DATASET INFERENCE FOR DATA PROVENANCE AND PRIVACY AUDITING IN TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) are increasingly deployed through black-box APIs and trained on large collections of real-world tabular datasets. As this training paradigm becomes more widely adopted, private or proprietary datasets may be incorporated into pre-training corpora, yet there are currently no dedicated methods for determining whether a given tabular dataset was used to train a TFM. As a solution, we introduce the first dataset inference method for TFMs, aiming to infer whether a suspect dataset was part of a model's pre-training data. We systematically analyze a broad collection of candidate signals that can be observed from a black-box TFM via input manipulation and find that individual signals provide only weak evidence of training data membership. However, by combining multiple complementary signals, we can infer dataset membership for several state-of-the-art TFMs trained on real tabular data, achieving up to 0.988 ROCAUC dependent on the TFM. We then study factors that influence dataset identification, including pre-training data composition, model capacity, and the use of real vs. synthetic data. Our results show that our dataset inference method is a practical auditing tool for detecting privacy leakage and the use of proprietary datasets in TFMs.

1 INTRODUCTION

Foundation models are a powerful paradigm for creating versatile models on various data modalities. As a prototype, large language models (LLMs) use transformer architectures (Vaswani et al., 2017) to process vast amounts of pre-training data, guided by scaling laws that predict performance improvements (Kaplan et al., 2020). In contrast to the rapid progress of this paradigm in the language domain, performance on predictive tabular data tasks has been dominated by gradient-boosted tree-based methods for most of the previous decade (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018). Only with the rise of prior-data fitted networks (Müller et al., 2022; Hollmann et al., 2023; 2025), a transformer-based architecture which exhibits scaling properties similar to LLMs (Ma et al., 2025), have tabular foundation models (TFMs) begun to convincingly outperform tree-based models (Erickson et al., 2025). TFMs are pre-trained on a wide variety of datasets, then for inference take a labeled dataset as context and directly predict on unlabeled examples via in-context learning (Brown et al., 2020), without requiring updates to model weights. This ease of use combined with their strong predictive power has rapidly made TFMs a go-to tool across industries where tabular data is prevalent.

One explanation for the delayed development of TFMs compared to foundation models for other modalities is the relative scarcity of large-scale and high-quality tabular datasets. Unlike text or images which are available in vast quantities on the internet, publicly available tabular datasets tend to be small. The first TFMs circumvented this issue by relying on synthetic data generated from a prior distribution over tabular datasets, which set the trend for many subsequent TFMs (Qu et al., 2025; 2026; Grinsztajn et al., 2025; Zhang et al., 2025b;c; Bouadi et al., 2025; Hosseinzadeh et al., 2026). However, scaling laws for TFMs (Ma et al., 2025) indicate that real-world datasets can lead to better scaling exponents, suggesting that it may be worthwhile to curate larger-scale real-world tabular datasets for pre-training. Recently, there has been increased effort from the research community in this direction (Vanschoren et al., 2014; Eggert et al., 2023; Spinaci et al., 2025).

Unfortunately, the drive to curate ever larger datasets for foundation model consumption has created controversy over ownership and licensing for scraped data (The New York Times, 2023; Carlini et al., 2023; Freeman et al., 2024; Ross et al., 2025). In the tabular space, further pressure has

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.

been added due to the rapid commercialization of TFMs hosted as black-box APIs (Prior Labs, 2025; Fundamental, 2025; Kumo AI, 2025; Neuralk-AI, 2026). Creators of black-box TFMs have an incentive to train on as much real-world tabular data as possible, without diligently obtaining permission from the respective data owners or subjects. Yet there is currently no dedicated method for data owners to determine whether their dataset was used to train a black-box TFM without permission.

This lack of training data transparency creates an auditing problem for both model developers and external data owners. Organizations training or fine-tuning TFMs on sensitive internal data will need to assess the risk of data leakage before releasing their models. Such auditing is especially important in domains such as finance, e-commerce, and healthcare where tabular data is prevalent and often contains highly sensitive information. Conversely, data owners may want to verify whether their datasets were incorporated into a TFM without authorization. However, training-data identification for TFMs is challenging: the models have diverse architectures, attention mechanisms, and may be trained on datasets that are repeatedly subsampled, augmented, transformed, and redefined into different prediction tasks, in the manner of self-supervised learning (Chen et al., 2020; Balestriero et al., 2023; Sui et al., 2024; Ma et al., 2025; Shaheen et al., 2026). These training procedures can substantially weaken direct evidence that a given dataset was part of pre-training.

For these reasons, we investigate the feasibility of *dataset inference* (DI) for TFMs. Similar to membership inference (Shokri et al., 2017; Carlini et al., 2022), which aims to determine whether an individual data point was part of a model's training set, DI asks whether an entire dataset was included in the training corpus of a model (Maini et al., 2021). DI has previously been studied for self-supervised vision models (Dziedzic et al., 2022), LLMs (Maini et al., 2024; Rossi et al., 2026; Zhao et al., 2025), and image generative models (Dubiński et al., 2025; Kowalczuk et al., 2025). TFMs, however, differ from these settings because their predictions depend jointly on the model parameters, the provided context dataset, and the unlabeled query points. This creates both new challenges and new opportunities for DI. In this work, we systematically design TFM-specific input manipulations and output signals to create the first training dataset inference method for TFMs.

Our main contributions are as follows:

1. We formulate the dataset inference task for TFMs and design TFM-specific signals for identification of training datasets.
2. We show that individual signals are weak, but that multi-signal inference can accurately identify training datasets in some state-of-the-art TFMs.
3. We analyze factors in the models and data that drive dataset detectability, and identify pre-training data composition as the main driver of detectability: real and real-derived synthetic datasets leave strong dataset-level traces, while model size alone has little effect and prior-synthetic data leaks only when very few datasets are used.
4. Our experiments highlight the potential of dataset inference as a practical auditing tool for detecting the use of private or proprietary datasets in TFMs.

2 Background and Related Work

2.1 Tabular Foundation Models

TFMs are highly diverse in their architectural details, data treatments, and training methods. The primary commonality is the in-context learning (ICL) prediction mechanism. Each TFM, denoted f_θ, is designed to take in a labeled *context* dataset $\mathcal{D}_{\text{ctx}} = \{X_{\text{ctx}}, y_{\text{ctx}}\}$, along with unlabeled *query* data points $\mathcal{D}_{\text{qy}} = \{X_{\text{qy}}\}$ from the same marginal distribution, and then predicts the corresponding labels y_{qy} in-context (without modifying the weights of f_θ). This process of prediction with TFMs is depicted in Figure 1. Note that TFMs are often designed to perform classification, regression, or both kinds of predictive tasks. More details on the diversity of architectures, data, and training used across TFMs are given in Appendix B.

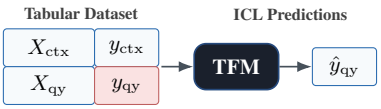


Figure 1: **Prediction with TFMs.**

The first TFMs were pre-trained exclusively on synthetic tabular data generated from a prior over datasets (Hollmann et al., 2023). In this case, there were no concerns over the privacy or ownership of pre-training data. However, training on real data has shown better scaling exponents (Ma et al., 2025), indicating that the ceiling for performance is higher with real tabular data (Spinaci et al., 2025; Garg et al., 2025), as has been the case for other modalities (Kaplan et al., 2020; Fan et al., 2024).

Additionally, organizations train TFMs on their own private datasets to tailor them to specialized domains (Breejen et al., 2024). In these cases the concerns of private information leakage from TFMs are extremely relevant. To tackle a similar problem, Carey et al. (2024) used LLMs to predict on tabular data and introduced a differentially private mechanism (Dwork et al., 2006) to protect data in the *prompt*. Byun et al. (2025) looked at privacy risks of data *generation* using foundation models, including tabular data generation. Neither work investigates pre-training dataset identification for predictive TFMs, which is the focus of our work.

2.2 Membership and Dataset Inference

Membership inference attacks (MIAs) attempt to answer the question of whether a suspect data point was used to train a given model (Shokri et al., 2017; Carlini et al., 2022), and have become the de facto standard for auditing training data privacy in machine learning (Steinke et al., 2023; Rossi et al., 2026; Nasr et al., 2023; Zanella-Beguelin et al., 2023). However, it has recently been shown that for foundation models, MIAs do not significantly outperform random guessing as training datasets grow (Duan et al., 2024; Das et al., 2025; Maini et al., 2024). Thus, MIAs are not suitable for identifying training datasets for these models.

As a solution, dataset inference (DI) (Maini et al., 2021) shifts the focus from individual data points to entire datasets. Originally proposed for resolving model ownership in supervised learning (Maini et al., 2021) and subsequently extended to self-supervised learning (Dziedzic et al., 2022), DI has recently been applied to identify training datasets in LLMs (Maini et al., 2024; Rossi et al., 2026; Zhao et al., 2025). Beyond LLMs, DI has also been used for other generative models, including diffusion models (Dubiński et al., 2025) and image autoregressive models (Kowalczuk et al., 2025). In these settings, DI typically extracts membership signals from individual examples in a suspect dataset, aggregates these signals across the dataset, and applies a statistical test to determine whether the dataset was used during training. TFMs, however, differ fundamentally from these models: they are predictive models pre-trained on collections of datasets, and their predictions are produced through an ICL mechanism. Consequently, the relevant unit of membership for TFMs is not an individual example, but an entire dataset. Because existing DI methods do not directly apply to this setting, we propose the first DI method for TFMs, designed to determine whether a given suspect dataset was used during pre-training.

