# TabPFN-3.5: Technical Report

Prior Labs Team (see Appendix A for the list of contributors)

We introduce **TabPFN-3.5**, our new flagship Tabular Foundation Model. It significantly outperforms its predecessor, TabPFN-3 [1], and all existing baselines across a broad range of tabular problems. TabPFN-3.5 sets a new state of the art on standard tabular prediction in TabArena [2], and extends it to the data practitioners encounter in practice: non-i.i.d. data with temporal or grouped splits [3], tables with strings [4], text and images [5], high-cardinality categorical features, and wide tables with many features [3]. These gains carry over to our model task-specific harnesses: state of the art on relational data [6] and stronger time-series forecasting [7]. For faster inference, our variant **TabPFN-3.5-Fast** runs up to 3× faster than TabPFN-3 while keeping most of the accuracy gains. In addition, we upgrade **TabPFN-3.5-Plus**, expanding our multimodal capabilities with advanced text and date handling alongside proprietary inference optimizations. Finally, we release a new version of our Thinking mode, **TabPFN-3.5-Thinking**, which scales inference-time computation to push the state of the art further. It benefits from our stronger base model and from inference-time improvements that make it up to 12× faster than TabPFN-3-Thinking.

Date September 14th 2026

License Code: Apache 2.0; weights: TABPFN-3.5 License v1.0 (see Section 4)

Docs https://docs.priorlabs.ai

**TabPFN-3.5: state of the art on all 7 benchmarks**

| Benchmark | Rank | Win rate against best other TFM | | Win rate against best non-TFM | |
|---|---|---|---|---|---|
| TabArena | 1 of 89 | TabFM+ | 57% | RealMLP | 96% |
| BeyondArena | 1 of 29 | TabICLv2 | 83% | RealMLP | 79% |
| STRABLE | 1 of 197 | TabICLv2 | 98% | XGBoost | 98% |
| MulTaBench | 1 of 14 | TabICLv2 | 90% | TabM | 95% |
| RelArena-$\alpha$ | 1 of 12 | RT-PluRel | 57% | KurveRSC | 71% |
| TALENT | 1 of 37 | TabICLv2 | 80% | CatBoost | 91% |
| ScoringBench | 1 of 52 | EXAONE | 78% | RealMLP | 91% |
| Mean | 1 | | 78% | | 89% |

**Table 1. TabPFN-3.5 against the strongest non-TabPFN competitor on each benchmark.** Rank is the position of the best TabPFN-3.5 member (TabPFN-3.5-Thinking on the first four benchmarks, TabPFN-Rel (3.5) on RelArena-$\alpha$, TabPFN-3.5 on TALENT and ScoringBench) among the methods on the benchmark's leaderboard. Win rates are the share of splits won against the best other Tabular Foundation Model (TFM) and the best non-foundation model, datasets weighted equally. Details in Section C.8.

# Contents

**1 Introduction** **3**

**2 Experimental Results** **3**

2.1 TabArena . . . . . 4

2.2 BeyondArena . . . . . 5

2.3 STRABLE . . . . . 6

2.4 MulTaBench . . . . . 6

2.5 RelArena-$\alpha$ . . . . . 7

2.6 Inference benchmarks . . . . . 7

**3 Model Design and Inference** **8**

3.1 Architecture . . . . . 8

3.2 Preprocessing . . . . . 9

3.3 Synthetic Prior . . . . . 11

3.4 TabPFN-3.5-Plus and Thinking mode . . . . . 11

**4 License and Availability** **11**

4.1 Distribution . . . . . 12

**A Contributors** **16**

**B Acknowledgements** **17**

**C More Experimental Results** **17**

C.1 Additional details on TabArena . . . . . 17

C.2 Additional results on BeyondArena . . . . . 25

C.3 Additional Results on STRABLE . . . . . 26

C.4 Additional Results on MulTaBench . . . . . 28

C.5 TALENT . . . . . 28

C.6 fev-bench . . . . . 30

C.7 ScoringBench . . . . . 31

C.8 Details on the benchmark overview table . . . . . 32

# 1 Introduction

Tabular Foundation Models (TFMs) have replaced gradient-boosted trees, for decades the default choice for tabular prediction that drives decision making across science and industry [8–13], as the strongest predictors on standard i.i.d. benchmarks: on TabArena [2], TFMs dominate the leaderboard, and the strongest default TFM outperforms AutoML systems that tune and ensemble models for hours [2, 14]. Successive TabPFN releases [1, 15–17] established and extended this paradigm, from a thousand clean numerical rows [15] to a million rows with heterogeneous features (categorical and numerical), missing values, and many-class targets [1, 16, 17]. Around these models, an active ecosystem has grown across time-series forecasting [18, 19], causal inference [20–22], graph learning [23, 24], relational data [6], interpretability [25, 26], survival analysis [27, 28], Bayesian optimization [29, 30], and reinforcement learning [31], with hundreds of published applications and millions of downloads; the TabPFN-3 report [1] surveys both. The field is maturing: within the last year, TabICLv2 [32], TabFM [33], EXAONE-Tabular [34], TabDPT-Turbo [35], and SAP-RPT-1 [36] have been released as competitive TFMs, and benchmarks have broadened from numerical i.i.d. data to non-i.i.d. [3, 37], string-valued [4], multimodal [5], and relational [6] data, and to calibrated predictive distributions [38]. These new benchmarks also show where TFMs still fall short. Purucker et al. [3] show that TFMs excel on clean i.i.d. data, yet even the best of them fall behind tuned conventional models on grouped or temporal splits, on tables with many rows or many features, and on tables with text or high-cardinality features. This is the data practitioners face every day. And the TFMs that have pushed accuracy furthest have also grown larger and slower, which makes them harder to deploy.

To address these open problems, we introduce TabPFN-3.5. It extends the lead of TFMs from clean i.i.d. benchmarks to all of these regimes, on which it is now competitive with heavily tuned gradient-boosted trees and MLPs. We observe these gains across benchmarks: TabPFN-3.5 sets a new state of the art on TabArena [2], BeyondArena [3], STRABLE [4] for tabular data with strings, MulTaBench [5] for multimodal tabular learning with text and images, ScoringBench [38] for predictive distributions, and, through TabPFN-Rel, RelArena [6] for relational data. At the same time, it retains what made TabPFN-3 practical: training sets of up to one million rows, many-class classification, calibrated predictive distributions from a single forward pass, and cached inference in which the training rows are processed once and test rows are then scored in well under a second. TabPFN-3.5's gains come from revised per-cell encodings, doubling the model width while keeping the cache size unchanged, a single checkpoint trained jointly for classification and regression, and a synthetic prior tuned towards high-cardinality, wide, and grouped data. Section 2 quantifies these gains.

We release two checkpoints compatible with the open-source `tabpfn` package: TabPFN-3.5 as the core model and TabPFN-3.5-Fast, an alpha checkpoint for latency-sensitive applications that runs up to 6× faster than TabPFN-3.5 (and up to 140× faster than TabFM) with a small drop in accuracy while still matching the previous state of the art on these benchmarks. The core model was designed to serve as the foundation of TabPFN-3.5-Plus, which expands our multimodal capabilities for text and date processing alongside proprietary inference optimizations, as well as TabPFN-3.5-Thinking, which also scales inference-time computation to deliver peak performance. TabPFN-3.5-Thinking is our most accurate model across all use cases and further strengthens grouped and temporal modeling. TabPFN-3.5-Plus, TabPFN-3.5-Fast, and TabPFN-3.5-Thinking are available through the Prior Labs API.

The remainder of this report presents the experimental results (Section 2); describes the architecture, preprocessing, inference characteristics, and synthetic prior of TabPFN-3.5, as well as TabPFN-3.5-Plus and its Thinking mode (Section 3); and details licensing (Section 4). For installation and usage, see our documentation.

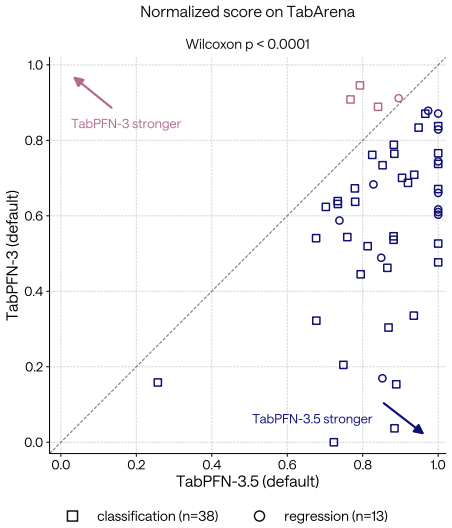
# 2 Experimental Results

TabPFN-3.5 improves performance across a wide range of settings:

- **Standard tabular benchmarks.** On TabArena, TabPFN-3.5 ranks first, and together with TabPFN-3.5-Fast and TabPFN-3.5-Thinking dominates the Elo-versus-time Pareto frontier (Section 2.1). It also ranks first on the 300-dataset TALENT benchmark (Section C.5).
- **Diverse and non-i.i.d. data.** On BeyondArena, which spans grouped and temporal splits, tiny

| Model | Rows | Features | Parameters | |
|---|---|---|---|---|
| | | | Clf. | Reg. |
| TabPFN-v1 | 1,000 | 100 | 26 M | — |
| TabPFN-v2 | 10,000 | 500 | 7 M | 11 M |
| TabPFN-2.5 | 100,000 | 2,000 | 11 M | 10 M |
| TabPFN-2.6 | 100,000 | 2,000 | 11 M | 13 M |
| TabPFN-3 | 1,000,000 | 2,000 | 53 M | 58 M |
| TabPFN-3.5 | 1,000,000 | 6,000 | 220 M | |
| TabPFN-3.5-Fast | 1,000,000 | 6,000 | 84 M | |

**(a) Overview of TabPFN releases**, with the largest recommended row and feature counts, and the models' parameter counts for each release. These are recommended limits for strong performance, not computational limits. Increasing the number of estimators allows support for up to 20K features.




**(b) TabPFN-3.5 substantially improves on TabPFN-3.** We report the per-dataset scores on TabArena [2]. The normalized score is based on a per-fold min-max normalization as explained in [1].

**Figure 1. Evolution and performance of the TabPFN model family.** (a) Supported input shapes and model size across releases. TabPFN-3.5 raises the recommended feature limit and can simultaneously handle 1M data points and 6k features. (b) Per-dataset scores on TabArena. Points below the diagonal indicate stronger TabPFN-3.5 performance; a Wilcoxon signed-rank test confirms the improvement is significant ($p < 0.0001$).

to million-row tables, and text and high-cardinality features, TabPFN-3.5 ranks first overall, and for every subset of non-large (<100K rows) datasets (Section 2.2).

- **Multimodal data.** TabPFN-3.5 ranks first on STRABLE, where tables carry string fields such as names, addresses and free-text notes (Section 2.3), and on MulTaBench, where tabular columns are paired with text and image fields (Section 2.4). TabPFN-3.5-Plus, which processes text columns natively, and TabPFN-3.5-Thinking extend the lead further on both benchmarks (Section 3.4).
- **Predictive distributions.** Beyond point predictions, TabPFN-3.5 produces better predictive distributions, ranking first on ScoringBench, which scores the full predicted distribution of regression models with proper scoring rules (Section C.7).
- **Task-specific harnesses.** The gains carry over to task-specific harnesses: TabPFN-Rel built on TabPFN-3.5-Plus ranks first on RelArena-$\alpha$ for relational data (Section 2.5), and the general TabPFN-3.5 checkpoint, with no time-series finetuning, beats the time-series-specific TabPFN-TS-3 on fev-bench (Section C.6).

Together, TabPFN-3.5, TabPFN-3.5-Fast and TabPFN-3.5-Thinking improve on TabPFN-3 at every time budget. TabPFN-3.5 itself runs up to 2× slower than TabPFN-3 on large training sets because of its doubled model width; TabPFN-3.5-Fast is up to 3× faster than TabPFN-3 while still about 150 Elo points ahead on TabArena. See Section 2.6 for inference-speed measurements and details on how our TabPFN-3.5-Plus offering makes TabPFN-3.5 and TabPFN-3.5-Fast even faster.

## 2.1 TabArena

TabArena [2] has become increasingly competitive with the arrival of strong tabular foundation models such as TabFM [33] and EXAONE-Tabular [34]. Indeed, every method in the current top ten is either a tabular foundation model in its default configuration or a highly optimized AutoML system.

Figure 2 shows the Pareto frontier for Elo and total fit-and-predict time; Figure 13 shows the corresponding improvability frontier. Against a strong set of recent baselines, TabPFN-3.5-Thinking, TabPFN-3.5 and TabPFN-3.5-Fast form the Pareto frontier: at every time budget one of them reaches the highest Elo. TabPFN-3.5 beats TabFM by 84 Elo points using one-tenth of its total time per sample, and outperforms

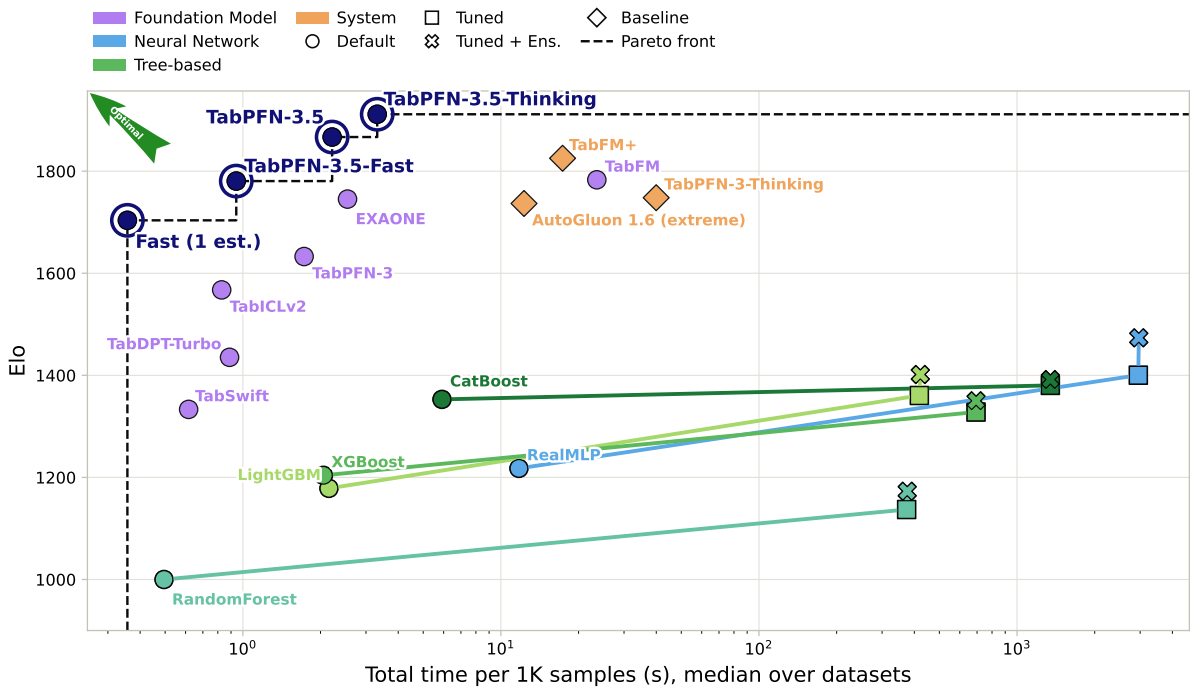



**Figure 2. TabArena Elo against total time.** Total time is the fit time per 1,000 training rows plus the prediction time per 1,000 test rows, median over the 51 datasets, on a log scale; Section C.1.4 describes the measurement.

