KumoRFM: A Foundation Model for In-Context Learning on Relational Data

Corresponding Authors

Matthias Fey* **Vid Kocijan** **Federico Lopez** **Jan Eric Lenssen** **Jure Leskovec***

Abstract

Foundation Models (FMs) have revolutionized unstructured data domains like natural language and images, yet enterprises' most valuable asset—structured and semi-structured relational data—largely miss out on this AI wave. Today, to use AI on enterprise data, one is forced to use conventional machine learning and build per-task and per-dataset specific models that take months to develop and tune.

In this paper, we present a *Relational Foundation Model (RFM)*: a pre-trained model capable of making accurate predictions over any relational database and any predictive task, all of this without requiring any data- or task-specific training. We introduce *Kumo Relational Foundation Model (KumoRFM)*, which extends principles of in-context learning to the complex, multi-table relational graph setting. KumoRFM employs a table agnostic encoding scheme and a novel Relational Graph Transformer to reason within arbitrary multi-modal data across tables. This allows KumoRFM to deliver accurate predictions (*e.g.*, churn, fraud, recommendations, forecasting) within one second, eliminates the need for labor-intensive model development, and paves the way for scalable and explainable AI on enterprise data.

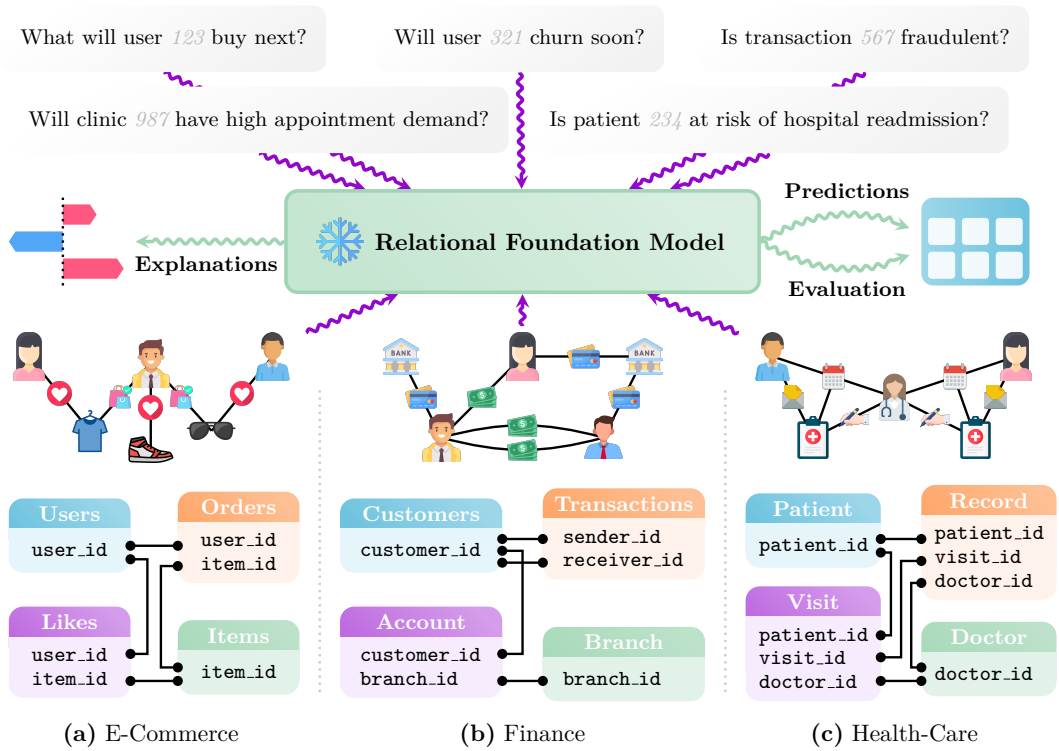
We demonstrate that KumoRFM outperforms conventional approaches, is advantageous over supervised Relational Deep Learning approaches despite requiring no dataset/task specific training, and substantially improves its performance further through fine-tuning capabilities. KumoRFM is available at `https://kumo.ai`.

1 Introduction

The advent of large pre-trained *Foundation Models (FMs)* (Bommasani et al., 2021; Zhou et al., 2024) has fundamentally transformed how users extract value from unstructured data like text, images, audio, and video. This progress, fueled by training on massive corpora, has led to models capable of encapsulating world knowledge and performing complex reasoning tasks (Touvron et al., 2023; Grattafiori et al., 2024; DeepSeek-AI et al., 2025). A key feature is their ability to adapt to previously unseen tasks via *in-context learning (ICL)* (Brown et al., 2020) by generalizing from a few examples, prompted at prediction time.

However, there exists no foundation model designed specifically for enterprise data. Enterprise data is typically stored in structured relational tables in data warehouses. Examples of such data are *e.g.*, customer records, transaction histories, supply chain interactions, product catalogs, financial ledgers, and electronic health records. Such data are arguably the most valuable asset that organizations use for predictive decision making. It is commonly used to answer questions, such as: "Is a given transaction fraudulent?", "What product is a customer likely to buy next?", "Will a customer churn?", "What will be the sales of a product next quarter?", "Will a shipment arrive late?".

*Correspondence to `matthias@kumo.ai`, `jure@kumo.ai`



(a) E-Commerce (b) Finance (c) Health-Care

Figure 1: **Key capabilities of Relational Foundation Models.** RFMs can be applied to new/unseen databases and schemas with highly varying structural characteristics, as found in **(a)** e-commerce, **(b)** finance, or **(c)** health care. Secondly, they can be applied to any predictive task type, ranging from one-off assessments (*e.g.* entity-level fraud prediction) to temporal predictive queries (*e.g.* temporal recommendation prediction). Thirdly, they generalize to new predictive tasks and give accurate predictions without any task-specific model tuning. Finally, not only do RFMs support prediction outputs, but they also offer insights into the reasoning processes via explanations, and build trust through extensive quantitative evaluation mechanisms.

Constructing such a foundation model presents a unique and substantial challenge. Such a model needs to effectively learn across complex database schemas with an arbitrary number of tables and columns, and handle the inherent heterogeneity of column types, which often have divergent semantic meanings. Crucially, there is a wide range of open questions: What should such a foundation model for relational data be capable of? Will it be able to generalize to new databases and tasks, even the ones it was never trained on? Could it provide accurate predictions from a few in-context examples? How does the neural network architecture look like? How should such a model be trained? What data shall one use to train it? How can one efficiently apply it in real-time?

In this work, we present kumo *Relational Foundation Model (KumoRFM)*, the first foundation model designed specifically for enterprise relational data. This model possesses several key advanced capabilities (*cf.* Fig. 1): First, it is able to seamlessly adapt to database schemas unseen during its training phase. This includes the ability to accommodate diverse structural characteristics, such as varying numbers of tables and different types of relationships (*e.g.*, one-to-many, many-to-many). Second, KumoRFM can handle a wide range of column types effectively, including columns that are proprietary or opaque, such as custom upstream embeddings or hashed identifiers. Finally, KumoRFM flexibly adapts to a spectrum of predictive tasks specified at inference time through a unified prompting interface. Not only are the predictions accurate; the model is able to perform them even for tasks it was never explicitly trained for. Consequently, the scope of tasks that KumoRFM can solve far exceeds rudimentary operations like missing cell imputation. For example, KumoRFM is capable of handling complex predictive tasks such as temporal forecasting—*e.g.*, predicting user churn. Another example is forecasting inventory demand that requires careful analysis of past sales, supplier reliability, seasonal trends, and macroeconomic indicators stored relationally. Furthermore,

KumoRFM goes beyond merely producing predictive outputs. It incorporates mechanisms for explainability, offering insights into the reasoning processes that lead to its predictions. To build further trust and avoid the fear of hallucinations, KumoRFM also facilitates quantitative prediction accuracy evaluation to let users judge the quality of predictions without relying on them blindly.

