PACE: PLUG-AND-PLAY CONTEXTUAL EMBEDDING FOR FEATURE SCREENING WITH PRETRAINED TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

In high-dimensional tabular learning, feature screening provides a lightweight, model-agnostic way to remove irrelevant features before model fitting. However, scoring raw values directly can miss nonlinear or distributional structure. We introduce PACE (Plug-and-Play Contextual Embedding), which inserts a frozen tabular foundation model (TFM) column encoder before an existing feature-scoring rule, expanding each feature into a higher-dimensional contextual representation. Across controlled studies, PACE improves raw-space screening of complex nonlinear dependence with only modest additional encoding cost. These gains translate to downstream prediction on TALENT datasets: PACE-DC improves binary AUC by 0.077 and multiclass macro-AUC by 0.064, with a median normalized RMSE improvement of 0.063 across ten learners. Matched random-weight and random-feature controls show that PACE gains from pretrained structure beyond generic dimensional expansion. PACE further achieves favorable performance–time trade-offs against task-fitted selectors and attribution-based methods, positioning pretrained column geometry as a reusable upstream primitive for tabular learning.

1 INTRODUCTION

High-dimensional tabular data are prevalent in scientific and industrial applications. Increasing dimensionality exacerbates estimation error, feature-acquisition costs, and computational burden (Fan et al., 2011). *Feature screening* addresses this problem by ranking columns (features) before model fitting and retaining a compact candidate set that can be reused across tree models, neural networks (NNs), and tabular foundation models (TFMs). Despite rapid progress in TFM scaling, high-dimensional tables remain a challenging regime (Purucker et al., 2026). For a concrete reference point, TabPFN v2.5 supports datasets with up to 2,000 features (Grinsztajn et al., 2025), while specialized approaches such as TabPFN-Wide (Kolberg et al., 2025) and GOTabPFN (Habib et al., 2026) accommodate much wider inputs by modifying the predictive pipeline through continued pretraining or feature ordering and compression. Rather than modifying the predictor to accommodate increasingly wide inputs, we take an upstream approach: deciding which original columns should reach the predictive pipeline before a downstream predictor is chosen.

Classical marginal screening assigns each feature a *utility* $U(X_j, Y)$ that quantifies its association with the response. The choice of U determines which aspects of dependence are emphasized. SIS, for example, ranks features by marginal correlation (Fan & Lv, 2008); when X is symmetric, the correlation between X and $Y = X^2$ can vanish despite statistical dependence. More flexible utilities, such as distance correlation, capture broader forms of dependence (Li et al., 2012). However, the choice of utility is only part of the problem; real-world tables contain heterogeneous inputs: continuous and categorical features have different structures, and missing entries alter the empirical distribution seen by a utility. Nonlinear, category-aware, and missingness-aware screeners address important cases (Cui et al., 2015; Li & Xu, 2024; Bai & Tang, 2026), but each still operates on a particular representation. This suggests a *representation bottleneck*: some screening failures may reflect the geometry presented to the utility rather than the utility alone. Pretrained TFMs offer a learned alternative to prespecified transformations. Broad pretraining yields transferable representations, and recent work has studied task-agnostic embeddings at multiple granularities, including row- and column-level representations (Hoppe et al., 2025; Vogel et al., 2026). We instead

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.

use a frozen per-column representation as input to an existing marginal screening utility. This leads to our central question: *can pretrained column geometry improve classical screening across diverse data regimes without exposing the encoder to the current-task response?*

We introduce PACE, a **P**lug-**A**nd-Play **C**ontextual **E**mbedding adapter for feature screening. PACE inserts a frozen TFM column encoder before an existing feature-scoring rule, lifting each raw scalar feature into a higher-dimensional per-observation representation. PACE leaves the rest of the screening pipeline unchanged while replacing raw-space geometry with pretrained column geometry. The resulting representation can expose nonlinear and distributional structure that is difficult to capture directly from raw values, improving feature ranking across heterogeneous and incomplete tabular data. PACE is compatible with a broad range of screening utilities and downstream predictive models.

Our contributions are threefold. **First**, we introduce PACE as a plug-and-play representation adapter for existing feature-screening utilities, producing screened tables reusable across downstream predictors while preserving original feature identity. **Second**, extensive experiments across screening utilities and downstream predictors demonstrate broad empirical benefits. PACE improves over its raw counterpart in 98.6% of classification and 83.9% of regression non-tied utility–setting comparisons. On TALENT datasets, PACE-DC improves downstream performance by 0.077 in binary AUC and 0.064 in multiclass macro-AUC, with a median normalized RMSE improvement of 0.063. **Third**, controlled representation ablations provide evidence that dimensional expansion alone does not explain these gains: pretrained representations improve over matched random-weight, projection, and random-feature controls, while backbone and extraction choices further affect screening performance.

2 Related Work

Marginal and representation-enhanced screening. Feature screening is a classical approach to ultrahigh-dimensional data. Sure independence screening (SIS) uses marginal correlation (Fan & Lv, 2008), while DC extends screening through distance correlation (Li et al., 2012). Subsequent work adapts screening to heterogeneous feature types (Cui et al., 2015; Xie et al., 2020; Li & Xu, 2024), heavy-tailed designs (Joudah et al., 2025; Jiang et al., 2025), missing responses and censoring (Bai & Tang, 2026; Zhang & Cui, 2026), and robust settings (Guo et al., 2022; Qin et al., 2025). Projection correlation (PC) (Liu et al., 2022) and improved projection correlation (IPC) (Lin et al., 2026) provide multivariate dependence utilities. A complementary direction transforms inputs before screening: kernel screening, random Fourier features (RFF), and InterDependence Scores (IDS) use prespecified kernel or feature maps (Poignard et al., 2022; Rahimi & Recht, 2007; Radhakrishnan et al., 2025); DeepFS learns a dataset-specific autoencoder representation (Li et al., 2023); and large language models provide task-agnostic row embeddings for downstream tasks (Hoppe et al., 2025; Vogel et al., 2026). PACE instead provides a plug-and-play representation adapter for existing screening methods.

Task-adaptive feature selection. Feature selection typically couples dimensionality reduction with current-task fitting. LASSO uses an ℓ_1 penalty (Tibshirani, 1996); Stochastic Gates (Yamada et al., 2020), LassoNet (Lemhadri et al., 2021), and Deep Lasso (Cherepanova et al., 2023) learn differentiable or neural selectors. GradEnFS (Liu et al., 2024) aggregates input gradients, SAND (Pad et al., 2025) uses additive-noise distortion, and xRFM-AGOP (Beaglehole et al., 2026) uses gradient outer products. Unlike the encoder in PACE, these selectors tailor their model to the specific task.

Model-based feature attribution. Model-based feature scores include tree importance (Breiman, 2001), SHAP (Lundberg & Lee, 2017), leave-one-covariate-out importance (Lei et al., 2018), and Shapley Additive Global Importance (Covert et al., 2020). Tabular transformers can expose marginal effects (Thielmann et al., 2025), while ExplainerPFN meta-learns zero-shot Shapley-style importance without target-model queries (Fonseca & Stoyanovich, 2026). Attribution scores provide feature rankings for screening; rankings derived from a task-fitted predictor are specific to that predictor.

Learning on wide tables. Tabular foundation models such as TabPFN (Grinsztajn et al., 2025), TabICL (Qu et al., 2026), and TabDPT (Ma et al., 2025) have expanded the supported feature count through pretraining. Approaches for wider tables include TabPFN sketching (Feuer et al., 2023), continued pretraining (Kolberg et al., 2025), feature ordering and compact tokenization (Habib et al., 2026), inference-time feature construction (Nishikawa et al., 2025), and test-time partitioning (Ye

et al., 2025; Team et al., 2026). More recent models (Tao et al., 2026; Grinsztajn et al., 2026; Eo et al., 2026) have adopted feature–screening mechanisms. PACE focuses on improving the representation used by screening utilities, with the resulting ranking reusable across downstream predictors.

3 METHODOLOGY

3.1 SETUP

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ contain N observations (table rows) and p candidate features (table columns), with a classification or regression response $Y = (y_1, \ldots, y_N)^\top$. Write $X_j = (x_{1j}, \ldots, x_{Nj})^\top$ for column j, which may be continuous or categorical. The sparse target support is

$$\mathcal{S} = \left\{ j : F_{Y|\boldsymbol{X}}(\cdot \mid \boldsymbol{x}) \text{ depends on } x_j \right\}, \qquad s = |\mathcal{S}| \ll p. \tag{1}$$

A conventional marginal screener assigns $\omega_j = \mathcal{U}(X_j, Y)$ and retains the top k columns or thresholds the scores, with the cutoff determined by a cost budget or false discovery rate control (Guo et al., 2023; Cai et al., 2026). SIS uses absolute sample correlation; other utilities use different measures of dependence.

3.2 PLUG-AND-PLAY CONTEXTUAL EMBEDDING

Classical nonlinear screeners use a prespecified dependence geometry, which may not expose all nonlinear or distributional patterns equally well. PACE (Plug-and-Play Contextual Embedding) instead adapts the feature representation before applying a chosen screening utility. Specifically, a frozen column-contextual encoder maps each feature column to a higher-dimensional representation:

$$G_{\theta,N} : \mathcal{X}^N \to \mathbb{R}^{N \times d}, \qquad \mathbf{H}_j = G_{\theta,N}(X_j), \tag{2}$$

where $\mathbf{H}_j$ represents the single original feature j; its d coordinates are latent dimensions, not new selection candidates. $G_{\theta,N}$ is column-local, permutation equivariant over observations, frozen across current tasks, and never receives their responses. Categorical and missing values are handled by the native preprocessing of the underlying TFM.

PACE accommodates both vector- and scalar-input screening utilities: the former score $\mathbf{H}_j$ jointly, while the latter score each coordinate separately, with the resulting scores averaged:

$$\omega_j^{\text{PACE}} = \begin{cases} \mathcal{U}(\mathbf{H}_j, Y), & \text{vector-input utility (e.g., DC),} \\ \dfrac{1}{d} \displaystyle\sum_{m=1}^{d} \mathcal{U}\Big(\mathbf{H}_j^{(:,m)}, Y\Big), & \text{scalar-input utility (e.g., SIS).} \end{cases} \tag{3}$$

Vector-input example: DC. Distance correlation (Li et al., 2012) uses pairwise Euclidean distances between the full d-dimensional row vectors of $\mathbf{H}_j$ to measure dependence on the response:

$$\widehat{\omega}_j^{\text{PACE-DC}} = \widehat{\mathrm{dCor}}_N(\mathbf{H}_j, \psi(Y)), \tag{4}$$

where $\psi(y) = y$ for regression, and $\psi(y)$ is a one-hot encoding of y for classification. Using the doubly centered Euclidean-distance matrices A_j and B of $\mathbf{H}_j$ and $\psi(Y)$, we compute $\widehat{\mathrm{dCor}}_N = \{\langle A_j, B\rangle_F / (\|A_j\|_F \|B\|_F)\}^{1/2}$ (zero if the denominator vanishes).

Scalar-input example: SIS. For regression, SIS (Fan & Lv, 2008) computes the absolute value of the sample Pearson correlation $\widehat{\rho}$ between each embedding coordinate and Y across the N observations, then averages the d scores:

$$\widehat{\omega}_j^{\text{PACE-SIS}} = \frac{1}{d} \sum_{m=1}^{d} \left| \widehat{\rho}\Big(\mathbf{H}_j^{(:,m)}, Y\Big) \right|. \tag{5}$$

For classification, each coordinate instead uses its largest absolute correlation with a one-versus-rest class indicator, followed by the same coordinate average. We also evaluate replacing this coordinate average with a maximum for PACE-SIS in Appendix F.4.

The original methods score raw features, whereas PACE applies the same utilities in a higher-dimensional embedding space. This parallels kernel methods, which map inputs into spaces where nonlinear structure can become more accessible (Cristianini & Shawe-Taylor, 2000; Balasubramanian et al., 2013). Rather than relying on a prespecified feature map, PACE uses a frozen column encoder pretrained on large collections of synthetically generated tabular tasks, allowing the representation to capture richer nonlinear and distributional regularities learned across diverse data-generating processes. The screening utility itself remains unchanged. Algorithm 1 (Appendix B) details the procedure, while Appendix B.1 defines the utilities. The following proposition formalizes the information boundary of this response-blind mapping.

Proposition 1 (No creation of marginal information)**.** *For a fixed feature j, let G_N be a fixed deterministic measurable column operator between standard Borel spaces. If $X_j \perp Y$, then $G_N(X_j) \perp Y$. If G_N is injective on a set of X_j-probability one, then $\sigma(G_N(X_j)) = \sigma(X_j)$ up to null sets, where $\sigma(\cdot)$ denotes the generated sigma-algebra.*

Proposition 1 shows that a fixed response-blind column operator preserves independence between the feature column and the response vector. Under almost-sure injectivity, the transformation also preserves the information in the original column while reorganizing its geometry. This provides a simple information-theoretic interpretation of PACE: gains can arise from making existing marginal structure more accessible to the screening utility.

3.3 Column encoders and computational cost

Extraction boundary and instantiation. PACE can be instantiated at column-aligned representation stages across different backbones and extraction depths. TabICL (Qu et al., 2025; 2026) and TabPFN v3 (Grinsztajn et al., 2026) provide compatible column-aligned representations. The main experiments use a frozen TabICL v1 encoder, whose shared column-wise module processes each feature independently; cross-feature interaction occurs only in the subsequent row-wise module. Each column is projected to $d = 128$ and processed by three induced self-attention blocks with $M = 128$ inducing points, with the main representation read out cumulatively after the second block (L_2). Ablations vary backbone, depth, execution path, and pretrained versus random weights.

Computational cost. TabICL v1 column encoding costs $O(NMp)$, linear in p. SIS-type scoring costs $O(Npd)$; distance and kernel utilities cost $O(N^2pd)$ and can dominate encoding. Appendix G specifies the implementation.

4 Experiments

We evaluate PACE from three perspectives. First, matched raw-to-PACE comparisons isolate the representation effect by holding the screening utility and downstream learner fixed, with controlled simulations measuring support recovery and nuisance-augmented TALENT datasets (Liu et al., 2025) assessing downstream transfer. Second, natural high-dimensional datasets provide a complementary real-data evaluation. Third, representation ablations test whether the gains arise from pretrained geometry rather than dimensional expansion alone, while cross-paradigm comparisons place PACE alongside reusable and task-fitted selectors with different fitting and response-access requirements.

4.1 Comparison methods and evaluation criteria

Comparison methods. We organize the evaluated selectors by the mechanism that produces their ranking. Appendix C.2 gives exact constructions, fitting protocols, and comparison scope. **Featurewise screening utilities.** The main text highlights matched raw/PACE versions of SIS, DC, PC, and HSIC. Appendix C.1 lists the full set of utilities. We also include IDS (Radhakrishnan et al., 2025) and other prespecified nonlinear mappings in Appendix C.2. **Fitted-model importance.** We rank features using importance scores from random forests (Breiman, 2001), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). **Predictive attribution.** We compute SHAP values (Lundberg & Lee, 2017) for LightGBM, CatBoost, and TabICL v2; TabICL-SHAP uses two estimators. We also evaluate Leave-One-Covariate-Out (LOCO) (Lei et al., 2018) and Shapley Additive Global Importance (SAGE) (Covert et al., 2020). **Supervised learned selectors.** LassoNet (Lemhadri et al., 2021), Deep Lasso (Cherepanova et al., 2023), GradEnFS (Liu et al.,

Table 1: **Selected controlled MMS: raw versus PACE-L_2.** Lower is better; s is the oracle MMS and IDS is a standalone expanded baseline. The tables in the appendix list the results in more detail.

Scenario	s	SIS	DC	PC	HSIC	IDS
Classification						
Additive Gaussian	8	$8 \to 8$	$8 \to 8$	$8 \to 8$	$8 \to 8$	8
Nonlinear, independent	4	$2179 \to 4$	$6 \to 4$	$18 \to 7$	$4 \to 5$	4
Severe imbalance	10	$1728 \to 10$	$86 \to 21$	$278 \to 72$	$26 \to 18$	10
$p = 100{,}000$	15	$86096 \to 15$	$15 \to 15$	$15 \to 15$	$15 \to 15$	15
Mixed, $K = 25$	6	$753 \to 62$	$30 \to 8$	$75 \to 8$	$118 \to 20$	70
MCAR 0.30	4	$2128 \to 22$	$20 \to 20$	$130 \to 86$	$7 \to 18$	5
Regression						
Nonlinear, Gaussian	4	$1461 \to 86$	$54 \to 14$	$92 \to 28$	$49 \to 27$	1054
$p = 100{,}000$	8	$88153 \to 19$	$164 \to 23$	$2531 \to 97$	$101 \to 53$	86604
Mixed, $K = 5$	4	$197 \to 6$	$40 \to 12$	$108 \to 108$	$74 \to 58$	1556
Block, MCAR 0.15	3	$18 \to 11$	$14 \to 12$	$24 \to 58$	$28 \to 54$	1064

2024), SAND (Pad et al., 2025), and xRFM-AGOP (Beaglehole et al., 2026) learn task-specific rankings from the current labels.

Evaluation metrics. The TALENT study uses AUC for binary classification, macro-averaged AUC for multiclass classification, and RMSE for regression. Natural high-dimensional classification is evaluated using accuracy. Controlled simulation studies use the minimum model size (MMS),

$$\mathrm{MMS} = \max_{j \in \mathcal{S}} r_j, \tag{6}$$

where r_j is the rank of feature j. MMS is the size of the smallest top-ranked set containing every active feature. Its oracle value is s, and lower is better. Table 1 reports medians across matched replications; Appendix C additionally reports interquartile ranges (IQRs).

For nuisance-augmented dataset q, let O_q index its original features and let $\widehat{\mathcal{S}}_{k,q}$ be the retained set. Original-feature retention is $\mathrm{Ret}_q = |\widehat{\mathcal{S}}_{k,q} \cap O_q|/|O_q|$; higher values mean that fewer injected nuisance coordinates displace original features. For learner ℓ, repeated split r, and TALENT task metric Q, the matched PACE-minus-raw effect is

$$\delta_{\ell q} = \frac{1}{|\mathcal{R}_{\ell q}|} \sum_{r \in \mathcal{R}_{\ell q}} \begin{cases} Q_{r\ell q}^{\mathrm{PACE}} - Q_{r\ell q}^{\mathrm{raw}}, & \text{classification,} \\ (\mathrm{RMSE}_{r\ell q}^{\mathrm{raw}} - \mathrm{RMSE}_{r\ell q}^{\mathrm{PACE}})/\mathrm{RMSE}_{r\ell q}^{\mathrm{clean}}, & \text{regression.} \end{cases} \tag{7}$$

Here, $\mathrm{RMSE}_{r\ell q}^{\mathrm{clean}}$ is the RMSE of learner ℓ on the corresponding clean, non-nuisance-augmented table. We average $\delta_{\ell q}$ over the available matched learners within each dataset, then report the mean across datasets for classification and the median for regression. Uncertainty intervals are obtained by bootstrapping base datasets. The broader comparison reports improvability, $100(E_m - E_*)/E_m$, within dataset–learner blocks, where E_m is the method's error and E_* is the lowest error among the included methods; lower is better. Coverage and aggregation details appear in Appendix E.

