

RelICL: Training-free Relational Learning with Tabular Foundation Models

Simon Forbat, Rainer Gemulla

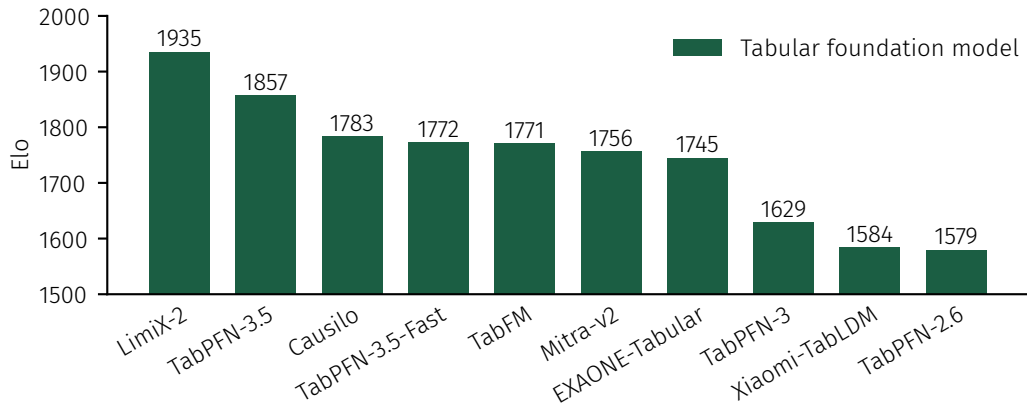
October 7, 2026

Chair of Practical Computer Science I: Data Analytics (Prof. Gemulla)

Tabular foundation models (TFMs) are strong and cheap

- Pretrained, predict via **in-context learning** (Brown et al., 2020)
- **No training** on the target dataset
- Highly active area of research: TabPFN (Hollmann et al., 2022, 2025; Jäger et al., 2026), TabICL (Qu et al., 2025, 2026), TabFM (Kong et al., 2026), LimiX (Zhang et al., 2025), ...
- Top of the TabArena leaderboard (Erickson et al., 2025), ahead of tuned GBDTs

The TabArena leaderboard is dominated by Tabular Foundation Models

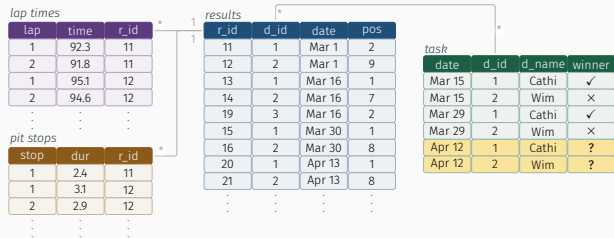


<https://huggingface.co/spaces/TabArena/leaderboard> (Oct 5, 2026)

Much real-world data is relational.

Relational data

- Many tables, linked by keys
- Racing, e-commerce, social networks, clinical trials, ... (Fey et al., 2024; Robinson et al., 2024)
- **Task:** predict a class or an attribute of the rows of one *task table*
(here: whether a driver wins)



What do we want to predict? (Robinson et al., 2024; Gu et al., 2026)

Database	Task	Question
rel-amazon	user-churn	Will the user review a product in the next 3 months?
rel-hm	item-sales	What are the total sales of an article next week?
rel-trial	study-outcome	Will the trial reach its desired outcome?
rel-avito	ad-ctr	What is the click-through rate of an ad?
⋮	⋮	⋮

- **Many task types:** classification, regression, forecasting, link prediction, recommendation, missing value imputation
- **Our focus:** (entity) classification and regression

Can we use TFMs for relational data? → Yes!

- Relational databases consist of (interconnected) **tables**

Why should we?

- TFMs yield **strong performance**
- TFMs require **no training** on the target database
- Better / customized TFMs directly give better relational predictions

Are there any alternatives?

