xRFM: Accurate, scalable, and interpretable feature learning models for tabular data

Daniel Beaglehole
Computer Science and Engineering
Halıcıoğlu Data Science Institute
UC San Diego
dbeaglehole@ucsd.edu

David Holzmüller
INRIA Paris
École Normale Supérieure
PSL University
david.holzmuller@inria.fr

Adityanarayanan Radhakrishnan
MIT Mathematics
Broad Institute of MIT and Harvard
aradha@mit.edu

Mikhail Belkin
Halıcıoğlu Data Science Institute
UC San Diego
mbelkin@ucsd.edu

Abstract

Inference from tabular data, collections of continuous and categorical variables organized into matrices, is a foundation for modern technology and science. Yet, in contrast to the explosive changes in the rest of AI, the best practice for these predictive tasks has been relatively unchanged and is still primarily based on variations of Gradient Boosted Decision Trees (GBDTs). Very recently, there has been renewed interest in developing state-of-the-art methods for tabular data based on recent developments in feature learning methods. In this work, we introduce xRFM, an algorithm that combines feature learning kernel machines with a tree structure to scale to essentially unlimited amounts of training data. On the TALENT benchmark, we show that xRFM achieves best performance across 100 regression datasets and is competitive across 200 classification datasets, outperforming GBDTs.

1 Introduction

Tabular data – collections of continuous and categorical variables organized into matrices – underlies all aspects of modern commerce and science from airplane engines to biology labs to bagel shops. Yet, while Machine Learning and AI for language and vision have seen unprecedented progress, the primary methodologies of prediction from tabular data have been relatively static, dominated by variations of Gradient Boosted Decision Trees (GBDTs), such as XGBoost [6]. Nevertheless, hundreds of tabular datasets have been assembled to form extensive regression and classification benchmarks [5, 13, 10, 25, 9], and, recently, there has been renewed interest in building state-of-the-art predictive models for tabular data [14, 15, 12].

In this work, we introduce xRFM, a tabular predictive model that combines recent advances in feature learning kernel machines with an adaptive tree structure, making it effective, scalable, and interpretable. xRFM builds upon the Recursive Feature Machine (RFM) algorithm from [20], which enabled *feature learning* (a form of supervised dimensionality reduction) in general machine learning models. Given training data, xRFM first builds a binary tree structure to split data into subsets based on features relevant for prediction within each split. When splits reach a certain size, we train a *leaf* RFM (a hyper-parameter and compute optimized version of RFM).

In practice, we show xRFM has the best performance across 100 tabular regression tasks and is competitive with state-of-the-art on 200 tabular classification tasks from the TALENT benchmark [25]. On the TabArena-Lite benchmark [9], we show xRFM empirically achieves one of the best tradeoffs (is on the empirical Pareto frontier) between performance and inference time among all methods in regression and is again competitive on classification. We show that xRFM achieves similar results on the largest datasets from the meta-test benchmark [15], where directly solving standard kernel machines becomes intractable on standard GPUs.

AI for Tabular Data workshop at EurIPS 2025.

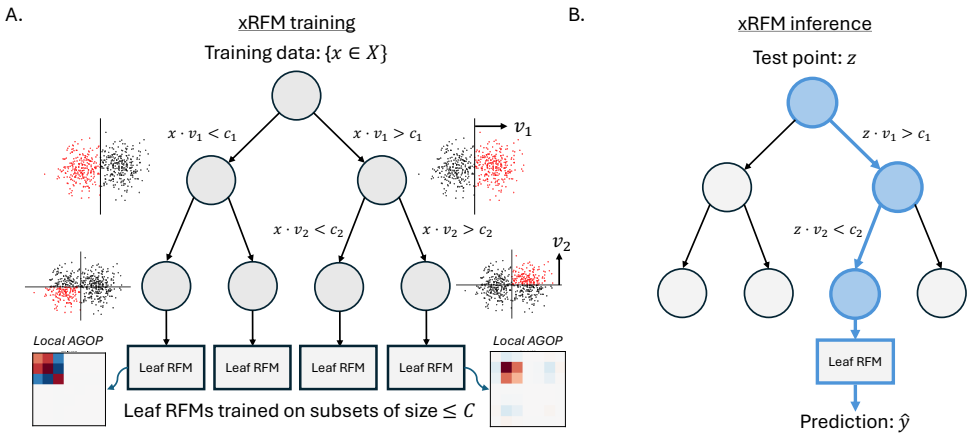

Figure 1: Overview of xRFM training and inference procedures. (A) xRFM is trained by splitting the data along the median projections (denoted c_1, c_2) onto computed split directions (denoted v_1, v_2). Data is split repeatedly into leaves, which contain at most C training samples. Leaf RFMs are trained on the data at each leaf. (B) During inference, test data is routed to the appropriate leaf RFM based on split directions. The prediction is generated by the selected leaf RFM.

2 Preliminaries

We review kernel machines and kernel-RFM, which we use to build xRFM.

Kernel machines A kernel machine is a non-parametric machine learning model [23, 2]. The idea behind kernel machines is that one can train a nonlinear predictive model by first transforming input data with a fixed, nonlinear feature map and then performing linear regression on the transformed data. Kernel machines make this procedure computationally tractable even for infinite dimensional feature maps by using kernel functions (inner products of feature mapped data). We describe kernel machines in the context of supervised learning below. Let $X \in \mathbb{R}^{n \times d}$ denote training inputs with ${x^{(i)}}^T$ denoting the example in the i^{th} row of X for $i \in [n]$ and $y \in \mathbb{R}^{n \times c}$ denote training labels. Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ denote a kernel function (a positive semi-definite, symmetric function). Given a regularization parameter $\lambda \geq 0$, a kernel machine trained on the data (X, y) is a predictor, $\hat{f} : \mathbb{R}^d \to \mathbb{R}^c$, of the form: $\hat{f}(x) = K(x, X)\alpha; \quad \alpha = (K(X, X) + \lambda I)^{-1} y$; where the notation $K(x, X) \in \mathbb{R}^{1 \times n}$ denotes an n-dimensional row vector with $K(x, X)_i = K(x, x^{(i)})$ and $K(X, X) \in \mathbb{R}^{n \times n}$ denotes a matrix with $K(X, X)_{ij} = K(x^{(i)}, x^{(j)})$. A typical kernel functions used in practice is, e.g., the Gaussian kernel ($K(x, z) = \exp(-\|x - z\|_2^2)/L^2$).

Recursive Feature Machines (RFMs) The ability to learn task-relevant features from data is key to building effective predictors [7, 11, 1]. RFM, introduced in [20], is an algorithm that enables feature learning in general machine learning models through a mathematical object known as the Average Gradient Outer Product (AGOP). Given a predictive model $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ and data $S = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^d$, the AGOP is defined as $\text{AGOP}(\hat{f}, S) = \frac{1}{n} \sum_{i=1}^{n} \nabla \hat{f}(x^{(i)}) \nabla \hat{f}(x^{(i)})^T \in \mathbb{R}^{d \times d}$ where $\nabla \hat{f}(x^{(i)})$ denotes the gradient of $\hat{f}$ at the point $x^{(i)}$. The AGOP is an estimate of the (un-centered) covariance of the gradients of $\hat{f}$ and intuitively captures the subspace along which the predictor highly varies [24, 16]. The RFM algorithm involves iterating between training a predictive model and using the AGOP of the trained model to select features and linearly transform input data.

In the case of kernel machines, which have no native mechanism for feature learning, RFM enables feature learning by adapting the kernel to the data. We describe the kernel-RFM algorithm below. Following the notation in the previous section, let (X, y) denote training inputs and labels, let K denote the kernel function, and λ denote the regularization parameter. Letting $M_1 = I$ and $c > 0$, kernel-RFM repeats the following two steps for T iterations: (Step 1) Compute $\hat{f}_t(x) = K(M_t x, X M_t)\alpha_t$ and $\alpha_t = [K(XM_t, XM_t) + \lambda I]^{-1} y$, (Step 2) Compute $M_{t+1} = \left[\text{AGOP}(\hat{f}_t(M_t x), X)\right]^c$. We provide additional details on the hyperparamters of this algorithm in Appendix B.2.

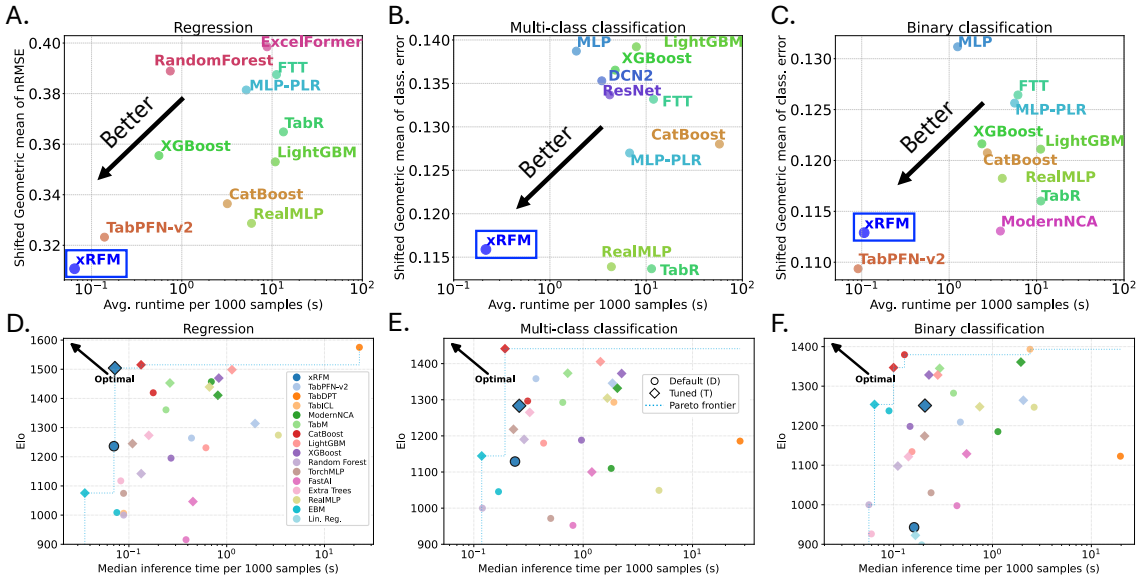

Figure 2: Performance and runtime of xRFM on the TALENT (Plots A-C) and TabArena-Lite benchmarks (Plots D-F). The x-axes are the average over all datasets of the training plus inference time per 1000 samples for just the best hyperparameter configuration (meaning if a dataset has n samples, we compute the training and inference time on the n samples divide the total time by $n/1000$).

