arXiv:2610.39506v1 [cs.LG] 29 Oct 2026

Interpretable by Design Tabular Prediction via In-Context Local Linear Learning

Ran Eisenberg
Faculty of Engineering
Bar-Ilan University
Ramat Gan, Israel

Steven Berger
Faculty of Engineering
Bar-Ilan University
Ramat Gan, Israel

Ofir Lindenbaum
Faculty of Engineering
Bar-Ilan University
Ramat Gan, Israel

Abstract

In-context learning (ICL) lets tabular foundation models make predictions without dataset-specific training or fine-tuning. These models achieve strong predictive accuracy, but interpreting how individual features contribute to the prediction remains difficult. In biology, for example, feature contributions can help identify biomarkers or risk genes. Post-hoc interpretability methods assign feature importance scores, but these scores do not indicate what the model actually uses and can lead to explanation errors. We introduce TICL^3 (***T**abular In-Context **L**ocal **L**inear **L**earning*), an interpretable-by-design predictor that uses a tabular foundation model to guide a simple linear prediction rule. TICL^3 first learns a shared linear model to capture dataset-level feature effects. Then, for each query, it retrieves similar training samples using representations from the foundation model and fits a second linear model to the residuals in that neighborhood. The shared model captures broad effects, while the local model corrects errors among similar samples. Their combined coefficients determine each feature contribution to the final query-specific linear score, so the same model makes and explains the prediction. Across 386 classification benchmark–dataset evaluations, TICL^3 trails TabICLv2 by 1.1 percentage points and outperforms CatBoost in 266 evaluations. It improves average support F1 over MAPLE by 4.7 points and achieves the best performance on both real-data selected-feature utility metrics.

1 Introduction

Making predictions on tabular data remains a challenge in machine learning, where interpretability of individual predictions is often as important as predictive accuracy. Classical algorithms, including sparse linear models, generalized additive models (GAMs), and explainable boosting machines (EBMs) [1], provide explicit feature contributions, although their structural constraints may limit predictive performance. Recent tabular foundation models [2, 3] adopt a different paradigm by leveraging labeled examples as context to predict new samples without requiring dataset-specific parameter updates. Related in-context regression approaches have also been extended to reinforcement learning [4]. While these models show strong predictive performance, multiple attention layers complicate attributing individual input feature contributions to each prediction.

Post-hoc interpretability methods such as LIME [5] and SHAP [6] explain predictors after training. Other approaches include gradient-based attribution methods [7, 8], FastSHAP [9], and learned local surrogate models [10]. While these methods can provide useful feature importance scores, they do not use these scores to make predictions and may not fully reflect how the model uses each feature.

TICL^3 is introduced as an interpretable-by-design predictor that leverages a tabular foundation model to infer a simple prediction rule. First, we train a shared ElasticNet [11] to learn a dataset-wide rule based on the foundation model's predictions. For each query, the foundation model's embedding identifies similar training samples, referred to as a neighborhood. We then fit a second ElasticNet to the residuals between the foundation model's predictions and the shared rule within

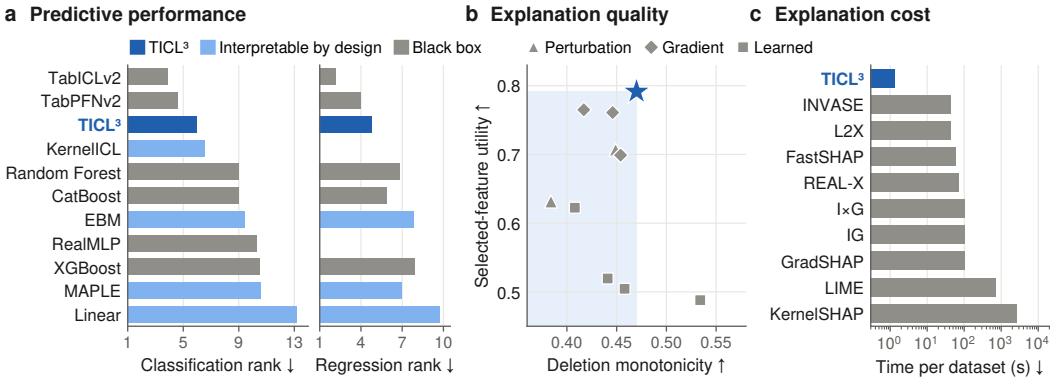


Figure 1: **Summary of results. (a)** Average per-dataset rank in classification and regression over all methods in Tables 2 and 6; a subset is shown. **(b)** Deletion monotonicity and selected-feature utility; the latter averages Gen-SVM and Gen-kmeans, reported separately in Table 5. **(c)** Per-dataset cost for 50 predictions and explanations, including method-specific explainer construction. All methods explain the same TICL^3 (global + residual) predictor, so its setup is excluded for every method. Online latency is similar for TICL^3 (global + residual) and the learned explainers (about 27 ms per query); the difference comes from explainer construction. LIME and KernelSHAP are extrapolated from the fixed 10-query-per-dataset subset.

Table 1: Conceptual positioning of TICL^3. *Interpretable by design*: the explanation is part of the deployed predictor; *explicit additive prediction*: prediction sums inspectable feature terms mapped to input columns; *query-specific deployed rule*: an explicit feature-level rule is fitted or selected per test point (sample weights or post-hoc explanations alone do not qualify); *ICL-guided locality*: an ICL representation defines locality. KernelICL uses ICL-derived query weights but no query-specific feature-level rule. ShapPFN sums per-feature contributions computed by its transformer but deploys no explicit feature-level rule; ExplainerPFN estimates attributions without access to a predictor.

Method	Interpretable by design	Explicit additive prediction	Query-specific deployed rule	ICL-guided locality
TabICLv2	✗	✗	✗	✗
SHAP/LIME on TabICLv2	✗	✗	✗	✗
ExplainerPFN	✗	✗	✗	✗
EBM/NAM	✓	✓	✗	✗
MAPLE	✓	✓	✓	✗
LLSPIN	✓	✓	✓	✗
KernelICL	✓	✗	✗	✓
ShapPFN	✓	✓	✗	✗
TICL^3 (global + residual)	✓	✓	✓	✓

References: TabICLv2 [12]; SHAP/LIME [5, 6]; ExplainerPFN [13]; EBM/NAM [1, 14]; MAPLE [15]; LLSPIN [16]; KernelICL [17]; ShapPFN [18].

this neighborhood. Integrating this local correction with the shared model produces a query-specific linear rule whose coefficients directly determine the final prediction. Figure 2 summarizes this procedure. In TICL^3, the foundation model's learned representation guides sample selection for local correction, while the shared and local coefficients specify each feature's contribution to the prediction.

The central idea of TICL^3 is to use the foundation model's learned representation to inform local correction fitting, with explicit coefficients specifying the prediction process. Table 1 compares TICL^3 with existing methods. In TICL^3, we derive feature contributions directly from the linear model used to compute each class score, rather than from a separate explanation technique. When combined with a baseline, these contributions sum precisely to the class score. The model further employs representations learned through tabular in-context learning (ICL) to select relevant training samples. However, these contributions do not elucidate how individual features influence sample selection or the determination of fitted coefficients. Figure 1 compares prediction accuracy, explanation quality, and computational cost.

For each query, we form a single linear model that is composed of the shared and local models. The coefficients, together with the processed feature values, determine each feature's contribution to the prediction score for a class, and we map these contributions back to the original columns of the table. We use an ElasticNet regularizer to encourage a sparse coefficient vector. The foundation model does not make the final prediction; it provides targets to fit the shared prediction rule and selects the rows to use for each query. The final prediction is made by the explicit prediction rule described above. The error in this local approximation can be bounded in terms of (i) smoothness of the residual, (ii) context shift, (iii) the radius of the neighborhood in the representation space, (iv) distance of the query to the weighted feature centroid, and (v) the amount of regularization.

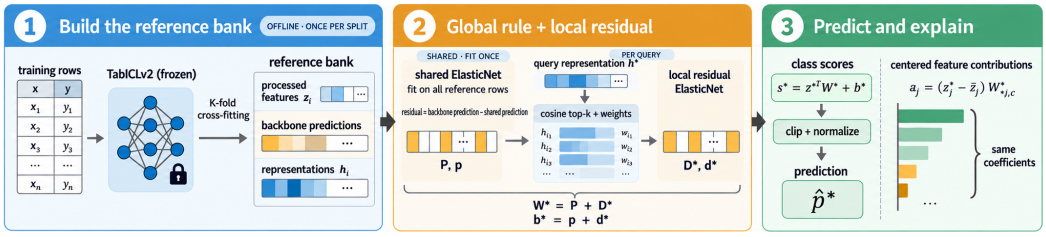


Figure 2: **Overview of TICL^3 (global + residual).** Offline we use a frozen TABICLV2 backbone with cross-fitting to build a reference bank of processed features, class probability targets and learned representations. We fit a shared ElasticNet model to the bank targets. For each query, we then use its representation to select similar reference rows and fit a second ElasticNet model to the shared model's remaining errors on those rows. We add the shared model and the local correction to give a single linear model for the query. We clip and normalize the class scores from this to make the final prediction. We also use the model coefficients and the centered feature values to compute the contribution of each feature to the class scores.

2 Background and Related Work

We group related work into three categories: predictive models, post-hoc explanation methods, and interpretable-by-design models.

Predictive models. Tabular ICL models such as TabPFN [2], TabPFNv2 [19], and TABICLV2 [12] demonstrate strong predictive performance, as do task-specific neural models like TabM and TabR [20, 21]. Some approaches combine neural and tree-based encoders to learn tabular representations [22]. While these models learn effective representations and prediction functions, they do not provide explicit feature contributions that directly determine predictions.

Post-hoc explanation methods. LIME [5], SHAP [6], gradient attributions [7, 8], and learned explainers clarify the outputs of a fixed predictor. ExplainerPFN [13] uses a pretrained tabular foundation model to estimate feature attributions for a new dataset without access to the explained predictor. These methods offer feature attributions or local approximations, but their explanations do not determine the final prediction.

Interpretable-by-design models. Lasso [23], EBMs [1], and NAMs [14] use explicit feature contributions within a shared model. LOESS [24] and MAPLE [15] fit local linear rules, defining neighborhoods or sample weights by input distances or tree-based representations. LSPIN and its locally linear variant LLSPIN [16] train a gating network that selects a sparse feature subset for each sample; with a linear prediction network, LLSPIN deploys a sample-specific sparse linear rule. ShapPFN [18] makes a tabular foundation model's prediction additive over per-feature contributions, although each contribution is computed by the transformer rather than by an explicit feature-level rule. Context-dependent feature-selection methods allow selected variables to vary with the observed context [25]. However, these methods do not leverage context-conditioned representations from a pretrained tabular ICL model to fit a query-specific prediction rule.

TICL^3 uses representations from a tabular ICL model to select a neighborhood for each query and fits a local linear correction to the residuals of a shared linear predictor. The combined coefficients directly determine the prediction. For more details on related work, see Appendix F.

3 Method

For a new test query, TICL^3 constructs an adaptive linear prediction rule using representations from a tabular foundation model. The representations allow TICL^3 to identify the training samples that are most relevant to the test query. The coefficients, together with the processed feature values and intercept, determine the class scores and feature contributions. We consider two variants of TICL^3, TICL^3 (global + residual) and TICL^3 (regional anchors). In TICL^3 (global + residual), we first fit a shared linear prediction model to capture the global effects of features. Then, for each test query, we use the representations to find training samples similar to the query and fit a second linear prediction model to the shared model's residuals on these samples. The final prediction rule for the test query consists of the sum of the shared linear model and the query-specific linear model. In TICL^3 (regional anchors), we avoid fitting a new prediction model for each test query by fitting regional prediction models upfront and combining them based on their similarity to the query. See

section 3.3 for details. We use the terms "linear model" and "linear predictor" to refer to linear models with an intercept that is not regularized.

Backbone outputs and reference bank. Suppose we have a labeled training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^{D_{\text{raw}}}$ is the i-th sample's feature vector and $y_i \in \{1, \dots, C\}$ is its class. We apply a deterministic mapping $\phi : \mathbb{R}^{D_{\text{raw}}} \to \mathbb{R}^D$ to obtain the processed features $\mathbf{z}_i = \phi(\mathbf{x}_i)$. In our experiments, ϕ encodes categorical features using ordinal encoding, imputes missing values with the corresponding training-set means, and applies scalar transformations fitted on the training set. We do not expand features, so each retained coordinate of $\mathbf{z}_i$ corresponds to a single coordinate of $\mathbf{x}_i$. See Appendix E.1 for details about the transformations and how we map feature contributions and recovered supports back to the original columns.

