KumoRFM-2: Scaling Foundation Models for Relational Learning

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, Matthias Fey

`https://kumo.ai`
`pip install kumoai`

Abstract

We introduce *KumoRFM-2*, the next iteration of a pre-trained foundation model for relational data. KumoRFM-2 supports in-context learning as well as fine-tuning and is applicable to a wide range of predictive tasks. In contrast to tabular foundation models, KumoRFM-2 natively operates on relational data, processing one or more connected tables simultaneously without manual table flattening or target variable generation, all while preserving temporal consistency.

KumoRFM-2 leverages a large corpus of synthetic and real-world data to pre-train across four axes: the row and column dimensions at the individual table level, and the foreign key and cross-sample dimensions at the database level. In contrast to its predecessor, KumoRFM-2 injects task information as early as possible, enabling sharper selection of task-relevant columns and improved robustness to noisy data.

Through extensive experiments on 41 challenging benchmarks and analysis around expressivity and sensitivity, we demonstrate that KumoRFM-2 outperforms supervised and foundational approaches by up to 8%, while maintaining strong performance under extreme settings of cold start and noisy data. To our knowledge, this is the first time a few-shot foundation model has been shown to surpass supervised approaches on common benchmark tasks, with performance further improving upon fine-tuning. Finally, while KumoRFM-1 was limited to small-scale in-memory datasets, KumoRFM-2 scales to billion-scale relational datasets.

1 Introduction

Foundation models are fundamentally changing the landscape of predictive modeling on structured data, achieving state of the art accuracy (Qu et al., 2025; Hollmann et al., 2023; Spinaci et al., 2025; Garg et al., 2025; Zhang et al., 2025a; Fey et al., 2025; Qu et al., 2026). The key enabling factor for this paradigm shift is the rise of *In-Context Learning (ICL)* (Brown et al., 2020), which enables solving novel predictive tasks in a single feed-forward step, avoiding expensive task-specific training.

Traditional supervised machine learning methods require training on large datasets containing labeled data for a single specific task. In contrast, ICL allows for *pre-training* a general model on large-scale synthetic and real data, which can adapt to novel, unseen tasks by showing it a few labeled examples — the *context* — at prediction time only. The consequences are dramatic: data requirements for solving unseen predictive tasks are substantially reduced. Similarly, time to deployment for novel tasks is reduced from several hours and days to seconds.

ICL was originally introduced in the context of *Large Language Models (LLMs)* Vaswani et al. (2017); Radford et al. (2019); Brown et al. (2020), where models are conditioned on a sequence of input/output examples at inference time. Despite their strong generalization capabilities, the language-based inductive bias of LLMs is not well aligned with predictive tasks over structured data

(*cf.* Sec. 4). Therefore, *Tabular Foundation Models (TFMs)* extended the principle, allowing for in-context learning on data populating a single table (Hollmann et al., 2023). They can be naively extended to the relational setting by flattening multiple tables via fixed-function, parameter-free encoders (Hayler et al., 2025; Eremeev et al., 2025; Xu et al., 2026; Wang et al., 2026).

However, these approaches remain limited for real-world applications: they rely on flattening relational data into single, row-wise representations. This either necessitates substantial feature engineering and target label generation (undermining their key advantage of training-free, ad-hoc predictions), or relies on generic, task-agnostic feature aggregation schemes that fail to exploit task-specific relational structure for downstream prediction. Enabling foundation models to operate directly on multi-table relational data would eliminate these limitations, allowing truly end-to-end, training-free predictions while fully leveraging the rich structure inherent in real-world datasets.

Here, we introduce *KumoRFM-2*, a foundation model for general data and tasks that live in relational databases, *e.g.*, predicting user churn, inventory demand, or detecting fraud. KumoRFM-2 is the next iteration of a "database-native" *Relational Foundation Model (RFM)* (Fey et al., 2025). We show that pre-training on relational data enables the model to effectively learn how to filter, correlate, and aggregate information across multiple tables. Compared to v1, KumoRFM-2 performs ICL over two separate attention stages, one for intra-table processing and one for aggregating information across tables. This design enables task-conditioned feature extraction at both the table and database level, leading to improved predictive performance.

In comprehensive experiments across 41 predictive tasks on real-world databases from RelBenchV1 (Robinson et al., 2024), RelBenchV2 (Gu et al., 2026), SALT (Klein et al., 2024) and 4DBInfer (Wang et al., 2024), we demonstrate that KumoRFM-2 outperforms both supervised models as well as few-shot foundation models across tabular and relational settings. Specifically, on RelBenchV1, KumoRFM-2 outperforms v1 by $\approx$10% and even surpasses its closest competitor, the task-specific supervised RelGNN (Chen et al., 2025), by $\approx$5% for both classification and regression tasks. Furthermore, on the SAP SALT enterprise benchmark, KumoRFM-2 achieves state-of-the-art performance, surpassing giant tabular AutoGluon ensembles (Erickson et al., 2020) by $\approx$8% and recent tabular foundation models (Qu et al., 2026) by $\approx$25%. Performance can be further improved by $\approx$16% through fine-tuning. This highlights the benefits of modeling relational structure directly rather than relying on flattened tabular representations. Finally, we analyze the robustness of KumoRFM-2 with respect to context size, subgraph size, feature and link scarcity, and feature noise. In general, we observe that KumoRFM-2 is able to preserve strong predictive capability even under extreme settings of cold start, as well as missing or incomplete data.

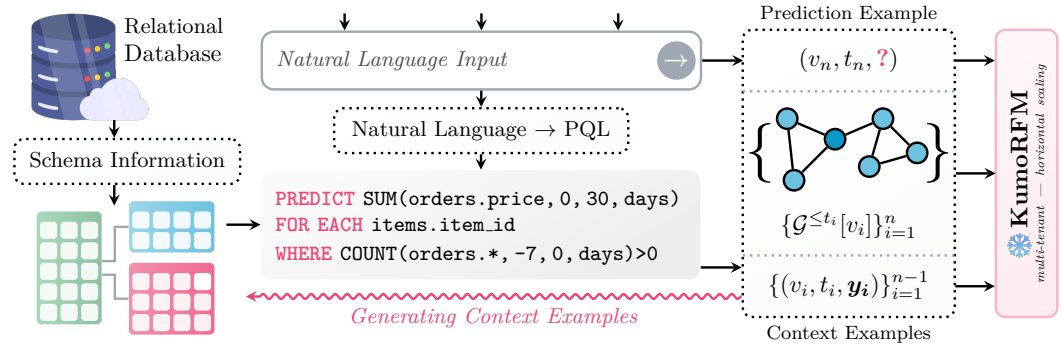
Overall, the results suggest that the relational domain is undergoing the same paradigm shift already witnessed in other domains, where few-shot foundation models are now able to rival carefully tuned, task-specific approaches. This, in turn, enables accurate on-demand predictions on relational data that can be seamlessly integrated into agentic workflows for real-time decision-making.

KumoRFM-2 is publicly available through our Python SDK. While previously limited to in-memory datasets, KumoRFM-2 supports direct connectivity to SQL databases and cloud data warehouses, enabling it to scale seamlessly to databases of arbitrary size. Specifically, data processing can either be pushed down to the underlying database or executed via an SSD-based graph engine that leverages memory-mapped I/O to enable high-throughput, low-latency access to databases with 500B+ rows.

2 Relational Foundation Models and KumoRFM

KumoRFM is a pre-trained relational foundation model (Fey et al., 2025), combining the concepts of *Relational Deep Learning* (Fey et al., 2024) and in-context learning (Brown et al., 2020):

Relational Deep Learning (RDL). RDL models (Fey et al., 2024) a relational database as a *temporal heterogeneous graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each record in any table corresponds to a node $v \in \mathcal{V}$, and the primary-foreign key links define the edges $(v, w) \in \mathcal{E}$. Every node is associated with a timestamp t, allowing us to define a time-consistent snapshot $\mathcal{G}^{\leq t}$ that contains only the records available up to that moment. We define a *training example* (v_i, t_i, y_i) for a specific node v_i at a specific point in time t_i, where y_i is the ground-truth target. A sequence of examples $\mathcal{T} = \{(v_i, t_i, y_i)\}_{i=1}^{n-1}$, called *task table*, implicitly defines a prediction task, used as training set for supervised learning.



(a) Database Integration (b) Prompting Interface (c) Context Generation

Figure 1: **Overview of the KumoRFM system. (a)** KumoRFM operates directly on a relational database, where the schema defines the underlying graph structure. **(b)** Users specify predictive tasks via a declarative Predictive Query Language or natural language. **(c)** KumoRFM constructs context examples (v_i, t_i, y_i) and corresponding input subgraphs $\mathcal{G}^{\leq t_i}[v_i]$, where targets y_i are derived according to the query definition. All data processing is pushed down to the underlying database for scalable execution, after which the constructed context is passed to the model for inference.

In single tabular learning, learning can be simply performed over rows in $\mathcal{T}$ (Hollmann et al., 2023; Erickson et al., 2020). The relational setting comes with additional challenges. Relevant input features are not given in the task table itself but live in multiple tables, connected to v_i via complex relational structures. Thus, for each v_i, task-relevant signals need to be extracted from related data sources, *i.e.* within local subgraphs $\mathcal{G}^{\leq t_i}[v_i]$ around v_i up to timestamp t_i. Recent work has made strong progress in processing such data through advances in model architectures (Dwivedi et al., 2026), graph rewiring techniques (Chen et al., 2025), and pre-training strategies (Ranjan et al., 2026).

