$ contains supervised rows for field j, and $\mathcal{J}_{\text{cat}} = \{j : t_j = \text{cat}, |\Omega_j| > 0\}$ contains categorical fields with supervision. The values $\hat{x}_{ij}, x_{ij}$ are numerical predictions and targets, while $\widehat{p}_{ij,\kappa_{ij}}$ is the predicted probability of the true category.

The moment-matching term is:

$$\mathcal{L}_{\text{mmm}} = \frac{1}{|\mathcal{J}_2|} \sum_{j\in\mathcal{J}_2} [(\widehat{\mu}_j - \mu_j)^2 + (\widehat{\sigma}_j - \sigma_j)^2], \tag{19}$$

where $\mathcal{J}_2 = \{j : t_j = \text{num}, |\Omega_j| \geq 2\}$ contains numerical fields with at least two masked targets in the batch. Within the batch, $\widehat{\mu}_j, \widehat{\sigma}_j$ are computed from predictions at Ω_j, and μ_j, σ_j from the corresponding targets. Standard deviations use population variance with 10^{-8} added before the square root. Empty supervision sets contribute zero, including in the utility loss. Structure-Semantic Representation Alignment losses cover both observed and masked fields.

B.7 Interleaved Missingness-View Training and Optimization

Inspired by epoch-wise dataset alternation in Ark$^+$ (Ma et al., 2025), Interleaved Missingness-View Training independently assigns each training row one of the 16 precomputed mechanism–severity views per epoch. Let $\mathcal{V}$ index these views and $v_i^{(e)} \sim \text{Unif}(\mathcal{V})$ denote the view drawn for row i at epoch e. The next epoch resamples this assignment, while validation retains a fixed per-row assignment. The same selected mask controls input masking and supervision, with no additional online masking. No mechanism or severity identifiers enter the model.

A curriculum (Bengio et al., 2009) trains alignment and reconstruction jointly. The default 200-epoch schedule allocates 30%, 20%, and 50% to alignment, transition, and reconstruction. During transition, $(w_n, w_c, w_m, w_f, w_r)$ changes from $(0.5, 0.5, 0, 0.02, 0.02)$ to $(1, 1, 0.3, 0.005, 0)$ by cosine interpolation, where the weights correspond to the terms in Eq. equation 7. The utility weight w_u is zero during the alignment phase and increases linearly to 0.2 over the first 10 transition epochs. Endpoints are evaluated only when $w_u > 0$; the gated main path is active throughout training.

We use AdamW with learning rate 3×10^{-4}, weight decay 2×10^{-3}, batch size 128, and gradient clipping at 1. A 10-epoch learning-rate warm-up from 10% of the target rate is followed by cosine decay to 10^{-5}. This warm-up lies inside the alignment phase. Validation runs after every epoch and monitors $q_e = \mathcal{L}_{\text{num}}^{\text{val}} + \mathcal{L}_{\text{cat}}^{\text{val}}$, where q_e is the unweighted validation reconstruction score at epoch e. Under the default indexed schedule, utility supervision starts at epoch 60, checkpoint eligibility at epoch 69 after utility warm-up, and patience accumulation at epoch 100. Early stopping uses patience 50. The selected checkpoint is reused without updates across test views. Algorithm 1 summarizes the procedure.

The parameter E is the maximum number of epochs, e_u is the first epoch eligible for checkpoint selection after utility warm-up, and e_r is the first epoch eligible for patience accumulation after both the transition and utility warm-up are complete. The parameter p is the early-stopping patience. The symbols $\mathcal{W}$ and $\mathcal{W}^*$ denote the current and selected model parameters, respectively. The score q_e is the validation reconstruction loss at epoch e, q^* is its best eligible value, and n_{wait} counts eligible evaluations without improvement.

B.8 Baseline Reproduction Protocol

We follow the official implementations, retaining each method's core estimation components while adapting its data interface to the benchmark. All methods use the same processed arrays, field

Algorithm 1 Training procedure for Interleaved Missingness-View Training.

Require: Training data $\mathcal{D}_{\text{tr}}$; mask bank $\mathcal{M} = \{\mathbf{m}_i^v\}$; fixed validation data $\mathcal{D}_{\text{val}}$; schedule $\{w_\bullet(e)\}$; $1 \le e_u \le e_r \le E$; patience p
Ensure: Selected checkpoint $\mathcal{W}^*$

```
Initialize W, q* ← ∞, n_wait ← 0
for e = 1, ..., E do
    Set {w•(e)} and draw v_i^(e) ~ Unif(V) for each training row
    for each batch B do
        Construct masked inputs using the selected row views and compute the shared encoding and gated prediction
        if w_u(e) > 0 then
            Compute stop-gradient endpoints P_0, P_1 and utility labels using Eq. equation 4
        end if
        Update W using Eq. equation 7
    end for
    q_e ← L_num^val + L_cat^val
    if e ≥ e_u and q_e < q* then
        (q*, W*, n_wait) ← (q_e, W, 0)
    else if e ≥ e_r then
        n_wait ← n_wait + 1
    end if
    break if n_wait ≥ p
end for
return W*
```

Method	Training input	Reuse across settings	Test fitting	Main settings
KPI (Wang et al., 2026)	complete train + current test table	per-setting fit	yes	k=10, 50 epochs, lr 5×10^{-4}, 2 pairs
ReMasker (Du et al., 2024)	complete train + current test table	per-setting fit	yes	600 epochs, width 32, depth 8/4, 4 heads, mask 0.5
DiffPuter (Zhang et al., 2025a)	complete training rows	1 checkpoint/dataset	no	10001 epochs, width 1024, 50 steps, 10 trials
UnIMP (Wang et al., 2025)	16-view pool	1 checkpoint/dataset	no	3 GNN layers, GNN width 64, 1000 epochs, Huber loss
CACTI (Gorla et al., 2025)	16-view pool	1 checkpoint/dataset	no	width 64, depth 10/4, 1000 epochs, mask ratio 0.9
RefiDiff (Ahamed et al., 2026)	16-view pool	1 checkpoint/dataset	XGBoost refiner	width 128, 10001 epochs, 50 steps, 5 trials
CtrTab (Li et al., 2026)	complete rows + 16-view control pool	1 checkpoint/dataset	no	VAE 600 epochs; diffusion 2000 and control 10000 steps

Table 6: **Reproduction settings of seven baselines are summarized.** "Test fitting" denotes optimization or auxiliary-model fitting on the observed portion of each incomplete test table.

ordering, data splits, test masks, and metric implementation. The common 16-view pool combines four missingness mechanisms with specified rates of 10%, 20%, 30%, and 40%; it supplies training views to mask-aware inductive methods. Complete-data density models instead train on complete rows, while transductive `fit_transform` methods are fitted separately to each incomplete test table. Table 6 lists the training budgets and principal settings; the method-specific procedures and benchmark adaptations are detailed below. Hidden test values are used only for scoring, never for fitting or model selection.