For each sample, our tabular foundation model (backbone) outputs a predicted class-probability vector $\widetilde{\mathbf{y}}_i \in \Delta^{C-1}$ and a representation $\mathbf{h}_i \in \mathbb{R}^{d_h}$, where $\Delta^{C-1} := \{\mathbf{u} \in \mathbb{R}^C_{\geq 0} : \mathbf{1}^\top \mathbf{u} = 1\}$ is the probability simplex and $\mathbf{1}$ is the all-ones vector. All vectors are column vectors. We use $\|\cdot\|_2$ to denote the ℓ_2 norm of a vector.

We use stratified cross-fitting to generate the backbone outputs so that a sample's class label is not included in the labeled context used to compute its class-probability vector and representation. Specifically, we use the labeled context from all other folds when generating the backbone outputs for samples in a given fold. We then aggregate the outputs from all folds to construct a reference bank $\{(\mathbf{z}_i, \widetilde{\mathbf{y}}_i, \mathbf{h}_i)\}_{i=1}^N$, in which each training sample's prediction and representation are generated without its own label in the context. See Appendix B for more details.

Given a new test input $\mathbf{x}^*$, we transform its features using the same mapping $\mathbf{z}^* = \phi(\mathbf{x}^*)$ and generate its representation $\mathbf{h}^*$ using the tabular foundation model with the entire training set as its labeled context. We fit the mapping ϕ using the entire training set and use this fitted mapping for all samples. We only cross-fit the backbone predictions and representations for the training samples.

3.1 Context-Conditioned Neighborhood

We use the backbone representations to identify training rows most similar to the test query. Cosine similarity is computed as $s_i(\mathbf{x}^*) = \mathbf{h}_i^\top \mathbf{h}^* / (\|\mathbf{h}_i\|_2 \|\mathbf{h}^*\|_2)$, and the k most similar rows are selected. For a neighborhood size $k \in \mathbb{N}_+$, $\mathcal{N}^* \subseteq \{1, \dots, N\}$ denotes the indices of the $k' = \min(k, N)$ largest similarities. This approach maintains a fixed number of rows for each local fit, rather than varying the count based on a similarity threshold. Greater weight is assigned to rows that are more similar to the query. For a temperature parameter $\tau > 0$, we define

$$\widetilde{w}_i^* = \frac{\exp(s_i(\mathbf{x}^*)/\tau)}{\sum_{j \in \mathcal{N}^*} \exp(s_j(\mathbf{x}^*)/\tau)}, \qquad i \in \mathcal{N}^*. \tag{1}$$

The weights are strictly positive and sum to one. Smaller τ assigns greater weight to the closest neighbors, whereas larger values result in more uniform weighting.

3.2 Shared Global Predictor

We fit a global predictor to capture the part of the backbone's predictions that can be explained by a single linear model across the entire dataset. Given the coefficient matrix $\mathbf{A} \in \mathbb{R}^{D \times C}$, we use the ElasticNet regularizer $\Omega_{\alpha,\rho}(\mathbf{A}) = \alpha\rho\|\mathbf{A}\|_1 + \alpha(1-\rho)\|\mathbf{A}\|_F^2/2$ with regularization strength $\alpha > 0$ and mixing parameter $\rho \in [0, 1]$. Here, $\|\mathbf{A}\|_1$ denotes the sum of the absolute values of its entries. The ℓ_1 term encourages sparsity, while the ℓ_2 term penalizes large coefficients. Specifically, we learn the dataset-wide model $(\mathbf{P}, \mathbf{p})$ by solving

$$(\mathbf{P}, \mathbf{p}) = \arg \min_{\mathbf{A} \in \mathbb{R}^{D \times C},\, \mathbf{a} \in \mathbb{R}^C} \frac{1}{2N} \sum_{i=1}^N \|\mathbf{A}^\top \mathbf{z}_i + \mathbf{a} - \widetilde{\mathbf{y}}_i\|_2^2 + \Omega_{\alpha,\rho}(\mathbf{A}),$$

where $\mathbf{P} \in \mathbb{R}^{D \times C}$ is the coefficient matrix, $\mathbf{p} \in \mathbb{R}^C$ is the unregularized intercept vector, and $\mathbf{P}^\top \mathbf{z} + \mathbf{p}$ is the prediction from the dataset-wide linear model for the processed feature vector $\mathbf{z}$. This model is fitted once per training split and shared across all test queries.

3.3 Query-Specific Residual Correction

The dataset-wide model may be unable to capture all of the variation in the backbone's predictions. To account for this, we train a second model to predict the residual $\mathbf{r}_i = \widetilde{\mathbf{y}}_i - (\mathbf{P}^\top \mathbf{z}_i + \mathbf{p})$ for the

reference rows. For a test query $\mathbf{x}^*$, the query-specific residual correction $(\mathbf{D}^*, \mathbf{d}^*)$ is learned by solving

$$(\mathbf{D}^*, \mathbf{d}^*) = \arg \min_{\mathbf{D}\in\mathbb{R}^{D\times C},\, \mathbf{d}\in\mathbb{R}^C} \frac{1}{2} \sum_{i\in\mathcal{N}^*} \widetilde{w}_i^* \|\mathbf{D}^\top \mathbf{z}_i + \mathbf{d} - \mathbf{r}_i\|_2^2 + \Omega_{\alpha,\rho}(\mathbf{D}), \tag{2}$$

where $\mathbf{d}^*$ is the unregularized intercept vector. This model is fitted separately for each test query. We use $\mathbf{W}^* = \mathbf{P} + \mathbf{D}^*$ and $\mathbf{b}^* = \mathbf{p} + \mathbf{d}^*$ to define the final linear prediction rule, where $\mathbf{P}$ contains the shared coefficients and $\mathbf{D}^*$ contains the query-specific correction learned from the query's neighborhood. This rule outputs a vector with one score per class, given by $(\mathbf{W}^*)^\top \mathbf{z}^* + \mathbf{b}^*$.

We next analyze the error in approximating the backbone's predictions by fitting the residuals left by the dataset-wide model. Let $g : \mathbb{R}^D \to \mathbb{R}^C$ be the dataset-wide linear model, where $g(\mathbf{z}) = \mathbf{P}^\top \mathbf{z} + \mathbf{p}$. We let $\boldsymbol{\eta}(\mathbf{x}; \mathcal{C})$ and $\mathbf{h}(\mathbf{x}; \mathcal{C})$ denote the backbone's class-probability vector and learned query representation, respectively, given a context of labeled examples $\mathcal{C}$. We assume that the representations are nonzero. For each reference row i, we let $\boldsymbol{\eta}_i = \boldsymbol{\eta}(\mathbf{x}_i; \mathcal{C}_i) = \widetilde{\mathbf{y}}_i$ and $\mathbf{h}_i = \mathbf{h}(\mathbf{x}_i; \mathcal{C}_i)$, where $\mathcal{C}_i$ consists of the training examples outside the fold containing $\mathbf{x}_i$. For a test query $\mathbf{x}^*$, we let $\boldsymbol{\eta}^* = \boldsymbol{\eta}(\mathbf{x}^*; \mathcal{C}^*)$ and $\mathbf{h}^* = \mathbf{h}(\mathbf{x}^*; \mathcal{C}^*)$, where $\mathcal{C}^*$ consists of all training examples. Let $\mathbf{r}_i = \boldsymbol{\eta}_i - g(\mathbf{z}_i)$ and $\mathbf{r}^* = \boldsymbol{\eta}^* - g(\mathbf{z}^*)$. We normalize the representations as $\widehat{\mathbf{h}}_i = \mathbf{h}_i / \|\mathbf{h}_i\|_2$ and $\widehat{\mathbf{h}}^* = \mathbf{h}^* / \|\mathbf{h}^*\|_2$. We use $\Delta_k(\mathbf{x}^*) = \max_{i\in\mathcal{N}^*} \|\widehat{\mathbf{h}}_i - \widehat{\mathbf{h}}^*\|_2$ for the representation-space radius of the retrieved neighborhood and $\overline{\mathbf{z}}_k = \sum_{i\in\mathcal{N}^*} \widetilde{w}_i^* \mathbf{z}_i$ for its weighted centroid in processed-feature space.

Theorem 1 (Residual approximation bound)**.** *Assume $\alpha(1-\rho) > 0$ and that there are constants $L_r \geq 0$ and $\epsilon_{\text{ctx}} \geq 0$ such that each retrieved residual is close to the query residual:*

$$\|\mathbf{r}_i - \mathbf{r}^*\|_2 \leq L_r \|\widehat{\mathbf{h}}_i - \widehat{\mathbf{h}}^*\|_2 + \epsilon_{\text{ctx}}, \qquad i \in \mathcal{N}^*.$$

Here L_r measures residual smoothness in representation space, and ϵ_{ctx} absorbs the shift between outputs produced under the out-of-fold contexts $\mathcal{C}_i$ and under the full context $\mathcal{C}^*$. Then the linear prediction before clipping and normalization,

$$\widehat{\boldsymbol{\eta}}^*_{\text{lin}} = g(\mathbf{z}^*) + (\mathbf{D}^*)^\top \mathbf{z}^* + \mathbf{d}^*,$$

satisfies

$$\|\boldsymbol{\eta}^* - \widehat{\boldsymbol{\eta}}^*_{\text{lin}}\|_2 \leq \big(L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\text{ctx}}\big) \left(1 + \frac{\|\mathbf{z}^* - \overline{\mathbf{z}}_k\|_2}{\sqrt{\alpha(1-\rho)}}\right). \tag{3}$$

The proof can be found in Appendix D. When ϵ_{ctx} is 0, the condition says that the Euclidean norm of the difference between the residual of the query and of each retrieved row is bounded by the distance between their normalized representations, multiplied by L_r. ϵ_{ctx} allows this residual difference to be larger, to account for differences arising from the labeled contexts used to obtain the backbone outputs. When $\mathbf{D}^*$ is all zeroes, the correction is the weighted mean of the neighborhood residuals $\overline{\mathbf{r}}_k$, and the error is bounded by $L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\text{ctx}}$. The theorem doesn't show that the slope will make the correction better, but it bounds the additional error that the slope may introduce through the ElasticNet regularization. The fitted slope, combined with the shared coefficients, can be used to compute feature contributions. We can set $g \equiv \mathbf{0}$ to get a similar bound for approximating the backbone outputs directly, under the corresponding smoothness assumption on those outputs. Using the shared model might make the term $L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\text{ctx}}$ smaller by reducing L_r or ϵ_{ctx}, but it doesn't necessarily do so, because the shared model is linear with respect to the processed features, while L_r describes variation with respect to distance in the representation space. Also, although we use cross-fitting so that the label for the row is not used in its context, we don't actually bound ϵ_{ctx}. Finally, the bound is only for the linear scores; next we describe how to clip and normalize them.

Prediction and feature contributions. Given a test query, we first obtain class scores for the query, given by $\widehat{\mathbf{z}}^*_{\text{cls}} = (\mathbf{W}^*)^\top \mathbf{z}^* + \mathbf{b}^* = \widehat{\boldsymbol{\eta}}^*_{\text{lin}}$, and then we clip the scores to $[0, 1]$ and normalize them to sum to one to obtain predicted probabilities, $\widehat{\mathbf{p}}^* = \text{clip}_{[0,1]}(\widehat{\mathbf{z}}^*_{\text{cls}}) / (\mathbf{1}^\top \text{clip}_{[0,1]}(\widehat{\mathbf{z}}^*_{\text{cls}}))$; if the normalization constant is zero, we set $\widehat{\mathbf{p}}^* = \mathbf{1}/C$.

We explain the model using the same set of coefficients: for each processed feature j and class c, the feature's contribution to the class score is defined as $a_{j,c}(\mathbf{x}^*) = (z_j^* - \overline{z}_j) W_{j,c}^*$, where $\overline{\mathbf{z}} \triangleq N^{-1} \sum_{i=1}^N \mathbf{z}_i$ is the average processed feature vector computed on the training set. This leads to

the decomposition: $\widehat{z}^*_{\mathrm{cls},c} = \underbrace{b^*_c + (\mathbf{W}^*_c)^\top \overline{\mathbf{z}}}_{\text{baseline}} + \sum_{j=1}^{D} W^*_{j,c}(z^*_j - \overline{z}_j)$, where the baseline is query-dependent.

The feature contributions therefore describe how much each processed feature contributes to the fitted linear class score for the query, relative to a query-dependent baseline; they do not decompose the normalized probability, because clipping and normalization are nonlinear, nor do they describe how the feature affected the query representation, which neighbors were retrieved, or the coefficients that were fit. (In particular, even if a feature is displayed to have zero coefficient, it does not mean that the entire query-adaptive prediction pipeline is independent of the feature value.) Unless the class is fixed by the evaluation protocol, we report feature contributions for the predicted class. We also map the contributions back to the raw, semantic feature columns.

Optimization. We solve the two ElasticNet problems with weighted centering to allow for an unregularized intercept, optimizing each coefficient matrix jointly across classes with FISTA [26] with backtracking line search and checking the KKT residual, for at most 300 iterations with tolerance 10^{-4}. When solving for the coefficients of the shared model, all rows of the reference set are weighted equally, and when solving for the query-specific correction, we weight the rows according to equation 1.