AutoGluon 1.6 extreme [14] by 130 Elo points in a fifth of the time. Section C.1.4 describes how the time axis is measured.

## 2.2 BeyondArena

BeyondArena [3] tests whether tabular models generalize beyond the small, IID settings emphasized by standard benchmarks. It includes datasets with different training and test distributions, spanning grouped splits (predicting on data from a subgroup unseen during training, e.g. a new country or a new hospital), and temporal splits (predicting on test samples observed later in time than the training samples). BeyondArena also contains tiny to million-row datasets; low and high-dimensional tables; and text and high-cardinality features. Earlier TFMs performed strongly on small IID datasets but remained behind tuned conventional models in several of these harder regimes.

TabPFN-3.5 establishes a new state of the art on the full BeyondArena core (Figure 3). It ranks first overall and improves over TabPFN-3 in every slice, with particularly strong results on IID, high-dimensional, high-cardinality, and text data. The confidence intervals are wide for some of the smaller slices, so their exact ordering should be interpreted cautiously. Tuned and ensembled MLPs retain the highest performance on grouped, temporal, and large datasets, but TabPFN-3.5 substantially narrows these gaps. The temporal (and to a lesser extent the grouped) subset of BeyondArena is confounded with size. When restricted to BeyondArena non-large (datasets with up to 100K rows), TabPFN-3.5 matches the best baselines for grouped and temporal datasets. See Appendix C.2.3 for more details on how we evaluate models on grouped datasets, which differs slightly from the current BeyondArena procedure.

In Appendix C.2.2, we compare to more recent TFMs that are absent from the official BeyondArena results and cannot run on the full benchmark. This comparison confirms its lead on the evaluated subset.

Finally, we also show the results of TabPFN-3.5-Thinking and TabPFN-3.5-Fast on BeyondArena in Figure 3. TabPFN-3.5-Thinking improves on TabPFN-3.5 by 20 Elo points, while TabPFN-3.5-Fast is 3x faster and scores only 40 Elo points lower.

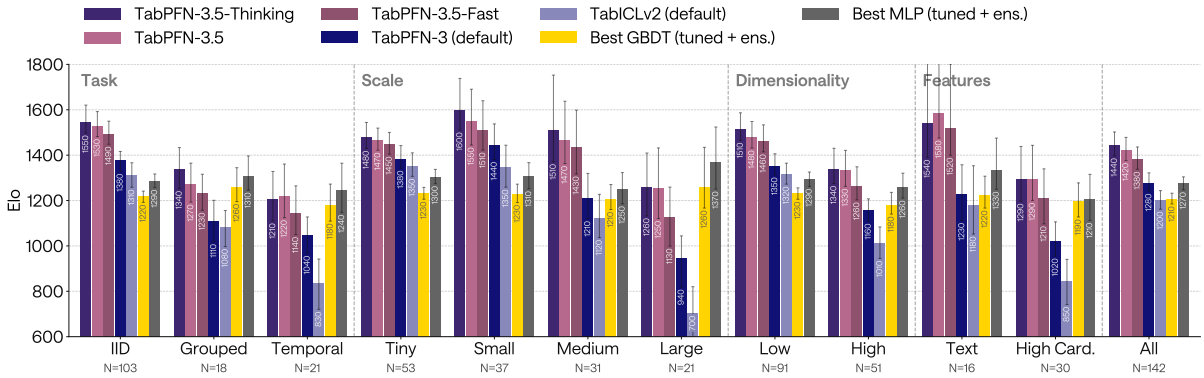



**Figure 3. Performance on BeyondArena core.** Elo scores compare TabPFN-3.5 with TabPFN-3, TabICLv2, and the strongest tuned and ensembled GBDT and MLP baselines, overall and by data slice. Error bars show bootstrap confidence intervals. The labels of the Elo scores are rounded to the nearest multiple of 10. The group identifier is exposed to the TabPFN models on label-per-sample grouped datasets; see Section C.2.3.

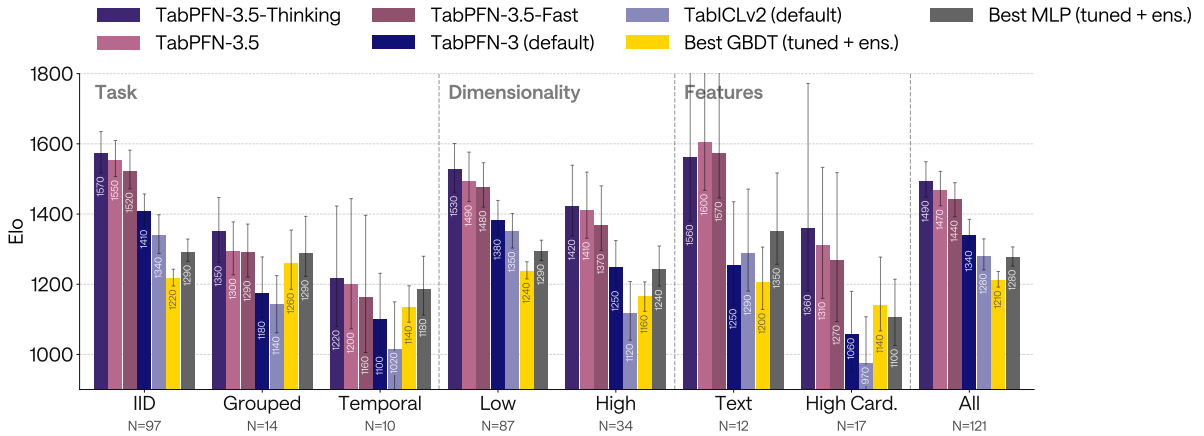



**Figure 4. BeyondArena without large tables.** TabPFN-3.5 leads overall and across the reported data regimes, while matching the strongest baseline on grouped data. Error bars show bootstrap confidence intervals. The labels of the Elo scores are rounded to the nearest multiple of 10. The group identifier is exposed to the TabPFN models on label-per-sample grouped datasets; see Section C.2.3.

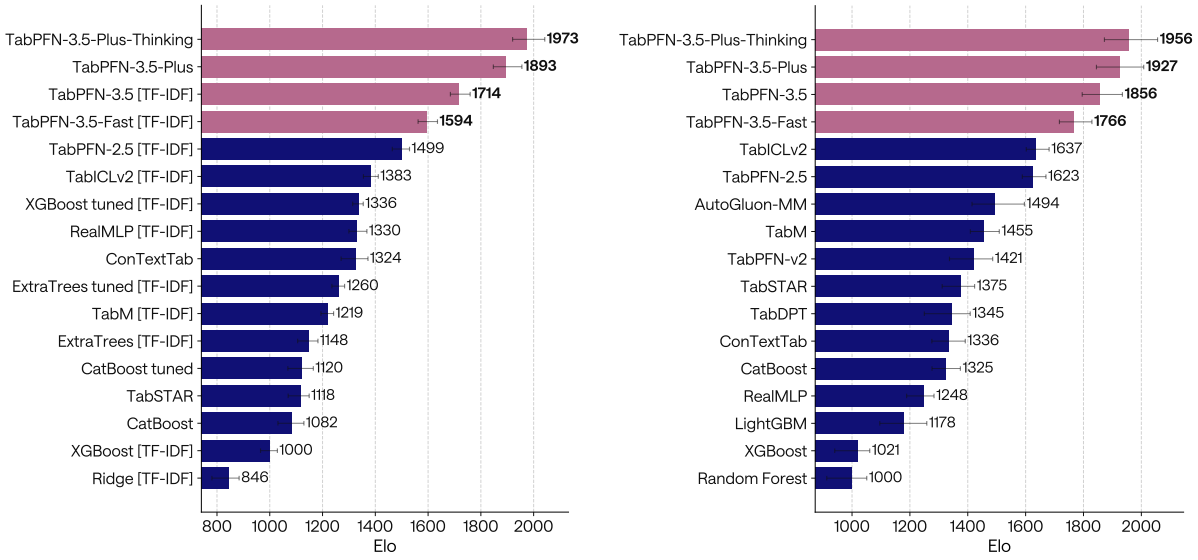
## 2.3 STRABLE

Enterprise tables frequently contain product names, addresses, job titles, and identifiers alongside numeric columns. STRABLE [4] is the largest benchmark of tabular learning with strings, comprising 108 datasets drawn from diverse application fields. We compare against the results published in the official paper, without re-running baselines; consequently, models released after the paper are absent from the comparison. Since TF-IDF was the strongest string encoding for STRABLE, we use it for models without native text handling, including TabPFN-3.5 and TabPFN-3.5-Fast; models with native text handling [36, 39, 40], as well as TabPFN-3.5-Plus and TabPFN-3.5-Thinking, receive the raw table.

Figure 5a shows that the TabPFN-3.5 family dominates the benchmark, with all variants significantly outperforming prior baselines. Appendix C.3 shows the leaderboard including variants using the embedding model *e5-small-v2* [41] for string encoding.

## 2.4 MulTaBench

In many high-impact domains, tabular learning problems are inherently multimodal, combining structured numerical features with unstructured inputs like free-text and images. MulTaBench [5] benchmarks this setting across 40 datasets: 20 with an image column and 20 with free-text columns. We compare against the results published in the official paper without re-running baselines; consequently, models released




**(a)** STRABLE with TF-IDF: 108 datasets, 3 folds each. **(b)** MulTaBench: 40 datasets, 5 folds each.

**Figure 5. Elo on STRABLE and MulTaBench**, over the results published with each benchmark. The TabPFN-3.5 family significantly outperforms all published baselines on both.

after the paper are absent from the comparison. In the official benchmark, baselines lacking native text support use frozen *e5-small-v2* [41] embeddings, and baselines lacking image support rely on frozen *DINOv3-small* [42] embeddings. Both of them are then reduced to 30 PCA [43] components. To ensure a fair comparison, we evaluate TabPFN-3.5 and TabPFN-3.5-Fast using the same preprocessing.

Figure 5b also includes TabPFN-3.5-Plus and TabPFN-3.5-Thinking, which process the raw text columns directly (Section 3.4) while relying on the frozen embeddings for image features. As shown in the figure, the TabPFN-3.5 family significantly outperforms all prior published methods. Appendix C.4 details the text-only half of the benchmark, where the performance gap between our models and the baselines widens further, thanks to our improved text support.

## 2.5 RelArena-$\alpha$

As observed in previous work [1], improvements in the base model tend to translate directly into stronger downstream performance on relational tasks through the TabPFN-Rel harness [1, 6]. Figure 6 illustrates this on the novel RelArena-$\alpha$ benchmark [6], which enables reproducible and comparable evaluation on RelBench v1 entity-level tasks [44]. Using TabPFN-3.5-Plus, TabPFN-Rel ranks first on both the models and model+systems leaderboards of RelArena-$\alpha$, outperforming the RT-PluRel system [45–47] while requiring significantly less runtime.

Building on these results, we continue to co-develop the TabPFN-Rel harness alongside the TabPFN base models. We additionally preview the latest internal version of the harness, which, when combined with TabPFN-3.5-Plus, improves performance by a further 36 Elo points, as also shown in Figure 6. We co-developed the newer versions of the TabPFN-Rel harness with TabPFN-3.5-Plus. Results with open-source checkpoints and with Thinking mode are work in progress.

## 2.6 Inference benchmarks

We benchmark our new models TabPFN-3.5 and TabPFN-3.5-Fast against TabPFN-3 on one NVIDIA RTX PRO 6000 Blackwell (96 GB). We run a single estimator, measure the model forward time, and multiply by the checkpoint's default number of estimators (8 for TabPFN-3 and TabPFN-3.5, 4 for TabPFN-3.5-Fast), see Figure 7. As the number of training rows grows from 1k to 1M, TabPFN-3.5 runs up to 2× slower than TabPFN-3 due to the increased number of attention heads. Our optimized TabPFN-3.5-Plus, however, reduces the model's latency and only has 1.25× the latency of TabPFN-3 due to faster FP8 attention [48]. TabPFN-3.5-Fast is up to 3 times faster than TabPFN-3 and the optimized

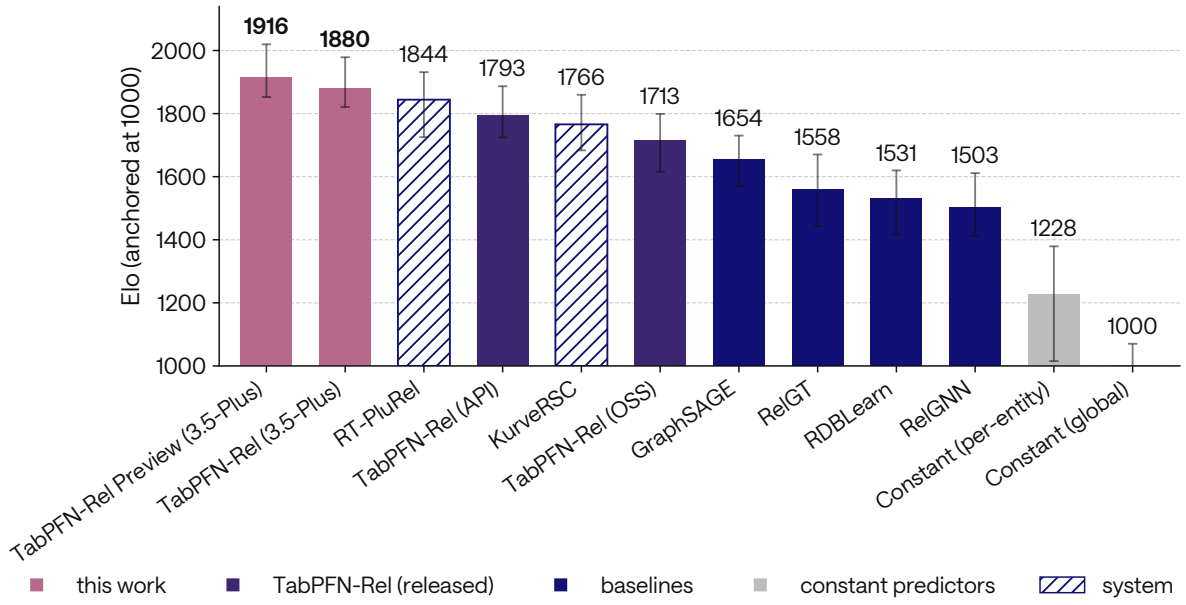



**Figure 6. Elo scores on the model + system leaderboard over 21 RelArena-$\alpha$ tasks:** Entries marked as systems (hatched) comply with RelArena-$\alpha$'s data states and evaluation regime but not with its standardized tuning regime (see Hayler et al. [6] for further details). We observe that updating the base model from TabPFN-3-Plus to TabPFN-3.5-Plus improves performance by nearly 90 Elo points. We also preview our latest (internal) version of TabPFN-Rel, co-developed with TabPFN-3.5, which yields an additional 36 Elo-point improvement.

Plus variant can run in a quarter of the time. All scale quadratically in the number of training rows for datasets larger than 128k rows.

Like our previous models, TabPFN-3.5 also supports cached inference for sub-second predictions. The expensive computations on the training rows are done only once and only the relevant tensors are cached. The models cache the per-block inducing states produced by the feature distribution embedder, the keys and values of the ICL self-attention blocks, and the inputs to the many-class decoder. For the latter, TabPFN-3 caches the final ICL layer's train embeddings. We optimized this and TabPFN-3.5 directly caches the input keys of the many-class decoder, reducing cache size as well as computation. The only additional new structure that needs caching for TabPFN-3.5 is for the ECDF features, for which we use a bucketed version to decouple its cache size from the number of training rows.

