Exploring Accuracy and Interpretability trade-off in Tabular Learning with Novel Attention-Based Models

Kodjo Mawuena Amekoe[1,3*], Hanane Azzag[1†], Zaineb Chelly Dagdia[2†], Mustapha Lebbah[2†], Gregoire Jaffre[3†]

*Corresponding author(s). E-mail(s): amekoe@lipn.univ-paris13.fr; Contributing authors: azzag@univ-paris13.fr; zaineb.chelly-dagdia@uvsq.fr; mustapha.lebbah@uvsq.fr; Gregoire.JAFFRE@bpce.fr;
†These authors contributed equally to this work.

Abstract

Apart from high accuracy, what interests many researchers and practitioners in real-life tabular learning problems (e.g., fraud detection, credit scoring) is uncovering hidden patterns in the data and/or providing meaningful justification of decisions made by machine learning models. In this concern, an important question arises: *should one use inherently interpretable models or explain full-complexity models such as XGBoost, Random Forest with post hoc tools?* Opting for the second choice is typically supported by the accuracy metric, but it is not always evident that the performance gap is sufficiently significant, especially considering the current trend of accurate and inherently interpretable models, as well as accounting for other real-life evaluation metrics such as faithfulness, stability, and computational cost of explanations. In this work, we show through benchmarking on 45 datasets that the relative accuracy loss is less than 4% in average when using intelligible models such as Explainable Boosting Machine (EBM). Furthermore, we propose a simple use of model ensembling to improve the expressiveness of TabSRALinear, a novel attention based inherently interpretable solution, and demonstrate both theoretically and empirically that it is a viable option for (1) generating stable or robust explanations, and (2) incorporating human knowledge during the training phase. Source code is available at https://github.com/anselmeamekoe/TabSRA.

Keywords: Tabular Data, Interpretability, Attention, Robust Explanation

1 Introduction

Since the promising results of the Transformer architecture on machine translation tasks (Vaswani et al. [1]), deep learning models continue to provide impressive performance, for example, in language modeling or computer vision. Convinced by the utility of the attention mechanism (used in the Transformer architecture), especially when modeling contextual information, many efforts have been made to use it in order to match or compete with the accuracy of boosted tree models such as XGBoost (Chen and Guestrin [2]) in tabular modeling (Somepalli et al. [3], Kossen et al. [4], Huang et al. [5], Gorishniy et al. [6]).

However, the models listed above typically use complex computation mechanisms or a large number of trees, making direct human inspections difficult. On the other hand, interpretability is usually (i) required by regulators in real-world applications (e.g., GDPR: Article 22 in Europe) and (ii) desired if the goal is to discover hidden patterns in the data (e.g., in fraud detection) or to ensure that the model does not learn a bias that may lead to significant drift in production. Therefore, recent studies, such as Lundberg and Lee [7], Ribeiro et al. [8], Lundberg et al. [9], have focused on developing post hoc methods to explain, at least locally, the predictions of full-complexity or black box models. Unfortunately, although these methods provide interesting properties, they are sometimes based on some computational mechanisms (e.g., exact Shapley value computation) or hypotheses (e.g., independence between features) that are difficult to achieve in practice, leading to biased explanations (Amoukou et al. [10], Kumar et al. [11]).

Still discussing interpretability, Rudin [12] provides a technical reason why an interpretable model might exist among the set of accurate models in any domain and encourages researchers to move toward finding this solution, especially for high-risk domains. As a result, a bench of inherently interpretable models (Nori et al. [13], Agarwal et al. [14], Chang et al. [15]) have been recently implemented and provide superior accuracy compared to classical statistical models (e.g., linear models). Some natural questions arise: *How much accuracy are we actually sacrificing when using an inherently interpretable model instead of a fully complex one?* In light of this question, we first show that the relative performance gap is less than 4%, considering the recent tabular learning benchmark of 45 datasets (59 tasks) introduced by Grinsztajn et al. [16]. Moreover, we show that accounting for some real-life metrics such as faithfulness, stability, and computational cost of explanations tends to favor the choice of inherently interpretable models. Second, after a deeper investigation, we propose the use of model ensembling to improve the modeling capacity of the TabSRALinear solution initially introduced by Amekoe et al. [17, 18] resulting in a new variant of TabSRAs. In addition, we demonstrate its superiority in terms of the stability of explanations and flexibility of incorporating human knowledge compared to existing solutions.