Implementation details. For the primary configuration (global residual), we use one TABICLV2 estimator for test-query predictions, using float32 precision (not automatic mixed precision). We use the class probabilities directly as the target when fitting the shared model (there is no label smoothing). When computing the representation, we use a single view at the input to the first ICL decoder block; when retrieving neighbors, we use a cached, cross-fitted reference bank (see appendix B) and exact top-k retrieval.

Our primary configuration uses $k = 400$ and $\tau = 0.1$; $\alpha = 0.01$ and $\rho = 0.25$ at both stages (implying ℓ_1 coefficient $\alpha\rho = 0.0025$, ridge weight $\alpha(1-\rho) = 0.0075$, and the coefficient on $\|\mathbf{A}\|_F^2$ is 0.00375); when there are fewer than $k = 400$ training rows, we use all of them but still weight them according to their representations. We use squared loss.

Regional anchor variant. In TICL^3 (regional anchors), we fit local linear models in advance rather than fitting a new model for each query. We cluster the reference-bank representations and select the training sample closest to each cluster center as an anchor. For each anchor, we fit an ElasticNet using nearby reference samples and their backbone probability targets. Unlike TICL^3 (global + residual), these models are fitted directly to the backbone predictions rather than to residuals of a shared model. For a new query, we use its backbone representation to weight the coefficients and intercepts of the pre-fitted models. Their weighted combination gives a query-specific linear rule for prediction and feature contributions, without fitting a new model at prediction time (Appendix E.5).

4 Experiments

We evaluate three qualities of TICL^3: whether the deployed global-plus-local predictor remains a competitive tabular classifier; whether its combined coefficients rank data-generating features when the generating support is known; and the deletion monotonicity, selected-feature utility, and stability of its centered contributions on real-world datasets. Predictive performance is evaluated on 386 benchmark–dataset pairs across four classification benchmarks: CC-18 [27] (69 datasets), TabArena-Lite [28] (38), TALENT [29] (181), and BeyondArena [30] (98). We evaluate feature-selection quality on controlled synthetic tasks with designated generating supports. Real-data interpretability is evaluated on a fixed CC-18 subset, with regional explanation diagnostics confined to Appendix E.5.

4.1 Experimental Setup

We report mean accuracy on benchmark-specific splits. CC-18 and TabArena-Lite use stratified 80/20 splits generated independently for seeds 0, 1, and 2. TALENT keeps its provided train/validation/test partition, with training and validation sets combined, and uses seed 42 for capped subsampling. BeyondArena keeps its provided splits at split/repeat positions 0, 1, and 2. In all suites, we cap the training and test sets at 1600 and 400 rows. TICL^3 uses one backbone estimator, a five-fold cross-fitted reference bank, and the primary hyperparameters of Section 3; Table 3 uses separate runtime operating points (Appendix E.5).

Table 2: Predictive performance on CC-18, TabArena-Lite, TALENT, and BeyondArena under Section 4.1. In that order, TABICLV2 uses 8/4/4/4 estimators and TabPFNv2 uses 8/8/4/8; both TICL^3 variants use one. The TICL^3 rows use a five-fold cross-fitted bank and the configurations in Section 3 and Appendix E.5. TICL^3 (regional anchors) is grouped by its predictor form, but its predicted-class explanation is all zero for 73.3% of synthetic and 43.8% of real-data queries (Appendix E.5). The $\pm$ values describe variation across dataset means.

Family	Method	CC-18↑	TabArena-Lite↑	TALENT↑	BeyondArena↑
Tabular foundation models	TABICLV2	$\mathbf{0.872 \pm 0.154}$	$\mathbf{0.880 \pm 0.087}$	$\mathbf{0.830 \pm 0.148}$	$\mathbf{0.839 \pm 0.132}$
	TabPFNv2	0.865 ± 0.157	$\mathbf{0.880 \pm 0.086}$	0.825 ± 0.149	0.835 ± 0.137
Interpretable by design	TICL^3 (global + residual)	0.856 ± 0.161	0.877 ± 0.089	0.817 ± 0.160	0.833 ± 0.134
	TICL^3 (regional anchors)	0.855 ± 0.162	0.874 ± 0.087	0.814 ± 0.164	0.834 ± 0.136
	MAPLE	0.830 ± 0.164	0.862 ± 0.091	0.790 ± 0.166	0.820 ± 0.137
	EBM	0.840 ± 0.159	0.865 ± 0.093	0.779 ± 0.186	0.825 ± 0.136
	NAM	0.827 ± 0.165	0.859 ± 0.094	0.774 ± 0.182	0.808 ± 0.144
Kernel ICL surrogates	KernelICL (kNN, k=20)	0.825 ± 0.206	0.872 ± 0.094	0.803 ± 0.177	0.823 ± 0.150
	KernelICL (Gaussian, scale 4)	0.826 ± 0.205	0.875 ± 0.091	0.806 ± 0.173	0.825 ± 0.152
Deep tabular models	TabR	0.840 ± 0.165	0.855 ± 0.095	0.797 ± 0.160	0.798 ± 0.154
	RealMLP	0.846 ± 0.164	0.862 ± 0.094	0.790 ± 0.174	0.816 ± 0.144
	ModernNCA	0.843 ± 0.165	0.862 ± 0.094	0.798 ± 0.164	0.819 ± 0.139
	FT-Transformer	0.836 ± 0.167	0.860 ± 0.092	0.783 ± 0.175	0.813 ± 0.145
Classical baselines	CatBoost	0.849 ± 0.161	0.866 ± 0.092	0.799 ± 0.166	0.824 ± 0.141
	Random Forest	0.842 ± 0.164	0.867 ± 0.091	0.797 ± 0.162	0.824 ± 0.139
	LightGBM	0.842 ± 0.165	0.862 ± 0.096	0.795 ± 0.165	0.817 ± 0.147
	XGBoost	0.842 ± 0.162	0.863 ± 0.093	0.794 ± 0.165	0.816 ± 0.148
	Logistic Regression	0.764 ± 0.199	0.806 ± 0.122	0.738 ± 0.205	0.780 ± 0.157

Table 3: CC-18 accuracy and mean fit/prediction time over 69 datasets and three seeds. The default TABICLV2 row uses eight estimators; the matched row uses one. Configuration and timing details are in Appendix E.5.

Method	Mean Acc.↑	Fit (s)↓	Predict (s)↓
TABICLV2 (8 estimators)	**0.8735**	0.950	5.462
TABICLV2 (1 estimator)	0.8710	**0.485**	**0.745**
TabPFNv2	0.8641	3.139	9.660
TICL^3 (global + residual)	0.8557	5.515	6.849
TICL^3-Fast (global + residual)	0.8609	5.842	1.495
TICL^3 (regional anchors)	0.8555	6.736	0.850

We compare against TABICLV2, KernelICL [17], TabPFNv2 [19], modern neural baselines [21, 31–33], interpretable additive models [1, 14], and classical tree/linear baselines. Appendix E.2 gives versions, fixed settings, validation use, and suite-specific ensemble sizes.

4.2 Predictive Performance

Table 2 compares the completed primary configuration with foundation-model, interpretable, neural, and classical baselines. TICL^3 (global + residual) is within 0.3–1.6 percentage points of TABICLV2 across the four suites and exceeds the strongest tree baseline by 0.7–1.8 points. TICL^3 (regional anchors) trades a small amount of accuracy for faster prediction, as quantified in Table 3. Its explanations are less informative, however: its anchor models use a ten-times larger ℓ_1 penalty and no shared component, and the blended predicted-class coefficients are all zero for 73.3% of synthetic and 43.8% of real-data queries, whose predictions then vary only through anchor routing and blended intercepts (Appendix E.5). We therefore treat it as a computational variant rather than a coequal explanation method.

In Table 3, the fast configuration reduces global-residual prediction time from 6.849 to 1.495 s; regional anchors require 0.850 s. Their speed advantage is relative to the eight-estimator backbone: all TICL variants predict more slowly than the matched one-estimator backbone. These batch timings differ from the explanation workload in Table 5.

4.3 Feature Selection

In real-world datasets, we rarely know which features generate each label. Deletion metrics therefore cannot establish whether an explanation identifies these features. Feature selection and support recovery have long been used for interpretable prediction [11, 23, 34, 35]. Recent work also considers self-supervised and group-structured unsupervised feature selection [36, 37], as well as sparse

Table 4: Generating-support F1 and accuracy on LSPIN E1–E4 (10 seeds; 100 queries per task–seed). Blocks distinguish own-predictor from masked-TABICLV2 accuracy; E4 uses group-conditional support. An all-zero coefficient vector selects no features and receives support F1 zero.

	E1		E2		E3		E4	
Method	**F1↑**	**Acc.↑**	**F1↑**	**Acc.↑**	**F1↑**	**Acc.↑**	**F1↑**	**Acc.↑**
Interpretable-by-design predictors (own deployed accuracy)								
TICL^3 (global + residual)	0.512	0.663	0.540	0.675	0.554	**0.787**	0.548	0.697
MAPLE	0.408	**0.665**	0.480	**0.684**	0.455	0.760	0.624	**0.712**
EBM	0.439	0.636	0.361	0.653	0.494	0.690	0.541	0.688
Logistic (ERM)	0.422	0.535	0.343	0.649	0.491	0.656	0.493	0.632
Post-hoc explainers of TABICLV2 *(masked-*TABICLV2 *accuracy)*								
LIME	0.517	0.645	**0.640**	0.674	**0.713**	0.743	**0.964**	0.703
KernelSHAP	0.525	0.638	0.616	0.656	0.689	0.768	0.556	0.690
Integrated Gradients	0.533	0.639	0.624	**0.684**	0.687	0.764	0.765	0.702
GradientSHAP	**0.540**	0.634	0.617	0.664	0.662	0.766	0.581	0.679

linear support recovery using randomized, refined, or projected-gate estimators [38–40]. We use four controlled binary tasks adapted from LSPIN [16], where the generating features are known for each query and vary across examples. E1–E3 use nonlinear gated rules with query-dependent support. E4 has two latent generating groups and 46 nuisance features. We test whether explanations rank the generating features highly for each query. In E4, we evaluate recovery of the support for the query's latent group. However, all four informative variables are shifted by group, so variables outside this support can still help predict the label when the group is unobserved.

We compare TICL^3 (global + residual), using one backbone estimator and the primary reference bank, with EBM, logistic regression, and MAPLE [15]. For each query, we rank features by the absolute values of the nonzero global-plus-residual coefficients for TICL^3, local coefficients for MAPLE, additive scores for EBM, and mean-relative standardized contributions for logistic regression. Each method selects up to the known number of generating features, giving all methods an oracle ranking budget. We report support F1 and the accuracy of each interpretable method's own predictions. We evaluate post-hoc explainers of TABICLV2 against the same supports and with the same budget, but measure accuracy after replacing unselected features with their means. Appendix E.10 provides the full setup, and Appendix E.5 reports regional diagnostics.

Across the four tasks, seeds 21–30, and 100 queries per task and seed, TICL^3 (global + residual) achieves mean support F1 of 0.539 ± 0.010 and accuracy of 0.706 ± 0.011. Standard errors are computed across seeds after averaging over tasks within each seed. Among the interpretable-by-design predictors, TICL^3 (global + residual) has the highest average support F1 and performs best on E1–E3. MAPLE has higher support F1 on E4 (0.624 versus 0.548). The post-hoc explainers of TABICLV2 are shown for reference rather than as direct competitors: their scores do not determine any prediction, whereas each interpretable-by-design method is evaluated on the rule it uses to predict.

4.4 Interpretability Evaluation

For each query, we sum the centered contributions of processed features that belong to the same original table column, and rank the columns by their signed contribution to the predicted-class score. This allows us to check whether removing the features that most support the prediction consistently decreases its probability. We use the same coefficients as the deployed predictor to obtain this ranking; Appendix E.9 describes the exact mapping to the original columns.

Table 5 uses 50 fixed queries from each of 10 CC-18 datasets (see Appendices E.6 and E.7). For both TICL^3 (global + residual) and the post-hoc methods, we evaluate explanations of the same complete TICL^3 (global + residual) predictor. We keep the predicted class fixed while progressively replacing features with their training-set means. Deletion Spearman measures the correlation between the number of features replaced and the cumulative decrease in the probability of that class, so it measures how consistently the probability decreases, rather than how much it decreases. Gen-SVM measures how well the selected features allow an SVM to reproduce the corresponding wrapper's predicted class, while Gen-kmeans measures how well clusters obtained from the selected features agree with the true labels. Neighbor stability measures how much the normalized attribution or selection scores change between nearby observed queries; lower values indicate greater stability.