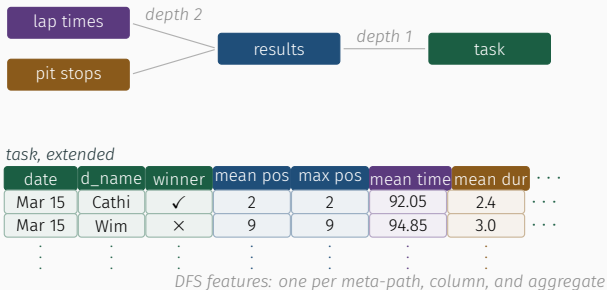
- Relational deep learning, e.g., GNNs (Fey et al., 2024; Chen et al., 2025), Graph Transformers (Dwivedi et al., 2025)
 - **Requires supervised training** for every task⇒ Strong but costly
- Relational foundation models, e.g., RT (Ranjan et al., 2025), PluRel (Kothapalli et al., 2026), KumoRFM (Fey et al., 2025)
 - Brand new models for relational data, **pretrained from scratch**⇒ Strong but costly

→ Neither approach makes use of the **knowledge available in TFMs**

How to use TFMs for relational data?

One answer: Deep Feature Synthesis (DFS) (Kanter & Veeramachaneni, 2015)

- Follow every **meta-path** from the task table
- Aggregate each **individual column** at its end (mean, min, max, mode, ...)
- Append the **aggregates as columns**, then run a TFM



Why use DFS?

- Training-free
- Yields strong performance (on RelBench) (Xu et al., 2026; Hayler et al., 2026)
- Works with **any tabular model**, so better TFMs directly give better relational predictions
- Implemented in **frameworks** such as RDBLearn (Xu et al., 2026), TabPFN-Rel (Hayler et al., 2026), OBM (Lam et al., 2017), ...

But: DFS-based methods have **two inherent limitations**.

Today:

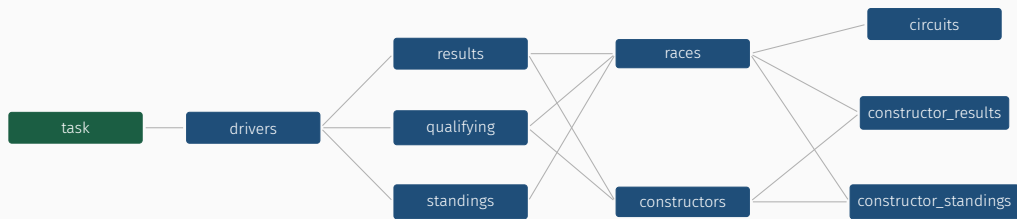
- What are those limitations?
- Can we still leverage strong TFMs for relational data without these limitations? → **RelICL**

Limitations of DFS

Limitations of DFS

Feature explosion

rel-f1 + task table: (from RelBench, Robinson et al. (2024))

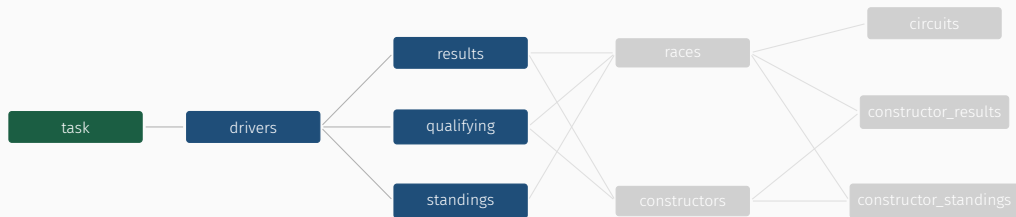


- **rel-f1**: 9 tables, 67 columns
- #DFS features $\approx$ #meta-paths $\times$ #columns $\times$ #aggregates

$\Rightarrow$ RDBLearn produces **7715** DFS features on **driver-top3** (unbounded depth)

Limitation (Feature Explosion). The number of DFS features grows with the complexity of the schema.

rel-f1 + task table:



Common hotfix: bound the depth

- Only follow meta-paths up to a fixed length (default 2 in most current tools)
- **Problem:** a signal further away is invisible

