BEYOND DIRECT LLM PREDICTION: OPEN-WORLD REUSE OF HETEROGENEOUS TABULAR LEARNWARES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) can draw on broad knowledge for prediction. Specialized tabular models offer complementary value even as LLM capabilities improve: they encode statistical relationships often learned from private data, while their explicitly defined inputs and targets support statistical auditing and validation. However, heterogeneous feature requirements and incomplete user inputs complicate their reuse. The *learnware* paradigm makes independently trained models identifiable and reusable by equipping each with a statistical specification of its training distribution, without sharing the original data. Building on this foundation, for the first time, we enable heterogeneous tabular learnwares to address open-world prediction queries whose observed features need not match any model's input requirements. Key to this capability is our finding that local statistical relationships encoded in different specifications can be jointly exploited to infer missing inputs required by other models. Our framework aligns specifications along shared features into a query-dependent context, from which a pretrained tabular prior infers missing inputs for model retrieval and prediction. Model-specific completion uncertainty guides filtering before aggregation, while statistical support under the same context determines whether to return the ensemble prediction or abstain. Across 41 tasks, our framework outperforms direct and refined LLM prediction and LLM-assisted model reuse. Controlled comparisons further show that statistical relationships encoded in other models' specifications can improve missing-feature completion and downstream prediction.

1 INTRODUCTION

Professional prediction queries can be expressed through LLM-based interfaces (Ji et al., 2026; Trirat et al., 2025). For example, a user might ask, "For a 52-year-old with a BMI of 29, a fasting glucose of 132 mg/dL, and an HbA1c of 5.9%, what is the risk of developing type 2 diabetes within five years?" LLMs can use broad knowledge to answer such questions, and their tabular prediction capabilities continue to develop (Hegselmann et al., 2023; Gardner et al., 2024). However, specialized tabular models offer complementary value: they encode statistical relationships often learned from private institutional data and define prediction through explicit variables and targets. These properties remain relevant even as LLM prediction improves.

First, specialized models can encode statistical relationships unavailable to a general-purpose LLM. Institutional records may reveal population-specific relationships between measured variables and outcomes that general domain knowledge does not establish. Such records are distributed across institutions and often cannot be shared. Reusing existing predictors offers access to this statistical knowledge without requiring the user to assemble a new labeled dataset (Borisov et al., 2024).

Second, tabular models organize prediction around human-defined variables and an explicit target. Defined measurements, outcome horizons, and intended populations provide concrete objects for statistical auditing, including assessment of calibration and subgroup errors on appropriate validation data (Riley et al., 2024). These also provide a basis for feature-level explanations, allowing users to examine how individual measurements contribute to a model's prediction (Lundberg et al., 2020). Task definition alone, however, does not establish accuracy or calibration.

Realizing this complementary value requires using existing models with the information a query provides. As Figure 1 illustrates, the observed features may not satisfy any model's input requirements,

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.

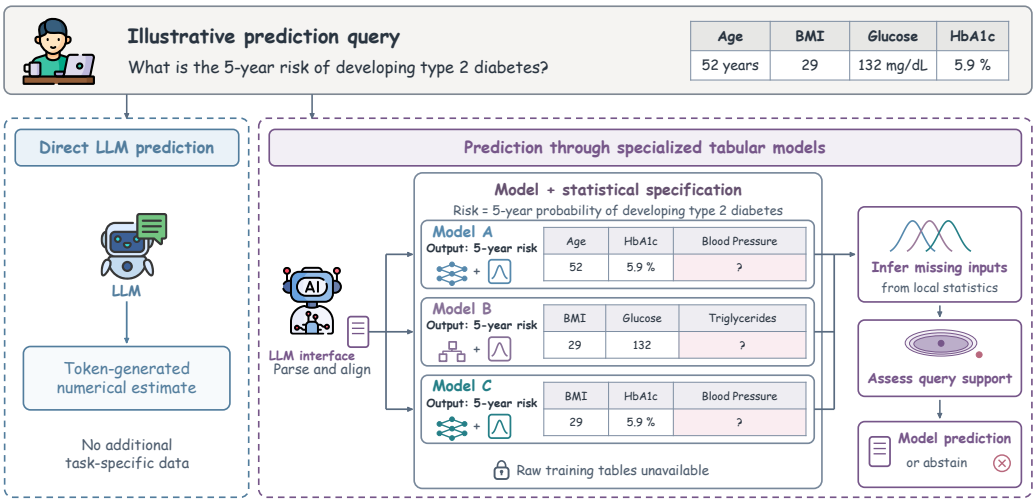

Figure 1: **A five-year diabetes-risk query and the barriers to model reuse.** Private-data models share a target but require different inputs. Although the query reports age, BMI, fasting glucose, and HbA1c, each model still requires one further input: blood pressure (A/C) or triglycerides (B). Their specifications relate missing inputs to observed features, enabling completion and query assessment without raw training tables; the system predicts or abstains.

and a query may lie in a region outside the model's capabilities. We study *open-world prediction* under these conditions, aiming to answer individual queries without labeled user examples while allowing abstention when statistical support is insufficient. Target compatibility and feature correspondences are assumed to have been established.

The *learnware* paradigm provides a foundation for addressing these barriers by equipping each model with a compact *statistical specification* of its training distribution (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026), enabling model identification and reuse without sharing the original data (Tan et al., 2024b). For tabular learnwares, reduced kernel mean embedding (RKME) specifications summarize training-input distributions through compact weighted sets of optimized synthetic feature vectors (Wu et al., 2023), which retain information about relationships among the variables observed by each developer. Prior work analyzes training-data preservation and bounds inversion risks for RKME specifications (Lei et al., 2024; 2026a). Feature-space heterogeneity nevertheless remains a central challenge: independently developed tabular models often operate on different sets of variables. Existing methods considered constrained heterogeneous scenarios: assuming predefined feature blocks, and available user-side task data to characterize the user's distribution and support cross-space matching (Tan et al., 2023; 2026a). We make the first attempt to face open-world heterogeneous prediction queries. Here, the observed feature combination can vary from query to query and need not satisfy any model's input requirements, while no additional user examples are available to construct a task specification or support adaptation. The central question is how the predictive capabilities and statistical knowledge distributed across a heterogeneous learnware collection can be jointly exploited to answer such queries, when potentially useful variable relationships are scattered across specifications covering different feature subsets.

Our key insight is that these local dependencies can be jointly exploited across models: a specification can supply evidence for inferring another model's missing inputs, even when its own predictor is not used. In Figure 1, C's specification relates blood pressure to BMI and HbA1c, so the observed BMI can inform completion of A's missing blood-pressure input even though A does not use BMI. Through shared variables, the library can thus provide not only predictors to select and combine, but also statistical context for using those predictors on incomplete queries. We develop specification-guided learnware reuse around this shared context. Statistical evidence guides both missing-input inference and the decision to return a prediction. Uncertainty about inferred inputs shapes how model predictions are used, and insufficient statistical support calls for abstention. The LLM serves as the request interface, while the reused models provide numerical predictions.

Our contributions are as follows:

- We enable open-world prediction with heterogeneous tabular learnwares without labeled user examples. Our key finding is that local statistical relationships encoded in different specifications can be jointly exploited to infer missing inputs required by other models, allowing a learnware to contribute even when its own predictor is not used.
- We develop a specification-guided prediction framework that aligns specifications into a query-dependent context for conditional input inference with a pretrained tabular prior. After specification-based model retrieval, uncertainty in each model's inferred inputs guides which predictions are combined. The completed query's estimated statistical support under the same context determines whether to return the ensemble prediction or abstain.
- Experiments across 41 tasks show improvements over direct and refined LLM prediction and LLM-assisted model reuse in task-group comparisons. Controlled comparisons further demonstrate that cross-specification statistical relationships can improve missing-feature completion and downstream prediction.

2 PRELIMINARY

Kernel mean embeddings provide a distribution representation in a reproducing kernel Hilbert space. For a positive-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, let $\mathcal{H}_k$ denote its associated space. A distribution P satisfying $\mathbb{E}_{X\sim P}\sqrt{k(X,X)} < \infty$ is represented by its kernel mean embedding (KME) (Smola et al., 2007):

$$\mu_P = \mathbb{E}_{X\sim P}[k(X,\cdot)] \in \mathcal{H}_k. \tag{1}$$

Replacing the expectation with an average over samples $\{x_j\}_{j=1}^n$ gives the empirical embedding $\widehat{\mu}_P = n^{-1}\sum_{j=1}^n k(x_j,\cdot)$. Distributional discrepancy is measured by the distance between embeddings: the maximum mean discrepancy (MMD) is $\|\mu_P - \mu_Q\|_{\mathcal{H}_k}$ (Gretton et al., 2012).

To obtain a compact statistical specification, reduced kernel mean embedding (RKME) replaces this empirical representation with a weighted set of synthetic support points (Wu et al., 2023). For a model f_i defined on $\mathcal{X}^{(i)}$, we write:

$$\sigma_i = \sum_{r=1}^{m_i} \beta_{ir} k(z_{ir},\cdot), \qquad z_{ir} \in \mathcal{X}^{(i)}, \quad m_i \ll n_i, \tag{2}$$

where n_i is the number of training samples. The support points and weights summarize the training input distribution and are stored with f_i as the learnware $L_i = (f_i, \sigma_i)$. They are optimized rather than retained as original training rows; Appendix B gives the construction objective. Specifications describe local distributions; combining them across feature spaces requires feature correspondence.

3 PROBLEM SETUP

We consider a prediction query from a user about an unobserved target Y with output space $\mathcal{Y}$. An LLM-based interface can extract the requested outcome and the available feature values. The LDS organizes tabular models with different feature subsets and potentially different output spaces. Before reuse, the system matches model targets to the requested outcome and establishes correspondences between the query attributes and model inputs (Tan et al., 2024b). These steps provide query-relevant models and shared feature indices for the procedures below. For classification, the output coordinates of the selected models refer to the same target labels.

Let $X_q \in \mathcal{X}_q$ denote the random variable for aligned feature q, with global feature space $\mathcal{X}_{\text{all}} = \prod_{q=1}^Q \mathcal{X}_q$. For an index set $A \subseteq \{1,\dots,Q\}$, write $\mathcal{X}_A = \prod_{q\in A}\mathcal{X}_q$. The parsed query supplies an unlabeled vector $x \in \mathcal{X}_{C_u}$, where $C_u \subseteq \{1,\dots,Q\}$ contains its observed features. Reuse is performed per query, without constructing a user specification from multiple user examples. Let N denote the number of query-relevant learnwares, collected as $\mathcal{L} = \{L_i = (f_i,\sigma_i)\}_{i=1}^N$. Here, $f_i : \mathcal{X}_{C_i} \to \mathcal{Y}$ is model i, with local input space $\mathcal{X}^{(i)} = \mathcal{X}_{C_i}$, and σ_i is its statistical specification. Model i was trained on private data $D_i = \{(x_{ij}, y_{ij})\}_{j=1}^{n_i}$, where $x_{ij} \in \mathcal{X}_{C_i}$ and $y_{ij} \in \mathcal{Y}$. These models can still require different feature sets C_i despite their compatible targets.

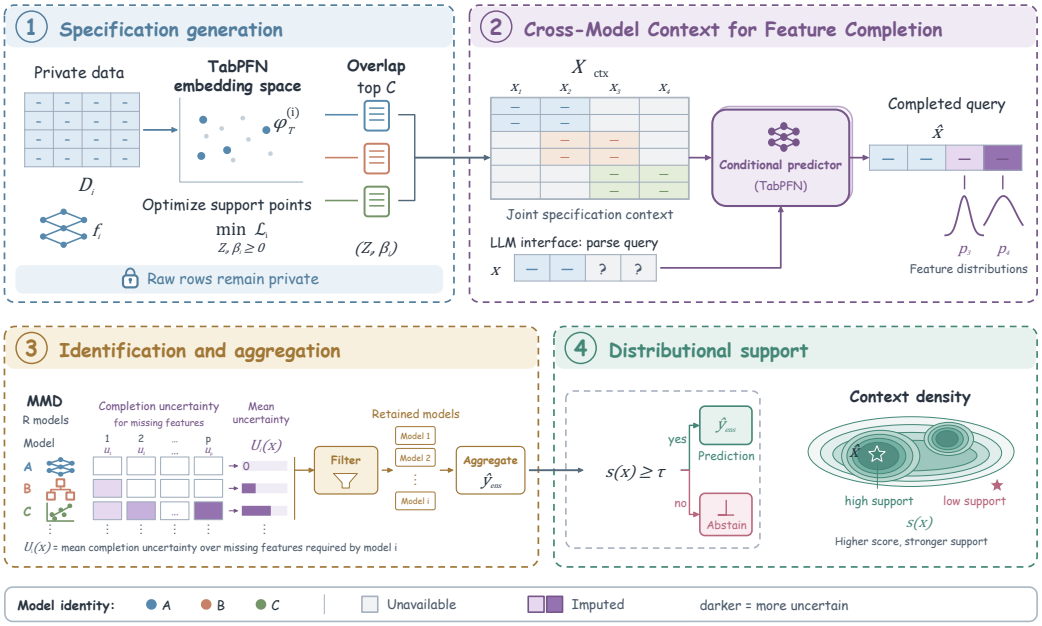

Figure 2: **Specification-guided reuse of heterogeneous tabular learnwares.** (1) Developers optimize compact support sets in a fixed TabPFN representation. (2) Specifications from feature-overlap candidates form a sparse context for input completion and uncertainty estimation. (3) MMD retrieves models using the completed query; models requiring highly uncertain inputs are filtered out before prediction averaging. (4) Support under the same context determines whether to return the ensemble prediction or abstain. Context gaps remain unfilled; $\mathcal{L}_i$ is the construction objective.

