SCORINGBENCH: A BENCHMARK FOR EVALUATING TABULAR FOUNDATION MODELS WITH PROPER SCORING RULES

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models such as TabPFN (Hollmann et al., 2025), TabICL (Qu et al., 2026), ExaOneTabular (Eo et al., 2026), and Nori (Synthefy et al., 2026) already produce full predictive distributions, yet prevailing regression benchmarks evaluate them almost exclusively via point-estimate metrics such as RMSE. This discards precisely the distributional information these models are designed to provide — a critical gap for high-stakes domains where not all kinds of errors are equally costly. We introduce **ScoringBench**[1], an open and extensible benchmark that evaluates tabular regression models under a comprehensive suite of proper scoring rules — including CRPS, interval score, energy score, and weighted CRPS — alongside standard point metrics. ScoringBench covers 101 regression datasets from diverse domains, supports transparent community contributions via a git-based leaderboard, and provides two complementary ranking protocols: an ordinal Demšar/autorank approach and a magnitude-preserving z-score ranking approach. Evaluating several models — spanning in-context learners, fine-tuned foundation models, gradient-boosted trees, and MLPs — we find that model rankings shift substantially depending on the scoring rule: the top-performing model under one proper scoring rule may rank noticeably lower under another and models that excel on point-estimate metrics can rank poorly on probabilistic ones. These results demonstrate that the choice of evaluation metric is not a technicality but a modelling decision — and, for applications where e.g. tail errors are disproportionately costly, a domain-specific requirement with direct consequences for model deployment.

1 INTRODUCTION

The emergence of Prior-Data Fitted Networks (TabPFN) (Hollmann et al., 2022; 2025) has shifted tabular learning towards in-context learners (Qu et al., 2026) that place attention on informative examples at inference time without model weight updates. Both leading models (TabPFN and TabICLv2) perform *discretized distributional regression* and produce full predictive distributions: TabPFN predicts a probability mass function over an adaptive grid, while TabICLv2 performs multi-quantile regression (approximating the CRPS training objective with a limited number of quantiles).

Despite this, prevailing tabular benchmarks (TabArena (Erickson et al., 2025), TALENT (Liu et al., 2025)) evaluate models almost exclusively through point-estimate metrics such as RMSE or R^2, implicitly rewarding models that elicit a good conditional mean while ignoring the quality of the full predictive distribution. This evaluation mismatch is not merely cosmetic. Gneiting & Raftery (2007) proposed to use optimal score estimation as a general framework for estimation. A scoring rule $S(\mathbb{P}, y)$ measures the quality of a probabilistic forecast $\mathbb{P}$ when a realization y is observed and is *strictly proper* if the expected scoring rule is minimized if and only if the forecast $\mathbb{P}$ equals the true distribution. A detailed overview of the scoring rules used in this article is given in Appendix A.1.

[1] ScoringBench is available at `https://anonymous.4open.science/r/ScoringBench-A25F/`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: ScoringBench: Evaluating tabular regression models with proper scoring rules.

There are infinitely many proper scoring rules, each of which penalizes deviations in a particular way. Thus different rules induce different inductive biases during finite-sample training, even though they share the same population minimizer (Landsgesell et al., 2026; Waghmare & Ziegel, 2025; Buchweitz et al., 2026). Consequently, different proper scoring rules yield different model rankings (Merkle & Steyvers, 2013; Landsgesell et al., 2026) and the choice of a scoring rule for training intuitively determines which aspects of the predictive distribution are learned first. For point estimation, Gneiting (2011) argues that "the common practice of requesting *some* point forecast, and then evaluating the forecasters by using *some* (set of) scoring function(s), is not a meaningful endeavor." Instead, effective forecasting requires clear directives, such as disclosing the evaluation metric *ex ante*. The same argument applies to distributional regression: requesting a predictive distribution but not disclosing the scoring rule for evaluation risks weighting errors suboptimally for the undisclosed scoring rule (e.g. it is unclear if tail accuracy or central tendency shall be rewarded), especially in the finite-sample regime.

This inductive bias arising from the choice of scoring rule in the tabular distributional regression setting has been demonstrated in Landsgesell et al. (2026):

$$\left.\begin{aligned}\theta_1 &= \arg\min_\theta \hat{\mathbb{E}}_{(X_n,Y_n)_{n=1}^N}[S(\mathbb{P}_\theta(X_n), Y_n)],\\ \theta_2 &= \arg\min_\theta \hat{\mathbb{E}}_{(X_n,Y_n)_{n=1}^N}[S'(\mathbb{P}_\theta(X_n), Y_n)]\end{aligned}\right\} \;\not\Rightarrow\; \mathbb{P}_{\theta_1} = \mathbb{P}_{\theta_2}.$$

Here, θ_1 and θ_2 are the parameter values obtained by minimizing the two scoring rules S and S'; $\mathbb{P}_\theta(X_n)$ denotes the model's predictive distribution at input X_n, Y_n is the observed outcome, and $\hat{\mathbb{E}}_{(X_n,Y_n)_{n=1}^N}$ is the sample average $\frac{1}{N}\sum_{n=1}^N(\cdot)$. The symbol $\not\Rightarrow$ indicates that, in finite samples, different empirical scoring rules can yield different fitted predictive mappings even when the scoring rules share the same population minimizer.

There is no universally optimal scoring rule: an appropriate choice depends on the application and the associated decision context. As Martin et al. (2022) put it in the context of model selection, "one is forced to confront the fact that no such 'true model' will be revealed, and that the criterion should be defined by a score that rewards the type of predictive accuracy that matters for the problem at hand" – an observation that applies equally to the choice of scoring rule for evaluating predictive distributions. This is particularly important in high-risk applications, where different types of forecast errors may have strongly asymmetric consequences (Gneiting, 2011): some such settings may require carefully designed custom proper scoring rules (Oesterheld & Conitzer, 2020; Johnstone et al., 2011), and even when the focus is on mean estimators, Bregman divergences can help induce desired residual structures (Gneiting, 2011).

These differences have practical implications for model selection and deployment, motivating benchmarks that report a comprehensive suite of proper scoring rules alongside standard point metrics.

ScoringBench (Figure 1) implements these ideas in an extensible, reproducible evaluation harness covering a broad set of scoring rules. Our fine-tuning experiments illustrate how changing the training score affects predictive behavior.

1.1 Contributions

We contribute

- ScoringBench (`https://anonymous.4open.science/r/ScoringBench-A25F/`), an easy-to-install benchmark for distributional regression with minimal dependencies.
- a git-based leaderboard (`https://incredible-truffle-9624b2.netlify.app/`), where updates happen transparently with full traceability and reproducibility.
- a rigorous benchmark protocol using Autorank (following Demšar (2006); Herbold (2020)).

2 Related Work

Tabular Deep Learning. Prior-Data Fitted Networks (TabPFN) (Hollmann et al., 2022; 2025) have shifted tabular learning from the dominant paradigm of gradient-boosted decision trees toward transformer-based in-context learners. Rather than updating model weights at fit time, PFNs perform in-context learning in a single forward pass. TabPFN predicts probability masses over an adaptive grid, while TabICLv2 (Qu et al., 2026), ExaOneTabular (Eo et al., 2026) and Nori (Synthefy et al., 2026) estimates multiple conditional quantiles.

Probabilistic Forecasting. In a seminal review and theoretical development, Gneiting & Raftery (2007) surveyed proper scoring rules and their foundations, and discussed examples for probabilistic forecasts of densities, distributions, quantiles, and intervals. Gneiting et al. (2007) further formalized the goal of probabilistic forecasting as maximizing *sharpness* subject to *calibration*, providing the conceptual basis for the diagnostics reported in ScoringBench. Importantly, while all strictly proper scoring rules share the same population-level minimizer, they induce different inductive biases during finite-sample training (Waghmare & Ziegel, 2025). Merkle & Steyvers (2013) demonstrated empirically that model rankings change substantially depending on which proper scoring rule is used (the Spearman correlation between Brier-score and log-score rankings in their study is only 0.15), and Buchweitz et al. (2026) showed that a broad class of proper scoring rules—including the logarithmic, CRPS, quadratic, and energy scores—penalize under- and overestimation asymmetrically. The scoring rules computed in ScoringBench span a wide range of sensitivities: the CRPS (Gneiting & Raftery, 2007) weights all quantile levels equally; the energy score (Gneiting & Raftery, 2007) generalizes CRPS via the parameter $\beta \in (0, 2)$, interpolating between median- and mean-optimal behavior (Landsgesell et al., 2026). The interval score (Gneiting & Raftery, 2007) targets specific central prediction intervals; and the weighted CRPS (Gneiting & Ranjan, 2011) allows practitioners to emphasize particular regions of the predictive distribution. Together, these rules enable a multifaceted assessment of forecast quality that no single metric can provide. Additionally, we provide scoring rules which evaluate the density estimate (density power divergence scores and pseudo spherical scores, see appendix).

Benchmarking Methodology. Two prominent tabular benchmarks, TabArena (Erickson et al., 2025) and TALENT (Liu et al., 2025), evaluate models exclusively on point-estimate metrics such as RMSE. These metrics reward accurate conditional-mean predictions but are blind to the quality of the full predictive distribution.

Landsgesell et al. (2026) showed that fine-tuning realTabPFNv2.5 (Garg et al., 2025; Grinsztajn et al., 2025) with a scoring rule different from the pretraining objective shifts the model's inductive bias and changes its rankings across models. Other recent work, including Izbicki & Rodrigues (2026), has similarly highlighted the limitations of relying solely on point-estimate metrics; Izbicki et al. propose an additional loss termed "conditional density estimation", which can be viewed as a Brier-type loss for continuous outcomes (Rudemo, 1982). Building on this line of work (Landsgesell et al., 2026), ScoringBench offers a broad framework in which any admissible proper scoring rule can be used for evaluating the quality of predictive distributions. The benchmark establishes a reproducible, unified evaluation protocol for distributional regression.

3 SCORINGBENCH SETUP

3.1 BENCHMARKING WITH PROPER SCORING RULES

We refer the reader to Gneiting & Raftery (2007); Gneiting & Ranjan (2011) for a comprehensive introduction to proper scoring rules and Landsgesell et al. (2026) for a discussion in the tabular distributional regression setting. All definitions follow Gneiting & Raftery (2007); Gneiting & Ranjan (2011) unless otherwise noted; full formulae, propriety classes, and implementation details are provided in the Appendix A.1 and Appendix A.2.

Since ScoringBench is not tied to a single downstream application, we report a representative suite of commonly used and complementary proper scoring rules alongside standard point metrics (R^2, RMSE, MAE): the **CRPS**, the β-**energy score** (ES_β, $\beta \in (0, 2)$), the **continuous ranked logarithmic score** (CRLS), the **density power divergence** (DPD), the **interval score** (IS), the **quantile-weighted CRPS** (wCRPS), and the **CDE/quadratic (Brier-type) loss**. We use the loss convention throughout, so lower values indicate better forecasts. The scoring rule and its weighting should be specified according to the aspects of the predictive distribution — e.g. tail behavior versus central tendency — that are relevant to the downstream application; we refer to Merkle & Steyvers (2013); Oesterheld & Conitzer (2020) for detailed discussions of the selection and design of scoring rules. As a representative example, given a predictive CDF $F_\mathbb{P}$ and an observation y, the CRPS is

$$\mathrm{CRPS}(\mathbb{P}, y) = \int_{-\infty}^{\infty} \left(F_\mathbb{P}(x) - \mathbb{1}_{\{x \geq y\}}\right)^2 \mathrm{d}x.$$

When $\mathbb{P}$ collapses to a point mass at some $\hat{y}$, the CRPS reduces to the absolute error $|y - \hat{y}|$, recovering point-estimate evaluation as a special case. For univariate predictive distributions, the CRPS is furthermore the special case $\beta = 1$ of the β-energy score. Formulae for the remaining scoring rules, together with their natural domains of definition and further implementation details, are given in Appendix A.1.

3.2 DATASETS

Many existing tabular benchmarks assemble datasets via semi-automated pipelines (Gijsbers et al., 2024; Jiang et al., 2026; Grinsztajn et al., 2022) or manual curation (Erickson et al., 2025; Grinsztajn et al., 2022). Building on these efforts, ScoringBench curates a heterogeneous collection of regression problems spanning finance, physics, engineering, and environmental science.