Experiment: a signal 3 hops away

Data (sketch): the label equals the **bit** in R_3

task		R_1		R_2		R_3	
label	r1_id	r1_id	r2_id	r2_id	r3_id	r3_id	bit
0	1	1	2	1	2	1	0
1	2	2	1	2	1	2	1
?	3	3	3	3	3	3	1

Experiment: a signal 3 hops away (results)

Method	AUROC
RDBLearn (depth bound 2)	50.8 ± 0.1
RelICL (TabICL, unbounded)	100.0 ± 0.0
RelICL (TabPFN, unbounded)	100.0 ± 0.0

- Test AUROC (%), mean $\pm$ SEM over three seeds
 - 1000 training rows, 300 query rows, 3 seeds
- **RDBLearn is at random guessing, RelICL solves the task**

Limitations of DFS

Interaction blindness

RDBLearn's argument

*“[...] we should **only compress along the vertical dimension within table columns** [...]. Meanwhile, the encoder should **not compress in the horizontal dimension** across table columns [...].”* (Xu et al., 2026)

Really?

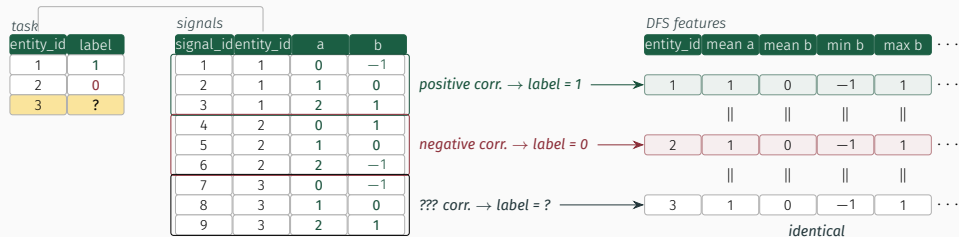
Why may this be problematic?

- Recall: Each DFS feature aggregates a **single** context-table column.
- ⇒ A signal that lies only in the interaction of columns is invisible.

Limitation (Interaction Blindness). DFS-based methods are oblivious to feature interactions.

Experiment: a signal in a column interaction

Data (sketch): each task row references three signal rows



- Label 1: a and b positively correlated; label 0: negatively
- Per column, both cases hold the same values, so every DFS feature is identical in both cases

Experiment: a signal in a column interaction (results)

Method	AUROC
RDBLearn	46.5 $\pm$ 3.3
RelICL (TabICL)	99.3 $\pm$ 0.7
RelICL (TabPFN)	99.8 $\pm$ 0.1

- *Setup*: 1 000 training rows, 300 query rows, 3 seeds; RDBLearn v1.1 in its default configuration; RelICL with TabICL and TabPFN
 - Test AUROC (%), mean $\pm$ SEM over three seeds
- **RDBLearn is at random guessing, RelICL solves the task**

RelICL

Key idea

Use the tabular foundation model not only to predict, but also to **extract** and **fuse** features.

But how?

Assumption: Row embeddings are good features

TFMs typically compute **row embeddings** internally: $\mathcal{T} = \mathcal{T}_{\text{pred}} \circ \mathcal{T}_{\text{enc}}$

- Encoder $\mathcal{T}_{\text{enc}}$: one embedding per row
- Prediction head $\mathcal{T}_{\text{pred}}$: predicts from these embeddings

Our key assumption: For a relation R ,

$$\mathcal{T}(\mathcal{T}_{\text{enc}}(R)) \approx \mathcal{T}(R).$$

Does this assumption hold?

Assumption: Row embeddings are good features (cont'd.)