Our objective is to answer the query using external models without accessing their training data or requiring labeled user examples. Open-world reuse involves missing inputs ($C_i \not\subseteq C_u$) and queries that may fall outside the regions represented by the library. Given target and feature correspondences, the system must identify and reuse suitable models and assess support before returning their prediction $\widehat{y}_{\text{out}}(x) \in \mathcal{Y}$ or abstaining with $\perp$. The LLM interface specifies the requested target and observed features; the reused models supply the numerical prediction.

4 Proposed Approach

Figure 2 outlines our framework. Local specifications are constructed for conditional inference and aligned across feature-overlap candidates to complete the query. The completed input supports model identification, followed by uncertainty filtering and aggregation. Support under the same context determines whether to return the ensemble prediction or abstain.

4.1 Specification Generation for Conditional Inference

Specifications are used here as context for conditional feature inference as well as for model identification. A specification optimized in the original feature space may be poorly suited to the representation used by a pretrained conditional predictor. We therefore construct support points by matching distributions in the task-conditioned representation of a frozen TabPFN model (Hollmann et al., 2025; Grinsztajn et al., 2025). Let $\phi_{\mathrm{T}}^{(i)}$ denote this embedding for developer i. We use out-of-fold feature extraction (Ye et al., 2025) to exclude the label of each training instance from its embedding context, reducing the discrepancy between representations extracted for labeled training data and unlabeled queries. The developer optimizes m_i support points $Z_i = \{z_{ir}\}_{r=1}^{m_i} \subseteq \mathcal{X}_{C_i}$ and nonnegative weights $\boldsymbol{\beta}_i = (\beta_{i1}, \ldots, \beta_{im_i})^\top$:

$$\min_{Z_i,\,\boldsymbol{\beta}_i\succeq 0}\left\|\frac{1}{n_i}\sum_{j=1}^{n_i}k\Big(\phi_{\mathrm{T}}^{(i)}(x_{ij}),\cdot\Big)-\sum_{r=1}^{m_i}\beta_{ir}k\Big(\phi_{\mathrm{T}}^{(i)}(z_{ir}),\cdot\Big)\right\|_{\mathcal{H}_k}^2. \tag{3}$$

This preserves the RKME reduced-set objective in the task-conditioned representation. Inspired by Wu et al. (2023), we alternate between optimizing weights and support coordinates; Appendix C.1 gives details. Model-generated pseudo labels $\widetilde{y}_{ir} = f_i(z_{ir})$ are used only to embed support points during developer-side generation. The stored specification is $(Z_i, \boldsymbol{\beta}_i)$, with coordinates in the original local feature space. At deployment, completion and support assessment use these coordinates without pseudo labels; kernel comparisons also use the weights. Developers share optimized support points, not their training records. For conventional RKME specifications, prior work studies resistance to training-data disclosure and inference attacks (Lei et al., 2024; 2026a).

4.2 Joint Specification Context and Query Completion

After query parsing and feature alignment, the LDS ranks query-relevant learnwares by Jaccard overlap (Jaccard, 1901), $o_i(x) = |C_u \cap C_i|/|C_u \cup C_i|$. It selects up to C with nonzero overlap in a candidate pool $\mathcal{P}(x)$. These candidates supply the completion context; statistical identification using feature values follows after completion.

Let $J = C_{\mathrm{joint}}(x) = C_u \cup \bigcup_{i\in\mathcal{P}(x)} C_i$ be the joint feature set. Aligning candidate support points by feature gives a context $A = X_{\mathrm{ctx}}(x)$ with one row per support point, indexed by (i, r):

$$A_{(i,r),q} = \begin{cases} [z_{ir}]_q, & q \in C_i, \\ \mathrm{NaN}, & q \in J \setminus C_i. \end{cases} \tag{4}$$

Shared coordinates connect local views of different feature subsets. Alignment thus combines the specifications into a cross-model context, but leaves both the context and query partially observed.

A pretrained conditional predictor supplies a prior for inference from this sparse context. We instantiate it with TabPFN-2.5 (Grinsztajn et al., 2025), treating each missing feature as a temporary prediction target. For $q \in J \setminus C_u$, use rows of A with observed q values. Their qth column forms a response $y^{(q)}$; their remaining columns form $X^{(q)}$, which may still contain missing entries. Let v be the current query, initialized with x on observed coordinates and missing values elsewhere, and let v_{-q} exclude coordinate q. Then

$$p_q(\cdot \mid v, A) = \mathcal{F}_\theta\Big(\cdot \mid v_{-q}; X^{(q)}, y^{(q)}\Big), \tag{5}$$

where $\mathcal{F}_\theta$ is the conditional predictor with pretrained parameters θ fixed. The context response is a feature value, not the query target Y or a developer pseudo label. These distributions fill missing coordinates while preserving observations; completed values can inform later feature predictions. The context itself remains sparse. Each candidate f_i receives the completed input $\widehat{x}_{C_i}$, and the resulting feature distributions quantify completion uncertainty. The resulting completions depend on both the available local relationships and the pretrained prior: overlapping specifications describe local variable relationships, but do not generally determine a unique joint distribution.

4.3 Learnware Identification and Uncertainty-Aware Aggregation

Feature overlap alone cannot distinguish training distributions. We represent the completed query $\widehat{x}$ as a unit-weight support point and compare it with the normalized specification of each candidate using MMD. Support points retain coordinates on C_i and copy the completed query elsewhere, adding no distance penalty. Appendix C.2 gives the full RKHS distance and weight normalization. Using RKME distance ranking (Tan et al., 2024b), we retrieve up to R nearest candidates in $\mathcal{R}(x)$.

Statistical proximity does not establish how strongly a prediction depends on uncertain completions. We therefore assess uncertainty only over the inputs each model needs. For missing feature q, let p_q be its conditional distribution from Eq. (5), expressed as categorical probabilities or numerical-bin masses over $\mathcal{A}_q$. Its normalized entropy is $U_q(x) = H(p_q)/\log|\mathcal{A}_q|$, where H is Shannon

entropy (Shannon, 1948); set $U_q(x) = 0$ when $|\mathcal{A}_q| = 1$. For model i, let $M_i(x) = C_i \setminus C_u$ be its required missing features. Their mean uncertainty is

$$U_i(x) = \frac{\sum_{q \in M_i(x)} U_q(x)}{\max\{1, |M_i(x)|\}}, \tag{6}$$

which is zero for an already compatible model. We filter out models in $\mathcal{R}(x)$ with high completion uncertainty $U_i(x)$ and denote the retained set by $\mathcal{I}(x)$. For regression, $\widehat{y}_{\text{ens}}(x) = |\mathcal{I}(x)|^{-1} \sum_{i \in \mathcal{I}(x)} f_i(\widehat{x}_{C_i})$; for classification, we return the class with the largest mean predicted probability. Filtering favors less uncertain inputs but does not guarantee a correct prediction or sufficient contextual support.

4.4 Specification-Based Abstention

Even a completed input with low uncertainty may have weak support under the training distributions of the models. We assess its support using the same conditional predictor and candidate context A used to complete that query. Restrict both A and $\widehat{x}$ to columns with observed support values, and let $\pi = (\pi_1, \ldots, \pi_d)$ order these d features. For each feature, the conditional predictor uses rows observed in that column to predict it from preceding features. The scored query is never added to the context. Let p_{T} denote the conditional predictions from this context. For a feature order π, we define $\ell_\pi(x)$ as the log of the resulting factorized estimate of joint support for the completed query:

$$\ell_\pi(x) = \sum_{j=1}^{d} \log p_{\text{T}}\left(\widehat{x}_{\pi_j} \mid \widehat{x}_{\pi_{<j}}, A\right), \tag{7}$$

where $\pi_{<j}$ denotes preceding feature indices and the first factor is marginal. Each factor is a density for numerical features or a probability mass for categorical features. Because the conditional estimates can depend on feature order, we combine the joint support estimates from orders in Π by averaging them before taking the logarithm. The resulting score $s(x)$ measures support for the completed query under its specification context. The system output $\widehat{y}_{\text{out}}(x)$ is the ensemble prediction when this score meets a threshold τ, chosen for the scored feature set and preprocessing, and an abstention otherwise:

$$s(x) = \log\left[\frac{1}{|\Pi|} \sum_{\pi \in \Pi} \exp\big(\ell_\pi(x)\big)\right], \qquad \widehat{y}_{\text{out}}(x) = \begin{cases} \widehat{y}_{\text{ens}}(x), & s(x) \geq \tau, \\ \perp, & s(x) < \tau. \end{cases} \tag{8}$$

A lower $s(x)$ indicates weaker contextual support. The threshold controls whether the entire ensemble output is released; it does not trigger another round of model selection. Support is assessed without access to raw developer data or query labels. Our implementations of feature completion and support assessment follow TabPFN Extensions (Prior Labs, 2026). Algorithm 1 summarizes the query-time procedure after specification generation and target and feature matching.

5 Experiments

We evaluate the framework on 41 prediction tasks through three research questions:

- **Q1:** Can variable relationships encoded in other specifications support feature completion?
- **Q2:** Does the full framework improve zero-shot prediction over direct or refined LLM prediction and model reuse with LLM-based completion?
- **Q3:** How do feature completion, uncertainty-aware aggregation, and specification-based abstention contribute to open-world reuse?

Appendix E tests Q1's cross-specification relationships with marginal-preserving shuffles and a bridge-chain control, then reports fixed-predictor results by task. It expands Q2 with task-level comparisons of the main prediction baselines and Ours. For Q3, it shows query-level effects of uncertainty filtering and evaluates abstention through risk–coverage analysis. Appendix F compares raw and TabPFN-aware specifications, tests direct TabPFN prediction against model reuse after completion, and varies feature availability to assess robustness across different levels of query completeness.

5.1 EXPERIMENTAL SETUP

Datasets. The benchmark comprises 41 prediction tasks: 30 General tasks from BeyondArena (Purucker et al., 2026) (16 classification and 14 regression tasks), 8 Financial classification tasks from FinBench (Yin et al., 2023), and 3 clinical prediction tasks from the MIMIC-III benchmark (Johnson et al., 2016; Harutyunyan et al., 2019) (2 classification and 1 regression task). The clinical tasks are in-hospital mortality (IHM), decompensation (DECOMP), and length-of-stay (LOS) prediction.

Task construction. For each task, developers form a separate library of LightGBM (Ke et al., 2017) models trained on local rows and heterogeneous feature subsets, submitting models and specifications without training data. Users provide partially observed test queries with a shared task target and feature schema; observed inputs need not match the interface of any model.

Contenders. *Direct LLM* predicts the target from observed features; *LLM-Refine* critiques and revises that prediction once using self-feedback (Madaan et al., 2023). Both use LLM-generated predictions without library models. *LLM-Completed Models* uses the LLM only to fill missing inputs, then averages overlap-retrieved models' probabilities or regression outputs. These baselines receive no specifications. *Ours* uses TabPFN v2.5 and the joint specification context for missing-input completion, RKME distance for model retrieval, and completion uncertainty for selective aggregation. Its target predictions come from library models. The main comparison evaluates predictions on all test queries without applying the abstention rule. Q3 separately examines how withholding predictions for low-support queries changes prediction risk as coverage decreases.

Configuration and evaluation. We evaluate Qwen3.8-flash (Alibaba Cloud, 2026) and DeepSeek-V4-flash (DeepSeek-AI, 2026) as LLM backbones; *LLM-Completed Models* uses Qwen3.8-flash for completion in the main comparison. *Ours* retrieves $R = 8$ models and discards those requiring highly uncertain feature completion, leaving a query-dependent number $K(x)$ of models for aggregation. Each task uses 200 test queries, with query rows and masks matched across methods within each of three random seeds. We report seed-averaged accuracy or RMSE; General regression summaries use target-standardized RMSE and within-task RMSE ranks. Construction, decoding, and aggregation details appear in Appendix D.