KPI. We retain the official iterative kernel-point estimator and its feature-wise model adaptation (Wang et al., 2026). Distance-weighted k-nearest-neighbor imputation with $k = 10$ initializes the missing entries. For each test setting, the estimator is fitted to the concatenation of complete training rows and the current incomplete test table, using 50 epochs and learning rate 5×10^{-4}. Optimization and iterate selection use observed entries and an internally sampled validation mask. Only the completed test rows enter evaluation; fitting is repeated for each setting.

ReMasker. We retain the masked-autoencoder encoder–decoder and its re-masking reconstruction objective (Du et al., 2024). The official `fit_transform` interface fits a new model to the concatenated complete training rows and incomplete test rows for each setting. Additional reconstruction masks are generated within this procedure, using mask ratio 0.5. The model uses width 32, encoder/decoder depths 8/4, four attention heads, and 600 training epochs. Imputed training rows are discarded before scoring, and a new fit is performed for the next test mask.

DiffPuter. We retain the EDM density model and conditional diffusion sampling procedure (Zhang et al., 2025a). The density model is trained once on complete preprocessed training rows, using width 1024 and 10001 epochs, and its checkpoint is reused across test settings. Observed test entries condition the sampler without updating model parameters. Each completion averages ten stochastic trials, with 50 denoising steps and 20 resampling steps per trial. Observed entries are restored before masked-entry scoring.

UnIMP. We retain UnIMP's language-model representations, cell-oriented hypergraph, bidirectional node–hyperedge message passing, XFusion module, and projection head (Wang et al., 2025). XFusion combines language and hypergraph features before the head predicts missing values. The graph component uses three layers and width 64. For this benchmark, training uses Huber loss for 1000 epochs over the common 16-view pool, with one view sampled per epoch. The resulting checkpoint is reused for all test mechanisms and rates without parameter updates.

CACTI. We retain CACTI's encoder–decoder, column-context inputs, and median-truncated copy-masking procedure (Gorla et al., 2025). The common 16-view pool supplies the training missingness patterns, with one view assigned to each row per epoch; copy masking and median truncation remain part of CACTI's reconstruction procedure. We use width 64, encoder/decoder depths 10/4, masking ratio 0.9, and 1000 training epochs. For semantic capacity matching, column context is encoded offline with the same frozen Qwen3-Embedding-0.6B backbone as SemImputer. This changes the source of column embeddings while retaining their role in CACTI. One checkpoint per dataset is reused across all test settings.

RefiDiff. We retain the predictive refinement stages and Mamba-based diffusion denoiser (Ahamed et al., 2026). Refined versions of the 16 training views supply the denoiser's inputs, with one view selected per row and epoch. The denoiser uses width 128 and 10001 training epochs; inference uses 50 denoising steps and five trials. At evaluation, the official refinement procedure fits column-wise XGBoost regressors using observed test entries, while the diffusion checkpoint remains fixed. Thus, the refinement models are fitted for each incomplete test table even though the trained diffusion model is shared across settings.

CtrTab. We retain the Transformer VAE, latent diffusion model, and ControlNet conditioning architecture (Li et al., 2026). The VAE and base diffusion model are fitted to complete training rows for 600 epochs and 2000 steps, respectively. To adapt this synthesis model to imputation, the control stage is trained for 10000 steps on partially observed views from the common mask pool. The VAE, diffusion model, and ControlNet are saved together and reused across test mechanisms and rates. Test observations provide conditioning information without further parameter fitting.

B.9 Evaluation Metrics and Utility Diagnostics

Imputation metrics. RMSE and MAE pool the masked numerical cells within each dataset and evaluation setting. The reported R^2 is squared Pearson correlation between predictions and targets over those same cells. For WD, we compute the one-dimensional Wasserstein distance between the

predicted and true values at masked positions separately for each numerical field. The main comparison and component ablation tables report the mean over fields with masked targets, corresponding to the evaluator's `wd_mean` output. The evaluator also exports the sum as `wd`; these two reductions have different scales. Categorical accuracy is the number of correct masked-field predictions divided by the number of masked categorical fields, counting each original field once rather than scoring its one-hot coordinates separately. Dataset-macro averages give each included dataset equal weight.

Positive utility labels and ranking. A target has any semantic benefit when $e^{(1)} < e^{(0)}$, equivalently $\Delta^{\mathrm{rel}} > 0$. The gate is supervised by the stricter event $y^u = 1$ in Eq. equation 4, requiring both the relative and type-specific absolute margins. Figure 3(a) reports any semantic benefit, while panel (b) calibrates scores against $y^u = 1$. AUROC measures how well ρ ranks positive utility labels above negative ones. The labels are defined by each checkpoint's paired predictions, so their prevalence can change across trained variants as well as utility margins. AUPRC is reported as average precision: the mean precision at the ranks occupied by positive targets. These utility metrics assess the gate, whereas RMSE and MAE assess the final imputed values.

Validation diagnostics for ablation and sensitivity. The utility diagnostics in Table 3 and Figure 5 are taken from the validation evaluation at the selected checkpoint's epoch. Test reconstruction metrics are computed separately across the 12 mechanism–rate settings. Both studies use Bike, Default, Students, and Heart, with datasets weighted equally. For the variant without counterfactual supervision, AUROC and AUPRC are computed post hoc from its validation gate scores and paired utility labels; its reconstruction-gradient path is specified in Appendix D.1. The positive fraction, mean score, and calibration error are

$$\begin{aligned} \pi_u &= \frac{1}{n}\sum_{k=1}^{n} y_k^u, \qquad \bar{\rho} = \frac{1}{n}\sum_{k=1}^{n} \rho_k, \\ \mathrm{ECE} &= \sum_{b=1}^{10} \frac{n_b}{n} \left|\bar{\rho}_b - \bar{y}_b\right|, \end{aligned} \tag{20}$$

where n is the number of masked validation targets, k indexes those targets, and y_k^u, ρ_k are their utility labels and scores. Index b denotes one of ten equal-width score bins on $[0, 1]$, n_b is its target count, and $\bar{\rho}_b, \bar{y}_b$ are its mean score and positive-label fraction. Empty bins contribute zero. The values π_u, $\bar{\rho}$, and ECE are proportions on a 0–1 scale; for example, $\pi_u = 0.054$ means that 5.4% of targets satisfy both margins. The equal-width bins define ECE for the validation diagnostics. In Figure 3(b), the displayed held-out test ECE uses the same weighted formula over ten equal-count score groups. In the sensitivity study, w_u denotes the final utility-loss weight reached after warm-up, while δ_{rel} changes which targets qualify as positive. AUROC values across different margins therefore refer to different positive-label definitions.