4.2 Controlled simulations across utilities

The controlled simulation study uses 16 classification and nine regression settings with known active sets. It covers additive and nonlinear signals, imbalance, mixed feature types, correlated features with values missing completely at random (MCAR), and widths up to 100,000. Regular settings use 50 matched replications; Appendix C gives the generators, counts, and complete results.

Table 1 separates settings where raw screening is already sufficient from those where representation becomes limiting. In simple additive designs, nearly all raw utilities already attain the oracle MMS, and PACE retains this ability to recover simple signals. Under nonlinear or distributional signals, however, raw SIS can rank active features extremely late, whereas PACE-SIS often reduces MMS below 100 and frequently to the oracle level. PACE-SIS reduces MMS from 2,179 to 4 in nonlinear

classification and, at $p = 100{,}000$, from 86,096 to 15 in classification and from 88,153 to 19 in regression. More flexible utilities such as DC, HSIC, and PC also improve across diverse challenging settings, with utility-dependent gains.

IDS provides a useful representation-based comparator. Both methods enrich the feature representation before measuring dependence, but IDS uses a fixed finite-dimensional map whereas PACE uses a pretrained column Transformer. PACE's lower MMS in several difficult settings is consistent with the ablations, which indicate benefits from pretrained structure beyond generic dimensional expansion.

Beyond the representative utilities highlighted above, we aggregate results over the full utility–setting grid. In classification, 32.2% of comparisons are unchanged, largely due to oracle-level ties in easier settings; among the remaining comparisons, 98.6% favor PACE. In regression, only 1.6% are unchanged, and PACE improves 83.9% of the remainder.

4.3 Nuisance-augmented TALENT tables

We augment 300 TALENT datasets (Liu et al., 2025) with independent $\mathcal{N}(0, 1)$ nuisance columns[1] until the total width reaches 2,000. Screening retains $k = \min(N - 1, 3p_{\text{orig}})$, where N is the training-set size and p_{orig} is the number of original features before augmentation. The common budget compares ranking quality at equal retained dimensionality, while the $3p_{\text{orig}}$ target allows slack beyond the original width. This construction provides an identifiable numerical-contamination stress test: original-feature retention directly measures how well a screener prevents injected nuisance variables from displacing the original columns. Appendix E.1 documents preprocessing and sampling details. We evaluate the selected columns using ten learners in three families: **Tree models:** LightGBM (Ke et al., 2017), XGBoost (Chen & Guestrin, 2016), and CatBoost (Prokhorenkova et al., 2018); **NN models:** a standard MLP, RealMLP (Holzmüller et al., 2024), and TabM (Gorishniy et al., 2025); and **TFM models:** TabICL v2 (Qu et al., 2026), TabPFN v3 (Grinsztajn et al., 2026), TabPFN v2.6 (Hollmann et al., 2025), and TabDPT (Ma et al., 2025).

4.3.1 Matched screening and downstream effects across utilities and learners

The top row of Figure 1 quantifies nuisance rejection. Averaged over all datasets, PACE-DC raises original-feature retention from 0.426 to 0.715; PACE-SIS improves from 0.671 to 0.719, while PACE-HSIC changes only slightly. Larger retention gains also tend to correspond to larger downstream improvements, linking nuisance rejection to predictive recovery. The bottom row shows learner-specific effects. Under the dataset-level aggregation in Table 5, PACE-DC improves mean binary AUC by 0.077 and mean multiclass macro-AUC by 0.064. PACE-SIS and PACE-HSIC show smaller positive improvements in classification across learners (Tables 6 and 7). In regression, PACE-DC achieves a median normalized RMSE improvement of 0.063, while SIS and HSIC show smaller positive point improvements.

Appendix E.2 further examines nuisance sensitivity across learner families. NNs and TFMs are substantially more affected than tree models: in binary classification, nuisance damage is 0.049 for trees versus 0.123 and 0.114 for NNs and TFMs, while clean-normalized regression damage rises from 0.114 for trees to 0.431 and 0.609, respectively. PACE reduces this burden before downstream prediction, with benefits extending across screening utilities and learner families. A complementary recovery analysis in Appendix E.2 shows that PACE restores a larger share of the predictive performance lost to nuisance features, with the strongest recovery observed for PACE-DC.

4.3.2 Comparison with classical and learned selectors

For the broader comparison, we impose a three-hour screening-time budget and rank methods that complete within this limit. Unless otherwise noted, all feature-selection times are measured on a single NVIDIA A100 GPU. Figure 2(a) shows that all three PACE variants lie on the displayed performance–time Pareto frontier, spanning complementary operating points: PACE-SIS, PACE-DC, and PACE-HSIC achieve 10.9%, 9.8%, and 8.0% improvability at screening times of 0.21, 1.12, and 6.60 seconds, respectively. PACE-DC achieves both lower improvability and lower screening time than the evaluated learned selectors, including xRFM-AGOP and LassoNet, while PACE-HSIC

[1]Numerical inputs are normalized before downstream prediction in our evaluation pipeline, so standard Gaussian noise provides a common, scale-controlled nuisance distribution across datasets.

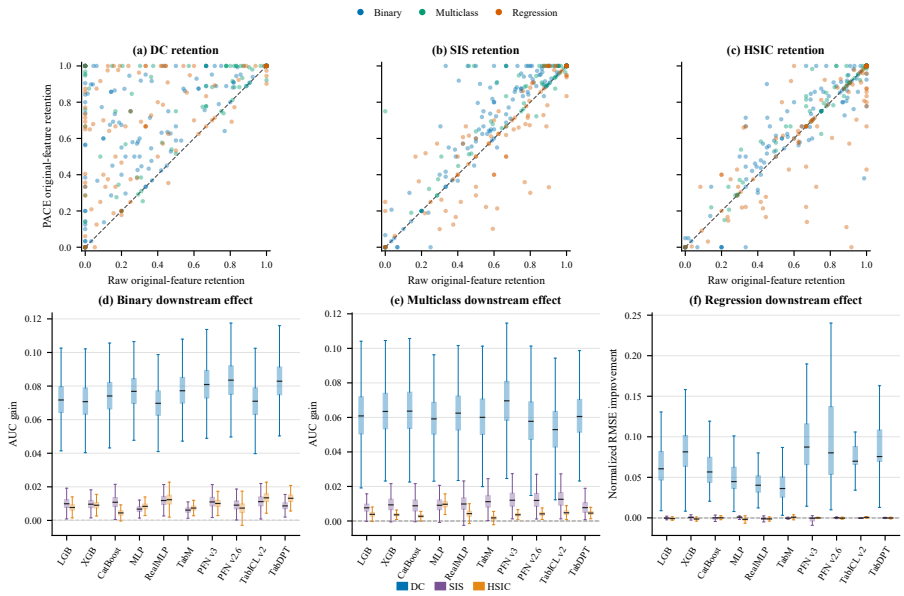


Figure 1: **Nuisance rejection and downstream recovery on TALENT.** Top: matched original-feature retention; points above the diagonal favor PACE. Bottom: bootstrap distributions of learner-specific PACE-minus-raw effects; positive values favor PACE. Effects are binary/multiclass AUC gains or clean-normalized RMSE improvement.

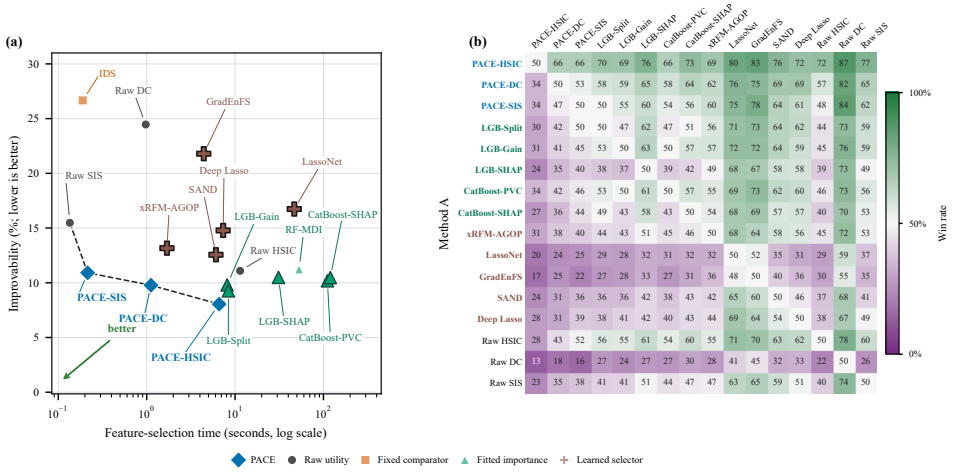


Figure 2: **TALENT performance–time trade-offs and pairwise comparisons.** (a) Time vs. improvability (lower-left is better); the Pareto frontier is shown, and marker shapes indicate method families. (b) Win rate of the row method against the column method (ties count as half a win). Results are averaged over learners; protocols are given in Appendix E.3.

further improves the trade-off relative to tree-based feature-importance methods such as LightGBM gain and CatBoost PVC. Among methods completing within the budget, PACE-HSIC attains the lowest improvability and the best average rank across all three tasks. In Figure 2(b), PACE-HSIC's aggregated win rates against the other displayed methods range from approximately 66% to 87%.

We next ask whether upstream screening also improves prediction with wide-table TFMs. Across the four general-purpose TFMs in Figure 3(a)–(b), screening increases mean classification AUC and reduces median relative RMSE gaps compared with the matched all-feature pipelines. On the shared classification support, PACE-HSIC with TabICL v2 achieves a task-balanced mean AUC

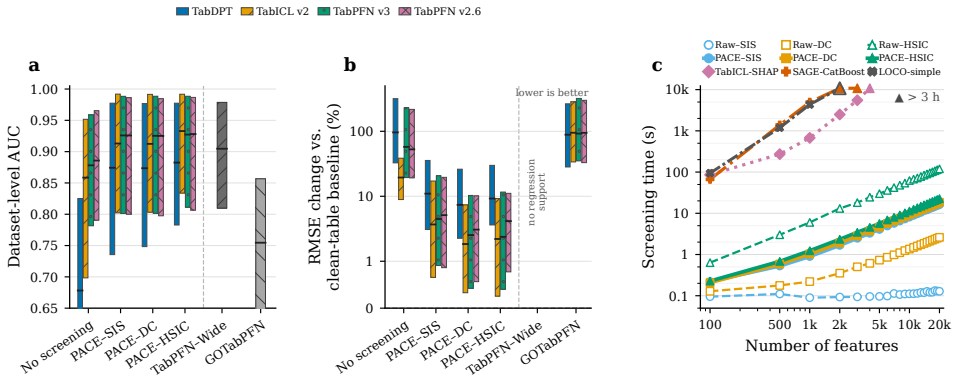


Figure 3: **Runtime scaling and wide-table TFM prediction.** (a) Dataset-level AUC on classification datasets shared by all compared pipelines. (b) Relative RMSE change from the corresponding clean-table TFM on the regression datasets; lower is better. Boxes show quartiles and medians. (c) Screening time at $N = 1{,}000$; triangles mark values above three hours.

of 0.891, compared with 0.879 for TabPFN-Wide and 0.753 for GOTabPFN. For regression, using clean TabICL v2 as the reference, PACE-DC reduces the median relative RMSE gap from 19.43% without screening to 1.84%; GOTabPFN yields a 95.11% gap, while TabPFN-Wide does not provide regression support. These results show that upstream screening can complement general-purpose TFMs while remaining reusable across downstream models.

The computational contrast is even larger for more expensive attribution procedures. TabICL-SHAP, LOCO, and SAGE exceed the three-hour budget in this comparison, whereas every PACE variant completes screening in under ten seconds. Thus, PACE improves the performance–time trade-off without fitting a task-specific selector or repeatedly querying a predictive model. This large gap motivates the next experiment, where we examine how the screening cost of PACE and attribution-based alternatives scales as feature dimensionality increases.

4.3.3 Feature-dimension scaling

To examine how this computational gap evolves with dimensionality, we fix $N = 1{,}000$ and vary the number of features p from 100 to 20,000 using a consumer-grade NVIDIA RTX 4080 SUPER GPU (Figure 3(c)). The PACE variants and traditional screening utilities scale approximately linearly with p. In contrast, TabICL-SHAP and SAGE-CatBoost show roughly quadratic growth beyond their initial overhead, and LOCO-simple also scales steeply because it repeatedly refits a random forest after removing each feature. Consequently, the runtime gap widens rapidly as dimensionality increases: at $p = 2{,}000$, the PACE variants require about 2 seconds, compared with $2{,}497$ seconds for TabICL-SHAP, $17{,}769$ seconds for SAGE-CatBoost, and $14{,}840$ seconds for LOCO-simple. These results show that PACE retains a favorable screening-time profile as feature dimensionality grows, complementing the performance–time trade-offs observed in Figure 2. Appendix E.4 additionally reports screening time as the sample size increases at fixed feature dimension.

Natural high-dimensional evaluation. We evaluate ten natural classification datasets with 500–$10{,}304$ features at a 30% selection budget, where TFMs can be sensitive to feature width (Purucker et al., 2026). PACE-SIS with TabPFN v3 (92.3%) and PACE-DC with TabICL v2 (92.5%) have the highest complete-coverage means for their respective downstream models (Appendix Table 12).

4.4 Representation ablations: backbone, pretraining, and extraction

Across controlled settings, Figure 4 examines whether the screening gains arise from the pretrained representation itself and how they depend on the extraction choice. We summarize each variant by first aggregating paired log-MMS ratios across utilities within a setting and then taking the median across settings; positive values indicate that fewer top-ranked features are required to recover the active set.

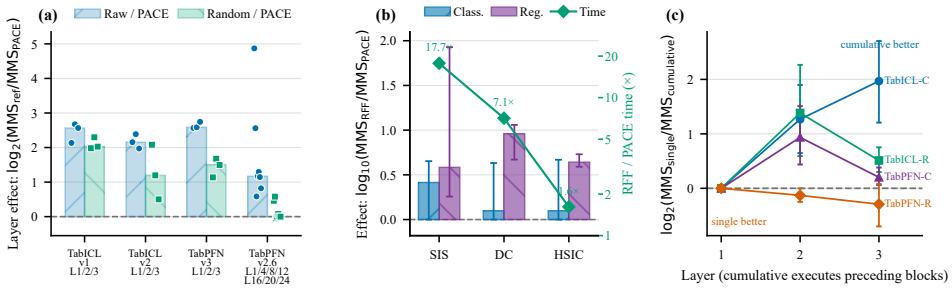


Figure 4: **Representation ablations.** (a) Backbone and random-weight controls: points denote layers and bars their medians. (b) RFF comparison: positive values favor PACE and labels give RFF-to-PACE time ratios. (c) Cumulative versus same-layer single-block extraction: C/R denote classification/regression.

In panel (a), for the main TabICL v1-L_2 representation, the resulting effects correspond to MMS reduction factors of 6.38 relative to raw input and 4.91 relative to its matched random-weight counterpart. The latter control preserves the Transformer architecture, representation dimension, and extraction procedure while replacing pretrained weights with random initialization, thereby isolating the contribution of pretraining from architectural expansion. Positive contrasts for TabICL v2 and TabPFN v3 further show that the gain extends beyond the main TabICL v1 backbone.

Panel (b) provides a complementary nonlinear-expansion control using RFF with the same output dimension as the main TabICL representation. PACE has lower aggregate MMS than RFF in all six task–utility summaries, while the panel also reports their relative screening costs. The RFF map uses random features with an input-dependent bandwidth. Together with the matched random-weight comparison, these results provide evidence that dimensional expansion alone does not explain the observed gains; the structure learned during TFM pretraining contribute additional screening value.

Finally, panel (c) separates cumulative extraction, which executes all preceding blocks, from applying the same target block in isolation. At L_2/L_3, cumulative execution lowers MMS for TabICL v2 and for TabPFN v3 classification, whereas single-block extraction tends to perform better for TabPFN v3 regression. The preferred extraction path varies across backbones and tasks, showing that both the pretrained representation and its extraction affect screening performance. Appendix Figures 12–14 provide the full backbone-, layer-, and utility-level results.

5 Conclusion

PACE provides a reusable feature-screening representation adapter using a frozen TFM column encoder. By preserving original feature identity while changing its representation, PACE helps existing utilities expose complex nonlinear and distributional signals while remaining compatible with diverse downstream predictors. Across controlled and real-data evaluations, the gains extend across utilities and learner families. PACE provides favorable performance–time trade-offs on wide tables, and ablations show that pretrained structure contributes beyond generic dimensional expansion.

Screening performance still depends on the representation, utility, backbone, and extraction strategy. Future work includes combining raw and pretrained representations, exploring additional pretrained components, and reducing the quadratic cost in sample size of distance- and kernel-based utilities. Because column-wise attention can induce dependence across embedded observations, establishing sure screening properties under this dependence is another important direction. More broadly, our results suggest that pretrained tabular representations can serve not only predictive models but also reusable upstream statistical procedures.

AI Use Statement. We used ChatGPT 5.6 to assist with literature searches, language polishing, grammar checking, software implementation, and code drafting. All AI-assisted code was reviewed, tested, and verified. All experiments and scientific analyses were conducted by the authors, and generative AI was not used to generate experimental results. We reviewed all AI-assisted content and take responsibility for the final manuscript.

REFERENCES

Yuliang Bai and Niansheng Tang. Model-free feature screening for ultrahigh dimensional data with responses missing not at random. *Journal of Multivariate Analysis*, pp. 105605, 2026.

Krishnakumar Balasubramanian, Bharath Sriperumbudur, and Guy Lebanon. Ultrahigh dimensional feature screening via rkhs embeddings. In *Artificial Intelligence and Statistics*, pp. 126–134. PMLR, 2013.

Daniel Beaglehole, David Holzmüller, Adityanarayanan Radhakrishnan, and Misha Belkin. xrfm: Accurate, scalable, and interpretable feature learning models for tabular data. In *International Conference on Learning Representations*, volume 2026, pp. 146422–146454, 2026.

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

Zhanrui Cai, Yingying Fan, and Lan Gao. Knockoff inference under privacy constraints. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, pp. qkag120, 2026.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Valeriia Cherepanova, Roman Levin, Gowthami Somepalli, Jonas Geiping, C Bayan Bruss, Andrew G Wilson, Tom Goldstein, and Micah Goldblum. A performance-driven benchmark for feature selection in tabular deep learning. *Advances in Neural Information Processing Systems*, 36: 41956–41979, 2023.

Ian Covert, Scott M Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. *Advances in neural information processing systems*, 33:17212–17223, 2020.

Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

Hengjian Cui, Runze Li, and Wei Zhong. Model-free feature screening for ultrahigh dimensional discriminant analysis. *Journal of the American Statistical Association*, 110(510):630–641, 2015.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. Exaone tabular 1.0 : Technical report, 2026.

Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(5):849–911, 2008.

Jianqing Fan, Yang Feng, and Rui Song. Nonparametric independence screening in sparse ultra-high-dimensional additive models. *Journal of the American Statistical Association*, 106(494):544–557, 2011.

Benjamin Feuer, Chinmay Hegde, and Niv Cohen. Scaling tabpfn: Sketching and feature selection for tabular prior-data fitted networks. *arXiv preprint arXiv:2311.10609*, 2023.

Joao Fonseca and Julia Stoyanovich. Explainerpfn: Towards tabular foundation models for model-free zero-shot feature importance estimations. *arXiv preprint arXiv:2601.23068*, 2026.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*, volume 2025, pp. 77899–77935, 2025.

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

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

Xu Guo, Runze Li, Wanjun Liu, and Lixing Zhu. Stable correlation and robust feature screening. *Science China Mathematics*, 65(1):153–168, 2022.

Xu Guo, Haojie Ren, Changliang Zou, and Runze Li. Threshold selection in feature screening for error rate control. *Journal of the American Statistical Association*, 118(543):1773–1785, 2023.

Al Zadid Sultan Bin Habib, Md Younus Ahamed, Prashnna Kumar Gyawali, Gianfranco Doretto, and Donald A Adjeroh. Gotabpfn: From feature ordering to compact tokenization for tabular foundation models on high-dimensional data. *arXiv preprint arXiv:2606.05441*, 2026.

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

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

Frederik Hoppe, Lars Kleinemeier, Astrid Franz, and Udo Göbel. Comparing task-agnostic embedding models for tabular data. *arXiv preprint arXiv:2511.14276*, 2025.

Yunlu Jiang, Xiaowen Huang, Jing Tan, and Ruizhe Jiang. Robust feature screening via grothendieck's correlation with fdr control. *Statistics and Computing*, 35(5):141, 2025.

Ibrahim Joudah, Samuel Muller, and Houying Zhu. Air-holp: adaptive regularized feature screening for high dimensional correlated data. *Statistics and Computing*, 35(3):63, 2025.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

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

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523): 1094–1111, 2018.

Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021. URL `http://jmlr.org/papers/v22/20-848.html`.

Kexuan Li, Fangfang Wang, Lingli Yang, and Ruiqi Liu. Deep feature screening: Feature selection for ultra high-dimensional data via deep neural networks. *Neurocomputing*, 538:126186, 2023.

Runze Li, Wei Zhong, and Liping Zhu. Feature screening via distance correlation learning. *Journal of the American Statistical Association*, 107(499):1129–1139, 2012.

Xingxiang Li and Chen Xu. Feature screening with conditional rank utility for big-data classification. *Journal of the American Statistical Association*, 119(546):1385–1395, 2024.

Jian Lin, Liping Zhu, and Tingyou Zhou. Model-free group screening and fdr control with deep knockoffs. *Science China Mathematics*, 69(5):1355–1372, 2026.

Kaiting Liu, Zahra Atashgahi, Ghada Sokar, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Supervised feature selection via ensemble gradient information from sparse neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 3952–3960. PMLR, 2024.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226): 1–16, 2025. URL `http://jmlr.org/papers/v26/25-0512.html`.

Wanjun Liu, Yuan Ke, Jingyuan Liu, and Runze Li. Model-free feature screening and fdr control with knockoff features. *Journal of the American Statistical Association*, 117(537):428–443, 2022.

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, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pIZxEOZCId.

Qing Mai and Hui Zou. The fused kolmogorov filter: A nonparametric model-free screening method. *The Annals of Statistics*, pp. 1471–1497, 2015.

Naoki Nishikawa, Yujin Song, Kazusato Oko, Denny Wu, and Taiji Suzuki. Nonlinear transformers can perform inference-time feature learning. In *Forty-second International Conference on Machine Learning*, 2025.

Pedram Pad, Hadi Hammoud, Mohamad Dia, Nadim Maamari, and L Andrea Dunbar. Sand: One-shot feature selection with additive noise distortion. *arXiv preprint arXiv:2505.03923*, 2025.

Rui Pan, Hansheng Wang, and Runze Li. Ultrahigh-dimensional multiclass linear discriminant analysis by pairwise sure independence screening. *Journal of the American Statistical Association*, 111(513):169–179, 2016.

Benjamin Poignard, Peter J Naylor, Héctor Climente-González, and Makoto Yamada. Feature screening with kernel knockoffs. In *International Conference on Artificial Intelligence and Statistics*, pp. 1935–1974. PMLR, 2022.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

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

Qi Qin, Erbo Li, Xingxiang Li, Yifan Sun, Wu Wang, and Chen Xu. Label-shift robust federated feature screening for high-dimensional classification. *arXiv preprint arXiv:2506.00379*, 2025.

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. *arXiv preprint arXiv:2602.11139*, 2026.

Adityanarayanan Radhakrishnan, Yajit Jain, Caroline Uhler, and Eric S Lander. Efficiently quantifying dependence in massive scientific datasets using interdependence scores. *Proceedings of the National Academy of Sciences*, 122(34):e2509860122, 2025.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

M. Robnik-Sikonja and Igor Kononenko. An adaptation of relief for attribute estimation in regression. In *International Conference on Machine Learning*, 1997.

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

TabLDM Team, Penghui Wang, Wei Liu, Hong Wang, Chengyue Huang, Yuxi Sun, Zirui Wang, Hongming Huang, Quan Wang, Chunxiao Liu, et al. Xiaomi-tabldm: A tabular foundation model technical report. *arXiv preprint arXiv:2609.03880*, 2026.

Anton Thielmann, Arik Reuter, and Benjamin Säfken. Beyond black-box predictions: Identifying marginal feature effects in tabular transformer networks. *arXiv preprint arXiv:2504.08712*, 2025.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

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

Jiujing Wu and Hengjian Cui. Model-free feature screening based on hellinger distance for ultrahigh dimensional data. *Statistical Papers*, 65(9):5903–5930, 2024.

Jinhan Xie, Yuanyuan Lin, Xiaodong Yan, and Niansheng Tang. Category-adaptive variable screening for ultra-high dimensional heterogeneous categorical data. *Journal of the American Statistical Association*, 115(530):747–760, 2020.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *International conference on machine learning*, pp. 10648–10659. PMLR, 2020.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Harry Chao. A closer look at tabpfn v2: Understanding its strengths and extending its capabilities. *Advances in Neural Information Processing Systems*, 38: 135605–135637, 2025.

Jing Zhang and Hengjian Cui. Fdr control in feature screening for ultrahigh-dimensional right-censored data via data-driven threshold selection. *Journal of Computational and Graphical Statistics*, pp. 1–14, 2026.

A Appendix Overview

The appendix is organized as follows. Appendix B gives the screening procedure, with utility formulas in Appendix B.1. Appendix C presents complete matched raw-to-PACE simulation results, implementation details for the cross-paradigm baselines, and setting-level MMS comparisons. Appendix D proves the information-boundary result and states its scope. Appendix E presents the full nuisance-augmented TALENT evaluation, including matched comparisons across downstream learners, model-family sensitivity, contamination recovery, a broad screening comparison, and natural high-dimensional classification. Appendix F extends the main-text ablation analysis across pretrained backbones, extraction layers, screening utilities, and mean versus maximum coordinate aggregation. Appendix G provides implementation, model, data, and responsible-use details.

Collectively, the empirical study covers 25 controlled settings; 300 nuisance-augmented TALENT datasets; ten natural high-dimensional classification datasets evaluated with TabPFN v3 (Grinsztajn et al., 2026) and TabICL v2 (Qu et al., 2026) using classical, learned, attribution-based, and tree-importance selectors, as well as with TabPFN-Wide (Kolberg et al., 2025) and GOTabPFN (Habib et al., 2026); and representation analyses across multiple backbones, layers, and screening utilities. Additional controls include five supervised learned selectors and matched pretrained and random-weight representations.

B Algorithmic Specification

Algorithm 1 PACE screening with a response-blind column encoder.

Require: Training feature columns $\{X_j\}_{j=1}^{p}$ and response Y; frozen column encoder $G_{\theta,N}$
Require: Prespecified screening utility $\mathcal{U}$; selection budget k
Ensure: Scores $\widehat{\boldsymbol{\omega}} \in \mathbb{R}^p$ and selected original-feature indices $\widehat{\mathcal{S}}_k$

for $j = 1, \ldots, p$ **in parallel do**
  $\widetilde{X}_j \leftarrow \textsc{Preprocess}(X_j)$ under the fixed study protocol
  $\mathbf{H}_j \leftarrow G_{\theta,N}(\widetilde{X}_j)$ ▷ Y is not passed to $G_{\theta,N}$
  if $\mathcal{U}$ accepts vector-valued input **then**
    $\widehat{\omega}_j \leftarrow \mathcal{U}(\mathbf{H}_j, Y)$
  else
    for $m = 1, \ldots, d$ **do**
      $u_{jm} \leftarrow \mathcal{U}(\mathbf{H}_j^{(:,m)}, Y)$
    end for
    $\widehat{\omega}_j \leftarrow d^{-1} \sum_{m=1}^{d} u_{jm}$
  end if
end for
$\pi \leftarrow \operatorname{argsort}_{j=1,\ldots,p}(\widehat{\omega}_j)$ in decreasing order
$\widehat{\mathcal{S}}_k \leftarrow \{\pi_1, \ldots, \pi_k\}$
return $\widehat{\boldsymbol{\omega}}$ and $\widehat{\mathcal{S}}_k$

The encoder $G_{\theta,N}$ uses feature values only, while the supervised utility $\mathcal{U}(\cdot, Y)$ also receives the training response. Appendix E.1 describes the real-data preprocessing.

B.1 Screening utility definitions

We define the screening utilities using a common notation. Let $Z = (z_1, \ldots, z_N)^\top \in \mathbb{R}^{N \times q}$ denote the input for one original feature: $q = 1$ for raw screening and $q = d$ for its PACE representation. For dependence measures with vector inputs, use the response representation $V = \psi(Y)$ defined in the main text, with rows $v_i^\top$.

For a scalar-valued feature z and a classification response, let $\mathcal{I}_r = \{i : y_i = r\}$, $N_r = |\mathcal{I}_r|$, and $\widehat{\pi}_r = N_r/N$, with $r = 1, \ldots, R$ indexing the observed classes. Define the empirical marginal,

class-conditional, and complementary-class distribution functions by

$$\widehat{F}(t) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(z_i \le t), \qquad \widehat{F}_r(t) = \frac{1}{N_r}\sum_{i\in\mathcal{I}_r} \mathbb{I}(z_i \le t),$$
$$\widehat{F}_{-r}(t) = \frac{1}{N - N_r}\sum_{i\notin\mathcal{I}_r} \mathbb{I}(z_i \le t). \tag{8}$$

We use $C_N = I_N - N^{-1}\mathbf{1}\mathbf{1}^\top$ for centering and $\langle A, B\rangle_F = \sum_{i,\ell} A_{i\ell}B_{i\ell}$ for the Frobenius inner product. Normalized scores are set to zero when their denominator vanishes.

SIS. SIS (Fan & Lv, 2008) uses absolute marginal Pearson correlation. For vectors $a, b \in \mathbb{R}^N$ with sample means $\bar{a}$ and $\bar{b}$, define

$$\widehat{\rho}(a, b) = \frac{\sum_{i=1}^{N}(a_i - \bar{a})(b_i - \bar{b})}{\left\{\sum_{i=1}^{N}(a_i - \bar{a})^2 \sum_{i=1}^{N}(b_i - \bar{b})^2\right\}^{1/2}}. \tag{9}$$

The regression score is $\mathcal{U}_{\mathrm{SIS}}(z, Y) = |\widehat{\rho}(z, Y)|$. For classification, we use the largest absolute one-versus-rest correlation,

$$\mathcal{U}_{\mathrm{SIS}}(z, Y) = \max_{1\le r\le R} \left|\widehat{\rho}\big(z, \mathbb{I}(Y = r)\big)\right|. \tag{10}$$

DC. Distance-correlation screening (Li et al., 2012) operates on pairwise Euclidean distances. Let $D^Z_{i\ell} = \|z_i - z_\ell\|_2$, $D^V_{i\ell} = \|v_i - v_\ell\|_2$, $A = C_N D^Z C_N$, and $B = C_N D^V C_N$. The sample distance correlation is

$$\mathcal{U}_{\mathrm{DC}}(Z, Y) = \left\{\frac{\langle A, B\rangle_F}{\|A\|_F \|B\|_F}\right\}^{1/2}. \tag{11}$$

Thus the distances are computed between scalar observations for raw screening and between full embedding vectors for PACE.

FKF. The Kolmogorov-filter family (Mai & Zou, 2015) compares conditional distribution functions. The classification implementation uses the largest one-versus-rest Kolmogorov discrepancy,

$$\mathcal{U}_{\mathrm{FKF}}(z, Y) = \max_{1\le r\le R} \sup_{t\in\mathbb{R}} \left|\widehat{F}_r(t) - \widehat{F}_{-r}(t)\right|. \tag{12}$$

The supremum is evaluated over the pooled observed feature values.

MV. The mean–variance index of Cui et al. (2015) measures variation in the conditional distribution function across response classes:

$$\mathrm{MV}(z \mid Y) = \sum_{r=1}^{R} \pi_r \int \{F_r(t) - F(t)\}^2 \, dF(t). \tag{13}$$

Here F, F_r, and π_r are the population counterparts of the quantities in Equation (8). The empirical counterpart of the mean–variance index is

$$\mathcal{U}_{\mathrm{MV}}(z, Y) = \frac{1}{N}\sum_{i=1}^{N}\sum_{r=1}^{R} \widehat{\pi}_r \left\{\widehat{F}_r(z_i) - \widehat{F}(z_i)\right\}^2. \tag{14}$$

PSIS. Pairwise screening (Pan et al., 2016) compares feature locations across response classes. Write $\widehat{\mu}_r = N_r^{-1}\sum_{i\in\mathcal{I}_r} z_i$ and $\widehat{m}_r = \mathrm{median}\{z_i : i \in \mathcal{I}_r\}$. The corresponding mean- and median-contrast scores are

$$\mathcal{U}^{\mathrm{mean}}_{\mathrm{PSIS}}(z, Y) = \max_{r<s} |\widehat{\mu}_r - \widehat{\mu}_s|, \qquad \mathcal{U}^{\mathrm{median}}_{\mathrm{PSIS}}(z, Y) = \max_{r<s} |\widehat{m}_r - \widehat{m}_s|. \tag{15}$$

The original method uses mean contrasts; our implementation uses the median-contrast variant, with the lower middle order statistic when a class has an even sample size.

CAVS. Category-adaptive screening (Xie et al., 2020) defines a class-specific utility for a continuous scalar feature X as

$$\tau_r = \mathbb{E}_X\{F_r(X)\} - \frac{1}{2} = \int F_r(t)\, dF(t) - \frac{1}{2}. \tag{16}$$

The expectation is taken over the marginal feature distribution. The original sample estimator is

$$\widehat{\tau}_r = \frac{1}{N+1}\sum_{k=1}^{N} \widehat{F}_r(z_k) - \frac{1}{2} = \frac{1}{(N+1)N_r}\sum_{k=1}^{N}\sum_{i\in\mathcal{I}_r} \mathbb{I}(z_i \le z_k) - \frac{1}{2}. \tag{17}$$

Aggregating the class-specific contrasts gives

$$\mathcal{U}_{\mathrm{CAVS}}(z, Y) = \max_{1\le r\le R} |\widehat{\tau}_r|. \tag{18}$$

The $N+1$ denominator makes each class-specific estimator mean-zero under independence for continuous features.

HSIC. HSIC-based screening (Poignard et al., 2022) uses the biased empirical Hilbert–Schmidt independence criterion

$$\mathcal{U}_{\mathrm{HSIC}}(Z, Y) = \frac{1}{N^2}\,\mathrm{tr}(KC_NLC_N), \qquad K_{i\ell} = \exp\left(-\frac{\|z_i - z_\ell\|_2^2}{2\sigma_Z^2}\right). \tag{19}$$

For classification, $L_{i\ell} = \mathbb{I}(y_i = y_\ell)$; for regression, $L_{i\ell} = \exp\{-(y_i - y_\ell)^2/(2\sigma_Y^2)\}$. The bandwidth for each RBF kernel is the median of its nonzero pairwise distances, computed from the training observations. A constant input receives a score of zero. PACE uses the full embedding vector when constructing K.

PC. The original sample projection correlation (Liu et al., 2022) averages angles around every sample anchor. For each anchor t, define the $N \times N$ angle matrix

$$a_{i\ell}^{Z,t} = \arccos\left(\frac{(z_i - z_t)^\top (z_\ell - z_t)}{\|z_i - z_t\|_2 \|z_\ell - z_t\|_2}\right), \qquad A^{Z,t} = C_N a^{Z,t} C_N, \tag{20}$$

with the angle set to zero whenever either difference vector vanishes. Define $A^{V,t}$ in the same way from V, and let

$$S_{\mathrm{P}}(Z, V) = \frac{1}{N^3}\sum_{t=1}^{N} \langle A^{Z,t}, A^{V,t}\rangle_F. \tag{21}$$

The projection-correlation utility is

$$\mathcal{U}_{\mathrm{PC}}(Z, Y) = \left\{\frac{S_{\mathrm{P}}(Z, V)}{\sqrt{S_{\mathrm{P}}(Z, Z) S_{\mathrm{P}}(V, V)}}\right\}^{1/2}. \tag{22}$$

The squared version induces the same feature ranking.

HD. Hellinger-distance screening (Wu & Cui, 2024) compares both the conditional CDF and its complement:

$$\begin{aligned}\mathcal{U}_{\mathrm{HD}}(z, Y) = \frac{1}{N}\sum_{i=1}^{N}\sum_{r=1}^{R} \widehat{\pi}_r \Big[&\left\{\sqrt{\widehat{F}_r(z_i)} - \sqrt{\widehat{F}(z_i)}\right\}^2 \\ &+ \left\{\sqrt{1 - \widehat{F}_r(z_i)} - \sqrt{1 - \widehat{F}(z_i)}\right\}^2\Big].\end{aligned} \tag{23}$$

For regression, let $Y^{(\ell)}$ be the categorical response obtained by partitioning Y at empirical quantiles into $\ell + 3$ slices. The implementation averages the resulting classification scores:

$$\mathcal{U}_{\mathrm{HD}}^{\mathrm{reg}}(z, Y) = \frac{1}{L}\sum_{\ell=1}^{L} \mathcal{U}_{\mathrm{HD}}(z, Y^{(\ell)}), \qquad L = \max\{1, \lceil \log N \rceil - 2\}. \tag{24}$$

Empty slices contribute zero. The CDFs are computed cumulatively over the sorted feature observations.