Overall, the key contribution of our work is to examine the trade-off between accuracy and interpretability in tabular learning by questioning the value of using post hoc tools to explain blackbox models in an era where accurate, inherently interpretable models are available. To achieve this:

1. We thoroughly evaluate and compare state-of-the-art inherently interpretable models and their full-complexity counterparts in terms of their relative predictive performance (Section 4.2.1).
2. We further compare baseline models based on some practical metrics (faithfulness and stability) and demonstrate the usefulness of inherently interpretable models (Section 4.2.2 and 4.2.3).
3. We propose a simple and intuitive method to improve the expressiveness of TabSRALinear, an attention-based, differentiable, and inherently interpretable solution (Section 3). We show how to incorporate some human knowledge during its training and demonstrate through real-world datasets its superiority in terms of stability and flexibility for human knowledge incorporation (Section 4.4).
4. We provide some recommendations on the choice of a tabular learning model in settings where interpretability is a key consideration (Section 5).

2 Background and related work

In this section, we delve into the state-of-the-art of tabular learning models and offer background information on existing interpretable solutions for tabular data. We contrast inherently interpretable models with those utilizing blackbox models and eXplainable Artificial Intelligence (XAI) tools.

2.1 State-of-the-art on supervised learning with tabular data

The accuracy based state-of-the-art. The debate on the best performing between decision tree-based models such as Random Forest (Breiman [19]), XGBoost (Chen and Guestrin [2]), LightGBM (Ke et al. [20]), CatBoost (Prokhorenkova et al. [21]) and Neural Networks (NNs) such as Trompt (Chen et al. [22]), FT-Transformer (Gorishniy et al. [6]) or SAINT (Somepalli et al. [3]), on tabular data is not settled. However, certain meta-features guide the choice of one over the other depending on the problem (Gorishniy et al. [6], Grinsztajn et al. [16], Borisov et al. [23], McElfresh et al. [24]). Regarding NNs, they are usually differentiable, making them easy to use, for example in multimodal settings (e.g., encoding tabular information with text, images, etc.) or multitask settings (Agarwal et al. [14]). Tree-based models, on the other hand, are capable of applying abrupt thresholding to features, making them more suitable for handling skewed feature distributions and other forms of dataset irregularities (Grinsztajn et al. [16], McElfresh et al. [24]). Moreover, in contrast to NNs, which generally require a long training time owing to the gradient back-propagation mechanism, tree-based models are known to be relatively fast to train. When it comes to understanding or explaining their outputs, these state-of-the-art models usually require post hoc tools because of their **blackbox** nature. However, can these post hoc tools really tell what the underlying model is doing?

Post hoc explanation of full-complexity models. The post hoc explanation of full-complexity models can be organized into two categories: feature selection and feature attribution (Huang and Marques-Silva [25]). Feature selection methods involve identifying a set (or subset) of features relevant for target prediction among which we have Anchors (Ribeiro et al. [26]) and Logic-based abduction (Marques-Silva and Ignatiev [27]). Feature attribution methods remain the most used eXplainable AI (XAI) techniques and assign importance to each feature. SHAP (Lundberg and Lee [7]) is a well-known unifying framework for feature attribution, based on the Shapley values. One interesting property of SHAP is additivity; that is, the marginal contribution of each feature sums to the output of the model (minus a constant baseline value). However, exact Shapley computation is difficult to achieve in practice, and as a result, an approximation of this quantity is usually estimated. Such an approximation is based on simulating the absence of features, a process that, depending on the nature of the data (correlation, interactions between features) and the underlying model, can result in biased approximation or huge computational costs (Chen et al. [28]). To alleviate this problem, model-specific SHAP has been developed, among which the well-known TreeSHAP solution (Lundberg et al. [9]) was designed for decision tree-based models. However, the question of the faithfulness of explanations remains (Amoukou et al. [10], Kumar et al. [11], Huang and Marques-Silva [25]).