Relational Foundation Models (RFMs). RFMs enable predictive modeling on any relational database without requiring task-specific training (Fey et al., 2025; Wang et al., 2026; Kothapalli et al., 2026; Xu et al., 2026; Ranjan et al., 2026). Specifically, RFMs perform few-shot predictions by treating training examples as *context examples* for in-context learning (Fey et al., 2025). That is, a single feed-forward step extracts signal from subgraphs within the context to inform predictions. More formally, the task table $\mathcal{T} = \{(v_i, t_i, y_i)\}_{i=1}^{n-1}$, containing the context examples, defines what information to use to predict the label $\hat{y}_n$ for a node v_n for a future point in time t_n:

$$\hat{y}_n = \text{RFM}_{\boldsymbol{\theta}}^{\text{❄}}(\mathcal{G}^{\leq t_n}[v_n], \{(\mathcal{G}^{\leq t_i}[v_i], y_i)\}_{i=1}^{n-1}).$$

Because the RFM parameters $\boldsymbol{\theta}$ are pre-trained and remain frozen during this process, the model must entirely rely on its ability to reason over the relational structure of input subgraphs $\mathcal{G}^{\leq t_i}[v_i]$ and outputs y_i to generalize to the specified task for the new example. This ability naturally extends the paradigm of tabular foundation models to the relational multi-table and temporal setting, building on recent advancements in synthetic data generation (Kothapalli et al., 2026; Wang et al., 2026; Hoppe et al., 2025; Hollmann et al., 2023; Zhang et al., 2025b; den Breejen et al., 2025; Qu et al., 2026), multi-scale processing (Bouadi et al., 2025), or semantic awareness (Spinaci et al., 2025).

The KumoRFM System. KumoRFM (*cf.* Fig. 1) is an end-to-end system built around a Relational Foundation Model for predictive tasks on structured data. Beyond the model itself, KumoRFM orchestrates a set of tightly coupled components to enable low-latency predictions and seamless integration into agentic workflows. In particular, KumoRFM manages the following components:

- **Database integration (Fig. 1(a)):** KumoRFM integrates natively with relational databases, allowing it to construct graphs and feature spaces directly from database metadata or a semantic model. This integration is a one-time setup that relies primarily on schema-level information, making it fast and lightweight in practice. Support for views and derived columns enables flexible graph rewiring without requiring costly data materialization or preprocessing pipelines. Crucially, this integration allows KumoRFM to scale directly with the underlying database, regardless of its size.

- **Prompting interface (Fig. 1(b)):** Interaction with KumoRFM is facilitated through the high-level declarative *Predictive Query Language (PQL)* (Kocijan et al., 2026). Queries are expressed as compositions of aggregations, filters, and binary operations, providing a concise and flexible abstraction for specifying predictive tasks. This interface supports a wide range of use cases, from basic tasks such as value imputation to more complex temporal prediction problems (*e.g.*, user churn and item demand forecasting). It also serves as a structured intermediate representation for natural language interfaces, avoiding the need for agents to construct model inputs directly. As such, PQL turns predictive modeling into a composable primitive for agents, allowing queries to be stacked and combined with SQL operations to express complex decision-making pipelines.

- **Automatic context generation (Fig. 1(c)):** Given a predictive query and graph metadata, the full context can be constructed ad-hoc by pushing computation to the underlying data backend. In particular, ground-truth labels $\{(v_i, t_i, y_i)\}_{i=1}^{n-1}$ can be obtained in two ways: either directly from existing database values (*e.g.*, in value imputation settings), or by computing them at earlier timestamps according to the query definition. Similarly, input subgraphs $\{\mathcal{G}^{\leq t_i}[v_i]\}_{i=1}^{n}$ are constructed by traversing the database schema along valid relational paths.

  As such, KumoRFM supports both static as well as temporal predictive tasks. For static tasks (*e.g.*, "predict user age"), the target for context examples can be generated by sampling users whose age is already known. For temporal tasks (*e.g.*, "forecast sales next quarter" or "predict transaction fraud probability at time t_i"), the context is effectively generated by "replaying" historical states of the database, sampling entity-centric subgraphs up to anchor time t_i and computing the corresponding target y_i based on events occurring after t_i (*cf.* Fig. 2). In practice, only a small number of context examples (*e.g.*, up to $10k$) is sufficient to provide strong signal for in-context learning while preserving low latency.

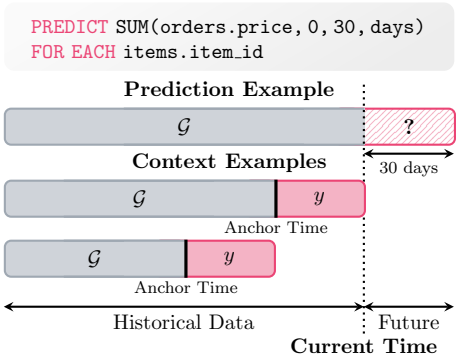


Figure 2: **Automatic context generation.** The query (Kocijan et al., 2026) defines how context input/output pairs can be automatically constructed from historical database states while preventing data leakage.

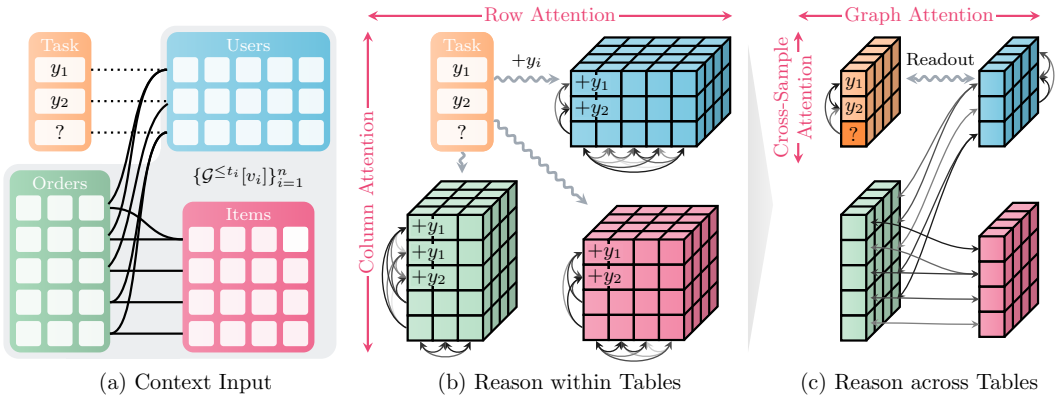
- **Scalability:** KumoRFM-2 is designed to support in-context learning natively over databases, either **(1)** in-memory, **(2)** via database and warehouse connectors, or **(3)** through a custom graph engine, optimized for fast ingestion and neighbor retrieval. The in-memory mode is ideal for exploration and small-scale experimentation, while database connectors and graph engine are designed for production workloads at scale. In the database connector mode, no intermediate graph representation is constructed. Instead, both context retrieval and subgraph construction are *pushed down* to the database via recursive SQL execution along metapaths. The graph engine builds an intermediate memory-mapped data structure, enabling KumoRFM-2 to scale to 500B+ rows with high throughput (up to 5GB/sec bandwidth, 20M lookups/sec) and low-latency requirements.

- **Stateless, multi-tenant model serving:** Kumo's relational foundation model can be deployed as a stateless inference service, decoupled from any specific database, task, or tenant. All required context is provided at request time, enabling strict isolation across workloads and eliminating the need for persistent state or per-tenant model instances. This design simplifies scaling, supports efficient multi-tenant scheduling, and allows horizontal replication without coordination. Inputs are processed ephemerally and are not retained beyond the scope of a request.

- **Predictions, embeddings, evaluations and explanations:** Beyond producing predictive outputs, KumoRFM returns entity embeddings for downstream use and built-in evaluation metrics to quantify predictive performance. Furthermore, the system incorporates mechanisms for explainability, offering insights into the reasoning processes that lead to its predictions.

Taken together, these components enable KumoRFM to function as an agent-ready system. To facilitate integration, we provide a portable collection of skills[1] for modern LLM-based tooling.

[1] KumoRFM skills: `https://github.com/kumo-ai/kumo-coding-agent`



(a) Context Input (b) Reason within Tables (c) Reason across Tables

Figure 3: **Overview of the KumoRFM-2 architecture. (a)** Given a task table with connected tables as context examples, **(b)** KumoRFM-2 first computes representations for all rows in the context via alternating column- and row-wise attention. These representations are made *task-conditioned* by incorporating context targets directly into the input tables. **(c)** The representations are then enriched via graph attention over primary–foreign key relationships and cross-sample attention over context examples, enabling in-context learning beyond single tables. Finally, the target is predicted from the resulting representation of the prediction example.

3 The KumoRFM-2 Architecture

KumoRFM-2 follows the general blueprint of RFMs as introduced in KumoRFM-1 (Fey et al., 2025) as well as in Sec. 2. A schematic overview of the architecture is shown in Fig. 3. The architecture is highly general, allowing it to process arbitrary relational databases and to make predictions in a wide range of formats. It is trained on a large distribution of tabular and relational tasks, enabling it to efficiently reason over diverse structural priors and capture complex non-linear dependencies between inputs and outputs. This allows the model to generalize effectively to unseen databases and tasks during inference, beyond those encountered during training.