5.2 Q1: CAN LOCAL SPECIFICATIONS SUPPORT FEATURE COMPLETION ACROSS MODELS?

For each query, we hold the downstream predictor f_1 and the observed inputs fixed across completion methods. *Mean-F* fills missing inputs with feature-wise means of the support points associated with f_1. Both remaining methods use TabPFN for conditional completion: *TabPFN-F-Spec* uses only the support points associated with f_1, whereas *TabPFN-All-Specs* uses aligned points from all candidate specifications. Each completed query is passed to f_1.

Table 1: **Local specifications as completion context.** Downstream prediction on 41 tasks, with the same predictor f_1 used across completion methods for each query.

Task group	Metric	Mean-F	TabPFN-F-Spec	TabPFN-All-Specs
General classification	Acc. (%) ↑	76.354 ± 0.865	78.948 ± 0.851	$\mathbf{79.344 \pm 0.641}$
General regression	RMSE rank ↓	2.643 ± 0.071	1.976 ± 0.180	$\mathbf{1.381 \pm 0.149}$
Financial classification	Acc. (%) ↑	80.562 ± 0.165	80.729 ± 0.567	$\mathbf{81.979 \pm 0.485}$
MIMIC-III classification	Acc. (%) ↑	88.833 ± 2.376	$\mathbf{89.333 \pm 0.520}$	89.250 ± 1.500
MIMIC-III regression	RMSE ↓	0.868 ± 0.030	0.873 ± 0.028	$\mathbf{0.859 \pm 0.052}$

With f_1, observed query values, and TabPFN held fixed, adding aligned support points from other specifications improves four of five group metrics (Table 1). Financial accuracy rises from 80.729% to 81.979%, and General regression rank improves from 1.976 to 1.381. These gains show that a specification can improve predictions from another model even when the model associated with that specification is not used. We also shuffle auxiliary specifications to disrupt variable relationships while preserving feature marginals; relationship recovery worsens on most evaluated tasks, supporting the value of relational information in the joint context (Appendix E.1.1).

5.3 Q2: Does the Full Framework Improve Zero-Shot Tabular Prediction?

We test whether specification-guided reuse improves on LLM prediction, refinement, and model reuse with LLM-based completion. Query rows, masks, and available libraries are paired within each task and seed; Q3 isolates individual components.

Table 2: **Zero-shot prediction across 41 tasks.** Mean accuracy/RMSE (NRMSE for General regression) averages task scores; average rank summarizes within-task method ranks (lower is better).

Metric	Direct LLM		LLM-Refine		LLM-Completed Models	Ours
	Qwen	DeepSeek	Qwen	DeepSeek		
General classification (16 tasks)						
Mean accuracy (%) ↑	57.490	62.594	58.115	62.615	78.115	**81.573**
Average rank ↓	4.781	4.250	4.719	4.250	1.813	**1.188**
General regression (14 tasks)						
Mean NRMSE ↓	1.179	1.069	1.467	1.116	0.859	**0.616**
Average rank ↓	4.571	3.679	5.143	3.679	2.929	**1.000**
Financial classification (8 tasks)						
Mean accuracy (%) ↑	68.021	65.958	70.167	66.083	79.771	**83.146**
Average rank ↓	4.875	4.563	3.625	4.438	2.125	**1.375**
MIMIC-III classification (2 tasks)						
Mean accuracy (%) ↑	61.417	68.417	70.667	68.750	71.333	**90.583**
Average rank ↓	5.500	4.000	2.500	4.000	4.000	**1.000**
MIMIC-III regression (1 task)						
Mean RMSE ↓	1.058	0.951	1.038	0.951	0.889	**0.849**
Average rank ↓	6.000	4.000	5.000	3.000	2.000	**1.000**

Table 2 shows that LLM-Completed Models improves mean accuracy or error over the strongest direct or refined LLM baseline in all five groups, demonstrating the value of reusing library predictors. Ours improves further in every group: General classification accuracy rises from 78.115% to 81.573%, and General regression NRMSE falls from 0.859 to 0.616. The largest classification gain occurs on MIMIC-III (90.583% vs. 71.333%). Ours leads on 35 of 41 tasks overall (Appendix E.2.1), supporting the added value of specification-guided reuse.

5.4 Q3: How Do the Components Support Open-World Reuse?

Feature completion. Reused predictors must cope with inputs that are absent from the query. We hold the model library, overlap retrieval, mean aggregation, queries, and masks fixed, and compare the four input-handling methods in Table 3. *Native Missing* passes missing values directly to LightGBM. *Mean* fills each missing feature with its mean in the joint specification context. *Random* draws a value from the marginal distribution of that feature in the same context. *TabPFN* infers missing features conditionally on the observed query values and the joint specification context. Table 3 shows that TabPFN completion improves on native handling in every group and on both simple fills in four of five group metrics. Completion settings appear in Appendix F.2.

Uncertainty-aware aggregation. To examine how completion uncertainty guides aggregation, we hold the eight retrieved models and their completed inputs fixed. For each tested retention size, we aggregate the least-uncertain models.

Table 4 compares Mean over all eight models with the best tested fixed-size result for each task. The best tested size improves on Mean in all eight tasks, while the preferred size varies across them. Additional aggregation details and query-level examples appear in Appendix E.3.1.

Specification-based abstention. We vary the support-score threshold (Section 4.4) to assess whether rejecting less-supported queries improves the risk–coverage trade-off (Geifman et al., 2019). Figure 3 shows seed-specific curves for Airline Satisfaction, 5G Energy, CD1, and IHM.

Table 3: **Feature-completion methods with overlap retrieval and mean aggregation fixed.** General regression ranks compare these four methods within each task and seed (1 is best).

Task group	Metric	Native Missing	Mean	Random	TabPFN
General classification (16)	Acc. (%) ↑	76.323 ± 0.621	78.177 ± 0.629	78.448 ± 0.611	**80.948 ± 1.063**
General regression (14)	RMSE rank ↓	3.857 ± 0.000	2.000 ± 0.189	2.905 ± 0.041	**1.238 ± 0.149**
Financial classification (8)	Acc. (%) ↑	79.167 ± 0.764	**82.875 ± 0.325**	81.104 ± 0.816	82.875 ± 1.985
MIMIC-III classification (2)	Acc. (%) ↑	57.417 ± 4.260	89.083 ± 3.503	90.417 ± 1.155	**90.500 ± 1.732**
MIMIC-III regression (1)	RMSE ↓	0.927 ± 0.042	0.870 ± 0.019	0.867 ± 0.009	**0.849 ± 0.023**

Table 4: **Uncertainty-aware aggregation on eight tasks.** Mean aggregation over all eight retrieved models is compared with the best result for each task among the tested fixed retention sizes.

Task	Metric	All 8 models	Best fixed-size result
CC3	Acc. (%) ↑	73.833 ± 1.528	**74.667 ± 1.607**
Fitness Club	Acc. (%) ↑	70.667 ± 2.566	**71.167 ± 1.893**
In-Vehicle Coupon	Acc. (%) ↑	63.833 ± 1.041	**65.667 ± 2.021**
SDSS-17	Acc. (%) ↑	86.167 ± 2.021	**86.667 ± 1.258**
5G Energy	RMSE ↓	0.606 ± 0.036	**0.603 ± 0.039**
Airfoil Noise	RMSE ↓	0.938 ± 0.083	**0.926 ± 0.083**
Fiat 500	RMSE ↓	0.634 ± 0.028	**0.631 ± 0.025**
Video Game FPS	RMSE ↓	0.117 ± 0.027	**0.116 ± 0.027**

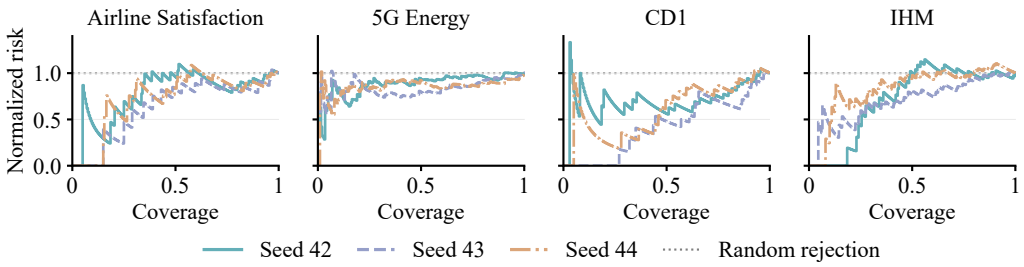

Figure 3: **Specification-based abstention across four tasks.** Coverage is the fraction of test queries receiving predictions. Each seed uses 200 queries. Risk is classification error or mean absolute error on retained queries, normalized by the full-coverage risk of the corresponding run; the dotted line denotes expected random rejection.

Across these 12 task–seed runs, normalized AURC gain ranges from 0.114 to 0.572, indicating lower risk integrated over coverage than random rejection. Across all 41 tasks, mean gain is positive on 25 tasks. The normalization and AURC definition appear in Appendix E.3.2.

6 Conclusion

This work studies open-world reuse of heterogeneous tabular learnwares, where incomplete queries may not match the inputs required by available models and original training data are unavailable to infer missing inputs. The learnware paradigm provides the needed statistical interface: specifications characterize individual models while their aligned local relationships help infer inputs required by other models. Their shared context supports feature completion, query-specific model retrieval, uncertainty-aware aggregation, and a decision to return the model prediction or abstain. Across 41 tasks, the framework leads the evaluated LLM prediction and model-reuse methods at the group level; controlled comparisons show that specifications from other learnwares improve feature completion and downstream prediction. Future work could calibrate support assessment under distribution shift and use completion uncertainty to guide requests for additional user measurements.

AI USE STATEMENT

The research questions, conceptual framing, method, experimental protocol, analyses, and conclusions in this work were developed by the authors. Generative AI tools supported language polishing and the writing of selected code segments. In the manuscript, they helped refine grammar, phrasing, and readability; the authors checked these edits against the intended technical meaning. The authors also reviewed AI-assisted code before use. The tools did not determine scientific claims or experimental choices. The authors remain responsible for the final text, code, results, and conclusions.

ETHICS STATEMENT

This work studies the reuse of tabular prediction models using existing benchmark datasets. The clinical experiments are conducted for methodological evaluation and are not intended to guide patient care; inferred feature values are not substitutes for clinical measurements. For experiments involving the Qwen3.8-Flash and DeepSeek-V4-Flash APIs, we handled data submitted to these services in accordance with the applicable dataset licenses, terms of use, and restrictions on sharing data with third-party services. At reuse time, the framework operates on trained models and optimized statistical specifications without accessing the original developer training records. Applications in healthcare, finance, and other consequential settings require appropriate data governance and evaluation of predictive reliability and potential biases. Support-based abstention provides an additional safeguard but does not eliminate prediction errors.

REPRODUCIBILITY STATEMENT

We describe the proposed method and experimental procedures in the main text and appendices, including dataset preparation, model-library construction, statistical specification generation, feature completion, and evaluation protocols. We report the model configurations, hyperparameters, random seeds, and aggregation procedures used in our experiments. Additional implementation details and task-level results are provided to facilitate reproduction and comparison with future work.

REFERENCES

Alibaba Cloud. Qwen3.8-Flash. Model Studio documentation, 2026. URL https://www.alibabacloud.com/help/en/model-studio/qwen3-8-flash. Accessed September 15, 2026.

Alan Arazi, Eilam Shapira, and Roi Reichart. TabSTAR: A tabular foundation model for tabular data with text fields. In *Advances in Neural Information Processing Systems 38*, pages 172108–172161, 2025.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, 2024.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. arXiv:2606.19348, 2026.

Josh Gardner, Juan C. Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. In *Advances in Neural Information Processing Systems 37*, pages 45155–45205, 2024.

Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. Bias-reduced uncertainty estimation for deep neural classifiers. In *The 7th International Conference on Learning Representations*, 2019.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski,

Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. arXiv:2511.08667, 2025.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, 2019.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, pages 5549–5581, 2023.

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.

Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37(142):547–579, 1901.

An-Yang Ji, Jun-Peng Jiang, De-Chuan Zhan, and Han-Jia Ye. TopBench: A benchmark for implicit predictive reasoning in tabular question answering. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pages 3146–3154, 2017.

Hao-Yi Lei, Zhi-Hao Tan, and Zhi-Hua Zhou. On the ability of developers' training data preservation of learnware. In *Advances in Neural Information Processing Systems 37*, pages 36471–36513, 2024.

Hao-Yi Lei, Zhi-Hao Tan, and Zhi-Hua Zhou. A statistical framework for analyzing specification resistance to learnware-inversion risks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a.

Hao-Yi Lei, Jin-Hui Wu, Zhi-Hao Tan, and Zhi-Hua Zhou. On the privacy-preserving capabilities of PAVE specifications in learnware. In *Proceedings of the 35th International Joint Conference on Artificial Intelligence*, pages 4465–4473, 2026b.

Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Identifying and reusing learnwares across different label spaces. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence*, pages 5734–5742, 2025.

Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems 38*, pages 172692–172722, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36*, pages 46534–46594, 2023.

Prior Labs. TabPFN Extensions. Software repository, version 0.6.2, 2026. URL https://github.com/PriorLabs/tabpfn-extensions. Accessed September 15, 2026.

Lennart Purucker, Andrej Tschalzev, Nick Erickson, Gioia Blayer, David Holzmüller, Alan Arazi, Alexander Pfefferle, Mustafa Tajjar, Gaël Varoquaux, and Frank Hutter. Beyond IID: How general are tabular foundation models, really? arXiv:2606.30410, 2026.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 50817–50847, 2025.

James Requeima, John Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. LLM processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems 37*, pages 109609–109671, 2024.

Richard D. Riley, Lucinda Archer, Kym I. E. Snell, Joie Ensor, Paula Dhiman, Glen P. Martin, Laura J. Bonnett, and Gary S. Collins. Evaluation of clinical prediction models (part 2): How to undertake an external validation study. *The BMJ*, 384:e074820, 2024.

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

Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

Hao-Yu Shi, Zhi-Hao Tan, Zi-Chen Zhao, Yang Yu, and Zhi-Hua Zhou. A study on PAVE specification for learnware. In *The 14th International Conference on Learning Representations*, 2026.

Aliaksandra Shysheya, John Bronskill, James Requeima, Shoaib Ahmed Siddiqui, Javier González, David Duvenaud, and Richard E. Turner. JoLT: Joint probabilistic predictions on tabular data using LLMs. arXiv:2502.11877, 2025.

Alexander Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*, pages 13–31, 2007.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. Handling learnwares developed from heterogeneous feature spaces without auxiliary data. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, pages 4235–4243, 2023.

Peng Tan, Hai-Tian Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Handling learnwares from heterogeneous feature spaces with explicit label exploitation. In *Advances in Neural Information Processing Systems 37*, pages 12767–12795, 2024a.

Peng Tan, Zhi-Hao Tan, Hai-Tian Liu, and Zhi-Hua Zhou. Handling learnwares from heterogeneous feature and label spaces with explicit label exploitation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026a. In press.

Peng Tan, Fei-Fan Yang, Zhi-Hao Tan, and Zhi-Hua Zhou. Tabular learnwares can be repurposed for seemingly irrelevant new tasks. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, volume 40, pages 25778–25786, 2026b.

Zhi-Hao Tan, Jian-Dong Liu, Xiao-Dong Bi, Peng Tan, Qin-Cheng Zheng, Hai-Tian Liu, Yi Xie, Xiao-Chuan Zou, Yang Yu, and Zhi-Hua Zhou. Beimingwu: A learnware dock system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5773–5782, 2024b.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. AutoML-Agent: A multi-agent LLM framework for full-pipeline AutoML. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 60099–60146, 2025.

Xi-Zhu Wu, Wenkai Xu, Song Liu, and Zhi-Hua Zhou. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):699–710, 2023.

Jiahuan Yan, Bo Zheng, Hongxia Xu, Yiheng Zhu, Danny Z. Chen, Jimeng Sun, Jian Wu, and Jintai Chen. Making pre-trained language models great on tabular prediction. In *The 12th International Conference on Learning Representations*, 2024.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. In *Advances in Neural Information Processing Systems 38*, pages 135605–135637, 2025.

Yuwei Yin, Yazheng Yang, Jian Yang, and Qi Liu. FinPT: Financial risk prediction with profile tuning on pretrained foundation models. arXiv:2308.00065, 2023.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. arXiv:2509.03505, 2025.

Zhi-Hua Zhou. Learnware: On the future of machine learning. *Frontiers of Computer Science*, 10 (4):589–590, 2016.

Zhi-Hua Zhou. Learnware and AI model management system. arXiv:2609.11656, 2026.

Zhi-Hua Zhou and Zhi-Hao Tan. Learnware: Small models do big. *Science China Information Sciences*, 67(1):112102, 2024.

A Related Work

A.1 Tabular Prediction with Language and Foundation Models

Tabular questions may ask for an unobserved outcome rather than a value already recorded in a table, a distinction examined by TopBench in text-based and agentic workflows (Ji et al., 2026). One response is to make the language model the predictor. TabLLM serializes features and values for zero-shot classification and fine-tunes with labeled examples in the few-shot setting (Hegselmann et al., 2023); TabuLa-8B trains on a large collection of tabular prediction tasks to transfer to unseen tables without target-task fine-tuning (Gardner et al., 2024). LLM Processes elicit numerical predictive distributions from observations and natural-language prior information (Requeima et al., 2024), while JoLT extends probabilistic LLM prediction to mixed-type tabular targets and missing inputs (Shysheya et al., 2025). These methods obtain predictions from the language model and its supplied context, establishing a route distinct from reusing separately trained tabular predictors.

Language models can also help construct tabular predictors. TP-BERTa connects numerical values and feature names through relative-magnitude tokenization and intra-feature attention (Yan et al., 2024); TabSTAR adapts a pretrained text encoder to learn target-aware representations for tables with textual fields (Arazi et al., 2025). At the workflow level, AutoML-Agent coordinates data retrieval, model development, and deployment from a task description (Trirat et al., 2025). These approaches improve representation transfer or automate model construction from available task data. Another route conditions a pretrained model on task examples at inference time.

Tabular foundation models provide such conditional inference from task-specific contexts. TabPFN learns a prior over synthetic tasks (Hollmann et al., 2025), and LimiX models variables and missingness jointly for conditional prediction and imputation (Zhang et al., 2025). TabICL scales in-context classification to larger tables (Qu et al., 2025), while TabDPT combines retrieval-based in-context learning with self-supervised pretraining on real tables for classification and regression (Ma et al., 2025). These models are closest to the conditional inference needed for our completion step, but their task contexts do not organize the capabilities and input requirements of an independently trained model library. We use TabPFN to infer missing inputs from aligned local specifications, then identify and combine the corresponding learnwares and assess whether their joint context supports the query. We next review specification-based model reuse underlying our approach.

A.2 Learnware

Learnware research develops ways to make independently trained models discoverable and reusable through specifications of their capabilities (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026). These specifications are generated with the help of machine learning and provide information for model identification without requiring access to the original training data of the developers. A learnware dock system (LDS) maintains the submitted models and their specifications; Beimingwu implements this infrastructure for identification and reuse (Tan et al., 2024b). In our setting, this infrastructure supplies both the predictors and the local statistical information used to infer their missing inputs.

RKME specifications represent local data distributions through compact weighted support sets (Wu et al., 2023). PAVE instead represents capabilities through parameter updates obtained by adapting a shared reference model to predictions from the submitted model, extending specification-based identification to high-dimensional inputs (Shi et al., 2026). Liu et al. (2025) use class-wise specifications to identify learnware combinations across different label spaces. They use labeled user examples to specify tasks and adapt the identified models. Lei et al. (2024; 2026a) analyze privacy protection and bound inference risks for RKME specifications, while Lei et al. (2026b) establish privacy guarantees for PAVE specifications under structural assumptions. Tan et al. (2026b) learn feature and label mappings from a small labeled user dataset to repurpose models across semantically unrelated tasks. We use RKME specifications to supply statistical context for model reuse.

For heterogeneous feature spaces, Tan et al. (2023) use RKME support sets to learn a unified subspace with mappings and reconstruction functions for predefined feature blocks, without requiring original developer data or auxiliary data containing all features together; users can specify tasks through unlabeled data. Tan et al. (2024a) further exploit model-generated labels to improve the shared representation and use labeled user examples for conditional matching. Tan et al. (2026a) extend identification and assembly to joint feature and label heterogeneity. These approaches require

a common representation to support both cross-space matching and reconstruction into the original feature spaces. In open-world reuse, these mappings must accommodate varying observed feature combinations, while queries may fall outside the support of the library. Matching and reconstruction alone do not resolve whether a prediction should be returned. Once the system has established target and feature correspondences, we use overlapping specifications to support missing-input completion, model identification, aggregation, and query support assessment.

B RKME BACKGROUND

This appendix gives the kernel formulation underlying the specification objective in Section 4.1. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive-definite kernel on a feature space $\mathcal{X}$, with reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$. For a distribution P over $\mathcal{X}$, its kernel mean embedding (KME) (Smola et al., 2007) is

$$\mu_P = \mathbb{E}_{x\sim P}[k(x, \cdot)] \in \mathcal{H}_k. \tag{9}$$

Given independent samples $\{x_j\}_{j=1}^n$ from P, the empirical embedding is $\widehat{\mu}_P = \frac{1}{n}\sum_{j=1}^n k(x_j, \cdot)$ (Smola et al., 2007). The RKHS distance $\|\mu_P - \mu_Q\|_{\mathcal{H}_k}$ is the maximum mean discrepancy between P and Q (Gretton et al., 2012).

The number of terms in an empirical embedding grows with the training set. A reduced-set construction instead searches for a compact collection of support locations and coefficients that approximates this embedding. For training inputs $D^X = \{x_j\}_{j=1}^n \subseteq \mathcal{X}$, the resulting RKME specification is

$$\sigma = \sum_{r=1}^m \beta_r k(z_r, \cdot), \qquad z_r \in \mathcal{X}, \tag{10}$$

where $Z = \{z_r\}_{r=1}^m$ is the reduced set and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_m)^\top$ contains its coefficients. The support points and coefficients are obtained by solving

$$\min_{Z,\boldsymbol{\beta}} \left\| \frac{1}{n}\sum_{j=1}^n k(x_j, \cdot) - \sum_{r=1}^m \beta_r k(z_r, \cdot) \right\|_{\mathcal{H}_k}^2. \tag{11}$$

The optimization learns $m \ll n$ support locations rather than retaining training records. Equation (11) defines the feature-space construction underlying RKME. Our objective in Eq. (3) applies this approximation in a task-conditioned representation with nonnegative coefficients. For a characteristic kernel, equality of population embeddings identifies equality of distributions (Gretton et al., 2012); a finite reduced-set approximation does not imply exact distribution recovery. Moreover, embeddings on different local feature spaces describe different marginals. Comparing or combining them requires an explicit correspondence between coordinates, as supplied by the alignment and sparse context in Section 4.2. Appendix A.2 discusses privacy results and their scope.

C ADDITIONAL METHOD DETAILS

C.1 SPECIFICATION GENERATION FOR CONDITIONAL INFERENCE

To construct specifications for conditional feature inference, we approximate the training distribution of each developer in the fixed TabPFN representation defined in Section 4.1. Here we describe the alternating optimization used to construct these specifications. Following Ye et al. (2025), we extract the representation of each training instance with its own label excluded from the context, so that the embedding procedure more closely matches that for an unlabeled query. A synthesized support point is embedded with its developer-model prediction $\widetilde{y}_{ir} = f_i(z_{ir})$. These labels condition developer-side construction; they are not target responses for deployment-time feature completion.

Write $h_{ij} = \phi_{\mathrm{T}}^{(i)}(x_{ij})$ and $g_{ir} = \phi_{\mathrm{T}}^{(i)}(z_{ir})$. For fixed support coordinates, define the Gram matrix and empirical cross-kernel vector as

$$[K_i]_{rt} = k(g_{ir}, g_{it}), \qquad [b_i]_r = \frac{1}{n_i}\sum_{j=1}^{n_i} k(h_{ij}, g_{ir}).$$

Expanding Eq. (3) gives $\boldsymbol{\beta}_i^\top K_i \boldsymbol{\beta}_i - 2b_i^\top \boldsymbol{\beta}_i + c_i$, where c_i depends only on the embedded training data. The weight step is therefore a nonnegative quadratic program. With the weights fixed, a gradient-based step updates the support coordinates (Wu et al., 2023). The pretrained predictor parameters remain fixed throughout. The generation objective does not impose $\sum_r \beta_{ir} = 1$; normalization is performed for the retrieval comparison below. The stored coordinates remain in the original local feature space, so they can be aligned into the context defined in Eq. (4).

C.2 Query-Specific RKME Distance

The following distance implements the query-specific comparison in Section 4.3. We represent $\widehat{x}$ as a single support point with unit weight. For comparison in $\mathcal{X}_{C_{\text{joint}}(x)}$, let $z_{ir}^{(x)}$ equal z_{ir} on C_i and take the completed query values $\widehat{x}$ on $C_{\text{joint}}(x) \setminus C_i$. Using the normalized weights $w_{ir} = \beta_{ir} / \sum_{t=1}^{m_i} \beta_{it}$, we compute

$$d_i(x) = \left\| \sum_{r=1}^{m_i} w_{ir} k_{\text{ret}}(z_{ir}^{(x)}, \cdot) - k_{\text{ret}}(\widehat{x}, \cdot) \right\|_{\mathcal{H}_{k_{\text{ret}}}}^2, \tag{12}$$

where $k_{\text{ret}}(a, b) = \exp(-\gamma_{\text{ret}} \|a - b\|_2^2)$ with $\gamma_{\text{ret}} > 0$. The identical coordinates outside C_i contribute no additional distance: the support points of each candidate are compared in feature coordinates with an ordinary kernel. Candidates are ranked separately for each query to retrieve up to R learnwares. Weight normalization assumes positive total support weight.