Inherently interpretable solutions. Although the term "inherently interpretable" may seem more general or vague, we consider in this work that an inherently interpretable model is not one whose explanation derives from a heuristic (e.g., attention weight aggregation across several layers or heads) but from an explicit hypothesis and/or formulation. A classical example of such models is a statistical model in which the data are assumed to have some properties (e.g., for linear models, the feature should be independent, have a Gaussian distribution, and have a linear relation with the target feature). In modern machine learning based inherently interpretable models, instead of making assumptions about the distribution of data that are not necessarily verified, some intelligible constraints are put in the model formulations resulting in transparent or partially transparent decision making.

Among these models, we mention Neural Additive Models (NAMs) (Agarwal et al. [14]) which present a neural implementation of the classic Generalized Additive Models (GAMs). In addition to the marginal feature contribution, NAMs provide a *shape function*, which can help understand the global contribution of a given feature by visualization. NODE-G^{2}AM (Chang et al. [15]) and GAMI-Net (Yang et al. [29]) improve NAMs by considering pairwise interactions among the features. Another well-known example is the Explainable Boosting Machine (EBM) (Nori et al. [13]), a framework that proposes the implementation of GAMs using piecewise constant approximations and the boosting mechanism. To solve the problem of effect identification when adding pairwise interactions, additional heredity constraints (Yang et al. [29]) or interaction purification (Lou et al. [30]) is used. Consequently, to the best of our knowledge, it is difficult to handle higher-order interactions using GAMs based solutions. To alleviate this problem, we propose TabSRAs, an attention-based solution that does not require interaction identification. It is important to point out that EBM can capture abrupt changes. The same flexibility is possible for NAMs (using ExU activation) or NODE-G^{2}AM by imitating the discrete thresholding of decision trees (Popov et al. [31]). Although this property can help increase the expressiveness or help detect bias in data (Chen et al. [32]), it may result in discontinuities that are sometimes difficult to justify. That is, similar explanations are expected for similar input data, the so-called stability or robustness property (Alvarez-Melis and Jaakkola [33]). With TabSRAs, the stability can be easily controlled using the range of attention coefficients.

2.2 Interpretable solutions for tabular data problems

In this part, we provide background on well-known existing interpretable solutions for tabular learning problems, starting with inherently interpretable models (Section 2.2.1). Subsequently, in Section 2.2.2, we describe post hoc interpretability solutions for full-complexity models with a focus on additive feature attribution methods, recalling that these are the most studied and used XAI techniques in the literature.

The challenge in most supervised tabular learning problems is to estimate the output $\hat{y} = \sigma^{-1}(f(\mathbf{x}))$ given the feature vector $\mathbf{x} = (x_1, ..., x_p) \in \mathbb{R}^p$. σ represents the link function (e.g., $\sigma(\mu) = \log(\frac{\mu}{1-\mu})$ for binary classification and $\sigma = \mathit{Identity}$ for regression tasks). The model f is usually learned using the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $y_i \in \{0,1\}$ for binary classification or $y_i \in \mathbb{R}$ for regression tasks.

2.2.1 Inherently interpretable models

Linear models. These are highly transparent models obtained by a simple linear combination of input features as follows:

$$\begin{aligned} \sigma(\hat{y}) &= \beta_0 + \boldsymbol{\beta} \cdot \mathbf{x} \\ &= \beta_0 + \beta_1 x_1 + ... + \beta_i x_i + ... + \beta_p x_p \end{aligned} \tag{1}$$

$\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)$ is the linear regression coefficients vector and β_0 is the intercept or bias term. β_i is a global importance measure of the i-th feature while $\beta_i x_i$ is the effect used in the explanation of individual output. **LR** refers to linear models in this study, that is, Linear Regression for regression and Logistic Regression for binary classification tasks.

Generalized Additive Models. The Generalized Additive Models (GAMs) are formalized as follows:

$$\sigma(\hat{y}) = \beta_0 + \sum_{i=1}^{p} f_i(x_i) \tag{2}$$

where f_i is a shape function that given the i-th input feature x_i produces the output $f_i(x_i) \in \mathbb{R}$. Linear models (LR) are a particular case of GAMs in which every shape function is linear, that is, $f_i(x_i) = \beta_i x_i$.

In this work, we consider among GAMs the Explainable Boosting Machine (EBM) (Nori et al. [13]), which is arguably one of the most established implementations. EBM uses piecewise constant functions and a boosting mechanism to fit the shape functions.

