VIP-COP: BLACK-BOX CONTEXT OPTIMIZATION FOR TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) have emerged as a powerful paradigm for in-context learning on structured data, enabling direct prediction on new tabular tasks without task-specific training. However, their effectiveness is constrained by context length limits, restricting application to medium-scale data and degrading performance when inference-time data exceed pretraining (size) distributions. Our work introduces VIP-COP, estimating the Value of Importance for Prediction of training examples and features for hard Context OPtimization for TFMs. Its explicit selection mechanism suppresses noise and isolates influential data, enabling the model to also benefit from data augmentation by prioritizing high-value augmented samples and features. VIP-COP is (i) fast, boosting performance often within minutes of optimization, based on an online KernelSHAP based regression with iterative refinement, value-guided context sampling, and multi-fidelity pruning; (ii) budget-aware and any-time, improving with additional test-time compute unlike heuristics that produce fixed context; and (iii) model-aware yet fully black-box, requiring no access to model internals, making it compatible with both proprietary and open-source TFMs; (iv) interpretable, identifying discrete "Very Important Predictors" (samples and features) that maximize signal-to-noise; which makes it (v) robust, isolating high-value data from noise. In contrast,soft-prompt optimization requires model gradients, produce abstract latent tokens,and lack explicit signal discrimination. Extensive experiments show that VIP-COP consistently outperforms heuristic and optimized baselines across large-scale high-dimensional testbeds, including data augmentation and data noise settings, establishing a new state of the art in test-time context refinement for TFMs.

1 INTRODUCTION

Machine learning (ML) has undergone a remarkable transformation with the advent of new model architectures and the rise of generative and foundation models that learn from massive datasets, exhibiting unprecedented generalization and in-context learning capabilities (Xie et al., 2021; Falck et al., 2024). This paradigm shift has permeated nearly all branches of AI, extending to tabular ML, where traditional methods are challenged by emerging Tabular Foundation Models (TFMs).

Pretrained TFMs aim to address classical problems *in context* (Nagler, 2023), including classification, regression, outlier detection, and density estimation (Hollmann et al., 2023; 2025; Qu et al., 2025; Zhang et al., 2025; Shen et al., 2025; Ding et al., 2026b), where the training data is ingested as context samples and estimations on test data is output through a mere forward pass—without requiring any ML model training or model selection (i.e. hyperparameter tuning).

This shift creates an important question: *which training examples and features should a TFM actually see?* This question arises naturally when the training data exceed the model's pretrained context capacity, but it remains important even when all available data fit within the context. Real-world datasets often contain irrelevant, noisy, corrupted, or redundant samples and features, making the full available context not necessarily optimal. Moreover, data augmentation, commonly used in practice to introduce additional variation, can inadvertently introduce further noise or uninformative examples (Yoo et al., 2023). We refer to selecting a discrete subset of training examples and/or features for a frozen TFM as *hard context optimization* (HardCOp).

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

More broadly, this challenge falls under prompt engineering or context optimization (COp) (Sahoo et al., 2024; Chen et al., 2025; Yao et al., 2023; Mei et al., 2025). In NLP, COp typically focuses on prompts, retrieving relevant documents, or selecting few-shot examples to elicit desired behaviors. In the tabular domain, the "instructions" are implicit in the data. Heuristic approaches to TFM context optimization aim to construct representative subsets or sketches of the data (Xu et al., 2025; Ye et al., 2025; Feuer et al., 2023), which however, optimize *model-agnostic* objectives. In contrast, soft prompt optimization (SoftCOp) (Ma et al., 2024b; Feuer et al., 2024) learn abstract context tokens that are optimized end-to-end to maximize performance over the full training set.

In this work, we introduce VIP-COP[1], a fast, any-time algorithm that identifies Very Important Predictors; i.e., the most influential samples and features in the input. The key insight is that the value of a context item is context-dependent: an item may be useful in one context but redundant in another. We therefore estimate item contributions using an additive surrogate inspired by KernelSHAP (Shapley et al., 1953; Lundberg & Lee, 2017) and iteratively importance-sample candidate contexts to refine these estimates. Our main contributions are as follows:

- **Black-box hard context optimization.** We formulate TFM context selection as a discrete optimization problem and introduce VIP-COP, which optimizes explicit samples and features *without modifying or accessing the internals of the pretrained backbone TFM*.
- **Value-guided, anytime and any-size selection.** VIP-COP estimates context-dependent item importance through an additive surrogate and importance sampling, producing a global ranking that supports flexible optimization budgets and context sizes.
- **Robust and efficient context refinement.** Extensive experiments show consistent gains under original, augmented, and noisy contexts, on classification and outlier detection tasks with a number of SOTA foundation models as backbone.
- **Generalization and interpretability.** We evaluate VIP-COP across TFM backbones, including TabPFN (Hollmann et al., 2023), TabICLv2 (Qu et al., 2026), and Outformer (Ding et al., 2026b), and show that its sample/feature rankings provide a practical mechanism for context interpretation and data selection.

2 PRELIMINARIES

2.1 TABULAR FOUNDATION MODELS (TFMS)

TFMs extend the foundation model paradigm to structured, tabular data by pretraining on synthetic tasks drawn from prior distributions over data-generating mechanisms. At inference time, a frozen TFM $q_{\boldsymbol{\theta}}$ takes a labeled training set $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ as *context* and predicts labels for unlabeled *query* points in a single forward pass, e.g., $q_{\boldsymbol{\theta}}(y \mid \mathbf{x}, \mathcal{D}_{\text{train}})$, without task-specific parameter optimization (Hollmann et al., 2023). We use $\mathcal{D}_{\text{val}}$ as a labeled query set for evaluating and selecting contexts, and $\mathcal{D}_{\text{test}}$ as the held-out test set for final evaluation. Thus, a candidate context $C \subseteq \mathcal{D}_{\text{train}}$ contains a subset of context samples and features and is shared across query points. TabICL and TabICLv2 (Qu et al., 2025; 2026) scale this mechanism to larger sample contexts via induced self-attention, while TabPFN-Wide (Kolberg et al., 2025) scales to more features through feature partitioning and latent representations. OutFormer (Ding et al., 2026b) extends the paradigm to outlier detection, where inlier examples form the context and query points are scored relative to the learned inlier structure. Despite these differences, TFM predictions depend directly on the samples and features included in the context, making context construction a natural target for optimization.

2.2 PROBLEM STATEMENT AND SETTINGS

Context limits: Although Transformers can process arbitrary numbers of tokens architecturally, TFMs are pretrained on datasets within finite ranges of sample and feature sizes. Using substantially larger contexts at inference therefore constitutes a distribution shift and can degrade generalization. Context optimization thus serves two purposes: **(1) distillation:** reducing oversized datasets to pretrained limit, and **(2) denoising and efficiency:** removing irrelevant/misleading samples or features even when the data fit within the context. This not only improves the signal-to-noise ratio but also reduces the quadratic computational overhead of self-attention.

[1] VIP-COP is hosted at `https://anonymous.4open.science/r/VIP-COp` for open-API TFMs.

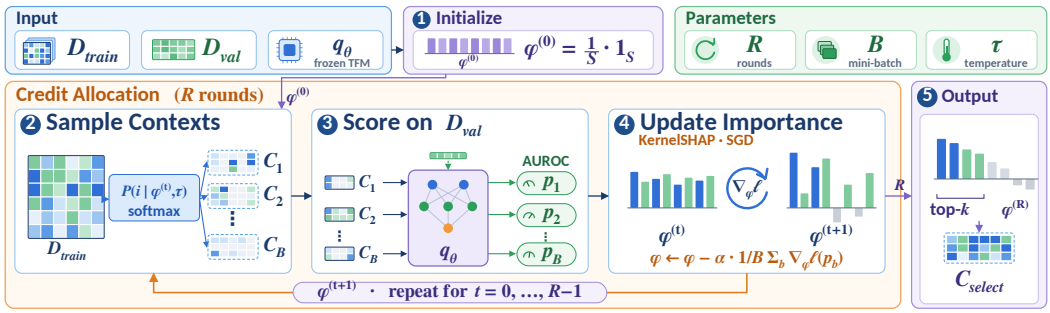


Figure 1: VIP-COP iteratively samples contexts from $\mathcal{D}_{\text{train}}$ via softmax(ϕ/τ), scores them on $\mathcal{D}_{\text{val}}$ with the frozen TFM, and updates importance values ϕ by KernelSHAP-style SGD; after R rounds, the top-k items by ϕ form C_{select}.

Problem Statement 1 (Hard Context Optimization (HardCOp)). *Given a frozen TFM q_θ with context limits (n_C, d_C) and $(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}})$, select a global subset of $n' \leq n_C$ samples and $d' \leq d_C$ features from $\mathcal{D}_{\text{train}}$ to maximize validation performance and generalize to $\mathcal{D}_{\text{test}}$.*

We optimize the context once and reuse it for all test queries, preserving efficient batched inference. We consider two settings, where same selection mechanism can both **exploit useful additional data** and **filter noisy or redundant information**.

Problem Statement 2 (HardCOp under Data Augmentation (DA) & Data Noise (DN)). ***HardCOp4DA:*** *$\mathcal{D}_{\text{train}}$ is augmented with additional samples and/or features, potentially exceeding (n_C, d_C), and HardCOp selects most informative subsets.* ***HardCOp4DN:*** *$\mathcal{D}_{\text{train}}$ contains noisy samples and/or irrelevant features; HardCOp selects subsets that suppress harmful information.*

2.3 Related Work

Context Optimization for Tabular Foundation Models (TFMs): Context optimization refines TFM inputs to improve predictions, analogous to prompt optimization for LLMs (Sahoo et al., 2024). Soft-prompt methods such as ICD (Ma et al., 2024b) and TuneTables (Feuer et al., 2024) compress training data into learned pseudo-points, but require white-box, gradient-based access; VIP-COP instead performs discrete, black-box selection over the original samples and features. LoCalPFN (Thomas et al., 2024) and TabDPT (Ma et al., 2024a) retrieve nearest-neighbor contexts per query, while Xu *et al.* (Xu et al., 2025) route queries to learned local subsets. Unlike our global context, such query-specific adaptation inhibits efficient batching. Other methods use model-agnostic sketching or dimensionality reduction (Feuer et al., 2023), or ensemble predictions over bootstrapped contexts (Liu & Ye, 2025), rather than explicitly optimizing a single context for the TFM. Closest to ours, Rundel *et al.* (Rundel et al., 2024) estimate sample and feature contributions using cached attention keys and values and prune low-value elements; unlike VIP-COP, this requires access to model internals.

Scaling TFMs: Complementary work enlarges TFM capacity through architectural changes. TabICL (Qu et al., 2025) uses Induced Self-Attention (Lee et al., 2019), reducing sample-wise attention from $O(N^2)$ to $O(NM)$ using M inducing/router points (Shen et al., 2025), while TabPFN-Wide (Kolberg et al., 2025) partitions features into blocks, reducing feature-wise attention from $O(P^2)$ to $O(PB)$. In contrast, VIP-COP improves the context *within* a fixed pretrained TFM rather than modifying its architecture.

Long-context Adaptation: Long-context LLM methods similarly expand context capacity through position interpolation (Chen et al., 2023a) or additional adaptation, as in LongLoRA and YaRN (Chen et al., 2023b; Peng et al., 2023). Unlike these approaches, VIP-COP requires neither architectural modification nor model fine-tuning or internal access.

3 VIP-COP: Very Important Predictors for Context Optimization

Given as input (i) a labeled training set $\mathcal{D}_{\text{train}}$ with n examples and d features, (ii) a (small) labeled validation set $\mathcal{D}_{\text{val}}$, and (iii) a checkpoint q_θ of a TFM with a context capacity of n_C examples and

Algorithm 1 VIP-COP: ***Hard Context Optimization for Black-Box Tabular Foundation Models***

Given: $\mathcal{D}_{\text{train}}$, (n, d)=size($\mathcal{D}_{\text{train}}$), $\mathcal{D}_{\text{val}}$, TFM check-point q_θ, (n_C, d_C)=Context_size(q_θ);

1: **User-specified Input:** budget rounds R, η (default $\eta = 2$), mini-batch size B

2: **Initialization:** $r_{\max} = \lfloor \log_\eta(R) \rfloor$ {*#parallel runs*}, $S = n \times \mathbb{I}[n > n_C] + d \times \mathbb{I}[d > d_C]$, $\boldsymbol{\phi}^{(0)} = \frac{1}{S}\mathbf{1}_S$ {*init. all items' (examples and/or features) values*}

3: **for** $r \in 2 \times \{r_{\max}, r_{\max}-1, \ldots, 0\}$ {*parallel runs w/ varying temperature*} **do**

4: temperature $\tau = \eta^{r-r_{\max}}$

5: **for** $t \in \{0, 1, \ldots, R-1\}$ **do**

6: $P(i \mid \boldsymbol{\phi}^{(t)}, \tau) = \frac{\exp\left(\phi_i^{(t)}/\tau\right)}{\sum_{j=1}^{S}\exp\left(\phi_j^{(t)}/\tau\right)} \;\; \forall i = 1 \ldots S$

7: **for** $b = 1 \ldots B$ {*parallel subset sampling and scoring*} **do**

8: $\mathcal{C}_b :=$ `sample` (n_C, d_C) (examples, features) from $\mathcal{D}_{\text{train}}$ w/ prob. $P(i \mid \boldsymbol{\phi}, \tau)$

9: $\widehat{\mathbf{y}}_{\text{val},b} :=$ `feed_forward` $q_\theta(\mathcal{D}_{\text{val}} \mid \mathcal{C})$

10: $p_b = \texttt{auroc}(\widehat{\mathbf{y}}_{\text{val},b}, \mathbf{y}_{\text{val}})$

11: $\mathbf{c}_b = \mathbf{1}_{[i \in \mathcal{C}_b]} \in \{0,1\}^S$

12: **end for**

13: $\boldsymbol{\phi}^{(t+1)} \leftarrow \boldsymbol{\phi}^{(t)} - \alpha \frac{1}{B}\sum_{b=1}^{B} \nabla_{\boldsymbol{\phi}} \ell\left(\boldsymbol{\phi}^{(t)}; \mathbf{c}_b, p_b\right)$ {*mini-batch SGD update*}

14: **end for**

15: $\mathcal{C}_r^{\text{select}}$:=top examples/features with the largest positive values $[\boldsymbol{\phi}^{(R)}]_+$

16: $\widehat{p}_{\text{val},r}^{\text{select}} = \sum_{i \in \mathcal{C}_r^{\text{select}}} \boldsymbol{\phi}_i^{(R)}$

17: **end for**

18: **return** context $\mathcal{C}_r^{\text{select}}$ with the largest $\widehat{p}_{\text{val},r}^{\text{select}}$

d_C features, our proposed VIP-COP estimates the Values of Importance for Prediction (VIP) of the training examples as well as the feature dimensions whose cardinality exceeds the context limits. Put differently, VIP-COP selects the examples and/or dimensions that are Very Important Predictors.