C.3 Query-Time Procedure

Algorithm 1 starts from the query and relevant learnwares after the system has established target and feature correspondences. The overlap candidates provide the context for both completion and support assessment, before MMD retrieval or uncertainty filtering.

Algorithm 1 Specification-Guided Model Reuse for One Query

Input: Query x and relevant learnwares $\mathcal{L}$ after target and feature matching; candidate and retrieval sizes C, R; an uncertainty-filtering rule; feature orders Π; support-score threshold τ
Output: Prediction $\widehat{y}_{\text{out}}(x)$ or abstention $\perp$

1: Form up to C overlap candidates $\mathcal{P}(x)$ (Section 4.2); **if** empty, **return** $\perp$
2: Construct $X_{\text{ctx}}(x)$ by Eq. (4)
3: Complete x as $\widehat{x}$ using Eq. (5); store feature distributions
4: Retrieve up to R candidates by query-specific MMD (Section 4.3)
5: Filter out retrieved models with high completion uncertainty in Eq. (6)
6: Compute $\widehat{y}_{\text{ens}}(x)$: mean retained outputs for regression, or $\arg\max$ of mean class probabilities
7: Compute the contextual support score $s(x)$ for the completed query using Eq. (8)
8: **return** the ensemble prediction if $s(x) \geq \tau$; otherwise return abstention $\perp$

D Experimental Protocol

Section 5.1 outlines the benchmark and methods. Here we specify the data splits, library and query construction, evaluation metrics, and LLM inputs used in that setup. Table 5 lists shared settings, and Table 6 identifies the factor varied in each component study.

D.1 Benchmark Tasks and Data Splits

BeyondArena. We use 30 tasks with semantic feature names: 16 classification and 14 regression tasks. For each task, we use the outer training–test split from the first fold of the first cross-validation repetition provided by BeyondArena (Purucker et al., 2026). This split is fixed across all scenario seeds. If a separate validation partition is unavailable, we reserve 15% of the outer training partition for model selection. The remaining 85% supplies developer training samples, and the outer test partition supplies user queries.

FinBench. We preserve the released training, validation, and test partitions for the 8 binary classification tasks `cc1`, `cc2`, `cc3`, `cd1`, `cf1`, `ld1`, `ld2`, and `ld3` (Yin et al., 2023). These cover customer churn, credit-card default and fraud, and loan default. Developer row samples are stratified; test queries preserve the original label distribution.

MIMIC-III. We retain the patient-level training, validation, and test splits of the clinical benchmark (Johnson et al., 2016; Harutyunyan et al., 2019). The tasks are in-hospital mortality using the first 48 hours (IHM), decompensation within the next 24 hours (DECOMP), and remaining length of stay after 24 hours (LOS). IHM and DECOMP are binary classification tasks; LOS is regression. Time-series observations are converted to fixed-window tabular features without using test labels. All features derived from the same clinical variable form one feature group. Query masking retains or hides every feature in a group together, as detailed in Appendix D.3.

D.2 Model-Library and Specification Construction

Each task and seed defines a separate model library with a common target and feature schema. Each developer trains a LightGBM predictor (Ke et al., 2017) on a local row subset and a feature subspace drawn from the training partition. Row and feature subsets may overlap across developers. General and Financial developers observe 70% of the features, while clinical developers observe 50% of the clinical feature groups. We draw 60% of each feature budget from a training-only importance-ranked pool and the remainder uniformly.

Classification models use stratified row sampling and balanced class weights. Eligible models have validation ROC-AUC more than 0.05 above chance, or positive validation R^2 for regression. Among eligible models, weights $1.0, 1.0, 0.25$ balance validation quality, feature coverage, and feature-subspace diversity. The resulting libraries contain 20 General or Financial models and 16 clinical models; local row limits are given in Table 5.

Developers submit models, feature metadata, and statistical specifications. Each specification summarizes at most 2,000 local rows using 80 support points for General and Financial tasks or 100 for clinical tasks. Predictor pseudo labels are used during TabPFN-aware specification generation, whereas query-time feature completion consumes aligned support coordinates without target pseudo labels (Section 4.1). The original developer training rows remain unavailable to the user.

Table 5: Main-experiment settings and hyperparameters.

Parameter	BeyondArena	FinBench	MIMIC-III
Random seeds		42, 43, 44	
Query-row seed		42	
Models per library	20	20	16
Maximum rows per developer	1,000	5,000	10,000
Developer-visible feature fraction	0.70	0.70	0.50 of clinical groups
Importance-biased fraction	0.60	0.60	0.60
Developer predictor		LightGBM	
Classification class weight		balanced	
Quality requirement		ROC-AUC $>$ chance $+0.05$, or $R^2 > 0$	
Quality / coverage / diversity weights		1.0 / 1.0 / 0.25	
Maximum specification rows	2,000	2,000	2,000
RKME support points	80	80	100
Missingness mechanism	MCAR	MCAR	Group-MCAR
User-visible feature fraction	0.70	0.70	0.50 of clinical groups
Test queries per task and seed	200	200	200
Context candidates	12	12	16
Retrieved models	8	8	8
Binary classification threshold		fixed 0.5	
Query class balancing		none	
TabPFN checkpoint		v2.5	
Direct/completion LLM		Qwen3.8-flash, non-thinking, temperature 0	
Critique/refinement LLM		Qwen3.8-flash / DeepSeek-V4-flash, matching the Direct LLM backbone	
Refinement rounds		1 (initial Direct LLM prediction reused)	
Alternative LLM		DeepSeek-V4-flash, non-thinking	

Table 6: Ablation and robustness settings.

Study	Varied factor
Specification context (Q1)	completion context
Specification generation	generation method
Feature completion	completion mechanism
Feature availability	observed-feature fraction
Uncertainty-aware selection	selected-model count
Specification-based abstention	score ranking / coverage

D.3 User Queries and Feature Availability

For each task, we sample 200 held-out test rows without replacement and keep their identities fixed across seeds. Seeds 42, 43, and 44 generate different model libraries and query masks. All methods receive identical query rows and masks within each task and seed. The repetitions therefore measure variation in library construction and feature availability.

General and Financial queries reveal 70% of feature columns sampled without replacement, independently of their values. Clinical queries reveal 50% of clinical feature groups under group-wise missing completely at random (Rubin, 1976) (Group-MCAR). A selected group makes all features derived from that clinical variable available; all features in the remaining groups are masked. This represents queries in which particular clinical variables have not been measured.

Classification queries follow the natural test label distribution. LOS queries are proportionally stratified across 10 target quantiles to cover the response range. Hidden feature values and labels are withheld from completion, retrieval, and prediction. Benchmark metadata supplies feature correspondence and compatible targets, so the experiments evaluate reuse after schema alignment.

D.4 Compared Methods

Direct LLM predicts the target from visible feature–value pairs. *LLM-Refine* reuses that prediction and applies one critique–refine round using the observed query information (Madaan et al., 2023). *LLM-Completed Models* retrieves models by Jaccard feature overlap, asks the LLM to fill their required missing inputs, and averages model class probabilities or regression outputs. These baselines receive no specifications. Appendix D.6 specifies their inputs, outputs, and prompt structure.

TabPFN completes missing inputs using aligned specification points. We compare feature-overlap retrieval with RKME-based identification (Appendix C.2). Mean averages all retrieved predictions, while selective aggregation filters models using completion uncertainty.

Qwen3.8-flash and DeepSeek-V4-flash are the main LLM backbones; LLM-Completed Models uses Qwen for feature completion. Main-table predictions precede abstention.

D.5 Metrics and Statistical Aggregation

The main experiment reports classification accuracy and task-level regression RMSE. General regression summaries use target-standardized RMSE (NRMSE) and within-task RMSE ranks because target scales differ. Where shown, standard deviations are computed across three seeds, each defining a model library and query masks, rather than across individual queries. Within each task, all regression methods are evaluated in the same target representation; the reported values are not pooled across target scales.

Let G be the General regression tasks and $\mathcal{S}$ the evaluation seeds (Table 5). For method m, task $d \in G$, and seed $s \in \mathcal{S}$, we normalize RMSE by the standard deviation of the evaluated test targets:

$$\mathrm{NRMSE}_{m,d,s} = \frac{\mathrm{RMSE}_{m,d,s}}{\mathrm{sd}(y^{\mathrm{test}}_{d,s})}, \qquad \overline{\mathrm{NRMSE}}_{m,G} = \frac{1}{|G|}\sum_{d\in G}\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\mathrm{NRMSE}_{m,d,s}.$$

Tasks and seeds are weighted equally. The target standard deviation is computed from the evaluated test targets with population normalization ($\mathrm{ddof} = 0$). Task-level results are reported as the mean $\pm$ sample standard deviation across three seeds.

For Table 1, task scores are averaged equally within each seed, then summarized by their mean and sample standard deviation across seeds. General regression ranks compare the three completion methods within each task and seed before averaging over tasks. Table 3 follows this seed-wise procedure with four completion methods. Table 2 instead first averages the scores for each task over the 3 seeds, then ranks all six displayed methods by descending accuracy or ascending RMSE. Average ranks weight tasks equally and assign average ranks to ties. Comparisons use unrounded scores. Group mean accuracy also weights tasks equally; raw RMSE is reported only for the single clinical regression task rather than averaged across different General regression targets.

D.6 LLM Inputs and Required Outputs

The experimental unit is one zero-shot user query. Table 7 summarizes Direct LLM, which predicts the target, and LLM-Completed Models, which uses Qwen3.8-flash to complete inputs for library predictors. Neither receives statistical specifications in the main comparison.

Table 7: **Inputs and outputs of the LLM prediction and completion routes.** Neither route receives developer training rows, hidden query values, or query labels.

Information	Direct LLM	LLM-Completed Models
Dataset/domain description	Supplied	Supplied
Observed feature names/values	Supplied	Supplied
Feature descriptions/encodings	Supplied	Supplied
Task	Target prediction	Missing-feature prediction
Required missing-feature names	Not required	Supplied
Output	One class or target value	JSON feature–value mapping
Output constraints	Valid class/numeric format	Valid feature codes/values

LLM-Refine. LLM-Refine critiques and revises the Direct LLM prediction on the same visible query. For each backbone, the same LLM performs one critique stage followed by one refinement stage. Critique receives the visible query and initial prediction and returns a short string beginning with `KEEP` or `REVISE`. Refine receives these inputs plus the critique and returns a valid class identifier or finite regression value. If no concrete reason for revision is identified, the prompt instructs it to retain the initial prediction. Neither stage receives specifications, library-model outputs, developer training rows, hidden feature values, or true query labels.

Prompt structure. The templates below show the information supplied for each query. Classification prompts specify valid class identifiers, and regression prompts specify the target encoding and units. Query labels are never supplied.

Direct LLM: per-query template

You are an expert data scientist. Your task is `<task description>`.
Dataset/domain: `<dataset>`. Prediction target: `<target description>`.
Below are the feature values for one sample. Some features may be missing.
Features: `<observed feature names and values>`.
Based on the feature values above, predict the target for this sample. `<valid class identifier or numeric target format>`. Do NOT output any explanation, just the number.

LLM-Completed Models: per-query template

You are a data imputation expert. Given the known feature values of a sample, estimate the missing feature values.
The sample comes from `<dataset>`.
Known features: `<observed feature names and values>`.
Missing features to impute: `<required feature names>`. `<allowed categorical encodings>`.
Output ONLY a JSON object mapping each missing feature name to its estimated numeric value. Do NOT include any explanation.

Critique and refinement instructions. The critique instruction asks the LLM to inspect target interpretation, feature reading, class encoding, units, numerical scale, and whether missing information was treated as an observed fact. It returns a short string beginning with `KEEP` or `REVISE`; `KEEP` is valid when no concrete reason for change is identified. The refinement instruction uses only the visible query, initial prediction, and critique, retaining the initial prediction if the critique provides no concrete basis for a change.

E Supplementary Results for Q1–Q3

Following Q1–Q3 in Section 5, this section reports relationship controls, task-level prediction results, and aggregation and abstention analyses.

E.1 Q1: Local Specifications as Completion Context

This extends Q1 (Section 5.2) with relationship controls and task-level results for a fixed predictor.

E.1.1 Local Relationship Recovery and Bridge-Chain Controls

The main Q1 comparison asks whether specifications from other models improve prediction through the relationships they encode. The controls below separate those relationships from extra support rows and feature marginals: real-task shuffles break cross-feature associations while preserving marginals, and a synthetic bridge chain tests whether two local views can support indirect completion without jointly observing the source and target features.