3 Problem Setup

Our work targets the important question: *Was a given dataset used to train a TFM?* This question is highly relevant both for external and internal audits. These setups, while conceptually similar, differ in their motivation and assumptions on what knowledge the respective parties have.

External Audits - Data Provenance. External parties who own tabular datasets and suspect that these datasets were used to train a TFM without their consent would like to verify this suspicion. They only have black-box access to the TFM, which is deployed behind an API. They hold the dataset that they suspect was used to train the TFM, and some additional datasets known to be non-members, which could be unreleased private datasets or randomly generated datasets from a prior, as are commonly used to train TFMs (Hollmann et al., 2023).

Internal Audits - Privacy Leakage. Organizations train or fine-tune TFMs on their own proprietary datasets. TFMs are most commonly used in domains where tabular data is abundant, including finance, e-commerce, and healthcare and these domains also happen to involve highly sensitive data. Therefore, it is important to internally audit the potential leakage of sensitive information from a TFM before publicly deploying it. An internal auditor needs to verify how much private information could be leaked from their hosted black-box API to an outside user in the worst case. For this setting, we assume the auditor only accesses the black-box API, but to mimic a worst case they assume knowledge of the pre-training dataset members, and also have non-member datasets which could be, for example, internal test datasets used to assess the performance of the TFM, or synthetic data.

We consider dataset inference for both external and internal audits, where the external setting is more challenging as it assumes less knowledge about the TFM's pre-training datasets.

4 Dataset Inference for TFMs

Our DI method for TFMs aims to answer the question of whether a labeled *suspect* dataset $\mathcal{D}_{\text{sus}}$ was used to train f_θ. Therefore, following prior work on DI (Maini et al., 2021; Dziedzic et al., 2022; Maini et al., 2024), it utilizes the trained model f_θ and additional *known* datasets $\{\mathcal{D}_{\text{known}}\}$. For

external audits, $\mathcal{D}_{\text{known}}$ are all non-members, whereas for internal audits $\mathcal{D}_{\text{known}}$ are a mix of members and non-members. Both internal and external audits use only black-box access to f_θ, as through an API. This means the auditor can control the inputs to f_θ, namely X^i_{qy} and $\mathcal{D}_{\text{ctx}}$, and receives the TFM's output distribution $p(y^i_{\text{qy}} \mid X^i_{\text{qy}}; \mathcal{D}_{\text{ctx}})$.

Using f_θ, the auditor extracts a single fingerprint vector $\bar{S}(\mathcal{D}) \in \mathbb{R}^d$ from each dataset $\mathcal{D}$, applied both to the suspect dataset $\mathcal{D}_{\text{sus}}$ and to the known datasets $\{\mathcal{D}_{\text{known}}\}$. Compared to DI in the language (Maini et al., 2024) or vision domains (Dubiński et al., 2025) which calculates membership signals *per sample* and then aggregates over $\mathcal{D}_{\text{sus}}$, in TFMs $\mathcal{D}_{\text{sus}}$ is treated as *a single unit*. This is because TFM pre-training performs batching at the dataset level. Given $\bar{S}(\mathcal{D}_{\text{sus}})$ and $\{\bar{S}(\mathcal{D}_{\text{known}})\}$, the auditor leverages these fingerprints to determine whether $\mathcal{D}_{\text{sus}}$ was used for training. The concrete procedure depends on the auditing setup, i.e., internal vs. external audit, and we instantiate both in the following sections.

Constructing $\bar{S}(\mathcal{D})$ is the central challenge. TFMs have diverse architectures, attention mechanisms, and augment their pre-training datasets in various ways. These factors disrupt direct membership signals that could be gathered from $\mathcal{D}_{\text{sus}}$, and mean that different signals can be informative or not based on the TFM. Below, we show how to carefully craft a wide variety of membership signals for TFMs that reflect the diversity in their pre-training, then turn to the two auditing setups.

4.1 Crafting TFM-Specific Dataset Membership Feature Vectors

A natural starting point is to port standard membership signals, like *loss* or *predictive confidence* (Shokri et al., 2017; Carlini et al., 2022), that have shown success in other domains to TFM DI. As Figure 2 shows for the loss, these signals on their own do not separate member from non-member datasets in TFMs. Since TFMs may see millions of datasets during pre-training, sometimes under many subsamplings, shuffles, and task redefinitions, any single readout captures only a faint trace of membership.

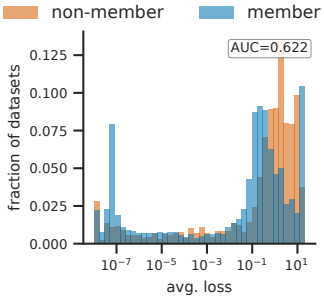


Figure 2: **Classical inference signals are weak predictors.** *Per-dataset* avg. loss does not clearly separate members and non-members of SAP-RPT-1-OSS.

Instead, we take full advantage of the powerful lever an auditor has, namely complete control over the inputs X^i_{qy} and $\mathcal{D}_{\text{ctx}}$. These inputs can be created from the dataset $\mathcal{D}$ (either $\mathcal{D}_{\text{sus}}$ or one of the $\mathcal{D}_{\text{known}}$) in a wide variety of ways, complementing the great diversity in TFM design. We therefore probe models using inputs crafted with specialized TFM augmentations, and assemble the responses into the fingerprint vector $\bar{S}(\mathcal{D}) \in \mathbb{R}^d$.

Each entry of $\bar{S}(\mathcal{D})$ is generated from a triple (manipulation, signal, reduction): a *manipulation* perturbs the TFM's inputs; a *signal* is a scalar read from the TFM's output; a *reduction* first varies the manipulation's parameters, then collapses the resulting sequence of signals into a single number. Stacking the reduction values across all valid triples yields the fingerprint $\bar{S}(\mathcal{D})$. We describe each of these axes now, starting from the outputs.

Signals are scalar functions of the TFM's output distribution $p(y^i_{\text{qy}} \mid X^i_{\text{qy}}; \mathcal{D}_{\text{ctx}})$, and are aggregated over the query points X^i_{qy}. Many classical MIA signals (correct-class confidence, max confidence, margin, loss) are useful in this context. However, since the auditor does not know details of the TFM architecture or training methodology, they cannot assume particular details such as the exact loss that f_θ was trained with. Hence, our signals cover a variety of quantities, including common losses. The full list of 14 signals we use is given in Table 3 in Appendix C.

Manipulations perturb the TFM's inputs X^i_{qy} and $\mathcal{D}_{\text{ctx}}$ in structured ways that take advantage of the common ICL prediction mechanism used by TFMs. Manipulations are designed so that they will generate different patterns of signals when applied to member or non-member datasets. Furthermore, each manipulation comes with parameters that can be adjusted, such as the fraction of $\mathcal{D}$ to split into query vs. context. The general idea is that when $\mathcal{D}$ is a member, f_θ may have internalized some parametric knowledge about the relationship between X and y that allows it to predict more correctly or confidently on queries. While all 19 manipulations we use are detailed in Table 4 in Appendix C, here we give select examples to build intuition:

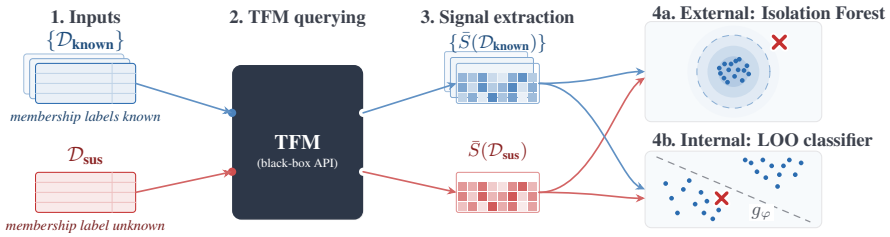


Figure 3: **Overview of our DI procedure**. The auditor queries the black-box TFM API with both known datasets $\{\mathcal{D}_{\text{known}}\}$ and a suspect dataset $\mathcal{D}_{\text{sus}}$ under structured input manipulations, yielding fingerprint vectors $\{\bar{S}(\mathcal{D}_{\text{known}})\}$ and $\bar{S}(\mathcal{D}_{\text{sus}})$. *(a) External audit:* All $\mathcal{D}_{\text{known}}$ are non-members. We fit an Isolation Forest on $\{\bar{S}(\mathcal{D}_{\text{known}})\}$ and score $\bar{S}(\mathcal{D}_{\text{sus}})$ as a candidate anomaly indicating membership. *(b) Internal audit:* $\{\mathcal{D}_{\text{known}}\}$ has a mix of member and non-members. We train a binary membership classifier g_φ on $\{\bar{S}(\mathcal{D}_{\text{known}})\}$, and predict the membership label of $\bar{S}(\mathcal{D}_{\text{sus}})$.

- **M3 Query only**: This manipulation splits $\mathcal{D}$, but then discards $\mathcal{D}_{\text{ctx}}$ and inputs only X_{qy} to f_θ. If $\mathcal{D}$ is a member, the TFM will have encountered inputs like X_{qy} before and may predict some labels correctly, whereas for non-members predictions are more likely to be random.
- **M17 Brainwash**: This manipulation selects rows of $\mathcal{D}_{\text{qy}}$ and copies them into $\mathcal{D}_{\text{ctx}}$ with corrupted labels y_{ctx}. If $\mathcal{D}$ is a non-member f_θ can only rely on the context and will likely predict y_{qy} incorrectly, but for a member dataset f_θ retains parametric knowledge that can overcome the corrupted context to make correct predictions. This is inspired by Brainwash (Wen et al., 2024a).