C Additional Results and Robustness

C.1 Dataset-Level Paired Comparisons

Figure 7 averages the 12 settings within each dataset before comparing methods, so each dataset receives equal weight. All 28 median paired effects favor SemImputer. Holm-adjusted improvements are significant against 7/7 baselines for RMSE and R^2, 4/7 for MAE, and 3/7 for WD. The positive medians but wider dispersion in several MAE and WD comparisons indicate more dataset-dependent gains for these metrics.

C.2 Extreme Missingness at 60% and 70%

At 60% missingness, SemImputer leads 13 of the 16 metric–mechanism comparisons and has the highest R^2 under all four mechanisms (Table 7). Its R^2 gains over the strongest baseline reach 22.36% under MNAR and 13.78% under MIXED. UnIMP has lower MAE under MAR and MIXED, and CtrTab has lower WD under MCAR.

At 70% missingness, SemImputer leads or ties in 14 of 16 comparisons, including every RMSE and R^2 comparison (Table 8). Its R^2 gains reach 10.66% under MAR and 12.87% under MNAR.

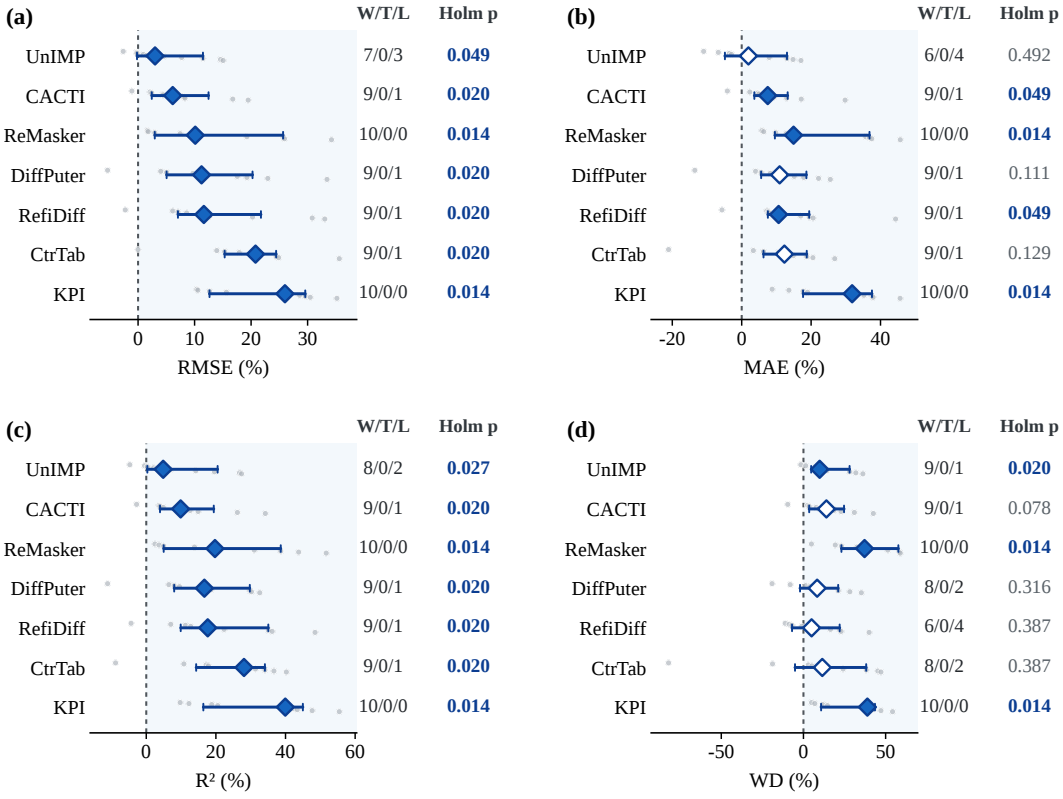


Figure 7: **Dataset-level gains of SemImputer are compared across baselines.** Panels report RMSE, MAE, R^2, and WD. Each gray point averages 12 settings within one dataset ($n = 10$); positive values favor SemImputer. Diamonds and bars show the median and 95% dataset-bootstrap confidence interval. W/T/L denotes dataset wins/ties/losses; filled diamonds mark Holm-adjusted two-sided Wilcoxon $p < 0.05$.

	MCAR				MAR				MNAR				MIXED			
Method	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓
KPI	1.012	0.638	0.038	0.407	0.865	0.479	0.292	0.312	1.036	0.647	0.031	0.412	1.008	0.585	0.119	0.345
ReMasker	0.920	0.569	0.155	0.418	0.958	0.559	0.229	0.362	0.930	0.567	0.155	0.417	0.908	0.536	0.198	0.375
DiffPuter	0.980	0.544	0.186	0.303	0.910	0.473	0.303	0.244	0.958	0.539	0.197	0.298	0.919	0.510	0.250	0.277
UnIMP	0.846	0.505	0.271	0.295	0.785	**0.406**	0.395	0.258	0.858	0.509	0.237	0.296	0.832	**0.457**	0.312	0.254
CACTI	0.880	0.537	0.211	0.395	0.822	0.456	0.354	0.334	0.893	0.538	0.206	0.394	0.865	0.503	0.262	0.353
RefiDiff	0.967	0.566	0.150	0.308	0.898	0.474	0.289	0.259	0.985	0.566	0.144	0.310	0.949	0.523	0.195	0.268
CtrTab	1.052	0.564	0.111	**0.281**	0.931	0.443	0.261	0.252	1.069	0.561	0.109	0.276	1.014	0.512	0.166	0.272
SemImputer	**0.843**	**0.492**	**0.283**	0.285	**0.763**	0.418	**0.414**	**0.217**	**0.843**	**0.487**	**0.290**	**0.273**	**0.795**	0.459	**0.355**	**0.249**
improv. (%)	*0.35*	*2.57*	*4.43*	*-1.42*	*2.80*	*-2.96*	*4.81*	*11.07*	*1.75*	*4.32*	*22.36*	*1.09*	*4.45*	*-0.44*	*13.78*	*1.97*

Table 7: **Numerical imputation performance of eight methods is compared at 60% missingness.** The specified masking rate follows Appendix A.2; values are macro-averaged over datasets. Bold and underlined values indicate the best and second-best results, respectively. The final row reports the relative improvement of SemImputer over the strongest baseline for each metric; negative values indicate that the strongest baseline performs better.

UnIMP retains a marginal MAE advantage under MAR, and CtrTab has lower WD under MNAR. Both severities lie beyond the 10%–40% view pool used during training.