Dataset	$\mathcal{T}(R)$	$\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$		
	AUROC (%)	AUROC (%)	Δ	
bioresponse	87.7 ± 1.0	87.5 ± 0.8	-0.3 ± 0.3	✓
churn	92.8 ± 0.8	92.6 ± 0.6	-0.2 ± 0.2	✓
compass	72.3 ± 0.7	72.4 ± 0.7	$+0.1 \pm 0.1$	✓
credit-g	82.2 ± 1.8	83.1 ± 0.4	$+0.9 \pm 1.4$	✓
jasmine	89.8 ± 1.1	86.8 ± 0.4	-3.0 ± 0.6	✗
kr-vs-kp	100.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	✓
philippine	93.6 ± 0.1	84.7 ± 0.6	-8.9 ± 0.6	✗
phoneme	97.6 ± 0.0	96.2 ± 0.2	-1.5 ± 0.2	✓
qsar-biodeg	92.4 ± 2.1	91.3 ± 2.4	-1.1 ± 0.4	✓
spambase	99.1 ± 0.0	98.8 ± 0.0	-0.4 ± 0.0	✓
$\emptyset$	90.8 ± 2.7	89.3 ± 2.6	-1.4 ± 0.9	

The assumption holds often (✓) but not always (✗).

Why does this assumption matter?

- Represent each context row by its **row embedding**
 - Embeddings depend on all columns of a row, so **interactions survive**
 - ⇒ No interaction blindness
- Pool the embeddings along each **foreign key**, one hop at a time
 - The number of added features depends **only on the local neighborhood**, not the whole schema
 - ⇒ No feature explosion

Overview

1. **Outward traversal:** Determine **relevant rows** and drop the rest to reduce inference cost.
2. **Inward traversal:** Determine and **transfer contextual information** hop-by-hop to the task table.
3. **Prediction:** Leverage a **strong pretrained TFM** for the final prediction.

lap times			*	1	results				*
lap	time	r_id			r_id	d_id	date	pos	
1	92.3	11			11	1	Mar 1	2	
2	91.8	11			12	2	Mar 1	9	
1	95.1	12			13	1	Mar 16	1	
2	94.6	12			14	2	Mar 16	7	
⋮	⋮	⋮			19	3	Mar 16	2	(irrelevant)
⋮	⋮	⋮			15	1	Mar 30	1	
⋮	⋮	⋮			16	2	Mar 30	8	
⋮	⋮	⋮			20	1	Apr 13	1	(future)
⋮	⋮	⋮			21	2	Apr 13	8	
⋮	⋮	⋮			⋮	⋮	⋮	⋮	
⋮	⋮	⋮			⋮	⋮	⋮	⋮	

pit stops			*
stop	dur	r_id	
1	2.4	11	
1	3.1	12	
2	2.9	12	
⋮	⋮	⋮	
⋮	⋮	⋮	
⋮	⋮	⋮	

task			
date	d_id	d_name	winner
Mar 15	1	Cathi	✓
Mar 15	2	Wim	×
Mar 29	1	Cathi	✓
Mar 29	2	Wim	×
Apr 12	1	Cathi	?
Apr 12	2	Wim	?

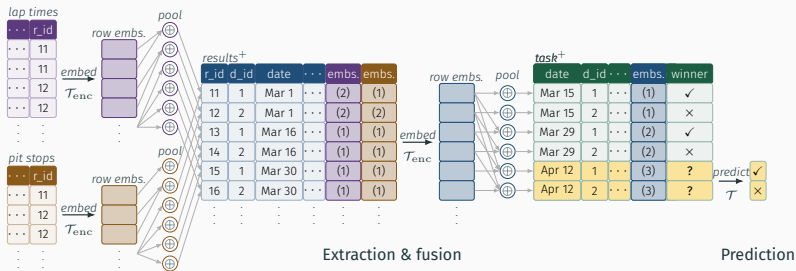
Input database (as of Apr 12)



Inference DAG (right-pointing edges) and traversal order

Inward traversal

- Walk the inference DAG in topological order
- Per table R : append pooled child embeddings and a count, then embed R^+ with $\mathcal{T}_{\text{enc}}$
- **Phase 3:** predict with the full TFM, $\mathcal{T}(T^+)$