KumoRFM enables users to generate accurate predictions, such as item recommendations, customer churn detection, or fraud identification, directly on *any* set of structured data tables without the need to manually engineer and train task-specific or dataset-specific models. Built on a generalized Transformer architecture, KumoRFM is pre-trained to learn and reason across multiple tables of structured data at once. KumoRFM delivers predictions in less than one second, achieving accuracy comparable to state-of-the-art supervised deep learning models. The model facilitates rapid experimentation and deployment across an almost boundless set of diverse use cases. KumoRFM can be further fine-tuned to a specific task, delivering even higher accuracy.

Specifically, KumoRFM builds on the principles of *Relational Deep Learning (RDL)* (Fey et al., 2024), which represents relational data as a temporal heterogeneous graph, where each entity is represented as a node, and the primary-foreign key links between entities define the edges. It utilizes table-wise attention mechanisms in the form of a *Relational Graph Transformer* (Dwivedi et al., 2025), which operates on the local subgraph centered around the entity of interest. Its table-invariant feature representation enables seamless transfer across tables with varying multi-modal column types, *i.e.* numerical values, (multi-)categorical variables, timestamps, free-form text, images, embeddings, anonymized inputs, such as hashed identifiers and custom upstream embeddings, where a natural language prior would invariably fail.

We introduce a novel declarative mechanism for prompting KumoRFM. Our *Predictive Query Language (PQL)* is a SQL-like syntax designed to define predictive modeling tasks. In PQL, a predictive query is structured to specify the target variable, the entity for which predictions are made, and optional filters to refine the dataset. PQL (and thus KumoRFM) supports various predictive tasks, including regression, binary classification, multi-class and multi-label classification, as well as link prediction. The flexibility of the language allows for diverse applications such as item recommendation, user engagement analysis, fraud detection, forecasting, and many others.

KumoRFM utilizes in-context learning, enabling it to generalize to previously unseen tasks from a collection of labeled examples to new, future examples directly during prediction time. KumoRFM executes a given Predictive Query with a powerful real-time in-context label generator that dynamically produces task-specific context labels for any given entity by leveraging temporal neighbor sampling (Wang et al., 2021; Fey et al., 2024). This mechanism is used both to construct input subgraphs and to derive in-context labels in a time-consistent manner. In-context learning is then applied in two complementary ways: **(1)** within the subgraph of an entity by attending to its own historical labels as well as those of its entities nearby; and **(2)** across sampled subgraphs, using subgraph-wise attention to capture broader contextual patterns. This dual mechanism not only reduces the number of context examples required, but also prioritizes context that is more relevant according to temporal and relational proximity.

Lastly, KumoRFM incorporates explainability at both the global data level and the individual entity level (Simonyan et al., 2013; Sundararajan, 2024; Ying et al., 2019). For any prediction, users can examine the features and columns most relevant to the prediction and assess the importance of individual nodes and edges within the subgraph of the entity.

We evaluate KumoRFM on 30 different predictive tasks coming from 7 diverse publicly available datasets in RELBENCH (Robinson et al., 2024). RELBENCH is a benchmark for Relational Deep Learning that comprises a wide range of relational databases and tasks. Importantly, KumoRFM was never trained nor tuned on any of the datasets or predictive tasks included in RELBENCH. Results show that across 30 tasks from 7 datasets, on average, KumoRFM outperforms both the de-facto gold-standard of feature engineering as well as end-to-end supervised deep learning approaches by 2% to 8% across three different task types. When fine-tuned on a specific task, KumoRFM can improve its performance further by 10% to 30% on average. Most importantly, KumoRFM is orders of magnitude faster than conventional approaches that rely on supervised training, and provides a zero-code solution to query *any* entity and *any* target at *any* future point in time.

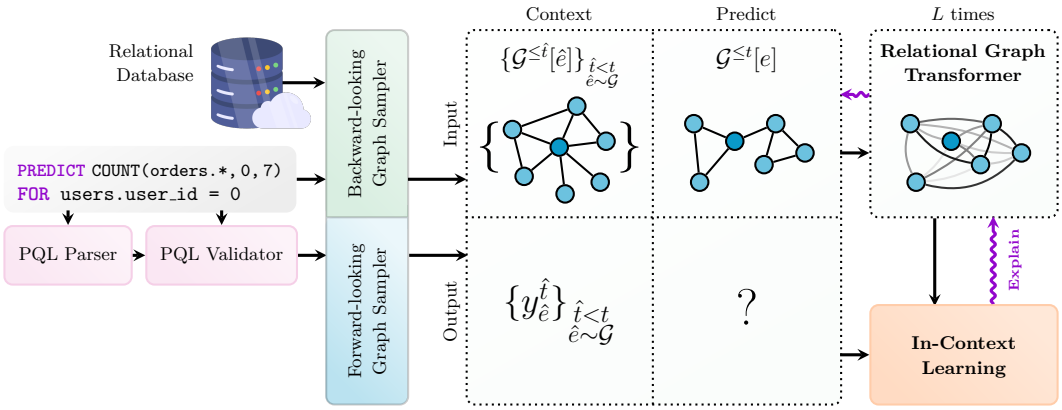


Figure 2: **Overview of the KumoRFM architecture.** Given a relational database and a predictive query, KumoRFM constructs context and prediction subgraphs by dynamically sampling from the database: it uses a backward-looking graph sampler to obtain time-consistent context and prediction subgraphs, and a forward-looking sampler to retrieve ground-truth labels. Context and prediction data are then fed into a Relational Graph Transformer, which learns generalized subgraph representations to enable in-context learning.

2 Overview of KumoRFM

KumoRFM is the first system to offer accurate, explainable, and trustworthy predictions on any database and any predictive task, all without any task/dataset-specific training or tuning. At the core of KumoRFM is the Predictive Query Language (PQL), an expressive declarative language for specifying predictive tasks (Sec. 2.1). With this, users can effectively "talk to their data"[2], issuing high-level queries for tasks such as forecasting or recommendation, and instantly receiving predictions, explanations and quantitative evaluation metrics. Explanations are available at fine-grained analytical and textual summary levels. The quantitative prediction accuracy evaluation process enables users to build confidence in the model's predictions through both performance-centric and behavioral metrics.

The KumoRFM architecture is composed of several tightly integrated components that operate in harmony to deliver accurate, fast, and responsive performance:

- A powerful real-time **in-context label generator** that dynamically curates task-specific context labels for *any* entity at *any* point in time.
- A novel pre-trained **Relational Foundation Model** that seamlessly integrates a table-width invariant column encoder and performs table-wise attention mechanisms.
- A comprehensive **explainability** module that leverages both analytical and gradient-based techniques to provide explanations at the global data level as well as for individual entities.
- A **fine-tuning** module designed to optimize a query for final large-scale production use-cases, ensuring it runs efficiently at the scale of billions of predictions.

Fig. 2 highlights the inter-play of the different building blocks into a unified system. KumoRFM is a running system connected to a per-enterprise specific relational database which stores its structured data (*e.g.*, product catalogs, transaction records, customer data, supply chain data). KumoRFM internally represents such data as a temporal, heterogeneous graph $\mathcal{G}$ (Fey et al., 2024). Based on this representation, KumoRFM has the capability to seamlessly traverse through the graph in real-time, and query or sample subgraphs around a given entity e (e.g. a customer or product) at any specific point in time t. This sampling approach allows (recursive) access to neighbors (*e.g.*, orders attached to a user or item) in constant time and is highly customizable: it allows for specifying the number of hops, tables, and metapaths to sample based on different temporal sampling strategies (*e.g.*, uniform, most recent, fixed time interval), and can be adaptive (*i.e.*, up to a certain node budget) to allow for maximally sized subgraphs even in cold start scenarios.

