PROXYBENCH: A TABULAR CLASSIFICATION BENCHMARK THAT MIRRORS ENTERPRISE DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Enterprises are increasingly deploying machine-learning models for tabular data. However, they face a structural evaluation gap: Very few proprietary enterprise datasets are publicly released, and public benchmarks often fail to capture enterprise deployment realities. Similarly, as model development within the scientific community relies on openly accessible benchmarks, it cannot directly address enterprise challenges and needs. To fill this gap, we introduce `ProxyBench`, an open-source enterprise-grade classification benchmark containing 134 tasks on 90 different tables. To ensure the similarity of `ProxyBench` with enterprise data without disclosing sensitive information, we first generate tasks from a large pool of public datasets. Then, we use a privacy-preserving data fingerprint to automatically identify the ones having the most similar characteristics to the tasks of an internal, proprietary benchmark. By comparing the resulting aggregated statistics, we show that `ProxyBench` is much more statistically similar to enterprise-grade data than existing open-source benchmarks. Furthermore, our experimental evaluation of eleven baseline models shows that `ProxyBench` offers a model ranking with a superior Spearman's correlation with the internal benchmark than open-source alternatives. `ProxyBench` enables reproducible and accessible evaluation of tabular models in an enterprise-grade context. It provides the research community with a tool to develop tabular learning methods that more closely reflect real-world enterprise data.

1 INTRODUCTION

Tabular data is the dominant format in enterprise settings and is essential for critical operations across industries such as finance, healthcare, retail, and logistics (van Breugel & van der Schaar, 2024; Klein & Hoffart, 2025). While prediction on tabular data has been traditionally dominated by tree-based approaches (Chen & Guestrin, 2016; Breiman, 2001), the last few years have seen a rapid evolution of transformer-based tabular foundation models (Hollmann et al., 2023; Qu et al., 2026; Spinaci et al., 2025) that outperform traditional approaches on publicly available benchmarks such as TabArena (Erickson et al., 2025). These new models open a lot of possibilities for companies, but also raise a critical question: Are public benchmarks used for their evaluation representative of real-world business environments?

Previous work (Vogelsgesang et al., 2018; Rubachev et al., 2025) and our analysis of seven widely used open source benchmarks show that the answer is no, and they differ both in terms of model ranking and statistical properties, as we will illustrate in detail in Section 4. The root cause is the lack of access: Even though enterprises possess massive volumes of business tabular data reflecting real-world use cases, many factors, e.g., legal constraints, privacy regulations, and competitive concerns, prevent their public release. Researchers, meanwhile, measure their progress against publicly available benchmarks, often curated years ago for different purposes (Kohli et al., 2024), highlighting a critical asymmetry: Researchers have the know-

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.

how to develop state-of-the-art models but limited access to real-world enterprise data, while companies possess large amounts of data that are mostly unused for public model evaluation.

To fill this gap, we introduce ProxyBench, an enterprise-grade tabular classification benchmark containing 134 tasks on 90 distinct tables. To avoid compromising confidentiality, we built ProxyBench using public data that mirrors characteristics of enterprise data. To this end, while actual enterprise data cannot be shared, we propose the use of a privacy-preserving *fingerprint* capturing its statistical properties to identify publicly available tables and tasks that are akin to the enterprise domain. We first curated the proprietary EGI-Bench, a set of 166 tabular classification tasks extracted from an internal collection of EntCorp [1], and RawTabs, a collection of 270k unlabeled open-source tables extracted from heterogeneous domains. Then, we approached the construction of ProxyBench as a table discovery problem: Identifying tasks inside RawTabs having fingerprints similar to tasks from EGI-Bench. Thanks to heavy parallelization and optimization, our pipeline can process the data and construct the ProxyBench data in less than 24 hours (see Section 3).

Our experimental evaluation spans eleven models, including tabular foundation models (Spinaci et al., 2025; Grinsztajn et al., 2025; Qu et al., 2026), gradient boosting approaches (Chen & Guestrin, 2016), and tree-based approaches (Breiman, 2001). We evaluate those models on EGI-Bench, ProxyBench, and seven widely used open source benchmarks. We show that, according to Spearman's ρ, the model ranking produced by ProxyBench better reflects EGI-Bench than open source competitors (0.873 vs. 0.555). We also observe that the average Total Variation Distance (TVD) between the fingerprint distributions of open-source benchmarks and EGI-Bench is almost three times larger than that of ProxyBench (0.422 vs. 0.153). These results suggest that (1) many enterprise challenges are not represented in current benchmarks, and (2) benchmarks such as ProxyBench are crucial to guide future model development.[2]

2 Related Benchmarks

Many benchmarks for tabular learning derived from open-source data are available. OpenML-CC18 (Bischl et al., 2021), OpenML-CTR23 (Fischer et al., 2023), PMLB (Olson et al., 2017), TALENT (Liu et al., 2025), and TabArena (Erickson et al., 2025) provide large curated collections of classification and regression tasks, but focus mostly on numerical values and require little handling of textual and categorical data. Other works address specific evaluation needs: TabZilla (McElfresh et al., 2023) focuses on challenging datasets, CARTE (Kim et al., 2024) and TextTabBench (Mráz et al., 2025) on semantically rich textual features, BeyondArena (Purucker et al., 2026) supports discipline-specific evaluation, and TEmBed (Vogel et al., 2026) assesses tabular representations at multiple levels.

These benchmarks cover a broad range of tasks, but are not fully representative of enterprise-grade scenarios as we will discuss in detail in Section 4. Indeed, prior studies (Vogelsgesang et al., 2018; Rubachev et al., 2025) suggest that public benchmarks often fail to represent real-world data, while current public enterprise-grade benchmarks are rather limited in size: The SALT benchmark (Klein et al., 2024), a curated dataset extracted from an ERP system, contains four linked tables and eight tasks, while TabRed (Rubachev et al., 2025) contains only eight dataset-task pairs. The limited size of existing enterprise-grade benchmarks calls for larger and more diverse collections, but confidentiality constraints restrict the release of real enterprise data. This motivates the need for a systematic approach to identifying prediction tasks on publicly available data that reflects the statistical characteristics of enterprise data.

[1] EntCorp is a large international enterprise software vendor, whose operations span finance, supply chain, customer analytics, and related domains. The real company name will be disclosed in case of acceptance.

[2] ProxyBench is available at https://anonymous.4open.science/r/ProxyBench-7FE0

Table 1: Characteristics used to compute each capability and capability domain.