Overall, the size of the cache remains largely unchanged between TabPFN-3 and TabPFN-3.5 even though the parameter count roughly quadrupled and prediction quality improved substantially. This is crucial for online inference, where a single test example is scored against the training context. This online use-case is bound by the memory bandwidth of moving the cache (and the weights). As we see in Figure 7 for a single test row, the cached predict times between TabPFN-3.5 and TabPFN-3 align exactly, whereas TabPFN-3.5-Fast is around three times faster.

# 3 Model Design and Inference

This section describes the changes from TabPFN-3 to TabPFN-3.5: revised per-cell encodings, an architecture optimized for joint multitask training, and a doubled model width (Section 3.1), simpler preprocessing (Section 3.2), and a synthetic prior tuned to high-cardinality, wide, and grouped data (Section 3.3). Section 3.4 introduces TabPFN-3.5-Plus and its Thinking mode.

## 3.1 Architecture

The architecture of TabPFN-3.5 retains TabPFN-3 's [1] overall design while revising several key components, summarized below.

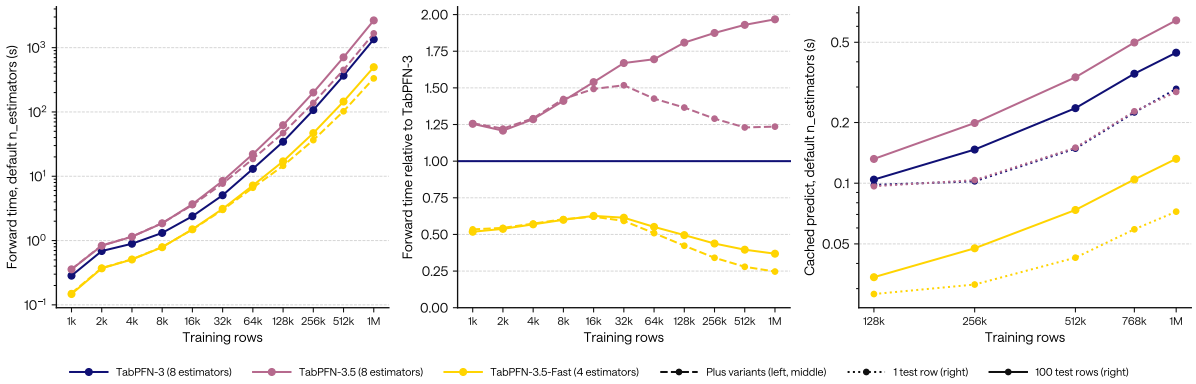



**Figure 7.** Inference speed at 100 columns and 1,024 test rows absolute (left) and relative to TabPFN-3 (middle). Dashed lines are the same checkpoints behind TabPFN-Plus. Right: predicting 1 or 100 test rows against a KV cache built over the training rows. On a single test row, TabPFN-3.5 matches the speed of TabPFN-3.

**Per-cell Encodings.** Cells of the table are embedded in three stages: per-cell preprocessing, feature grouping, and group embedding (Figure 8). We inherit the missing/infiniteness indicators and the feature grouping from TabPFN-3 and add two more signals: first, the standardized values are encoded with Fourier features, inspired by TabFM [33], rather than projected only linearly: each value is multiplied by a bank of learned frequencies and passed through sine and cosine, which captures small differences better than a linear projection of the raw number. This is particularly advantageous for modeling ordinal-encoded categorical variables with high cardinality. Second, we introduce in-context *Empirical Cumulative Distribution Function (ECDF)* features: each cell is ranked against the cell values of the training rows of its own column, and the resulting $u \in [0, 1]$ is expanded into low-frequency sine and cosine terms. This places a value in its column's distribution rather than on its raw scale: midranks are unchanged by any strictly increasing transformation of a column, so a skewed feature and its logarithm yield identical ECDF terms, and a heavy tail that standard scaling crushes against the $\pm 100$ clip is spread evenly over $[0, 1]$. As in TabPFN-3, the standardized values are also passed through, so the exact magnitudes survive alongside the two new encodings. The target embedding is unchanged from TabPFN-3: a learned embedding per class and a linear projection for regression targets.

**Multitask Model.** We modified our model to be trained jointly on classification and regression, with the task type supplied as an input. Only the label encoder and the output head are task-specific; the cell encoder, per-column distribution embedder, feature aggregator and in-context transformer are shared. Classification is read out by attention over the one-hot training labels, regression by a predicted distribution over fixed bins. One run therefore yields a single checkpoint covering both tasks, reducing training cost and simplifying the inference stack. To keep training stable across the two tasks and the larger model width, we also added some new normalization layers throughout the network. Specifically, we added QK-norm (an RMSNorm on queries and keys) in the attention blocks, a LayerNorm after the input encoding, and an RMSNorm before the task-specific head.

**Increased Capacity.** We widen the in-context transformer from 512 to 1024 dimensions compared to TabPFN-3 by using eight CLS aggregation tokens per row instead of four. The attention head count doubles with it, from 8 to 16, so the per-head dimension stays at 64, and test rows still attend through a single 64-dimensional key/value head. Doubling the dimension quadruples the parameter count in the linear layers, which dominate the overall parameter count. However, for large datasets (many rows), the inference time only doubles, as it is linear in the width rather than the parameter count. Furthermore, the grouped-query attention for the test rows ensures that the size of the KV cache, and thus the cached-prediction time remains largely unchanged compared to TabPFN-3, at significantly improved accuracy, see Section 2.6.

## 3.2 Preprocessing

Unlike previous TabPFN versions, TabPFN-3.5 does not rely on additional feature transformations such as quantile transformations or robust scaling, and does not augment the dataset with singular value decomposition (SVD) components. The improved cell encodings, described above and shown in Figure 8,

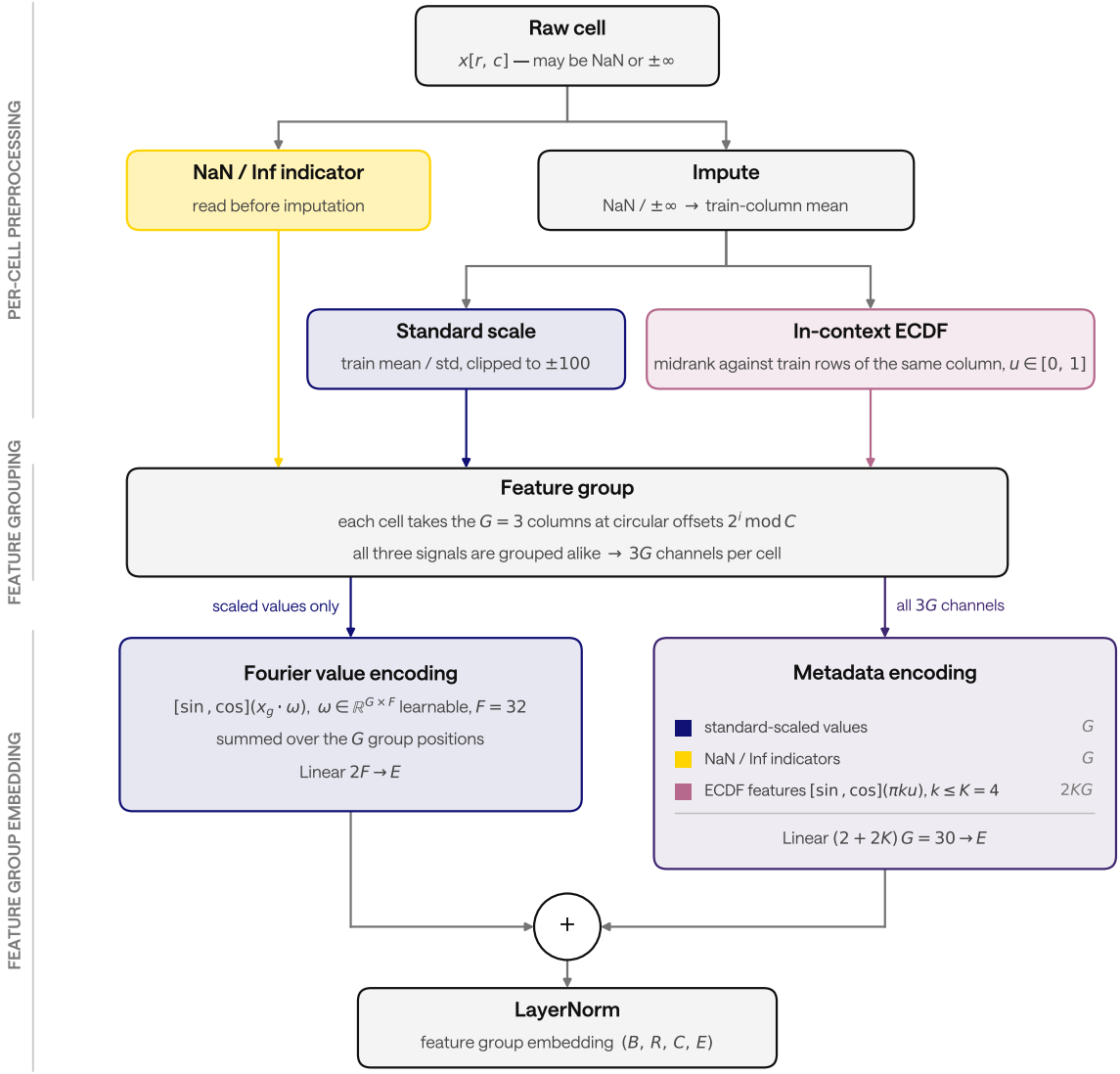



**Figure 8. Per-cell encoding in TabPFN-3.5.** B is the batch size, R and C the numbers of rows and columns, E = 128 the cell embedding width; each group spans G = 3 columns, the Fourier encoder uses F = 32 learned frequencies, and the ECDF encoder K = 4 harmonics. Preprocessing (top) records a missing/infinite indicator before imputing with the training-column mean, then derives two signals from the imputed value: a standard-scaled value and an in-context ECDF rank $u \in [0, 1]$ obtained by midranking against the training rows of the same column. Feature grouping (middle) collects the $G = 3$ columns at circular offsets $2^j$ mod $C$ for all three signals, giving $3G$ channels per cell. Group embedding (bottom) encodes the scaled values with learnable Fourier features summed over the $G$ group positions, while a linear metadata encoder reads all $3G$ channels: the scaled values passed through, the missing/infinite indicators, and the ECDF ranks expanded into low-frequency sine and cosine terms. The two encodings are summed and layer-normalised into the feature-group embedding.

provide a sufficiently expressive representation of each cell for the model to handle diverse datasets natively. We still use multiple estimators to form a final prediction: by default, TabPFN-3.5 aggregates eight and TabPFN-3.5-Fast four estimators that operate on distinct permutations of the dataset.

### 3.3 Synthetic Prior

Our synthetic data generation pipeline closely follows TabPFN-3 [1]. We make some general improvements to diversity and scalability, alongside taking inspiration from the TabICLv2 prior [32]. Furthermore, we put a specific emphasis on generating more realistic datasets that exhibit high-cardinality categorical variables in conjunction with our encoding improvements detailed in Section 3.1. Moreover, we adjusted our prior to generate more realistic datasets with large feature counts and grouped data, where the test set is from a different group than the train set. Figure 4 shows the effect on the grouped, high-feature-count, and high-cardinality slices of BeyondArena.

### 3.4 TabPFN-3.5-Plus and Thinking mode

On top of TabPFN-3.5, which we release open-source, our API and enterprise deployments provide access to TabPFN-3.5-Plus and TabPFN-3.5-Thinking. These variants are fully compatible with the open-source TabPFN-3.5 interface and can be used as a drop-in replacement, while offering additional capabilities.

While our open-source release now supports preprocessing of dates and text features, **TabPFN-3.5-Plus** further improves on multimodal data. Furthermore, TabPFN-3.5-Plus provides latency-optimized inference, most notably through FP8 attention that accelerates inference on datasets with many rows [48].

**TabPFN-3.5-Thinking** applies additional inference-time computation on top of TabPFN-3.5-Plus to push prediction quality further. Thinking mode composes with native multimodal support, so a single call can handle mixed numerical, categorical, date, and text columns under the same inference-time-compute regime. We emphasize that our Thinking mode achieves this strong performance while relying only on TabPFN-3.5, without using LLMs, real data, internet search, or any other model.

These models substantially improve on TabPFN-3-Plus and TabPFN-3-Thinking [1]. First, both benefit directly from the stronger base model. Second, we have significantly improved their implementation. Notably, TabPFN-3.5-Thinking is twelve times faster than TabPFN-3-Thinking, and TabPFN-3.5-Plus now comes with inference speed optimizations. The implementation details of TabPFN-3.5-Plus and TabPFN-3.5-Thinking are proprietary and are intentionally not described in this report.

TabPFN-3.5-Plus and TabPFN-3.5-Thinking are available through our API and through enterprise deployments including on-prem, Virtual Private Cloud (VPC) deployment on AWS SageMaker and Azure AI Foundry, and the SAP Generative AI Hub; see Section 4.1 and Section 4 for deployment options, licensing and access. Benchmark results are reported in Section 2.

## 4 License and Availability

Our open-source release has two parts under two licenses: the code, and the model weights.

**Code.** Our open-source code, including the `tabpfn` package[1] that runs the models and the Python client SDK[2], is released under the Apache License 2.0.

**Model weights.** We release the TabPFN-3.5 and TabPFN-3.5-Fast weights under the TABPFN-3.5 License v1.0, designed to be permissive for academic use, research, and evaluation in commercial settings. The license explicitly allows testing, evaluation, and internal benchmarking, so an organization can download the models and run preliminary assessments on its own datasets without a commercial agreement. The key restriction is that the models, their derivatives, and their outputs cannot be used for

[1] https://github.com/PriorLabs/TabPFN
[2] https://github.com/PriorLabs/tabpfn-client

commercial or production purposes. This includes, but is not limited to, revenue-generating products, competitive benchmarking for procurement decisions, client deliverables, and using model outputs as inputs to internal commercial decision-making. The full TABPFN-3.5 License v1.0 text is available at https://huggingface.co/Prior-Labs/tabpfn_3_5/blob/main/LICENSE.

**Commercial use.** For production use, we offer a Commercial Enterprise License, available for our managed API, the SAP Generative AI Hub, Virtual Private Cloud deployments (at the time of publication: AWS SageMaker & Azure AI Foundry), and on-prem or other custom deployment modes across other software platforms such as Databricks. The Commercial Enterprise License provides access to our proprietary high-speed inference engine, dedicated support, integration tooling, additional internal models, and the TabPFN-3.5-Plus and TabPFN-3.5-Thinking variants, which are not available as part of the open-source release. The managed API runs on our optimized GPU infrastructure and is the recommended option for users without dedicated local GPUs; it is accessible via the Python client SDK (`pip install tabpfn-client`) or a standard REST API. For commercial licensing inquiries, please contact sales@priorlabs.ai.

### 4.1 Distribution

TabPFN-3.5 and TabPFN-3.5-Fast are available through the open-source PyPI distribution for evaluation and non-commercial use, and through a managed API for commercial workloads. The models are listed on the AWS SageMaker Marketplace[3], the Azure AI Foundry Model Catalog[4] and the SAP Generative AI Hub with full support for batch and real-time inference on classification and regression tasks; TabPFN-3.5 will become available on these marketplaces with this report. A reference integration for Databricks is available through the Databricks Industry Solutions repository[5]. See Section 4 for license terms, commercial-use scope, and whom to contact about production deployment.