Our dataset collection incorporates the three OpenML (Bischl et al., 2025) regression suites 269, 297, and 299 (Dheur & Taieb, 2023). We supplement these suites with datasets from three additional repositories: **PMLB** (Olson et al., 2017), **KEEL** (Derrac et al., 2015), and **TALENT** (Liu et al., 2025). We drop duplicated datasets by filtering the names and also drop datasets that have only few unique target values (classification, not regression). Lightweight, uniform preprocessing is applied (e.g. numeric median imputation); see Appendix B.3 for details on the imputation and preprocessing pipeline. The complete list of datasets retained after deduplication and validation, together with per-dataset details and selection criteria, is provided in Appendix B.

For each dataset, models (see Section 3.3) are evaluated using 5-fold cross-validation with a fixed random seed. We limit each training fold to at most 2400 observations and each test fold to at most 600 observations. An ablation with regards to the number of observations is provided in Appendix E.

3.3 EVALUATED MODELS

We evaluate a mix of in-context learners and conventional, non-pretrained baselines. At the time of writing, the in-context learners that support distributional prediction are **realTabPFNv2.5** (and several fine-tuned variants of realTabPFNv2.5), **TabPFNv2.6**, **TabPFNv3** (Grinsztajn et al., 2026), **TabICLv2** (Qu et al., 2026), **ExaOneTabular** (Eo et al., 2026), and Nori (Synthefy et al., 2026). We expect this list to grow as further models add distributional prediction.[2] Our non-pretrained

[2]This is precisely the advantage of maintaining ScoringBench as a living benchmark: rather than being fixed at publication time, it continues to track newly released tabular foundation models on the git-based leaderboard. Since this manuscript was written, **TabDPT** (Hosseinzadeh et al., 2026), **Mitra-v2** (Tao

baselines include **XGBoost vector-leaf** models adapted for distributional regression, **XGBoostLLS** (März, 2019), and **CatBoost** (Prokhorenkova et al., 2018). We also use **CREPES** (Boström, 2024) to construct conformal predictive distributions from XGBoost, CatBoost, and TabICLv2 predictions. We also evaluate **TabM** (Gorishniy et al., 2024), a compact multilayer perceptron that leverages parameter-efficient ensembling, and **RealMLP** (Holzmüller et al., 2024), an MLP architecture with extensive hyperparameter tuning designed for robust out-of-the-box performance on diverse tabular tasks.

3.4 Implementation

The models under consideration originally represent their predictive distributions in different forms, including histograms, quantiles, samples, and probability density functions. For evaluation in ScoringBench, model-specific wrappers convert these outputs into a common histogram representation. For a given observation i, this representation consists of bins $I_{ik} = [a_{ik}, b_{ik}]$ and associated probability masses p_{ik}, with $\sum_k p_{ik} = 1$. For density-based scoring rules and models without native histogram representations, we adapt Izbicki & Rodrigues (2026) to enable reliable conditional density estimation: we monotonically interpolate the cumulative distribution function (CDF), create a histogram with equal bin widths, and compute the probability mass in each bin by evaluating the interpolated CDF at the bin edges ($p_{ik} = F(b_{ik}) - F(a_{ik})$) which is effectively a smoothing scheme.

Within each bin, the probability is assumed to be distributed uniformly, resulting in the piecewise-constant density $f_i(x) = p_{ik}/(b_{ik} - a_{ik})$, $x \in I_{ik}$. This unified representation allows the required quantities to be evaluated analytically within each bin. For example, for positive-width bins, the first term of the CRPS is then computed as

$$\mathbb{E}|X - y_i| = \sum_k p_{ik} \frac{1}{b_{ik} - a_{ik}} \int_{a_{ik}}^{b_{ik}} |x - y_i| \, \mathrm{d}x,$$

which can be computed analytically. Other scoring rules are calculated analogously.

3.5 Ranking

We use two complementary ranking strategies; full details are in Appendix D. The benchmark produces scores $s_{m,d,f,p}$ for each tuple

$$(\text{model } m,\ \text{dataset } d,\ \text{fold } f,\ \text{metric } p),$$

across M models, D datasets, F folds, and P metrics (in total $M \cdot D \cdot F \cdot P$ observations). Before ranking, we collapse across folds by averaging,

$$\bar{s}_{m,d,p} \ = \ \frac{1}{F} \sum_{f=1}^{F} s_{m,d,f,p} \, ,$$

reducing the data to per-(model, dataset, metric) scores. This step is essential to avoid *pseudoreplication*: fold-level scores within a dataset are correlated, so treating them as independent observations would inflate the effective sample size and distort subsequent tests (Lazic, 2010). After fold aggregation the data has shape $M \times D \times P$, with entries $\bar{s}_{m,d,p}$ denoting the fold-averaged score of model m on dataset d for metric p.

The two ranking strategies differ in how they aggregate these per-dataset summaries across datasets:

Demšar–autorank approach (Autorank). Following Demšar (Demšar, 2006) and the `autorank` library (Herbold, 2020), models are ranked within each dataset (best = rank 1), mean ranks are computed across datasets, and a non-parametric Friedman omnibus test evaluates whether mean ranks differ across models. If the omnibus test is significant, Nemenyi post-hoc comparisons (visualized via a critical-difference diagram) identify which model pairs differ; the Nemenyi test is automatically selected by autorank as the appropriate non-parametric post-hoc procedure for

et al., 2026), **LimiX-2** (Zhang et al., 2026), **Causilo** (Cho et al., 2026), and **Xiaomi-TabLDM** (Team et al., 2026) have already been added; up-to-date results for these and future models are available at `https://incredible-truffle-9624b2.netlify.app/`.

this Demšar-based approach. Autorank also reports Akinshin's γ as a non-parametric analogue of Cohen's d.[3] Thresholds on the absolute effect size $|\gamma|$ used to categorize magnitudes in the Autorank library follow the common practice for Cohen's d: $|\gamma| < 0.2$ is negligible; $0.2 \leq |\gamma| < 0.5$ is small; $0.5 \leq |\gamma| < 0.8$ is medium; $|\gamma| \geq 0.8$ is large. This ordinal approach is robust to outliers and to heteroskedastic score scales, but it discards magnitude information by relying on ranks. Some more details are provided in Appendix D.1. For results see the Autorank-based heatmap in Figure 2a.

Per-dataset standardized performance approach (z-score ranking). To preserve magnitude information while controlling for dataset difficulty, we standardize the fold-averaged scores within each dataset (compute z-scores across the M models, negating the sign for metrics where smaller is better), then average these standardized scores across datasets and rank models by their mean standardized performance. The full details are given in Appendix D.2. This produces magnitude-preserving, comparable scores across datasets that account for differing score scales and difficulties. See the effect-size heatmap in Figure 2b.

We refer to the Demšar–autorank procedure as **Autorank** and the per-dataset standardized performance procedure as **z-score ranking**; subsequent figures and tables use these labels. The two approaches are complementary (ordinal significance vs. magnitude-aware ranking); for implementation details see Appendix D. Since both produce highly correlated rankings across metrics (mean Pearson $r \approx 0.97$; see Appendix D.3), we report only the **Autorank**-based leaderboards in the main text. Full z-score rankings and per-metric correlation statistics are available on the live leaderboard and may be relevant for readers interested in magnitude-aware differences between models.

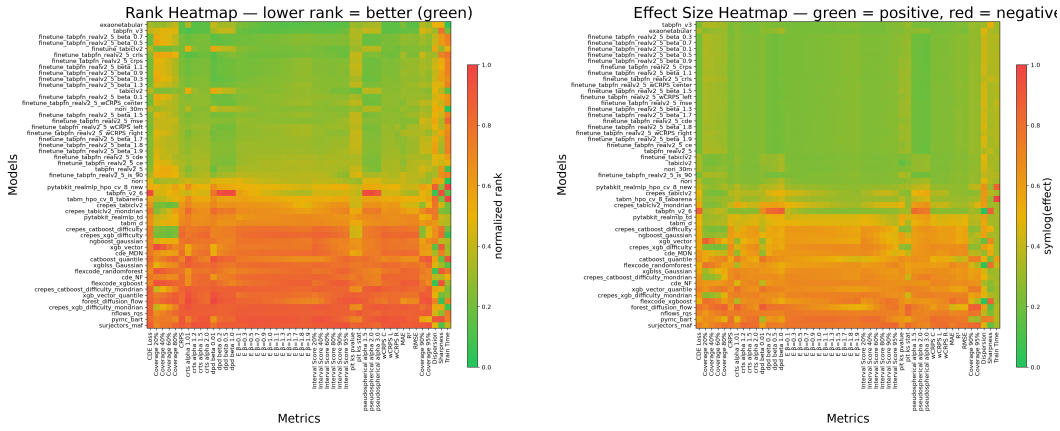
4 Results of Benchmark 2026

The reported results represent a snapshot from 8th September 2026.

Our website

`hhttps://incredible-truffle-9624b2.netlify.app/`

reports the live leaderboard with all models and their respective rankings. Hyperparameters of the models are documented in the git repository. The leaderboard is updated with every new pull request, and all changes are tracked in the git history for full reproducibility, transparency and agility.



(a) Autorank (b) z-score ranking

Figure 2: Ranking heatmaps summarizing the performance of different models on different scoring rules.

[3] `https://aakinshin.net/posts/nonparametric-effect-size/`

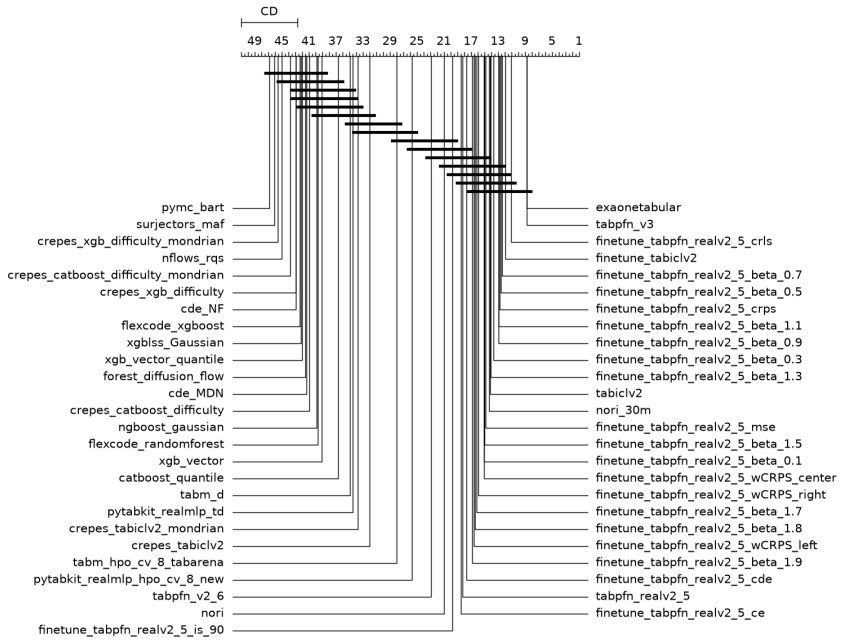


Figure 3: Critical Difference (CD) diagram for CRPS: Models are positioned by autorank on a horizontal axis according to their mean rank $\bar{r}_{m,p}$ (see Eq. (1)), and those whose mean-rank difference does not exceed CD are connected by a horizontal bar, indicating no statistically significant difference in performance at the $\alpha = 0.05$ level. Critical difference (CD) is defined in Eq. (2) (Demšar, 2006).

The results confirm the central finding of Landsgesell et al. (2026): the ranking of different models depends on the choice of scoring rule, and reporting only point-estimate metrics risks obscuring these differences.

Table 3 reports Autorank-based rankings for CRPS (lower is better); column headers are defined in the table caption. The leading models are `exaonetabular` and `tabpfn_v3` (Grinsztajn et al., 2026) (mean ranks 8.72 and 8.73), followed by `finetune_tabpfn_realv2_5_crls` (mean rank 11.08). Several fine-tuned TabPFN variants also rank ahead of the unfine-tuned `tabpfn_realv2_5` (mean rank 18.21). While Akinshin's γ values relative to the top-ranked model remain in the "negligible" range for these models, the rank ordering is consistent across datasets and robust to dataset size (see Appendix E), suggesting that fine-tuning induces subtle but stable shifts in predictive behavior. Magnitude categories use $|\gamma|$ and follow the common practice for Cohen's d according to Romano et al. (2006).

The critical-difference diagram in Figure 3 shows that tabular foundation models provide substantial improvements over traditional baselines: the top (fine-tuned) variants form a cluster that is statistically significantly better than the baselines such as `xgb_vector` and `xgblss_Gaussian`.