Capability	Characteristics	Capability Domain
Height	Num. rows	Small, medium, large
Width	Num. cols	Small, medium, large
Data type	Dtype distr.	Most num., most non-num., balanced
Feature entropy	Feature entropy	Low, high
Target entropy	Target entropy	Low, high
Cardinality	Target cardinality	Low, moderate, high
Imbalance	MMR median	Low, moderate, high
Rare classes	Class freq.	Rare, very rare, no rare
Covariate drift	Feature Normalized Compression Distance	Low, moderate, high
Label drift	Target Normalized Compression Distance	Low, moderate, high

3 PROXYBENCH CONSTRUCTION

We approach the construction of `ProxyBench` as a table discovery problem: First, we curated `EGI-Bench`, a proprietary benchmark consisting of a variety of real-world enterprise predictive tasks. Next, we assembled `RawTabs`, a large collection of publicly available tables, however, without a prediction target. Our goal is to identify prediction tasks in `RawTabs` that resemble those in `EGI-Bench`. This requires building a compact task representation and an efficient search procedure to identify tasks with similar representations. In the following, we introduce a statistical task fingerprint that allows us to estimate table and target similarity. Based on this, we first use target-agnostic characteristics to narrow the search across `RawTabs` to a set of candidate tables for each task in `EGI-Bench`. Finally, we identify, rank, and validate classification tasks within those candidate tables to obtain a suitable predictive task.

3.1 TASK FINGERPRINT

A *task* is composed of a table, a target column, and a split. To capture the statistical properties of `EGI-Bench` without disclosing sensitive information, we built a fingerprint describing enterprise tasks using a set of data capabilities, i.e., categorical values derived from scalar data characteristics. In this section, we outline target-agnostic and target-aware characteristics that we selected and how we derived the capabilities used to build `ProxyBench`. Table 1 provides a high-level overview of what characteristics are used to compute which capability and the domain of each capability.

Target-agnostic characteristics. First, we collect simple table-level information such as the number of rows and columns, complexity, and data types. That is, we collect dimensional information using the number of r rows and c columns, and content-wise information by measuring, at a high level, the data-type distribution across features. Specifically, we collect the percentages of non-numerical features, $\rho_{\text{non-num}}$, and numerical features, ρ_{num}. We consider all text-based columns as non-numerical and do not distinguish between categorical and free text features.

Finally, we estimate the table complexity by approximating the feature entropy. As enterprise tables contain heterogeneous data types beyond simple numerical and categorical features, a precise statistical estimate of the entropy is unfeasible. Therefore, as a proxy, we use the average compressed bits per cell of the serialized features via a column-wise serialization and compression procedure. That is, given the MessagePack serialization of a column, we calculate the average size in bits per cell after compression using the Zstandard compression across all columns. With this, we obtain a proxy estimate of the feature entropy H_F.

Target-aware characteristics. To obtain task-level information, we collect target-aware characteristics. In the following, we refer to the features as $\mathbf{X}$, to the target as $\mathbf{Y}$, and to the set of target classes as $\mathcal{Y}$.

We partition each task's data into training, validation, and test sets containing 70%, 10%, and 20% of the rows, respectively. We stratify these splits by target class to approximately preserve the full dataset's class proportions in each subset. We denote the features and targets of the combined training and validation sets as $\mathbf{X}_{tv}$ and $\mathbf{Y}_{tv}$, and those of the test set as $\mathbf{X}_t$ and $\mathbf{Y}_t$, respectively.

We collect information about the target distribution using two measures: the target cardinality $|\mathcal{Y}|$, i.e., the distinct number of classes in the target column, and the target entropy, i.e., the average information of the label distribution. Similarly to the feature entropy, we estimate it via a compression proxy, by calculating the bits per cell after compressing the MessagePack serialization of the target. While, strictly speaking, for a classification target, a standard statistical estimate is possible, this would lead to an incompatibility in comparing the feature and target entropies, or calculating a proxy for covariate drift, as we detail below. To quantify the imbalance of the target class, in addition to reporting the set of class probabilities $P = \{p_c : c \in \mathcal{Y}\}$, we use the Max-to-Median Ratio (MMR), i.e., the ratio of the probability of the most frequent class over that of the median class frequency.

To estimate the drift in the feature and target column distribution across splits, we use the Normalized Compression Distance (NCD) (Cilibrasi & Vitányi, 2005), again using the average column-wise compression after MessagePack serialization, both for features and target, denoted as $\mathrm{NCD}(\mathbf{X}_{tv}, \mathbf{X}_t)$ and $\mathrm{NCD}(\mathbf{Y}_{tv}, \mathbf{Y}_t)$ respectively. Lower values indicate greater similarity under the chosen compressor.

Capabilities. To obtain a coarse, privacy-preserving fingerprint, we cluster the calculated table and target characteristics into so-called capabilities, summarized in Table 1. Most of them are obtained through simple binning of a data characteristic, i.e., given a characteristic s and two thresholds $\tau_1, \tau_2 \in \mathbb{R}^+ : \tau_1 < \tau_2$, the value of a capability c_{bin} is obtained as:

$$c_{\text{bin}} = \begin{cases} \text{small (low)} & \text{if } s \leq \tau_1 \\ \text{medium (moderate)} & \text{if } \tau_1 < s \leq \tau_2 \\ \text{large (high)} & \text{if } s > \tau_2 \end{cases} \tag{1}$$

For example, we obtain the *Width* capability by discretizing the attribute *Num. columns* into three intervals: $small$ (at most ten columns), $medium$ (more than ten columns and at most one hundred), and $large$ (more than one hundred columns). The thresholds are different for each capability and presented in full detail in Appendix A.

The capabilities obtained differently are *Data type* and *Rare classes*. *Data type* can be either $mostly_numeric$, $mostly_non_numeric$, or $balanced_dtype$, depending on the most frequent data type. The *Rare classes* capability uses two thresholds on relative class frequency, with a lower threshold for very rare classes and a higher threshold for rare classes. A task is labeled `Very_rare` if at least one target class falls below the lower threshold. It is labeled `Rare` if at least one class falls below the higher threshold but none falls below the lower threshold, and `No_rare` if no class falls below the higher threshold.

In the following, we refer to the statistical fingerprint composed of all the capabilities of a task as f, a vector of categoricals having $|\mathcal{C}|$ dimensions. The full list of thresholds used is provided in Appendix A. Next, we will use those table and task fingerprints to identify classification tasks in a large collection of open source tables that mirror the internal data of `EntCorp`.

3.2 Setup and search space reduction

Setup. To build `EGI-Bench`, we use a large internal collection of expert-curated, enterprise-grade tasks extracted from heterogeneous business areas connected to decision making in business operations, e.g., manufacturing, sales, or supply chain management. From this, we curated a subset of 166 classification tasks that are (1) non-trivial, i.e., a base predictor achieves an accuracy of at most 99%, and (2) contain predictive signal, i.e., a base predictor improves over a naive baseline's accuracy by at least the minimum

between ten percentage points and 20% of the maximum possible improvement. As the base predictor, we use AutoGluon (Erickson et al., 2020), as this AutoML system ensembles and stacks a multitude of tabular predictors, it introduces less bias towards a specific model architecture as compared to using a single model.

Our unlabeled collection RawTabs contains around 270k tables that we extracted from a variety of publicly available sources, which are available under commercially reusable licenses. To ensure that these tables are free of sensitive information, we performed extensive Personally Identifiable Information (PII) removal as detailed in Appendix B.2. For all tasks in EGI-Bench we computed the full set of characteristics and capabilities as previously introduced. Furthermore, for the (unlabeled) tables in RawTabs, we initially compute only the target-agnostic characteristics.

