
GOTabPFN: From Feature Ordering to Compact Tokenization for Tabular Foundation Models on High-Dimensional Data

Al Zadid Sultan Bin Habib¹ Md Younus Ahamed¹ Prashnna Kumar Gyawali¹ Gianfranco Doretto²
Donald A. Adjeroh¹

Abstract

We investigate how to make small tabular foundation models effective for High-Dimensional, Low-Sample Size (HDLSS) tabular prediction without retraining large backbones. We introduce Graph-guided Ordering with Local Refinement (GO-LR), show its equivalence to weighted Minimum Linear Arrangement, and interpret the practical solver as a TSP-path-style surrogate. We propose GOTabPFN, which builds on GO-LR, and a Neuro-Inspired Subunit Compression (NSC) unit to pool locally adjacent ordered features into meta-features, yielding a compact representation that makes TabPFN-style prediction practical in HDLSS regimes. Across tabular benchmarks, GOTabPFN improves stability and accuracy under tight token budgets.

1. Introduction

High-Dimensional, Low-Sample Size (HDLSS) tabular prediction remains a challenge: when $m \gg n$ (with m =no. of features, n =no. of samples), both learning and representation become costly. Tabular foundation models such as TabPFN and its variants are strong general-purpose baselines, but popular versions (e.g., TabPFN-2.5 (Grinsztajn et al., 2025)) are designed and benchmarked for inputs with up to roughly 2,000 features, leaving many HDLSS domains (e.g., gene expression with $m \gg 2,000$) outside their intended operating range without prior feature selection or compression. This motivates representation strategies that reduce dimensionality under tight sample budgets while

preserving predictive structure, so TabPFN-style learners remain effective in truly high-dimensional regimes.

Permutation learning seeks an ordering of a finite set that improves a downstream objective, typically via differentiable relaxations that approximate discrete permutations in end-to-end neural training (Barthel et al., 2025; Jurewicz & Derczynski, 2022). For tabular data, the lack of inherent spatial or temporal structure weakens inductive bias relative to vision or language, especially in HDLSS settings. Although tree-based methods remain strong baselines, learning cross-feature dependencies without overfitting is difficult; even simple models (e.g., MLPs or Lasso) can outperform advanced tabular approaches in $n \ll m$ regimes (ProtoGate (Jiang et al., 2024)). This suggests that feature selection alone is often insufficient; we also need a learnable feature ordering that organizes correlated features into neighborhoods amenable to structured compression. We therefore formulate the Column Permutation Problem (CPP) (Fogel et al., 2013; Lima et al., 2024; Tegze & Vlach, 1986; Liiv, 2010; Behrisch et al., 2016): learn a data-driven column order that reduces redundancy, reveals long-range dependencies, and induces a useful sequential structure for downstream modules. In practice, CPP can be tackled via attention-based pointer mechanisms and graph-aware variants that generate permutations while encoding relational structure (Vinyals et al., 2015; Yang et al., 2022b; Veličković et al., 2020).

Feature ordering has a long history in pattern recognition and is central to Incremental Attribute Learning (IAL), where features arrive sequentially and must be ranked before training (Wang & Guan, 2013). Unlike set-based models that assume order invariance (Zaheer et al., 2017), column order can expose redundancy and shape how models capture dependencies; even simple Fisher/correlation/entropy rankings reduce interference and error over unordered baselines (Wang et al., 2015c;b), motivating learned, task-aware ordering (Wang et al., 2015a; 2014). In deep tabular learning, Mambular (Thielmann et al., 2024) underscored the impact of ordering and Habib et al. (2024; 2026b) introduced explicit ordering algorithms in TabSeq and DynaTab, respectively. Other related efforts show brittleness to col-

¹ Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA

² Scientific Computing and Imaging Institute & Department of Biomedical Informatics, The University of Utah, Salt Lake City, UT 84112, USA. Correspondence to: Al Zadid Sultan Bin Habib <ah00069@mix.wvu.edu>.

umn permutations, prompting permutation-invariant architectures (Eremeev et al., 2025; Brahmavar et al., 2025) and TabICL (Jingang et al., 2025), which ensembles across permutations. Beyond supervised prediction, COPER (Eisenberg et al., 2025) uses a permutation-based correlation objective for multi-view (image-table) clustering, and ROTATOR-LLM (Wang et al., 2025) studies feature ordering for LLM-based tabular inference.

While ordering can expose local structure, HDLSS tables introduce a second bottleneck: even a “good” permutation still leaves m raw features to process, which is prohibitive when $m \gg n$. To make TabPFN-style predictors practical in this regime without changing the backbone, we introduce Neuro-Inspired Subunit Compression (NSC), motivated by subunit-style integration in cortical dendrites (Poirazi et al., 2003; Schiller et al., 2000; Major et al., 2013; Kastellakis et al., 2015; Kirchner & Gjorgjieva, 2021; Ujfalussy & Makara, 2020; Wu et al., 2018). NSC groups adjacent features along the GO-LR (Graph-guided Ordering with Local Refinement) axis into contiguous subunits and pools each into a meta-feature, reducing dimensionality from m to M ($M \ll m$), with M tied to intrinsic dimension estimates from the covariance spectrum (Roy & Vetterli, 2007; Halko et al., 2011; Levina & Bickel, 2004). Naïve compression often produces latent components without a stable coordinate system, yielding run and subsample-dependent representations that are not effective for TabPFN-style models, which assume a fixed, consistently parameterized input space (Hollmann et al., 2023; 2025). We therefore design a structure-constrained compression interface that yields reproducible latent features within the feature budgets targeted by recent TabPFN variants (Grinsztajn et al., 2025; Liu & Ye, 2025; Kolberg et al., 2025).

Our contributions:

- We cast feature ordering as a combinatorial optimization problem, prove its NP-hardness, and propose MinLA-grounded ordering via GO-LR.
- We introduce scalable HDLSS compression via NSC, a neuro-inspired subunit-style pooling that is controlled by intrinsic-dimension estimates.
- Building on the above, we propose GOTabPFN for analyzing HDLSS tabular data. Across HDLSS benchmarks, GOTabPFN improves accuracy and stability under tight feature budgets in high-dimensions.

2. Related Work

In Appendix A, we provide more details on related work, including on tabular foundation models, the TabPFN family, HDLSS-specific models, and LLM-based tabular models.

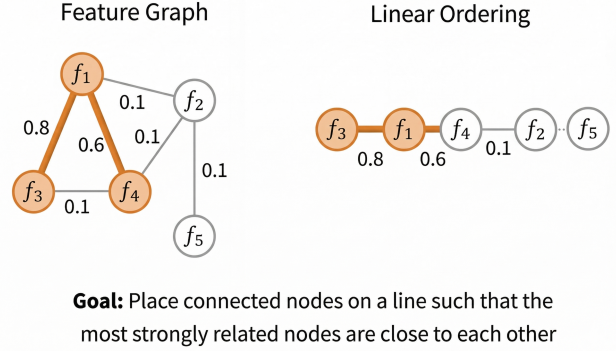


Figure 1. **Graph-based feature ordering.** GO-LR linearizes a weighted feature graph to keep related features nearby for local segmentation and compression. It uses NNPath for local initialization, then refines the order with a global MinLA-style objective over pairwise placements. See Appendix T for more clarifications.

Existing approaches often struggle in HDLSS settings with $m \gg n$, since they either assume moderate feature counts or rely primarily on feature selection and task-specific tuning to cope with very high dimensionality. GOTabPFN bridges this gap by coupling MinLA-grounded ordering (GO-LR) with subunit-style compression (NSC), yielding stable, low-dimensional representations that enable TabPFN-style predictors to operate effectively in truly high-dimensional regimes without modifying the TabPFN backbone.

3. Methodology

Problem formulation. Let $X \in \mathbb{R}^{n \times m}$ be the input matrix with n samples and m features. We define the sample partition $\{I_c\}_{c=1}^k$ obtained by clustering the samples, and the cluster-restricted matrices in Eq. 1.

$$X^{(c)} = X[I_c, :] \in \mathbb{R}^{n_c \times m}, \quad n_c = |I_c| \quad (1)$$

For each $X^{(c)}$, we construct the corresponding cluster-wise feature graph $G_c = (V, E, w^{(c)})$, where $V = \{1, \dots, m\}$ is the shared feature set and $w_{ij}^{(c)}$ measures feature dissimilarity within cluster c . The local permutation π_c is obtained by minimizing a MinLA-style dispersion objective on G_c , and the final global permutation Π^* is obtained by aggregating local ranks across clusters. All permutations are over features, and GO-LR outputs a single global feature ordering Π^* , not separate feature spaces that must later be rearranged across clusters. Π^* is then used for NSC segmentation and compression. Figs. 1, 2, and 3 summarize the pipeline: GO-LR linearizes feature graphs, NSC segments and compresses contiguous ordered neighborhoods into meta-features, and the resulting tokens are passed to a frozen TabPFN-2.5 head within GOTabPFN.

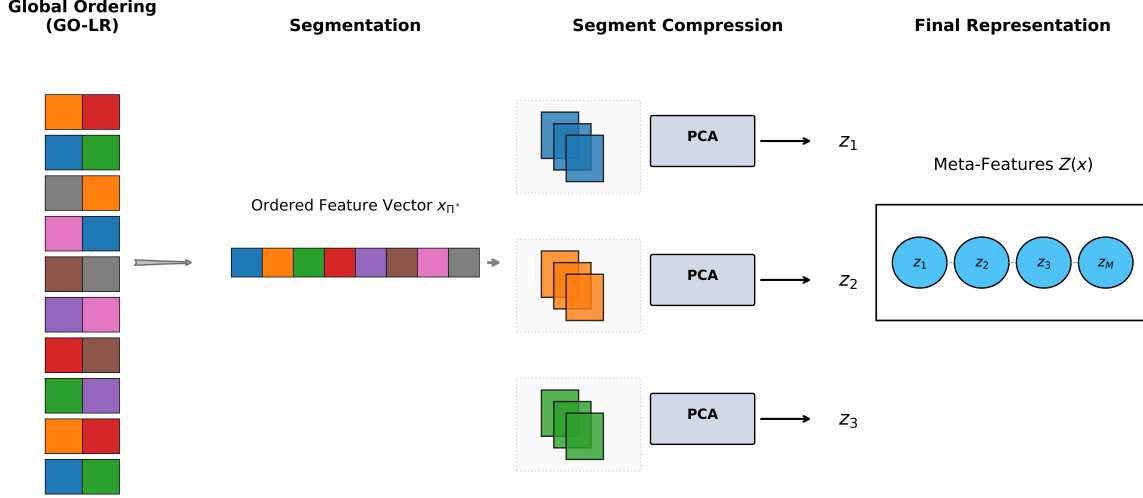


Figure 2. **Meta-feature construction.** GO-LR first orders features globally; NSC then segments the ordered axis into contiguous neighborhoods and compresses each segment by PCA into a scalar meta-feature. The final vector $Z(x) = (z_1, \dots, z_M)$ is passed to the frozen TabPFN-2.5 head. See Appendix T for additional clarifications.

3.1. Feature Ordering as a Combinatorial Optimization Problem.

Problem Setup: Feature Ordering by Graph Dispersion. In this section, we show that GO-LR-based feature ordering corresponds to the Minimum Linear Arrangement (MinLA) problem, is NP-hard, and strictly generalizes TSP-path. Here, TSP-path refers to the Traveling Salesman (TSP) path problem: given a complete weighted graph, find a Hamiltonian path σ that minimizes $\text{PathCost}(\sigma) = \sum_{t=1}^{m-1} d_{\sigma_t, \sigma_{t+1}}$. We further show that the practical GO-LR algorithm provides a TSP-path style initialization, which is then locally refined under the dispersion objective. We connect GO-LR-based feature ordering to classical combinatorial optimization, including linear arrangement and seriation problems (Díaz et al., 2002; Seminaroti, 2016; Fogel et al., 2013). It is MinLA (NP-hard), admits a TSP-path heuristic implementation, and strictly generalizes TSP-path via an exact embedding.

Theorem 3.1 (Theoretical Characterization of GO-LR). *GO-LR-based feature ordering corresponds to a weighted MinLA problem, is NP-hard in the number of features, and strictly generalizes the TSP-path problem.*

Proof sketch. Theorem follows from Lemma 3.8, Lemma 3.9, and Theorem 3.12 as described below. $\square$

Moreover, the practical GO-LR algorithm uses a nearest-neighbor TSP-path heuristic for initialization and then applies a local refinement step (direction selection and adjacent swaps) that monotonically decreases the MinLA dispersion objective. The remainder of this section establishes this characterization through a sequence of equivalence and reduction results.

Definition 3.2 (Local Feature Graph). Given cluster c with samples $X^{(c)} \in \mathbb{R}^{n_c \times m}$, we define a weighted feature graph $G_c = (V, E, w)$ where $V = \{1, \dots, m\}$ indexes features and $w_{ij} \geq 0$ quantifies dissimilarity between features i and j computed from $X^{(c)}$ (e.g., $1 - |\text{corr}|$; see App. T, JS(Jensen-Shannon divergence)/KL(Kullback-Leibler divergence), cosine/Euclidean/Manhattan). We write $(i, j) \in E$ whenever a pair is included (typically $E = V \times V \setminus \{(i, i)\}$ for a complete graph, or a sparse neighborhood graph).

Definition 3.3 (Dispersion Objective (GO-LR Local Ordering)). A local ordering is a bijection $\pi : V \rightarrow \{0, \dots, m-1\}$ assigning each feature to a position. The cluster-wise dispersion of π is in Eq. 2. The GO-LR local ordering problem is to compute the local order with minimum dispersion, as shown in Eq. 3.

$$D_{G_c}(\pi) = \sum_{(i,j) \in E} w_{ij} |\pi(i) - \pi(j)| \quad (2)$$

$$\pi_c^* \in \arg \min_{\pi} D_{G_c}(\pi) \quad (3)$$

Definition 3.4 (GO-LR Local Refinement Operator). Let $\pi^{(0)} \leftarrow \text{NNPath}(G_c)$ be the nearest-neighbor initialization (a permutation of V). We define $\text{rev}(\pi)$ as the reversed permutation and let $\mathcal{N}(\pi)$ denote the set of permutations obtained by one adjacent transposition (Eq. 4). GO-LR first performs direction selection (Eq. 5) and then applies P passes of adjacent-swap descent (Eq. 6), with early stopping if $\pi^{(p+1)} = \pi^{(p)}$. The refined local ordering is $\pi_c \leftarrow \pi^{(P)}$. In Eq. 4, $\text{swap}_t(\pi)$ is the adjacent-transposition operator that returns the permutation obtained by swapping the entries at positions t and $t+1$ in π .

$$\mathcal{N}(\pi) = \{\text{swap}_t(\pi) : t = 0, \dots, m-2\} \quad (4)$$

$$\pi^{(0)} \leftarrow \arg \min \left\{ D_{G_c}(\pi^{(0)}), D_{G_c}(\text{rev}(\pi^{(0)})) \right\} \quad (5)$$

$$\pi^{(p+1)} \leftarrow \text{SweepRefine}(\pi^{(p)}, G_c), \quad p = 0, \dots, P-1 \quad (6)$$

SweepRefine. Initialize $\tilde{\pi} \leftarrow \pi^{(p)}$ and scan $t = 0, \dots, m-2$. Compute swap gain $\Delta_t := D_{G_c}(\text{swap}_t(\tilde{\pi})) - D_{G_c}(\tilde{\pi})$ via an incremental update (no full recomputation). If $\Delta_t < 0$, set $\tilde{\pi} \leftarrow \text{swap}_t(\tilde{\pi})$ immediately. Return $\pi^{(p+1)} \leftarrow \tilde{\pi}$ and early-stop if a full sweep makes no changes.

Remark 3.5. Each update in Eqs. (5)–(6) is chosen to not increase D_{G_c} ; hence GO-LR yields $D_{G_c}(\pi_c) \leq D_{G_c}(\pi^{(0)})$.

Remark 3.6. Eq. (2) (together with Eq. (3)) defines the dispersion objective used in GO-LR. It is a standard linear arrangement / seriation-type criterion (Díaz et al., 2002; Fogel et al., 2013): pairs with larger weights w_{ij} are penalized more when placed far apart, hence the ordering tends to place high-weight pairs closer in their index.

Complexity: Equivalence to Minimum Linear Arrangement. The MinLA problem is a classical graph layout problem, and has been well studied (Shiloach, 1979).

Definition 3.7 (Weighted Minimum Linear Arrangement (MinLA)). Given a weighted graph $G = (V, E, w)$, the weighted MinLA problem is

$$\min_{\pi: V \rightarrow \{0, \dots, |V|-1\} \text{ bijective}} \sum_{(i,j) \in E} w_{ij} |\pi(i) - \pi(j)| \quad (7)$$

Lemma 3.8 (GO-LR Local Ordering is MinLA). *For each cluster c , the GO-LR local ordering objective in Eq. (3) is exactly the weighted MinLA objective on G_c .*

Proof sketch. Both problems optimize over bijections $\pi : V \rightarrow \{0, \dots, m-1\}$ and share the identical objective $\sum_{(i,j) \in E} w_{ij} |\pi(i) - \pi(j)|$ (Eq. (2) and Eq. (7)). Hence they are the same optimization problem. $\square$

Lemma 3.9 (NP-hardness). *The GO-LR local feature ordering problem (Eq. (3)) is NP-hard in m .*

Proof sketch. Weighted MinLA is NP-hard (Garey et al., 1976); since GO-LR local ordering is exactly MinLA, it is NP-hard. $\square$

An Exact Equivalence Case: TSP-path as a Special Case of Feature Ordering.

Definition 3.10 (TSP-path Objective on a Complete Graph). Given a complete weighted graph $\mathcal{K} = (V, \binom{V}{2}, d)$ with

edge weights $d_{ij} \geq 0$, define the path cost of a permutation $\sigma = (\sigma_1, \dots, \sigma_m)$ by

$$\text{PathCost}(\sigma) = \sum_{t=1}^{m-1} d_{\sigma_t, \sigma_{t+1}} \quad (8)$$

First, we establish GO-LR as a TSP-path heuristic that outputs a Hamiltonian path. (see Appendix B). Then, below, we show the connection between our feature ordering and TSP-path.

The GO-LR objective in Eq. (2) is a general seriation / linear arrangement criterion. We now exhibit a non-circular special case in which Feature Ordering becomes exactly the TSP-path problem, implying that Feature Ordering strictly generalizes TSP-path (Carmona et al., 2023).

Definition 3.11 (Path-Edge Feature Ordering (Adjacency-by-Position)). Fix m and define the path-edge set on positions

$$E_{\text{path}} = \{(t, t+1) : t = 1, \dots, m-1\} \quad (9)$$

Given a complete weighted graph $\mathcal{K} = (V, \binom{V}{2}, d)$ with $|V| = m$, a permutation $\sigma = (\sigma_1, \dots, \sigma_m)$ induces an ordering map $\pi_\sigma : V \rightarrow \{1, \dots, m\}$ via $\pi_\sigma(\sigma_t) = t$. We define the path-edge feature ordering objective:

$$D_{\text{path}}(\pi_\sigma) = \sum_{t=1}^{m-1} d_{\sigma_t, \sigma_{t+1}} \quad (10)$$

This is equivalent to restricting Eq. (2) to adjacency-by-position interactions.

Theorem 3.12 (Exact Equivalence to TSP-path). *Minimizing the path-edge feature ordering objective in Eq. (10) over all permutations σ is exactly the TSP-path problem on $\mathcal{K}$ with path cost given by $\text{PathCost}(\sigma)$ in Eq. (8).*

Proof sketch. For any permutation σ , Eq. (10) can be rewritten as $\sum_{t=1}^{m-1} d_{\sigma_t, \sigma_{t+1}} = \text{PathCost}(\sigma)$ by definition. Thus, the minimizers coincide. $\square$

Corollary 3.13 (TSP-path embeds into Feature Ordering). *TSP-path is a special case of feature ordering. Consequently, feature ordering (strictly) generalizes TSP-path.*

Remark 3.14. This equivalence holds for the path-edge special case. In GO-LR, the practical objective remains the full dispersion in Eq. (2) (MinLA), while the nearest-neighbor constructor corresponds to a TSP-path heuristic for initialization, and GO-LR then applies local refinement under the full dispersion objective in Eq. (2) on a complete graph built from the chosen dissimilarity metric.

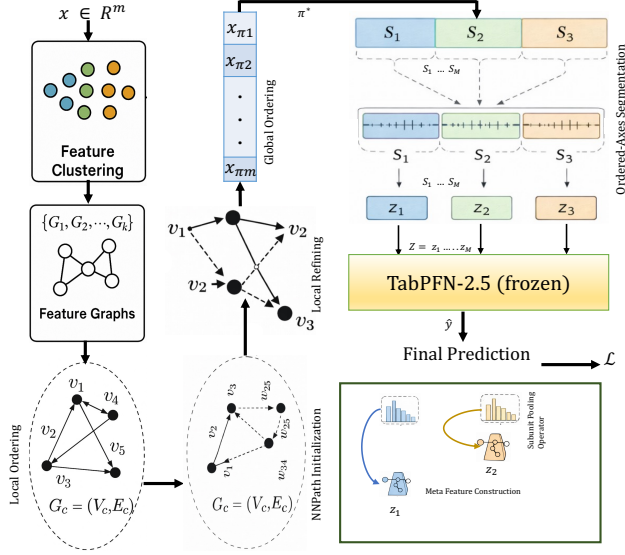


Figure 3. End-to-end architecture of GOTabPFN. The feature clustering block denotes the discovery of local feature-dependence groups, implemented by estimating cluster-wise feature graphs G_c from local sample contexts; GO-LR then obtains a global order Π^* , and NSC compresses contiguous ordered segments into meta-features $Z(x)$, which are passed to a frozen TabPFN-2.5 head.

Global Aggregation (Mean-Rank Integration). Let π_c be a local ordering for cluster c and let $r_c(j)$ be the rank (position) of feature j in π_c . With cluster weights $\alpha_c \geq 0$ and $\sum_{c=1}^k \alpha_c = 1$, GO-LR forms a global order by Eq. 11. This aggregation produces a single global permutation consistent with the set of local cluster-wise permutations.

$$\bar{r}(j) = \sum_{c=1}^k \alpha_c r_c(j), \quad \Pi^* = \underset{j=1}{\operatorname{argsort}}^m \bar{r}(j) \quad (11)$$

Algorithm 1 captures the steps in the proposed GO-LR algorithm.

3.2. Neuro-Inspired Subunit Compression (NSC)

Motivation. We design a representation interface that allows TabPFN to scale to HDLSS tabular data without re-training or architectural modification. Cortical pyramidal neurons receive on the order of 20,000-30,000 synaptic inputs (Poirazi et al., 2003), yet these inputs are not integrated as a single linear sum (Major et al., 2013). Instead, inputs are organized into multiple dendritic subunits (Kastellakis et al., 2015), each acting as a nonlinear integration compartment. Here, correlated synapses may exhibit local clustering (Ujfalussy & Makara, 2020) and trigger N-methyl-D-aspartate (NMDA)-mediated plateau potentials (Schiller et al., 2000) that pool dozens to hundreds of inputs into a single subunit-level signal (Kirchner & Gjorgjieva, 2021). This subunit-based organization provides a canonical biological mechanism (Beniaguev et al., 2021) for compressing extremely high-dimensional inputs into a compact set of functional representations (Wu et al., 2018). This locality-

Algorithm 1 Graph-guided Ordering with Local Refinement (GO-LR)

Require: $X \in \mathbb{R}^{n \times m}$, clusters k , metric ϕ , passes P
Ensure: Global order Π^* and local orders $\{\pi_c\}_{c=1}^k$

- 1: $\{X^{(c)}\}_{c=1}^k \leftarrow \text{Cluster}(X, k)$; $\mu^{(c)} \leftarrow \text{mean}(X^{(c)})$
- 2: **for** $c = 1$ to k **do**
- 3: $G_c \leftarrow \text{Sym}(\text{FeatureDissimilarity}(X^{(c)}, \phi)) \in \mathbb{R}^{m \times m}$ // undirected
- 4: $\pi_c \leftarrow \text{NNPath}(G_c)$
- 5: $\pi_c \leftarrow \text{Refine}(\pi_c, G_c, P)$ // direction-select + P passes
- 6: $r_c(j) \leftarrow \text{rank}(j \text{ in } \pi_c)$ **for** $j = 1, \dots, m$
- 7: **end for**
- 8: $\tilde{\alpha}_c \leftarrow (\varepsilon + \text{mean}_{c'} \|\mu^{(c)} - \mu^{(c')}\|_2)^{-1}$; $\alpha_c \leftarrow \tilde{\alpha}_c / \sum_{c'} \tilde{\alpha}_{c'}$
- 9: $\bar{r}(j) \leftarrow \sum_{c=1}^k \alpha_c r_c(j)$; $\Pi^* \leftarrow \text{argsort}_j \bar{r}(j)$
- 10: **return** Global feature order Π^* and local orders $\{\pi_c\}_{c=1}^k$

driven compression view is also consistent with prior signal-compression work, where edge-aware prediction has been used to exploit local structure in high-dimensional hyperspectral imagery (Jain & Adjeroh, 2007). We adopt this principle as an algorithmic inductive bias for HDLSS tabular data (Baln et al., 2019). See Alg. 2 for steps in NSC.

Ordered-Axis Segmentation. Let $x \in \mathbb{R}^m$ denote a tabular sample and let Π^* be the global feature permutation produced by GO-LR (Section 3.1). We define the reordered feature vector in Eq. 12. Given a target number of meta-features M , we set the segment length $s = \lceil m/M \rceil$ (Eq. 13) and define contiguous segments $\{S_t\}_{t=1}^M$ by Eq. 14, which partition $\{1, \dots, m\}$ into ordered neighborhoods (subunits).

$$x^\Pi = (x_{\Pi^*(1)}, x_{\Pi^*(2)}, \dots, x_{\Pi^*(m)}) \quad (12)$$

$$s = \left\lceil \frac{m}{M} \right\rceil \quad (13)$$

$$S_t = \{(t-1)s + 1, \dots, \min(ts, m)\}, \quad t = 1, \dots, M \quad (14)$$

Adaptive segmentation. To let segment boundaries follow “transitions” in the ordered feature axis, we first summarize pairwise dissimilarities into a 1D signal. We reuse the global feature dissimilarity matrix $\bar{W} \in \mathbb{R}^{m \times m}$ already computed for GO-LR on the dataset (and keep it fixed at inference), e.g., $\bar{W} := \text{FeatureDissimilarity}(X, \phi)$ or $\bar{W} := \sum_{c=1}^k \alpha_c W_c$, so NSC itself does not introduce any additional $O(m^2)$ cost. For adjacent positions along the GO-LR order, we define the transition dissimilarity $\delta_t := \bar{W}_{\Pi^*(t), \Pi^*(t+1)}$ for $t = 1, \dots, m-1$; large δ_t indicates a sharp change between neighboring features. We then form the cumulative transition mass $c_t := \sum_{i=1}^{t-1} \delta_i$ for $t = 1, \dots, m$ with total $C := c_m = \sum_{i=1}^{m-1} \delta_i$. Given a desired number of segments M , we place cutpoints

Algorithm 2 Neuro-Inspired Subunit Compression (NSC)

Require: Training matrix $X_{\text{train}} \in \mathbb{R}^{n \times m}$, sample $x \in \mathbb{R}^m$, global order Π^* , ID threshold $\tau \in (0, 1)$, bypass threshold m_0 , M -rule hyperparameters $(\gamma, M_{\min}, M_{\max})$, segmentation rule $\text{Seg}(\cdot)$ with $\ell_{\min}$, and (if transition-aware) dissimilarities $\Delta \in \mathbb{R}_+^{m-1}$.

Ensure: Compressed tokens $Z(x) \in \mathbb{R}^M$.

- 1: **Reorder:** $X^\Pi \leftarrow X_{\text{train}}[:, \Pi^*]$, $x^\Pi \leftarrow x[\Pi^*]$.
- 2: **PCA-ID:** compute $G = \frac{1}{n-1} X^\Pi (X^\Pi)^\top$; let $\{\lambda_i\}$ be eigenvalues of G (descending).
- 3: $\hat{d} \leftarrow \min\{k : \sum_{i=1}^k \lambda_i \geq \tau \sum_i \lambda_i\}$; IDF $\leftarrow \hat{d}/m$
- 4: **if** $m \leq m_0$ **then**
- 5: $M \leftarrow m$
- 6: **else**
- 7: $M \leftarrow \text{clip}(\lceil 2\hat{d} \rceil, M_{\min}, \min(M_{\max}, m))$ // or $\lceil \gamma \hat{d} \rceil / \text{IDF-rule}$
- 8: **end if**
- 9: **Segment:** $\{\mathcal{S}_t\}_{t=1}^M \leftarrow \text{Seg}(m, M, \Delta, \ell_{\min})$ // uniform / equal-mass / largest-jump
- 10: **for** $t = 1$ **to** M **do**
- 11: **Fit (once):** center $X_{[:, \mathcal{S}_t]}^\Pi$ to get mean μ_t ; compute first PC direction v_t (CPU SVD), fix sign deterministically.
- 12: **Tokenize:** $z_t(x) \leftarrow (x_{\mathcal{S}_t}^\Pi - \mu_t)^\top v_t$ // scalar
- 13: **end for**
- 14: **return** $Z(x) \leftarrow (z_1(x), \dots, z_M(x))$

$1 \leq \tau_1 < \dots < \tau_{M-1} < m$ along this 1D signal in two ways: (i) a largest-jump rule that selects the indices of the $M-1$ largest δ_t values (subject to a minimum segment length $\ell_{\min}$), and (ii) an equal-mass rule that treats c_t as a discrete CDF and chooses cutpoints by Eq. 15.

$$\tau_\ell := \min \left\{ t \in \{2, \dots, m-1\} : c_t \geq (\ell/M) C \right\}, \quad (15)$$

$$\ell = 1, \dots, M-1$$

Again enforcing $\ell_{\min}$ with a uniform fallback if needed. Finally, we materialize segments as $\mathcal{S}_1 = \{1, \dots, \tau_1\}$, $\mathcal{S}_t = \{\tau_{t-1} + 1, \dots, \tau_t\}$ for $t = 2, \dots, M-1$, and $\mathcal{S}_M = \{\tau_{M-1} + 1, \dots, m\}$, so that each subunit is an ordered neighborhood bounded by large transitions in the feature axis.

Subunit Pooling and Meta-Feature Construction.

Segment descriptors. Beyond mean and variance, we optionally summarize each segment u_t using a richer descriptor $\psi(u_t)$ that includes higher-order and robust statistics (e.g., skewness, kurtosis, median, and interquartile range), enabling NSC to capture distributional shape within each ordered region at negligible extra cost.

Let $\psi : \mathbb{R}^{|\mathcal{S}_t|} \rightarrow \mathbb{R}^q$ denote a (possibly learn-free) segment descriptor of dimension q , and let $g_\theta : \mathbb{R}^q \rightarrow \mathbb{R}^d$ be a shared lightweight pooling network that maps each descriptor to a d -dimensional meta-feature. In practice, g_θ

may be a shallow MLP (or linear map) applied to $\psi(u_t)$; the same g_θ is reused across segments to enforce parameter sharing and stability. The t -th meta-feature is defined in Eq. 16. NSC outputs the compressed meta-feature sequence $Z(x) = (z_1, \dots, z_M)$, which is subsequently provided to the TabPFN predictor head.

$$z_t = g_\theta(\psi(u_t)), \quad u_t := x_{\mathcal{S}_t}^\Pi \quad (16)$$

$$Z(x) = (z_1, \dots, z_M) \in \mathbb{R}^{M \times d} \quad (17)$$

NSC variants. We instantiate NSC in four variants (details in Appendix D): (i) NSC: uniform segments + learned pooling, (ii) NSC-P: same with PCA-based intrinsic-dimension rule for M , (iii) NSC-SP: PCA-based segment (SegPCA) pooling with a fixed M , and (iv) NSC-pSP: PCA-based intrinsic-dimension rule for M combined with SegPCA pooling. GOTabPFN uses NSC-pSP in the experiments.

Choosing the Number of Meta-Features (NSC-pSP). To adapt the compression level to dataset complexity, we tie the meta-feature budget M to an estimate of the intrinsic dimensionality $\hat{d}$ of the training data. Let $\hat{X} \in \mathbb{R}^{n \times m}$ denote the standardized training matrix (zero mean, unit variance per feature), and let $\Sigma = \frac{1}{n-1} \hat{X}^\top \hat{X}$ be its empirical covariance (or correlation) matrix with nonzero eigenvalues $\{\lambda_i\}_{i=1}^r$, $r \leq \min(n, m)$.

For the NSC-pSP variant used in our main experiments, we estimate $\hat{d}$ via a PCA cumulative-variance rule (Hotelling, 1933). We define the explained-variance ratio and its cumulative sum in Eqns. 18 and 19. Given a target variance-retention level $\tau \in (0, 1)$ (e.g., $\tau \in \{0.90, 0.95, 0.99, 0.9975\}$), the PCA-based intrinsic dimension is defined by Eq. 20 and NSC-pSP sets $\hat{d} = \hat{d}_{\text{PCA}}(\tau)$. We then choose the meta-feature budget via Eq. 21 where $\text{clip}(x, a, b) = \min(\max(x, a), b)$. For non-HDLSS regimes (e.g., $m \leq 400$), we bypass compression by setting $M = m$. This rule ensures that the number of meta-features scales with intrinsic, rather than ambient, dimensionality, yielding aggressive compression in highly redundant HDLSS settings while avoiding unnecessary bottlenecks when features are already compact. Implementation details and alternative intrinsic-dimension rules used by the other NSC variants are given in Appendix C.

$$\text{EVR}_i = \frac{\lambda_i}{\sum_{j=1}^r \lambda_j}, \quad i = 1, \dots, r \quad (18)$$

$$\text{CUM}(k) = \sum_{i=1}^k \text{EVR}_i, \quad k = 1, \dots, r \quad (19)$$

$$\hat{d}_{\text{PCA}}(\tau) = \min \left\{ k \in \{1, \dots, r\} : \text{CUM}(k) \geq \tau \right\} \quad (20)$$

$$M = \text{clip}(\lceil 2\hat{d} \rceil, 32, \min(512, m)) \quad (21)$$

PCA-centric post-segmentation pooling (SegPCA). For the PCA-centric NSC variants (NSC-SP and NSC-pSP), once the meta-feature budget M is determined (Sec. 3.2) and the ordered segments $\{\mathcal{S}_t\}_{t=1}^M$ are formed (Eqs. 13-14), we construct one scalar token per segment by projecting onto a segment specific first principal direction learned on the training set. Let $u_t(x) := x_{\mathcal{S}_t}^\Pi \in \mathbb{R}^{|\mathcal{S}_t|}$ and let $X^\Pi \in \mathbb{R}^{n \times m}$ denote the standardized training matrix after applying Π^* . We define the training submatrix for segment t as $X_t := X_{:, \mathcal{S}_t}^\Pi \in \mathbb{R}^{n \times |\mathcal{S}_t|}$. We compute the segment mean and covariance by Eq. 22 and take the first principal direction using Eq. 23. The t -th meta-feature is then the centered projection (Eq. 24), yielding a $d=1$ token sequence $Z_{\text{SegPCA}}(x)$ (Eq. 25). Optionally, we apply a deterministic sign convention to v_t (e.g., flipping v_t so that segment scores positively correlate with a fixed reference such as the within-segment sample mean), which leaves the subspace unchanged but improves reproducibility.

$$\mu_t = \frac{1}{n} \sum_{i=1}^n X_{t,i}, \quad \mu_t \in \mathbb{R}^{|\mathcal{S}_t|}, \quad (22)$$

$$\Sigma_t = \frac{1}{n-1} (X_t - \mathbf{1}\mu_t^\top)^\top (X_t - \mathbf{1}\mu_t^\top)$$

$$v_t = \arg \max_{\|v\|_2=1} v^\top \Sigma_t v \in \mathbb{R}^{|\mathcal{S}_t|} \quad (23)$$

$$z_t(x) = (u_t(x) - \mu_t)^\top v_t \in \mathbb{R}, \quad t = 1, \dots, M \quad (24)$$

$$Z_{\text{SegPCA}}(x) = (z_1(x), \dots, z_M(x)) \in \mathbb{R}^{M \times 1} \quad (25)$$

Summary. NSC acts as a shared piecewise pooling operator, defined by Prop. C.1 (App. C). NSC transforms GO-LR-ordered high-dimensional tabular inputs into a compact sequence of structured meta-features through contiguous segmentation and shared pooling, introducing an HDLSS-friendly inductive bias inspired by subunit-based cortical computation while remaining purely statistical and computationally efficient. By compressing m raw features into $M \ll m$ meta-features, NSC reduces effective sequence length presented to TabPFN-style backbones (e.g., TabPFN-2.5 (Grinsztajn et al., 2025) or other variants), yielding lower compute & memory cost while preserving order-induced locality to make original TabPFN versions usable for HDLSS regime. Per sample, NSC is $O(m)$ when ψ uses linear-time statistics (e.g., moments); robust summaries such as quantiles can be computed approximately in linear time/exactly with a mild $O(|\mathcal{S}_t| \log |\mathcal{S}_t|)$ overhead if sorting is used.

TabPFN-2.5 Head (non-differentiable). NSC module

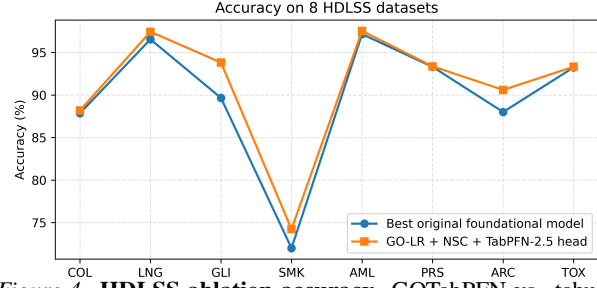


Figure 4. **HDLSS ablation accuracy.** GOTabPFN vs. tabular foundation models on 8 HDLSS datasets.

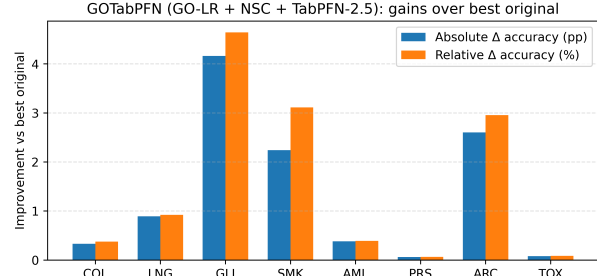


Figure 5. **Ablation gains.** Absolute and relative gains of GOTabPFN over the best original foundation-model head.

compresses each sample into a fixed-dimensional representation $Z(x) \in \mathbb{R}^M$ (or $Z(x) \in \mathbb{R}^{M \times d}$, flattened to $\mathbb{R}^{Md}$). We then use TabPFN-2.5 as the predictor head: for each train/validation split, we fit TabPFN-2.5 on $\{(Z(x_i), y_i)\}_{i \in \mathcal{I}_{\text{train}}}$ and evaluate on $Z(x_j)$ for $j \in \mathcal{I}_{\text{val}}$ without backpropagation through the head. This design treats NSC as a compression interface that maps HDLSS inputs into a feature budget compatible with TabPFN variants, while retaining strong tabular foundation models by Hollmann et al. (2023; 2025); Grinsztajn et al. (2025).

4. Experimental Results

Algorithms and datasets used. We use eight biomedical HDLSS datasets (e.g., Arcene, Colon, GLI-85, Lung, etc.) from the repository of Li et al. (2018), also used by ProtoGate (Jiang et al., 2024). We follow the standard HDLSS setting where features far exceed samples ($m \gg n$). To study when feature ordering helps, we propose an empirical locality-based criterion; see Appendix F for the criterion and usage guidance. We compare against 55 baselines spanning classical ML/GBDT, HDLSS feature selection, deep tabular models, and small tabular foundation models. Modern methods including TANDEM (Naor & Lindenbaum, 2025), TabPFN Wide (Kolberg et al., 2025), TabDPT (Ma et al., 2025), TabICL (Jingang et al., 2025), BETA (Liu & Ye, 2025), TuneTables (Feuer et al., 2024), and ProtoGate perform strongly on HDLSS classification, while classical baselines (MLP, Lasso) remain competitive, consistent with ProtoGate. Full baseline details are in Appendix G.

Experimental set up. We use 5×5 nested cross-validation (25 repeats) on the HDLSS datasets, matching ProtoGate’s protocol, for all baselines. Experiments run on the TI-

Table 1. Top-10 performance on 8 HDLSS datasets (mean accuracy with subscripted standard deviation over 5×5 CV). **Bold** denotes the best result per dataset and underline denotes the second-best. Rank is the average rank across datasets (lower is better), computed with standard tie-breaking. Dataset abbreviations: COL = Colon, LNG = Lung, GLI = GLI-85, SMK = SMK_CAN_187, AML = ALLAML, PRS = Prostate-GE, ARC = Arcene, TOX = TOX-171. Model abbreviations: *GOTabPFN = our method, TabPFN-W = TabPFN Wide, TTables = TuneTables, BETA = TabPFN Unleashed. See Table G.1 in Appendix G for full results against 55 baselines.

MODEL/DB	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX	RANK
#SAMPLES	62	203	85	187	72	102	200	171	—
#FEATURES	2000	3312	22283	19993	7129	5966	10000	5748	—
#CLASSES	2	5	2	2	2	2	2	4	—
*GOTABPFN	88.18 ± 10.05	97.44 ± 2.32	93.82 ± 5.81	74.23 ± 5.17	97.54 ± 3.86	93.37 ± 4.48	90.60 ± 3.97	93.33 ± 4.74	1.00 ± 0.00
TANDEM	86.15 ± 7.75	96.46 ± 2.88	<u>91.53</u> ± 6.02	<u>72.72</u> ± 5.69	95.81 ± 5.53	91.55 ± 4.32	86.90 ± 6.34	93.08 ± 2.61	<u>3.63</u> ± 1.32
TABPFN-W	87.85 ± 7.28	<u>96.55</u> ± 2.15	88.47 ± 5.75	68.78 ± 8.60	<u>97.16</u> ± 4.10	93.10 ± 5.92	<u>88.00</u> ± 5.20	89.35 ± 4.95	<u>3.75</u> ± 2.38
TABDPT	86.26 ± 7.27	96.05 ± 2.57	87.76 ± 5.86	71.99 ± 7.32	96.32 ± 4.15	90.94 ± 5.72	82.10 ± 6.48	<u>93.25</u> ± 3.44	4.88 ± 1.69
TABICL	84.62 ± 10.52	96.36 ± 2.61	87.06 ± 6.23	68.73 ± 7.28	95.52 ± 5.59	90.17 ± 5.93	82.60 ± 6.14	88.78 ± 5.92	7.63 ± 2.29
BETA	84.73 ± 9.36	94.38 ± 3.34	86.21 ± 8.91	70.21 ± 5.61	95.67 ± 7.54	87.53 ± 4.67	86.45 ± 5.92	90.38 ± 6.42	8.13 ± 4.31
TTABLES	86.80 ± 2.14	94.37 ± 2.35	89.66 ± 3.12	70.28 ± 6.46	95.80 ± 2.14	<u>93.31</u> ± 2.83	81.40 ± 3.66	77.96 ± 2.55	8.38 ± 7.70
LASO	79.40 ± 10.18	94.47 ± 4.39	85.88 ± 4.71	61.19 ± 13.72	87.24 ± 3.39	91.18 ± 6.39	81.00 ± 3.39	91.86 ± 6.03	11.13 ± 5.06
MLP	83.95 ± 9.80	96.47 ± 2.69	85.41 ± 8.00	59.05 ± 7.44	89.98 ± 9.17	89.20 ± 6.07	78.40 ± 4.05	92.48 ± 4.28	11.63 ± 5.45
PROTOGATE	83.95 ± 9.82	93.44 ± 6.37	82.48 ± 5.68	60.16 ± 5.10	86.12 ± 3.34	90.58 ± 5.72	81.50 ± 5.10	92.34 ± 5.67	12.06 ± 4.77

Table 2. Performance on 8 cross-domain datasets (mean accuracy with subscripted standard deviation over 5×5 CV). **Bold** denotes the best result per dataset and underline denotes the second-best. Rank is the avg. rank across datasets (lower is better), computed with standard tie-breaking; Some datasets use only 50- 60 Optuna trials due to compute limits; “—” denotes OOM/unsupported runs, ranked last. ProtoGate targets very few sample and scales less favorably with larger n . Dataset abbreviations: ORL = orlraws10P, BAS = BASEHOCK, REL = RELATHE, PCM = PCMAC, CCY = Cell Cycle, CIF = CIFAR-10, DF-R = DrivFace-Regression, DF-C = DrivFace-Classification, REG = Regression (R^2). Model abbreviations: *GOTabPFN = our method, TabPFN-W = TabPFN Wide, TTables = TuneTables.