While the importance of aligning the scoring rule used for training with application-specific utility is well-established among leading statisticians — most notably popularized by Gneiting & Raftery (2007); Gneiting & Ranjan (2011); Gneiting (2011) and colleagues — its practical implementation in terms of optimal score estimation remains frequently overlooked.

In general, we observe—consistent with (Landsgesell et al., 2026)—that the ranking of models depends on the specific scoring rule used to penalize errors (or the chosen scoring function for point estimates (Gneiting, 2011)). This can also be seen by comparing the different leaderboards Table 3 and Table 6, see e.g. the entries for TabPFNv2.6. In view of Figure 2, we observe correlations between different scoring rules, in particular between scoring rules belonging to the same scoring rule family. For an investigation of this effect, we point to (Landsgesell et al., 2026, Figure 10).

We also observe that conformal methods like CREPES, when coupled with XGBoost, CatBoost, or TabICLv2, can achieve excellent nominal coverage (see Table 5). However, while coverage improves, the interval score and other proper scoring rules deteriorate (see Table 4).

This tradeoff reflects a fundamental distinction between two different evaluation objectives. Conformal methods optimize for agreement with nominal coverage, while proper scoring rules reward both calibration *and* sharpness simultaneously (Gneiting & Raftery, 2007). Since TabICLv2 is trained on CRPS, wrapping it with a coverage-focused method compromises the distributional quality it was designed to achieve.

5 Conclusion

ScoringBench is an open, extensible benchmark that closes the evaluation gap between the full predictive distributions produced by tabular foundation models and the point estimates measured by prevailing benchmarks. By reporting a comprehensive suite of proper scoring rules alongside standard metrics, ScoringBench enables a more faithful assessment of probabilistic forecast quality, especially for applications where the cost of errors is asymmetric.

Our results support three main findings. First, model rankings depend on the chosen scoring rule: point-estimate metrics (scoring functions) such as RMSE can paint a different picture than proper scoring rules that evaluate full predictive distributions. Second, fine-tuning realTabPFN with alternative scoring rules tends to produce small effect sizes (Akinshin's γ often in the negligible range) but yields consistent directional shifts across datasets and metrics. Third, pretrained in-context models achieve substantial improvements over gradient-boosted baselines on probabilistic metrics, indicating genuine recent progress in distributional regression.

We therefore recommend that benchmarks include a diverse suite of proper scoring rules and present both rank-based and magnitude-aware summaries. For high-risk applications, practitioners should select (Gneiting, 2011) or design (Merkle & Steyvers, 2013; Oesterheld & Conitzer, 2020) scoring rules that encode domain-specific utility. For example, weighted CRPS suits settings where tail errors are disproportionately costly.

A natural next step is multivariate distributional regression, where predictive quality depends on the dependence structure between targets and not only on the marginals. Multivariate proper scoring rules such as the energy score (Gneiting & Raftery, 2007) and the variogram score (Scheuerer & Hamill, 2015) assess this joint structure, but their discriminative power weakens as the dimension grows (Pinson & Tastu, 2013). The choice of scoring rule also matters beyond regression: training language models with strictly proper scoring rules other than the logarithmic score can improve generation quality (Shao et al., 2024).

We see ScoringBench as a living resource and welcome community contributions of new datasets, models, and scoring rules via pull request. The code and contribution guide are available at `https://anonymous.4open.science/r/ScoringBench-A25F/`. The live leaderboard can be found at `https://incredible-truffle-9624b2.netlify.app/`.

Limitations ScoringBench currently takes into account only scalar valued output datasets. However this does not seem to be a structural limitation, since there are also proper scoring rules for vector valued outputs.

AI use statement

In this work, we used generative AI tools for providing feedback on experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, interpreting results. "Formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs" are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures, suggesting experimental parameters, creating and editing software code, drafting parts of a research paper, summarizing or analysing existing literature,

sourcing/searching for information, editing a research paper to improve readability, identifying relevant literature.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested by all three authors, all papers that were found by AI were reviewed, understood and approved each by at least two authors. the drafted text was reviewed, manually edited and approved by all three authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

ScoringBench evaluates probabilistic regression models to improve understanding of how their performance depends on the choice of scoring rule. We do not identify immediate ethical risks arising from the benchmark itself. However, benchmark performance alone does not establish that a model is suitable for a particular application, especially where errors may affect people. Users should also assess fairness, privacy, and the consequences of prediction errors in their intended setting.

REPRODUCIBILITY STATEMENT.

The ScoringBench code is available at the link provided in the paper. The repository README explains how to rerun the benchmark and reproduce the reported results. We describe the computational resources used for the experiments in the paper, and the ScoringBench website is publicly available.

REFERENCES

Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, et al. OpenML: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7), 2025.

Henrik Boström. Conformal prediction in python with crepes. In *Proc. of the 13th Symposium on Conformal and Probabilistic Prediction with Applications*, pp. 236–249. PMLR, 2024.

Erez Buchweitz, João Vitor Romano, and Ryan J Tibshirani. Asymmetric penalties underlie proper loss functions in probabilistic forecasting. *Biometrika*, pp. asag042, 2026.

Minyong Cho, Minho Jeong, Dooho Lee, Jinmo Lee, and Jaemin Yoo. Causilo technical report. 2026. URL `https://arxiv.org/abs/2609.22866`.

Claudia Czado, Tilmann Gneiting, and Leonhard Held. Predictive model assessment for count data. *Biometrics*, 65(4):1254–1261, 2009.

A Philip Dawid. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2): 278–290, 1984.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

J Derrac, S Garcia, L Sanchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Logic Soft Comput*, 17:255–287, 2015.

Victor Dheur and Souhaib Ben Taieb. A large-scale study of probabilistic calibration in neural network regression. In *International Conference on Machine Learning*, pp. 7813–7836. PMLR, 2023.

Francis X. Diebold, Todd A. Gunther, and Anthony S. Tay. Evaluating density forecasts with applications to financial risk management. *International Economic Review*, 39(4):863–883, 1998.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE Tabular 1.0: Technical Report. *arXiv preprint arXiv:2608.25774*, 2026.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025.

Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. OpenML-CTR23–a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023.

Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. Real-TabPFN: Improving tabular foundation models via continued pre-training with real-world data. *arXiv preprint arXiv:2507.03971*, 2025.

Abhik Ghosh and Ayanendranath Basu. Robust estimation for independent non-homogeneous observations using density power divergence with applications to linear regression. 2013.

Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

Tilmann Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106(494):746–762, 2011.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Tilmann Gneiting and Roopesh Ranjan. Comparing density forecasts using threshold-and quantile-weighted scoring rules. *Journal of Business & Economic Statistics*, 29(3):411–422, 2011.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(2): 243–268, 03 2007. ISSN 1369-7412. doi:10.1111/j.1467-9868.2007.00587.x. URL https://doi.org/10.1111/j.1467-9868.2007.00587.x.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing tabular deep learning with parameter-efficient ensembling. *arXiv preprint arXiv:2410.24210*, 2024.

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.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. TabPFN-3: Technical Report. *arXiv preprint arXiv:2605.13986*, 2026.

Steffen Herbold. Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173, 2020. doi:10.21105/joss.02173. URL https://doi.org/10.21105/joss.02173.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

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.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by Default: Strong Pre-Tuned MLPs and Boosted Trees on Tabular Data. *Advances in Neural Information Processing Systems*, 37: 26577–26658, 2024.

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.

Rafael Izbicki and Ann B Lee. Nonparametric conditional density estimation in a high-dimensional regression setting. *Journal of Computational and Graphical Statistics*, 25(4):1297–1316, 2016.

Rafael Izbicki and Pedro LC Rodrigues. Benchmarking tabular foundation models for conditional density estimation in regression. *arXiv preprint arXiv:2603.26611*, 2026.

Dihong Jiang, Ruoqi Cao, Zhiyuan Dang, Li Huang, Qingsong Zhang, Zhiyu Wang, Shihao Piao, Shenggao Zhu, Jianlong Chang, Zhouchen Lin, et al. OmniTabBench: Mapping the empirical frontiers of GBDTs, neural networks, and foundation models for tabular data at scale. *arXiv preprint arXiv:2604.06814*, 2026.

David J Johnstone, Victor Richmond R Jose, and Robert L Winkler. Tailored scoring rules for probabilities. *Decision Analysis*, 8(4):256–268, 2011.

I. Juutilainen, S Tamminen, and J Röning. Exceedance probability score: A novel measure for comparing probabilistic predictions. *Journal of Statistical Theory and Practice*, 6(3):452–467, 2012.

Fabian Krüger, Sebastian Lerch, Thordis Thorarinsdottir, and Tilmann Gneiting. Predictive inference based on markov chain monte carlo output. *International Statistical Review*, 89(2): 274–301, 2021. doi:https://doi.org/10.1111/insr.12405. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12405.

Jonas Landsgesell, Pascal Knoll, and Tizian Wenzel. Distributional regression with tabular foundation models: Evaluating probabilistic predictions via proper scoring rules. *arXiv preprint arXiv:2603.08206*, 2026.

Stanley E Lazic. The problem of pseudoreplication in neuroscientific studies: is it affecting your analysis? *BMC neuroscience*, 11(1):5, 2010.

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.

Gael M. Martin, Rubén Loaiza-Maya, Worapree Maneesoonthorn, David T. Frazier, and Andrés Ramírez-Hassan. Optimal probabilistic forecasts: When do they work? *International Journal of Forecasting*, 38(1):384–406, 2022.

Alexander März. XGBoostLSS–an extension of XGBoost to probabilistic forecasting. *arXiv preprint arXiv:1907.03178*, 2019.

Edgar C. Merkle and Mark Steyvers. Choosing a strictly proper scoring rule. *Decision Analysis*, 10 (4):292–304, 2013.

Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

Caspar Oesterheld and Vincent Conitzer. Decision scoring rules. In *WINE*, pp. 468, 2020.

Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1):36, 2017.

Pierre Pinson and Julija Tastu. Discrimination ability of the energy score. 2013.

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.

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. URL https://openreview.net/forum?id=SxsyLjIfWB.

Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, and Jeff Skowronek. Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys. In *annual meeting of the Florida Association of Institutional Research*, volume 177, 2006.

Mats Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, pp. 65–78, 1982.

Michael Scheuerer and Thomas M Hamill. Variogram-based proper scoring rules for probabilistic forecasts of multivariate quantities. *Monthly Weather Review*, 143(4):1321–1334, 2015.

Chenze Shao, Fandong Meng, Yijin Liu, and Jie Zhou. Language generation with strictly proper scoring rules. *arXiv preprint arXiv:2405.18906*, 2024.

Synthefy, Po-han Li, Aditya Narayanan, Sai Shankar Narasimhan, Raghav Mallampalli, Aahan Agrawal, Bekzat Ajan, Raimi Shah, and Shubhankar Agarwal. Synthefy nori: Tabular foundation model for regression, June 2026. URL https://doi.org/10.5281/zenodo.20710462.

Yefan Tao, Xiyuan Zhang, Xinyi Liu, Boran Han, Danielle Maddix, Haoyang Fang, Zhen Han, Jiading Gai, Xuanqing Liu, Michael Bohlke-Schneider, et al. Mitra-v2 technical report. *arXiv preprint arXiv:2609.04540*, 2026.

TabLDM Team, Penghui Wang, Wei Liu, Hong Wang, Chengyue Huang, Yuxi Sun, Zirui Wang, Hongming Huang, Quan Wang, Chunxiao Liu, et al. Xiaomi-TabLDM: A Tabular Foundation Model Technical Report. *arXiv preprint arXiv:2609.03880*, 2026.

Kevin Tran, Willie Neiswanger, Junwoong Yoon, Qingyang Zhang, Eric Xing, and Zachary W Ulissi. Methods for comparing uncertainty quantifications for material property predictions. *Machine Learning: Science and Technology*, 1(2):025006, 2020.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Kartik Waghmare and Johanna Ziegel. Proper scoring rules for estimation and forecast evaluation. *Annual Review of Statistics and Its Application*, 13, 2025.

Xingxuan Zhang, Gang Ren, Hao Yuan, Hao Zou, Hongze Tan, Hui Wang, Jianhao Song, Jiansheng Li, Jiayao Zhang, Jinghan Zhang, et al. Limix-2: A contextual mechanism network towards general structured-data intelligence. *arXiv preprint arXiv:2609.17488*, 2026.

A Scoring Rules and Diagnostic Metrics

A.1 Scoring rules

This appendix collects concise definitions of the scoring rules and diagnostic metrics used in ScoringBench. Full implementation notes are available in the repository. We use the loss convention throughout, so lower values indicate better forecasts.