Reductions harness the various parameters of manipulations to create a sequence of signals which are then converted into a single scalar feature for $\bar{S}(\mathcal{D})$. We use a library of 11 reductions organised by the structure of the manipulation's parameters, given in Table 5 of Appendix C. When parameters correspond to a numeric intensity axis (e.g. noise scale, context size), the sequence forms a curve and we reduce based on its shape, for example taking its *slope* or the *area under the curve*. When parameters lack a directionality (e.g. permutations), we can still reduce with statistical metrics such as *mean*, *standard deviation*, and *range*. Finally, for manipulations that only affect the context, predictions on the same query row are directly comparable across parameter settings, and we add cross-parameter agreement scores like *consistency* and *rank correlation* to capture prediction stability.

In summary, we form a large number of (manipulation, signal, reduction) triples by taking many combinations from the three sets of options (1,021 in total). The resulting reduction values are the features in the fingerprint $\bar{S}(\mathcal{D})$ used for DI. As an example, the feature `brainwash_loss_slope` corresponds to the Brainwash manipulation, measuring the loss signal as the Brainwash parameter (fraction of duplicated rows) is changed, and then computes the slope of that signal curve in parameter space to give a value which can be added to $\bar{S}(\mathcal{D})$.

4.2 External Auditing

For external auditing, the auditor, i.e., the data owner, has black-box TFM access and known non-member datasets $\{\mathcal{D}_{\text{known}}\}$, along with their suspect dataset $\mathcal{D}_{\text{sus}}$. Since the auditor only knows *non-member* datasets and does not already know any *members*, we frame DI as anomaly detection calibrated only on non-members as depicted in Figure 3. We fit an Isolation Forest (Liu et al., 2008; 2012) on the known datasets' fingerprints $\{\bar{S}(\mathcal{D}_{\text{known}})\}$ to characterize the typical non-member behaviour, and score the suspect dataset $\bar{S}(\mathcal{D}_{\text{sus}})$ against this baseline. If the suspect falls outside the normality region, it is flagged as likely to be a pre-training member. This formulation requires neither examples of member datasets, nor training expensive shadow models (Shokri et al., 2017).

Retaining all available features in $\bar{S}$ would create unnecessary noise due to features that measure distribution shifts within non-members, rather than membership information. To ensure each anomaly detection feature is meaningful we randomly split $\{\bar{S}(\mathcal{D}_{\text{known}})\}$ in halves, and as a baseline measure the separation a feature produces. Then, that feature is retained only if its suspect-versus-known separation exceeds the 95th percentile of the baseline.

To evaluate the method, we score a set of $\{\mathcal{D}_{\text{sus}}\}$ which are equally distributed between members and non-members, none of which are part of $\{\mathcal{D}_{\text{known}}\}$. Given the Isolation Forest scores $S_{\text{known}} = \{s_i\}_{i=1}^n$ for $\{\mathcal{D}_{\text{known}}\}$ we turn an individual score s_{sus} for a dataset $\mathcal{D}_{\text{sus}}$ into a conformal p-value (Vovk et al., 2005; Bates et al., 2023), i.e., $p(s_{\text{sus}}) = \left(1 + \sum_{i=1}^n \mathbb{1}[s_i \geq s_{\text{sus}}]\right)/(n+1)$ which can be thresholded.

4.3 INTERNAL AUDITING

In addition to black-box access an internal auditor also has ground truth knowledge of which datasets were used for training the TFM. This follows the setup from prior DI works (Maini et al., 2021; 2024; Kowalczuk et al., 2025; Dubiński et al., 2025). Therefore, unlike for the external auditor, $\{\mathcal{D}_{\text{known}}\}$ contains non-member and member datasets. For the sake of testing, the auditor removes one dataset from $\{\mathcal{D}_{\text{known}}\}$ to play the role of $\mathcal{D}_{\text{sus}}$, ignores its membership label, and tries to determine the label from other available information. We evaluate the DI method holistically by applying leave-one-out cross-validation (LOOCV), in which we iterate the choice of $\mathcal{D}_{\text{sus}}$ over the whole set $\{\mathcal{D}_{\text{known}}\}$. In each iteration, given $\{\mathcal{D}_{\text{known}}\}$ with their membership labels $y_i \in \{0, 1\}$ (non-member vs. member) and fingerprints $\{\bar{S}(\mathcal{D}_{\text{known}})\}$, we train a binary classifier g_φ to predict y_i, and apply it to $\bar{S}(\mathcal{D}_{\text{sus}})$, as depicted in Figure 3. We then report the ROCAUC of the left-out predictions over the LOOCV setup.

Again, for internal auditing we filter out features from $\bar{S}$ to reduce noise. Within each LOOCV fold we use a validation split to rank features by mean absolute SHAP value (Lundberg & Lee, 2017; Lundberg et al., 2019) based on a fit to $\bar{S}$ and the membership labels of that fold's training datasets, using a gradient-boosted tree for g_φ (Chen & Guestrin, 2016). Then we retain only the top K features, with K maximizing the validation ROCAUC divided by $\log_2(K + 1)$.

4.4 BLIND BASELINES

As we cannot assume that the member and non-member dataset follow the same distribution we employ *blind baselines* (Das et al., 2025) when evaluating both external and internal auditing power. If member and non-member datasets come from different distributions, it may be possible for an auditing procedure to distinguish them purely from distributional differences, not from membership information revealed by a TFM. This would constitute a false positive for identifying information leakage from a TFM. Hence, the blind baseline's purpose is to differentiate signals due to distribution shifts from signals induced by training on the data.

For this purpose we replace the target TFM with a "blind" TFM that has entirely different pre-training members, but we use the same datasets $\{\mathcal{D}_{\text{known}}\}$ for calibration, and $\{\mathcal{D}_{\text{sus}}\}$ for testing without changing their membership labels. Since none of $\{\mathcal{D}_{\text{sus}}\}$ are true members of the blind TFM, any observed differences between predictions on $\{\mathcal{D}_{\text{sus}}\}$ and $\{\mathcal{D}_{\text{known}}\}$ must be attributed to the data distribution itself, not membership leakage from a TFM. The excess accuracy on the target TFM over the blind baseline indicates a true DI signal.

5 EXPERIMENTAL EVALUATION

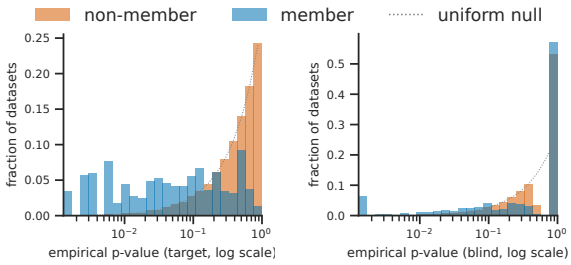
5.1 EXPERIMENTAL SETUP

We present DI experiments on four TFMs: SAP-RPT-1-OSS (Spinaci et al., 2025), TabDPT (Ma et al., 2025), nanoTabPFN (Pfefferle et al., 2025), and Real-TabPFN-2.5 (Grinsztajn et al., 2025). For SAP-RPT-1-OSS, TabDPT, and Real-TabPFN-2.5, we use the pre-trained model weights released by their authors. For nanoTabPFN we train the model using default settings. These four TFMs were each trained on a fixed corpus of datasets which define the members.[1] Any dataset not used for pre-training a given TFM is a valid non-member, which can include held-out test datasets, and synthetic datasets generated from a prior. Details on the member and non-member datasets actually used for experiments with each model are fully described in Appendix D.1. Notably, SAP-RPT-1-OSS and TabDPT were trained entirely on real tabular datasets, whereas nanoTabPFN was trained on a fixed set of synthetic datasets. Real-TabPFN-2.5 was pre-trained on synthetic data drawn fresh from a prior every batch and then fine-tuned on a small set of real tables. We use the real tables as members.

As a blind baseline (Das et al., 2025), we employ TabPFN-2.5 (Grinsztajn et al., 2025), which was not trained with a fixed corpus, but on datasets drawn from a prior for each batch. Hence, TabPFN-2.5 does not share any member datasets in common with the above four TFMs. For internal auditing, we experiment with three tabular algorithms for the membership classifier g_φ: logistic regression, XGBoost (Chen & Guestrin, 2016), and TabPFN-2.5.[2]

[1] Note that the four TFMs have *different* fixed corpora. Hence experimental results are not directly comparable.

[2] We use a TFM for g_φ because TFMs are the strongest available classifiers. We do not wish to confuse the TFM g_φ with the target TFM f_θ that is being audited or the blind baseline; they play independent roles. When g_φ is a TFM, it is not actually trained; instead it uses $\{\bar{S}(\mathcal{D}_{\text{known}})\}$ as context and $\bar{S}(\mathcal{D}_{\text{sus}})$ as query.



(a) Target TFM (b) Blind TFM

Figure 4: **External audit DI.** Empirical p-values from the Isolation Forest fit on $\{\bar{S}(\mathcal{D}_{\text{known}})\}$, and evaluated on various $\mathcal{D}_{\text{sus}}$. (a) shows the target TFM (SAP-RPT-1-OSS) and (b) shows the blind baseline. Suspect datasets that are members concentrate in the low-p tail on the target but cluster at $p \approx 1$ on the blind model.

Table 1: **External-audit DI AUC.** For each target TFM, we show the ROCAUC of membership detection by the Isolation Forest fit on $\{\bar{S}(\mathcal{D}_{\text{known}})\}$. The *Blind* column reports the same pipeline run against TabPFN-2.5 and excess of target over blind quantifies the actual DI signal.