## References


[1] Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report, 2026. URL https://arxiv.org/abs/2605.13986.

[2] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 17285–17350. Curran Associates, Inc., 2025. doi: 10.52202/085713-0519. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1697e3fb412da11dc9488249f9e7bbc9-Paper-Datasets_and_Benchmarks_Track.pdf.

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

[4] Gioia Blayer, Myung Jun Kim, Félix Lefebvre, Lennart Purucker, Alan Arazi, Eilam Shapira, Roi Reichart, Frank Hutter, Marine Le Morvan, David Holzmüller, and Gaël Varoquaux. STRABLE: Benchmarking tabular machine learning with strings. *arXiv preprint arXiv:2605.12292*, 2026.

[5] Alan Arazi, Eilam Shapira, Shoham Grunblat, Mor Ventura, Elad Hoffer, Gioia Blayer, David Holzmüller, Lennart Purucker, Gaël Varoquaux, Frank Hutter, and Roi Reichart. Multabench:


[3] https://aws.amazon.com/marketplace/pp/prodview-chfhncrdzlb3s
[4] https://ai.azure.com/catalog/models/TabPFN-2.5
[5] https://github.com/databricks-industry-solutions/tabpfn-databricks

Benchmarking multimodal tabular learning with text and image, 2026. URL https://arxiv.org/abs/2605.10616.

[6] Adrian Hayler, Klemens Flöge, Alan Arazi, Rishabh Ranjan, Jure Leskovec, Felix Birkel, Brendan Roof, Anurag Garg, Kristina Collins, Lydia Sidhoum, Jonas Kübler, Siyuan Guo, Oscar Key, Jan Hendrik Metzen, Rylee Grace, David Salinas, Arthur Cahu, Simon Bing, Benjamin Jäger, Tuana Çelik, Mihir Manium, Vitor Monteiro, Jake Robertson, Jerry Chen, Eliott Kalfon, Tomás Pereda, Lilly Wehrhahn, Dominik Safaric, Tobias Schroeder, Georg Grab, Diana Kriuchkova, Clara Cornu, Philipp Singer, Nick Erickson, Vahid Balazadeh, Marie Salmon, Simone Alessi, Kürşat Kaya, Philipp Jund, Léo Grinsztajn, Yann LeCun, Bernhard Schölkopf, Madelon Hulsebos, Lennart Purucker, Sauraj Gambhir, Frank Hutter, and Noah Hollmann. Advancing open and reproducible relational learning: RelArena-$\alpha$, TabPFN-Rel and RPI. *arXiv preprint arXiv:2608.16319*, 2026. URL https://arxiv.org/abs/2608.16319.

[7] Oleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. fev-bench: A realistic benchmark for time series forecasting. *arXiv preprint arXiv:2509.26468*, 2025.

[8] Katharine E Henry, David N Hager, Peter J Pronovost, and Suchi Saria. A targeted real-time early warning score (trewscore) for septic shock. *Science Translational Medicine*, 7(299):299ra122, 2015.

[9] Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[10] Thyago P. Carvalho, Fabrízzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, and Symone G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.

[11] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1):4308, 2014.

[12] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[13] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35: 507–520, 2022.

[14] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[15] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

[16] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08328-6. URL https://doi.org/10.1038/s41586-024-08328-6.

[17] Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models, 2025. URL https://arxiv.org/abs/2511.08667.

[18] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. The tabular foundation model TabPFN outperforms specialized time series forecasting models based on simple features. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

[19] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: Extending TabPFN-v2 to time series forecasting, 2025. URL https://arxiv.org/abs/2501.02945.

[20] Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. Do-PFN: In-context learning for causal effect estimation. In *Advances in Neural Information Processing Systems*, volume 38, pages 193605–193642, 2025. doi: 10.52202/085713-5814. URL https://arxiv.org/abs/2506.06039.

[21] Vahid Balazadeh, Hamidreza Kamkari, Valentin Thomas, Benson Li, Junwei Ma, Jesse C. Cresswell, and Rahul G. Krishnan. Causalpfn: Amortized causal effect estimation via in-context learning, 2025. URL https://arxiv.org/abs/2506.07918.

[22] Yuchen Ma, Dennis Frauen, Emil Javurek, and Stefan Feuerriegel. Foundation models for causal inference via prior-data fitted networks, 2025. URL https://arxiv.org/abs/2506.10914.

[23] 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. doi: 10.48550/arXiv.2509.07143. URL https://arxiv.org/abs/2509.07143.

[24] Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. Graphpfn: A prior-data fitted graph foundation model, 2025. URL https://arxiv.org/abs/2509.21489.

[25] David Rundel, Julius Kobialka, Constantin von Crailsheim, Matthias Feurer, Thomas Nagler, and David Rügamer. Interpretable machine learning for TabPFN. In *World Conference on Explainable Artificial Intelligence*, pages 465–476. Springer, 2024.

[26] Lars Henry Berge Olsen and Dennis Christensen. Computing conditional shapley values using tabular foundation models, 2026. URL https://arxiv.org/abs/2602.09489.

[27] Shi-ang Qi, Vahid Balazadeh, Michael Cooper, Russell Greiner, and Rahul G Krishnan. Survivalpfn: Amortizing survival prediction via in-context bayesian inference. *arXiv preprint arXiv:2605.15488*, 2026.

[28] Dmitrii Seletkov, Paul Hager, Georgios Kaissis, Rickmer Braren, Daniel Rueckert, and Raphael Rehms. Survival in-context: Amortized bayesian survival analysis via prior-fitted networks, 2026. URL https://arxiv.org/abs/2603.29475.

[29] Rosen Ting-Ying Yu, Cyril Picard, and Faez Ahmed. GIT-BO: High-dimensional bayesian optimization with tabular foundation models. *arXiv preprint arXiv:2505.20685*, 2025. doi: 10.48550/arXiv.2505.20685. URL https://arxiv.org/abs/2505.20685.

[30] Theodore Rogers and Srividya Ponnada. Zero-shot Bayesian optimization with TabPFN: Competitive with state-of-the-art without per-task training. In *AutoML Conference*, 2026. URL https://www.amazon.science/publications/zero-shot-bayesian-optimization-with-tabpfn-competitive-with-state-of-the-art-without-per-task-

[31] David Schiff, Ofir Lindenbaum, and Yonathan Efroni. ICR-RL: Deep reinforcement learning via in-context regression. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2509.11259.

[32] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *International Conference on Machine Learning*, 2026.

[33] Google Research. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research Blog, June 2026. URL https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/. Accessed 2026-09-03.

[34] Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE tabular 1.0: Technical report, 2026. URL https://arxiv.org/abs/2608.25774.

[35] Rasa Hosseinzadeh, Alex Labach, Zexin Xue, Shuyi Han, Valentin Thomas, and Anthony L Caterini. Tabdpt-turbo: Efficient in-context learning for tabular prediction. *arXiv preprint arXiv:2608.01400*, 2026.

[36] Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. ConTextTab: A semantics-aware tabular in-context learner. In *Advances in Neural Information Processing Systems*, volume 38, pages 163185–163220, 2025. doi: 10.52202/085713-4918.

[37] Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. Tabred: Analyzing pitfalls and filling the gaps in tabular deep learning benchmarks. In *International Conference on Learning Representations*, 2025.

[38] Jonas Landsgesell, Pascal Knoll, and Tizian Wenzel. ScoringBench: A benchmark for evaluating tabular foundation models with proper scoring rules, 2026. URL https://arxiv.org/abs/2603.29928.

[39] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31, 2018.

[40] Alan Arazi, Eilam Shapira, and Roi Reichart. TabSTAR: A Tabular Foundation Model for Tabular Data with Text Fields. In *Advances in Neural Information Processing Systems*, volume 38, pages 172108–172161, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/faf6e23e198314c7728eaa6ac44ae079-Paper-Conference.pdf.

[41] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[42] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

[43] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[44] Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E. Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. RelBench: A benchmark for deep learning on relational databases. In *Advances in Neural Information Processing Systems*, volume 37, pages 21330–21341, 2024. doi: 10.52202/079017-0672. URL https://arxiv.org/abs/2407.20060.

[45] Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos Kanatsoulis, Roshan Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational Transformer: Toward Zero-Shot Foundation Models for Relational Data. In *International Conference on Learning Representations (ICLR)*, 2026. URL http://arxiv.org/abs/2510.06377.

[46] Rishabh Ranjan, Vignesh Kothapalli, Harshvardhan Agarwal, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Large-Scale Pretraining unlocks Few-Shot Prediction for Relational Data. In *2nd ICML Workshop on Foundation Models for Structured Data*, 2026. URL https://openreview.net/forum?id=oQINTd9din.

[47] 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. In *International Conference on Machine Learning (ICML)*, 2026. URL http://arxiv.org/abs/2602.04029.

[48] Jonas M. Kübler, Benjamin Jäger, Klemens Flöge, Noah Hollmann, and Frank Hutter. Attention quantization for tabular foundation models. *arXiv preprint arXiv:2609.13031*, 2026.

[49] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[50] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[51] 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*, pages 785–794, 2016.

[52] David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/2ee1c87245956e3eaa71aaba5f5753eb-Abstract-Conference.html.

[53] Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

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

[55] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

[56] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. URL http://dx.doi.org/10.1023/A%3A1010933404324.

[57] Zhiqiang Tang, Haoyang Fang, Su Zhou, Taojiannan Yang, Zihan Zhong, Tony Hu, Katrin Kirchhoff, and George Karypis. AutoGluon-Multimodal (AutoMM): Supercharging multimodal AutoML with foundation models. In *International Conference on Automated Machine Learning (AutoML)*, 2024.

[58] Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226): 1–16, 2025. URL http://jmlr.org/papers/v26/25-0512.html.

# A Contributors

Benjamin Jäger led the development of TabPFN-3.5 and is listed first. The order of the remaining members of the TabPFN-3.5 model development team, Nick Erickson, Léo Grinsztajn, Felix Birkel and Klemens Flöge, was randomized. The rest of the Prior Labs team is listed below, appearing in random order per group.

**Research/Model Dev:** Benjamin Jäger, Nick Erickson, Léo Grinsztajn, Felix Birkel, Klemens Flöge, Lennart Purucker, Arthur Cahu, Vahid Balazadeh, Jonas Kübler, Alan Arazi, Philipp Jund, Siyuan Guo, Anurag Garg, Jan Hendrik Metzen, Mihir Manium, Jake Robertson, David Salinas, Andrej Tschalzev, Simon Bing, Adrian Hayler, Tobias Schröder, Oscar Key.

**Engineering/Platform:** Brendan Roof, Georg Grab, Simone Alessi, Dominik Safaric.

**Applied:** Eliott Kalfon, Philipp Singer.

**GTM:** Clara Cornu, Diana Kriuchkova, Lilly Wehrhahn, Tuana Çelik, Vitor Monteiro.

**Ops/People/Legal:** Rylee Grace, Adèle Frankel, Kürşat Kaya, Kristina Collins, Lydia Sidhoum, Marie Salmon, Tomás Pereda, Jerry Chen.

**Scientific Advisors:** Madelon Hulsebos, Yann LeCun, Bernhard Schölkopf.

**Founders:** Frank Hutter, Sauraj Gambhir, Noah Hollmann.

# B Acknowledgements

We acknowledge the EuroHPC Joint Undertaking for awarding this project access to the EuroHPC supercomputer LUMI, hosted by CSC (Finland) and the LUMI consortium through a EuroHPC Regular Access call.

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them. This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement No 101214398 (ELLIOT).

# C More Experimental Results

## C.1 Additional details on TabArena

### C.1.1 Tuning trajectories and pairwise win rates on TabArena

Figure 9 places the TabPFN-3.5 family on TabArena's improvability-against-time plane together with the tuning trajectories of the classical baselines: each connected path shows a model's tuned-and-ensembled portfolio as the number of random configurations grows from one to all, left to right, with the default configuration as the left-most point. Improvability is how much a model would gain by switching to the best model on each dataset. Figure 10 gives the pairwise win rates of the same methods over the 816 tasks. Figures 11 and 12 repeat both views on TabArena-medium, the 15 datasets with 10k–100k training rows.

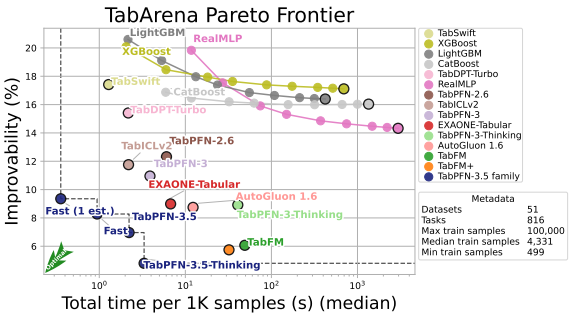



**Figure 9. Pareto frontier on TabArena**: improvability against the total training plus inference time per 1,000 rows, median over the 51 datasets, log scale. Connected points are the tuning trajectories of the classical baselines; single points are default configurations and the TabPFN-3.5 family. Section C.1.4 describes the time measurement.

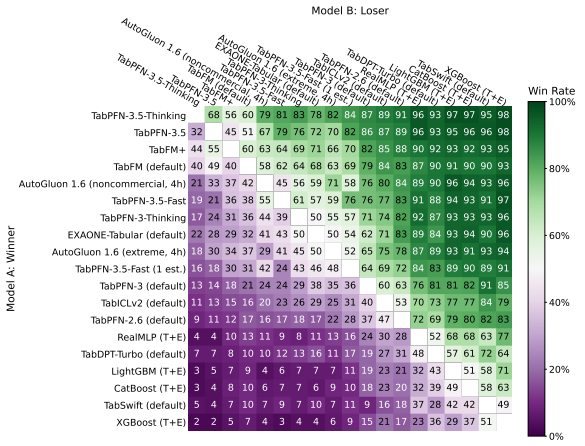



**Figure 10. Pairwise win rates on TabArena** for the strongest models on TabArena: the share of the 816 tasks on which the row method beats the column method. See Section C.1.2 for the full results.

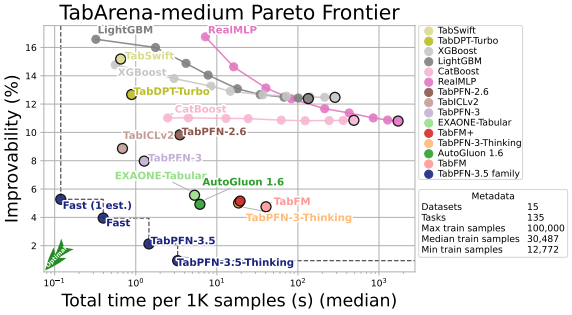



**Figure 11. Pareto frontier on TabArena-medium** (10k–100k training rows): improvability against the total training plus inference time per 1,000 rows, median over the 15 datasets, log scale. Same layout as Figure 9.

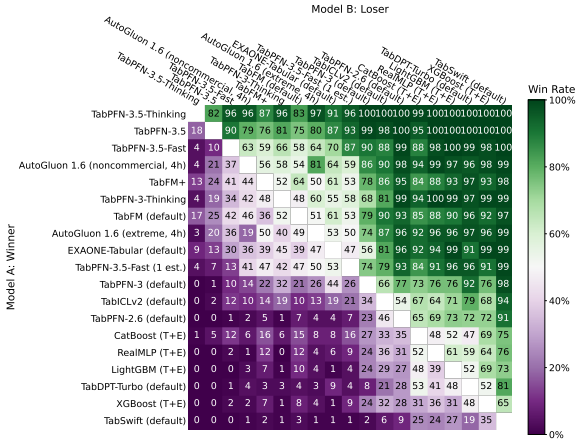



**Figure 12. Pairwise win rates on TabArena-medium** (10k–100k training rows) for the strongest models on TabArena, over the 135 tasks. See Section C.1.2 for the full results.