⇒ Relies on the key assumption: row embeddings are good features

Experiments

Experimental Setup

- **Data:**
 - RelBench v2 (Gu et al., 2026): **21 entity tasks on 7 databases** (100k to 44M rows)
 - 12 binary classification (AUROC), 9 regression (MAE, R^2)
- **Baselines:**
 - DFS / pretrained tabular:
 - RDBLearn (Xu et al., 2026)
 - TabPFN-Rel (Hayler et al., 2026)
 - Pretrained relational:
 - RT (Ranjan et al., 2025)
 - PluRel (Kothapalli et al., 2026)
 - Excluded (hosted, closed-source models):
 - KumoRFM-2 (Hudovernik et al., 2026)
 - TabPFN-Rel API
- **One configuration for all 21 tasks, chosen once** by a hyperparameter search on 12 RelBench tasks; RDBLearn and TabPFN-Rel tune per task (Xu et al., 2026; Hayler et al., 2026)

Model	Pretrained (rel.)		Pretrained (tab.)		
	RT	PluRel	RDBLearn	TabPFN-Rel	RelICL
RT	–	6 / 18*	0 / 8	2 / 10	7 / 18*
PluRel	11 / 18*	–	0 / 8	2 / 10	5 / 18*
RDBLearn	8 / 8	8 / 8	–	1 / 15	5 / 15
TabPFN-Rel	8 / 10	8 / 10	13 / 15	–	10 / 21
RelICL	11 / 18*	13 / 18*	10 / 15	11 / 21	–

Cell: tasks the row model won against the column model, out of those both report.

* R^2 instead of MAE on some tasks.

RelICL is ahead on every pair.

Conclusion

Takeaways

- DFS **leverages strong TFMs** for relational prediction but **suffers from feature explosion and interaction blindness**.
- RelICL also builds on strong TFMs and **avoids both limitations** by using the TFM for feature propagation and step-by-step fusion.
- Both approaches are training-free and work with most TFMs out of the box.
- **RelICL is ahead of every baseline pair by pair** and on par with the strongest one, TabPFN-Rel, using one configuration for all tasks (while TabPFN-Rel tunes per task and adds engineered features).



[https://arxiv.org/
abs/2610.01725](https://arxiv.org/abs/2610.01725)

Appendix

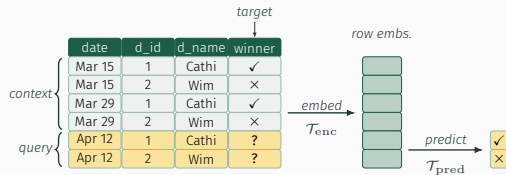
Appendix

- ICL
- RDBLearn's argument, full quote
- Full method sketch
- Regression results
- Classification results
- All models, regression
- All models, classification
- Backbones, regression
- Backbones, classification
- OpenML datasets
- RelBench databases
- RelBench classification tasks

Appendix: ICL

Tabular in-context learning (ICL)

- Input: labeled *context* rows and unlabeled *query* rows
- One forward pass with the TFM $\mathcal{T}$ predicts the query labels without training
- Here: assume $\mathcal{T} = \mathcal{T}_{\text{pred}} \circ \mathcal{T}_{\text{enc}}$:
 - Encoder $\mathcal{T}_{\text{enc}}$: one embedding per row
 - Prediction head $\mathcal{T}_{\text{pred}}$: predicts from these embeddings
- Holds for TabICL, TabPFN, TabFM, ...