	MCAR				MAR				MNAR				MIXED			
Method	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓
KPI	1.029	0.660	0.016	0.435	0.880	0.492	0.269	0.334	1.055	0.669	0.011	0.440	1.029	0.603	0.095	0.370
ReMasker	0.941	0.587	0.130	0.444	0.979	0.575	0.205	0.386	0.951	0.585	0.130	0.443	0.927	0.553	0.172	0.401
DiffPuter	1.003	0.584	0.145	0.342	0.879	0.465	0.330	0.268	1.019	0.583	0.139	0.348	0.984	0.544	0.198	0.312
UnIMP	0.875	0.534	0.236	0.327	0.809	**0.421**	0.366	0.252	**0.887**	0.537	0.202	0.327	0.860	0.483	0.275	0.286
CACTI	0.905	0.559	0.182	0.424	0.843	0.473	0.326	0.358	0.918	0.559	0.177	0.423	0.889	0.524	0.231	0.383
RefiDiff	0.993	0.587	0.124	0.339	0.922	0.491	0.263	0.279	1.012	0.586	0.119	0.341	0.975	0.541	0.166	0.298
CtrTab	1.077	0.587	0.086	0.319	0.952	0.458	0.235	0.274	1.094	0.583	0.085	**0.312**	1.038	0.532	0.140	0.298
SemImputer	**0.874**	**0.525**	**0.245**	**0.295**	**0.785**	0.423	**0.405**	**0.236**	**0.887**	**0.522**	**0.228**	0.313	**0.855**	**0.478**	**0.282**	**0.279**
improv. (%)	*0.11*	*1.69*	*3.81*	*7.52*	*2.97*	*-0.48*	*10.66*	*6.35*	*0.00*	*2.79*	*12.87*	*-0.32*	*0.58*	*1.04*	*2.55*	*2.45*

Table 8: **Numerical imputation performance of eight methods is compared at 70% missingness.** The specified masking rate follows Appendix A.2; values are macro-averaged over datasets. Bold and underlined values indicate the best and second-best results, respectively. The final row reports the relative improvement of SemImputer over the strongest baseline for each metric; negative values indicate that the strongest baseline performs better.

C.3 Generalization to Held-Out Missingness Mechanisms

Tables 9 and 10 compare SemImputer and UnIMP after training each method on the same two-mechanism subset of the 10%–40% view pool. We hold out MNAR when training on MCAR+MAR and hold out MAR when training on MCAR+MNAR. Tests at 30% and 40% isolate transfer to an unseen mechanism at trained severities; 50% additionally tests an unseen severity. Both methods are evaluated without parameter updates.

	30%				40%				50%			
Method	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓
UnIMP	0.772	0.428	0.380	2.279	0.810	0.443	0.324	2.307	0.859	0.470	0.279	2.395
SemImputer	**0.732**	**0.408**	**0.439**	**1.828**	**0.779**	**0.430**	**0.367**	**1.991**	**0.829**	**0.460**	**0.319**	**2.195**
SemImputer gain (%)	*5.24*	*4.75*	*15.50*	*19.81*	*3.83*	*2.99*	*13.30*	*13.71*	*3.51*	*2.03*	*14.26*	*8.32*

Table 9: **Generalization of SemImputer and UnIMP to held-out MNAR is compared.** Values are macro-averaged over ten datasets. Both methods train on MCAR and MAR and use the same test masks. The final row reports the relative gain of SemImputer over UnIMP.

	30%				40%				50%			
Method	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓	RMSE↓	MAE↓	R^2 ↑	WD↓
UnIMP	0.769	0.421	0.400	1.259	0.750	0.411	0.438	1.219	0.773	0.477	0.394	1.210
SemImputer	**0.684**	**0.370**	**0.517**	**0.788**	**0.695**	**0.371**	**0.513**	**0.851**	**0.715**	**0.436**	**0.473**	**0.820**
SemImputer gain (%)	*11.02*	*11.92*	*29.28*	*37.40*	*7.40*	*9.65*	*17.28*	*30.23*	*7.53*	*8.56*	*20.18*	*32.26*

Table 10: **Generalization of SemImputer and UnIMP to held-out MAR is compared.** Values are macro-averaged over ten datasets. Both methods train on MCAR and MNAR and use the same test masks. The final row reports the relative gain of SemImputer over UnIMP.

SemImputer outperforms UnIMP on every reported metric and severity in both transfer directions. For held-out MNAR, its gains range from 3.51% to 5.24% in RMSE and from 8.32% to 19.81% in WD. For held-out MAR, the corresponding ranges are 7.40%–11.02% and 30.23%–37.40%. The advantage remains at 50%, with smaller RMSE and WD gaps under MNAR than under MAR.

C.4 Dataset-Wise Results across Missingness Conditions

Figures 8–15 expand the aggregate comparison in Table 1 by dataset and missingness mechanism. Each bar averages 30%–50%, while its error bar is the standard deviation across these three severities. The error bars therefore measure sensitivity to missingness severity, not uncertainty across random seeds. Lower values are better for RMSE, MAE, and WD, whereas higher values are better for R^2.