3 xRFM algorithm overview

We now describe our algorithm xRFM, which consists of the following two components (Fig. 1): (1) An improved kernel-RFM, termed *leaf* RFM, that is trained on the subset of data, and (2) A binary tree that splits the data into subsets, termed *leaves*, of a maximum size (*leaf size*) that is independent of the number of training samples. Leaf RFMs are then trained on the subset at each leaf. The splits stratify data based on features relevant for prediction. We detail these two components below.

Leaf RFM Here, we describe the changes we made to the original kernel RFM algorithm from [20] to produce leaf RFMs. The kernel RFM model was primarily built using kernels that were invariant to orthonormal transformations of data. Such invariance does not effectively leverage a special property of tabular data – the fact that each coordinate can be independently meaningful. To account for this, we introduce the following modifications: (1) We tune over a more general class of kernels $K_{p,q}(x, x') = \exp(-\|x - x'\|_p^q / L^q)$ that are positive definite for $0 < q \leq p \leq 2$ [22, Theorems 1, 5], and (2) we tune over using the full AGOP and just the diagonal of the AGOP.

In addition to the above changes, we also implemented optimizations to speed up computations for categorical variables and an adaptive approach for tuning bandwidth separately for data on each leaf. Additional details and the selected hyperparameter search spaces are provided in Appendix B.

Tree-based data partitioning We now explain how xRFM builds a binary tree to partition data. First, given a dataset S with n samples, we subsample m points and train a leaf RFM (called a *split* model) on this subsample. We extract the top eigenvector, v, of the AGOP from the split model and create two subsets of the n datapoints: $S_1 = \{x \in S \; ; \; v^T x > \text{Median}(v^T z \text{ for } z \in S)\}$ and $S_2 = \{x \in S \; ; \; v^T x \leq \text{Median}(v^T z \text{ for } z \in S)\}$. We repeat this procedure on S_1 and S_2 and their corresponding children until all leaves are less than a maximum leaf size C. We finally train a leaf RFM on each of these leaves. This procedure is illustrated in Fig. 1A and detailed in the 'TreePartition' procedure of Algorithm B.2.

A key advantage of our split approach is that it groups together data points based on the features most relevant for prediction, as is captured by the top eigenvector of the AGOP. Prior works have also split data using the random forest procedure. For example, the authors of [14] train TabPFN-v2 on the samples routed to a given leaf of a tree in a random forest. The procedure in xRFM differs as it stratifies samples by projection onto a direction rather than individual coordinates and produces only one balanced tree instead of a forest.

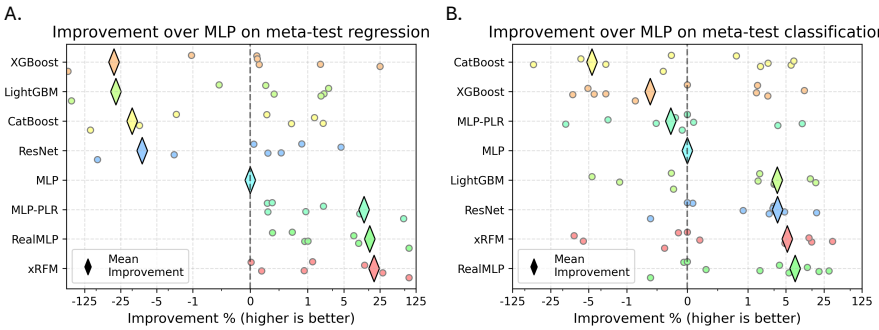

Figure 3: Performance comparison across the 17 large datasets from meta-test (70,000-500,000 samples). (A, B) Percentage improvement over MLP error on (A) regression and (B) classification datasets. Average percentage improvement is denoted as a diamond. Points denote individual dataset results (points are jittered for visibility).

4 xRFM results

We now apply xRFM to three tabular data benchmarks: the TALENT benchmark [25], the TabArena-Lite benchmark [9], and large datasets from the meta-test benchmark [15]. We use TALENT and TabArena-Lite to evaluate xRFM on datasets of various sizes between 500 and $150,000$ training samples. We use the meta-test benchmark to evaluate xRFM on larger datasets with at least $70,000$ and up to $500,000$ samples (a setting in which direct linear solvers become intractable, taking more than 40GB VRAM). We describe the performance measures used for these benchmarks in Appendix B.

Results on the TALENT and TabArena-Lite benchmarks. The TALENT benchmark consists of 300 total datasets comparing 31 different supervised learning algorithms [25]. Among these 31 algorithms are strong, widely used predictive models including Gradient Boosted Decision Tree (GBDT) variants (like XGBoost, CatBoost, LightGBM), hyper-parameter optimized neural networks (RealMLP), and recent transformer-based foundation models (TabPFN-v2). TabArena-Lite contains 51 total datasets comparing 15 different supervised learning algorithms including many of those in TALENT.

xRFM is the best performing method on TALENT regression tasks according to all aggregation metrics over RMSEs on individual datasets (Fig. 2A). It is also the fastest method per configuration, although TabPFN-v2 does not incur a $100\times$ overhead for hyperparameter tuning. xRFM is also competitive on classification datasets (Fig. 2B, C). We also compare the performance (in Elo) and inference time of xRFM using the TabArena-Lite benchmark (Fig. 2). In particular, when compared to all default and tuned methods for this benchmark, tuned xRFM is among the top three methods for regression while being orders of magnitude faster for inference. Indeed, xRFM lies along the empirical Pareto frontier, meaning that there is no method that dominates xRFM in both performance and inference time for these tasks (Fig. 2D). For classification tasks, xRFM is near the empirical Pareto frontier (Fig. 2E, F).

Results on large datasets from the meta-test benchmark. To analyze xRFM performance on large datasets beyond those in TALENT and TabArena, we consider the 17 largest datasets in the meta-test benchmark from [15] (Table C.6). We compare the performance of xRFM to other models with results reported in the literature (Fig. 3). On this benchmark, xRFM is best on regression tasks, and second best on classification tasks. All results are presented in Tables C.6, C.7.

Interpretability with xRFM An advantage of xRFM is that it immediately provides a means of identifying features relevant for prediction (without stacking on any additional interpretability methods). Namely, we can extract learned AGOPs of leaf RFMs and visualize the features they select. We show examples of this interpretability in Appendix A.

5 Discussion

Overall, we have shown that xRFM is an effective algorithm for inference from tabular data that scales to essentially unlimited data sizes and achieves performance exceeding or comparable to the current state-of-the-art. It combines the advantages of tree-based methods with the power and elegance of feature-learning kernel machines. We envision that xRFM will be used for both high-performing predictive modeling and uncovering heterogeneous structure in large-scale tabular data.

References

[1] E. Abbe, E. B. Adsera, and T. Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In P.-L. Loh and M. Raginsky, editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 4782–4887. PMLR, 02–05 Jul 2022.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[3] D. Beaglehole, A. Radhakrishnan, E. Boix-Adserà, and M. Belkin. Toward universal steering and monitoring of ai models, 2025.

[4] D. Beaglehole, P. Súkeník, M. Mondelli, and M. Belkin. Average gradient outer product as a mechanism for deep neural collapse. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 130764–130796. Curran Associates, Inc., 2024.

[5] B. Bischl, G. Casalicchio, T. Das, M. Feurer, S. Fischer, P. Gijsbers, S. Mukherjee, A. C. Müller, L. Németh, L. Oala, L. Purucker, S. Ravi, J. N. van Rijn, P. Singh, J. Vanschoren, J. van der Velde, and M. Wever. Openml: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7), 2025.

[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[7] A. Damian, J. Lee, and M. Soltanolkotabi. Neural networks can learn representations with gradient descent. In P.-L. Loh and M. Raginsky, editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 5413–5452. PMLR, 02–05 Jul 2022.

[8] A. E. Elo. The proposed uscf rating system, its development, theory, and applications. *Chess life*, 22(8):242–247, 1967.

[9] N. Erickson, L. Purucker, A. Tschalzev, D. Holzmüller, P. M. Desai, D. Salinas, and F. Hutter. Tabarena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025.

[10] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[11] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari. When do neural networks outperform kernel methods?*. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124009, Dec. 2021.

[12] Y. Gorishniy, A. Kotelnikov, and A. Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling, 2025.

[13] L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 507–520. Curran Associates, Inc., 2022.

[14] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[15] D. Holzmüller, L. Grinsztajn, and I. Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37:26577–26658, 2024.

[16] S. Kpotufe, A. Boularias, T. Schultz, and K. Kim. Gradients weights improve regression and classification. *Journal of Machine Learning Research*, 17(22):1–34, 2016.

[17] S. Ma and M. Belkin. Kernel machines that adapt to gpus for effective large batch training. *Proceedings of Machine Learning and Systems*, 1:360–373, 2019.

[18] N. Mallinar, D. Beaglehole, L. Zhu, A. Radhakrishnan, P. Pandit, and M. Belkin. Emergence in non-neural models: grokking modular arithmetic via average gradient outer product, 2024.

[19] A. Narasimha, B. Vasavi, and H. M. Kumar. Significance of nuclear morphometry in benign and malignant breast aspirates. *International Journal of Applied and Basic Medical Research*, 3(1):22–26, 2013.

[20] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

[21] A. Radhakrishnan, M. Belkin, and D. Drusvyatskiy. Linear recursive feature machines provably recover low-rank matrices. *Proceedings of the National Academy of Sciences*, 122(13):e2411325122, 2025.