KumoRFM-2 powers a *database-native* design to simultaneously master three distinct scales of information: It reasons over input context examples at the **(1)** *row/column level* within individual tables to understand and relate specific multi-modal attributes, at the **(2)** *foreign-key level* to capture the structural topology and relationships across tables, and at the **(3)** *cross-sample level* to recognize and share patterns across context examples. Context targets are provisioned to the process early, allowing the model to perform *task-conditioned* processing throughout the model. KumoRFM-2 represents a significant improvement over KumoRFM-1. Below, we summarize its key changes:

Hierarchical Attention Scheme. Optimizing performance across these various scales requires hierarchical attention at the table and graph levels. A lightweight network first extracts task-conditioned row embeddings from individual tables through alternating column and row attention. A larger network then distributes and relates these embeddings across tables and context samples via foreign key and cross-sample attention. This staged hierarchical design produces better cell- and structure-aware representations while enabling the removal of noisy data early on. It also avoids the quadratic complexity of attending to each table cell. Instead, the attention scheme scales to large context sizes while preserving the ability to attend across row, column, foreign key and cross-sample dimensions.

Smarter Context and Lagged Targets. Running foundation models over millions of context examples remains an open research question, making effective context selection critical at scale. KumoRFM-2 addresses this with a structured context selection strategy that combines *local* context drawn from the prediction entity itself with *global* context from the most recent database snapshots. This allows the model to condition on its own lagged targets and prior subgraphs while incorporating up-to-date global information from other entities in the database.

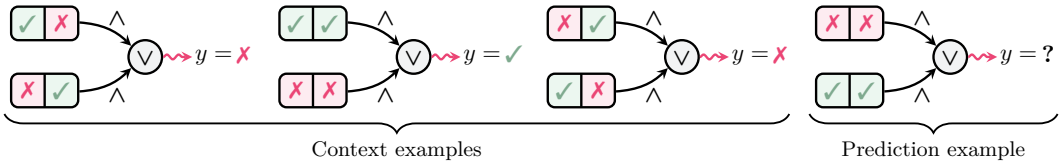


Figure 4: **An adversarial example for fixed-function column-wise encoders.** The label is positive iff both child features co-occur within at least one connected row. Row-level alignment is required to determine the label, which column-wise encoders (Kanter & Veeramachaneni, 2015) fail to capture.

Training Data. KumoRFM-2 is pre-trained on an expanded combination of synthetic and real-world data. Synthetic tables and graphs are generated via *Structural Causal Models* (Pearl, 2009; Peters et al., 2017; Hollmann et al., 2023; Hoppe et al., 2025; Kothapalli et al., 2026), while real-world pre-training data consists of publicly available relational databases paired with a diverse set of predictive queries, executed over a set of random entities at different points in time. Pre-training is conducted in multiple stages, transitioning from single tables to more complex relational structures.

Ensembling. In contrast to v1, KumoRFM-2 enables more fine-grained control over ensembling. This includes randomized column and class order shuffles to reduce sensitivity to input permutations, as well as ensembling over the number of hops and various post-processing techniques. Together, these mechanisms reduce variance across runs and improve robustness to arbitrary input orderings.

Expressivity. Recent approaches apply tabular foundation models to multi-table relational tasks via propositionalization (Kramer et al., 2001; Kanter & Veeramachaneni, 2015), which pre-aggregates relational subgraphs into a single flat table (Hayler et al., 2025; Eremeev et al., 2025; Xu et al., 2026; Wang et al., 2026; Meyer et al., 2026). We argue that such task-agnostic, fixed-function pipeline induces a fundamental expressivity gap. For example, the *deep feature synthesis (DFS)* (Kanter & Veeramachaneni, 2015) utilized in Xu et al. (2026) and in Wang et al. (2026) applies column-wise pre-defined aggregations, thereby limited to capture interactions across rows.

To illustrate this, we consider a database with a primary table T_1 and a child table T_2 containing binary features A and B. For each entity in T_1, we define the target $y = 1$ iff A and B co-occur in at least one related row of T_2 (*cf.* Fig. 4). Consequently, any model on top of column-wise encoders (Xu et al., 2026; Wang et al., 2026) fails to distinguish the two classes due to uninformative column-wise marginals (**AUROC = 0.5**), whereas KumoRFM-2 with its task-conditioned feature extraction scheme achieves perfect separation (**AUROC = 1.0**).

Fine-Tuning. KumoRFM-2 can be fine-tuned and specialized to a specific dataset and task to achieve maximum performance. In this setting, the model weights now encode domain- and task-specific information, resulting in richer parametric knowledge. Consequently, fine-tuning reduces reliance on in-context learning and context selection. This is particularly beneficial in large-scale regimes, where the available training data far exceeds current context size limitations.

4 Experimental Results

For the experimental evaluation, we assess the in-context learning capabilities of KumoRFM-2 on four different benchmark suites: RelBenchV1 (Robinson et al., 2024), RelBenchV2 (Gu et al., 2026), SALT (Klein et al., 2024), and 4DBInfer (Wang et al., 2024). None of these datasets were used during pre-training, which guarantees no leakage of information. We consider binary classification (Sec. 4.1), multi-class classification (Sec. 4.2), and regression tasks (Sec. 4.3). Besides raw performance, we analyze the robustness of KumoRFM-2 across varying data regimes, specifically analyzing its performance under conditions of extreme data scarcity, high feature sparsity, and structural noise (Sec. 4.4). Finally, we analyze the impact of fine-tuning on model performance (Sec. 4.5).

We closely follow the evaluation protocols of the utilized benchmarks, ensuring that KumoRFM-2 adheres to the same temporal constraints and data splits as prior models. Since KumoRFM-2 does not require task-specific training, we utilize data from both the training and validation splits (if provided)

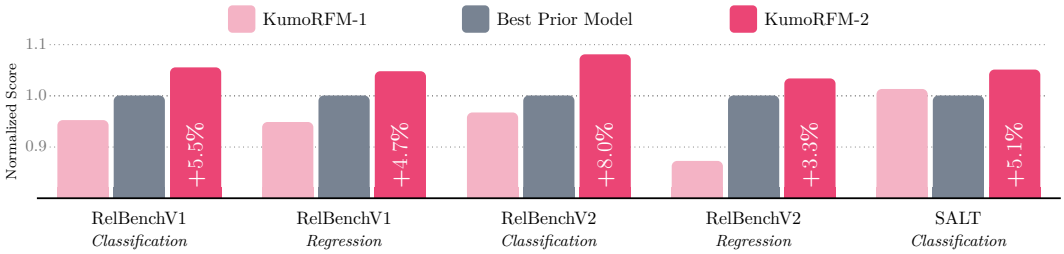


Figure 5: **Performance overview of KumoRFM-2**, where scores are normalized relative to the strongest (supervised) model on each benchmark suite. KumoRFM-2 shows a significant improvement over KumoRFM-1 and consistently outperforms prior supervised or foundational few-shot approaches. Further details are provided in Tables 3, 4, 6, 5, 7, and 8.

Table 1: **Database summary and statistics across RelBenchV1, RelBenchV2, SALT and 4DBInfer.** Datasets vary in the number of tables, rows, columns, and available training examples (#Train). ICL Coverage denotes the fraction of training data used for in-context learning, reaching as low as 0.2%.

	Dataset	Domain	Tasks	#Tasks	#Tables	#Rows	#Cols	#Train	ICL Coverage
RelBenchV1	f1	Sports	Race outcome	3	9	74,063	67	≤11,977	83.5%
	avito	E-commerce	User engagement, CTR	3	8	20,679,117	42	≤116,598	8.6%
	event	Social	Event participation	3	5	41,328,337	128	≤23,424	42.7%
	trial	Medical	Clinical trial outcome	3	15	5,434,924	140	≤12,954	77.2%
	amazon	E-commerce	Churn, LTV	4	3	15,000,713	15	≤5,142,347	0.2%
	stack	Social	Community activities	3	7	4,247,264	52	≤3,633,674	0.3%
	hm	E-commerce	Churn, LTV	2	3	16,664,809	37	≤3,947,966	0.3%
RelBenchV2	mimic	Medical	Length-of-stay	1	6	2,424,751	54	16,999	58.8%
	ratebeer	E-commerce	Engagement, Churn	4	13	13,787,005	221	≤2,563,053	0.4%
	arxiv	Academic	Citations, Publications	2	6	2,146,112	21	≤249,784	4.0%
SALT	SALT	ERP	Business outcome	8	4	4,257,145	31	≤1,916,610	0.5%
4DBInfer	AB	E-commerce	Churn	1	3	24,291,489	15	1,194,773	0.8%
	OB	Social	CTR	1	8	2,170,441,217	31	78,424	12.4%
	RR	E-commerce	CVR	1	3	23,033,676	11	90,003	11.1%
	SE	Social	Churn, Popularity	2	7	5,399,818	49	≤347,285	2.9%
	Total			41	100	2,349,210,440	914	≤68,020,630	0.6%

to populate the model's context. We use at most $10k$ context examples to reflect realistic settings and ensure fair comparison with prior work. Unless otherwise stated, no further training or fine-tuning of KumoRFM-2 is performed, and all predictions are obtained solely through in-context learning on the base model. Benchmark scripts for all experiments are publicly available[2]. An overview of KumoRFM-2's performance across the different benchmarks is shown in Fig. 5.

Databases and Tasks. We evaluate KumoRFM-2 on a diverse collection of benchmark suites comprising 15 relational databases and 41 temporal prediction tasks. These benchmarks span a wide range of domains, including e-commerce, Q&A platforms, medical, academic, enterprise resource planning (ERP), and sports. The tasks cover a variety of temporal prediction problems, such as churn, lifetime value (LTV), click-through rate (CTR), and conversion rate (CVR).