CRU. Conditional-rank screening (Li & Xu, 2024) measures class-specific deviations in mean marginal rank. For a continuous scalar feature X with CDF F, its population utility can be written as

$$\begin{aligned}\mathrm{CRU}(X,Y) &= \sum_{r=1}^{R}\left[\mathbb{E}\{F(X)\mathbb{I}(Y=r)\}-\frac{\pi_r}{2}\right]^2 \\ &= \sum_{r=1}^{R}\pi_r^2\left[\mathbb{E}\{F(X)\mid Y=r\}-\frac{1}{2}\right]^2.\end{aligned} \tag{25}$$

For pooled observations, the component $\theta_r = \mathbb{E}\{F(X)\mathbb{I}(Y = r)\}$ has the unbiased pairwise estimator

$$\widehat{\theta}_r = \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\ell\neq i}\mathbb{I}(z_\ell \le z_i)\mathbb{I}(y_i = r). \tag{26}$$

Substitution gives the empirical score

$$\mathcal{U}_{\mathrm{CRU}}(z,Y) = \sum_{r=1}^{R}\left(\widehat{\theta}_r - \frac{\widehat{\pi}_r}{2}\right)^2. \tag{27}$$

GCor. The GCor score (Jiang et al., 2025) is constructed from the Grothendieck kernel

$$K^Z_{i\ell} = \arcsin\left(\frac{1+z_i^\top z_\ell}{\sqrt{(1+\|z_i\|_2^2)(1+\|z_\ell\|_2^2)}}\right). \tag{28}$$

For $N > 3$, first set the diagonal of K^Z to zero, obtaining $K^{Z,0}$, and apply U-centering:

$$\begin{aligned}\widetilde{K}^Z_{i\ell} &= K^{Z,0}_{i\ell} - \frac{\sum_b K^{Z,0}_{ib}}{N-2} - \frac{\sum_a K^{Z,0}_{a\ell}}{N-2} + \frac{\sum_{a,b}K^{Z,0}_{ab}}{(N-1)(N-2)}, \quad i\neq\ell, \\ \widetilde{K}^Z_{ii} &= 0.\end{aligned} \tag{29}$$

Construct $\widetilde{K}^V$ analogously and set

$$S_{\mathrm{G}}(Z,V) = \frac{\langle \widetilde{K}^Z, \widetilde{K}^V\rangle_F}{N(N-3)}, \qquad \mathcal{U}_{\mathrm{GCor}}(Z,Y) = \left\{\frac{|S_{\mathrm{G}}(Z,V)|}{\sqrt{S_{\mathrm{G}}(Z,Z)S_{\mathrm{G}}(V,V)}}\right\}^{1/2}. \tag{30}$$

The absolute value specifies the implemented treatment of a negative finite-sample cross-covariance estimate.

LR-FFS. The pooled-data LR-FFS score (Qin et al., 2025) uses the largest unweighted class-specific rank contrast. Define the empirical pairwise rank probability using strict comparisons:

$$\widehat{\gamma}_r(z) = \frac{1}{N_r(N-N_r)}\sum_{i\in\mathcal{I}_r}\sum_{\ell\notin\mathcal{I}_r}\mathbb{I}(z_i < z_\ell). \tag{31}$$

The score is

$$\mathcal{U}_{\text{LR-FFS}}(z,Y) = \max_{1\le r\le R}\left|\widehat{\gamma}_r(z) - \frac{1}{2}\right|, \tag{32}$$

with $\widehat{\gamma}_r$ as in Equation (31).

IPC. For IPC (Lin et al., 2026), the implemented transformed-Gram score first standardizes each input coordinate. Let $\widetilde{z}_i$ denote a row after centering and division by its coordinatewise sample standard deviation. Define

$$\sigma_Z^2 = \mathrm{median}_i\,\|\widetilde{z}_i\|_2^2, \qquad g_i^Z = \frac{\widetilde{z}_i}{\sigma_Z\sqrt{\sigma_Z^2 + \|\widetilde{z}_i\|_2^2}}, \qquad M_Z = G_Z G_Z^\top, \tag{33}$$

where G_Z has rows $(g_i^Z)^\top$. Constant coordinates are standardized to zero. Apply the same construction to V to obtain M_V, assigning a score of zero if either scale parameter vanishes. Otherwise, the score is the normalized centered Gram-matrix inner product

$$\mathcal{U}_{\mathrm{IPC}}(Z,Y) = \frac{\langle C_N M_Z C_N, C_N M_V C_N\rangle_F}{\|C_N M_Z C_N\|_F\|C_N M_V C_N\|_F}. \tag{34}$$

Raw and PACE inputs. DC, HSIC, PC, GCor, and IPC admit the vector-valued input $Z = \mathbf{H}_j$ directly. For the scalar utilities SIS, FKF, MV, PSIS, CAVS, HD, CRU, and LR-FFS, the PACE interface applies the utility to each embedding coordinate and averages the resulting scores:

$$\widehat{\omega}_j^{\mathrm{raw}} = \mathcal{U}(X_j, Y), \qquad \widehat{\omega}_j^{\mathrm{PACE}} = \begin{cases} \mathcal{U}(\mathbf{H}_j, Y), & \text{vector-input utility,} \\ d^{-1}\sum_{m=1}^{d} \mathcal{U}(\mathbf{H}_j^{(:,m)}, Y), & \text{scalar-input utility.} \end{cases} \tag{35}$$

Larger scores receive higher ranks, and the selected indices always refer to original feature columns.

C Controlled Simulation Studies

C.1 Raw and PACE versions of feature-screening utilities

The controlled raw/PACE evaluation contains 25 settings. Within each raw/PACE comparison, a replication evaluates the paired utilities on the same generated dataset. The known target support enters the post-ranking MMS calculation. Regular raw/PACE settings use 50 replications. Ultrahigh-dimensional settings use 25 replications.

The primary analysis presents classification utilities in the chronological order of their cited methodological sources: SIS (Fan & Lv, 2008), DC (Li et al., 2012), FKF (Mai & Zou, 2015), MV (Cui et al., 2015), PSIS (Pan et al., 2016), CAVS (Xie et al., 2020), HSIC (Poignard et al., 2022), PC (Liu et al., 2022), HD (Wu & Cui, 2024), CRU (Li & Xu, 2024), GCor (Jiang et al., 2025), LR-FFS (Qin et al., 2025), and IPC (Lin et al., 2026). The regression set contains SIS, DC, HSIC, PC, HD, GCor, and IPC in the same order. The matched design compares each utility on raw values and pretrained representations under the same screening budget.

C.1.1 Classification data-generating processes

The 16 classification settings instantiate six design families.

exp1: strong additive classification ($N = 200$, $p = 3000$, $s = 8$). This design tests whether screening preserves readily detectable class mean-shift signals across different feature distributions. Draw $Y \sim \mathrm{Bernoulli}(1/2)$ and let $\mathcal{S} = \{1, \ldots, 8\}$. Active features receive opposing class shifts of magnitude 0.6; inactive features have identical class-conditional distributions. The three variants use Gaussian, t_3, and log-normal base distributions, respectively.

exp2: nonlinear latent classification ($N = 300$, $p = 3000$, $s = 4$). This design tests recovery of nonlinear signals combining linear and quadratic effects under different feature correlations and noise distributions. Draw $\boldsymbol{X} \sim \mathcal{N}(0, \Sigma)$ and

$$Y^* = 2X_1 + 2X_2 + 2.5(X_3^2 - 1) + 2.5(X_4^2 - 1) + 3\varepsilon, \qquad Y = \mathbb{I}(Y^* > 0).$$

The three variants use identity covariance with Gaussian errors; AR(1) covariance, $\Sigma_{ab} = 0.3^{|a-b|}$, with t_3 errors; and banded covariance with adjacent correlation 0.3 and Gaussian errors, respectively.

exp3: three-class bimodal signal ($N = 300$, $p = 2000$, $s = 10$). This design tests recovery of class-dependent distributional signals with equal class means under varying class imbalance. Features 1–5 carry a class-1 signal and features 6–10 carry a class-2 signal. For each active feature, observations from its signal-bearing class follow $0.5\mathcal{N}(-2, 0.25) + 0.5\mathcal{N}(2, 0.25)$, whereas observations from the remaining classes and all inactive features follow $\mathcal{N}(0, 1)$. The three variants use class probabilities $(0.70, 0.20, 0.10)$, $(0.85, 0.10, 0.05)$, and $(0.33, 0.34, 0.33)$, respectively.

exp4: ultrahigh-dimensional classification ($N = 400$, $s = 15$). This design tests recovery of mean- and scale-shift signals as the number of nuisance features increases. For 12 active features, the class-conditional distributions are $\mathcal{N}(-4.5, 1)$ and $\mathcal{N}(4.5, 1)$. Three additional active features have equal means but class-dependent scales, $\mathcal{N}(0, 1)$ versus $\mathcal{N}(0, 2.5^2)$. All inactive features are standard Gaussian, and $p \in \{10\mathrm{K}, 50\mathrm{K}, 100\mathrm{K}\}$.

exp5: mixed continuous–categorical classification ($N = 300$, $p = 2000$, $s = 6$). This design tests screening of mixed continuous and categorical signals as the number of categorical levels varies. Three active continuous features have opposite class means. Three active categorical features take K levels, are uniform in class 0, and follow a nonmonotone reweighting over a random subset of levels in class 1. The two variants use $(K, \mu, c) = (25, 0.8, 4)$ and $(40, 0.6, 5)$, respectively; inactive features mix Gaussian and categorical noise.

exp6: missing feature values ($N = 300$, $p = 3000$, $s = 4$). This design tests nonlinear signal recovery with correlated features and increasing rates of missing values. These two variants use the latent response form of exp2 with AR(1) covariance, $\Sigma_{ab} = 0.3^{|a-b|}$, and independently mask each feature value with probabilities 0.15 and 0.30, respectively.

C.1.2 Full classification results

Table 2: **Controlled classification: raw versus PACE screening.** Thirteen utilities span exp1–exp6, with parameter sets ordered within each experiment. Panels (a) and (b) apply each utility to raw features and PACE-L_2 representations, respectively. Cells report median MMS (IQR); lower is better.

(a) Raw feature screening

Setting	s	SIS	DC	FKF	MV	PSIS	CAVS	HSIC	PC	HD	CRU	GCor	LR-FFS	IPC
exp1/1 (Gaussian)	8	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
exp1/2 (t_3)	8	9 (22)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
exp1/3 (log-normal)	8	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
exp2/1 (independent)	4	2179 (1063)	6 (4)	21 (34)	2208 (1146)	1782 (1374)	2208 (1146)	4 (2)	18 (22)	4 (1)	2204 (1154)	18 (22)	2209 (1146)	2053 (1116)
exp2/2 (AR(1)/t_3)	4	1736 (1306)	6 (4)	28 (40)	1827 (1446)	1907 (1331)	1825 (1444)	4 (1)	23 (26)	4 (2)	1824 (1441)	21 (27)	1824 (1446)	1872 (1109)
exp2/3 (banded)	4	1664 (1368)	5 (1)	18 (24)	1668 (1331)	1628 (1335)	1668 (1331)	4 (0)	10 (10)	4 (0)	1668 (1336)	12 (10)	1668 (1330)	1671 (1347)
exp3/1 (moderate imbalance)	10	1784 (213)	16 (19)	12 (4)	1842 (272)	13 (18)	1836 (236)	10 (1)	72 (109)	12 (4)	1843 (272)	75 (118)	1826 (224)	1828 (256)
exp3/2 (severe imbalance)	10	1728 (256)	86 (114)	45 (74)	1792 (263)	48 (96)	1774 (200)	26 (31)	278 (294)	86 (128)	1791 (265)	833 (1378)	1782 (219)	1785 (312)
exp3/3 (balanced)	10	1868 (167)	10 (0)	10 (0)	1840 (174)	10 (1)	1845 (160)	10 (0)	10 (0)	10 (0)	1836 (175)	10 (0)	1840 (177)	1836 (216)
exp4/1 (10K features)	15	7826 (2818)	15 (0)	15 (1)	7002 (2489)	4653 (4359)	7002 (2489)	15 (0)	15 (0)	15 (0)	7006 (2496)	15 (2)	7000 (2488)	5388 (3716)
exp4/2 (50K features)	15	32136 (9411)	15 (0)	16 (3)	34780 (13380)	26131 (21731)	34774 (13392)	15 (0)	15 (1)	15 (0)	34840 (13326)	22 (12)	34801 (13394)	30811 (16266)
exp4/3 (100K features)	15	86096 (28678)	15 (0)	22 (16)	87210 (12109)	84574 (27024)	87207 (12109)	15 (0)	15 (1)	15 (0)	87186 (11906)	24 (14)	87199 (12038)	79168 (14886)
exp5/2 ($K = 25$)	6	753 (1455)	30 (96)	12 (48)	654 (1186)	471 (1460)	568 (1144)	118 (569)	75 (304)	28 (130)	752 (1425)	89 (135)	516 (1074)	696 (1458)
exp5/3 ($K = 40$)	6	898 (1144)	85 (234)	60 (202)	884 (1165)	726 (290)	744 (1164)	292 (992)	272 (663)	82 (298)	894 (1330)	144 (1208)	746 (1159)	982 (1251)
exp6/1 (MCAR 15%)	4	1991 (1296)	7 (7)	1200 (971)	1752 (1054)	3000 (0)	1643 (1302)	4 (1)	31 (62)	6 (6)	1908 (1136)	30 (55)	1810 (1116)	1708 (1245)
exp6/2 (MCAR 30%)	4	2128 (1046)	20 (48)	2010 (1125)	1860 (1300)	3000 (0)	1818 (1129)	7 (25)	130 (224)	28 (51)	2112 (1063)	86 (181)	2018 (1034)	2061 (1454)

(b) Matched PACE-L_2 feature screening

Setting	s	SIS	DC	FKF	MV	PSIS	CAVS	HSIC	PC	HD	CRU	GCor	LR-FFS	IPC
exp1/1 (Gaussian)	8	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
exp1/2 (t_3)	8	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
exp1/3 (log-normal)	8	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
exp2/1 (independent)	4	4 (2)	4 (1)	8 (10)	6 (5)	148 (143)	9 (16)	5 (4)	7 (11)	4 (2)	6 (5)	4 (2)	9 (16)	6 (6)
exp2/2 (AR(1)/t_3)	4	4 (1)	4 (0)	5 (5)	5 (4)	130 (175)	11 (14)	4 (1)	6 (5)	4 (1)	5 (4)	4 (1)	11 (14)	5 (3)
exp2/3 (banded)	4	4 (0)	4 (0)	4 (1)	4 (1)	66 (90)	6 (5)	4 (0)	4 (2)	4 (0)	4 (1)	4 (0)	6 (5)	4 (1)
exp3/1 (moderate imbalance)	10	10 (0)	10 (1)	10 (0)	10 (0)	10 (0)	10 (2)	10 (0)	12 (11)	10 (0)	10 (0)	10 (0)	10 (0)	16 (16)
exp3/2 (severe imbalance)	10	10 (1)	21 (30)	10 (2)	12 (6)	11 (5)	16 (11)	18 (15)	72 (89)	10 (2)	12 (6)	16 (28)	16 (11)	84 (126)
exp3/3 (balanced)	10	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
exp4/1 (10K features)	15	15 (0)	15 (0)	15 (0)	15 (0)	62 (51)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)
exp4/2 (50K features)	15	15 (0)	15 (0)	15 (0)	15 (0)	364 (320)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)
exp4/3 (100K features)	15	15 (0)	15 (0)	15 (0)	15 (0)	768 (1642)	16 (2)	15 (0)	15 (0)	15 (0)	15 (0)	15 (0)	16 (2)	15 (0)
exp5/2 ($K = 25$)	6	62 (224)	8 (11)	6 (0)	29 (121)	148 (358)	27 (139)	20 (96)	8 (20)	6 (1)	32 (141)	8 (15)	26 (134)	35 (215)
exp5/3 ($K = 40$)	6	194 (570)	26 (129)	8 (5)	120 (323)	473 (619)	105 (319)	86 (446)	54 (140)	14 (17)	128 (337)	50 (80)	98 (290)	144 (381)
exp6/1 (MCAR 15%)	4	5 (3)	4 (1)	220 (329)	8 (16)	170 (62)	22 (48)	4 (1)	10 (15)	4 (1)	8 (16)	4 (2)	32 (52)	8 (13)
exp6/2 (MCAR 30%)	4	22 (66)	20 (30)	650 (906)	61 (114)	444 (673)	115 (162)	18 (34)	86 (210)	6 (11)	53 (118)	42 (143)	116 (168)	59 (139)

C.1.3 Regression data-generating processes

The nine regression settings instantiate four design families.

exp1: heterogeneous nonlinear regression ($N = 200$, $p = 2000$, $s = 4$). This design tests recovery of heterogeneous nonlinear effects under Gaussian and heavy-tailed response noise. With independent Gaussian features,

$$Y = 2X_1 + 2X_2^2 + 2\exp(X_3) + 2\log|X_4| + 4\varepsilon,$$

where the two variants use Gaussian and t_3 errors, respectively.

exp2: ultrahigh-dimensional regression ($N = 400$, $s = 8$). This design tests recovery of linear and quadratic signals as the number of nuisance features increases. With independent Gaussian features, $Y = 2\sum_{j=1}^{4} X_j + 2\sum_{j=5}^{8}(X_j^2 - 1) + 3\varepsilon$ and $p \in \{10\text{K}, 50\text{K}, 100\text{K}\}$.

exp3: mixed continuous–categorical regression ($N = 400$, $p = 2000$, $s = 4$). This design tests recovery of continuous threshold effects and unordered categorical level effects at different cardinalities. Two active features are Gaussian, and two are categorical with K levels. Each

categorical level-effect vector η_r is an independent permutation of evenly spaced values on $[-1.5, 1.5]$, and

$$Y = 5X_1\mathbb{I}(X_1 > 1.5) + 5X_2 + 5\{\eta_3[X_3] + \eta_4[X_4]\} + 5\varepsilon.$$

The two variants use $K = 5$ and $K = 7$, respectively.

exp4: missing feature values under collinearity ($N = 300$, $p = 3000$, $s = 3$). This design tests signal recovery under the combined effects of strong collinearity and missing feature values. Draw $\boldsymbol{X} \sim \mathcal{N}(0, \Sigma)$ and set $Y = 0.3\{X_1 + X_2^2 + \exp(X_3)\} + 3\varepsilon$. The first variant uses block-AR(1) covariance with $\rho = 0.98$ and an MCAR rate of 0.15; the second uses AR(1) covariance with $\rho = 0.95$ and an MCAR rate of 0.20.

C.1.4 Full regression results

Table 3: **Controlled regression: raw versus PACE screening.** Panels (a) and (b) give matched raw and PACE results, respectively, across exp1–exp4 for each applicable utility in Table 2. Cells report median MMS (IQR); lower is better.