[22] I. J. Schoenberg. Positive definite functions on spheres. *Duke Mathematical Journal*, 9(1):96 – 108, 1942.

[23] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[24] S. Trivedi, J. Wang, S. Kpotufe, and G. Shakhnarovich. A consistent estimator of the expected gradient outerproduct. In *UAI*, pages 819–828, 2014.

[25] H.-J. Ye, S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, and D.-C. Zhan. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024.

[26] L. Zhu, D. Davis, D. Drusvyatskiy, and M. Fazel. Iteratively reweighted kernel machines efficiently learn sparse functions, 2025.

A Features learned by xRFM on tabular data

We use two approaches for identifying features selected by the AGOP. The first is to identify the elements along the diagonal of the AGOP with highest magnitude. By definition, these entries indicate how much a leaf RFM's predictions vary when perturbing a given coordinate. As such, they are a natural measure of feature importance. The second approach is to examine the loadings onto the top eigenvectors of the AGOP. This approach allows us to identify joint effects of feature perturbations on the prediction. Namely, if coordinate i of the top eigenvector is positive and coordinate j is negative, then increasing one of these features and decreasing the other changes the prediction.

We use both approaches above to understand what features are learned by xRFM on tabular data from scikit-learn and meta-test (Fig. A.1). As examples, we study the AGOPs for four such datasets: (1) California housing - a regression task for predicting the average price of a house, (2) Covertype - a multiclass classification task for identifying the dominant tree species in a given location, (3) NYC Taxi Tipping - a regression task for predicting the dollar tip amount in a taxi ride, and (4) Breast cancer - a classification task for identifying malignancy from features of biopsy images.

As the California housing and breast cancer datasets contain fewer than 50k samples (the parameter we used for leaf size), we visualize a single AGOP. For covertype, we visualize one of the AGOPs from a leaf RFM (for this dataset, the leaf RFM AGOPs generally indicate the same pattern), and for taxi tipping we visualize several leaf RFM AGOPs. Upon visualization, it is apparent that AGOPs indicate low rank structure: either they highlight a subset of coordinates relevant for prediction (Fig. A.1A, B) or they have a decay in the eigenvalue spectrum (Fig. A.1C).

In Fig. A.1A and B, we list feature names for the features with highest diagonal AGOP entries (darker shades of red indicate higher values). In all cases, we find that AGOP identifies sensible features for prediction. For the California housing dataset, xRFM identifies longitude, or east-west location, as the most important feature for predicting the average price of a house. Given that beach fronts (and major cities) in California are typically located to the west, the importance of house longitude is consistent with the hypothesis that homes closer to the beach are more expensive on average. For the Covertype prediction dataset, the example AGOP from a leaf RFM shows that elevation, distance to roadways, and distance to firepoints are the most important features. This finding is consistent with the hypothesis that elevations with different climates and the existence of fires / roadways can significantly affect viability of different tree species. For the taxi tipping dataset, we observe that leaf RFMs identify varying local features. For example, one leaf RFM (denoted Taxi Tipping 1 in Fig. A.1B) selected pickup location as an important feature, while this feature was less important for a different leaf RFM (Taxi Tipping 3). Furthermore, fare code and the MTA tax have varying feature relationships at each leaf: in leaf RFM 1 and 2, the fare code and MTA tax has neutral or synergistic

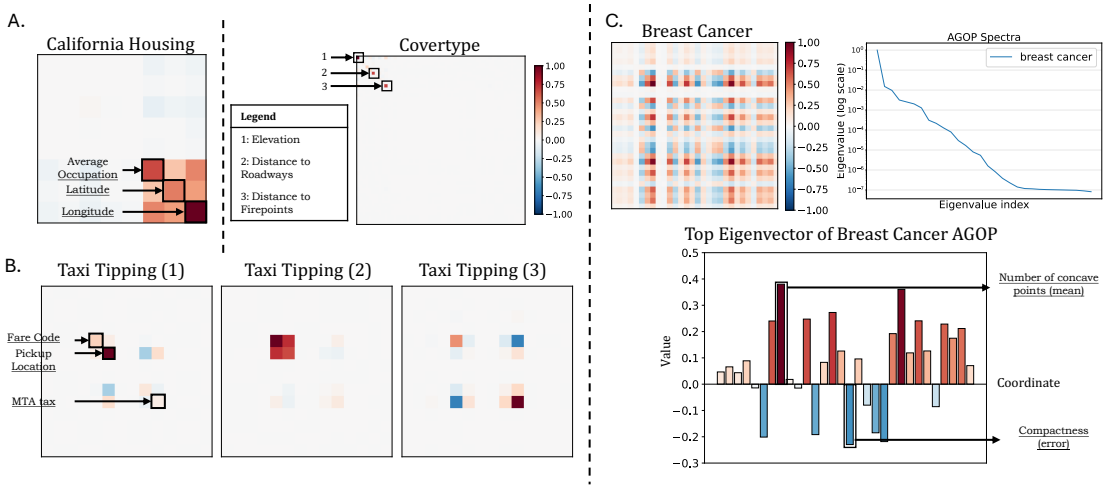

Figure A.1: Interpreting xRFM through the AGOP of its constituent Leaf RFM models. (A) Examining the most important features for xRFM trained on California Housing (price prediction) and Covertype (dominant tree species prediction) datasets, based on the magnitude of diagonal entries. (B) Examining the features identified across three different Leaf RFM models for the NYC Taxi Tipping dataset. (C) Examining features learned for Breast Cancer detection from processed FNA imaging. The spectrum of this AGOP is plotted and the top eigenvector is shown in a bar plot. The most positive and negative entries of this eigenvector are boxed.

effects on the prediction value, while for leaf RFM 3, increasing fare code has the opposite effect on prediction as increasing MTA tax (shown as a blue square in Fig. A.1B).

For the breast cancer dataset, the most important feature found by xRFM is the average number of concave points in the biopsy image, which has been shown to be a significant indicator of malignancy [19]. When examining the top eigenvector of the AGOP, we find that standard error in the compactness measurements of cell nuclei is the feature with highest importance that has an opposite effect on the malignancy label as concavity (Fig. A.1C). This finding suggests that benign cells have less uniform compactness than malignant cells.

B Methods

B.1 Code availability

Code for xRFM (following a scikit-learn-style API) is available at: `https://github.com/dmbeaglehole/xRFM`.

B.2 Additional leaf-RFM modifications

Typical choices of the T and c parameters for kernel-RFM in practice are $T \leq 10$ and $c \in \{\frac{1}{4}, \frac{1}{2}\}$ [20, 3, 4, 18]. (In practice, we return the f_t with best validation performance rather than returning f_T.) When training labels, y, are multi-dimensional, we use the Jacobian of $\hat{f}$ instead of the gradient in Step (2) (i.e., averaging the AGOP over output dimensions). Kernel-RFM is particularly effective at identifying low dimensional subspaces (or subsets of variables) relevant for prediction [21, 26], making it a useful interpretability tool.

Below, we provide further detail on the RFM modifications we made to implement leaf RFM.

(1) We tune over a more general class of kernels $K_{p,q}(x, x') = \exp(-\|x - x'\|_p^q / L^q)$ that are positive definite for $0 < q \leq p \leq 2$ [22, Theorems 1, 5]. We typically search over $p \in \mathcal{U}(0.7, 1.4)$ and $p \in \{q, 2\}$ (Table B.1).

(2) We tune over using the full AGOP and just the diagonal of the AGOP. The latter is known to be a theoretically grounded approach for coordinate selection [26], and introduces an axis-aligned bias that has been observed to match the structure of tabular data [13].

We also make the following changes to improve leaf RFM runtime on tabular data and to allow xRFM to adapt to the variance of input variables at each leaf:

(3) We speed up computation for kernels with $q = 1$ on categorical variables taking c values by precomputing possible kernel entries as follows. We restrict the AGOP matrix to be block diagonal with a block for the $c \times c$ entries corresponding to the categorical variable. Letting $M_c \in \mathbb{R}^{c \times c}$ denote this block and $e_i \in \{0, 1\}^c$ denote the one-hot embedding of the categorical variable when it takes on value i, we precompute $M_c^{1/2}(e_i - e_j)$ for all $i, j \in [c]$.

(4) We tune over whether or not to use *adaptive bandwidth*, which involves separately scaling L at each leaf RFM by $(\text{Median}(\|x - x'\|_p)))^{-1}$ for $x \neq x'$ at every leaf independently. Adaptive scaling allows xRFM to adapt to the variance of covariates at different leaves.

B.3 Metrics

Performance evaluation measures. When measuring performance on these datasets, we use Root Mean Square Error (RMSE) for regression tasks (as is used in TALENT), and we use classification error ($1 -$ classification accuracy) for classification tasks. For TabArena-Lite, binary classification performance for individual datasets are measured by AUROC, while multi-class datasets are evaluted with log-loss. To measure performance on aggregate over TALENT, we consider the following aggregation metrics on individual dataset performance measures: Shifted Geometric Mean, Arithmetic Mean, and Normalized Arithmetic Mean. TabArena-Lite uses Elo [8], a rating system that calculates the relative skill levels of each method based on the probability of winning when pairs of methods are compared on individual datasets (see [9] for how Elo is estimated in this benchmark).

Following [15], we report the shifted geometric mean of the error, which is the geometric mean of the error after shifting by a small value to prevent over-sensitivity to datasets with small errors. This metric is defined as follows.

Definition 1. *For a given set of errors on a benchmark $\varepsilon_1, \ldots, \varepsilon_N$, the shifted geometric mean (SGM_ε) with parameter ε takes value:*

$$\text{SGM}_\varepsilon = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\log(\varepsilon + \varepsilon_i)\right) .$$

In this work, as in [15], we use $\varepsilon = 0.01$.

For regression tasks, we use normalized Root-mean-square-error (nRMSE), which is defined as follows.