Search space reduction. Since computing the capabilities of all possible target columns of all the 270k tables in RawTabs would be too computationally demanding, we reduce the search space by identifying for each of the 166 EGI-Bench tasks the 128 most similar tables in RawTabs.

To compute the table similarity, we represent each table as a vector of target-agnostic characteristics: number of rows, number of columns, percentage of numeric and non-numeric columns, and feature entropy. We then use the cosine similarity between these vectors as the similarity measure. As the characteristics contain values having substantially different domains, e.g., rows can be millions while data type percentages are between zero and one, we align their values using a quantile transformer preprocessor and use the cosine similarity to provide the ranking. Having obtained a list of candidate unlabeled tables, we need to identify potential target columns to create suitable predictive tasks.

3.3 Target-Aware Search

Capability computation and ranking. After identifying for each EGI-Bench task the 128 most similar tables in RawTabs, for each table with less than 10% duplicate rows, we parse each column to identify columns that are suitable classification targets. A column is a suitable classification target if it contains booleans, low-cardinality integers, or low-cardinality short strings. Columns that do not match this criterion are not considered as a potential prediction target. Classification targets with more than 30% null values or fewer than thirty non-null values are skipped as well. Each pairing of a table from RawTabs with a suitable target column defines a task, and we compute its splits and fingerprint as described in Section 3.1.

Then, all the tasks identified are ranked by fingerprint similarity with their corresponding EGI-Bench task. Given the set of capabilities $\mathcal{C}$, the fingerprint f_{os} of a task from RawTabs, and the fingerprint f_{en} of a task from EGI-Bench, we compute the fingerprint similarity as

$$\text{sim}(f_{\text{os}}, f_{\text{en}}) = \frac{1}{|\mathcal{C}|} \sum_{g \in \mathcal{C}} \mathbf{1}\left[f_{\text{os}}(g) = f_{\text{en}}(g)\right] . \tag{2}$$

Task validation and pruning. After ranking the suitable tasks, we aim at identifying, for each EGI-Bench task, ten valid tasks from RawTabs. We process ranked candidates sequentially, training a naive classifier and an AutoGluon predictor with a 45-minute time budget per task, and retain a task only if it satisfies the same non-triviality and predictive-signal criteria used for EGI-Bench (see Section 3.2). To incentivize diversity, we allow at most one valid task per table.

To obtain a final set of tasks that are matched 1-to-1 with the enterprise ones and are also aligned in terms of model performance with the tasks in EGI-Bench, we use the performance of AutoGluon on the reference task to re-rank the tasks from RawTabs. Then, we use a round-robin approach to select for each task in EGI-Bench one task from the validated candidate tasks.

Expert validation. Given the heterogeneous and open nature of the data contained in RawTabs, its tables present multiple data quality issues. While the aim of ProxyBench is offering statistical similarity and not

Table 2: Size-related benchmark statistics. The exact number of columns and rows of EGI-Bench tables cannot be shared due to privacy regulations.

	# Tables	# Tasks	Number of Rows			Number of Columns		
			Avg	Min	Max	Avg	Min	Max
EGI-Bench	100	166	–	–	–	–	–	–
ProxyBench	90	134	49 616	57	3 934 879	22	6	254
OS	128	152	14 162	500	150 000	93	4	1776
OS-Industry	4	11	1 753 129	141 794	2 319 540	117	21	700

semantic similarity with enterprise data, to ensure the quality of the proposed benchmark, all the tables went through manual revision, aiming at removing poor quality tasks. Our process removed 32 tasks, mostly due to issues related to regression-in-disguise classification targets, uninformative disguised missing values, and hard-to-interpret tables.

Thanks to the heavy parallelization, the construction of ProxyBench, excluding expert validation, required less than 24 hours in total. This automated process enables lightweight iterations. We plan to keep the benchmark aligned with the evolution of the enterprise data at EntCorp. More details about the hardware used, parallelization level, and runtime are provided in Appendix B.3.

4 Evaluation

In this section, we measure how well the capability distributions and model rankings of ProxyBench align with EGI-Bench. We compare ProxyBench against a collection of tabular benchmarks based on open source tables that we call OS and a small collection of enterprise-like benchmarks that we call OS-Industry. Our results suggest that ProxyBench is much closer to our EGI-Bench than existing publicly available alternatives.

4.1 Experimental Setup

Data sources. The OS benchmark collection contains classification tasks extracted from five well-known benchmarks, namely CARTE (Kim et al., 2024), OpenML-CC18 (Bischl et al., 2021), TabArena-Lite (Erickson et al., 2025), Talent-Tiny (Liu et al., 2025), and TextTab (Mráz et al., 2025). The OS-Industry benchmark collection, instead, includes the classification tasks of SALT (Klein et al., 2024) benchmark and TabRed (Rubachev et al., 2025). We call OS-Full the union of OS-Industry and OS. Statistics about the number of tables and size of the OS, OS-Industry, ProxyBench, and EGI-Bench are provided in Table 2. Further technical details about EGI-Bench cannot be disclosed due to privacy regulations.

Models. To produce the model ranks used in Section 4.3, we measure the performance of heterogeneous families of models with a particular focus on recent Tabular Foundation Models, including TabPFN (Grinsztajn et al., 2025), TabICL (Qu et al., 2026), and SAP-RPT-OSS (Spinaci et al., 2025). Due to their lack of native text handling, we evaluate TabICL and TabPFN in both their native variant as well as when combined with a default preprocessor from the skrub library (Cerda et al., 2018). SAP-RPT-OSS, instead, is evaluated in the base setting as it handles textual features natively. We also include RealMLP (Holzmüller et al., 2024) and XGBoost (Chen & Guestrin, 2016), a gradient boosting-based model. Finally, we also evaluate simple baselines, namely, random forest (Breiman, 2001), K-Nearest Neighbors (K=5), a linear classifier, and a naive majority-class predictor. More details about the model configurations and training are provided in Appendix D.2.