### C.1.2 TabArena leaderboard tables

We present the leaderboard tables for TabArena, TabArena-small, TabArena-medium, TabArena-classification, and TabArena-regression below. Elo, win counts and improvability are computed over the full TabArena field; the tables show the methods of the report's figures. Train and predict times are medians over the datasets of each view of the time per 1,000 rows, measured as described in Section C.1.4.

In all five views, TabPFN-3.5-Thinking ranks first in Elo. TabPFN-3.5 ranks second in every view except TabArena-small, where TabFM+ places between the two, and TabPFN-3.5-Fast reaches the Elo of TabFM on the full benchmark at a fraction of its train and predict time.

**Table 2. TabArena leaderboard using all 51 datasets with 816 total tasks.**

| Model | Elo (↑) | #wins (↑) | Improvability (↓) | Train time per 1K [s] | Predict time per 1K [s] |
|---|---|---|---|---|---|
| TabPFN-3.5-Thinking | **1910**$_{-82,+135}$ | **13.6** | **4.8%** | 2.00 | 1.47 |
| TabPFN-3.5 | **1866**$_{-65,+87}$ | 3.3 | 7.0% | 1.83 | 0.44 |
| TabFM+ | **1823**$_{-81,+104}$ | **8.4** | **5.8%** | 9.24 | 7.63 |
| AutoGluon 1.6 (noncommercial, 4h) | 1786$_{-57,+97}$ | 0.8 | 8.2% | 14.48 | 0.82 |
| TabPFN-3.5-Fast | 1780$_{-59,+83}$ | 0.7 | 8.3% | 0.78 | 0.16 |
| TabFM (D) | 1780$_{-95,+99}$ | **8.0** | **6.1%** | 17.11 | 7.00 |
| TabPFN-3-Thinking | 1748$_{-50,+72}$ | 1.0 | 8.9% | 36.84 | 2.43 |
| EXAONE-Tabular (D) | 1747$_{-54,+70}$ | 2.3 | 9.0% | 1.94 | 0.59 |
| AutoGluon 1.6 (extreme, 4h) | 1737$_{-51,+86}$ | 0.7 | 8.8% | 11.58 | 0.61 |
| TabPFN-3.5-Fast (1 est.) | 1703$_{-54,+66}$ | 1.0 | 9.4% | 0.30 | 0.05 |
| TabPFN-3 (D) | 1631$_{-48,+66}$ | 0.2 | 11.0% | 1.33 | 0.38 |
| TabPFN-2.6 (D) | 1579$_{-43,+57}$ | 0.1 | 12.3% | 5.48 | 0.55 |
| TabICLv2 (D) | 1565$_{-54,+64}$ | 0.3 | 11.8% | 0.66 | 0.13 |
| RealMLP (T+E) | 1474$_{-41,+43}$ | 0.2 | 14.3% | 2950.72 | 11.97 |
| TabDPT-Turbo (D) | 1434$_{-45,+49}$ | 0.3 | 15.4% | 0.65 | 0.18 |
| LightGBM (T+E) | 1404$_{-28,+28}$ | 0.0 | 16.4% | 417.05 | 2.64 |
| RealMLP (T) | 1401$_{-47,+38}$ | 0.1 | 15.6% | 2950.72 | 0.66 |
| CatBoost (T+E) | 1394$_{-33,+33}$ | 0.1 | 16.1% | 1346.21 | 0.34 |
| CatBoost (T) | 1382$_{-33,+30}$ | 0.0 | 16.3% | 1346.21 | 0.04 |
| LightGBM (T) | 1362$_{-26,+25}$ | 0.0 | 17.1% | 417.05 | 0.33 |
| CatBoost (D) | 1355$_{-39,+39}$ | 0.0 | 16.9% | 5.88 | 0.02 |
| XGBoost (T+E) | 1352$_{-30,+28}$ | 0.0 | 17.1% | 693.49 | 1.69 |
| TabSwift (D) | 1333$_{-48,+60}$ | 0.0 | 17.4% | 0.55 | 0.07 |
| XGBoost (T) | 1329$_{-28,+29}$ | 0.0 | 17.4% | 693.49 | 0.31 |
| RealMLP (D) | 1217$_{-38,+33}$ | 0.1 | 19.8% | 10.06 | 1.69 |
| XGBoost (D) | 1203$_{-35,+34}$ | 0.0 | 20.1% | 1.94 | 0.12 |
| LightGBM (D) | 1178$_{-29,+32}$ | 0.0 | 20.6% | 1.96 | 0.14 |

**Table 3. TabArena-small leaderboard on the 36 smallest datasets in TabArena**, with 500–10k training samples, evaluated on the full 681 tasks.

| Model | Elo (↑) | #wins (↑) | Improvability (↓) | Train time per 1K [s] | Predict time per 1K [s] |
|---|---|---|---|---|---|
| TabPFN-3.5-Thinking | **1824**$_{-68,+119}$ | **5.3** | **6.4%** | 1.52 | 2.14 |
| TabFM+ | **1806**$_{-80,+126}$ | **7.5** | **6.0%** | 10.68 | 6.54 |
| TabPFN-3.5 | **1796**$_{-57,+92}$ | 2.0 | 9.0% | 2.11 | 0.44 |
| TabFM (D) | 1753$_{-85,+142}$ | **6.3** | **6.6%** | 15.08 | 5.38 |
| AutoGluon 1.6 (noncommercial, 4h) | 1734$_{-54,+72}$ | 0.4 | 9.8% | 19.64 | 0.92 |
| TabPFN-3.5-Fast | 1723$_{-51,+94}$ | 0.6 | 10.1% | 0.89 | 0.16 |
| EXAONE-Tabular (D) | 1708$_{-59,+71}$ | 1.4 | 10.4% | 1.85 | 0.55 |
| TabPFN-3-Thinking | 1693$_{-45,+81}$ | 0.6 | 10.6% | 51.76 | 2.69 |
| AutoGluon 1.6 (extreme, 4h) | 1689$_{-47,+55}$ | 0.5 | 10.4% | 15.56 | 0.63 |
| TabPFN-3.5-Fast (1 est.) | 1656$_{-55,+82}$ | 0.6 | 11.1% | 0.45 | 0.05 |
| TabPFN-3 (D) | 1611$_{-51,+68}$ | 0.2 | 12.2% | 1.97 | 0.40 |
| TabPFN-2.6 (D) | 1572$_{-38,+57}$ | 0.1 | 13.4% | 7.03 | 0.55 |
| TabICLv2 (D) | 1553$_{-74,+90}$ | 0.3 | 13.0% | 0.78 | 0.14 |
| RealMLP (T+E) | 1455$_{-43,+49}$ | 0.2 | 15.8% | 3771.85 | 21.90 |
| TabDPT-Turbo (D) | 1430$_{-52,+68}$ | 0.3 | 16.6% | 0.86 | 0.19 |
| RealMLP (T) | 1382$_{-42,+46}$ | 0.1 | 17.2% | 3771.85 | 1.78 |
| LightGBM (T+E) | 1372$_{-33,+28}$ | 0.0 | 18.1% | 892.49 | 2.57 |
| TabSwift (D) | 1350$_{-51,+58}$ | 0.0 | 18.4% | 0.57 | 0.06 |
| CatBoost (T+E) | 1349$_{-38,+35}$ | 0.1 | 18.2% | 1739.17 | 0.40 |
| LightGBM (T) | 1338$_{-32,+29}$ | 0.0 | 18.5% | 892.49 | 0.35 |
| CatBoost (T) | 1337$_{-33,+31}$ | 0.0 | 18.4% | 1739.17 | 0.05 |
| XGBoost (T+E) | 1311$_{-35,+33}$ | 0.0 | 19.1% | 884.48 | 2.37 |
| CatBoost (D) | 1305$_{-38,+34}$ | 0.0 | 19.3% | 8.46 | 0.05 |
| XGBoost (T) | 1294$_{-40,+32}$ | 0.0 | 19.3% | 884.48 | 0.39 |
| RealMLP (D) | 1212$_{-43,+37}$ | 0.1 | 21.1% | 15.69 | 4.69 |
| LightGBM (D) | 1158$_{-44,+43}$ | 0.0 | 22.3% | 3.61 | 0.17 |
| XGBoost (D) | 1157$_{-38,+25}$ | 0.0 | 22.3% | 3.29 | 0.25 |

**Table 4. TabArena-medium leaderboard on the 15 largest datasets in TabArena**, with 10k–100k training samples, evaluated on the full 135 tasks with 9 splits per dataset.

| Model | Elo (↑) | #wins (↑) | Improvability (↓) | Train time per 1K [s] | Predict time per 1K [s] |
|---|---|---|---|---|---|
| TabPFN-3.5-Thinking | **2494**$_{-114,+232}$ | **8.3** | **0.9%** | 2.33 | 0.99 |
| TabPFN-3.5 | **2293**$_{-92,+225}$ | **1.2** | **2.1%** | 1.06 | 0.40 |
| TabPFN-3.5-Fast | **2117**$_{-105,+213}$ | 0.1 | **3.9%** | 0.30 | 0.10 |
| AutoGluon 1.6 (noncommercial, 4h) | 2097$_{-78,+140}$ | 0.4 | 4.3% | 7.01 | 0.63 |
| TabPFN-3-Thinking | 2067$_{-56,+99}$ | 0.4 | 5.0% | 15.90 | 2.12 |
| AutoGluon 1.6 (extreme, 4h) | 2022$_{-64,+105}$ | 0.2 | 4.9% | 5.21 | 0.52 |
| TabFM+ | 2003$_{-108,+251}$ | 0.9 | 5.1% | 3.53 | 13.80 |
| EXAONE-Tabular (D) | 1995$_{-73,+137}$ | 0.9 | 5.6% | 3.93 | 1.21 |
| TabFM (D) | 1986$_{-129,+270}$ | **1.7** | 4.8% | 26.32 | 13.12 |
| TabPFN-3.5-Fast (1 est.) | 1980$_{-87,+152}$ | 0.3 | 5.3% | 0.09 | 0.03 |
| TabPFN-3 (D) | 1788$_{-129,+194}$ | 0.0 | 8.0% | 0.58 | 0.21 |
| TabPFN-2.6 (D) | 1691$_{-61,+108}$ | 0.0 | 9.8% | 2.76 | 0.70 |
| TabICLv2 (D) | 1689$_{-103,+168}$ | 0.0 | 8.9% | 0.41 | 0.12 |
| RealMLP (T+E) | 1608$_{-95,+91}$ | 0.0 | 10.8% | 1719.82 | 1.67 |
| CatBoost (T+E) | 1600$_{-77,+99}$ | 0.0 | 10.9% | 489.93 | 0.25 |
| CatBoost (T) | 1586$_{-84,+100}$ | 0.0 | 11.1% | 489.93 | 0.02 |
| LightGBM (T+E) | 1572$_{-51,+59}$ | 0.0 | 12.4% | 131.56 | 2.64 |
| CatBoost (D) | 1572$_{-109,+114}$ | 0.0 | 11.0% | 2.47 | 0.01 |
| XGBoost (T+E) | 1534$_{-61,+82}$ | 0.0 | 12.5% | 282.13 | 0.56 |
| RealMLP (T) | 1522$_{-85,+102}$ | 0.0 | 11.9% | 1719.82 | 0.08 |
| TabDPT-Turbo (D) | 1514$_{-134,+141}$ | 0.0 | 12.7% | 0.46 | 0.18 |
| LightGBM (T) | 1490$_{-49,+60}$ | 0.0 | 13.5% | 131.56 | 0.13 |
| XGBoost (T) | 1488$_{-58,+60}$ | 0.0 | 13.0% | 282.13 | 0.07 |
| XGBoost (D) | 1366$_{-99,+90}$ | 0.0 | 14.8% | 0.49 | 0.05 |
| TabSwift (D) | 1337$_{-103,+110}$ | 0.0 | 15.2% | 0.42 | 0.13 |
| RealMLP (D) | 1264$_{-80,+76}$ | 0.0 | 16.8% | 6.75 | 0.23 |
| LightGBM (D) | 1260$_{-60,+46}$ | 0.0 | 16.6% | 0.29 | 0.04 |

**Table 5. TabArena-classification leaderboard on the 38 classification datasets in TabArena**, evaluated on the full 594 tasks.

| Model | Elo (↑) | #wins (↑) | Improvability (↓) | Train time per 1K [s] | Predict time per 1K [s] |
|---|---|---|---|---|---|
| TabPFN-3.5-Thinking | **1872**$_{-88,+131}$ | **7.7** | **6.2%** | 2.37 | 1.54 |
| TabPFN-3.5 | **1845**$_{-71,+106}$ | 2.0 | 8.9% | 1.86 | 0.49 |
| TabFM+ | **1803**$_{-89,+117}$ | **7.3** | **7.2%** | 10.68 | 7.80 |
| TabPFN-3.5-Fast | 1784$_{-69,+102}$ | 0.5 | 10.0% | 0.78 | 0.16 |
| TabFM (D) | 1768$_{-114,+121}$ | **7.5** | **7.2%** | 18.58 | 7.06 |
| AutoGluon 1.6 (noncommercial, 4h) | 1757$_{-53,+80}$ | 0.2 | 10.2% | 12.89 | 0.79 |
| EXAONE-Tabular (D) | 1752$_{-58,+79}$ | 2.1 | 10.8% | 1.58 | 0.51 |
| TabPFN-3-Thinking | 1744$_{-51,+88}$ | 0.8 | 11.0% | 35.78 | 2.42 |
| TabPFN-3.5-Fast (1 est.) | 1710$_{-53,+97}$ | 0.8 | 11.2% | 0.32 | 0.05 |
| AutoGluon 1.6 (extreme, 4h) | 1709$_{-42,+61}$ | 0.2 | 10.9% | 9.70 | 0.57 |
| TabPFN-3 (D) | 1625$_{-65,+71}$ | 0.1 | 13.5% | 1.32 | 0.38 |
| TabPFN-2.6 (D) | 1574$_{-53,+55}$ | 0.1 | 14.8% | 5.17 | 0.54 |
| TabICLv2 (D) | 1570$_{-64,+73}$ | 0.3 | 14.1% | 0.68 | 0.14 |
| RealMLP (T+E) | 1461$_{-36,+51}$ | 0.1 | 17.1% | 2879.46 | 12.48 |
| TabDPT-Turbo (D) | 1419$_{-54,+71}$ | 0.2 | 18.5% | 0.69 | 0.19 |
| LightGBM (T+E) | 1411$_{-30,+44}$ | 0.0 | 18.8% | 382.09 | 1.49 |
| CatBoost (T+E) | 1395$_{-42,+47}$ | 0.1 | 18.5% | 1107.00 | 0.33 |
| RealMLP (T) | 1390$_{-42,+49}$ | 0.1 | 18.5% | 2879.46 | 0.60 |
| CatBoost (T) | 1386$_{-40,+45}$ | 0.0 | 18.7% | 1107.00 | 0.04 |
| CatBoost (D) | 1372$_{-44,+43}$ | 0.0 | 19.0% | 5.41 | 0.03 |
| LightGBM (T) | 1370$_{-30,+36}$ | 0.0 | 19.5% | 382.09 | 0.25 |
| XGBoost (T+E) | 1359$_{-43,+39}$ | 0.0 | 19.6% | 685.87 | 1.45 |
| TabSwift (D) | 1338$_{-58,+55}$ | 0.0 | 20.5% | 0.56 | 0.06 |
| XGBoost (T) | 1334$_{-38,+35}$ | 0.0 | 19.9% | 685.87 | 0.21 |
| RealMLP (D) | 1230$_{-39,+41}$ | 0.1 | 22.7% | 10.48 | 1.71 |
| XGBoost (D) | 1220$_{-47,+40}$ | 0.0 | 22.5% | 1.77 | 0.12 |
| LightGBM (D) | 1182$_{-41,+49}$ | 0.0 | 23.4% | 1.79 | 0.12 |