Definition 2. *The normalized Root-mean-square-error is defined as,*

$$\textit{nRMSE} \;=\; \sigma_y^{-1}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\big(y_i - \hat{y}_i\big)^2}, \qquad \textit{where } \sigma_y = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

When we refer to other normalized metrics, such as those used in Table C.1, Table C.2, Table C.3, Table C.4, Table C.5, we are min-max normalizing errors across methods for each dataset (see below).

Definition 3. *The normalized error $\tilde{E}_j$ for method j on a given dataset, where the un-normalized error is E_j, has the form:*

$$\tilde{E}_j \;=\; \frac{E_j - E_{\min}}{E_{\max} - E_{\min}}, \qquad E_{\min} = \min_k E_k, \quad E_{\max} = \max_k E_k \; .$$

B.4 Hyperparameters

Data pre-processing. For the TALENT benchmark, each method's hyperparameters are tuned after fixing a single data normalization and categorical encoding scheme. For xRFM, kernel ridge regression, and standard kernel RFM, we one-hot encode categorical features and z-score input coordinates separately. For meta-test, methods also tune over choice of ordinal or one-hot encoding of categorical variables. On meta-test, we also normalize data by z-scoring input coordinates separately prior to tuning xRFM parameters.

Hyperparameter	TALENT	TabArena-Lite	Meta-test
Bandwidth	$\log \mathcal{U}(1, 200)$	$\log \mathcal{U}(0.5, 200)$	$\log \mathcal{U}(0.4, 80)$
Bandwidth Mode	{constant}	{constant}	{constant, adaptive}
Categorical Transformations	{one_hot}	{one_hot}	{ordinal_encoding, one_hot}
Diagonal	{False, True}	{False, True}	{False, True}
Early Stop Multiplier	1.1	1.1	1.1
Exponent q	$\mathcal{U}(0.7, 1.4)$	$\mathcal{U}(0.7, 1.4)$	$\mathcal{U}(0.7, 1.3)$
Kernel Type	{80%: $K_{p,q}$, 20%$K_{2,q}$}	{80%: $K_{p,q}$, 20%$K_{2,q}$}	{80%: $K_{p,q}$, 20%$K_{2,q}$}
p (when kernel type is $K_{p,q}$)	$\mathcal{U}(q, q + 0.8(2 - q))$	$\mathcal{U}(q, q + 0.8(2 - q))$	$\mathcal{U}(q, q + 0.8(2 - q))$
Normalization	{standard}	{standard}	{standard}
Regularization	$\log \mathcal{U}(10^{-6}, 1)$	$\log \mathcal{U}(10^{-6}, 10)$	$\log \mathcal{U}(10^{-5}, 50)$
Refill size (N_{val})	1500	1500	1500

Table B.1: Search spaces for xRFM on the TALENT (Figure 2), TabArena-Lite (Figure 2), and Meta-test benchmarks (Figure 3).

Details of various methods. For scaling kernel ridge regression and the standard kernel RFM to large TALENT datasets, we used Eigenpro-2 (EP2) [17] that was initialized with the coefficients obtained from directly solving RFM on a random sample of 70,000 points. To tune the optimization hyperparameters for EP2, we tuned the model parameters for 100 trials on a random 70,000 sample subset, then for the final run, we used the tuned hyperparameters (and for kernel RFM the AGOP from the best iteration). For TabPFN-v2, we used the code provided from the benchmark github (`https://github.com/LAMDA-Tabular/TALENT`), which subsampled to 10,000 samples to avoid out-of-memory issues. The provided code did not apply TabPFN-v2 to datasets with more than 10 classes. For efficient L_p^q kernel computations on GPU we used the KerMac library (`https://github.com/Kernel-Machines/kermac`).

B.5 Algorithmic details

Note on AGOP computation For computing gradients of the predictor on training data (for AGOP computation), we omit the contribution to the gradient from the kernel evaluation between each training point and itself. This is because the kernel function is often not differentiable (e.g. Laplace kernels) when evaluated for two identical points.

Algorithm B.1 Leaf RFM

Require:
- $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d, y \in \mathbb{R}^{n \times c}$: Train data
- $X_{\text{val}} \in \mathbb{R}^{m \times d}, y_{\text{val}} \in \mathbb{R}^{m \times c}$: Validation data
- $K(\cdot, \cdot; L, p)$: Kernel parametrized by bandwidth $L \in \mathbb{R}^+$ and exponent $p \in (0, 2]$
- $\tau \in \mathbb{Z}^+$: Number of iterations
- $\lambda \in \mathbb{R}^+$: Ridge parameter
- $\varepsilon \in \mathbb{R}^+$: Stability parameter
- use_diag: Boolean indicating whether to use diagonal of AGOP only
- adapt_bandwidth: Boolean indicating whether to adapt bandwidth

$M_0 \leftarrow I_{d \times d}$
$X = [x^{(1)}, \ldots, x^{(n)}]^\top \in \mathbb{R}^{n \times d}$
if adapt_bandwidth **then**
  $L \leftarrow \text{AdaptBandwidth}(L)$ ▷ Adapt bandwidth if enabled
end if
for $t = 0, \ldots, \tau - 1$ **do**
  if use_diag **then**
    $X_M \leftarrow X \odot \text{diag}(M_t)^{1/2}$
    Solve α_t such that $(K(X_M, X_M) + \lambda I)\alpha_t = y$
    $f^{(t)}(x) = K(x \odot \text{diag}(M_t)^{1/2}, X_M)\alpha_t$ ▷ $\odot$ denotes element-wise multiplication
  else
    $X_M \leftarrow X M_t^{1/2}$
    Solve α_t such that $(K(X_M, X_M) + \lambda I)\alpha_t = y$
    $f^{(t)}(x) = K(M_t^{1/2} x, X_M)\alpha_t$
  end if
  Compute $E_t \leftarrow \text{Error}(f^{(t)}, X_{\text{val}}, y_{\text{val}})$ ▷ Validate model
  $M_{t+1} \leftarrow \frac{1}{n} \sum_{i=1}^n \nabla_x f^{(t)}(x^{(i)}) \nabla_x f^{(t)}(x^{(i)})^\top \in \mathbb{R}^{d \times d}$ ▷ Feature matrix (AGOP) computation
  $M_{t+1} \leftarrow M_{t+1} / (\varepsilon + \max_{i,j} M_{t+1}[i, j])$ ▷ Normalize feature matrix
end for
$t^* \leftarrow \arg\min_t E_t$
return α_{t^*}, M_{t^*} ▷ KRR coefficients: α_{t^*}, feature matrix: M_{t^*} from best iteration on val. set

Algorithm B.2 TreePartition

Require:
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$: Full dataset with $x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}^c$
- $K(\cdot, \cdot; L, p)$: Kernel parametrized by bandwidth $L \in \mathbb{R}^+$ and exponent $p \in (0, 2]$
- $N \in \mathbb{Z}^+$: Number of sample points for Leaf RFM
- $L \in \mathbb{Z}^+$: Maximum leaf size
- $\lambda \in \mathbb{R}^+$: Ridge parameter

function TREEPARTITION($\mathcal{D}$)
  if $|\mathcal{D}| \leq L$ **then**
    return Leaf node with dataset $\mathcal{D}$
  end if
  Sample N points $\mathcal{S} = \{(x^{(a_1)}, y^{(a_1)}), \ldots, (x^{(a_N)}, y^{(a_N)})\}$ from $\mathcal{D}$
  $X_s = [x^{(a_1)}, \ldots, x^{(a_N)}]^\top \in \mathbb{R}^{N \times d}$
  $y_s = [y^{(a_1)}, \ldots, y^{(a_N)}]^\top \in \mathbb{R}^{N \times c}$
  Solve α such that $(K(X_s, X_s) + \lambda I)\alpha = y$ ▷ Fit Leaf RFM on sampled data
  Define predictor $f(x) = K(x, X_s)\alpha$
  Compute AGOP: $M \leftarrow \frac{1}{N} \sum_{i=1}^{N} \nabla_x f(x^{(a_i)}) \nabla_x f(x^{(a_i)})^\top \in \mathbb{R}^{d \times d}$
  Extract top eigenvector v_1 of M ▷ Principal direction
  Project all data points: $p^{(i)} \leftarrow v_1^\top x^{(i)}$ for $i = 1, \ldots, |\mathcal{D}|$
  Compute median projection: $m \leftarrow \text{Median}(\{p^{(1)}, \ldots, p^{(|\mathcal{D}|)}\})$
  Split dataset:
    $\mathcal{D}_{\text{left}} \leftarrow \{(x^{(i)}, y^{(i)}) \in \mathcal{D} : v_1^\top x^{(i)} \leq m\}$
    $\mathcal{D}_{\text{right}} \leftarrow \{(x^{(i)}, y^{(i)}) \in \mathcal{D} : v_1^\top x^{(i)} > m\}$
  left_child $\leftarrow$ TreePartition($\mathcal{D}_{\text{left}}$) ▷ Recursive call
  right_child $\leftarrow$ TreePartition($\mathcal{D}_{\text{right}}$) ▷ Recursive call
return Internal node with splitting vector v_1, threshold m, and children
end function

Algorithm B.3 Route (*find the leaf that contains a point*)

Require:
- $\mathcal{T}$: a (possibly trained) tree whose internal nodes store
  - splitting vector $v_1 \in \mathbb{R}^d$
  - threshold $m \in \mathbb{R}$
- $x \in \mathbb{R}^d$: query point

function ROUTE($x, \mathcal{T}$)
  $r \leftarrow \mathcal{T}$.root ▷ Initialize current node r from the tree
  while r is an internal node **do**
    if $v_1^\top x \leq r$.threshold **then** ▷ Check if projection is less than (median) threshold
      $r \leftarrow r$.left_child
    else
      $r \leftarrow r$.right_child
    end if
  end while
  return r ▷ r is now the leaf node that contains x
end function

Algorithm B.4 xRFM (*training*)