Databases are summarized in Table 1, which vary significantly in the numbers of tables, rows, columns, and available training examples. We also report the coverage of utilized training examples $\frac{\text{\#context examples}}{\text{\#available training examples}}$ for in-context learning in foundation models . Notably, in the largest settings, only up to 0.2% of available training data is actually being used as context examples.

[2]Benchmark scripts: `https://github.com/kumo-ai/kumo-rfm`

Table 2: **Overview of utilized baselines**. We compare against supervised tabular models (including hand-crafted and automatic feature engineering pipelines for flattening multi-table data), state-of-the-art relational deep learning models, and language-based, tabular, and relational foundation models.

	Category	Model	Description
Supervised Tabular ML	*Gradient Boosted*	XGBoost (Chen & Guestrin, 2016)	Ensemble of decision trees trained via gradient boosting, where each tree corrects the residual errors of the previous ones, forming a competitive baseline on tabular data.
	Decision Tree	LightGBM (Ke et al., 2017)	
		CatBoost (Prokhorenkova et al., 2018)	
	Deep Tabular ML	CARTE (Kim et al., 2024)	Deep representation learning on tabular data, trained in a (self-)supervised fashion.
	AutoML	AutoGluon (Erickson et al., 2020)	Ensemble of diverse learners (*e.g.*, Neural Networks, GBDTs) with automated hyperparameter optimization. Generally provides best performance among tabular models.
	Manual Feature Engineering	DS+LightGBM DS+AutoGluon DS+TabPFN-2.5 …(Robinson et al., 2024)	Current gold-standard. A data scientist expert that engineers task-specific features (*e.g.*, temporal aggregations, multi-hop joins) to capture relational dependencies, followed by a tabular model.
	Automatic Feature Engineering	DFS+XGBoost DFS+AutoGluon (Kanter & Veeramachaneni, 2015)	Task-agnostic, fixed-function, column-wise encoder to flatten multi-tables based on deep feature synthesis, followed by a tabular model.
Sup. Relational ML	*Graph Neural Network*	GraphSAGE (Hamilton et al., 2017)	Message-passing networks based on different aggregation schemes and information flows, trained in a supervised fashion based on the relational deep learning paradigm.
		GAT (Veličković et al., 2018)	
		PNA (Corso et al., 2020)	
		RelGNN (Chen et al., 2025)	
	Graph Transformer	HGT/HGT$_{\text{PE}}$ (Hu et al., 2020)	Self-attention networks, trained in a supervised fashion based on the relational deep learning paradigm.
		RelGT (Dwivedi et al., 2026)	
Foundation Models	*Language Model*	LLM$_1$ (Ranjan et al., 2026)	Language models adapted for relational reasoning through structured prompts and context encoding.
		LLM$_2$ (Wydmuch et al., 2024)	
	Tabular Foundation Model	TabPFN-2.5 (Hollmann et al., 2025)	In-context learning over single tabular data based on a pre-trained transformer architecture.
		TabICLv2 (Qu et al., 2026)	
	Relational Foundation Model	Griffin (Wang et al., 2025)	Pre-trained relational foundation model that leverages large-scale meta-learning.
		RT$_{\text{zero}}$ (Ranjan et al., 2026)	Relational transformer for "zero"-shot predictions, pre-trained via masked token prediction over relational data.
		GNN+TabPFN-2.5 (Meyer et al., 2026)	Randomly initialized GNN to flatten multi-tables, followed by a tabular foundation model.
		RDBLearn (Xu et al., 2026)	Task-agnostic deep feature synthesis, followed by a tabular foundation model.
		KumoRFM-1 (Fey et al., 2025)	The first iteration of KumoRFM.

Specifically, we evaluate on all binary classification and regression tasks in RelBenchV1 (Robinson et al., 2024), the standard benchmark for relational deep learning, and its recent extension RelBenchV2 (Gu et al., 2026). In addition, we include the enterprise-scale SALT database (Klein et al., 2024), which captures real-world customer interactions in ERP systems, where common missing fields in business processes must be predicted. ERP systems are critical for managing core business operations, including finance, human resources, production, and supply chains. SALT includes 8 large multi-class classification tasks, with up to 589 classes in the most challenging setting. Finally, we evaluate on binary classification tasks in 4DBInfer (Wang et al., 2024), which extends RelBench with additional databases and tasks, particularly for relationship attribute prediction.

Baselines. We compare KumoRFM-2 against a diverse set of baselines, including supervised tabular models, state-of-the-art relational deep learning models, as well as language-based, tabular, and relational foundation models. To ensure strong and well-tuned baseline performance, we rely on publicly reported benchmark numbers for baselines as much as possible. Baselines evaluated by ourselves are marked with *. Detailed information about all baselines is provided in Table 2.

Specifically, we evaluate tabular (foundation) models in three settings. Unless stated otherwise, tabular models are run on entity table features only, without access to information from related tables. This setting highlights the limitations of tabular models in capturing relational and temporal dependencies without additional feature engineering. To address this, we consider two extended

Table 3: **Test results on the binary classification tasks in RelBenchV1.** Higher is better (AUROC). denotes the best, and the second-best among all models and tasks.

	Method	f1		avito		event		trial	amazon		stack		hm	Avg	Rank
		dnf	top3	click	visit	repeat	ignore	out	user	item	eng	badge	churn	↑	↓
Sup. TabML	LightGBM	68.56	73.92	53.60	53.05	68.04	79.93	70.09	52.22	62.54	63.39	63.43	55.21	63.67	16.75
	DS+LightGBM	69.80	82.40	64.27	64.46	75.42	84.23	72.00	67.60	81.80	90.30	86.20	69.00	75.62	7.75
	DS+AutoGluon*	70.01	82.53	65.66	65.87	74.85	82.31	70.51	68.12	80.45	89.94	85.38	68.71	75.36	8.50
	DS+TabPFN-2.5*	71.16	77.70	64.21	64.53	76.43	85.70	70.75	66.28	79.83	89.07	85.08	68.31	74.92	9.50
Sup. RelationalML	GraphSAGE	72.62	75.54	65.90	66.20	76.89	81.62	68.60	70.42	82.81	90.59	88.86	69.88	75.83	6.42
	HGT	70.77	70.75	63.76	64.32	64.96	82.47	58.37	66.43	77.97	88.47	86.08	66.95	71.78	12.50
	HGT_{PE}	71.17	76.27	64.57	64.95	65.36	81.61	59.21	66.19	78.03	88.17	85.66	65.69	72.24	11.75
	RelGNN	75.29	85.69	68.23	66.18	79.61	86.18	71.24	70.99	82.64	90.75	88.98	70.93	**78.06**	**3.75**
	RelGT	75.87	83.52	68.30	66.78	76.09	81.57	68.61	70.39	82.55	90.53	86.32	69.27	76.65	6.00
Foundational	LLM_1	75.80	91.40	59.80	62.70	71.40	69.30	57.40	58.10	62.10	78.00	80.00	59.80	68.82	13.92
	LLM_2	80.03	87.11	53.36	54.07	70.11	68.65	59.17	62.55	73.41	81.23	79.99	63.81	69.46	13.92
	TabPFN-2.5*	61.21	77.35	54.44	51.45	69.10	70.91	71.74	57.14	59.30	59.66	52.26	54.99	61.63	16.42
	Griffin	57.70	82.50	45.90	60.70	71.88	83.27	51.00	62.30	69.00	77.50	73.50	60.20	66.29	15.42
	RT_{zero}	81.20	89.30	59.50	61.80	73.22	77.47	51.80	64.00	70.90	75.70	80.10	62.80	67.73	13.25
	GNN+TabPFN	64.26	74.67	60.68	65.35	73.58	84.13	56.68	64.30	77.89	87.48	85.34	64.34	71.56	12.58
	RDBLearn	70.87	79.69	69.04	65.49	75.04	82.52	71.58	67.57	82.07	89.39	85.26	68.05	75.97	8.08
	KumoRFM-1	82.41	91.07	64.85	64.11	76.08	89.20	70.79	67.29	79.93	87.09	80.00	67.71	76.71	8.33
	KumoRFM-2	84.59	92.18	67.42	69.41	81.66	90.83	72.03	69.10	82.17	89.40	87.15	69.27	**79.60**	**3.08**

settings for applying tabular models to relational data. First, we leverage an expert data scientist (DS) with MSc degree, 4.0 GPA and 5 years of experience of building machine learning models to manually design features (*e.g.*, temporal aggregations, multi-hop joins) for specific tasks, incorporated into a single wide table (Robinson et al., 2024). Since this process needs to be repeated individually for each task and requires hours to days of human labor, it is not available for all tasks. Secondly, we explore automatic feature engineering pipelines via fixed-function encoders, *e.g.*, based on the deep feature synthesis (DFS) framework (Kanter & Veeramachaneni, 2015) or via randomly initialized graph neural networks (GNNs) (Meyer et al., 2026). Afterwards, these single wide tables can be fed into any tabular model, denoted by prepending DS+ and DFS+ to the tabular model, respectively. Notably, RDBLearn can be interpreted as DFS+TabPFN-2.5 (Xu et al., 2026).

On the foundation model side, we compare against language-based models, tabular foundation models (following the recipe above), and dedicated relational foundation models. Language models (Wydmuch et al., 2024; Ranjan et al., 2026) are prompted with task instructions followed by a series of text-serialized database subgraphs and their targets used as context examples for in-context learning. Relational foundation baselines, such as Griffin (Wang et al., 2025), RT_{zero} (Ranjan et al., 2026) or KumoRFM-1 (Fey et al., 2025), are specifically pre-trained end-to-end on relational data.