Target TFM	Target	Blind	Excess
SAP-RPT-1-OSS	0.652 $\pm$0.041	0.442 $\pm$0.013	0.210 $\pm$0.045
TabDPT	0.619 $\pm$0.055	0.560 $\pm$0.028	0.059 $\pm$0.040
nanoTabPFN	0.498 $\pm$0.004	0.500 $\pm$0.005	$-$0.002 $\pm$0.002
Real-TabPFN-2.5	0.565 $\pm$0.089	0.557 $\pm$0.096	0.008 $\pm$0.045

Our implementation of the methods is available at `anonymous.4open.science/r/tfm-di-D651`.

5.2 Dataset Inference Performance

External Auditing. Figure 4 shows empirical p-values produced by our anomaly detection method (Section 4.2) on fingerprints from SAP-RPT-1-OSS. On the target TFM, the distribution of anomaly scores differs significantly based on whether a given $\mathcal{D}_{\text{sus}}$ is a true member or not. Many member dataset fingerprints have anomaly scores that sit in the low-p tail and can be confidently inferred as members, while the non-member distribution closely follows the uniform reference (dotted). On the blind TFM, both member and non-member p-values pile near 1, exactly as expected when the TFM has seen neither $\mathcal{D}_{\text{sus}}$ nor $\mathcal{D}_{\text{known}}$ during pre-training.

Table 1 reports the corresponding ROCAUC values across all target TFMs alongside their blind baselines. On SAP-RPT-1-OSS, there is a clear signal that the DI method captures membership information from the target TFM in excess of what a blind baseline achieves. However, this excess is much weaker for TabDPT, showing that it leaks less membership information. NanoTabPFN and Real-TabPFN-2.5 do not leak any membership information under this protocol. There are of course two possible explanations: the auditing procedure does not have enough discriminatory power to extract the membership information; or, these TFMs do not retain any significant membership information in the first place. We investigate the differences between TFMs more in Section 5.4.

Internal Auditing. Figure 5 demonstrates the positive effect of using many TFM-specialized membership features in the fingerprint $\bar{S}(\mathcal{D})$, as compared to Figure 2 which used single classic membership signals. In this demonstration we show 19,888 SAP-RPT-1-OSS member and non-member datasets scored by the logistic regression classifier g_φ, as described in Section 4.3. Using TFM-specialized features greatly increases the predictive power of the audit. Figure 7 in Appendix D.2 shows the same comparison for TabDPT.

Figure 6 reports the internal audit ROCAUC across target TFMs using the TabPFN-2.5 classifier g_φ compared to the blind TFM. Table 2 shows the same information across all classifiers. We find that the classifiers g_φ acting on fingerprints from blind baselines can achieve non-trivial performance, which highlights the necessity of accounting for distributional differences between $\mathcal{D}_{\text{sus}}$ and $\{\mathcal{D}_{\text{known}}\}$. Hence, we focus on the excess as true membership leakage. Success varies substantially: nanoTabPFN and Real-TabPFN-2.5 sit only just above random (AUC $\sim 0.50 - 0.55$) for both the target TFM and the blind baseline, suggesting little to no leakage. In contrast, SAP-RPT-1-OSS reaches significantly higher discrimination than the blind baseline (e.g. AUC 0.99 vs. 0.76), as does TabDPT (e.g. AUC 0.70 vs. 0.59) indicating genuine membership leakage.

These results demonstrate for the first time that (*i*) some TFMs internalize membership information about which datasets they are pre-trained on; and (*ii*) our dataset inference methods can extract that information in the external and internal audit settings.

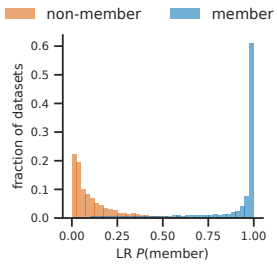


Figure 5: **Multiple signals together succeed.** Our method (Section 4.3) with multiple TFM-specific signals leads to clear separation, where single signals did not (cf. Figure 2).

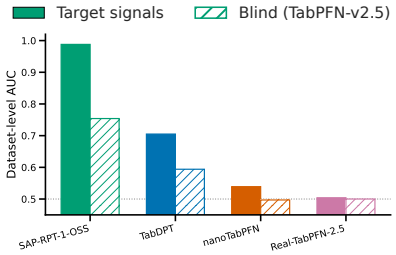


Figure 6: **Internal-audit DI ROCAUC.** DI success varies strongly between target TFMs, from no power (Real-TabPFN-2.5) to an almost perfect ROCAUC of 0.99 (SAP-RPT-1-OSS). Excess over the blind baseline indicates membership leakage.

Table 2: **Internal-audit DI AUC by classifier.** For each target TFM and classifier, we report the test ROCAUC of dataset-membership prediction; the *Blind TFM* column reports the same pipeline run against fingerprints from TabPFN-2.5. Excess of target over blind quantifies the actual DI signal, accounting for distributional differences between $\mathcal{D}_{\text{sus}}$ and $\{\mathcal{D}_{\text{known}}\}$.

Target TFM f_θ	Classifier g_φ	Target TFM	Blind TFM	Excess
SAP-RPT-1-OSS	LR	0.947 ± 0.002	0.683 ± 0.006	$+0.264 \pm 0.006$
	XGB	0.984 ± 0.001	0.760 ± 0.005	$+0.224 \pm 0.005$
	TabPFN-2.5	0.988 ± 0.001	0.754 ± 0.005	$+0.234 \pm 0.005$
TabDPT	LR	0.739 ± 0.040	0.622 ± 0.046	$+0.117 \pm 0.046$
	XGB	0.693 ± 0.042	0.612 ± 0.044	$+0.081 \pm 0.054$
	TabPFN-2.5	0.705 ± 0.041	0.594 ± 0.046	$+0.111 \pm 0.045$
nanoTabPFN	LR	0.510 ± 0.004	0.499 ± 0.004	$+0.011 \pm 0.004$
	XGB	0.549 ± 0.004	0.503 ± 0.004	$+0.046 \pm 0.005$
	TabPFN-2.5	0.539 ± 0.004	0.497 ± 0.004	$+0.042 \pm 0.005$
Real-TabPFN-2.5	LR	0.547 ± 0.092	0.540 ± 0.095	$+0.007 \pm 0.014$
	XGB	0.536 ± 0.079	0.572 ± 0.082	-0.037 ± 0.048
	TabPFN-2.5	0.504 ± 0.095	0.500 ± 0.094	$+0.003 \pm 0.026$

5.3 Which Features Are Most Revealing?

To identify which features within the fingerprints $\bar{S}(\mathcal{D})$ drive accurate membership prediction, we compute SHAP values (Shapley, 1953; Lundberg & Lee, 2017) for the classifier g_φ used for internal auditing. Table 6 in Appendix D.3 shows the top 10 features with the greatest SHAP values computed for the XGB classifier trained on fingerprints from the SAP-RPT-1-OSS target TFM, as compared to the TabPFN-2.5 blind TFM (see Table 7 for TabDPT). A wide variety of manipulations and reductions contribute highly, emphasizing the importance of the auditor having full control over TFM inputs. In contrast, one signal stands out as most informative: the chi-squared loss. It measures the occurrence of output frequencies of the correct classes or, for regression, correct binned output values. Therefore, if the model outputs the correct *label marginals* for a certain dataset, it will result in a lower chi-squared loss. Notably, the blind baseline relies on an entirely different set of features as seen by the low correlation between contribution rankings. This provides further evidence that true membership signals are distinct from mere distributional signals.

5.4 What Drives Inference Success?

We noted in Section 5.2 that the default nanoTabPFN and Real-TabPFN-2.5 models did not leak significant membership information. These models differ from SAP-RPT-1-OSS and TabDPT in several key ways: nanoTabPFN is much smaller, less performant at tabular tasks, and was trained on a very large number (690k) of fixed synthetic datasets rather than real data; Real-TabPFN-2.5 was pre-trained on non-fixed synthetic data and only fine-tuned on real data using an L2 penalty with respect to the initial TabPFN-2.5 weights (Purucker, 2026). We find it likely that these factors entail that nanoTabPFN and Real-TabPFN-2.5 do not encode a meaningful amount of membership information. The alternative is that our DI method lacked discriminatory power to extract the membership information retained in these models, however we find this less likely given that our DI method successfully extracted information from two TFMs reaching up to 0.988 AUC.

On the other hand, we observed greater levels of leakage from SAP-RPT-1-OSS compared to TabDPT. While both models were entirely trained on real data, they too have important differences. SAP-RPT-1-OSS was trained on a corpus of 2.18M tables, while TabDPT used 123 (Appendix D.1). To make up for this small corpus, TabDPT introduced an SSL-based task generation mechanism in which each time a dataset is used, the column labeled as y is randomly swapped with one of the columns in X, while other columns are randomly shuffled or dropped. These manipulations disrupt the normal relationship between X and y, potentially obfuscating membership information.

To isolate the reasons why some TFMs are more robust to membership leakage, we run ablations changing how nanoTabPFN is trained. Specifically, we: (a) reduce the number of pre-training datasets, so that each dataset has a relatively larger influence; (b) increase the model capacity, as a larger model could memorize more aspects of its training data; and (c) train on real instead of synthetic data.

Number of synthetic datasets. We decrease the number of datasets in the pre-training corpus, while keeping the total number of gradient updates fixed to increase the exposure of each individual dataset to the model. Decreasing the number of synthetic datasets from $50,000$ to 100 increases DI success only at the extreme: with 100 synthetic member datasets we observe an increased ROCAUC of 0.72. The synthetic datasets have relatively few rows (~ 50) and the loss plateaus quickly, with no overfitting occurring like in traditional learning methods, so only when extremely few datasets are used is there a measurable difference in membership leakage (see Table 9 in Appendix D.4).