Require:
- $\mathcal{D}_{\text{train}},\ \mathcal{D}_{\text{val}}$: training and validation sets
- $K(\cdot,\cdot; L, p)$: kernel (bandwidth L, exponent p)
- TreeHyp $= \{N,\ L_{\max},\ \lambda\text{split}\}$: hyper-parameters for TREEPARTITION
- LeafHyp $= \{\tau,\ \lambda_{\text{leaf}},\ \varepsilon, \textit{use\_diag}\}$: hyper-parameters for LEAFRFM

function xRFM-FIT($\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$)
  $\mathcal{T} \leftarrow$ TREEPARTITION($\mathcal{D}_{\text{train}}$, K, TreeHyp) ▷ Alg. B.2
  for all leaf node $\ell \in$ LEAVES($\mathcal{T}$) **do**
    $\mathcal{D}_\ell^{\text{train}} \leftarrow$ data stored in ℓ
    $\mathcal{D}_\ell^{\text{val}} \leftarrow \{(x,y) \in \mathcal{D}_{\text{val}} : \text{ROUTE}(x, \mathcal{T}) = \ell\}$
    $(\alpha_\ell, M_\ell) \leftarrow$ LEAFRFM($\mathcal{D}_\ell^{\text{train}}, \mathcal{D}_\ell^{\text{val}}$, K, LeafHyp) ▷ Alg. B.1
    Define predictor $f_\ell(x)$ according to (α_ℓ, M_ℓ) and store it in ℓ
  end for
  return $\mathcal{T}$ ▷ Tree whose leaves now carry trained predictors
end function

Algorithm B.5 xRFM (*inference*)

Require:
- $\mathcal{T}$: trained tree returned by xRFM-FIT (Alg. B.4)
- $x \in \mathbb{R}^d$: test point

function xRFM-PREDICT($\mathcal{T}, x$)
  $\ell \leftarrow$ ROUTE(x, $\mathcal{T}$) ▷ Alg. B.3
  $\hat{y} \leftarrow f_\ell(x)$ ▷ Leaf predictor stored in ℓ
  return $\hat{y}$
end function

C All results

Method	Rank	Score	Norm. Score	Top-1 (%)	Top-3 (%)	Top-5 (%)	Top-8 (%)	SGM_ε
xRFM	**4.70**	**0.311**	**0.036**	**20.0**	**56.0**	**69.0**	**84.0**	**0.311**
PFN-v2	6.55	0.323	0.067	**20.0**	45.0	57.0	72.0	0.323
CatBoost	7.79	0.336	0.053	7.00	24.0	41.0	68.0	0.336
RealMLP	8.20	0.329	0.076	8.00	21.0	37.0	60.0	0.329
ModernNCA	9.39	0.365	0.097	14.0	30.0	38.0	57.0	0.365
LightGBM	9.91	0.353	0.062	6.00	16.0	34.0	52.0	0.353
XGBoost	10.6	0.355	0.071	2.00	12.0	32.0	47.0	0.355
TabR	10.7	0.365	0.133	5.00	20.0	34.0	46.0	0.365
FTT	12.0	0.388	0.149	3.00	11.0	17.0	33.0	0.388
Laplace KRR	12.3	0.373	0.102	3.00	11.0	21.0	35.0	0.373
MLP-PLR	12.7	0.381	0.142	0.0	8.00	15.0	34.0	0.381
Excelformer	12.7	0.399	0.157	0.0	2.00	9.00	28.0	0.399
PTARL	13.1	0.390	0.114	2.00	6.00	9.00	20.0	0.390
RandomForest	13.3	0.389	0.088	5.00	12.0	22.0	35.0	0.389
AutoInt	14.3	0.399	0.158	1.00	2.00	6.00	12.0	0.399
Node	14.4	0.451	0.195	0.0	6.00	18.0	31.0	0.451
MLP	15.2	0.418	0.167	1.00	3.00	4.00	11.0	0.418
DCN2	15.3	0.473	0.227	0.0	2.00	10.0	20.0	0.473
ResNet	15.6	0.426	0.175	0.0	4.00	8.00	15.0	0.426
Tangos	16.3	0.432	0.173	0.0	1.00	4.00	9.00	0.432
SNN	17.2	0.419	0.174	0.0	2.00	4.00	8.00	0.419
kNN	19.1	0.459	0.190	2.00	4.00	4.00	11.0	0.459
TabNet	21.5	0.475	0.227	0.0	0.0	0.0	0.0	0.475
GrowNet	23.0	0.619	0.323	0.0	0.0	0.0	0.0	0.619
SVM	23.2	0.612	0.346	0.0	0.0	1.00	1.00	0.612
TabTransformer	26.4	1.07	0.750	1.00	1.00	2.00	3.00	1.07
SwitchTab	26.9	1.29	0.745	0.0	0.0	0.0	1.00	1.29
Danets	27.2	1.12	0.830	0.0	1.00	1.00	3.00	1.12

Table C.1: Full TALENT Regression results across 100 datasets. Rank is the average rank among the ordered methods over all datasets. Score is the metric we use to compare methods in Figure 2, in this case SGM_ϵ. Normalized score is the arithmetic mean of the normalized nRMSE. Top-X (%) is the percentage of datasets for which that method is in the top X ranks. The final column is the shifted geometric mean error (SGM_ε).

Method	Rank	Score	Norm. Score	Top-1 (%)	Top-3 (%)	Top-5 (%)	Top-8 (%)	SGM_ε
PFN-v2	**7.15**	0.823	**0.935**	**25.0**	**51.5**	**63.2**	**70.6**	0.108
xRFM	7.60	0.825	0.933	11.8	29.4	48.5	66.2	0.107
RealMLP	7.64	0.823	0.918	10.3	20.6	48.5	66.2	0.107
TabR	7.98	**0.828**	0.932	8.82	27.9	41.2	60.3	**0.106**
ModernNCA	8.88	0.825	0.912	10.3	30.9	44.1	61.8	0.106
CatBoost	10.3	0.819	0.905	1.47	19.1	36.8	51.5	0.117
LightGBM	11.7	0.813	0.883	5.88	22.1	33.8	44.1	0.125
XGBoost	12.1	0.815	0.886	4.41	19.1	26.5	41.2	0.124
ResNet	12.3	0.807	0.879	0.0	4.41	16.2	36.8	0.127
FTT	12.8	0.810	0.868	0.0	5.88	10.3	29.4	0.125
MLP-PLR	13.2	0.815	0.881	0.0	7.35	11.8	25.0	0.119
Laplace KRR	13.5	0.804	0.864	5.88	16.2	20.6	33.8	0.133
DCN2	14.0	0.806	0.865	1.47	4.41	10.3	26.5	0.127
MLP	14.2	0.805	0.864	0.0	4.41	8.82	19.1	0.131
SNN	14.9	0.805	0.858	0.0	0.0	4.41	8.82	0.129
AutoInt	15.7	0.804	0.858	0.0	0.0	1.47	7.35	0.130
RandomForest	15.7	0.797	0.831	5.88	8.82	13.2	27.9	0.137
Excelformer	16.1	0.805	0.853	0.0	0.0	4.41	17.6	0.127
Tangos	16.5	0.797	0.844	0.0	1.47	4.41	13.2	0.133
TabCaps	16.5	0.797	0.837	0.0	1.47	5.88	13.2	0.134
Danets	17.7	0.795	0.829	0.0	1.47	2.94	5.88	0.137
PTARL	19.1	0.796	0.808	0.0	1.47	1.47	4.41	0.139
kNN	20.1	0.786	0.772	2.94	7.35	10.3	14.7	0.152
LogReg	20.2	0.767	0.755	1.47	4.41	7.35	13.2	0.162
TabTransformer	21.5	0.771	0.739	0.0	1.47	1.47	2.94	0.155
Node	22.0	0.767	0.742	0.0	4.41	7.35	10.3	0.173
SVM	23.4	0.750	0.697	2.94	5.88	5.88	8.82	0.182
GrowNet	24.4	0.691	0.596	0.0	0.0	2.94	5.88	0.216
SwitchTab	25.0	0.748	0.674	0.0	1.47	1.47	2.94	0.187
TabNet	26.0	0.753	0.615	0.0	0.0	0.0	0.0	0.183
NaiveBayes	29.4	0.609	0.327	0.0	0.0	0.0	0.0	0.321
NCM	30.5	0.627	0.278	0.0	0.0	1.47	1.47	0.326

Table C.2: Full TALENT Multiclass classification (≤ 10 classes) results. Rank is the average rank among the ordered methods over all datasets. (Normalized) Score is the metric we use to compare methods in Figure 2. In this case score is the mean classification accuracy (1 minus the error in Figure 2). Top-X (%) is the percentage of datasets for which that method is in the top X ranks. The final column is the shifted geometric mean error (SGM_ε).