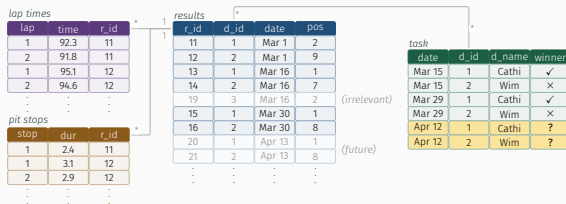


Appendix: RDBLearn's argument

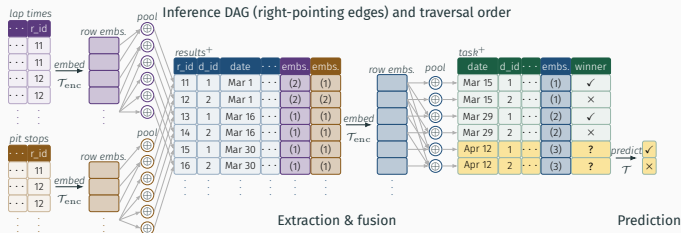
*“There is no free lunch; RDB subgraphs may be large so any encoder must compress somewhere. Our insight is that at the encoder level, prior to seeing labeled ICL samples, we should **only compress along the vertical dimension within table columns**, where shared units facilitate interpretable aggregation guided by known PK-FK relations. Meanwhile, the encoder should **not compress in the horizontal dimension across table columns**, where roles, data types, and predictive relevance are broadly different and yet indeterminate without adequate per-dataset/task label info.”*

(Xu et al., 2026)

Appendix: Full method sketch



Input database (as of Apr 12)



Appendix: regression results

Task	Pretrained (rel.)		Pretrained (tab.)		
	RT	PluRel	RDBLearn	TabPFN-Rel	RelICL
item-ltv	– / 32.5	– / 40.5	48.56 / –	47.80 / –	43.97 / 28.6
user-ltv	– / 36.9	– / 18.5	14.54 / –	14.40 / –	14.78 / 21.0
ad-ctr	– / 4.5	– / 4.9	0.03 / –	0.03 / –	0.04 / 18.3
user-attendance	–	–	–	0.24 / –	0.24 / 5.1
driver-position	– / 52.4	– / 55.5	–	3.76 / –	3.68 / 25.2
item-sales	– / 14.0	– / 20.0	0.06 / –	0.06 / –	0.04 / 69.2
post-votes	– / 33.9	– / 25.5	0.07 / –	0.07 / –	0.06 / 25.6
site-success	– / 4.5	– / 38.6	0.42 / –	0.39 / –	0.40 / -22.0
study-adverse	– / 2.6	– / 1.6	43.91 / –	42.60 / –	32.42 / 54.0

Appendix: classification results

Task	Pretrained (rel.)		Pretrained (tab.)		
	RT	PluRel	RDBLearn	TabPFN-Rel	RelICL
item-churn	70.9	72.5	82.1	82.8	78.5
user-churn (ama)	64.0	65.0	67.6	70.2	65.4
user-clicks	59.5	47.9	69.0	61.5	64.3
user-visits	61.8	63.4	65.5	66.9	66.2
user-ignore	–	–	–	70.1	88.5
user-repeat	–	–	–	76.9	79.3
driver-dnf	81.2	81.0	–	71.5	73.6
driver-top3	89.3	88.4	–	79.3	81.4
user-churn (hm)	62.8	66.0	68.0	70.6	68.4
user-badge	80.1	82.0	85.3	86.4	86.3
user-engagement	75.7	86.2	89.4	90.6	90.0
study-outcome	51.8	51.8	71.6	73.1	73.4