Increased model capacity. We increase nanoTabPFN's capacity by adding more layers ($\{6, 8, 18\}$), increasing embedding dimensionality ($\{192, 384, 768\}$), adding attention heads ($\{6, 8, 12\}$), and increasing the MLP hidden dimension ($\{768, 1536, 3072\}$). The smallest hyperparameter settings match the default nanoTabPFN parameters, while the largest hyperparameters match SAP-RPT-1-OSS, the most leaky model (although other training details still differ). Overall, we vary model size between 3.72 M and 115.8 M parameters. Even with nanoTabPFN at the same size as SAP-RPT-1-OSS, DI is not successful, see Table 10 in Appendix D.4.

Real vs. synthetic data. We replace the fixed synthetic datasets drawn from a prior with real data from T4 (Gardner et al., 2024), matching a subset of the SAP-RPT-1-OSS dataset. When pre-trained with real data, ROCAUC of the DI reaches 0.811 with a single feature, and further increases to 0.92 when all available features are used, see Table 11 in Appendix D.4. This shows that characteristics of real data not captured by synthetic priors lead to fingerprintable behavior differences in what the model learns.

Pre-train vs. FT on real data. As a middle ground between real and prior-synthetic data, we use a generative model to create synthetic data based on the real datasets. In particular, we employ a Gaussian copula data synthesizer which fits an empirical cumulative distribution function (CDF) per column, scales the columns based on a standard normal, fits a multivariate Gaussian on the transformed columns, samples from that Gaussian, and inverts the CDFs to go back to the original column space. The result preserves marginals and pairwise correlations of the data. We fit one Gaussian copula for each real dataset and sample the same number of rows as each had for use in the DI method. Synthetic data generated from real datasets still yields a detectable membership signal (Table 11 in Appendix D.4), suggesting that the signal is not solely tied to individual training rows, but can also arise from dataset-level statistical structure such as marginals and correlations.

6 Conclusion

In this work, we introduced dataset inference for tabular foundation models as a black-box auditing method for determining whether a suspect tabular dataset was part of a model's pre-training corpus. By exploiting the in-context prediction interface of TFMs, we constructed dataset-level fingerprints from diverse input manipulations, output signals, and reductions, and showed that while individual classical membership signals are weak, a combination of TFM-specialized signals can reliably reveal dataset membership for TFMs trained on real tabular data, reaching near-perfect ROCAUC on SAP-RPT-1-OSS. Our ablations indicate that membership leakage is driven less by model capacity, and more by the use of real data in pre-training corpora. In other words, TFMs trained on fixed synthetic datasets are largely resistant to membership leakage except in the regime of extremely low numbers of distinct training datasets. Real data, and real-derived synthetic datasets, leave substantially stronger dataset-level traces that can be extracted by our auditing methods. These results suggest that dataset inference can serve as a practical tool for provenance and privacy auditing of deployed TFM APIs, helping data owners and model developers assess unauthorized data use and leakage risk.

AI USE STATEMENT

In this work, we used generative AI tools for coding assistance, implementation of methods, feedback on experimental methods, and editing of the paper in select parts. We have not used generative AI tools for writing the bulk of the paper, generating synthetic data sets, cleaning data sets, supporting data analysis, interpreting results, or proposing or refining hypotheses, and tasks related to theory, mathematical claims, proofs, or translation are not applicable to this work. Additionally, we used generative AI tools for producing code to plot experimental data, conducting literature searches, and providing feedback on the overall paper. We have reviewed all AI-assisted work, for example by manually curating papers from the literature search into our bibliography and discussions of related work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work has dual-use implications. While we frame dataset inference as a tool for provenance and privacy auditing, similar techniques could be misused to probe whether sensitive datasets contributed to a deployed API. We do not condone such use, but believe that studying these risks is necessary to help data owners and model providers assess leakage, improve transparency, and develop appropriate privacy safeguards.

REPRODUCIBILITY STATEMENT

We have made significant efforts to ensure that this work is reproducible. Our methodology is thoroughly explained in Sections 4 and 5. Additional experimental details have been included in Appendix D. Finally, we have included source code for our implementation at `anonymous.4open.science/r/tfm-di-D651`, which includes scripts for reproducing important parts of our work. This codebase has been released publicly in a non-anonymized version.

REFERENCES

AutoML. TFM-Playground, 2026. URL `https://github.com/automl/TFM-Playground/`. Accessed: 2026-05-05.

Vahid Balazadeh, Hamidreza Kamkari, Valentin Thomas, Benson Li, Junwei Ma, Jesse C. Cresswell, and Rahul G. Krishnan. CausalPFN: Amortized Causal Effect Estimation via In-Context Learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A Cookbook of Self-Supervised Learning . *arXiv:2304.12210*, 2023.

Stephen Bates, Emmanuel Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. Testing for outliers with conformal p-values. *The Annals of Statistics*, 51(1):149 – 178, 2023. doi: 10.1214/22-AOS2244.

Bernd Bischl, Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan van Rijn, and Joaquin Vanschoren. OpenML Benchmarking Suites. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Mohamed Bouadi, Pratinav Seth, Aditya Tanna, and Vinay Kumar Sankarapu. Orion-MSP: Multi-Scale Sparse Attention for Tabular In-Context Learning. *arXiv:2511.02818*, 2025.

Felix den Breejen, Sangmin Bae, Stephen Cha, and Se-Young Yun. Fine-tuned in-context learning transformers are excellent tabular data classifiers. *arXiv: 2405.13396*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Jessup Byun, Xiaofeng Lin, Joshua Ward, and Guang Cheng. Risk In Context: Benchmarking Privacy Leakage of Foundation Models in Synthetic Tabular Data Generation. *arXiv:2507.17066*, 2025.

Alycia N Carey, Karuna Bhaila, Kennedy Edemacu, and Xintao Wu. DP-TabICL: In-Context Learning with Differentially Private Tabular Data. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 1552–1557. IEEE, 2024.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy*, pp. 1897–1914, 2022. doi: 10.1109/SP46214.2022.9833649.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SigKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 1597–1607, 2020.

Debeshee Das, Jie Zhang, and Florian Tramèr. Blind Baselines Beat Membership Inference Attacks for Foundation Models. In *2025 IEEE Security and Privacy Workshops (SPW)*, pp. 118–125, 2025. doi: 10.1109/SPW67851.2025.00016.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? In *First Conference on Language Modeling*, 2024.

Jan Dubiński, Antoni Kowalczuk, Franziska Boenisch, and Adam Dziedzic. CDI: Copyrighted data identification in diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18674–18684, 2025.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, pp. 265–284. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-32732-5.

Adam Dziedzic, Haonan Duan, Muhammad Ahmad Kaleem, Nikita Dhawan, Jonas Guan, Yannis Cattan, Franziska Boenisch, and Nicolas Papernot. Dataset inference for self-supervised models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12058–12070, 2022.

Gus Eggert, Kevin Huo, Mike Biven, and Justin Waugh. TabLib: A dataset of 627M tables with context. *arXiv: 2310.07875*, 2023.

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 *Advances in Neural Information Processing Systems*, 2025.

Lijie Fan, Kaifeng Chen, Dilip Krishnan, Dina Katabi, Phillip Isola, and Yonglong Tian. Scaling Laws of Synthetic Images for Model Training ... for Now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7382–7392, June 2024.

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

Joshua Freeman, Chloe Rippe, Edoardo Debenedetti, and Maksym Andriushchenko. Exploring Memorization and Copyright Violation in Frontier LLMs: A Study of the New York Times v. OpenAI 2023 Lawsuit. *arXiv:2412.06370*, 2024.

Fundamental. Fundamental Nexus Model, 2025. URL `https://fundamental.tech/nexus`. Accessed: 2026-04-14.

Josh Gardner, Juan C. Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. In *Advances in Neural Information Processing Systems*, volume 37, pp. 45155–45205, 2024. doi: 10.52202/079017-1435.

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:2507.03971*, 2025.

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. *arXiv: 2511.08667*, 2025.

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.

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.

Rasa Hosseinzadeh, Alex Labach, Zexin Xue, Shuyi Han, Valentin Thomas, and Anthony L. Caterini. TabDPT-Turbo: Efficient In-Context Learning for Tabular Prediction. *arXiv:2608.01400*, 2026.

Sukriti Jaitly, Tanay Shah, Ashish Shugani, and Razik Singh Grewal. Towards better serialization of tabular data for few-shot classification with large language models. *arXiv:2312.12464*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Antoni Kowalczuk, Jan Dubiński, Franziska Boenisch, and Adam Dziedzic. Privacy attacks on image autoregressive models. In *Forty-second International Conference on Machine Learning*, 2025.

Kumo AI. Introducing kumorfm: A foundation model for in-context learning on relational data, 2025. URL `https://kumo.ai/company/news/kumo-relational-foundation-model/`. Accessed: 2026-04-14.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. doi: 10.1109/ICDM.2008.17.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.

Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. Consistent Individualized Feature Attribution for Tree Ensembles. *arXiv:1802.03888*, 2019.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling Tabular Foundation Models on Real Data. In *Advances in Neural Information Processing Systems*, 2025.

Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *International Conference on Learning Representations*, 2021.

Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. LLM Dataset Inference: Did you train on my dataset? In *Advances in Neural Information Processing Systems*, volume 37, pp. 124069–124092, 2024. doi: 10.52202/079017-3941.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 1631–1648, 2023.