(a) Raw feature screening

Setting	s	SIS	DC	HSIC	PC	HD	GCor	IPC
exp1/2 (Gaussian error)	4	1461 (819)	54 (122)	49 (109)	92 (181)	149 (396)	185 (463)	1482 (854)
exp1/3 (t_3 error)	4	1442 (828)	142 (172)	128 (222)	180 (212)	321 (410)	244 (1224)	1470 (881)
exp2/1 (10K features)	8	8505 (2038)	26 (39)	15 (15)	170 (114)	56 (80)	299 (314)	8993 (2801)
exp2/2 (50K features)	8	41714 (11155)	102 (151)	26 (140)	856 (1089)	226 (434)	1521 (2037)	42398 (9969)
exp2/3 (100K features)	8	88153 (20315)	164 (232)	101 (337)	2531 (2455)	1681 (2440)	4341 (4754)	85691 (26980)
exp3/2 ($K = 5$)	4	197 (835)	40 (211)	74 (154)	108 (315)	134 (309)	374 (1234)	836 (1027)
exp3/3 ($K = 7$)	4	576 (1299)	94 (288)	116 (260)	93 (252)	96 (325)	192 (1124)	750 (1129)
exp4/2 (block-AR, MCAR 15%)	3	18 (30)	14 (44)	28 (196)	24 (110)	26 (116)	46 (217)	56 (248)
exp4/3 (AR(1), MCAR 20%)	3	12 (36)	12 (60)	58 (260)	30 (118)	40 (292)	64 (238)	56 (400)

(b) Matched PACE-L_2 feature screening

Setting	s	SIS	DC	HSIC	PC	HD	GCor	IPC
exp1/2 (Gaussian error)	4	86 (270)	14 (41)	27 (70)	28 (62)	42 (179)	40 (52)	34 (81)
exp1/3 (t_3 error)	4	226 (379)	52 (86)	91 (164)	56 (158)	108 (289)	81 (115)	74 (168)
exp2/1 (10K features)	8	9 (6)	8 (1)	12 (41)	30 (70)	20 (37)	25 (34)	25 (54)
exp2/2 (50K features)	8	20 (18)	11 (7)	31 (110)	163 (522)	97 (403)	77 (174)	133 (340)
exp2/3 (100K features)	8	19 (20)	23 (63)	53 (111)	97 (156)	50 (141)	79 (132)	101 (156)
exp3/2 ($K = 5$)	4	6 (26)	12 (48)	58 (161)	108 (371)	66 (479)	62 (627)	86 (390)
exp3/3 ($K = 7$)	4	10 (28)	10 (32)	27 (99)	96 (258)	37 (198)	75 (325)	98 (271)
exp4/2 (block-AR, MCAR 15%)	3	11 (15)	12 (31)	54 (146)	58 (204)	34 (250)	40 (110)	54 (124)
exp4/3 (AR(1), MCAR 20%)	3	8 (17)	14 (48)	74 (190)	103 (217)	50 (271)	46 (191)	60 (161)

C.1.5 Utility-specific improvement frequency and paired inference

The raw-to-PACE effect depends on the screening utility, so Table 4 reports each utility separately. In classification, SIS records 14 wins and two ties; MV, CAVS, CRU, IPC, and LR-FFS each record 13 wins and three ties. DC records eight wins and eight ties, and HD records six wins and ten ties, so neither utility loses a classification setting. HSIC produces three wins, eleven ties, and two losses. In regression, SIS and GCor win all nine settings, while DC and IPC each win eight. Paired tests are performed separately for each utility, using settings as the analysis units.

Table 4: **Raw-to-PACE improvement frequency and exploratory paired tests.** A win is lower median MMS under PACE; Wilson intervals count ties as non-wins. Median gain is $\log_2(\mathrm{MMS}_{\mathrm{raw}}/\mathrm{MMS}_{\mathrm{PACE}})$, so positive values favor PACE. Exact sign tests exclude ties; one-sided sign-flip tests use mean log gain T. Tests are unadjusted and use settings as analysis units.

Task	Utility	Settings	Win/tie/loss	PACE win share [95% CI]	Median log gain	Sign test (+/nonzero, p)	Sign-flip test (T, p)
Classification	SIS	16	14/2/0	87.5% [64.0, 96.5]	7.512	14/14, < 0.001	$T = 6.425, < 0.001$
	DC	16	8/8/0	50.0% [28.0, 72.0]	0.161	8/8, 0.004	$T = 0.539, 0.004$
	FKF	16	11/5/0	68.8% [44.4, 85.8]	0.776	11/11, < 0.001	$T = 1.069, < 0.001$
	MV	16	13/3/0	81.2% [57.0, 93.4]	7.524	13/13, < 0.001	$T = 6.290, < 0.001$
	PSIS	16	12/4/0	75.0% [50.5, 89.8]	2.441	12/12, < 0.001	$T = 2.685, < 0.001$
	CAVS	16	13/3/0	81.2% [57.0, 93.4]	7.084	13/13, < 0.001	$T = 5.947, < 0.001$
	HSIC	16	3/11/2	18.8% [6.6, 43.0]	0.000	3/5, 0.500	$T = 0.198, 0.188$
	PC	16	9/7/0	56.2% [33.2, 76.9]	0.959	9/9, 0.002	$T = 1.059, 0.002$
	HD	16	6/10/0	37.5% [18.5, 61.4]	0.000	6/6, 0.016	$T = 0.684, 0.016$
	CRU	16	13/3/0	81.2% [57.0, 93.4]	7.523	13/13, < 0.001	$T = 6.321, < 0.001$
	GCor	16	11/5/0	68.8% [44.4, 85.8]	1.280	11/11, < 0.001	$T = 1.558, < 0.001$
	LR-FFS	16	13/3/0	81.2% [57.0, 93.4]	7.086	13/13, < 0.001	$T = 5.932, < 0.001$
	IPC	16	13/3/0	81.2% [57.0, 93.4]	7.178	13/13, < 0.001	$T = 6.015, < 0.001$
Regression	SIS	9	9/0/0	100.0% [70.1, 100.0]	5.037	9/9, 0.002	$T = 5.781, 0.002$
	DC	9	8/0/1	88.9% [56.5, 98.0]	1.737	8/9, 0.020	$T = 1.790, 0.006$
	HSIC	9	6/0/3	66.7% [35.4, 87.9]	0.351	6/9, 0.254	$T = 0.390, 0.117$
	PC	9	5/1/3	55.6% [26.7, 81.1]	1.684	5/8, 0.363	$T = 1.100, 0.078$
	HD	9	7/0/2	77.8% [45.3, 93.7]	1.376	7/9, 0.090	$T = 1.429, 0.008$
	GCor	9	9/0/0	100.0% [70.1, 100.0]	2.209	9/9, 0.002	$T = 2.455, 0.002$
	IPC	9	8/0/1	88.9% [56.5, 98.0]	4.312	8/9, 0.020	$T = 4.718, 0.006$

C.2 Comparison with statistical and machine-learning baselines

C.2.1 Implementation of the comparison methods

The methods below are evaluated in the controlled comparisons or the TALENT study (Table 11). Within each controlled replication, the compared methods receive the same generated dataset.

InterDependence Scores. IDS (Radhakrishnan et al., 2025) expands each feature and the response into six Gaussian-damped polynomial coordinates and takes the largest absolute cross-correlation:

$$\phi_a(u) = \exp(-0.5u^2)\frac{u^a}{\sqrt{a!}}, \qquad \mathcal{U}_{\mathrm{IDS}}(X_j, Y) = \max_{\substack{0\le a,b\le 5\\ 1\le c\le q_Y}} \left|\widehat{\rho}\big(\phi_a(X_j), \phi_b(V^{(:,c)})\big)\right|. \tag{36}$$

Here $\widehat{\rho}$ and V follow Appendix B.1, with $q_Y = 1$ response coordinate for regression and $q_Y = R$ for classification. Each expansion is applied elementwise, and the correlation standardizes its resulting coordinates. Extending the polynomial order beyond 8 yields near-zero terms, so we keep the same setting as the original IDS method.

Neighborhood and fixed-representation comparators. ReliefF and RReliefF (Robnik-Sikonja & Kononenko, 1997) use at most 1,000 reference observations and ten nearest neighbors. Continuous-feature distances are normalized by their training ranges, categorical mismatches contribute unit distance, and the regression variant weights neighbors by rank and response difference. The prespecified representation families comprise degree-8 Legendre polynomials, cubic B-splines, quantile-centered RBFs, 128 random Fourier features (Rahimi & Recht, 2007), and 16-bin quantile indicators. Representations are padded with zeros or truncated to 128 coordinates and paired with the dependence score named in each method label; Legendre-WMC and Spline-WMC use ridge-whitened multiple correlation. UniTree-CV ranks a feature by the five-fold loss reduction of a depth-3 univariate decision tree relative to an intercept-only predictor, with a minimum leaf size of $\max(5, \lceil 0.02N \rceil)$.

For continuous columns, RFF uses $\sqrt{2/d}\cos(\omega_m x + b_m)$ with $d = 128$, Gaussian frequencies $\omega_m \sim \mathcal{N}(0, \sigma^{-2})$, and phases $b_m \sim \mathrm{Unif}(0, 2\pi)$. The bandwidth σ is the median pairwise absolute distance within the training column, with the column's standard deviation and then one as fallbacks if this median is zero; output coordinates are centered. Categorical columns use one-hot indicators, merging levels with fewer than $\max(2, \lceil 0.005N \rceil)$ observations before dimension alignment. Thus the control matches the output dimension but uses a different map from the Transformer; its bandwidth and categorical vocabulary depend on feature values, not responses.

Fitted-model importance. Random forests (Breiman, 2001), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) use 300 trees or boosting iterations and a maximum depth of 6. Random forests use bootstrap sampling and a feature fraction of 0.8; LightGBM uses a learning rate of 0.05 and row and feature fractions of 0.8; CatBoost uses a learning rate of 0.05 and a feature fraction of 0.8. The reported rankings use random-forest mean decrease in impurity, LightGBM split count or total gain, and CatBoost PredictionValuesChange, respectively.

Predictive attribution. LightGBM- and CatBoost-SHAP (Lundberg & Lee, 2017) rank features by their mean absolute TreeSHAP values on at most 1,000 validation observations. TabICL-SHAP fits two estimators, uses a single all-missing row as the masking background, evaluates at most 256 validation observations, and averages absolute permutation-SHAP values across observations and classes. The TALENT comparison additionally includes LOCO (Lei et al., 2018), which uses the increase in validation loss after refitting a random forest without each feature; LOCO-CV averages the same loss increase over five folds. SAGE (Covert et al., 2020), also evaluated on TALENT, uses 32 feature permutations and up to 32 marginal imputations from a training-fold background for fitted LightGBM or CatBoost predictors.

Supervised learned selectors. xRFM-AGOP (Beaglehole et al., 2026) fits one hard-routing tree with a maximum leaf size of 60,000 and ranks features by the diagonal of the training-fold average gradient outer product. LassoNet (Lemhadri et al., 2021) uses one 64-unit hidden layer, hierarchy parameter $M = 10$, 300 dense epochs, 30 path epochs with early stopping, and the official regularization-path importance. GradEnFS (Liu et al., 2024) is classification-only and trains a 512-unit sparse hidden layer for 50 epochs; its accumulated input-gradient scores define the ranking. SAND (Pad et al., 2025) trains for 100 epochs with noise scale 0.5 and gate budget $k_0 = 100$, then ranks features by their projected gate values. Deep Lasso (Cherepanova et al., 2023) uses two 128-unit hidden layers, dropout 0.1, gradient regularization 0.1, and 100 epochs; its validation-fold input-gradient importance defines the ranking.

C.2.2 Setting-level cross-paradigm method landscapes

The setting-level landscapes complement the matched raw/PACE analysis by comparing method families within the controlled simulation settings. The primary raw-to-PACE results are reported in Tables 2–4.

Figures 5–9 place methods on the horizontal axis and MMS on a logarithmic vertical axis. Each figure contains two experiment families, with up to three data-generating parameter settings per family. The plots include IDS as a standalone fixed-expansion comparator and compare PACE with fixed or neighborhood representations, fitted-model importance, predictive attribution, and learned or deep selectors. Crosses at the top of a panel indicate that no finite result is available.

Figures 5 and 6 show the clearest separation in nonlinear and ultrahigh-dimensional classification: the three PACE variants have lower MMS than many fitted and learned selectors, with oracle MMS in several nonlinear and ultrahigh-dimensional settings. Figure 7 shows greater variation in mixed-type and missing-value classification. Figures 8 and 9 show that PACE-SIS and PACE-DC often rank the active set early relative to ambient width, with notable gaps from the oracle in nonlinear and missing-value regression; several fitted alternatives either rank the active set later or return no finite result.

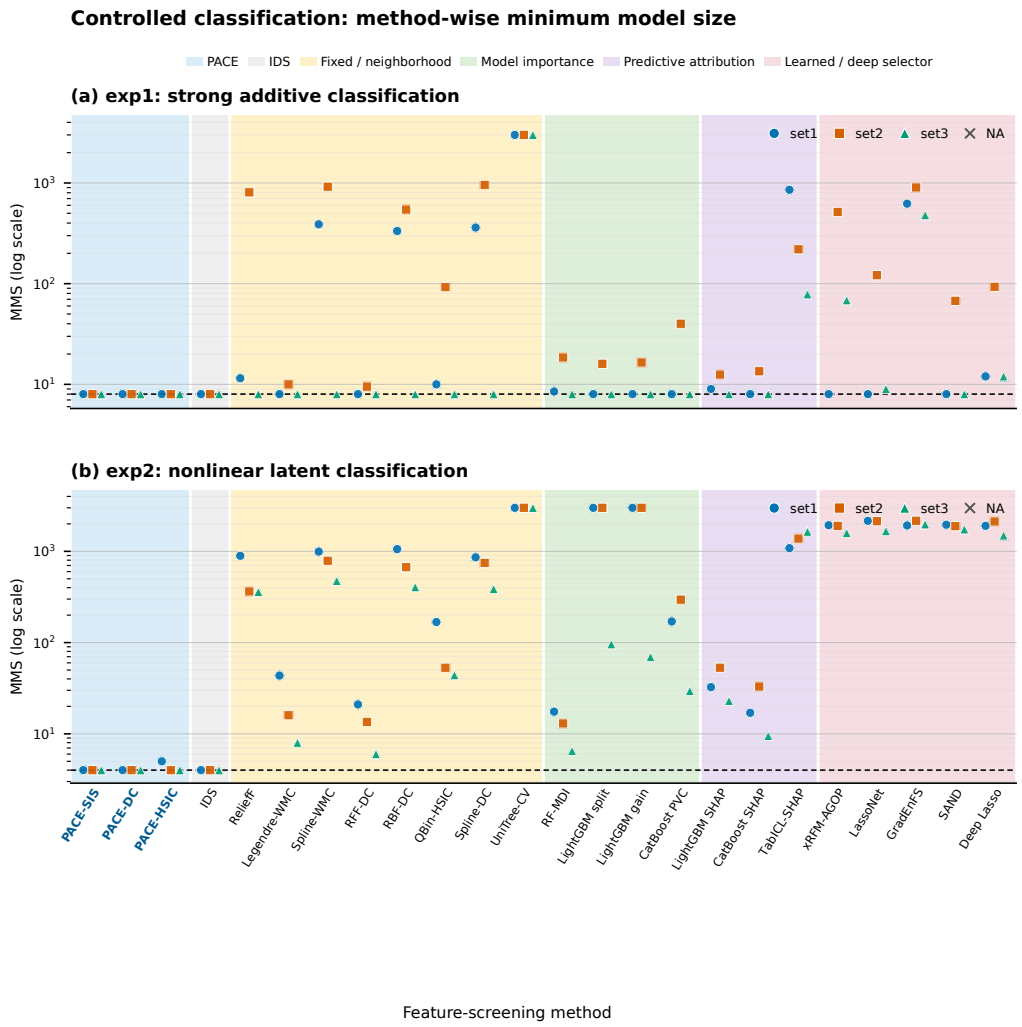


Figure 5: **Controlled classification exp1–exp2: cross-paradigm MMS.** Median MMS by method on a log scale; lower is better. Dashed lines mark oracle support, colors/markers denote parameter sets, shading denotes method families, and top crosses mark missing results. (a) exp1: Gaussian, t_3, log-normal. (b) exp2: independent, AR(1)/t_3, banded.

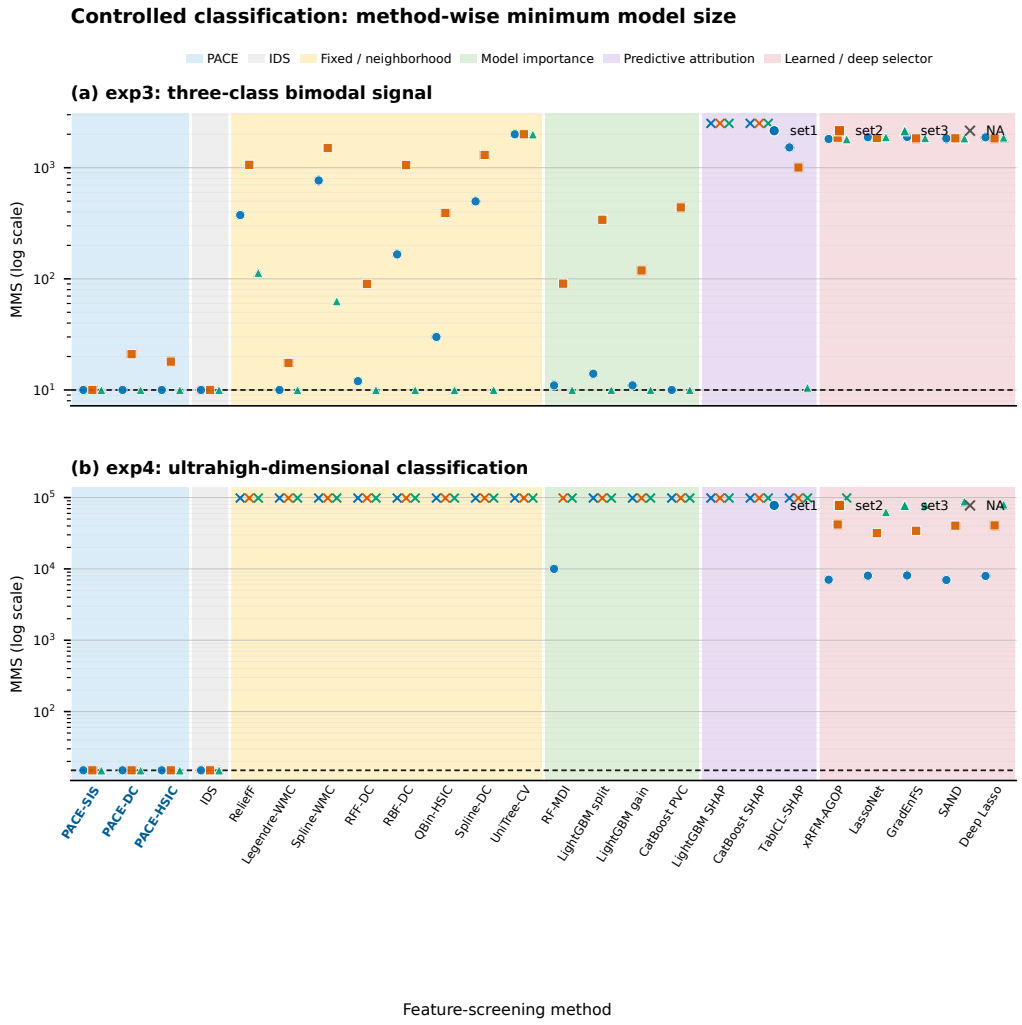


Figure 6: **Controlled classification exp3–exp4: cross-paradigm MMS.** Median MMS by method on a log scale; lower is better. Dashed lines mark oracle support, colors/markers denote parameter sets, shading denotes method families, and top crosses mark missing results. (a) exp3: moderate imbalance, severe imbalance, balanced. (b) exp4: 10K, 50K, 100K features.