**Table 6. TabArena-regression leaderboard on the 13 regression datasets in TabArena**, evaluated on the full 222 tasks.

| Model | Elo (↑) | #wins (↑) | Improvability (↓) | Train time per 1K [s] | Predict time per 1K [s] |
|---|---|---|---|---|---|
| TabPFN-3.5-Thinking | **2250**$_{-135,+241}$ | **5.9** | **0.8%** | 1.54 | 0.99 |
| TabPFN-3.5 | **2103**$_{-130,+245}$ | **1.3** | **1.4%** | 1.55 | 0.37 |
| AutoGluon 1.6 (noncommercial, 4h) | **2056**$_{-141,+190}$ | 0.6 | 2.1% | 22.20 | 1.73 |
| TabFM+ | 2055$_{-130,+186}$ | **1.1** | **1.7%** | 7.28 | 7.19 |
| AutoGluon 1.6 (extreme, 4h) | 2002$_{-107,+161}$ | 0.5 | 2.6% | 19.36 | 1.63 |
| TabFM (D) | 1975$_{-85,+161}$ | 0.5 | 2.8% | 14.60 | 6.47 |
| TabPFN-3.5-Fast | 1912$_{-94,+172}$ | 0.2 | 3.3% | 0.66 | 0.13 |
| TabPFN-3-Thinking | 1907$_{-130,+158}$ | 0.3 | 2.8% | 37.75 | 2.43 |
| EXAONE-Tabular (D) | 1868$_{-96,+131}$ | 0.2 | 3.9% | 2.47 | 0.78 |
| TabPFN-3.5-Fast (1 est.) | 1816$_{-106,+175}$ | 0.1 | 3.9% | 0.20 | 0.05 |
| TabPFN-3 (D) | 1784$_{-123,+205}$ | 0.0 | 3.6% | 1.33 | 0.38 |
| TabPFN-2.6 (D) | 1723$_{-47,+95}$ | 0.0 | 5.1% | 8.52 | 0.70 |
| TabICLv2 (D) | 1672$_{-134,+213}$ | 0.0 | 5.1% | 0.64 | 0.11 |
| RealMLP (T+E) | 1636$_{-56,+99}$ | 0.0 | 6.1% | 3995.01 | 10.05 |
| TabDPT-Turbo (D) | 1594$_{-88,+175}$ | 0.1 | 6.3% | 0.49 | 0.18 |
| RealMLP (T) | 1540$_{-68,+98}$ | 0.0 | 7.1% | 3995.01 | 0.84 |
| CatBoost (T+E) | 1479$_{-57,+78}$ | 0.0 | 9.0% | 2229.21 | 0.38 |
| LightGBM (T+E) | 1472$_{-68,+71}$ | 0.0 | 9.4% | 700.19 | 9.32 |
| CatBoost (T) | 1456$_{-58,+83}$ | 0.0 | 9.1% | 2229.21 | 0.06 |
| LightGBM (T) | 1411$_{-74,+75}$ | 0.0 | 10.0% | 700.19 | 0.97 |
| XGBoost (T+E) | 1402$_{-39,+49}$ | 0.0 | 10.0% | 834.93 | 2.61 |
| TabSwift (D) | 1395$_{-110,+146}$ | 0.0 | 8.3% | 0.49 | 0.07 |
| XGBoost (T) | 1381$_{-49,+58}$ | 0.0 | 10.1% | 834.93 | 0.39 |
| CatBoost (D) | 1375$_{-83,+83}$ | 0.0 | 10.6% | 8.73 | 0.02 |
| RealMLP (D) | 1221$_{-87,+102}$ | 0.0 | 11.5% | 8.90 | 1.64 |
| LightGBM (D) | 1201$_{-44,+29}$ | 0.0 | 12.4% | 2.11 | 0.27 |
| XGBoost (D) | 1182$_{-82,+83}$ | 0.0 | 13.0% | 2.24 | 0.24 |

### C.1.3 Additional results on TabArena

**What drives improvability differences?** Figure 13 shows the improvability as a function of total fit-and-predict time. Although TabFM and TabFM+ achieve slightly higher total improvability than TabPFN-3.5, the per-dataset comparison in fig. 14 shows that the remaining gap is dominated by two datasets: `hazelnut-spread-contaminant-detection` and `polish-companies-bankruptcy`. Both are already nearly perfectly solved, with ROC AUC above 0.997, which leads to improvability inflating the small absolute residual differences.

### C.1.4 How the time axis of the TabArena Pareto frontier is computed

**Protocol.** All times in Figure 2 follow TabArena's default protocol [2]: every method is fitted as an 8-fold bagging ensemble with a final refit on the full training set, in a fresh process per split, on one node with 24 CPU cores and one NVIDIA RTX Pro 6000 GPU. The fit time of a split is the wall-clock time of the whole bagged fit; the prediction time is the wall-clock time of the refitted model predicting the test third of the dataset. While the 8-fold bagging is not used for TFMs in general, as they are refitted on the full training set during inference, we keep it to make the comparison fairer to other non-TFM models, which could also be fitted without bagging (although with weaker performance). Systems (TabPFN-3.5-Thinking, TabPFN-3-Thinking, TabFM+, AutoGluon 1.6) are instead called once per split and do their own validation and ensembling, so their fit time might look a bit better than those of the bagged configurations; this is why TabFM+ shows a lower total time than TabFM in Figure 2 although it runs the same network.

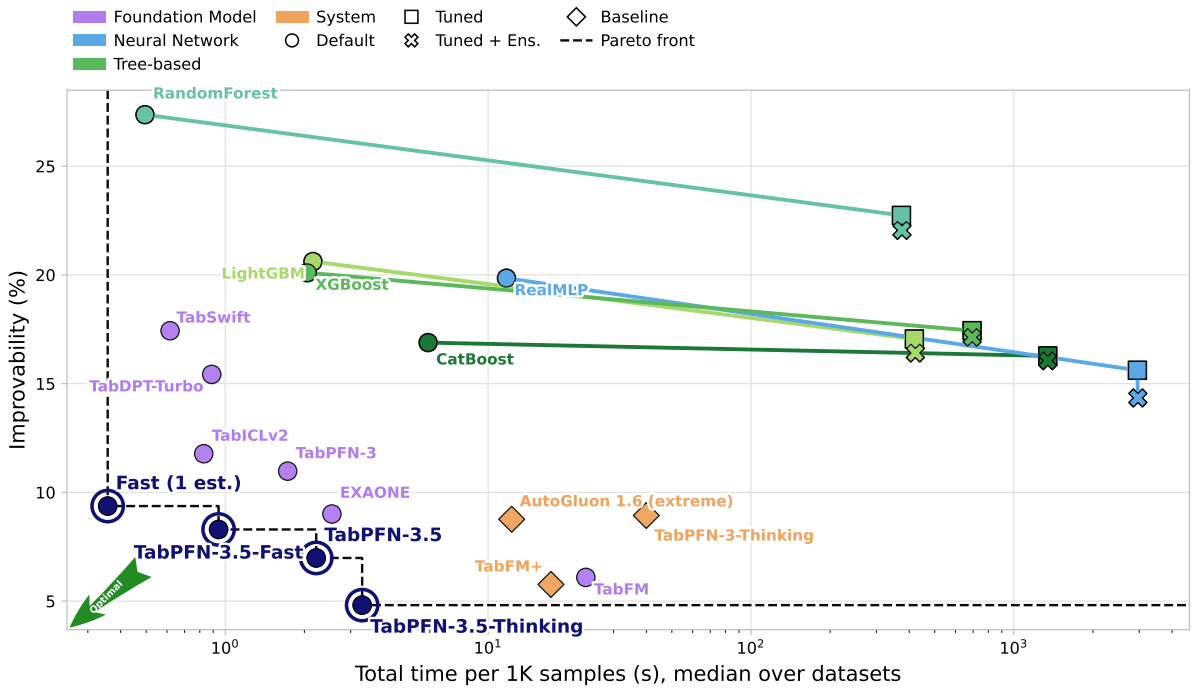



**Figure 13. TabArena improvability against total time.** Lower is better. Same methods and time axis as Figure 2; Section C.1.4 describes the measurement.

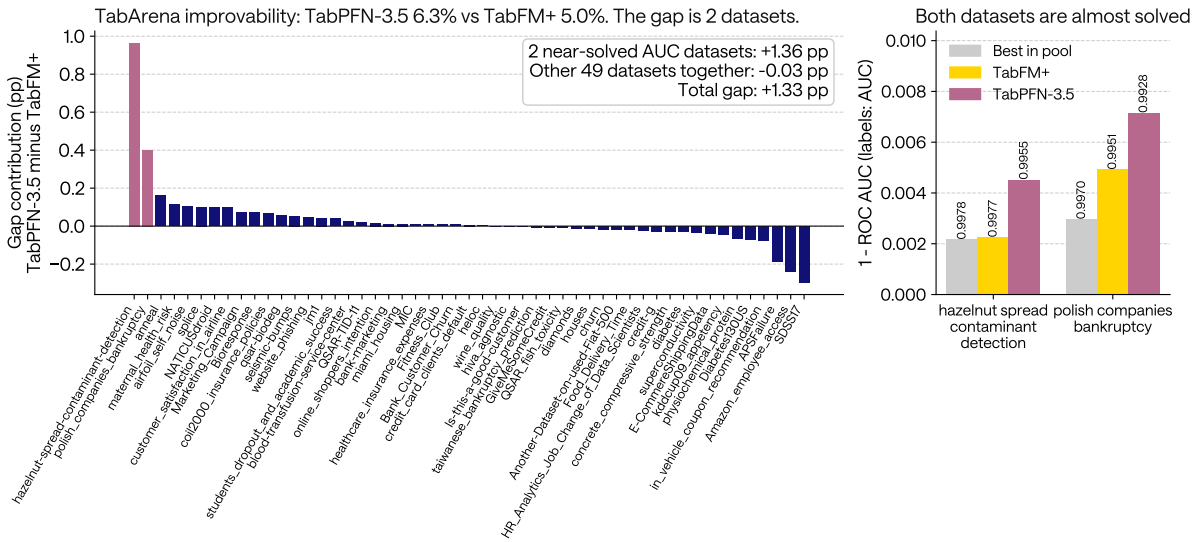



**Figure 14.** Improvability vs. TabFM+, we can see that the improvability gap in the final Pareto Frontier boils down to `hazelnut-spread-contaminant-detection` and `polish-companies-bankruptcy`, both of which are almost perfectly solved, with ROC AUC exceeding 0.997.

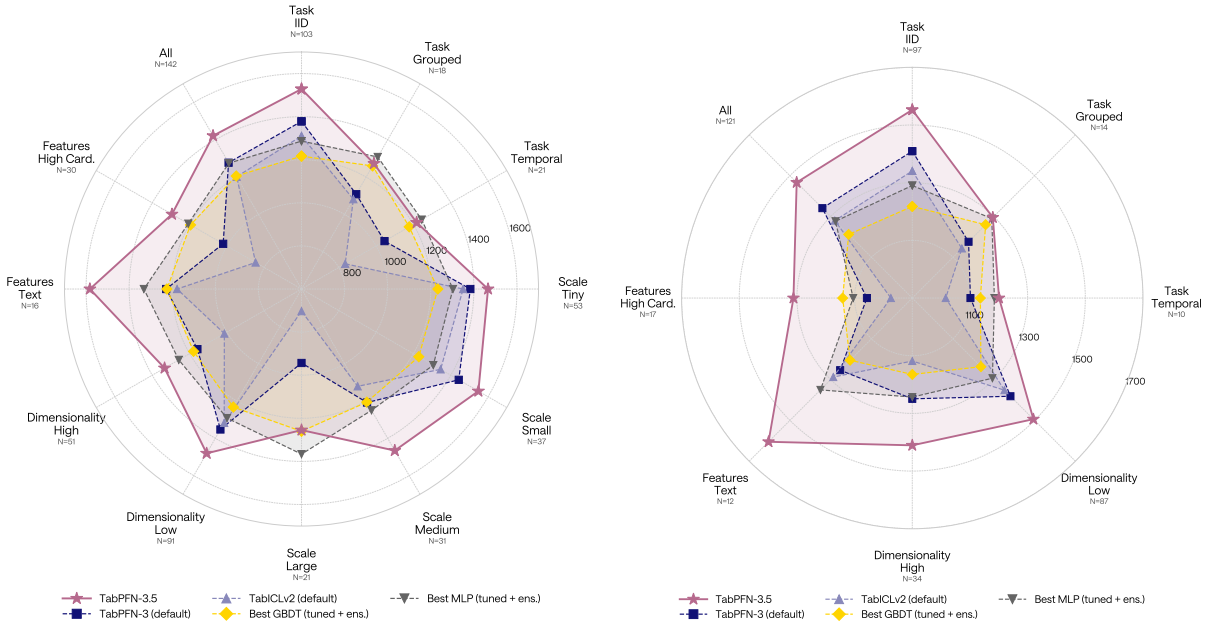
**Normalization.** Per split, fit time is expressed per 1,000 training rows and prediction time per 1,000 test rows, averaged over folds per dataset and reported as the median over the 51 datasets, as in the TabArena leaderboard [2]. The total time of a split is the sum of the two per-1,000-row rates, the definition tabarena's code uses for its combined time axis.

**Re-timing of the foundation models.** The arena's wrappers for tabular foundation models carried overheads that are not part of running the model and were often larger than actual train times for small datasets, making it hard to compare different TFMs' speeds: the network was rebuilt from its checkpoint and copied to the GPU for each of the nine bagging children, the child models were saved with the full network weights inside, and the library import and the first forward pass of the process fell inside the timed fit. We removed these in the wrappers (one network per process shared across the children, no weights in the saved children, and an untimed warm-up on fixed synthetic data) and re-timed TabPFN-3.5, TabPFN-3.5-Fast, TabFM, TabFM+, EXAONE-Tabular, TabPFN-3, TabICLv2, TabDPT-Turbo [35] and TabSwift on the first repeat of every dataset (3 folds, 153 splits), with the Python environment on the node's local disk. Predictions are unchanged, so the Elo scores are those of the full 816-split evaluations. The tree-based and neural baselines use the arena's own measurements. TabPFN-3-Thinking keeps the Elo and time of its published run from May 2026, which predates these fixes, so its time is an upper bound.

## C.2 Additional results on BeyondArena

### C.2.1 Radar view of the BeyondArena slices

Figure 15 shows the Elo scores of Figure 3 and Figure 4 as radar charts, with one axis per data slice. The numbers are the same as in the bar charts.