Method	Rank	Score	Norm. Score	Top-1 (%)	Top-3 (%)	Top-5 (%)	Top-8 (%)	SGM_ε
RealMLP	**3.75**	**0.730**	**0.964**	8.33	**50.0**	**91.7**	**91.7**	**0.164**
xRFM	5.46	0.718	0.949	25.0	50.0	66.7	83.3	0.183
TabR	6.00	0.720	0.948	16.7	50.0	50.0	66.7	0.172
ResNet	6.33	0.720	0.949	0.0	16.7	50.0	83.3	0.178
ModernNCA	6.83	0.713	0.913	**33.3**	41.7	50.0	83.3	0.168
FTT	8.79	0.709	0.919	0.0	8.33	25.0	50.0	0.187
MLP-PLR	8.79	0.712	0.925	0.0	0.0	25.0	50.0	0.185
MLP	9.58	0.710	0.922	0.0	0.0	8.33	41.7	0.190
Laplace KRR	10.8	0.679	0.889	0.0	16.7	41.7	50.0	0.218
DCN2	11.2	0.707	0.915	0.0	8.33	8.33	8.33	0.193
AutoInt	12.2	0.701	0.899	0.0	0.0	0.0	16.7	0.194
SNN	12.2	0.704	0.906	0.0	0.0	0.0	33.3	0.197
CatBoost	12.8	0.679	0.842	8.33	25.0	33.3	33.3	0.218
Danets	15.9	0.687	0.882	0.0	0.0	0.0	0.0	0.215
Tangos	16.1	0.660	0.823	0.0	8.33	8.33	16.7	0.225
XGBoost	16.2	0.678	0.874	8.33	8.33	8.33	25.0	0.236
Excelformer	16.6	0.675	0.851	0.0	0.0	0.0	16.7	0.217
TabCaps	16.8	0.670	0.853	0.0	0.0	0.0	0.0	0.231
LightGBM	16.9	0.668	0.850	0.0	0.0	0.0	16.7	0.261
PTARL	18.7	0.656	0.823	0.0	0.0	0.0	0.0	0.233
kNN	20.2	0.637	0.800	0.0	0.0	0.0	8.33	0.273
RandomForest	20.7	0.620	0.782	0.0	8.33	8.33	8.33	0.311
TabTransformer	20.8	0.582	0.702	0.0	0.0	8.33	8.33	0.307
LogReg	22.0	0.538	0.622	0.0	0.0	0.0	0.0	0.338
SVM	25.1	0.486	0.523	0.0	8.33	8.33	8.33	0.385
SwitchTab	25.2	0.525	0.603	0.0	0.0	0.0	0.0	0.372
Node	26.2	0.536	0.611	0.0	0.0	0.0	0.0	0.402
GrowNet	26.2	0.402	0.477	0.0	0.0	0.0	0.0	0.565
TabNet	26.3	0.486	0.516	0.0	0.0	0.0	0.0	0.368
NaiveBayes	29.5	0.410	0.380	0.0	0.0	0.0	0.0	0.545
NCM	30.2	0.399	0.339	0.0	0.0	0.0	0.0	0.552

Table C.3: Full TALENT Multiclass classification (> 10 classes) results. Rank is the average rank among the ordered methods over all datasets. (Normalized) Score is the metric we use to compare methods in Figure 2. In this case score is the mean classification accuracy (1 minus the error in Figure 2). Top-X (%) is the percentage of datasets for which that method is in the top X ranks. The final column is the shifted geometric mean error (SGM_ε).

Method	Rank	Score	Norm. Score	Top-1 (%)	Top-3 (%)	Top-5 (%)	Top-8 (%)	SGM_ε
xRFM	**5.96**	**0.845**	**0.981**	**29.6**	**55.6**	**59.3**	74.1	0.119
RealMLP	7.44	0.839	0.951	7.41	14.8	29.6	**77.8**	0.125
TabR	7.83	0.844	0.970	18.5	33.3	55.6	70.4	**0.116**
ModernNCA	9.69	0.843	0.963	3.70	29.6	40.7	59.3	0.120
CatBoost	10.2	0.827	0.891	7.41	25.9	33.3	55.6	0.128
LightGBM	10.5	0.826	0.884	7.41	18.5	40.7	55.6	0.128
PFN-v2	10.9	0.834	0.923	7.41	22.2	25.9	48.1	0.129
MLP-PLR	11.2	0.827	0.890	0.0	0.0	18.5	40.7	0.131
XGBoost	12.1	0.824	0.875	3.70	18.5	33.3	51.9	0.129
FTT	12.4	0.829	0.900	0.0	11.1	18.5	29.6	0.135
MLP	12.7	0.833	0.915	0.0	7.41	14.8	29.6	0.130
ResNet	13.3	0.834	0.920	0.0	3.70	7.41	29.6	0.130
DCN2	13.4	0.832	0.919	0.0	3.70	3.70	22.2	0.130
PTARL	13.4	0.831	0.911	0.0	3.70	7.41	14.8	0.131
AutoInt	13.7	0.831	0.911	0.0	0.0	3.70	11.1	0.130
Excelformer	14.7	0.821	0.858	3.70	7.41	7.41	11.1	0.143
SNN	14.8	0.831	0.912	0.0	3.70	7.41	18.5	0.131
Laplace KRR	17.0	0.826	0.879	3.70	11.1	14.8	14.8	0.138
Danets	17.4	0.827	0.887	0.0	0.0	0.0	0.0	0.133
TabCaps	18.4	0.822	0.862	0.0	3.70	7.41	7.41	0.136
TabTransformer	19.2	0.816	0.796	0.0	3.70	7.41	18.5	0.160
RandomForest	19.6	0.806	0.772	0.0	3.70	7.41	18.5	0.144
Node	20.1	0.793	0.724	3.70	3.70	3.70	7.41	0.151
LogReg	20.5	0.807	0.778	0.0	3.70	18.5	25.9	0.159
Tangos	20.7	0.811	0.798	0.0	0.0	3.70	3.70	0.139
SVM	21.8	0.802	0.754	0.0	0.0	3.70	18.5	0.162
GrowNet	22.7	0.804	0.771	0.0	0.0	0.0	3.70	0.160
TabNet	23.2	0.814	0.818	0.0	0.0	0.0	0.0	0.142
kNN	24.3	0.787	0.673	0.0	0.0	0.0	7.41	0.166
SwitchTab	27.3	0.786	0.670	0.0	0.0	0.0	0.0	0.163
NaiveBayes	30.2	0.678	0.154	0.0	0.0	0.0	0.0	0.283
NCM	31.3	0.679	0.174	0.0	0.0	0.0	0.0	0.281

Table C.4: Full TALENT Binary classification ($> 10,000$ samples) results. Rank is the average rank among the ordered methods over all datasets. (Normalized) Score is the metric we use to compare methods in Figure 2. In this case score is the mean classification accuracy (1 minus the error in Figure 2). Top-X (%) is the percentage of datasets for which that method is in the top X ranks. The final column is the shifted geometric mean error (SGM_ε).

Method	Rank	Score	Norm. Score	Top-1 (%)	Top-3 (%)	Top-5 (%)	Top-8 (%)	SGM_ε
PFN-v2	**5.02**	**0.864**	**0.947**	**26.9**	**59.1**	**72.0**	**80.6**	**0.104**
xRFM	8.75	0.856	0.893	7.53	29.0	49.5	63.4	0.111
ModernNCA	10.3	0.856	0.886	11.8	22.6	31.2	50.5	0.111
CatBoost	10.3	0.845	0.867	4.30	23.7	34.4	54.8	0.119
LightGBM	10.4	0.845	0.869	7.53	18.3	35.5	54.8	0.119
TabR	10.5	0.851	0.873	4.30	17.2	30.1	43.0	0.116
XGBoost	11.4	0.844	0.851	5.38	18.3	35.5	49.5	0.120
RealMLP	11.7	0.850	0.864	1.08	7.53	18.3	35.5	0.116
FTT	13.3	0.836	0.813	1.08	6.45	17.2	32.3	0.124
MLP-PLR	14.2	0.838	0.817	0.0	3.23	11.8	29.0	0.124
RandomForest	14.3	0.836	0.824	3.23	10.8	19.4	33.3	0.129
AutoInt	15.2	0.834	0.803	0.0	3.23	5.38	16.1	0.129
Tangos	15.4	0.834	0.800	0.0	3.23	7.53	16.1	0.131
DCN2	15.5	0.839	0.816	1.08	2.15	9.68	21.5	0.127
Laplace KRR	15.7	0.836	0.801	1.08	16.1	20.4	26.9	0.133
MLP	16.0	0.836	0.797	1.08	3.23	8.60	17.2	0.132
ResNet	16.1	0.839	0.811	1.08	3.23	8.60	19.4	0.129
TabCaps	16.2	0.835	0.804	1.08	3.23	5.38	14.0	0.130
Excelformer	16.6	0.830	0.778	0.0	1.08	7.53	16.1	0.131
SNN	16.6	0.836	0.800	1.08	3.23	4.30	7.53	0.129
Node	17.0	0.824	0.757	0.0	5.38	15.1	25.8	0.137
PTARL	17.0	0.830	0.780	0.0	1.08	3.23	8.60	0.134
Danets	18.0	0.829	0.759	0.0	3.23	6.45	14.0	0.137
TabTransformer	19.7	0.820	0.715	1.08	1.08	6.45	12.9	0.148
LogReg	20.1	0.819	0.723	3.23	9.68	12.9	17.2	0.146
kNN	20.8	0.811	0.677	5.38	10.8	14.0	18.3	0.158
SVM	21.6	0.816	0.698	2.15	3.23	6.45	10.8	0.150
GrowNet	22.2	0.817	0.714	0.0	0.0	0.0	2.15	0.151
SwitchTab	24.0	0.809	0.675	0.0	0.0	0.0	0.0	0.156
TabNet	25.8	0.802	0.633	0.0	0.0	0.0	1.08	0.157
NaiveBayes	28.6	0.695	0.281	1.08	2.15	3.23	4.30	0.253
NCM	29.9	0.723	0.230	0.0	0.0	0.0	2.15	0.257

Table C.5: Full TALENT Binary classification ($\leq 10,000$ samples) results. Rank is the average rank among the ordered methods over all datasets. (Normalized) Score is the metric we use to compare methods in Figure 2. In this case score is the mean classification accuracy (1 minus the error in Figure 2). Top-X (%) is the percentage of datasets for which that method is in the top X ranks. The final column is the shifted geometric mean error (SGM_ε).

Dataset	xRFM	XGB	CatBoost	LGBM	RealMLP	MLP-PLR	MLP-RTDL	ResNet-RTDL
Airlines_DepDelay_10M	0.9818	0.9813	0.9796	0.9798	**0.9786**	0.9795	0.9824	0.9818
Allstate_Claims_Severity	**0.6489**	0.6547	0.6510	0.6530	0.6495	0.6537	0.6557	0.6537
black_friday	0.6881	0.6807	0.6792	**0.6787**	0.6859	0.6862	0.6929	0.6892
Buzzinsocialmedia_Twitter	**0.2080**	0.2134	0.3147	0.2789	0.2566	0.2553	0.2840	0.2906
nyc-taxi-green-dec-2016	**0.5834**	0.6649	0.6567	0.6489	0.6142	0.6523	0.6657	0.6365
wave_energy	**0.0020**	0.0918	0.0499	0.0821	0.0024	0.0073	0.0254	0.0434
Yolanda	**0.7816**	0.8012	0.8094	0.7970	0.7869	0.7897	0.7927	0.7856

Table C.6: Meta-test regression datasets with more than 70,000 samples. Error reported is nRMSE averaged over five train/test splits using `pytabkit`.