Appendix: all models, regression

Kind	Model	item-ltv	user-ltv	ad-ctr	user-attendance	driver-position	item-sales	post-votes	site-success	study-adverse
Trained	GraphSAGE	50.05 / -	14.31 / -	0.04 / -	0.26 / -	4.02 / -	0.06 / -	0.07 / -	0.40 / -	44.47 / -
Trained	RelGNN	48.77 / -	14.23 / -	0.04 / -	0.24 / -	3.80 / -	0.05 / -	0.07 / -	0.30 / -	44.46 / -
Trained	RelGT	48.92 / -	14.27 / -	0.04 / -	0.25 / -	3.92 / -	0.05 / -	0.07 / -	0.33 / -	43.99 / -
Trained	DS+LightGBM	41.12 / -	13.93 / -	0.04 / -	0.28 / -	3.96 / -	0.04 / -	0.07 / -	0.41 / -	40.58 / -
Pretrained (rel.)	RT	- / 32.5	- / 36.9	- / 4.5	-	- / 52.4	- / 14.0	- / 33.9	- / 4.5	- / 2.6
Pretrained (rel.)	RT (target DB)	- / 32.2	- / 38.3	- / 8.0	-	- / 58.7	- / 30.9	- / 35.0	- / 5.1	- / 3.1
Pretrained (rel.)	PluRel	- / 40.5	- / 18.5	- / 4.9	-	- / 55.5	- / 20.0	- / 25.5	- / 38.6	- / 1.6
Pretrained (rel.)	KumoRFM-2	46.99 / -	13.92 / -	0.03 / -	0.23 / -	2.85 / -	0.03 / -	0.06 / -	0.43 / -	43.29 / -
Pretrained (tab.)	RDBLearn	48.56 / -	14.54 / -	0.03 / -	-	-	0.06 / -	0.07 / -	0.42 / -	43.91 / -
Pretrained (tab.)	TabPFN-Rel (API)	46.80 / -	14.40 / -	0.03 / -	0.24 / -	3.77 / -	0.06 / -	0.07 / -	0.41 / -	39.80 / -
Pretrained (tab.)	TabPFN-Rel	47.80 / -	14.40 / -	0.03 / -	0.24 / -	3.76 / -	0.06 / -	0.07 / -	0.39 / -	42.60 / -
Pretrained (tab.)	RelICL (TabICL)	43.97 / 28.6	14.78 / 21.0	0.04 / 18.3	0.24 / 5.1	3.68 / 25.2	0.04 / 69.2	0.06 / 25.6	0.40 / -22.0	32.42 / 54.0
Pretrained (tab.)	RelICL (TabPFN)	43.41 / 29.2	15.48 / 16.5	0.05 / 0.1	0.30 / 17.2	4.00 / 13.4	0.04 / 50.1	0.08 / 17.0	0.43 / -7.1	29.97 / 56.9

Appendix: all models, classification

Kind	Model	item-churn	user-churn (ama)	user-clicks	user-visits	user-ignore	user-repeat	driver-dnf	driver-top3	user-churn (hm)	user-badge	user-engagement	study-outcome
Trained	GraphSAGE	82.8	70.4	65.9	66.2	81.6	76.9	72.6	75.5	69.9	88.9	90.6	68.6
Trained	RelGNN	82.6	71.0	68.2	66.2	86.2	79.6	75.3	85.7	70.9	89.0	90.8	71.2
Trained	RelGT	82.5	70.4	68.3	66.8	81.6	76.1	75.9	83.5	69.3	86.3	90.5	68.6
Trained	DS+LightGBM	81.8	67.6	64.3	64.5	84.2	75.4	69.8	82.4	69.0	86.2	90.3	72.0
Pretrained (rel.)	RT	70.9	64.0	59.5	61.8	–	–	81.2	89.3	62.8	80.1	75.7	51.8
Pretrained (rel.)	PluRel	72.5	65.0	47.9	63.4	–	–	81.0	88.4	66.0	82.0	86.2	51.8
Pretrained (rel.)	KumoRFM-2	82.2	69.1	67.4	69.4	90.8	81.7	84.6	92.2	69.3	87.2	89.4	72.0
Pretrained (tab.)	RDBLearn	82.1	67.6	69.0	65.5	–	–	–	–	68.0	85.3	89.4	71.6
Pretrained (tab.)	TabPFN-Rel (API)	82.8	70.9	67.5	66.8	87.9	75.9	73.2	77.1	70.5	88.0	90.6	76.5
Pretrained (tab.)	TabPFN-Rel	82.8	70.2	61.5	66.9	70.1	76.9	71.5	79.3	70.6	86.4	90.6	73.1
Pretrained (tab.)	RelICL (TabICL)	78.5	65.4	64.3	66.2	88.5	79.3	73.6	81.4	68.4	86.3	90.0	73.4
Pretrained (tab.)	RelICL (TabPFN)	78.4	64.6	58.7	65.7	87.9	77.7	74.2	78.4	67.8	84.9	89.5	73.6