[2]A natural language interface to PQL converter is also available, though we omit its details here for brevity.

Once a predictive query is issued, it gets parsed, validated and transformed into an abstract syntax tree. Such a predictive query unambiguously defines the task type (node-level prediction *vs.* link-level prediction) and query type (one-off assessment *vs.* temporal prediction) (Sec. 2.1). KumoRFM then builds upon the principles of in-context learning, and connects it to the Relational Deep Learning setting. That is, given an entity e and a timestamp t, we sample the k-hop subgraph $\mathcal{G}_k^{\leq t}[e] \subseteq \mathcal{G}$ around entity e up to timestamp t, and use it as input to make a prediction $\tilde{y}_e^t$ according to the user-defined predictive query. In the conventional RDL setup, such a model is trained from scratch, using offline-generated historical labels $y_{\hat{e}}^{\hat{t}}$, $\hat{t} < t$, $\hat{e} \sim \mathcal{G}$, for supervision. However, in the in-context learning setup, these historical labels are instead prompted during prediction time to a pre-trained but frozen KumoRFM:

$$\tilde{y}_e^{(t)} = \text{KumoRFM}_{\boldsymbol{\theta}}^{\textcolor{cyan}{\text{❄}}} \left(\mathcal{G}_k^{\leq t}[e], \left\{ \left(\mathcal{G}_k^{\leq \hat{t}}[\hat{e}], \; y_{\hat{e}}^{\hat{t}} \right) \right\}_{\substack{\hat{t} < t \\ \hat{e} \sim \mathcal{G}}} \right), \tag{1}$$

where $\hat{t}$ and $\hat{e}$ refer to randomly sampled timestamps and entity nodes, respectively. This approach fundamentally departs from conventional supervised learning where model parameters $\boldsymbol{\theta}$ are trained for a specific dataset and task, with the final model being applied to unseen examples at inference time. Instead, KumoRFM is pre-trained to reason about the non-linear dependency between (subgraph, label) pairs in a single forward pass to derive a prediction for a test subgraph, and thus can naturally generalize to unseen datasets and tasks. Importantly, to accommodate for real-time use-cases, all required input data (*i.e.* the entity subgraph, context subgraphs and their labels) are generated and combined on-the-fly (Sec. 2.2).

KumoRFM is built on a generalized Transformer architecture, and offers two key innovations: **(1)** context labels are dynamically attached to each subgraph in order to favor context that is relevant according to temporal and relational proximity; **(2)** a *Relational Graph Transformer* (Dwivedi et al., 2025) that utilizes this input data to perform self-attention in order to exchange information across multiple tables and context labels; KumoRFM supports node (binary, multi-class, multi-label) classification, node regression, and link prediction tasks, relying on individual routines to map them into the in-context learning setup (Sec. 2.3). KumoRFM is trained on a mix of both publicly available real-world databases and synthetic data. No private enterprise data was used during its training phase.

Finally, we overcome one of the major limitations of in-context learning via *fine-tuning* capabilities: while in-context learning allows for rapid experimentation and obtaining accurate predictions for a pre-defined set of entities, it does not scale well to billions of task-specific predictions requested at once. Here, fine-tuning converts the in-context workflow into a supervised production pipeline, ensuring efficient deployment at scale by specializing the model to a given predictive query. We briefly describe these internals as well as the support for explanations and evaluations in Sec. 2.4.

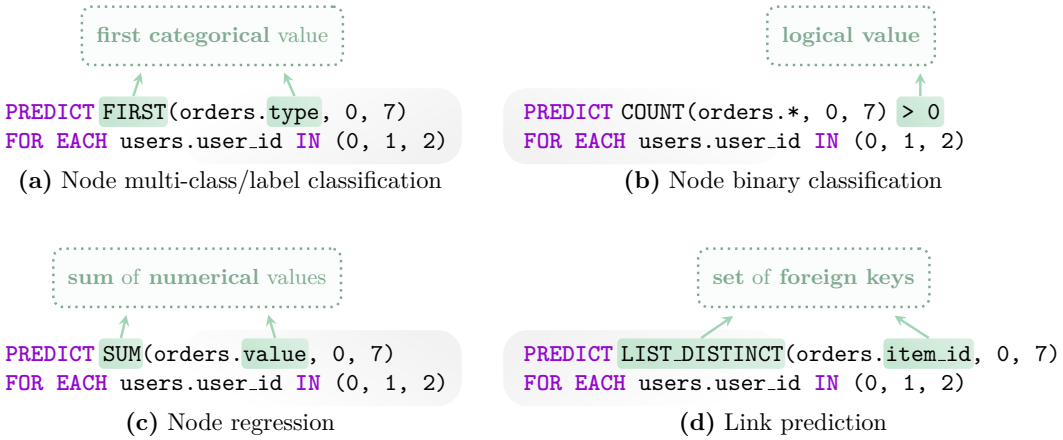
2.1 Prompting KumoRFM: The Predictive Query Language Interface

To enable users to explicitly and unambiguously specify a predictive task, we introduce the *Predictive Query Language (PQL)*. While RDL eliminates the need for tedious feature engineering, PQL eliminates the need for training set and label generation. In contrast to traditional query languages designed for data manipulation (*e.g.*, SQL (Chamberlin & Boyce, 1975)), PQL abstracts away physical data manipulation and instead focuses solely on the domain and predictive label definitions, leaving out unnecessary details such as data sampling that do not require explicit user input.

Every query has a clearly separated label (`PREDICT` clause) and entity definition (`FOR` clause), defining both the entities for which predictions are madehe label for every individual entity. Additional filters (`WHERE` clause) can be applied to both the label and entity clause. PQL supports aggregations (`SUM`, `COUNT`, `AVG`, `LIST_DISTINCT`, *etc.*), binary operations ($=$, $<$, `CONTAINS`, *etc.*), and logical operations across different time granularities (hours, days, months). For more details on the PQL's structure and syntax refer to its official documentation[3].

PQL is capable of handling a broad set of task types, such as node (binary, multi-class, multi-label) classification, node regression, and link prediction, ranging from missing cell imputation to complex forecasts. We find that these task types cover the large majority of practically-encountered predictive problems, allowing users to use KumoRFM for most practical problems. Examples of predictive queries and their corresponding mappings to a task type are given in Fig. 3.

[3]PQL documentation: `https://docs.kumo.ai/docs/pquery-structure`



(a) Node multi-class/label classification **(b)** Node binary classification

(c) Node regression **(d)** Link prediction

Figure 3: **Mapping from predictive queries to task types.** The aggregation scheme and the semantic type of the target column uniquely determine the underlying machine learning task.

Defining the predictive problem through PQL offers key advantages for robust foundation model predictions:

- The label definition is independent of the entity and time, allowing us to generate additional labeled input data for additional entities, even if they are not specified by the input.
- The language is intentionally restrictive around manipulation of time to ensure that past labels can safely be computed for any timestamp with an unambiguous forward- and backward-looking time frame. This is essential to computing valid labels and preventing data leakage.
- The syntax and graph specification alone define the task type, allowing us to automatically pick a suitable model head for down-stream processing.