4.1 KumoRFM-2 on Binary Classification Tasks

We evaluate KumoRFM-2 on the binary classification tasks of RelBenchV1, RelBenchV2 and 4DBInfer, comparing against data scientist-engineered tabular models, recent relational deep learning models, and various paradigms of foundation models. All models follow the pre-defined benchmark splits, but may operate on different kinds of input representations, including single tables, flattened feature-engineered tables, or relational subgraphs. All relational deep learning models, including KumoRFM-2, use two-hop subgraphs to ensure a fair comparison.

Table 3 presents the results on the 12 RelBenchV1 binary classification tasks. KumoRFM-2 achieves the highest average AUROC of 79.60 and the best overall rank of 3.08 among all evaluated methods, outperforming the best tabular model by 3.98 points, the previously best foundation model KumoRFM-1 by 2.89 points, and even the best supervised relational model RelGNN by 1.54 points.

Comparison to Tabular Models. As expected, tabular models such as LightGBM or TabPFN-2.5, without additional engineered features (*e.g.*, "time since last purchase"), fail to capture predictive signals in these inherently temporal tasks, consistently ranking lowest across all models. With data

scientist–engineered features, tabular models achieve substantially stronger performance and become competitive with supervised relational models. In this setting, DS+LightGBM, DS+AutoGluon, and DS+TabPFN-2.5 perform similarly on average, with minor variations across tasks. However, no such model consistently outperforms relational approaches, suggesting that they do not capture additional predictive signals beyond what is already learned by end-to-end relational models. Moreover, overall performance does not reflect the significant manual effort required for feature engineering. In contrast, KumoRFM-2 captures complex interactions and temporal patterns without manual labor, improving results by 3.98 points. This ability to maintain strong performance across diverse tasks without task-specific engineering and tuning highlights the effectiveness of our relational foundation approach.

Comparison to Supervised Relational Models. KumoRFM-2 outperforms even the strongest supervised baseline by 1.54 points on average across all tasks. To our knowledge, this is the first demonstration of a few-shot foundation model surpassing supervised approaches on relational benchmarks. Notably, this is achieved without any task-specific training and with as little as 0.2% of the available data (context examples *vs.* full training set), making it significantly more practical than supervised methods. KumoRFM-2 is particularly effective in smaller training data regimes (`f1`, `avito`, `event`, `trial`), where it consistently outperforms any supervised approach, often by large margins. We observe that supervised relational models retain a slight advantage on `amazon`, `stack`, and `hm` due to their larger training set sizes, where task-specific tuning can still provide incremental gains. We expect this gap to diminish or reverse when scaling context sizes (*cf.* Sec. 4.4). Notably, while supervised relational models eliminated the need for manual feature engineering, KumoRFM-2 now eliminates the need for additional training, which can range from hours to days.

Comparison to LLMs. KumoRFM-2 consistently outperforms LLM-based baselines on predictive tasks. Overall, LLMs underperform in this setting, as they are not designed for structured prediction and lack appropriate inductive biases for tabular and relational data. We observe competitive performance only in cases with potential data leakage (*e.g.*, `f1`) due to memorization effects in large pre-trained models. Even in these scenarios, KumoRFM-2 significantly outperforms LLM-based approaches, demonstrating the advantage of models explicitly designed for relational prediction.

Comparison to Relational Foundation Models. Existing relational foundation models such as Griffin (Wang et al., 2025) and RT_{zero} (Ranjan et al., 2026) generally tend to underperform on RelBenchV1 despite their strong relational prior. A key limitation lies in their formulation of in-context learning as *label propagation*, which restricts effective context utilization and often reduces predictions to its historical mean. Fixed-function, training-free encoder pipelines as used in GNN+TabPFN (Meyer et al., 2026) and RDBLearn (Xu et al., 2026) instead rely on automatically flattening relational data into single wide tables, effectively deferring most of the modeling capacity to the underlying tabular foundation model. As a result, their performance remains bounded by that of feature-engineered tabular approaches, lacking the ability to perform explicit relational reasoning (*cf.* Sec. 3). In contrast, KumoRFM-2 directly models relational structure and captures complex dependencies beyond column-level aggregations, resulting in average improvements of 8.04 points and 3.63 points in AUROC over these approaches. Remarkably, KumoRFM-2 represents a significant improvement over KumoRFM-1. On average, KumoRFM-2 outperforms v1 by 2.89 points, with consistent gains over all tasks.

KumoRFM-2 on RelBenchV2. We further evaluate KumoRFM-2 on RelBenchV2 (*cf.* Table 4). Due to the recency of the benchmark, publicly reported results are limited. We compare against a subset of supervised tabular and relational approaches, as well as to KumoRFM-1. Again, KumoRFM-2 achieves the highest overall performance, surpassing LightGBM by 8.17 points, GraphSAGE by 1.77 points, and KumoRFM-1 by 3.74 points. Notably, KumoRFM-2 performs strongest on all tasks of the `ratebeer` database, despite it having the largest training set. We also observe substantial gains over KumoRFM-1 on this dataset.

Table 4: **Test results on the binary classification tasks in RelBenchV2.** Higher is better (AUROC). ■ denotes the best, and ■ the second-best among all models and tasks.

Method	mimic	ratebeer			arxiv	Avg	Rank
	stay	beer-churn	user-churn	dormant	citation	↑	↓
LightGBM	51.81	76.21	83.92	75.79	71.21	71.79	3.80
GraphSAGE	55.01	78.67	94.27	80.51	82.50	**78.19**	**2.00**
KumoRFM-1	56.33	75.06	91.38	77.10	80.62	76.22	2.80
KumoRFM-2	55.28	83.84	97.43	80.65	81.71	**79.96**	**1.75**

As `ratebeer` contains the highest number of columns (221), this highlights the importance of task-conditioned feature processing to effectively leverage high-dimensional inputs.

KumoRFM-2 on 4DBInfer. For the 4DBInfer benchmark, we report AUROC results across five binary classification tasks, with supervised baselines sourced from Wang et al. (2024) (*cf.* Table 5). We compare against tabular models with and without automatic feature engineering (DFS+XGBoost and DFS+AutoGluon), as well as supervised relational models and few-shot foundation models.

Table 5: **Test results on the binary classification tasks in 4DBInfer.** Higher is better (AUROC). denotes the best, and the second-best among all models and tasks.

	Method	AB	OB	RR	SE		Avg	Rank
		churn	ctr	cvr	upvote	churn	↑	↓
Sup. TabML	XGBoost	50.00	50.00	50.00	49.68	50.84	50.10	10.6
	AutoGluon	50.00	49.69	50.96	50.81	50.00	50.29	10.4
	DFS+XGBoost	69.22	54.21	79.06	86.75	82.51	74.35	9.0
	DFS+AutoGluon	72.91	54.94	80.08	88.49	83.96	76.08	7.6
Sup. RelML	GAT	76.22	61.46	82.84	88.53	86.45	79.10	5.8
	HGT	77.30	62.60	84.95	88.17	86.70	79.94	3.4
	PNA	76.45	62.49	83.67	88.96	86.64	79.64	4.0
	GraphSAGE	75.71	62.39	84.70	88.61	85.58	79.40	4.6
FDN	RDBLearn	77.41	54.47	84.69	88.45	87.96	78.60	3.8
	KumoRFM-1	75.59	61.11	84.36	88.92	87.87	79.17	4.6
	KumoRFM-2	77.06	61.97	84.36	88.50	87.92	79.96	3.6

KumoRFM-2 performs effectively on par with the best supervised relational models, achieving the highest average performance (79.96 AUROC) and a top overall rank. While it does not dominate any single task, it consistently ranks among the top performers, indicating strong robustness across diverse datasets. In contrast, the best-performing model varies across tasks, suggesting that no single approach generalizes uniformly well.

Notably, KumoRFM-2 matches or exceeds competitive relational baselines on most tasks while avoiding task-specific training. This consistency is particularly evident when compared to models such as RDBLearn or KumoRFM-1, which achieve strong performance on individual tasks but exhibit larger variability overall. Furthermore, approaches based on deep feature synthesis show weaker performance on this benchmark, suggesting the benefits of end-to-end relational modeling.

4.2 KumoRFM-2 on Multi-Class Classification Tasks

We now evaluate KumoRFM-2 in the multi-class classification setting on the SALT benchmark suite. This enterprise resource planning dataset comprises 8 predictive tasks, characterized by significant class imbalances, a wide range of class counts (up to 589 for the "Sales Group" task), and challenges such as label diversity, noise, and distributional drift. Notably, SALT is provided in both multi-table and joined single-table formats, making it an ideal testbed for both tabular and relational models. One key challenge for foundation models specifically is the large class count, in which close to 600 classes

Table 6: **Test results on the multi-class classification tasks in SALT.** Higher is better (MRR). denotes the best, and the second-best among all models and tasks.