Dataset	xRFM	XGB	CatBoost	LGBM	RealMLP	MLP-PLR	MLP-RTDL	ResNet-RTDL
airlines	0.3341	0.3292	0.3315	**0.3287**	0.3344	0.3342	0.3342	0.3339
covertype	**0.0257**	0.0420	0.0612	0.0333	0.0280	0.0364	0.0404	0.0385
Higgs	0.2640	0.2576	0.2576	0.2549	0.2473	0.2528	0.2515	**0.2435**
jannis	**0.2701**	0.2799	0.2816	0.2779	0.2702	0.2764	0.2865	0.2795
MiniBooNE	0.0539	0.0529	0.0538	0.0525	**0.0484**	0.0504	0.0503	0.0488
numerai28.6	0.4778	0.4812	0.4790	0.4782	0.4800	0.4775	**0.4771**	0.4800
porto-seguro	**0.0380**	**0.0380**	0.0381	0.0381	**0.0380**	**0.0380**	**0.0380**	**0.0380**
dionis	0.0926	0.1219	0.1041	0.1076	**0.0887**	0.1257	0.1110	0.0907
Fashion-MNIST	0.0889	0.0928	0.0969	**0.0895**	0.0913	0.1064	0.1041	0.1011
kick	0.0972	0.0965	**0.0956**	0.0964	0.0976	0.0978	0.0979	0.0970

Table C.7: Meta-test classification datasets with greater than 70,000 samples. Error reported is classification error averaged over five train/test splits using `pytabkit`.

Model	Elo (↑)	Norm. score (↑)	Avg. rank (↓)	Harm. mean rank (↓)	#wins (↑)	Improva-bility (↓)	Train time per 1K [s]	Predict time per 1K [s]
AutoGluon 1.3 (4h)	1779	0.673	5.2	3.2	1	3.6%	1734.20	7.06
RealMLP (T+E)	1721	0.660	6.6	5.4	0	3.1%	6860.54	7.68
ModernNCA (T+E)	1626	0.622	9.2	2.6	3	5.4%	3811.43	7.58
LightGBM (T+E)	1602	0.478	10.1	7.5	0	6.0%	686.46	5.48
TabDPT (D)	1575	0.515	10.9	3.8	2	3.9%	16.97	8.70
xRFM (T+E)	1563	0.492	11.5	5.3	1	5.1%	365.57	0.72
CatBoost (T+E)	1558	0.435	11.8	8.9	0	5.7%	2895.38	1.32
TabM (T+E)	1529	0.439	12.8	8.5	0	4.3%	4228.53	1.19
CatBoost (T)	1515	0.406	13.3	6.8	0	5.9%	2895.38	0.07
xRFM (T)	1504	0.406	13.8	9.7	0	5.7%	365.57	0.09
LightGBM (T)	1498	0.354	13.9	10.2	0	6.8%	686.46	0.74
XGBoost (T+E)	1470	0.328	15.2	13.9	0	6.7%	848.99	2.38
XGBoost (T)	1470	0.325	15.2	13.3	0	6.7%	848.99	0.47
ModernNCA (D)	1457	0.323	15.6	11.0	0	8.1%	16.07	0.29
TabM (T)	1453	0.311	16.0	12.6	0	5.2%	4228.53	0.13
RealMLP (T)	1439	0.306	16.6	13.3	0	5.9%	6860.54	0.32
CatBoost (D)	1419	0.278	17.4	12.0	0	7.9%	8.35	0.09
ModernNCA (T)	1411	0.347	17.6	6.7	0	7.9%	3811.43	0.45
TabPFNv2 (T+E)	1377	0.410	19.1	2.7	4	4.9%	3805.62	10.41
TabM (D)	1361	0.232	20.1	16.5	0	6.9%	13.90	0.12
TabPFNv2 (T)	1314	0.281	22.1	6.9	0	6.4%	3805.62	0.26
TorchMLP (T+E)	1301	0.141	22.7	19.1	0	8.1%	4452.11	0.85
ExtraTrees (T+E)	1297	0.169	23.1	16.9	0	11.1%	161.73	0.78
RealMLP (D)	1274	0.085	24.0	21.6	0	8.2%	23.30	1.44
ExtraTrees (T)	1273	0.174	24.0	16.7	0	11.4%	161.73	0.12
TabPFNv2 (D)	1264	0.262	24.3	7.7	0	7.6%	2.78	0.32
TorchMLP (T)	1245	0.124	25.2	20.8	0	8.7%	4452.11	0.09
xRFM (D)	1236	0.149	25.8	8.6	1	11.7%	1.65	0.08
LightGBM (D)	1231	0.069	25.9	24.5	0	9.7%	2.03	0.30
XGBoost (D)	1195	0.104	27.4	24.5	0	10.4%	2.15	0.18
RandomForest (T+E)	1193	0.056	27.5	25.5	0	12.1%	526.17	0.77
RandomForest (T)	1142	0.046	29.9	27.5	0	12.8%	526.17	0.12
EBM (T+E)	1133	0.122	30.0	16.4	0	14.7%	2124.78	0.12
ExtraTrees (D)	1117	0.056	30.6	27.7	0	13.0%	0.42	0.06
FastaiMLP (T+E)	1086	0.005	31.8	30.6	0	13.1%	527.21	2.83
EBM (T)	1076	0.131	32.2	9.4	1	15.3%	2124.78	0.01
TorchMLP (D)	1075	0.015	32.4	30.3	0	12.9%	20.50	0.08
FastaiMLP (T)	1046	0.000	33.3	32.3	0	13.6%	527.21	0.31
EBM (D)	1009	0.071	34.7	30.3	0	16.0%	7.25	0.04
TabICL (D)	1006	0.000	34.8	34.2	0	14.2%	0.63	0.06
RandomForest (D)	1000	0.000	34.8	34.2	0	14.2%	0.63	0.06
FastaiMLP (D)	916	0.000	37.2	36.5	0	17.8%	3.08	0.29
KNN (T+E)	533	0.000	43.8	43.6	0	37.3%	2.25	0.15
Linear (T+E)	485	0.000	44.3	44.2	0	35.5%	46.50	0.14
KNN (T)	434	0.000	44.9	44.7	0	38.0%	2.25	0.03
Linear (T)	426	0.000	44.9	44.8	0	35.7%	46.50	0.04
Linear (D)	312	0.000	46.0	46.0	0	38.1%	1.16	0.08
KNN (D)	263	0.000	46.5	46.2	0	41.6%	0.04	0.02

Table C.8: Full regression results on TabArena-Lite.

Model	Elo (↑)	Norm. score (↑)	Avg. rank (↓)	Harm. mean rank (↓)	#wins (↑)	Improvability (↓)	Train time per 1K [s]	Predict time per 1K [s]
AutoGluon 1.3 (4h)	**1521**	**0.586**	**9.0**	**3.0**	**2**	**9.7%**	4917.81	4.05
CatBoost (T)	**1441**	0.439	**12.5**	9.6	0	**11.9%**	3307.58	0.14
CatBoost (T+E)	**1441**	**0.446**	**12.4**	9.4	0	**11.1%**	3307.58	1.18
TabPFNv2 (T+E)	1419	**0.518**	13.5	**4.2**	**1**	13.8%	2584.13	12.37
LightGBM (T+E)	1418	0.393	13.2	7.5	0	13.6%	1280.01	4.08
LightGBM (T)	1405	0.381	13.9	6.9	0	13.6%	1280.01	1.05
TabM (T+E)	1387	0.405	14.8	9.7	0	15.7%	5568.31	1.78
XGBoost (T+E)	1377	0.310	15.2	13.3	0	14.0%	2029.77	4.11
TabM (T)	1373	0.401	15.5	9.1	0	16.0%	5568.31	0.37
XGBoost (T)	1373	0.317	15.6	13.1	0	13.7%	2029.77	1.04
RealMLP (T+E)	1359	0.337	16.2	5.3	**1**	16.8%	6866.35	10.40
TabPFNv2 (D)	1358	0.361	16.4	4.7	**1**	12.5%	5.48	0.35
xRFM (T+E)	1357	0.369	16.2	7.0	0	14.8%	515.01	1.68
TabPFNv2 (T)	1346	0.366	17.0	6.5	0	14.9%	2584.13	0.41
ModernNCA (T+E)	1345	0.302	16.9	10.4	0	15.3%	6684.65	9.59
ModernNCA (T)	1332	0.248	17.8	12.4	0	15.6%	6684.65	0.75
RealMLP (T)	1304	0.253	19.1	15.5	0	18.4%	6866.35	0.92
CatBoost (D)	1296	0.201	19.2	16.9	0	16.1%	43.10	0.25
TabICL (D)	1293	0.325	19.9	**4.5**	**1**	20.5%	11.51	1.95
TabM (D)	1293	0.284	19.9	9.4	0	19.7%	17.09	0.15
ExtraTrees (T+E)	1286	0.289	20.2	11.0	0	17.7%	728.32	2.44
xRFM (T)	1283	0.273	20.4	5.2	**1**	16.4%	515.01	0.20
ExtraTrees (T)	1265	0.262	21.2	11.2	0	17.3%	728.32	0.36
RandomForest (T+E)	1260	0.321	20.9	6.1	0	15.9%	729.17	1.83
TorchMLP (T+E)	1260	0.188	21.2	17.5	0	19.6%	3646.83	2.16
TorchMLP (T)	1218	0.156	23.9	17.8	0	21.6%	3646.83	0.19
RandomForest (T)	1190	0.241	24.9	12.1	0	17.6%	729.17	0.33
XGBoost (D)	1188	0.102	24.9	21.5	0	18.1%	4.93	0.59
TabDPT (D)	1186	0.259	25.2	4.7	**1**	23.3%	33.52	20.75
FastaiMLP (T+E)	1182	0.205	25.2	12.1	0	24.2%	2721.87	12.59
LightGBM (D)	1180	0.158	25.4	19.6	0	21.3%	5.12	0.44
EBM (T+E)	1168	0.166	26.0	17.1	0	24.6%	1471.12	0.27
EBM (T)	1144	0.160	27.4	19.0	0	24.9%	1471.12	0.03
xRFM (D)	1129	0.094	28.4	22.1	0	23.9%	2.22	0.20
ModernNCA (D)	1110	0.051	29.1	26.8	0	25.9%	17.24	0.57
FastaiMLP (T)	1100	0.104	29.6	20.3	0	25.8%	2721.87	1.08
RealMLP (D)	1049	0.071	32.0	24.5	0	26.3%	26.02	4.18
EBM (D)	1046	0.095	32.0	24.3	0	27.7%	6.16	0.08
RandomForest (D)	1000	0.000	33.9	32.0	0	33.3%	0.74	0.15
TorchMLP (D)	972	0.004	35.0	33.6	0	28.5%	14.37	0.36
FastaiMLP (D)	952	0.038	35.9	31.9	0	33.4%	8.37	0.66
ExtraTrees (D)	862	0.013	38.6	35.1	0	39.1%	0.76	0.15
Linear (T+E)	752	0.000	41.8	41.5	0	45.6%	170.51	0.20
Linear (T)	718	0.000	42.3	41.9	0	45.8%	170.51	0.13
KNN (T+E)	714	0.000	42.4	41.0	0	52.5%	2.99	0.17
Linear (D)	689	0.000	42.9	42.7	0	46.9%	3.89	0.16
KNN (D)	544	0.000	45.4	45.1	0	69.5%	0.33	0.05
KNN (T)	541	0.000	45.6	45.5	0	58.3%	2.99	0.06