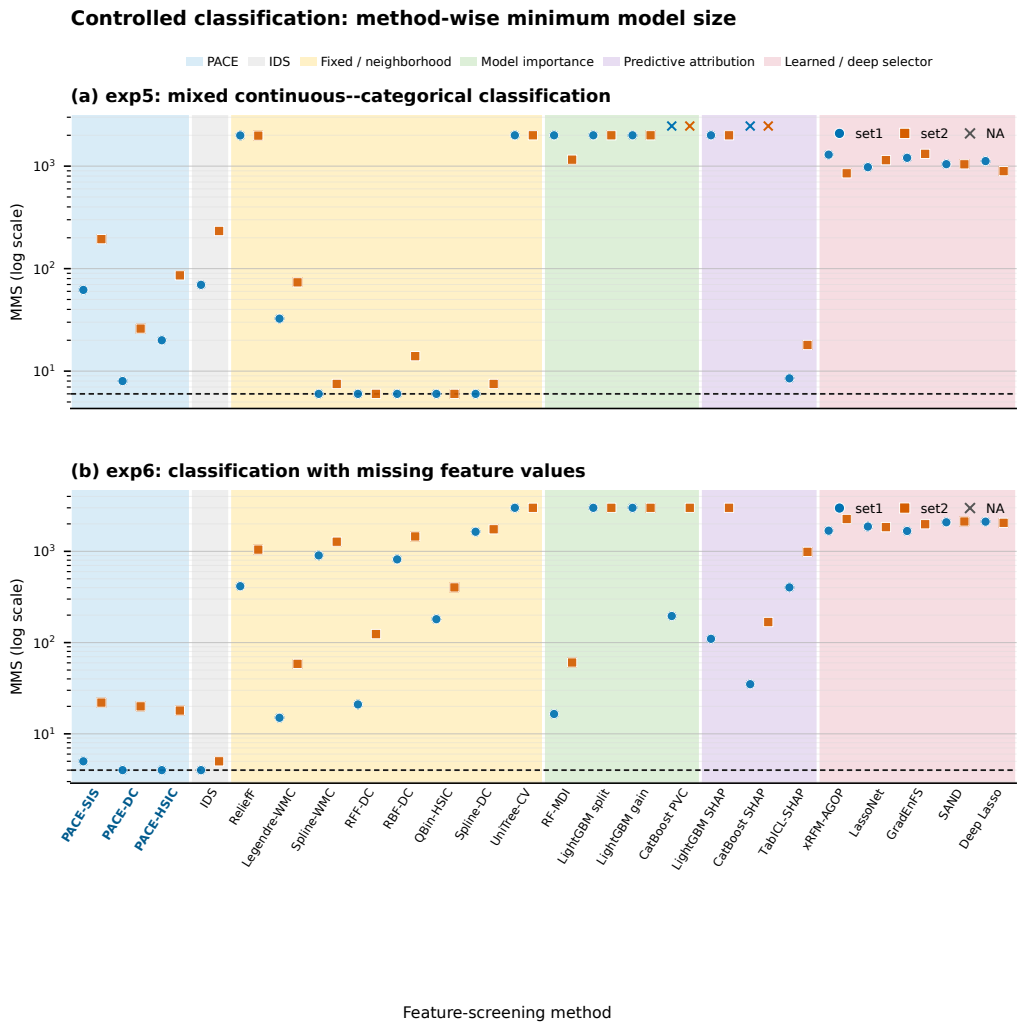


Figure 7: **Controlled classification exp5–exp6: cross-paradigm MMS.** Median MMS by method on a log scale; lower is better. Dashed lines mark oracle support, colors/markers denote parameter sets, shading denotes method families, and top crosses mark missing results. (a) exp5: $K = 25, 40$. (b) exp6: MCAR 15%, 30%.

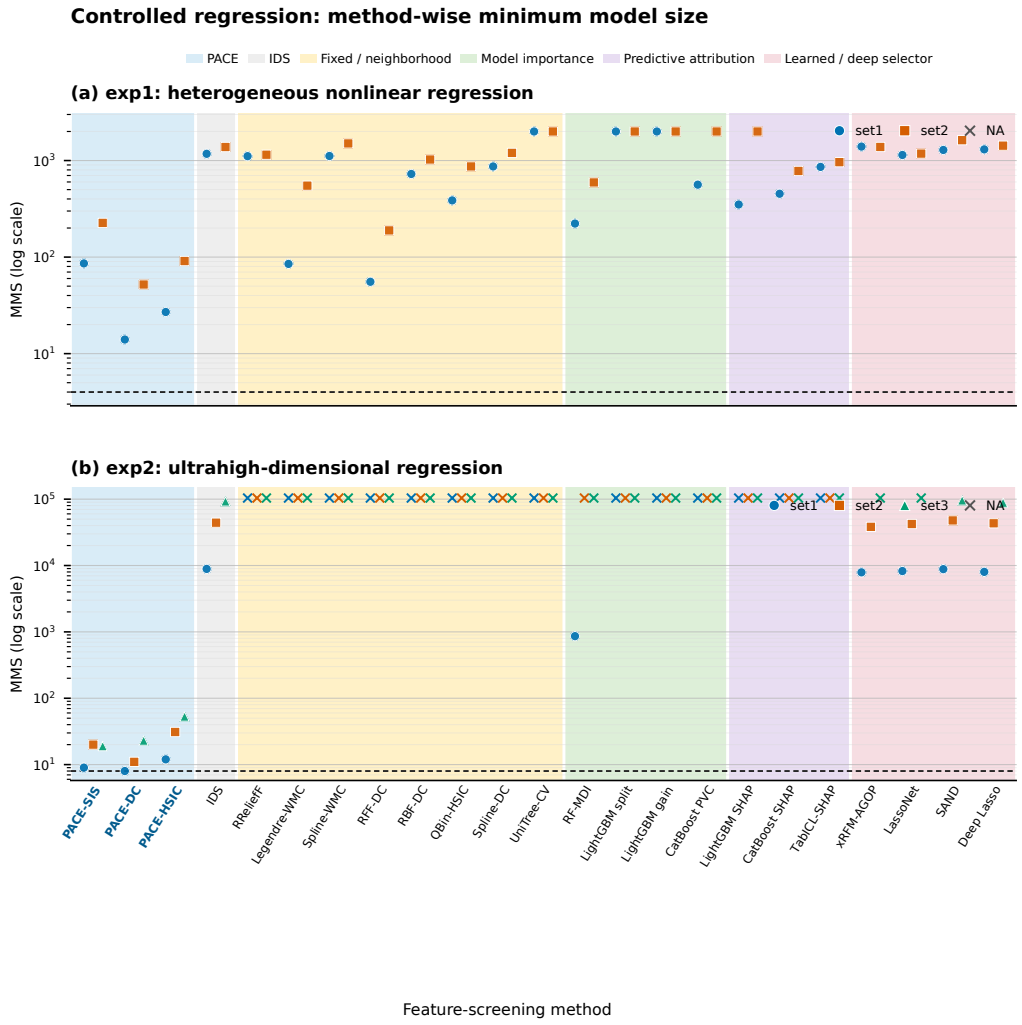


Figure 8: **Controlled regression exp1–exp2: cross-paradigm MMS.** Median MMS by method on a log scale; lower is better. Dashed lines mark oracle support, colors/markers denote parameter sets, shading denotes method families, and top crosses mark missing results. (a) exp1: Gaussian/t_3 error. (b) exp2: 10K, 50K, 100K features.

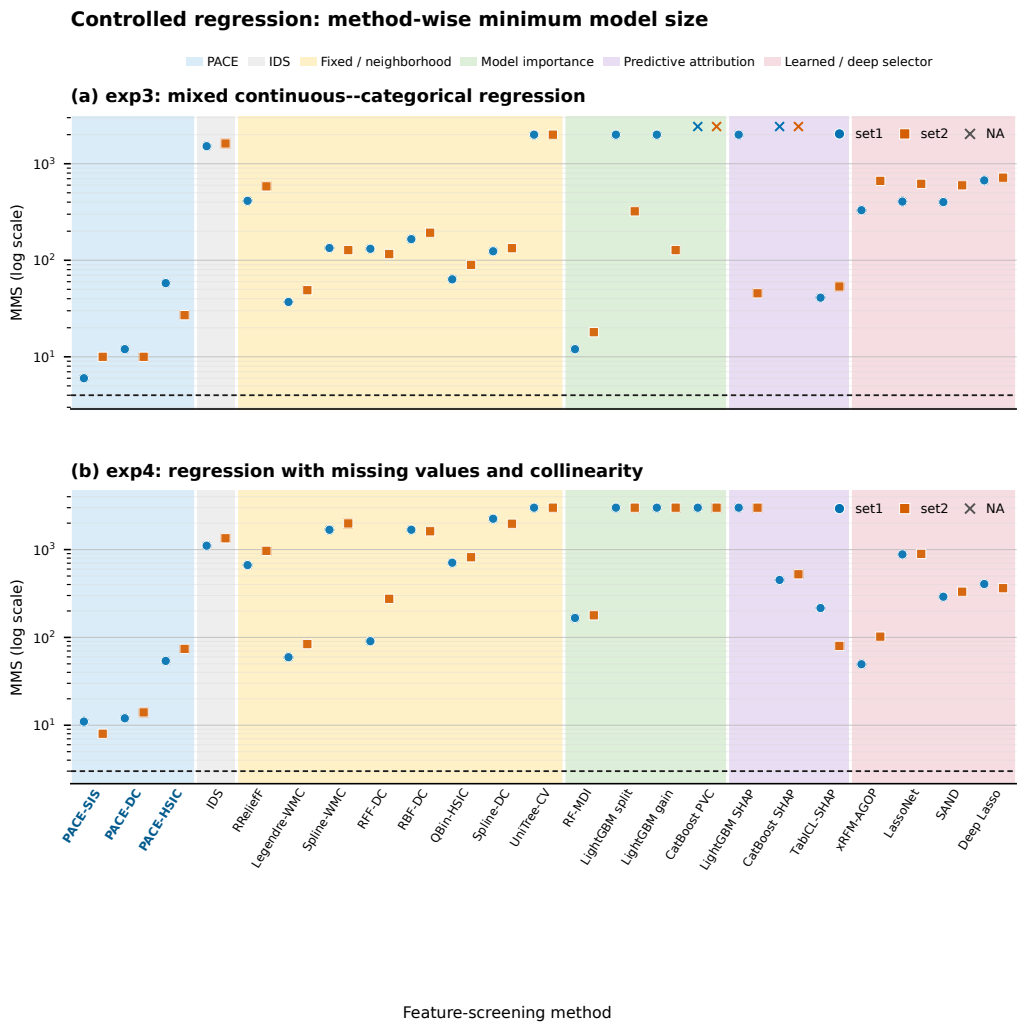


Figure 9: **Controlled regression exp3–exp4: cross-paradigm MMS.** Median MMS by method on a log scale; lower is better. Dashed lines mark oracle support, colors/markers denote parameter sets, shading denotes method families, and top crosses mark missing results. (a) exp3: $K = 5, 7$. (b) exp4: block-AR/MCAR 15%, AR(1)/MCAR 20%.

D Formal Information Boundary

D.1 Proof of Proposition 1

Using the proposition's column notation, let B and C be measurable sets in the representation and response-vector spaces, respectively. Measurability of G_N and $X_j \perp Y$ give

$$\begin{aligned} P\{G_N(X_j) \in B, Y \in C\} &= P\{X_j \in G_N^{-1}(B), Y \in C\} \\ &= P\{X_j \in G_N^{-1}(B)\}P\{Y \in C\}, \end{aligned}$$

which proves $G_N(X_j) \perp Y$.

For i.i.d. observations, marginal feature–response independence yields the column-level condition through

$$P_{X_j,Y} = \prod_{i=1}^{N} P_{X_{ij},Y_i} = \left(\prod_{i=1}^{N} P_{X_{ij}}\right)\left(\prod_{i=1}^{N} P_{Y_i}\right) = P_{X_j}P_Y.$$

If extraction uses randomness $R \perp (X_j, Y)$, the same independence argument gives $G_N(X_j, R) \perp Y$. For the injectivity statement, R must be fixed or observed and conditioned upon. Injectivity conditional on an unobserved random seed does not imply marginal information preservation.

If G_N is almost surely injective between standard Borel spaces, its inverse on the image is measurable up to a null set. Therefore X_j is measurable with respect to $G_N(X_j)$, and the reverse inclusion follows from measurability of G_N; thus $\sigma(G_N(X_j)) = \sigma(X_j)$ up to null sets. This gives a sufficient condition for information preservation.

D.2 Scope of Proposition 1

The proposition establishes that a fixed measurable column operator preserves independence between a feature column and the response vector. Finite-sample rankings can still select inactive features by chance. The empirical study evaluates finite-sample ordering of target and inactive features in the transformed geometry.

E Real-Data Evaluation Details and Additional Results

E.1 Study overview and evaluation protocol

The real-data evaluation uses two complementary constructions. The nuisance-augmented TALENT study estimates matched raw-to-PACE representation effects across binary classification, multiclass classification, and regression. The natural high-dimensional study evaluates end-to-end screening pipelines on tables with continuous features.

For TALENT, the *clean table* is the original table, the *nuisance-augmented table* adds independent $\mathcal{N}(0,1)$ features until its total width is 2,000, and the *screened table* contains the $k = \min(N - 1, 3p_{\text{orig}})$ highest-ranked features, where N is the training-set size. Raw-to-PACE contrasts match the selection budget, downstream learner, base dataset, and repeated split. Binary tasks use AUC, multiclass tasks use macro-averaged one-versus-rest AUC, and regression tasks use RMSE. Three random seeds are used. Representation extraction and utility estimation use at most 2,000 training rows: classification subsampling is stratified and regression subsampling is without replacement, both controlled by the corresponding random seed.

The natural high-dimensional experiment uses ten datasets and five random seeds, retaining the top 30% of features ranked using the training partition. Dataset-level results require all five seeds. For TabICL-SHAP and SAGE-CatBoost, an incomplete result indicates that at least one seed exceeded two hours; completed cells report mean feature-selection time directly in Table 12. TabPFN-Wide and GOTabPFN operate directly on the full table and have complete coverage on the same nine-dataset support. On 20-class COIL20, all six PACE pipelines complete all five seeds; both dedicated wide predictors have incomplete five-seed coverage. Scores are averaged within each dataset, and each dataset receives equal weight in complete-coverage means.

Table 5: **PACE-DC versus raw DC across TALENT.** Original-feature retention is $|\widehat{\mathcal{S}}_k \cap O|/|O|$. Effects are computed from matched splits and screening budgets, averaging over learners within each dataset. For classification, we report the mean gain in binary AUC or multiclass macro-AUC across datasets; for regression, we report the median clean-table-normalized RMSE improvement. Intervals are obtained by bootstrapping base datasets.

Task	Original-feature retention: raw → PACE	Downstream effect [95% interval]
Binary	$0.353 \rightarrow 0.665$	$+0.0773\ [0.0559, 0.1005]$
Multiclass	$0.582 \rightarrow 0.819$	$+0.0641\ [0.0365, 0.0962]$
Regression	$0.388 \rightarrow 0.693$	$+0.0627\ [0.0333, 0.1618]$

E.2 NUISANCE-AUGMENTED TALENT: MATCHED REPRESENTATION EFFECTS

Table 5 shows positive PACE-DC effects on retention and downstream prediction in all three tasks. Tables 6 and 7 report the classification effects by utility and downstream learner.

Table 6: **Learner-conditioned PACE-minus-raw effects: binary classification.** Each cell is the mean AUC gain followed by a 95% base-dataset bootstrap interval. The intervention replaces a raw utility with its PACE counterpart at the same screening budget; positive values favor PACE.

Learner	SIS	DC	HSIC
LightGBM	+0.010 [+0.004, +0.017]	+0.072 [+0.051, +0.096]	+0.008 [+0.003, +0.013]
XGBoost	+0.010 [+0.004, +0.017]	+0.071 [+0.050, +0.095]	+0.009 [+0.005, +0.014]
CatBoost	+0.011 [+0.004, +0.020]	+0.074 [+0.053, +0.098]	+0.004 [+0.001, +0.008]
MLP	+0.007 [+0.003, +0.011]	+0.077 [+0.056, +0.099]	+0.008 [+0.004, +0.012]
RealMLP	+0.012 [+0.006, +0.020]	+0.070 [+0.049, +0.092]	+0.012 [+0.005, +0.020]
TabM	+0.006 [+0.003, +0.010]	+0.078 [+0.057, +0.101]	+0.007 [+0.004, +0.011]
TabPFN v3	+0.011 [+0.005, +0.020]	+0.081 [+0.058, +0.106]	+0.010 [+0.005, +0.016]
TabPFN v2.6	+0.009 [+0.003, +0.017]	+0.084 [+0.061, +0.109]	+0.007 [-0.001, +0.014]
TabICL v2	+0.012 [+0.005, +0.020]	+0.072 [+0.050, +0.095]	+0.014 [+0.007, +0.021]
TabDPT	+0.009 [+0.004, +0.014]	+0.083 [+0.061, +0.108]	+0.013 [+0.008, +0.019]

Table 7: **Learner-conditioned PACE-minus-raw effects: multiclass classification.** Each cell is the mean macro-AUC gain followed by a 95% base-dataset bootstrap interval. The intervention replaces a raw utility with its PACE counterpart at the same screening budget; positive values favor PACE.