Table 5: Explanation quality and online prediction-plus-explanation cost on 10 CC-18 datasets. We evaluate quality using 500 queries; all rows use the same 422 queries with valid values for deletion and 493 for the other metrics. We exclude undefined values rather than setting them to zero. We average runtime across datasets, giving each dataset equal weight and excluding one-off setup. For LIME and KernelSHAP attribution time, we use a fixed subset of 100 queries (10 per dataset, three repeats); for the common predictor latency and the remaining rows, we use 50 queries per dataset.

Method	Deletion Spearman↑	Gen-SVM↑	Gen-kmeans↑	Neighbour stab.↓	Online ms/query↓
TICL^3 (global + residual)	0.470	**0.886**	**0.697**	0.032	**26.974**
KernelSHAP	0.384	0.708	0.555	0.037	53,686.935
LIME	0.449	0.813	0.601	0.032	14,607.496
FastSHAP	0.408	0.608	0.637	0.025	27.197
IG on surrogate	0.417	0.861	0.669	0.031	27.482
GradientSHAP on surrogate	0.446	0.856	0.666	0.031	27.560
Input × Gradient on surrogate	0.454	0.830	0.568	0.034	27.112
INVASE	0.458	0.514	0.495	0.007	27.191
L2X	0.441	0.503	0.536	0.010	27.213
REAL-X	**0.534**	0.500	0.476	**0.004**	27.192

We report the regional results and the different sets of queries for which their metrics are valid in Appendix E.5. Appendix E.8 gives the formal definitions.

TICL^3 (global + residual) has the highest Gen-SVM agreement and Gen-kmeans score, and the second-highest deletion Spearman correlation. Its neighbor stability is similar to LIME and the gradient-based methods, although the learned selectors are more stable. The learned post-hoc explainers add little online latency after the common predictor call, but they require 41.1–105.0 s per dataset for one-off construction. For 50 queries, excluding the predictor setup shared by all methods, TICL^3 (global + residual) takes 1.35 s, while the fastest post-hoc alternative takes 42.44 s (see Appendix G).

Component and configuration ablations. On the ten explanation-benchmark datasets, adding a local correction increases the shared model's accuracy from 76.13% to 89.90%. However, the global-only model has higher deletion Spearman (0.610 versus 0.530) and Gen-SVM agreement (95.4% versus 88.8%). This means that a model can score higher on these metrics even when its accuracy is lower, so we need to consider them together with accuracy.

When compared with the local-only model using the same $k = 400$, the combined model has slightly lower accuracy (89.90% versus 90.18%), and we don't find a clear difference in deletion monotonicity in the paired comparison. However, the combined model gives a nonzero explanation for every query, while the local-only model does so for only 68% of queries. On this subset, the main benefit we observe from adding the shared model is that the explanation is nonzero for every query.

Using $k = 400$ rather than $k = 30$ improves paired deletion Spearman by 0.130 (95% interval $[0.027, 0.234]$), while $k = 30$ gives 1.29 percentage points higher accuracy on this subset. On CC-18, the fast configuration ($k = 30$, 30 local iterations) also gives 0.52 percentage points higher accuracy and is $4.6\times$ faster at prediction (Table 3). Reducing the local iteration cap from 300 to 30 shows no clear difference. Appendix G.1 reports all nine configurations and their paired comparisons.

4.5 Regression

To test whether the method can also be used for continuous targets, we use a scalar prediction from the backbone as the teacher target. We build the reference bank using five-fold cross-fitting with shuffled K-fold splits instead of stratified folds. We standardize all teacher targets using the target mean and scale learned by the full-context backbone, so that the fixed regularization strength is comparable across datasets. TICL^3 (global + residual) fits its shared and local squared-loss ElasticNets to these standardized targets. For TICL^3 (regional anchors), we use the same 64-anchor configuration as in classification, fit each anchor model to the standardized out-of-fold teacher targets, and combine the eight closest anchor models. Both methods transform their predictions back to the original target scale, without clipping or probability normalization. Table 6 reports predictive performance on the 119 TALENT regression datasets; we don't evaluate explanation quality here.

TICL^3 (global + residual) has an nRMSE of 0.515, trailing the TabICLv2 backbone model (0.482 nRMSE for the one-estimator case), but improving over the strongest tree baseline (0.537 nRMSE) and the other evaluated interpretable predictors. TICL^3 (regional anchors) is also competitive (0.521

Table 6: TALENT regression on 119 splits. nRMSE uses the test-target standard deviation; ± reports across-dataset variation.

Family	Method	Mean nRMSE↓	Mean R^2↑
Tabular foundation models	TabICLv2	$\mathbf{0.479 \pm 0.300}$	$\mathbf{0.681}$
	TabICLv2 (1 estimator)	$\underline{0.482 \pm 0.303}$	$\underline{0.676}$
	TabPFNv2	0.506 ± 0.310	0.648
Interpretable by design	TICL^3 (global + residual)	0.515 ± 0.287	0.653
	TICL^3 (regional anchors)	0.521 ± 0.275	0.654
	MAPLE	0.571 ± 0.298	0.586
	EBM	0.601 ± 0.273	0.564
Classical baselines	CatBoost	0.537 ± 0.297	0.623
	Random Forest	0.550 ± 0.301	0.608
	LightGBM	0.566 ± 0.333	0.569
	XGBoost	0.587 ± 0.394	0.501
	Ridge Regression	0.702 ± 0.272	0.434

nRMSE), with a slightly higher mean R^2 score (0.654 vs. 0.653 for TICL^3 (global + residual)). We leave further tuning of these approaches for regression tasks to future work.

4.6 Limitations and Future Work

TICL^3 (global + residual) is limited by the expressiveness of the linear model, and may struggle to capture highly interactive local residual functions in the retrieved neighborhood, particularly for higher-dimensional and multi-class problems. Our approach does not directly model categorical features, and we ordinally encode any categorical columns before treating them as numerical. As a result, our feature contributions are for the entire encoded column, rather than for specific categories. Handling categorical features directly and providing attributions for groups of categorical features could improve the interpretability of TICL^3 (global + residual) in this setting. Similarly, we do not currently evaluate our approach on text input. We leave evaluating TICL^3 (global + residual) with direct categorical handling and on text inputs, and augmenting the approach with richer local interpretable model components, to future work.

While our provided feature contributions exactly decompose the query-specific linear score, our explanations do not decompose the final normalized probability score or account for how features influence the learned representation, retrieval, and the fitted coefficients. Sparsity in the learned coefficients does not imply that the entire query-adaptive model only depends on the features with nonzero coefficients. Our retrieval approach (top-k) can also introduce discontinuities into our explanations, which can shift substantially based on small changes in the query point; the active set for ElasticNet may also vary. Further, our regional anchor model approach often does not provide a feature-level explanation, as the coefficients for the blended model (corresponding to the predicted class) are all zeros for 73.3% of queries on the synthetic data and 43.8% on real data.

Our real data interpretability study is limited to a single backbone model and a fixed subset of 10 datasets in a capped low-data regime. The studied metrics do not establish causal feature relevance; notably, Gen-SVM can utilize both the locations and values of retained features. Additionally, the gradient-based rows utilize a differentiable surrogate with unquantified approximation error. We also do not evaluate the interpretability of our regression approach. Our methods also require access to the internal representations of the tabular in-context learning backbone, and do not apply directly to fully black-box models.

5 Conclusion

We propose TICL^3 (global + residual), an interpretable-by-design predictor leveraging a tabular foundation model to guide a linear predictor. TICL^3 (global + residual) learns a single ElasticNet model to capture global feature effects, and leverages the tabular foundation model to retrieve a similar set of samples for each query, learning a second ElasticNet to capture the residuals of the first. Together, these produce a query-specific linear prediction rule. TICL^3 (global + residual) provides feature contributions and a baseline that exactly sum to the linear prediction score for each class. We theoretically bound TICL^3 (global + residual)'s local approximation error, under a smoothness assumption on the residual in the learned representation space that includes an additive term for differences between labeled contexts.

Across four classification benchmarks, TICL^3 (global + residual) is competitive with the TabICLv2 backbone (within 0.3 to 1.6 percentage points in accuracy) and outperforms the strongest

tree-based baseline on each. Across a synthetic benchmark for feature selection, TICL^3 (global + residual) is the top-performing interpretable predictor among those evaluated on average in support F1 score. On the real-data explanation benchmark, TICL^3 (global + residual) scores highest among the evaluated methods on two metrics measuring the utility of the selected features, and second on a metric measuring deletion monotonicity. We also propose TICL^3 (regional anchors), which combines pre-fitted regional linear models; TICL^3 (regional anchors) is slower than a comparable single-estimator backbone model, but faster than the corresponding eight-estimator backbone model.

Our approach is limited by the fact that our explanations are only for the linear class scores, and by the expressiveness of local linear models. We leave exploring richer local interpretable models, leveraging additional tabular foundation models, and evaluating our approach's explanation quality for regression to future work.

Reproducibility statement

We specify the details of our predictor and its primary hyperparameters in Section 3 and the specifics of our benchmark selection, data splits, seeds, and row capping in Section 4.1. In Appendix E.1, we detail the data preprocessing and raw feature mapping. In Appendix E.2, we detail the configuration of our predictive baselines. In Appendix B, we detail the construction of our cross-fitted reference bank. In Appendix C, we detail our ElasticNet solver. In Appendix E.5, we detail our runtime protocol and the configuration of our regional anchors. In Appendix E.6, we detail our explanation benchmark; in Appendix E.7, we detail our explanation baselines; and in Appendix E.8, we provide metric definitions. In Appendix E.10, we detail our synthetic task generation. In Appendix D, we provide a proof of Theorem 1.

AI use statement for this paper

We used generative AI to write, debug, and run code for this paper, including code for the implementation of our method, synthetic task generators, preprocessing for our explanation benchmark tasks, and code to tabulate results. We note that all experiments in this paper were designed by the authors and not by generative AI. The generative AI tools were not used to generate proof ideas, hypotheses, or conceptual frameworks, to design experiments, or to translate text. The generative AI tools were used to assist with literature searches, structure the paper, generate a draft, and edit the paper (i.e., change wording or the placement of author-written text, such as in proofs). They also helped check whether descriptions of theoretical claims and experimental results were consistent with the proofs and reported numbers. All code written by the generative AI was reviewed and tested by the authors to ensure it performed as expected. All numbers reported in this paper were checked to make sure they match the corresponding recorded output from the experiments. All proofs were checked by hand by the authors. All edits to the text were approved by the authors. The authors take full responsibility for this work, including all text, claims, and artifacts generated with the help of AI.

References

[1] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. InterpretML: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[2] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=cp5PvcI6w8_`.

[3] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[4] David Schiff, Ofir Lindenbaum, and Yonathan Efroni. ICR-RL: Deep reinforcement learning via in-context-regression. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016. doi: 10.1145/2939672.2939778.

[6] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774, 2017.

[7] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3319–3328, 2017.

[8] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3145–3153, 2017.

[9] Neil Jethani, Mukund Sudarshan, Ian Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time Shapley value estimation. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Zq2G_VTV53T`.

[10] Simone Piaggesi, Riccardo Guidotti, Fosca Giannotti, and Dino Pedreschi. Explanations go linear: Post-hoc explainability for tabular data with interpretable meta-encoding. In *Proceedings of the IEEE International Conference on Data Mining*, pages 663–672, 2025. doi: 10.1109/ICDM65498.2025.00074.

[11] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. doi: 10.1111/j.1467-9868.2005.00503.x.

[12] 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. doi: 10.48550/arXiv.2602.11139.

[13] João Fonseca and Julia Stoyanovich. ExplainerPFN: Towards tabular foundation models for model-free zero-shot feature importance estimations. *arXiv preprint arXiv:2601.23068*, 2026. doi: 10.48550/arXiv.2601.23068.

[14] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Benjamin J. Lengerich, Rich Caruana, and Geoffrey E. Hinton. Neural additive models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems*, volume 34, pages 4699–4711, 2021.

[15] Gregory Plumb, Denali Molitor, and Ameet S. Talwalkar. Model agnostic supervised local explanations. In *Advances in Neural Information Processing Systems*, volume 31, pages 2520–2529, 2018.

[16] Junchen Yang, Ofir Lindenbaum, and Yuval Kluger. Locally sparse neural networks for tabular biomedical data. In *Proceedings of the 39th International Conference on Machine Learning*, pages 25123–25153, 2022.

[17] Ratmir Miftachov, Bruno Charron, and Simon Valentin. Interpretable tabular foundation models via in-context kernel regression. *arXiv preprint arXiv:2602.02162*, 2026. doi: 10.48550/arXiv.2602.02162.

[18] Luan Borges Teodoro Reis Sena and Francisco Galuppo Azevedo. Real-time explanations for tabular foundation models. *arXiv preprint arXiv:2603.29946*, 2026. doi: 10.48550/arXiv.2603.29946.

[19] 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. doi: 10.1038/s41586-024-08328-6.

[20] Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=Sd4wYYOhmY`.