Neuralk-AI. Neuralk: The prediction layer for large scale structured data., 2026. URL `https://www.neuralk.ai/`. Accessed: 2026-04-14.

Alexander Pfefferle, Johannes Hog, Lennart Purucker, and Frank Hutter. nanoTabPFN: A Lightweight and Educational Reimplementation of TabPFN. *arXiv:2511.03634*, 2025.

Prior Labs. TabPFN Scaling Mode: Large Data Models, 2025. URL `https://priorlabs.ai/tabpfn/large-data`. Accessed: 2026-04-14.

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

Lennart Purucker. Private communication, 2026.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv:2602.11139*, 2026.

Brendan Leigh Ross, Hamidreza Kamkari, Tongzi Wu, Rasa Hosseinzadeh, Zhaoyan Liu, George Stein, Jesse C. Cresswell, and Gabriel Loaiza-Ganem. A geometric framework for understanding memorization in generative models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Lorenzo Rossi, Bartłomiej Marek, Franziska Boenisch, and Adam Dziedzic. Natural identifiers for privacy and data audits in large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=doaAUf9Pi7`.

Nour Shaheen, Junwei Ma, Alex Labach, Frank Hutter, Valentin Thomas, and Anthony L. Caterini. Understanding the surprising generalization properties of tabular foundation models. *arXiv:2608.17957*, 2026.

Lloyd S. Shapley. A VALUE FOR n-PERSON GAMES. *Contributions to the Theory of Games*, 2 (28):307–317, 1953.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pp. 3–18, 2017. doi: 10.1109/SP.2017.41.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. Contexttab: A semantics-aware tabular in-context learner. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy auditing with one (1) training run. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49268–49280, 2023.

Christopher Stith, Medha Barath, Vahid Balazadeh, Jesse C Cresswell, and Rahul G Krishnan. Causal Foundation Models with Continuous Treatments. *arXiv:2605.15133*, 2026.

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, Haoxuan Lan, Jiaming Tian, Jing Yuan, Junbo Zhao, Junlin Zhou, Kaizhe Shou, Liangyu Zha, Lin Long, Liyao Li, Pengzuo Wu, Qi Zhang, Qingyi Huang, Saisai Yang, Tao Zhang, Wentao Ye, Wufang Zhu, Xiaomeng Hu, Xijun Gu, Xinjie Sun, Xiang Li, Yuhang Yang, and Zhiqing Xiao. TableGPT2: A Large Multimodal Model with Tabular Data Integration. *arXiv:2411.02059*, 2024.

Yi Sui, Tongzi Wu, Jesse Cresswell, Ga Wu, George Stein, Xiaoshi Huang, Xiaochen Zhang, and Maksims Volkovs. Self-supervised representation learning from random data projectors. In *International Conference on Learning Representations*, 2024.

The New York Times. The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work , 2023. URL `https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html`. Accessed: 2026-04-14.

Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2779–2792, 2022. ISBN 9781450394505. doi: 10.1145/3548606.3560554.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, New York, 2005. doi: 10.1007/b106715.

Ruiyu Wang, Zifeng Wang, and Jimeng Sun. UniPredict: Large Language Models are Universal Tabular Classifiers. *arXiv:2310.03266*, 2023.

Yongyao Wang, Ziqi Miao, Lu Yang, Haonan Jia, Wenting Yan, Chen Qian, and Lijun Li. TabSieve: Explicit In-Table Evidence Selection for Tabular Prediction. *arXiv:2602.11700*, 2026.

Rui Wen, Zheng Li, Michael Backes, and Yang Zhang. Membership inference attacks against in-context learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 3481–3495, 2024a. ISBN 9798400706363. doi: 10.1145/3658644.3690306.

Xumeng Wen, Han Zhang, Shun Zheng, Wei Xu, and Jiang Bian. From supervised to generative: A novel paradigm for tabular deep learning with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 3323–3333, 2024b. ISBN 9798400704901. doi: 10.1145/3637528.3671975.

Xumeng Wen, Shun Zheng, Zhen Xu, Yiming Sun, and Jiang Bian. Scalable in-context learning on tabular data via retrieval-augmented large language models. *arXiv:2502.03147*, 2025.

Santiago Zanella-Beguelin, Lukas Wutschitz, Shruti Tople, Ahmed Salem, Victor Rühle, Andrew Paverd, Mohammad Naseri, Boris Köpf, and Daniel Jones. Bayesian Estimation of Differential Privacy. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40624–40636. PMLR, 2023.

Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, Jifan Yu, Shu Zhao, Juanzi Li, and Jie Tang. TableLLM: Enabling Tabular Data Manipulation by LLMs in Real Office Usage Scenarios. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10315–10344. Association for Computational Linguistics, 2025a. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.538.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. Limix: Unleashing structured-data modeling capability for generalist intelligence. *arXiv:2509.03505*, 2025b.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*, 2025c.

Bihe Zhao, Pratyush Maini, Franziska Boenisch, and Adam Dziedzic. Unlocking post-hoc dataset inference with synthetic data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=a5Kgv47d2e.

A Limitations and Ethical Considerations

Our work has several limitations. First, successful dataset inference requires aggregating many weak features: individual signals such as loss or confidence provide only limited evidence of membership. Second, our method is mainly relevant for TFMs trained on fixed corpora of real or real-derived tabular datasets; many current TFMs, including the most performant ones, are trained on non-fixed synthetic data where there may be no meaningful finite training corpus to audit, and our experiments show little to no inference success on fixed synthetic data except in extreme low-diversity regimes. Third, our approach is model-agnostic and does not exploit architecture- or training-specific details, so stronger attacks may be possible when more information about the audited TFM is available.

B Extended Background on Tabular Foundation Models

In Section 2.1 we introduced the commonalities between existing TFMs, namely their ICL prediction mechanism. We also discussed the use of real and synthetic data for pre-training. Here, we give more detail on the diverse architectural details, data treatments, and training methods used for TFMs currently.

Internally, TFMs use an attention mechanism (Vaswani et al., 2017), where predictions on X^i_{qy} can attend to context datapoints, but not to other query datapoints. The actual implementation varies greatly, with some choices being to attend across rows (datapoints indexed by i) (Hollmann et al., 2023; Ma et al., 2025), across columns (elements of X^i), or some hybrid combination (Hollmann et al., 2025; Qu et al., 2025; Spinaci et al., 2025; Bouadi et al., 2025). These choices are further complicated due to the ways data can be encoded. Since tabular data is heterogeneous, where columns of X could be floats, ints, dates, or categorical, among others, TFMs require an encoder as a first step to create a uniform vector representation. Choosing what and how to encode is a major design decision, but the most common choice is to tokenize entire rows (Hollmann et al., 2023; Ma et al., 2025), as opposed to individual cells.

To enhance the variety of training tasks, the prediction target y can be randomly chosen out of all columns in the dataset, a form of self-supervised learning (Ma et al., 2025; Shaheen et al., 2026). This style of task creation has been successfully extended to foundation models for causal inference which also act on tabular data (Balazadeh et al., 2025; Stith et al., 2026).

We also note a separate line of research focused on tabular prediction using LLMs. Due to the general capabilities of LLMs to reason and perform ICL, efforts have been made to adapt them for tabular prediction tasks through ICL (Jaitly et al., 2023; Wen et al., 2024b), or by post-training LLMs on tabular prediction tasks (Wang et al., 2023; Gardner et al., 2024; Su et al., 2024; Zhang et al., 2025a; Wen et al., 2025; Wang et al., 2026). Because LLMs are pre-trained on vast quantities of non-tabular data and fundamentally process text tokens, we consider this setting to be distinct, and instead focus solely on TFMs.

Table 3: Output **signals** for dataset inference on TFMs. **C** - classification tasks; **R** - Regression tasks. $\hat{y}^i$ denotes the point prediction $\arg\max_y p(y)$ for query row i.

Signal Name	Type	Formula	Description
S1: Correct class confidence	**C**	$p(y) := p(y \mid X^i_{\text{qy}}; \mathcal{D}_{\text{ctx}})$	Likelihood assigned to the correct class
S2: Max confidence	**C**	$\max_y p(y)$	Highest likelihood assigned to any label
S3: Prediction margin	**C**	$p(y_{(1)}) - p(y_{(2)})$	Difference of highest two class likelihoods
S4: Accuracy	**C**	$\mathbb{1}(\hat{y}^i = y^i_{\text{qy}})$	Correctness of predicted class
S5: Cross entropy	**C**	$-\sum_{k=1}^{K} y^i_{\text{qy}} \log p_k(y)$	Cross entropy loss
S6: Absolute error	**R**	$\lvert\hat{y}^i - y^i_{\text{qy}}\rvert$	Absolute error loss
S7: Mean squared error	**R**	$(\hat{y}^i - y^i_{\text{qy}})^2$	Mean squared error loss
S8: R^2	**R**	$1 - \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$	Coefficient of Determination
S9: Prediction Spread	**R**	$\sigma(\hat{y})/\sigma(y)$	Relative standard deviation
S10: Chi-Squared†	**CR**	$\sum_{k=1}^{K} (p(y)_k - q(y)_k)^2 / q(y)_k$	Comparison of class frequencies to $q(y)$
S11: Entropy	**CR**	$-\sum_y p(y) \log p(y)$	Shannon entropy of class probabilities
S12: KL Divergence	**CR**	$\sum_y p(y) \log \frac{p(y)}{q(y)}$	KL Divergence to a distribution $q(y)$
S13: Reverse KL	**CR**	$\sum_y q(y) \log \frac{q(y)}{p(y)}$	KL Divergence from a distribution $q(y)$
S14: Wasserstein Distance	**CR**	$W_1(\hat{y}, y)$	Wasserstein-1 to reference output y

†: Computed in 30 bins for **R**.