Learner	SIS	DC	HSIC
LightGBM	+0.008 [+0.003, +0.014]	+0.062 [+0.033, +0.095]	+0.004 [+0.001, +0.007]
XGBoost	+0.010 [+0.003, +0.020]	+0.064 [+0.036, +0.096]	+0.004 [+0.002, +0.006]
CatBoost	+0.009 [+0.002, +0.020]	+0.064 [+0.036, +0.096]	+0.003 [+0.001, +0.005]
MLP	+0.010 [+0.003, +0.019]	+0.060 [+0.034, +0.089]	+0.010 [+0.005, +0.014]
RealMLP	+0.010 [+0.002, +0.020]	+0.063 [+0.037, +0.093]	+0.004 [-0.0001, +0.009]
TabM	+0.012 [+0.004, +0.022]	+0.061 [+0.033, +0.092]	+0.002 [-0.001, +0.004]
TabPFN v3	+0.013 [+0.004, +0.025]	+0.070 [+0.039, +0.105]	+0.004 [+0.002, +0.006]
TabPFN v2.6	+0.012 [+0.004, +0.025]	+0.058 [+0.030, +0.091]	+0.004 [+0.002, +0.007]
TabICL v2	+0.013 [+0.004, +0.025]	+0.054 [+0.026, +0.087]	+0.005 [+0.002, +0.008]
TabDPT	+0.008 [+0.003, +0.017]	+0.061 [+0.035, +0.090]	+0.005 [+0.002, +0.007]

Sensitivity across downstream model families. Table 8 shows that nuisance features reduce predictive performance more strongly for neural networks and TFMs than for tree ensembles in every task, while PACE-DC improves all nine task–family combinations. Tables 6 and 7 show that these classification gains extend across downstream learners; SIS intervals lie above zero and HSIC point estimates are positive throughout, with three HSIC intervals crossing zero.

Table 9 connects the two effects: nuisance damage and the matched PACE-DC gain are positively correlated in every task–family cell, and seven of the nine intervals exclude zero. Thus, datasets on

Table 8: **Nuisance damage and PACE-DC gains by model family.** Damage is clean-minus-augmented binary/multiclass AUC for classification and the median clean-normalized RMSE increase for regression; PACE-DC effects follow Equation (7). Brackets give 95% bootstrap intervals; positive values mean greater damage or gain.

	Binary		Multiclass		Regression	
Family	Nuisance damage	PACE-DC effect	Nuisance damage	PACE-DC effect	Nuisance damage	PACE-DC effect
Tree	.049 [.039, .059]	+.073 [.051, .095]	.030 [.023, .038]	+.063 [.035, .095]	.114 [.078, .152]	+.069 [.031, .135]
NN	.123 [.111, .136]	+.075 [.054, .097]	.115 [.097, .134]	+.056 [.032, .084]	.431 [.262, .602]	+.038 [.017, .178]
TFM	.114 [.098, .129]	+.071 [.050, .094]	.084 [.068, .099]	+.038 [.016, .065]	.609 [.381, 1.047]	+.061 [.032, .189]

Table 9: **Association between nuisance damage and PACE-DC gain.** Cells report the Spearman correlation between dataset-level nuisance damage and the matched PACE-DC-minus-raw effect, with 95% base-dataset bootstrap intervals. Bold intervals exclude zero.

Task	Tree	NN	TFM
Binary	.214 [.040, .384]	.166 [−.012, .340]	.125 [−.066, .314]
Multiclass	.234 [.018, .449]	.278 [.051, .497]	.334 [.087, .571]
Regression	.340 [.137, .524]	.349 [.024, .632]	.285 [.062, .490]

which the downstream learner is more affected by nuisance width tend to show larger PACE-DC gains.

Contamination recovery. Let Q denote binary AUC or multiclass macro-AUC and R denote RMSE. For each matched learner–dataset–split combination, the restored fraction of nuisance damage is

$$\frac{Q_{\text{screened}} - Q_{\text{aug}}}{Q_{\text{clean}} - Q_{\text{aug}}} \quad \text{or} \quad \frac{R_{\text{aug}} - R_{\text{screened}}}{R_{\text{aug}} - R_{\text{clean}}},$$

where the subscripts denote the clean, nuisance-augmented, and screened tables. The estimand is the matched PACE-minus-raw recovery difference for the same utility. Ratios are evaluated when nuisance damage exceeds ϵ: damage is $Q_{\text{clean}} - Q_{\text{aug}}$ for classification and $(R_{\text{aug}} - R_{\text{clean}})/R_{\text{clean}}$ for regression. Recovery differences are averaged over learners within each base dataset before bootstrap resampling.

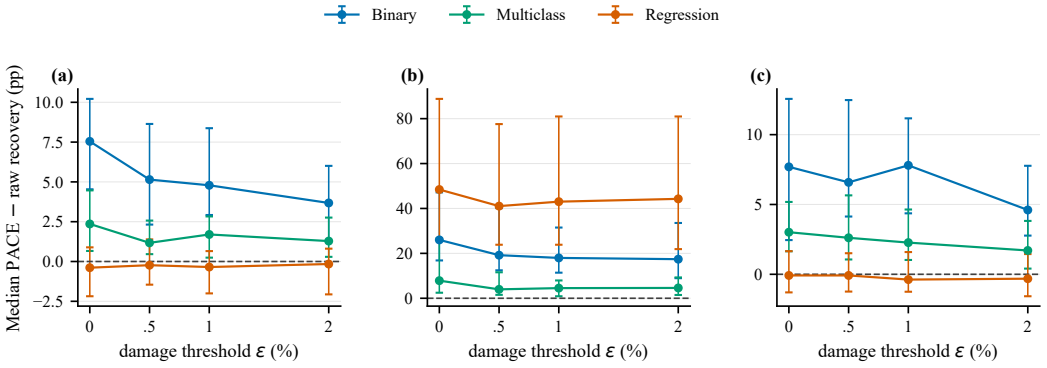


Figure 10: **Contamination recovery across damage thresholds.** Panels (a)–(c) show SIS, DC, and HSIC. Points are median PACE-minus-raw recovery differences with 95% bootstrap intervals; thresholds are absolute classification-score loss or regression loss relative to clean-table RMSE. Intervals lie above zero for all classification utilities and for DC in regression; regression SIS and HSIC remain near zero.

Figure 10 and Table 10 show positive recovery differences for all three utilities in classification. At $\epsilon = 0.01$, PACE-DC recovers an additional 17.97 percentage points of binary nuisance damage and 43.03 points in regression. At this threshold, the median regression differences for PACE-SIS and PACE-HSIC are below half a percentage point in magnitude, and both intervals include zero.

Table 10: **Contamination recovery at $\epsilon = .01$.** Recovery is the fraction of performance loss caused by nuisance augmentation that is restored, computed per matched split–learner–dataset combination before aggregation. Raw/PACE columns report marginal medians. Column 5 reports the median paired recovery difference in percentage points with a 95% bootstrap interval; the final column gives its native-scale counterpart. Ratios are unclipped.

Task	Utility	Raw recovery	PACE recovery	PACE − raw [95% interval]	Native-scale Δ
Binary	SIS	0.382	0.528	+4.79 [+2.92, +8.37]	+0.0027
Binary	DC	-0.111	0.559	+17.97 [+11.40, +31.50]	+0.0126
Binary	HSIC	0.497	0.652	+7.80 [+4.37, +11.17]	+0.0035
Multiclass	SIS	0.723	0.739	+1.70 [+0.24, +2.83]	+0.0009
Multiclass	DC	0.669	0.737	+4.52 [+0.90, +7.93]	+0.0032
Multiclass	HSIC	0.722	0.777	+2.27 [+1.03, +4.64]	+0.0020
Regression	SIS	0.569	0.573	-0.35 [-2.00, +0.65]	0.0000
Regression	DC	-0.020	0.657	+43.03 [+23.84, +80.97]	+0.1642
Regression	HSIC	0.638	0.649	-0.38 [-1.25, +1.59]	0.0000

E.3 TALENT SCREENING COMPARISON: RANK, RETENTION, AND TIME

Main-text Figure 2 presents the performance–time comparisons and the common-support pairwise matrix. Table 11 provides task-specific ranks for the 18 ranked methods and records the five methods exceeding the three-hour screening budget.

Table 11: **TALENT comparison across 23 screening methods.** Ranks are computed within learner–dataset blocks and averaged by dataset; lower is better. Methods exceeding the three-hour screening budget are marked $> 3\,\mathrm{h}$ and excluded from ranking. Retention is the task-equal mean over available panels. Runtimes are in seconds unless labeled in hours; dashes denote unavailable or inapplicable results.

Method	Family	Binary rank	Multiclass rank	Regression rank	Retention (%)	Time
Raw SIS	Raw utility	10.28	9.38	8.86	66.4	0.14
PACE-SIS	PACE	7.46	7.48	9.43	71.5	0.21
Raw DC	Raw utility	13.38	11.34	12.96	45.7	0.98
PACE-DC	PACE	7.55	8.05	7.11	71.2	1.12
Raw HSIC	Raw utility	8.76	8.26	7.38	68.7	11.39
PACE-HSIC	PACE	5.79	6.28	6.82	70.7	6.60
IDS	Fixed comparator	12.58	10.94	12.46	52.5	0.19
RF-MDI	Model-dependent	9.85	9.27	8.33	55.5	52.89
LGB-Split	Model-dependent	8.81	9.01	7.59	57.5	8.41
LGB-Gain	Model-dependent	8.58	9.19	7.77	57.0	8.12
CatBoost-PVC	Model-dependent	8.74	9.18	7.73	55.4	110.85
LGB-SHAP	Predictive attribution	10.23	9.42	7.98	56.3	30.97
CatBoost-SHAP	Predictive attribution	9.19	9.44	7.95	57.0	119.55
TabICL-SHAP	Predictive attribution	–	–	–	–	$> 3\,\mathrm{h}$
LOCO-simple	Predictive attribution	–	–	–	–	$> 3\,\mathrm{h}$
LOCO-CV	Predictive attribution	–	–	–	–	$> 3\,\mathrm{h}$
SAGE-LGB	Predictive attribution	–	–	–	–	$> 3\,\mathrm{h}$
SAGE-CatBoost	Predictive attribution	–	–	–	–	$> 3\,\mathrm{h}$
xRFM-AGOP	Learned selector	9.21	9.72	8.69	66.3	1.69
LassoNet	Learned selector	11.07	10.50	12.67	51.9	46.75
GradEnFS	Learned selector	10.75	12.58	–	48.4	4.42
SAND	Learned selector	9.25	10.48	9.50	59.1	6.08
Deep Lasso	Learned selector	9.54	10.47	9.76	60.0	7.33

Within each downstream learner and base dataset, methods are ranked after repeated-split aggregation; learner-specific ranks are then averaged within each dataset. Improvability and pairwise win rates are computed by averaging over learners within datasets and then weighting tasks equally; GradEnFS is evaluated only on classification. Retention summaries use the available task-specific panels, and timing summaries use each method's available TALENT base datasets.

PACE-HSIC ranks first in all three tasks in Table 11, with average ranks of 5.79 for binary classification, 6.28 for multiclass classification, and 6.82 for regression.

Feature-dimension timing. For Figure 3(c), GPU-based computations use an NVIDIA GeForce RTX 4080 SUPER. LOCO-simple is run with 650 training and 350 validation observations from the same $N = 1{,}000$ synthetic-data construction. Each run fits one full random forest and p reduced forests, scoring every feature by its held-out log-loss increase after deletion. Forests have 300 trees, a maximum depth of six, and a candidate-feature fraction of 0.8. The curve averages two repetitions at each completed width. Timing includes model fitting, prediction, and loss evaluation, and excludes data generation and checkpoint I/O. Triangles indicate completed measurements above the three-hour display ceiling.

E.4 Runtime scaling with sample size

We fix $p = 1{,}000$ and vary $N \in \{100, 200, 500, 1{,}000, 2{,}000, 5{,}000\}$ under the same data-generation and timing protocol as the feature-dimension experiment.

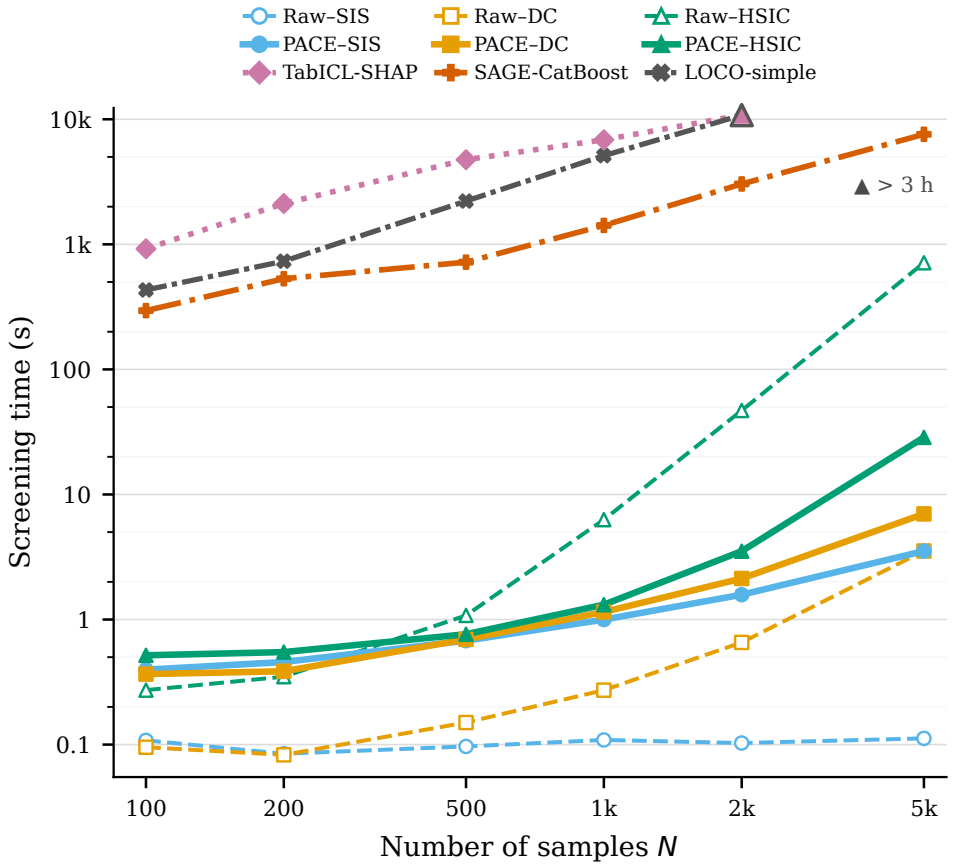


Figure 11: **Screening time versus sample size at $p = 1{,}000$.** Solid and dashed curves denote PACE and raw utilities. Nested triangles mark the $>$ 3-hour budget bounds for TabICL-SHAP and LOCO-simple at $N = 2{,}000$.

PACE remains substantially faster than the attribution methods across the measured sample sizes (Figure 11). At $N = 5{,}000$, PACE-SIS, PACE-DC, and PACE-HSIC take 3.53, 6.99, and 28.61 seconds, respectively, compared with 7,582 seconds for SAGE-CatBoost—speedups of approximately 2,149×, 1,084×, and 265×. TabICL-SHAP and LOCO-simple exceed the three-hour budget at $N = 2{,}000$.

PACE-SIS grows gradually with N, while DC and HSIC become more expensive as pairwise computations increase. DC utility evaluation grows approximately quadratically at larger sample sizes; HSIC grows more steeply because its current implementation includes dense kernel-matrix products. SAGE-CatBoost grows roughly linearly over the larger settings but remains much slower. Distributed computation, as explored in CRU and LR-FFS (Li & Xu, 2024; Qin et al., 2025), offers a further route to accelerating screening on large samples.

E.5 NATURAL HIGH-DIMENSIONAL CLASSIFICATION

The natural high-dimensional panel contains five binary and five multiclass datasets with 500–10,304 features. Screened pipelines retain 30% of the features ranked on the training partition and are evaluated on the same five splits using TabPFN v3 and TabICL v2. Table 12 compares full-table prediction with matched raw and PACE versions of SIS, DC, and HSIC; IDS; learned and attribution-based selectors; LightGBM gain importance; and the end-to-end TabPFN-Wide and GOTabPFN predictors.

Table 12: **Natural high-dimensional classification on ten datasets.** Screeners retain the top 30% of training-ranked features; Wide/Go are end-to-end TabPFN-Wide/GOTabPFN. Entries are five-split mean accuracy (%), ordered by p/n. Means weight all ten datasets equally and require complete coverage; em dashes mark incomplete coverage, and parenthetical T-SHAP/SAGE values give selection time (missing if any split exceeds two hours). All PACE pipelines complete 20-class COIL20. Bold marks the best result for each dataset–backbone pair and each mean row. LNet, Grad, D-L, T-SHAP, and SAGE denote LassoNet, GradEnFS, Deep Lasso, TabICL-SHAP, and SAGE-CatBoost.