MODEL/DB	ORL	BAS	REL	PCM	CCY	CIF	DF-R	DF-C	RANK
#SAMPLES	100	1993	1427	1943	1067	11000	606	606	—
#FEATURES	10304	4862	4322	3289	42728	2048	6400	6400	—
#CLASSES	10	2	2	2	3	10	REG.	7	—
*GOTABPFN	100.00 ± 0.00	97.11 ± 1.00	88.87 ± 1.32	89.51 ± 2.24	79.94 ± 2.53	88.45 ± 0.89	0.6548 ± 0.0992	86.70 ± 2.48	1.25 ± 0.66
TABDPT	97.32 ± 0.68	97.00 ± 0.68	88.26 ± 1.49	87.58 ± 0.96	77.52 ± 2.44	88.00 ± 0.40	<u>0.6505</u> ± 0.0820	83.57 ± 3.04	4.44 ± 1.49
TABPFN-W	96.10 ± 6.52	<u>97.04</u> ± 0.72	87.84 ± 3.90	88.56 ± 0.71	<u>78.67</u> ± 2.42	88.12 ± 0.60	0.6430 ± 0.0772	85.42 ± 2.66	<u>3.88</u> ± 1.62
TTABLES	93.14 ± 1.02	93.14 ± 1.02	88.26 ± 2.31	84.27 ± 3.06	78.30 ± 1.13	78.05 ± 0.15	0.6332 ± 0.0675	84.62 ± 2.43	5.94 ± 1.91
TABICL	<u>99.20</u> ± 1.87	96.84 ± 0.75	88.15 ± 1.68	88.68 ± 1.94	—	87.61 ± 1.00	—	<u>85.55</u> ± 2.45	5.31 ± 2.46
TANDEM	99.00 ± 2.45	96.72 ± 0.86	89.42 ± 1.42	<u>88.99</u> ± 1.33	77.31 ± 1.97	87.80 ± 0.36	0.6488 ± 0.0770	84.42 ± 2.55	4.00 ± 1.94
LASO	96.00 ± 3.82	96.74 ± 0.88	86.87 ± 1.46	88.85 ± 1.80	77.86 ± 2.42	86.42 ± 1.10	0.3194 ± 0.0668	77.95 ± 3.03	6.13 ± 1.69
MLP	91.00 ± 8.37	96.91 ± 1.15	<u>89.12</u> ± 1.85	87.71 ± 1.95	76.38 ± 1.42	<u>88.15</u> ± 0.54	0.5682 ± 0.1258	80.59 ± 3.72	5.25 ± 2.17
PROTOGATE	66.40 ± 11.32	—	58.40 ± 3.15	—	—	—	−0.1636 ± 0.0738	66.70 ± 10.20	8.81 ± 0.35

Table 3. Colon ablation. Accuracy over 5×5 CV. All NSC variants use NSC-pSP unless noted.

Variant	Acc. (%)
GO-LR+NSC-pSP (tokens)	88.18 ± 10.05
No NSC (TabPFN-2.5)	86.85 ± 9.16
Rand. order + NSC-pSP	84.21 ± 10.34
GO-LR+NSC-pSP + LogReg	82.67 ± 11.05
Id. order + NSC-pSP	81.67 ± 11.91
GO-LR+NSC-pSP (uniform seg.)	80.00 ± 11.30
Mean-pool NSC-pSP, no GO-LR	64.59 ± 3.67

TAN cluster (x86_64 CPU, 188,GB RAM, TITAN RTX 24,GB) with PyTorch 2.4.1+cu121. We disable AMP for GOTabPFN, but some transformer baselines require AMP and/or DP/DDP to fit GPU memory. We tune only GO-LR and NSC in GOTabPFN via Optuna (Akiba et al., 2019) (150 trials/dataset), following standard tabular tuning practice (Gorishniy et al., 2025; 2024; 2022; 2021); the TabPFN-2.5 head remains frozen. For baselines, we use authors’

recommended settings when tuning is unnecessary; when tuning is recommended, we also run Optuna (150 trials) for fairness. Among TabPFN-based models, TuneTables likewise uses lightweight Optuna tuning. See Fig. A.1.

HDLSS classification performance. Table 1 reports mean accuracy ($\pm$ std) over 5×5 repeated CV on eight HDLSS benchmarks. GOTabPFN is best on all datasets (8/8) with the lowest average rank (1.00 ± 0.00), indicating consistent dominance. Relative to the strongest TabPFN variants (TabPFN-Wide, TuneTables, BETA), gains are largest on harder/noisier datasets (e.g., SMK, TOX) and smaller on near-saturated ones (e.g., ALLAML, Prostate), where headroom is limited. GOTabPFN also shows comparable or lower split-to-split variance, suggesting improved robustness in the low-sample regime. Full results against 55 baselines are in Table G.1 (Appendix G). Across the 8

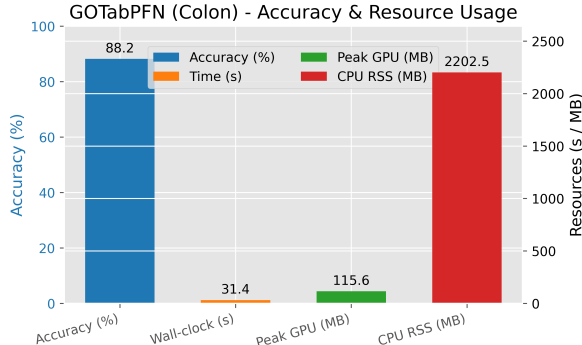


Figure 6. **Accuracy-resource profile on Colon.** Wall-clock time, peak GPU memory, and CPU RSS for GOTabPFN.

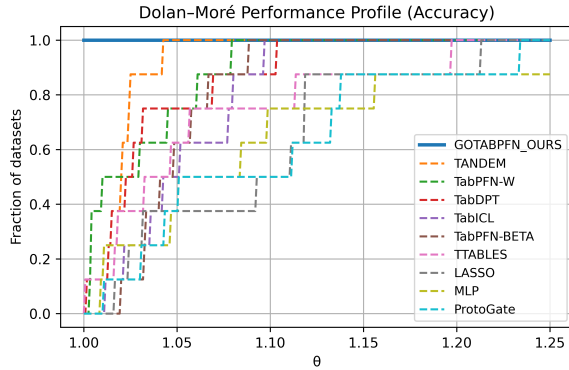


Figure 7. **Dolan-Moré profiles.** Performance profiles over 8 HDLSS datasets against the top-10 baselines.

cross-domain(App. T) high-dimensional datasets (Table 2), GOTabPFN achieves the best average rank (1.25 ± 0.66) and obtains the top result on 7/8 tasks, including image-derived, biological, text-like, and camera-sensor datasets. The strongest competing methods are TabPFN-W, TANDEM, TabDPT, and MLP, but their gains are less consistent across domains. These results suggest that GO-LR+NSC provides a robust ordering-aware compression interface for diverse HDLSS and related high-dimensional regimes.

Statistical significance. Across 8 HDLSS datasets, GOTabPFN achieves the best average rank and is separated from competing baselines by Friedman (Friedman, 1937)/Nemenyi (Nemenyi, 1963) critical-difference analysis (Fig. I.1). While paired Wilcoxon signed-rank tests (Demšar, 2006) yield consistent directional improvements (all $p_{\text{raw}} = 0.00781$), significance does not always survive Holm correction due to the small number of datasets and the resulting conservativeness of multiple-comparison control (Table I.1). Additional statistical details are in Appendix I.

Runtime and computational complexity. For n samples, m features, k sample-clusters, and M NSC tokens, GO-LR costs $\mathcal{O}(nmkI + m^2n)$, plus $\mathcal{O}(km^2b)$ for KL graph construction; refinement/integration adds $\mathcal{O}(kPm^2 + k^2m)$, where P is no. of Sweep Refine passes. In HDLSS ($n \ll m$), NSC fits per-segment PC1 directions in $\mathcal{O}(n^2m)$ and tokenizes in $\mathcal{O}(nm)$. The TabPFN-2.5 head on M tokens is dominated by attention over n context points, scaling as $\tilde{\mathcal{O}}(n^2M)$ per split. On Colon, GOTabPFN achieves 88.2%

Table 4. **GOTabPFN gains over foundation-model heads.** Accuracy on 8 HDLSS datasets under 5×5 CV. “Best orig” is the best among TabDPT, TabPFN-Wide, BETA, TuneTables, and TabICL.

Dataset	Best orig	GOTabPFN	Δ_{abs}	Δ_{rel}
Colon	87.85	88.18	0.33	0.38
Lung	96.55	97.44	0.89	0.92
GLI85	89.66	93.82	4.16	4.64
SMK	71.99	74.23	2.24	3.11
ALLAML	97.16	97.54	0.38	0.39
Prostate	93.31	93.37	0.06	0.06
Arcene	88.00	90.60	2.60	2.95
TOX	93.25	93.33	0.08	0.09

accuracy in 31.4s with modest peak GPU use (115.6MB; Fig. 6); memory is primarily CPU-side (RSS $\approx$ 2202.5MB), consistent with GO-LR graph construction/refinement.

Ablations. Figs. 4-5 and Table 4 quantify adding GO-LR ordering and NSC compression before a frozen TabPFN-2.5 head: GOTabPFN matches or exceeds the best original tabular foundation model on all 8 HDLSS datasets, with largest gains on GLI-85 (+4.16 pp), Arcene (+2.60 pp), and SMK (+2.24 pp), and near-saturated improvements on ALLAML/Prostate. On Colon (Table 3), removing NSC lowers accuracy, and replacing GO-LR with identity/random orders drops more, indicating NSC benefits from structure-revealing orderings; transition-aware segmentation and PCA-based token embeddings outperform uniform segmentation or mean-pooling, while swapping TabPFN-2.5 for logistic regression substantially degrades performance, suggesting both the ordering+compression pipeline and a strong TabPFN-style predictor are needed. The Dolan-Mor’e profile (Dolan & Moré, 2002) (Fig. 7, following TANDEM (Naor & Lindenbaum, 2025)) shows stronger cross-dataset consistency: GOTabPFN stays closest to the per-dataset best (curve at 1.0), whereas others need larger tolerance θ . Additional ablations appear in App. J.

Limitations. GOTabPFN inherits constraints from its frozen TabPFN-2.5 backbone, including its limit of up to 10 classes and sample-size limit of 50K samples. GO-LR+NSC adds ordering and compression before TabPFN inference; runtime can increase for larger sample sizes. Thus, GOTabPFN is most suitable for HDLSS and related low-sample, high-dimensional regimes, rather than high-sample regimes.

5. Conclusion

We present **GOTabPFN**, which makes TabPFN-style small tabular foundation models effective in HDLSS regimes. GOTabPFN couples MinLA-grounded feature ordering (GO-LR) with a neuro-inspired stable, locality-preserving compression interface (NSC) that converts high-dimensional tables into compact token sequences. Without retraining or modifying the TabPFN-2.5 backbone, this ordering-to-tokenization pipeline improves accuracy and robustness under tight token budgets across diverse HDLSS benchmarks. GOTabPFN provides a theory-grounded, practical route to scalable in-context tabular prediction when $m \gg n$.

Acknowledgements

This work was supported in part by the US National Science Foundation under Awards #1920920, #2125872, and #2223793. We thank the anonymous ICML reviewers for their valuable feedback and suggestions.

Impact Statement

GOTabPFN aims to make tabular foundation models more usable in HDLSS settings by introducing an ordering-aware compression interface that reduces feature dimensionality while preserving local structure. This can benefit scientific and biomedical domains where data are scarce but feature spaces are large. However, as with any predictive model, deployment in sensitive domains should include careful validation, bias assessment, and domain-expert oversight.

Software and Data

The project webpage is available at <https://www.zadidhabib.com/gotabpfn.html>. Code, notebooks, and installation instructions are available at <https://github.com/zadid6pretam/GOTabPFN>; the package can also be installed with `pip install gotabpfn`. Our experiments use TabPFN-2.5 as the frozen backbone with `tabpfn==6.3.1`; newer default `tabpfn` installations may install TabPFN-3 or later, so reproducing our results requires `pip install tabpfn==6.3.1` and may require Prior Labs/Hugging Face checkpoint access. Most HDLSS datasets are from the scikit-feature dataset repository (Li et al., 2018) (<https://jundongli.github.io/scikit-feature/datasets.html>); DrivFace is from UCI (Hernández-Sabat et al., 2016); CIFAR-10 embeddings are derived from the Kaggle CIFAR-10 dataset (Cukierski, 2013); and Cell Cycle is from Mahdessian et al. (2021) via GEO (NCBI, 2021).

References

- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. doi: 10.48550/arXiv.2303.08774.
- Ahamed, M. A. and Cheng, Q. MambaTab: A Plug-and-Play Model for Learning Tabular Data. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 369–375. IEEE, 2024. doi: 10.1109/MIPR62202.2024.00065.
- Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019. doi: 10.1145/3292500.3330701.
- Aoshima, M., Shen, D., Shen, H., Yata, K., Zhou, Y.-H., and Marron, J. S. A Survey of High Dimension Low Sample Size Asymptotics. *Australian & New Zealand Journal of Statistics*, 60(1):4–19, 2018. doi: 10.1111/anzs.12212.
- Arabie, P. and Hubert, L. J. Combinatorial Data Analysis. *Annual Review of Psychology*, 43:169–203, 1992. doi: 10.1146/annurev.ps.43.020192.001125.
- Arik, S. Ö. and Pfister, T. TabNet: Attentive Interpretable Tabular Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021. doi: 10.1609/aaai.v35i8.16826.
- Atkins, J. E., Boman, E. G., and Hendrickson, B. A Spectral Algorithm for Seriation and the Consecutive Ones Problem. *SIAM Journal on Computing*, 28(1):297–310, 1998. doi: 10.1137/S0097539795285771.
- Balin, M. F., Abid, A., and Zou, J. Concrete Autoencoders: Differentiable Feature Selection and Reconstruction. In *International Conference on Machine Learning*, pp. 444–453. PMLR, 2019.
- Barthel, K. U., Barthel, F. T., and Eisert, P. Permutation Learning with Only N Parameters: From SoftSort to Self-Organizing Gaussians. In *2025 33rd European Signal Processing Conference (EUSIPCO)*, pp. 1892–1896. IEEE, 2025. doi: 10.23919/EUSIPCO63237.2025.11226796.
- Behrisch, M., Bach, B., Henry Riche, N., Schreck, T., and Fekete, J.-D. Matrix Reordering Methods for Table and Network Visualization. In *Computer Graphics Forum*, volume 35, pp. 693–716. Wiley Online Library, 2016. doi: 10.1111/cgf.12935.
- Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*, 2020. doi: 10.48550/arXiv.2004.05150.
- Beniaguev, D., Segev, I., and London, M. Single Cortical Neurons as Deep Artificial Neural Networks. *Neuron*, 109(17):2727–2739, 2021. doi: 10.1016/j.neuron.2021.07.002.
- Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Mohammadi Sepahvand, N., Raff, E., Madan, K., Voleti, V., Kahou, S. E., Michalski, V., Arbel, T., Pal, C., Varoquaux, G., and Vincent, P. Accounting for Variance in Machine Learning Benchmarks. *Proceedings of Machine Learning and Systems*, 3: 747–769, 2021.

- Brahmavar, S. B., Li, Y., and Oliva, J. Towards Universal Neural Inference. *arXiv preprint arXiv:2508.09100*, 2025. doi: 10.48550/arXiv.2508.09100.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- Carmona, M., Chepoi, V., Naves, G., and Pr  a, P. A Simple and Optimal Algorithm for Strict Circular Seriation. *SIAM Journal on Mathematics of Data Science*, 5(1): 201–221, 2023. doi: 10.1137/22M1495342.
- Casella, G. and Berger, R. *Statistical Inference*. Chapman and Hall/CRC, 2024. doi: 10.1201/9781003456285.
- Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to Explain: An Information-Theoretic Perspective on Model Interpretation. In *International Conference on Machine Learning*, pp. 883–892. PMLR, 2018.
- Chen, J., Liao, K., Wan, Y., Chen, D. Z., and Wu, J. DANets: Deep Abstract Networks for Tabular Data Classification and Regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3930–3938, 2022. doi: 10.1609/aaai.v36i4.20309.
- Chen, K.-Y., Chiang, P.-H., Chou, H.-R., Chen, T.-W., and Chang, T.-H. Trompt: Towards A Better Deep Neural Network for Tabular Data. In *International Conference on Machine Learning*, pp. 4392–4434. PMLR, 2023.
- Chen, T. and Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016. doi: 10.1145/2939672.2939785.
- Child, R., Gray, S., Radford, A., and Sutskever, I. Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509*, 2019. doi: 10.48550/arXiv.1904.10509.
- Christofides, N. Worst-Case Analysis of A New Heuristic for the Travelling Salesman Problem. In *Operations Research Forum*, volume 3, pp. 20. Springer, 2022. doi: 10.1007/s43069-021-00101-z.
- Cover, T. and Hart, P. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1): 21–27, 1967. doi: 10.1109/TIT.1967.1053964.
- Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. John Wiley & Sons, Hoboken, NJ, USA, 2 edition, 2006. doi: 10.1002/047174882X.
- Cox, D. R. The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 20(2):215–232, 1958. doi: 10.1111/j.2517-6161.1958.tb00292.x.
- Cukierski, W. CIFAR-10 - Object Recognition in Images. <https://kaggle.com/competitions/cifar-10>, 2013. Kaggle.
- Dangond, F. Chips around the World: Proceedings from the Nature Genetics Microarray Meeting. *Physiological Genomics*, 2(2):53–58, 2000. doi: 10.1152/physiolgenomics.2000.2.2.53.
- Dasgupta, S. and Gupta, A. An Elementary Proof of A Theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. doi: 10.1002/rsa.10073.
- Davies, D. L. and Bouldin, D. W. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):224–227, 1979. doi: 10.1109/TPAMI.1979.4766909.
- Dem  sar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- Deng, R., Li, Z., and Wang, M. GeoAggregator: An Efficient Transformer Model for Geo-Spatial Tabular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 11572–11580, 2025. doi: 10.1609/aaai.v39i11.33259.
- Devijver, E. and Gallopin, M. Block-Diagonal Covariance Selection for High-Dimensional Gaussian Graphical Models. *Journal of the American Statistical Association*, 113(521):306–314, 2018. doi: 10.1080/01621459.2016.1247002.
- Devroye, L., Gy  rfi, L., and Lugosi, G. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer New York, NY, 1996. doi: 10.1007/978-1-4612-0711-5.
- D  az, J., Petit, J., and Serna, M. A Survey of Graph Layout Problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356, 2002. doi: 10.1145/568522.568523.
- Dinh, T., Zeng, Y., Zhang, R., Lin, Z., Gira, M., Rajput, S., Sohn, J.-y., Papailiopoulous, D., and Lee, K. LIFT: Language-Interfaced Fine-Tuning for Non-Language Machine Learning Tasks. *Advances in Neural Information Processing Systems*, 35:11763–11784, 2022.
- Dolan, E. D. and Mor  , J. J. Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi: 10.1007/s101070100263.

- Dorigo, M. and Gambardella, L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 2002. doi: 10.1109/4235.585892.
- Eisenberg, R., Svirsky, J., and Lindenbaum, O. COPER: Correlation-based Permutations for Multi-View Clustering. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Eremeev, D., Bazhenov, G., Platonov, O., Babenko, A., and Prokhorenkova, L. Turning Tabular Foundation Models into Graph Foundation Models. In *NeurIPS 2025 New Perspectives in Graph Machine Learning Workshop*, 2025.
- Feuer, B., Schirrmeister, R. T., Cherepanova, V., Hegde, C., Hutter, F., Goldblum, M., Cohen, N., and White, C. TuneTables: Context Optimization for Scalable Prior-Data Fitted Networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024.
- Fogel, F., Jenatton, R., Bach, F., and d’Aspremont, A. Convex Relaxations for Permutation Problems. *Advances in Neural Information Processing Systems*, 26, 2013.
- Friedman, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. doi: 10.1080/01621459.1937.10503522.
- Garey, M., Johnson, D., and Stockmeyer, L. Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1(3):237–267, 1976. doi: 10.1016/0304-3975(76)90059-1.
- Garey, M. R., Johnson, D. S., and Stockmeyer, L. Some Simplified NP-Complete Problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, pp. 47–63, 1974. doi: 10.1145/800119.803884.
- Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting Deep Learning Models for Tabular Data. *Advances in Neural Information Processing Systems*, 34: 18932–18943, 2021.
- Gorishniy, Y., Rubachev, I., and Babenko, A. On Embeddings for Numerical Features in Tabular Deep Learning. *Advances in Neural Information Processing Systems*, 35: 24991–25004, 2022.
- Gorishniy, Y., Rubachev, I., Kartashev, N., Shlenskii, D., Kotelnikov, A., and Babenko, A. TabR: Tabular Deep Learning Meets Nearest Neighbors. In *(The Twelfth International Conference on Learning Representations)*, 2024.
- Gorishniy, Y., Kotelnikov, A., and Babenko, A. TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.
- Grinsztajn, L., Flöge, K., Key, O., Birkel, F., Jund, P., Roof, B., Jäger, B., Safaric, D., Alessi, S., Hayler, A., et al. TabPFN-2.5: Advancing the State of the Art in Tabular Foundation Models. *arXiv preprint arXiv:2511.08667*, 2025. doi: 10.48550/arXiv.2511.08667.
- Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization (IJCAI-17), 2017. doi: 10.24963/ijcai.2017/239.
- Guo, S., Deng, C., Wen, Y., Chen, H., Chang, Y., and Wang, J. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. In *International Conference on Machine Learning*, pp. 16813–16848. PMLR, 2024.
- Guo, S., Liu, H., Chen, X., Xie, Y., Zhang, L., Han, T., Chen, H., Chang, Y., and Wang, J. Optimizing Case-Based Reasoning System for Functional Test Script Generation with Large Language Models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’25)*, Volume 2, pp. 4487–4498, 2025. doi: 10.1145/3711896.3737254.
- Habib, A. Z. S. B., Wang, K., Hartley, M.-A., Doretto, G., and Adjeroh, D. A. TabSeq: A Framework for Deep Learning on Tabular Data via Sequential Ordering. In *International Conference on Pattern Recognition*, pp. 418–434. Springer, 2024. doi: 10.1007/978-3-031-78128-5_27.
- Habib, A. Z. S. B., Ahamed, M. Y., Gyawali, P. K., Doretto, G., and Adjeroh, D. A. BSTabDiff: Block-Subunit Diffusion Priors for High-Dimensional Tabular Data Generation. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficiency*, 2026a.
- Habib, A. Z. S. B., Doretto, G., and Adjeroh, D. A. DynaTab: Dynamic Feature Ordering as Neural Rewiring for High-Dimensional Tabular Data. In *Proceedings of the First Workshop on NeuroAI Multimodal Intelligence @ AAAI 2026*, volume 308 of *Proceedings of Machine Learning Research*, pp. 27–57. PMLR, 2026b. URL <https://proceedings.mlr.press/v308/habib26a.html>.

- Habib, A. Z. S. B., Tasnim, T., Islam, M. E., and Tabasum, M. ZAYAN: Disentangled Contrastive Transformer for Tabular Remote Sensing Data. *arXiv preprint arXiv:2604.27606*, 2026c. doi: 10.48550/arXiv.2604.27606.
- Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring Network Structure, Dynamics, and Function Using NetworkX. In *Proceedings of the Python in Science Conference*, pp. 11–15. SciPy, 2008. doi: 10.25080/tcww9851.
- Hahsler, M., Hornik, K., and Buchta, C. Getting Things in Order: An Introduction to the R Package Seriation. *Journal of Statistical Software*, 25:1–34, 2008. doi: 10.18637/jss.v025.i03.
- Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, 2011. doi: 10.1137/090771806.
- Hall, P., Marron, J. S., and Neeman, A. Geometric Representation of High Dimension, Low Sample Size Data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(3):427–444, 2005. doi: 10.1111/j.1467-9868.2005.00510.x.
- Han, S., Yoon, J., Arik, S. O., and Pfister, T. Large Language Models Can Automatically Engineer Features for Few-Shot Tabular Learning. In *International Conference on Machine Learning*, pp. 17454–17479. PMLR, 2024.
- Hanley, J. A. and McNeil, B. J. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143(1):29–36, 1982. doi: 10.1148/radiology.143.1.7063747.
- Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. TabLLM: Few-Shot Classification of Tabular Data with Large Language Models. In *International Conference on Artificial Intelligence and Statistics*, pp. 5549–5581. PMLR, 2023.
- Hernández-Sabat, A., López, A. M., and Diaz-Chito, K. DrivFace. UCI Machine Learning Repository, 2016. doi: 10.24432/C5XC7Q.
- Hinton, G. E. and Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647.
- Hollmann, N., Müller, S., Eggersperger, K., and Hutter, F. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*, 2023.
- Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate Predictions on Small Data with a Tabular Foundation Model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6.
- Holzmüller, D., Grinsztajn, L., and Steinwart, I. Better by Default: Strong Pre-Tuned MLPs and Boosted Trees on Tabular Data. *Advances in Neural Information Processing Systems*, 37:26577–26658, 2024.
- Hotelling, H. Analysis of A Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24(6):417, 1933. doi: 10.1037/h0071325.
- Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *arXiv preprint arXiv:2012.06678*, 2020. doi: 10.48550/arXiv.2012.06678.
- Jain, S. K. and Adjero, D. A. Edge-based Prediction for Lossless Compression of Hyperspectral Images. In *2007 Data Compression Conference (DCC’07)*, pp. 153–162. IEEE, 2007. doi: 10.1109/DCC.2007.36.
- Jefferies, A., Liu, T., Crabbé, J., Imrie, F., and van der Schaar, M. TANGOS: Regularizing Tabular Neural Networks through Gradient Orthogonalization and Specialization. In *The Eleventh International Conference on Learning Representations*, 2023.
- Jethani, N., Sudarshan, M., Aphinyanaphongs, Y., and Ranganath, R. Have We Learned to Explain?: How Interpretability Methods Can Learn to Encode Predictions in Their Interpretations. In *International Conference on Artificial Intelligence and Statistics*, pp. 1459–1467. PMLR, 2021.
- Jiang, X., Margeloiu, A., Simidjievski, N., and Jamnik, M. ProtoGate: Prototype-based Neural Networks with Global-to-local Feature Selection for Tabular Biomedical Data. In *International Conference on Machine Learning*, pp. 21844–21878. PMLR, 2024.
- Jingang, Q., Holzmüller, D., Varoquaux, G., and Le Morvan, M. TabICL: A Tabular Foundation Model for In-Context Learning on Large Data. In *International Conference on Machine Learning*. PMLR, 2025.
- Johnson, W. B. and Lindenstrauss, J. Extensions of Lipschitz Mappings into A Hilbert Space. *Contemporary Mathematics*, 26(189-206):1, 1984.
- Jordan, K. On the Variance of Neural Network Training with respect to Test Sets and Distributions. In *The Twelfth International Conference on Learning Representations*, 2024.

- Jung, S. and Marron, J. PCA Consistency in High Dimension, Low Sample Size Context. *The Annals of Statistics*, 37(6B):4104–4130, 2009. doi: 10.1214/09-AOS709.
- Jurewicz, M. and Derczynski, L. Set Interdependence Transformer: Set-to-Sequence Neural Networks for Permutation Learning and Structure Prediction. In *Proceedings on the International Joint Conferences on Artificial Intelligence (IJCAI-22)*, 2022. doi: 10.24963/ijcai.2022/434.
- Kastellakis, G., Cai, D. J., Mednick, S. C., Silva, A. J., and Poirazi, P. Synaptic Clustering within Dendrites: An Emerging Theory of Memory Formation. *Progress in Neurobiology*, 126:19–35, 2015. doi: 10.1016/j.pneurobio.2014.12.002.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30, 2017.
- Kirchner, J. H. and Gjorgjieva, J. Emergence of Local and Global Synaptic Organization on Cortical Dendrites. *Nature Communications*, 12(1):4005, 2021. doi: 10.1038/s41467-021-23557-3.
- Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by Simulated Annealing. *Science*, 220(4598): 671–680, 1983. doi: 10.1126/science.220.4598.671.
- Kolberg, C., Eggensperger, K., and Pfeifer, N. TabPFN-Wide: Continued Pre-Training for Extreme Feature Counts. In *EurIPS 2025 Workshop: AI for Tabular Data*, 2025.
- Kontschieder, P., Fiterau, M., Criminisi, A., and Buló, S. R. Deep Neural Decision Forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1467–1475, 2015.
- Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., and Dizdarevic, S. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*, 13(2):129–170, 1999. doi: 10.1023/A:1006529012972.
- Levina, E. and Bickel, P. Maximum Likelihood Estimation of Intrinsic Dimension. *Advances in Neural Information Processing Systems*, 17, 2004.
- Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. Feature Selection: A Data Perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2018. doi: 10.1145/3136625.
- Li, S., Harner, E. J., and Adjeroh, D. A. Random KNN Feature Selection-A Fast and Stable Alternative to Random Forests. *BMC Bioinformatics*, 12(1):450, 2011. doi: 10.1186/1471-2105-12-450.
- Liiv, I. Seriation and Matrix Reordering Methods: An Historical Overview. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(2):70–91, 2010. doi: 10.1002/sam.10071.
- Lima, J. R., Santos, V. G. M., and Carvalho, M. A. M. A Δ -Evaluation Function for Column Permutation Problems. *arXiv preprint arXiv:2409.04926*, 2024. doi: 10.48550/arXiv.2409.04926.
- Liu, S. and Ye, H.-J. TabPFN Unleashed: A Scalable and Effective Solution to Tabular Classification Problems. In *Forty-second International Conference on Machine Learning*, 2025.
- Ma, J., Thomas, V., Hosseinzadeh, R., Labach, A., Cresswell, J. C., Golestan, K., Yu, G., Caterini, A. L., and Volkovs, M. TabDPT: Scaling Tabular Foundation Models on Real Data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- Maaten, L. v. d. and Hinton, G. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9(Nov): 2579–2605, 2008.
- Maguire, L. H., Handelman, S. K., Du, X., Chen, Y., Pers, T. H., and Speliotes, E. K. Genome-wide Association Analyses Identify 39 New Susceptibility Loci for Diverticular Disease. *Nature Genetics*, 50(10):1359–1365, 2018. doi: 10.1038/s41588-018-0203-z.
- Mahdessian, D., Cesnik, A. J., Gnann, C., Danielsson, F., Stenström, L., Arif, M., Zhang, C., Le, T., Johansson, F., Schutten, R., et al. Spatiotemporal Dissection of the Cell Cycle with Single-Cell Proteogenomics. *Nature*, 590(7847):649–654, 2021. doi: 10.1038/s41586-021-03232-9.
- Major, G., Larkum, M. E., and Schiller, J. Active Properties of Neocortical Pyramidal Neuron Dendrites. *Annual Review of Neuroscience*, 36(1):1–24, 2013. doi: 10.1146/annurev-neuro-062111-150343.
- Manikandan, H., Jiang, Y., and Kolter, J. Z. Language Models Are Weak Learners. *Advances in Neural Information Processing Systems*, 36:50907–50931, 2023.
- McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. When Do Neural Nets Outperform Boosted Trees on Tabular Data? *Advances in Neural Information Processing Systems*, 36:76336–76369, 2023.
- McInnes, L., Healy, J., and Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*, 2018. doi: 10.48550/arXiv.1802.03426.

- Nam, J., Kim, K., Oh, S., Tack, J., Kim, J., and Shin, J. Optimized Feature Generation for Tabular Data via LLMs with Decision Tree Reasoning. *Advances in Neural Information Processing Systems*, 37:92352–92380, 2024a.
- Nam, J., Song, W., Park, S. H., Tack, J., Yun, S., Kim, J., Oh, K. H., and Shin, J. Tabular Transfer Learning via Prompting LLMs. In *First Conference on Language Modeling*, 2024b.
- Naor, E. and Lindenbaum, O. Hybrid Autoencoders for Tabular Data: Leveraging Model-Based Augmentation in Low-Label Settings. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- NCBI. GEO Accession Viewer: GSE146773 — National Center for Biotechnology Information. <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146773>, 2021. Accessed: 2026-04-22.
- Nemenyi, P. B. *Distribution-Free Multiple Comparisons*. PhD thesis, Princeton University, Princeton, NJ, USA, 1963.
- Neyman, J. and Pearson, E. S. IX. On the Problem of the Most Efficient Tests of Statistical Hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933. doi: 10.1098/rsta.1933.0009.
- Ohlsson, M., Hellmark, T., Bengtsson, A. A., Theander, E., Turesson, C., Klint, C., Wingren, C., and Ekstrand, A. I. Proteomic Data Analysis for Differential Profiling of the Autoimmune Diseases SLE, RA, SS, and ANCA-Associated Vasculitis. *Journal of Proteome Research*, 20(2):1252–1260, 2020. doi: 10.1021/acs.jproteome.0c00657.
- OpenTabular Contributors. deeptab: Tabular Deep Learning Made Simple. <https://github.com/OpenTabular/DeepTab>, 2025. [Online; accessed 2025-07-05].
- Petersen, R. C., Aisen, P. S., Beckett, L. A., Donohue, M. C., Gamst, A. C., Harvey, D. J., Jack Jr, C., Jagust, W. J., Shaw, L. M., Toga, A. W., Trojanowski, J. Q., and Weiner, M. W. Alzheimer’s Disease Neuroimaging Initiative (ADNI): Clinical Characterization. *Neurology*, 74(3): 201–209, 2010. doi: 10.1212/WNL.0b013e3181cb3e25.
- Poirazi, P., Brannon, T., and Mel, B. W. Pyramidal Neuron as Two-Layer Neural Network. *Neuron*, 37(6):989–999, 2003. doi: 10.1016/S0896-6273(03)00149-1.
- Popov, S., Morozov, S., and Babenko, A. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. In *International Conference on Learning Representations*, 2020.
- Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. CatBoost: Unbiased Boosting with Categorical Features. *Advances in Neural Information Processing Systems*, 31, 2018.
- Rosenkrantz, D. J., Stearns, R. E., and Lewis, II, P. M. An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. doi: 10.1137/0206041.
- Rousseeuw, P. J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. doi: 10.1016/0377-0427(87)90125-7.
- Roy, O. and Vetterli, M. The Effective Rank: A Measure of Effective Dimensionality. In *2007 15th European Signal Processing Conference*, pp. 606–610. IEEE, 2007.
- Rubachev, I., Kartashev, N., Gorishniy, Y., and Babenko, A. TabReD: Analyzing Pitfalls and Filling the Gaps in Tabular Deep Learning Benchmarks. In *International Conference on Learning Representations*, volume 2025, pp. 35166–35202, 2025.
- Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., et al. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*, 2022.
- Schiller, J., Major, G., Koester, H. J., and Schiller, Y. NMDA Spikes in Basal Dendrites of Cortical Pyramidal Neurons. *Nature*, 404(6775):285–289, 2000. doi: 10.1038/35005094.
- Seminaroti, M. *Combinatorial Algorithms for the Seriation Problem*. PhD thesis, Tilburg University, 2016.
- Shi, J. and Malik, J. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi: 10.1109/34.868688.
- Shi, R., Gu, H., Ye, H., Dai, Y., Shen, X., and Wang, X. Latte: Transferring LLMs’ Latent-level Knowledge for Few-shot Tabular Learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-25)*, pp. 6173–6181, 2025. doi: 10.24963/ijcai.2025/687.
- Shiloach, Y. A Minimum Linear Arrangement Algorithm for Undirected Trees. *SIAM Journal on Computing*, 8(1): 15–32, 1979. doi: 10.1137/0208002.

- Si, S., Hsieh, C.-J., and Dhillon, I. S. Memory Efficient Kernel Approximation. *Journal of Machine Learning Research*, 18(20):1–32, 2017.
- Simon, N. and Tibshirani, R. A Permutation Approach to Testing Interactions in Many Dimensions. *arXiv preprint arXiv:1206.6519*, 2012. doi: 10.48550/arXiv.1206.6519.
- Somepalli, G., Schwarzschild, A., Goldblum, M., Bruss, C. B., and Goldstein, T. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. In *NeurIPS 2022 First Table Representation Workshop*, 2022.
- Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., and Tang, J. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1161–1170, 2019. doi: 10.1145/3357384.3357925.
- Strawderman, W. E. Sufficient Statistic: Theoretical Background. *Wiley StatsRef: Statistics Reference Online*, 2014. doi: 10.1002/9781118445112.stat05976.
- Tegze, M. and Vlach, M. On the Matrix Permutation Problem. *Zeitschrift für Operations Research*, 30:A155–A159, 1986. doi: 10.1007/BF01919177.
- Thielmann, A. F. and Samiee, S. On the Efficiency of NLP-Inspired Methods for Tabular Deep Learning. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*, pp. 532–539. PMLR, 2024.
- Thielmann, A. F., Kumar, M., Weisser, C., Reuter, A., Säfken, B., and Samiee, S. Mambular: A Sequential Model for Tabular Deep Learning. *arXiv preprint arXiv:2408.06291*, 2024. doi: 10.48550/arXiv.2408.06291.
- Thomas, V., Ma, J., Hosseinzadeh, R., Golestan, K., Yu, G., Volkovs, M., and Caterini, A. Retrieval & fine-tuning for in-context tabular models. *Advances in Neural Information Processing Systems*, 37:108439–108467, 2024.
- Ujfalussy, B. B. and Makara, J. K. Impact of Functional Synapse Clusters on Neuronal Response Selectivity. *Nature Communications*, 11(1):1413, 2020. doi: 10.1038/s41467-020-15147-6.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017.
- Veličković, P., Buesing, L., Overlan, M., Pascanu, R., Vinyals, O., and Blundell, C. Pointer Graph Networks. *Advances in Neural Information Processing Systems*, 33: 2232–2244, 2020.
- Venna, J. and Kaski, S. Neighborhood Preservation in Non-linear Projection Methods: An Experimental Study. In *International Conference on Artificial Neural Networks*, pp. 485–491. Springer Berlin Heidelberg, 2001. doi: 10.1007/3-540-44668-0_68.
- Ver Steeg, G., Harutyunyan, H., Moyer, D., and Galstyan, A. Fast Structure Learning with Modular Regularization. *Advances in Neural Information Processing Systems*, 32, 2019.
- Vinyals, O., Fortunato, M., and Jaitly, N. Pointer Networks. *Advances in Neural Information Processing Systems*, 28, 2015.
- Wang, G., Chen, Y., Chen, H., Fan, X., Wang, J., Li, X., Hu, M., Chang, C.-Y., and Hu, X. Advancing Table Understanding of Large Language Models via Feature Re-ordering. *ACM SIGKDD Explorations Newsletter*, 27(1):112–123, 2025. doi: 10.1145/3748239.3748248.
- Wang, R., Fu, B., Fu, G., and Wang, M. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD’17*, pp. 1–7. ACM, 2017. doi: 10.1145/3124749.3124754.
- Wang, T. and Guan, S.-U. Feature Ordering for Neural Incremental Attribute Learning Based on Fisher’s Linear Discriminant. In *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pp. 507–510. IEEE, 2013. doi: 10.1109/IHMSC.2013.268.
- Wang, T., Guan, S.-U., Man, K. L., and Ting, T. EEG Eye State Identification Using Incremental Attribute Learning with Time-Series Classification. *Mathematical Problems in Engineering*, 2014(1):365101, 2014. doi: 10.1155/2014/365101.
- Wang, T., Guan, S.-U., Ma, J., and Liu, F. Linear Feature Sensibility for Output Partitioning in Ordered Neural Incremental Attribute Learning. In *International Conference on Intelligent Science and Big Data Engineering*, pp. 373–383. Springer, 2015a. doi: 10.1007/978-3-319-23862-3_37.
- Wang, T., Guan, S.-U., Man, K. L., Park, J. H., and Hsu, H.-H. Output Effect Evaluation Based on Input Features in Neural Incremental Attribute Learning for Better Classification Performance. *Symmetry*, 7(1):53–66, 2015b. doi: 10.3390/sym7010053.
- Wang, T., Zhu, X., Guan, S.-U., Man, K. L., and Ting, T. Regression Based on Neural Incremental Attribute Learning with Correlation-based Feature Ordering. In *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on*

- Robotics, Automation and Mechatronics (RAM)*, pp. 109–113. IEEE, 2015c. doi: 10.1109/ICCIS.2015.7274557.
- Wang, Y., Huang, H., Rudin, C., and Shaposhnik, Y. Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization. *Journal of Machine Learning Research*, 22(201):1–73, 2021.
- Wen, X., Zhang, H., Zheng, S., Xu, W., and Bian, J. 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*, pp. 3323–3333, 2024. doi: 10.1145/3637528.3671975.
- Wu, X., Liu, X., Li, W., and Wu, Q. Improved Expressivity through Dendritic Neural Networks. *Advances in Neural Information Processing Systems*, 31, 2018.
- Yamada, Y., Lindenbaum, O., Negahban, S., and Kluger, Y. Feature Selection Using Stochastic Gates. In *International Conference on Machine Learning*, pp. 10648–10659. PMLR, 2020.
- Yan, J., Zheng, B., Xu, H., Zhu, Y., Chen, D., Sun, J., Wu, J., and Chen, J. Making Pre-Trained Language Models Great on Tabular Prediction. In *The Twelfth International Conference on Learning Representations*, 2024.
- Yan, J., Chen, J., Hu, C., Zheng, B., Hu, Y., Sun, J., and Wu, J. Small Models are LLM Knowledge Triggers for Medical Tabular Prediction. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Yang, J., Lindenbaum, O., and Kluger, Y. Locally Sparse Neural Networks for Tabular Biomedical Data. In *International Conference on Machine Learning*, pp. 25123–25153. PMLR, 2022a.
- Yang, T., Wang, Y., Yue, Z., Yang, Y., Tong, Y., and Bai, J. Graph Pointer Neural Networks. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 36, pp. 8832–8839, 2022b. doi: 10.1609/aaai.v36i8.20864.
- Yata, K. and Aoshima, M. PCA Consistency for Non-Gaussian Data in High Dimension, Low Sample Size Context. *Communications in Statistics-Theory and Methods*, 38(16-17):2634–2652, 2009. doi: 10.1080/03610910902936083.
- Ye, H., Li, J., Zhao, H., Guo, D., and Chang, Y. LLM Meeting Decision Trees on Tabular Data. In *Advances in Neural Information Processing Systems*, 2025a.
- Ye, H.-J., Liu, S.-Y., and Chao, W.-L. H. A Closer Look at TabPFN v2: Understanding Its Strengths and Extending Its Capabilities. *Advances in Neural Information Processing Systems*, 38:135605–135637, 2025b.
- Ye, H.-J., Yin, H.-H., Zhan, D.-C., and Chao, W.-L. Revisiting Nearest Neighbor for Tabular Data: A Deep Tabular Baseline Two Decades Later. In *The Thirteenth International Conference on Learning Representations*, 2025c.
- Yoon, J., Jordon, J., and Van der Schaar, M. INVASE: Instance-Wise Variable Selection Using Neural Networks. In *International Conference on Learning Representations*, 2019.
- Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep Sets. *Advances in Neural Information Processing Systems*, 30, 2017.
- Zhang, T., Zhang, Z. A., Fan, Z., Luo, H., Liu, F., Liu, Q., Cao, W., and Jian, L. OpenFE: Automated Feature Generation with Expert-Level Performance. In *International Conference on Machine Learning*, pp. 41880–41901. PMLR, 2023.