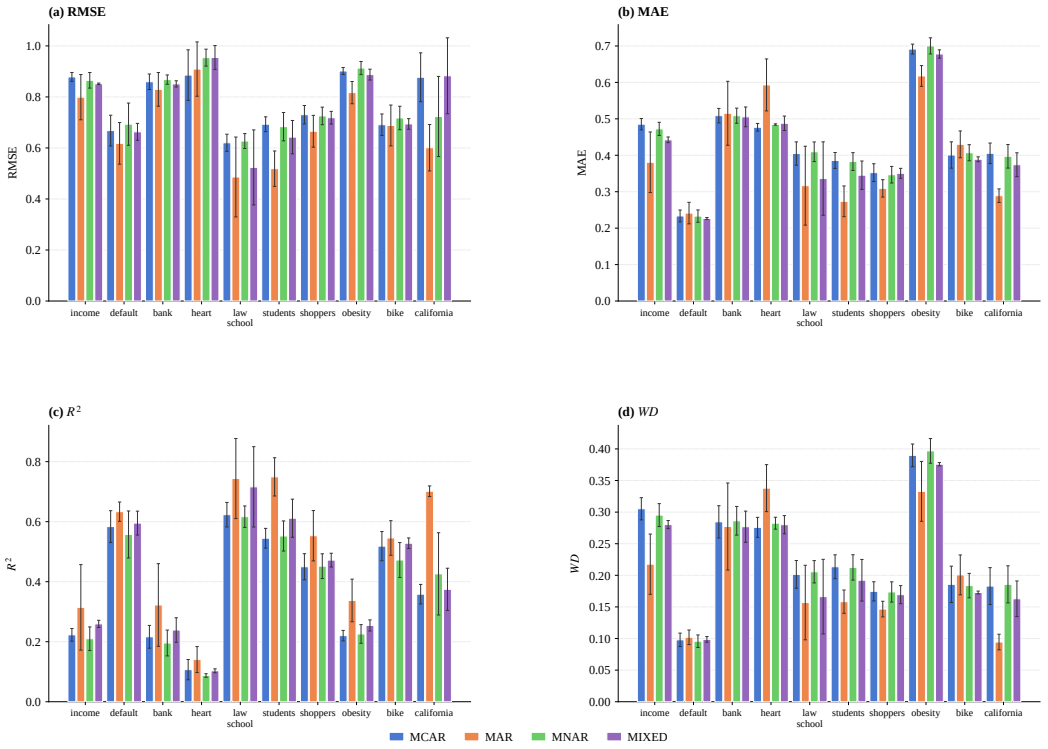


Figure 8: **Numerical imputation performance of SemImputer is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, on each dataset under MCAR, MAR, MNAR, and MIXED missingness. Bars show the mean over 30%–50% missingness, and error bars show the corresponding standard deviation across severities. One dataset-specific SemImputer checkpoint is used unchanged for all mechanisms and severities.

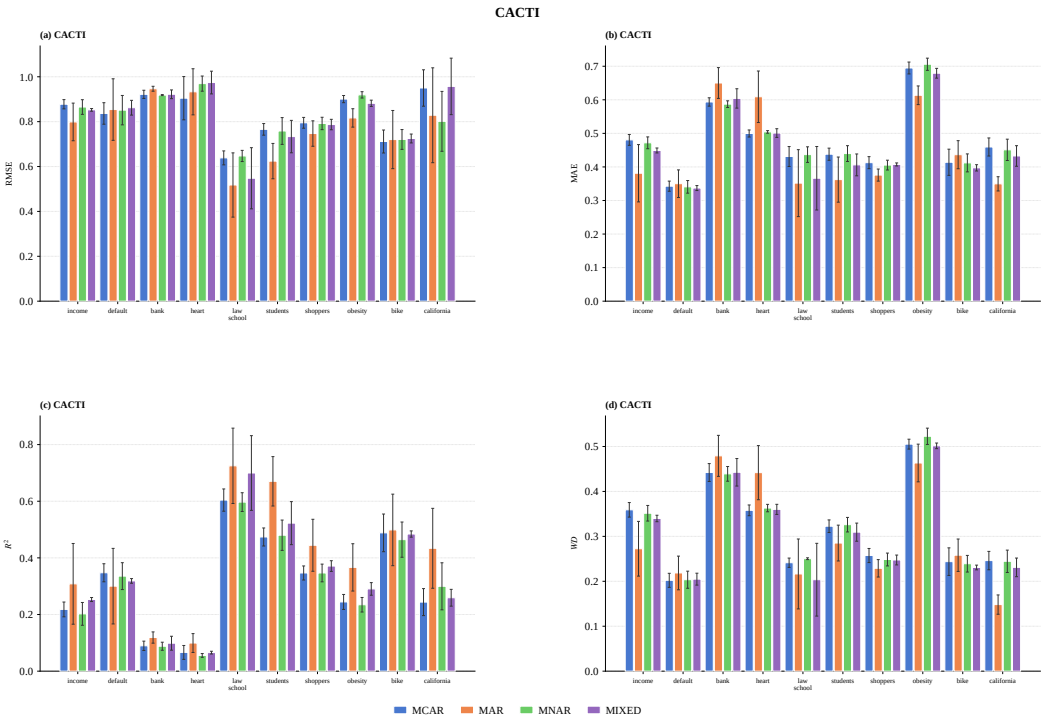


Figure 9: **Numerical imputation performance of CACTI is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

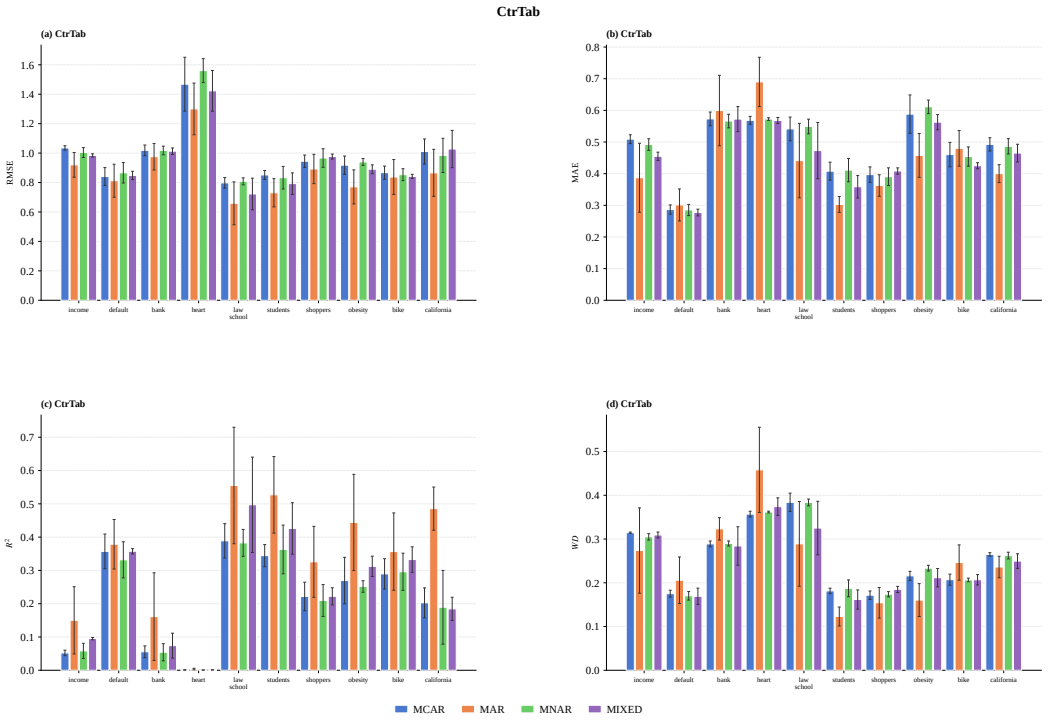


Figure 10: **Numerical imputation performance of CtrTab is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