Fixed-predictor comparison. Q1 isolates completion context by passing each method's completed inputs to the same predictor per query. We choose f_1 by developer-side validation quality

among high-overlap candidates, without using query labels, and share it across the three methods. *Mean-F* uses feature-wise means of the support points associated with f_1; *TabPFN-F-Spec* uses these points for conditional completion; *TabPFN-All-Specs* uses aligned support points from the candidate set. Table 1 reports downstream prediction, while Table 10 gives the task-level results.

Relation-destruction controls. We compare intact joint context with two interventions on auxiliary specifications, keeping the specification associated with the target predictor unchanged. *Marginal shuffle* independently permutes the finite values of each auxiliary feature column within its specification block. *Bridge shuffle* applies this operation only to features present in at least two specifications. Both preserve the empirical marginal of each column, missing-value positions, and support-row count, while disrupting relationships between columns. The target predictor, query rows, feature masks, and context budgets remain fixed across conditions.

For originally observed query values hidden by masking, we measure recovery of observed–missing feature relationships. Association MAE is the mean absolute difference between true and completed Spearman rank correlations (Spearman, 1904) over eligible feature pairs. Each pair requires at least 20 valid observations; comparisons use pairs eligible in all conditions. Conditional-mean NRMSE compares true and completed missing-feature means within up to 10 quantile bins of an observed feature. Bins require at least two observations, and RMSE is normalized by the standard deviation of the true values of the missing feature. Table 8, Panel A, summarizes the 41 tasks. Intact context lowers observed–missing association error on 38 tasks against either shuffle control; conditional-mean error is lower on 32 and 33 tasks against marginal and bridge shuffling, respectively. For DECOMP in seed 43, one anomalous query is excluded from all three conditions, leaving 199 paired queries for this control. Table 9 gives the task-wise association errors.

Controlled bridge chain. This synthetic control tests whether shared intermediate variables can connect local relationships when both views are available in an aligned context. We construct two local views from independent samples of $A \to B \to C$, where B contains eight noisy linear measurements of A, and C is a noisy linear combination of B. One view observes (A, B); the other observes C and 0, 1, 2, 4, or 8 coordinates of B. No context row jointly observes A and C. Each view contains 100 context rows, and each seed evaluates 200 queries exposing only A. TabPFN receives the two aligned views and completes C. The shuffled control independently permutes finite values within each feature column of both views, preserving their marginals, missing-value patterns, and support-row counts.

Panel B of Table 8 reports completion NRMSE, normalized by the query standard deviation of C, and Pearson correlation with C. With no shared bridge variable, intact and shuffled NRMSE are 1.053 and 1.054. With a shared bridge, intact context yields NRMSE 0.389–0.571, compared with 0.998–1.057 after shuffling. The gain shows that shared B variables allow the completion module to exploit relationships distributed across the two views, even though A and C are never observed together in a context row. Together with the real-task shuffles, this control supports the value of cross-feature relationships in aligned specification context for missing-input completion.

Table 8: **Relationship-destruction and bridge-chain controls.** Panel A counts tasks favoring intact context. Panel B reports completion NRMSE and Pearson correlation with C as the number of shared B features varies.

Panel A: Real-task relation controls				
Metric	**Marginal shuffle**		**Bridge shuffle**	
Observed–missing association MAE	38/41		38/41	
Conditional-mean NRMSE	32/41		33/41	
Panel B: Controlled bridge chain				
Bridge variables	**Intact NRMSE**	**Shuffled NRMSE**	**Intact ρ**	**Shuffled ρ**
0	1.053 $\pm$ 0.048	1.054 $\pm$ 0.029	-0.247 $\pm$ 0.231	0.037 $\pm$ 0.060
1	0.536 $\pm$ 0.405	1.014 $\pm$ 0.019	0.971 $\pm$ 0.010	-0.040 $\pm$ 0.263
2	0.389 $\pm$ 0.088	0.998 $\pm$ 0.018	0.934 $\pm$ 0.045	0.121 $\pm$ 0.166
4	0.510 $\pm$ 0.275	1.057 $\pm$ 0.020	0.972 $\pm$ 0.014	-0.108 $\pm$ 0.096
8	0.571 $\pm$ 0.334	1.042 $\pm$ 0.034	0.971 $\pm$ 0.016	-0.047 $\pm$ 0.056

Table 9: **Task-level relationship recovery across 41 tasks.** Observed–missing Spearman association MAE (lower is better).

Task	Marginal shuffle	Bridge shuffle	Intact
General classification (16 tasks)			
Airline	0.177 ± 0.004	0.181 ± 0.009	$\mathbf{0.122 \pm 0.005}$
Amazon Access	0.275 ± 0.015	0.272 ± 0.005	$\mathbf{0.136 \pm 0.016}$
Bank Churn	0.153 ± 0.012	$\mathbf{0.151 \pm 0.003}$	0.157 ± 0.003
Churn	0.148 ± 0.012	0.153 ± 0.010	$\mathbf{0.119 \pm 0.012}$
E-commerce	0.168 ± 0.008	0.180 ± 0.023	$\mathbf{0.159 \pm 0.007}$
Fitness Club	0.232 ± 0.073	0.229 ± 0.043	$\mathbf{0.185 \pm 0.051}$
HR Analytics	0.236 ± 0.023	0.243 ± 0.020	$\mathbf{0.166 \pm 0.016}$
Coupon	0.167 ± 0.009	0.171 ± 0.015	$\mathbf{0.144 \pm 0.005}$
Iranian Churn	0.285 ± 0.039	0.289 ± 0.025	$\mathbf{0.107 \pm 0.008}$
JM1	0.509 ± 0.040	0.459 ± 0.017	$\mathbf{0.083 \pm 0.012}$
Marketing	0.247 ± 0.021	0.248 ± 0.020	$\mathbf{0.121 \pm 0.016}$
Online Shoppers	0.204 ± 0.016	0.204 ± 0.010	$\mathbf{0.132 \pm 0.004}$
SDSS-17	0.200 ± 0.017	0.200 ± 0.020	$\mathbf{0.101 \pm 0.016}$
Seismic Bumps	0.234 ± 0.021	0.236 ± 0.016	$\mathbf{0.084 \pm 0.006}$
Student Dropout	0.171 ± 0.008	0.173 ± 0.007	$\mathbf{0.127 \pm 0.005}$
Website Phishing	0.180 ± 0.029	0.166 ± 0.011	$\mathbf{0.141 \pm 0.025}$
General regression (14 tasks)			
5G Energy	0.207 ± 0.030	0.212 ± 0.031	$\mathbf{0.164 \pm 0.021}$
Airfoil Noise	0.279 ± 0.084	0.256 ± 0.003	$\mathbf{0.218 \pm 0.076}$
Concrete	0.201 ± 0.023	0.191 ± 0.022	$\mathbf{0.142 \pm 0.010}$
Diamonds	0.242 ± 0.030	0.239 ± 0.024	$\mathbf{0.099 \pm 0.022}$
Fiat 500	0.337 ± 0.004	0.333 ± 0.035	$\mathbf{0.209 \pm 0.058}$
Food Delivery	0.179 ± 0.019	0.160 ± 0.001	$\mathbf{0.089 \pm 0.010}$
Houses	0.230 ± 0.017	0.229 ± 0.029	$\mathbf{0.106 \pm 0.049}$
ImmoScout	0.176 ± 0.009	0.172 ± 0.014	$\mathbf{0.132 \pm 0.006}$
Miami Housing	0.204 ± 0.013	0.207 ± 0.033	$\mathbf{0.076 \pm 0.003}$
Protein	0.345 ± 0.048	0.289 ± 0.041	$\mathbf{0.087 \pm 0.015}$
Superconductivity	0.215 ± 0.006	0.189 ± 0.013	$\mathbf{0.039 \pm 0.000}$
Video Game FPS	0.194 ± 0.004	0.197 ± 0.002	$\mathbf{0.067 \pm 0.004}$
Video Transcode	0.262 ± 0.009	0.254 ± 0.017	$\mathbf{0.086 \pm 0.006}$
Wine Quality	0.194 ± 0.017	0.189 ± 0.004	$\mathbf{0.113 \pm 0.005}$
Financial classification (8 tasks)			
CC1	0.165 ± 0.031	$\mathbf{0.164 \pm 0.029}$	0.171 ± 0.024
CC2	$\mathbf{0.147 \pm 0.011}$	0.159 ± 0.014	0.168 ± 0.009
CC3	0.178 ± 0.011	0.175 ± 0.017	$\mathbf{0.113 \pm 0.016}$
CD1	0.208 ± 0.040	0.213 ± 0.046	$\mathbf{0.179 \pm 0.032}$
CF1	0.289 ± 0.014	0.286 ± 0.018	$\mathbf{0.097 \pm 0.006}$
LD1	0.153 ± 0.014	0.153 ± 0.008	$\mathbf{0.145 \pm 0.013}$
LD2	0.217 ± 0.030	0.216 ± 0.034	$\mathbf{0.139 \pm 0.020}$
LD3	0.174 ± 0.009	0.180 ± 0.002	$\mathbf{0.105 \pm 0.006}$
MIMIC-III classification (2 tasks)			
IHM	0.172 ± 0.003	0.175 ± 0.009	$\mathbf{0.136 \pm 0.009}$
DECOMP	0.171 ± 0.003	0.175 ± 0.008	$\mathbf{0.148 \pm 0.010}$
MIMIC-III regression (1 task)			
LOS	0.183 ± 0.005	0.179 ± 0.006	$\mathbf{0.139 \pm 0.003}$

Across the 41 tasks, intact specifications yield lower observed–missing Spearman association error than either shuffle on 38 tasks. This pattern holds for all 14 General regression tasks and all three clinical tasks; on JM1, error rises from 0.083 with intact context to 0.509 and 0.459 under marginal and bridge shuffling. Bank Churn, CC1, and CC2 are the only reversals. These task-level results support the value of cross-feature relationships in the aligned context, with task-dependent effects.

E.1.2 SPECIFICATION-CONTEXT COMPARISON

The following table reports all 41 tasks underlying the prediction panel of Table 1. The downstream predictor is fixed within each query; each entry summarizes 3 seeds.

Table 10: **Task-level specification-context results.** Classification reports accuracy (%); regression reports RMSE. Each method uses the same predictor for a given query.

Task	Mean-F	TabPFN-F-Spec	TabPFN-All-Specs
General classification			
Amazon Employee Access	$\mathbf{93.000 \pm 1.000}$	90.833 ± 0.289	91.000 ± 0.500
Bank Customer Churn	$\mathbf{82.000 \pm 1.803}$	79.500 ± 3.279	82.000 ± 3.041
Churn	86.667 ± 2.021	$\mathbf{90.000 \pm 0.866}$	89.833 ± 1.528
Airline Satisfaction	79.500 ± 3.775	87.500 ± 1.323	$\mathbf{88.333 \pm 2.021}$
E-commerce Shipping	64.333 ± 3.055	64.500 ± 4.359	$\mathbf{65.000 \pm 2.784}$
Fitness Club	69.833 ± 5.508	$\mathbf{70.500 \pm 2.291}$	66.667 ± 3.253
HR Analytics	$\mathbf{71.167 \pm 1.756}$	69.000 ± 1.500	69.833 ± 1.258
In-Vehicle Coupon	$\mathbf{63.667 \pm 1.893}$	63.000 ± 1.500	63.167 ± 2.517
Iranian Churn	85.667 ± 4.252	$\mathbf{91.333 \pm 2.754}$	91.000 ± 2.179
JM1	70.667 ± 3.884	74.500 ± 1.323	$\mathbf{75.000 \pm 1.000}$
Marketing Campaign	73.500 ± 3.122	$\mathbf{81.167 \pm 4.537}$	79.333 ± 1.607
Online Shoppers	82.167 ± 4.010	$\mathbf{86.833 \pm 1.258}$	86.167 ± 1.528
Student Dropout	66.500 ± 3.969	75.333 ± 1.258	$\mathbf{76.833 \pm 0.764}$
SDSS-17	80.167 ± 1.258	83.500 ± 2.500	$\mathbf{87.000 \pm 1.500}$
Seismic Bumps	$\mathbf{89.000 \pm 2.179}$	87.667 ± 1.258	88.333 ± 0.764
Website Phishing	63.833 ± 4.072	68.000 ± 5.635	$\mathbf{70.000 \pm 3.041}$
General regression			
5G Energy	0.616 ± 0.050	0.630 ± 0.023	$\mathbf{0.595 \pm 0.013}$
Airfoil Noise	1.013 ± 0.066	0.944 ± 0.043	$\mathbf{0.941 \pm 0.031}$
Concrete Strength	0.731 ± 0.058	0.755 ± 0.019	$\mathbf{0.687 \pm 0.044}$
Diamonds	0.440 ± 0.006	0.208 ± 0.019	$\mathbf{0.208 \pm 0.003}$
Fiat 500	0.676 ± 0.041	0.643 ± 0.008	$\mathbf{0.638 \pm 0.013}$
Food Delivery Time	1.000 ± 0.060	$\mathbf{0.988 \pm 0.040}$	0.989 ± 0.042
Houses	0.942 ± 0.035	0.705 ± 0.011	$\mathbf{0.700 \pm 0.028}$
ImmoScout	0.805 ± 0.009	0.794 ± 0.075	$\mathbf{0.774 \pm 0.015}$
Miami Housing	0.548 ± 0.018	0.489 ± 0.022	$\mathbf{0.452 \pm 0.016}$
Protein Structure	0.971 ± 0.029	0.908 ± 0.034	$\mathbf{0.903 \pm 0.046}$
Superconductivity	0.710 ± 0.045	0.521 ± 0.029	$\mathbf{0.485 \pm 0.043}$
Video Game FPS	0.462 ± 0.013	$\mathbf{0.133 \pm 0.013}$	0.145 ± 0.053
Video Transcoding	0.749 ± 0.024	0.688 ± 0.092	$\mathbf{0.668 \pm 0.053}$
Wine Quality	0.869 ± 0.007	0.865 ± 0.015	$\mathbf{0.851 \pm 0.005}$
Financial classification			
CC1	61.667 ± 1.258	61.167 ± 3.253	$\mathbf{64.167 \pm 2.754}$
CC2	80.333 ± 0.764	78.833 ± 2.309	$\mathbf{80.667 \pm 4.163}$
CC3	$\mathbf{76.500 \pm 3.606}$	72.667 ± 1.041	73.333 ± 2.566
CD1	84.333 ± 0.289	$\mathbf{88.000 \pm 0.866}$	87.667 ± 1.893
CF1	$\mathbf{99.667 \pm 0.577}$	99.167 ± 0.289	99.333 ± 0.289
LD1	91.167 ± 0.764	91.333 ± 1.041	$\mathbf{91.500 \pm 1.500}$
LD2	84.000 ± 2.500	85.833 ± 1.528	$\mathbf{86.667 \pm 2.517}$
LD3	66.833 ± 2.754	68.833 ± 2.566	$\mathbf{72.500 \pm 1.000}$
MIMIC-III classification			
IHM	79.500 ± 4.272	$\mathbf{80.833 \pm 0.577}$	80.333 ± 3.253
DECOMP	98.167 ± 0.577	97.833 ± 0.764	$\mathbf{98.167 \pm 0.289}$
MIMIC-III regression			
LOS	0.868 ± 0.030	0.873 ± 0.028	$\mathbf{0.859 \pm 0.052}$