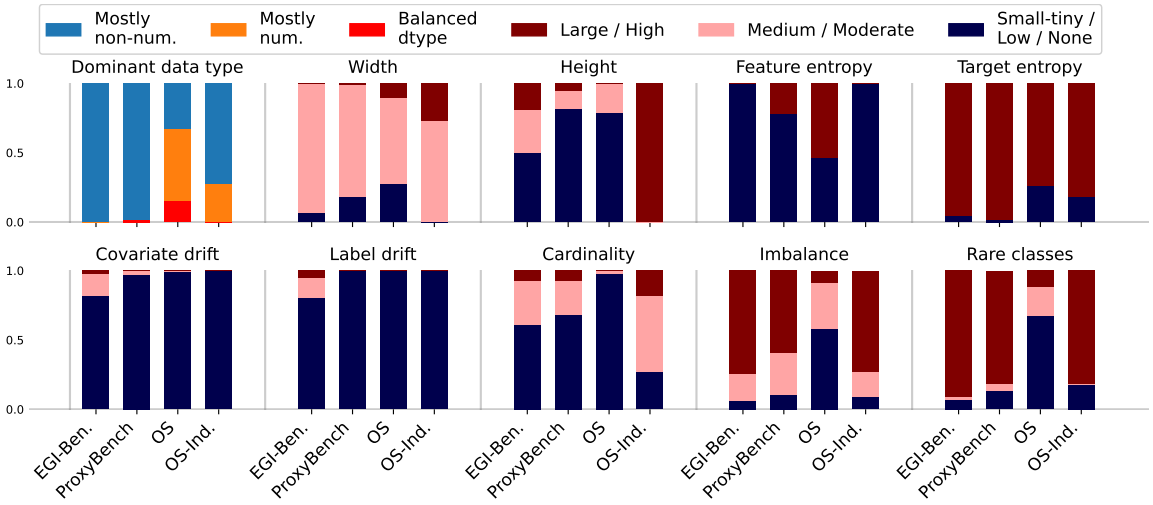


Figure 1: Capability distribution across different benchmarks: Each sub-plot reports a summary of the value distributions of a capability group in `EGI-Bench`, `ProxyBench`, `OS`, and `OS-Industry`. Columns are colored to show what percentage of the tasks of each benchmark falls into a specific bin, for data types, bins are either *mostly non-numeric* or *mostly numeric*, for table size-related characteristics, large, medium, and small, and for the others, low, moderate, and high. As some tables do not have rare classes at all, in their table, the *low* value is interpreted as *none*.

4.2 Capability Alignment

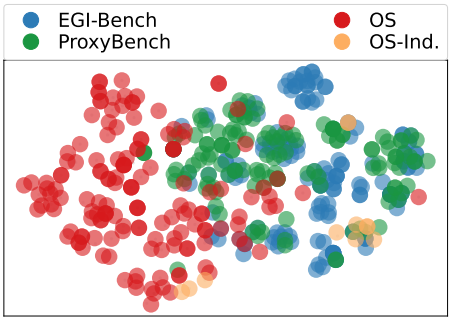
To compare data statistics across benchmarks, we compute the *distribution* of each capability within each benchmark, i.e., the fraction of its tasks that fall into each capability value (see Figure 1).

The capability distributions of `ProxyBench` are generally the most aligned with `EGI-Bench`, except for *Height*, *Feature entropy*, and *Imbalance*. The *Height* mismatch is due to the removal of ill-formed rows during curation. For some tasks, this reduced the row count enough to move them into a lower *Height* category. The *Feature entropy* distribution, instead, suggests that enterprise tables have generally lower information density than open tables; indeed `OS-Industry` shows the same distribution as `EGI-Bench`. Finally, even though the *Imbalance* distribution of `ProxyBench` is less aligned with `EGI-Bench` than `OS-Industry`, it is comparably close. However, note that `OS-Industry` contains only 11 classification tasks. Notably, all the benchmarks except `EGI-Bench` have the same *Label drift* distribution, suggesting that current benchmarks, even enterprise-grade ones, are still mostly IID (Independent and Identically Distributed) and different split strategies should be considered.

To numerically quantify the capability alignment of `ProxyBench`, `OS`, and `OS-Industry`, we use a holistic metric that compares all the capabilities: the average Total Variation Distance (TVD). Specifically, we first compute the TVD of each capability distribution of each benchmark with `EGI-Bench`, then, for each benchmark, we average the results obtained to get one comprehensive number. The per-capability and average TVDs of each benchmark are shown in Figure 2a. The numbers show that the average TVD of `ProxyBench` is almost three times lower than the average TVD of `OS` and lower than that of `OS-Industry`. In every capability, `ProxyBench` is either the closest or the second closest benchmark to `EGI-Bench`. More details about how TVD changes across benchmarks composing `OS` and `OS-Industry` are provided in Appendix C.

Capability	Proxy Bench	OS	OS-Ind.
Width	0.120	0.315	0.273
Height	0.478	0.289	0.807
Cardinality	0.080	0.365	0.336
Covariate Drift	0.151	0.174	0.181
Feature Entropy	0.216	0.539	0.000
Rare Classes	0.096	0.791	0.116
Imbalance	0.150	0.661	0.031
Label Drift	0.193	0.193	0.193
Data type	0.015	0.671	0.273
Target Entropy	0.027	0.221	0.140
Average	**0.153**	0.422	0.235

(a) TVD across benchmarks.



(b) t-SNE of task fingerprints across benchmarks. Each dot represents a different task.

Figure 2: Total Variation Distance (TVD) and t-SNE visualization of fingerprint distributions.

Finally, to compare the task fingerprints, in Figure 2b we show their t-SNE visualization: The public benchmarks lie in completely different parts of the chart, while `ProxyBench` provides superior capability alignment with `EGI-Bench`.

4.3 Ranking alignment

To compare the ranking alignment of the models introduced in Section 4.1 across benchmarks, we first measure the accuracy score of each model on all the tasks in the benchmarks, then we produce a per-task ranking from zero to ten, and finally, for each model and each benchmark, we average the ranks obtained.

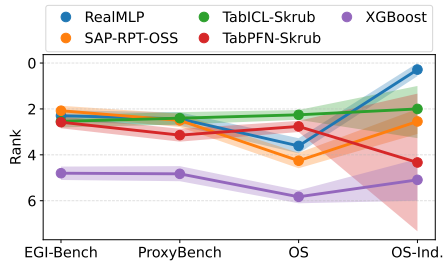
Figure 3b shows how the ranking of a subset of models changes across benchmarks. From `EGI-Bench` to `ProxyBench`, the ranks change the least, with narrow confidence bands. On `OS`, rankings are equally stable but substantially differ from those on `EGI-Bench`, e.g., SAP-RPT-OSS drops from the top positions to around rank four. On `OS-Industry`, instead, the confidence bands are wide and overlap strongly, as its eleven tasks are too few to reliably separate the models. Since rankings on `OS-Industry` alone are not reliable, in Figures 3a and 3c we merge `OS-Industry` and `OS` into `OS-Full`.

Figure 3a summarizes the results of our analysis by comparing for each model its rank on `EGI-Bench` against its rank on `ProxyBench` and `OS-Full` by showing the rank difference of each model. We also report Spearman's ρ with respect to the ranking of `EGI-Bench`. `ProxyBench` offers the highest Spearman's ρ and shows the best alignment across most non-naive baselines. The native variants of TabICL and TabPFN, though, see a large improvement in the rank, achieving +1.6 and +1.4, respectively. This suggests that data in `ProxyBench` is more machine-learning ready than the data in `EGI-Bench`; the same holds for `OS-Full`. The low Spearman's ρ of `OS-Full` highlights the limitations of current open source benchmarks working with enterprise data. To confirm this alignment under a different metric, we also show LogLoss-based Elo scores. As shown in Figure 3c, for most models, their Elo score on `EGI-Bench` is closer to that on `ProxyBench` than `OS-Full`.