Overview: TFMs act as "algorithms-in-a-box," using in-context learning (ICL) to predict query examples from a training context in a single forward pass. Because the implicitly learned prediction rule changes with the context, each sample or feature has a context-dependent contribution. We therefore formulate hard context optimization as a **credit allocation problem** and estimate sample- and feature-level value using Shapley values (Shapley et al., 1953).

Specifically, the **collective value** of a context $\mathcal{C} \in \mathbb{R}^{n_c \times d_c}$, where $\mathcal{C} \subset \mathcal{D}_{\text{train}} \in \mathbb{R}^{n \times d}$ with $n_c \leq n$ and $d_c \leq d$, is its validation performance when used to predict the query set $\mathcal{Q} := \mathcal{D}_{\text{val}}$. Shapley values aggregate an item's context-dependent marginal contributions across all possible subsets into a single value, which we call its VIP.

We approximate these values using Kernel SHAP (Lundberg & Lee, 2017), formulating a linear regression where binary inputs $\mathbf{c}i = \mathbb{I}[i \in \mathcal{C}] \in 0, 1; \forall i$ indicate item inclusion and validation performance $y\text{val}$ is the response.[2] The resulting coefficients approximate VIP and become exact when all possible subsets are enumerated.

Because exhaustive subset enumeration is infeasible, we iteratively refine VIP estimates by alternating between **(1)** updating the regression coefficients (VIP estimates) and **(2)** sampling new subsets according to the current VIP values. Each subset is evaluated on $\mathcal{D}_{\text{val}}$ with a single TFM forward pass, and its score is incorporated via mini-batch SGD. Temperature-guided importance sampling favors potentially high-VIP items while maintaining exploration, enabling efficient online refinement. Algorithm 1 summarizes VIP-COP (See visual overview in Fig. 1):

User-specified Input (Line 1): VIP-COP proceeds for a user-specified budget of R rounds of mini-batch SGD updates with B new importance-sampled subsets each. Our open source implementation takes around one second per round, so R can be thought of as the time limit in seconds. B is chosen based on available memory, where each subset (i.e., context) is drawn and fed-forward through the TFM to acquire validation performance in parallel.

Initialization (Line 2): We optimize the context jointly over training examples and feature dimensions, yielding $S = n + d$ regression coefficients when both exceed context limits, and n or d coefficients otherwise. We initialize the coefficients uniformly by default; more informative initial-

[2]Original KernelSHAP performs a carefully weighted linear regression to accommodate varying size subsets. As we consider a *fixed* size context (with n_C rows and d_C columns), we perform *unweighted* least-squares.

izations include proportional to attention-based scores (Hao et al., 2021), representative examples forming a coreset (Mirzasoleiman et al., 2020) or features highly correlated with the target variable.

Temperature-guided Importance Sampling (Lines 3–4) plays a key role in our framework, as subsets are populated by selecting items according to their current VIP estimates, which may be unreliable in early rounds. If temperature τ is too low, sampling may prematurely concentrate on a few spuriously high-VIP items, limiting exploration and biasing the estimates. Conversely, if τ is too high, sampling explores too broadly, producing high-variance estimates under the finite budget of $R{\times}B$ subsets.

The appropriate temperature depends on the dataset, particularly the distribution of true VIP values and the quality of VIP initialization. Given a user-specified η that controls available resources, we therefore run VIP-COP in parallel with $\lfloor \log_\eta(R) \rfloor{+}1$ different temperature schedules[3] (Lines 3–17), and return the best context across runs based on estimated validation performance (Line 18).

Iterative VIP Refinement (Lines 5–14): Over R rounds (Line 5), we alternate between **(1)** enriching the regression with B new subsets (Lines 7–12) and **(2)** updating the regression coefficients (VIP estimates) (Line 13). New subsets are constructed by importance-sampling items with probability proportional to their temperature-scaled VIP estimates (Line 6). The sampling temperature τ, scheduled in Line 4, controls the trade-off between exploiting high-VIP items and exploring alternatives.

Context Selection (Lines 15–18): The linear regression yields an *additive decomposition* of model performance, where the sum of VIP scores over a context equals its predicted validation performance. After R rounds of SGD updates, we select the training examples and feature dimensions with the largest positive VIP estimates as the final context (Line 15). Their summed VIPs give the estimated validation performance $\widehat{p}$val (Line 16). Finally, we return the context with the largest $\widehat{p}$val across the independent temperature runs (Line 18).

4 EVALUATION

4.1 EXPERIMENT SETUP

TFM Backbone: We evaluate VIP-COP across a deliberately diverse set of state-of-the-art TFMs spanning different context scales and application settings. Specifically, we use **TabPFN-v1** (Hollmann et al., 2023), pretrained with contexts of up to $1{,}024$ samples and 100 features, and **TabICLv2** (Qu et al., 2026), which scales to substantially larger contexts of up to 48K samples and 100 features. We further evaluate **OutFormer** (Ding et al., 2026b) for outlier detection, extending VIP-COP beyond standard classification tasks. Unlike classification, **OutFormer** uses clean inlier samples as context, providing a distinct evaluation setting with different context semantics.

Datasets: For classification tasks, we source (38) multi-class classification datasets with $n{>}1,000$ from the TALENT[4] benchmark (Ye et al., 2025) and organize them into two: (i) **small-d** includes 24 datasets with $d{<}100$, of which 16 have $n{<}50K$ and 8 have $n{>}50K$; and (ii) **Large-d** includes 14 datasets with $d{>}100$. Appdx. A.1 and Table 2 provide further dataset details. For outlier detection tasks, we source (240) outlier detection datasets with $n{>}6000$ from the MacrOData [5] benchmark (specifically, OddBench) (Ding et al., 2026a). Appdx. A.1 and Table 3 provide further dataset details. We use the standard train/val/test splits for all datasets.

Baselines: We compare VIP-COP with both heuristic **(H)** and optimized context-selection **(O)** strategies. All methods are evaluated in the same black-box TFM setting. We briefly describe baselines H1-H3 and O1-O2 as follows, with further details in Appdx. A.3.

Heuristic: While there are many possible model-unaware heuristics, we prioritize common, easy to implement three: **(H1) Random** selects items (training examples and/or features, whichever exceeds context limits) uniformly at random. We report average test performance across 15 random draws. **(H2) Ensemble** (Liu & Ye, 2025) averages the predictions across 20 random contexts. **(H3) XLContext** feeds $\mathcal{D}_{\text{train}}$ fully as context without subsampling, potentially inducing out-of-(pretraining-)distribution context sizes.

[3] This design is inspired by Hyperband (Li et al., 2018), which terminates underperforming jobs at multiple training stages using multi-fidelity time-budget allocation to accommodate varying learning speeds.

[4] `https://github.com/LAMDA-Tabular/TALENT`

[5] `https://huggingface.co/spaces/MacrOData-CMU/MacrOData`

Optimized: **(O1) KMeansReps** fits a n_C-means clustering on the examples and selects the centroids as the context items for sample-level optimization. For feature-level optimization, we instead perform d_C-means clustering on the features and select the centroids as the context features. Validation performance is used to select the best clustering among those trained by 10 randomly initialized centroids. **(O2) DT+TFM** As baselined in (Ye et al., 2025), a shallow decision tree with minimum leaf size $n_{\min}$=n_C is trained to partition $\mathcal{D}_{\text{train}}$ into subsets. Each validation example is routed to its corresponding leaf, where TFM uses the leaf's examples as context to produce the prediction. We modify DT+TFM baseline for feature-level optimization by training a DT with maximum depth $d_{\max}$=50 and selecting up to 100 unique split features level-by-level starting from the root. Validation performance is used to select the best DT among those trained by choosing a random feature at each split from among the top-three as ranked by information gain in both scenarios. *Note: this baseline is not viable under outlier detection settings since only inlier exists in the training data.*

Metrics: We evaluate methods on tabular multi-class classification datasets w.r.t. *balanced* accuracy and relative improvement (%) over baselines; and methods on tabular outlier detection datasets w.r.t. AUROC and improvement (%) over baselines. Besides performance, we use paired permutation tests to assess significance of differences across datasets and report optimization time to quantify the test-time cost of context selection.

4.2 Evaluation Settings

Data Augmentation (DA): We enrich each $\mathcal{D}_{\text{train}}$ with augmented samples and/or features prior to context optimization (Bengio et al., 2017; Ericsson et al., 2022). Specifically, datasets with $n < 50$K and $d < 100$ are augmented to $n = 50$K and $d = 100$; those with $n > 50$K and $d < 100$ are feature-augmented only to $d = 100$; and those with $n < 50$K and $d > 100$ are sample-augmented only to $n = 50$K. For sample augmentation **($\mathbf{S}_{\text{aug}}$)**, inspired by SMOTE (Chawla et al., 2002), we generate $\mathbf{x}_{\text{aug}} = \alpha\mathbf{x}_k + (1-\alpha)\mathbf{x}_l$ from randomly sampled $\mathbf{x}_k, \mathbf{x}_l \in \mathcal{D}_{\text{train}}$ with $\alpha \sim \mathcal{U}(0,1)$, assigning $y_{\text{aug}} = y_k$ if $\alpha \leq 0.5$ and $y_{\text{aug}} = y_l$ otherwise. For feature augmentation **($\mathbf{F}_{\text{aug}}$)**, we apply random projections of the form $\mathbf{X}_{\text{train}} \cdot \mathbf{R}$ to create d_{aug} new features, where $\mathbf{X}_{\text{train}} \in \mathbb{R}^{n \times d}$ and $\mathbf{R} \in \mathbb{R}^{d \times d_{\text{aug}}}$ with $\mathbf{R}_{ij} \sim \mathcal{N}(0, 1/d_{\text{aug}})$. ***Remark:*** augmentation choice is itself a model-selection problem, as different augmentations may be useful or irrelevant depending on the dataset (Cubuk et al., 2019; Zoph et al., 2020; Ottoni et al., 2023). Rather than optimizing augmentation policies, which is beyond our scope, we use standard off-the-shelf augmentations as a proof of concept. Thus, our results provide a conservative estimate of the potential gains from better-designed augmentations, while showing that VIP-COP can mitigate harmful augmentations by filtering uninformative examples.

Data Noise (DN): To stress-test robustness to irrelevant or corrupted context, we perturb the training data while preserving its original size. For small-d datasets, we randomly drop half of the samples and replace them with noisy samples to recover the original n; for large-d datasets, we randomly drop half of the features and replace them with noisy features to recover the original d. We consider two sample-noise strategies: **($\mathbf{S1}_{\text{noi}}$)** samples each feature independently from its empirical marginal, and **($\mathbf{S2}_{\text{noi}}$)** draws samples from a global Gaussian whose mean vector and covariance matrix are estimated from the original data. Both assign labels sampled from the empirical label distribution. For feature noise, **($\mathbf{F1}_{\text{noi}}$)** copies a randomly chosen existing feature and adds Gaussian noise with three times its original variance, while **($\mathbf{F2}_{\text{noi}}$)** randomly permutes the values of an existing feature across samples. ***Remark:*** DN is designed as a controlled robustness stress test rather than a dataset-specific model of realistic corruption: constructing realistic noise would require knowledge of each dataset's underlying data-generating process, which is impractical at our study scale. Importantly, most baselines degrade substantially under DN, indicating that the setting is sufficiently challenging, whereas VIP-COP remains comparatively robust by filtering irrelevant or corrupted context.

Outlier Detection (OD): To evaluate whether VIP-COP generalizes beyond standard supervised classification, we consider OutFormer (Ding et al., 2026b), a TFM designed for semi-supervised outlier detection with context size 5K including the context and the query/test sets in up to 100 features. It is pretrained with clean context consisting of inlier examples only, while test points are evaluated for anomalousness relative to the learned inlier structure.

Corruption Sensitivity (CS): We hypothesize that real inlier pools may contain outliers mislabeled as inliers. Since such label errors are unobservable, we mimic them by replacing randomly selected inliers with real, held-out outliers and treating them as inliers, creating data-derived contamination. We retain 83 OddBench datasets with at least 6K inliers and 1,200 labeled outliers, and construct

disjoint 6K-point context pools, 1K query sets, and 5K test sets. We replace $\varepsilon n_{\text{pool}}$ of randomly selected inlier points with outliers absent from both evaluation splits, using $\varepsilon \in \{0, 0.05, 0.1, 0.2\}$. We then run VIP-COP with true query labels for 100 rounds of 2,000 coalitions of size 500 and retain the top $k = 3{,}000$ points by ϕ. Keeping the pool and evaluation sets fixed across ε isolates the effect of contamination. This setting creates more organic noise than the Data Noise (DN) setting.

4.3 MAIN RESULTS

Originial Classification: With the TabPFN-v1 as backbone and original classification tasks, VIP-COP consistently improves over the evaluated context-selection baselines across the 38 datasets providing 1.54% to 14.68% performance improvement; full per-dataset results available in Table 4 of Appdx. B . Further evaluation on the larger-context TabICLv2 backbone demonstrates that VIP-COP outperforms all five baselines (see Table 5 in Appdx. B for full results). Paired statistical tests in Fig. 2 and Table 1 confirm this finding, showing that VIP-COP achieves significant gains over every baseline (p<0.01). Although XLContext (H3) and DT+FM (O2) emerge as effective heuristics with similar performance to one another (p=0.344), both ultimately remain inferior to VIP-COP. Similar results on TabPFNv1 are in Appdx. Table 13 and Fig. 9.

Data Augmentation (HardCOp4DA): For TabPFNv1 (see details in Appdx. B, Table 7 and Table 8), we report individual performances under sample and feature augmentation, respectively. Across the HardCOp4DA experiments, VIP-COP improves over the baselines by 4.80% to 15.05%. Table 13 shows that VIP-COP outperforms all baselines across the data augmentation settings except one, where the difference between KMeansReps and VIP-COP is not statistically significant ($p = 0.277$).

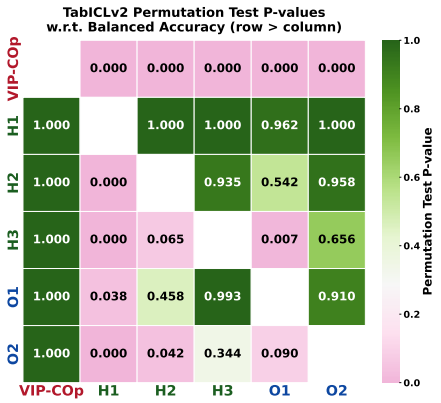


Figure 2: Paired comparisons between all context optimization methods based on p-values of permutation tests (TabICLv2 as backbone). TabPFNv1 results in Fig. 9.