C Dataset Inference Details

In this Appendix, we give complete details on how our dataset inference method is implemented for TFMs. As described in Section 4, from each dataset $\mathcal{D}$ we create a fingerprint vector $\bar{S}(\mathcal{D})$, where each element is a unique (manipulation, signal, reduction)-triple. These elements are used to train a model to perform anomaly detection of member datasets, or predict binary dataset member labels.

Signals are observable outputs from a TFM. In Table 3 we list the 14 signals constructed from the output of the TFM, which is the predictive distribution over labels $p(y^i_{\text{qy}} \mid X^i_{\text{qy}}; \mathcal{D}_{\text{ctx}})$. Signals include common measures for determining how well f_θ performs at the predictive task, like per-datapoint accuracy/error, predictive confidence, commonly used losses, and distributional measures like entropy and KL divergence. The KL and reverse-KL signals compare the output distribution $p(y^i_{\text{qy}} \mid X^i_{\text{qy}}; \mathcal{D}_{\text{ctx}})$ to a reference distribution $q(y)$. For concreteness, we use the distribution of true labels y_{qy}, although other options could be considered.

Manipulations are operations that can be performed on the inputs of f_θ, namely X_{qy} and $\mathcal{D}_{\text{ctx}}$. The auditor is free to construct these inputs from the dataset $\mathcal{D}$ (either $\mathcal{D}_{\text{sus}}$ or $\mathcal{D}_{\text{known}}$) as they wish, which provides strong leverage for interrogating the TFM using only a black-box API. Manipulations are designed so that the predictive distribution will behave differently depending on if $\mathcal{D}$ is a member or non-member. In Table 4 we list the 19 different types of input manipulation we perform. Truth Serum is inspired by Tramèr et al. (2022). Brainwash is inspired by Wen et al. (2024a).

Finally, reductions first vary a parameter of the manipulation, observe a sequence of corresponding signals, and aggregate that sequence to create a feature, one of the elements of $\bar{S}(\mathcal{D})$ used for dataset inference downstream. Some reductions are designed to match properties of certain manipulations, like their intensity as applicable. In Table 5 we detail the 11 types of reductions that we applied.

Table 4: Input **manipulations** for dataset inference on TFMs and their parameters. **C** - classification tasks; **R** - Regression tasks

Manipulation Name	Type	Description	Parameters
M1: No manipulation	**CR**	Only split rows of $\mathcal{D}_{\text{sus}}$ into disjoint sets $(X_{\text{ctx}}, y_{\text{ctx}})$ and $(X_{\text{qy}}, y_{\text{qy}})$	Splitting fraction (applies to all below)
M2: Query = Context	**CR**	Repeat the context inputs X_{ctx} as X_{qy}	-
M3: Query only	**CR**	Provide only X_{qy}, with $\mathcal{D}_{\text{ctx}} = \varnothing$	-
M4: Row sampling	**CR**	Subsample the rows of $\mathcal{D}_{\text{ctx}}$	Fraction of rows sampled
M5: Row shuffling	**CR**	Randomly permute the rows of $\mathcal{D}_{\text{ctx}}$	Permutation
M6: Feature sampling	**CR**	Subsample the columns of X_{ctx}	Fraction of columns sampled
M7: Feature shuffling	**CR**	Randomly permute the columns of X_{ctx}	Permutation
M8: Feature noising	**CR**	Randomly perturb the columns of $\tilde{X}_{\text{ctx}}=X_{\text{ctx}}+\varepsilon$	Noise type (e.g. Gaussian); Noise level
M9: Feature transform	**CR**	Transform a column ($\mathcal{D}_{\text{ctx}}$ and $\mathcal{D}_{\text{qy}}$) with a fixed function	Column; Function (e.g. $\log$)
M10: Task swap	**CR**	Redefine the label y as one of the columns of X	Column
M11: Class permutation	**C**	Re-encode classes as $y' = \pi(y)$ with a permutation π	Permutation
M12: Label corruption	**CR**	Randomly select context rows and change y_{ctx} to an incorrect label	Fraction corrupted
M13: Constant labels	**CR**	Set all context labels to a fixed value $y_{\text{ctx}} = a$	Value
M14: Row duplication	**CR**	Randomly duplicate some rows of $\mathcal{D}_{\text{ctx}}$	Fraction duplicated
M15: Truth serum	**CR**	Randomly duplicate some rows of $\mathcal{D}_{\text{ctx}}$, but with corrupted y_{ctx}	Fraction duplicated
M16: Query leakage	**CR**	Randomly copy some rows of $\mathcal{D}_{\text{qy}}$ into $\mathcal{D}_{\text{ctx}}$	Fraction copied
M17: Brainwash	**CR**	Randomly copy some rows of $\mathcal{D}_{\text{qy}}$ into $\mathcal{D}_{\text{ctx}}$, but with corrupted y_{ctx}	Fraction copied
M18: Nearest neighbours	**CR**	Compute NN within $\mathcal{D}$, and split them across $\mathcal{D}_{\text{ctx}}$ and $\mathcal{D}_{\text{qy}}$	Number of neighbours
M19: Retrieval	**CR**	Retrieve NN of X_{qy} from $\mathcal{D}$ as X_{ctx}	Number of neighbours

D Additional Experimental Details and Results

D.1 Tabular Foundation Models and Datasets

Our DI experiments used four TFMs: TabDPT (Ma et al., 2025) (Apache 2.0 license), SAP-RPT-1-OSS (Spinaci et al., 2025) (Apache 2.0 license), nanoTabPFN (Pfefferle et al., 2025) (Apache 2.0 license via TFM-Playground (AutoML, 2026)), and Real-TabPFN-2.5 (Grinsztajn et al., 2025) (TABPFN-2.5 License v1.1, which allows our research uses). Each of these models was trained on a different set of datasets. We describe the member datasets for each TFM below.

TabDPT (Ma et al., 2025) was trained on a collection of 123 datasets from OpenML (Vanschoren et al., 2014) (CC-BY license). For held-out non-members we use CC18 (Bischl et al., 2021) and CTR23 (Fischer et al., 2023), also from OpenML, which TabDPT used as a test set, not for pre-training.

SAP-RPT-1-OSS (Spinaci et al., 2025) was trained on a subset of the T4 dataset (Gardner et al., 2024) (available under a custom license). This subset first excluded any dataset with fewer than 150 rows, leaving 2.18M tables available from pre-training. As this amount of tables is excessively large for testing DI, we select tables in chunk-002 from T4 (with ¿150 rows) as the members. The datasets with 150 or fewer rows are known non-members. Given this setup, we note that DI is trivial if row count is used as a predictive feature. Hence, when performing DI we ensure any feature related to row count or table size is excluded, and also use blind baselines to account for the performance level that can be obtained from distributional differences alone.

Table 5: **Reductions** vary the parameters of a manipulation to create a sequence of signals that are then summarised into a scalar entry of $\bar{S}(\mathcal{D})$. The applicable family depends on the structure of the manipulation's parameters: *ordered* along an intensity axis, *unordered*, or based on prediction *stability*.

Type	Reduction	Formula	Captures
Default	**R1:** No Reduction	Signal is used directly as feature	Predictive power of signal directly
Ordered	**R2:** Slope	Regression coef. on (log-spaced) intensity	Rate of signal change as manipulation intensity increases
	R3: AUC	Normalised trapezoidal integral	Average signal value across manipulation intensity range
	R4: Gain	Final value minus initial value	Total signal change between highest and lowest intensity
	R5: Early	Value at 25% intensity, normalised by max	Signal response at low-intensity manipulation
	R6: Late gain	Final value minus second-to-last	End-of-curve flattening or runaway signal
Unordered	**R7:** Mean	Mean over parameter range	Typical signal value
	R8: St. dev.	Sample st. deviation over parameter range	Signal spread
	R9: Range	Max minus min over parameter range	Signal range
Stability	**R10:** Consistency	Pairwise signal agreement	Cross-parameter signal stability
	R11: Rank corr	Mean pairwise Spearman correlation across per-row signals	Cross-parameter ranking stability

nanoTabPFN (Pfefferle et al., 2025) and its pre-training datasets are available in the TFM-Playground repository (AutoML, 2026) (Apache 2.0 license). Using default settings, this model is trained on synthetic data from a prior, but unlike other TFMs (Grinsztajn et al., 2025) which generate new synthetic data for each batch, nanoTabPFN trains on a fixed set of synthetic tables that are recycled each epoch. The fixed set consists of 690k generated tables as members, and another 690k as non-members.

Real-TabPFN-2.5 (Grinsztajn et al., 2025) starts from the pre-trained TabPFN-2.5 model, which was trained entirely on synthetic datasets drawn fresh from a prior each batch, and fine-tunes on a set of 43 real tables from OpenML and Kaggle. We use the 20 datasets from OpenML as training members, and include 51 datasets from TabArena (Erickson et al., 2025) (originally from OpenML) as non-members, since Real-TabPFN-2.5 used them only for testing. We note a few relevant points about how Real-TabPFN-2.5 was fine-tuned which were not disclosed in the original paper: fine-tuning lasted 25,000 gradient steps and did not use gradient aggregation; the loss included an L2 penalty with respect to the starting weights of TabPFN-2.5 (Purucker, 2026).