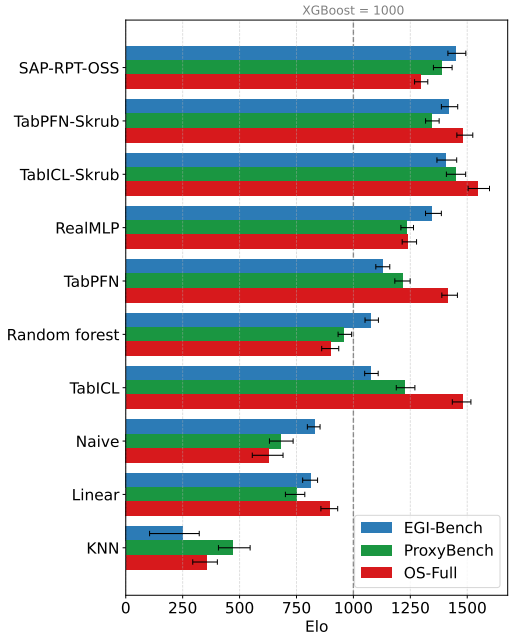
`ProxyBench` does not aim to provide accuracy-level alignment, but only ranking alignment. Details on how the accuracy score varies across benchmarks are provided in Appendix D.3, along with further details on how model rankings change across benchmarks. In Appendix D.1, we extract subsets of `ProxyBench` and `EGI-Bench` having a specific capability value, e.g., all the tasks having *Num cols. medium*, and show that the model ranking stays mostly stable even in subsets of the benchmarks.

Model	EGI-Bench	ProxyBench	OS-Full
SAP-RPT-OSS	2.1	-0.4	-2.1
RealMLP	2.3	-0.1	-1.2
TabICL-Skrub	2.5	+0.1	+0.3
TabPFN-Skrub	2.6	-0.6	-0.2
Random forest	3.7	-1.6	-2.1
XGBoost	4.8	-0.0	-1.0
TabPFN	5.5	+1.4	+2.9
TabICL	5.8	+1.6	+3.0
Linear	7.9	-0.6	+0.4
KNN	8.5	+0.4	+0.3
Naive	9.3	-0.2	+0.1
	Spearman	0.873	0.555

(a) Accuracy-based rank variation.



(b) Accuracy-based rank variation with confidence band.



(c) LogLoss-based Elo scores.

Figure 3: Model rankings across benchmarks. (a) Average accuracy-based rank on `EGI-Bench` and rank difference on `ProxyBench` and `OS-Full` (positive values indicate a better rank than on `EGI-Bench`); the last row reports Spearman's ρ with respect to `EGI-Bench`. (b) Average ranks of selected models with confidence bands. (c) LogLoss-based Elo scores, anchored to XGBoost $= 1000$ (dashed line).

5 Conclusion

In this work, we proposed `ProxyBench`, an enterprise-grade benchmark of 134 classification tasks on 90 open-source tables. To ensure similarity with `EGI-Bench`, a proprietary benchmark of 166 classification tasks extracted from `EntCorp`'s data, while keeping it confidential, we used a privacy-preserving statistical fingerprint to identify tasks with similar characteristics within `RawTabs`, a collection of around 270k publicly available unlabeled tables. The whole table discovery process required less than 24 hours. Our experimental analysis shows that current public benchmarks differ substantially from enterprise data, both in their statistical characteristics and in the model rankings they produce. `ProxyBench`, instead, is substantially closer to enterprise data in both respects, yielding model rankings consistent with `EGI-Bench`.

Limitations and future work. `ProxyBench` is designed to capture the statistical properties of enterprise tables through a privacy-preserving fingerprint; semantic, relational, and temporal aspects of enterprise data are outside the scope of this first version. Moreover, `ProxyBench` is derived from the data of a single large enterprise, although our pipeline is generic and can be applied by other organizations. We plan to make `ProxyBench` a living benchmark that follows the evolution of `EntCorp`'s data, and to extend it along the dimensions left out of this first version: Semantic similarity, by also matching tasks on the meaning of their columns and targets; relational structure, by including tasks over linked tables; and temporal dynamics, by adopting time-based splits that reflect distribution shifts. Future releases will also include regression tasks and stricter data quality controls.

AI USE STATEMENT

In this work, we used generative AI tools for the following task requiring disclosure: implementing methods.

We have not used generative AI tools for the following other tasks requiring disclosure: developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting results. The remaining tasks requiring disclosure, namely generating synthetic datasets and assisting with translation, were not applicable to this work.

Additionally, we used generative AI tools for the following tasks for which disclosure is recommended but not required: creating or modifying scientific figures or images, creating or editing software code, and editing the manuscript to improve readability.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness using human-designed tests. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

`ProxyBench` is constructed from existing publicly available tables rather than released proprietary enterprise records. `EGI-Bench` remains confidential and serves as a reference for selecting public tasks through coarse statistical fingerprints and for aggregate evaluation. To reduce the risk of exposing personal information in the public data, we apply the column-level PII screening and removal procedure described in Appendix B.2. Automated screening and coarse statistical summaries should not be interpreted as guarantees that all privacy risks have been eliminated. Use and redistribution of the source datasets remain subject to their respective licenses. Public datasets may contain societal biases, and similarity to the reference enterprise benchmark does not establish fairness, safety, or suitability for a particular deployment. `ProxyBench` is intended to support comparative model evaluation; downstream applications require context-specific assessment.

REPRODUCIBILITY STATEMENT

`ProxyBench` is available at `https://anonymous.4open.science/r/ProxyBench-7FE0`. Section 3 describes the task fingerprints and benchmark construction procedure, with capability thresholds provided in Appendix A. Table 4 summarizes the data sources, Appendix B.2 describes the PII-cleaning procedure, and Appendix B.3 reports computational resources and construction runtimes. Section 4.1 and Appendix D.2 describe the evaluation setup, model versions, configurations, and preprocessing. These details support reproduction of the methodology and evaluation on accessible datasets. `EGI-Bench` contains proprietary enterprise data and cannot be released; experiments involving its records, therefore, cannot be independently rerun using public materials alone. Reproducing the original benchmark-selection process also requires access to the reference information derived from the private benchmark.

REFERENCES

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*,

2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c7e1249ffc03eb9ded908c236bd1996d-Abstract-round2.html.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. Similarity encoding for learning with dirty categorical variables. *Machine Learning*, 107(8):1477–1494, 2018. URL https://skrub-data.org/. Available at: https://skrub-data.org/.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pp. 785–794. ACM, 2016.

Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Trans. Inf. Theory*, 51(4):1523–1545, 2005. doi: 10.1109/TIT.2005.844059. URL https://doi.org/10.1109/TIT.2005.844059.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

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

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. Openml-ctr23–a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023.

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

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations 2023*, 2023.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. In *NeurIPS*, 2024.

Myung Jun Kim, Léo Grinsztajn, and Gaël Varoquaux. CARTE: pretraining and transfer for tabular learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2 024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23843–23866. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/kim24d.html.