**(a)** Full BeyondArena core.

**(b)** BeyondArena core without large tables.

**Figure 15. BeyondArena Elo by data slice, radar view.** Same Elo fits as Figure 3 (left) and Figure 4 (right): TabPFN-3.5 against TabPFN-3, TabICLv2, and the strongest tuned and ensembled GBDT and MLP baselines. The group identifier is exposed to the TabPFN models on label-per-sample grouped datasets; see Section C.2.3.

### C.2.2 Comparison to recent TFMs

We additionally compare TabPFN-3.5 with TabFM [33] and EXAONE-Tabular [34] on the subset of BeyondArena that both models can run (Figure 16). The subset is BeyondArena core without large tables

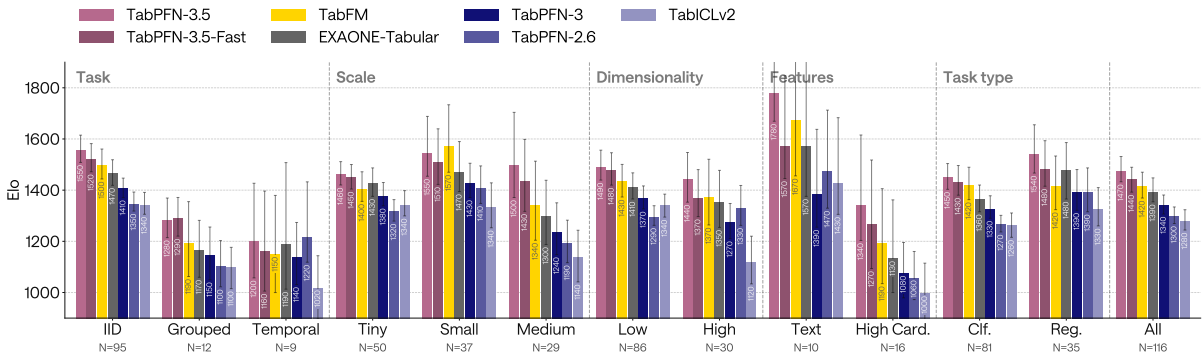



**Figure 16. Comparison with recent tabular foundation models on BeyondArena.** Elo scores on the 116 BeyondArena core datasets without large tables that TabPFN-3.5, TabFM, and EXAONE-Tabular all support (see text), reported overall and by data slice. TabPFN-3.5 leads overall and across nearly every slice. The group identifier is exposed to the TabPFN models and to TabFM on label-per-sample grouped datasets, not to EXAONE-Tabular; see Section C.2.3. The labels of the Elo scores are rounded to the nearest multiple of 10.

(up to 100K rows), minus five datasets: `asp_potassco_classification`, `ljubljana_primary_tumor` and `micro_mass` have more than ten classes, which TabFM does not support, and `sf_permit_time` (TabFM and EXAONE-Tabular) and `pva_revenue_prediction_kddcup98` (TabFM) run out of memory on a 96 GB GPU. This leaves 116 datasets, on which every method is scored on the same splits. TabPFN-3.5 leads both models overall and across every data slice except small datasets, where TabFM leads. Again, wide confidence intervals make some slice-level differences uncertain.

### C.2.3 Exposing the group identifier on label-per-sample grouped tasks

On grouped datasets, BeyondArena applies a shared preprocessing step which drops the group id before the table reaches a model. On *label-per-group* tasks where every row of a group shares its label, BeyondArena also adds transductive feature aggregates within each group, which still encode which group a row belongs to. For *label-per-sample* tasks, where labels vary within a group, no other grouped-specific preprocessing is done besides dropping the group id, so the group structure is not passed to the model. Not every method is affected in the same way: tuned and early-stopped baselines still exploit the group structure, because their validation splits are group-aware, whereas models which are fit on the full training set (TFMs) cannot exploit the group structure.

To make the comparison fairer and closer to what practitioners can use in practice, our aim on label-per-sample tasks is to score every model in the configuration that serves it best. We therefore tried exposing the group key to every model family with several encodings (float, categorical, string), passing it raw or marking unseen test groups as missing. Simply passing the group id as a categorical variable helps every TabPFN model and we score TabPFN-3.5, TabPFN-3.5-Fast, TabPFN-3.5-Thinking and TabPFN-3 with it. We score TabFM with the group id exposed as a float, unseen test groups marked as missing, which we find helps the most for this model. In our evaluations, no other model gains from passing the group id in any encoding, consistent with the results reported in the BeyondArena paper [3], where only TabPFN-2.6 benefits from removing the grouped preprocessing for label-per-sample grouped datasets. For each model, we only evaluate whether passing the group id helps for the default model, not the tuned and ensembled version, so we cannot exclude that these versions would benefit from the group id (though they arguably already use the group-aware validation splits).

On grouped data without large tables, when the group id is included as input, TabPFN-3.5 moves from 1240 to 1288 Elo (rank 6 to rank 1), TabPFN-3 from 1139 to 1176, and TabFM from 1167 to 1187. On grouped data including large tables, TabPFN-3.5 moves from 1187 to 1267 Elo (rank 9 to rank 4) and TabPFN-3 from 1078 to 1108.

## C.3 Additional Results on STRABLE

To evaluate performance on the 108 datasets of STRABLE (3 folds each), Elo ratings were computed jointly over a total of 200 methods. Specifically, we include only the 196 baselines from the original

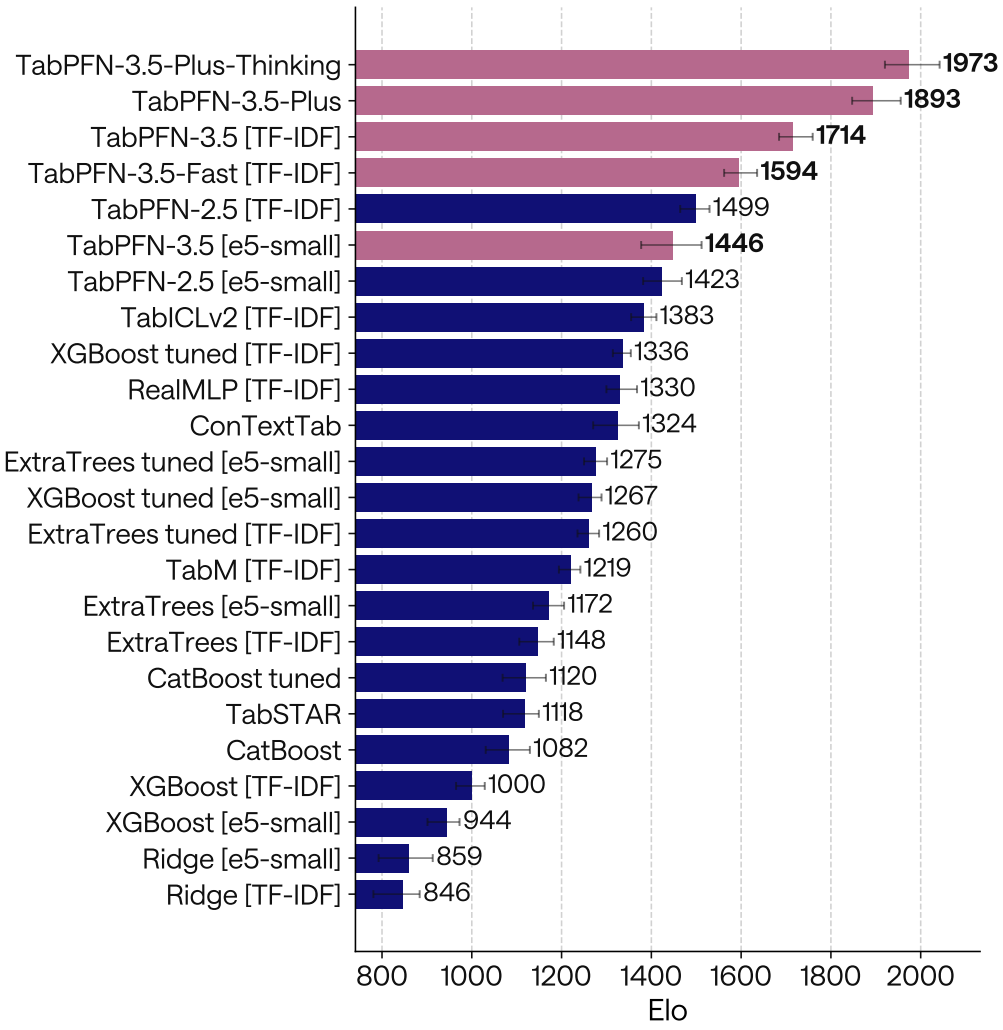



**Figure 17.** Elo on STRABLE. Elo ratings for a representative subset of methods, derived from a joint fit of all methods across all 108 datasets.

paper that successfully completed all 324 runs (108 datasets × 3 folds). On top of these, we added TabPFN-3.5 and TabPFN-3.5-Fast with TF-IDF, TabPFN-3.5 with *e5-small-v2* [41], TabPFN-3.5-Plus, and TabPFN-3.5-Thinking.

Figure 17 displays a selected list of baselines, focusing specifically on models that are either end-to-end or utilize the stronger string encoding methods. Because the Elo ratings are derived from the comprehensive joint fit over all methods, the values remain identical to those presented in the main paper's figure, which simply filtered out a few of these pipelines for visual clarity.

Alongside our models, the 19 representative published pipelines shown include Ridge [49], ExtraTrees [50], XGBoost [51], RealMLP [52], TabM [53], TabICLv2 [32], and TabPFN-2.5 [17], all utilizing TF-IDF string encodings; four of these also feature variants with *e5-small-v2* embeddings. Additionally, we include baselines that process raw text directly: CatBoost [39], ConTextTab [36], and TabSTAR [40].

When evaluating pipelines equipped with *e5-small-v2* embeddings, the relative rankings remain consistent: TabPFN-3.5 narrowly outperforms TabPFN-2.5, albeit by a smaller margin than observed with TF-IDF. Crucially, TabPFN-3.5-Thinking and TabPFN-3.5-Plus, which process the raw text directly, retain the first and second positions on the overall leaderboard by a substantial margin. Finally, consistent with the official STRABLE benchmark findings, TF-IDF generally proves to be the stronger text representation. This aligns with the nature of the strings in the benchmark, which are mostly short entries rather than long-form free text.

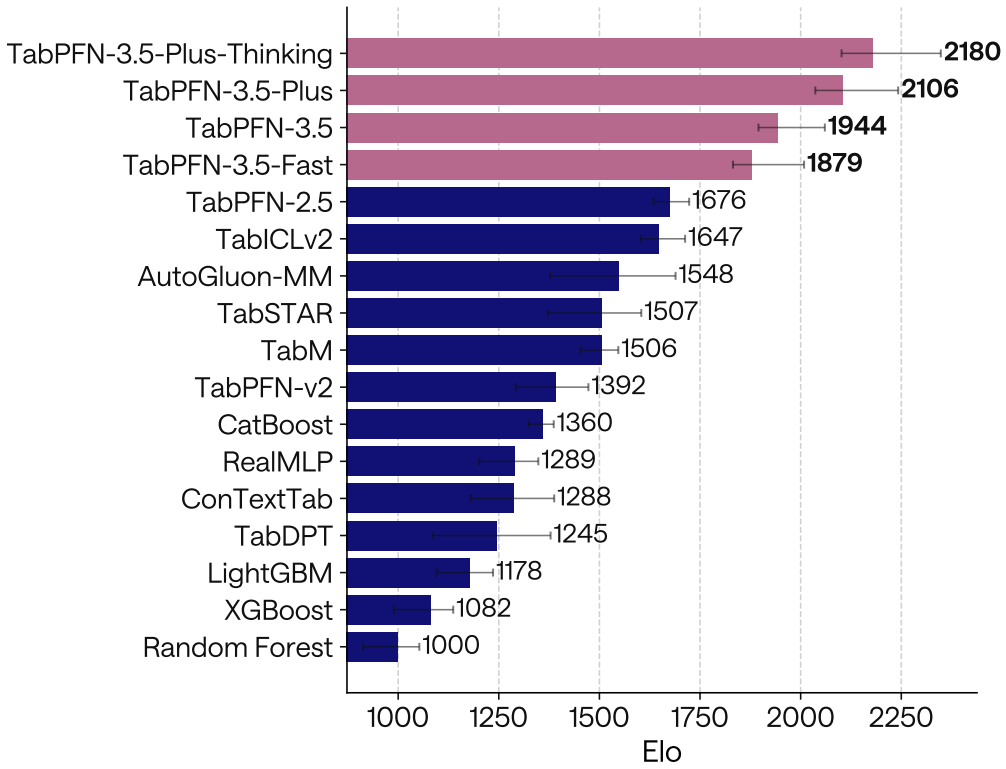



**Figure 18. Elo on MulTaBench text subset.** Elo ratings for all evaluated methods, computed jointly across the 20 text-focused datasets with 5 folds each. The TabPFN-3.5 family dominates the benchmark.

## C.4 Additional Results on MulTaBench

Figure 18 displays the Elo rankings evaluated specifically on the subset of 20 text-focused datasets of MulTaBench (5 folds each). The Elo ratings are computed jointly over the 13 baseline models published in the official benchmark: TabICLv2 [32], TabPFN-2.5 [17], TabPFN-v2 [16], TabM [53], TabDPT [54], CatBoost [39], RealMLP [52], LightGBM [55], XGBoost [51], and Random Forest [56], all of which, like TabPFN-3.5, utilize *e5-small-v2* [41] embeddings, as well as AutoGluon-MM [57], ConTextTab [36], and TabSTAR [40], which process the raw text directly. In addition, we run our own models: TabPFN-3.5, TabPFN-3.5-Fast, TabPFN-3.5-Plus, and TabPFN-3.5-Thinking.

Focusing exclusively on these text datasets, the performance gap between our models and the baselines widens further compared to the full benchmark. TabPFN-3.5 and TabPFN-3.5-Fast, despite relying on the same frozen *e5-small-v2* features as the non-native baselines, extend their lead over prior tabular foundation models such as TabPFN-2.5 and TabICLv2. Crucially, TabPFN-3.5-Plus and TabPFN-3.5-Thinking achieve an even greater advantage, securing the first and second position by a substantial margin.

## C.5 TALENT

TALENT [58] is a large collection of tabular datasets from many domains. Its base benchmark has 300 datasets spanning classification and regression tasks, and the authors later added extensions with larger and higher-dimensional datasets. Every dataset comes with a fixed split of 64% training, 16% validation and 20% test rows. All methods are ranked on each dataset, and the average rank is reported with a 95% bootstrap confidence interval over datasets. A method that has no result on some datasets gets those cells filled with the score of the KNN baseline; its bar is hatched and its label says how many cells were filled. The TabPFN-3 report [1] describes these steps in detail.

We show two views of TALENT.

**TabICLv2 evaluation protocol.** Figure 19 follows the evaluation protocol of the TabICLv2 paper [32]: the 300 base datasets minus the 26 that were used during the development of TabPFN-2 and TabICLv2, 274 datasets in total, with the baselines the TabICLv2 authors ran. This is the same setup as the main TALENT figure of the TabPFN-3 report, with TabPFN-3.5 added. TabPFN-3.5 ranks first with an average rank of 2.95, ahead of TabPFN-3 at 4.48 and TabICLv2 at 5.35. The confidence intervals of TabPFN-3.5 and TabPFN-3 do not overlap.

**All TALENT datasets up to one million training rows.** Figure 20 uses every TALENT dataset with published baselines and at most one million training rows: the 300 base datasets plus the large-data and high-dimensional extensions, 337 datasets in total. The baselines are the results published by the TALENT authors. The TabICLv2 results used in Figure 19 do not cover the extensions, and the TALENT tables list earlier TabPFN versions only on a small part of the datasets, so this view compares against classical and deep tabular baselines only. TabPFN-3.5 again ranks first, with an average rank of 2.13 against 3.20 for TabPFN-3.