2.2 Online Context Generation

The Predictive Query Language uniquely defines a predictive task. As such, it also defines the procedure on how to obtain ground truth labels from historical snapshots of the data, which we can use to generate context labels to perform in-context learning within KumoRFM. Since KumoRFM is a running system designed to answer any predictive task in real-time, task-specific context-labels need to be dynamically produced in real-time as well. Such context labels inherit a similar structure compared to the training table $\hat{T}$ introduced in RDL, *i.e.* it is given by a set of triplets $\{(e_i, t_i, y_i)\}_{i=1}^{n}$ that tie an entity e and timestamp t to a ground-truth label y. However, the RDL framework relies on training tables to be curated and generated offline. In our case, the system does not know the predictive query in advance so such pre-processing is not possible nor required.

In order to meet real-time requirements, we enabled predictive queries to be issued *online*. For this, we rely on temporal sampling procedures based on fixed time intervals on a set of metapaths, as parsed from the predictive query definition (*i.e.* the metapaths to reach target definitions and entity filter constraints). Afterwards, target definitions and entity filters can be efficiently computed directly on the GPU in a mini-batch fashion. As such, we rely on *forward-looking* graph samplers to generate the in-context labels, while *backward-looking* graph samplers generate the corresponding input subgraphs. On average across a diverse set of predictive queries, we can generate $\approx$ 2M in-context labels in less than one second.

In order to answer a query in KumoRFM with high accuracy, we dynamically update the graph on-the-fly, grouping historical labels into a dynamically constructed training table $\hat{T}$ and attaching it via primary key-foreign key connections to the entity table. This training table contains labels of historical timestamps of all entities sampled within the input test subgraph (including the seed entity itself). This mechanism models both temporal proximity in the form of autoregressive labels (Box & Jenkins, 1976) (*e.g.*, the sum of monthly transactions of the seed user over the past year) and relational proximity (*e.g.*, the sum of monthly transactions of nearby users over the past year). Such

relational proximity tends to be especially helpful in homophilous tasks, in which entities are likely to be connected to others who share similar characteristics, providing invaluable learning signal.

2.3 The KumoRFM Architecture

KumoRFM embeds entity-centric subgraphs followed by an in-context learning module. In particular, it consists of three main stages: **(1)** a table-invariant row embedding module that can operate on multi-modal data of all types (numerical, categoricals, free text, embeddings, *etc*), **(2)** a Relational Graph Transformer (Dwivedi et al., 2025) to perform attention-based message passing between different nodes in the graph, and **(3)** a final ICL module that performs in-context learning from context subgraphs to test subgraphs. We now describe this architecture in more detail.

Table-Invariant Row-level Representations. The initial module encodes individual rows in the database into dense vector representations. Since tables within a database (and across databases) have different width and diverse multi-modal column types, the KumoRFM encoding module is designed to be agnostic to these kinds of variances. In particular, each column is encoded w.r.t. its *semantic type* which specifies the "modality" of the column. A variety of semantic types are supported to handle column types, such as numerical (*e.g.*, price and age columns), categorical (*e.g.*, gender and educational-level columns), multi-categorical (*e.g.*, movie genres), timestamp (*e.g.*, date of events), text (*e.g.* product descriptions) or embeddings (*e.g.* custom upstream embeddings) (Hu et al., 2024).

A database holds a collection of tables $\mathcal{T} = \{T_1, \ldots, T_n\}$. The encoder is used to encode each cell in every table $T \in \mathcal{T} \cup \{\hat{T}\}$ into a representation $\mathbf{C}_T \in \mathbb{R}^{N_T \times C_T \times F}$ of fixed-length F, where N_T refers to the number of rows and C_T refers to the number of columns in table T. Afterwards, we utilize a Transformer (Vaswani et al., 2017; Lee et al., 2019) on the two-dimensional grid $\mathbf{C}_T$ to obtain row-level representations $\mathbf{H}_T^{(0)} \in \mathbb{R}^{N_T \times F}$ for every table T, which is then used for subsequent processing. Importantly, thanks to its attention formulation, the Transformer is agnostic to table size, allowing it to generalize across arbitrary table schemas and dimensions.

Table-wise Interaction. In order to exchange information across tables, we utilize a graph-based representation in which nodes corresponds to rows in all tables and edges capture primary-foreign key relationships. In order to capture rich structural patterns and enabling long-range modeling capabilities, we rely on a *Relational Graph Transformer* (Dwivedi et al., 2025), which applies self-attention within the full (sub-)graph, including the dynamically attached context table $\hat{T}$. For this, the row-wise embeddings $\mathbf{H}_T^{(0)} \in \mathbb{R}^{N_T \times F}$, $T \in \mathcal{T} \cup \{\hat{T}\}$, are extended by various positional encoding schemes that define the final token representation, *i.e.*:

- A **node type encoder** encodes the table type of each table $T \in \mathcal{T} \cup \{\hat{T}\}$.
- A **hop encoder** captures the structural proximity between the entity node and other nodes in the subgraph.
- A **time encoder** encodes the relative time of facts/events w.r.t. the prediction time t.
- A **subgraph encoder** captures the local fine-grained graph structure. This effectively captures parent-child relationships or important structural relationships such as cycles.

Finally, after L layers of table-wise interaction via the Relational Graph Transformer, we read out the representation $\mathbf{h}_{T,e}^{(L)} \in \mathbb{R}^F$ of the entity e in table T of interest.

In-Context Learning. In order to apply the final stage of in-context learning, each of the subgraphs in the training/context set and in the test set are individually encoded into fixed vector representations, following the procedure described above. We stack these representations into context representations $\mathbf{H}_{\text{train}}^{(L)}$ (obtained from earlier timestamps) and test representations $\mathbf{H}_{\text{test}}^{(L)}$, which we finally feed together with context labels $\mathbf{y}_{\text{train}}$ into an ICL module built upon a Transformer architecture to obtain class probabilities or regression values.

Link Prediction. To cast link prediction tasks such as recommendation into the in-context learning framework, KumoRFM utilizes fully-inductive pair-wise representations of users and items Yuan et al. (2025). Specifically, both user and item representations are read out from the user-centric subgraph

using the Relational Graph Transformer, and are then fed into the in-context learning module for classification. To address scalability challenges, item representations are uniformly sampled to a fixed context size according to sampling depth, ensuring the model learns signals from items across a diverse range of hops (*e.g.*, repeated purchases, collaborative patterns).

2.4 Advanced Capabilities

Next, we showcase three advanced capabilities enabled by KumoRFM:

Model Prediction Explainability. KumoRFM can provide explanations to every prediction, both at the global data level as well as for individual entities. Explanations can be turned on by user request, and are made available as visualizations or textual summaries.

At the global data level, we introduce an analytical framework that organizes column-level context data into *cohorts* and links their distributions to the ground-truth labels. For instance, this allows us to assess how user age correlates with churn behavior, or how purchased product categories influence lifetime value predictions. Importantly, this formulation accommodates column-level data across all tables by leveraging *weighted* cohorts in adjacent tables. To quantify the importance of each column, we compute the variance of model predictions across its cohorts—higher variance suggests greater relevance of the column to the downstream prediction task.

Since KumoRFM is fully differentiable, we can leverage gradient-based explanation techniques (Simonyan et al., 2013; Sundararajan, 2024; Ying et al., 2019; Chen et al., 2020; Huang et al., 2020) to understand the fine-grained impact of individual cells within an input subgraph on the model's final prediction. These methods reveal how changes in the input affect the model's predictions. We use *saliency* (Simonyan et al., 2013) as our gradient-based method of choice, due to its strong trade-off between interpretability and computational efficiency (Amara et al., 2022). Saliency works by computing the gradient of the model's prediction w.r.t. each input feature, highlighting the most influential components. A key novelty of our approach is the adaptation of gradient-based explanations to multi-modal, cell-level inputs. Instead of assigning importance scores at the feature level, we compute scores per cell—each of which may span multiple features—using specialized aggregation routines tailored to each semantic type. This provides actionable insights on the level of cells for individual predictions. For example, we can identify which product categories of past product purchases most strongly influenced a recommendation, or detect risk indicators in fraud detection scenarios, such as anomalous transaction times or unusually high purchase amounts.