With f_1 fixed, *TabPFN-All-Specs* leads on 25 of 41 tasks, including 12 of 14 General regression and 5 of 8 Financial tasks. It trails *TabPFN-F-Spec* on Fitness Club (66.667% versus 70.500%) and Video Game FPS (RMSE 0.145 versus 0.133). Auxiliary context can help the same predictor, with task-dependent gains.

E.2 Q2: Zero-Shot Prediction

This extends Q2 (Section 5.3) with task-level LLM prediction and model-reuse comparisons.

E.2.1 Main Prediction Comparison

Table 11 reports the six methods in main Table 2 on all 41 tasks. The LLM baselines receive no specifications, and LLM-Completed Models uses model-library predictions.

Table 11: **Task-level zero-shot prediction across 41 tasks.** Classification reports accuracy (%); regression reports RMSE.

Task	Metric	Direct LLM		LLM-Refine		LLM-Completed Models	Ours
		Qwen	DeepSeek	Qwen	DeepSeek		
General classification (16 tasks)							
Airline	Acc. (%) ↑	68.17 ± 0.29	73.83 ± 1.89	68.83 ± 3.82	73.83 ± 1.89	76.83 ± 1.26	**89.33 ± 1.44**
Amazon Access	Acc. (%) ↑	6.00 ± 0.00	24.67 ± 7.01	6.00 ± 0.00	24.67 ± 7.01	94.00 ± 0.00	**94.17 ± 0.58**
Bank Churn	Acc. (%) ↑	66.83 ± 2.36	68.83 ± 1.26	65.67 ± 2.84	68.83 ± 1.26	83.17 ± 2.08	**85.17 ± 2.47**
Churn	Acc. (%) ↑	84.00 ± 1.50	81.83 ± 2.47	81.00 ± 3.12	81.83 ± 2.47	86.17 ± 2.75	**90.00 ± 0.87**
E-commerce	Acc. (%) ↑	39.17 ± 1.61	48.67 ± 2.36	40.00 ± 1.50	48.50 ± 2.29	67.50 ± 4.44	**68.00 ± 2.60**
Fitness Club	Acc. (%) ↑	52.33 ± 2.31	51.00 ± 5.07	52.17 ± 2.47	51.00 ± 5.07	**71.33 ± 2.08**	70.83 ± 2.02
HR Analytics	Acc. (%) ↑	59.50 ± 4.27	62.67 ± 2.02	63.17 ± 3.06	62.33 ± 1.89	64.83 ± 2.89	**72.33 ± 2.52**
Coupon	Acc. (%) ↑	50.83 ± 3.82	59.67 ± 1.89	51.83 ± 4.01	59.67 ± 1.89	62.33 ± 1.89	**64.00 ± 0.87**
Iranian Churn	Acc. (%) ↑	52.67 ± 4.25	80.17 ± 1.61	61.17 ± 1.89	80.33 ± 1.89	86.67 ± 4.07	**93.50 ± 2.50**
JM1	Acc. (%) ↑	62.83 ± 2.36	61.83 ± 2.52	63.00 ± 2.18	61.83 ± 3.01	77.17 ± 2.47	**80.83 ± 1.44**
Marketing	Acc. (%) ↑	81.17 ± 0.58	77.50 ± 3.00	78.67 ± 0.29	77.67 ± 2.75	82.00 ± 1.50	**82.17 ± 0.76**
Online Shoppers	Acc. (%) ↑	83.83 ± 1.04	86.17 ± 1.53	79.83 ± 0.29	86.17 ± 1.53	**90.00 ± 0.50**	89.67 ± 0.58
SDSS-17	Acc. (%) ↑	50.17 ± 2.02	47.83 ± 7.29	59.33 ± 4.65	48.00 ± 7.00	77.50 ± 7.21	**86.67 ± 1.26**
Seismic Bumps	Acc. (%) ↑	68.17 ± 3.06	80.50 ± 1.80	68.50 ± 2.18	80.33 ± 1.61	**91.33 ± 1.15**	90.83 ± 1.04
Student Dropout	Acc. (%) ↑	60.67 ± 1.44	67.00 ± 1.73	59.17 ± 2.02	67.67 ± 2.89	72.67 ± 1.26	**76.50 ± 0.87**
Website Phishing	Acc. (%) ↑	33.50 ± 1.00	29.33 ± 2.57	31.50 ± 1.50	29.17 ± 2.52	66.33 ± 4.91	**71.17 ± 3.40**
General regression (14 tasks)							
5G Energy	RMSE ↓	0.989 ± 0.003	0.964 ± 0.017	0.996 ± 0.024	0.976 ± 0.029	0.750 ± 0.047	**0.605 ± 0.036**
Airfoil Noise	RMSE ↓	1.063 ± 0.010	1.157 ± 0.019	1.093 ± 0.038	1.164 ± 0.021	1.107 ± 0.051	**0.924 ± 0.082**
Concrete	RMSE ↓	0.955 ± 0.015	0.853 ± 0.047	0.991 ± 0.145	0.853 ± 0.047	1.064 ± 0.081	**0.736 ± 0.025**
Diamonds	RMSE ↓	1.142 ± 0.272	1.884 ± 0.527	3.439 ± 0.296	2.419 ± 0.159	0.364 ± 0.029	**0.184 ± 0.008**
Fiat 500	RMSE ↓	1.050 ± 0.006	0.940 ± 0.050	1.027 ± 0.046	0.927 ± 0.041	0.789 ± 0.044	**0.632 ± 0.028**
Food Delivery	RMSE ↓	1.066 ± 0.037	1.040 ± 0.011	1.067 ± 0.045	1.042 ± 0.016	1.133 ± 0.018	**0.929 ± 0.010**
Houses	RMSE ↓	1.132 ± 0.163	0.886 ± 0.047	1.082 ± 0.119	0.882 ± 0.037	0.955 ± 0.069	**0.661 ± 0.032**
ImmoScout	RMSE ↓	1.382 ± 0.128	1.164 ± 0.042	1.363 ± 0.139	1.164 ± 0.042	0.949 ± 0.011	**0.725 ± 0.020**
Miami Housing	RMSE ↓	1.886 ± 1.610	0.904 ± 0.051	3.626 ± 1.138	1.014 ± 0.102	0.778 ± 0.069	**0.394 ± 0.010**
Protein	RMSE ↓	0.998 ± 0.000	1.020 ± 0.003	1.058 ± 0.036	1.021 ± 0.003	0.958 ± 0.022	**0.813 ± 0.045**
Superconductivity	RMSE ↓	1.088 ± 0.010	1.109 ± 0.048	1.080 ± 0.007	1.109 ± 0.048	0.554 ± 0.001	**0.427 ± 0.009**
Video Game FPS	RMSE ↓	1.212 ± 0.083	0.887 ± 0.037	1.212 ± 0.063	0.880 ± 0.048	0.885 ± 0.089	**0.116 ± 0.027**
Video Transcode	RMSE ↓	1.075 ± 0.040	0.984 ± 0.018	1.079 ± 0.053	0.981 ± 0.020	0.694 ± 0.029	**0.664 ± 0.038**
Wine Quality	RMSE ↓	1.160 ± 0.279	0.985 ± 0.022	1.003 ± 0.013	0.984 ± 0.021	0.868 ± 0.014	**0.819 ± 0.017**
Financial classification (8 tasks)							
CC1	Acc. (%) ↑	57.83 ± 1.61	58.83 ± 4.54	61.50 ± 2.18	58.83 ± 4.54	**69.33 ± 5.97**	63.00 ± 6.76
CC2	Acc. (%) ↑	59.67 ± 4.25	66.67 ± 0.76	63.17 ± 2.93	66.50 ± 0.87	77.00 ± 0.50	**81.67 ± 3.01**
CC3	Acc. (%) ↑	63.50 ± 3.28	64.67 ± 2.52	67.17 ± 1.53	64.67 ± 2.52	71.17 ± 0.76	**75.50 ± 2.18**
CD1	Acc. (%) ↑	87.00 ± 4.36	62.17 ± 3.79	79.83 ± 4.31	62.17 ± 3.79	**90.50 ± 1.80**	90.00 ± 2.18
CF1	Acc. (%) ↑	58.17 ± 4.07	67.67 ± 3.01	65.00 ± 4.44	67.67 ± 3.01	99.33 ± 0.76	**99.67 ± 0.29**
LD1	Acc. (%) ↑	78.67 ± 0.29	71.83 ± 2.84	77.50 ± 1.80	71.83 ± 2.84	79.17 ± 1.26	**93.67 ± 0.29**
LD2	Acc. (%) ↑	75.17 ± 0.76	75.50 ± 1.73	78.17 ± 2.08	75.67 ± 1.89	74.67 ± 3.51	**87.50 ± 1.32**
LD3	Acc. (%) ↑	64.17 ± 3.75	60.33 ± 1.26	69.00 ± 4.50	61.33 ± 1.76	**77.00 ± 0.00**	74.17 ± 2.89
MIMIC-III classification (2 tasks)							
IHM	Acc. (%) ↑	50.83 ± 2.89	61.00 ± 3.50	64.33 ± 3.79	62.17 ± 2.02	49.17 ± 6.53	**82.67 ± 5.48**
DECOMP	Acc. (%) ↑	72.00 ± 1.50	75.83 ± 1.04	77.00 ± 5.00	75.33 ± 4.75	93.50 ± 3.61	**98.50 ± 0.00**
MIMIC-III regression (1 task)							
LOS	RMSE ↓	1.058 ± 0.001	0.951 ± 0.043	1.038 ± 0.014	0.951 ± 0.044	0.889 ± 0.035	**0.849 ± 0.023**

Ours leads on 35 of 41 tasks, including all 14 General regression tasks and all three clinical tasks. LLM-Completed Models leads on the other six, all classification tasks; the largest gaps are CC1 (69.33% versus 63.00%) and LD3 (77.00% versus 74.17%). Direct and refined LLM predictions do not lead any task. The task-level comparison complements the group-level results while identifying classification cases where LLM-based completion remains competitive.

E.3 Q3: Aggregation and Abstention

This extends Q3 (Section 5.4) with aggregation, abstention, and query-level analyses.