	Method	Plant	Shipping Point	Item Incoterm	Header Incoterm	Sales Office	Sales Group	Payment Terms	Shipping Condition	Avg ↑	Rank ↓
Base	Random	0.59	0.11	0.62	0.62	0.99	0.02	0.02	0.12	0.39	10.38
	Majority	0.59	0.54	0.62	0.62	0.99	0.05	0.23	0.41	0.51	9.63
Sup. TabML	DS+XGBoost	0.99	0.95	0.70	0.70	0.99	0.51	0.57	0.68	0.76	5.25
	DS+LightGBM	0.61	0.28	0.73	0.73	0.99	0.02	0.10	0.51	0.50	9.00
	DS+CatBoost	0.99	0.80	0.80	0.81	0.99	0.16	0.44	0.71	0.71	4.63
	DS+CARTE	0.99	0.97	0.75	0.77	0.99	0.46	0.62	0.74	0.79	3.88
	DS+AutoGluon	0.99	0.98	0.78	0.78	0.99	0.34	0.52	0.74	0.77	3.75
Sup. RelML	GraphSAGE_1	0.99	0.97	0.64	0.59	0.99	0.20	0.39	0.59	0.67	6.88
	GraphSAGE_2	0.99	0.98	0.69	0.62	1.00	0.16	0.37	0.57	0.67	5.88
FDN	DS+TabICLv2*	0.94	0.91	0.80	0.74	1.00	0.07	0.50	0.73	0.71	5.63
	KumoRFM-1	0.99	0.99	0.79	0.81	1.00	0.38	0.66	0.78	**0.80**	**1.88**
	KumoRFM-2	0.99	0.98	0.78	0.81	1.00	0.61	0.68	0.79	**0.83**	**1.38**

Table 7: **Test results on the regression tasks in RelBenchV1.** Lower is better (MAE). Average performance $\boldsymbol{\mu}_n$ is normalized relative to the LightGBM baseline. denotes the best, and the second-best among all models and tasks.

	Method	f1	avito	event	trial		amazon		stack	hm	$\boldsymbol{\mu}_n$	Rank
		pos	ctr	attend	adverse	success	user	item	votes	sales	↓	↓
Base	Global Median	4.399	0.043	0.264	57.533	0.462	16.783	64.234	0.068	0.076	1.062	12.78
	Entity Median	8.519	0.046	0.269	57.930	0.441	17.423	66.436	0.069	0.078	1.190	14.00
Sup. TabML	LightGBM	4.170	0.041	0.264	44.011	0.425	16.783	60.569	0.068	0.076	1.000	10.22
	DS+LightGBM	3.963	0.044	0.284	40.581	0.407	13.928	41.122	0.065	0.036	0.879	5.11
	DS+AutoGluon*	4.251	0.045	0.256	44.706	0.430	14.399	45.390	0.068	0.043	0.918	7.11
	DS+TabPFN-2.5*	4.373	0.044	0.318	47.168	0.432	15.631	47.908	0.080	0.059	1.009	10.11
Sup. RelationalML	GraphSAGE	4.022	0.041	0.258	44.473	0.400	14.313	50.053	0.065	0.056	0.918	6.33
	HGT	4.226	0.046	0.264	45.169	0.443	15.412	55.868	0.068	0.064	0.988	10.44
	HGT_{PE}	4.392	0.048	0.261	42.648	0.440	15.864	55.849	0.068	0.064	0.992	10.00
	RelGNN	3.798	0.037	0.238	44.461	0.301	14.230	48.767	0.065	0.054	**0.861**	**4.44**
	RelGT	3.917	0.035	0.250	43.992	0.326	14.267	48.922	0.065	0.054	0.869	4.67
Foundational	TabPFN-2.5*	4.446	0.044	0.527	55.187	0.469	17.513	60.996	0.104	0.096	1.258	14.78
	Griffin	4.460	0.050	0.461	78.232	0.463	35.590	53.214	0.092	0.151	1.471	15.44
	RT_{zero}	2.901	0.058	0.379	73.999	0.455	18.802	57.996	0.110	0.089	1.240	14.00
	RDBLearn	3.834	0.034	0.237	43.913	0.424	14.540	48.559	0.068	0.064	0.906	5.89
	KumoRFM-1	2.747	0.035	0.264	58.231	0.417	16.161	55.254	0.065	0.040	0.908	7.22
	KumoRFM-2	2.854	0.034	0.235	43.293	0.433	13.921	46.992	0.064	0.034	**0.822**	**2.67**

need to be inferred based of only $10k$ context examples. To address this, we adopt a hierarchical classification strategy (Silla & Freitas, 2011) similar to recent tabular foundation models.

Results are reported in Table 6 using Mean Reciprocal Rank (MRR). KumoRFM-2 performs strongly out-of-the-box, outperforming giant tabular model ensembles such as AutoGluon by 6 points, as well as pre-trained and fine-tuned tabular models such as CARTE by 4 points. Notably, KumoRFM-2 outperforms its closest competitor CARTE on all tasks. We observe particularly strong gains from KumoRFM-1 to KumoRFM-2 on the "Sales Group" task, with the largest number of classes. While v1 struggled to achieve competitive performance in this setting, KumoRFM-2 handles large multi-class problems much more effectively, massively outperforming both supervised and foundation models.

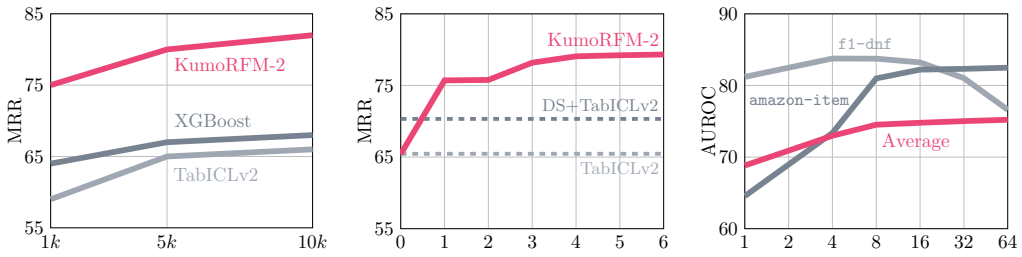
4.3 KumoRFM-2 on Regression Tasks

Finally, we evaluate KumoRFM-2 on the 11 regression tasks from RelBenchV1 and RelBenchV2 (*cf.* Tables 7 and 8). We compare against a similar set of baselines and adopt the same experimental setup as in the binary classification experiments on RelBench (*cf.* Sec. 4.1). We report the additional "Global Median" and "Entity Median" baselines as they provide strong sanity checks regarding model performance. For all experiments, we report task-specific Mean Absolute Error (MAE), and compute normalized average error μ_n relative to the naive LightGBM baseline (lower is better).

Table 8: **Test results on the regression tasks in RelBenchV2.** Lower is better (MAE). Average performance $\boldsymbol{\mu}_n$ is normalized relative to the LightGBM baseline. denotes the best, and the second-best among all models and tasks.

Method	ratebeer	arxiv	$\boldsymbol{\mu}_n$	Rank
	user-count	publication	↓	↓
Global Median	15.124	0.577	0.872	4.5
Entity Median	13.079	0.874	1.079	5.0
LightGBM	20.350	0.577	1.000	5.0
GraphSAGE	7.374	0.513	**0.626**	**2.0**
KumoRFM-1	11.063	0.518	0.712	3.0
KumoRFM-2	7.298	0.487	**0.601**	**1.0**

On RelBenchV1, KumoRFM-2 achieves the lowest normalized MAE ($\mu_n = 0.822$) and the best overall rank across the suite, outperforming the strongest baseline by 4.7% on average. Furthermore, it obtains the top performance in 5 out of the 9 regression tasks. Compared to KumoRFM-1, KumoRFM-2 improves performance by 10.5%, representing a substantial gain. On RelBenchV2, KumoRFM-2 again achieves the lowest score ($\mu_n = 0.601$) with a perfect rank of 1.0, while its predecessor KumoRFM-1 trails behind the supervised GraphSAGE model, highlighting the improved competitiveness of KumoRFM-2 against strong supervised relational baselines.



(a) Context size (b) Subgraph depth (c) Subgraph breadth

Figure 6: **Robustness of KumoRFM-2 with respect to context construction**. KumoRFM-2 exhibits strong sample efficiency, and generally benefits from increased relational subgraph breadth and depth. We identify subgraph breadth as a key factor for optimizing task-specific performance.

Overall, regression tasks exhibit trends similar to those observed in classification on RelBench (*cf.* Sec. 4.1). One notable difference is the increased competitiveness of the data scientist-engineered pipelines. These specific tasks are characterized by high absolute error, suggesting that in regimes where the target is highly volatile, expert-driven manual feature engineering can occasionally outperform automated graph-based approaches. We attribute this to the difficulty of capturing long-term, coarse-grained statistical patterns from fine-grained relational subgraphs at the level of individual transactions. KumoRFM-2 mitigates this limitation through its support for lagged targets within its smarter context selection strategy (*cf.* Sec. 3).