[21] Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. TabR: Tabular deep learning meets nearest neighbors. In *International Conference on Learning Representations*, 2024.

[22] Erel Naor and Ofir Lindenbaum. Hybrid autoencoders for tabular data: Leveraging model-based augmentation in low-label settings. In *Advances in Neural Information Processing Systems*, 2025. URL `https://arxiv.org/abs/2511.06961`.

[23] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x.

[24] William S. Cleveland and Susan J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988. doi: 10.1080/01621459.1988.10478639.

[25] Ram Dyuthi Sristi, Ofir Lindenbaum, Shira Lifshitz, Maria Lavzin, Jackie Schiller, Gal Mishne, and Hadas Benisty. Contextual feature selection with conditional stochastic gates. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 46375–46392. PMLR, 2024. URL `https://proceedings.mlr.press/v235/sristi24a.html`.

[26] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. doi: 10.1137/080716542.

[27] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021. URL `https://openreview.net/forum?id=OCrD8ycKjG`.

[28] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. doi: 10.48550/arXiv.2506.16791.

[29] Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning methods on tabular datasets. *arXiv preprint arXiv:2407.00956*, 2024. doi: 10.48550/arXiv.2407.00956.

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

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

[32] Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later. In *International Conference on Learning Representations*, 2025.

[33] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[34] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[35] Uri Shaham, Ofir Lindenbaum, Jonathan Svirsky, and Yuval Kluger. Deep unsupervised feature selection by discarding nuisance and correlated features. *Neural Networks*, 152:34–43, 2022. doi: 10.1016/j.neunet.2022.04.002.

[36] Daniel Segal, Ofir Lindenbaum, and Ariel Jaffe. Spectral self-supervised feature selection. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=t0EJiOd9Lg.

[37] Shira Lifshitz, Ofir Lindenbaum, Gal Mishne, Ron Meir, and Hadas Benisty. Unsupervised feature selection through group discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23496–23504, 2026. doi: 10.1609/aaai.v40i28.39521. URL https://doi.org/10.1609/aaai.v40i28.39521.

[38] Ofir Lindenbaum and Stefan Steinerberger. Randomly aggregated least squares for support recovery. *Signal Processing*, 180:107858, 2021. doi: 10.1016/j.sigpro.2020.107858. URL https://doi.org/10.1016/j.sigpro.2020.107858.

[39] Ofir Lindenbaum and Stefan Steinerberger. Refined least squares for support recovery. *Signal Processing*, 195:108493, 2022. doi: 10.1016/j.sigpro.2022.108493. URL https://doi.org/10.1016/j.sigpro.2022.108493.

[40] Soham Jana, Henry Li, Yutaro Yamada, and Ofir Lindenbaum. Support recovery with projected stochastic gates: Theory and application for linear models. *Signal Processing*, 213:109193, 2023. doi: 10.1016/j.sigpro.2023.109193. URL https://doi.org/10.1016/j.sigpro.2023.109193.

[41] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. In *British Machine Vision Conference*, 2018.

[42] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[43] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, 2020. doi: 10.18653/v1/2020.acl-main.408.

[44] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for PyTorch. *arXiv preprint arXiv:2009.07896*, 2020.

[45] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 883–892, 2018.

[46] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJg_roAcK7`.

[47] Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pages 2650–2658, 2021.

[48] Jianqing Fan and Irène Gijbels. *Local Polynomial Modelling and Its Applications*, volume 66 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, 1996.

[49] Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning*, pages 10648–10659, 2020.

[50] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4077–4087, 2017.

A Broader Impacts

Positive impacts. Interpretable machine learning is especially valuable in high-stakes decision domains such as healthcare, credit scoring, and legal risk assessment, where practitioners need to understand and audit individual predictions. TICL^3 lowers the barrier to inspecting tabular predictions by producing additive per-feature contributions directly, without a separate post-hoc explainer pipeline. Such decompositions may make it easier for domain experts and regulators to audit model behaviour, provided that their empirical fidelity and stability are validated for the deployed configuration.

Negative impacts. Like other prediction and explanation tools, the method may be misused in high-stakes settings or lend unwarranted confidence to automated decisions. Practitioners may also over-interpret coefficients as causal evidence rather than predictive associations. We frame them explicitly as contributions to the deployed query-specific linear score; they do not separately decompose representation-based retrieval, coefficient fitting, or the clipping-and-normalization step, and they do not imply causality.

B Reference-Bank Construction

The reference bank is constructed once for each training split using five-fold stratified cross-fitting. We partition the training rows into five folds. For each fold, we fit the backbone on the other four folds, present the held-out rows as unlabeled queries, and record their class probabilities and representations at the input to the first ICL decoder block. Concatenating the held-out outputs gives every training row one out-of-fold probability target and one query-role representation. This prevents a row's target and representation from being produced by a backbone that was fitted on that row's label, and it places reference rows in the same unlabeled-query role used at test time. A separate backbone fitted to the complete training split supplies test-query representations. The folds are used only to construct the reference bank; they do not create an ensemble of deployed predictors. The processed design vectors used by both ElasticNet stages are constructed with the complete-training preprocessing map. The retained-column mask, processed-coordinate order, and raw-column origin map are then held fixed; fold-specific backbones provide out-of-fold targets and representations, not alternative linear-design coordinates. Thus a stored row and a test query enter the linear rule in the same processed feature space even though their representations are obtained under different labeled contexts. Cross-fitting prevents direct label leakage and places each reference row in an unlabeled-query role, but it does not guarantee that representations produced under different fold contexts are directly comparable to one another or to a test representation produced under the complete context. Theorem 1 therefore states the required compatibility condition on the actual context-specific outputs and includes an additive term for the resulting cross-context shift.

C Weighted ElasticNet Optimization

Both stages of TICL^3 use an unpenalized intercept and the same ElasticNet penalty. Consider a generic problem with M observations, processed feature vectors $\mathbf{z}_i \in \mathbb{R}^D$, target vectors $\mathbf{t}_i \in \mathbb{R}^C$, and normalized weights $w_i \geq 0$ satisfying $\sum_{i=1}^M w_i = 1$. Define the weighted means $\bar{\mathbf{z}}_w = \sum_{i=1}^M w_i \mathbf{z}_i$ and $\bar{\mathbf{t}}_w = \sum_{i=1}^M w_i \mathbf{t}_i$. Let $\mathbf{Z}_w \in \mathbb{R}^{M\times D}$ and $\mathbf{T}_w \in \mathbb{R}^{M\times C}$ have rows $\sqrt{w_i}(\mathbf{z}_i - \bar{\mathbf{z}}_w)^\top$ and $\sqrt{w_i}(\mathbf{t}_i - \bar{\mathbf{t}}_w)^\top$, respectively. The centered objective is

$$\min_{\mathbf{A}\in\mathbb{R}^{D\times C}} \ \frac{1}{2}\|\mathbf{Z}_w\mathbf{A} - \mathbf{T}_w\|_F^2 + \alpha\rho\|\mathbf{A}\|_1 + \frac{\alpha(1-\rho)}{2}\|\mathbf{A}\|_F^2, \qquad \mathbf{a} = \bar{\mathbf{t}}_w - \mathbf{A}^\top\bar{\mathbf{z}}_w.$$

The recovered unpenalized intercept is $\mathbf{a} \in \mathbb{R}^C$. The implementation jointly optimizes all class columns using FISTA with backtracking and a Karush–Kuhn–Tucker residual check. It uses at most 300 iterations and tolerance 10^{-4}. The shared model uses uniform weights and the probability targets in the primary reference bank; the query-specific correction uses the normalized similarity weights from Eq. (1) and residual targets $\mathbf{r}_i$. Both use $\alpha = 0.01$ and $\rho = 0.25$, corresponding to an ℓ_1 coefficient $\alpha\rho = 0.0025$ and ridge weight $\alpha(1-\rho) = 0.0075$; because the ridge term is written with a factor of $1/2$, the multiplier on $\|\mathbf{A}\|_F^2$ is 0.00375.

D Proof of the Residual Approximation Bound

Proof of Theorem 1. For brevity, write $\mathbf{D} = \mathbf{D}^*$, $\mathbf{d} = \mathbf{d}^*$, $\widetilde{w}_i = \widetilde{w}_i^*$, and $\bar{\mathbf{r}}_k = \sum_{i \in \mathcal{N}^*} \widetilde{w}_i \mathbf{r}_i$. Because the intercept in Eq. (2) is unpenalized, its first-order optimality condition gives $\mathbf{d} = \bar{\mathbf{r}}_k - \mathbf{D}^\top \bar{\mathbf{z}}_k$. The shared term cancels from the approximation error, so

$$\begin{aligned}
&\|\boldsymbol{\eta}^* - \widehat{\boldsymbol{\eta}}^*_{\mathrm{lin}}\|_2 \\
&\quad = \|\mathbf{r}^* - (\mathbf{D}^\top \mathbf{z}^* + \mathbf{d})\|_2 \\
&\quad \le \|\mathbf{r}^* - \bar{\mathbf{r}}_k\|_2 + \|\bar{\mathbf{r}}_k - (\mathbf{D}^\top \mathbf{z}^* + \mathbf{d})\|_2.
\end{aligned}$$

For the first term, the assumption on the retrieved residuals implies

$$\begin{aligned}
\|\mathbf{r}^* - \bar{\mathbf{r}}_k\|_2 &\le \sum_{i \in \mathcal{N}^*} \widetilde{w}_i \|\mathbf{r}^* - \mathbf{r}_i\|_2 \\
&\le L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\mathrm{ctx}}.
\end{aligned}$$

For the second term, the intercept identity yields

$$\|\bar{\mathbf{r}}_k - (\mathbf{D}^\top \mathbf{z}^* + \mathbf{d})\|_2 \le \|\mathbf{z}^* - \bar{\mathbf{z}}_k\|_2 \|\mathbf{D}\|_F.$$

It remains to bound $\|\mathbf{D}\|_F$. By optimality, the objective at $(\mathbf{D}, \mathbf{d})$ is no larger than at the feasible local-constant choice $(\mathbf{0}, \bar{\mathbf{r}}_k)$. Dropping the nonnegative data-fit and ℓ_1 terms from the optimum gives

$$\begin{aligned}
\frac{\alpha(1-\rho)}{2} \|\mathbf{D}\|_F^2 &\le \frac{1}{2} \sum_{i \in \mathcal{N}^*} \widetilde{w}_i \|\mathbf{r}_i - \bar{\mathbf{r}}_k\|_2^2 \\
&\le \frac{1}{2} \sum_{i \in \mathcal{N}^*} \widetilde{w}_i \|\mathbf{r}_i - \mathbf{r}^*\|_2^2 \\
&\le \frac{\big(L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\mathrm{ctx}}\big)^2}{2}.
\end{aligned}$$

The middle inequality holds because the weighted mean $\bar{\mathbf{r}}_k$ minimizes weighted squared distance over constant vectors. Hence $\|\mathbf{D}\|_F \le \big(L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\mathrm{ctx}}\big) / \sqrt{\alpha(1-\rho)}$. Substitution into the two error terms above proves Eq. (3). □

The theorem states its assumption on the actual context-specific residuals, and the proof uses it only for pairs formed by a retrieved row and the query; no condition is imposed among the reference rows. Without the additive term ϵ_{ctx}, any nonzero context-induced shift between a reference row and the query at vanishing representation distance would force L_r to be arbitrarily large. The first error term alone is the error of the local-constant correction $(\mathbf{0}, \bar{\mathbf{r}}_k)$, which is therefore at most $L_r \Delta_k(\mathbf{x}^*) + \epsilon_{\mathrm{ctx}}$; the result bounds how much the regularized slope can add to this error rather than establishing a first-order improvement from fitting it. It is tighter than an analogous target-smoothness result only when the shared model reduces local variation. The bound applies to the linear score, not to the subsequent clipping and normalization. No claim is made that cross-fitting bounds ϵ_{ctx} or that the residual smoothness condition holds uniformly on every dataset.

E Additional Experimental Details

E.1 Input Preprocessing and Raw-Feature Mapping

Each benchmark loader preserves a fixed raw-column order and converts categorical columns to ordinal integer codes. For CC-18, codes are created when the full OpenML table is downloaded (a missing category becomes NaN); for TabArena-Lite, the encoder is fitted to the full table before the seeded split. TALENT fits the encoder on its supplied training split and applies it to validation and test, while BeyondArena fits it on each provided training split; both assign unseen categories code -1. After the benchmark's training block is formed, remaining NaNs are replaced by training-column means; for TALENT this block is the concatenated supplied training and validation data. No one-hot, target, or interaction expansion is applied. Labels are integer encoded independently of the feature preprocessing.