E.3.1 Uncertainty-Aware Aggregation

This experiment expands the uncertainty-aware aggregation analysis in Section 5.4. We retrieve $R = 8$ models and vary the number selected for aggregation, $K \in \{1, 2, 3, 4, 6\}$. For each query, models are ranked by completion uncertainty; aggregation uses the K lowest-uncertainty models. Mean aggregation uses all 8 models. The queries, masks, and trained predictors are shared across settings. Main-text Table 4 reports the task-level comparison; the query-level examples below show how filtering changes individual predictions.

Query-level illustrations. Figure 4 shows the completion uncertainty and prediction error of each retrieved model for four queries, tracing how uncertainty-aware filtering changes the ensemble prediction.

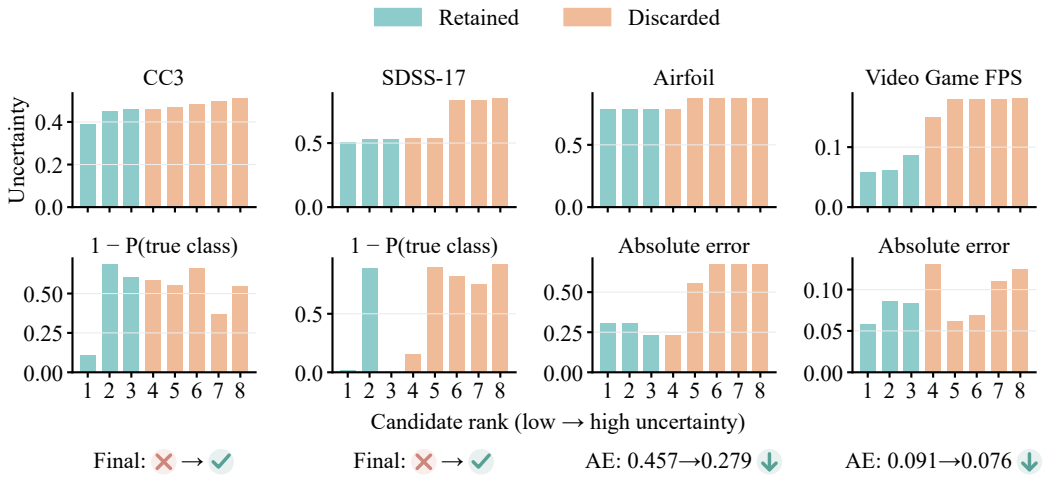

Figure 4: **Query-level effects of uncertainty-aware selection.** Candidates are ranked by increasing completion uncertainty. Lower panels show $1 - P(\text{true class})$ for classification and absolute error for regression. Footers show Mean → Selective correctness or final absolute error (AE).

Filtering corrects the CC3 and SDSS-17 predictions. It also reduces absolute error from 0.457 to 0.279 on Airfoil and from 0.091 to 0.076 on Video Game FPS.

E.3.2 Specification-Based Abstention

For task d and seed s, let q_i and ℓ_i denote the support score and loss of query i, respectively, where $\ell_i = \mathbf{1}[\hat{y}_i \neq y_i]$ for classification and $\ell_i = |\hat{y}_i - y_i|$ for regression. Let π order the n queries by decreasing support. At target coverage c,

$$k_c = \max\{1, \lceil cn \rceil\}, \qquad C_{ds}(c) = \frac{k_c}{n}, \qquad R_{ds}(c) = \frac{1}{k_c} \sum_{j=1}^{k_c} \ell_{\pi(j)}. \tag{13}$$

The system retains the k_c most supported queries and abstains on the remainder. We summarize risk–coverage behavior using AURC (Geifman et al., 2019). For each task and seed, risk is normalized

by the full-coverage risk:

$$r_{ds}(c) = \frac{R_{ds}(c)}{R_{ds}(1)}, \qquad \mathrm{nAURC}_{ds} = \frac{1}{n}\sum_{k=1}^{n} \frac{R_{ds}(k/n)}{R_{ds}(1)}, \qquad G_{ds} = 1 - \mathrm{nAURC}_{ds}\,. \tag{14}$$

Here, G_{ds} measures the reduction in normalized AURC relative to random query ranking, whose expected normalized risk is 1 and expected gain is 0. Positive gain indicates lower average normalized risk than random ranking. Normalized risk is undefined when full-coverage loss is zero; such task–seed runs are omitted from normalized summaries.

F Component Ablations and Robustness Analyses

These studies extend the prediction and component analyses in Sections 5.3 and 5.4. They examine specification generation, feature completion, and feature availability.

F.1 TabPFN-Aware Specification Generation

We compare two methods for generating the reduced specification. *Raw-RKME* constructs the specification in the original feature space (Wu et al., 2023); *TabPFN-aware RKME* optimizes the reduced set using TabPFN representations. Both use RKME retrieval and Mean aggregation with TabPFN completion. Developer splits, predictors, reduced-set size, queries, and masks are fixed. We report classification accuracy, within-task General regression RMSE ranks, and clinical regression RMSE.

Table 12: **Downstream prediction with raw-space and TabPFN-aware specifications.** RKME retrieval and Mean aggregation are fixed; General regression reports within-task ranks.

Task group	Metric	Raw-RKME	TabPFN-aware RKME
General classification (16)	Acc. (%) ↑	80.354 ± 1.303	$\mathbf{81.240 \pm 0.966}$
General regression (14)	RMSE rank ↓	$\mathbf{1.452 \pm 0.082}$	1.548 ± 0.082
Financial classification (8)	Acc. (%) ↑	81.750 ± 0.962	$\mathbf{82.875 \pm 1.657}$
MIMIC-III classification (2)	Acc. (%) ↑	90.083 ± 0.946	$\mathbf{90.583 \pm 2.742}$
MIMIC-III regression (1)	RMSE ↓	0.875 ± 0.023	$\mathbf{0.850 \pm 0.025}$

TabPFN-aware construction raises mean classification accuracy in all three domains: from 80.354% to 81.240% on General, 81.750% to 82.875% on Financial, and 90.083% to 90.583% on clinical tasks. Clinical regression RMSE also decreases from 0.875 to 0.850. These comparisons support generating specifications for the representation used by the completion model.

F.2 Feature-Completion Methods

We isolate missing-feature handling while fixing the model library, retrieval, aggregation, query rows, and masks. *Native Missing* passes unobserved inputs as missing values to LightGBM. *Mean* and *Random* use a feature-wise context mean or a draw from the context marginal, respectively. *TabPFN* conditions jointly on the observed query features and aligned specification context. This four-way comparison appears as Table 3 in Section 5.4.

Direct prediction versus learnware reuse. *TabPFN-ICL* pairs the joint support context with model-generated pseudo labels and predicts the target directly. Ours uses the context to complete missing inputs, then obtains target predictions from retrieved learnware predictors. Table 13 compares these end-to-end routes. Ours raises General classification accuracy from 77.365% to 81.573%, lowers General regression NRMSE from 0.703 to 0.616, and lowers clinical regression RMSE from 0.870 to 0.849. These results support combining specification-guided completion with trained predictors in these settings.

Table 13: **Direct TabPFN prediction and model reuse.** TabPFN-ICL predicts the target from pseudo labels; Ours completes inputs for library predictors.

Task group	Metric	TabPFN-ICL	Ours
General classification (16)	Acc. (%) ↑	77.365 ± 0.331	$\mathbf{81.573 \pm 0.578}$
General regression (14)	NRMSE ↓	0.703 ± 0.011	$\mathbf{0.616 \pm 0.008}$
Financial classification (8)	Acc. (%) ↑	$\mathbf{83.458 \pm 0.730}$	83.146 ± 1.814
MIMIC-III classification (2)	Acc. (%) ↑	89.833 ± 0.946	$\mathbf{90.583 \pm 2.742}$
MIMIC-III regression (1)	RMSE ↓	0.870 ± 0.010	$\mathbf{0.849 \pm 0.023}$

F.3 Sensitivity to Feature Availability

We fix RKME retrieval, TabPFN completion, and mean aggregation of all retrieved predictions (RKME-TabPFN Mean), then vary only the observed-feature fraction over $\{0.3, 0.5, 0.7, 0.9\}$. Clinical masking operates on variable groups. Table 14 reports the complete three-seed results. Figure 5 shows that increasing the observed-feature fraction from 0.3 to 0.9 raises Financial accuracy from 80.125% to 84.312%, raises clinical accuracy from 89.583% to 91.583%, and lowers clinical regression RMSE from 0.894 to 0.831.

Table 14: **RKME-TabPFN Mean under varying feature visibility.** General regression ranks compare the four visibility conditions within each task and seed.

Task group	Metric	0.3	0.5	0.7	0.9
General classification (16)	Acc. (%)	74.021 ± 0.614	78.458 ± 0.308	81.240 ± 0.966	$\mathbf{84.156 \pm 0.317}$
General regression (14)	RMSE rank	3.964 ± 0.000	3.036 ± 0.000	2.000 ± 0.000	$\mathbf{1.000 \pm 0.000}$
Financial classification (8)	Acc. (%)	80.125 ± 0.348	82.104 ± 1.041	82.875 ± 1.657	$\mathbf{84.312 \pm 0.380}$
MIMIC-III classification (2)	Acc. (%)	89.583 ± 3.744	90.583 ± 2.742	90.333 ± 0.946	$\mathbf{91.583 \pm 0.520}$
MIMIC-III regression (1)	RMSE	0.894 ± 0.003	0.850 ± 0.025	0.854 ± 0.014	$\mathbf{0.831 \pm 0.018}$

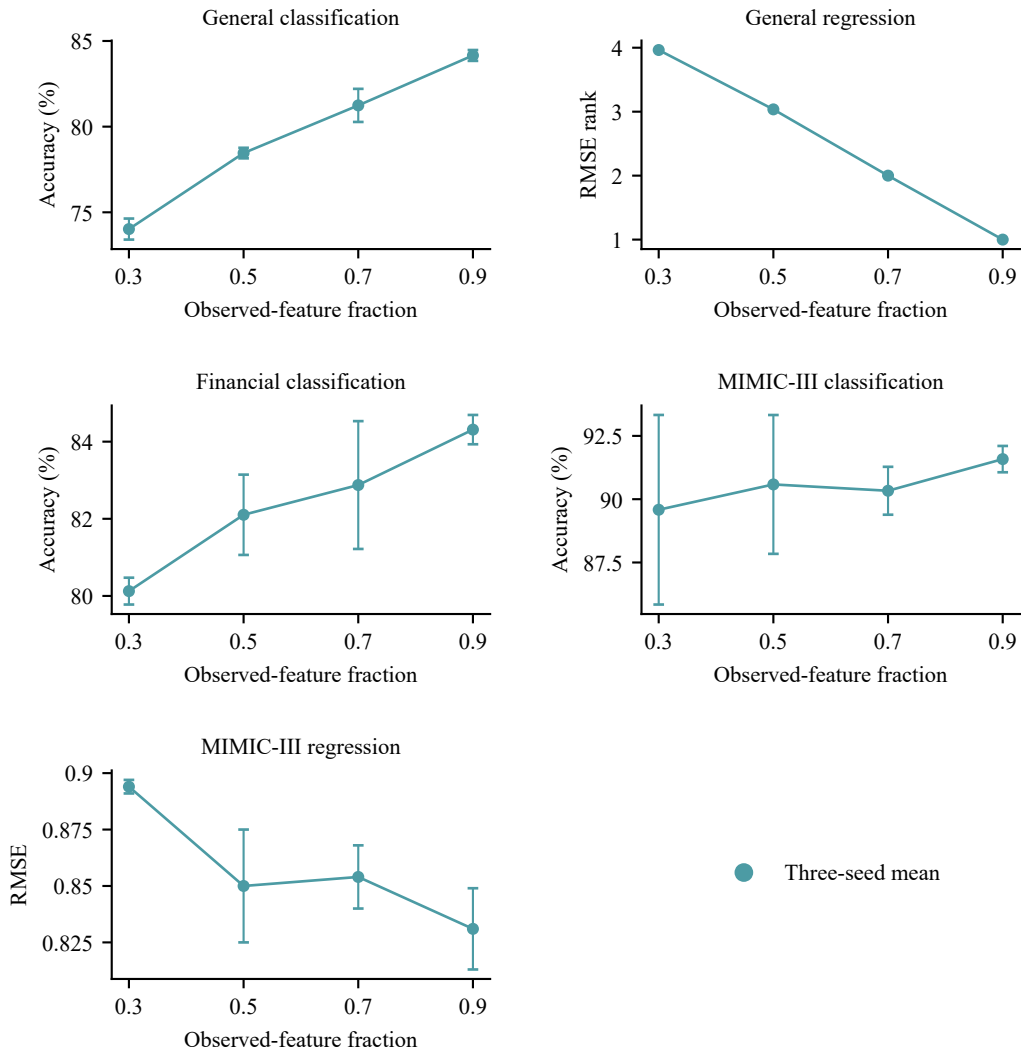

Figure 5: **RKME-TabPFN Mean under varying feature visibility.** General regression ranks compare the four fractions within each task and seed.