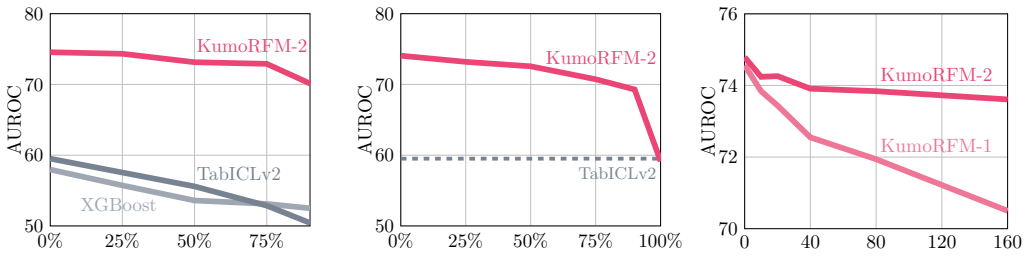
4.4 Robustness of KumoRFM-2

To better understand the performance and robustness of KumoRFM-2, we conduct two sets of studies. First, we analyze the key factors driving model performance, focusing on context size as well as the depth and breadth of relational subgraphs (*cf.* Fig. 6). Second, we evaluate model robustness under challenging conditions, including data scarcity (simulating cold-start scenarios) and increasing feature noise (*cf.* Fig. 7). The results demonstrate the model's ability to effectively leverage relational structure while remaining robust across diverse settings. We now discuss each study in detail:

KumoRFM-2 achieves strong sample efficiency and scales effectively. We evaluate average model performance on a subset of SALT under varying data regimes by adjusting the context size, *i.e.* the number of training/context examples (*cf.* Fig. 6(a)). We compare against XGBoost (supervised) and TabICLv2 (in-context) baselines. Overall, KumoRFM-2 exhibits strong sample efficiency, performing competitively even with small context sizes of $1k$ (0.05% of training examples). Furthermore, KumoRFM-2 follows similar scaling trends as XGBoost and TabICLv2, showing more rapid initial gains followed by more gradual improvements as context/supervision increases. Notably, KumoRFM-2 continues to improve more consistently than the baselines at larger context sizes. We attribute this to the richer context representations, where subgraphs capture more diverse relational structure than flat representations, and thus enable more effective use of larger contexts.

KumoRFM-2 benefits from increasing relational context. We evaluate average model performance on a subset of SALT under varying subgraph depths from zero hops (effectively reducing to a single table setting) up to six hops (*cf.* Fig. 6(b)). We compare against TabICLv2 and DS+TabICLv2. Notably, KumoRFM-2 matches the performance of TabICLv2 at zero hops, demonstrating strong performance as a single tabular foundation model even without relational information. We observe a sharp performance increase when moving from zero to one hop, highlighting the importance of incorporating immediate relational context. From three hops onward, subgraphs cover all available tables in the SALT dataset. Importantly, performance continues to improve as deeper relationships are incorporated, eventually saturating at around four hops.

KumoRFM-2 can benefit from task-dependent neighborhood sizes. We evaluate model performance on a subset of RelBenchV1-classification under varying neighborhood sizes per hop, ranging from 1 to 64, while fixing the subgraph depth to two hops (*cf.* Fig. 6(c)). We report both average performance across tasks and representative task-specific examples. Overall, increasing the



(a) Feature scarcity (b) Link scarcity (c) Number of noisy columns

Figure 7: **Robustness of KumoRFM-2 to sparse, incomplete and noisy data**. KumoRFM-2 maintains strong resilience to data scarcity, and remains robust to noise compared to its predecessor.

neighborhood size leads to steady improvements in average performance, indicating that broader subgraphs enable the model to aggregate more informative relational signals. However, the optimal neighborhood size is inherently task-dependent. For instance, performance on the `driver-dnf` task in `f1` peaks early with a highly localized neighborhood ($n = 4$) and gradually declines as more distant context is introduced. In contrast, the `item-churn` task on `amazon` exhibits a strong positive correlation with neighborhood size, consistently benefiting from a wider relational field. These trends align with intuition: race completion tasks depend primarily on recent driver performance, while tasks such as item demand forecasting benefit from broader historical context. Despite these task-level variations, KumoRFM-2 maintains strong performance across the full range of neighborhood sizes. At the same time, subgraph breadth is a key factor for optimizing task-specific performance.

KumoRFM-2 remains robust under feature scarcity. We evaluate average model performance on a subset of RelBenchV1-classification under varying levels of feature scarcity, where missing values are randomly introduced into the context at increasing rates (*cf.* Fig. 7(a)). We compare against XGBoost (supervised) and TabICLv2 (in-context) baselines. Tabular models are highly sensitive to this perturbation, with performance degrading rapidly as sparsity increases. In contrast, KumoRFM-2 exhibits significantly stronger resilience. Due to its relational structure, the model can access multiple alternative paths to relevant signals across tables, reducing its reliance on any single feature.

KumoRFM-2 maintains performance under link scarcity and cold-start settings. We evaluate average model performance on a subset of RelBenchV1-classification under varying levels of link scarcity, where foreign key relationships are randomly removed at increasing rates (*cf.* Fig. 7(b)). This setup effectively models cold-start scenarios with limited or no historical interactions. Notably, KumoRFM-2 remains highly robust under such structural degradation, maintaining strong performance even when up to 75% of edges are removed. In the extreme case, as link scarcity approaches 100%, the model reduces to a single-table setting, where its performance aligns with that of recent tabular foundation models such as TabICLv2.

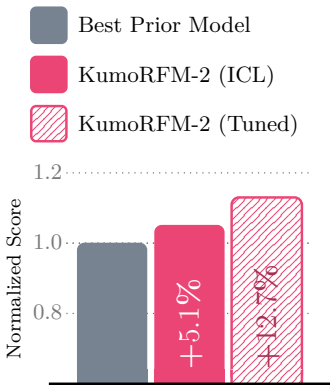
KumoRFM-2 is highly robust to noisy and uninformative features. We evaluate average model performance on a subset of RelBenchV1-classification under varying levels of noise, where a number of noisy columns are injected into the context at increasing rates (*cf.* Fig. 7(c)). While KumoRFM-1 exhibits a steady decline in performance as noise increases, KumoRFM-2 remains significantly more resilient, with performance remaining nearly constant even under heavy noise injection. We attribute this behavior to the task-conditioned intra-table processing of KumoRFM-2, which enables the model to filter out irrelevant information early in the pipeline.

4.5 Fine-Tuning KumoRFM-2

Finally, we evaluate the fine-tuning capabilities of KumoRFM-2 to scale beyond current context size limits and leverage the full training set. We use the SALT benchmark, where 100% of labels are available, compared to only 0.5% in the in-context setting. We compare against the strongest baselines, DS+AutuGluon and DS+Carte, as well as the KumoRFM-2 in-context base model.

Table 9: **Test results on SALT when fine-tuning KumoRFM-2.** Higher is better (MRR). denotes the best, and the second-best among all models and tasks.

Task	DS+Auto Gluon	DS+ CARTE	KumoRFM-2 ICL	KumoRFM-2 Tuned
Plant	0.99	0.99	0.99	1.00
Shipping Point	0.98	0.97	0.98	0.99
Item Incoterm	0.78	0.75	0.78	0.82
Header Incoterm	0.78	0.77	0.81	0.88
Sales Office	0.99	0.99	1.00	1.00
Sales Group	0.34	0.46	0.61	0.74
Payment Terms	0.52	0.62	0.68	0.86
Shipping Condition	0.74	0.74	0.79	0.84
Average (↑)	0.77	0.79	**0.83**	**0.89**



Results are reported in Table 9 using Mean Reciprocal Rank (MRR). Fine-tuning KumoRFM-2 yields substantial performance gains, improving MRR by 10 points over the best supervised baseline and by 6 points over the KumoRFM base model. The fine-tuned model consistently outperforms all baselines across tasks. Gains are particularly pronounced on the "Sales Group" and "Payment Terms" tasks, which denotes the tasks with the largest number of classes. These results demonstrate that fine-tuning effectively overcomes the challenges of in-context learning in large-scale settings. Notably, fine-tuning completes in under two minutes per task.

5 KumoRFM in Practice

KumoRFM-2 is available through our SDK[3] from Python 3.10 onwards via `pip install kumoai`. It exposes a unified interface for graph construction, task specification, and model inference. In addition to direct use through the Python SDK, the same abstractions are exposed as a collection of skills for modern agentic workflow integration. Users can leverage in-context learning through our public inference endpoint, while fine-tuning capabilities are available via enterprise deployments.

A typical workflow is illustrated in Example 1. The SDK constructs a relational graph abstraction directly from database schema information. For example, KumoRFM infers semantic structure such as primary keys, time columns, and inter-table links. This automatic graph construction minimizes manual schema engineering while remaining fully overrideable through a declarative interface.

Once a graph is defined, users can instantiate KumoRFM-2 and issue predictions using a high-level predictive query language. This interface allows users to express complex relational and temporal prediction tasks without explicitly materializing training datasets. Alternatively, users can provide explicit context and prediction tables via a `TaskTable` abstraction, enabling tighter control over context examples. Key runtime parameters allow users to trade off performance and efficiency, including `run_mode` (context budget and model depth), `num_neighbors` (subgraph sampling breadth and depth), `anchor_time` (prediction timestamp), and `inference_config` (ensemble configuration).

In contrast to traditional machine learning pipelines, KumoRFM does not require an explicit separation between training and prediction data. Instead, context examples and prediction examples refer to the *same* underlying graph. For practitioners, this changes the primary abstraction from "building and maintaining data pipelines for model training" to "specifying prediction tasks over a single relational database" in which feature construction and context selection are handled automatically.