This supplementary document supports our main paper *GOTabPFN: From Feature Ordering to Compact Tokenization for Tabular Foundation Models on High-Dimensional Data* (Submitted to the Forty-Third International Conference on Machine Learning (ICML) 2026). Specifically, it includes:

- Detailed Related Work in Sec. [A](#)
- Theoretical Characterization of GO-LR: Complexity and TSP Connections in Sec. [B](#)
- NSC Variants and NSC as a Shared Piecewise Pooling Operator in Sec. [C](#)
- NSC as a Structured Dimensionality Reduction Layer in Sec. [D](#)
- Why Feature Ordering? Local Neighborhoods Enable Structure-Aware Compression in Sec. [E](#)
- Feature Ordering - When to Use? Through the Lens of Locality in Sec. [F](#)
- Detailed Comparative Results in Sec. [G](#)
- GOTabPFN Hyperparameters in Sec. [H](#)
- Statistical Significance Analysis in Sec. [I](#)
- Additional Ablation Analysis in Sec. [J](#)
- Representation Quality via t-SNE in Sec. [K](#)
- Inference Level Ablation on Calibration and Robustness in Sec. [L](#)
- Sanity and Stress Diagnostics in Sec. [M](#)
- Additional Reliability and Interpretability Diagnostics in Sec. [N](#)
- Theory-Inspired Representation Diagnostics in Sec. [O](#)
- OOD and Local Sensitivity Diagnostics in Sec. [P](#)
- Deployment-Oriented Triage Diagnostics in Sec. [Q](#)
- Extension beyond TabPFN in Sec. [R](#)
- TabPFN Seed Sensitivity in Sec. [S](#)
- Additional Clarifications in Sec. [T](#)

A. Detailed Related Work

Tabular Models. Recent tabular deep learning has also advanced rapidly beyond early attention/MLP baselines. In-context learning-based model such as TabICL ([Jingang et al., 2025](#)) aim to provide strong out-of-the-box performance (especially in low-data regimes), while methods like TabR ([Gorishniy et al., 2024](#)) and TabM ([Gorishniy et al., 2025](#)) improve classical deep tabular pipelines via nearest-neighbor augmentation or parameter-efficient ensembling. Hybrid and low-label settings are explored by TANDEM ([Naor & Lindenbaum, 2025](#)), and context optimization for scalable prior-fitted networks is studied in TuneTables ([Feuer et al., 2024](#)). TabDPT pre-trains a row-token tabular foundation model by combining retrieval-based in-context learning with self-supervised column-masking on large-scale real datasets, enabling strong generalization to unseen tasks without per-dataset tuning ([Ma et al., 2025](#)). At the same time, strong “simple” baselines remain highly competitive: carefully pre-tuned MLPs such as Real-MLP ([Holzmüller et al., 2024](#)) can be surprisingly hard to beat, and gradient-boosted decision trees, especially XGBoost ([Chen & Guestrin, 2016](#)), LightGBM ([Ke et al., 2017](#)), and CatBoost ([Prokhorenkova et al., 2018](#)) continue to thrive as robust, high-performing workhorses on many tabular benchmarks.

Feature Selection-based HDLSS Specific Models. Feature-selection methods are particularly relevant for HDLSS tabular learning, where selecting a compact, informative subset of features is often more critical than scaling model capacity. ProtoGate ([Jiang et al., 2024](#)) addresses this regime with prototype-guided gating that selects features by aligning samples with class prototypes, improving robustness when samples are scarce. Several works perform instance-wise or differentiable

subset selection: INVASE (Yoon et al., 2019) learns a sample-dependent feature selector trained with prediction and selection regularization, while STG (Yamada et al., 2020) uses stochastic gates to enable end-to-end feature selection with sparsity control. LSPIN/LLSPIN (Yang et al., 2022a) further emphasizes interpretable, per-instance gating for tabular inputs. Complementarily, L2X (Chen et al., 2018) and REAL-X (Jethani et al., 2021) learn differentiable selection/explanation mechanisms that identify a small set of features sufficient for prediction, offering a principled way to trade accuracy for sparsity and interpretability in low-sample settings. RKNN-FS (Li et al., 2011) addresses HDLSS learning through feature selection, using a random k -nearest neighbor (RKNN) strategy to identify informative features before prediction.

LLMs for Tabular Data. Motivated by general-purpose LLMs (Brown et al., 2020; Achiam et al., 2023; Guo et al., 2024; 2025), recent work adapts them to tabular prediction via fine-tuning (e.g., LIFT (Dinh et al., 2022) on GPT-3 (Brown et al., 2020), TabLLM (Hegselmann et al., 2023) on T0 (Sanh et al., 2022)) and tabular-centric pretraining for transfer/instruction following (e.g., TP-BERTa (Yan et al., 2024), GTL (Wen et al., 2024)). Other lines use prompting or hybrid pipelines, including correlated-text semi-supervision (P2T (Nam et al., 2024b)), weak-learner boosting (Summary (Manikandan et al., 2023)), feature synthesis (FeatLLM (Han et al., 2024)), synergy learning with tabular backbones (SERSAL (Yan et al., 2025)), LLM-guided rule/feature generation (Nam et al., 2024a; Zhang et al., 2023), rule refinement without LLM fine-tuning (Ye et al., 2025a), metadata-driven distillation (Shi et al., 2025), and order-bias mitigation (ROTATOR-LLM (Wang et al., 2025)). Despite encouraging efforts, current tabular LLMs remain poorly suited to HDLSS and very high-dimensional tables, since attention-based architectures incur quadratic cost in the token/feature dimension.

TabPFN and Variants. TabPFN introduced the idea of a tabular foundation model that performs in-context learning for small tabular classification problems (Hollmann et al., 2023). This was later substantially extended and validated at scale in TabPFN v2 (Hollmann et al., 2025). Recent follow-ups push accuracy and scalability along multiple axes: TabPFN-2.5 advances the state of the art in tabular foundation models (Grinsztajn et al., 2025), TabPFN Unleashed (BETA) targets practical scalability and effectiveness on broader settings (Liu & Ye, 2025), and TabPFN-Wide explores continued pre-training for extreme feature counts (Kolberg et al., 2025). Orthogonally, LoCalPFN investigates retrieval and fine-tuning mechanisms for in-context tabular models (Thomas et al., 2024). Our work is complementary to these efforts: rather than modifying TabPFN itself, we develop a stable, structure-aware compression interface that enables TabPFN-style predictors to operate reliably in HDLSS regimes with $m \gg n$ and very large feature counts.

Other Models. MLP-PLR is a strong MLP baseline for tabular data that replaces raw continuous inputs with learnable piecewise-linear (PLR) feature embeddings, improving expressivity and performance over standard MLPs (Gorishniy et al., 2022). Other tabular models such as TabNet (Arik & Pfister, 2021), TabTransformer (Huang et al., 2020)/FT-Transformer (Gorishniy et al., 2021), SAINT (Somepalli et al., 2022), AutoInt (Song et al., 2019), and interaction-centric networks (DeepFM (Guo et al., 2017), DCN (Wang et al., 2017)) differ primarily in how they represent columns and capture cross-feature dependencies: TabNet performs step-wise attentive feature selection via sparse masks for interpretability, while transformer families tokenize columns (especially categorical features) and use self-attention to model contextual interactions, with FT-Transformer providing a lighter mixed-type variant; SAINT further strengthens tabular transformers via augmentation and contrastive/self-supervised pretraining, and AutoInt targets high-order interactions directly through attention. A complementary line treats tabular inputs as feature sequences, including ordering-based approaches (TabSeq (Habib et al., 2024)) and recurrent processing (TabulaRNN (Thielmann & Samiee, 2024)), which impose position-aware inductive bias but may depend on the quality of the chosen order. More recent sequence backbones replace attention with state-space modeling e.g., Mambular (Thielmann et al., 2024), MambaTab (Ahamed & Cheng, 2024), and hybrids like MambAttention (Thielmann & Samiee, 2024) leveraging Mamba-style selective state-space layers for efficient long-range dependency modeling. Tree-inspired inductive biases remain competitive through differentiable ensembles (NODE (Popov et al., 2020), ENODE (OpenTabular Contributors, 2025), NDTF (Kontschieder et al., 2015)) that emulate decision trees with end-to-end training, alongside classical ensembles (Random Forest, AdaBoost, GBM) that provide strong, stable baselines. Finally, representation and regularization advances such as DANets (Chen et al., 2022), ResNetTabular (OpenTabular Contributors, 2025), CategoryEmbedding (Gorishniy et al., 2022), TANGOS (Jeffares et al., 2023), and metric/contrastive methods (ModernNCA (Ye et al., 2025c), Tromp (Chen et al., 2023)) improve robustness and optimization on noisy, small-sample settings, while standard baselines (Naive Bayes, KNN, SVM, Decision Tree, Lasso, MLP, 1-D CNN) remain important reference points due to their interpretability and well-characterized trade-offs. GeoAggregator (Deng et al., 2025) and ZAYAN (Habib et al., 2026c) represent recent domain-specific geospatial tabular deep learning models, with GeoAggregator targeting spatially aware geospatial regression and ZAYAN focusing on feature-level contrastive learning for tabular remote sensing and environmental data.

Feature Ordering and Permutation Sensitivity. Mambular (Thielmann et al., 2024) first highlighted that random permutations of tabular feature order can induce brittleness in predictive performance, even under fixed seeds. Concurrently, TabSeq (Habib et al., 2024) explicitly introduced a feature ordering algorithm for tabular deep learning, establishing

SOTA tabular models grouping as per tuning requirement

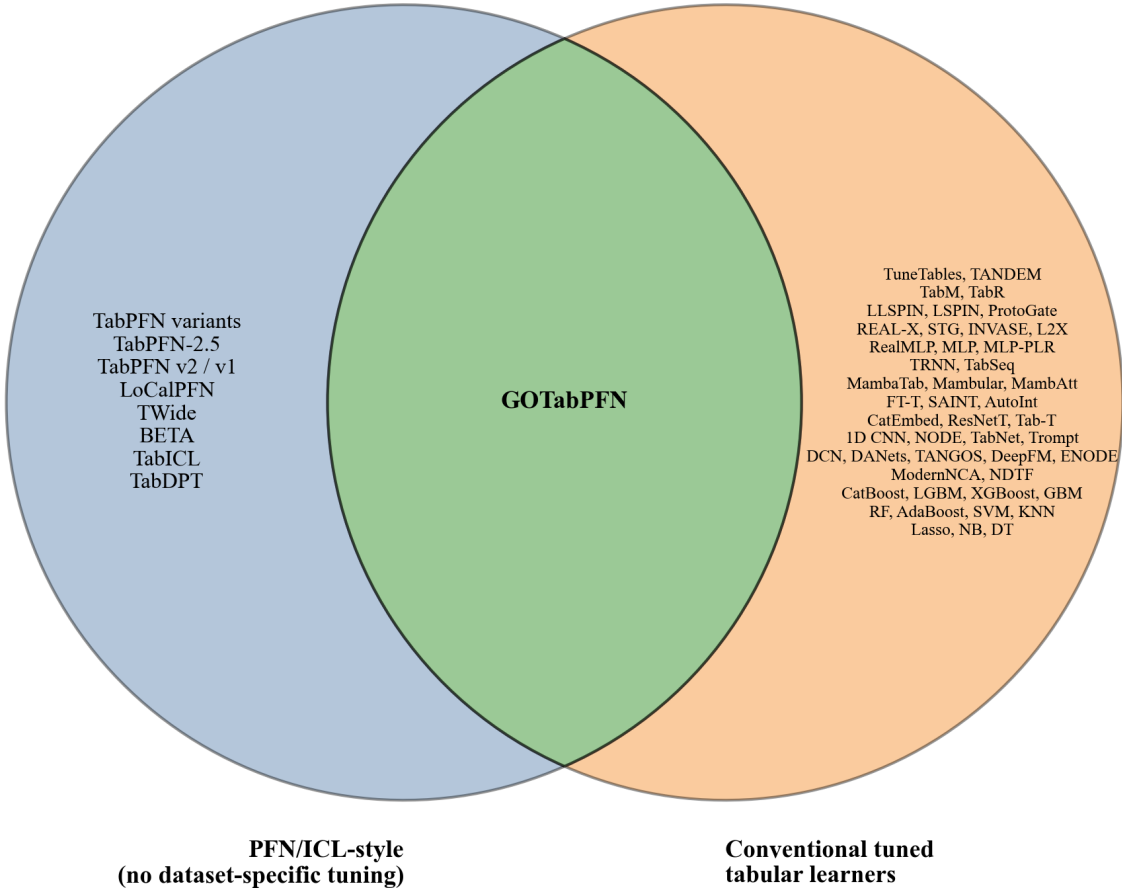


Figure A.1. **GOTabPFN tuning regime.** GOTabPFN sits between frozen PFN/ICL-style tabular foundation models and fully tuned tabular learners: only the GO-LR+NSC front-end is tuned, while TabPFN-2.5 remains frozen. See Appendix T for more clarifications.

column permutation as a learnable design choice rather than a nuisance factor. Later ROTATOR-LLM (Wang et al., 2025) extended this direction by studying feature ordering for LLM-based tabular inference. DynaTab (Habib et al., 2026b) systematically studied when feature ordering matters in high-dimensional tabular learning, introducing an Intrinsic Dimensionality Factor (IDF) and feature-to-sample ratio $\rho = m/n$ based categorization of dataset regimes. It proposed a neuroscience-inspired Dynamic Feature Ordering (DFO) algorithm and showed that sequence-sensitive backbones such as Transformers, LSTMs, denoising autoencoders, and SSM-style models can benefit from adaptive ordering. However, its use of vanilla sequence backbones incurs substantial memory and runtime costs, motivating more compact ordering-aware tokenization and compression strategies. We use Fig. A.1 to position GOTabPFN within this landscape according to hyperparameter tuning requirements for tabular models: unlike PFN/ICL-style models that require no dataset-specific tuning and conventional tabular learners that are fully tuned, GOTabPFN occupies the middle regime by tuning only the GO-LR+NSC front-end while retaining a frozen TabPFN-2.5 backbone.

B. Theoretical Characterization of GO-LR: Complexity and TSP Connections

B.1. GO-LR as a TSP-Style Initialization with Local Refinement.

GO-LR does not solve MinLA exactly; instead, our implementation constructs a permutation by greedy seriation over a pairwise dissimilarity matrix. The initialization coincides with a nearest-neighbor heuristic for a TSP-path objective defined on a complete graph, and GO-LR then locally refines the resulting permutation under the dispersion objective.

Definition B.1 (TSP-path Objective on a Complete Graph). Given a complete weighted graph $\mathcal{K} = (V, \binom{V}{2}, d)$ with edge

Table B.1. **GO-LR vs. classic metaheuristics on Colon.** Lower runtime, TSP cost, and MinLA are better; higher accuracy is better. Accuracy is reported over 5×5 CV. Christofides and Simulated Annealing are implemented using NetworkX (Hagberg et al., 2008).

Ordering Method _{+NSC}	Runtime	TSP ↓	MinLA ↓	Accuracy ↑
GO-LR (ours)	10.07	21958.78	1.4743e10	0.8818 ± 0.1005
Simulated Annealing	15.01	11712.75	1.4803e10	0.8405 ± 0.0944
Genetic Algorithm	206.59	11712.75	1.4803e10	0.8405 ± 0.0944
Ant Colony Optimization	1501.44	11792.08	1.4760e10	0.8595 ± 0.1000
Christofides	1424.83	11715.06	1.4994e10	0.8364 ± 0.0911

weights $d_{ij} \geq 0$, we define the path cost of a permutation $\sigma = (\sigma_1, \dots, \sigma_m)$ by Eq. 8, repeated below as Eq. 26 for brevity.

$$\text{PathCost}(\sigma) = \sum_{t=1}^{m-1} d_{\sigma_t, \sigma_{t+1}} \quad (26)$$

Lemma B.2 (Nearest-Neighbor Heuristic). *The greedy procedure “start at $\arg \min_i \sum_j d_{ij}$ and repeatedly append the nearest unvisited node” returns a Hamiltonian path in $\mathcal{K}$ and is a standard nearest-neighbor heuristic for minimizing Eq. (8) (Rosenkrantz et al., 1977).*

Proof sketch. At each step exactly one new unvisited node is appended; thus the resulting sequence visits each node exactly once and forms a Hamiltonian path. The choice “nearest unvisited” is precisely the nearest-neighbor rule for minimizing Eq. (8). $\square$

Theorem B.3 (The Initialization Step of GO-LR Can Be Used as a TSP-path Heuristic). *For any complete weighted graph $\mathcal{K}$, the GO-LR local seriation rule (nearest-neighbor) can be used as a TSP-path heuristic that outputs a Hamiltonian path σ for $\mathcal{K}$. By Lemma B.2, the output is a Hamiltonian path.*

Proof sketch. Given $\mathcal{K}$, run the greedy nearest-neighbor construction on its distance matrix $[d_{ij}]$. By Lemma B.2, the output is a Hamiltonian path and corresponds to a permutation σ . $\square$

Remark B.4 (What is (and is not) “equivalent to TSP” here). GO-LR’s local objective Eq. (2) is MinLA, while the nearest-neighbor constructor optimizes a TSP-path surrogate for initialization, while GO-LR subsequently refines the ordering using the dispersion objective in Eq. (2). Empirically, the surrogate tends to produce low-dispersion permutations under Eq. (2) when d_{ij} is chosen to be compatible with w_{ij} (e.g., monotone transforms or neighborhood sparsification).

GO-LR vs. classic metaheuristics. To assess whether GO-LR is overly limited by its greedy construction, we replace GO-LR with stronger stochastic/metaheuristic orderings while keeping the downstream NSC + TabPFN-2.5 pipeline fixed. On Colon, although Simulated Annealing (Kirkpatrick et al., 1983), Genetic Algorithm (Larranaga et al., 1999), Ant Colony Optimization (Dorigo & Gambardella, 2002), and Christofides-based ordering (Christofides, 2022) often achieve lower TSP-style surrogate cost, they do not improve downstream accuracy; GO-LR obtains the best 5×5 CV accuracy (0.8818 ± 0.1005), the lowest runtime (10.07s), and the best MinLA-style dispersion objective. We observe the same trend on the larger Cell Cycle (Mahdessian et al., 2021; NCBI, 2021) RNA-seq dataset ($n = 1067, m = 42728$): GO-LR+NSC+TabPFN-2.5 achieves 79.94 ± 2.53 accuracy, 92.36 ± 1.36 AUC, and 79.95 ± 2.51 macro-F1, compared to 76.45 ± 2.29 , 92.76 ± 1.10 , and 76.42 ± 2.29 for Simulated Annealing+NSC+TabPFN-2.5. Consistent with our theory, GO-LR attains the lower MinLA cost on Cell Cycle (8.14×10^{11} vs. 8.51×10^{11}), while Simulated Annealing attains the lower TSP-path cost; this is expected, since GO-LR is designed to optimize the MinLA-style dispersion objective rather than a TSP surrogate alone. These results suggest that optimizing a TSP surrogate more aggressively does not necessarily yield better feature orderings for NSC+TabPFN-2.5; GO-LR is better aligned with the MinLA-style objective and provides a stronger accuracy-efficiency tradeoff in practice.

On global optimality. GO-LR does not guarantee a globally optimal ordering, since the underlying MinLA-style feature ordering problem is combinatorial and NP-hard. Instead, it is a structured approximation strategy: clustering induces local feature graphs, NNPath provides an efficient initialization, and local refinement explicitly reduces the MinLA-style dispersion objective. Empirically, replacing GO-LR with stronger metaheuristics such as Simulated Annealing, Genetic Algorithm, Ant Colony Optimization, and Christofides-based ordering does not improve downstream NSC+TabPFN-2.5 performance. On both Colon and the larger Cell Cycle transcriptomic dataset, GO-LR achieves better downstream accuracy

and lower MinLA cost despite some alternatives attaining lower TSP-style surrogate cost, suggesting that GO-LR is better aligned with the objective that matters for ordering-aware compression and prediction.

GO-LR cluster-size sensitivity. We ablate the GO-LR cluster size k on Colon while fixing all other hyperparameters to the best tuned configuration. As shown in Table B.2, performance peaks at $k = 10$, reproducing the best Colon result (88.18 ± 10.05) in a fresh run. The trend is non-monotonic: small k values appear too coarse to capture local sample heterogeneity, while overly large k values fragment the data into noisier local graphs, weakening the aggregated global ordering. This supports using a moderate cluster size as the best trade-off.

Table B.2. **GO-LR cluster-size sensitivity on Colon.** All settings except k are fixed to the best tuned configuration. Values are mean accuracy with subscripted standard deviation over 5×5 CV.

k	3	4	5	7	10	12	15
Acc.	82.72 ± 10.69	84.38 ± 9.82	83.95 ± 9.54	84.33 ± 8.95	88.18 ± 10.05	83.72 ± 10.16	82.69 ± 11.08

C. NSC Variants and NSC as a Shared Piecewise Pooling Operator

C.1. Intrinsic-Dimension Rules and Budgets for NSC Variants

For completeness we describe the intrinsic-dimension estimators and budget rules used by the remaining NSC variants. Recall that $\tilde{X} \in \mathbb{R}^{n \times m}$ is the standardized training matrix and we get Eq. 27 which denotes its covariance (or correlation) matrix with nonzero eigenvalues $\{\lambda_i\}_{i=1}^r$, $r \leq \min(n, m)$.

$$\Sigma = \frac{1}{n-1} \tilde{X}^\top \tilde{X} \in \mathbb{R}^{m \times m} \quad (27)$$

Effective-rank intrinsic dimension. Besides the PCA cumulative-variance rule in Eqs. (18)-(20), we also consider an effective-rank estimate (Roy & Vetterli, 2007; Halko et al., 2011). We get Eq. 28 and define Eq. 29 with a small constant $\epsilon > 0$ for numerical stability. Some variants set $\hat{d} = d_{\text{eff}}$ instead of $\hat{d}_{\text{PCA}}(\tau)$.

$$p_i = \frac{\lambda_i}{\sum_{j=1}^r \lambda_j} \quad (28)$$

$$d_{\text{eff}} = \exp\left(-\sum_{i=1}^r p_i \log(p_i + \epsilon)\right) \quad (29)$$

IDF-based budget modulation. We define the IDF (Habib et al., 2026b) in Eq. 30 which compares intrinsic and ambient dimensionality. An IDF-based budget rule optionally used in NSC and NSC-P is defined by Eq. 31 where γ , $M_{\min}$, and $M_{\max}$ control redundancy allowance and token-budget bounds.

$$\text{IDF} = \frac{\hat{d}}{m}, \quad \hat{d} \in \{d_{\text{eff}}, \hat{d}_{\text{PCA}}(\tau)\} \quad (30)$$

$$M = \text{clip}\left(\lceil (1 + \beta(1 - \text{IDF})) \hat{d} \rceil, M_{\min}, \min(M_{\max}, m)\right), \quad \beta \in [0, 1] \quad (31)$$

Variant summary.

- **NSC:** uses a fixed, user-chosen M (no intrinsic-dimension estimate).
- **NSC-P:** chooses $\hat{d}$ via either d_{eff} or $\hat{d}_{\text{PCA}}(\tau)$ and then applies an IDF- or γ -based rule such as Eq. 32

$$M = \text{clip}(\lceil \gamma \hat{d} \rceil, M_{\min}, \min(M_{\max}, m)) \quad (32)$$

- **NSC-SP:** uses a fixed M but applies SegPCA pooling (Sec. 3.2).
- **NSC-pSP:** uses the PCA-based rule in Eqs. (20)-(21) (described in the main text).

Computational note. In HDLSS regimes ($m \gg n$), we avoid forming a full eigen-decomposition of the $m \times m$ matrix Σ in Eq. (27) by computing the nonzero spectrum via the $n \times n$ Gram matrix $\tilde{X}\tilde{X}^\top$ or using randomized methods (Halko et al., 2011). The resulting eigenvalues are then re-used in the intrinsic-dimension rules above.

Proposition C.1 (NSC as a Shared Piecewise Pooling Operator). *Let Π^* be a fixed global feature permutation and let $\{\mathcal{S}_t\}_{t=1}^M$ be the contiguous segments defined by NSC. The Neuro-Inspired Subunit Compression defines a mapping in Eq. 33 which is given by Eq. 34 where the same pooling function g_θ is shared across all segments.*

$$F_{\text{NSC}} : \mathbb{R}^m \rightarrow \mathbb{R}^{M \times d} \quad (33)$$

$$F_{\text{NSC}}(x) = \left(g_\theta(\psi(x_{\mathcal{S}_1}^\Pi)), \dots, g_\theta(\psi(x_{\mathcal{S}_M}^\Pi)) \right) \quad (34)$$

Then F_{NSC} is a piecewise pooling operator with the following properties: (1) **Locality**. Each output token depends only on a contiguous subset of the ordered feature axis; (2) **Parameter sharing**. The number of trainable parameters is independent of m and depends only on g_θ ; and (3) **Linear complexity**. For fixed pooling depth, F_{NSC} can be evaluated in $O(m)$ time and $O(Md)$ space per sample.

Proposition C.1 shows that NSC induces a structured, order-aware compression operator that preserves locality while remaining scalable to extreme HDLSS regimes.

Proof sketch. By construction, the ordered feature vector x^Π is partitioned into M disjoint contiguous segments whose union covers $\{1, \dots, m\}$. Each segment is processed independently by the same pooling function g_θ , establishing locality and parameter sharing. Since each feature appears in exactly one segment and g_θ has constant depth, the total number of operations scales linearly with m , yielding $O(m)$ time complexity. The output representation stores M vectors of dimension d , giving $O(Md)$ space complexity. $\square$

D. NSC as a Structured Dimensionality Reduction Layer

In this section we view the NSC (Section 3.2) as an explicit Dimensionality Reduction (DR) layer that maps high-dimensional GO-LR-ordered features into a low-dimensional latent space. We first formalize NSC as a structured DR map, then relate it to classical DR methods, and finally outline and instantiate an empirical protocol comparing NSC to Principal Component Analysis (PCA) (Hotelling, 1933), Random Projections (RP) (Johnson & Lindenstrauss, 1984), Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018), Pairwise Controlled Manifold Approximation (PaCMAP) (Wang et al., 2021), and Autoencoders (AE) (Hinton & Salakhutdinov, 2006) quantitatively.

D.1. NSC as a Structured DR Map

Recall that GO-LR produces a global feature permutation Π^* that approximately solves a weighted MinLA problem (Section 3.1), bringing highly correlated or low-dissimilarity features into local neighborhoods along the ordered axis. NSC then segments this ordered axis into M contiguous subunits $\{\mathcal{S}_t\}_{t=1}^M$ and applies a shared pooling operator $g_\theta \circ \psi$ to each segment (Eqs. 12-17). For a sample $x \in \mathbb{R}^m$ we again write Eq. 35 and define segments $\mathcal{S}_t \subset \{1, \dots, m\}$ that partition the ordered axis. The t -th meta-feature is defined by Eq. 36 yielding a meta-feature sequence $Z(x) = (z_1, \dots, z_M) \in \mathbb{R}^{M \times d}$ (Eqs. 16-17). In our implementation we often set $d = 1$ (scalar tokens), and flatten $Z(x)$ into a vector in $\mathbb{R}^M$. Formally, NSC again defines a mapping in Eq. 37 as summarized in Proposition C.1. When $d = 1$ and we flatten across segments, this reduces to a DR map by Eq. 38. Thus NSC acts as a structured DR layer whose output dimensionality $M \ll m$ is explicitly controlled by the meta-feature budget.

$$x^\Pi = (x_{\Pi^*(1)}, \dots, x_{\Pi^*(m)}) \quad (35)$$

$$z_t = g_\theta(\psi(u_t)), \quad u_t := x_{\mathcal{S}_t}^\Pi \quad (36)$$

$$F_{\text{NSC}} : \mathbb{R}^m \rightarrow \mathbb{R}^{M \times d}, \quad F_{\text{NSC}}(x) = \left(g_\theta(\psi(x_{\mathcal{S}_1}^\Pi)), \dots, g_\theta(\psi(x_{\mathcal{S}_M}^\Pi)) \right) \quad (37)$$

$$\Phi_{\text{NSC}} : \mathbb{R}^m \rightarrow \mathbb{R}^M, \quad \Phi_{\text{NSC}}(x) = \text{flatten}(F_{\text{NSC}}(x)) \quad (38)$$

Structured locality. Unlike global DR methods such as PCA (Hotelling, 1933), Φ_{NSC} has built-in locality: each coordinate of the compressed representation depends only on a contiguous block of GO-LR-ordered features. In particular, the t -th coordinate of $\Phi_{\text{NSC}}(x)$ is a pooled summary of the subunit $u_t = x_{\mathcal{S}_t}^\Pi$, where $\mathcal{S}_t$ corresponds to a locally redundant neighborhood in the GO-LR ordering. This induces a piecewise pooling structure: different coordinates of the latent vector correspond to different ordered blocks rather than global linear mixtures of all features.

Linear and nonlinear regimes. If both ψ and g_θ are linear maps, NSC reduces to a structured linear DR operator. Writing $\psi(u_t) = A_t u_t$ and $g_\theta(v) = w^\top v$, we obtain Eq. 39 where P_Π is the permutation matrix for Π^* and $P_{\mathcal{S}_t}$ selects segment indices. Stacking across t yields a linear map $W_{\text{NSC}}x$ with a structured block-sparse pattern aligned with the ordered segments. When ψ or g_θ include nonlinear statistics (e.g., quantiles, skewness, kurtosis, shallow MLP), NSC becomes a local nonlinear DR layer with the same segmentation structure.

$$z_t = w^\top A_t u_t = w^\top A_t x_{\mathcal{S}_t}^\Pi = w^\top A_t P_{\mathcal{S}_t} P_\Pi x \quad (39)$$

Intrinsic dimension-aware budget. Section 3.2 ties the meta-feature budget M to an estimate of intrinsic dimensionality $\hat{d}$ via effective rank (Eqs. 27-29). Our default rule (Eq. 21) sets Eq. 40 or optionally uses the IDF-based budget (Eq. 31) to modulate M by redundancy. In HDLSS regimes with $m \gg n$, we estimate $\hat{d}$ via the $n \times n$ Gram matrix or randomized methods to avoid forming a full $m \times m$ covariance. In all cases, M scales with intrinsic rather than ambient dimension, so NSC compresses more aggressively when the effective rank is low.

$$M = \text{clip}(\lceil 2\hat{d} \rceil, 32, \min(512, m)) \quad (40)$$

D.2. Relation to Classical Dimensionality Reduction

NSC is conceptually related to, but distinct from, standard DR techniques.

PCA and RP. PCA (Hotelling, 1933) computes a global linear projection $x \mapsto U_k^\top x$ that optimizes variance preservation in $\mathbb{R}^k$. RP maps $x \mapsto Rx$ with a dense random matrix R and approximately preserves pairwise distances by Johnson–Lindenstrauss guarantees (Johnson & Lindenstrauss, 1984). Both treat coordinates symmetrically and mix all features into each latent dimension. By contrast, NSC first reorders features via GO-LR so that correlated variables are neighbors, then pools each contiguous neighborhood into a meta-feature. This yields:

- **Local support:** Each latent coordinate depends on a small, interpretable block of features rather than all m .
- **Order-aware pooling:** GO-LR ensures that each block tends to group features with small dispersion under the MinLA objective, so pooled statistics are computed over structurally coherent neighborhoods.
- **Flexible statistics:** NSC variants can incorporate robust or higher-order statistics via ψ without changing the dimensionality M . NSC variants also inherit the power of PCA for IDF or post segmentation summarization.

In the special case of linear ψ and g_θ , NSC is a constrained linear DR method whose projection matrix has a block structure aligned with the GO-LR ordering, whereas PCA uses a dense orthogonal mixing and RP uses an unstructured dense matrix.

Autoencoders. Autoencoders (AE) (Hinton & Salakhutdinov, 2006) learn a parametric encoder-decoder pair (f_ϕ, h_ψ) with a low-dimensional bottleneck $z \in \mathbb{R}^k$ optimized to minimize reconstruction error. While AE can capture nonlinear structure, they require training, are sensitive to sample size, and their bottleneck dimensions often entangle information from all features. NSC can be seen as a deterministic, data-dependent encoder with no decoder: it compresses features into M meta-features using shared, shallow pooling and no reconstruction objective. In HDLSS regimes, NSC has two advantages: (i) it does not require fitting a heavy parametric model on small n , and (ii) its pooling structure is constrained by GO-LR ordering, which reduces overfitting and enforces a biologically motivated inductive bias.

UMAP. UMAP (McInnes et al., 2018) is a nonlinear manifold-learning method that constructs a neighborhood graph and optimizes a low-dimensional embedding to preserve local topological structure. Although this makes UMAP effective for visualization and neighborhood preservation, the learned coordinates are global embedding dimensions whose relation to the original features is indirect. In contrast, NSC preserves an explicit feature-axis interpretation: each meta-feature is computed from a contiguous GO-LR-induced neighborhood, making the compression more directly tied to feature locality and block structure in HDLSS settings.

PaCMAP. PaCMAP (Wang et al., 2021) is another nonlinear DR method that balances nearby, mid-near, and far pair relationships to preserve both local and global geometry in the embedded sample space. Like UMAP, it learns low-dimensional coordinates over samples rather than constructing interpretable feature-block summaries. NSC instead operates along the feature dimension: after GO-LR induces a locality-aware ordering, NSC pools contiguous feature segments into meta-features, which is better aligned with our goal of preserving order-induced feature neighborhoods for downstream TabPFN-style prediction.

D.3. Empirical Behavior on HDLSS Benchmarks

We instantiate the protocol below on block structured synthetic HDLSS datasets, using $M = 32$ as an aggressive DR setting. For each dataset we compare NSC (GO-LR ordering + Optuna (Akiba et al., 2019) tuned segmentation, descriptor, and pooling) with PCA, AE, UMAP, PaCMAP, and RP using:

- Linear-probe accuracy (Logistic Regression) (Cox, 1958),
- k NN accuracy in latent space (Cover & Hart, 1967),
- Silhouette (Rousseeuw, 1987) and Davies–Bouldin (Davies & Bouldin, 1979) scores for class separability.

D.4. Block-Structured Synthetic HDLSS Model

To better understand when NSC should dominate classical DR, we also study a synthetic family of HDLSS distributions with explicit block structure (Ver Steeg et al., 2019). This controlled ablation is inspired by the block-subunit HDLSS modeling philosophy of BStabDiff (Habib et al., 2026a), which views high-dimensional tabular data as arising from latent feature blocks governed by shared subunit factors. In our setting, we adapt this idea only as a diagnostic synthetic benchmark: features are generated from block-correlated latent factors, then randomly permuted, allowing us to test whether GO-LR + NSC can recover useful local neighborhoods for compression and downstream prediction. Concretely, we construct datasets where m features are partitioned into B contiguous blocks, each governed by a shared latent factor plus Gaussian noise (Devijver & Gallopin, 2018). Class information is injected via mean shifts on a small subset of blocks, while the remaining blocks are pure noise. After generation, we randomly permute feature indices so that class-informative blocks are no longer contiguous in the raw feature space (Simon & Tibshirani, 2012). Under this model, GO-LR with a correlation-based metric tends to recover an ordering that approximately re-groups highly correlated features into contiguous neighborhoods, effectively reconstructing the underlying blocks. NSC then segments along this ordering and pools each block into one or a few meta-features. As a result, the NSC embedding approximates block-level sufficient statistics (block means and low-order summaries), whereas PCA, AE, RP, UMAP, and PaCMAP operate through global mixing, parametric encoding, random projection, or sample-space manifold embedding, and have no explicit bias toward recovering feature-block boundaries.

We instantiate this block-structured synthetic model in the HDLSS regime ($n \ll m$) and compare four NSC variants (NSC, NSC-P, NSC-SP, NSC-pSP) against PCA, AE, RP, UMAP, and PaCMAP using the same evaluation suite as above (linear and k NN probe accuracy, silhouette, and Davies–Bouldin; see section D.3). Over 10 independent repetitions (Table D.1), the NSC family achieves the best mean result on all four metrics: NSC-pSP obtains the highest mean linear accuracy (0.8488 ± 0.0406), best silhouette (0.0460 ± 0.0257), and lowest DB (4.1198 ± 0.9657), while NSC-SP achieves the highest mean k NN accuracy (0.7313 ± 0.0722). Thus, NSC-pSP is the strongest overall variant, with NSC-SP providing the best neighborhood-probe performance (Table D.1, Panel B). Consistently, Friedman tests detect significant differences across the nine methods for all metrics (all $p \leq 9.79 \times 10^{-5}$; Table D.2). Using one-sided Wilcoxon tests with NSC-pSP as the reference, NSC-pSP shows a clear linear-probe advantage over all baselines and NSC variants, including PCA ($p=9.77 \times 10^{-4}$), AE ($p=0.003906$), RP ($p=9.77 \times 10^{-4}$), UMAP ($p=0.004883$), PaCMAP ($p=0.002930$), NSC ($p=0.001953$), NSC-P ($p=0.003906$), and NSC-SP ($p=0.03711$).

For neighborhood-based structure, NSC-pSP significantly improves k NN accuracy over PCA and RP (both $p=0.001953$), NSC ($p=0.04492$), and NSC-P ($p=0.004883$), while differences versus AE ($p=0.3467$), NSC-SP ($p=0.9561$), UMAP ($p=0.6152$), and PaCMAP ($p=0.5771$) are not significant at $\alpha=0.05$. For clustering separability, NSC-pSP yields higher silhouette than PCA, RP, and NSC-P (all $p=9.77 \times 10^{-4}$), AE ($p=0.002930$), and NSC ($p=0.01953$), whereas gains versus NSC-SP, UMAP, and PaCMAP are not significant. Finally, NSC-pSP achieves significantly lower DB than PCA, RP, and NSC-P (all $p=9.77 \times 10^{-4}$), AE ($p=0.002930$), NSC ($p=0.009766$), and PaCMAP ($p=0.01367$), with no significant

difference versus NSC-SP or UMAP. Overall, this controlled experiment highlights that the PCA-IDF + segment-wise PCA tokenization in NSC-pSP better matches the block-correlated generative structure, yielding improved predictive separability and strong local/clustering structure relative to both classical and nonlinear DR baselines, as well as earlier NSC variants. Table D.3 further shows that NSC-pSP consistently outperforms PCA, AE, UMAP, and PaCMAP on linear accuracy, with positive Δ CIs and favorable effect sizes; it also clearly improves DB over PCA, AE, and PaCMAP, while kNN accuracy and silhouette are essentially tied against UMAP/PaCMAP.

D.4.1. A STYLIZED BLOCK-SUBUNIT HDLSS MODEL

Inspired by BSTabDiff (Habib et al., 2026a), we formalize a simple generative model that matches the inductive bias of GO-LR + NSC.

Definition D.1 (Block-subunit HDLSS model). Let m features be partitioned into M disjoint blocks $\{\mathcal{S}_t\}_{t=1}^M$ of equal size s , so that $m = Ms$ and $\mathcal{S}_t \subset \{1, \dots, m\}$, $\mathcal{S}_t \cap \mathcal{S}_{t'} = \emptyset$ for $t \neq t'$. For each block t , we define a latent block signal $h_t \in \mathbb{R}$ and independent noise variables $\{\epsilon_j\}_{j \in \mathcal{S}_t}$ with Eq. 41. The observed features are defined by Eq. 42. The label Y is conditionally independent of X given the block signals in Eq. 43.

$$\epsilon_j \sim \mathcal{N}(0, \sigma^2), \quad \text{i.i.d. across } j \quad (41)$$

$$X_j = h_t + \epsilon_j, \quad j \in \mathcal{S}_t, \quad t = 1, \dots, M \quad (42)$$

$$P(Y | X) = P(Y | h_1, \dots, h_M) \quad (43)$$

We consider a setting where GO-LR, applied with a correlation-based metric, recovers an ordering Π^* that makes the blocks contiguous (up to permutation of the blocks). NSC then segments the GO-LR axis so that each segment $\mathcal{S}_t$ corresponds to one block.