For TabICLv2 (see details in Appdx. B, Tables 6), VIP-COP improves over the baselines by approximately 1.9%–5.8%, with p<0.01 (see Table 1). These results indicate that explicit context selection can exploit an expanded context without assuming that every augmented item is informative. Tables 7 and 8 show that VIP-COP benefits particularly from sample augmentation as compared to feature augmentation, which motivate further research into data augmentation strategies for TFMs.

Data Noise (HardCOp4DN): We next introduce controlled sample- and feature-level noise to reduce the signal-to-noise ratio of the context. For TabPFNv1, (see details in Appdx. B and full results in Table 9, Table 10, and Table 12) reports detailed results under the three data noising schemes. VIP-COP yields consistent improvements over the baselines, ranging from 1.92% to 27.38% under sample noise and from 154.11% to 240.81% under feature noise.

Table 1: Pairwise comparison between each baseline and VIP-COP (TabICLv2 as backbone) based on p-values of permutation tests over datasets across all HardCOp settings (TabPFNv1 results in Appdx. 13).

	Rand **(H1)**	Ens **(H2)**	XL **(H3)**	KRep **(O1)**	DT+FM **(O2)**
Original	<0.01	<0.01	<0.01	<0.01	<0.01
DA w/ S_{aug}	<1e-3	<1e-3	<1e-3	<1e-3	<1e-3
DN w/ S_{noi}	<1e-6	<1e-6	<1e-6	<1e-6	<1e-6
DN w/ F_{noi}	<0.01	<0.01	<0.01	<0.01	<0.01
Overall	**<1e-6**	**<1e-6**	**<1e-6**	**<1e-6**	**<1e-6**

Across the 24 TabICLv2 datasets, VIP-COP improves over the baselines by approximately 7.0–8.7%, with all aggregate comparisons significant at $p < 0.001$. This supports the use of explicit selection to suppress samples and features that are irrelevant or harmful to prediction. Table 9, Table 10, and Table 12 indicate that a lower signal-to-noise ratio degrades the performance of all methods, with the baselines experiencing more pronounced deterioration, particularly under feature noise.

Outlier Detection: Beyond classification, on 240 outlier detection tasks from Oddbench using the Outformer (Ding et al., 2026b) backbone, VIP-COP also improves over all the five baselines by approximately 9.13%-23.75% (full per-dataset results available in Appdx. B Table 14). These

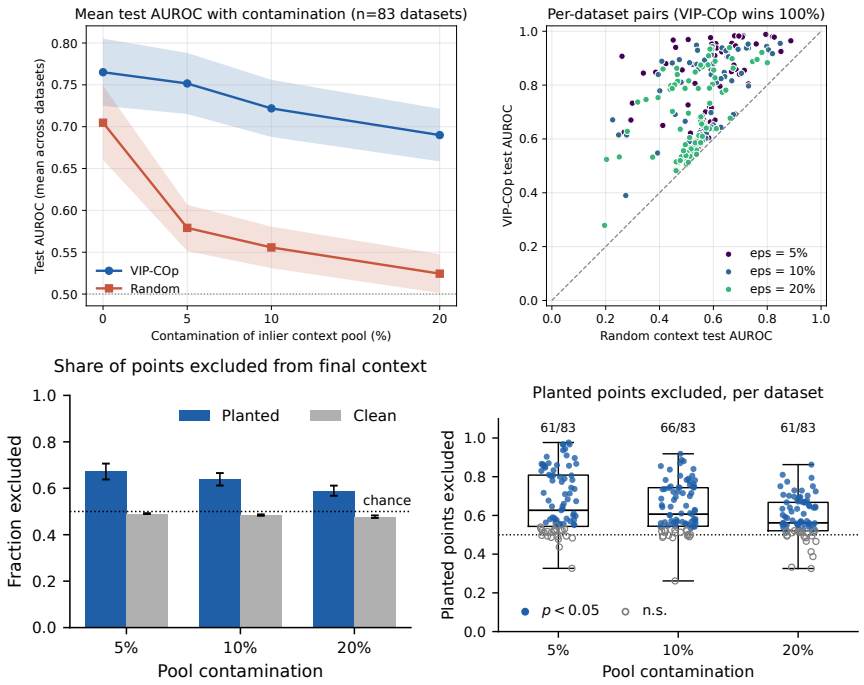


Figure 3: VIP-COP **degrades slower under context-pool corruption and preferentially excludes planted outliers.** Held-out real outliers are planted into the inlier-only context pool of 83 ODD-Bench datasets, disguised as inliers: **(a,b)** a random context collapses toward chance as contamination rises while VIP-COP largely holds, winning on every dataset at every rate; **(c,d)** clean points are excluded from the final context at the chance rate while planted points are excluded well above it, though the enrichment weakens as contamination grows.

results demonstrate that context optimization extends to TFMs with substantially different objectives and inference settings, including semi-supervised outlier detection.

Robustness to Context Contamination: We further test whether VIP-COP remains effective when the context pool is contaminated with outliers. Across 83 OddBench datasets, we replace $\epsilon \in 5\%, 10\%, 20\%$ of the 6,000-point context pool with held-out outliers, from which VIP-COP selects 3,000 points. As shown in Fig. 3a, random-context AUROC drops from 0.705 on the clean pool to 0.579 and 0.525 at 5% and 20% contamination, whereas VIP-COP declines from 0.765 to 0.752, 0.722, and 0.690, widening its advantage from 0.06 to 0.17 AUROC (Fig. 3b). VIP-COP also excludes 67%, 64%, and 59% of planted outliers, compared with ∼49% of clean points (shown in Fig. 3c); this filtering is significant ($p < 0.05$) on 61, 66, and 61 datasets, respectively (Fig. 3d).

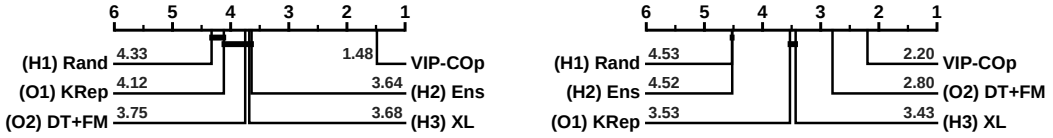
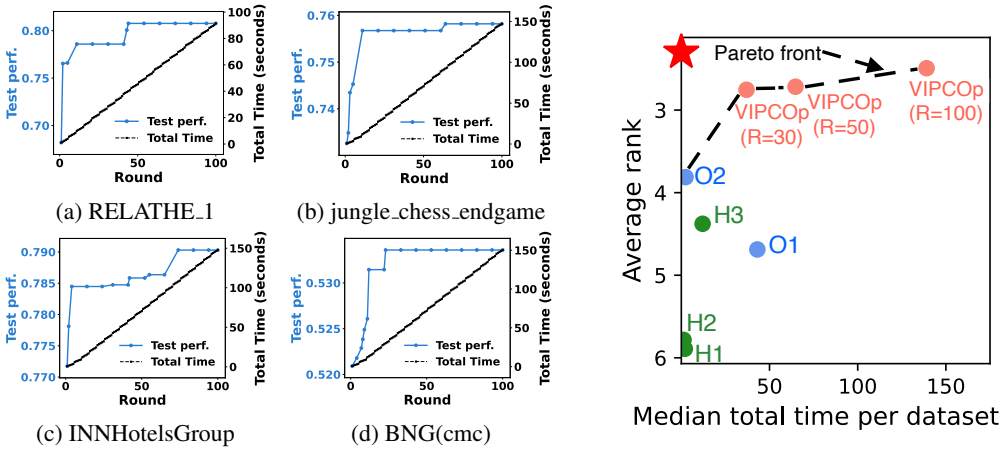


Figure 4: CD diagrams comparing methods across all settings with the TabICLv2 (left) and TabPFNv1 (right). With both backbones, VIP-COP achieves the best average rank and significantly outperforms all baselines. The baseline ordering depends on the backbone: the baselines are largely indistinguishable with TabICLv2. But VIP-COP outperforms with both backbones.

Summary of Results: Fig. 4 reveals a consistent advantage for VIP-COP across diverse context-selection settings and model backbones. Despite substantial variation in the relative performance of the baselines across backbones, VIP-COP **consistently remains the top-ranked method, indicating that its gains are not tied to a particular TFM architecture.**

4.4 Ablation Study and Efficiency Analysis

Compress Beyond the Context Limit: We examine the context size–performance trade-off by truncating VIP-COP's global data-point ranking after optimization. Using TabICLv2, we compare



(a) RELATHE_1 (b) jungle_chess_endgame

(c) INNHotelsGroup (d) BNG(cmc)

Figure 5: (left) VIP-COP **running time** (black) *increases linearly* with optimization rounds (≈1.4s per round), while **test performance** (blue) *improves monotonically*. (right) VIP-COP achieves the best **performance-time trade-off**, where its anytime variants occupy the Pareto frontier.

ranking prefixes with a context containing all samples with positive estimated coefficients, up to 48K samples. The top 24K samples reduce performance by only 0.51%, while the top 5K (about 10% of the context limit) incur only a 1.80% reduction, showing that much of the predictive signal can be retained with substantially smaller contexts without rerunning optimization.

Interpreting VIP Rankings - A Case Study: The VIP ranking provides an interpretable view of which samples and features contribute to TFM predictions. To illustrate this capability, we conduct a feature-selection case study on the UCI Communities and Crime dataset containing features for predicting violent crime. After converting the continuous target into a balanced binary classification task, we run VIP-COP with sample selection disabled and optimize feature selection for 100 rounds using TabICLv2. The resulting ranking closely agrees with the dataset creator's univariate Pearson-correlation analysis: the top four VIP features exactly match the four features with the largest absolute correlations, and 9 of the creator's top 10 univariate predictors appear in the VIP-COp top 20. VIP also distinguishes among correlated features, capturing context-dependent contributions beyond marginal correlation. Full results are in Appdx. C.

Running Time, Optimization Efficiency, and Performance: Context optimization adds test-time compute and latency, therefore it is important for VIP-COP to provide useful gains within a practical budget. Fig. 5 (left) analyzes VIP-COP's performance over increasing time budget. Its running time increases *linearly* with optimization rounds, averaging approximately 1.4, 2.5, and 1.7 seconds per round for TabPFN v1, TabICLv2, and Outformer, respectively, while test performance *improves monotonically*. Thus, VIP-COP is an *anytime* procedure that can trade optimization budget for performance. For TabICLv2, 100 rounds take approximately 4 minutes, while 58 of 64 dataset/setting combinations reach their best observed context by round 75, including 22 within the first 25 rounds (see Appdx. Table 15 and Fig. 10). Finally, Fig. 5 (right) illustrates the performance–time trade-off for all methods, showing average rank (lower is better) versus median running time (lower is better) across all datasets. VIP-COP and its anytime variants, respectively at rounds 30, 50 and 100, achieve the best performance-runtime trade-off, effectively defining the Pareto frontier.

5 Conclusion

Tabular foundation models (TFMs) offer zero-shot classification, regression, and outlier detection in a single forward pass; however, they trade dataset-specific training for careful context design. We address this central challenge with VIP-COP, a principled and scalable approach that identifies the most influential samples and features via black-box, model-aware value attribution. Across diverse settings, VIP-COP consistently distills oversized, noisy, or uninformative candidate pools into compact, high-signal contexts, yielding robust performance gains under practical test-time budgets. Its anytime nature further enables flexible trade-offs between accuracy and latency. By unifying effectiveness, efficiency, and robustness, VIP-COP elevates context optimization from a heuristic practice to a core component of TFMs, enabling reliable in-context learning on structured data.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing and to draft sections of the paper. We did not use generative AI tools for retrieval or discovery, research ideation, experimental design or execution, data generation, or generating synthetic datasets. All AI-assisted content was reviewed and verified by the authors. In particular, AI-assisted text was checked for factual accuracy, consistency, and appropriate attribution, and any AI-assisted code was manually reviewed and tested by the authors for correctness before use. The authors take full responsibility for the final content of this work, including all claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the complete algorithmic details of VIP-COP, including the optimization procedure, VIP estimation, temperature-guided importance sampling, and context selection, at this link: `https://anonymous.4open.science/r/VIP-COp`.

REFERENCES

Yoshua Bengio, Ian Goodfellow, Aaron Courville, et al. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. ISSN 1076-9757. doi: 10.1613/jair.953. URL http://dx.doi.org/10.1613/jair.953.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering for large language models. *Patterns*, 6(6), 2025.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023a.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023b.

Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019.

Xueying Ding, Simon Klüttermann, Haomin Wen, Yilong Chen, and Leman Akoglu. ¡scp¿macrodata:¡/scp¿ new benchmarks of thousands of datasets for tabular outlier detection. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 8777–8788. ACM, August 2026a. doi: 10.1145/3770855.3817520. URL http://dx.doi.org/10.1145/3770855.3817520.

Xueying Ding, Haomin Wen, Simon Klütterman, and Leman Akoglu. From zero to hero: Advancing zero-shot foundation models for tabular outlier detection. In *International Conference on Machine Learning*. PMLR, 2026b.

Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. Why do self-supervised models transfer? on the impact of invariance on downstream tasks. In *BMVC*, pp. 509. BMVA Press, 2022.

Fabian Falck, Ziyu Wang, and Christopher C. Holmes. Is in-context learning in large language models bayesian? a martingale perspective. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 12784–12805. PMLR, 2024.

Benjamin Feuer, Niv Cohen, and Chinmay Hegde. Scaling TabPFN: Sketching and feature selection for tabular prior-data fitted networks. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

Benjamin Feuer, Robin T Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12963–12971, 2021.

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

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Christopher Kolberg, Katharina Eggensperger, and Nico Pfeifer. Tabpfn-wide: Continued pre-training for extreme feature counts. *arXiv preprint arXiv:2510.06162*, 2025.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52, 2018.

Si-Yang Liu and Han-Jia Ye. Tabpfn unleashed: A scalable and effective solution to tabular classification problems. *arXiv preprint arXiv:2502.02527*, 2025.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L Caterini. TabDPT: Scaling tabular foundation models. *arXiv preprint arXiv:2410.18164*, 2024a.

Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with tabpfn. *arXiv preprint arXiv:2402.06971*, 2024b.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Thomas Nagler. Statistical foundations of prior-data fitted networks. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023.

André Luiz C Ottoni, Raphael M de Amorim, Marcela S Novo, and Dayana B Costa. Tuning of data augmentation hyperparameters in deep learning to building construction image classification with small datasets. *International Journal of Machine Learning and Cybernetics*, 14(1):171–186, 2023.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

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

Michael Redmond. Communities and Crime. UCI Machine Learning Repository, 2002. DOI: https://doi.org/10.24432/C53W3X.