For the single-estimator TICL models, the resulting numerical matrix is passed through the TabICL 2.0.3 preprocessing pipeline fitted on the training block. The map removes constant columns, standardizes each retained column and clips the standardized value to $[-100, 100]$, applies the Yeo–Johnson power transform with standardization, and applies TabICL's two-stage soft outlier clipping with threshold four. The same fitted transformations are applied to reference rows and test queries. With one estimator, the recorded first-member feature permutation fixes the processed-coordinate order. Consequently, every retained processed coordinate has exactly one originating raw column; a raw constant column has no retained coordinate and receives score zero.

Let $g(j)$ denote the raw-column origin of processed coordinate j. For class c, the displayed raw-column contribution is

$$A_{r,c}(\mathbf{x}^*) = \sum_{j:g(j)=r} (z_j^* - \overline{z}_j) W_{j,c}^*, \qquad \mathbf{W}^* = \mathbf{P} + \mathbf{D}^*.$$

Thus grouping uses a *sum of contributions*, which preserves the exact linear-score decomposition. In the reported experiments the sum contains at most one term because preprocessing does not expand a column. Synthetic support recovery instead attaches each transformed-space combined coefficient $W_{j,c}^*$ to its raw origin before taking absolute values and ranking. It does not sum coefficients, rescale them into raw measurement units, or use the centered contribution formula. This distinction prevents the raw-feature display operation from changing the coefficient-ranking estimand.

E.2 Predictive-Baseline Configurations

Table 7 records the fixed configurations used for Table 2. Runs used Python 3.10 with TabICL 2.0.3, TabPFN 2.2.1, scikit-learn 1.6.1, CatBoost 1.2.10, XGBoost 3.1.0, LightGBM 4.6.0, InterpretML 0.7.8, AutoGluon 1.5.0, pytabkit 1.7.3, TabArena 0.0.1, and PyTorch 2.9.1. None of the methods received a benchmark-wide hyperparameter search; validation below means within-fit model selection or early stopping, never selection on the reported test split.

Table 7: Configurations for the predictive baselines in Table 2. Suite order for ensemble counts is CC-18/TabArena-Lite/TALENT/BeyondArena. "Defaults" means the named package's released default configuration except for the explicitly listed settings.

Method	Fixed configuration	Selection/validation
TabICLv2	TabICL; 8/4/4/4 estimators; packaged defaults otherwise.	No tuning or validation selection.
TabPFNv2	TabPFN; 8/8/4/8 estimators; native multiclass model for at most 10 classes and one-vs-rest above 10; packaged defaults otherwise.	No tuning or validation selection.
KernelICL	Repository implementation with four TabICL estimators; kNN uses $k = 20$ and Gaussian uses scale 4.	Fixed a priori; no calibration or search.
MAPLE	scikit-learn random forest with 200 trees, `max_features`=0.5, minimum leaf size 10; per-query ridge penalty 0.001.	Stratified 80/20 training/validation split; retained count chosen by validation RMSE from candidates up to $\min(D, 100)$, using at most 50 validation queries.
EBM	InterpretML; 128 bins, 3000 maximum rounds, learning rate 0.01, and no interactions.	Fixed; package-internal fitting only, with no external search.
NAM	One hidden layer of width 32 per feature; at most 300 epochs; batch size 256; AdamW with learning rate 10^{-3} and weight decay 10^{-4}.	Stratified 80/20 training/validation split; validation-loss early stopping with patience 30.
TabR	pytabkit `TabR_S_D_Classifier` defaults.	No external search; package training procedure.
RealMLP, ModernNCA, FT-Transformer	AutoGluon RealMLP and FT-Transformer, and the TabArena ModernNCA wrapper, using released defaults. FT-Transformer worker counts were set to zero.	Fixed stratified 80/20 training/validation split passed to each package; no configuration search.
CatBoost	300 iterations, depth 6, learning rate 0.1.	Fixed; no validation set or early stopping.
Random Forest	scikit-learn; 300 trees, remaining defaults.	Fixed; no validation set.
LightGBM/XGBoost	300 trees, maximum depth 6, learning rate 0.1; histogram XGBoost.	Fixed; no validation set or early stopping.
Logistic regression	scikit-learn; $C = 1$, at most 1000 iterations; SAGA for CC-18, TabArena-Lite, and BeyondArena, and L-BFGS for TALENT.	Fixed; no validation set.

FT-Transformer used its released 300-feature guard. On BeyondArena tasks above that limit, a deterministic training-variance screen retained the 300 highest-variance features. On CC-18 tasks above that limit, the guard was instead disabled and training used batch size eight; this was an execution setting, not test-set tuning.

E.3 Compute Resources

Experiments were distributed across NVIDIA GPUs on university compute servers. The fit/prediction measurements in Table 3 were collected on NVIDIA L4 GPUs, while the explanation measurements in Table 5, synthetic evaluation in Table 4, and regression evaluation in Table 6 were collected on NVIDIA A100 80 GB GPUs. Timing protocols use device synchronization; parallel benchmark execution was used only to reduce wall-clock time and is not treated as a latency measurement.

E.4 Dataset Licences

All benchmark datasets are publicly available. The CC-18 benchmark datasets are sourced from OpenML [27]; individual dataset licenses vary by uploader, ranging from CC0 (public domain) to CC BY 4.0. The TABICLV2 codebase [12] is used under the BSD 3-Clause License. All other software dependencies (scikit-learn, NumPy, etc.) are used under their respective open-source licenses (BSD/MIT).

E.5 Prediction Runtime

Table 3 compares three TICL runtime configurations. TABICLV2 and TabPFNv2 use their default ensemble sizes of eight and four estimators, respectively. All TICL variants use one TABICLV2 estimator and a batched five-fold out-of-fold reference bank. Both global-residual variants use $\alpha = 0.01$ and $\rho = 0.25$ at both stages, with 300 iterations for the shared fit. The standard configuration uses $k = 400$ and 300 local iterations; TICL-Fast uses $k = 30$ and 30 local iterations. In Table 3, it reduces prediction time from 6.849 to 1.495 s and increases accuracy by 0.52 percentage points relative to the standard global-residual configuration. Separate ablations show an accuracy–deletion-monotonicity trade-off for smaller neighbourhoods (Appendix G.1). Regional settings are given below. These runtime configurations use fixed-step FISTA without the backtracking used by the primary global model.

Timings were measured on NVIDIA L4 GPUs with one worker per GPU and one CPU math-library thread. CUDA synchronization brackets each timed operation, and measurements are averaged over three repetitions. Fit time includes model construction, deterministic preprocessing, backbone fitting, construction of the prediction-and-representation bank, and fitting of the shared ElasticNet or regional anchor models. Prediction time covers the full held-out batch, including query representations, neighbour retrieval and residual ElasticNet fitting for the global variant, or anchor selection and coefficient blending for the regional variant, followed by class-score clipping, probability normalization, and contribution construction. Metric evaluation is excluded. Seeds are averaged within each dataset before datasets are weighted equally. For reference, the one-estimator TABICLV2 operating point attains 0.8710 accuracy with 0.485 s fit and 0.745 s prediction time; the fast and regional variants therefore predict faster than the eight-estimator row in Table 3, but not than its single-estimator counterpart.

Regional anchor configuration. The regional variants in Tables 2, 3, and 6, and the diagnostics below, use up to 64 anchors. We apply k-means to normalized reference-bank representations, select the training row closest to each cluster centre, and remove duplicate selections. Each classification anchor model fits the backbone probability targets on its 200 nearest reference rows using weighted squared loss and ElasticNet with $\alpha = 0.05$ and $\rho = 0.5$. Neighbour weights use a softmax over cosine similarities with temperature 0.1. For each query, the eight most similar anchors receive softmax blending weights with temperature 0.05. Their coefficients and intercepts are combined before computing class probabilities and feature contributions. For regression, the same construction fits standardized scalar out-of-fold teacher predictions and omits probability clipping and normalization. Anchor fitting uses at most 300 iterations in Tables 2 and 6 and in the diagnostics below, and 50 in Table 3. The runtime measurements use the same L4 protocol described above.

Regional explanation diagnostics. We treat the anchor model as a computational variant rather than a coequal explanation method because its fitted bank often produces an all-zero predicted-class contribution vector. On the synthetic tasks, its support F1/accuracy pairs are $0.040/0.659$, $0.069/0.689$, $0.067/0.784$, and $0.204/0.708$ on E1–E4, respectively. Its mean support F1 is 0.095 ± 0.008 and its mean accuracy is 0.710 ± 0.009; only 26.7% of queries have nonzero coefficient vectors (14.5%, 20.3%, 22.9%, and 49.0% on E1–E4), with a median of one selected feature

when nonempty. Predictions can nevertheless vary through query-dependent routing and blended intercepts.

On the real-data benchmark, the regional model obtains deletion Spearman 0.381, Gen-SVM 0.715, Gen-kmeans 0.632, neighbour stability 0.044, and 1.828 ms/query online latency. Only 281 of 500 queries have nonzero contribution vectors; deletion and stability cover eight datasets, while generalization covers seven. These values use the regional model's own valid population and are therefore descriptive rather than directly comparable to Table 5.

E.6 Explanation Benchmark Setup

The explanation benchmark used in the main paper is a fixed 10-dataset subset selected before score-function tuning. Each dataset uses a stratified 80/20 split with seed 42, with the training and test sets capped at 1600 and 400 rows, and 50 explained test queries per dataset drawn with query seed 123 (500 queries in total). The subset, split, and query indices were then held fixed for all reported comparisons. The datasets are: `balance-scale`, `banknote-authentication`, `har`, `mnist_784`, `kr-vs-kp`, `vowel`, `eucalyptus`, `texture`, `letter`, and `jungle_chess_2pcs_raw_endgame_complete`.

E.7 Explanation Baselines and Deletion Protocol

The post-hoc explanation methods and TICL^3 (global + residual) target the same complete TICL^3 (global + residual) wrapper; TICL^3 (regional anchors) targets its own deployed regional wrapper. They are evaluated on the same ordered query blocks, with the same masking operator and the same fill value for a removed feature: the training-set mean of that raw feature on the corresponding split. The class predicted by the relevant unperturbed wrapper is frozen and used as the explained class for class-specific methods. Features are ranked once per query and removed cumulatively in that order to a depth of $\min(10, D_{\text{raw}})$, and we record the drop in the frozen-class probability after each step. During a perturbation, the original ordered query block is fixed and only the focal row is replaced. Prediction calls rerun the full relevant wrapper because the query representation and, for TICL^3 (global + residual), the neighborhood and residual correction may change. This is a deployment-oriented benchmark rather than a test of causal relevance.

For LIME, we use `LimeTabularExplainer` on the original training table with `discretize_continuous=False`, `sample_around_instance=True`, and 500 perturbation samples per explained query. For SHAP, we use KernelSHAP with a random background subset of 32 training examples and 64 evaluation samples per explained query, seeded independently per query. The global-residual wrapper is not differentiable end-to-end because it contains top-k retrieval and a per-query ElasticNet fit. Integrated Gradients (24 steps), GradientSHAP (24 samples), and Input$\times$Gradient are therefore computed on a differentiable surrogate, a two-hidden-layer MLP (width 256) fitted to the wrapper's class probabilities on the training rows only. The reported gradient rows consequently conflate attribution behavior with surrogate approximation error; held-out surrogate fidelity was not separately measured. FastSHAP and experimental adaptations of INVASE, L2X, and REAL-X are trained on the training rows with the wrapper's class probabilities as targets. These settings are kept fixed across datasets.

For TICL^3 (global + residual), processed-feature contributions are aggregated to raw semantic columns using the preprocessing map. TICL^3 (global + residual), LIME, KernelSHAP, FastSHAP, and the surrogate-gradient methods return a signed score per raw feature for the frozen class, which we rank in descending order. INVASE, L2X, and REAL-X instead return nonnegative, class-agnostic selector scores in these adaptations; we rank those scores in descending order. Their rows therefore evaluate selected-feature rankings, not signed class contributions.

E.8 Interpretability Metric Definitions

Let $\mathcal{Q}$ index the queries in the 10-dataset interpretability benchmark. For $q \in \mathcal{Q}$, let $\mathbf{x}_q \in \mathbb{R}^{D_{\text{raw}}}$ be the raw query, $\widehat{y}_q \in \{1, \ldots, C\}$ its frozen predicted class, and $\boldsymbol{\psi} : \mathbb{R}^{D_{\text{raw}}} \to \mathbb{R}^{D_{\text{raw}}}$ the raw-feature attribution map for class $\widehat{y}_q$ produced by a class-specific method, or the selector-score map for a selector method. We abbreviate its value at the query as $\boldsymbol{\psi}^{(q)} := \boldsymbol{\psi}(\mathbf{x}_q)$ and let π_q be the permutation of $\{1, \ldots, D_{\text{raw}}\}$ that orders features by decreasing method-specific score $\psi_j^{(q)}$, with deterministic tie breaking. Let $f_c(\mathbf{x}) \in [0, 1]$ denote the probability that the deployed wrapper for the evaluated row assigns to class c (TICL^3 (global + residual) for the non-regional rows and

TICL[3] (regional anchors) for its own row), and let $\mu_j = |\mathcal{T}|^{-1} \sum_{\mathbf{x} \in \mathcal{T}} x_j$ be the mean of raw feature j over the corresponding training split $\mathcal{T}$. Every metric is first averaged over the valid queries of a dataset and then averaged with equal weight across the contributing datasets. Undefined values are excluded rather than set to zero. The non-regional Gen-SVM, Gen-kmeans, and neighbour-stability entries use the 493-query intersection for which every included method returns a finite, nonzero attribution vector. Their deletion entry additionally requires a defined deletion correlation, leaving 422 common-valid queries across nine datasets. The regional row instead uses its own defined population: 281 queries across eight datasets for deletion and neighbour stability. Its generalization entries have 556 valid calibration rows and 281 valid evaluation rows, with both metrics defined on seven datasets. Consequently, the regional quality entries are not on the same query population as the non-regional rows.