Appendix: backbones, regression

Task	MAE (↓)	
	RelICL (TabICL)	RelICL (TabPFN)
item-ltv	43.97	43.41
user-ltv	14.78	15.48
ad-ctr	0.04	0.05
user-attendance	0.24	0.30
driver-position	3.68	4.00
item-sales	0.04	0.04
post-votes	0.06	0.08
site-success	0.40	0.43
study-adverse	32.42	29.97

- Same configuration, TabPFN v3 instead of TabICL v2
- **RelICL works with another backbone too**
- Classification (next slide): same picture

Appendix: backbones, classification

Task	AUROC (%)	
	RelICL (TabICL)	RelICL (TabPFN)
item-churn	78.5	78.4
user-churn (ama)	65.4	64.6
user-clicks	64.3	58.7
user-visits	66.2	65.7
user-ignore	88.5	87.9
user-repeat	79.3	77.7
driver-dnf	73.6	74.2
driver-top3	81.4	78.4
user-churn (hm)	68.4	67.8
user-badge	86.3	84.9
user-engagement	90.0	89.5
study-outcome	73.4	73.6

Appendix: OpenML datasets

Dataset	OpenML ID	Rows	Features	Positive class (%)
bioresponse	4134	3751	1776	54.2
churn	40701	5000	20	14.1
compass	42193	5278	13	47.0
credit-g	31	1000	20	70.0
jasmine	41143	2984	144	50.0
kr-vs-kp	3	3196	36	52.2
philippine	41145	5832	308	50.0
phoneme	1489	5404	5	29.3
qsar-biodeg	1494	1055	41	33.7
spambase	44	4601	57	39.4

Appendix: RelBench databases

Dataset	Tables	Rows	Columns
rel-amazon	3	23,218,245	15
rel-avito	8	24,653,915	42
rel-event	5	44,822,992	131
rel-f1	9	97,606	67
rel-hm	3	16,931,173	37
rel-stack	7	5,399,818	51
rel-trial	15	5,852,157	140

Appendix: RelBench classification tasks

Dataset	Task	Train	Val	Test	Pos. rate (%)
rel-amazon	item-churn	2,536,014	177,689	166,842	36.9
rel-amazon	user-churn	4,708,383	409,792	351,885	60.6
rel-avito	user-clicks	59,454	21,183	47,996	1.5
rel-avito	user-visits	86,619	29,979	36,129	85.1
rel-event	user-ignore	19,239	2,013	1,958	13.0
rel-event	user-repeat	3,842	268	246	44.7
rel-f1	driver-dnf	11,411	566	702	70.5
rel-f1	driver-top3	1,353	588	726	17.6
rel-hm	user-churn	3,832,692	76,556	74,575	82.6
rel-stack	user-badge	3,386,276	247,398	255,360	2.6
rel-stack	user-engagement	1,360,850	85,838	88,137	2.7
rel-trial	study-outcome	11,994	960	825	58.5

Appendix: RelBench regression tasks

Dataset	Task	Train	Val	Test	Target (mean $\pm$ SD)
rel-amazon	item-ltv	2,707,679	166,978	178,334	77.13 $\pm$ 664.61
rel-amazon	user-ltv	4,708,383	409,792	351,885	16.78 $\pm$ 55.44
rel-avito	ad-ctr	5,100	1,766	1,816	0.05 $\pm$ 0.11
rel-event	user-attendance	19,239	2,013	1,958	0.26 $\pm$ 0.64
rel-f1	driver-position	7,453	499	760	11.93 $\pm$ 5.21
rel-hm	item-sales	5,488,184	105,542	105,542	0.08 $\pm$ 0.58
rel-stack	post-votes	2,453,921	156,216	160,903	0.07 $\pm$ 0.37
rel-trial	site-success	151,407	19,740	22,617	0.46 $\pm$ 0.48
rel-trial	study-adverse	43,335	3,596	3,098	57.93 $\pm$ 249.73

References

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite Message Passing for Relational Deep Learning. In *Forty-Second International Conference on Machine Learning*, June 2025.
- 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*, October 2025.