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

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 1, 2024.

Lloyd S Shapley et al. A value for n-person games. 1953.

Yuchen Shen, Haomin Wen, and Leman Akoglu. Fomo-0d: A foundation model for zero-shot tabular outlier detection. *Transactions on Machine Learning Research*, 2025.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony Caterini. Retrieval & fine-tuning for in-context tabular models. *Advances in Neural Information Processing Systems*, 37:108439–108467, 2024.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Derek Qiang Xu, F Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hantao Yao, Rui Zhang, and Changsheng Xu. Visual-language prompt tuning with knowledge-guided context optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6757–6767, 2023.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at tabpfn v2: Understanding its strengths and extending its capabilities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jaemin Yoo, Tiancheng Zhao, and Leman Akoglu. Data augmentation is a hyperparameter: Cherry-picked self-supervision for unsupervised anomaly detection is creating the illusion of success. *Trans. Mach. Learn. Res.*, 2023, 2023.

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

Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *ECCV*, pp. 566–583. Springer, 2020.

APPENDIX

A EXPERIMENT SETUP DETAILS

A.1 CLASSIFICATION DATASETS

We sourced classification datasets from the TALENT and TALENT-EXTENSION benchmarks Ye et al. (2025). For *small-d* datasets ($d \leq 100$), we included all publicly available datasets within the benchmark whose training size exceeded 1,000 samples. For high-dimensional datasets ($d > 100$), we included all publicly available datasets with more than 1,000 samples. We note that TALENT-EXTENSION MANY-CLASS benchmark is excluded as the corresponding data sources were not publicly accessible at the time of experimentation Details in Table 2.

Table 2: Dataset Summary Statistics; categorized by $\mathcal{D}_{\text{train}}$ sample size n and dimensions d (count in parentheses). ID refers to the identifier within the TALENT benchmarks Ye et al. (2025).

ID	Name (Source)	#classes	n	d
	small-d (16)			
19	Amazon_employee_access	2	26,215	7
145	INNHotelsGroup	2	29,020	17
41	BNG(breast-w)	2	31,492	9
47	BNG(tic-tac-toe)	2	31,492	9
55	Click_prediction	2	31,958	3
109	FOREX_audcad-hour-High	2	35,060	10
112	FOREX_audjpy-hour-High	2	35,060	10
113	FOREX_audsgd-hour-High	2	35,060	10
114	FOREX_audusd-hour-High	2	35,060	10
116	FOREX_cadjpy-hour-High	2	35,060	10
155	jungle_chess_endgame	3	35,855	6
30	bank	2	36,168	16
93	electricity	2	36,249	8
188	mobile_c36_oversampling	2	41,408	6
42	BNG(cmc)	3	44,236	9
248	shuttle	7	46,400	9
	small-d (n>50K) (8)			
147	internet_firewall	4	52,425	7
52	Cardiovascular-Disease	2	56,000	11
77	Credit_c	3	80,000	22
239	SDSS17	3	80,000	12
80	diabetes_us_hospitals	2	81,412	20
15	airline_satisfaction	2	103,904	21
229	Rain_in_Australia	3	116,368	18
8	accelerometer	4	122,403	4
	Large-d (14)			
15 [VLS]	nomao	2	27,572	118
121	gas-drift	6	8,902	128
20 [VLS]	volkert	10	46,648	180
177	mfeat-factors	10	1,600	216
144	Indian_pines	8	7,315	220
181	mfeat-pixel	10	1,600	240
171	madeline	2	2,512	259
8 [VLS]	fabert	7	6,589	800
10 [VLS]	gina_agnostic	2	2,774	970
7 [VLS]	dilbert	5	8,000	2,000
12 [HD]	PCMAC_1	2	1,554	3,289
14 [HD]	RELATHE_1	2	1,141	4,322
3 [HD]	BASEHOCK_1	2	1,594	4,862
6	gisette_1	2	5,600	5,000

A.2 Outlier Detection Datasets

We sourced all outlier detection datasets from the MacroData OddBench Ding et al. (2026a) that are publicly available whose training size exceeded 6,000 samples.

Table 3: Dataset Summary Statistics; Datasets in Alphabetical order Ding et al. (2026a)

Name (Source)	n_{rows}	n_{features}	Name (Source)	n_{rows}	n_{features}
AbandonedMineLand	13183	5	AdPerformance	20426	3
AdRequestFraud	25150	77	AgilePriority	17166	5
AgriculturalYield	39448	7	AircraftSensorAnomalies	173935	19

Continued on next page

Table 3 – continued from previous page

Name (Source)	n_{rows}	n_{features}	Name (Source)	n_{rows}	n_{features}
AmbulanceDeployment	133267	7	AnomalyClassification	262144	4
ArtCollections	19282	3	ArtSalesPlatform	46867	4
ArxivProcessing	93811	3	AstronomicalObservations	136625	11
AutoencoderAnomaly	166776	6	BackupSession	29498	5
BankTransactions	82163	3	BankruptcyStatus	78682	19
BarExamPrediction	19178	10	BaseballEvents	145585	21
BearingFault	25296	10	BettingIntegrity	16309	5
BikeSharingTemperature	17379	6	BioSequenceAnomaly	24814	5
BiologicalClassifications	26031	13	BirdMoult	19156	12
BirdTracking	17904	3	BlacklistSpam	93113	6
BlockchainTransaction	67601	3	BoliviaUnpaidLabor	16839	64
BondMarketAnomalies	14412	12	BuildingCalls	51010	10
BuildingDamage	15504	15	BuildingInspection	14940	24
BuildingInspections	27619	5	BuildingPoints	124818	8
BusinessFraud	50000	5	BusinessInspections	16103	4
BusinessProcessActivity	118721	4	CampaignOutcome	29149	7
CancerTreatment	124394	4	CandidateVetting	14575	18
CarMakeAnomaly	25997	4	CardFraudDetection	49264	4
CardioCatchBMI	68129	7	CarrefourSalesDefects	39058	8
CasePriority	24510	27	ChecksumValidation	141804	3
ChicagoFoodInspection	46030	11	ChronicDiseaseStats	20349	10
CodeDefects	19337	41	CogLoadReaction	76612	11
ContactTypeMonitoring	234512	5	ConversationAnomalies	82766	6
CoralDisease	213853	8	CountyHealthMetrics	180583	4
CourseEvaluation	419829	4	CourseRegistration	25435	5
CovidEpidemiology	16926	4	CovidTestingData	15315	5
CovtypeAnomaly	286048	10	CreditCardFraud	59367	5
CreditDefault	29019	14	CreditRiskModel	145741	8
CrimeAnomalies	94383	5	CrimeClearanceStatus	60748	13
CrimeFraudDetection	13307	10	CropProduction	33275	9
CustomerCampaignOutcome	21161	6	CustomerOrderPriority	33349	8
CustomerServiceTransactions	88186	4	CybersecurityVulnerability	21045	6
DallasBankGeography	62964	5	DamageFunctionSources	32580	3
DataValidationEntries	38876	70	DeathCauses	27622	32
DeliveryRootCause	79464	4	DentalConditionAnomaly	32046	14
DiabetesSpecialty	48819	14	DiamondClarity	18002	10
DifferentialExpression	104014	6	DisasterIncidentType	56981	4
DoctorVisits	23713	7	DragonFlySystem	22109	4
DurhamVotersAge	28379	4	EcommerceOrderPriority	33359	7
ElectricBuildingUsage	312958	10	ElectronicTesting	75387	12
EmailPhishingDetection	45793	16	EmotionalIncidents	36614	6
EmployeeAttrition	51350	11	EmployeeLocationAnomalies	333952	3
EndangeredSpeciesClassification	173476	3	GPUResourceUsage	21886	4
GeographicSeverity	122604	3	GeologicalLithology	225455	4
GeospatialCrimeResolution	129957	10	GlobalSharkAttacks	19974	3
GlobalTerrorismAttack	33620	5	HealthMetricsAnomalies	96476	8
HealthPulse	44119	28	HealthcareMessages	24378	8
HockeyGameplay	125494	5	HospitalSeverity	109369	9
HousePriority	16849	3	HousingConstruction	21424	9
HuntingCriteria	14486	5	ImageCategoryAnomaly	82273	6
IndustrialEquipment	29418	10	IndustrialFanVibration	440178	6
ItemIncineration	23080	14	JobStatus	38081	19
JobStatusMonitoring	32696	4	KidneyDisease	42941	9
LabTestAbnormality	18225	7	LandParcels	35865	3
LandShapeTrade	73481	8	LanguageAnomaly	17760	14
LanguageLocationAnomaly	368218	3	LargeWoodMeasurement	54331	9
LaunchStatus	54573	3	LegalChargeAnomalies	23571	5
LegalEducation	27472	6	LoanCreditScore	102617	5
LoanFraudDetection	20269	9	LoyaltyFraud	100000	7
MaritimeVessel	229235	8	MaterialFlow	382500	9
MechanicalTesting	28014	6	MedicalClassificationAnomalies	31103	4
MicrostructureDefect	32712	18	MineralComposition	27347	17
MobileMoneyTransactions	15005	7	MojaveDune	14760	17
MouseInteractionPhase	29155	5	MovieGenreAnomalies	20007	4
MusicFamiliarity	22025	5	NestSuccess	30947	12
NetworkMalware	40837	15	NetworkResponseAnomalies	14413	9

Continued on next page

Table 3 – continued from previous page

Name (Source)	n_{rows}	n_{features}	Name (Source)	n_{rows}	n_{features}
NetworkThreat	12993	4	NetworkTrafficDOS	197342	4
NetworkTrafficSpam	181803	4	NewbornReferral	15488	3
NonComplianceInspection	57410	5	OceanicStatus	37752	20
OptimizationError	25481	17	OrderTransmission	31864	3
OzoneLevels	342858	6	PatientReadmission	78887	11
PhotoAnomalyDetection	12952	3	PlayerMatchEvents	335787	9
PokemonTypeAnomaly	117184	8	PoliceFraudIncidents	508947	4
PoliticalInterestMisinformation	23680	9	PopulationComparison	80035	6
PostcodeValidation	15435	9	PrisonBooking	38408	5
ProcessState	18581	5	ProductionRunStatus	43992	7
ProgrammingAnomalies	32618	6	ProjectSeverity	16548	4
PropertyCondition	24649	7	ProteinCodonAnalysis	32057	14
ProtestEvents	24366	4	ProtestViolence	20224	7
PsychTrialAccuracy	55738	5	PsychologicalTrialValidity	16128	5
PsychologyExperiment	23562	11	RealEstateCondition	87400	5
RealEstateProperty	18599	25	RegionalPriceMonitoring	20943	6
ReportResolution	87154	3	ResidueLigandClassification	14498	9
RestaurantViolation	12756	6	RiskVulnerability	35154	3
RoutingAnomaly	66473	6	SalesOrderPriority	33386	7
SalesTaskStatus	14796	15	SatallaxBenchmark	20948	4
SaudiWeatherAnomalies	244536	9	SchedulerJobState	57595	7
SensorimotorAdaption	23322	19	SentencingFraud	15174	53
ServerLoadTest	21831	3	ServiceErrorLocation	16215	4
ShapeTrackingAnomaly	115129	6	ShellfishQuality	100000	5
ShuttleOutlier	41638	9	SierraLeoneWaterPoints	24173	6
SmallBusinessLoans	29151	12	SnowDepthMonitoring	32986	3
SoccerActions	45566	25	SoccerMatchAnomalies	17792	8
SoilFloodProperties	20301	4	SpamDetection	67881	5
SpeciesObservation	49688	8	SpecimenStatus	27296	6
StatisticalTestResults	42601	10	StoreCrowd	844338	3
StormBehavior	18512	6	StudentAssessment	117017	10
StudentEnrollment	56106	4	StudentProgramAnomaly	169579	3
SummaryAnomalies	35050	3	SuperStoreOrderPriority	33339	10
SystemFailure	28719	11	SystemFailureMonitoring	25010	6
SystemLogs	21834	3	SystemProcessAnomalies	23121	3
SystemStatusAnomaly	19789	5	TechnicalAnomalies	20020	8
TelecomMarketing	37471	7	TestCycleAnomalies	56763	3
TVShowAnomalies	107613	3	TicketPriority	30333	7
TorontoPoliceContacts	25802	5	TrafficIncident	56429	5
TrafficStopAnomalies	49787	4	TransactionFeeAnomalies	44688	3
TransactionService	18311	5	TrialConditionAnomaly	126330	3
TurnaroundValidation	34615	3	TwitchStreamAnomalies	131024	20
UserEventAnomalies	20738	3	VehicleNetworkTraffic	24895	59
VenezuelaBeneficiaries	16841	6	VideoGameViolence	19627	3
VolcanicAnomalies	27475	4	VolleyballMatch	26305	38
VoteOutcome	26898	8	WaterMeterAnomaly	19812	5
WaterPointFunctionality	26076	13	WaterQualityAssessment	121722	8
WeatherEvents	25533	7	WikipediaVandalism	29646	3
WindTurbineFault	47389	4	WindmillAnomaly	17220	17
WindowsEditionAnomaly	49384	26	WorkOvertimePatterns	18151	5
YouTubeFaceDetection	55036	5			

A.3 BASELINE CONFIGURATIONS

All baseline experiments are implemented through a unifed TabPFN wrapper interface and performance is evaluated using balanced accuracy. The global seed is fixed to 42 for all operations that involve stochasticity unless otherwise stated. Each method receives the same train/validation/test split and uses identical TabPFN-v1 backbone model. For experiments involving data augmentation (DA) or data noise (DN), the processed datasets and their associated splits are generated once and stored, which are then reused across all baseline methods to ensure an identical evaluation setup. During inference, test examples are evaluated in batches of 2000 samples per forward pass. The max context configuration, denoted (*MAX_CONTEXT*, *MAX_FEATURES*) is set to (*1000*, *100*) for TabPFN-v1.

H1: Random. H1 performs 15 independent random context runs. If $n >$ *MAX_CONTEXT*, each run samples exactly *MAX_CONTEXT* training examples uniformly without replacement; otherwise, all training examples are used towards context. If $d >$ *MAX_FEATURES*, a size *MAX_FEATURES* feature subset is sampled once per run using the fixed seed. The final result is the average of test balanced accuracy across the 15 runs.

H2: Ensemble. H2 performs 20 runs and assembles the predictions. For each run, if $n >$ *MAX_CONTEXT*, one random sample subset of size *MAX_CONTEXT* is selected. Similarly, one random feature subset of size *MAX_FEATURES* is selected if $d >$ *MAX_FEATURES*. Predictions from 20 independent samples are averaged at the probability level before computing balanced accuracy.