Best Practices. KumoRFM is designed to fully leverage relational and temporal structure, and excels in settings where rich interactions and histories are available. In particular, datasets with meaningful entity relationships and temporal dynamics (*e.g.*, user behavior over time) naturally unlock its full potential. To get started, we recommend starting with a focused predictive query over a small set of relevant tables, and progressively expanding the graph as needed. This iterative approach enables rapid prototyping while maintaining clarity over the signal captured by the model. Care

[3]Documentation: `https://kumo.ai/docs`

Example 1: **Using KumoRFM on Snowflake**: From data to predictions in a few lines of code.

```python
from kumoai import rfm

# Connect to a data warehouse and register a graph from its tables ######
graph = rfm.Graph.from_snowflake(
    connection=rfm.backend.snow.connect(...),
    tables=["users", "orders", "items"],
)

# Optionally refine the inferred schema ##################################
# Set primary keys, semantic types, links, or derived columns:
graph["users"].primary_key = "user_id"
graph["items"]["category"].stype = "categorical"
graph.link(src_table="orders", fkey="user_id", dst_table="users")
graph["orders"].add_column(name="total", expr="price * quantity")

# Initialize KumoRFM. Any data compute is pushed down to the database ###
model = rfm.KumoRFM(graph)

# 1) Run ad-hoc predictions using Kumo's predictive query language ######
result = model.predict(
    # Predict if a user will place zero orders in the next 30 days:
    "PREDICT COUNT(orders.*, 0, 30, days)=0 FOR EACH users.user_id",
    indices=[0, 1, 2],          # User IDs to score
    anchor_time="2026-04-07",   # Reference timestamp for prediction
    explain=True,               # Whether to explain the prediction
    return_embeddings=True,     # Whether to return embeddings
    run_mode="fast",            # Context budget: fast|normal|best
    num_neighbors=[32, 32],     # Sampled subgraph width and depth
    lag_timesteps=10,           # Lagged targets per prediction example
    inference_config=dict(      # Ensemble/inference settings:
        num_estimators=4, column_shuffle=True, class_shuffle=True
    ),
)

# 2) Run predictions from explicit context and prediction tables ########
task = rfm.TaskTable(
    task_type="regression",
    context_df=pd.DataFrame({    # Examples used as context
        "user_id": [...], "time": [...], "target": [...]
    }),
    pred_df=pd.DataFrame({       # Examples to score
        "user_id": [...], "time": [...]
    }),
    entity_table_name="users",   # Entity table to predict for
    entity_column="user_id",     # Entity identifier column
    time_column="time",          # Context/prediction timestamp column
    target_column="target",      # Supervision/target column
)
result = model.predict_task(task)
```

should be taken to ensure that relevant information is indeed reachable within a bounded number of hops, which can be controlled via `num_neighbors`. Once a query is stable, advanced parameters such as `run_mode` and `inference_config` can be tuned to optimize performance.

Resources. We provide a collection of notebooks[4] to demonstrate common workflows, including:

- **Database Integration:** Connecting to databases and constructing graphs from SQL tables

[4]Notebooks: `https://github.com/kumo-ai/kumo-rfm`

- **Predictive Query:** Formulating predictive tasks using the Predictive Query Language
- **Explanations:** Understanding and interpreting the different explanation methods in KumoRFM
- **MCP:** Using the KumoRFM MCP server to get started with LLM-based workflows
- **Evaluation:** Evaluating KumoRFM on benchmark datasets and analyzing its performance
- **Single Tabular Data:** Applying KumoRFM in non-relational settings on single tabular data

6 Conclusion

We presented KumoRFM-2, an improved relational foundation model to provide training-free, ad-hoc predictions on relational databases, able to scale to production environments with 500B+ rows. Our results indicate that the relational domain is reaching a critical inflection point. A single pre-trained model can now rival or exceed the performance of task-specific pipelines. This suggests a future where foundation models serve as the standard starting point for predictive modeling in databases.

Acknowledgements

We thank the entire Kumo team for their invaluable support. Special thanks go to Josh Przybylko, Adi Wadaskar, Hema Raghavan, Manush Murali, Raja Rao DV, Keti Jovanovska, Zack Drach, Vanja Josifovski, Myungwhan Kim, Blaž Stojanovič, Siyang Xie, Alan Krumholz, Effy Fang, Abdullah Al-chihabi, Disha Dubey, Devan Johnson, Joseph Oliveira, Angela Liu, Miha Pelko, Vedaant Jain, Sally Liu, Aleksandar S. Sokolovski, Alex Porter, and Ramona Bendias.

References

Mohamed Bouadi, Pratinav Seth, Aditya Tanna, and Vinay kumar Sankarapu. ORION-MSP: Multi-scale sparse attention for tabular in-context learning. In *EurIPS 2025 Workshop: AI for Tabular Data*, 2025. URL https://openreview.net/forum?id=i350aVAClr.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

T. Chen, C. Kanatsoulis, and J. Leskovec. RelGNN: Composite message passing for relational deep learning. In *ICML*, 2025.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020.

Felix den Breejen, Sangmin Bae, Stephen Cha, and Se-Young Yun. Fine-tuned in-context learning transformers are excellent tabular data classifiers, 2025. URL https://arxiv.org/abs/2405.13396.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico Lopez, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2d3j6bt21A.

Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models. *arXiv preprint arXiv:2508.20906*, 2025.

N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. AutoGluon-Tabular: Robust and accurate automl for structured data. *CoRR*, 2003.06505, 2020.

M. Fey, W. Hu, K. Huang, J. E. Lenssen, R. Ranjan, J. Robinson, R. Ying, J. You, and J. Leskovec. Relational Deep Learning: Graph representation learning on relational databases. In *ICML*, 2024.

Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. Kumorfm: A foundation model for in-context learning on relational data., 2025. URL `https://kumo.ai/research/kumo_relational_foundation_model.pdf`.

Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data, 2025. URL `https://arxiv.org/abs/2507.03971`.

Justin Gu, Rishabh Ranjan, Charilaos Kanatsoulis, Haiming Tang, Martin Jurkovic, Valter Hudovernik, Mark Znidar, Pranshu Chaturvedi, Parth Shroff, Fengyu Li, and Jure Leskovec. Relbench v2: A large-scale benchmark and repository for relational data, 2026. URL `https://arxiv.org/abs/2602.12606`.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Adrian Hayler, Xingyue Huang, İsmail İlkan Ceylan, Michael Bronstein, and Ben Finkelshtein. Bringing graphs to the table: Zero-shot node classification via tabular foundation models. *arXiv preprint arXiv:2509.07143*, 2025.

N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *ICLR*, 2023.

N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 2025.

Frederik Hoppe, Astrid Franz, Lars Kleinemeier, and Udo Göbel. Generating synthetic relational tabular data via structural causal models. In *Proceedings of the 4th Table Representation Learning Workshop*, pp. 13–18, 2025.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 2704–2710, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380027. URL `https://doi.org/10.1145/3366423.3380027`.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017.

M. J. Kim, L. Grinsztajn, and G. Varoquaux. CARTE: Pretraining and transfer for tabular learning. In *ICML*, 2024.

T. Klein, C. Biehl, M. Costa, A. Sres, J. Kolk, and J. Hoffart. SALT: Sales autocompletion linked business tables dataset. In *NeurIPS (Table Representation Learning Workshop)*, 2024.

Vid Kocijan, Jinu Sunil, Jan Eric Lenssen, Viman Deb, Xinwei Xe, Federico Reyes Gomez, Matthias Fey, and Jure Leskovec. Predictive query language: A domain-specific language for predictive modeling on relational databases, 2026. URL `https://arxiv.org/abs/2602.09572`.

Vignesh Kothapalli, Rishabh Ranjan, Valter Hudovernik, Vijay Prakash Dwivedi, Johannes Hoffart, Carlos Guestrin, and Jure Leskovec. PluRel: synthetic data unlocks scaling laws for relational foundation models. *arXiv preprint arXiv:2602.04029*, 2026.

Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. *Relational data mining*, pp. 262–291, 2001.

Joe Meyer, , Tom Palczewski, Afreen Shaikh, Mahmoud Mohammadi, Dinesh K. Ramprasath, Karan Paresh, Roshan U. Reddy, and Mark Li. Relational in-context learning on structured data via neighborhood aggregation and structural information. In *Proceedings of the AAAI-26 Summer Symposium Series: AI in Business: Intelligent Transformation and Management*, 2026.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2nd edition, 2009. ISBN 9780521895606.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. Adaptive Computation and Machine Learning series. The MIT Press, Cambridge, MA, November 2017. ISBN 9780262037310.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

J. Qu, D. Holzmüller, G. Varoquaux, and M. L. Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *ICML*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model, 2026. URL https://arxiv.org/abs/2602.11139.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rpPtgMC5s9.

J. Robinson, R. Ranjan, W. Hu, K. Huang, J. Han, A. Dobles, M. Fey, J. E. Lenssen, Y. Yuan, Z. Zhang, X. He, and J. Leskovec. RelBench: A benchmark for deep learning on relational databases. In *NeurIPS*, 2024.

C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 2011.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. Contexttab: A semantics-aware tabular in-context learner. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=kGMRb4jbTP.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

Minjie Wang, Quan Gan, David Wipf, Zhenkun Cai, Ning Li, Jianheng Tang, Yanlin Zhang, Zizhao Zhang, Zunyao Mao, Yakun Song, et al. 4dbinfer: A 4d benchmarking toolbox for graph-centric predictive modeling on rdbs. *Advances in Neural Information Processing Systems*, 37:27236–27273, 2024.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=TxeCxVb3cL.

Yanbo Wang, Jiaxuan You, Chuan Shi, and Muhan Zhang. Relational in-context learning via synthetic pre-training with structural prior, 2026. URL https://arxiv.org/abs/2603.03805.

M. Wydmuch, Ł. Borchmann, and F. Graliński. Tackling prediction tasks in relational databases with llms. *CoRR*, 2411.11829, 2024.

Linjie Xu, Yanlin Zhang, Quan Gan, Minjie Wang, and David Wipf. No need to train your rdb foundation model, 2026. URL https://arxiv.org/abs/2602.13697.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, et al. Limix: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025a.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=t8YRsWY6HM.