Prediction Accuracy Evaluation. To build trust in model predictions and take away the fear of hallucinations, KumoRFM supports a quantitative prediction accuracy evaluation mode that allows users to assess the quality of a prediction for a given query. Specifically, for a temporal prediction, we evaluate its performance using most recent historical data snapshots for which ground-truth labels are reliably known. For one-off assessments, we simulate missing values by masking known cells and measuring the model's imputation accuracy.

KumoRFM reports both performance-oriented metrics—such as AUROC, AP, MAE, MAPE, or MAP@k—as well as behavioral metrics that capture qualities like diversity and mitigating popularity bias. Such evaluation can be used to expand the predictive query to fit user intent more precisely.

Fine-Tuning. While in-context learning is a powerful mechanism to query a model in an *a*d hoc online fashion, it becomes inefficient for repeated execution of the same predictive query that runs predictions over billions of entities.

A common strategy to speed-up computations is to reuse context embeddings $\mathbf{H}^{(L)}_{\text{train}}$ or cache their key-value pairs in the final Transformer layer, enabling a partial separation between training and inference. KumoRFM automatically adopts this caching strategy when making predictions over multiple test inputs. However, the final prediction head must still attend to *all* in-context examples, which continues to limit scalability for large prediction workloads.

To address this issue, we further introduce fine-tuning capabilities into KumoRFM. Here, fine-tuning refers to the process of specializing KumoRFM to a single dataset and a single task (Howard & Ruder, 2018). In order to make KumoRFM task-specific, we replace its table-invariant encoders with dataset-specific ones, and substitute the final in-context learning head with a task-specific one, *e.g.*, as

described in Yuan et al. (2025). We then follow the general RDL blueprint to fine-tune the model in a supervised fashion based on a pre-generated training table. We found fine-tuning especially helpful in link-level prediction tasks, as it allows us to shift from an inductive to a transductive setting for improved model performance.

3 Related Work

Our work on KumoRFM builds on two primary lines of research: Representation learning over relational databases, and the emerging field of foundation models for structured data.

3.1 Relational Deep Learning

The field of *Relational Deep Learning (RDL)* (Fey et al., 2024) has marked a significant paradigm shift towards end-to-end deep learning over *raw* relational databases, alleviating the need for manual feature engineering that typically bottlenecks conventional machine learning pipelines. Its core idea is to represent relational tables as a temporal, heterogeneous graph, on which *Graph Neural Networks (GNNs)* (Gilmer et al., 2017; Hamilton et al., 2017; Yuan et al., 2025; Chen et al., 2025) or *Graph Transformers* (Zhang et al., 2020; Ying et al., 2021; Rampášek et al., 2022; Dwivedi et al., 2025) are used to learn data-driven representations and obtain autonomous predictive models. Each table corresponds to a node type, each record corresponds to a node, and primary-foreign key links define the edges. While RDL approaches have demonstrated strong performance in learning from complex relational structures, they generally adhere to a paradigm where distinct models are trained from scratch and in isolation for each new dataset and task. In contrast, KumoRFM presents a foundation model that adapts to new tasks via in-context learning.

3.2 Foundation Models for Structured Data

The pursuit of general-purpose models that can adapt to diverse tasks on previously unseen data has recently expanded from natural language and vision into the realms of structured data. Our work on KumoRFM builds upon and diverges from several key research trajectories in this space, which we broadly categorize into three main areas:

LLMs on Textified Structured Data. One direct approach to leveraging the power of LLMs for structured data involves transforming the data into a textual format. Here, tables or graphs are serialized into formats like JSON or natural language descriptions, which are then fed as input prompt together with a question to an LLM (Hegselmann et al., 2023; Fang et al., 2024; Zhao et al., 2023; Wydmuch et al., 2024). This strategy aims to directly harness the impressive zero/few-shot capabilities and reasoning abilities of LLMs. However, these approaches generally undergo brittle *prompt engineering*, suffer from large context windows and hallucination (Gardner et al., 2024), fail to handle datasets that lack textual grounding, struggle to capture numerical patterns (Thawani et al., 2021), and require careful evaluation due to potential leakage concerns (Bordt et al., 2024). At the core, there is a fundamental mismatch between the training objective (*i.e.* next token prediction) and the task they are aiming to complete (*i.e.* minimizing the error of a forecast).

Graph Foundation Models via Text-Attributed Graphs. Another emerging line of research seeks to identify an analogue of natural language prompting for graph machine learning tasks. In particular, it tackles the problem of heterogeneity in graph foundation models through the use of *text-attributed graphs* (Chen et al., 2024b; He et al., 2024; Wang et al., 2025), where textual node attributes are encoded via (frozen) LLMs to project nodes into a unified embedding space (Tang et al., 2023; Chai et al., 2023; Chen et al., 2024a; Fatemi et al., 2024). Prominent approaches involve augmenting the graph with *prompt nodes* and *label nodes* that connect readout nodes from both context and test subgraphs (Huang et al., 2023; Liu et al., 2023, 2024a). A GNN is then applied to the resulting *super-graph* to predict labels for the test subgraphs. However, studies have shown that even when textual representations are aligned, positive transfer remains limited to within-domain scenarios, as graphs from different domains often exhibit substantially divergent structural patterns (Liu et al., 2024b). Moreover, these methods impose a non-negligible amount of computational cost, as each node must be individually encoded into a high-dimensional space by an LLM, and are inherently challenging to tune jointly for many practical graph sizes and LLM choices.

Table 1: **Statistics of RELBENCH datasets.** Datasets vary significantly in the number of tables, total number of rows, and number of columns.

Dataset	Domain	#Tasks	Tables			Timestamp (year-month-day)	
			#Tables	#Rows	#Columns	Start	End
rel-amazon	E-commerce	7	3	15,000,713	15	2008-01-01	2016-01-01
rel-avito	E-commerce	4	8	20,679,117	42	2015-04-25	2015-05-14
rel-event	Social	3	5	41,328,337	128	1912-01-01	2012-11-29
rel-f1	Sports	3	9	74,063	67	1950-05-13	2010-01-01
rel-hm	E-commerce	3	3	16,664,809	37	2019-09-07	2020-09-14
rel-hm	Social	5	7	4,247,264	52	2009-02-02	2021-01-01
rel-trial	Medical	5	15	5,434,924	140	2000-01-01	2021-01-01
Total		30	51	103,466,370	489	1912-01-01	2020-09-14

Tabular Foundation Models. Recently, foundation models for single data tables have been introduced which do not rely on text-attributed inputs (Hollmann et al., 2023, 2025; Qu et al., 2025). These models identify column-wise and row-wise relationships among cells in a table directly within a single forward pass of a pre-trained Transformer, and derive the non-linear dependency between inputs and outputs from a large set of context examples. However, such tabular foundation models still face several key limitations: They are restricted to small-scale datasets due to limitations in context length, number of features, and output class size. They rely on complex input normalization schemes and feature shuffling, which necessitates data wrangling and ensembling to maintain robustness. And last, they are confined to single, flat tabular representation, which means that multiple data tables still need to be joined and flattened via manual feature engineering. KumoRFM moves this paradigm to the next level and provides a foundation model for arbitrary relational databases containing tables.

4 Results

For the experimental evaluation, we utilize 7 relational datasets and 30 predictive tasks introduced in RELBENCH (Robinson et al., 2024). RELBENCH datasets, covering E-commerce, social, medical, and sports domains. The databases vary significantly in the numbers of rows (*i.e.* data scale), the number of tables and columns, as well as time ranges. The databases are summarized in Table 1.