H3: XLContext. H3 uses the full training set as context, i.e., without applying *MAX_CONTEXT* sample restriction. However, if $d >$ *MAX_FEATURES*, features are randomly subsampled to *MAX_FEATURES* using the fixed seed. If the inference triggers a CUDA out-of-memory error (depending on GPU memory), the context size is multiplied by 0.9. The method then samples a random subset of the reduced size and retries inference. This process repeats until inference succeeds.

O1: KMeansReps. O1 uses the KMeans implementation from *sklearn.cluster* for both feature and sample selection, with feature selection performed before sample selection. The number of K-means initializations is defaulted to 5. For feature-level optimization, if $d >$ *MAX_FEATURES*, each feature is treated as a point by transposing the training matrix. KMeans is then run with $n_clusters =$ *MAX_FEATURES*. For each centroid, the nearest unique feature is selected. Each initialization is evaluated on the validation set using balanced accuracy, and the best feature subset is retained. Then, if $n >$ *MAX_CONTEXT*, Kmeans is run on the reduced training matrix with $n_clusters =$ *MAX_CONTEXT*. The nearest unique training example to each centroid is selected. Validation balanced accuracy is used to choose the best initialization. The selected feature indices and context indices are then fixed for final test evaluation.

O2: DT+TFM. O2 implementation uses a decision tree for sample routing. If $d >$ *MAX_FEATURES*, features are first selected uniformly at random without replacement using the fixed seed. Then, if $n >$ *MAX_CONTEXT*, a DecisionTreeClassifier is fitted on the reduced training matrix with gini impurity, $min_samples_leaf =$ *MAX_CONTEXT*, and $random_state =$ seed. During evaluation, each test example is routed to a leaf, and the training samples assigned to that leaf are used as its local TabPFN context. If a leaf contains more than *MAX_CONTEXT* examples, only the first *MAX_CONTEXT* examples are retained.

Hardware Configuration: We based our VIP-COP and baseline experiments on 4 NVIDIA Tesla V100-SXM2-32GB GPUs with 2 Intel Xeon Gold 6248 20-Core Processors.

B Detailed Experiment Results

Table 4: **TabPFNv1** Performance results using each method on individual datasets under the **original HardCOp** setting. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COp against each Baseline is highlighted in color.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
			small-d (16)				
19	Amazon_employee_access	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
145	INNHotelsGroup	0.790	0.775	0.788	0.793	0.790	**0.820**
41	BNG(breast-w)	0.978	0.978	0.980	**0.983**	0.980	0.982
47	BNG(tic-tac-toe)	0.704	0.691	0.683	0.715	0.713	**0.757**
55	Click_prediction	**0.504**	0.500	0.500	0.501	0.500	0.500
109	FOREX_audcad-hour-High	0.676	0.611	0.630	**0.679**	0.606	0.667
112	FOREX_audjpy-hour-High	**0.618**	0.511	0.500	**0.618**	0.591	0.534
113	FOREX_audsgd-hour-High	**0.537**	0.501	0.500	0.500	0.520	0.522
114	FOREX_audusd-hour-High	**0.520**	0.500	0.500	0.500	0.500	0.502
116	FOREX_cadjpy-hour-High	0.573	0.504	**0.598**	0.500	0.527	0.515
155	jungle_chess_endgame	0.758	0.662	0.670	0.669	0.718	**0.793**
30	bank	**0.752**	0.619	0.588	0.619	0.644	0.634
93	electricity	0.765	0.759	0.759	0.757	0.764	**0.817**
188	mobile_c36_oversampling	0.904	0.906	0.909	0.914	0.909	**0.955**
42	BNG(cmc)	0.534	0.523	0.518	0.542	**0.545**	0.529
248	shuttle	0.673	0.555	0.550	0.729	**0.929**	0.619
Avg. % Improv. over Baseline		–	**7.56%**	**6.80%**	**3.08%**	**1.49%**	**2.29%**
			small-d (n>50K) (8)				
147	internet_firewall	**0.498**	**0.498**	**0.498**	0.497	0.250	**0.498**
52	Cardiovascular-Disease	0.705	0.709	**0.721**	0.709	0.674	**0.721**
77	Credit_c	0.669	0.591	0.580	0.608	0.638	**0.685**
239	SDSS17	0.956	0.950	0.958	**0.961**	0.943	0.956
80	dabetes_us_hospitals	0.596	0.576	0.592	0.594	0.594	**0.597**
15	airline_satisfaction	0.886	0.890	0.893	0.925	0.892	**0.940**
229	Rain_in_Australia	**0.688**	0.550	0.484	0.635	0.660	0.652
8	accelerometer	0.699	0.440	0.241	0.447	0.436	**0.729**
Avg. % Improv. over Baseline		–	**12.53%**	**30.62%**	**8.75%**	**21.78%**	**-1.13%**
			Large-d (14)				
15 [VLS]	nomao	0.934	0.919	0.928	**0.944**	0.932	0.938
121	gas-drift	0.980	0.966	0.985	0.983	0.973	**0.986**
20 [VLS]	volkert	0.339	0.296	0.295	0.330	0.321	**0.391**
177	mfeat-factors	0.950	0.953	**0.970**	0.958	0.955	0.958
144	Indian_pines	0.765	0.685	0.717	0.762	0.709	**0.803**
181	mfeat-pixel	0.933	0.933	**0.948**	0.938	0.938	0.938
171	madeline	0.610	0.581	0.540	0.614	0.608	**0.637**
8 [VLS]	fabert	0.275	0.234	0.217	0.261	**0.281**	0.250
10 [VLS]	gina_agnostic	0.876	0.802	0.836	0.814	0.843	**0.882**
7 [VLS]	dilbert	0.881	0.831	0.830	**0.927**	0.904	**0.927**
12 [HD]	PCMAC_1	**0.865**	0.671	0.578	0.670	0.717	0.703
14 [HD]	RELATHE_1	**0.825**	0.579	0.569	0.579	0.676	0.707
3 [HD]	BASEHOCK_1	**0.890**	0.658	0.657	0.677	0.670	0.815
6	gisette_1	0.944	0.844	0.899	0.871	**0.947**	0.926
Avg. % Improv. over Baseline		–	**13.23%**	**14.56%**	**8.51%**	**6.30%**	**2.18%**
Overall Avg. % Improv. over Baseline		–	**10.69%**	**14.68%**	**6.26%**	**7.53%**	**1.54%**

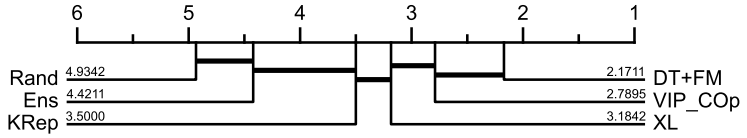


Figure 6: CD diagram across datasets (numbers depict average rank) in the original HardCOp setting.

Table 5: **TabICLv2** Performance results using each method on individual datasets under the **original HardCOp** setting. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline highlighted in color.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (KMeans)	O2 (DT+TFM)
			small-d (16)				
19	Amazon_employee_access	**0.595**	0.585	0.585	0.585	0.585	0.585
145	INNHotelsGroup	0.874	**0.882**	**0.882**	**0.882**	**0.882**	**0.882**
41	BNG(breast-w)	0.987	**0.988**	**0.988**	**0.988**	**0.988**	**0.988**
47	BNG(tic-tac-toe)	**0.789**	0.784	0.784	0.784	0.784	0.784
55	Click_prediction	**0.578**	0.521	0.521	0.521	0.521	0.521
109	FOREX_audcad-hour-High	**0.714**	0.703	0.703	0.703	0.703	0.703
112	FOREX_audjpy-hour-High	**0.721**	0.716	0.716	0.716	0.716	0.716
113	FOREX_audsgd-hour-High	**0.707**	0.656	0.656	0.656	0.656	0.656
114	FOREX_audusd-hour-High	**0.706**	0.705	0.705	0.705	0.705	0.705
116	FOREX_cadjpy-hour-High	**0.716**	0.712	0.712	0.712	0.712	0.712
155	jungle_chess_endgame	**0.858**	0.841	0.841	0.841	0.841	0.841
30	bank	**0.827**	0.757	0.757	0.757	0.757	0.757
93	electricity	0.889	**0.914**	**0.914**	**0.914**	**0.914**	**0.914**
188	mobile_c36_oversampling	0.989	**0.992**	**0.992**	**0.992**	**0.992**	**0.992**
42	BNG(cmc)	**0.578**	0.563	0.563	0.563	0.563	0.563
248	shuttle	**1.000**	0.991	0.991	0.991	0.991	0.991
Avg. % Improv. over Baseline		–	**2.18%**	**2.18%**	**2.18%**	**2.18%**	**2.18%**
			small-d (n>50K) (8)				
147	internet_firewall	**0.622**	0.571	0.571	0.571	0.571	0.582
52	Cardiovascular-Disease	**0.735**	0.733	0.733	0.734	0.733	0.732
77	Credit_c	**0.796**	0.807	0.802	0.805	**0.808**	0.801
239	SDSS17	**0.973**	0.972	0.972	0.972	0.972	0.971
80	dabetes_us_hospitals	0.633	0.631	0.631	0.631	**0.633**	0.630
15	airline_satisfaction	**0.974**	0.944	0.938	0.949	0.944	0.944
229	Rain_in_Australia	**0.705**	0.679	0.681	0.679	0.680	0.680
8	accelerometer	0.804	0.802	**0.807**	0.803	0.804	0.804
Avg. % Improv. over Baseline		–	**1.95%**	**1.98%**	**1.87%**	**1.82%**	**1.74%**
Overall Avg. % Improv. over Baseline		–	**2.10%**	**2.11%**	**2.08%**	**2.06%**	**2.03%**

Table 6: **TabICLv2** Performance results using each method on individual datasets under the **HardCOp4DA** setting with **sample augmentation** S_{aug}. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in color. Effect of sample augmentation is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (KMeans)	O2 (DT+TFM)
			small-d (16)				
19	Amazon_employee_access	**0.606**	0.564	0.557	0.585	0.572	0.569
145	INNHotelsGroup	**0.888**	0.865	0.874	0.881	0.868	0.872
41	BNG(breast-w)	**0.987**	0.985	0.985	**0.987**	0.979	0.984
47	BNG(tic-tac-toe)	**0.797**	0.772	0.775	0.779	0.776	0.780
55	Click_prediction	**0.596**	0.514	0.514	0.512	0.509	0.512
109	FOREX_audcad-hour-High	**0.728**	0.707	0.709	0.707	0.701	0.706
112	FOREX_audjpy-hour-High	**0.737**	0.720	0.721	0.720	0.716	0.722
113	FOREX_audsgd-hour-High	**0.724**	0.705	0.708	0.702	0.699	0.701
114	FOREX_audusd-hour-High	**0.721**	0.706	0.710	0.705	0.701	0.707
116	FOREX_cadjpy-hour-High	**0.731**	0.713	0.715	0.712	0.707	0.714
155	jungle_chess_endgame	**0.860**	0.810	0.814	0.839	0.840	0.844
30	bank	**0.898**	0.718	0.721	0.736	0.728	0.728
93	electricity	0.898	0.876	0.885	**0.909**	0.898	0.896
188	mobile_c36_oversampling	**0.996**	0.975	0.977	0.977	0.967	0.976
42	BNG(cmc)	**0.588**	0.565	0.568	0.567	0.564	0.570
248	shuttle	**1.000**	0.903	0.916	0.987	0.950	0.916
Avg. % Improv. over Baseline		–	**5.80%**	**5.42%**	**4.17%**	**5.23%**	**5.00%**
Overall Avg. % Δ from Original (Table 5)		**1.91%**	**-1.43%**	**-1.08%**	**0.05%**	**-0.95%**	**-0.70%**

Table 7: **TabPFNv1** Performance results using each method on individual datasets under the **Hard-COp4DA** setting with **sample augmentation** S_{aug}. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in color. Effect of sample augmentation is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
			small-d (16)				
19	Amazon_employee_access	0.508	0.500	0.505	0.500	0.500	**0.593**
145	INNHotelsGroup	0.805	0.788	0.790	0.812	0.792	**0.853**
41	BNG(breast-w)	0.980	0.979	0.979	0.983	0.980	**0.984**
47	BNG(tic-tac-toe)	0.706	0.710	0.707	0.733	0.720	**0.749**
55	Click_prediction	**0.582**	0.500	0.500	0.509	0.507	0.512
109	FOREX_audcad-hour-High	0.666	0.615	0.500	**0.677**	0.601	0.598
112	FOREX_audjpy-hour-High	0.608	0.526	0.500	**0.617**	0.589	0.553
113	FOREX_audsgd-hour-High	0.537	0.504	0.500	0.500	**0.538**	0.516
114	FOREX_audusd-hour-High	**0.531**	0.500	0.500	0.500	0.508	0.522
116	FOREX_cadjpy-hour-High	**0.582**	0.500	0.500	0.500	0.504	0.525
155	jungle_chess_endgame	0.777	0.755	0.748	0.789	0.779	**0.815**
30	bank	**0.794**	0.760	0.759	0.781	0.778	0.721
93	electricity	0.774	0.761	0.759	0.760	0.769	**0.847**
188	mobile_c36_oversampling	0.904	0.905	0.909	0.910	0.915	**0.973**
42	BNG(cmc)	0.537	0.526	0.513	**0.546**	0.527	0.521
248	shuttle	**0.997**	0.981	0.984	0.914	0.939	0.880
Avg. % Improv. over Baseline		–	**5.34%**	**7.47%**	**2.92%**	**3.70%**	**1.98%**
			Large-d (14)				
15 [VLS]	nomao	0.932	0.927	0.929	0.950	0.930	**0.951**
121	gas-drift	0.974	0.968	0.978	**0.983**	0.982	0.981
20 [VLS]	volkert	0.399	0.347	0.358	0.415	0.423	**0.509**
177	mfeat-factors	0.948	0.949	**0.958**	0.955	0.948	0.803
144	Indian_pines	**0.917**	0.893	0.862	0.797	0.796	0.750
181	mfeat-pixel	0.925	0.927	**0.948**	0.923	0.940	0.838
171	madeline	0.622	0.568	0.556	0.620	0.631	**0.708**
8 [VLS]	fabert	0.249	0.269	0.244	**0.300**	0.248	0.284
10 [VLS]	gina_agnostic	**0.859**	0.796	0.836	0.814	0.839	0.809
7 [VLS]	dilbert	0.877	0.820	0.804	**0.927**	0.898	0.885
12 [HD]	PCMAC_1	**0.835**	0.659	0.564	0.507	0.526	0.484
14 [HD]	RELATHE_1	**0.797**	0.582	0.650	0.471	0.480	0.500
3 [HD]	BASEHOCK_1	**0.935**	0.639	0.624	0.675	0.760	0.665
6	gisette_1	**0.950**	0.841	0.894	0.873	0.912	0.888
Avg. % Improv. over Baseline		–	**11.31%**	**11.91%**	**12.33%**	**11.31%**	**13.34%**
Overall Avg. % Improv. over Baseline		–	**8.13%**	**9.57%**	**7.32%**	**7.27%**	**7.28%**
Overall Avg. % Δ from Original (Table 4)		**3.43%**	**5.88%**	**4.81%**	**2.60%**	**0.65%**	**0.40%**