In the following, let $F_{\mathbb{P}}$ be the cumulative distribution function (CDF) of the probability distribution $\mathbb{P}$, let $f_{\mathbb{P}}$ be the predictive density of $\mathbb{P}$ and let y be an observation. Before giving the definitions of the most important scoring rules, we recall that (strict) propriety is relative to the class of admissible predictive distributions (Krüger et al., 2021; Gneiting & Raftery, 2007). For example, the logarithmic score is strictly proper with respect to the class $\mathcal{L}_1$ of absolutely continuous probability measures, while the CRPS is strictly proper with respect to the class $\mathcal{M}_1$ of Borel probability measures that have finite first moment (Krüger et al., 2021; Gneiting & Raftery, 2007). For details on the scoring rules and their properties, we refer to (Gneiting & Raftery, 2007). This distinction matters in practice: a scoring rule that is inapplicable, or numerically unstable can favor or penalize models in an inappropriate way. Therefore, when comparing models across scoring rules, as ScoringBench does, it is therefore important to verify that the kind of output prediction of the model is valid for the scoring rule used. For density-based scores, we smooth potential atoms (e.g., from tied quantiles) by monotone CDF interpolation and equal-width binning (Izbicki & Rodrigues, 2026). CRPS and energy scores use native forecasts without this smoothing.

Continuous Ranked Probability Score (CRPS). The CRPS is given as

$$\mathrm{CRPS}(\mathbb{P}, y) = \int_{-\infty}^{\infty} (F_{\mathbb{P}}(x) - \mathbb{1}_{\{x \geq y\}})^2 \, \mathrm{d}x.$$

For univariate probability distributions, the CRPS is a special case of the β-energy score $ES_{\beta=1}$.

β-energy score. For any $\beta \in (0, 2)$, the β-energy score is given as

$$ES_\beta(\mathbb{P}, y) = \mathbb{E}_{X \sim \mathbb{P}} \|X - y\|^\beta - \tfrac{1}{2} \, \mathbb{E}_{X, X' \sim \mathbb{P}} \|X - X'\|^\beta,$$

where $X, X' \sim \mathbb{P}$ are independent draws from $\mathbb{P}$. Smaller values of β reduce the relative influence of large errors, while values of β closer to 2 emphasize large deviations more strongly. In the limiting case $\beta \to 2$, the MSE is obtained.

Continuous Ranked Logarithmic Score (CRLS). Following Juutilainen et al. (2012), the CRLS is given as:

$$\mathrm{CRLS}(\mathbb{P}, y) = -\int_{-\infty}^{\infty} \log |F_{\mathbb{P}}(x) + \mathbb{1}_{\{y \leq x\}} - 1| \, \mathrm{d}x,$$

Density Power Divergence (DPD) scoring rule. Following Ghosh & Basu (2013), the DPD is given as:

$$DPD_\beta(\mathbb{P}, y) = \int_{-\infty}^{\infty} f_{\mathbb{P}}(t)^{1+\beta} \, \mathrm{d}t - \left(1 + \frac{1}{\beta}\right) f_{\mathbb{P}}(y)^\beta, \qquad \beta > 0.$$

In the limiting case $\beta \to 0$ this recovers the log score up to an additive constant:

$$DPD_0(\mathbb{P}, y) = -\log f_{\mathbb{P}}(y) + \text{const}.$$

Note for $\beta = 1$ this corresponds to the CDE loss, which is a continuous generalization of the Brier score.

Interval Score. The Interval Score (IS) (Gneiting & Raftery, 2007) evaluates the $(1-\alpha)$ central prediction interval:

$$S_\alpha(\mathbb{P}, y) = (u - l) + \frac{2}{\alpha}(l - y)\mathbb{1}_{\{y < l\}} + \frac{2}{\alpha}(y - u)\mathbb{1}_{\{y > u\}},$$

where $[l, u]$ are the $\alpha/2$ and $1-\alpha/2$ quantiles of the CDF $F_{\mathbb{P}}$.

Quantile Weighted CRPS (wCRPS). The weighted CRPS can be written as an integral of the pinball loss ρ_α over quantile levels $\alpha \in [0, 1]$. Using a weight function $w(\alpha)$ to emphasise regions of interest, a convenient representation is

$$L_{\mathrm{wCRPS}}(\mathbb{P}, y) = 2 \int_0^1 w(\alpha)\, \rho_\alpha\big(y - F_{\mathbb{P}}^{-1}(\alpha)\big)\, \mathrm{d}\alpha,$$

where the pinball loss is defined by

$$\rho_\alpha(u) = (\alpha - \mathbb{1}_{\{u<0\}})\, u.$$

The weight functions for the left, center, and right cases are given, respectively, by $w(\alpha) = (1-\alpha)^2$, $w(\alpha) = \alpha(1-\alpha)$, $w(\alpha) = \alpha^2$.

The CDE Loss as Continuous Brier Score The Conditional Density Estimation (CDE) loss (sometimes quadratic score) (Rudemo, 1982; Izbicki & Lee, 2016) generalises the Brier Score (Gneiting & Raftery, 2007) to continuous outcomes by minimising the L^2 distance between the true density f and an estimator $\hat{f}$:

$$L(f, \hat{f}) = \int \big(f(z) - \hat{f}(z)\big)^2\, \mathrm{d}z.$$

Dropping the estimator-independent term $\int f^2(z)\, \mathrm{d}z$ yields the modified CDE loss functional used for estimation:

$$\mathcal{L}_{\mathrm{CDE}}(\hat{f}, y) = \int \hat{f}^2(z)\, \mathrm{d}z - 2\, \hat{f}(y).$$

This expression is directly analogous to the discrete Brier loss $\sum_k p_k^2 - 2p_y$, where the integral of the squared density replaces the sum of squared probabilities and the density value at the observation replaces the "hit" probability. Importantly, numerical stability of $\mathcal{L}_{\mathrm{CDE}}$ depends on well-defined supports and strictly monotonic bin boundaries (for example, sorted quantiles in discretised estimators). Because $\mathcal{L}_{\mathrm{CDE}}$ is strictly proper, it incentivises calibrated predictive densities.

A.2 Diagnostic Metrics

Sharpness. Sharpness measures the concentration of the predictive distribution. Following Tran et al. (2020) we compute sharpness from the model's estimated predictive CDFs $\widehat{F}_n$ for each test point n as the average predicted standard deviation:

$$\mathrm{Sharp}(\widehat{F}) = \frac{1}{N} \sum_{n=1}^{N} \sqrt{\mathrm{var}(\widehat{F}_n)} = \frac{1}{N} \sum_{n=1}^{N} \sqrt{\mathbb{E}_{\widehat{F}_n}[X^2] - \big(\mathbb{E}_{\widehat{F}_n}[X]\big)^2}.$$

Here, $\widehat{F}_n$ denotes the model's estimated predictive CDF for test sample n, and we write $X \sim \widehat{F}_n$. Define $\sigma_n = \sqrt{\mathrm{var}(\widehat{F}_n)}$ and the mean predicted uncertainty $\bar{\sigma} = \frac{1}{N}\sum_{n=1}^{N} \sigma_n$. Sharpness is not a proper scoring rule; it should be considered jointly with other scoring rules. A well-calibrated model with low sharpness (small predicted standard deviations) provides more actionable uncertainty estimates than one that is calibrated but diffuse (Tran et al., 2020).

Dispersion. Dispersion measures heterogeneity in the model's predicted uncertainties across the test set. Using the notation above, let $\sigma_n = \sqrt{\mathrm{var}(\widehat{F}_n)}$ be the predicted standard deviation for sample n. Then dispersion is the cross-sample standard deviation of these predicted uncertainties:

$$\mathrm{Disp}(\widehat{F}) = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} (\sigma_n - \bar{\sigma})^2},$$

where $\bar{\sigma} = \frac{1}{N}\sum_{n=1}^{N} \sigma_n$. Dispersion is not a proper scoring rule; it answers the question "how much does the model's confidence vary across samples?". A low-dispersion model assigns similar uncertainty estimates to all instances, while a high-dispersion model distinguishes some instances as

more predictable than others. Dispersion complements sharpness: two models with the same average sharpness may differ substantially in dispersion.

Higher dispersion is favorable only when true underlying errors are heteroscedastic (i.e., when prediction difficulty genuinely varies across samples). If errors are homoscedastic, a model with high dispersion wastes capacity on spurious confidence variation. We report dispersion alongside sharpness and other metrics for completeness, but treat it as a secondary diagnostic tool (not a primary optimization target) that is only meaningful for datasets with heteroscedastic errors.

Coverage. For a nominal $(1-\alpha)$ prediction interval $[l, u]$ extracted from the predictive CDF, empirical coverage is the fraction of test points falling inside the interval:

$$\mathrm{Cov}_{1-\alpha} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{l_n \le y_n \le u_n}.$$

Here, for the n-th test sample, $[l_n, u_n]$ denotes the nominal $(1-\alpha)$ predictive interval (lower and upper bounds) extracted from the model's predictive CDF for that sample, and y_n is the observed realization of the target. We report coverage at the 90% level (Cov_{90}, target 0.90) and the 95% level (Cov_{95}, target 0.95). Concretely, for Cov_{90} we set $\alpha = 0.10$ (nominal level $1-\alpha = 0.90$) and for Cov_{95} we set $\alpha = 0.05$ (nominal level $1-\alpha = 0.95$); α denotes the total tail probability excluded from a central $(1-\alpha)$ interval $[l_n, u_n]$. A well-calibrated model achieves coverage close to the nominal level; the interval score (IS) jointly penalizes over-wide intervals and coverage shortfalls.

PIT Kolmogorov–Smirnov test. The probability integral transform (PIT) assesses the calibration of the full predictive distribution rather than just an interval or a single moment (Dawid, 1984; Diebold et al., 1998). For test sample n with predictive CDF $\widehat{F}_n$ and observed outcome y_n, the PIT value is

$$p_n = \widehat{F}_n(y_n).$$

If the predictive distributions are ideal (i.e., y_n is truly drawn from $\widehat{F}_n$ for every n), the PIT values $\{p_n\}_{n=1}^{N}$ are i.i.d. $\mathrm{Uniform}(0,1)$, regardless of the shape of $\widehat{F}_n$ itself (Dawid, 1984). We test this null hypothesis with a one-sample Kolmogorov–Smirnov test against $\mathrm{Uniform}(0,1)$, reporting the test statistic

$$\text{PIT-KS} = \sup_{p\in[0,1]} \left|\widehat{G}_N(p) - p\right|,$$

where $\widehat{G}_N$ is the empirical CDF of $\{p_n\}_{n=1}^{N}$, together with the corresponding p-value $\text{PIT-KS}_{\text{pval}}$. A PIT-KS statistic near zero (equivalently, a large p-value) indicates that the predicted distributions are statistically indistinguishable from calibrated; a small p-value indicates systematic miscalibration (e.g., predictive distributions that are too narrow, too wide, or biased). Following the histogram-based prediction representation, PIT values falling within a bin are obtained by linear interpolation of the piecewise-constant predictive density, and PIT values landing exactly on a probability atom (zero-width bin) are assigned the mid-CDF value 0.5, the standard non-randomized convention for discrete or tied predictive mass (Czado et al., 2009). PIT-KS is not a proper scoring rule: it is a pure calibration diagnostic and, unlike sharpness or dispersion, is agnostic to how concentrated the predictive distribution is; a model can be perfectly calibrated (PIT-KS ≈ 0) while still being uninformative (low sharpness). We therefore report PIT-KS jointly with sharpness, dispersion, and coverage to separate calibration from concentration.

B Benchmark Datasets

We draw candidate datasets from four complementary public repositories (Section B.1), apply a multi-level deduplication procedure to remove redundant entries (Section B.2.1), and filter the remaining candidates to retain only well-formed regression tasks (Section B.2.2). All datasets are available through public APIs (OpenML, PMLB GitHub, KEEL) or direct download URLs without requiring authentication, ensuring full reproducibility. Our guiding principles follow recent calls for higher-quality tabular benchmarks (Erickson et al., 2025; Grinsztajn et al., 2022): we prioritize diversity across domains and feature dimensionalities, favour datasets derived from genuine data-collection processes, and deduplicate aggressively to avoid inflating the effective benchmark size with near-identical tasks.