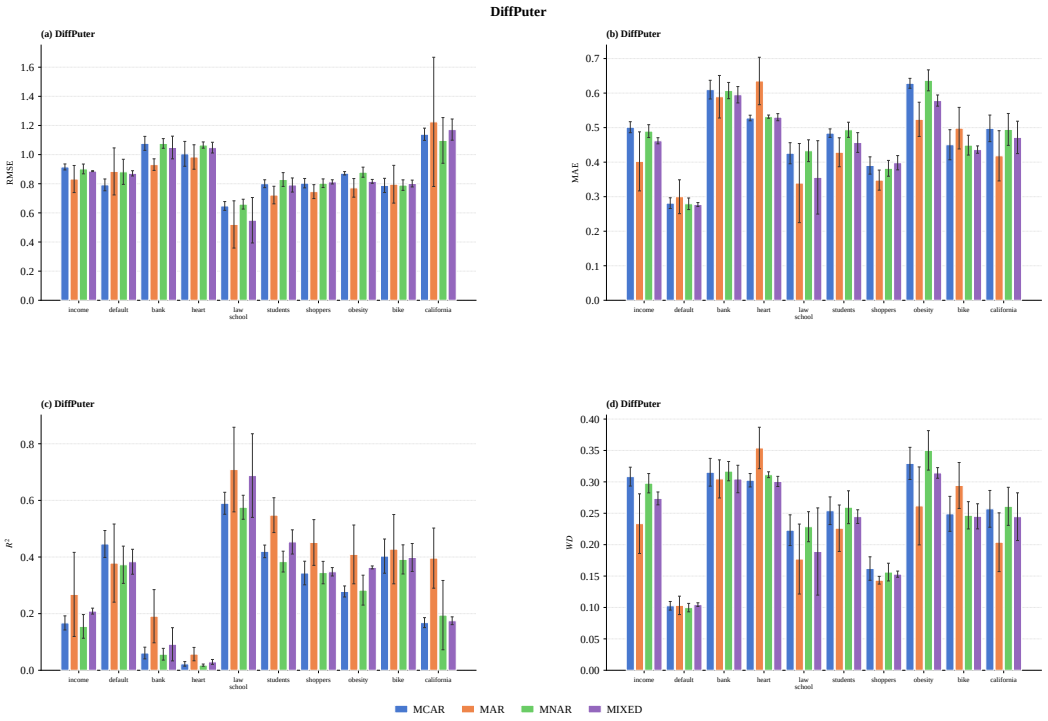


Figure 11: **Numerical imputation performance of DiffPuter is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

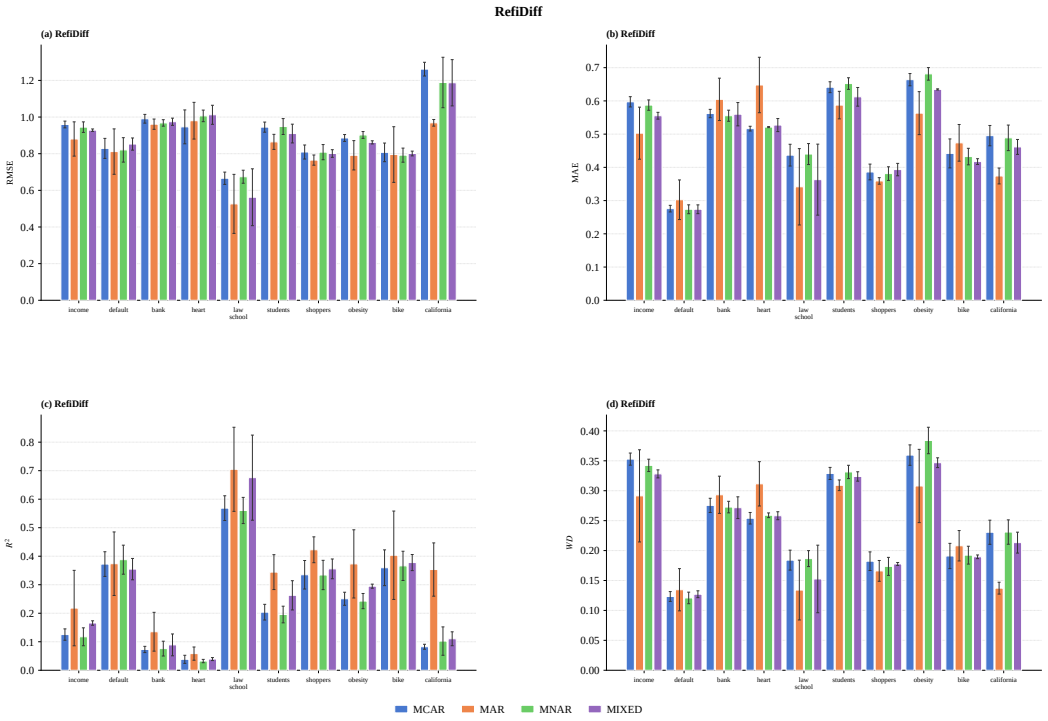


Figure 12: **Numerical imputation performance of RefiDiff is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

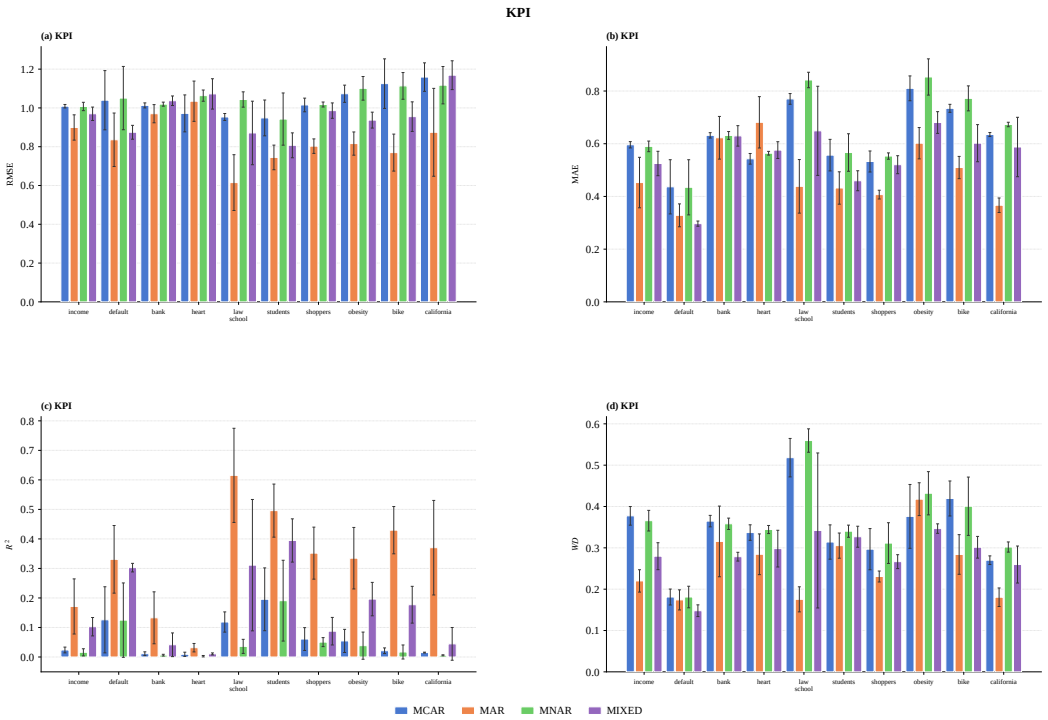


Figure 13: **Numerical imputation performance of KPI is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