Table 8: **TabPFNv1** Performance results using each method on individual datasets under the **HardCOp4DA** setting with **feature augmentation** F_{aug}. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in color. Effect of feature augmentation is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
			small-d (16)				
19	Amazon_employee_access	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
145	INNHotelsGroup	**0.788**	0.773	0.769	0.782	0.756	0.787
41	BNG(breast-w)	0.979	0.979	0.978	**0.983**	0.948	0.916
47	BNG(tic-tac-toe)	**0.717**	0.682	0.699	0.709	0.685	0.675
55	Click_prediction	**0.569**	0.492	0.500	0.500	0.488	0.471
109	FOREX_audcad-hour-High	0.556	0.512	0.500	0.657	**0.676**	0.599
112	FOREX_audjpy-hour-High	0.558	0.501	0.500	0.573	**0.692**	0.550
113	FOREX_audsgd-hour-High	0.530	0.500	0.500	0.500	**0.669**	0.534
114	FOREX_audusd-hour-High	0.521	0.500	0.500	0.500	**0.671**	0.531
116	FOREX_cadjpy-hour-High	0.552	0.502	0.500	0.506	**0.674**	0.537
155	jungle_chess_endgame	**0.757**	0.650	0.609	0.667	0.688	0.697
30	bank	**0.764**	0.603	0.561	0.609	0.672	0.602
93	electricity	0.761	0.762	0.762	**0.767**	0.740	0.755
188	mobile_c36_oversampling	0.909	0.906	0.904	**0.920**	0.890	0.904
42	BNG(cmc)	**0.535**	0.513	0.525	**0.535**	0.523	0.503
248	shuttle	0.575	0.499	0.445	0.474	0.688	**0.743**
Avg. % Improv. over Baseline		–	**7.86%**	**9.65%**	**4.68%**	**-3.44%**	**3.07%**
			small-d (n>50K) (8)				
147	internet_firewall	0.497	0.416	0.497	0.250	0.249	**0.509**
52	Cardiovascular-Disease	0.705	0.700	0.696	**0.708**	0.654	0.687
77	Credit_c	**0.661**	0.582	0.613	0.601	0.600	0.649
239	SDSS17	0.952	0.939	0.940	**0.960**	0.932	0.916
80	dabetes_us_hospitals	0.599	0.585	0.569	**0.603**	0.583	0.572
15	airline_satisfaction	0.887	0.894	0.880	**0.930**	0.892	0.893
229	Rain_in_Australia	**0.680**	0.615	0.628	0.660	0.641	0.616
8	accelerometer	**0.687**	0.438	0.243	0.447	0.400	0.471
Avg. % Improv. over Baseline		–	**13.02%**	**25.93%**	**19.87%**	**24.97%**	**8.29%**
Overall Avg. % Improv. over Baseline		–	**9.58%**	**15.05%**	**9.75%**	**6.02%**	**4.80%**
Overall Avg. % Δ from Original (Table 4)		**-1.43%**	**-1.86%**	**-1.80%**	**-3.88%**	**1.95%**	**-4.11%**

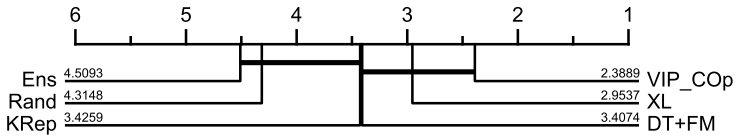


Figure 7: CD diagram across datasets (numbers depict average rank) in the **HardCOp4DA** setting.

Table 9: **TabPFNv1** Performance results using each method on individual datasets under the **Hard-COp4DN** setting with **sample noising** $S1_{noi}$. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in color. Effect of sample noise is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
				small-d (16)			
19	Amazon_employee_access	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
145	INNHotelsGroup	0.786	0.677	0.693	0.730	0.734	**0.841**
41	BNG(breast-w)	**0.981**	0.961	0.966	0.967	0.968	0.966
47	BNG(tic-tac-toe)	0.704	0.500	0.500	0.500	0.500	**0.716**
55	Click_prediction	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
109	FOREX_audcad-hour-High	**0.673**	0.502	0.500	0.500	0.497	0.563
112	FOREX_audjpy-hour-High	**0.601**	0.500	0.500	0.500	0.480	0.528
113	FOREX_audsgd-hour-High	**0.542**	0.500	0.500	0.500	0.500	0.517
114	FOREX_audusd-hour-High	**0.531**	0.500	0.500	0.500	0.500	0.512
116	FOREX_cadjpy-hour-High	**0.572**	0.500	0.500	0.500	0.500	0.520
155	jungle_chess_endgame	0.723	0.463	0.516	0.459	0.510	**0.763**
30	bank	**0.672**	0.500	0.500	0.500	0.513	0.591
93	electricity	0.767	0.709	0.711	0.732	0.723	**0.794**
188	mobile_c36_oversampling	0.911	0.874	0.880	0.899	0.884	**0.930**
42	BNG(cmc)	0.525	0.340	0.333	0.333	0.333	**0.537**
248	shuttle	0.634	0.432	0.285	0.428	0.296	**0.748**
Overall Avg. % Improv. over Baseline		–	**21.65%**	**25.35%**	**21.10%**	**24.54%**	**1.92%**
Overall Avg. % Δ from Original (Table 4)		**-1.44%**	**-11.21%**	**-13.03%**	**-13.78%**	**-15.54%**	**-0.80%**

Table 10: **TabPFNv1** Performance results using each method on individual datasets under the **Hard-COp4DN** setting with **sample noising** $S2_{noi}$. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in color. Effect of sample noise is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
				small-d (16)			
19	Amazon_employee_access	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
145	INNHotelsGroup	0.795	0.705	0.721	0.739	0.750	**0.828**
41	BNG(breast-w)	**0.979**	0.942	0.953	0.938	0.963	0.954
47	BNG(tic-tac-toe)	**0.710**	0.500	0.500	0.500	0.577	0.703
55	Click_prediction	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
109	FOREX_audcad-hour-High	**0.682**	0.502	0.500	0.500	0.500	0.544
112	FOREX_audjpy-hour-High	**0.603**	0.501	0.500	0.500	0.500	0.528
113	FOREX_audsgd-hour-High	**0.561**	0.500	0.500	0.500	0.528	0.520
114	FOREX_audusd-hour-High	0.505	0.500	0.500	0.500	0.500	**0.513**
116	FOREX_cadjpy-hour-High	**0.558**	0.500	0.500	0.500	0.500	0.522
155	jungle_chess_endgame	0.690	0.488	0.513	0.531	0.531	**0.747**
30	bank	**0.686**	0.508	0.500	0.513	0.545	0.567
93	electricity	0.771	0.642	0.642	0.693	0.707	**0.809**
188	mobile_c36_oversampling	0.913	0.873	0.859	0.900	0.891	**0.924**
42	BNG(cmc)	0.530	0.346	0.333	0.389	0.357	**0.533**
248	shuttle	**0.681**	0.422	0.279	0.423	0.303	0.552
Overall Avg. % Improv. over Baseline		–	**22.21%**	**27.38%**	**19.33%**	**21.69%**	**5.17%**
Overall Avg. % Δ from Original (Table 4)		**-1.09%**	**-11.39%**	**-13.71%**	**-12.80%**	**-13.46%**	**-3.51%**

Table 11: **TabICLv2** Performance results using each method on individual datasets under the **HardCOp4DN** setting with $S2_{noi}$ + $S2_{noi}$ **sample noising**. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in blue. Effect of sample noise is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
			small-d (16)				
19	Amazon_employee_access	**0.506**	0.500	0.500	0.500	0.500	0.500
145	INNHotelsGroup	**0.810**	0.808	0.808	0.808	0.808	0.808
41	BNG(breast-w)	**0.975**	0.967	0.967	0.967	0.967	0.967
47	BNG(tic-tac-toe)	**0.691**	0.684	0.684	0.684	0.684	0.684
55	Click_prediction	0.500	0.500	0.500	0.500	0.500	0.500
109	FOREX_audcad-hour-High	**0.702**	0.666	0.666	0.666	0.666	0.666
112	FOREX_audjpy-hour-High	**0.703**	0.671	0.671	0.671	0.671	0.671
113	FOREX_audsgd-hour-High	**0.692**	0.608	0.608	0.608	0.608	0.608
114	FOREX_audusd-hour-High	**0.691**	0.548	0.548	0.548	0.548	0.548
116	FOREX_cadjpy-hour-High	**0.675**	0.661	0.661	0.661	0.661	0.661
155	jungle_chess_endgame	**0.522**	0.487	0.487	0.487	0.487	0.487
30	bank	**0.668**	0.553	0.553	0.553	0.553	0.553
93	electricity	**0.820**	0.816	0.816	0.816	0.816	0.816
188	mobile_c36_oversampling	**0.922**	0.918	0.918	0.918	0.918	0.918
42	BNG(cmc)	**0.498**	0.477	0.477	0.477	0.477	0.477
248	shuttle	**0.896**	0.659	0.659	0.659	0.659	0.659
Avg. % Improv. over Baseline		–	**7.82%**	**7.82%**	**7.82%**	**7.82%**	**7.82%**
			small-d ($n > 50K$) (8)				
147	internet_firewall	**0.656**	0.487	0.488	0.490	0.489	0.609
52	Cardiovascular-Disease	**0.733**	0.719	0.720	0.722	0.721	0.721
77	Credit_c	**0.759**	0.708	0.720	0.710	0.707	0.710
239	SDSS17	**0.971**	0.734	0.741	0.737	0.732	0.739
80	dabetes_us_hospitals	**0.678**	0.621	0.623	0.619	0.619	0.622
15	airline_satisfaction	**0.957**	0.808	0.819	0.809	0.835	0.811
229	Rain_in_Australia	**0.685**	0.642	0.654	0.649	0.648	0.658
8	accelerometer	**0.794**	0.721	0.747	0.743	0.730	0.788
Avg. % Improv. over Baseline		–	**15.02%**	**13.65%**	**14.29%**	**14.27%**	**9.93%**
Overall Avg. % Improv. over Baseline		–	**10.22%**	**9.76%**	**9.97%**	**9.97%**	**8.52%**
Overall Avg. % Δ from Original (Table 4)		**-6.64%**	**-12.96%**	**-12.59%**	**-12.79%**	**-12.80%**	**-11.64%**

Table 12: **TabPFNv1** Performance results using each method on individual datasets under the **HardCOp4DN** setting with **feature noising** F_{noi} that mixes $F1_{noi}$+$F2_{noi}$. Best performance per dataset is shown in **bold**, with average percentage (%) improvement by VIP-COP against each Baseline is highlighted in blue. Effect of feature noise is shown as the average percentage (%) difference from performance on the original datasets, highlighted in green.

ID	Name (Source)	VIP_COp (ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	O1 (Kmeans)	O2 (DT+TFM)
			Large-d (14)				
15 [VLS]	nomao	**0.924**	0.583	0.647	0.593	0.660	0.690
121	gas-drift	**0.980**	0.124	0.191	0.121	0.190	0.245
20 [VLS]	volkert	**0.336**	0.113	0.135	0.100	0.099	0.087
177	mfeat-factors	**0.963**	0.111	0.223	0.098	0.240	0.118
144	Indian_pines	**0.703**	0.163	0.131	0.167	0.186	0.138
181	mfeat-pixel	**0.945**	0.144	0.088	0.143	0.173	0.133
171	madeline	**0.614**	0.529	0.524	0.533	0.513	0.542
8 [VLS]	fabert	**0.247**	0.154	0.148	0.156	0.149	0.156
10 [VLS]	gina_agnostic	**0.850**	0.474	0.550	0.482	0.598	0.503
7 [VLS]	dilbert	**0.857**	0.263	0.276	0.249	0.328	0.241
12 [HD]	PCMAC_1	**0.705**	0.521	0.513	0.527	0.524	0.513
14 [HD]	RELATHE_1	**0.668**	0.487	0.486	0.487	0.516	0.487
3 [HD]	BASEHOCK_1	**0.702**	0.443	0.493	0.439	0.514	0.424
6	gisette_1	**0.901**	0.463	0.457	0.500	0.611	0.500
Overall Avg. % Improv. over Baseline		–	**229.18%**	**207.91%**	**240.81%**	**154.11%**	**212.34%**
Overall Avg. % Δ from Original (Table 4)		**-6.11%**	**-50.23%**	**-46.44%**	**-52.11%**	**-48.13%**	**-54.09%**

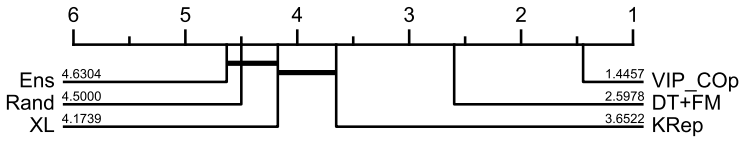


Figure 8: CD diagram across datasets (numbers depict average rank) in the **HardCOp4DN** setting.

Table 13: Pairwise comparison between each baseline and VIP-COP (TabPFN-v1 as backbone) based on p-value of permutation test over datasets across all HardCOp settings; Original, Data Augmentation (DA), Data Noise (DN). VIP-COP significantly outperforms the baselines in the majority of cases and overall, while none of the baselines consistently achieve competitive performance.

	Rand (H1)	Ens (H2)	XL (H3)	KRep (O1)	DT+FM (O2)
Original	<1e-6	<1e-6	0.002	0.012	0.207
DA w/ S_{aug}	<1e-6	<1e-6	0.007	0.001	0.021
DA w/ F_{aug}	1e-5	<1e-6	0.014	0.277	0.044
DN w/ $S1_{noi}$	2e-05	3e-05	6e-05	9e-05	0.335
DN w/ $S2_{noi}$	5e-05	3e-05	2e-05	5e-05	0.052
DN w/ F_{noi}	6e-05	9e-05	1e-05	2e-04	4e-05
Overall	**<1e-6**	**<1e-6**	**<1e-6**	**<1e-6**	**<1e-6**

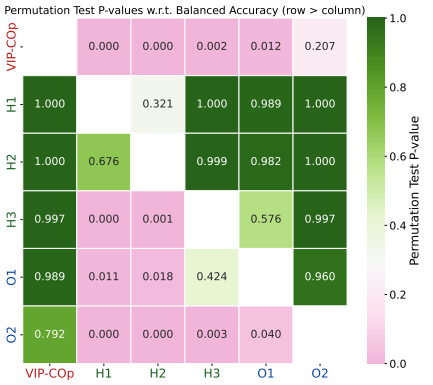


Figure 9: Paired comparisons between all context optimization methods based on p-values of permutation tests over all (38) datasets in the original HardCOp setting, with TabPFNv1 as backbone.