NSC configuration in the block model. In this setting, NSC with: (i) segmentation aligned with the blocks $\{\mathcal{S}_t\}_{t=1}^M$, and (ii) a simple mean descriptor $\psi(u_t) = \frac{1}{|\mathcal{S}_t|} \sum_{j \in \mathcal{S}_t} u_t[j]$ with identity pooling $g_\theta(v) = v$, produces meta-features by Eq. 44. Stacking across t yields the NSC embedding $\Phi_{\text{NSC}}(X) = (z_1, \dots, z_M) \in \mathbb{R}^M$.

$$z_t = \frac{1}{|\mathcal{S}_t|} \sum_{j \in \mathcal{S}_t} X_j, \quad t = 1, \dots, M \quad (44)$$

Proposition D.2 (Block means are sufficient for Bayes classification). *Consider the block model in Definition D.1 in a two-class mean-shift setting where Eq. 45 with σ^2 known and $\mu_{t,0}, \mu_{t,1}$ possibly varying across blocks t . Then for each block $\mathcal{S}_t$ the block sum (Casella & Berger, 2024) in Eq. 46 is a sufficient statistic for $(\mu_{t,0}, \mu_{t,1})$, and the joint log-likelihood ratio (Neyman & Pearson, 1933) for Y based on all features X depends only on the collection $\{\mathcal{S}_t\}_{t=1}^M$. Consequently, any Bayes-optimal classifier (Devroye et al., 1996) based on X can be written as a function of the NSC block means $z_t = S_t/s$.*

$$X_j | Y = y \sim \mathcal{N}(\mu_{t,y}, \sigma^2), \quad j \in \mathcal{S}_t, \quad t = 1, \dots, M, \quad y \in \{0, 1\} \quad (45)$$

$$S_t := \sum_{j \in \mathcal{S}_t} X_j \quad (46)$$

Proof sketch. Within each block t , conditional on $Y = y$, the observations $\{X_j\}_{j \in \mathcal{S}_t}$ are i.i.d. Gaussian with mean $\mu_{t,y}$ and variance σ^2 . The joint likelihood within block t is defined by Eq. 47. Rewriting the exponent shows that this likelihood depends on the data only through the block sum $S_t = \sum_{j \in \mathcal{S}_t} X_j$ (equivalently the block mean z_t). Thus S_t (or z_t) is a sufficient statistic for $\mu_{t,y}$ in the exponential-family sense. Across blocks, conditional independence implies Eq. 48 so the global log-likelihood ratio $\log p(X | Y = 1) - \log p(X | Y = 0)$ depends on X only through $\{\mathcal{S}_t\}_{t=1}^M$, i.e., only through $\{z_t\}_{t=1}^M$. Therefore any Bayes-optimal decision rule $\text{sign}(\log p(Y = 1 | X) - \log p(Y = 0 | X))$ can be written as a function of $(z_1, \dots, z_M)$.

$$p(X_{\mathcal{S}_t} | Y = y) = \prod_{j \in \mathcal{S}_t} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(X_j - \mu_{t,y})^2}{2\sigma^2}\right) \quad (47)$$

$$p(X | Y = y) = \prod_{t=1}^M p(X_{S_t} | Y = y) \quad (48)$$

□

Remark D.3. Proposition D.2 exhibits a family of HDLSS distributions where there exists an NSC configuration (aligned segments, block means) such that the M -dimensional NSC embedding $\Phi_{\text{NSC}}(X)$ is information-preserving for Bayes classification, even though the ambient dimension $m = Ms$ can be arbitrarily large.

Lemma D.4 (SNR gain of NSC vs Random Projection in a block). *Consider one block $\mathcal{S}$ of size s under a simple two-class Gaussian mean-shift model (Eq. 49) with independent coordinates and fixed $\Delta \neq 0$, $\sigma^2 > 0$ (Dasgupta & Gupta, 2003; Devroye et al., 1996; Johnson & Lindenstrauss, 1984).*

$$X_j | Y = 0 \sim \mathcal{N}(0, \sigma^2), \quad X_j | Y = 1 \sim \mathcal{N}(\Delta, \sigma^2), \quad j \in \mathcal{S} \quad (49)$$

1. The NSC block mean (Eq. 50) has class-conditional mean difference $\mathbb{E}[z_{\text{NSC}} | Y = 1] - \mathbb{E}[z_{\text{NSC}} | Y = 0] = \Delta$ and variance $\text{Var}(z_{\text{NSC}} | Y) = \sigma^2/s$, hence we get Eq. 50 and 51

$$z_{\text{NSC}} = \frac{1}{s} \sum_{j \in \mathcal{S}} X_j \quad (50)$$

$$\text{SNR}_{\text{NSC}} := \frac{(\mathbb{E}[z_{\text{NSC}} | Y = 1] - \mathbb{E}[z_{\text{NSC}} | Y = 0])^2}{\text{Var}(z_{\text{NSC}} | Y)} = \frac{\Delta^2 s}{\sigma^2} \quad (51)$$

2. Let $u = (u_j)_{j \in \mathcal{S}}$ be a random projection direction with $\sum_{j \in \mathcal{S}} u_j^2 = 1$ and entries of order $1/\sqrt{s}$, and define Eq. 52. Then we get Eq. 52 and Eq. 53. For typical random u , $\mathbb{E}[(\sum_j u_j)^2] = 1$, so the typical SNR of z_{RP} is defined by Eq. 54.

$$z_{\text{RP}} = \sum_{j \in \mathcal{S}} u_j X_j \quad (52)$$

$$\mathbb{E}[z_{\text{RP}} | Y = 1] - \mathbb{E}[z_{\text{RP}} | Y = 0] = \Delta \sum_{j \in \mathcal{S}} u_j, \quad \text{Var}(z_{\text{RP}} | Y) = \sigma^2 \quad (53)$$

$$\text{SNR}_{\text{RP}} := \frac{(\mathbb{E}[z_{\text{RP}} | Y = 1] - \mathbb{E}[z_{\text{RP}} | Y = 0])^2}{\text{Var}(z_{\text{RP}} | Y)} \approx \frac{\Delta^2}{\sigma^2} \quad (54)$$

Consequently, in this block model (Eq. 55 i.e., NSC's block mean enjoys an SNR gain of a factor s over a typical random projection coordinate.

$$\frac{\text{SNR}_{\text{NSC}}}{\text{SNR}_{\text{RP}}} \approx s \quad (55)$$

Proof sketch. Part (1) follows by linearity of expectation and the variance of the average of s i.i.d. Gaussians. For part (2), the mean and variance of z_{RP} are obtained by linearity and the constraint $\sum_j u_j^2 = 1$. For random u with roughly i.i.d. components of variance $1/s$, we have $\mathbb{E}[(\sum_j u_j)^2] = s \cdot \mathbb{E}[u_j^2] = 1$, so the typical squared signal is Δ^2 , leading to the stated SNR. The ratio then simplifies to s . □

Remark D.5. Lemma D.4 shows that, even at the level of a single block, NSC pooling yields a multiplicative SNR improvement of order s over random projections. In HDLSS regimes where classification is heavily SNR-limited, this translates into a substantial advantage in sample efficiency and Bayes error.

Proposition D.6 (HDLSS stability of NSC block statistics). *In the block model of Definition D.1, suppose the block size s is fixed while the ambient dimension $m = Ms$ may grow. For each block $\mathcal{S}_t$, the NSC block mean (Eq. 56) satisfies Eq. 57 and its empirical estimate based on n samples converges to its population value at rate $O_p(n^{-1/2})$, independently of m . By contrast, global linear DR methods such as PCA or dense linear encoders must estimate directions in $\mathbb{R}^m$ based on the empirical covariance matrix or large weight matrices of size $O(m)$ or $O(m^2)$, which is known to be unstable in the HDLSS regime $m \gg n$ without strong structural assumptions (Jung & Marron, 2009; Yata & Aoshima, 2009; Strawderman, 2014;*

Cover & Thomas, 2006). Thus, in this stylized setting, NSC’s local block statistics remain well-conditioned as m grows, while unconstrained global DR can become ill-posed.

$$z_t = \frac{1}{s} \sum_{j \in \mathcal{S}_t} X_j \quad (56)$$

$$z_t = h_t + \bar{\epsilon}_t, \quad \bar{\epsilon}_t := \frac{1}{s} \sum_{j \in \mathcal{S}_t} \epsilon_j \sim \mathcal{N}(0, \sigma^2/s) \quad (57)$$

D.5. Comparison and Evaluation Protocol

To empirically position NSC as a structured DR layer, we adopt the following protocol for both real and synthetic experiments.

Experimental setup. For each dataset (real HDLSS or synthetic block-structured):

1. **GO-LR ordering.** Compute the GO-LR global feature permutation Π^* on the training set using a chosen metric (e.g., correlation, cosine, euclidean, manhattan, or KL divergence), with local refinement passes as in Algorithm 1.
2. **Intrinsic dimension.** Estimate $\hat{d}$ via effective rank (Eqs. 27-29), and compute $\text{IDF} = \hat{d}/m$ (Eq. 30).
3. **NSC configuration.** Configure NSC with Π^* , using:
 - *Default compression:* M chosen by Eq. 21 (e.g., yielding $M \approx 2\hat{d}$ and resulting in tens to a few hundred meta-features).
 - *Aggressive compression:* fixed $M \in \{32, 64\}$ to emulate strong dimensionality reduction.

Apply NSC to obtain embeddings $X_{\text{NSC}} \in \mathbb{R}^{n \times M}$.

Baselines. For each dataset and target dimension M (e.g., $M = 32$), we construct the following DR baselines:

- **PCA:** compute k principal components with $k = M$, producing $X_{\text{PCA}} \in \mathbb{R}^{n \times M}$.
- **RP:** apply a dense Gaussian projection $R \in \mathbb{R}^{M \times m}$, normalized to preserve variance, yielding $X_{\text{RP}} = X R^\top$.
- **AE:** train a shallow autoencoder with bottleneck dimension M on the training set, and use the bottleneck activations $X_{\text{AE}} \in \mathbb{R}^{n \times M}$ as the DR representation.
- **UMAP:** learn a nonlinear manifold embedding with target dimension M , preserving local neighborhood structure in the sample space and producing $X_{\text{UMAP}} \in \mathbb{R}^{n \times M}$.
- **PaCMAP:** learn a nonlinear embedding with target dimension M by balancing nearby, mid-near, and far pair constraints, yielding $X_{\text{PaCMAP}} \in \mathbb{R}^{n \times M}$.

NSC variants and naming conventions. We evaluate a family of NSC variants that share a common pipeline (i) determine an IDF to set the internal granularity which is a ratio of Intrinsic Dimension (ID) to the actual dimension, (ii) segment the feature sequence into contiguous blocks, and (iii) compress each block into an M -dimensional representation, but differ in how IDF is estimated and how each segment is summarized. We denote the default variant as NSC, which uses an effective-rank (data-driven) ID estimate and applies statistical descriptor-based pooling within each segment. To isolate the impact of a PCA-inspired ID heuristic while keeping the same descriptor-based segment summarization, we use **NSC-P** (also written as NSC-PCA). To study the role of replacing descriptors with explicit linear projection inside segments, we define **NSC-SP** (also written as NSC-SegPCA), which retains the effective rank-based IDF but applies PCA within each segment after segmentation. Finally, our proposed **NSC-pSP** (also written as NSC-PIDF-SegPCA) combines both modifications: a PCA-inspired IDF estimate together with per-segment PCA compression after segmentation. In all cases, the suffixes indicate the modification relative to NSC: **P** denotes PCA-inspired IDF, **SP** denotes segmented per-block PCA, and **pSP** denotes the combination of both.

Quantitative comparisons. To compare NSC variants against PCA/RP/AE/UMAP/PaCMAP, we use:

1. **Linear-probe accuracy.** Train a logistic regression classifier on each latent space $X_{\text{NSC}}, X_{\text{PCA}}, X_{\text{RP}}, X_{\text{AE}}, X_{\text{UMAP}}, X_{\text{PaCMAP}}$ using the same stratified cross-validation protocol. This measures how well each DR method preserves label-relevant structure in M dimensions.
2. **k NN classification in latent space.** Evaluate k NN accuracy using the compressed embeddings under the same stratified cross-validation protocol to assess neighborhood quality.
3. **Label-based separability metrics.** Compute silhouette and Davies-Bouldin scores on each latent representation using ground-truth class labels as the partition.
4. **Statistical comparison across datasets.** Aggregate per-dataset metrics and apply nonparametric tests, using a Friedman test (Friedman, 1937) across methods followed by one-sided Wilcoxon signed-rank comparisons with NSC-pSP as the reference (see Tables D.1, D.2, D.3).

Evaluation protocol. We report only quantitative DR-style evaluations under a fixed aggressive budget of $M=32$. For each method (NSC variants, PCA, AE, RP, UMAP, PaCMAP), we first compute a 32-dimensional latent representation in $\mathbb{R}^{32}$, and then assess (i) linear-probe accuracy via logistic regression, (ii) k NN accuracy in latent space, and (iii) label-based separability via silhouette and Davies-Bouldin scores, using the same stratified cross-validation protocol across methods (Tables D.1-D.2). Overall, these experiments treat NSC as a first-class dimensionality reduction layer: it produces an explicit $\mathbb{R}^M$ representation that can be consumed directly by TabPFN-style predictor under strict token budgets, while still supporting standard DR comparisons to PCA/RP/AE/UMAP/PaCMAP via probe and clustering metrics. In HDLSS settings, NSC couples GO-LR’s MinLA-motivated ordering with subunit-wise pooling to achieve (i) aggressive compression ($M \ll m$) and (ii) a locality-preserving structured embedding. Empirically, NSC is typically competitive with PCA, AE, UMAP, and PaCMAP and consistently outperforms RP on real HDLSS datasets; on the block-structured synthetic model, the NSC family yields the best mean results across the evaluated metrics (Table D.1) and shows significantly stronger predictive, neighborhood, and separability structure in several comparisons (Table D.2), highlighting a regime where its locality bias matches the data-generating process.

Synthetic block-model validation and takeaways. To connect the empirical trends to our theoretical picture, we evaluate NSC as an explicit DR layer on a synthetic HDLSS block model aligned with its inductive bias: m features are generated in $B=40$ latent-factor blocks (shared block signal + noise), class information is injected via mean shifts on a small subset of blocks, and then feature indices are randomly permuted to destroy contiguity in the raw space. Under an aggressive budget ($M=32$) and over 10 independent repetitions (Table D.1), NSC-style embeddings remain strongly discriminative while preserving neighborhood/cluster structure competitively against unstructured and nonlinear DR: nonparametric tests confirm significant differences across methods and show consistent gains in latent-space geometry (e.g., higher k NN accuracy versus PCA/RP and improved separability via silhouette/DB; Table D.2). Overall, these results reinforce the DR interpretation of NSC: GO-LR constructs a locality-revealing axis, and NSC compresses it via subunit-wise pooling into interpretable meta-features whose coordinates summarize coherent redundant neighborhoods, contrasting with the global mixing of PCA/RP, many AE encoders, and sample-space manifold embeddings such as UMAP/PaCMAP. Thus, its primary advantage under strong compression is retaining local geometry and cluster structure while staying competitive in discriminative accuracy; among variants, NSC-pSP is the most consistently competitive on the block model (Table D.3), while on real HDLSS benchmarks at the same $M=32$ budget it is broadly comparable to PCA/AE/UMAP/PaCMAP with smaller, dataset-dependent differences.

GO-LR+NSC vs. PCA. Global PCA (Hotelling, 1933) $\rightarrow$ TabPFN-2.5 (Grinsztajn et al., 2025) is a natural control for testing whether the gains of GOTabPFN come merely from dimensionality reduction. We therefore compare GO-LR+NSC+TabPFN-2.5 against Global PCA+TabPFN-2.5 while keeping the same frozen TabPFN-2.5 predictor and changing only the front-end compression interface. As shown in Table D.4, GO-LR+NSC outperforms Global PCA on all 8 HDLSS datasets, suggesting that the improvement is not explained by generic global compression alone, but by locality-aware ordering and structured neighborhood compression.

GO-LR+NSC vs. Lasso-selected features. We compare GO-LR+NSC against Lasso-selected features under the same frozen TabPFN-2.5 (Grinsztajn et al., 2025) predictor and identical 5×5 CV protocol. Specifically, we evaluate

Lasso+TabPFN-2.5 (LT) across sparsity levels $C \in \{0.01, 0.02, 0.05\}$. As shown in Table D.5, GOTabPFN outperforms all LT variants on most HDLSS datasets, while LT only slightly exceeds GOTabPFN on SMK and AML under the weakest sparsity setting ($C = 0.05$). This suggests that locality-aware feature ordering and structured neighborhood compression provide more robust gains than sparsity-based feature selection alone in HDLSS settings.

E. Why Feature Ordering? Local Neighborhoods Enable Structure-Aware Compression

A common critique is that tabular features form an unordered set, so learning a column order may appear arbitrary. In this work, ordering is not introduced to impose a fictional semantics (e.g., time), but to construct an algorithmic coordinate system: a 1D axis on which local neighborhoods become meaningful via a structure-revealing layout objective from seriation / graph layout (rather than positional meaning) (Arabie & Hubert, 1992; Hahsler et al., 2008; Díaz et al., 2002; Atkins et al., 1998). This is essential for our pipeline because NSC is an explicitly local operator it pools contiguous segments into meta-features (Sec. 3.2). Without an ordering that places related features nearby, contiguity-based pooling becomes arbitrary aggregation and can destroy predictive signal. Therefore, feature ordering is primarily justified as a neighborhood construction mechanism that enables structured compression and stable tokenization under tight budgets.

E.1. Ordering Constructs Coherent Neighborhoods That NSC Can Pool

NSC partitions the ordered feature axis into M contiguous segments and pools each segment into a token (Sec. 3.2). This implicitly assumes that adjacency along the axis corresponds to statistical relatedness. GO-LR is designed exactly to enforce this locality: it approximately minimizes a MinLA/seriation-style dispersion objective that penalizes placing strongly related feature pairs far apart, aligning index locality with a similarity graph (Díaz et al., 2002; Atkins et al., 1998; Garey et al., 1974). As a result, features that are close under the global dissimilarity structure become near neighbors in index, so NSC pooling operates on coherent neighborhoods and yields compressed tokens that preserve informative local structure. In contrast, if features are left in raw or arbitrary order, contiguous segments mix unrelated variables and pooling becomes a lossy averaging operation. This is precisely the failure mode we can worry about: ordering only matters if downstream modules use contiguity. Since NSC explicitly uses contiguity, ordering becomes a necessary part of the representation interface.

E.2. Ordering Improves Compression by Reducing Cross-Segment Boundary Cuts

The role of ordering can be formalized through the segmentation-induced boundary cut. Let $\bar{W} \in \mathbb{R}^{m \times m}$ denote the global dissimilarity matrix used to define neighborhoods (Sec. 3.2), and let Π^* be the learned ordering. Consider a segmentation into M contiguous segments $\{\mathcal{S}_t\}_{t=1}^M$ along the ordered axis. Define the cross-segment boundary cost by Eq. 58. Intuitively, Cut measures how often highly related features are split across segment boundaries; the definition parallels cut objectives used in graph-based segmentation/layout (Shi & Malik, 2000; Díaz et al., 2002). A smaller cut indicates that each segment captures a coherent neighborhood, so pooled tokens retain structure. Since GO-LR reduces dispersion, it typically also reduces boundary cuts for reasonable segmentations; random or raw orders inflate boundary cuts, making local pooling ineffective.

$$\text{Cut}(\Pi^*, \{\mathcal{S}_t\}) = \sum_{t=1}^{M-1} \sum_{i \in \mathcal{S}_t} \sum_{j \in \mathcal{S}_{t+1}} \bar{W}_{\Pi^*(i), \Pi^*(j)} \quad (58)$$

E.3. What Ordering Is Not: We Do Not Make Tabular Data “Sequential”

We do not claim that tabular columns possess an intrinsic order like words or pixels. Rather, we learn an order as a layout that makes neighborhood-based operators (pooling, segmentation, local filters) well-defined. This mirrors classic seriation/linear arrangement goals: the objective is not positional semantics, but an ordering that makes locality meaningful for downstream computation (Arabie & Hubert, 1992; Hahsler et al., 2008; Atkins et al., 1998; Díaz et al., 2002). In our case, ordering is valuable specifically because NSC is local along the constructed axis.

E.4. Empirical Diagnostics for Neighborhood Preservation (Order-Only)

We report order-only diagnostics that quantify whether an ordering induces meaningful local neighborhoods independently of any classifier or tokenizer. All diagnostics are computed on the 8 HDLSS datasets using standardized features. Since explicitly materializing a dense pairwise dissimilarity/Gram matrix scales as $\mathcal{O}(m^2)$ in memory (and time), forming

$\bar{W} \in \mathbb{R}^{m \times m}$ quickly becomes impractical at HDLSS dimensions (e.g., gene-expression arrays with $\sim 10^4$ - $10^4.5$ genes and GWAS with $\geq 10^5$ SNP markers). (Si et al., 2017; Dangond, 2000; Maguire et al., 2018), we use a lightweight proxy $\bar{W}_{ij} \triangleq 1 - |\text{corr}(x_i, x_j)|$, estimated from the standardized data (and for adjacent deltas from a small row subset for stability/speed) where lower is better, indicating that neighbors along the axis are more mutually similar. Across the 8 datasets, Π^* achieves significantly lower adjacency dissimilarity than random permutations (paired by dataset; Wilcoxon signed-rank $p = 0.00390625$; Fig. E.1d,f).

Local adjacency coherence (path-length objective). Given a feature ordering Π^* and a dissimilarity matrix $\bar{W}$, we define the adjacent dissimilarity $\delta_t = \bar{W}_{\Pi^*(t), \Pi^*(t+1)}$. We define the adjacency coherence as the mean adjacent dissimilarity along the ordering, which is the Hamiltonian path (TSP-path) length objective used in seriation, up to normalization (Hahsler et al., 2008). We measure $\text{AdjCoh}(\Pi^*)$ by Eq. 59.

$$\text{AdjCoh}(\Pi^*) = \frac{1}{m-1} \sum_{t=1}^{m-1} \delta_t = \frac{1}{m-1} \sum_{t=1}^{m-1} \bar{W}_{\Pi^*(t), \Pi^*(t+1)} \quad (59)$$

Neighborhood hit-rate for top- k neighbors. For each feature i , let $\mathcal{N}_k(i)$ be its top- k nearest neighbors under $\bar{W}$. For an ordering Π^* , define the window neighborhood $\mathcal{W}_h(i) = \{j : |\text{pos}_{\Pi^*}(j) - \text{pos}_{\Pi^*}(i)| \leq h\}$. We compute $\text{HitRate}_{k,h}(\Pi^*)$ by Eq. 60 where higher is better. To avoid $O(m^2)$ complexity, we compute $\mathcal{N}_k(i)$ on a feature subsample (default 2048 features) and average over features and (when applicable) multiple random seeds. Π^* yields consistently higher hit-rate than random orderings (Wilcoxon signed-rank $p = 0.0078125$ for a representative $(k, h) = (10, 16)$; Fig. E.1c,e,f), and the advantage persists over multiple (k, h) choices (Fig. E.1e) (Venna & Kaski, 2001).

$$\text{HitRate}_{k,h}(\Pi^*) = \frac{1}{m} \sum_{i=1}^m \frac{|\mathcal{N}_k(i) \cap \mathcal{W}_h(i)|}{k} \quad (60)$$

Segmentation boundary cut. To test alignment between the ordering and contiguity-based pooling, we evaluate a boundary-cut proxy under common segmentation rules (uniform, equal-mass, largest-jump) with $M=32$ and $l_{\min}=8$ (matching our NSC configuration). Given segments $\{\mathcal{S}_t\}$ along Π^* , we summarize the average dissimilarity at segment boundaries via the adjacent deltas (Eq. 61) where $\mathcal{B}$ are boundary indices between consecutive segments. Lower cut indicates that segment boundaries fall on weaker connections, i.e., stronger within-segment neighborhood coherence. Across datasets, Π^* yields favorable boundary alignment relative to random permutations (Fig. E.1b) (Shi & Malik, 2000).

$$\text{Cut}(\Pi^*, \{\mathcal{S}_t\}) \approx \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \delta_{b-1} \quad (61)$$

E.5. Ablations That Isolate Why Ordering Helps NSC

To separate the effect of ordering from the effect of compression/tokenization, we evaluate NSC under controlled ordering perturbations while keeping the tokenizer/compressor fixed (same M , segmentation rule, pooling, and tuned hyperparameters). We use five random seeds for permutation-based controls.

Ordered vs. un-ordered. We compare the same NSC configuration under: (i) GO-LR order Π^* , (ii) the raw/original column order, (iii) random permutations (averaged over seeds). This directly tests whether NSC benefits from neighborhood structure rather than from compression alone. In parallel, the order-only diagnostics ($\text{AdjCoh}/\text{HitRate}/\text{Cut}$) show that Π^* is systematically more neighborhood-preserving than random, providing a mechanistic explanation for the observed gains (Fig. E.1c,d,f).

Destroy global layout while partially preserving locality (block shuffle). Starting from Π^* , we partition indices into contiguous blocks of size b and randomly permute the blocks. This preserves within-block neighborhoods but disrupts long-range arrangement. If NSC relies primarily on local neighborhoods, performance/diagnostics should improve as b increases. Consistent with this, normalized hit-rate increases monotonically with block size: 0.740 ± 0.184 ($b=8$), 0.889 ± 0.177 ($b=16$), 0.968 ± 0.097 ($b=32$), 1.011 ± 0.095 ($b=64$), relative to the corresponding Π^* value per dataset (Fig. E.1a,f). This supports the locality hypothesis: preserving larger local neighborhoods recovers the Π^* advantage.

Keep order fixed, break contiguity (round-robin segments). We keep Π^* but break contiguity by assigning features to segments in a round-robin manner and then concatenating segments. This retains the same set and global ordering statistics, but destroys the contiguous neighborhood pooling assumption. The resulting drop in hit-rate (and corresponding degradation in order-sensitive behavior) isolates contiguity as the operative mechanism (Fig. E.1c,d).

Optional representation probes. When needed, we complement the order-only metrics with lightweight probes on the produced tokens (e.g., linear probe) under the ablations above. These probes are used only to verify that improved neighborhood preservation translates into higher-quality representations, while the primary claim remains anchored in the classifier-free diagnostics.

Summary. Across datasets, the learned ordering Π^* consistently preserves local neighborhoods better than baselines: the hit-rate is highest under Π^* , typically followed by the raw order, while randomization and explicitly breaking contiguity degrade neighborhood agreement (Fig. E.1(c)); this is mirrored by adjacency coherence, where Π^* attains the lowest (best) AdjCoh and random orderings are worst (Fig. E.1(d)). The robustness heatmap further shows that Π^* yields uniformly positive gains over random across all tested (k, h) , with larger improvements for wider locality windows (larger h) and smaller k (Fig. E.1(e)). For segmentation alignment, Π^* tends to reduce boundary cut under “equal_mass” (and is roughly neutral under “largest_jump”), whereas “uniform” can be inconsistent and often flips the advantage (negative median ΔCut), suggesting uniform boundaries may not match the induced contiguous structure (Fig. E.1(b)). Finally, the block-shuffle experiment shows a clear scale effect: when features are shuffled within small blocks, the normalized hit-rate drops substantially, but it recovers toward Π^* as block size increases (e.g., rising from ≈ 0.67 at $b=8$ to ≈ 0.90 at $b=64$), indicating that locality is largely preserved within coarse blocks and mainly disrupted by fine-grained shuffles (Fig. E.1(a)).

E.6. Beyond NSC: When Ordering Can Improve Accuracy

While our primary motivation is NSC’s contiguity-based pooling, the same locality principle applies to any order-sensitive learner that introduces architectural locality over feature tokens (e.g., local attention windows, relative position bias, or convolutional mixing) (Vaswani et al., 2017; Child et al., 2019; Huang et al., 2020; Gorishniy et al., 2021; Somepalli et al., 2022). In such models, the feature order acts as a computational layout: by placing statistically related features nearby, the model concentrates informative interactions into small neighborhoods, which can reduce sample complexity and improve generalization in low-sample regimes.

Experimental evidence (accuracy changes only when the backbone uses locality). We run controlled ordering experiments on two biomedical $n < m$ datasets (AI-d_case5 (Ohlsson et al., 2020) and ADNLAD123 (Petersen et al., 2010)) using the same backbone and training protocol, changing only the column order applied consistently to train/val/test. As an order-sensitive backbone, we use a local-window Transformer whose attention is restricted to a fixed neighborhood around each feature token (Beltagy et al., 2020), making performance dependent on index-locality. We evaluate multiple ordering strategies: our GO-LR order Π^* , a TabSeq-style ordering (Habib et al., 2024), the raw column order, random permutations (averaged over seeds followed by TabICL (Jingang et al., 2025)), a light version of ROTATOR (Wang et al., 2025) and controlled perturbations that partially preserve locality (block-shuffle) or explicitly destroy contiguity while keeping the same global order statistics (round-robin “break contiguity”). Figure E.2 (top) and Table E.1 show that ordering yields non-trivial AUC changes for the local-window Transformer, and GO-LR produces the strongest gains among the tested ordering methods on these datasets. We use a fixed training protocol with a single train/val/test split and fixed hyperparameters (no dataset-specific tuning or HPO); all results are produced with a fixed seed (SEED=42), except the random-permutation baseline which is averaged over 5 permutation seeds.

Permutation-invariant sanity check. To verify that gains are not artifacts of reindexing, we repeat the same experiment using a permutation-invariant control model (set encoder / invariant pooling) where consistent reordering should not systematically affect performance (Zaheer et al., 2017). As expected, Figure E.2 (middle) shows near-zero deltas across orderings, supporting the interpretation that improvements arise from locality-aware computation rather than from accidental leakage or inconsistent preprocessing. This perspective is also compatible with permutation-ensemble approaches such as TabICL, which averages predictions over multiple random feature permutations to approximate invariance (Jingang et al., 2025).

Mechanistic link: better neighborhood preservation $\Rightarrow$ better accuracy. Finally, we connect accuracy changes to order-only locality diagnostics. Figure E.2 (bottom) shows that orderings with higher neighborhood hit-rate ($\text{HitRate}_{k,h}$) tend to yield higher AUC under the local-window backbone, supporting the hypothesis that ordering helps by increasing the density of meaningful local interactions. In other words, ordering can improve accuracy whenever the downstream architecture uses locality; when the architecture is invariant, ordering should not matter such as tree-based models or set-based models (Zaheer et al., 2017).

Relation to prior ordering methods. This finding aligns with prior work that explicitly learns or uses feature layouts to benefit order-sensitive tabular learners (e.g., TabSeq (Habib et al., 2024) ordering heuristics and ordering strategies in recent LLM-based tabular pipelines such as ROTATOR-LLM (Wang et al., 2025)).

Summary. Figure E.2 provides a causal/mechanistic check that ordering only matters when the backbone uses locality. In the order-sensitive local-window Transformer, GO-LR (Π^*) yields the most consistent positive ΔAUC versus random across both datasets, while disrupting locality either by random permutations, breaking contiguity, or (to a lesser extent) block shuffling reduces or erases these gains; in contrast, the permutation-invariant control shows ΔAUC values clustered near zero with no systematic advantage for any ordering, indicating that improvements are not an artifact of reindexing features but arise from the model’s locality bias (Fig. E.2a). The mechanism plot further supports this explanation: for the local model, downstream AUC increases with neighborhood preservation ($\text{HitRate}_{k,h}$), with higher-performing orderings (e.g., GO-LR/ Π^*) occupying the high-HitRate/high-AUC region, whereas random or contiguity-breaking variants sit at lower HitRate and correspondingly lower AUC (Fig. E.2b). Together, these results suggest that learned orderings improve accuracy or overall classification performance beyond NSC-based compression specifically by aligning informative feature neighborhoods with the local attention window, and that when locality is removed (invariant control), ordering ceases to provide systematic benefit (Fig. E.2).

F. Feature Ordering - When to Use? Through the Lens of Locality

When Feature Ordering Matters. Deep Sets (Zaheer et al., 2017) is designed to be permutation-invariant for genuinely unordered inputs (e.g., point clouds, MIL, and chemoinformatics). In contrast, we study high-dimensional tabular settings where the chosen column layout can materially affect learning. A well-chosen permutation can reduce redundancy, expose latent dependencies among features, and ultimately improve predictive performance or help in contiguity based-process like NSC where locality matters. This is particularly pertinent for high-dimensional biological measurements (e.g., gene expression), EEG and other sensor data, remote sensing and climate datasets, and multimodal or heavily engineered feature tables domains that often exhibit sparsity, redundancy, and hidden structure, and are thus natural targets for sequence-dependent models. DynaTab (Habib et al., 2026b) initiated a systematic study of when feature ordering is useful in high-dimensional tabular learning, primarily from the perspective of ordering sensitivity and sequence-dependent modeling. We extend this view through the lens of locality: feature ordering is useful not only because sequence-sensitive backbones depend on token order, but also because a good permutation can create contiguous neighborhoods of statistically related features, enabling locality-based operators such as NSC to compress related features into informative meta-features.

Dataset Categorization Rules. There is no universally agreed upon numerical cutoff that uniquely determines when a dataset should be labeled HDLSS. In much of the HDLSS literature, the term is used broadly for regimes where the ambient dimension (number of variables) is far larger than the sample size, and is often formalized through HDLSS asymptotics (Aoshima et al., 2018) in which the dimension grows while the sample size is fixed (or grows much more slowly) (Hall et al., 2005; Jung & Marron, 2009). Consequently, the simple rule “ $m > n$ ” is a useful heuristic but too coarse to capture the practical spectrum of high dimensionality. To make this notion more operational, we follow DynaTab’s (Habib et al., 2026b) empirical regime stratification and use the feature-to-sample ratio $\rho = m/n$, which helps distinguish qualitatively different regimes beyond a binary HDLSS vs. non-HDLSS split. Let n denote the number of samples, m the number of features, and $\rho = \frac{m}{n}$ the feature-to-sample ratio. We assign each dataset to one of five regimes using the following empirical ρ thresholds:

HDLSS:	$m > 1000, n < 1000, \rho > 2.$
HDHSS:	$m > 1000, n > 10^4, 0.005 < \rho \leq 2.$
LDHSS:	$m \leq 100, n > 10^4, \rho \leq 0.01.$
LDLSS:	$m \leq 100, n \leq 1000, \rho \leq 0.05.$
MixedRegime:	otherwise.

F.1. Intrinsic Dimensionality Factor as a Proxy for Locality Exploitability

Our primary justification for feature ordering in this paper is locality: ordering constructs an algorithmic 1D axis on which contiguity corresponds to statistical relatedness, making neighborhood-based operators (e.g., NSC’s contiguous pooling) well-defined (Sec. 3.2, Appendix E). This motivates a complementary question: when should we expect ordering to provide tangible benefit? We connect to this locality view via the IDF. Let m be the ambient number of features and let $\hat{d}$ denote an estimate of the dataset’s intrinsic dimensionality (e.g., the number of principal components required to reach a fixed cumulative variance threshold, or an effective-rank estimate). In other words, the minimal number of features capturing core variability (Chen et al., 2022) to its total feature count. Following DynaTab (Habib et al., 2026b), we use Eq. 62 to compute IDF. Intuitively, IDF measures how compact the data are relative to the ambient dimension. A small IDF indicates substantial redundancy/low effective rank, which we hypothesize corresponds to stronger, more compressible correlation structure and thus a greater ability to induce local neighborhoods via a 1D layout.

$$\text{IDF} = \frac{\hat{d}}{m} \quad (62)$$

Complexity score (IDF-normalized compactness). Following DynaTab (Habib et al., 2026b), we summarize dataset compactness and “ordering opportunity” by the IDF-normalized score in Eq. 63, where $\text{CumVar}(\hat{d})$ is the cumulative variance explained at $\hat{d}$ components and p is a tunable sensitivity parameter. Larger values indicate that a small intrinsic subspace captures substantial variance, suggesting higher redundancy and greater potential for ordering-based locality.

$$\text{ComplexityScore} = \frac{\text{CumVar}(\hat{d})}{\text{IDF}^p} \quad (63)$$

F.2. Feature Ordering Effectiveness and Success Probability

Following DynaTab (Habib et al., 2026b), we use the Feature Ordering Effectiveness (FOE) as a composite indicator of ordering benefit, given in Eq. 64, where κ is a dataset-specific scaling factor and AUC denotes the area under the IDF-variance curve, estimated by trapezoidal integration over discrete IDF-variance pairs. We choose κ by minimizing the deviation from a target value (set to 1) via Eq. 65. Setting $p=2$ introduces quadratic sensitivity (Hinton & Salakhutdinov, 2006), amplifying penalties when variance grows slowly with intrinsic dimension. AUC is estimated using the trapezoidal rule (Hanley & McNeil, 1982) for efficient integration of discrete IDF–variance pairs. While κ and AUC vary by dataset, FOE preserves an inverse dependence on IDF by Eq. 66. We also report a simple success-probability proxy by Eq. 67. As $\hat{d} \rightarrow m$, p_{succ} decreases, reflecting limited room for ordering to expose structure beyond what is already “fully spread” across features.

$$\text{FOE} = \frac{\kappa}{(\text{AUC} \cdot \text{IDF})^p} \quad (64)$$

$$\text{Loss}(\kappa) = \left(\frac{\kappa}{(\text{AUC})^p} - 1 \right)^2 \quad (65)$$

$$\text{FOE} \propto \frac{1}{\text{IDF}} \quad (\text{for fixed } \kappa \text{ and AUC}) \quad (66)$$

$$p_{\text{succ}} = 1 - \text{IDF} = 1 - \frac{\hat{d}}{m} \quad (67)$$

F.3. Linking IDF/FOE to Locality: Testable Predictions

The locality view yields a mechanistic interpretation of IDF/FOE: ordering helps when the dataset admits a linear layout that concentrates strong relations into short-range neighborhoods. We formalize this using order-only locality diagnostics (Appendix E.4). Let Π^* be the learned GO-LR ordering and let $\Pi^{(r)}$ denote random permutations.

Locality gains relative to random orderings. We define three locality gains by Eqs. 68, 69, 70.

$$\Delta \text{AdjCoh} = \mathbb{E}_r [\text{AdjCoh}(\Pi^{(r)})] - \text{AdjCoh}(\Pi^*) \quad (68)$$

$$\Delta \text{HitRate}_{K,h} = \text{HitRate}_{K,h}(\Pi^*) - \mathbb{E}_r [\text{HitRate}_{K,h}(\Pi^{(r)})] \quad (69)$$

$$\Delta \text{Cut} = \mathbb{E}_r [\text{Cut}(\Pi^{(r)})] - \text{Cut}(\Pi^*) \quad (70)$$

Positive values indicate that Π^* induces stronger locality than random orderings: lower adjacent dissimilarity (better AdjCoh), higher neighborhood recovery (better HitRate), and lower cross-segment boundary cost (better Cut).

Locality Exploitability Score (LES). Optionally, we aggregate these into a single dataset-level score by z-normalizing each gain across datasets and averaging by Eq. 71 where LES measures how much local neighborhood structure a dataset allows ordering to unlock.

$$\text{LES} = \frac{1}{3} \left(\text{zscore}(\Delta \text{AdjCoh}) + \text{zscore}(\Delta \text{HitRate}_{K,h}) + \text{zscore}(\Delta \text{Cut}) \right) \quad (71)$$

For a benchmark suite with multiple datasets, the $\text{zscore}(\cdot)$ terms in Eq. 71 are computed across the evaluated dataset collection. For single-dataset diagnostics, cross-dataset z-normalization is not defined. We therefore report the raw finite-diagnostic aggregate

$$\text{LES}_{\text{single}} = \frac{1}{|\mathcal{D}_{\text{fin}}|} \sum_{d \in \mathcal{D}_{\text{fin}}} d, \quad \mathcal{D}_{\text{fin}} = \{\Delta \text{AdjCoh}, \Delta \text{HitRate}_{K,h}, \Delta \text{Cut}\} \cap \mathbb{R}_{\text{finite}} \quad (72)$$

Here, $\mathcal{D}_{\text{fin}}$ contains only finite locality diagnostics, so unavailable quantities such as $\Delta \text{HitRate}_{K,h}$ or ΔCut for very small feature dimensions are omitted from the average. Thus, Eq. 71 is benchmark-relative, while Eq. 72 provides a dataset-level locality summary when only one dataset is evaluated.

Predictions. Under the locality hypothesis, IDF/FOE provide opportunity indicators for locality gains, rather than deterministic guarantees. We summarize this diagnostic expectation as in Eq. 73.

$$\text{IDF} \downarrow, \text{FOE} \uparrow, p_{\text{succ}} \uparrow \implies \text{higher expected opportunity for locality gains} \quad (73)$$

Importantly, Eq. 73 is diagnostic rather than deterministic: IDF/FOE indicate when ordering is worth trying, while LES measures whether the learned GO-LR ordering actually realizes locality gains over random orderings.

For single-dataset diagnostics, the same opportunity-based interpretation can be applied to $\text{LES}_{\text{single}}$ from Eq. 72, but it should be treated as an empirical diagnostic rather than a guaranteed monotonic relationship. Intuitively, small IDF suggests that variance concentrates in a low-dimensional subspace, which may co-occur with stronger feature redundancy or community structure in the similarity graph. When such structure is present, GO-LR can align related features into contiguous neighborhoods that NSC can pool effectively.

F.4. Experimental Validation Protocol (Locality as the Bridge)

We validate the bridge “When $\Rightarrow$ Why” by testing whether IDF/FOE predict order-only locality gains on the same dataset suite used throughout the paper.

Correlation tests (dataset-level). Across datasets, we compute Spearman correlations between IDF (or p_{succ} / FOE) and each locality gain by Eq. 74.

$$\rho(\text{IDF}, \Delta \text{HitRate}_{K,h}), \rho(\text{IDF}, \Delta \text{AdjCoh}), \rho(\text{IDF}, \Delta \text{Cut}), \rho(\text{FOE}, \text{LES}) \quad (74)$$

We expect negative correlations for IDF (smaller IDF $\Rightarrow$ larger gains) and positive correlations for FOE and p_{succ} .

Link to downstream NSC behavior. To directly connect locality gains to NSC’s contiguity-based pooling, we also test whether datasets with larger LES obtain larger ordering-induced improvements under NSC by Eq. 75.

$$\rho(\text{LES}, \Delta\text{Perf}_{\text{NSC}}), \quad \Delta\text{Perf}_{\text{NSC}} = \text{Perf}(\text{NSC} + \Pi^*) - \mathbb{E}_r[\text{Perf}(\text{NSC} + \Pi^{(r)})] \quad (75)$$

This closes the chain:

low IDF / high FOE $\rightsquigarrow$ higher opportunity for locality gains $\xrightarrow{\text{validated by LES}}$ effective contiguity-based pooling (NSC)

Summary. We use locality as the operational criterion for deciding “when” feature ordering should help. Concretely, we first compute an opportunity proxy from intrinsic dimensionality: we estimate $\hat{d}$ from the PCA cumulative-variance curve at a fixed threshold, then we define $\text{IDF} = \hat{d}/m$, and form FOE by combining IDF with the IDF-variance curve area AUC (with κ optimized via Eq. 65). Intuitively, HDLSS/HDHSS datasets typically exhibit very small IDF and hence large FOE (Table F.1, top ranks), indicating strong redundancy/low effective rank and thus substantial room for an ordering algorithm to expose coherent local neighborhoods. In contrast, low-dimensional or near-full-rank datasets tend to have $\text{IDF} \approx 1$ (and $P_{\text{success}} = 1 - \text{IDF} \approx 0$), suggesting limited remaining structure for ordering to uncover; in such cases, ordering is expected to be less critical unless the locality diagnostics indicate otherwise. Additionally, Table F.2 shows that orlraws10P has the strongest expected benefit from ordering, with the lowest IDF and highest FOE among the additional cross-domain datasets. Cell Cycle also exhibits a relatively high FOE despite being MixedRegime, suggesting substantial compressible structure. In contrast, RELATHE, BASEHOCK, and PCMAC have lower FOE scores, indicating that feature ordering is expected to provide more limited gains for these datasets. Second, beyond this screen, we quantify whether the opportunity is realized by the ordering algorithm: we learn a GO-LR ordering Π^* and measure order-only locality gains against random permutations using ΔAdjCoh , $\Delta\text{HitRate}_{K,h}$, and ΔCut , whose z-normalized average defines LES (Eqs. 68-71). The scatter plots (Fig. F.1) show that IDF/FOE are best interpreted as capacity measures: they separate regimes where ordering is plausibly useful (low IDF/high FOE, often HDLSS) from regimes where it is unlikely to matter (high IDF, typically low-dimensional), while LES diagnoses whether GO-LR successfully linearizes the dataset’s similarity structure into short-range neighborhoods that contiguity-based operators (e.g., NSC segmentation/pooling) can exploit. In summary, ordering is most relevant in HDLSS-like regimes with low IDF or high FOE, and it is most likely to help when GO-LR also produces positive locality improvements over random orderings, reflected by higher LES.; conversely, for low-dimensional/high-IDF datasets, ordering is generally not required. Practically, we recommend a two-stage test: use low IDF / high FOE to flag datasets where ordering may help, and use positive locality gains (high LES) to predict when ordering will actually benefit architectures that rely on contiguity or local neighborhoods along the input sequence e.g., local-window attention/Transformer variants, state-space sequence models (e.g., Mamba-style SSMs), recurrent models (LSTM/GRU), and sequence-based LLM backbones since these models implicitly assume that nearby tokens/features should interact more strongly than distant ones. In our pipeline, NSC is not a backbone but a compression / dimensionality-reduction operator whose contiguous segmentation and pooling explicitly depends on meaningful neighborhoods; thus, when GO-LR induces stronger locality than random orderings (positive ΔAdjCoh , $\Delta\text{HitRate}_{K,h}$, ΔCut and higher LES), NSC-style compression, and potentially other locality-sensitive sequence models, are expected to benefit.