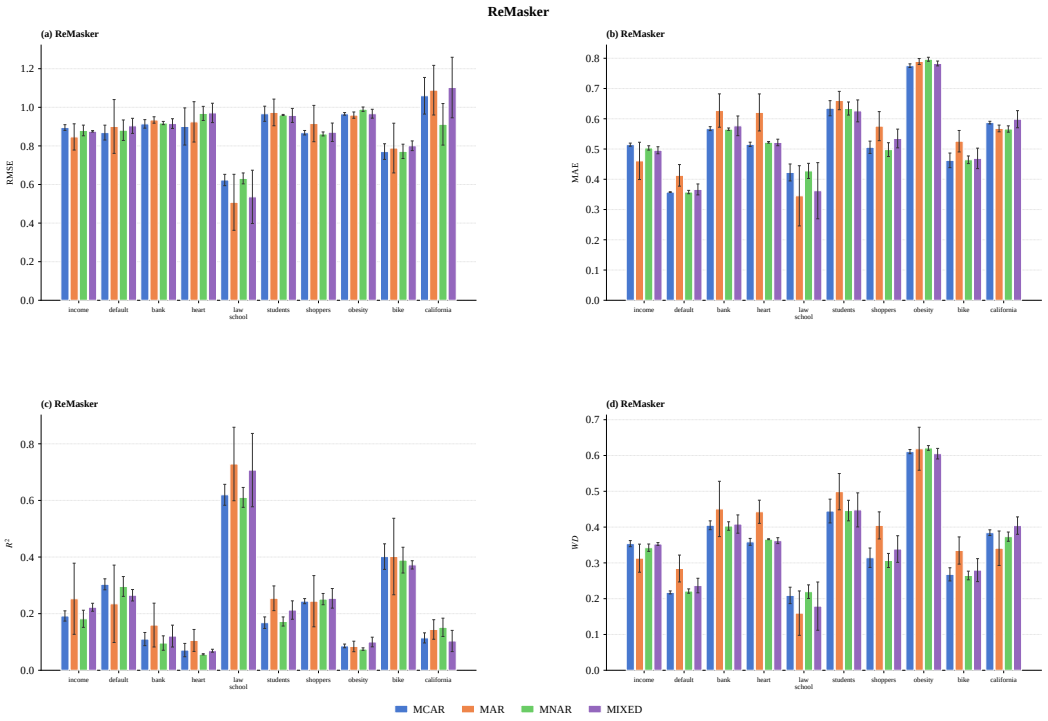


Figure 14: **Numerical imputation performance of ReMasker is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

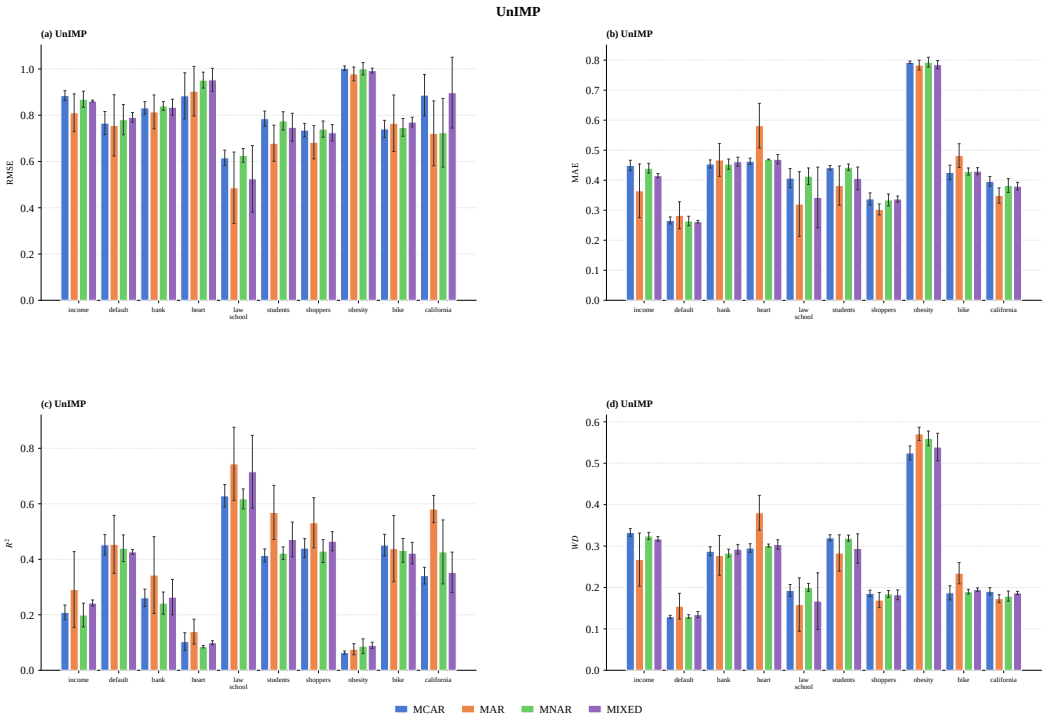


Figure 15: **Numerical imputation performance of UnIMP is shown by dataset.** Panels (a)–(d) report RMSE, MAE, R^2, and WD, respectively, under MCAR, MAR, MNAR, and MIXED missingness. Bars and error bars denote the mean and standard deviation over 30%–50% for each dataset.

Absolute difficulty varies across datasets: the same method can retain predictive signal on one table but have low R^2 on another. MAR often gives lower errors, although the ordering of MCAR, MAR, MNAR, and MIXED changes across datasets and methods. The spread across missingness severities is also concentrated in particular dataset–mechanism pairs. SemImputer performs well across several mechanisms, consistent with the aggregate and dataset-paired comparisons. Among the baselines, DiffPuter and RefiDiff are often stronger on WD than on RMSE or R^2; distributional and pointwise accuracy therefore yield different rankings.

D Further Analysis of Alignment and Utility Learning

D.1 Alignment and Utility Ablations

Figure 4 evaluates Structure-Semantic Representation Alignment by retaining only $\mathcal{L}_{\text{field}}$, only $\mathcal{L}_{\text{row}}$, or neither. All three variants retain semantic inputs. Table 3 separately evaluates counterfactual supervision and observation context within Counterfactual Utility-Guided Fusion; their validation diagnostics follow Appendix B.9.