Dataset	p/n	Full	SIS		DC		HSIC		IDS	xRFM-AGOP	LNet	Grad	SAND	D-L	T-SHAP	SAGE	LGB-Gain	Wide	Go
			Raw	PACE	Raw	PACE	Raw	PACE											
TabPFN v3																			
orlraws10P	103.0	99.0	97.0	99.0	**100.0**	99.0	98.0	**100.0**	97.0	99.0	99.0	98.0	98.0	98.0	— (> 2 h)	— (> 2 h)	98.0	**100.0**	99.0
leukemia	98.2	92.0	93.3	93.3	93.3	93.3	93.3	93.3	92.0	93.3	92.0	93.3	93.3	92.0	93.3 (40.7 min)	— (> 2 h)	93.3	**94.7**	86.7
GLIOMA	88.7	64.0	64.0	**70.0**	64.0	64.0	60.0	60.0	62.0	60.0	66.0	60.0	66.0	64.0	62.0 (9.8 min)	— (> 2 h)	66.0	64.0	60.0
arcene	50.0	88.5	85.5	89.0	88.0	89.5	90.0	86.5	87.5	83.0	88.0	**92.0**	86.0	88.0	— (> 2 h)	— (> 2 h)	90.0	89.5	89.0
lung	16.3	93.7	94.6	95.6	95.6	94.1	95.6	95.1	95.6	94.1	95.6	93.7	**96.1**	95.1	**96.1** (16.7 min)	— (> 2 h)	95.1	94.6	93.7
warpPIE10P	11.5	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0** (10.7 min)	**100.0** (1.44 h)	**100.0**	**100.0**	**100.0**
BASEHOCK	2.4	89.3	95.5	96.5	96.7	96.0	96.5	96.4	96.0	95.0	—	95.9	95.9	94.8	— (> 2 h)	— (> 2 h)	93.8	**98.3**	97.2
PCMAC	1.7	83.1	90.5	89.5	90.5	91.1	89.8	90.2	89.9	89.7	—	88.7	89.8	91.2	90.6 (1.85 h)	— (> 2 h)	89.0	**92.0**	89.0
COIL20	0.7	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	99.9	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0** (24.0 min)	**100.0** (1.86 h)	**100.0**	—	—
madelon	0.2	89.5	89.9	90.1	90.1	89.9	90.1	90.1	70.8	**90.4**	77.4	89.5	90.1	87.8	90.2 (10.7 min)	90.3 (33.6 min)	90.1	87.4	67.7
Mean	—	89.9	91.0	**92.3**	91.8	91.7	91.3	91.2	89.1	90.5	—	91.1	91.5	91.1	—	—	91.5	—	—
TabICL v2																			
orlraws10P	103.0	**100.0**	98.0	**100.0**	98.0	**100.0**	99.0	**100.0**	98.0	98.0	99.0	99.0	**100.0**	98.0	— (> 2 h)	— (> 2 h)	99.0	**100.0**	99.0
leukemia	98.2	90.7	**97.3**	96.0	94.7	94.7	**97.3**	94.7	**97.3**	94.7	94.7	94.7	96.0	96.0	96.0 (40.7 min)	— (> 2 h)	90.7	94.7	86.7
GLIOMA	88.7	52.0	58.0	64.0	**66.0**	**66.0**	62.0	62.0	62.0	60.0	60.0	62.0	64.0	54.0	56.0 (9.8 min)	— (> 2 h)	56.0	64.0	60.0
arcene	50.0	82.0	88.0	86.5	88.5	87.0	87.0	87.5	84.5	75.0	75.5	85.5	83.0	80.0	— (> 2 h)	— (> 2 h)	84.0	**89.5**	89.0
lung	16.3	94.6	95.1	94.1	95.1	**95.6**	**95.6**	95.1	94.1	94.6	**95.6**	94.6	**95.6**	**95.6**	95.1 (16.7 min)	— (> 2 h)	94.6	94.6	93.7
warpPIE10P	11.5	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0** (10.7 min)	**100.0** (1.44 h)	**100.0**	**100.0**	**100.0**
BASEHOCK	2.4	93.1	98.2	98.0	98.2	98.4	98.2	98.2	98.5	98.2	—	97.9	98.3	**98.6**	— (> 2 h)	— (> 2 h)	97.6	98.3	97.2
PCMAC	1.7	90.7	94.2	**94.3**	93.9	93.7	93.8	93.8	94.0	93.6	—	92.5	94.2	93.9	93.3 (1.85 h)	— (> 2 h)	91.6	92.0	89.0
COIL20	0.7	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0** (24.0 min)	**100.0** (1.86 h)	**100.0**	—	—
madelon	0.2	89.5	**89.9**	89.5	89.4	89.7	**89.9**	89.2	74.8	89.7	77.0	89.4	89.3	88.5	89.8 (10.7 min)	89.6 (33.6 min)	89.5	87.4	67.7
Mean	—	89.3	91.9	92.3	92.4	**92.5**	92.3	92.1	90.3	90.4	—	91.6	92.0	90.5	—	—	90.3	—	—

On the ten evaluated datasets, Table 12 shows that a PACE utility has the highest complete-coverage mean with each downstream TFM, while effects vary by utility and dataset. Under TabPFN v3, PACE-SIS reaches 0.923, compared with 0.910 for raw SIS and 0.899 for the full table, and LightGBM gain and SAND share the highest complete-coverage accuracy among learned or model-based selectors, at 0.915. Under TabICL v2, PACE-DC reaches 0.925, compared with 0.924 for raw DC, 0.923 for raw HSIC, 0.920 for SAND, and 0.893 for the full table.

All six raw/PACE utility pipelines in Table 12 complete all ten datasets for both downstream TFMs. On the nine datasets shared by the dedicated wide predictors, PACE-DC with TabICL v2 reaches 0.917, compared with 0.912 for TabPFN-Wide and 0.869 for GOTabPFN; PACE-SIS with TabPFN v3 reaches 0.914. On 20-class COIL20, the six PACE pipelines complete all five seeds, with mean accuracies ranging from 99.93% to 100%, whereas both dedicated wide predictors are incomplete.

For the same panel in Table 12, PACE-SIS completes feature screening in 30.36 seconds on average across datasets and splits, compared with the two-hour per-split execution cap for TabICL-SHAP and SAGE-CatBoost.

F Representation Ablations: Backbones, Layers, and Utilities

Main-text Figure 4 examines backbone, pretraining, and extraction effects. Panel (a) uses comparisons with raw input and random-weight controls to assess whether dimensional expansion alone is sufficient. Panel (b) compares PACE with a fixed RFF map under matched utilities and reports the ratio of their TALENT runtimes. Panel (c) assesses whether executing the preceding blocks changes performance relative to executing the same block in isolation.

F.1 Pretrained representations versus raw input and matched random-weight controls

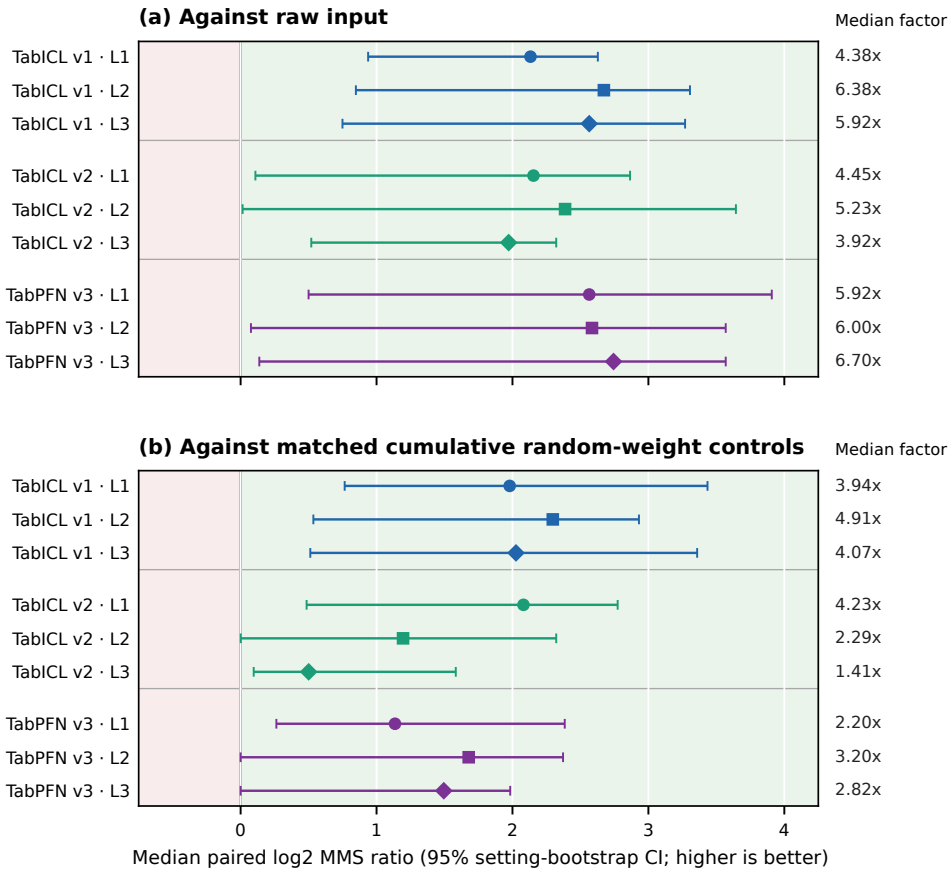


Figure 12: **Pretraining versus raw and random-weight representations.** Points are medians across settings of within-setting median paired log-MMS ratios; a gain of one on the $\log_2$ scale corresponds to a twofold MMS reduction. Exponentiating the TabICL v1-L_2 estimates gives MMS reduction factors of 6.38 relative to raw input (top) and 4.91 relative to its matched cumulative random-weight control (bottom). Bars are descriptive 95% setting-bootstrap intervals.

F.2 Backbone–layer comparisons

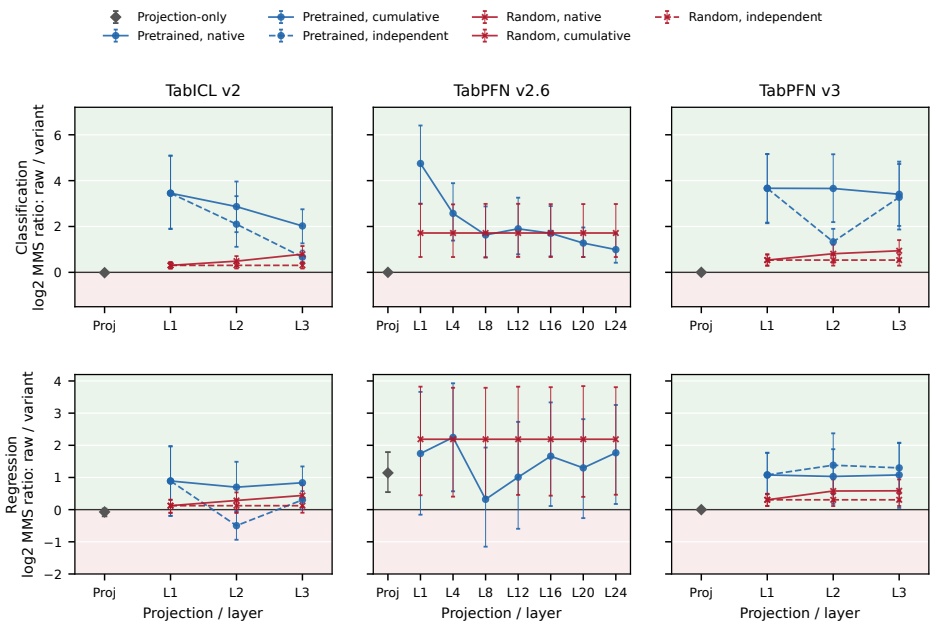


Figure 13: **Backbone and layer effects.** A positive value of $\log_2(\mathrm{MMS}_{\mathrm{raw}}/\mathrm{MMS}_{\mathrm{variant}})$ favors the representation. Points show utility-level log-ratios averaged within each setting and then across settings within each task. Projection-only, pretrained, and matched random-weight variants are shown with descriptive 95% setting-bootstrap intervals.

F.3 Transfer across screening utilities

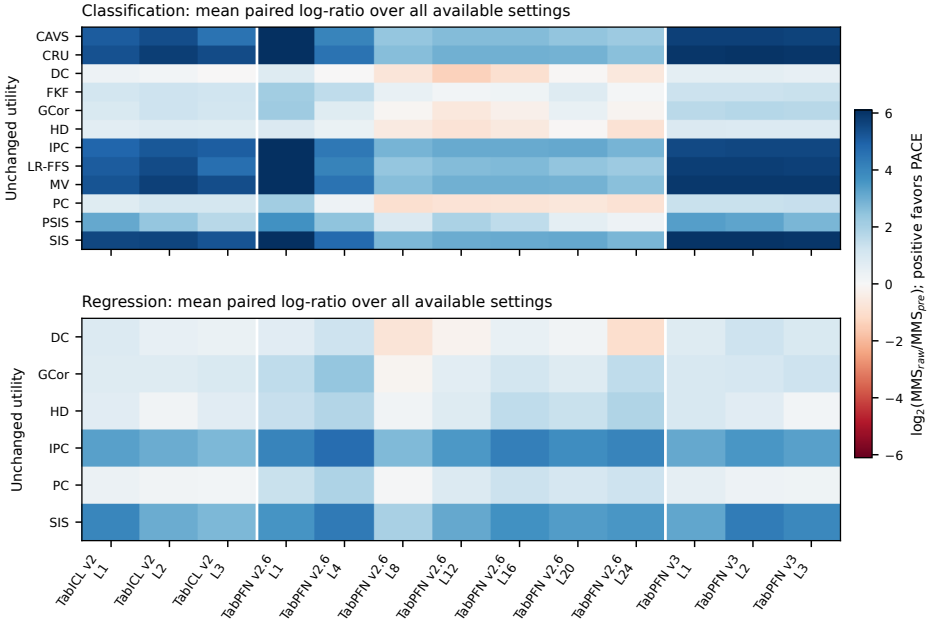


Figure 14: **Transfer across utilities in the controlled ablation setting.** Cells show mean paired $\log_2(\mathrm{MMS}_{\mathrm{raw}}/\mathrm{MMS}_{\mathrm{PACE}})$; blue favors PACE. Classification: CAVS, CRU, DC, FKF, GCor, HD, IPC, LR-FFS, MV, PC, PSIS, SIS. Regression: DC, GCor, HD, IPC, PC, SIS.

F.4 MEAN VERSUS MAXIMUM COORDINATE AGGREGATION

Maximum aggregation uses the embedding coordinate with the strongest empirical association with the response. This can prevent a signal concentrated in a few coordinates from being diluted by averaging, strengthening PACE-SIS in several nonlinear classification settings. The maximum also gives greater weight to large associations caused by noise, increasing sensitivity to isolated extreme scores. Using the notation of Equation (3), we compare

$$\omega_j^{\mathrm{mean}} = \frac{1}{d}\sum_{m=1}^{d} \mathcal{U}_{\mathrm{SIS}}\left(\mathbf{H}_j^{(:,m)}, Y\right), \qquad \omega_j^{\max} = \max_{1\le m\le d} \mathcal{U}_{\mathrm{SIS}}\left(\mathbf{H}_j^{(:,m)}, Y\right).$$

Table 13: **Coordinate aggregation on TALENT.** Mean and Max are across-dataset means after averaging three paired splits. Classification reports AUC (↑); regression reports $R = \mathrm{RMSE}/\mathrm{RMSE}^{\mathrm{clean}}$ for the same learner and split (↓), where $R = 1$ matches clean-table performance. Retention is the percentage of original features retained (↑). The paired t statistic tests Max minus Mean, using datasets as analysis units.

	Binary AUC ↑			Multiclass AUC ↑			Regression R ↓		
Learner	Mean	Max	t	Mean	Max	t	Mean	Max	t
LightGBM	0.806	0.807	+0.20	0.891	0.890	-0.29	1.282	1.975	+1.39
XGBoost	0.802	0.803	+0.83	0.896	0.896	+0.99	1.281	1.978	+1.36
CatBoost	0.814	0.815	+0.24	0.898	0.897	-1.46	1.303	2.000	+1.37
MLP	0.764	0.769	+1.73	0.844	0.852	+3.30	1.359	1.745	+1.23
RealMLP	0.778	0.774	-2.11	0.882	0.883	+0.56	1.333	2.072	+1.21
TabM	0.796	0.798	+1.04	0.880	0.884	+0.93	1.109	1.333	+1.06
TabPFN v3	0.832	0.835	+1.50	0.925	0.925	+0.19	1.492	3.046	+1.22
TabPFN v2.6	0.833	0.834	+0.04	0.924	0.924	+0.67	1.335	2.505	+1.28
TabICL v2	0.831	0.834	+0.98	0.923	0.924	+0.95	1.739	5.142	+1.10
TabDPT	0.804	0.809	+2.16	0.899	0.898	-0.34	1.699	3.392	+1.17
Retention (%) ↑	69.40	66.46	-2.69	83.47	80.28	-4.07	60.43	56.40	-2.37

Classification and retention. Max raises mean binary AUC for nine of ten learners, with the largest gains for MLP and TabDPT (0.005 AUC). Multiclass MLP improves from 0.844 to 0.852; the four TFMs change by less than 0.001 AUC each. Original-feature retention decreases by 2.94, 3.19, and 4.04 percentage points in binary, multiclass, and regression tasks, respectively.

Regression stability. Mean aggregation limits large regression losses. Max increases mean clean-normalized RMSE for every learner; for TabICL v2, mean R rises from 1.739 to 5.142. The largest losses are concentrated in a small subset of datasets.

Controlled support recovery. The simulation ablation uses a fresh set of paired replicates, evaluating both aggregations on the same generated data and embeddings. Table 14 reports median MMS (IQR).

The two aggregation strategies yield broadly similar support-recovery performance across most settings. The largest classification gains from maximum aggregation occur with mixed feature types and missingness: median MMS decreases from 52.5 to 7 and from 115.5 to 13 in exp5, and from 27 to 4 at 30% missingness in exp6, relative to mean aggregation. Aggregation also affects variability: in classification exp2/1, both strategies achieve median MMS 5, while the IQR increases from 2 under mean aggregation to 9.75 under maximum aggregation.

Mean remains the default aggregation. Max improves support recovery in mixed-type and incomplete designs, while mean reduces the influence of isolated extreme scores and provides more stable performance on TALENT.

Table 14: **Mean versus maximum aggregation in controlled simulations.** Entries report median MMS (IQR); lower is better. Experiment and setting identifiers follow Appendix C.

Exp./setting	Mean MMS	Max MMS
Classification		
exp1/1	8 (0)	8 (0)
exp1/2	8 (0)	8 (0)
exp1/3	8 (0)	8 (0)
exp2/1	5 (2)	5 (9.75)
exp2/2	5 (2)	4 (0)
exp2/3	4 (0)	4 (0)
exp3/1	10 (0)	10 (0)
exp3/2	10 (1)	10 (0)
exp3/3	10 (0)	10 (0)
exp4/1	15 (0)	15 (0)
exp4/2	15 (0)	15 (0)
exp4/3	15 (0)	15 (0)
exp5/2	52.5 (323.5)	7 (26.75)
exp5/3	115.5 (359.25)	13 (66.75)
exp6/1	5 (2.75)	4 (0)
exp6/2	27 (37)	4 (3)
Regression		
exp1/2	80.5 (127)	15.5 (71.25)
exp1/3	161.5 (441.5)	93 (361)
exp2/1	9 (3)	9 (4)
exp2/2	13 (12)	16 (37)
exp2/3	26 (64)	53 (505)
exp3/2	8 (17.25)	4.5 (5)
exp3/3	9.5 (24.25)	4 (8.75)
exp4/2	9 (13)	8 (8)
exp4/3	7 (25)	6 (14.25)

G REPRODUCIBILITY

Implementation details. PACE returns rankings over original input-column indices for evaluation by a prespecified downstream learner. Encoder configuration, preprocessing, screening budget, repeated splits, and downstream configuration are fixed within each matched raw/PACE comparison. TALENT uses three random seeds, and the natural high-dimensional experiments use five. If a TALENT training partition exceeds 2,000 rows, embedding and utility estimation use at most 2,000 rows selected by stratified sampling for classification and by sampling without replacement for regression, using the corresponding random seed in both cases. TabPFN-Wide and GOTabPFN are evaluated as end-to-end prediction procedures. Timing results throughout the paper and appendix were recorded on an NVIDIA A100, except for the feature-dimension scaling experiment, which used an NVIDIA GeForce RTX 4080 SUPER shared by all compared methods.

Preprocessing and information access. Utility estimation and feature ranking use training responses; representation extraction uses feature values only. The TALENT vocabulary is constructed from categorical feature levels pooled across the training, validation, and test partitions, yielding transductive, label-free preprocessing.

Data availability. The study uses synthetic data specified in this paper and public datasets from the TALENT collection, subject to the original access conditions and licenses.

Ethics and human subjects. The study uses synthetic and previously released tabular datasets, with no new participant recruitment or collection of personal data. Dataset-specific licenses and use restrictions apply.