G. Detailed Comparative Results

Table G.1 reports the full 5×5 cross-validation results (mean accuracy with subscripted standard deviation) for all 8 HDLSS benchmarks and the complete set of 50+ baselines. Our method, GOTabPFN, attains the highest mean accuracy on every dataset (Colon, Lung, GLI, SMK, ALLAML, Prostate, Arcene, TOX), leading to an average rank of $1.00_{\pm 0.00}$ in the rightmost column; that is, it is consistently ranked first across all tasks and all CV folds. The absolute accuracies are also strong: GOTabPFN achieves at least 90% mean accuracy on 6 of 8 datasets (Lung, GLI, ALLAML, Prostate, Arcene, TOX), while maintaining competitive performance even on the most challenging HDLSS cases, such as SMK and ARC. On ARC, for instance, GOTabPFN reaches 90.60% accuracy, whereas the best competing methods remain in the mid-80% range, and on SMK it still leads the next-best model by a non-trivial margin. Across all datasets, the standard deviations of GOTabPFN are comparable to or smaller than those of the strongest baselines, indicating that the gains are not the result of a few lucky splits but are stable across repeated 5×5 CV.

The immediate competitors are other TabPFN-style models and modern HDLSS-focused baselines. TANDEM and TabPFN Wide form the closest group, with average ranks of $3.63_{\pm 1.32}$ and $3.75_{\pm 2.38}$, respectively. However, even these strong baselines lag behind GOTabPFN on every individual dataset: they never surpass our method on any of Colon, Lung, GLI,

SMK, ALLAML, Prostate, Arcene, or TOX, and their average ranks remain strictly higher. A second tier of competitive models includes TabDPT, TabICL, BETA, TuneTables, and well-regularized neural and boosted-tree baselines such as RealMLP, LGBM, and CatBoost, with average ranks roughly in the 4–18 range. These methods often perform reasonably on some datasets (e.g., LGBM and CatBoost on PRS and TOX, RealMLP on AML), but they either fall short on at least one particularly difficult HDLSS dataset (e.g., SMK or ARC) or show larger variability across splits, which results in clearly worse average ranks compared to GOTabPFN.

Classical shallow models and generic deep architectures occupy the middle of the table. Linear or margin-based methods (Lasso, SVM), tree ensembles (RF, XGBoost, GBM, AdaBoost), k -NN, and simple MLP variants (MLP, MLP-PLR, RealMLP) typically achieve moderate performance on most datasets, with average ranks in the low-to-mid 10–20 range. They can be competitive on a subset of benchmarks (e.g., RF and GBM on some of the easier tasks, SVM on GLI), but they do not exhibit the uniformly strong behavior of GOTabPFN and often degrade substantially on the most extreme HDLSS settings (e.g., SMK and ARC). Models designed primarily with feature selection or explainability in mind (STG, L2X, INVASE, REAL-X, ENODE, ModernNCA) also tend to underperform in this regime: while they occasionally match classical baselines on certain datasets, their overall average ranks (typically > 18 and often > 40) indicate that their inductive biases are not sufficient on their own to close the gap to GOTabPFN.

The bottom portion of the table is dominated by recent transformer- and Mamba-based tabular architectures originally developed and tuned for larger, non-HDLSS datasets. Methods such as FT-Transformer, SAINT, TabM, AutoInt, Category Embedding, ResNet Tabular, TabNet, NODE, DeepFM, DCN, DANets, TANGOS, Tab-Transformer, NDTF, MambaTab, Mambular, MambAttention, 1D CNN, and TabSeq generally obtain average ranks in the 30–45+ range and substantially lower accuracies on several HDLSS benchmarks. In many cases, these models struggle to surpass 70% accuracy on the hardest datasets and can even drop close to chance levels on some splits, highlighting a clear mismatch between their inductive biases (e.g., heavy overparameterization and large-context attention) and the HDLSS setting. Finally, the rows with “N/A” entries correspond to TabPFN variants for which results were not available on all 8 datasets (e.g., TabPFN-2.5 on COL only, and TabPFN v1/v2 and LoCalPFN without HDLSS evaluations); for completeness, we report their observed accuracies and overall average ranks in the rightmost column, but we exclude them from cross-dataset comparisons. Taken together, these detailed results show that GOTabPFN is not merely competitive but uniformly dominant across a broad and challenging suite of HDLSS benchmarks, outperforming both specialized TabPFN-style baselines and a wide spectrum of modern tabular architectures.

Evaluation beyond accuracy. To evaluate whether the gains of GOTabPFN extend beyond accuracy, we compare GOTabPFN with two strong tabular foundation model baselines, TabICL (Jingang et al., 2025) and TabDPT (Ma et al., 2025), using ROC-AUC and macro-F1 on the same 8 HDLSS datasets. As shown in Table G.2, GOTabPFN obtains the best ROC-AUC on 6/8 datasets and the best macro-F1 on 6/8 datasets, indicating that the proposed GO-LR+NSC representation improves not only top-line accuracy but also ranking quality and class-balanced predictive performance.

H. GOTabPFN Hyperparameters

HDLSS datasets. Table H.1 reports the best-performing GOTabPFN configurations across eight HDLSS benchmarks, showing that the GO-LR stage adapts its distance metric (euclidean/manhattan/correlation/KL) and clustering granularity ($k=4$ –12) per dataset with only 1–3 refinement passes, while NSC consistently favors high retention thresholds ($\tau \approx 0.99$, except ALLAML at 0.95) and typically uses the gamma-based rule (with Colon using IDF and Lung/SMK using the default rule). Segmentation is dataset-dependent uniform for several datasets, but equal-mass for Colon/TOX and largest-jump for ALLAML/Prostate indicating that both tokenization strategy and compression hyperparameters ($\gamma, \beta, M_{\min/\max}, l_{\min}$) must be tuned to match the underlying feature geometry; only SMK employs feature subsampling and only SMK/Arcene enable assume_standardized, while TabPFN random seeds vary modestly across datasets.

Cross-domain datasets. Table H.2 reports the best-performing GOTabPFN configurations on the 8 additional cross-domain datasets. Similar to the HDLSS setting, GO-LR adapts both the metric and clustering granularity to each dataset, using cosine, Manhattan, correlation, and KL-based dissimilarities with $k=4$ –11 clusters and only 1–2 refinement passes. Most cross-domain datasets favor uniform NSC segmentation and the default M -rule, while Cell Cycle and both DrivFace tasks use the IDF rule, and DrivFace additionally benefits from largest-jump segmentation. Feature subsampling is used for most high-dimensional cross-domain datasets, especially ORL, RELATHE, PCMAC, Cell Cycle, CIFAR-10, and DrivFace, reflecting the larger feature spaces in this evaluation. The selected configurations also show that standardization is useful

for most cross-domain settings except ORL, while TabPFN seeds vary modestly across datasets. Overall, the table shows that GOTabPFN remains flexible across text, image-feature, camera sensor, and RNA-seq domains by adapting the feature graph, segmentation rule, and compression budget to the geometry of each dataset. Prior Labs notes that TabPFN inference is deterministic for a fixed seed in the same environment, while small differences may occur across different hardware configurations.¹

I. Statistical Significance Analysis

Significance analysis on HDLSS datasets. We evaluate statistical differences across methods using (i) a Friedman test (Friedman, 1937) over per-dataset ranks, followed by a Nemenyi post-hoc critical-difference (CD) analysis (Nemenyi, 1963) (Fig. I.1), and (ii) pairwise Wilcoxon signed-rank tests (Demšar, 2006) comparing GOTabPFN to each baseline across the same 8 datasets, with Holm correction to control family-wise error (Table I.1). The Friedman test indicates a significant overall effect across methods, and the CD diagram visualizes the separation in average ranks, where GOTabPFN attains the lowest (best) average rank. For pairwise tests, the raw Wilcoxon p -values are identical across baselines ($p_{\text{raw}} = 0.00781$), reflecting that GOTabPFN improves over each comparator on all datasets with no sign reversals (a common outcome when $n = 8$ and per-dataset differences are consistently positive). After Holm correction, the adjusted p -values become more conservative ($p_{\text{Holm}} = 0.0703$), so we do not claim strict significance at $\alpha = 0.05$ under family-wise error control; nevertheless, the combination of uniform wins, rank dominance, and consistent positive paired differences supports the robustness of the observed improvements.

Extended statistical significance analysis. We further evaluate statistical significance on an expanded 16-dataset benchmark formed by combining the 8 HDLSS datasets with the 8 cross-domain datasets, using the strongest common comparison set from the main HDLSS and cross-domain experiments. Table I.2 summarizes the average ranks: GOTabPFN achieves the best average rank by a clear margin (1.12), followed by TabPFN-Wide (3.62), TANDEM (3.69), and TabDPT (4.34). The same ranking trend is visualized in Fig. I.2, where GOTabPFN is separated from the nearest competing methods by more than two average-rank points. As summarized in Table I.3, the omnibus Friedman test over the 9-method comparison is strongly significant ($\chi^2 = 52.55$, $p = 1.32 \times 10^{-8}$; 11 complete rows used), rejecting the null hypothesis that all methods have equal rank distributions across the expanded benchmark. We then compare GOTabPFN against each baseline using pairwise Wilcoxon signed-rank tests with Holm correction. Table I.4 shows that GOTabPFN remains statistically significant against every baseline after correction, with Holm-adjusted p -values between 2.44×10^{-4} and 6.10×10^{-4} . The win/tie/loss counts are uniformly favorable: 16/0/0 against Lasso, TabDPT, TabPFN-Wide, and TuneTables; 14/0/0 against TabICL on the common subset; 12/0/0 against ProtoGate on the common subset; and 15/0/1 against MLP and TANDEM. Table I.5 further reports the mean accuracy gain and Holm-corrected pairwise significance of GOTabPFN against each strong baseline on the expanded 16-dataset benchmark. The Holm-adjusted p -values remain below 0.01 for all pairwise comparisons, indicating that GOTabPFN is significantly better than each baseline after controlling for multiple comparisons. Together, these results complement the 55-baseline analysis on the original 8 HDLSS datasets by showing that GOTabPFN remains robust across a broader 16-dataset evaluation against the strongest repeated baselines.

J. Additional Ablation Analysis

Fig. J.7 reports accuracy distributions across CV splits for the top-10 methods, showing that GOTabPFN achieves the strongest central tendency with competitive dispersion, i.e., high average performance without depending on a small number of favorable splits. Consistently, the average-rank vs. global-mean scatter in Fig. J.4 places GOTabPFN in the top-left regime (lowest average rank and highest global accuracy). The normalized per-dataset accuracies in Fig. J.2 and the dataset-wise rank breakdown as a function of sample size in Fig. J.3 further indicate that the gains persist across datasets of different sizes rather than being driven by a single benchmark. Fig. J.6 summarizes the best-second-best gaps per dataset, highlighting where the leading method separates more clearly from the runner-up (notably on the harder benchmarks), while Fig. J.5 localizes improvements by visualizing ΔAcc (ours – baseline) across datasets and competitors, revealing broadly positive deltas with the largest separations against simpler baselines on challenging tasks. Finally, Fig. J.1 characterizes distributional shape via skewness and kurtosis computed over per-dataset accuracies, where negative skewness reflects a small number of difficult datasets that pull performance downward and higher kurtosis for several baselines suggests heavier tails and greater

¹Prior Labs FAQ: <https://docs.priorlabs.ai/faq>. See also the PriorLabs/TabPFN reproducibility discussion, issue #266: <https://github.com/PriorLabs/TabPFN/issues/266>.

instability compared to the most consistent top performers.

K. Representation Quality via t-SNE

Figure K.1 visualizes the NSC token/latent representation learned by GOTabPFN on the Colon dataset using a 2D t-SNE (Maaten & Hinton, 2008) embedding (points colored by class). Each point corresponds to a sample after GO-LR ordering and NSC compression (PCA-based segmentation). The plot indicates that the compressed representation preserves class-discriminative structure in a low-dimensional manifold: samples from the two classes exhibit partially separated regions with limited overlap, suggesting that NSC produces a structured embedding that is more amenable to the downstream TabPFN-2.5 head under the HDLSS regime.

L. Inference Level Ablation on Calibration and Robustness

On Colon, we complement the main accuracy results with inference-time diagnostics that probe robustness, selectivity, neighborhood structure, and calibration. First, robustness to feature perturbations (Table L.1 and Fig. L.1b) shows a graceful degradation as the perturbed fraction increases: accuracy remains near-ceiling under mild corruption (e.g., $\leq 10\%$) but drops substantially under heavy perturbations, with shuffling generally more harmful than mean-imputation at moderate rates (e.g., at 50%: 74.19% vs. 98.39%). Second, selective prediction behaves as expected (Fig. L.1a): increasing the confidence threshold τ raises accuracy on the retained “confident” subset while reducing coverage, indicating that model confidence meaningfully ranks predictions by correctness. Third, local consistency in the NSC latent space remains stable (Table L.1 and Fig. L.1d), with kNN label agreement around 73.87% at $k=5$ across this evaluation, suggesting a reasonably coherent neighborhood geometry after compression. Finally, the reliability diagram in Fig. N.1 reports a low expected calibration error (ECE ≈ 0.033), indicating that predicted probabilities are well-aligned with empirical accuracy on this dataset.

M. Sanity and Stress Diagnostics

Figure M.1 summarizes additional sanity and stress tests for GOTabPFN on Colon. As shown in Fig. M.1a, performance is near-ceiling on the full input (98.39%), but drops sharply under degenerate signals (all-zero or global-mean inputs both 64.52%), and further degrades when the feature rows are randomly permuted (54.84%), confirming that predictions depend on meaningful sample-specific structure rather than trivial priors. Notably, accuracy remains high under strong additive noise (95.16%), suggesting robustness to moderate distributional corruption in feature values. We also evaluate a simple tabular test-time augmentation (TTA) procedure (Fig. M.1b): majority voting over $n_{\text{aug}}=5$ noisy/dropout augmentations matches the base accuracy (both 98.39%), and only 3.23% of samples change their predicted label under any augmentation, indicating high prediction stability. Table M.1 reports these stress-test and stability numbers alongside per-class accuracy, showing uniformly strong performance across classes (97.5% on the majority class with support 40, and 100% on the minority class with support 22), consistent with the robustness patterns observed in Fig. M.1.

N. Additional Reliability and Interpretability Diagnostics

On the Colon benchmark, GOTabPFN achieves a baseline Top-1 accuracy of 98.39% and reaches 100% Top-2 accuracy (Fig. N.1b). The normalized confusion matrix indicates a single error case, with the most frequent confusion being true class $0 \rightarrow 1$ occurring once (Fig. N.1c). Confidence is well-separated: the mean margin $p_{\text{top1}} - p_{\text{top2}}$ is high for correct predictions (0.949) and near-zero for the lone incorrect prediction (0.025), yielding a clear bimodal separation (Fig. N.1a). Finally, per-feature permutation importance on this small evaluation set shows 0.0 percentage-point accuracy drop for the top-ranked features, consistent with near-saturated accuracy and limited headroom for measurable single-feature perturbation effects under this diagnostic protocol.

O. Theory-Inspired Representation Diagnostics

We analyze the learned embedding geometry and confidence behavior of GOTabPFN on Colon, where the model attains 98.39% top-1 accuracy. The embedding spectrum in Fig. O.1a is strongly low-rank, with an effective dimension (participation ratio) of 3.5, and the cumulative curve in Fig. O.1b shows that only 2/4/6/11/29 components capture 50/80/90/95/99% of the variance, respectively, indicating a highly concentrated representation. Despite this compression, local-neighborhood classifiers are insufficient: leave-one-out kNN in embedding space peaks at 82.26% (at $k=5$) and degrades for larger k

(Fig. O.1c), remaining well below the parametric TabPFN head, suggesting the decision rule leverages more than simple Euclidean locality. Finally, margin diagnostics confirm strong separation and calibrated confidence: the mean normalized margin is 0.9707 for correct predictions versus -0.0359 for incorrect ones, and the conditional error stays near 0 across a wide range of margin thresholds while maintaining near-full coverage (Fig. O.1d-e). Together, these results support that GOTabPFN forms a low-dimensional, sharply separated embedding while relying on a richer (non-kNN) parametric decision mechanism.

P. OOD and Local Sensitivity Diagnostics

On Colon, GOTabPFN achieves 98.39% ID accuracy and exhibits highly confident and low-entropy predictions on ID inputs (mean max-softmax confidence = 0.967, mean entropy = 0.106; Fig. P.1a-P.1b). Under synthetic OOD-style tabular inputs, uncertainty increases: Gaussian noise and column-wise permutation both reduce confidence (mean ≈ 0.810 and 0.795) and raise entropy (mean ≈ 0.424 and 0.446), while constant “blank” features yield intermediate behavior (confidence 0.883, entropy 0.360), indicating that the model does not collapse to uniformly overconfident predictions off-manifold (Fig. P.1a-P.1b). Finally, a local Lipschitz-like probe under small feature perturbations ($\epsilon = 0.10$) shows a distribution concentrated at low $\|\Delta\text{probs}\|_2/\|\Delta x\|_2$ with a modest tail (mean 0.041, median 0.015; Fig. P.1c), suggesting that the learned predictor is generally stable to small tabular noise while allowing occasional locally sensitive regions.

Q. Deployment-Oriented Triage Diagnostics

To clarify deployment behavior beyond multiclass accuracy, we cast Colon as a triage task by treating class 1 as the positive (“high-risk”) class and sweeping a decision threshold over the predicted probability $p(y=1 \mid x)$. As shown in Fig. Q.1, the resulting ROC achieves $\text{AUC} = 1.000$, indicating perfect separability between positives and negatives on this evaluation set ($N = 62$). Using a sensitivity-driven operating constraint (target sensitivity 0.95), the selected threshold is $\text{th}^* = 0.7885$, which attains sensitivity = 1.000 and specificity = 1.000; the induced confusion matrix is $TN=40, FP=0, FN=0, TP=22$, yielding 100% precision/recall and 100% binary accuracy at this operating point. Finally, we report wall-clock latency over 20 random mini-batches (batch size 64) with mean $\approx 639\text{ms}$ per batch (p50 $\approx 638\text{ms}$, p90 $\approx 644\text{ms}$, p99 $\approx 649\text{ms}$), providing a coarse throughput reference for deployment-oriented settings.

R. Extension beyond TabPFN

GO-LR + NSC is not tied to TabPFN; it is a model-agnostic representation interface that can be paired with other tabular foundation models. We use TabPFN-2.5 (Grinsztajn et al., 2025) as the main backbone because its feature-dimensionality bottleneck is especially clear, but the same front-end also transfers to TabICL (Jingang et al., 2025), as shown in Table R.1. Across the same 8 HDLSS datasets, GO-LR + NSC + TabICL improves over vanilla TabICL on 5/8 datasets in accuracy, 7/8 in ROC-AUC, and 5/8 in macro-F1. Importantly, it also reduces runtime on all 8 datasets, with especially large gains on very high-dimensional cases such as GLI, SMK, and ARC. These results suggest that GO-LR + NSC acts as a broader HDLSS-oriented representation layer rather than a TabPFN-specific preprocessing trick.

S. TabPFN Seed Sensitivity

Dataset-seed vs. TabPFN-seed robustness. We distinguish two sources of randomness in our evaluation. A dataset seed controls the train/validation/test partition or CV folds, and therefore measures data-split robustness: whether conclusions persist across different sampled splits of the same small HDLSS dataset. This is the main source of evaluation variability in our setting, since small tabular datasets can be highly split-sensitive (Rubachev et al., 2025; Grinsztajn et al., 2022; Bouthillier et al., 2021); accordingly, our main experiments use repeated 5×5 CV following ProtoGate (Jiang et al., 2024) to average over multiple dataset splits. In contrast, a TabPFN seed controls the `random_state` or equivalent stochastic components inside the TabPFN inference/configuration pipeline, and therefore measures model/inference stochasticity while holding the data split fixed; such run and distribution-level variance is also a known concern in neural-network evaluation (Jordan, 2024). Prior Labs provides the official TabPFN classification interface,² and maintainers discuss fixed-`random_state` reproducibility for TabPFN in the implementation repository.³ Recent TabPFN studies also report

²Prior Labs classification documentation: <https://docs.priorlabs.ai/capabilities/classification>.

³PriorLabs/TabPFN reproducibility discussion, issue #266: <https://github.com/PriorLabs/TabPFN/issues/266>.

averages over multiple seeds (Ye et al., 2025b). Thus, varying dataset seeds tests evaluation robustness, whereas varying TabPFN seeds isolates model-seed sensitivity. In our main HDLSS experiments, we prioritize repeated 5×5 CV for split robustness and use the Optuna-tuned best TabPFN seed per dataset to avoid underestimating TabPFN from an unfavorable inference seed; fixed-split multi-TabPFN-seed results are supplementary analyses of model-seed variance.

Findings from TabPFN-seed robustness analysis. Tables S.1, S.2, S.3, S.4, and S.5 show that GOTabPFN remains consistently strong across TabPFN seeds, not only under one favorable random state. On the 8 HDLSS datasets, GOTabPFN achieves the best average accuracy for every tested seed: 90.57, 89.66, 89.93, 89.79, and 89.66, compared with the strongest competing averages of roughly 88-88.5 from TabPFN-Wide (Kolberg et al., 2025) or TuneTables (Feuer et al., 2024). The detailed per-dataset table further shows that GOTabPFN is especially effective on difficult high-dimensional datasets such as GLI, ARC, SMK, TOX, AML, and LNG, although some baselines occasionally win on individual datasets such as PRS or TOX for particular seeds. On the 8 cross-domain datasets, GOTabPFN is also the strongest complete method across all seeds, with averages around 86.6-86.9, while TabPFN-Wide and TuneTables vary more substantially across seeds. The only higher BETA (Liu & Ye, 2025) average is reported for seed 42 over 6 datasets only, because Cell Cycle and DrivFace-Regression were omitted due to runtime; therefore, that number is not directly comparable to the complete 8-dataset averages. For ROC-AUC on the HDLSS benchmark, GOTabPFN again achieves the best average in 4 of 5 seeds and is nearly tied with TabPFN-Wide at seed 93 (93.71 vs. 93.74), indicating that the gains are not limited to accuracy but also largely persist in ranking quality. Overall, these results distinguish GOTabPFN from other high-dimensional TabPFN variants: TabPFN-Wide, BETA, and TuneTables can process larger feature spaces, but they still rely primarily on the foundation-model predictor or tuning strategy to absorb high-dimensional structure. GOTabPFN instead first reorganizes the feature space through GO-LR and compresses locally coherent neighborhoods through NSC, yielding a compact and more stable representation before the frozen TabPFN-2.5 head. This structured front-end explains why GOTabPFN remains competitive or superior across seeds: it reduces the burden on the predictor, preserves locality among related features, and provides a more robust HDLSS-specific interface than simply widening, tuning, or directly applying TabPFN-style models to high-dimensional inputs.

Pareto frontier of runtime and performance. We further evaluate the accuracy-efficiency trade-off among high-dimensionality compatible TabPFN-family methods using mean runtime and mean performance under TabPFN seed 42. As shown in Fig. S.1, GOTabPFN lies on the Pareto frontier on both benchmarks. On the original 8 HDLSS datasets, GOTabPFN achieves the highest mean performance while remaining substantially faster than BETA and only moderately slower than TabPFN-Wide and TuneTables, yielding the best overall trade-off among the compared methods. On the additional 8 cross-domain datasets, where both sample size and dimensionality increase beyond the original HDLSS-only benchmark, GOTabPFN again remains Pareto optimal: it achieves the best mean performance while requiring lower runtime than TabPFN-Wide and TuneTables. These results indicate that the GO-LR+NSC front-end does not merely improve accuracy by adding excessive computation; rather, it provides an efficient structured compression interface that improves the performance-runtime balance of TabPFN-style prediction in high-dimensional and larger-sample regimes.

Random seed sensitivity on Colon. We further isolate TabPFN inference-seed sensitivity on the Colon dataset by fixing the best GO-LR+NSC configuration from Table H.1 and varying only the TabPFN inference seed over $\{0, 1, 2, 3, 4, 7, 11, 17, 23, 42\}$. Preprocessing, GO-LR, NSC, and the exact same 5×5 CV splits are kept fixed, so any variation comes only from the TabPFN seed. As shown in Table S.6, the results are tightly clustered: the across-seed mean accuracy is 87.19, with an across-seed standard deviation of only 0.46 and a full range of 86.56-88.18 percentage points. Thus, while TabPFN seed introduces mild variation, the Colon result is not driven by an exceptionally favorable stochastic realization. This is particularly relevant because the strongest Colon baselines are also TabPFN-family methods: GOTabPFN (88.18), TabPFN-Wide (87.85), and TuneTables (86.80). Under this like-for-like comparison, GOTabPFN remains the best-performing TabPFN-family method, supporting that its gain comes from the GO-LR+NSC representation interface rather than seed luck alone.

T. Additional Clarifications

Clarity of graph-based feature ordering. To make the graph-based ordering step easier to follow, the main paper introduces GO-LR with an intuitive figure before the formal MinLA-based development. Here, we provide additional

clarification. The goal of GO-LR is to place statistically related features close to one another on a one-dimensional axis, so that subsequent NSC segmentation groups coherent neighborhoods rather than arbitrary columns. Equivalently, GO-LR treats features as nodes in a weighted feature graph, where stronger edges indicate stronger feature relationships, and seeks a linear arrangement in which strongly connected nodes remain nearby. For example, if f_1 is strongly related to both f_3 and f_4 , while f_5 is comparatively independent, an ordering such as $[f_3, f_1, f_4, f_2, f_5]$ is preferable to $[f_1, f_2, f_3, f_4, f_5]$, because the related features become contiguous and can be compressed more meaningfully. This intuition is illustrated in Fig. 1 in the main paper. Formally, this corresponds to a Minimum Linear Arrangement (MinLA)-style objective that penalizes placing strongly related features far apart. Since exact optimization is combinatorial and intractable at scale, GO-LR uses a TSP-style nearest-neighbor path as an efficient initialization and then applies local refinement under the MinLA-style dispersion objective.

Clarifying meta-feature construction. To make the NSC representation interface more self-contained, we explicitly define a meta-feature as the low-dimensional token obtained by compressing one contiguous segment of the GO-LR-ordered feature axis. Let Π^* denote the global feature ordering, $\{\mathcal{S}_t\}_{t=1}^M$ the contiguous ordered segments, and $u_t = x_{\mathcal{S}_t}^{\Pi^*}$ the subvector of sample x restricted to segment $\mathcal{S}_t$. The t -th meta-feature is then $z_t = g(u_t)$, where $g(\cdot)$ is a segment-level pooling or projection operator. In our main NSC-pSP instantiation, $g(\cdot)$ is implemented by segment-wise PCA projection, producing one scalar token per ordered segment. The final compressed representation is therefore $Z(x) = (z_1, \dots, z_M)$, which is passed to the frozen TabPFN-2.5 head. Fig. 2 in the main paper illustrates this process: GO-LR first reorders the original features, NSC partitions the ordered axis into contiguous neighborhoods, and each neighborhood is compressed into a meta-feature. Additional details on ordered segmentation, subunit pooling, and meta-feature construction are provided in Sec. 3.2 in the main paper.

Similarity vs. dissimilarity metric. We use $1 - |\text{corr}(i, j)|$ as a dependence-aware dissimilarity so that strongly coupled feature pairs have small distance regardless of sign. Concretely, both strongly positive and strongly negative correlations satisfy $|\text{corr}(i, j)| \approx 1$, hence $d_{ij} = 1 - |\text{corr}(i, j)| \approx 0$. This is the intended behavior for GO-LR: the goal is not to preserve the sign of association, but to place strongly dependent or redundant features into the same local neighborhood before compression. This choice is also consistent with our neuro-inspired motivation. In Sec. 3.2, we discuss evidence that dendritic inputs are organized into local subunits rather than summed globally, and that correlated synapses may exhibit local clustering within such compartments. Our algorithmic analogue is therefore that GO-LR first brings strongly coupled features close along the ordered axis, and NSC then pools these local neighborhoods into subunit-level meta-features. In this sense, $1 - |\text{corr}(i, j)|$ should be read as a practical measure of lack of coupling, chosen to support local clustering and subunit-style aggregation, rather than as a broader semantic notion of dissimilarity.

Cross-domain evaluation. To evaluate whether GOTabPFN generalizes beyond biomedical HDLSS datasets, we extend the benchmark with 8 additional cross-domain datasets spanning text, face images, camera-sensor data, image features, and RNA-seq measurements. These datasets cover HDLSS, HDHSS, and mixed regimes under the empirical categorization rule adopted from DynaTab (Habib et al., 2026b) and expanded in Appendix F. This extension is important because recent HDLSS-specific tabular models are still evaluated largely on biomedical benchmarks, with only limited coverage of text, face, or sensor domains. For example, ProtoGate (Jiang et al., 2024) evaluates primarily on 7 biomedical datasets, while LSPIN/LLSPIN (Yang et al., 2022a) reports real-world experiments on 3 text and 3 biomedical datasets. Similarly, high-dimensional TabPFN-style extensions remain limited in domain coverage: TabPFN-Wide (Kolberg et al., 2025) uses 4 biomedical datasets, BETA (Liu & Ye, 2025) includes a mixture of biomedical, text, and face-image datasets, and TuneTables (Feuer et al., 2024) is evaluated through the broader TabZilla benchmark (McElfresh et al., 2023). We did not identify a clear real-world financial HDLSS dataset suitable for inclusion in this evaluation. The resulting cross-domain suite, summarized in Table T.1, therefore broadens the empirical scope of our study while retaining high-dimensional settings where feature ordering and locality-aware compression are relevant.

Cross-domain dominance. Fig. T.1 shows that GOTabPFN generalizes beyond its primary HDLSS target regime to a broader cross-domain benchmark. Across the 8 additional datasets, GOTabPFN ranks first on 7/8 datasets, with positive margins over the strongest competing method on ORL (+0.80), BAS (+0.07), PCM (+0.52), Cell Cycle (+1.27), CIFAR-10

(+0.30), DrivFace-R (+0.0043), and DrivFace-C (+1.15). The only exception is RELATHE, where GOTabPFN remains competitive and trails the best baseline by only 0.55 points. These results indicate that, even with limited tuning, the GO-LR+NSC pipeline remains highly effective on HDLSS-style datasets while also preserving competitive performance in adjacent high-dimensional and mixed-regime settings. The larger margins on ORL, Cell Cycle, and DrivFace-C further suggest that locality-aware feature ordering and compression are particularly beneficial when the data retain strong high-dimensional structure.

Clarifying tuning fairness. Our tuning protocol follows a distinction that is common in recent tabular-learning practice and is summarized in Fig. A.1 in Appendix A. On one side are PFN/ICL-style tabular foundation models, which are typically used close to off-the-shelf because much of the modeling burden is absorbed during pre-training. For example, the official TabICL (Jingang et al., 2025) repository states that TabICL does not require preprocessing or hyperparameter tuning,⁴ and the official TabDPT (Ma et al., 2025) repository similarly describes TabDPT as an ICL-based tabular foundation model that generalizes to new tasks without additional training or hyperparameter tuning.⁵ On the other side are conventional tuned tabular learners, such as TabR (Gorishniy et al., 2024), TabM (Gorishniy et al., 2025), RealMLP (Holzmüller et al., 2024), XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), and LightGBM (Ke et al., 2017), whose standard use often involves dataset-specific hyperparameter search. GOTabPFN naturally lies between these two regimes: the GO-LR+NSC front-end is dataset-adaptive and therefore tunable, while the downstream TabPFN-2.5 (Grinsztajn et al., 2025) predictor remains a frozen pre-trained backbone that is neither retrained nor structurally modified. Thus, the substantive dataset-specific adaptation in GOTabPFN occurs only in the representation interface before TabPFN inference, not in the foundation-model backbone itself. Regarding the TabPFN seed in Table H.1, this seed is not a trainable parameter and does not modify the backbone weights; it is a fixed inference-time configuration selected from the same predefined set under the same outer Optuna (Akiba et al., 2019) study as the front-end hyperparameters (see Appendix S for TabPFN seed sensitivity). Therefore, GOTabPFN should be viewed as a hybrid tunable front-end attached to a frozen PFN-style predictor, rather than as a fully tuned end-to-end tabular learner. See Table T.2 for the source URLs of baselines.

Evaluation scope and statistical validation. We provide a consolidated view of the expanded benchmark and statistical analyses supporting our empirical conclusions. The main paper reports results on the original 8 HDLSS datasets in Sec. 4, including the top-model summary in Table 1; Appendix G further provides the full 55-baseline comparison in Table G.1. To broaden the benchmark beyond the original biomedical-heavy HDLSS setting, we add 8 cross-domain datasets spanning text, face-image, camera-sensor, image-feature, and RNA-seq domains, with results summarized in Table 2. Thus, the final evaluation covers both the targeted HDLSS regime and a broader 16-dataset cross-domain setting. For statistical validation, Appendix I provides an expanded significance analysis: Fig. I.2 and Table I.2 show that GOTabPFN achieves the best average rank on the 16-dataset benchmark; Table I.3 reports a significant omnibus Friedman test across the 9-method comparison; and Tables I.4 and I.5 show that GOTabPFN remains significantly better than each strong repeated baseline after Holm correction. We also contextualize our evaluation protocol relative to closely related HDLSS and high-dimensional tabular studies. Prior HDLSS-specific work often relies on small but carefully curated benchmark suites and reports rank-based summaries to compare methods across heterogeneous datasets. For example, ProtoGate (Jiang et al., 2024) evaluates on 7 biomedical HDLSS datasets against 16 baselines and reports average rank as a primary aggregate measure, while LSPIN/LLSPIN (Yang et al., 2022a) evaluates on 6 real-world high-dimensional datasets, including 3 text and 3 biomedical datasets, and summarizes performance using median rank. Following this established practice, we report average ranks and statistical tests, but we further expand the evidence by evaluating GOTabPFN against 55 baselines on the original 8 HDLSS datasets and against the strongest repeated baselines on an expanded 16-dataset benchmark. Overall, the conclusions are not based only on the original 8-dataset rank summary, but are supported by detailed 55-baseline HDLSS comparisons, an additional 8-dataset cross-domain evaluation, and both omnibus and pairwise statistical tests on the expanded 16-dataset benchmark.

GOTabPFN in extreme dimensionality. GOTabPFN is evaluated across a broad high-dimensional range, with feature counts spanning from $m = 2,000$ at the lower end of our HDLSS benchmark to $m = 42,728$ on the Cell Cycle RNA-seq

⁴TabICL official repository: <https://github.com/soda-inria/tabicl>.

⁵TabDPT official repository: <https://github.com/layer6ai-labs/TabDPT-inference>.

dataset with $n = 1067$ samples. On this largest dataset, GOTabPFN remains fully operational and achieves 79.94 ± 2.53 accuracy, 92.36 ± 1.36 AUC, and 79.95 ± 2.51 macro-F1. These results show that the GO-LR+NSC representation interface scales beyond moderate HDLSS feature counts and remains effective in substantially larger transcriptomic feature spaces, where direct use of TabPFN-style predictors would otherwise be constrained by the extreme dimensionality.

Theoretical grounding, novelty, and HDLSS relevance. The theoretical results in Sec. 3.1 are not intended as isolated complexity-theoretic contributions; rather, they formalize why GO-LR is a principled ordering mechanism. Specifically, the MinLA formulation identifies the objective that GO-LR approximates, the NP-hardness result explains why exact scalable optimization is unrealistic, and the TSP-style path construction motivates an efficient surrogate initialization before local MinLA-based refinement. The overall contribution is therefore best understood as an integrated HDLSS-oriented framework: GO-LR provides a theoretically grounded feature ordering objective, NSC converts the ordered axis into stable meta-features through structured local compression, and the resulting representation interface enables a frozen TabPFN-style backbone to operate effectively beyond its native feature counts’ limits. This integration yields a new HDLSS-oriented tabular foundation model interface in which feature ordering, locality-preserving compression, and frozen TabPFN-style inference are jointly aligned to overcome the dimensionality bottleneck of existing tabular foundation models.

Fine-tuning as future work. An important future direction is to pretrain or fine-tune a TabPFN-style backbone directly on structured representations produced by GO-LR+NSC. In principle, this could be done by generating large collections of synthetic HDLSS-style tasks, applying graph-guided ordering and subunit compression, and then adapting the backbone to operate natively in the resulting meta-feature space. A further extension would be to initialize from the existing TabPFN checkpoint and pretrain or fine-tune the full GO-LR+NSC+TabPFN pipeline end-to-end, so that the entire model becomes a pretrained HDLSS-oriented foundation model rather than a tuned front-end attached to a frozen predictor. Such a model could potentially reduce or eliminate dataset-specific tuning at inference time, because the ordering, compression, and prediction components would be jointly adapted during pretraining. However, this would shift the focus from representation-side adaptation of an existing frozen backbone to the development of a new HDLSS-specific tabular foundation model. We therefore view backbone adaptation or full-pipeline pretraining on GO-LR+NSC representations as a promising but distinct direction beyond the present scope.

Table D.1. NSC variants vs. PCA/AE/RP/UMAP/PaCMAP on the block-structured synthetic HDLSS model with $M = 32$ (10 independent reps).

Method	Dim.	Lin. Acc.	kNN Acc.	Sil.	DB	Method	Dim.	Lin. Acc.	kNN Acc.	Sil.	DB
BlockModel-Rep0						BlockModel-Rep5					
NSC-pSP (ours)	32	0.825	0.675	0.047	3.802	NSC-pSP (ours)	32	0.875	0.688	0.064	3.577
NSC-SP (ours)	32	0.825	0.662	0.044	3.963	NSC-SP (ours)	32	0.838	0.788	0.048	3.729
NSC (ours)	32	0.775	0.713	0.035	4.291	NSC (ours)	32	0.775	0.625	0.030	4.717
NSC-P (ours)	32	0.825	0.688	0.038	4.130	NSC-P (ours)	32	0.600	0.588	0.021	5.910
PCA	32	0.725	0.588	0.035	4.354	PCA	32	0.713	0.562	0.028	4.842
AE	32	0.762	0.725	0.041	4.094	AE	32	0.738	0.700	0.036	4.433
RP	32	0.713	0.650	0.038	4.098	RP	32	0.700	0.650	0.030	4.365
UMAP	32	0.789	0.652	0.068	3.502	UMAP	32	0.684	0.639	0.001	5.213
PaCMAP	32	0.776	0.742	0.116	3.783	PaCMAP	32	0.697	0.584	-0.004	5.677
BlockModel-Rep1						BlockModel-Rep6					
NSC-pSP (ours)	32	0.825	0.688	0.043	3.986	NSC-pSP (ours)	32	0.825	0.675	0.053	3.684
NSC-SP (ours)	32	0.762	0.775	0.045	3.984	NSC-SP (ours)	32	0.863	0.725	0.058	3.525
NSC (ours)	32	0.713	0.588	0.034	4.493	NSC (ours)	32	0.788	0.625	0.034	4.525
NSC-P (ours)	32	0.675	0.613	0.028	5.008	NSC-P (ours)	32	0.750	0.575	0.023	5.688
PCA	32	0.763	0.625	0.032	4.643	PCA	32	0.775	0.675	0.030	4.596
AE	32	0.725	0.638	0.036	4.418	AE	32	0.825	0.625	0.032	4.562
RP	32	0.588	0.550	0.022	5.119	RP	32	0.812	0.675	0.028	4.715
UMAP	32	0.736	0.612	-0.006	5.827	UMAP	32	0.855	0.811	0.088	3.398
PaCMAP	32	0.749	0.637	-0.007	5.720	PaCMAP	32	0.842	0.847	0.061	4.598
BlockModel-Rep2						BlockModel-Rep7					
NSC-pSP (ours)	32	0.875	0.713	0.046	3.886	NSC-pSP (ours)	32	0.850	0.800	0.071	3.034
NSC-SP (ours)	32	0.862	0.800	0.053	3.539	NSC-SP (ours)	32	0.775	0.650	0.034	4.357
NSC (ours)	32	0.725	0.600	0.025	4.786	NSC (ours)	32	0.850	0.800	0.057	3.628
NSC-P (ours)	32	0.613	0.575	0.022	5.891	NSC-P (ours)	32	0.738	0.613	0.028	4.858
PCA	32	0.750	0.600	0.030	4.690	PCA	32	0.838	0.675	0.030	4.608
AE	32	0.688	0.625	0.030	4.775	AE	32	0.850	0.825	0.036	4.144
RP	32	0.688	0.538	0.026	4.801	RP	32	0.788	0.688	0.028	4.648
UMAP	32	0.894	0.797	0.091	3.423	UMAP	32	0.723	0.705	0.021	4.561
PaCMAP	32	0.868	0.834	0.047	4.643	PaCMAP	32	0.684	0.649	0.029	4.820
BlockModel-Rep3						BlockModel-Rep8					
NSC-pSP (ours)	32	0.850	0.725	0.031	4.630	NSC-pSP (ours)	32	0.850	0.650	0.038	4.283
NSC-SP (ours)	32	0.900	0.788	0.041	4.176	NSC-SP (ours)	32	0.725	0.775	0.045	3.958
NSC (ours)	32	0.775	0.662	0.035	4.517	NSC (ours)	32	0.725	0.725	0.048	3.904
NSC-P (ours)	32	0.725	0.588	0.019	6.260	NSC-P (ours)	32	0.863	0.700	0.030	4.636
PCA	32	0.775	0.600	0.030	4.672	PCA	32	0.762	0.613	0.028	4.745
AE	32	0.800	0.763	0.034	4.493	AE	32	0.763	0.762	0.030	4.739
RP	32	0.688	0.600	0.023	5.047	RP	32	0.650	0.625	0.018	5.819
UMAP	32	0.696	0.666	0.021	4.404	UMAP	32	0.762	0.719	0.038	4.193
PaCMAP	32	0.670	0.584	0.037	4.591	PaCMAP	32	0.828	0.782	0.089	4.093
BlockModel-Rep4						BlockModel-Rep9					
NSC-pSP (ours)	32	0.863	0.675	0.033	4.437	NSC-pSP (ours)	32	0.850	0.688	0.034	4.579
NSC-SP (ours)	32	0.812	0.750	0.032	4.620	NSC-SP (ours)	32	0.762	0.762	0.033	4.527
NSC (ours)	32	0.763	0.625	0.033	4.596	NSC (ours)	32	0.750	0.625	0.029	4.755
NSC-P (ours)	32	0.675	0.613	0.024	5.742	NSC-P (ours)	32	0.600	0.562	0.019	6.010
PCA	32	0.725	0.650	0.029	4.729	PCA	32	0.775	0.600	0.026	4.853
AE	32	0.712	0.650	0.031	4.631	AE	32	0.725	0.638	0.030	4.658
RP	32	0.688	0.612	0.020	5.389	RP	32	0.712	0.575	0.018	5.266
UMAP	32	0.802	0.744	0.038	4.196	UMAP	32	0.723	0.705	0.059	3.659
PaCMAP	32	0.723	0.715	0.037	4.748	PaCMAP	32	0.737	0.677	0.005	5.489
Method	Dim.	Lin. Acc. ($\uparrow$)		kNN Acc. ($\uparrow$)		Sil. ($\uparrow$)		DB ($\downarrow$)			
NSC-pSP (ours)	32	0.8488 ± 0.0406		0.7063 ± 0.0590		0.0460 ± 0.0257		4.1198 ± 0.9657			
NSC-SP (ours)	32	0.8263 ± 0.0542		0.7313 ± 0.0722		0.0432 ± 0.0209		4.1806 ± 0.8407			
AE	32	0.7688 ± 0.0652		0.6950 ± 0.0825		0.0340 ± 0.0102		4.4959 ± 0.5068			
UMAP	32	0.7663 ± 0.0651		0.7050 ± 0.0621		0.0415 ± 0.0321		4.2374 ± 0.7654			
NSC (ours)	32	0.7638 ± 0.0614		0.6538 ± 0.0932		0.0348 ± 0.0187		4.5613 ± 1.1626			
PCA	32	0.7588 ± 0.0553		0.6188 ± 0.0659		0.0296 ± 0.0071		4.7186 ± 0.4223			
PaCMAP	32	0.7575 ± 0.0657		0.7050 ± 0.0907		0.0410 ± 0.0375		4.8160 ± 0.6112			
NSC-P (ours)	32	0.7263 ± 0.1021		0.6225 ± 0.0671		0.0270 ± 0.0125		5.0121 ± 1.5534			
RP	32	0.7163 ± 0.0921		0.6113 ± 0.0781		0.0251 ± 0.0108		5.0218 ± 0.7847			

Overall results across repetitions (Mean $\pm$ Std.; higher is better for Acc./Sil., lower is better for DB).