Table 14: **Performance results** using each method on individual datasets.

Name (Source)	VIP_COp (Ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	H4 (CMRS)	O1 (KMeans)
AbandonedMineLand	**0.749**	0.479	0.477	0.472	0.492	0.534
AdPerformance	**0.780**	0.672	0.681	0.676	0.664	0.767
AdRequestFraud	**0.756**	0.528	0.524	0.502	0.629	0.589
AgilePriority	**0.616**	0.545	0.547	0.544	0.560	0.565
AgriculturalYield	0.998	0.993	0.999	0.998	**1.000**	0.995
AircraftSensorAnomalies	1.000	0.999	0.999	0.998	1.000	1.000
AmbulanceDeployment	**0.868**	0.848	0.846	0.859	0.703	0.220
AnomalyClassification	1.000	1.000	1.000	1.000	1.000	1.000
ArtCollections	**0.820**	0.584	0.615	0.604	0.631	0.664
ArtSalesPlatform	**0.725**	0.505	0.538	0.534	0.586	0.573
ArxivProcessing	0.875	0.690	**0.973**	0.882	0.964	0.741
AstronomicalObservations	1.000	1.000	1.000	1.000	1.000	0.997
AutoencoderAnomaly	**0.549**	0.515	0.514	0.526	0.517	0.526
BackupSession	**0.908**	0.858	0.854	0.900	0.871	0.849
BankTransactions	**0.781**	0.546	0.540	0.596	0.644	0.553
BankruptcyStatus	**0.605**	0.552	0.552	0.553	0.562	0.552
BarExamPrediction	**0.794**	0.602	0.620	0.630	0.700	0.684
BaseballEvents	**0.702**	0.627	0.631	0.617	0.620	0.616
BearingFault	**0.989**	0.948	0.959	0.943	0.975	0.892
BettingIntegrity	0.650	**0.722**	0.682	0.777	0.630	0.729
BikeSharingTemperature	0.988	0.963	0.979	0.967	0.995	**0.998**

Continued on next page

Table 14: **Performance results** using each method on individual datasets.

Name (Source)	VIP_COp (Ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	H4 (CMRS)	O1 (KMeans)
BioSequenceAnomaly	**0.853**	0.602	0.587	0.580	0.692	0.673
BiologicalClassifications	**0.974**	0.868	0.905	0.864	0.969	0.904
BirdMoult	**0.933**	0.894	0.902	0.896	0.913	0.891
BirdTracking	0.999	0.999	0.999	0.999	0.999	0.999
BlacklistSpam	**0.918**	0.444	0.394	0.464	0.689	0.776
BlockchainTransaction	**0.646**	0.579	0.574	0.584	0.598	0.623
BoliviaUnpaidLabor	**0.789**	0.710	0.716	0.701	0.718	0.729
BondMarketAnomalies	0.927	0.941	0.943	0.909	**0.974**	0.907
BuildingCalls	1.000	1.000	1.000	1.000	1.000	1.000
BuildingDamage	**0.621**	0.547	0.550	0.539	0.543	0.514
BuildingInspection	0.912	0.874	0.881	0.859	0.887	**0.917**
BuildingInspections	0.743	0.506	0.627	0.466	0.771	**0.772**
BuildingPoints	1.000	1.000	1.000	1.000	1.000	1.000
BusinessFraud	**0.543**	0.487	0.485	0.483	0.506	0.483
BusinessInspections	0.443	0.479	0.471	0.455	0.478	**0.498**
BusinessProcessActivity	**0.683**	0.611	0.567	0.529	0.610	0.556
CampaignOutcome	1.000	1.000	0.983	1.000	1.000	1.000
CancerTreatment	0.568	0.555	0.559	0.553	**0.574**	0.574
CandidateVetting	**0.568**	0.534	0.535	0.537	0.534	0.534
CarMakeAnomaly	**0.781**	0.662	0.695	0.717	0.675	0.741
CardFraudDetection	0.965	0.949	0.944	0.933	**0.967**	0.951
CardioCatchBMI	0.998	0.995	0.983	0.992	0.998	0.998
CarrefourSalesDefects	**0.623**	0.563	0.579	0.562	0.579	0.578
CasePriority	0.997	0.990	0.997	0.996	0.996	0.993
ChecksumValidation	0.950	0.949	0.933	0.948	0.950	0.949
ChicagoFoodInspection	**0.606**	0.582	0.587	0.574	0.585	0.577
ChronicDiseaseStats	0.999	0.999	0.997	0.999	**1.000**	0.999
CodeDefects	**0.666**	0.623	0.595	0.613	0.653	0.614
CogLoadReaction	0.997	0.997	0.999	0.995	**0.999**	0.983
ContactTypeMonitoring	0.874	0.957	0.949	**0.998**	0.873	0.997
ConversationAnomalies	0.871	0.850	0.887	0.913	**0.920**	0.913
CoralDisease	**0.871**	0.515	0.507	0.474	0.614	0.427
CountyHealthMetrics	0.926	0.923	0.916	**0.943**	0.904	0.909
CourseEvaluation	0.619	0.172	0.326	0.183	0.214	**0.757**
CourseRegistration	**0.693**	0.555	0.572	0.520	0.678	0.569
CovidEpidemiology	0.655	0.644	0.699	0.641	0.663	**0.860**
CovidTestingData	**0.931**	0.809	0.807	0.841	0.880	0.720
CovtypeAnomaly	**0.998**	0.988	0.995	0.987	0.997	0.988
CreditCardFraud	0.789	0.770	0.776	0.771	0.767	**0.798**
CreditDefault	**0.638**	0.585	0.585	0.578	0.586	0.554
CreditRiskModel	0.677	0.462	0.413	0.549	0.478	**0.768**
CrimeAnomalies	**0.530**	0.522	0.521	0.523	0.528	0.510
CrimeClearanceStatus	0.582	0.551	0.568	0.565	0.581	**0.670**
CrimeFraudDetection	0.998	0.998	0.999	0.997	**0.999**	0.962
CropProduction	**0.948**	0.801	0.816	0.840	0.879	0.768
CustomerCampaignOutcome	1.000	0.985	0.972	0.966	1.000	1.000
CustomerOrderPriority	**0.945**	0.918	0.930	0.920	0.926	0.904
CustomerServiceTransactions	**0.591**	0.369	0.422	0.339	0.433	0.567
CybersecurityVulnerability	1.000	0.974	0.972	0.980	0.997	1.000
DallasBankGeography	0.569	**0.581**	0.535	0.566	0.579	0.456
DamageFunctionSources	0.500	0.500	0.492	0.500	0.500	0.500
DataValidationEntries	**0.623**	0.584	0.588	0.591	0.596	0.579
DeathCauses	**0.648**	0.574	0.576	0.571	0.593	0.571
DeliveryRootCause	0.595	0.566	0.558	0.575	**0.595**	0.517
DentalConditionAnomaly	**0.547**	0.523	0.517	0.522	0.526	0.513
DiabetesSpecialty	0.849	0.827	0.851	**0.867**	0.844	0.743
DiamondClarity	**0.932**	0.820	0.833	0.827	0.849	0.800
DifferentialExpression	0.500	**1.000**	0.860	**1.000**	0.500	**1.000**
DisasterIncidentType	0.327	0.370	0.358	0.171	0.394	**0.509**
DoctorVisits	0.998	0.969	0.978	0.955	0.999	**1.000**
DragonFlySystem	0.533	0.501	0.524	0.515	0.524	**0.564**
DurhamVotersAge	0.500	0.487	0.595	0.500	0.477	**0.962**
EcommerceOrderPriority	**0.686**	0.568	0.557	0.571	0.574	0.534
ElectricBuildingUsage	**0.719**	0.530	0.518	0.522	0.514	0.645

Continued on next page

Table 14: **Performance results** using each method on individual datasets.

Name (Source)	VIP_COp (Ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	H4 (CMRS)	O1 (KMeans)
ElectronicTesting	**0.869**	0.841	0.852	0.846	0.855	0.835
EmailPhishingDetection	**0.982**	0.937	0.953	0.925	0.976	0.950
EmotionalIncidents	**0.629**	0.477	0.472	0.457	0.503	0.456
EmployeeAttrition	0.898	0.761	0.807	0.801	0.775	**0.923**
EmployeeLocationAnomalies	**0.550**	0.531	0.528	0.531	0.524	0.532
EndangeredSpeciesClassification	**0.658**	0.561	0.537	0.524	0.595	0.510
GPUResourceUsage	**0.721**	0.426	0.479	0.441	0.553	0.503
GeographicSeverity	0.946	0.772	0.899	0.932	0.964	**0.954**
GeologicalLithology	0.535	0.502	0.604	0.464	0.734	**0.740**
GeospatialCrimeResolution	**0.732**	0.578	0.585	0.625	0.627	0.675
GlobalSharkAttacks	0.616	0.508	0.560	0.500	0.616	**0.657**
GlobalTerrorismAttack	**0.665**	0.502	0.552	0.519	0.571	0.642
HealthMetricsAnomalies	0.505	0.501	0.500	0.502	**0.507**	0.498
HealthPulse	**0.672**	0.606	0.616	0.590	0.616	0.622
HealthcareMessages	**0.774**	0.677	0.726	0.560	0.748	0.688
HockeyGameplay	0.511	0.505	0.508	0.508	0.511	**0.521**
HospitalSeverity	0.511	0.507	0.507	0.503	0.505	**0.521**
HousePriority	0.500	0.500	0.498	0.500	0.500	0.499
HousingConstruction	0.775	0.767	0.783	0.758	**0.805**	0.749
HuntingCriteria	0.979	0.953	0.952	0.950	**0.982**	0.981
ImageCategoryAnomaly	**0.607**	0.508	0.489	0.512	0.514	0.456
IndustrialEquipment	0.998	0.988	0.995	0.998	**0.999**	0.998
IndustrialFanVibration	**0.984**	0.982	0.983	0.983	0.984	0.982
ItemIncineration	0.663	0.643	0.653	0.660	0.663	**0.770**
JobStatus	**0.998**	0.998	0.998	0.995	0.997	0.993
JobStatusMonitoring	**0.879**	0.796	0.786	0.781	0.823	0.796
KidneyDisease	0.998	0.992	0.991	0.994	**1.000**	0.999
LabTestAbnormality	0.683	**0.732**	0.683	0.628	0.671	0.693
LandParcels	**0.739**	0.504	0.582	0.408	0.724	0.697
LandShapeTrade	0.551	0.536	0.542	0.528	0.547	**0.555**
LanguageAnomaly	**0.604**	0.577	0.573	0.580	0.579	0.567
LanguageLocationAnomaly	**0.904**	0.385	0.553	0.494	0.487	0.870
LargeWoodMeasurement	1.000	0.999	1.000	0.999	1.000	0.999
LaunchStatus	**0.734**	0.580	0.608	0.566	0.670	0.640
LegalChargeAnomalies	**0.979**	0.923	0.940	0.901	0.969	0.978
LegalEducation	**0.679**	0.592	0.587	0.567	0.617	0.609
LoanCreditScore	**0.710**	0.631	0.636	0.628	0.657	0.630
LoanFraudDetection	0.756	0.736	0.760	0.692	**0.769**	0.692
LoyaltyFraud	**0.878**	0.830	0.846	0.823	0.845	0.790
MaritimeVessel	0.990	0.989	**0.997**	0.996	0.987	0.974
MaterialFlow	**0.791**	0.674	0.679	0.667	0.692	0.707
MechanicalTesting	**0.942**	0.680	0.735	0.670	0.755	0.865
MedicalClassificationAnomalies	0.584	0.756	0.863	0.771	**0.991**	0.987
MicrostructureDefect	**0.744**	0.717	0.724	0.707	0.743	0.679
MineralComposition	0.998	0.996	0.997	0.993	0.998	0.988
MobileMoneyTransactions	**0.615**	0.543	0.548	0.544	0.558	0.539
MojaveDune	0.995	0.995	**0.997**	0.994	0.995	0.991
MouseInteractionPhase	0.927	0.832	0.879	0.834	0.881	**0.996**
MovieGenreAnomalies	**0.750**	0.670	0.677	0.674	0.684	0.684
MusicFamiliarity	**0.546**	0.525	0.527	0.522	0.536	0.532
NestSuccess	0.999	0.997	0.999	0.997	0.998	0.997
NetworkMalware	**0.796**	0.690	0.695	0.687	0.725	0.693
NetworkResponseAnomalies	1.000	0.908	0.939	0.894	1.000	1.000
NetworkThreat	0.696	0.617	0.604	0.622	**0.737**	0.439
NetworkTrafficDOS	0.999	0.985	0.977	0.960	0.999	0.996
NetworkTrafficSpam	0.829	0.322	0.598	0.398	**0.950**	0.760
NewbornReferral	**0.505**	0.426	0.416	0.349	0.453	0.467
NonComplianceInspection	0.655	0.593	0.613	0.610	0.623	**0.670**
OceanicStatus	0.998	0.995	0.998	0.993	0.997	0.995
OptimizationError	0.998	0.999	**1.000**	0.999	0.999	0.997
OrderTransmission	**0.493**	0.467	0.470	0.471	0.479	0.465
OzoneLevels	0.987	0.899	0.918	0.887	**0.988**	0.962
PatientReadmission	**0.581**	0.526	0.528	0.527	0.541	0.515
PhotoAnomalyDetection	**0.841**	0.366	0.376	0.368	0.387	0.407