It is important to note that KumoRFM has not seen any RELBENCH dataset during its pre-training phase, which guarantees *no* leakage of information.

We evaluate KumoRFM on entity classification, entity regression and recommendation tasks, and compare to the following baselines methods (whenever applicable for a given task type):

LightGBM A gradient boosting framework that builds an ensemble of decision trees, trained in a supervised fashion (Ke et al., 2017).

Data Scientist An expert data scientist that solves each task by manually designing features and feeds them into a supervised tabular model, the prior gold-standard for building predictive models on relational databases (Robinson et al., 2024).

RDL An end-to-end supervised GNN baseline as introduced in Robinson et al. (2024), utilizing a heterogeneous GRAPHSAGE (Hamilton et al., 2017) model. For recommendation tasks, we further report the performance of a traditional two-tower GraphSAGE model as well as the performance of the pair-wise NBFNET (Zhu et al., 2021) model, both utilizing the same GNN backbone.

LLM A direct application of a LLAMA 3.2 3B model (Grattafiori et al., 2024) as introduced in (Wydmuch et al., 2024), receiving short descriptions of the relational database schema and the task, and a number of in-context subgraphs in JSON format. The LLM is then asked to make a prediction for a test example. In the same spirit as KumoRFM, no training is taking place here. However, we want to point out potential leakage concerns of this baseline, since all datasets we evaluate on are publicly available (*e.g.*, predicting driver positions in F1 races).

Table 2: Test results (AUROC) on the entity classification tasks in RELBENCH. Higher is better. **KumoRFM in-context** is making predictions from in-context provided examples. The model was not trained on these tasks or datasets. **KumoRFM fine-tuned** was fine-tuned on each specific task.

		SUPERVISED			FOUNDATIONAL		
Dataset	**Task**	**LightGBM**	**Data Scientist**	**RDL**	**LLM**	**KumoRFM** (in-context)	**KumoRFM** (fine-tuned)
rel-amazon	user-churn	52.22	67.60	**70.42**	62.55	67.29	**70.47**
	item-churn	62.54	81.80	**82.81**	73.41	79.93	**82.83**
rel-avito	user-visits	53.05	—	66.20	53.36	64.85	**78.30**
	user-clicks	53.60	—	65.90	54.07	64.11	**66.83**
rel-event	user-repeat	53.05	—	76.89	53.36	76.08	**80.64**
	user-ignore	79.93	—	81.62	68.65	**89.20**	**89.43**
rel-f1	driver-dnf	68.86	69.80	72.62	80.03	**82.41**	**82.63**
	driver-top3	73.93	82.40	75.54	87.11	**91.07**	**99.62**
rel-hm	user-churn	55.21	69.00	69.88	63.81	67.71	**71.23**
rel-stack	user-engagement	63.39	90.30	90.59	81.23	87.09	**90.70**
	user-badge	63.43	86.20	88.86	79.99	80.00	**89.86**
rel-trial	study-outcome	70.09	**72.00**	68.60	59.17	70.79	71.16
Average ↑		62.44	—	75.83	68.06	**76.71**	**81.14**

We evaluate pre-trained KumoRFM on the provided test sets of the RELBENCH tasks, using only historical context from earlier timestamps. Importantly, no further training is performed at this stage. Additionally, we report the performance of KumoRFM after fine-tuning it specifically for each dataset and task.

4.1 Entity Classification Tasks

The entity classification tasks involve predicting binary labels for a given entity at a specific anchor time across 12 distinct tasks. Performance is measured using AUROC, with higher values indicating better predictive accuracy.

Table 2 presents the results of KumoRFM's in-context and fine-tuned models compared to supervised baselines (LightGBM, Data Scientist, RDL) and a no-training LLM baseline. Notably, KumoRFM's in-context learning model performs strongly out-of-the-box compared to supervised Data Scientist and RDL baselines, even exceeding the supervised RDL approach on average (75.83 AUROC on average in RDL *vs.* 76.71 AUROC on average in KumoRFM in-context). KumoRFM also outperforms the foundational LLM baseline on every dataset and task. Overall, we found the LLM baseline to be underperforming except on datasets where there exists clear leakage concerns (*i.e.* `rel-f1`) due to the LLM's memorization capabilities. Even in those cases, KumoRFM significantly outperforms the LLM.

Lastly, we observe that fine-tuning KumoRFM has the potential to improve its in-context mode further, especially on datasets with large number of labels available. On average, fine-tuned models achieve a 7% relative improvement over the best-performing supervised baseline.

4.2 Entity Regression Tasks

Entity regression tasks require predicting a continuous scalar for a given entity at a specific anchor time. We evaluate performance using Mean Absolute Error (MAE), where lower values indicate more accurate predictions.

Table 3 presents the results of KumoRFM for both its in-context and fine-tuned modes, alongside the supervised baselines LightGBM, feature engineered Data Scientist models, and RDL. Unlike

Table 3: Test results (MAE) on the entity regression tasks in RELBENCH. Lower is better. **KumoRFM in-context** is making predictions from in-context provided examples. The model was not trained on these tasks or datasets. **KumoRFM fine-tuned** was fine-tuned on each specific task.

		SUPERVISED			FOUNDATIONAL	
Dataset	**Task**	**LightGBM**	**Data Scientist**	**RDL**	**KumoRFM** (in-context)	**KumoRFM** (fine-tuned)
rel-amazon	user-ltv	16.783	**13.928**	14.313	16.161	14.226
	item-ltv	60.569	**41.122**	50.053	55.254	48.670
rel-avito	ad-ctr	0.041	—	0.041	**0.035**	**0.034**
rel-event	user-attendance	0.264	—	0.258	0.264	**0.238**
rel-f1	driver-position	4.170	3.963	4.022	**2.747**	**2.731**
rel-hm	item-sales	0.076	0.036	0.056	0.040	**0.034**
rel-stack	post-votes	0.068	**0.065**	**0.065**	**0.065**	**0.065**
rel-trial	study-adverse	44.011	**40.581**	44.473	58.231	44.225
	site-success	0.425	0.407	0.400	0.417	**0.301**
Normalized Average w.r.t. RDL ↓		1.100	—	1.000	**0.984**	**0.862**

classification, no foundational LLM baseline is included here, as LLMs lack the capacity to directly produce reliable numerical outputs for such tasks.

KumoRFM's in-context mode achieves very promising results, obtaining a relative gain of 1.6% over the RDL baseline on average. However, we observe that the regression tasks pose unique challenges. The Data Scientist baseline performs best on 3 out of the 9 tasks, exclusively on tasks with high MAE where fine-grained graph reasoning may offer limited additional value. This gives rise to future investigation how deep learning models across the board can better adapt to these settings. However, this strength does not generalize across the board.

Notably, once fine-tuned, KumoRFM consistently delivers even stronger results, outperforming all baselines on 5 out of 9 tasks. This highlights its capacity to adapt to the numerical prediction setting, especially once given access to sufficient task-specific data.

4.3 Recommendation Tasks

Recommendation tasks involve predicting a ranked list of top-k target entities for a given source entity at a specific anchor time. The value of k is predefined and varies across tasks in RELBENCH (Robinson et al., 2024). Performance is measured using Mean Average Precision at k (MAP@k), where higher values indicate better results.

Following prior work, we evaluate a LightGBM baseline that concatenates user and item features, along with two graph-based methods: a two-tower GNN that computes pairwise scores via inner product (Hamilton et al., 2017), and NBFNET (Zhu et al., 2021), which derives pairwise embeddings from user-centric subgraphs.