Table D.2. Nonparametric comparisons on the block-structured synthetic HDLSS model ($M=32$, 10 reps). Friedman tests compare all nine methods; Wilcoxon signed-rank tests are one-sided with **NSC-pSP** as the reference (higher is better for accuracies/silhouette; lower is better for DB). Here, DB = Davies-Bouldin, sil. = silhouette.

Accuracy metrics			Clustering metrics		
Comparison	Stat.	p	Comparison	Stat.	p
Friedman (linear_acc; 9 methods)	31.879	9.791e-05	Friedman (sil.; 9 methods)	32.335	8.110e-05
NSC-pSP > AE (linear)	36.0	0.003906	NSC-pSP > AE (sil.)	53.0	0.002930
NSC-pSP > NSC (linear)	45.0	0.001953	NSC-pSP > NSC (sil.)	40.0	0.01953
NSC-pSP > NSC-P (linear)	44.0	0.003906	NSC-pSP > NSC-P (sil.)	55.0	0.0009766
NSC-pSP > NSC-SP (linear)	38.0	0.03711	NSC-pSP > NSC-SP (sil.)	26.0	0.5693
NSC-pSP > PCA (linear)	55.0	0.0009766	NSC-pSP > PCA (sil.)	55.0	0.0009766
NSC-pSP > RP (linear)	55.0	0.0009766	NSC-pSP > RP (sil.)	55.0	0.0009766
NSC-pSP > UMAP (linear)	52.0	0.004883	NSC-pSP > UMAP (sil.)	31.0	0.3848
NSC-pSP > PaCMAP (linear)	53.0	0.002930	NSC-pSP > PaCMAP (sil.)	27.0	0.5391
Friedman (knn_acc; 9 methods)	34.190	3.753e-05	Friedman (DB; 9 methods)	43.680	6.539e-07
NSC-pSP > AE (kNN)	32.0	0.3467	NSC-pSP < AE (DB)	2.0	0.002930
NSC-pSP > NSC (kNN)	37.0	0.04492	NSC-pSP < NSC (DB)	5.0	0.009766
NSC-pSP > NSC-P (kNN)	52.0	0.004883	NSC-pSP < NSC-P (DB)	0.0	0.0009766
NSC-pSP > NSC-SP (kNN)	11.0	0.9561	NSC-pSP < NSC-SP (DB)	31.0	0.6523
NSC-pSP > PCA (kNN)	45.0	0.001953	NSC-pSP < PCA (DB)	0.0	0.0009766
NSC-pSP > RP (kNN)	45.0	0.001953	NSC-pSP < RP (DB)	0.0	0.0009766
NSC-pSP > UMAP (kNN)	25.0	0.6152	NSC-pSP < UMAP (DB)	28.0	0.5391
NSC-pSP > PaCMAP (kNN)	26.0	0.5771	NSC-pSP < PaCMAP (DB)	6.0	0.01367

Table D.3. Additional competitiveness analyses of NSC-pSP vs. PCA/AE/UMAP/PaCMAP on the block-structured synthetic HDLSS model ($M=32$, 10 reps). Δ denotes paired improvement of NSC-pSP over the baseline (for Acc./Sil.: $\Delta = \text{NSC-pSP} - \text{base}$; for DB: $\Delta = \text{base} - \text{NSC-pSP}$ so that $\Delta > 0$ favors NSC-pSP). 95% CIs are paired bootstrap percentile intervals of the mean Δ (over reps). W/T/L counts wins/ties/losses across reps. Wilcoxon p is paired two-sided; r_{rb} is the matched-pairs rank-biserial effect size. Avg. ranks are computed over all 9 methods per rep (1=best).

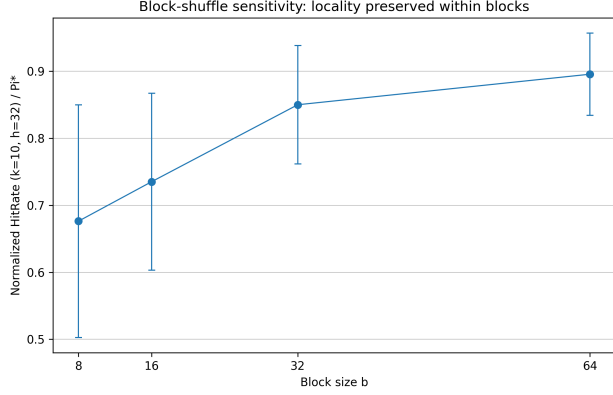
Metric	Base	Δ (mean)	95% CI	W/T/L	Wilcoxon p (2s)	r_{rb}	Avg. Rank (ours/base)
Lin. Acc.	PCA	0.0887	[0.0624, 0.1149]	10/0/0	0.001953	1.000	1.85/4.85
Lin. Acc.	AE	0.0900	[0.0525, 0.1263]	8/2/0	0.007812	1.000	1.85/5.10
Lin. Acc.	UMAP	0.0825	[0.0403, 0.1245]	8/0/2	0.009766	0.891	1.85/5.10
Lin. Acc.	PaCMAP	0.0913	[0.0471, 0.1346]	9/0/1	0.005859	0.927	1.85/5.40
kNN Acc.	PCA	0.0789	[0.0513, 0.1044]	9/1/0	0.003906	1.000	3.85/6.85
kNN Acc.	AE	0.0026	[-0.0337, 0.0376]	5/0/5	0.695312	0.164	3.85/3.60
kNN Acc.	UMAP	-0.0073	[-0.0534, 0.0384]	5/0/5	0.845703	-0.091	3.85/4.10
kNN Acc.	PaCMAP	-0.0073	[-0.0747, 0.0609]	5/0/5	0.921875	-0.055	3.85/4.00
Sil.	PCA	0.0162	[0.0091, 0.0244]	10/0/0	0.001953	1.000	2.95/6.40
Sil.	AE	0.0124	[0.0056, 0.0199]	9/0/1	0.005859	0.927	2.95/4.60
Sil.	UMAP	0.0045	[-0.0171, 0.0274]	5/0/5	0.769531	0.127	2.95/4.40
Sil.	PaCMAP	0.0050	[-0.0210, 0.0302]	4/0/6	1.000000	-0.018	2.95/4.40
DB	PCA	0.6834	[0.4172, 0.9733]	10/0/0	0.001953	1.000	2.90/6.40
DB	AE	0.5049	[0.2642, 0.7506]	9/0/1	0.005859	0.927	2.90/4.50
DB	UMAP	0.2476	[-0.3096, 0.8761]	3/0/7	1.000000	-0.018	2.90/3.20
DB	PaCMAP	0.8262	[0.3524, 1.3164]	7/0/3	0.027344	0.782	2.90/6.00

Table D.4. **GO-LR+NSC vs. PCA.** Accuracy comparison using the same frozen TabPFN-2.5 predictor, where only the front-end compression method differs. Values are mean accuracy with subscripted standard deviation over 5×5 CV.

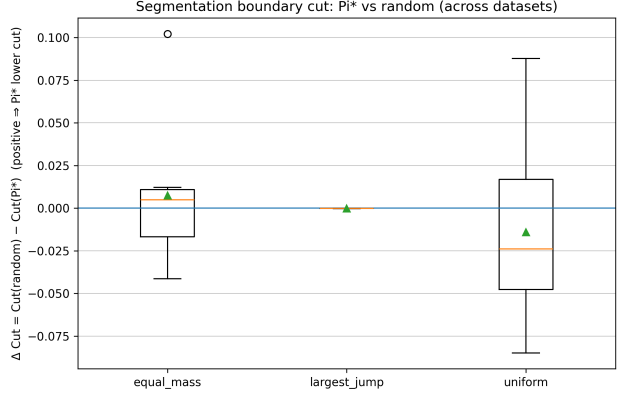
Model / DB	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX
GO-LR+NSC+TabPFN-2.5	88.18 ± 10.05	97.44 ± 2.32	93.82 ± 5.81	74.23 ± 5.17	97.54 ± 3.86	93.37 ± 4.48	90.60 ± 3.97	93.33 ± 4.74
Global PCA+TabPFN-2.5	78.51 ± 10.13	96.16 ± 2.85	86.35 ± 6.73	69.71 ± 7.70	95.52 ± 4.90	89.21 ± 5.23	81.90 ± 5.37	89.47 ± 6.45

Table D.5. **GO-LR+NSC vs. Lasso-selected features.** Accuracy comparison using the same frozen TabPFN-2.5 predictor. LT denotes Lasso-selected features + TabPFN-2.5, evaluated at different sparsity levels C . Values are mean accuracy with subscripted standard deviation over 5×5 CV.

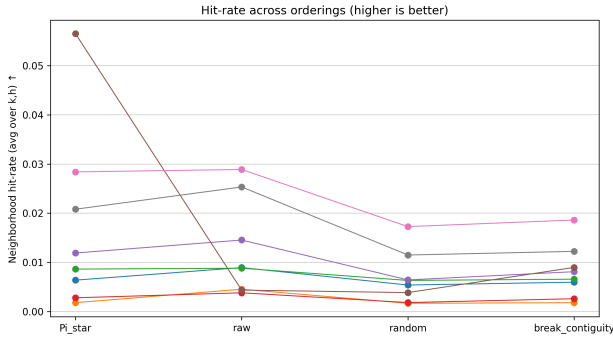
Method / DB	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX
GOTabPFN	88.18 ± 10.05	97.44 ± 2.32	93.82 ± 5.81	74.23 ± 5.17	97.54 ± 3.86	93.37 ± 4.48	90.60 ± 3.97	93.33 ± 4.74
LT ($C = 0.01$)	85.59 ± 9.80	96.76 ± 2.63	91.76 ± 5.88	68.53 ± 7.54	96.10 ± 4.52	93.33 ± 5.06	89.60 ± 3.73	90.06 ± 5.22
LT ($C = 0.02$)	85.59 ± 9.80	96.75 ± 2.44	91.76 ± 5.88	68.53 ± 7.54	96.10 ± 4.52	93.33 ± 5.06	89.60 ± 3.73	92.28 ± 5.79
LT ($C = 0.05$)	85.59 ± 10.05	97.15 ± 2.33	92.71 ± 5.44	75.14 ± 6.10	98.04 ± 3.21	92.35 ± 5.08	90.30 ± 4.23	92.74 ± 3.90



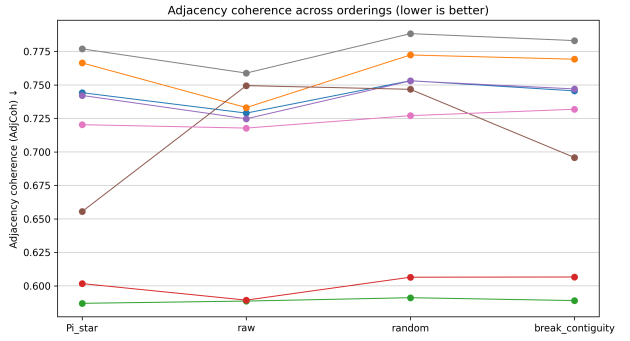
(a) **Block-shuffle sensitivity.** Normalized $\text{HitRate}_{k,h}$ (relative to Π^*) vs. block size b .



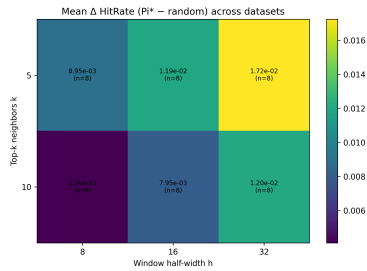
(b) **Boundary-cut advantage.** $\Delta \text{Cut} = \text{Cut}(\text{random}) - \text{Cut}(\Pi^*)$ across datasets (positive $\Rightarrow \Pi^*$ lower cut).



(c) **Hit-rate across order families.** $\text{HitRate}_{k,h}$ is highest under Π^* and drops when contiguity is broken.



(d) **Adjacency coherence across families.** AdjCoh is lowest (best) under Π^* and worse under random.



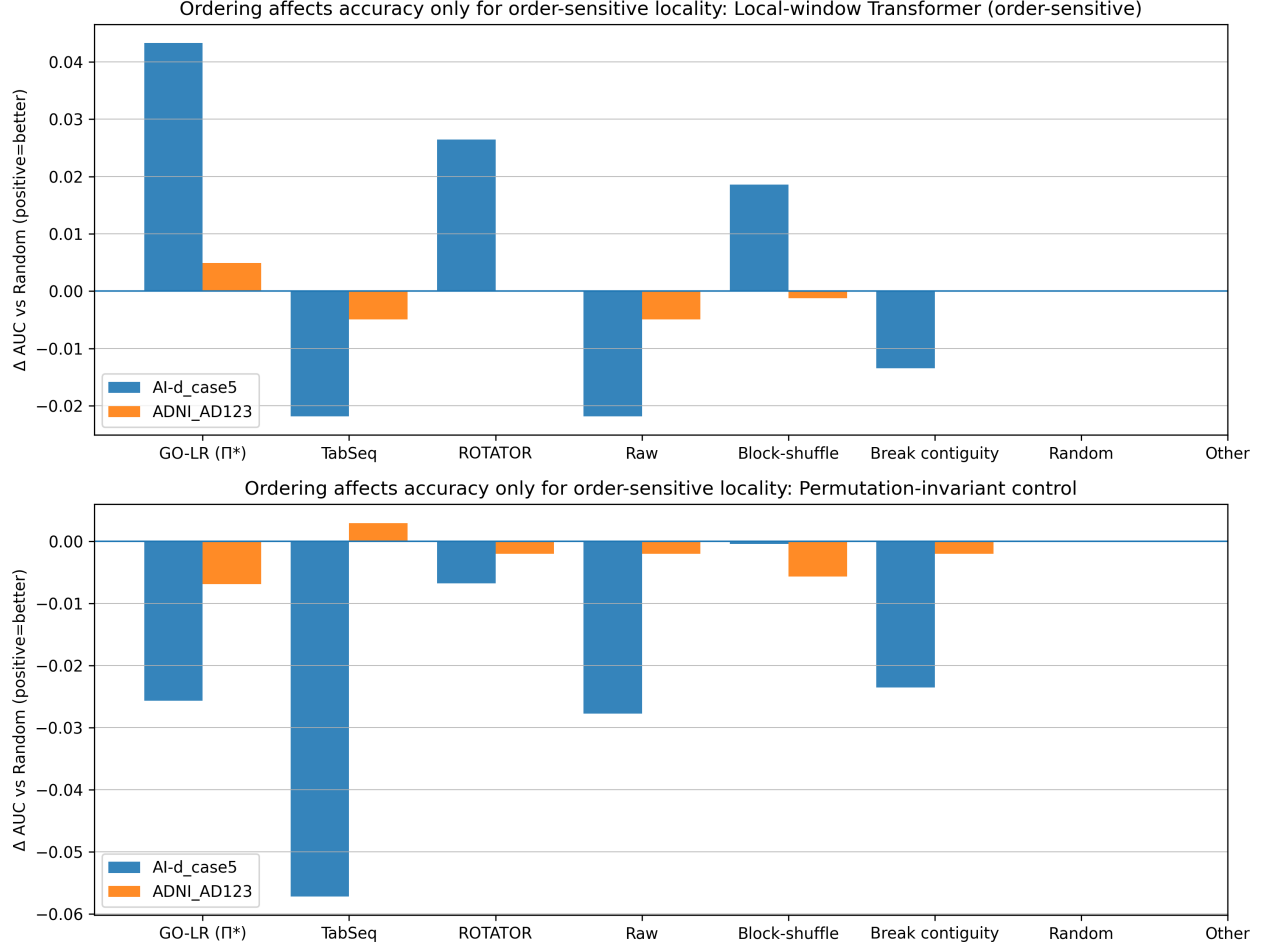
(e) **Robust neighborhood gains.** $\text{Mean } \Delta \text{HitRate}_{k,h} = \text{HitRate}(\Pi^*) - \text{HitRate}(\text{random})$ over (k, h) .

Figure E.1. Order-only neighborhood diagnostics and controls. GO-LR ordering Π^* improves local coherence (AdjCoh), increases neighborhood recovery (HitRate), yields favorable segmentation alignment (Cut), and degrades predictably under block-shuffle and contiguity-breaking controls.

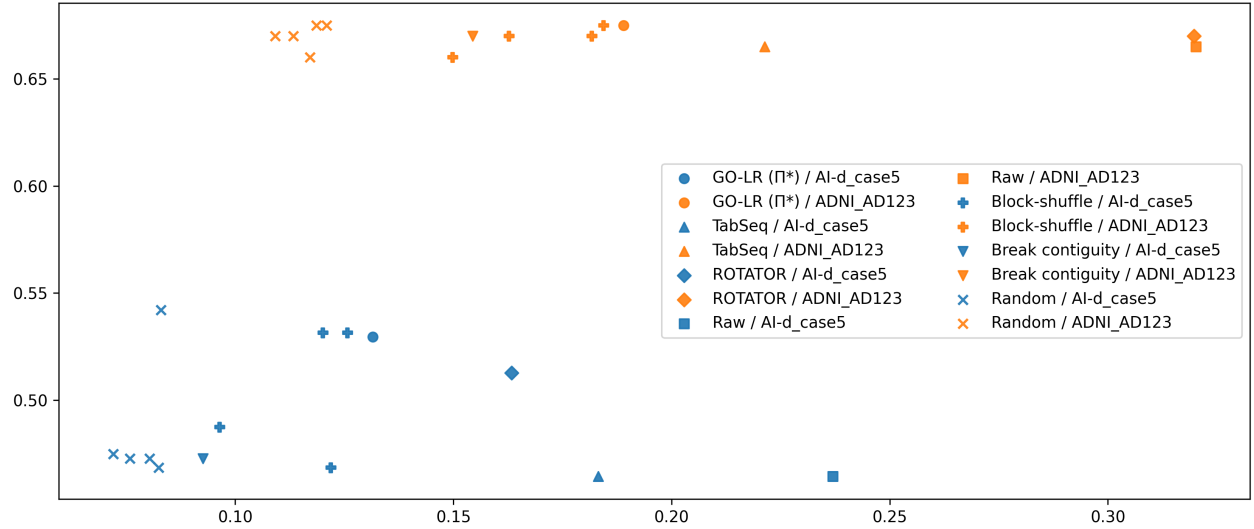
Test / Statistic	Value
Wilcoxon p ($\text{AdjCoh}: \Pi^* < \text{random}$)	0.00390625
Wilcoxon p ($\text{HitRate}: \Pi^* > \text{random}$)	0.0078125

Block size b	Norm. HitRate (mean $\pm$ std)
8	0.740448 ± 0.184317
16	0.888879 ± 0.176857
32	0.967536 ± 0.096630
64	1.010504 ± 0.094868

(f) **Aggregated stats.** Wilcoxon tests ($n=8$ datasets) and block-shuffle summary (normalized to Π^*).



(a) **Causal check.** Δ AUC vs. random permutations for an order-sensitive local-window Transformer (top) and a permutation-invariant control (bottom). Ordering changes performance only when the backbone uses locality.



(b) **Mechanism.** Neighborhood preservation ($\text{HitRate}_{k,h}$) correlates with downstream AUC for the local model, supporting the locality hypothesis.

Figure E.2. Ordering can improve accuracy beyond NSC. Learned orderings matter for architectures that introduce locality over feature tokens; invariant controls do not exhibit systematic gains.

Table E.1. Ordering improves AUC for an order-sensitive local-window Transformer, but not for a permutation-invariant control, on two $n < m$ datasets. Values are mean $\pm$ std where multiple runs exist (e.g., random permutations); parentheses show Δ AUC relative to the random baseline within each dataset (local model).

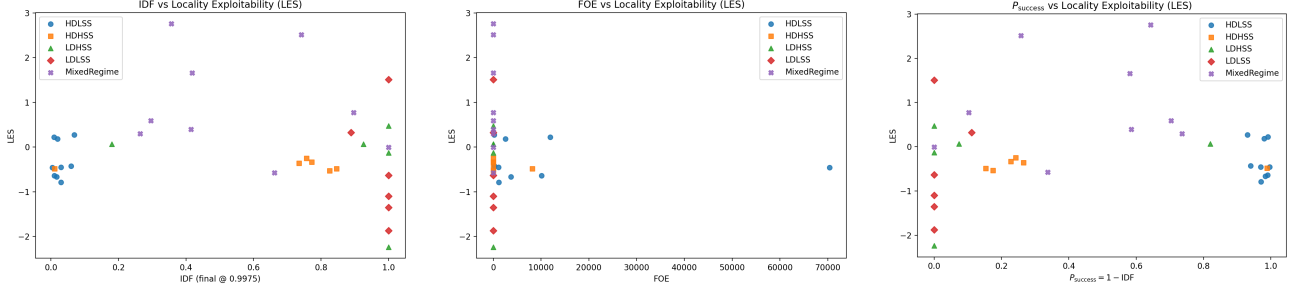
Ordering	Local window Transformer (AUC)		Permutation-invariant control (AUC)	
	AI-d_case5	ADNI_AD123	AI-d_case5	ADNI_AD123
GO-LR (Π^*)	0.529 (+0.043)	0.675 (+0.005)	0.466	0.665
TabSeq	0.464 (-0.022)	0.665 (-0.005)	0.435	0.675
ROTATOR	0.513 (+0.026)	0.670 (+0.000)	0.485	0.670
Raw	0.464 (-0.022)	0.665 (-0.005)	0.464	0.670
Block-shuffle ($b = 8$)	0.487 (+0.001)	0.660 (-0.010)	0.508	0.660
Block-shuffle ($b = 16$)	0.532 (+0.045)	0.670 (+0.000)	0.513	0.670
Block-shuffle ($b = 32$)	0.468 (-0.018)	0.675 (+0.005)	0.471	0.665
Block-shuffle ($b = 64$)	0.532 (+0.045)	0.670 (+0.000)	0.475	0.670
Break contiguity	0.473 (-0.013)	0.670 (+0.000)	0.468	0.670
Random (baseline)	0.486 $\pm$ 0.031	0.670 $\pm$ 0.006	0.492 $\pm$ 0.032	0.672 $\pm$ 0.003

Table F.1. When to use ordering through locality: FOE-sorted datasets with IDF/FOE/ P_{success} and order-only locality gains/LES (not used for sorting). Here, AUC denotes the area under the cumulative explained-variance-IDF curve, computed via trapezoidal integration over discrete pairs ($IDF_k = k/n_{\text{total}}$, $CV\text{ar}(k)$). Here, HDLSS = High-Dimensional Low-Sample Size, HDHSS = High-Dimensional High-Sample Size, LDLSS = Low-Dimensional Low-Sample Size, LDHSS = Low-Dimensional High-Sample Size.

Rank	Dataset	Cat.	IDF $\downarrow$	FOE $\uparrow$	$P_{\text{success}} \uparrow$	$\Delta\text{AdjCoh}\uparrow$	$\Delta\text{HitRate}\uparrow$	$\Delta\text{Cut}\uparrow$	LES $\uparrow$	AUC
1	GLI-85	HDLSS	3.770e-03	7.037e+04	0.996	5.942e-03	1.465e-04	0.0105	-0.457	1.027e-03
2	SMK_CAN_187	HDLSS	9.153e-03	1.194e+04	0.991	4.770e-03	6.445e-04	0.102	0.222	4.629e-03
3	ALLAML	HDLSS	9.959e-03	1.008e+04	0.99	8.937e-03	1.387e-03	-0.0169	-0.639	2.681e-03
4	DeepLesion+	HDHSS	0.011	8.19e+03	0.989	7.085e-03	9.307e-03	0.000	-0.481	5.144e-03
5	Prostate-GE	HDLSS	0.0164	3.71e+03	0.984	4.189e-03	5.176e-04	-0.0168	-0.667	8.838e-03
6	Arcene	HDLSS	0.0197	2.58e+03	0.98	0.0912	0.0368	0.0122	0.185	6.477e-03
7	TOX171	HDLSS	0.0292	1.17e+03	0.971	0.0109	5.498e-03	-0.0413	-0.789	0.0116
8	Colon	HDLSS	0.03	1.11e+03	0.97	6.735e-03	9.160e-03	4.156e-03	-0.452	0.0124
9	Lung	HDLSS	0.0598	280	0.94	0.0114	7.168e-03	5.703e-03	-0.428	0.0251
10	DriveFace	HDLSS	0.0692	209	0.931	0.112	0.058	-5.279e-03	0.274	0.0597
11	MiniBooNE	LDHSS	0.18	30.9	0.82	0.0413	-0.0208	0.0712	0.0648	0.136
12	EEG-FE	MixedRegime	0.264	14.4	0.736	0.0827	0.0933	-7.165e-03	0.296	0.231
13	MOF	MixedRegime	0.296	11.4	0.704	0.0492	0.0653	0.0755	0.593	0.137
14	EEG-PD	MixedRegime	0.357	7.85	0.643	0.202	0.244	0.134	2.76	0.301
15	AI-D (Case 5)	MixedRegime	0.415	5.81	0.585	0.0293	0.068	0.0606	0.394	0.348
16	ADNI (AD123)	MixedRegime	0.418	5.72	0.582	0.11	0.158	0.111	1.66	0.28
17	CNAE9	MixedRegime	0.662	2.28	0.338	6.173e-04	-2.290e-03	9.856e-07	-0.575	0.269
18	MINIST+	HDHSS	0.735	1.85	0.265	5.163e-03	6.973e-03	0.0192	-0.36	0.635
19	WDBC	MixedRegime	0.742	1.82	0.258	0.197	nan	nan	2.52	0.47
20	HAM10000	HDHSS	0.757	1.74	0.243	9.772e-03	0.0114	0.0278	-0.249	0.646
21	Fashion MNIST+	HDHSS	0.772	1.68	0.228	3.848e-03	6.934e-03	0.0238	-0.333	0.663
22	Dog vs Cat+	HDHSS	0.825	1.47	0.175	0.0164	0.0131	-0.0158	-0.531	0.702
23	CIFAR-10+	HDHSS	0.846	1.4	0.154	0.012	0.0162	-9.114e-03	-0.487	0.71
24	Glass	LDLSS	0.889	1.27	0.111	0.0542	nan	nan	0.326	0.219
25	Cargo	MixedRegime	0.897	1.24	0.103	0.0956	0.156	3.076e-03	0.773	0.405
26	Forest Cover	LDHSS	0.926	1.17	0.0741	0.0309	4.444e-03	0.0605	0.065	0.197
27	Iris	LDLSS	1	1	0.000	-0.0893	nan	nan	-1.87	0.493
28	Higgs	LDHSS	1	1	0.000	0.064	nan	nan	0.477	0.275
29	BUPA Liver	LDLSS	1	1	0.000	-8.495e-03	nan	nan	-0.634	0.163
30	Adult	LDHSS	1	1	0.000	-0.113	nan	nan	-2.24	0.138
31	Pima Indian	LDLSS	1	1	0.000	-0.0553	nan	nan	-1.35	0.122
32	Water Potability	MixedRegime	1	1	0.000	0.0325	nan	nan	-5.418e-03	0.106
33	Poker Hand	LDHSS	1	1	0.000	0.0243	nan	nan	-0.131	0.0955
34	Hayes-Roth	LDLSS	1	0.000	0.000	0.131	nan	nan	1.51	0.000
35	Monks-1	LDLSS	1	0.000	0.000	-0.0388	nan	nan	-1.1	0.000

Table F.2. Ordering-locality diagnostics for additional cross-domain datasets. Datasets are sorted by FOE score. Categories follow the empirical regime rule using n , m , and $\rho = m/n$. AUC is computed under the cumulative explained variance-IDF curve via trapezoidal integration. LES is standardized within this five-dataset subset.

Rank	Dataset	Cat.	IDF $\downarrow$	FOE $\uparrow$	$P_{\text{success}} \uparrow$	$\Delta\text{AdjCoh}\uparrow$	$\Delta\text{HitRate}\uparrow$	$\Delta\text{Cut}\uparrow$	LES $\uparrow$	AUC
1	orlraws10P	HDLSS	9.317e-03	1.152e+04	0.991	3.487e-03	1.641e-03	-0.0223	-0.170	5.118e-03
2	Cell Cycle	MixedRegime	0.0244	1.681e+03	0.976	8.670e-03	1.758e-04	7.770e-04	0.659	4.773e-03
3	RELATHE	MixedRegime	0.292	11.77	0.709	-1.728e-03	-5.078e-04	-4.570e-03	-0.664	0.148
4	BASEHOCK	MixedRegime	0.362	7.649	0.638	2.262e-04	1.270e-04	5.039e-03	0.0345	0.185
5	PCMAC	MixedRegime	0.513	3.796	0.487	-3.030e-04	2.520e-03	-0.0111	0.141	0.261



(a) IDF vs. LES. (b) FOE vs. LES. (c) P_{success} vs. LES.
 Figure F.1. Locality Exploitability Score (LES) against intrinsic-dimension/compression proxies.

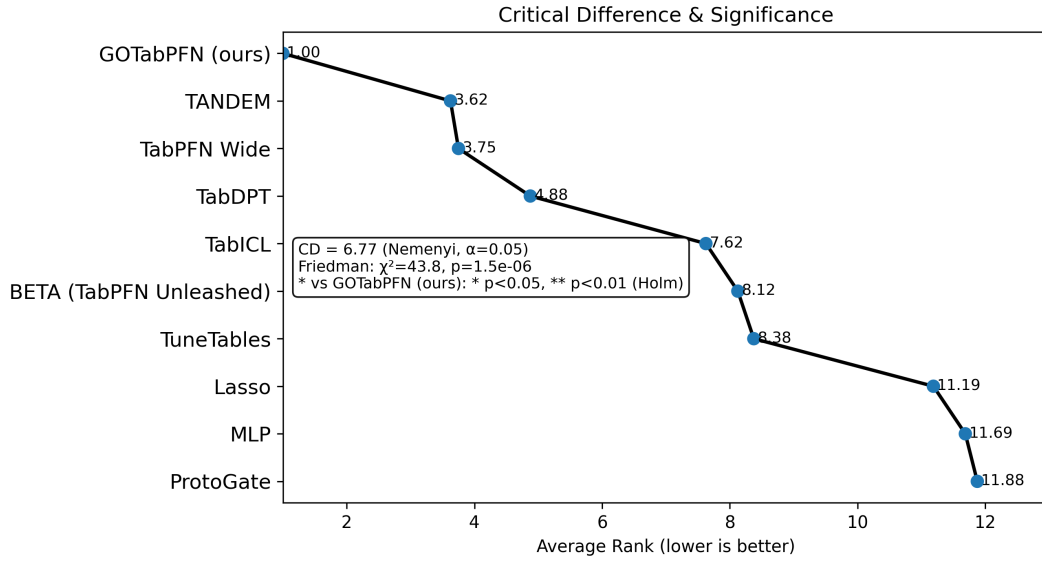


Figure I.1. **Average-rank comparison on the 8 HDLSS datasets.** Lower rank is better. Friedman/Nemenyi analysis shows GOTabPFN as the best-ranked method on the original HDLSS benchmark.

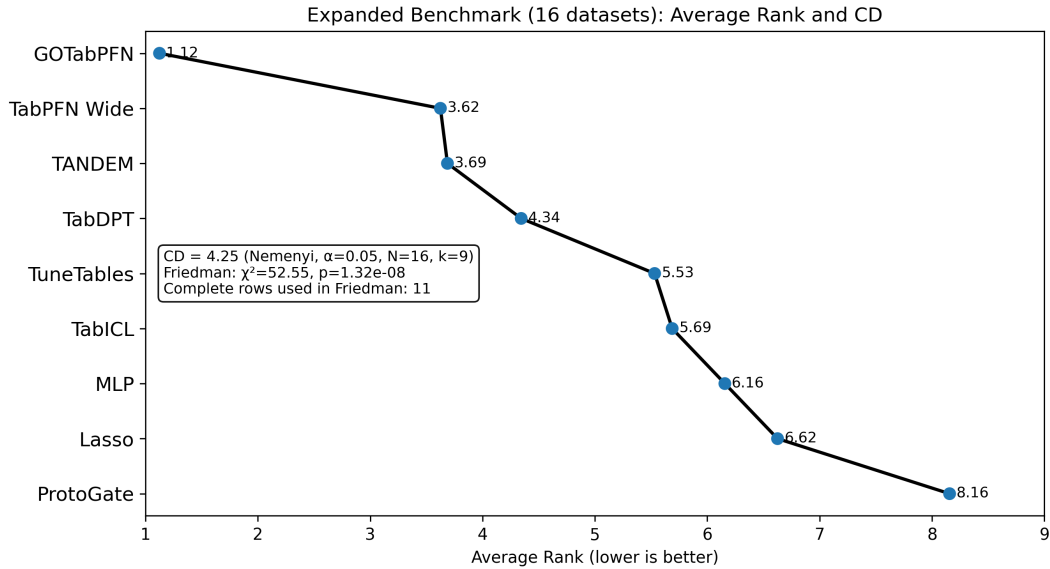


Figure I.2. **Average-rank comparison on the expanded 16-dataset benchmark.** Lower rank is better. GOTabPFN achieves the best average rank (1.12), followed by TabPFN-Wide (3.62) and TANDEM (3.69), with a significant Friedman test ($\chi^2 = 52.55$, $p = 1.32 \times 10^{-8}$).

Table G.1. Performance of the models on 8 HDLSS datasets (mean accuracy with subscripted standard deviation over 5×5 CV). Dataset abbreviations: COL = Colon, LNG = Lung, GLI = GLI-85, SMK = SMK_CAN_187, AML = ALLAML, PRS = Prostate-GE, ARC = Arcene, TOX = TOX-171. Model abbreviations: GOTabPFN_{ours} = our method, TWide = TabPFN Wide, TTables = TuneTables, BETA = TabPFN Unleashed, PGate = ProtoGate, TRNN = TabulaRNN, RF = Random Forest, NB = Naive Bayes, DT = Decision Tree, MambAtt = MambAttention, FT-T = FT-Transformer, CatEmbed = CategoryEmbedding, ResNetT = ResNetTabular, Tab-T = TabTransformer.