Tassilo Klein and Johannes Hoffart. Foundation models for tabular data within systemic contexts need grounding. *CoRR*, abs/2505.19825, 2025.

Tassilo Klein, Clemens Biehl, Margarida Costa, Andre Sres, Jonas Kolk, and Johannes Hoffart. SALT: Sales autocompletion linked business tables dataset. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=UZbELpkWIr.

Ravin Kohli, Matthias Feurer, Katharina Eggensperger, Bernd Bischl, and Frank Hutter. Towards quantifying the effect of datasets for benchmarking: A look at tabular machine learning. In *ICLR Workshop*, volume 2, pp. 6, 2024.

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. *J. Mach. Learn. Res.*, 26:226:1–226:16, 2025.

Duncan C. McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C., Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? In *NeurIPS*, 2023.

Microsoft. Presidio - data protection and de-identification sdk, 2020. URL https://github.com/microsoft/presidio. Version 2.2.362.

Ines Montani, Matthew Honnibal, Matthew Honnibal, Adriane Boyd, Sofie Van Landeghem, and Henning Peters. explosion/spacy: v3.7.2: Fixes for apis and requirements, October 2023. URL https://doi.org/10.5281/zenodo.10009823.

Martin Mráz, Breenda Das, Anshul Gupta, Lennart Purucker, and Frank Hutter. Towards benchmarking foundation models for tabular data with text. *CoRR*, abs/2507.07829, 2025.

Randal S. Olson, William G. La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10 (1):36:1–36:13, 2017. doi: 10.1186/S13040-017-0154-4. URL https://doi.org/10.1186/s13040-017-0154-4.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

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

Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. Tabred: Analyzing pitfalls and filling the gaps in tabular deep learning benchmarks. In *ICLR*. OpenReview.net, 2025.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. Contexttab: A semantics-aware tabular in-context learner. In *NeurIPS*, 2025.

Boris van Breugel and Mihaela van der Schaar. Position: Why tabular foundation models should be a research priority. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48976–48993. PMLR / OpenReview.net, 2024.

Liane Vogel, Kavitha Srinivas, Niharika S. D'Souza, Sola Shirai, Oktie Hassanzadeh, and Horst Samulowitz. Towards universal tabular embeddings: A benchmark across data tasks. *CoRR*, abs/2604.21696, 2026.

Adrian Vogelsgesang, Michael Haubenschild, Jan Finis, Alfons Kemper, Viktor Leis, Tobias Mühlbauer, Thomas Neumann, and Manuel Then. Get real: How benchmarks fail to represent the real world. In *DBTest@SIGMOD*, pp. 1:1–1:6. ACM, 2018.

A Capability thresholds

Table 3 specifies the thresholds and conditions used to map the characteristics introduced in Section 3.1 to categorical capability values. The same mapping is applied across benchmarks so that their task fingerprints use a common representation.

Most capabilities are discretized into three ordered categories using two thresholds, $\tau_1 < \tau_2$: the lowest category includes values at or below τ_1, the intermediate category includes values above τ_1 and at or below τ_2, and the highest category includes values above τ_2. Feature entropy and target entropy instead use a single threshold, producing two categories, *Low* and *High*. Table 3 specifies the threshold values and interval boundaries for each capability.

The *Data type* and *Rare classes* capabilities follow separate rules. A table is classified as mostly numerical or mostly non-numerical when the corresponding feature type accounts for more than half of its columns, and as balanced when the proportions are equal. For *Rare classes*, a task is assigned *Has very rare* if at least one target class has a relative frequency below 5%. It is assigned *Has rare* if at least one class has a frequency below 10% but none falls below 5%, and *Has no rare* otherwise. These conditions make the three categories mutually exclusive.

B Benchmark construction

B.1 Dataset sources

While `RawTabs` consists of publicly available tables, the collection itself is an internal asset of `EntCorp` and cannot be disclosed. We therefore report only the provenance of the tasks retained in `ProxyBench` after automated selection and expert validation in Table 4, including both the number of tasks and the number of distinct source tables. The selected tasks originate from five sources, with Zenodo contributing the largest number of tasks and tables. We report task and table counts separately because a single table can support multiple classification tasks through different target columns.

B.2 PII-Cleaning Procedure

While all the tables in `RawTabs` are available under commercially reusable licenses, they are collected from heterogeneous sources and there is no guarantee that they are free of sensitive information. To prevent PII leakage, we parsed the whole `RawTabs` collection using a distributed Named Entity Recognition (NER)-based approach that, given a table, identifies columns containing sensitive data and removes them.

We approach the PII identification as an NER problem, i.e., we flag as PII the columns that contain entities belonging to sensitive categories, e.g., EMAIL, CREDIT_CARD, PHONE, LOCATION, and PERSON. Given a table, first, we flag as PII-free all the numerical columns and drop columns that cannot be cast to a string. Then, each remaining column is scanned using the spaCy (Montani et al., 2023) and Presidio-analyzer (Microsoft, 2020) systems. Presidio uses regular expressions to identify entities belonging to its default PII list in version 2.2.3, including EMAIL_ADDRESS, PHONE_NUMBER, CREDIT_CARD,

Table 3: Capabilities computed for each candidate task with thresholds.

Capability	Value	Condition
HEIGHT	SMALL	num_rows $\leq 10\,000$
	MEDIUM	$10\,000 <$ num_rows $\leq 100\,000$
	LARGE	num_rows $> 100\,000$
WIDTH	SMALL	num_cols ≤ 10
	MEDIUM	$10 <$ num_cols ≤ 100
	LARGE	num_cols > 100
DATA_TYPE	MOSTLY_NUMERIC	rate_num > 0.5
	MOSTLY_NON_NUMERIC	rate_non_num > 0.5
	BALANCED	rate_non_num $==$ rate_num
FEATURE_ENTROPY	LOW	feature_entropy ≤ 10.0
	HIGH	feature_entropy > 10.0
TARGET_ENTROPY	LOW	target_entropy ≤ 1.0
	HIGH	target_entropy > 1.0
CARDINALITY	LOW	num_target_classes ≤ 10
	MODERATE	$10 <$ num_target_classes ≤ 100
	HIGH	num_target_classes > 100
IMBALANCE	LOW	mmr_median ≤ 1.5
	MODERATE	$1.5 <$ mmr_median ≤ 3.0
	HIGH	mmr_median > 3.0
RARE_CLASSES	VERY_RARE	any class_freq < 0.05
	RARE	any class_freq < 0.10 and no class_freq < 0.05
	NO_RARE	no class_freq < 0.10
COVARIATE_DRIFT	LOW	Feature NCD ≤ 0.05
	MODERATE	$0.05 <$ Feature NCD ≤ 0.10
	HIGH	Feature NCD > 0.10
LABEL_DRIFT	LOW	Target NCD ≤ 0.05
	MODERATE	$0.05 <$ Target NCD ≤ 0.10
	HIGH	Target NCD > 0.10

Source	Link	# Tasks	# Tables
Australian Government Data	data.gov.au	35	14
europa.eu	data.europa.eu	15	13
huggingface	huggingface.co	6	4
texas	data.texas.gov	4	1
zenodo.org	zenodo.org	74	58

Table 4: Number of tasks and distinct tables per source in ProxyBench.