Deletion monotonicity. Let $M = \min(10, D_{\text{raw}})$ and let $\mathbf{x}_q^{(m)}$ denote the query with its first m features under ranking π_q replaced by their training-set means. Define the cumulative probability drop

$$\Delta_m^{(q)} \;=\; f_{\widehat{y}_q}(\mathbf{x}_q) - f_{\widehat{y}_q}(\mathbf{x}_q^{(m)}), \qquad m = 1, \ldots, M.$$

Deletion Spearman is the per-query correlation between m and $\Delta_m^{(q)}$. Higher values indicate that progressive removal of highly ranked features produces a more consistently increasing decrease in the predicted-class probability. It measures monotonicity rather than the magnitude of the decrease: an arbitrarily small but consistently increasing drop can score highly. Deletion-based evaluation is used broadly in prior work [41–43]; these citations do not imply that each work uses this exact Spearman statistic.

Gen-SVM and Gen-kmeans. Following the local feature-selection evaluation of LSPIN [16], we use the budget $K = \min(10, \max(1, D_{\text{raw}} - 1))$. Let $J_q^{(K)}$ be the set containing the first K indices under π_q, and let $\mathbf{x}_q^{J_q^{(K)}}$ denote the masked input that retains $\mathbf{x}_q$ on $J_q^{(K)}$ and replaces every other entry j with μ_j. *Gen-SVM* trains a linear SVM ($C = 1$) on a calibration block of 100 training rows, each masked by its own explanation and labeled with the wrapper's predicted class. It reports agreement with $\widehat{y}_q$ from the masked evaluation queries. *Gen-kmeans* runs k-means with C clusters on the masked evaluation queries and reports the accuracy of the Hungarian-matched cluster assignment against the true labels. The two metrics therefore target different labels and are reported separately. Moreover, because every row is masked according to its own selected set, Gen-SVM can exploit the pattern of retained locations as well as retained values; we did not run a mask-only control.

Neighbour stability. Neighbour stability compares normalized attribution differences between nearby observed queries. Write $\mathbf{u} = \boldsymbol{\psi}/\|\boldsymbol{\psi}\|_1$. Let $\widetilde{\mathbf{h}}_q$ be the retrieval representation of query q, standardized with the training-row mean and standard deviation, and let $\mathcal{N}_3(q)$ contain the three nearest other evaluation queries in Euclidean distance. Then

$$L_{\text{nb}}(\mathbf{x}_q) = \max_{q' \in \mathcal{N}_3(q)} \frac{\|\mathbf{u}(\mathbf{x}_q) - \mathbf{u}(\mathbf{x}_{q'})\|_2}{\|\widetilde{\mathbf{h}}_q - \widetilde{\mathbf{h}}_{q'}\|_2},$$

and lower values indicate more similar explanations for nearby queries.

E.9 Feature-Importance Scoring

For class c, the processed-feature contribution is $a_{j,c}(\mathbf{x}^*) = (z_j^* - \bar{z}_j) W_{j,c}^*$ with $\mathbf{W}^* = \mathbf{P} + \mathbf{D}^*$. The raw-column contribution sums these terms over the common origin map g defined in Appendix E.1. Unless a protocol fixes c in advance, we use the class predicted for the unperturbed query. Raw contributions are ranked by signed value in the real-data benchmark (Appendix E.7); the synthetic support-recovery tasks instead rank the absolute transformed-space combined coefficients after associating them with their raw origins (Appendix E.10).

For the synthetic-task notation below, $\sigma(t) := (1 + \exp(-t))^{-1}$ is the logistic function, $\mathcal{N}(\mu, v)$ denotes a normal distribution with mean μ and variance v, q indexes a test query, and S_q is its designated generating-support set of raw-feature indices.

E.10 LSPIN Synthetic Tasks: E1–E4

Tasks E1–E4 are regime-dependent binary classification benchmarks adapted from the LSPIN synthetic evaluation [16]. Each task is generated with ten independent seeds, 21–30. Unless noted

otherwise, all features are drawn from independent standard normal distributions, and labels follow $y \sim \text{Bernoulli}(\sigma(\ell))$ where σ is the logistic function and ℓ is the task-specific logit defined below.

E1–E3: regime-switching with a gate feature (11 features). Each task uses $n = 2000$ samples with features $x_1, \ldots, x_{11} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$. The gate feature x_{11} determines which rule is active for each instance: the *gate branch* applies when $x_{11} < 0$ and the *off-gate branch* applies otherwise. Because x_{11} governs branch selection, it is included in the local support S_q of every query. Training uses 1800 samples and testing uses 200 samples; 100 test queries are sampled per seed.

E1. Gate branch ($x_{11} < 0$): $\ell = x_1 x_2$; support $S_q = \{x_1, x_2, x_{11}\}$ ($|S_q| = 3$). Off-gate branch ($x_{11} \geq 0$): $\ell = x_3^2 + x_4^2 + x_5^2 + x_6^2 - 4$; support $S_q = \{x_3, x_4, x_5, x_6, x_{11}\}$ ($|S_q| = 5$).

E2. Gate branch ($x_{11} < 0$): $\ell = x_1 x_2$; support $S_q = \{x_1, x_2, x_{11}\}$ ($|S_q| = 3$). Off-gate branch ($x_{11} \geq 0$): $\ell = -10 \sin(0.2\, x_7) + |x_8| + x_9 + e^{-x_{10}} - 2.4$; support $S_q = \{x_7, x_8, x_9, x_{10}, x_{11}\}$ ($|S_q| = 5$).

E3. Gate branch ($x_{11} < 0$): $\ell = x_3^2 + x_4^2 + x_5^2 + x_6^2 - 4$; support $S_q = \{x_3, x_4, x_5, x_6, x_{11}\}$ ($|S_q| = 5$). Off-gate branch ($x_{11} \geq 0$): $\ell = -10 \sin(0.2\, x_7) + |x_8| + x_9 + e^{-x_{10}} - 2.4$; support $S_q = \{x_7, x_8, x_9, x_{10}, x_{11}\}$ ($|S_q| = 5$).

E4: two-group design with 46 nuisance features (50 features total). E4 uses $n = 1000$ samples drawn from two equally-sized groups. Features $x_5, \ldots, x_{50}$ are independent nuisance variables with $x_j \sim \mathcal{N}(0, 0.5)$ (variance 0.5). The first four features form the informative core, with group-shifted distributions that make group membership recoverable from the data:

Group A ($n = 500$): $x_1, x_2, x_3, x_4 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(+1,\, 0.5)$. Logit: $\ell = x_1 x_2 - 0.9$. Generating-group conditional support: $S_q = \{x_1, x_2\}$ ($|S_q| = 2$).

Group B ($n = 500$): $x_1, x_2, x_3, x_4 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(-1,\, 0.5)$. Logit: $\ell = x_3^2 + x_4^2 - 2.5$. Generating-group conditional support: $S_q = \{x_3, x_4\}$ ($|S_q| = 2$).

Support follows the latent generating group. Because all four core variables shift with group, variables outside this conditional support can reveal group membership and therefore carry information about the observable $P(y \mid \mathbf{x})$; S_q should not be read as its unambiguous support. We randomly split each group into 450 training and 50 test rows, yielding 900 training rows and 100 balanced test queries.

Evaluation protocol. All tasks use one TABICLV2 estimator, one representation view, the primary reference bank, and squared loss. The TICL^3 (global + residual) row uses $k = 400$, $\tau = 0.1$, $\alpha = 0.01$, and $\rho = 0.25$; TICL^3 (regional anchors) uses the anchor configuration in Appendix E.5. For query q, let A_q contain up to $|S_q|$ raw-feature indices with the largest nonzero absolute predicted-class coefficients after associating each processed coefficient with its raw-column origin. The reported metric is conventional set F1,

$$\text{F1}_q = \frac{2|A_q \cap S_q|}{|A_q| + |S_q|}, \qquad \text{F1}_q = 0 \text{ when } A_q = \varnothing.$$

For post-hoc explainers, the selected features are used to compute masked-TABICLV2 accuracy. Results average queries and seeds per task. Aggregate standard errors use ten seed-level averages over the four tasks.

TICL^3 (global + residual) ranks nonzero absolute combined raw coefficients; MAPLE ranks absolute local coefficients; EBM ranks absolute additive scores (no interactions); and logistic regression ranks absolute mean-relative standardized contributions. LIME, KernelSHAP, Integrated Gradients, and GradientSHAP use the absolute scores returned for the teacher's predicted class. Interpretable models report their own accuracy; post-hoc methods report TABICLV2 accuracy after mean-masking unselected features. MAPLE selects its feature count on a validation split within the training data.

Gate-feature interpretation. The gate x_{11} belongs to every E1–E3 generating support, but may have a small combined contribution when a neighborhood lies within one regime. Generating support and local contribution magnitude therefore need not coincide.

F Additional Background and Related Work

We organize prior work along three axes that are central to this paper: whether the predictor is *interpretable by design* or requires post-hoc analysis, whether the predictive form exposes explicit feature contributions, and whether predictions adapt locally to each query. We distinguish post-hoc explanations, which analyze a fixed predictor after inference, from interpretability by design, where the prediction itself is computed through an inspectable structure. Table 1 summarizes representative methods under this view and positions TICL^3 as an interpretable-by-design, global-plus-local linear predictor.

Tabular in-context learning models. Recent work has introduced a new paradigm for tabular prediction in which a labeled dataset is provided as context at inference time. Given a labeled context $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with raw input $\mathbf{x}_i \in \mathbb{R}^{D_{\text{raw}}}$ and class label $y_i \in \{1, \dots, C\}$, and a query input $\mathbf{x}^* \in \mathbb{R}^{D_{\text{raw}}}$, these models directly predict

$$p_\theta\big(y^* \mid \mathbf{x}^*, \{(\mathbf{x}_i, y_i)\}_{i=1}^N\big)$$

in a single forward pass, without dataset-specific training. Here y^* is the unknown query label, θ denotes the pretrained model parameters, and $p_\theta(\cdot \mid \cdot)$ is the resulting conditional class distribution. This corresponds to in-context learning, in which adaptation to a new task is performed by conditioning on labeled examples rather than by gradient updates.

During pretraining, models are exposed to a distribution of datasets and optimized to predict labels for held-out examples given their context, encouraging a general-purpose inference procedure. Architecturally, different designs vary in how tabular data is represented and how interactions are modeled. TabPFN [2] represents rows as tokens and performs attention across samples, while TabPFNv2 [19] introduces a cell-based design with alternating row and column attention, increasing flexibility at a higher computational cost. TABICL [3] improves scalability through a two-stage design that first constructs row embeddings and then performs attention over them. Throughout this work, we use TABICLV2 [12], the successor of TABICL, as our backbone and primary baseline.

These models are strong predictors, but their decision-making process is not interpretable by design: predictions are computed via multi-layer attention rather than through explicit feature-level operations. In Table 1, they therefore lie outside the interpretable-by-design, linear, and local region. Rather than attempting to explain these models post hoc, TICL^3 reuses their context-conditioned representation geometry to define locality, while constructing a separate predictor whose computation is interpretable by design.

Post-hoc local explanation methods. Post-hoc explanation methods approximate a trained predictor after it has produced a prediction. Model-agnostic approaches such as LIME [5] and SHAP [6] construct local surrogate models or estimate feature attributions by perturbing the input. These methods operate at the input-output level and do not expose the internal computation that produces the prediction. As a result, they require repeated evaluations of an external predictor and define locality independently of its internal representation.

Gradient-based attribution methods, including Integrated Gradients [7], DeepLIFT and Input $\times$ Gradient [8], and GradientSHAP-style estimators [44], instead use differentiability to assign feature importance scores. A separate line of work amortizes explanation through learned selector–predictor models that return instance-wise feature subsets, including L2X [45], INVASE [46], REAL-X [47], and FastSHAP [9]; ILLUME [10] similarly learns meta-representations for local surrogate explanations. LIME, SHAP, and gradient attribution analyze a fixed predictor after inference. The original L2X, INVASE, and REAL-X formulations instead train selector and predictor components jointly or in coordinated objectives; our Table 5 adaptations train their selectors against the fixed TICL^3 (global + residual) wrapper outputs. In each case, the selector or explainer score is distinct from the explicit rule used by that wrapper to make its prediction.