MODEL	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX	AVG. RANK
GOTABPFN _{OURS}	88.18 _{±10.05}	97.44 _{±2.32}	93.82 _{±5.81}	74.23 _{±5.17}	97.54 _{±3.86}	93.37 _{±4.48}	90.60 _{±3.97}	93.33 _{±4.74}	1.00 _{±0.00}
TANDEM	86.15 _{±7.75}	96.46 _{±2.88}	91.53 _{±6.02}	72.72 _{±5.69}	95.81 _{±5.53}	91.55 _{±4.32}	86.90 _{±6.34}	93.08 _{±2.61}	3.63 _{±1.32}
TWIDE	87.85 _{±7.28}	96.55 _{±2.15}	88.47 _{±5.75}	68.78 _{±8.60}	97.16 _{±4.10}	93.10 _{±5.92}	88.00 _{±5.20}	89.35 _{±4.95}	3.75 _{±2.38}
TABDPT	86.26 _{±7.27}	96.05 _{±2.57}	87.76 _{±5.86}	71.99 _{±7.32}	96.32 _{±4.15}	90.94 _{±5.72}	82.10 _{±6.48}	93.25 _{±3.44}	4.88 _{±1.69}
TABICL	84.62 _{±10.52}	96.36 _{±2.61}	87.06 _{±6.23}	68.73 _{±7.28}	95.52 _{±5.59}	90.17 _{±5.93}	82.60 _{±6.14}	88.78 _{±5.92}	7.63 _{±2.29}
BETA	84.73 _{±9.36}	94.38 _{±3.34}	86.21 _{±8.91}	70.21 _{±5.61}	95.67 _{±7.54}	87.53 _{±4.67}	86.45 _{±5.92}	90.38 _{±6.42}	8.13 _{±4.31}
TTABLES	86.80 _{±2.14}	94.37 _{±2.35}	89.66 _{±3.12}	70.28 _{±6.46}	95.80 _{±2.14}	93.31 _{±2.83}	81.40 _{±3.66}	77.96 _{±2.55}	8.38 _{±7.70}
LASSO	79.40 _{±10.18}	94.47 _{±4.39}	85.88 _{±4.71}	61.19 _{±13.72}	87.24 _{±3.39}	91.18 _{±6.39}	81.00 _{±3.39}	91.86 _{±6.03}	11.13 _{±5.06}
MLP	83.95 _{±9.80}	96.47 _{±2.69}	85.41 _{±8.00}	59.05 _{±7.44}	89.98 _{±9.17}	89.20 _{±6.07}	78.40 _{±4.05}	92.48 _{±4.28}	11.63 _{±5.45}
PGATE	83.95 _{±9.82}	93.44 _{±6.37}	82.48 _{±5.68}	60.16 _{±5.10}	86.12 _{±3.34}	90.58 _{±5.72}	81.50 _{±5.10}	92.34 _{±5.67}	12.06 _{±4.77}
TRNN	84.20 _{±6.50}	90.50 _{±4.80}	79.68 _{±6.68}	60.02 _{±3.18}	88.92 _{±2.02}	90.50 _{±6.00}	81.50 _{±5.10}	85.80 _{±4.70}	14.81 _{±6.03}
REALMLP	76.28 _{±13.16}	93.39 _{±3.81}	82.59 _{±13.31}	64.93 _{±9.86}	96.65 _{±5.78}	85.17 _{±11.94}	79.70 _{±6.01}	88.67 _{±5.32}	15.13 _{±7.36}
LGBM	76.60 _{±11.67}	93.42 _{±5.91}	85.88 _{±11.53}	58.85 _{±10.14}	85.81 _{±5.67}	91.38 _{±5.71}	80.50 _{±5.79}	81.98 _{±6.25}	16.00 _{±6.08}
CATBOOST	72.65 _{±10.12}	91.57 _{±5.74}	84.71 _{±12.11}	58.28 _{±12.16}	91.71 _{±8.22}	90.24 _{±6.87}	81.00 _{±2.00}	81.95 _{±7.47}	17.38 _{±6.38}
STG	79.55 _{±10.53}	93.30 _{±6.28}	82.48 _{±4.56}	57.25 _{±8.82}	86.08 _{±5.60}	89.38 _{±5.85}	74.40 _{±6.90}	87.95 _{±5.01}	18.25 _{±4.92}
RF	80.05 _{±10.37}	91.73 _{±6.61}	85.88 _{±8.80}	58.29 _{±10.61}	85.71 _{±5.71}	90.38 _{±7.31}	74.00 _{±1.22}	79.78 _{±7.10}	18.50 _{±5.57}
REAL-X	76.75 _{±12.21}	93.27 _{±4.32}	83.24 _{±5.56}	56.48 _{±4.90}	84.16 _{±5.68}	86.75 _{±6.68}	77.30 _{±6.10}	90.79 _{±4.75}	19.50 _{±6.18}
LLSPIN	79.35 _{±7.74}	70.10 _{±12.31}	84.42 _{±7.12}	61.16 _{±7.92}	88.12 _{±1.26}	88.71 _{±5.98}	80.80 _{±4.90}	81.67 _{±9.01}	19.50 _{±8.50}
LSPIN	81.30 _{±7.97}	76.92 _{±9.38}	83.48 _{±6.62}	58.92 _{±6.78}	84.46 _{±3.36}	87.75 _{±6.74}	78.60 _{±5.80}	83.47 _{±8.59}	19.88 _{±5.16}
TABR	80.75 _{±8.40}	86.70 _{±6.40}	81.42 _{±6.64}	58.46 _{±6.68}	80.84 _{±2.24}	84.50 _{±8.00}	75.85 _{±6.50}	86.50 _{±5.00}	21.56 _{±4.15}
KNN	71.65 _{±12.03}	91.06 _{±5.41}	83.53 _{±5.76}	52.05 _{±13.13}	79.05 _{±8.02}	78.78 _{±9.20}	82.50 _{±5.00}	83.86 _{±7.07}	22.75 _{±9.36}
MLP-PLR	71.41 _{±11.32}	77.32 _{±12.81}	80.94 _{±9.50}	60.79 _{±8.28}	92.84 _{±7.68}	86.89 _{±8.69}	66.30 _{±12.52}	81.85 _{±8.32}	22.88 _{±8.33}
ADABOOST	78.97 _{±10.96}	78.32 _{±1.93}	85.88 _{±7.97}	58.28 _{±9.55}	84.38 _{±8.32}	89.19 _{±4.94}	75.50 _{±5.34}	57.85 _{±9.01}	23.38 _{±8.62}
XGBOOST	72.60 _{±12.59}	86.61 _{±8.72}	77.06 _{±7.80}	55.59 _{±6.34}	90.38 _{±9.39}	82.55 _{±10.22}	81.50 _{±4.06}	70.13 _{±7.85}	24.38 _{±8.51}
GBM	77.31 _{±14.43}	91.59 _{±2.52}	80.00 _{±8.80}	58.31 _{±9.72}	82.10 _{±6.67}	82.24 _{±5.34}	82.00 _{±3.32}	53.85 _{±15.31}	24.50 _{±9.80}
SVM	70.75 _{±13.93}	72.77 _{±8.33}	85.88 _{±2.88}	61.09 _{±11.78}	83.14 _{±13.37}	85.75 _{±6.63}	77.00 _{±2.92}	66.75 _{±7.86}	24.50 _{±10.49}
INVASE	75.40 _{±10.10}	91.22 _{±6.16}	80.14 _{±4.56}	36.42 _{±4.46}	78.90 _{±2.26}	88.00 _{±6.50}	71.20 _{±7.80}	79.94 _{±6.60}	26.63 _{±7.76}
NB	83.85 _{±10.56}	84.24 _{±3.99}	82.35 _{±3.72}	59.32 _{±14.95}	88.57 _{±2.86}	60.86 _{±14.63}	53.50 _{±8.31}	58.49 _{±8.14}	27.13 _{±12.07}
DT	83.85 _{±9.15}	85.16 _{±5.58}	78.82 _{±9.56}	56.63 _{±6.94}	75.14 _{±10.18}	82.33 _{±5.13}	73.50 _{±5.39}	45.68 _{±8.70}	28.63 _{±9.01}
MAMBATB	80.75 _{±8.40}	87.85 _{±6.00}	80.16 _{±6.64}	54.98 _{±9.20}	68.16 _{±4.90}	58.52 _{±15.60}	64.00 _{±9.60}	82.30 _{±6.00}	28.81 _{±9.28}
TABSEQ	72.00 _{±11.24}	86.81 _{±3.98}	75.29 _{±9.98}	65.16 _{±7.50}	77.28 _{±14.49}	65.24 _{±10.76}	65.30 _{±6.45}	47.95 _{±7.27}	30.50 _{±10.64}
MAMBULAR	83.55 _{±7.10}	89.90 _{±5.10}	46.92 _{±4.56}	32.90 _{±15.12}	60.80 _{±5.77}	81.12 _{±8.02}	69.65 _{±8.20}	84.95 _{±5.20}	30.50 _{±12.46}
MAMBATT	81.90 _{±8.00}	78.00 _{±9.10}	76.18 _{±7.78}	52.18 _{±4.46}	70.16 _{±6.28}	61.99 _{±14.45}	49.90 _{±12.90}	80.90 _{±6.40}	31.75 _{±8.82}
FT-T	69.25 _{±12.50}	67.30 _{±12.20}	52.46 _{±8.92}	56.45 _{±9.68}	56.12 _{±12.10}	85.00 _{±7.00}	52.30 _{±12.30}	79.45 _{±6.90}	35.63 _{±7.45}
SAINT	67.60 _{±13.10}	78.00 _{±9.10}	78.56 _{±9.36}	50.34 _{±12.16}	52.92 _{±14.15}	61.99 _{±14.45}	57.10 _{±11.20}	75.10 _{±8.20}	35.63 _{±5.75}
TABM	65.80 _{±13.70}	74.70 _{±10.10}	60.46 _{±4.42}	42.90 _{±6.36}	66.67 _{±3.34}	75.03 _{±10.08}	66.10 _{±9.10}	70.20 _{±9.60}	35.94 _{±3.57}
AUTOINT	65.80 _{±13.70}	74.70 _{±10.10}	48.44 _{±5.65}	49.68 _{±9.80}	58.34 _{±9.78}	83.47 _{±7.53}	67.90 _{±8.70}	66.80 _{±10.60}	36.00 _{±5.27}
CATEMBED	60.50 _{±15.50}	72.95 _{±10.60}	69.14 _{±9.46}	59.16 _{±6.68}	78.12 _{±6.90}	45.01 _{±19.50}	54.80 _{±11.80}	57.80 _{±13.30}	36.38 _{±9.50}
RESNETT	64.10 _{±14.30}	74.70 _{±10.10}	66.67 _{±12.16}	52.16 _{±4.46}	75.10 _{±3.30}	48.43 _{±18.70}	42.60 _{±14.70}	63.30 _{±11.70}	38.88 _{±5.67}
1D CNN	58.70 _{±16.10}	63.50 _{±13.30}	56.92 _{±5.62}	40.92 _{±10.42}	60.42 _{±9.92}	70.00 _{±12.00}	54.80 _{±11.80}	71.85 _{±9.10}	39.19 _{±4.35}
NODE	54.95 _{±17.40}	55.10 _{±15.70}	64.92 _{±12.34}	46.58 _{±10.45}	76.92 _{±10.08}	58.52 _{±15.60}	59.40 _{±10.70}	65.10 _{±11.10}	39.94 _{±5.36}
TABNET	56.75 _{±15.20}	80.14 _{±12.23}	55.29 _{±10.26}	48.67 _{±2.17}	63.89 _{±4.17}	66.55 _{±15.33}	50.00 _{±7.55}	41.68 _{±9.03}	40.38 _{±6.02}
L2X	57.60 _{±13.48}	50.02 _{±14.26}	56.92 _{±3.58}	45.68 _{±9.78}	76.28 _{±2.36}	61.78 _{±13.69}	72.95 _{±7.30}	31.72 _{±9.11}	40.75 _{±7.50}
TROMPT	58.70 _{±16.10}	61.55 _{±13.90}	43.96 _{±12.16}	46.65 _{±12.12}	46.12 _{±4.46}	69.08 _{±12.20}	40.10 _{±15.30}	66.80 _{±10.60}	42.38 _{±4.90}
DCN	48.30 _{±19.20}	57.25 _{±15.10}	42.86 _{±8.84}	34.92 _{±14.10}	43.78 _{±16.10}	85.02 _{±7.05}	61.75 _{±10.10}	51.90 _{±15.00}	43.31 _{±8.41}
DANETS	50.60 _{±18.60}	76.35 _{±9.60}	35.90 _{±6.78}	29.16 _{±16.10}	56.98 _{±4.48}	78.58 _{±9.10}	30.30 _{±17.80}	59.60 _{±12.80}	43.63 _{±7.03}
TANGOS	38.20 _{±21.70}	48.10 _{±17.50}	64.84 _{±10.42}	32.68 _{±6.80}	50.12 _{±8.86}	65.47 _{±13.30}	27.90 _{±18.40}	68.50 _{±10.10}	44.94 _{±7.20}
DEEPPM	56.90 _{±16.80}	59.40 _{±14.50}	43.28 _{±7.82}	36.12 _{±12.26}	48.16 _{±12.56}	45.01 _{±19.50}	64.00 _{±9.60}	55.90 _{±13.90}	45.00 _{±4.70}
ENODE	52.80 _{±18.00}	50.45 _{±16.90}	58.92 _{±13.12}	26.12 _{±5.68}	71.16 _{±4.46}	52.05 _{±17.95}	35.20 _{±16.50}	45.40 _{±16.80}	45.75 _{±5.33}
MODERNNCA	40.85 _{±21.10}	43.10 _{±18.80}	27.68 _{±14.10}	31.16 _{±11.67}	33.47 _{±10.24}	71.95 _{±11.12}	32.80 _{±17.10}	68.50 _{±10.10}	46.69 _{±7.53}
TABPFN-2.5	86.85 _{±9.16}	N/A	N/A	N/A	N/A	N/A	N/A	N/A	46.75 _{±16.54}
TAB-T	46.44 _{±16.84}	21.01 _{±12.53}	47.77 _{±17.71}	50.26 _{±8.56}	53.70 _{±15.05}	51.01 _{±10.64}	48.20 _{±7.57}	23.66 _{±7.84}	46.88 _{±5.04}
NDTF	48.30 _{±19.20}	63.50 _{±13.30}	26.14 _{±16.10}	36.18 _{±4.48}	43.14 _{±6.94}	54.98 _{±16.80}	37.70 _{±15.90}	43.10 _{±17.30}	48.00 _{±2.93}
TABPFN v2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	53.13 _{±0.33}
LOCALPFN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	53.13 _{±0.33}
TABPFN v1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	53.13 _{±0.33}

Table G.2. Evaluation beyond accuracy on 8 HDLSS datasets. ROC-AUC and macro-F1 comparisons under 5×5 CV. Values are mean with subscripted standard deviation. Bold denotes the best result per dataset/metric.

Metric	Model	COL	LNG	AML	TOX	PRS	ARC	SMK	GLI
ROC-AUC	GOTabPFN	91.40 ± 10.12	99.57 ± 0.60	99.36 ± 1.70	99.23 ± 0.63	94.27 ± 5.87	95.77 ± 2.35	80.55 ± 5.67	90.97 ± 7.09
	TabICL	89.37 ± 10.24	99.39 ± 0.46	98.27 ± 4.05	98.62 ± 1.15	93.92 ± 5.56	91.48 ± 3.84	73.27 ± 8.14	91.41 ± 7.25
	TabDPT	90.80 ± 7.28	99.27 ± 0.97	99.28 ± 0.62	99.93 ± 0.14	94.07 ± 3.50	91.33 ± 4.51	76.34 ± 7.90	93.33 ± 7.76
Macro-F1	GOTabPFN	87.14 ± 10.93	95.73 ± 5.37	96.17 ± 5.97	92.91 ± 4.01	90.32 ± 7.63	88.92 ± 4.92	75.46 ± 5.12	93.54 ± 4.84
	TabICL	78.84 ± 14.16	93.72 ± 7.09	92.90 ± 9.29	88.89 ± 5.91	90.13 ± 5.92	79.58 ± 6.53	71.04 ± 6.85	90.91 ± 4.62
	TabDPT	84.17 ± 10.91	94.71 ± 4.91	95.45 ± 4.89	93.62 ± 2.37	90.27 ± 6.42	82.08 ± 6.64	70.07 ± 7.99	85.25 ± 9.20

Table H.1. Best GOTabPFN hyperparameters for 8 HDLSS datasets (COL = Colon, LNG = Lung, GLI = GLI-85, SMK = SMK.CAN_187, AML = ALLAML, PRS = Prostate-GE, ARC = Arcene, TOX = TOX-171). GO-LR metric: eucl.=euclidean, manh.=manhattan, corr.=correlation, KL=kl.divergence. NSC seg: unif.=uniform, eq-mass=equal_mass, lrg-jump=largest_jump. NSC rule: gam.=gamma, def.=default. Std? indicates assume_standardized.

PARAM	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX
GO-LR METRIC	EUCL.	MANH.	CORR.	CORR.	KL	MANH.	EUCL.	KL
GO-LR k (CLUSTERS)	10	11	7	9	5	12	4	10
GO-LR REFINE PASSES	3	1	3	1	2	2	1	3
GO-LR DIR SELECT	T	T	F	F	T	T	F	T
GO-LR FEAT-SUB	—	—	—	3000	—	—	—	—
NSC SEG	EQ-MASS	UNIF.	UNIF.	UNIF.	LRG-JUMP	LRG-JUMP	UNIF.	EQ-MASS
NSC RULE	IDF	DEF.	GAM.	DEF.	GAM.	GAM.	GAM.	GAM.
NSC τ	0.99	0.99	0.99	0.99	0.95	0.99	0.99	0.99
NSC γ	1.76	2.38	2.92	2.90	2.40	2.91	2.26	2.86
NSC β	0.22	0.79	0.17	0.08	0.32	0.84	0.19	0.65
NSC $M_{\min}$	64	64	64	48	64	48	16	32
NSC $M_{\max}$	384	384	640	384	512	256	640	384
NSC $l_{\min}$	16	12	12	16	16	8	16	16
STD? (ASSUME STD.)	F	F	F	T	F	F	T	F
TABPFN SEED	42	42	42	2	3	3	4	4

Table H.2. Best GOTabPFN hyperparameters for 8 cross-domain datasets (ORL = orlraws10P, BAS = BASEHOCK, REL = RELATHE, PCM = PCMAC, CCY = Cell Cycle, CIF = CIFAR-10, DF-R = DrivFace-Regression, DF-C = DrivFace-Classification). GO-LR metric: cos.=cosine, manh.=manhattan, corr.=correlation, KL=kl.divergence. NSC seg: unif.=uniform, lrg-jump=largest_jump. NSC rule: def.=default. Std? indicates assume_standardized.

PARAM	ORL	BAS	REL	PCM	CCY	CIF	DF-R	DF-C
GO-LR METRIC	COS.	KL	MANH.	COS.	CORR.	MANH.	MANH.	MANH.
GO-LR k (CLUSTERS)	5	10	6	4	4	11	5	5
GO-LR REFINE PASSES	1	2	2	2	1	2	1	1
GO-LR DIR SELECT	F	F	T	T	F	F	F	F
GO-LR FEAT-SUB	3000	—	2000	2000	3000	2000	2000	2000
NSC SEG	UNIF.	UNIF.	UNIF.	UNIF.	UNIF.	UNIF.	LRG-JUMP	LRG-JUMP
NSC RULE	DEF.	DEF.	DEF.	DEF.	IDF	DEF.	IDF	IDF
NSC τ	0.99	0.99	0.95	0.95	0.95	0.95	0.99	0.99
NSC γ	2.05	1.98	2.12	1.07	2.26	1.60	2.65	2.65
NSC β	0.39	0.16	0.06	0.50	0.58	0.21	0.04	0.04
NSC $M_{\min}$	32	48	16	48	48	32	16	16
NSC $M_{\max}$	384	640	384	384	384	384	256	256
NSC $l_{\min}$	12	12	12	12	12	12	12	12
STD? (ASSUME STD.)	F	T	T	T	T	T	T	T
TABPFN SEED	42	3	1	4	4	3	3	3

Table I.1. Pairwise significance of GOTabPFN against the top baselines across the 8 HDLSS datasets. ΔAcc denotes the mean accuracy improvement of GOTabPFN over each baseline (percentage points) averaged across datasets. We report Wilcoxon signed-rank p -values (p_{raw}) and Holm-corrected p -values (p_{Holm}) for multiple comparisons; Sig. indicates significance after Holm correction ($\alpha = 0.05$).

Baseline	ΔAcc (pp)	n_{ds}	p_{raw}	p_{Holm}	Sig.
TANDEM	1.79	8	0.00781	0.0703	n.s.
TabPFN Wide	2.41	8	0.00781	0.0703	n.s.
TabDPT	2.98	8	0.00781	0.0703	n.s.
TabICL	4.33	8	0.00781	0.0703	n.s.
BETA (TabPFN Unleashed)	4.12	8	0.00781	0.0703	n.s.
TuneTables	4.87	8	0.00781	0.0703	n.s.
Lasso	7.04	8	0.00781	0.0703	n.s.
MLP	6.70	8	0.00781	0.0703	n.s.
ProtoGate	7.24	8	0.00781	0.0703	n.s.

Table I.2. Average-rank summary on the expanded 16-dataset benchmark. Lower rank is better. The Friedman test over the 9-method comparison is significant ($\chi^2 = 52.55$, $p = 1.32 \times 10^{-8}$; 11 complete rows).

Model	Avg. Rank	Std. Rank
GOTabPFN	1.12	0.50
TabPFN-Wide	3.62	1.75
TANDEM	3.69	1.62
TabDPT	4.34	1.49
TuneTables	5.53	2.29
TabICL	5.69	2.02
MLP	6.16	2.43
Lasso	6.62	1.59
ProtoGate	8.16	1.23

Table I.3. Omnibus Friedman test on the expanded 16-dataset benchmark. The test evaluates whether the 9 methods have equal rank distributions across datasets; 11 complete rows were used because Friedman requires all compared methods to be present.

# Datasets	# Methods	Complete Rows	Friedman χ^2	p -value
16	9	11	52.55	1.32×10^{-8}

Table I.4. Pairwise significance of GOTabPFN on the expanded 16-dataset benchmark. ΔAcc denotes the mean accuracy improvement of GOTabPFN over each baseline in percentage points, averaged over the datasets used for that comparison. W/T/L counts wins/ties/losses in favor of GOTabPFN. We report raw Wilcoxon signed-rank p -values and Holm-corrected p -values across the 8 pairwise comparisons.

Baseline	ΔAcc (pp)	n_{ds}	W/T/L	p_{raw}	p_{Holm}	Sig.
TANDEM	1.33	16	15/0/1	0.000305	0.000610	Yes
TabPFN-Wide	1.76	16	16/0/0	0.000031	0.000244	Yes
TabDPT	2.20	16	16/0/0	0.000031	0.000244	Yes
TabICL	2.81	14	14/0/0	0.000122	0.000488	Yes
TuneTables	4.36	16	16/0/0	0.000031	0.000244	Yes
MLP	4.65	16	15/0/1	0.000153	0.000488	Yes
Lasso	4.78	16	16/0/0	0.000031	0.000244	Yes
ProtoGate	11.90	12	12/0/0	0.000488	0.000610	Yes

Table I.5. Pairwise significance on the expanded 16-dataset benchmark. ΔAcc denotes the mean accuracy improvement of GOTabPFN over each baseline in percentage points. We report the number of datasets used (n_{ds}), raw Wilcoxon signed-rank p -values, and Holm-corrected p -values across the 8 pairwise comparisons. All comparisons remain significant after Holm correction.

Baseline	ΔAcc (pp)	n_{ds}	p_{raw}	p_{Holm}	Sig.
TANDEM	1.33	16	3.05×10^{-4}	6.10×10^{-4}	< 0.01
TabPFN-Wide	1.76	16	3.05×10^{-5}	2.44×10^{-4}	< 0.01
TabDPT	2.20	16	3.05×10^{-5}	2.44×10^{-4}	< 0.01
TabICL	2.81	14	1.22×10^{-4}	4.88×10^{-4}	< 0.01
TuneTables	4.36	16	3.05×10^{-5}	2.44×10^{-4}	< 0.01
MLP	4.65	16	1.53×10^{-4}	4.88×10^{-4}	< 0.01
Lasso	4.78	16	3.05×10^{-5}	2.44×10^{-4}	< 0.01
ProtoGate	11.90	12	4.88×10^{-4}	6.10×10^{-4}	< 0.01

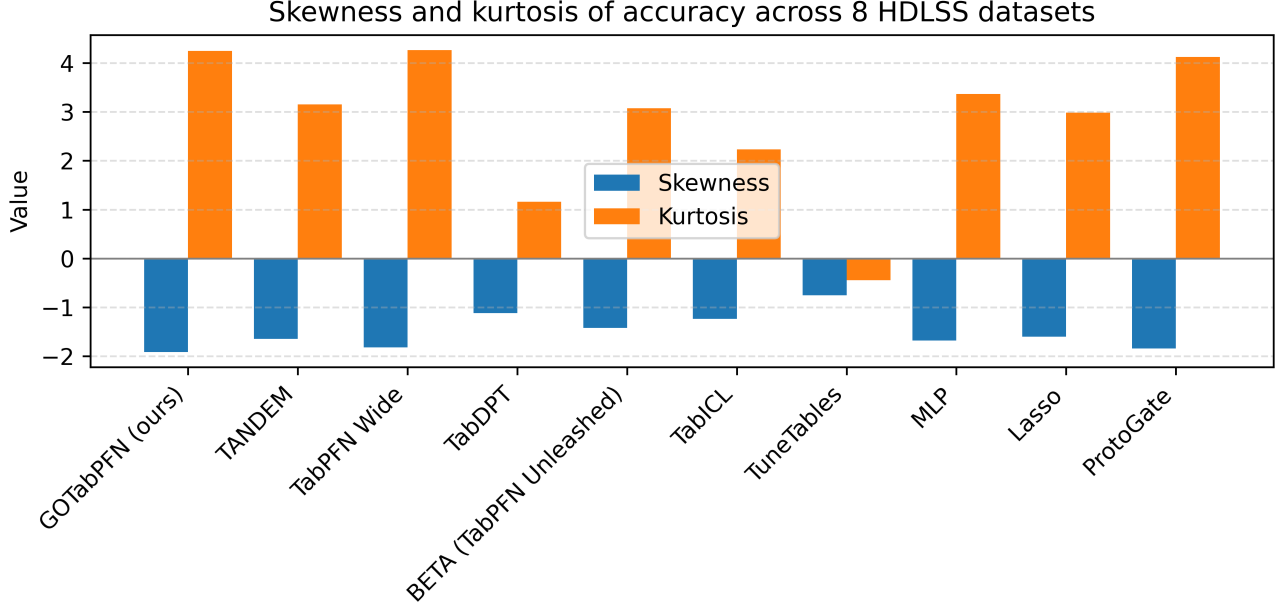


Figure J.1. Skewness/kurtosis for the top-10 methods on the 8 HDLSS benchmarks.

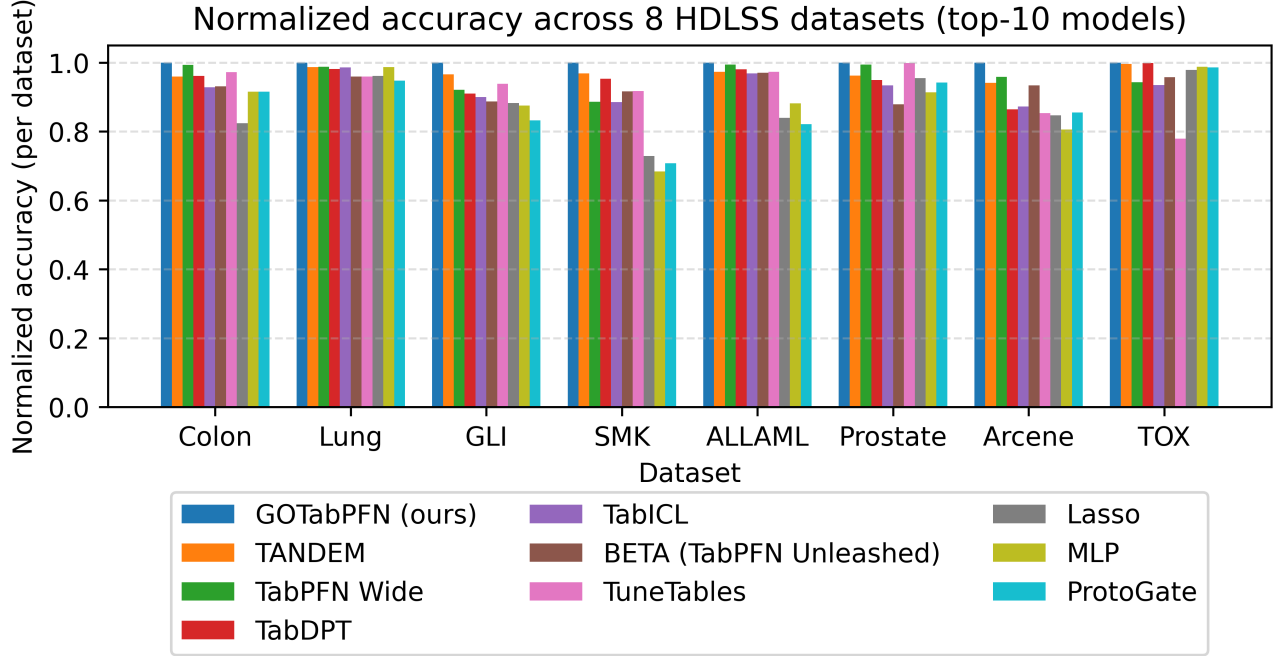


Figure J.2. Normalized accuracy for the top-10 methods on the 8 HDLSS benchmarks.

Table L.1. GOTabPFN (Colon) inference-time robustness to feature perturbations and latent-space neighborhood consistency. Tab Shuffle randomly permutes a fraction of feature columns across samples; Tab Drop replaces a fraction with the global feature mean. Higher is better for accuracy and kNN agreement.

Fraction perturbed	Tab Shuffle (%)	Tab Drop/mean (%)	kNN-Agree@5 (%)
0.00	98.39	98.39	73.87
0.10	96.77	100.00	73.87
0.25	88.71	93.55	73.87
0.50	74.19	98.39	73.87
0.75	62.90	61.29	73.87
1.00	58.06	64.52	73.87

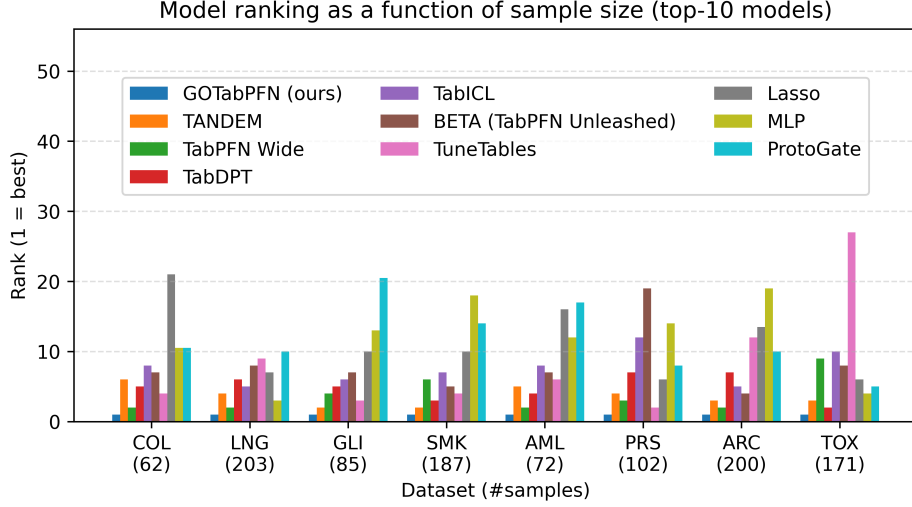


Figure J.3. Model rank versus sample size for the top-10 methods on the 8 HDLSS benchmarks.

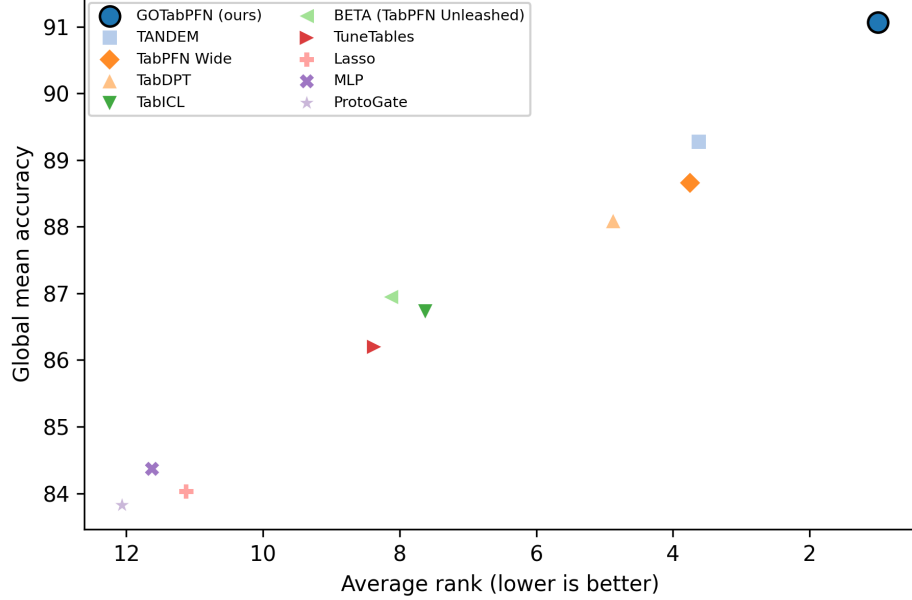


Figure J.4. Avg. rank vs. global accuracy
Table M.1. Sanity/stress and stability diagnostics for GOTabPFN on Colon dataset.

Sanity / stress mode	Accuracy (%)
Full input	98.39
All-zero input	64.52
Global-mean input	64.52
Shuffle rows	54.84
Heavy noise	95.16
TTA stability ($n_{\text{aug}}=5$)	
Base accuracy	98.39
TTA majority-vote accuracy	98.39
Any label change across aug (%)	3.23
Per-class accuracy	
Class 0 (support 40)	97.5
Class 1 (support 22)	100.0

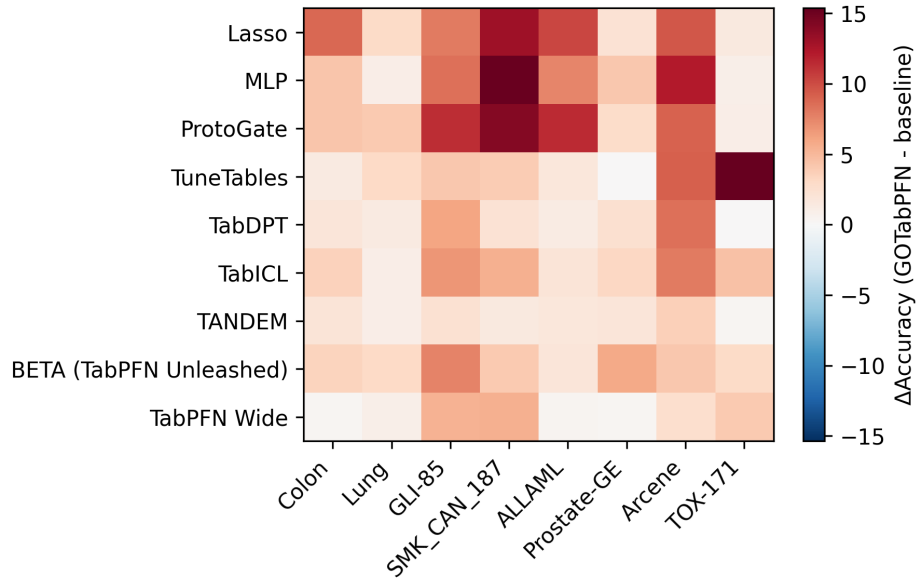


Figure J.5. Δ Acc heatmap (ours – baseline).

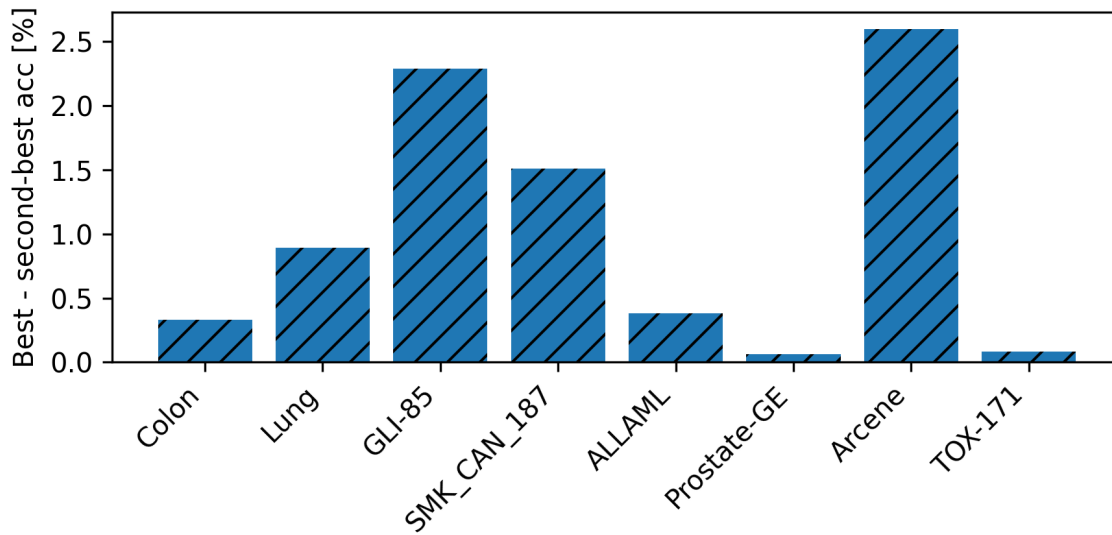


Figure J.6. Best-second-best margin across the 8 HDLSS benchmarks.

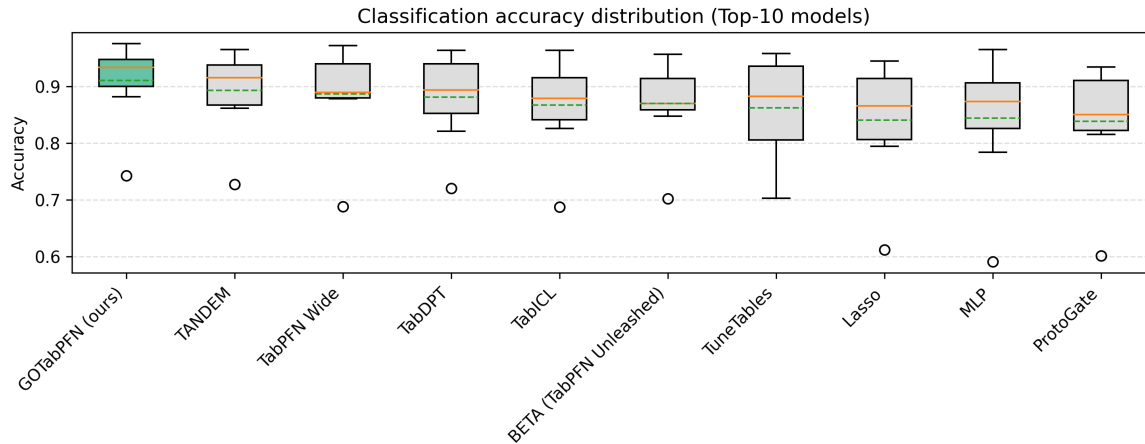


Figure J.7. Accuracy distributions across CV splits for the top-10 methods on the 8 HDLSS benchmarks.

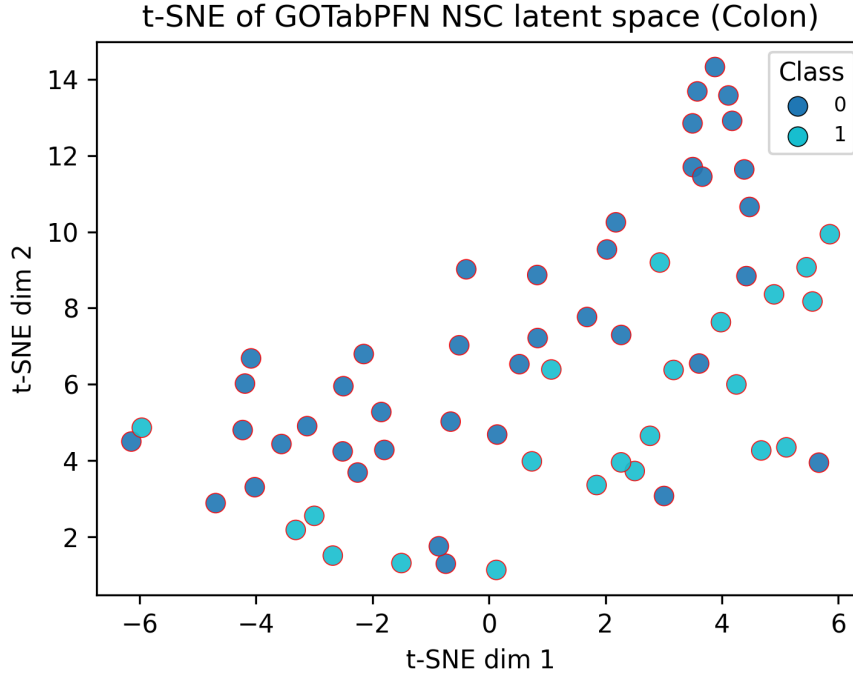


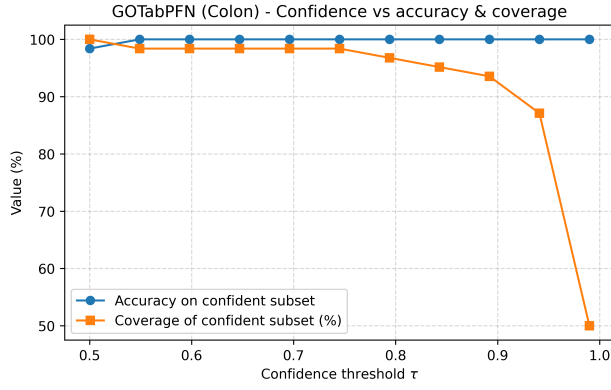
Figure K.1. t-SNE visualization of the NSC latent space in GOTabPFN on Colon (colored by class).

 Table R.1. **GO-LR + NSC as a model-agnostic front-end for TabICL.** We compare GO-LR + NSC + TabICL against vanilla TabICL on 8 HDLSS datasets under 5×5 CV. Values are mean with subscripted standard deviation for accuracy, ROC-AUC, and macro-F1; runtime is total seconds. Bold denotes the better value between the two methods for each metric/dataset.

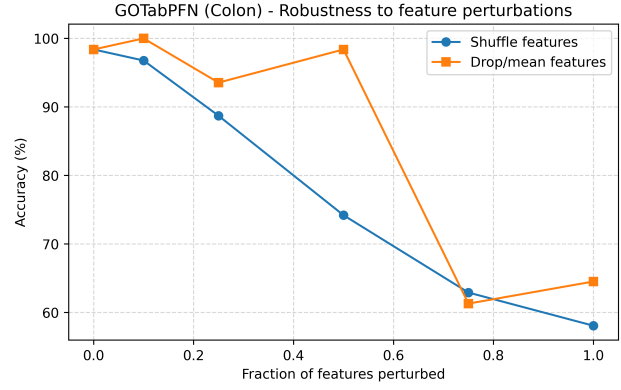
Dataset	Model	Accuracy $\uparrow$	ROC-AUC $\uparrow$	Macro-F1 $\uparrow$	Runtime $\downarrow$
COL	GO-LR+NSC+TabICL	86.56 ± 10.77	89.83 ± 10.36	81.36 ± 14.87	382.17
	TabICL	84.62 ± 10.52	89.38 ± 10.03	78.84 ± 13.87	942.21
LNG	GO-LR+NSC+TabICL	96.05 ± 2.58	99.62 ± 0.42	92.17 ± 7.57	382.35
	TabICL	96.36 ± 2.61	99.59 ± 0.45	93.72 ± 6.95	2602.42
GLI	GO-LR+NSC+TabICL	87.76 ± 7.39	92.62 ± 6.19	91.45 ± 5.15	117.65
	TabICL	87.06 ± 6.23	91.47 ± 6.98	90.91 ± 4.53	14598.50
SMK	GO-LR+NSC+TabICL	70.69 ± 5.46	76.00 ± 6.95	73.01 ± 5.23	315.10
	TabICL	68.73 ± 7.28	73.29 ± 7.99	70.92 ± 6.73	25011.40
AML	GO-LR+NSC+TabICL	94.78 ± 6.35	98.75 ± 3.49	92.22 ± 9.42	242.97
	TabICL	95.52 ± 5.59	98.27 ± 3.96	92.89 ± 9.10	2847.26
PRS	GO-LR+NSC+TabICL	88.25 ± 6.94	93.04 ± 6.17	87.87 ± 8.09	286.36
	TabICL	90.17 ± 5.93	93.92 ± 5.45	90.12 ± 5.80	2469.72
ARC	GO-LR+NSC+TabICL	85.20 ± 3.88	93.10 ± 2.25	83.21 ± 4.16	358.59
	TabICL	82.60 ± 6.14	91.48 ± 3.76	79.61 ± 6.26	7671.89
TOX	GO-LR+NSC+TabICL	90.87 ± 5.97	99.08 ± 0.98	90.95 ± 5.92	386.01
	TabICL	88.78 ± 5.92	98.62 ± 1.12	88.89 ± 5.79	2900.00

 Table S.1. **Average accuracy ($\uparrow$) across the 8 HDLSS datasets for different TabPFN seeds.** Values are mean accuracy with subscripted standard deviation across datasets.

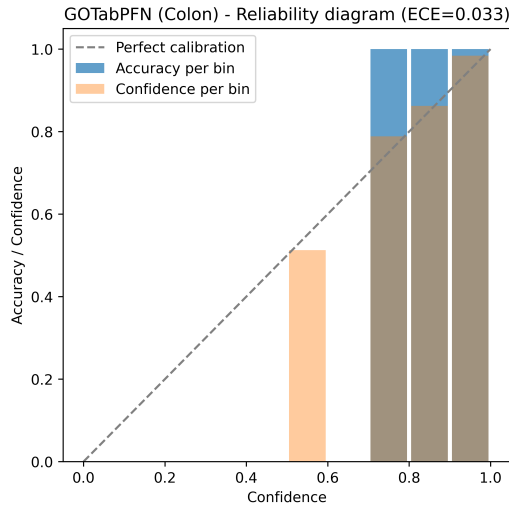
Model	Seed 42	Seed 77	Seed 82	Seed 93	Seed 147
TabPFN-Wide	87.51 ± 6.04	88.43 ± 4.87	88.33 ± 5.59	87.59 ± 5.60	87.69 ± 5.11
BETA	85.03 ± 8.70	86.10 ± 8.33	85.36 ± 8.36	85.87 ± 7.54	85.70 ± 7.65
TuneTables	88.12 ± 5.72	87.90 ± 5.65	88.12 ± 5.70	88.47 ± 5.08	88.20 ± 5.21
GOTabPFN	90.57 ± 5.54	89.66 ± 5.72	89.93 ± 5.17	89.79 ± 5.30	89.66 ± 5.59



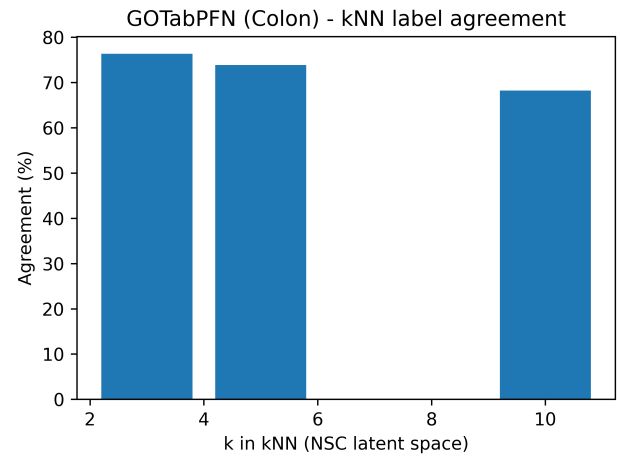
(a) Confidence vs. accuracy & coverage.



(b) Robustness to perturbations.

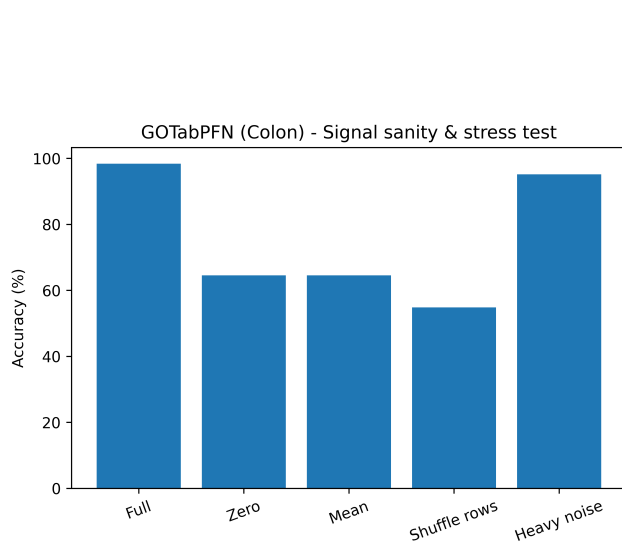


(c) Reliability diagram (ECE).