B.1 Data Sources

Candidate datasets are drawn from four complementary repositories:

1. **OpenML benchmark suites.** We include three curated regression suites hosted on OpenML (Vanschoren et al., 2014):
   - Suite 297 (OpenML-CTR23): a community-curated tabular regression benchmark (Fischer et al., 2023);
   - Suite 299: an additional OpenML regression collection;
   - Suite 269: an OpenML regression collection collating further community-vetted datasets.

   These suites are fetched programmatically via the OpenML Python API and serve as the base layer of our collection.
2. **PMLB (Penn Machine Learning Benchmarks).** Regression datasets from PMLB (Olson et al., 2017), distributed as compressed TSV files on GitHub. PMLB provides a standardised, version-controlled archive of datasets widely used in AutoML research.
3. **KEEL (Knowledge Extraction based on Evolutionary Learning)** Regression datasets from the KEEL data-mining repository (Derrac et al., 2015), distributed as zipped `.dat` files. KEEL datasets add coverage of domains less represented in OpenML and PMLB, such as financial time series and energy consumption.
4. **TALENT / OpenML verified regression datasets.** Additional OpenML datasets drawn from the TALENT benchmark (Liu et al., 2025) and independently verified as regression tasks. This set expands domain coverage to areas including real estate, book reviews, sensor fusion, and sports analytics.

B.2 Dataset construction and filtering

B.2.1 Deduplication Procedure

A central challenge in multi-source dataset aggregation is that the same underlying data frequently appears under different names, numeric identifiers, or preprocessing variants. To avoid having highly correlated (identical) datasets, we apply a four-level deduplication strategy:

1. **Exact normalised match.** Dataset names are normalised by lowercasing, stripping leading numeric prefixes (e.g., `197_cpu_act` $\rightarrow$ `cpuact`), removing separators and special characters, and collapsing repeated characters. Datasets whose normalised names are identical are treated as duplicates.
2. **Substring match.** For names longer than three characters, we check whether either normalised name is a substring of the other. This catches common patterns such as `housing` matching `californiahousing`.
3. **Fuzzy match.** We compute a pairwise similarity ratio (via Python's `SequenceMatcher`) and flag pairs with $\geq 85\,\%$ similarity as duplicates.
4. **Explicit deduplication keys.** Many datasets are annotated with known aliases (e.g., `cpu_act` $\leftrightarrow$ `cpuact`). All checks above are re-applied against these keys.

When a duplicate is detected, we only retain one version and discard the others, using the (arbitrary) precedence order (OpenML $>$ PMLB $>$ KEEL $>$ scikit-learn).

B.2.2 Validation Filters

We exclude Santander_transaction_value (OpenML ID 42572) because of excessive runtimes for some models, including RealMLP with hyperparameter optimization. After deduplication and this exclusion, the remaining candidate datasets are downloaded and subjected to the following checks:

- **Low-cardinality target removal.** For successfully loaded targets, datasets with fewer than four distinct non-missing values are excluded. Thus, constant, binary, and three-valued

targets are all removed. This is a target-cardinality screening heuristic, not a definitive classification-versus-regression test. In the reported benchmark, this excludes the WLAN 3D-estimation dataset (OpenML ID 45720) and `chscase_foot` (OpenML ID 703), each with three distinct target values. Starting from 104 deduplicated candidates, excluding Santander and these two datasets leaves 101 datasets.

- **Target integrity.** We favour datasets derived from genuine data-collection processes. Purely synthetic or deterministic-function datasets are included only when they appear in established regression benchmark suites and provide useful diversity in feature–target relationships. We exclude datasets which have non-tabular modalities (images, raw text, audio) and only accept datasets with pure tabular representation. Rows with missing or non-numeric target values are removed. If the target column is stored as a string or categorical type, we convert it to numeric and drop any rows where conversion fails.

B.3 PREPROCESSING

All datasets undergo lightweight, uniform preprocessing to ensure compatibility with all benchmark models:

- Missing numeric features are imputed with the column median.
- Missing categorical features are imputed with the most frequent category.
- Categorical columns are label-encoded into integer values.
- No feature scaling or normalisation is applied at the benchmark level; individual model wrappers are free to apply their own transformations internally.

We note that preprocessing choices—imputation strategies, categorical encoding, and any scaling—act as modelling hyperparameters that can materially affect downstream performance. Different model families benefit from different strategies, and several modern algorithms support native handling of missing values. In a future release we will provide an alternative benchmark configuration that omits benchmark-level imputation, enabling direct comparison of methods that handle missing values natively.

B.4 FINAL DATASET OVERVIEW

Table 1 enumerates the datasets in ScoringBench after deduplication and validation.

Table 1: The 101 datasets included in the reported ScoringBench results.

Dataset	**Dataset**
1000-Cameras-Dataset	1027_ESL
1028_SWD	1029_LEV
1030_ERA	1193_BNG_lowbwt
1199_BNG_echoMonths	1201_BNG_breastTumor
1203_BNG_pwLinear	215_2dplanes
218_house_8L	225_puma8NH
227_cpu_small	294_satellite_image
344_mv	503_wind
529_pollen	547_no2
564_fried	595_fri_c0_1000_10
AIRFOIL	Ailerons
Airlines_DepDelay_10M	Allstate_Claims_Severity
Another-Dataset-on-used-Fiat-500-(1538-rows)	BIAS_CORRECTION
BNG(mv)	BNG(stock)
Bike_Sharing_Demand	Brazilian_houses
Buzzinsocialmedia_Twitter	CPS1988
CookbookReviews	Ele2
Energy_Efficiency	Goodreads-Computer-Books
IEEE80211aa-GATS	Infrared_Thermography_Temperature

continued on next page

Table 1 – *continued from previous page*

Dataset	Dataset
Job_Profitability	Laser
MIP-2016-regression	Mercedes_Benz_Greener_Manufacturing
MiamiHousing2016	Moneyball
Mortgage	NASA_PHM2008
OnlineNewsPopularity	PRODUCTIVITY
Parkinsons_Telemonitoring	QSAR-TID-10980
QSAR-TID-11	QsarFishToxicity
SAT11-HAND-runtime-regression	SGEMM_GPU_kernel_performance
Student_Performance	Treasury
Wizmir	Yolanda
abalone	analcatdata_supreme
avocado_sales	bank32nh
bank8FM	black_friday
boston	california
colleges	concrete_compressive_strength
cpu_act	dataset_sales
debutanizer	diamonds
elevators	fifa
house_16H	house_prices_nominal
house_sales	houses
isolet	kin8nm
mauna-loa-atmospheric-co2	medical_charges
nyc-taxi-green-dec-2016	particulate-matter-ukair-2017
pol	puma32H
quake	sensory
socmob	space_ga
stock_fardamento02	sulfur
superconduct	tecator
topo_2_1	us_crime
visualizing_soil	weather_izmir
wine_quality	year
yprop_4_1	

The final number of datasets used in any given experiment depends on which candidates survive all deduplication and validation steps. The authoritative, machine-readable dataset list is exported as `datasets.json` at the start of every benchmark run. If any datasets fails on too many models (e.g., due to memory errors), it is automatically removed from the final results to prevent skewing the analysis.

C Experimental protocol

C.1 Cross-Validation Protocol

Each dataset is evaluated using K-fold cross-validation (CV) with $K = 5$ folds and a fixed random seed (`seed=42`) for reproducibility. The number of CV repeats defaults to 1 but can be increased with `--n_repeats_cv` to obtain tighter confidence estimates.

C.2 Downsampling after CV split

To ensure computational feasibility and to prevent very large datasets from dominating the benchmark, each cross-validation repeat draws a random subsample of at most $N_{\max} = 3000 = 2400 + 600$ (train/test) observations (configurable via `--sample_size`). This design may slightly favor in-context learners, as more traditional parametric baselines (e.g., XGBoost, MLPs) cannot fully exploit their scaling behavior on larger datasets.

Subsampling is performed inside the CV loop so that each repeat sees a different random subset, improving robustness through less correlation in the folds. Future versions of the benchmark may study sample efficiency by measuring how model performance scales with training-set size; this

requires evaluating multiple training-sample sizes per dataset. As an ablation, we also evaluate the effect of capping the total dataset size (training plus test) at 1000, 2000 and 8000 observations, see Appendix E.

C.3 Evaluation

For each CV split and each model, we compute the fold-averaged score $\bar{s}_{m,d,p}$ on held out data (from the CV split) for each metric p on dataset d.

C.4 Experimental runtime

All experiments were conducted on a cluster of 40 GPU nodes equipped with NVIDIA RTX 6000 GPUs, with a total runtime of approximately 36 hours. A significant amount of the computational cost is attributed to ***RealMLP***, which is considerably slower than other methods due to its extensive hyperparameter optimization. Further details on training and inference times can be found in the online leaderboard.

D Ranking Approaches

D.1 Demšar–autorank approach (Autorank)

Autorank strictly follows the Demšar protocol (Demšar, 2006) and relies on the implementation in the `autorank` library (Herbold, 2020). This approach discards score magnitudes and operates solely on ordinal ranks. Let $m \in \{1, \dots, M\}$ denote a model, $d \in \{1, \dots, D\}$ a dataset, and p a specific performance metric. The procedure is as follows (Demšar, 2006):

- For each dataset d and metric p, the models m are ranked based on their fold-averaged scores $\bar{s}_{m,d,p}$. The best-performing model receives rank 1, the second-best rank 2, and so on. This transforms the data into the shape $(\text{model } m,\ \text{dataset } d,\ \text{rank } r_{m,d,p})$.[4]
- Next, the mean rank $\bar{r}_{m,p}$ for each model m on metric p is computed by averaging across all D datasets, yielding the mean rank

$$\bar{r}_{m,p} = \frac{1}{D} \sum_{d=1}^{D} r_{m,d,p} \tag{1}$$

- The non-parametric Friedman test then evaluates whether these mean ranks $\bar{r}_{m,p}$ differ significantly across the M models for a given metric p.
- If the omnibus Friedman test is significant at level $\alpha = 0.05$, the post-hoc Nemenyi test (Nemenyi, 1963) identifies which specific pairs of models differ. The Nemenyi test defines a critical difference (CD) threshold (Demšar, 2006):

$$\mathrm{CD} = q_\alpha \sqrt{\frac{M(M+1)}{6D}}\,, \tag{2}$$

  where q_α is the critical value from the studentised range distribution divided by $\sqrt{2}$.
- Two models are considered statistically significantly different if the absolute difference between their mean ranks exceeds the value CD computed in Eq. (2).

Autorank automatically selects appropriate summary statistics, confidence-interval methods, and effect-size measures for reporting. Furthermore, it automatically creates critical difference (CD) diagrams like the one in Figure 3.

To partially mitigate the loss of magnitude information inherent in rank-based tests, `autorank` augments each model's evaluation with Akinshin's γ, a non-parametric effect size analogous to Cohen's d. Akinshin's γ quantifies the standardized difference between two score distributions without parametric assumptions; we report it computed against the top-ranked model.

[4] In contrast to the scoring rules, minimizing coverage is not desirable but we need to judge goodness by its agreement with the nominal level. For ranking we, therefore, have to transform coverage and we rank according to $|\mathrm{Cov}_{1-\alpha} - (1-\alpha)|$ instead of $\mathrm{Cov}_{1-\alpha}$.

D.2 STANDARDISED MAGNITUDE-BASED RANKING (z-SCORE RANKING)

The second approach retains performance magnitude information by standardising the fold-averaged scores $\bar{s}_{m,d,p}$ (see Eq. (3)) within each dataset before aggregating them. Starting with the data in the shape $(\text{model } m,\ \text{dataset } d,\ \text{mean score } \bar{s}_{m,d,p})$, we compute a z-score to normalize for dataset difficulty:

$$z_{m,d,p} \;=\; \frac{\bar{s}_{m,d,p} - \mu_{d,p}}{\sigma_{d,p} + \epsilon}\,, \tag{3}$$

where the dataset- and metric-specific mean $\mu_{d,p}$ and standard deviation $\sigma_{d,p}$ are defined across the M models as:

$$\mu_{d,p} = \frac{1}{M}\sum_{m=1}^{M} \bar{s}_{m,d,p}\,, \qquad \sigma_{d,p} = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(\bar{s}_{m,d,p} - \mu_{d,p}\right)^2}\,.$$

Here, $\epsilon = 10^{-9}$ prevents division by zero, and $\sigma_{d,p}$ is estimated with $M-1$ degrees of freedom. This standardisation transforms the data into the shape:

$$(\text{model } m,\ \text{dataset } d,\ z\text{-score } z_{m,d,p})\,,$$

expressing each model's performance in units of the cross-model standard deviation on dataset d. Crucially, this standardisation centers and scales scores within each dataset (subtracting the dataset mean and dividing by the cross-model standard deviation), removing dataset-specific location and scale so that resulting $z_{m,d,p}$ values reflect relative performance across models independent of metric scale or dataset difficulty.