The variant without counterfactual supervision uses the `--no-tsug` configuration, which sets $w_u(e) = 0$ throughout training and retains the utility gate. Its injection scale is $\gamma_{ij} = m_{ij}\rho_{ij}\lambda_{t_j}$, with no stop-gradient on ρ_{ij}. Reconstruction gradients therefore update G_ϕ and the context mappings P_g, P_c, C_m; the aligned inputs to the gate also remain attached to the computation graph. Paired endpoint errors provide post-hoc validation labels for this control, rather than a training loss.

This configuration uses standard linear-layer initialization for the gate and pools the semantic row view as $\mathbf{v}_i^s = Q_s(F^{-1}\sum_j \rho_{ij}\mathbf{a}_{ij}^s)$. Consequently, row-alignment gradients also reach the gate while $w_r(e) > 0$. Checkpoint selection uses the same validation reconstruction score, with eligibility starting at epoch index 1; under the default 200-epoch schedule, patience still starts at epoch 100.

D.2 Stability of Counterfactual Utility Learning

Figure 16 tracks validation AUROC for the margin-qualified utility label in diagnostic runs that activate utility supervision at epoch 40. The default schedule in Appendix B.7 begins utility learning at epoch 60. All dataset curves rise above chance and settle at different levels; the means over the last ten recorded epochs of each run range from 0.773 to 0.895 by dataset. Variation around activation generally decreases later in training. These trajectories use fixed validation masks; Figure 3(b) reports the separate held-out test diagnostic.

E Missingness-View Interleaving and Leakage Controls

E.1 Isolating Missingness-View Interleaving

Interleaved Missingness-View Training assigns each training row one mask from the 16-view bank at every epoch. A mini-batch can therefore contain different mechanisms and severities, and the same row appears under different observation contexts over training. Validation retains a fixed assignment. The model receives no view identity, and one selected checkpoint is used for all test settings.

The fixed-view control assigns each row one stratified view from the same bank before training and reuses it at every epoch. Architecture, objectives, optimizer, validation masks, checkpoint selection, and evaluation remain fixed. Relative to this control, SemImputer reduces RMSE, MAE, and WD by 7.8%, 10.2%, and 29.3%, respectively, and raises R^2 from 0.3334 to 0.4677 (Figure 4). The comparison supports a benefit from varying the observation context presented with each training row.

E.2 Leakage Controls

Complete rows are partitioned into training and test splits, with duplicate rows kept in the same split; validation is drawn from the training split. Numerical scaling is fitted on training data, and

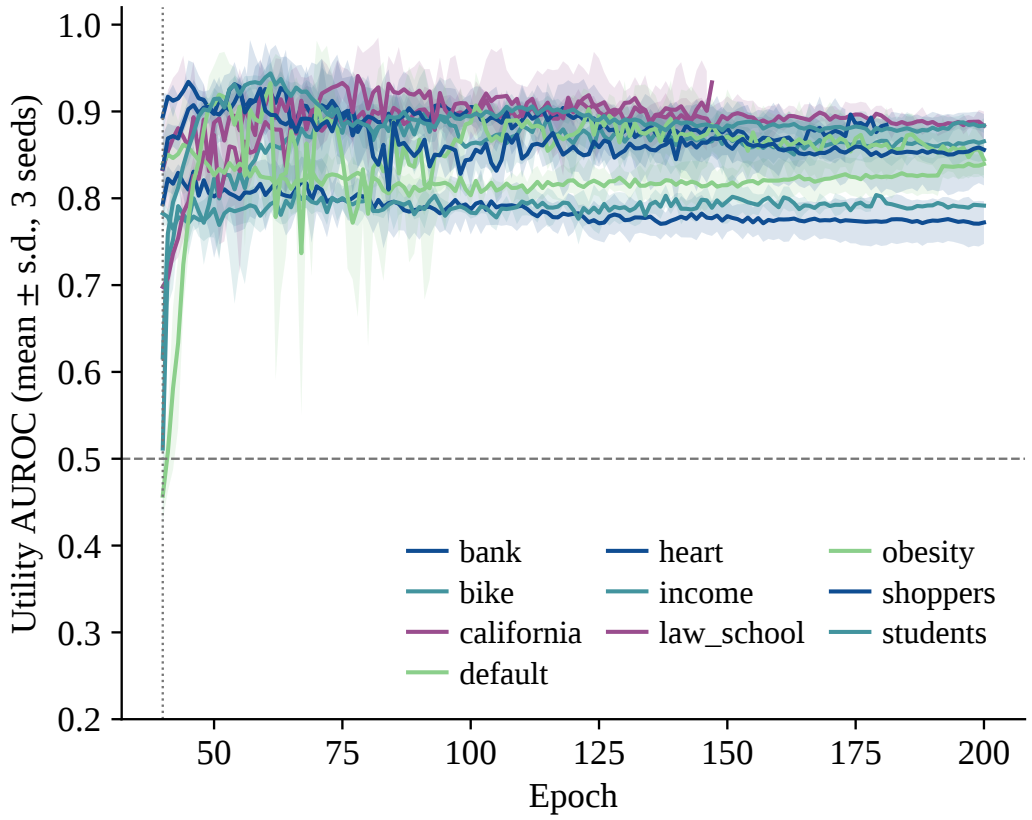


Figure 16: **Validation AUROC of counterfactual utility scores during training.** Curves evaluate the margin-qualified utility label on each dataset; shaded regions denote the mean ± standard deviation across the available seeds at each epoch. The vertical dotted line marks activation of utility supervision at epoch 40 for these diagnostic runs, and the horizontal dashed line marks chance performance. The default protocol in Appendix B.7 activates supervision at epoch 60. Runs may terminate at different epochs under early stopping.

categorical vocabularies are fixed schema metadata. Training masks are constructed for training rows; test-scenario masks are loaded after checkpoint selection.

For each selected mask, hidden positions are zeroed in the structural input and used as reconstruction targets. Observed fields use value-conditioned semantic embeddings; hidden fields use field descriptors without realized values. The frozen encoder processes each cached cell embedding independently, and the runtime mask selects the descriptor at hidden positions. Missingness context uses the mask and statistics from the masked input, without mechanism or severity labels.

During training, complete values enter the reconstruction loss and the no-gradient P_0/P_1 comparison that constructs utility labels. Validation targets select the checkpoint through fixed masks. Held-out utility diagnostics evaluate P_0/P_1 without parameter updates and use hidden values to calculate diagnostic errors and labels. Deployment evaluates only the gated prediction from the observed row, its mask, and the selected observation-aware semantic inputs.