Table 4 presents the results. Remarkably, KumoRFM's in-context mode achieves strong performance despite the inherent difficulty of in-context recommendation. This mode operates purely in the inductive setting—without learning shallow embeddings, which are commonly used to enhance model expressiveness (Wang et al., 2019; He et al., 2020; Yuan et al., 2025). On average, KumoRFM outperforms both GRAPHSAGE and NBFNET by a wide margin (7.29 MAP@k vs. 1.85 MAP@k vs. 6.74 MAP@k), and leads on 5 out of the 9 tasks.

When fine-tuned, KumoRFM transitions from an inductive to a transductive setting (Yuan et al., 2025), further boosting performance. In this mode, it achieves state-of-the-art results on all 9 tasks.

Table 4: Test results (MAP@k) on the recommendation tasks in RELBENCH. Higher is better. **KumoRFM in-context** is making predictions from in-context provided examples. The model was not trained on these tasks or datasets. **KumoRFM fine-tuned** was fine-tuned on each specific task.

		SUPERVISED			FOUNDATIONAL	
Dataset	**Task**	**LightGBM**	**RDL**		**KumoRFM**	
			GRAPHSAGE	NFBNET	(in-context)	(fine-tuned)
rel-amazon	user-item-purchase	0.16	0.74	0.10	**1.72**	**2.93**
	user-item-rate	0.17	0.87	0.12	**1.14**	**2.25**
	user-item-review	0.09	0.47	0.09	0.22	**1.63**
rel-avito	user-ad-visit	0.06	0.02	3.66	**4.02**	**4.17**
rel-hm	user-item-purchase	0.38	0.80	2.81	2.73	**3.14**
rel-stack	user-post-comment	0.04	0.11	12.72	11.83	**13.34**
	post-post-related	2.00	0.07	10.83	**11.80**	**12.21**
rel-trial	condition-sponsor-run	4.82	2.89	11.36	11.29	**11.65**
	site-sponsor-run	8.40	10.70	19.00	**20.83**	**28.02**
Average ↑		1.79	1.85	6.74	**7.29**	**8.82**

4.4 Explanations

Next, we present illustrative examples that demonstrate KumoRFM's capability for explainable predictions. Specifically, we compute importance scores using both global column-level features and localized subgraph cells, leveraging a combination of analytical and gradient-based explanation methods. These signals are then passed through an LLM to generate natural language summaries.

Below, we show the generated explanation for a user churn query on the `rel-hm` dataset, where the user is predicted to churn:

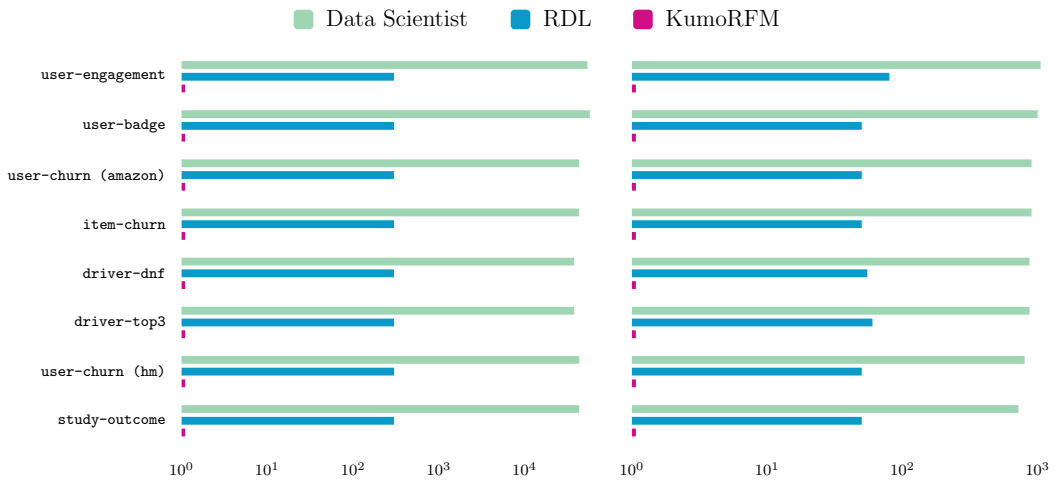
```
PREDICT COUNT(orders.*, 0, 30) > 0 FOR users.user_id=1
```

The model predicts that the user has a moderate likelihood of placing at least one order in the upcoming month. Key factors influencing this prediction include:

- ***Order Count:*** *Users with only a few past orders have a very low likelihood of ordering soon, while those with more orders show increased probabilities.*
- ***Order Date Recency:*** *Recent orders (6-12 months ago) greatly increase the chance of placing new orders soon.*
- ***Fashion News Frequency and Club Membership:*** *Users who regularly receive fashion news or have active club membership status show higher probabilities of ordering.*

The explanations accurately highlight key signals in the data, emphasizing on **Order Count**, **Order Date Recency**, and **Fashion News Frequency and Club Membership** status. These factors suggest why the model predicts the user is likely to churn: few historical orders, no recent activity, and a lack of engagement through fashion news or club membership.

Explainability is also available in the recommendation setting, enabling insights into why certain items are suggested based on past user behavior or external factors. In the textual summary below, the model justifies its emphasis on items such as jackets by referencing prior user interactions, including **Recent Views and Past Order History**. Moreover, it indicates a strong likelihood that the user will act on these recommendations soon based on **User Status**, relating the suggestions with associated output probabilities:



(a) Hours Human Labor (in seconds, log-scaled) (b) Lines of Code (log-scaled)

Figure 4: **Time to first prediction in (a) hours of human labor and (b) lines of code (LoC), illustrated on node-level classification tasks.** KumoRFM is orders of magnitude faster ($\approx$ 1 second *vs.* $\approx$ 30 minutes *vs.* $\approx$ 12.3 hours) and requires zero-code to get to accurate predictions (1 LoC *vs.* $\approx$ 56 LoC *vs.* $\approx$ 878 LoC) compared to Data Scientist and RDL baselines, respectively.

```
PREDICT LIST_DISTINCT(orders.item_id, 0, 7) FOR users.user_id=2
```

The model predicts the likely distinct items that the user will order in the next week. Key points supporting this prediction include:

- ***Recent Views and Past Order History:*** *The descriptions of items recently viewed by the user (e.g., jackets, t-shirts, jumpers) are among the most important features, with importance scores above 95%. The user's interest in specific product types (such as jackets) is a strong predictor of future orders.*
- ***User Status:*** *The user is currently active, which supports the prediction that they will make purchases soon.*

4.5 Time To First Prediction

A major advantage of KumoRFM is that it empowers end-users to effortlessly explore various use-cases, identify promising queries quickly, and generate predictions within seconds—no prior machine learning expertise needed.

In contrast, even a data scientist with a Stanford CS MSc degree (4.0 GPA) and five years of experience building machine learning models (Robinson et al., 2024) typically requires several hours to produce a first working model. This process involves the following time-consuming steps[4]:

- **EDA:** Understanding the data by examining the schema, key relationships, and distributions.
- **Feature Ideation:** Brainstorming potential features manually, often with pen and paper.
- **Feature Engineering:** Implementing the designed features by writing and running SQL queries.
- **Model Training:** Training a decision-tree model, including extensive hyperparameter tuning.

While RDL has substantially reduced the cost of human effort through representation learning, it still incurs the overhead of managing task-specific models—covering training, inference, re-training on updated datasets, and other maintenance steps. With KumoRFM, none of these burdens remain.

[4]Data Scientist Notebooks: `https://github.com/snap-stanford/relbench-user-study`

To validate this, we compare KumoRFM against the data scientist and RDL baselines in terms of **(1)** hours of human effort required and **(2)** number of new lines of code needed to complete each task. As shown in Fig. 4, KumoRFM delivers predictions in under one second, while a data scientist requires an average of 12.3 hours, and RDL takes approximately 30 minutes per task.