Interpretable-by-design models and local predictors. Interpretable-by-design models expose feature-level structure directly in the deployed predictor. Classic examples include sparse linear models, such as Least Absolute Shrinkage and Selection Operator (Lasso) [23], and additive models, such as Explainable Boosting Machines (EBMs) and Neural Additive Models (NAMs) [1, 14]. These models are interpretable by design and often partially linear, but they are typically global: a single model is applied across all queries.

Local modeling approaches instead fit simple predictors in a neighborhood around each query. Methods such as LOESS [24] and local polynomial regression [48] often use input-space distance. MAPLE [15] learns a supervised forest geometry and derives query-dependent proximity weights before fitting a local model. Its forest is fixed after training, but its neighborhood is label-informed and changes with the query. In contrast, TICL^3 recomputes neighborhoods from representations conditioned on the current labeled ICL context; the distinction is the context-conditioned geometry, not an absence of query adaptation in MAPLE.

LSPIN and its locally linear variant LLSPIN [16] obtain query adaptation differently. Building on stochastic-gate feature selection [49], a gating network trained jointly with the prediction network maps each sample to gates that select a sparse subset of its features. When the prediction network is linear (LLSPIN), the deployed model is a sample-specific sparse linear rule whose coefficients are the shared weights multiplied by the sample's gates. Locality is thus learned from the labels by a dataset-specific gating network, whereas TICL^3 obtains it from the representation of a pretrained in-context model and fits each local rule with a per-query ElasticNet. Our synthetic tasks and selected-feature utility metrics are adapted from this work (Section 4.3 and Appendix E.8).

TICL^3 differs from these prior directions in how locality is defined and reused. A shared ElasticNet model captures dataset-level structure, while context-conditioned representations from a pretrained tabular in-context learner determine which training rows fit each query's residual correction. The deployed predictor is the sum of this global coefficient matrix and a query-specific ElasticNet correction in the processed feature space. It is therefore interpretable by design and query-adaptive, with locality grounded in the foundation model's learned geometry rather than raw input distances.

Interpretable extensions of tabular in-context models. Recent work has explored incorporating explanations directly into tabular in-context models. ShapPFN [18], a TabPFN-style model pretrained on synthetic datasets, decodes a base term and per-feature, per-class contributions from the test-row embeddings and sums them to obtain its logits. Its prediction is therefore additive over features by construction, and predictions and contributions are produced in a single forward pass. ExplainerPFN [13] instead targets model-free feature-importance estimation: it predicts Shapley-style attributions for a new dataset without access to the model being explained.

ShapPFN is interpretable by design in the additive sense of Table 1, because its displayed contributions sum to its output. Each contribution, however, is computed by attention layers over the full labeled context, so how a term depends on its feature value is not itself inspectable, and no explicit feature-level rule is deployed. In TICL^3, the contribution of processed feature j to class c is the product $(z_j^* - \overline{z}_j)W_{j,c}^*$ of the centered feature value and an explicit coefficient; the foundation model only supplies prediction targets and defines the neighborhood. ExplainerPFN, like other post-hoc methods, produces attributions that do not determine a deployed prediction.

Example-based explanations. Example-based methods explain predictions by identifying similar training instances, as in prototype and metric-based approaches [50]. These methods provide sample-level evidence but do not produce explicit feature-level contributions. TICL^3 retains the neighborhood view but uses it to adapt a shared linear predictor through a residual correction, yielding both sample-level locality and feature-level contributions within a single interpretable-by-design model.

G Explanation Runtime Accounting

The runtime column of Table 5 reports equal-dataset means. The online boundary includes the complete predictor call and construction of the attribution, but excludes one-off model and explainer setup. TICL and the seven learned or gradient explainers use the same 50 original queries per dataset, measured with device synchronization over three repeats on two NVIDIA A100 80 GB GPUs. LIME and KernelSHAP use the first 10 fixed original queries from each of the same 10 datasets, also with three isolated, device-synchronized repeats; their attribution times are added to the common predictor latency measured on the 50-query blocks. No stability-perturbed rows enter this timing subset. LIME retains 500 perturbations per explanation and KernelSHAP retains 64 evaluation samples with a 32-row background. Extrapolating the measured equal-dataset mean attribution times to 500 queries and three repetitions gives approximately 28.5 A100-hours for LIME and KernelSHAP combined, motivating the fixed 100-query timing subset.

Table 8: Component and configuration ablations. Bold identifies the primary configuration, not column-wise best results. Iteration caps apply to the local correction; shared global fits retain 300 iterations. A nonzero explanation has a finite contribution vector with maximum absolute contribution above 10^{-12}. Explanation metrics use configuration-specific valid populations and equally weight contributing datasets. Gen-SVM targets each predictor's own classes; Gen-kmeans uses true labels. Evaluation counts appear in Table 9.

Configuration	Acc. (%)↑	Nonzero expl. (%)↑	Deletion Spearman↑	Gen-SVM (%)↑	Gen-kmeans (%)↑
Global and local components					
Local ElasticNet only ($k = 400$, matched)	90.18	68.0	0.5207	70.72	55.77
Global ElasticNet only	76.13	100.0	0.6097	95.40	67.80
TICL — primary ($k = 400$, 300 iterations)	**89.90**	**100.0**	**0.5297**	**88.80**	**71.40**
Local reference configuration					
Local ElasticNet reference ($k = 100$)	90.29	41.8	0.2807	53.03	52.97
Correction neighbourhood and iteration cap					
TICL ($k = 30$, 300 iterations)	91.19	100.0	0.3993	81.60	69.40
TICL ($k = 400$, 30 iterations)	89.92	100.0	0.5323	89.20	72.20
TICL ($k = 30$, 30 iterations)	90.97	100.0	0.4194	82.60	69.60
Regional blending					
Regional — nearest anchor	87.88	44.6	0.2867	61.27	52.06
Regional — eight-anchor blend	89.21	55.4	0.3975	71.69	61.83

The exact query-parallel TICL^3 (global + residual) implementation prepares the GPU query state once per call and parallelizes only the independent residual ElasticNet solves. It uses the primary $k = 400$, 300-iteration, 10^{-4}-tolerance configuration. On all ten datasets and all repeats, its predicted classes matched the frozen classes, and its probabilities, coefficients, and contributions had zero measured difference across repeats. Its equal-dataset mean joint prediction and contribution-construction latency is 26.974 ms/query. The corresponding latency of TICL^3 (regional anchors) is 1.828 ms/query; all 500 outputs are finite, while 281 have a nonzero predicted-class contribution vector. The corresponding end-to-end means are 27.197 for FastSHAP, 27.482 for Integrated Gradients, 27.560 for GradientSHAP, 27.112 for Input×Gradient, 27.191 for INVASE, 27.213 for L2X, and 27.192 ms/query for REAL-X. After adding the common predictor call, the matched LIME and KernelSHAP measurements are 14,607.496 and 53,686.935 ms/query, respectively. Across the three complete repeats, the standard deviations of the equal-dataset means are 19.407 and 149.744 ms/query.

One-off method-specific construction is kept separate from online latency because its per-query amortization depends on deployment volume. The measured equal-dataset mean construction times are 0.003 s for LIME, 1.024 s for KernelSHAP, 58.366 s for FastSHAP, 104.961 s for the shared differentiable surrogate used by Integrated Gradients, GradientSHAP, and Input×Gradient, 41.077 s for INVASE, 41.181 s for L2X, and 71.341 s for REAL-X. The gradient-surrogate values retain their cached construction provenance; all other construction values are means of the three new construction measurements. The backbone, reference-bank, and shared-ElasticNet fit is common to every method and is not charged only to TICL^3 (global + residual).

Panel (c) of Figure 1 adds method-specific construction to the 50-query online workload while excluding shared predictor construction. This gives 1.349 s for TICL^3 (global + residual) and 42.436 s for INVASE, the fastest post-hoc total, corresponding to a $31.5\times$ advantage. The analogous LIME and KernelSHAP totals, extrapolated from their fixed 10-query-per-dataset timing subset, are 730.378 and 2,685.371 s, or $542\times$ and $1,991\times$ the TICL^3 (global + residual) total.

G.1 Component and Configuration Ablations

We evaluate nine configurations on the ten explanation-benchmark datasets, using seed 42, full-test accuracy and 50 fixed explanation queries per dataset. All configurations share the same single-estimator FP32 TabICLv2 teacher and batched five-fold OOF reference bank. Unlike the ablation, the regional evaluation in Appendix E.5 uses automatic mixed precision and sequential five-fold OOF. The separate regional evaluation and this ablation use the same 500 query identities but different fitted banks, yielding 281 and 277 nonzero regional explanations, respectively. Explanation metrics follow Appendix E.8, with each configuration explaining its own predictor.

The primary configuration combines a shared global ElasticNet with a local residual correction using $k = 400$ and at most 300 iterations. Matched local-only fitting removes the global component while retaining the local neighbourhood, regularization and iteration budget. Global-only fitting removes the local correction. The regional comparison holds the fitted anchor bank fixed and changes only the blending rule.

Table 9: Evaluation populations for the ablation. Every configuration has 500 attempted explanation queries and full-test accuracy on all ten datasets. Dataset counts refer to deletion Spearman, Gen-SVM and Gen-kmeans, respectively.

Configuration	Valid deletion /500	Contributing datasets
Matched local K400	336	9 / 9 / 9
Global-only	500	10 / 10 / 10
TICL — primary	488	10 / 10 / 10
Local K100	209	7 / 6 / 6
TICL ($k = 30$, 300 iterations)	493	10 / 10 / 10
TICL ($k = 400$, 30 iterations)	488	10 / 10 / 10
TICL ($k = 30$, 30 iterations)	491	10 / 10 / 10
Regional — nearest anchor	222	7 / 5 / 6
Regional — eight-anchor blend	277	8 / 7 / 7

Global and local components. Adding the local correction raises accuracy by 13.77 percentage points over global-only fitting. Against matched local K400, the primary model produces more nonzero explanations but has slightly lower accuracy. Its paired Deletion Spearman difference is 0.027, with a 95% dataset-bootstrap interval of $[-0.078, 0.114]$, indicating no clear difference in deletion monotonicity. Higher reported Gen scores use different eligible populations and should not be interpreted as matched comparisons. Global-only fitting has higher deletion and Gen-SVM scores despite lower accuracy, illustrating the need to assess explanation diagnostics alongside predictive performance. Local K100 is a reference setting rather than an isolated component ablation because its regularization also differs.

Neighbourhood and iteration cap. At 300 iterations, K30 improves accuracy by 1.29 percentage points, while K400 improves paired Deletion Spearman by $0.130\,[0.027, 0.234]$. This supports an accuracy–deletion-monotonicity trade-off on this subset. At K400, 30 iterations yield similar observed results: the paired Spearman difference for 30 minus 300 iterations is $-0.0003\,[-0.0104, 0.0093]$. The larger cap therefore has no demonstrated advantage in this experiment; the interval does not establish equivalence.

Regional blending. Using eight anchors instead of the nearest anchor raises accuracy from 87.88% to 89.21% and the fraction of nonzero explanations from 44.6% to 55.4%. Its paired Deletion Spearman difference is $0.031\,[-0.116, 0.157]$, so improved deletion monotonicity is not established. This comparison isolates blending within the regional model, not regionalization against the primary model. These ablations do not measure runtime or establish a universally optimal configuration.

Settings. Matched local K400 and all global models use $\alpha = 0.01$ and L1 ratio 0.25; local K100 uses $\alpha = 0.05$ and L1 ratio 0.5. Local and global models use safeguarded FISTA with tolerance 10^{-4}; neighbour weighting uses temperature 0.1. Shared global and local-only fits retain 300-iteration caps. Regional variants share up to 64 anchors fitted on 200 neighbours with FISTA, 300 iterations, $\alpha = 0.05$ and L1 ratio 0.5. Eight-anchor blending uses temperature 0.05. These settings are distinct from runtime configurations using different solvers or iteration budgets.

Valid populations and uncertainty. Empty explanations and undefined correlations are excluded rather than assigned zero. The primary model's 12 undefined deletion correlations have constant probability-drop curves; all its explanations are nonzero. The primary value 0.5297 above uses its 488 own-valid queries across all ten datasets. Table 5 instead reports 0.470 on the 422-query cross-method intersection over nine contributing datasets; on its own valid population there, the primary score is 0.5283. Paired comparisons use common-valid query indices, with each model's original predicted class fixed separately. Intervals use 10,000 dataset-bootstrap resamples; they are exploratory, unadjusted for multiple comparisons and conditional on one split seed.