For metrics p where lower values indicate better performance, the sign of $z_{m,d,p}$ is negated so that higher standardised scores uniformly represent superior performance (e.g. for CRPS, lower is better, so we use $-z_{m,d,p}$ instead of $z_{m,d,p}$). The final aggregated ranking score for model m on metric p is obtained by averaging its standardised z-scores across all D datasets:

$$\bar{z}_{m,p} \;=\; \frac{1}{D}\sum_{d=1}^{D} z_{m,d,p}\,.$$

Models are then ranked in descending order based on $\bar{z}_{m,p}$.

We compare both ranking approaches in Figure 4. The figure shows, for every model, how many times it was placed Top1 and how many times it was placed in the Top3 under each approach. Similar Top1 and Top3 counts across the two methods indicate stability — the same models are repeatedly identified as winners and leaders; large differences imply the ranking choice materially changes which models are considered top performers. Additionally, a Pearson correlation between the two rank vectors is computed to quantify their agreement, see Appendix D.3.

D.3 RANKING STABILITY BETWEEN BOTH RANKING APPROACHES

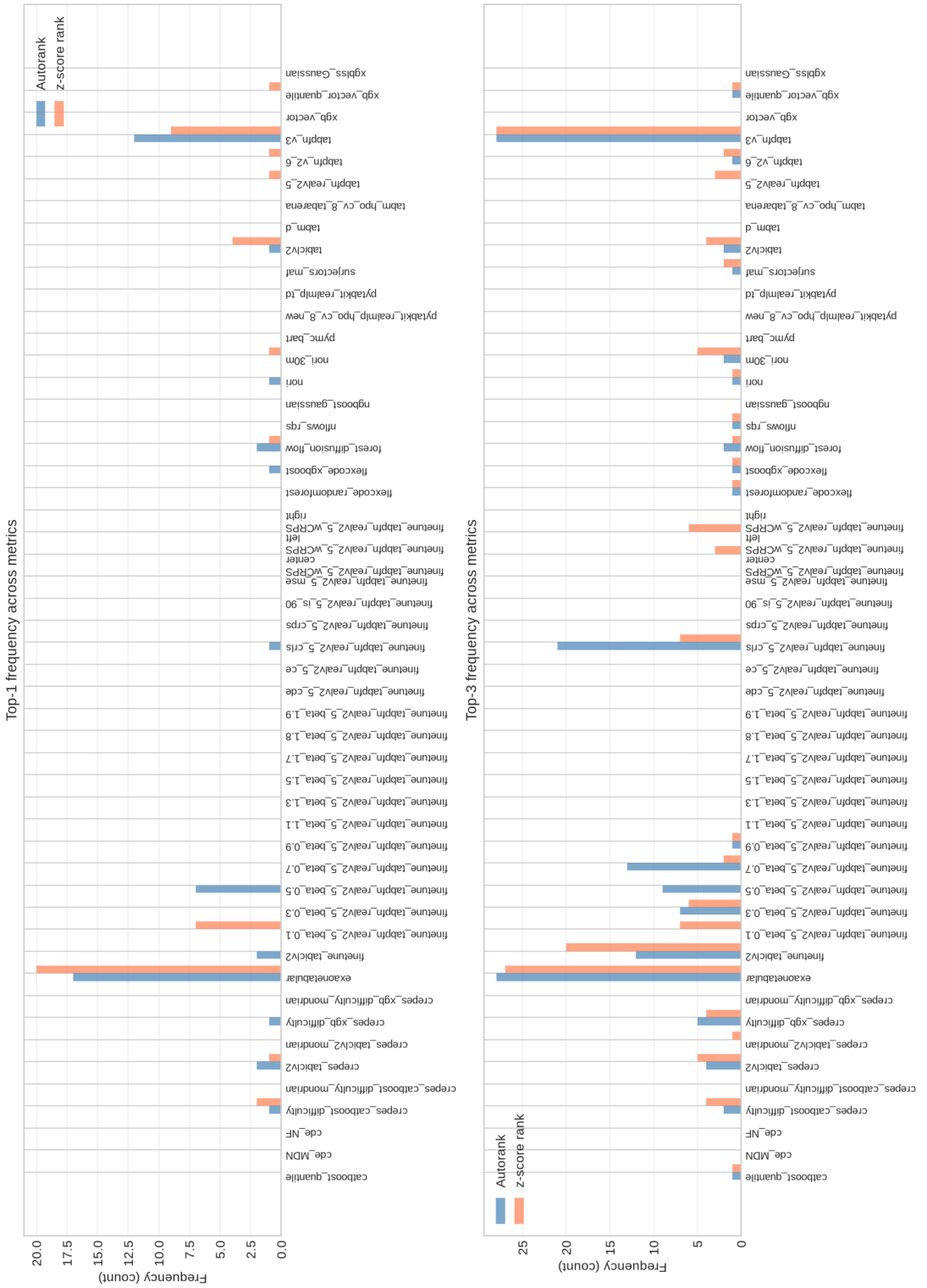


Figure 4: Winner stability between ranking methods. Evaluation of Top-1 and Top-3 agreement as described in the text.

To assess whether our conclusions depend on the ranking method, we compare Autorank and z-score rankings across all metrics. For each metric, we record which model is ranked first and which models appear in the top three under each method (Figure 4).

The two approaches largely agree: across the 44 metrics reported in Table 2, the Pearson correlation between their rank vectors has a mean of 0.9667 and a median of 0.9762, indicating broad agreement between their overall model orderings.

Given this overall agreement, reporting only the Autorank-based leaderboards in the main text is justified; both rankings for every metric are available on the live leaderboard at `https://incredible-truffle-9624b2.netlify.app/`.

Table 2: Pearson correlation of ranks between Autorank and Zscore ranking across metrics (computed over 51 models).

Metric	Rank Correlation (r)
cde_loss	0.9743
coverage_20	0.9414
coverage_40	0.8845
coverage_60	0.9186
coverage_80	0.9610
coverage_90	0.9844
coverage_95	0.8955
crps	0.9910
dispersion	0.9680
dpd_beta_0.01	0.9880
dpd_beta_0.2	0.9720
dpd_beta_0.5	0.9716
dpd_beta_1.0	0.9743
energy_score_beta_0.1	0.9820
energy_score_beta_0.3	0.9871
energy_score_beta_0.5	0.9924
energy_score_beta_0.7	0.9925
energy_score_beta_0.9	0.9920
energy_score_beta_1.0	0.9910
energy_score_beta_1.1	0.9902
energy_score_beta_1.3	0.9890
energy_score_beta_1.5	0.9664
energy_score_beta_1.7	0.9507
energy_score_beta_1.8	0.9500
energy_score_beta_1.9	0.9460
interval_score_20	0.9878
interval_score_40	0.9879
interval_score_60	0.9879
interval_score_80	0.9894
interval_score_90	0.9825
interval_score_95	0.9768
mae	0.9716
pit_ks_pvalue	0.9298
pit_ks_stat	0.8119
pseudospherical_alpha_1.5	0.9757
pseudospherical_alpha_2.0	0.9773
pseudospherical_alpha_3.0	0.9836
r2	0.9593
rmse	0.9539
sharpness	0.9636
train_time	0.9720
wcrps_center	0.9919
wcrps_left	0.9892
wcrps_right	0.9906

D.4 Additional ranking results

In the main text, the CRPS-based leaderboard was reported in Table 3. Here we report the Autorank leaderboards for RMSE, the 90% interval score, and absolute deviation from nominal 90% coverage (Tables 4 to 6). RMSE provides the point-estimate comparison to CRPS; the interval-score and coverage results illustrate the distinction between distributional quality and nominal coverage. The z-score-based leaderboards are available at `https://incredible-truffle-9624b2.netlify.app/`. These tables match the structure of Table 3.

	MR	MED	MAD	CI	γ	Magnitude
exaonetabular	8.723	0.378	0.362	[0.073, 1.597]	0.000	negligible
tabpfn_v3	8.733	0.373	0.356	[0.092, 1.608]	0.010	negligible
finetune_tabpfn_realv2_5_crls	11.084	0.377	0.363	[0.091, 1.652]	0.001	negligible
finetune_tabiclv2	11.871	0.392	0.377	[0.092, 1.600]	-0.025	negligible
finetune_tabpfn_realv2_5_beta_0.7	12.386	0.379	0.364	[0.092, 1.651]	-0.002	negligible
finetune_tabpfn_realv2_5_beta_0.5	12.569	0.379	0.365	[0.092, 1.651]	-0.003	negligible
finetune_tabpfn_realv2_5_crps	12.698	0.381	0.367	[0.092, 1.649]	-0.006	negligible
finetune_tabpfn_realv2_5_beta_1.1	12.871	0.381	0.367	[0.092, 1.651]	-0.007	negligible
finetune_tabpfn_realv2_5_beta_0.9	12.911	0.381	0.366	[0.092, 1.650]	-0.006	negligible
finetune_tabpfn_realv2_5_beta_0.3	13.663	0.379	0.365	[0.092, 1.651]	-0.003	negligible
finetune_tabpfn_realv2_5_beta_1.3	14.015	0.382	0.367	[0.092, 1.656]	-0.007	negligible
tabiclv2	14.119	0.392	0.378	[0.096, 1.602]	-0.026	negligible
nori_30m	14.337	0.393	0.377	[0.074, 1.621]	-0.028	negligible
finetune_tabpfn_realv2_5_mse	14.817	0.378	0.363	[0.092, 1.653]	-0.000	negligible
finetune_tabpfn_realv2_5_beta_1.5	14.980	0.382	0.368	[0.092, 1.656]	-0.008	negligible
finetune_tabpfn_realv2_5_beta_0.1	15.030	0.379	0.365	[0.092, 1.652]	-0.003	negligible
finetune_tabpfn_realv2_5_wCRPS_center	15.069	0.380	0.366	[0.092, 1.654]	-0.005	negligible
finetune_tabpfn_realv2_5_wCRPS_right	15.851	0.384	0.369	[0.092, 1.647]	-0.012	negligible
finetune_tabpfn_realv2_5_beta_1.7	16.139	0.382	0.367	[0.092, 1.655]	-0.008	negligible
finetune_tabpfn_realv2_5_beta_1.8	16.416	0.382	0.368	[0.092, 1.656]	-0.008	negligible
finetune_tabpfn_realv2_5_wCRPS_left	16.510	0.387	0.373	[0.092, 1.653]	-0.017	negligible
finetune_tabpfn_realv2_5_beta_1.9	16.797	0.383	0.369	[0.092, 1.655]	-0.010	negligible
finetune_tabpfn_realv2_5_cde	17.589	0.381	0.366	[0.092, 1.653]	-0.005	negligible
tabpfn_realv2_5	18.208	0.378	0.363	[0.080, 1.655]	-0.001	negligible
finetune_tabpfn_realv2_5_ce	18.441	0.381	0.366	[0.092, 1.655]	-0.005	negligible
finetune_tabpfn_realv2_5_is_90	19.728	0.381	0.366	[0.092, 1.658]	-0.006	negligible
nori	20.941	0.396	0.379	[0.074, 1.639]	-0.034	negligible
tabpfn_v2_6	22.911	0.394	0.378	[0.105, 1.652]	-0.030	negligible
pytabkit_realmlp_hpo_cv_8_new	25.693	0.401	0.386	[0.096, 1.658]	-0.042	negligible
tabm_hpo_cv_8_tabarena	28.000	0.426	0.397	[0.097, 1.679]	-0.086	negligible
crepes_tabiclv2	31.960	0.408	0.391	[0.099, 1.730]	-0.054	negligible
crepes_tabiclv2_mondrian	33.723	0.429	0.412	[0.100, 1.752]	-0.090	negligible
pytabkit_realmlp_td	34.485	0.451	0.423	[0.108, 1.787]	-0.125	negligible
tabm_d	34.851	0.442	0.427	[0.122, 2.076]	-0.110	negligible
catboost_quantile	36.594	0.482	0.462	[0.106, 2.381]	-0.169	negligible
xgb_vector	39.030	0.475	0.449	[0.131, 2.385]	-0.160	negligible
flexcode_randomforest	39.614	0.482	0.461	[0.120, 2.354]	-0.169	negligible
ngboost_gaussian	39.782	0.473	0.451	[0.128, 3.046]	-0.157	negligible
crepes_catboost_difficulty	40.851	0.472	0.450	[0.126, 2.591]	-0.155	negligible
cde_MDN	41.317	0.560	0.524	[0.125, 2.353]	-0.273	small
forest_diffusion_flow	41.436	0.475	0.449	[0.119, 2.429]	-0.161	negligible
xgb_vector_quantile	41.980	0.514	0.491	[0.119, 2.606]	-0.213	small
xgblss_Gaussian	42.119	0.540	0.510	[0.131, 3.165]	-0.247	small
flexcode_xgboost	42.327	0.563	0.527	[0.141, 2.578]	-0.276	small
cde_NF	42.851	0.606	0.556	[0.138, 2.308]	-0.328	small
crepes_xgb_difficulty	42.901	0.467	0.446	[0.118, 2.683]	-0.148	negligible
crepes_catboost_difficulty_mondrian	43.683	0.496	0.477	[0.136, 2.768]	-0.188	negligible
nflows_rqs	44.941	0.548	0.513	[0.155, 2.411]	-0.259	small
crepes_xgb_difficulty_mondrian	45.584	0.506	0.484	[0.122, 2.810]	-0.202	small
surjectors_maf	46.079	0.564	0.528	[0.152, 2.577]	-0.278	small
pymc_bart	46.792	0.894	0.837	[0.190, 4.376]	-0.540	medium