IBAN_CODE, IP_ADDRESS, and LOCATION. SpaCy uses static embeddings to identify entities belonging to the categories PERSON and NORP, which are more ambiguous than the others. While each column must be labeled with only one entity type, spaCy and Presidio can assign entities only at a cell level, and the majority class is selected as representative. If the dominant category is a PII category, the column is flagged as PII. The category PERSON, though, since it is the most ambiguous and names of people may appear in a broader context without necessarily being PII, receives a special treatment, i.e., the bert-base-NER variant of the BERT (Devlin et al., 2019) model is used to validate the PII hypothesis.

B.3 Runtime and resources

All operations ran on Azure CPU clusters, with each node providing 8 CPU cores and 64 GB of RAM.

The search space reduction operation required 36 CPU nodes, running 166 parallel jobs, one for each EGI-Bench task, requiring 54 minutes.

The capability computation and ranking operations required more than twenty thousand parallel jobs, one for each pair of an EGI-Bench task and a RawTabs table, requiring 11 hours and 43 minutes. The long runtime confirms the need for the search space reduction discussed in Section 3.2. Note that the increase in computation is not only driven by the fact that each table may have dozens or hundreds of potential targets, but also that the target-aware capabilities are expensive to compute.

The task validation and pruning operations required running 166 parallel jobs, one for each EGI-Bench task, for 10 hours and 29 minutes.

Overall, the benchmark construction pipeline required less than 24 hours in total.

C Extra capability analysis

Figure 4 extends the capability alignment analysis by reporting the Total Variation Distance (TVD) from EGI-Bench for each individual benchmark included in OS and OS-Industry. Each row compares the categorical distributions of a capability, while the final row reports the average TVD across all ten capabilities. Lower values indicate closer alignment.

ProxyBench achieves the lowest average TVD at 0.153, followed by SALT at 0.228. Among the individual benchmarks in OS, TextTab is the closest to EGI-Bench, with an average TVD of 0.331, while the others range from 0.418 to 0.467. ProxyBench also achieves the lowest TVD for target cardinality, at 0.080, and is closer to EGI-Bench in rare-class prevalence and imbalance than any individual benchmark in OS.

The individual results reveal differences that are less apparent in the aggregate comparison. SALT closely matches several enterprise characteristics, including data type and feature entropy, both with a TVD of 0.000, and rare classes, with a TVD of 0.090. However, it differs substantially in *Height* and target cardinality, with TVDs of 0.807 and 0.608, respectively. TabRed also matches the feature entropy distribution, but

Capability	Proxy Bench	OS	OS-Ind.	Carte	OpenML CC18	Tab-Arena Lite	Talent Tiny	TextTab	SALT	TabRed
Width	0.120	0.315	0.273	0.752	0.316	0.302	0.203	0.156	0.066	1.000
Height	0.478	0.289	0.807	0.325	0.382	0.237	0.440	0.325	0.807	0.807
Cardinality	0.080	0.365	0.336	0.392	0.333	0.392	0.392	0.392	0.608	0.392
Covariate Drift	0.151	0.174	0.181	0.090	0.181	0.181	0.181	0.181	0.181	0.181
Feature Entropy	0.216	0.539	0.000	0.909	0.544	0.368	0.538	0.778	0.000	0.000
Rare Classes	0.096	0.791	0.116	0.934	0.851	0.752	0.756	0.465	0.090	0.600
Imbalance	0.150	0.661	0.031	0.940	0.703	0.668	0.593	0.414	0.253	0.747
Label Drift	0.193	0.193	0.193	0.193	0.193	0.193	0.193	0.193	0.193	0.193
Data type	0.015	0.671	0.273	0.091	0.750	0.632	0.885	0.333	0.000	1.000
Target Entropy	0.027	0.221	0.140	0.042	0.027	0.932	0.004	0.069	0.083	0.291
Average	**0.153**	0.422	0.235	0.467	0.428	0.466	0.418	0.331	0.228	0.521

Figure 4: Extended TVD analysis across benchmarks in `OS-Industry` and `OS`.

has a TVD of 1.000 for both *Width* and data type, contributing to its average TVD of 0.521. These results show that benchmarks drawn from business applications can still differ considerably from the specific enterprise reference represented by `EGI-Bench`. In contrast, `ProxyBench` is consistently among the closest benchmarks to `EGI-Bench`: Its TVD is among the lowest in eight out of ten capabilities, whereas every individual public benchmark that closely matches some capabilities diverges substantially on others.

The analysis also identifies limitations of `ProxyBench`. Its *Height* TVD of 0.478 exceeds that of every individual benchmark in `OS`, and its feature entropy distribution is less aligned than those of SALT and TabRed, which obtain a perfect TVD. Finally, a TVD of zero indicates matching proportions within the defined capability categories; it does not imply identical distributions of the underlying scalar characteristics.

D Additional ranking analysis

D.1 Ranking stability analysis

We assess how ranking alignment with `EGI-Bench` changes when `ProxyBench` and `EGI-Bench` are restricted to subsets of tasks defined by capability-based criteria. Figure 5 reports the number of tasks and Spearman correlation for each subset, using the full benchmark of 134 tasks, with a Spearman correlation of 0.873, as a reference. Subsets that are empty in either `ProxyBench` or `EGI-Bench` are omitted, as no ranking comparison can be computed for them.

For the rare-class sensitivity analysis, we use nested subsets to reflect that very rare classes also satisfy the broader definition of rare classes. The *Rare classes* subset includes all tasks with at least one target class whose relative frequency is below 10%, including those with very rare classes. The *Very rare classes* subset contains tasks with at least one class below 5% and is therefore a subset of the former. This grouping allows us to assess ranking alignment at two levels of class rarity. These nested subsets are used specifically for this analysis; the fingerprint retains the mutually exclusive categories defined in Appendix A.

Several subsets retain or exceed the correlation obtained with the full benchmark. Tasks with low feature entropy retain a Spearman correlation of 0.873 across 102 tasks, while those with small *Height* achieve a Spearman correlation of 0.900 across 109 tasks. Strong alignment also occurs in substantially smaller subsets: moderate target cardinality yields a Spearman correlation of 0.909 across 32 tasks, moderate imbalance yields a Spearman correlation of 0.907 across 40 tasks, and medium *Height* yields a Spearman correlation of 0.882 across 18 tasks. These results show that strong ranking alignment is observed across multiple benchmark subsets with different statistical characteristics.