Table C.9: Full multiclass results on TabArena-Lite.

Model	Elo (↑)	Norm. score (↑)	Avg. rank (↓)	Harm. mean rank (↓)	#wins (↑)	Improva-bility (↓)	Train time per 1K [s]	Predict time per 1K [s]
AutoGluon 1.3 (4h)	**1779**	**0.673**	**5.2**	**3.2**	1	3.6%	1734.20	7.06
RealMLP (T+E)	**1721**	**0.660**	**6.6**	5.4	0	**3.1%**	6860.54	7.68
ModernNCA (T+E)	**1626**	**0.622**	**9.2**	**2.6**	**3**	5.4%	3811.43	7.58
LightGBM (T+E)	1602	0.478	10.1	7.5	0	6.0%	686.46	5.48
TabDPT (D)	1575	0.515	10.9	3.8	**2**	**3.9%**	16.97	8.70
xRFM (T+E)	1563	0.492	11.5	5.3	1	5.1%	365.57	0.72
CatBoost (T+E)	1558	0.435	11.8	8.9	0	5.7%	2895.38	1.32
TabM (T+E)	1529	0.439	12.8	8.5	0	4.3%	4228.53	1.19
CatBoost (T)	1515	0.406	13.3	6.8	0	5.9%	2895.38	0.07
xRFM (T)	1504	0.406	13.8	9.7	0	5.7%	365.57	0.09
LightGBM (T)	1498	0.354	13.9	10.2	0	6.8%	686.46	0.74
XGBoost (T+E)	1470	0.328	15.2	13.9	0	6.7%	848.99	2.38
XGBoost (T)	1470	0.325	15.2	13.3	0	6.7%	848.99	0.47
ModernNCA (D)	1457	0.323	15.6	11.0	0	8.1%	16.07	0.29
TabM (T)	1453	0.311	16.0	12.6	0	5.2%	4228.53	0.13
RealMLP (T)	1439	0.306	16.6	13.3	0	5.9%	6860.54	0.32
CatBoost (D)	1419	0.278	17.4	12.0	0	7.9%	8.35	0.09
ModernNCA (T)	1411	0.347	17.6	6.7	0	7.9%	3811.43	0.45
TabPFNv2 (T+E)	1377	0.410	19.1	**2.7**	**4**	4.9%	3805.62	10.41
TabM (D)	1361	0.232	20.1	16.5	0	6.9%	13.90	0.12
TabPFNv2 (T)	1314	0.281	22.1	6.9	0	6.4%	3805.62	0.26
TorchMLP (T+E)	1301	0.141	22.7	19.1	0	8.1%	4452.11	0.85
ExtraTrees (T+E)	1297	0.169	23.1	16.9	0	11.1%	161.73	0.78
RealMLP (D)	1274	0.085	24.0	21.6	0	8.2%	23.30	1.44
ExtraTrees (T)	1273	0.174	24.0	16.7	0	11.4%	161.73	0.12
TabPFNv2 (D)	1264	0.262	24.3	7.7	0	7.6%	2.78	0.32
TorchMLP (T)	1245	0.124	25.2	20.8	0	8.7%	4452.11	0.09
xRFM (D)	1236	0.149	25.8	8.6	1	11.7%	1.65	0.08
LightGBM (D)	1231	0.069	25.9	24.5	0	9.7%	2.03	0.30
XGBoost (D)	1195	0.104	27.4	24.5	0	10.4%	2.15	0.18
RandomForest (T+E)	1193	0.056	27.5	25.5	0	12.1%	526.17	0.77
RandomForest (T)	1142	0.046	29.9	27.5	0	12.8%	526.17	0.12
EBM (T+E)	1133	0.122	30.0	16.4	0	14.7%	2124.78	0.12
ExtraTrees (D)	1117	0.056	30.6	27.7	0	13.0%	0.42	0.06
FastaiMLP (T+E)	1086	0.005	31.8	30.6	0	13.1%	527.21	2.83
EBM (T)	1076	0.131	32.2	9.4	1	15.3%	2124.78	0.01
TorchMLP (D)	1075	0.015	32.4	30.3	0	12.9%	20.50	0.08
FastaiMLP (T)	1046	0.000	33.3	32.3	0	13.6%	527.21	0.31
EBM (D)	1009	0.071	34.7	30.3	0	16.0%	7.25	0.04
TabICL (D)	1006	0.000	34.8	34.2	0	14.2%	0.63	0.06
RandomForest (D)	1000	0.000	34.8	34.2	0	14.2%	0.63	0.06
FastaiMLP (D)	916	0.000	37.2	36.5	0	17.8%	3.08	0.29
KNN (T+E)	533	0.000	43.8	43.6	0	37.3%	2.25	0.15
Linear (T+E)	485	0.000	44.3	44.2	0	35.5%	46.50	0.14
KNN (T)	434	0.000	44.9	44.7	0	38.0%	2.25	0.03
Linear (T)	426	0.000	44.9	44.8	0	35.7%	46.50	0.04
Linear (D)	312	0.000	46.0	46.0	0	38.1%	1.16	0.08
KNN (D)	263	0.000	46.5	46.2	0	41.6%	0.04	0.02

Table C.10: Full binary classification results on TabArena.

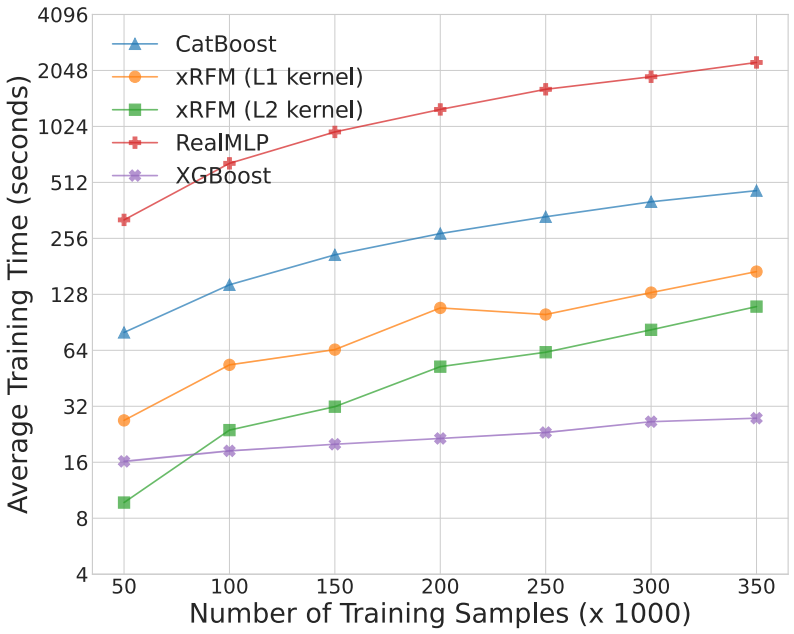

Figure C.1: Runtime comparison as a function of the number of training samples on the covertype dataset. Here, L1 kernel refers to the L_p^p kernel.

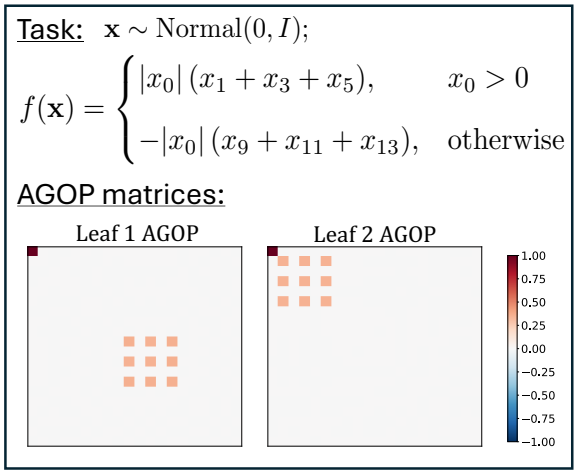

Figure C.2: Training xRFM on synthetic data where splitting on the top AGOP direction enables xRFM to learn locally relevant features.