Table 3: Autorank leaderboard for CRPS. Columns: MR = mean rank across datasets (lower is better); MED = median score; MAD = median absolute deviation; CI = confidence interval for the median; γ = Akinshin's gamma effect size relative to the top-ranked model. *Note:* MED, MAD, and γ are computed by autorank without further standardization, thus reflecting the original score scale. While this may hamper direct comparison across datasets with different units, we report it for transparency; ranking is unaffected as autorank sorts by mean rank (scale-independent). The z-score ranking approach properly standardizes for dataset difficulty and produces similar ordering as the autorank approach due to high correlation between both methods (Appendix D.3) while having interpretable z-scores; Live rankings available at `https://incredible-truffle-9624b2.netlify.app/`.

	MR	MED	MAD	CI	γ	Magnitude
exaonetabular	8.604	2.963	2.816	[0.663, 12.150]	0.000	negligible
tabpfn_v3	10.327	3.007	2.811	[0.690, 12.049]	-0.011	negligible
finetune_tabpfn_realv2_5_crls	10.678	2.968	2.833	[0.688, 12.285]	-0.001	negligible
finetune_tabiclv2	11.020	3.006	2.879	[0.689, 12.238]	-0.010	negligible
tabiclv2	12.584	2.997	2.852	[0.685, 12.168]	-0.008	negligible
finetune_tabpfn_realv2_5_beta_0.7	13.485	3.001	2.824	[0.692, 12.252]	-0.009	negligible
nori_30m	13.535	2.988	2.832	[0.601, 12.407]	-0.006	negligible
finetune_tabpfn_realv2_5_beta_0.5	13.569	3.001	2.839	[0.691, 12.258]	-0.009	negligible
finetune_tabpfn_realv2_5_beta_0.3	13.614	3.002	2.840	[0.691, 12.259]	-0.009	negligible
finetune_tabpfn_realv2_5_crps	13.688	3.002	2.837	[0.692, 12.261]	-0.009	negligible
finetune_tabpfn_realv2_5_beta_0.9	14.198	2.998	2.833	[0.692, 12.255]	-0.008	negligible
finetune_tabpfn_realv2_5_beta_1.1	14.277	3.004	2.838	[0.692, 12.259]	-0.010	negligible
finetune_tabpfn_realv2_5_wCRPS_right	14.465	3.028	2.864	[0.692, 12.291]	-0.016	negligible
finetune_tabpfn_realv2_5_beta_1.3	14.470	3.017	2.851	[0.692, 12.269]	-0.013	negligible
finetune_tabpfn_realv2_5_beta_1.5	14.505	3.029	2.864	[0.691, 12.276]	-0.016	negligible
finetune_tabpfn_realv2_5_beta_0.1	14.851	3.012	2.850	[0.691, 12.270]	-0.012	negligible
finetune_tabpfn_realv2_5_is_90	15.530	3.020	2.858	[0.691, 12.289]	-0.014	negligible
finetune_tabpfn_realv2_5_mse	15.876	3.063	2.876	[0.692, 12.330]	-0.024	negligible
finetune_tabpfn_realv2_5_beta_1.7	16.198	3.067	2.903	[0.691, 12.285]	-0.025	negligible
finetune_tabpfn_realv2_5_wCRPS_center	16.228	3.043	2.874	[0.691, 12.277]	-0.019	negligible
finetune_tabpfn_realv2_5_wCRPS_left	16.431	3.057	2.890	[0.690, 12.335]	-0.022	negligible
finetune_tabpfn_realv2_5_beta_1.8	16.861	3.057	2.894	[0.690, 12.293]	-0.022	negligible
finetune_tabpfn_realv2_5_beta_1.9	17.629	3.066	2.904	[0.689, 12.293]	-0.024	negligible
tabpfn_realv2_5	17.802	3.059	2.877	[0.695, 12.309]	-0.023	negligible
finetune_tabpfn_realv2_5_cde	18.114	3.069	2.907	[0.696, 12.247]	-0.025	negligible
finetune_tabpfn_realv2_5_ce	18.391	3.071	2.909	[0.696, 12.330]	-0.025	negligible
nori	19.228	3.011	2.848	[0.693, 12.571]	-0.011	negligible
tabpfn_v2_6	22.752	3.067	2.845	[0.885, 13.411]	-0.025	negligible
pytabkit_realmlp_hpo_cv_8_new	26.802	3.124	2.987	[0.741, 15.931]	-0.038	negligible
tabm_hpo_cv_8_tabarena	27.980	3.528	3.263	[0.781, 18.557]	-0.125	negligible
pytabkit_realmlp_td	33.069	3.678	3.499	[0.944, 17.701]	-0.152	negligible
crepes_tabiclv2	33.743	3.764	3.522	[0.797, 19.249]	-0.169	negligible
tabm_d	34.119	3.609	3.411	[1.012, 20.939]	-0.139	negligible
crepes_tabiclv2_mondrian	35.772	3.876	3.695	[0.866, 20.670]	-0.188	negligible
ngboost_gaussian	36.723	3.816	3.629	[1.161, 24.067]	-0.177	negligible
xgb_vector	36.851	4.619	4.334	[1.155, 20.140]	-0.306	small
cde_MDN	37.693	4.640	4.357	[1.058, 23.694]	-0.308	small
crepes_catboost_difficulty	39.020	4.045	3.834	[1.111, 24.556]	-0.217	small
cde_NF	40.822	6.031	5.714	[1.471, 26.977]	-0.459	small
catboost_quantile	40.901	4.686	4.474	[1.087, 24.938]	-0.311	small
xgblss_Gaussian	40.950	4.388	4.205	[1.196, 25.127]	-0.269	small
crepes_xgb_difficulty	41.158	4.128	3.925	[1.076, 25.667]	-0.230	small
crepes_catboost_difficulty_mondrian	42.861	4.936	4.631	[1.199, 39.635]	-0.347	small
xgb_vector_quantile	43.347	5.580	5.344	[1.310, 30.291]	-0.413	small
flexcode_randomforest	43.604	5.263	5.030	[1.638, 28.275]	-0.381	small
pymc_bart	44.030	7.531	7.125	[1.474, 42.480]	-0.569	medium
nflows_rqs	44.327	6.592	6.125	[1.538, 29.673]	-0.514	medium
crepes_xgb_difficulty_mondrian	44.634	5.257	4.913	[1.288, 34.131]	-0.386	small
flexcode_xgboost	45.604	6.912	6.715	[2.081, 38.746]	-0.517	medium
forest_diffusion_flow	45.752	5.723	5.421	[1.393, 31.649]	-0.431	small
surjectors_maf	47.327	7.705	7.323	[1.677, 35.535]	-0.577	medium

Table 4: Autorank leaderboard for the 90% interval score (lower is better). Column definitions as in Table 3.

	MR	MED	MAD	CI	γ	Magnitude
crepes_catboost_difficulty	13.376	0.007	0.004	[0.003, 0.012]	0.000	negligible
crepes_xgb_difficulty	14.228	0.007	0.005	[0.003, 0.013]	-0.096	negligible
finetune_tabiclv2	14.337	0.007	0.005	[0.004, 0.015]	-0.096	negligible
crepes_tabiclv2	14.931	0.007	0.004	[0.004, 0.012]	-0.109	negligible
tabiclv2	15.317	0.010	0.007	[0.004, 0.020]	-0.336	small
finetune_tabpfn_realv2_5_beta_0.3	16.545	0.009	0.007	[0.003, 0.020]	-0.246	small
finetune_tabpfn_realv2_5_beta_0.1	16.812	0.010	0.008	[0.004, 0.021]	-0.384	small
finetune_tabpfn_realv2_5_beta_0.5	16.975	0.008	0.006	[0.003, 0.021]	-0.214	small
finetune_tabpfn_realv2_5_beta_0.9	17.525	0.009	0.007	[0.004, 0.022]	-0.270	small
nori_30m	17.634	0.011	0.009	[0.004, 0.022]	-0.394	small
finetune_tabpfn_realv2_5_beta_0.7	17.698	0.010	0.007	[0.004, 0.021]	-0.386	small
finetune_tabpfn_realv2_5_cde	17.703	0.010	0.008	[0.004, 0.023]	-0.325	small
nori	17.896	0.010	0.007	[0.005, 0.021]	-0.379	small
finetune_tabpfn_realv2_5_crps	17.911	0.010	0.007	[0.004, 0.020]	-0.415	small
finetune_tabpfn_realv2_5_beta_1.1	18.599	0.011	0.007	[0.005, 0.020]	-0.502	medium
finetune_tabpfn_realv2_5_beta_1.3	19.040	0.011	0.006	[0.005, 0.019]	-0.539	medium
finetune_tabpfn_realv2_5_wCRPS_right	19.119	0.011	0.007	[0.005, 0.021]	-0.463	small
finetune_tabpfn_realv2_5_beta_1.5	19.436	0.012	0.007	[0.006, 0.019]	-0.602	medium
exaonetabular	19.624	0.010	0.006	[0.006, 0.025]	-0.401	small
finetune_tabpfn_realv2_5_wCRPS_left	20.713	0.012	0.007	[0.007, 0.020]	-0.678	medium
finetune_tabpfn_realv2_5_beta_1.7	20.738	0.013	0.008	[0.006, 0.021]	-0.664	medium
finetune_tabpfn_realv2_5_wCRPS_center	20.772	0.012	0.007	[0.006, 0.020]	-0.614	medium
finetune_tabpfn_realv2_5_ce	20.837	0.013	0.009	[0.005, 0.023]	-0.556	medium
finetune_tabpfn_realv2_5_is_90	21.401	0.011	0.007	[0.005, 0.025]	-0.480	small
finetune_tabpfn_realv2_5_beta_1.9	21.465	0.014	0.009	[0.008, 0.023]	-0.681	medium
finetune_tabpfn_realv2_5_beta_1.8	21.649	0.013	0.008	[0.006, 0.021]	-0.723	medium
finetune_tabpfn_realv2_5_crls	22.005	0.013	0.009	[0.006, 0.025]	-0.614	medium
finetune_tabpfn_realv2_5_mse	23.104	0.014	0.009	[0.008, 0.023]	-0.722	medium
tabpfn_realv2_5	23.455	0.015	0.010	[0.006, 0.029]	-0.681	medium
tabpfn_v3	23.950	0.014	0.010	[0.008, 0.026]	-0.695	medium
tabm_hpo_cv_8_tabarena	25.663	0.024	0.017	[0.009, 0.044]	-0.935	large
pytabkit_realmlp_td	26.391	0.022	0.017	[0.009, 0.046]	-0.831	large
tabpfn_v2_6	26.960	0.017	0.011	[0.009, 0.040]	-0.786	medium
pymc_bart	29.158	0.036	0.021	[0.015, 0.065]	-1.291	large
ngboost_gaussian	29.520	0.031	0.016	[0.020, 0.055]	-1.419	large
pytabkit_realmlp_hpo_cv_8_new	30.381	0.027	0.015	[0.014, 0.053]	-1.197	large
cde_MDN	30.901	0.043	0.027	[0.018, 0.071]	-1.272	large
xgb_vector	32.010	0.051	0.030	[0.020, 0.076]	-1.400	large
tabm_d	32.094	0.042	0.019	[0.025, 0.060]	-1.758	large
crepes_tabiclv2_mondrian	34.896	0.038	0.008	[0.032, 0.065]	-3.182	large
cde_NF	37.089	0.102	0.087	[0.031, 0.291]	-1.051	large
xgblss_Gaussian	37.797	0.073	0.031	[0.048, 0.110]	-2.053	large
flexcode_randomforest	38.787	0.076	0.017	[0.051, 0.085]	-3.654	large
flexcode_xgboost	39.762	0.077	0.013	[0.064, 0.088]	-4.896	large
crepes_catboost_difficulty_mondrian	40.960	0.074	0.012	[0.066, 0.106]	-5.135	large
crepes_xgb_difficulty_mondrian	41.045	0.075	0.011	[0.066, 0.119]	-5.372	large
catboost_quantile	44.554	0.201	0.046	[0.162, 0.244]	-4.040	large
nflows_rqs	44.851	0.246	0.152	[0.121, 0.461]	-1.500	large
xgb_vector_quantile	45.614	0.228	0.078	[0.161, 0.310]	-2.706	large
surjectors_maf	46.332	0.323	0.182	[0.173, 0.513]	-1.656	large
forest_diffusion_flow	46.441	0.424	0.099	[0.344, 0.519]	-4.029	large

Table 5: Autorank leaderboard for absolute deviation from nominal 90% coverage (lower is better). Column definitions as in Table 3.