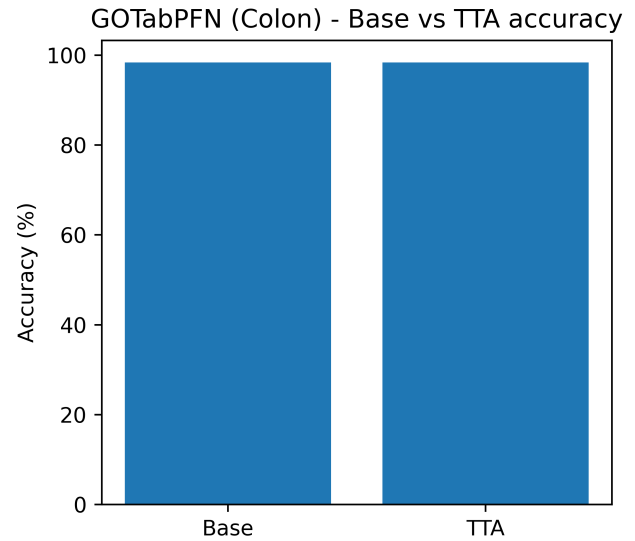


(d) k NN label agreement.

Figure L.1. GOTabPFN (Colon): confidence/coverage, calibration, robustness, and latent-space neighborhood agreement.



(a) Signal sanity & stress tests.



(b) Base vs. TTA accuracy.

Figure M.1. Sanity and stress diagnostics on Colon.

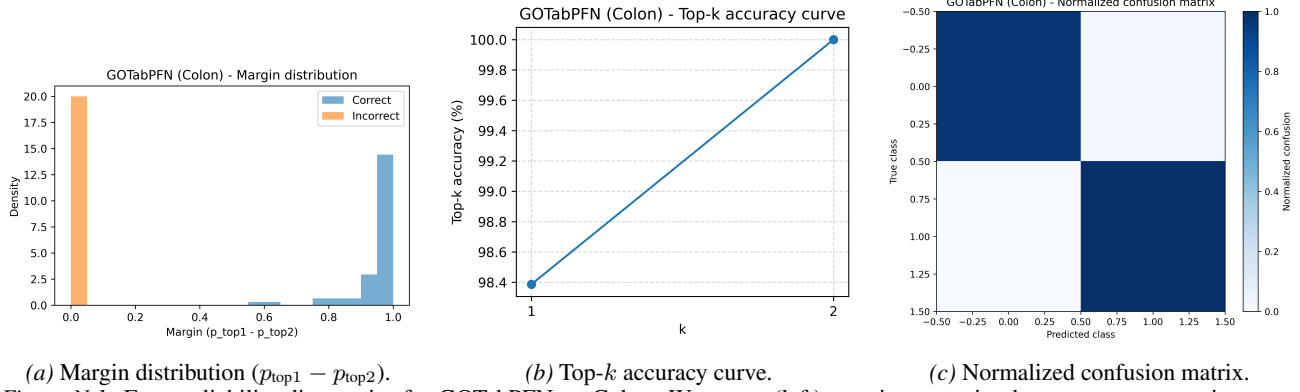


Figure N.1. Extra reliability diagnostics for GOTabPFN on Colon. We report (left) margin separation between correct vs. incorrect predictions, (middle) Top- k accuracy, and (right) normalized confusion matrix.

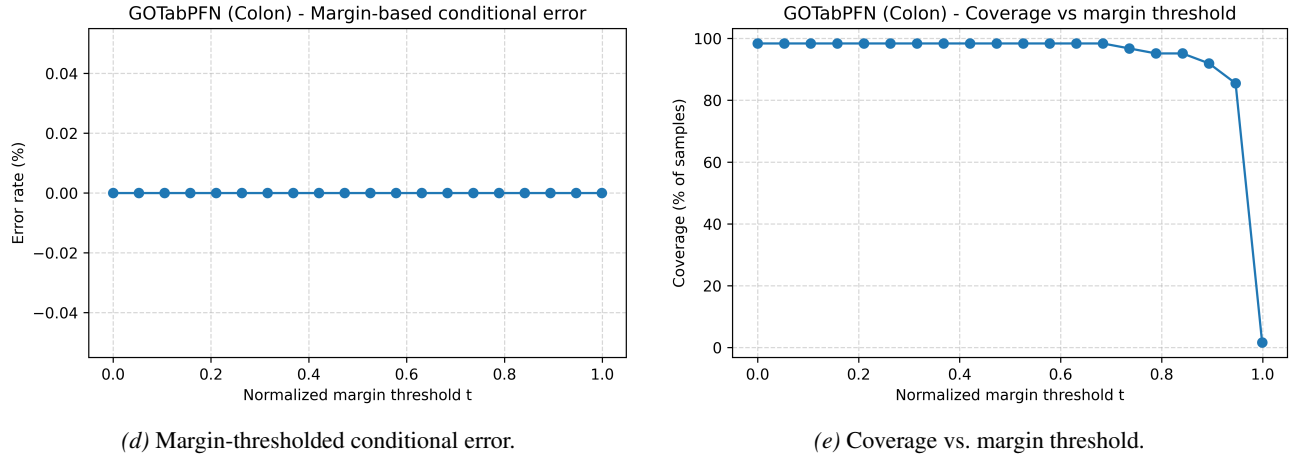
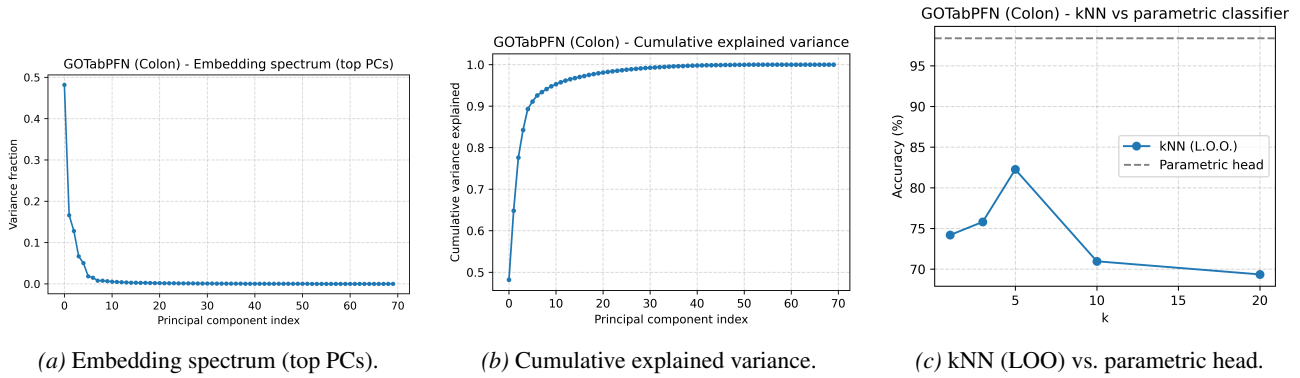
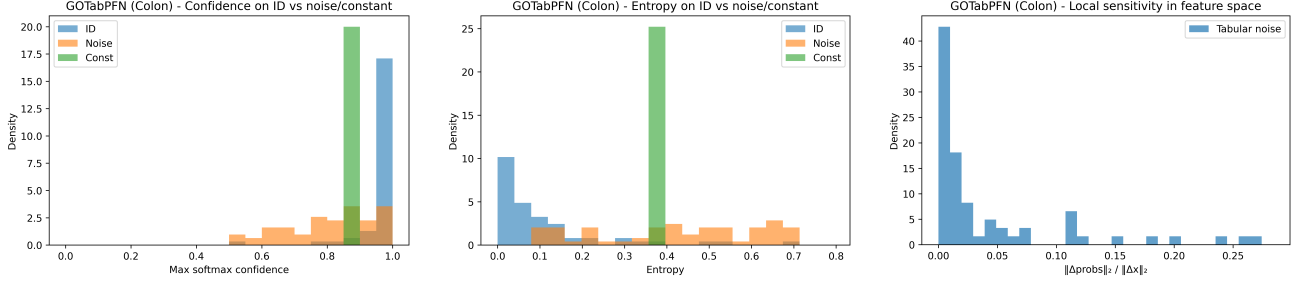


Figure O.1. **Theory-inspired representation diagnostics for GOTabPFN (Colon).** (a) Variance spectrum indicates a sharp concentration of energy in the leading PCs. (b) Cumulative explained variance shows rapid saturation. (c) Leave-one-out kNN in embedding space underperforms the parametric head, suggesting performance is not explained by simple local geometry alone. (d-e) Margin-based conditional error and coverage demonstrate high-confidence predictions over most samples.

Table S.2. **TabPFN-seed robustness on 8 cross-domain datasets.** Average accuracy across the 8 additional cross-domain datasets for different TabPFN seeds. Values are mean with subscripted standard deviation across datasets. BETA is omitted because it was not run on all datasets and required substantially longer runtime.

Model	Seed 42	Seed 77	Seed 82	Seed 93	Seed 147
TabPFN-W	86.60 $\pm$ 2.95	84.84 $\pm$ 3.35	85.72 $\pm$ 3.04	83.71 $\pm$ 3.24	86.84 $\pm$ 3.10
TuneTables	83.65 $\pm$ 2.10	82.43 $\pm$ 2.33	83.19 $\pm$ 2.20	81.68 $\pm$ 2.28	83.60 $\pm$ 2.22
GOTabPFN	86.85 $\pm$ 2.55	86.58 $\pm$ 2.85	86.94 $\pm$ 2.50	86.90 $\pm$ 2.55	86.88 $\pm$ 2.66



(a) Max-softmax confidence: ID vs. Noise/Const. (b) Predictive entropy: ID vs. Noise/Const. (c) Local sensitivity in feature space.

Figure P.1. OOD and sensitivity-style diagnostics for GOTabPFN on Colon. Confidence/entropy histograms compare in-distribution (ID) inputs with synthetic OOD tabular perturbations (Noise, Const), while the right panel reports a Lipschitz-like local sensitivity score $\|\Delta \text{probs}\|_2 / \|\Delta x\|_2$ under small feature-space noise.

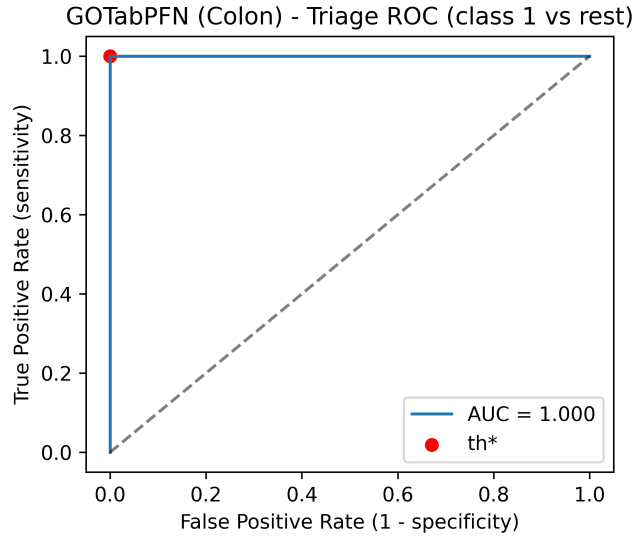
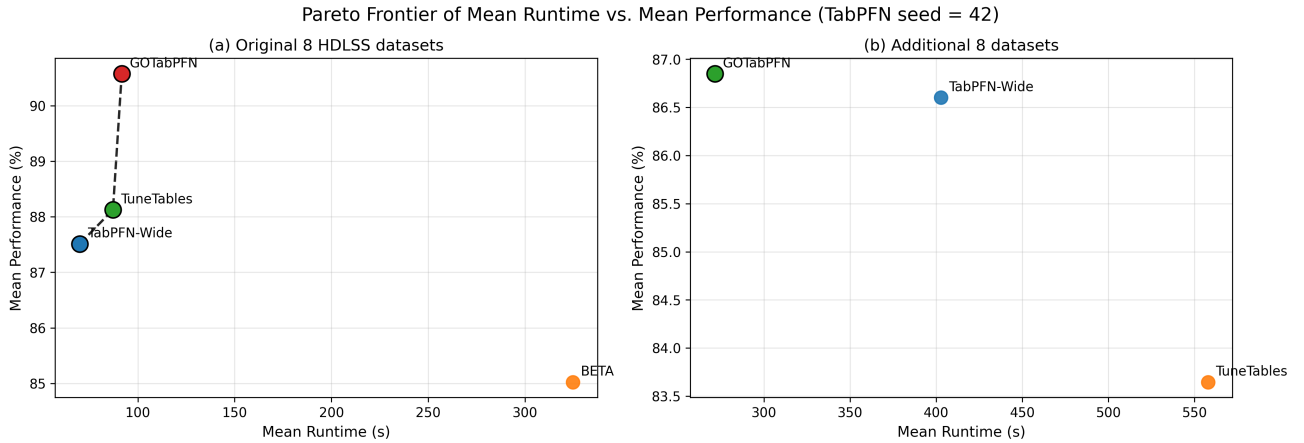


Figure Q.1. Deployment-style triage diagnostic on Colon (class 1 vs rest). ROC curve for treating class 1 as the “high-risk” positive class. The marked operating point (th*) corresponds to the selected threshold achieving the target sensitivity criterion, with the best specificity among feasible thresholds.



Dashed lines connect Pareto-optimal methods. BETA is excluded from panel (b) because it was not run on Cell Cycle and DrivFace Regression, and it required substantially longer runtime.

Figure S.1. Pareto frontier of mean runtime vs. mean performance for high-dimensionality compatible TabPFN-family methods. Results use TabPFN seed 42. Lower runtime and higher performance are better. Dashed lines connect Pareto optimal methods. Left: original 8 HDLSS datasets. Right: additional 8 cross-domain datasets. BETA is excluded from the right panel because it was not run on Cell Cycle and DrivFace-Regression and required substantially longer runtime.

GOTabPFN: From Feature Ordering to Compact Tokenization for Tabular Foundation Models on High-Dimensional Data

Table S.3. Accuracy comparison across 8 HDLSS datasets for different TabPFN seeds. Values are mean accuracy with subscripted standard deviation over 5×5 CV. The Avg. column reports the mean across the 8 datasets, with the subscript giving the mean of the corresponding standard deviations.

Seed 42									
Model	COL	LNG	AML	TOX	PRS	ARC	SMK	GLI	Avg.
TabPFN-Wide	87.05 $\pm$ 7.44	96.04 $\pm$ 2.85	95.90 $\pm$ 3.74	84.15 $\pm$ 6.17	92.19 $\pm$ 7.21	86.00 $\pm$ 2.85	70.53 $\pm$ 9.73	88.24 $\pm$ 8.32	87.51 $\pm$ 6.04
BETA	84.80 $\pm$ 13.01	95.00 $\pm$ 3.41	89.40 $\pm$ 5.82	83.20 $\pm$ 12.73	88.40 $\pm$ 9.89	84.50 $\pm$ 6.20	69.80 $\pm$ 10.10	85.10 $\pm$ 8.40	85.03 $\pm$ 8.70
TuneTables	86.20 $\pm$ 6.80	96.30 $\pm$ 2.50	94.10 $\pm$ 5.20	92.10 $\pm$ 4.80	91.10 $\pm$ 6.30	85.40 $\pm$ 3.90	71.20 $\pm$ 8.80	88.60 $\pm$ 7.50	88.12 $\pm$ 5.72
GOTabPFN	88.18 $\pm$ 10.05	97.44 $\pm$ 2.32	97.54 $\pm$ 4.31	92.75 $\pm$ 4.07	91.95 $\pm$ 7.38	90.30 $\pm$ 3.70	73.61 $\pm$ 5.71	92.82 $\pm$ 6.81	90.57 $\pm$ 5.54
Seed 77									
TabPFN-Wide	85.38 $\pm$ 7.07	96.55 $\pm$ 2.85	95.81 $\pm$ 3.83	90.10 $\pm$ 5.94	92.24 $\pm$ 5.44	88.50 $\pm$ 4.87	70.60 $\pm$ 3.10	88.24 $\pm$ 5.88	88.43 $\pm$ 4.87
BETA	83.00 $\pm$ 10.37	94.20 $\pm$ 2.93	91.40 $\pm$ 8.52	88.00 $\pm$ 12.46	87.40 $\pm$ 9.77	85.20 $\pm$ 7.10	70.10 $\pm$ 6.30	89.50 $\pm$ 9.20	86.10 $\pm$ 8.33
TuneTables	84.90 $\pm$ 8.50	96.10 $\pm$ 2.90	94.80 $\pm$ 5.40	91.20 $\pm$ 4.50	90.90 $\pm$ 5.80	86.40 $\pm$ 5.10	70.80 $\pm$ 5.60	88.10 $\pm$ 7.40	87.90 $\pm$ 5.65
GOTabPFN	87.21 $\pm$ 9.28	97.33 $\pm$ 2.38	96.99 $\pm$ 5.18	92.04 $\pm$ 4.91	89.39 $\pm$ 6.94	90.10 $\pm$ 3.57	74.11 $\pm$ 6.37	90.12 $\pm$ 7.15	89.66 $\pm$ 5.72
Seed 82									
TabPFN-Wide	86.92 $\pm$ 11.33	97.05 $\pm$ 3.20	95.71 $\pm$ 3.91	88.89 $\pm$ 5.25	91.19 $\pm$ 3.99	88.00 $\pm$ 3.26	70.63 $\pm$ 6.55	88.24 $\pm$ 7.20	88.33 $\pm$ 5.59
BETA	82.80 $\pm$ 13.01	94.80 $\pm$ 1.83	90.60 $\pm$ 9.99	86.20 $\pm$ 10.96	86.40 $\pm$ 9.48	84.80 $\pm$ 6.80	70.40 $\pm$ 7.20	86.90 $\pm$ 7.60	85.36 $\pm$ 8.36
TuneTables	85.10 $\pm$ 9.40	96.80 $\pm$ 3.20	94.30 $\pm$ 6.10	92.80 $\pm$ 4.90	90.20 $\pm$ 4.70	85.90 $\pm$ 4.80	71.50 $\pm$ 5.90	88.40 $\pm$ 6.60	88.12 $\pm$ 5.70
GOTabPFN	87.21 $\pm$ 9.06	97.34 $\pm$ 2.15	98.10 $\pm$ 3.67	93.10 $\pm$ 4.03	90.19 $\pm$ 6.19	89.60 $\pm$ 3.59	73.58 $\pm$ 6.59	90.35 $\pm$ 6.09	89.93 $\pm$ 5.17
Seed 93									
TabPFN-Wide	84.10 $\pm$ 10.90	96.06 $\pm$ 1.34	97.14 $\pm$ 3.91	90.62 $\pm$ 3.87	91.16 $\pm$ 3.37	85.50 $\pm$ 6.22	70.00 $\pm$ 11.73	86.12 $\pm$ 3.42	87.59 $\pm$ 5.60
BETA	82.80 $\pm$ 13.01	95.20 $\pm$ 2.80	94.00 $\pm$ 5.10	86.20 $\pm$ 9.93	87.20 $\pm$ 8.35	84.90 $\pm$ 7.00	70.30 $\pm$ 8.50	86.34 $\pm$ 5.65	85.87 $\pm$ 7.54
TuneTables	86.40 $\pm$ 7.90	96.30 $\pm$ 2.10	95.60 $\pm$ 4.30	94.50 $\pm$ 3.80	91.30 $\pm$ 5.20	84.80 $\pm$ 5.10	71.00 $\pm$ 6.80	87.90 $\pm$ 5.40	88.47 $\pm$ 5.08
GOTabPFN	87.85 $\pm$ 9.04	97.34 $\pm$ 2.58	97.52 $\pm$ 3.88	93.57 $\pm$ 4.38	90.16 $\pm$ 6.69	89.80 $\pm$ 3.88	72.42 $\pm$ 5.73	89.65 $\pm$ 6.19	89.79 $\pm$ 5.30
Seed 147									
TabPFN-Wide	83.97 $\pm$ 5.25	96.56 $\pm$ 2.20	95.81 $\pm$ 3.83	87.71 $\pm$ 4.38	91.34 $\pm$ 3.82	89.50 $\pm$ 2.09	68.42 $\pm$ 12.07	88.24 $\pm$ 7.20	87.69 $\pm$ 5.11
BETA	80.20 $\pm$ 11.60	95.00 $\pm$ 3.41	93.00 $\pm$ 4.43	86.54 $\pm$ 10.30	88.60 $\pm$ 8.05	85.10 $\pm$ 6.50	69.90 $\pm$ 9.80	87.30 $\pm$ 7.10	85.70 $\pm$ 7.65
TuneTables	83.20 $\pm$ 5.90	96.20 $\pm$ 2.30	94.40 $\pm$ 5.60	93.80 $\pm$ 3.90	90.80 $\pm$ 4.20	86.70 $\pm$ 4.40	71.60 $\pm$ 9.30	88.90 $\pm$ 6.10	88.20 $\pm$ 5.21
GOTabPFN	88.49 $\pm$ 9.60	97.14 $\pm$ 2.12	97.28 $\pm$ 4.77	92.28 $\pm$ 4.85	89.18 $\pm$ 6.98	90.10 $\pm$ 3.78	72.94 $\pm$ 6.36	89.88 $\pm$ 6.24	89.66 $\pm$ 5.59

Table S.4. Accuracy comparison across 8 cross-domain datasets for different TabPFN seeds. Values are mean accuracy with subscripted standard deviation over 5×5 CV. The Avg. column reports the mean across datasets within each seed block, with the subscript giving the mean of the corresponding standard deviations. BETA is reported only for seed 42 and its average is computed over 6 datasets because Cell Cycle and DrivFace-Regression were omitted due to substantially longer runtime.

Seed 42									
Model	ORL	BAS	REL	PCM	CCY	CIF	DF-R	DF-C	Avg.
TabPFN-W	95.80 $\pm$ 6.30	96.70 $\pm$ 0.70	89.20 $\pm$ 3.60	88.65 $\pm$ 0.60	77.90 $\pm$ 2.30	88.20 $\pm$ 0.60	70.10 $\pm$ 7.00	86.27 $\pm$ 2.50	86.60 $\pm$ 2.95
TuneTables	92.40 $\pm$ 1.10	92.90 $\pm$ 1.00	89.10 $\pm$ 2.10	83.87 $\pm$ 2.80	77.50 $\pm$ 1.20	78.20 $\pm$ 0.20	68.90 $\pm$ 6.20	86.30 $\pm$ 2.20	83.65 $\pm$ 2.10
BETA	93.00 $\pm$ 5.10	96.80 $\pm$ 1.17	89.20 $\pm$ 2.04	89.13 $\pm$ 1.94	-	88.20 $\pm$ 0.75	-	80.60 $\pm$ 1.36	89.49 $\pm$ 2.06 [†]
GOTabPFN	100.00 $\pm$ 0.00	97.04 $\pm$ 1.11	88.79 $\pm$ 1.37	89.28 $\pm$ 2.25	79.34 $\pm$ 2.68	88.50 $\pm$ 0.93	65.51 $\pm$ 9.83	86.34 $\pm$ 2.22	86.85 $\pm$ 2.55
Seed 77									
TabPFN-W	96.30 $\pm$ 6.90	96.50 $\pm$ 0.80	85.60 $\pm$ 4.10	89.30 $\pm$ 0.80	79.40 $\pm$ 2.60	87.50 $\pm$ 0.50	60.20 $\pm$ 8.20	83.90 $\pm$ 2.90	84.84 $\pm$ 3.35
TuneTables	94.80 $\pm$ 1.00	93.80 $\pm$ 1.10	86.20 $\pm$ 2.50	86.90 $\pm$ 3.20	79.20 $\pm$ 1.00	77.60 $\pm$ 0.10	58.20 $\pm$ 7.10	82.70 $\pm$ 2.60	82.43 $\pm$ 2.33
GOTabPFN	100.00 $\pm$ 0.00	97.05 $\pm$ 1.13	88.56 $\pm$ 1.60	89.52 $\pm$ 2.35	79.57 $\pm$ 2.85	86.27 $\pm$ 2.39	65.39 $\pm$ 10.08	86.27 $\pm$ 2.39	86.58 $\pm$ 2.85
Seed 82									
TabPFN-W	94.60 $\pm$ 6.40	97.80 $\pm$ 0.70	88.40 $\pm$ 3.80	87.70 $\pm$ 0.70	76.80 $\pm$ 2.20	88.90 $\pm$ 0.60	65.50 $\pm$ 7.50	86.10 $\pm$ 2.40	85.72 $\pm$ 3.04
TuneTables	93.10 $\pm$ 1.10	93.20 $\pm$ 1.00	88.70 $\pm$ 2.20	84.50 $\pm$ 3.00	78.40 $\pm$ 1.20	78.40 $\pm$ 0.20	64.10 $\pm$ 6.60	85.10 $\pm$ 2.30	83.19 $\pm$ 2.20
GOTabPFN	100.00 $\pm$ 0.00	96.99 $\pm$ 1.01	88.80 $\pm$ 1.62	89.63 $\pm$ 2.05	79.32 $\pm$ 2.34	88.66 $\pm$ 0.81	65.66 $\pm$ 9.60	86.44 $\pm$ 2.53	86.94 $\pm$ 2.50
Seed 93									
TabPFN-W	90.20 $\pm$ 6.80	96.90 $\pm$ 0.60	84.90 $\pm$ 4.00	88.80 $\pm$ 0.70	81.30 $\pm$ 2.50	87.80 $\pm$ 0.50	57.10 $\pm$ 8.00	82.70 $\pm$ 2.80	83.71 $\pm$ 3.24
TuneTables	91.90 $\pm$ 1.00	92.70 $\pm$ 1.10	87.30 $\pm$ 2.40	82.80 $\pm$ 3.10	76.90 $\pm$ 1.10	77.80 $\pm$ 0.10	60.10 $\pm$ 6.90	83.90 $\pm$ 2.50	81.68 $\pm$ 2.28
GOTabPFN	100.00 $\pm$ 0.00	97.09 $\pm$ 1.02	88.63 $\pm$ 1.41	89.52 $\pm$ 2.02	79.44 $\pm$ 2.67	88.38 $\pm$ 0.89	65.62 $\pm$ 9.69	86.54 $\pm$ 2.70	86.90 $\pm$ 2.55
Seed 147									
TabPFN-W	97.60 $\pm$ 6.20	96.70 $\pm$ 0.70	91.10 $\pm$ 3.70	88.90 $\pm$ 0.70	77.90 $\pm$ 2.40	87.80 $\pm$ 0.60	68.60 $\pm$ 7.90	86.10 $\pm$ 2.60	86.84 $\pm$ 3.10
TuneTables	93.50 $\pm$ 1.00	93.10 $\pm$ 1.00	90.00 $\pm$ 2.30	84.10 $\pm$ 2.90	79.50 $\pm$ 1.20	78.30 $\pm$ 0.20	65.30 $\pm$ 6.80	85.00 $\pm$ 2.40	83.60 $\pm$ 2.22
GOTabPFN	99.80 $\pm$ 1.00	97.08 $\pm$ 1.08	88.75 $\pm$ 1.78	89.31 $\pm$ 2.12	79.64 $\pm$ 2.58	88.34 $\pm$ 0.74	65.55 $\pm$ 9.69	86.54 $\pm$ 2.33	86.88 $\pm$ 2.66

[†] For BETA at seed 42, Cell Cycle and DrivFace-Regression are unavailable because the runs were prohibitively time-consuming; its average is computed over the remaining 6 datasets.

Table S.5. ROC-AUC comparison across 8 HDLSS datasets for different TabPFN seeds. Values are mean ROC-AUC with subscripted standard deviation over 5×5 CV. The Avg. column reports the mean across the 8 datasets, with the subscript giving the mean of the corresponding standard deviations.

Seed 42									
Model	COL	LNG	AML	TOX	PRS	ARC	SMK	GLI	Avg.
TabPFN-Wide	88.25 $\pm$ 6.27	99.49 $\pm$ 0.55	98.40 $\pm$ 3.58	97.72 $\pm$ 1.77	96.07 $\pm$ 5.63	93.56 $\pm$ 2.63	77.97 $\pm$ 8.85	94.76 $\pm$ 4.93	93.28 $\pm$ 4.28
BETA	92.60 $\pm$ 5.54	99.00 $\pm$ 1.10	94.40 $\pm$ 3.38	96.60 $\pm$ 2.94	95.20 $\pm$ 4.90	91.80 $\pm$ 3.40	76.10 $\pm$ 8.20	93.40 $\pm$ 6.10	92.39 $\pm$ 4.44
TuneTables	90.10 $\pm$ 6.20	99.10 $\pm$ 0.80	97.20 $\pm$ 3.10	98.80 $\pm$ 1.20	95.80 $\pm$ 4.90	92.10 $\pm$ 3.10	76.80 $\pm$ 8.00	94.20 $\pm$ 6.20	93.01 $\pm$ 4.19
GOTabPFN	91.40 $\pm$ 10.12	99.58 $\pm$ 0.60	99.44 $\pm$ 1.47	99.23 $\pm$ 0.63	94.32 $\pm$ 5.39	95.85 $\pm$ 2.29	80.92 $\pm$ 5.63	90.97 $\pm$ 7.09	93.96 $\pm$ 4.15
Seed 77									
TabPFN-Wide	86.75 $\pm$ 11.31	99.84 $\pm$ 0.27	99.60 $\pm$ 0.89	98.21 $\pm$ 1.89	97.13 $\pm$ 3.14	95.08 $\pm$ 4.13	76.03 $\pm$ 7.48	91.39 $\pm$ 8.83	93.00 $\pm$ 4.74
BETA	89.80 $\pm$ 7.49	99.40 $\pm$ 0.49	99.20 $\pm$ 1.60	96.80 $\pm$ 2.71	94.80 $\pm$ 4.70	91.20 $\pm$ 4.60	75.20 $\pm$ 7.50	92.30 $\pm$ 7.90	92.34 $\pm$ 4.62
TuneTables	89.20 $\pm$ 9.80	99.60 $\pm$ 0.60	99.10 $\pm$ 1.40	98.90 $\pm$ 1.50	95.90 $\pm$ 4.20	92.80 $\pm$ 4.10	75.90 $\pm$ 7.30	92.60 $\pm$ 7.40	93.00 $\pm$ 4.54
GOTabPFN	91.75 $\pm$ 9.86	99.65 $\pm$ 0.50	99.12 $\pm$ 2.39	99.18 $\pm$ 0.72	93.93 $\pm$ 5.13	95.82 $\pm$ 2.26	81.03 $\pm$ 5.66	91.07 $\pm$ 6.94	93.94 $\pm$ 4.18
Seed 82									
TabPFN-Wide	83.38 $\pm$ 15.27	99.75 $\pm$ 0.29	99.56 $\pm$ 0.99	97.83 $\pm$ 1.64	95.78 $\pm$ 4.29	96.61 $\pm$ 2.55	78.74 $\pm$ 4.11	95.21 $\pm$ 6.63	93.36 $\pm$ 4.47
BETA	90.40 $\pm$ 6.62	99.20 $\pm$ 0.75	95.80 $\pm$ 5.15	96.80 $\pm$ 2.71	95.10 $\pm$ 3.90	92.00 $\pm$ 2.80	78.20 $\pm$ 4.80	92.80 $\pm$ 6.20	92.54 $\pm$ 4.12
TuneTables	88.90 $\pm$ 7.20	99.70 $\pm$ 0.40	98.90 $\pm$ 1.80	99.10 $\pm$ 1.10	95.60 $\pm$ 3.50	93.40 $\pm$ 2.70	79.10 $\pm$ 4.50	93.40 $\pm$ 6.40	93.51 $\pm$ 3.45
GOTabPFN	91.03 $\pm$ 10.30	99.51 $\pm$ 0.84	99.04 $\pm$ 2.52	99.32 $\pm$ 0.68	94.20 $\pm$ 5.32	95.84 $\pm$ 2.22	79.06 $\pm$ 5.57	90.47 $\pm$ 7.82	93.56 $\pm$ 4.41
Seed 93									
TabPFN-Wide	91.13 $\pm$ 10.24	99.65 $\pm$ 0.46	99.56 $\pm$ 0.99	97.85 $\pm$ 1.70	97.85 $\pm$ 3.48	93.86 $\pm$ 5.30	76.68 $\pm$ 5.77	93.33 $\pm$ 7.36	93.74 $\pm$ 4.41
BETA	90.40 $\pm$ 8.01	99.20 $\pm$ 0.75	96.40 $\pm$ 2.65	96.00 $\pm$ 2.61	95.30 $\pm$ 4.10	91.00 $\pm$ 3.90	75.50 $\pm$ 6.90	92.50 $\pm$ 7.20	92.04 $\pm$ 4.52
TuneTables	90.50 $\pm$ 10.20	99.50 $\pm$ 0.50	98.90 $\pm$ 1.60	99.00 $\pm$ 1.30	96.20 $\pm$ 4.00	92.40 $\pm$ 3.50	75.80 $\pm$ 6.40	92.90 $\pm$ 7.00	93.15 $\pm$ 4.31
GOTabPFN	90.68 $\pm$ 10.34	99.61 $\pm$ 0.60	99.28 $\pm$ 2.07	99.24 $\pm$ 0.76	94.28 $\pm$ 5.41	95.67 $\pm$ 2.47	79.93 $\pm$ 5.40	90.98 $\pm$ 6.88	93.71 $\pm$ 4.24
Seed 147									
TabPFN-Wide	83.87 $\pm$ 11.80	99.44 $\pm$ 0.42	99.56 $\pm$ 0.99	98.23 $\pm$ 1.44	96.42 $\pm$ 3.44	96.11 $\pm$ 1.51	79.18 $\pm$ 12.89	95.36 $\pm$ 5.21	93.52 $\pm$ 4.71
BETA	90.20 $\pm$ 6.01	99.40 $\pm$ 0.49	97.80 $\pm$ 4.40	96.80 $\pm$ 3.12	95.60 $\pm$ 3.80	92.40 $\pm$ 3.10	78.50 $\pm$ 12.10	93.20 $\pm$ 5.90	92.99 $\pm$ 4.86
TuneTables	88.10 $\pm$ 11.40	99.30 $\pm$ 0.60	98.70 $\pm$ 2.20	99.20 $\pm$ 1.10	95.90 $\pm$ 3.60	93.10 $\pm$ 3.20	78.90 $\pm$ 11.80	93.80 $\pm$ 5.50	93.38 $\pm$ 4.93
GOTabPFN	91.23 $\pm$ 10.75	99.57 $\pm$ 0.56	99.20 $\pm$ 2.24	99.08 $\pm$ 0.99	93.88 $\pm$ 5.35	95.55 $\pm$ 2.47	79.64 $\pm$ 5.94	90.92 $\pm$ 7.02	93.63 $\pm$ 4.42

Table S.6. TabPFN inference-seed sensitivity of GOTabPFN on Colon. GO-LR, NSC, preprocessing, and the exact same 5×5 CV splits are fixed; only the TabPFN inference seed is varied. Values are mean accuracy with subscripted standard deviation over 5×5 CV.

Seed	0	1	2	3	4	7	11	17	23	42
GOTabPFN	87.51 $\pm$ 8.72	86.56 $\pm$ 9.25	87.18 $\pm$ 8.72	86.90 $\pm$ 8.34	86.87 $\pm$ 8.71	86.85 $\pm$ 8.36	87.15 $\pm$ 9.34	87.51 $\pm$ 9.36	87.21 $\pm$ 8.73	88.18 $\pm$ 10.05
Across seeds	Mean = 87.19, std. = 0.46, range = 86.56-88.18 (1.62 pp)									

Table T.1. Additional cross-domain datasets. Datasets are categorized using the empirical regime rule in Appendix F, following DynaTab (Habib et al., 2026b). Here, n is the number of instances, m is the number of features, and $\rho = m/n$ is the feature-to-sample ratio. CIFAR-10 uses subsampled ResNet-50 image embeddings.

Dataset	Source	Type	n	m	#Classes	ρ	Category
BASEHOCK	scikit-feature (Li et al., 2018) ¹	Text (bag-of-words)	1993	4862	2	2.44	MixedRegime
PCMAC	scikit-feature (Li et al., 2018) ¹	Text (bag-of-words)	1943	3289	2	1.69	MixedRegime
RELATHE	scikit-feature (Li et al., 2018) ¹	Text (bag-of-words)	1427	4322	2	3.03	MixedRegime
orlraws10P	scikit-feature (Li et al., 2018) ¹	Face image	100	10304	10	103.04	HDLSS
DrivFace-C	UCI (Hernández-Sabat et al., 2016) ²	Camera sensor	606	6400	7	10.56	HDLSS
DrivFace-R	UCI (Hernández-Sabat et al., 2016) ²	Camera sensor	606	6400	Reg.	10.56	HDLSS
CIFAR-10	Kaggle (Cukierski, 2013) ³	Image features	11000	2048	10	0.19	HDHSS
Cell Cycle	NCBI (Mahdessian et al., 2021) ⁴	RNA-seq	1067	42728	3	40.05	MixedRegime

¹ scikit-feature dataset repository: <https://jundongli.github.io/scikit-feature/datasets.html>. ² DrivFace dataset: <https://archive.ics.uci.edu/dataset/378/drivface>. ³ Kaggle CIFAR-10 dataset: <https://www.kaggle.com/competitions/cifar-10>. ⁴ Cell Cycle GEO accession: <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146773>. **Categorization rule.** A dataset is labeled **HDLSS** if $m > 1000$, $n < 1000$, and $\rho > 2$; **HDHSS** if $m > 1000$, $n > 10^4$, and $0.005 < \rho \leq 2$; **LDHSS** if $m \leq 100$, $n > 10^4$, and $\rho \leq 0.01$; **LDLSS** if $m \leq 100$, $n \leq 1000$, and $\rho \leq 0.05$; and **MixedRegime** otherwise.

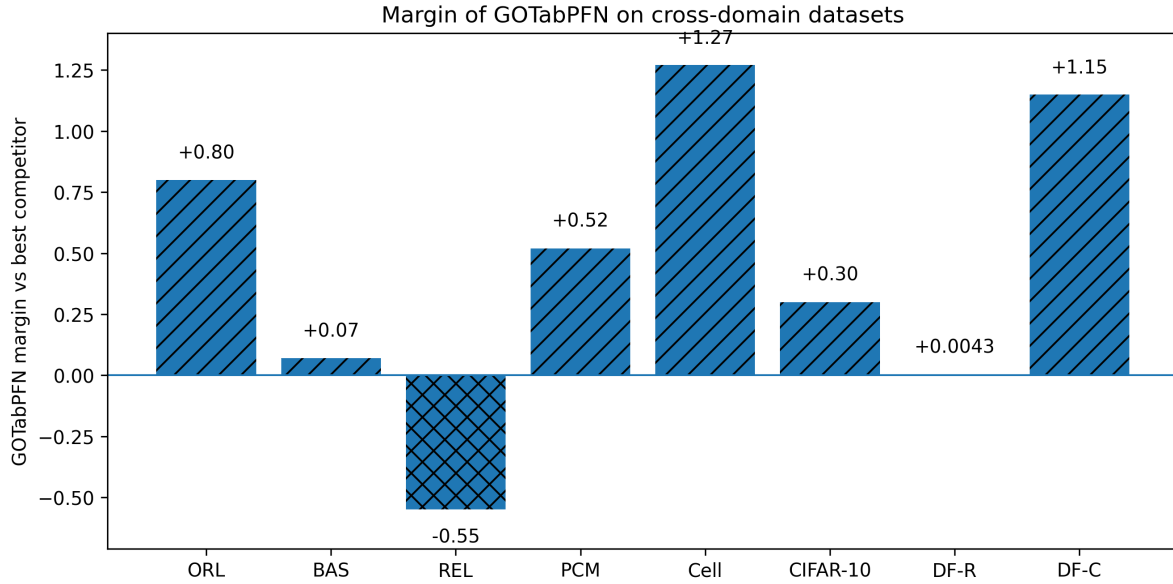


Figure T.1. Signed margin of GOTabPFN on cross-domain datasets. Positive bars show GOTabPFN’s margin over the runner-up when it ranks first; the negative bar shows how far it trails the best baseline when it does not rank first. GOTabPFN ranks first on 7/8 additional datasets, with especially strong margins on ORL, Cell Cycle, and DrivFace-C, and remains competitive on RELATHE.

Table T.2. List of 55 baseline models and their source URLs.

Model	Source URL	Model	Source URL
Naive Bayes	https://scikit-learn.org/stable/supervised_learning.html	AutoInt	https://github.com/OpenTabular/DeepTab
KNN	https://scikit-learn.org/stable/supervised_learning.html	TabR	https://github.com/OpenTabular/DeepTab
SVM	https://scikit-learn.org/stable/supervised_learning.html	ProtoGate	https://github.com/SilenceX12138/ProtoGate
Decision Tree	https://scikit-learn.org/stable/supervised_learning.html	LSPIN	https://github.com/jcyang34/lspin
Lasso	https://scikit-learn.org/stable/supervised_learning.html	LLSPIN	https://github.com/jcyang34/lspin
MLP	https://scikit-learn.org/stable/supervised_learning.html	INVASE	https://github.com/vanderschaarlab/INVASE
1-D CNN	https://github.com/harryjdavies/Python1D_CNNs	L2X	https://github.com/Jianbo-Lab/L2X
Random Forest	https://scikit-learn.org/stable/supervised_learning.html	Mambular	https://github.com/OpenTabular/DeepTab
AdaBoost	https://scikit-learn.org/stable/supervised_learning.html	DANets	https://github.com/manujosephv/pytorch_tabular
GBM	https://scikit-learn.org/stable/supervised_learning.html	STG	https://github.com/runopti/stg
LGBM	https://github.com/microsoft/LightGBM	REAL-X	https://github.com/rajesh-lab/realx
XGBoost	https://github.com/dmlc/xgboost	TabM	https://github.com/OpenTabular/DeepTab
CatBoost	https://github.com/catboost/catboost	ModernNCA	https://github.com/OpenTabular/DeepTab
TabNet	https://github.com/dreamquark-ai/tabnet	Trompt	https://github.com/OpenTabular/DeepTab
TabTransformer	https://github.com/lucidrains/tab-transformer-pytorch	TabulaRNN	https://github.com/OpenTabular/DeepTab
FT-Transformer	https://github.com/lucidrains/tab-transformer-pytorch	MambAttention	https://github.com/OpenTabular/DeepTab
TabSeq	https://github.com/zadid6pretam/TabSeq	MambaTab	https://github.com/OpenTabular/DeepTab
TANGOS	https://github.com/OpenTabular/DeepTab	NDTF	https://github.com/OpenTabular/DeepTab
NODE	https://github.com/OpenTabular/DeepTab	ENODE	https://github.com/OpenTabular/DeepTab
SAINT	https://github.com/OpenTabular/DeepTab	ResNetTabular	https://github.com/OpenTabular/DeepTab
DeepFM	https://github.com/shenweichen/DeepCTR-Torch	CategoryEmbedding	https://github.com/manujosephv/pytorch_tabular
DCN	https://github.com/shenweichen/DeepCTR-Torch	TANDEM	https://github.com/erelnaor3/tandem
TabICL	https://github.com/soda-inria/tabicl	LoCalPFN	https://github.com/layer6ai-labs/LoCalPFN
BETA	https://github.com/LAMDA-Tabular/BETA	TuneTables	https://github.com/penfever/TuneTables
TabPFN-Wide	https://github.com/pfeiferAI/TabPFN-Wide	RealMLP	https://github.com/dholzmueller/pytabkit
MLP-PLR	https://github.com/dholzmueller/pytabkit	TabDPT	https://github.com/layer6ai-labs/TabDPT-inference
TabPFN v1	https://github.com/PriorLabs/TabPFN	TabPFN v2	https://github.com/PriorLabs/TabPFN
TabPFN-2.5	https://github.com/PriorLabs/TabPFN		