Subset	Benchmark	Num Tasks	Spearman
-	ProxyBench	134	0.873
Mostly non-numerical	ProxyBench	132	0.873
Height small	ProxyBench	109	0.900
Height medium	ProxyBench	18	0.882
Height large	ProxyBench	7	0.510
Width small	ProxyBench	24	0.818
Width medium	ProxyBench	109	0.873
Feature entropy low	ProxyBench	102	0.873
Feature entropy high	ProxyBench	32	0.373
Target entropy low	ProxyBench	2	0.562
Target entropy high	ProxyBench	132	0.873
Covariate drift low	ProxyBench	127	0.900
Covariate drift moderate	ProxyBench	7	0.687
Has rare classes	ProxyBench	116	0.873
Has very rare classes	ProxyBench	109	0.882
No rare classes	ProxyBench	18	0.755
Cardinality low	ProxyBench	92	0.900
Cardinality moderate	ProxyBench	32	0.909
Cardinality high	ProxyBench	10	0.873
Imbalance low	ProxyBench	14	0.727
Imbalance moderate	ProxyBench	40	0.907
Imbalance high	ProxyBench	80	0.827

Figure 5: Spearman's ρ of different subsets of `ProxyBench`.

However, alignment is not uniform across all capability values. The largest decrease occurs for high feature entropy, where the 32-task subset yields a Spearman correlation of 0.373, compared with 0.873 for the low-entropy subset. Large *Height*, low target entropy, and moderate covariate drift yield Spearman correlations of 0.510, 0.562, and 0.687 across 7, 2, and 7 tasks, respectively. The very small sizes of these latter subsets warrant caution when interpreting their ranking differences.

This analysis suggests strong alignment across many subsets while identifying groups for which the aggregate result is less representative, particularly tasks with high feature entropy. Since the subsets differ in size and may also differ along other capabilities, the observed changes cannot be attributed exclusively to the capability used to define each subset.

D.2 Model details

We follow the evaluation protocol and baseline implementations of ConTextTab (Spinaci et al., 2025), updated to recent model versions.

TabPFN: We evaluate TabPFNv3 using the official PyPI package version 8.0.8. For classification tasks beyond the native 160 class limit, we use the many-class extension from `tabpfn-extensions`. We use default parameters, in particular, using 8 estimators for inference.

TabICL: We evaluate TabICLv2, using the PyPI package version 2.0.1. We use default parameters, in particular, using 8 estimators for inference.

SAP-RPT-OSS: We evaluate SAP-RPT-OSS using the official repository and checkpoints at git version tag v1.1.2[3].

PytabKit: For the evaluation of RealMLP and XGBoost, we use the recent 1.7.3 version of the PyPI `PytabKit` package (Holzmüller et al., 2024) with search spaces following the TabArena setup (Erickson et al., 2025). For RealMLP, we report 5-fold inner-CV tuned results (Erickson et al., 2025). For XGBoost, we report results using the Tuned-Default variant.

Scikit-learn: For the naive, KNN, linear, and Random Forest baselines, we use the corresponding scikit-learn (Pedregosa et al., 2011) estimators with their default settings. We use scikit-learn version 1.5.2.

For preprocessing, we use a default AutoGluon-based preprocessor for all models that do not natively handle modalities beyond numerical values. In particular, we combine all scikit-learn and PyTabkit baselines with an AutoGluon-based standard preprocessor. For textual features, this preprocessor creates ngram-based features. For all Tabular Foundation Models, we provide results with their internal, native encoding. However, for a better treatment of textual features with TabPFN and TabICL, we also report results when combined with the default skrub preprocessor (Cerda et al., 2018).

D.3 Ranking details

Table 5, Table 6, and Table 7 report the average model ranks and accuracies on `EGI-Bench`, `ProxyBench`, and `OS-Full`, respectively. Average ranks summarize relative model performance across tasks, with lower values indicating better performance.

The same four models achieve the best average ranks on both `EGI-Bench` and `ProxyBench`: SAP-RPT-OSS, RealMLP, TabICL-Skrub, and TabPFN-Skrub, although their ordering differs. SAP-RPT-OSS obtains an average rank of 2.1 on `EGI-Bench` and 2.5 on `ProxyBench`, while RealMLP obtains 2.3 and 2.4, respectively. On `OS-Full`, their average ranks deteriorate to 4.1 and 3.5, while the native variants of TabPFN and TabICL achieve stronger relative positions.

The results also highlight differences associated with preprocessing. On `EGI-Bench`, TabPFN-Skrub improves the average rank of native TabPFN from 5.5 to 2.6, while TabICL-Skrub improves that of native TabICL from 5.8 to 2.5. Both improvements are also observed on `ProxyBench`, where the respective ranks improve from 4.1 to 3.1 and from 4.1 to 2.4. This pattern is less consistent on `OS-Full`: TabICL-Skrub remains better ranked than native TabICL, but native TabPFN achieves a slightly better average rank than its Skrub variant.

The alignment between `ProxyBench` and `EGI-Bench` is not perfect at the level of individual models. For instance, random forest obtains an average rank of 3.7 on `EGI-Bench` but 5.3 on `ProxyBench`, while native TabPFN and TabICL rank better on `ProxyBench`. Moreover, as `ProxyBench` targets ranking rather than accuracy alignment, similar ranks do not imply similar absolute accuracies: XGBoost has an average rank of 4.8 on both benchmarks, while its accuracy increases from 0.780 on `EGI-Bench` to 0.847 on `ProxyBench`. Nevertheless, the overall structure of the ranking is preserved: the same four models form the top group on both benchmarks, the same three baselines (KNN, linear classifier, and naive predictor) occupy the last positions, and XGBoost obtains the same average rank. This agreement is reflected in the Spearman correlation of 0.873 between the two rankings, compared with 0.555 for `OS-Full`.

[3] `https://github.com/SAP-samples/contexttab/`

Table 5: `EGI-Bench` model ranking.

Model	Avg. Rank	Accuracy
SAP-RPT-OSS	2.1	0.866
RealMLP	2.3	0.857
TabICL-Skrub	2.5	0.858
TabPFN-Skrub	2.6	0.865
Random forest	3.7	0.841
XGBoost	4.8	0.780
TabPFN	5.5	0.764
TabICL	5.8	0.737
Linear	7.9	0.516
KNN	8.5	0.476
Naive	9.3	0.504

Table 6: `ProxyBench` model ranking.

Model	Avg. Rank	Accuracy
TabICL-Skrub	2.4	0.893
RealMLP	2.4	0.882
SAP-RPT-OSS	2.5	0.883
TabPFN-Skrub	3.1	0.890
TabPFN	4.1	0.851
TabICL	4.1	0.853
XGBoost	4.8	0.847
Random forest	5.3	0.844
KNN	8.1	0.635
Linear	8.5	0.520
Naive	9.5	0.467

Table 7: `OS-Full` model ranking.

Model	Avg. Rank	Accuracy
TabICL-Skrub	2.2	0.877
TabPFN	2.6	0.867
TabICL	2.7	0.868
TabPFN-Skrub	2.8	0.877
RealMLP	3.5	0.867
SAP-RPT-OSS	4.1	0.862
XGBoost	5.8	0.843
Random forest	5.8	0.848
Linear	7.5	0.784
KNN	8.2	0.772
Naive	9.2	0.579