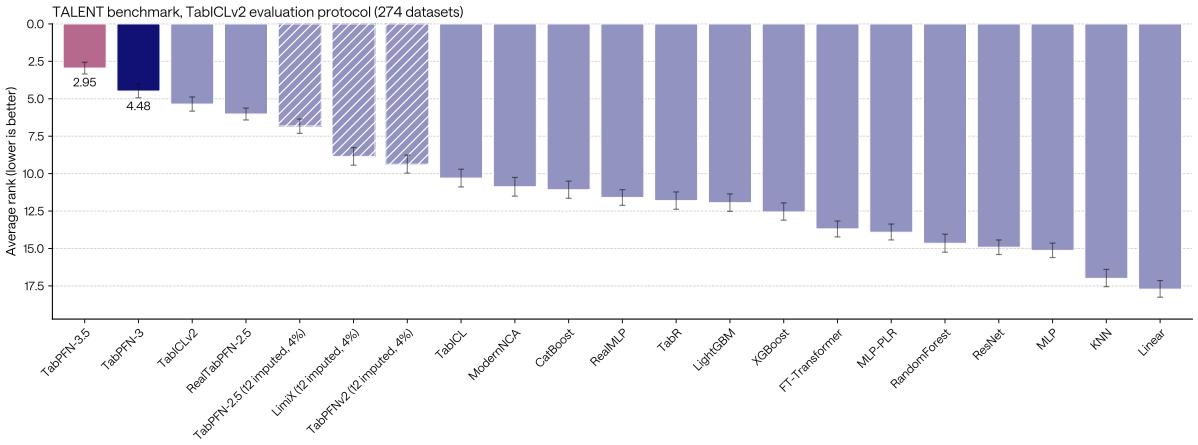



**Figure 19. Average rank on the TALENT benchmark, using the TabICLv2 evaluation protocol [32] (274 datasets).** Same datasets and baselines as the main TALENT figure of the TabPFN-3 report [1], with TabPFN-3.5 added. Lower is better; TabPFN-3.5 in plum, TabPFN-3 in dark blue. Error bars are 95% bootstrap confidence intervals over datasets. Hatched bars have KNN-imputed cells, counted in the label.

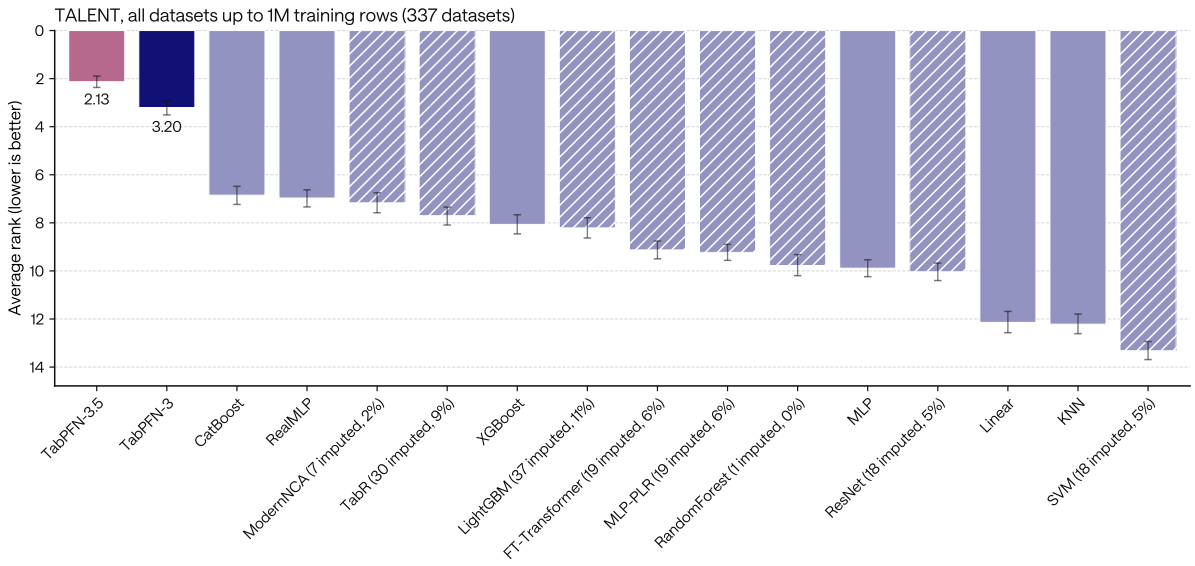



**Figure 20. Average rank on all TALENT datasets with up to one million training rows (337 datasets).** The 300 base datasets plus the large-data and high-dimensional extensions, against the baselines published by the TALENT authors [58]. Lower is better; TabPFN-3.5 in plum, TabPFN-3 in dark blue. Error bars are 95% bootstrap confidence intervals over datasets. Hatched bars have KNN-imputed cells, counted in the label.

## C.6 fev-bench

fev-bench [7] is a forecasting benchmark of 100 tasks on public series, scored as skill over Seasonal Naive with the SQL (quantile) and MASE (point) metrics. We run TabPFN-3.5 through TabPFN-TS [18], which casts each series as a regression over calendar and seasonal features, with the inference config of the published TabPFN-TS-3 entry (context 32768, 12 AutoSeasonal periods), and score against the current upstream leaderboard field of 29 methods. Figure 21 shows the result: TabPFN-3.5 places sixth, 1.1 points above TabPFN-TS-3 at 2.4× its speed, behind TimesFM-3, the two Chronos-2 variants, TiRex-2 and Toto-2.0. The checkpoint is the general tabular model with no time-series finetuning; TabPFN-TS-3 is a time-series-specific checkpoint.

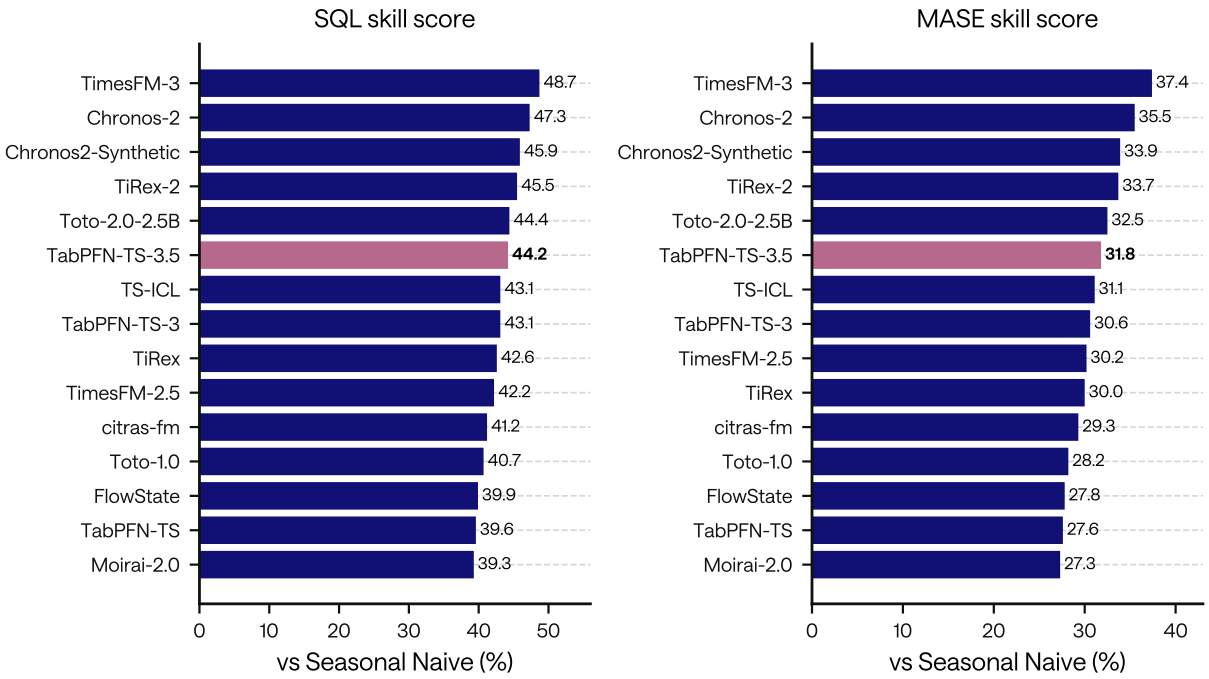



**Figure 21. fev-bench.** SQL (left) and MASE (right) skill score relative to Seasonal Naive on the 100 fev-bench tasks. Top 15 of 29 methods; each panel is sorted on its own metric.

## C.7 ScoringBench

ScoringBench [38] evaluates the full predictive distribution of regression models with proper scoring rules rather than point error. It has 101 OpenML regression datasets, each subsampled to 3,000 rows and scored with 5-fold cross-validation. Its headline metric is the continuous ranked probability score (CRPS); it also reports interval scores, coverage, energy scores and density-based rules. All methods are ranked on each dataset, with folds averaged first, and the average rank over the 101 datasets is reported.

We ran TabPFN-3.5 and TabPFN-3.5-Fast through the benchmark's own harness, with their default inference configuration and no tuning on the benchmark. The 51 baselines are the results the ScoringBench authors publish with the benchmark, including TabPFN-3, TabICLv2 and EXAONE-Tabular; we did not re-run them. To check that our runs are comparable with these results, we re-ran the released TabPFN-3 under the same code: it reproduces its published CRPS to within $10^{-3}$ relative and lands on the same mean rank.

Figure 22 shows the CRPS ranking. TabPFN-3.5 places first with a mean rank of 2.85 and TabPFN-3.5-Fast second at 5.76; TabPFN-3.5 improves on TabPFN-3 (mean rank 7.64) on 85 of the 101 datasets.

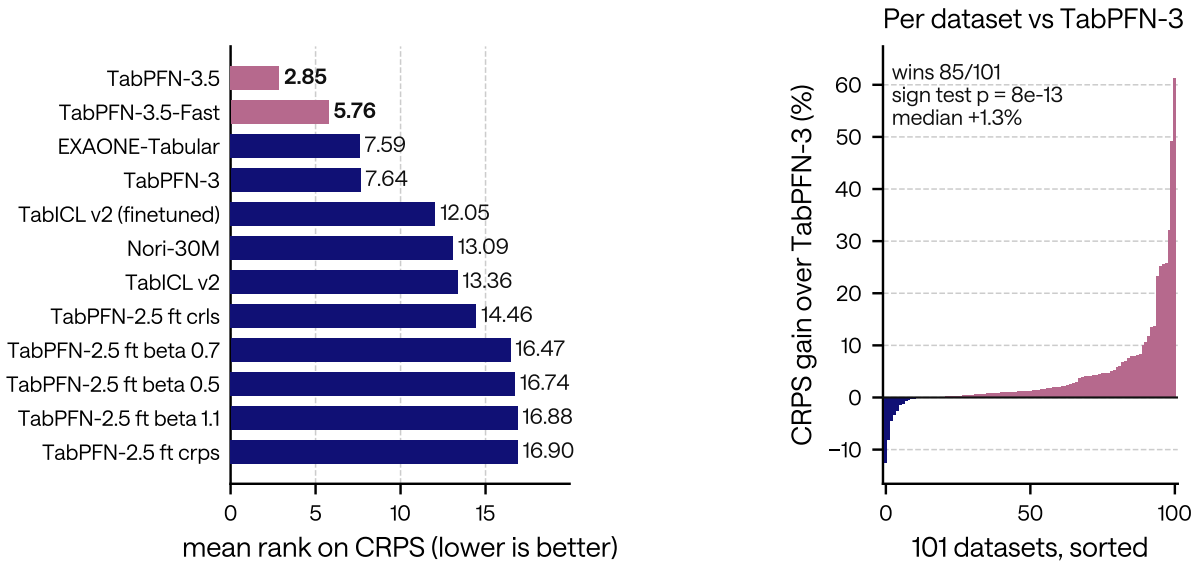



**Figure 22. ScoringBench.** Left: mean rank on CRPS across the 101 datasets for the top 12 of 53 methods; folds are averaged per dataset, methods ranked within each dataset, and ranks averaged. Lower is better; TabPFN-3.5 and TabPFN-3.5-Fast in plum. Right: per-dataset relative CRPS improvement of TabPFN-3.5 over TabPFN-3, sorted; positive bars are datasets TabPFN-3.5 wins (85 of 101, one-sided sign test $p = 8 \times 10^{-13}$, median +1.3%). Published methods are ScoringBench's own committed results; TabPFN-3 re-run under the same code reproduces its published CRPS to within $10^{-3}$ relative.

## C.8 Details on the benchmark overview table

Table 1 summarises the seven tabular benchmarks of this report in one table. This section states how each column is computed.

**Family member.** Each benchmark is represented by the best TabPFN-3.5 family member evaluated on it: TabPFN-3.5-Thinking on TabArena, BeyondArena, STRABLE and MulTaBench (on STRABLE and MulTaBench inside TabPFN-3.5-Plus, which handles the raw text columns); TabPFN-Rel (3.5), the internal preview of the TabPFN-Rel harness on TabPFN-3.5-Plus, on RelArena-$\alpha$; and TabPFN-3.5 on TALENT and ScoringBench, where no other family member was evaluated.

**Rank.** The position of that member on the benchmark's headline ranking as reported in this report: Elo on TabArena, BeyondArena, STRABLE, MulTaBench and RelArena-$\alpha$, mean rank on TALENT and ScoringBench. The denominator counts every method on that ranking except the other TabPFN-3.5 family members: the 88 entries of the TabArena field, every tuning variant counted separately (Section C.1.2); the 28 pinned BeyondArena baselines; the 196 published STRABLE pipelines of the joint Elo fit (Section C.3); the 13 published MulTaBench models; the 11 official RelArena-$\alpha$ baselines; the 35 TALENT baselines published by the TALENT authors plus TabPFN-3 (Section C.5); and the 51 published ScoringBench methods (Section C.7).

**Competitors.** The best other tabular foundation model and the best non-foundation model are the highest-placed such methods on the same ranking. Systems that contain foundation models (AutoGluon 1.6 on TabArena, AutoGluon-MM on MulTaBench) are left out of both columns; TabPFN-3.5-Thinking wins 79% of TabArena splits against AutoGluon 1.6 (noncommercial) and 82% against AutoGluon 1.6 (extreme), and 77% of MulTaBench splits against AutoGluon-MM. On RelArena-$\alpha$ the foundation-model column holds RT-PluRel, a Relational Transformer fine-tuned per task from a PluRel-pretrained checkpoint, the closest to a foundation model in that field, and the non-foundation column holds KurveRSC, a GraphReduce feature search with CatBoost; both are system entries there. The 337-dataset TALENT view has no other foundation model, so its TabICLv2 entry is taken from the 274-dataset TabICLv2-protocol view (Figure 19).

**Win rate.** As on TabArena: every split (dataset and fold) is compared on its own, a tie counts as half a win, the per-split results are averaged within a dataset and then over datasets, so every dataset has the same weight. On RelArena-$\alpha$ and TALENT a dataset has one split. TabArena has 51 datasets and 816 splits, BeyondArena 142 and 507 (the core view, group identifier exposed to the TabPFN models, Section C.2.3), STRABLE 108 and 324, MulTaBench 40 and 200, RelArena-$\alpha$ 21 tasks, TALENT 337 datasets and ScoringBench 101 datasets with 5 folds.