In comparison, many other TFMs are not trained on a fixed set of datasets, but instead draw new synthetic datasets from a prior for each batch during pre-training. Such synthetic datasets generally do not come with privacy or provenance concerns, so we do not consider DI on these TFMs. However, this fact also makes them suitable as blind baselines. In particular, we use the highly-performant **TabPFN-2.5** (Grinsztajn et al., 2025), with model weights released by the authors under the custom TABPFN-2.5 License v1.1.

D.2 Single Feature Dataset Inference on TabDPT

Figure 7 reproduces Figure 2 on TabDPT (n=224 datasets), showing the strength of individual features for dataset inference. Single-signal AUCs are again close to chance (0.61 for both average loss and average correct confidence).

D.3 Feature Contributions

Table 6 shows the top 10 features in $\bar{S}(\mathcal{D})$ for the classifier g_φ for internal auditing, using SAP-RPT-1-OSS as the target TFM, as discussed in Section 5.3. In Table 7 we show the same analysis using TabDPT as the target TFM, with all other settings unchanged. Like for Table 6, dataset inference

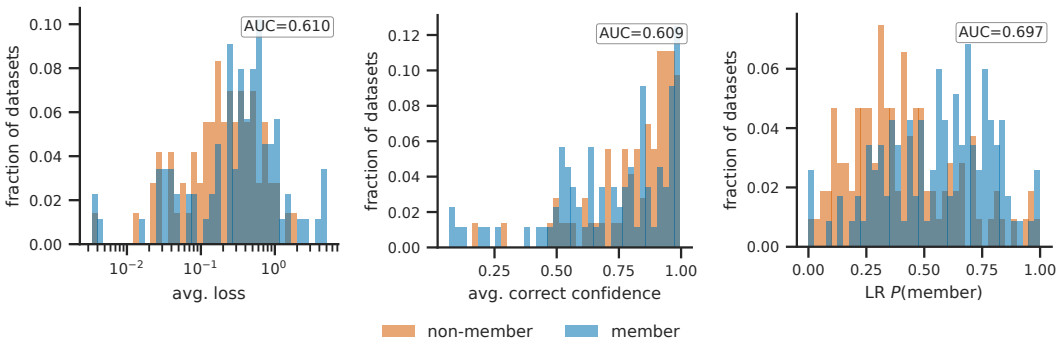


Figure 7: **Single- and multi-feature dataset inference on TabDPT.** The *per-dataset* average loss (Left) and correct-class confidence (Middle) computed for member and non-member datasets of TabDPT (Ma et al., 2025) (n=224 datasets) do not clearly separate member and non-member datasets, as in Figure 2. (Right) The internal auditing method used as in Figure 5 with many TFM-specific features achieves higher ROCAUC.

Table 6: **Fingerprint Feature Contributions.** Mean absolute SHAP value of the top 10 features from $\bar{S}(\mathcal{D})$, computed for the SAP-RPT-1-OSS target TFM and TabPFN-2.5 blind TFM, with the XGB classifier. Larger values indicate that the feature contributes more to membership classification. Identifiers for signals, manipulations, and reductions reference Tables 3, 4, and 5 in Appendix C.

Fingerprint Feature	Manipulation	Signal	Reduction	Target	Blind
Retrieval_ChiSquared_Mean	M19	S10	R7	0.6409	0.0009
TaskSwap_ChiSquared_Mean	M10	S10	R7	0.4718	0.0156
NN_ChiSquared_Default	M18	S10	R1	0.3287	0.0033
RowSampling_ChiSquared_Gain	M4	S10	R4	0.3245	0.0821
ClassPermutation_ChiSquared_Range	M11	S10	R9	0.3154	0.0652
RowSampling_ChiSquared_Slope	M4	S10	R2	0.3104	0.0449
ClassPermutation_ChiSquared_Mean	M11	S10	R7	0.2910	0.1389
Retrieval_KLD_Mean	M19	S12	R7	0.2735	0.0039
LabelCorruption_ChiSquared_Gain	M12	S10	R4	0.2480	0.0044
Brainwash_ReverseKLD_Early	M17	S13	R5	0.2283	0.0021

on the blind TFM relies on an entirely different set of features than for the target TFM. Compared to Table 6, inference on TabDPT has heaviest reliance on a different set of manipulations, signals, and reductions. However, the auditor need not know the identity of the TFM they are querying for dataset inference to work; our tests in Table 2 are model agnostic and generate a large number of features in $\bar{S}(\mathcal{D})$, relying on the classifier g_φ to automatically choose which features are predictive. Strong classifiers like XGBoost are fully capable of filtering out noisy features when appropriate regularization is used.

D.4 Ablations

We perform ablations on nanoTabPFN to determine under which conditions it leaks membership information under our DI method. We use nanoTabPFN as it is modularly designed and relatively lightweight to train, finishing a training run in a few minutes on a single H100 as Table 8 shows. We ablate over architecture size: we interpolate model size between the default size and the $32\times$ larger SAP-RPT-1-OSS configuration. First we reduce the number of datasets. Table 9 shows that only with very few datasets (100) do we see a significant increase in AUC. Table 10 shows that increased model size does not lead to membership leakage. Finally, we train nanoTabPFN on real data, and Table 11 shows this allows differentiating member and non-member data. We also derive synthetic data from the real data using a Gaussian Copula generator. This barely reduces AUC from 0.92 for real data to 0.90 for the real data-derived synthetic data.

Table 7: **Fingerprint Feature Contributions.** Mean absolute SHAP value of the top 10 features from $\bar{S}(\mathcal{D})$, computed for the TabDPT target TFM and TabPFN-2.5 blind TFM, with the XGB classifier. Larger values indicate that the feature contributes more to membership classification. Identifiers for signals, manipulations, and reductions reference Tables 3, 4, and 5 in Appendix C.

Fingerprint Feature	Manipulation	Signal	Reduction	Target	Blind
FeatureSample_KLD_AUC	M6	S12	R3	0.4358	0.0000
SplitFraction_CELoss_Early	M1	S5	R5	0.4125	0.0000
RowSample_KLD_Early	M4	S12	R5	0.4018	0.0000
SplitFraction_Confidence_Slope	M1	S1	R2	0.3576	0.0000
FeatureNoise_ChiSquared_AUC	M8	S10	R3	0.1962	0.0225
SplitFraction_CELoss_Slope	M1	S5	R2	0.1838	0.0000
FeatureSample_Wasserstein_LateGain	M6	S14	R6	0.1810	0.0000
Brainwash_CELoss_LateGain	M17	S5	R6	0.1795	0.0000
Brainwash_ChiSquared_AUC	M17	S10	R3	0.1766	0.0102
QueryLeak_ReverseKLD_Early	M16	S13	R5	0.1723	0.0133

Table 8: **NanoTabPFN training cost.** Single-GPU (NVIDIA H100) training time for the default checkpoint and all ablation variants used in the paper. Architecture is given as L / E / H / M (layers, embedding dim, attention heads, MLP hidden dim). All variants are trained for 80 epochs of 25 steps each at batch size 50 with AdamW ($\eta = 10^{-4}$).

Variant	L/E/H/M	Prior tables	Time/epoch	Compute
`default`	6/192/6/768	50 000	0.55 s	44 s
`data25k`	6/192/6/768	25 000	0.54 s	43 s
`data10k`	6/192/6/768	10 000	0.52 s	42 s
`data5k`	6/192/6/768	5 000	0.52 s	42 s
`arch_medium`	8/384/8/1536	50 000	1.34 s	1.8 min
`arch_large`	12/768/12/3072	50 000	5.35 s	7.1 min

Table 9: **NanoTabPFN training corpus size ablation.** Dataset inference results for nanoTabPFN models trained on different sized corpora of prior-synthetic tables. For each row, the number of training steps and batch size is held constant, meaning that when the corpus is smaller, the model is exposed to each dataset more times. DI is run on a held-out synthetic eval set with equal numbers of member and non-member tables. Top-1 is the best single-features AUC; multivariate is logistic regression on the top-50 features.

$\lvert\mathcal{D}_{\text{train}}\rvert$	eval n	top-1 AUC	multivariate AUC
50,000	100,000	0.505	0.501 ± 0.003
25,000	50,000	0.511	0.506 ± 0.004
10,000	20,000	0.516	0.514 ± 0.005
5,000	10,000	0.531	0.523 ± 0.009
2,000	4,000	0.530	0.516 ± 0.019
1,000	2,000	0.536	0.541 ± 0.037
500	1,000	0.570	0.563 ± 0.037
100	200	0.632	0.720 ± 0.061

Table 10: **NanoTabPFN architecture size ablation.** All variants are trained on the same 50 000 synthetic prior tables; only the parameter count differs. "L/E/H/M" denotes layers, embedding dimension, attention heads, and feed-forward dimension. DI evaluation is run on the same held-out 50 000 member / 50 000 non-member split as in Table 9.

Variant	L/E/H/M	params	top-1 AUC	multivariate AUC
Small (default)	6/192/6/768	3.7M	0.505	0.501 ± 0.003
Medium	8/384/8/1536	19.5M	0.506	0.505 ± 0.003
Large	12/768/12/3072	115.8M	0.505	0.503 ± 0.004

Table 11: **NanoTabPFN training-corpus ablation.** DI results on two nanoTabPFN models with the default architecture, trained on the real T4 corpus and on a Gaussian-copula synthetic copy of T4.

Training corpus	members	non-mem.	top-1 ROCAUC	multivariate ROCAUC
Real T4	9,321	3,725	0.811	0.923 ± 0.006
Synthetic, Gaussian copula	8,867	2,437	0.871	0.897 ± 0.008