Continued on next page

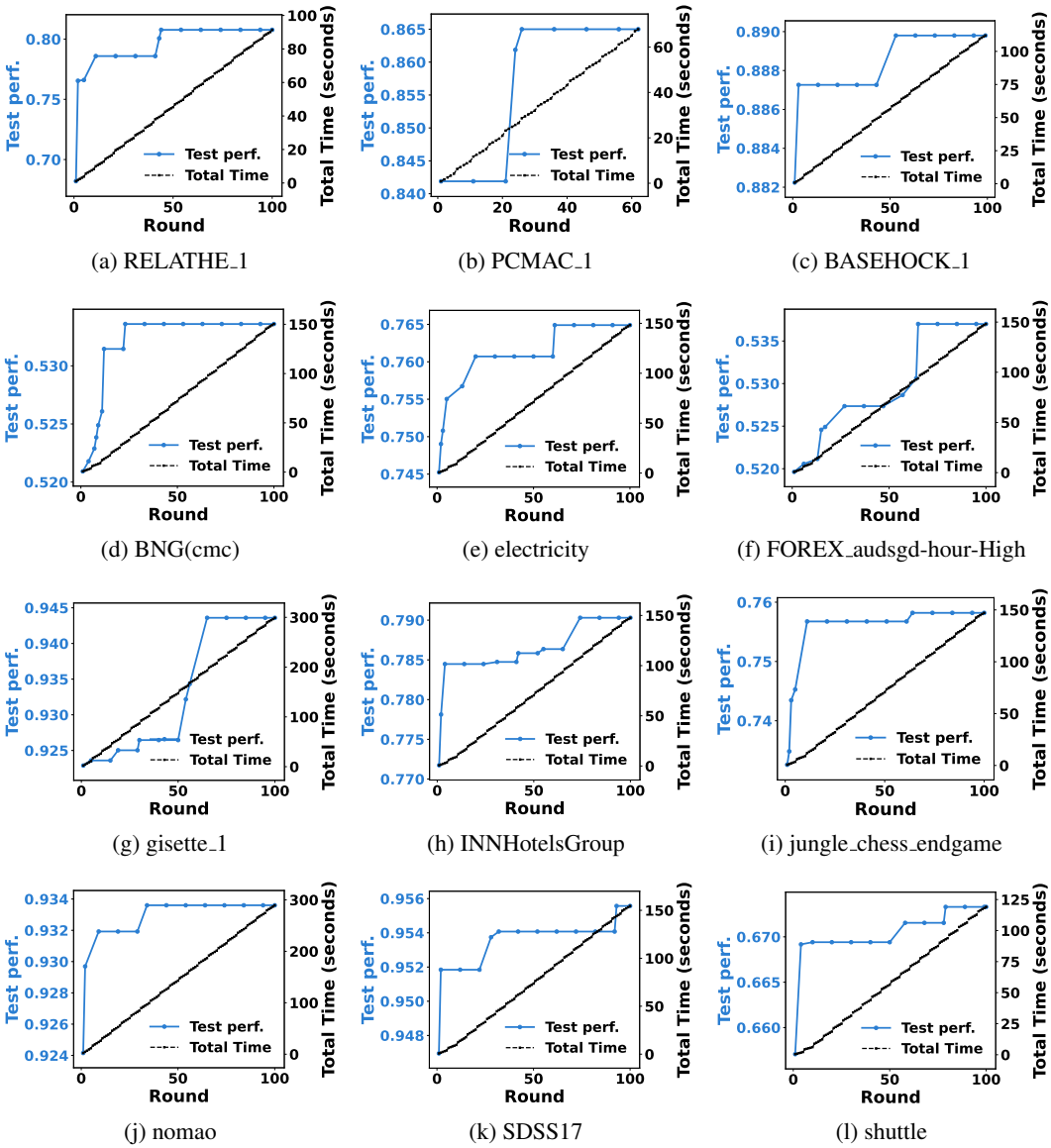
Table 14: **Performance results** using each method on individual datasets.

Name (Source)	VIP_COp (Ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	H4 (CMRS)	O1 (KMeans)
PlayerMatchEvents	**0.647**	0.621	0.624	0.620	0.624	0.635
PokemonTypeAnomaly	0.953	0.935	0.939	0.940	0.955	**0.968**
PoliceFraudIncidents	0.486	0.388	0.420	0.391	0.424	**0.489**
PoliticalInterestMisinformation	**0.941**	0.904	0.902	0.899	0.918	0.898
PopulationComparison	**0.975**	0.241	0.323	0.045	0.161	0.872
PostcodeValidation	0.481	0.555	0.644	**0.670**	0.523	0.481
PrisonBooking	**0.522**	0.485	0.487	0.491	0.501	0.509
ProcessState	**0.626**	0.533	0.532	0.536	0.541	0.542
ProductionRunStatus	**0.987**	0.809	0.872	0.792	0.975	0.955
ProgrammingAnomalies	0.528	0.523	0.526	0.522	**0.548**	0.532
ProjectSeverity	0.595	0.520	0.566	0.522	0.525	**0.722**
PropertyCondition	**0.491**	0.465	0.465	0.469	0.478	0.458
ProteinCodonAnalysis	**0.928**	0.582	0.501	0.616	0.688	0.619
ProtestEvents	0.618	0.501	0.558	0.500	0.516	**0.732**
ProtestViolence	**0.809**	0.677	0.696	0.614	0.773	0.740
PsychTrialAccuracy	**0.559**	0.505	0.516	0.502	0.522	0.551
PsychologicalTrialValidity	0.833	0.612	0.699	0.612	0.743	**0.849**
PsychologyExperiment	**0.553**	0.538	0.539	0.550	0.548	0.544
RealEstateCondition	**0.717**	0.637	0.652	0.636	0.658	0.669
RealEstateProperty	**0.912**	0.757	0.765	0.757	0.790	0.734
RegionalPriceMonitoring	0.985	0.984	0.976	0.984	0.985	**1.000**
ReportResolution	**0.646**	0.624	0.619	0.635	0.643	0.613
ResidueLigandClassification	0.871	0.866	0.876	0.818	0.865	**0.889**
RestaurantViolation	**0.541**	0.370	0.391	0.363	0.450	0.381
RiskVulnerability	0.500	0.500	0.494	0.500	0.500	0.500
RoutingAnomaly	0.983	0.982	0.982	0.983	0.983	**0.992**
SalesOrderPriority	**0.609**	0.551	0.545	0.549	0.576	0.532
SalesTaskStatus	**0.748**	0.559	0.567	0.555	0.621	0.583
SatallaxBenchmark	**0.799**	0.572	0.638	0.502	0.747	0.773
SaudiWeatherAnomalies	0.279	0.298	0.285	**0.643**	0.194	0.368
SchedulerJobState	0.993	0.967	0.961	0.963	0.992	**0.995**
SensorimotorAdaption	**0.993**	0.856	0.849	0.849	0.895	0.881
SentencingFraud	**0.811**	0.699	0.705	0.694	0.715	0.742
ServerLoadTest	0.500	0.500	0.534	0.500	0.500	**0.650**
ServiceErrorLocation	**0.615**	0.584	0.588	0.601	0.595	0.600
ShapeTrackingAnomaly	0.738	0.750	0.766	0.770	0.728	**0.806**
ShellfishQuality	**0.567**	0.504	0.486	0.512	0.499	0.457
ShuttleOutlier	0.990	0.987	0.992	0.987	**1.000**	0.979
SierraLeoneWaterPoints	**0.682**	0.554	0.552	0.575	0.596	0.581
SmallBusinessLoans	**0.993**	0.916	0.943	0.912	0.964	0.899
SnowDepthMonitoring	**0.642**	0.620	0.619	0.621	0.632	0.625
SoccerActions	**0.697**	0.689	0.694	0.685	0.688	0.664
SoccerMatchAnomalies	0.983	0.995	0.996	**0.999**	0.985	0.992
SoilFloodProperties	0.899	0.776	0.852	0.848	0.887	**0.953**
SpamDetection	**0.597**	0.440	0.447	0.435	0.508	0.418
SpeciesObservation	0.992	0.982	0.993	0.990	0.981	0.969
SpecimenStatus	**0.994**	0.772	0.832	0.731	0.983	0.836
StatisticalTestResults	1.000	0.998	1.000	0.999	1.000	0.998
StoreCrowd	0.500	0.500	0.493	0.500	0.500	0.500
StormBehavior	**0.590**	0.336	0.354	0.354	0.387	0.343
StudentAssessment	0.542	0.521	0.517	0.520	0.529	**0.551**
StudentEnrollment	**0.732**	0.547	0.553	0.538	0.592	0.565
StudentProgramAnomaly	**0.687**	0.427	0.420	0.388	0.525	0.357
SummaryAnomalies	**0.518**	0.501	0.494	0.502	0.507	0.503
SuperStoreOrderPriority	**0.576**	0.533	0.542	0.530	0.538	0.490
SystemFailure	**0.713**	0.575	0.578	0.573	0.602	0.546
SystemFailureMonitoring	0.740	0.709	0.762	0.705	0.736	**0.910**
SystemLogs	**0.398**	0.306	0.330	0.343	0.319	0.372
SystemProcessAnomalies	**0.615**	0.557	0.568	0.613	0.613	0.498
SystemStatusAnomaly	**0.821**	0.435	0.585	0.460	0.725	0.744
TechnicalAnomalies	**0.855**	0.824	0.817	0.822	0.836	0.801
TelecomMarketing	0.999	0.992	0.981	0.994	0.998	**1.000**
TestCycleAnomalies	**0.903**	0.729	0.719	0.625	0.797	0.739
TVShowAnomalies	**0.732**	0.472	0.475	0.466	0.510	0.488

Continued on next page

Table 14: **Performance results** using each method on individual datasets.

Name (Source)	VIP_COp (Ours)	H1 (Mean15)	H2 (Ensem20)	H3 (XXL)	H4 (CMRS)	O1 (KMeans)
TicketPriority	**0.675**	0.559	0.557	0.574	0.565	0.650
TorontoPoliceContacts	0.438	0.493	0.486	0.482	**0.512**	0.442
TrafficIncident	**0.790**	0.609	0.640	0.609	0.664	0.679
TrafficStopAnomalies	**0.819**	0.525	0.520	0.508	0.755	0.297
TransactionFeeAnomalies	**0.649**	0.634	0.637	0.634	0.641	0.643
TransactionService	0.992	0.982	0.982	0.991	**0.998**	0.993
TrialConditionAnomaly	**0.572**	0.536	0.536	0.523	0.554	0.541
TurnaroundValidation	**0.667**	0.562	0.559	0.565	0.569	0.558
TwitchStreamAnomalies	**0.926**	0.827	0.852	0.828	0.889	0.878
UserEventAnomalies	0.987	0.961	0.939	0.969	**0.989**	0.861
VehicleNetworkTraffic	**0.927**	0.799	0.790	0.770	0.903	0.871
VenezuelaBeneficiaries	0.500	0.497	0.614	0.581	**0.745**	0.639
VideoGameViolence	0.500	0.500	0.581	0.500	0.500	**0.836**
VolcanicAnomalies	**0.908**	0.516	0.566	0.512	0.716	0.563
VolleyballMatch	**0.821**	0.654	0.666	0.657	0.656	0.771
VoteOutcome	**0.981**	0.970	0.976	0.969	0.975	0.950
WaterMeterAnomaly	**0.947**	0.891	0.889	0.891	0.907	0.881
WaterPointFunctionality	**0.674**	0.560	0.563	0.571	0.556	0.586
WaterQualityAssessment	0.999	0.999	**1.000**	0.999	1.000	1.000
WeatherEvents	**0.940**	0.926	0.923	0.927	0.935	0.921
WikipediaVandalism	0.514	0.537	0.508	0.555	0.525	0.440
WindTurbineFault	**0.598**	0.511	0.521	0.510	0.532	0.535
WindmillAnomaly	**0.957**	0.772	0.783	0.774	0.804	0.778
WindowsEditionAnomaly	0.883	0.934	0.950	0.929	**0.961**	0.935
WorkOvertimePatterns	0.479	0.398	0.516	0.400	0.484	**0.816**
YouTubeFaceDetection	**0.605**	0.591	0.574	0.592	0.602	0.534
Avg. % Improv. over Baseline	-	**16.37%**	**12.41%**	**23.75%**	**10.01%**	**9.13%**



(a) RELATHE_1 (b) PCMAC_1 (c) BASEHOCK_1

(d) BNG(cmc) (e) electricity (f) FOREX_audsgd-hour-High

(g) gisette_1 (h) INNHotelsGroup (i) jungle_chess_endgame

(j) nomao (k) SDSS17 (l) shuttle

Figure 10: **Running time** of VIP-COP (black) *increases linearly* over optimization rounds, averaging approximately 1.4 seconds per round. Accordingly, **test performance** (blue) *improves monotonically*, indicating that VIP-COP functions as an anytime method that can accommodate varying inference-time latency requirements.

Table 15: Dataset/setting combination count by **round of saturation** (TabICLv2 backbone)

Dataset	**0–25 rounds**	**25–50 rounds**	**50–75 rounds**	**75–100 rounds**	**Total Count**
Original	6	9	7	2	24
Augmentation	3	6	5	2	16
Noise	13	5	4	2	24

C QUALITATIVE ANALYSIS OF SELECTED FEATURES: A CASE STUDY

VIP-COP supports interpretability. Unlike standard TFMs, which are largely black-box models, it provides feature-importance-based rankings of both features and samples, which can help explain decisions and guide the design of new features.

To demonstrate this capability, We evaluated VIP-COP's feature-selection mechanism on the UCI Communities and Crime Dataset Redmond (2002) to showcase VIP-COP 's interpretability in practical settings. The dataset contains 1,994 US communities (samples), 122 named attributes (features), and a continuous violent-crime-rate target. (`https://archive.ics.uci.edu/dataset/183/communities+and+crime`)

During preprocessing, we removed 22 attributes with more than 50

The resulting VIP-COP ranking closely agreed with the creator-published univariate Pearson correlations despite being computed by entirely different mechanisms. VIP-COP**'s top 4 features (*PctKids2Par*, *racepctblack*, *racePctWhite*, *PctIlleg*) were exactly the four attributes with the largest $|r|$ in the creator's analysis.** Of the Creator's 10 strongest univariate predictors, 9 appeared in VIP-COP's top 20. We noticed that major criminological constructs like family-structure, divorce triplet, investment income, and racial-composition were represented in the top 20.

Beyond univariate correlation, we noticed VIP-COP also captures feature redundancy. Looking closely on where the two rankings diverge, the creator's correlation table scored the 4 two-parent-household variants nearly identically ($|r| = 0.74/0.71/0.66/0.67$) because marginal correlation cannot see redundancy. In contrast, VIP-COP assigns sharply decreasing importance scores across this group (0.0089, 0.0039, 0.0027, and 0.0006), which reflects their diminishing marginal contributions. Similar behavior appeared in the poverty and economic block where *pctWInvInc* was selected at rank 5 while its strongly correlated counterpart *pctWPubAsst* was 84th. Conversely, several variables with relatively weak univariate correlations are promoted above stronger marginal predictors, suggesting that VIP-COP captures complementary information that is overlooked by univariate analyses.

From a practical perspective, the full analysis was a single unmodified run of the VIP-COP pipeline, which produces a complete ranked list with continuous scores. These scores naturally support feature-level interpretation while providing practitioners with a principled basis for understanding the model's feature preferences.

One limitation of this analysis is that it was conducted using a single random seed. A stability check across seeds could be done to add robustness to this analysis.