	MR	MED	MAD	CI	γ	Magnitude
exaonetabular	8.564	0.758	0.726	[0.181, 3.681]	0.000	negligible
tabpfn_v3	9.851	0.763	0.732	[0.183, 3.809]	-0.005	negligible
finetune_tabpfn_realv2_5_crls	12.649	0.774	0.742	[0.180, 3.832]	-0.015	negligible
finetune_tabiclv2	13.287	0.837	0.805	[0.182, 3.768]	-0.070	negligible
finetune_tabpfn_realv2_5_mse	14.104	0.770	0.738	[0.181, 3.791]	-0.012	negligible
finetune_tabpfn_realv2_5_beta_0.5	14.282	0.770	0.738	[0.181, 3.829]	-0.012	negligible
finetune_tabpfn_realv2_5_beta_0.7	14.644	0.770	0.738	[0.181, 3.835]	-0.011	negligible
nori_30m	14.881	0.762	0.727	[0.180, 3.734]	-0.004	negligible
finetune_tabpfn_realv2_5_beta_0.3	15.267	0.773	0.742	[0.181, 3.829]	-0.014	negligible
finetune_tabpfn_realv2_5_beta_0.9	15.683	0.770	0.738	[0.181, 3.834]	-0.011	negligible
finetune_tabpfn_realv2_5_crps	15.807	0.772	0.741	[0.181, 3.807]	-0.013	negligible
finetune_tabpfn_realv2_5_beta_1.1	15.871	0.773	0.742	[0.181, 3.806]	-0.014	negligible
tabiclv2	15.901	0.834	0.798	[0.188, 4.023]	-0.068	negligible
finetune_tabpfn_realv2_5_beta_0.1	16.248	0.774	0.743	[0.181, 3.829]	-0.015	negligible
finetune_tabpfn_realv2_5_beta_1.8	16.644	0.770	0.739	[0.181, 3.848]	-0.011	negligible
finetune_tabpfn_realv2_5_beta_1.5	16.693	0.772	0.741	[0.181, 3.812]	-0.013	negligible
finetune_tabpfn_realv2_5_beta_1.3	16.827	0.769	0.738	[0.181, 3.811]	-0.011	negligible
finetune_tabpfn_realv2_5_beta_1.9	16.847	0.772	0.741	[0.181, 3.846]	-0.013	negligible
finetune_tabpfn_realv2_5_wCRPS_center	16.891	0.768	0.737	[0.181, 3.822]	-0.009	negligible
finetune_tabpfn_realv2_5_beta_1.7	16.980	0.770	0.739	[0.181, 3.839]	-0.011	negligible
finetune_tabpfn_realv2_5_wCRPS_left	17.797	0.774	0.743	[0.181, 3.826]	-0.015	negligible
tabpfn_v2_6	18.079	0.772	0.737	[0.182, 4.018]	-0.014	negligible
finetune_tabpfn_realv2_5_wCRPS_right	18.436	0.777	0.746	[0.181, 3.819]	-0.018	negligible
finetune_tabpfn_realv2_5_cde	18.520	0.776	0.745	[0.181, 3.829]	-0.017	negligible
tabpfn_realv2_5	19.267	0.789	0.752	[0.181, 3.822]	-0.029	negligible
nori	19.614	0.766	0.732	[0.181, 3.759]	-0.008	negligible
finetune_tabpfn_realv2_5_ce	19.619	0.776	0.745	[0.181, 3.829]	-0.017	negligible
finetune_tabpfn_realv2_5_is_90	20.233	0.788	0.757	[0.181, 3.859]	-0.027	negligible
pytabkit_realmlp_hpo_cv_8_new	22.723	0.769	0.735	[0.187, 3.728]	-0.011	negligible
crepes_tabiclv2	23.208	0.846	0.804	[0.188, 3.886]	-0.078	negligible
tabm_hpo_cv_8_tabarena	26.713	0.846	0.773	[0.188, 3.906]	-0.080	negligible
crepes_tabiclv2_mondrian	29.307	0.851	0.808	[0.191, 3.918]	-0.082	negligible
catboost_quantile	33.703	0.849	0.807	[0.210, 4.982]	-0.080	negligible
forest_diffusion_flow	35.525	0.862	0.784	[0.210, 5.007]	-0.093	negligible
tabm_d	35.634	0.847	0.807	[0.197, 4.997]	-0.079	negligible
pytabkit_realmlp_td	35.960	0.858	0.802	[0.206, 4.896]	-0.089	negligible
ngboost_gaussian	37.941	0.911	0.834	[0.238, 5.659]	-0.132	negligible
xgb_vector	38.970	0.888	0.823	[0.246, 5.254]	-0.113	negligible
crepes_catboost_difficulty	39.198	0.891	0.839	[0.243, 5.200]	-0.115	negligible
xgb_vector_quantile	39.921	0.898	0.817	[0.229, 5.225]	-0.122	negligible
xgblss_Gaussian	40.277	1.089	1.028	[0.246, 6.058]	-0.251	small
cde_MDN	42.158	0.969	0.934	[0.250, 6.554]	-0.171	negligible
crepes_catboost_difficulty_mondrian	42.218	0.906	0.862	[0.252, 6.248]	-0.125	negligible
flexcode_randomforest	42.406	0.923	0.851	[0.269, 5.555]	-0.141	negligible
crepes_xgb_difficulty	42.752	0.919	0.866	[0.242, 5.388]	-0.136	negligible
cde_NF	42.772	0.966	0.928	[0.251, 5.416]	-0.168	negligible
pymc_bart	44.396	1.386	1.301	[0.287, 8.595]	-0.403	small
nflows_rqs	44.465	0.968	0.906	[0.267, 5.539]	-0.172	negligible
crepes_xgb_difficulty_mondrian	45.040	0.939	0.886	[0.251, 5.568]	-0.151	negligible
flexcode_xgboost	45.356	1.118	1.081	[0.320, 6.612]	-0.264	small
surjectors_maf	45.871	0.922	0.837	[0.336, 5.392]	-0.142	negligible

Table 6: Autorank leaderboard for RMSE (lower is better). Column definitions as in Table 3.

E ABLATION STUDY WITH RESPECT TO DATASET SIZE

To investigate if our ranking results are robust to the choice of dataset size, we perform an ablation study where we re-run the benchmark on all datasets but with capped dataset sizes of 1000, 2000, 3000 and 8000 samples (test + train size). The results are shown in Figure 5.

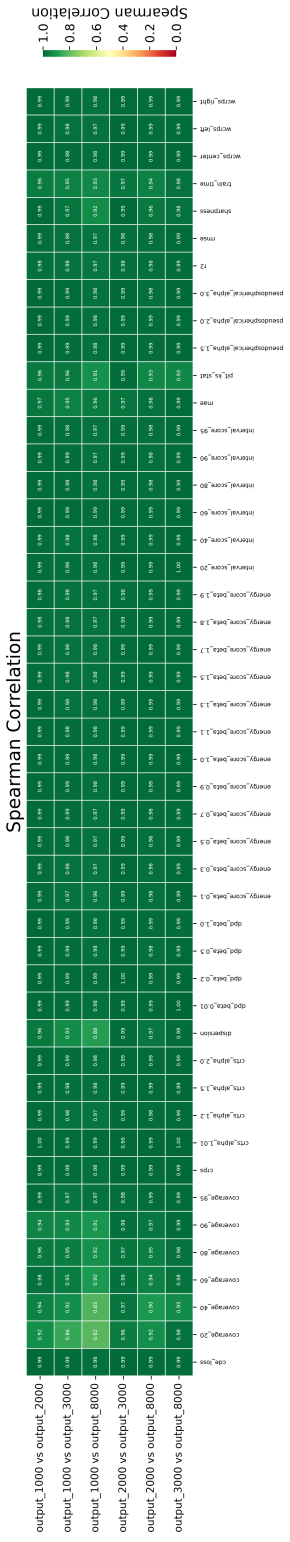


Figure 5: Ablation study on dataset size. We compare model rankings across all six pairs of sample-size caps (1000, 2000, 3000 and 8000 training plus test observations) using Spearman correlation over the 51 models common to all four runs. The heatmap displays 47 metrics, excluding the PIT KS test p-value, and shows high correlations across sample sizes.

We observe some minor variation across metrics and models; this is expected because different scoring rules can have different finite-sample efficiencies and may converge at different speeds during finetuning, which can affect rankings at smaller sample sizes (Waghmare & Ziegel, 2025).

Let $\mathcal{M}_{a,b}^{(p)}$ be the set of models common to sample-size outputs (folders) a and b for metric p, where $n_{a,b}^{(p)} = |\mathcal{M}_{a,b}^{(p)}|$. For each model $i \in \mathcal{M}_{a,b}^{(p)}$, let $r_i^{(a,p)}$ and $r_i^{(b,p)}$ be its ranks in folders a and b for metric p. The Spearman rank correlation $\rho_{a,b}^{(p)}$ is defined as:

$$\rho_{a,b}^{(p)} = \frac{\sum_{i=1}^{n_{a,b}^{(p)}} \left(r_i^{(a,p)} - \bar{r}^{(a,p)}\right)\left(r_i^{(b,p)} - \bar{r}^{(b,p)}\right)}{\sqrt{\sum_{i=1}^{n_{a,b}^{(p)}} \left(r_i^{(a,p)} - \bar{r}^{(a,p)}\right)^2}\sqrt{\sum_{i=1}^{n_{a,b}^{(p)}} \left(r_i^{(b,p)} - \bar{r}^{(b,p)}\right)^2}}$$

where $\bar{r}^{(a,p)}$ and $\bar{r}^{(b,p)}$ are the mean ranks. Ties are handled by assigning average ranks.

F Workflow for contributing to the benchmark

In order to facilitate contributions to our benchmark from other researchers, we briefly elaborate on the workflow to do so.

1. `git clone --recurse-submodules https://anonymous.4open.science/r/ScoringBench-A25F/`, make sure to have git-lfs installed.
2. Add your custom wrapper with a unique name to the folder `scoringbench/wrapper/` following the template of existing wrappers. Ensure it adheres to the expected interface for seamless integration.
3. Run the benchmarking via python `run_bench_regression.py` (or parallelize via slurm with `sbatch array=0103 run_benchmark.sbatch`)
4. Run the ranking method by executing `python autorank_leaderboard.py` which will evaluate both Autorank and z-score ranking and generate the leaderboard outputs.
5. Commit your model parquet file (documenting each run) and the updated leaderboard CSVs in `/output/figures/leaderboard/`. Since the output repository is separate from the main repository, push to both. This serves as a public ledger and allows traceability.
6. Create a pull request to the ScoringBench repository for review; contributions that meet standards will be merged.
7. Upon merge, `https://incredible-truffle-9624b2.netlify.app/` will automatically display the updated leaderboard and data is archived in the git lfs repository for reproducibility.

G Broader Impact

Our work evaluates tabular foundation models, their fine-tuned variants, and additional baseline methods under a range of proper scoring rules, and compiles the results into a comprehensive benchmark. While our findings highlight that the choice of evaluation metric constitutes an important modeling decision with potential implications across various applications, we do not identify any specific broader impacts that require particular emphasis.