From a coding perspective, data scientists write an average of 878 ± 77 lines to solve a single task. Even with RDL—excluding the architecture and training boilerplate—registering a new task and its training table still requires around 56 ± 8.8 lines of code. In contrast, KumoRFM reduces this to a *single* line of code for defining a predictive query via PQL.

These results highlight the core value proposition of a relational foundation model: enabling real-time predictions with minimal effort, paving the way for a new generation of predictive systems that can be stacked, queried, and operationalized to drive faster and smarter business decisions.

5 Conclusion

We defined a blueprint for a Relational Foundation Model, a pre-trained model capable of making accurate predictions over any relational database and any predictive task without requiring any supervised training. kumo Relational Foundation Model is the first system to implement this vision by applying in-context learning principles to the multi-table relational graph setting. KumoRFM delivers accurate, fast and responsive performance by integrating Relational Graph Transformers and in-context learning modules into a unified architecture. It is accessible through the Predictive Query Language, from which it dynamically generates context labels of historical snapshots in real time. In future iterations, we are eager to close the gap between in-context and fine-tuned model performance.

A KumoRFM team members (alphabetical)

Alexa Murray, Blaž Stojanovič, Dong Wang, Eric Tang, Federico Lopez, Jan Eric Lenssen, Jure Leskovec, Kexin Huang, Matthias Fey, Siyang Xie, Vid Kocijan, Viman Deb, Xinwei He, Zecheng Zhang

We thank the entire Kumo.AI team for their invaluable support.

References

K. Amara, Z. Ying, Z. Zhang, Z. Han, Y. Zhao, Y. Shan, U. Brandes, S. Schemm, and C. Zhang. GraphFramEx: Towards systematic evaluation of explainability methods for graph neural networks. In *LOG*, 2022.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, M. S. Chatterji, A. S. Chen, K. A. Creel, J. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. E. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. F. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. P. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. F. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. H. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. P. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. A. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models. *CoRR*, 2108.07258, 2021.

S. Bordt, H. Nori, and R. Caruana. Elephants never forget: Testing language models for memorization of tabular data. *CoRR*, 2403.06644, 2024.

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

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.

Z. Chai, T. Zhang, L. Wu, K. Han, X. Hu, X. Huang, and Y. Yang. GraphLLM: Boosting graph reasoning ability of large language model. *CoRR*, 2310.05845, 2023.

D. D. Chamberlin and R. F. Boyce. SEQUEL: A structured english query language. *Communications of the ACM*, 18(6):377–385, 1975.

J. Chen, Y. Miao, and X. Chen. Parameterized explainer for graph neural network. In *NeurIPS*, 2020.

R. Chen, T. Zhao, A. Kumar Jaiswal, N. Shah, and Z. Wang. LLaGA: Large language and graph assistant. In *PMLR*, 2024a.

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

Z. Chen, H. Mao, J. Liu, Y. Song, B. Li, W. Jin, B. Fatemi, A. Tsitsulin, B. Perozzi, H. Liu, and J. Tang. Text-space graph foundation models: Comprehensive benchmarks and new insights. *CoRR*, 2406.10727, 2024b.

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

V. P. Dwivedi, S. Jaladi, Y. Shen, F. López, C. I. Kanatsoulis, R. Puri, M. Fey, and J. Leskovec. Relational graph transformer. *CoRR*, 2505.10960, 2025.

X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, and C. Faloutsos. Large language models (LLMs) on tabular data: Prediction, generation, and understanding - a survey. *PMLR*, 2024.

B. Fatemi, J. Halcrow, and B. Perozzi. Talk like a graph: Encoding graphs for large language models. In *ICLR*, 2024.

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.

J. Gardner, J. C. Perdomo, and L. Schmidt. Large scale transfer learning for tabular data via language modeling. In *NeurIPS*, 2024.

J. Gilmer, S. S. Schönholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, and B. et al. Chern. The Llama 3 herd of models, 2024.

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

X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 2020.

X. He, X. Bresson, T. Laurent, A. Perold, Y. LeCun, and B. Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *ICLR*, 2024.

S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *AISTATS*, 2023.

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.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

W. Hu, Y. Yuan, Z. Zhang, A. Nitta, K. Cao, V. Kocijan, J. Leskovec, and M. Fey. PyTorch Frame: A modular framework for multi-modal tabular learning. *CoRR*, abs/2404.00776, 2024.

Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang. GraphLIME: Local interpretable model explanations for graph neural networks. *CoRR*, 2001.06216, 2020.

Q. Huang, H. Ren, P. Chen, G. Kržmanc, D. Zeng, P. Liang, and J. Leskovec. PRODIGY: enabling in-context learning over graphs. In *NeurIPS*, 2023.

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.

J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang. One for all: Towards training one graph model for all classification tasks. In *ICLR*, 2024a.

J. Liu, H. Mao, Z. Chen, W. Fan, M. Ju, T. Zhao, N. Shah, and J. Tan. One model for one graph: A new perspective for pretraining with cross-domain graphs. *CoRR*, 2412.00315, 2024b.

Z. Liu, X. Yu, Y. Fang, and X. Zhang. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*, 2023.

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.

L. Rampášek, M. Galkin, V. Prakash, A. T. Luu, G. Wold, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. In *NeurIPS*, 2022.

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.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, 1312.6034, 2013.

A. Yan Q. Sundararajan, M. Taly. Axiomatic attribution for deep networks. In *ICML*, 2024.

J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang. GraphGPT: Graph instruction tuning for large language models. *CoRR*, 2310.13023, 2023.

A. Thawani, J. Pujara, P. A. Szekely, and F. Ilievski. Representing numbers in NLP: a survey and a vision. In *NAACL*, 2021.

H. Touvron, L. Martin, K. R. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. M. Kloumann, A. Korenev, P. S. Koura, M. A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, Pu. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

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.

S. Wang, J. Huang, Z. Chen, Y. Song, W. Tang, H. Mao, W. Fan, H. Liu, X. Liu, D. Yin, and Q. Li. Graph machine learning in the era of large language models (LLMs). *TIST*, 2025.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019.

Y. Wang, Y. Cai, Y. Liang, H. Ding, C. Wang, and B. Hooi. Time-aware neighbor sampling for temporal graph networks. *CoRR*, abs/2112.09845, 2021.

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

C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T. Y. Liu. Do transformers really perform bad for graph representation? In *NeurIPS*, 2021.

R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*, 2019.

Y. Yuan, Z. Zhang, X. He, A. Nitta, W. Hu, D. Wang, M. Shah, S. Huang, B. Stojanovič, A. Krumholz, J. E. Lenssen, J. Leskovec, and M. Fey. ContextGNN: Beyond two-tower recommendation systems. In *ICLR*, 2025.

J. Zhang, H. Zhang, C. Xia, and L. Sun. Graph-BERT: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140, 2020.

J. Zhao, L. Zhuo, Y. Shen, M. Qu1, K. Liu, M. Bronstein, Z. Zhu, and J. Tang. GraphText: Graphs as natural language for learning tasks. *CoRR*, 2310.01089, 2023.

C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, H. Peng, J. Li, J. Wu, Z. Liu, P. Xie, C. Xiong, J. Pei, P. S. Yu, and L. Sun. A comprehensive survey on pretrained foundation models: a history from BERT to ChatGPT. *IJMLC*, 2024.

Z. Zhu, Z. Zhang, L. P. Xhonneux, and J. Tang. Neural Bellmann-Ford networks: A general graph neural network framework for link prediction. In *NeurIPS*, 2021.