Generalized Additive Models with Pairwise interactions. In order to improve the expressiveness of GAMs, pairwise interactions are used following Equation 3.

$$\sigma(\hat{y}) = \beta_0 + \sum_{i=1}^{p} f_i(x_i) + \sum_{i<j} f_{ij}(x_i, x_j) \tag{3}$$

In the EBM framework (Nori et al. [13]), the maximum number of interaction terms f_{ij} is controllable, helping to preserve the intelligible aspect of the overall model. We call **EBM** the Explainable Boosting Machine model with pairwise interaction and **EBM_S** the one without interaction terms.

Decision Trees. Apart from the above listed additive inherently interpretable models, we also distinguish non-additive solutions. In this category, we cite **Decision Trees (DT)**, which follow the tree path to produce interpretable decision rules, especially when the depth of the tree is reasonable. In this study, the Scikit-Learn implementation (Buitinck et al. [34]) was used.

2.2.2 Full-complexity models

In this part, we present some post hoc interpretability solutions for accuracy based state-of-the-art models (Gorishniy et al. [6], Grinsztajn et al. [16], Borisov et al. [23], McElfresh et al. [24]). We split these models into two categories: Decision tree ensemble and Neural Nets models (NNs).

Decision tree ensemble. Model ensembling (or ensemble methods) consists of aggregating the decision or prediction from different individual models (called base or weak learners) to obtain better predictive performance or expressiveness. In the decision tree ensemble, the base learner is a decision tree model. Generally, a decision tree ensemble with an important number/size of learners requires post hoc tools when explaining their decisions.

Owing to the node-path structure, decision tree ensembles can be easily combined with some model-specific post hoc interpretability tools. For example, TreeSHAP (Lundberg et al. [9]) is a well-suited and computationally efficient SHAP explanation framework for tree-based models. Another framework, Xreason (Ignatiev et al. [35]), aims at reasoning (formally) regarding explanations for tree ensembles with acceptable scalability.

In our evaluation, we consider:

- **Random Forest (RF)** (Breiman [19]). RF is one of the most popular decision tree ensembles. RF fits a number of independent decision trees on various subsamples of the dataset (bagging) and uses aggregation to improve the predictive accuracy and control overfitting. We consider the scikit-learn implementation[34].
- **XGBoost** (Chen and Guestrin [2]). In XGBoost, decision trees are sequentially combined using a boosting mechanism. XGBoost remains the leading state-of-the-art model for several real-life use cases and tabular learning ML competitions.

- **CatBoost** (Prokhorenkova et al. [21]). Similarly to XGBoost, CatBoost is also based on the boosting mechanism. In CatBoost, the *ordered boosting* strategy is utilized to address the target leakage issue that can arise in the original implementation of XGBoost or LightGBM. Additionally, CatBoost's implementation incorporates a built-in method for handling categorical and text features. In recent studies, such as Chen et al. [22], McElfresh et al. [24], CatBoost has been shown to provide slightly better predictive performance compared to XGBoost. Therefore, we have included it in our predictive performance evaluation

Neural Nets. NNs (especially deep NNs) usually require post hoc techniques for explaining their decisions. Among these tools, LIME (Ribeiro et al. [8]) and KernelSHAP (Lundberg and Lee [7]) are model-agnostic. DeepSHAP is a Shapley value based specific tool for NNs, inspired by DeepLIFT (Shrikumar et al. [36]). In the theoretical design of these tools, the features are assumed to be independent, and the learned functions are assumed to be linear, at least locally. These assumptions are generally difficult to meet in real-life tabular settings; consequently, these solutions are generally less reliable than their tree-based counterparts.
Based on the results from previous studies (Gorishniy et al. [6], Grinsztajn et al. [16], Borisov et al. [23], McElfresh et al. [24]), we consider in our study the following NNs designed specifically for tabular data:

- **MultiLayer Perceptron (MLP)**. It is a full complexity architecture that can model nonlinear effects and interactions. It somehow provides us the achievable accuracy by shallow and differentiable architectures.
- **ResNet**. ResNet is similar to MLP but includes skip connections (Gorishniy et al. [6], Grinsztajn et al. [16]).
- **FT-Transformer** (Gorishniy et al. [6]). FT-Transformer combines a Feature Tokenizer module and the classical Transformer block (Vaswani et al. [1]) resulting in superior accuracy over the classical MLP (Gorishniy et al. [6], Grinsztajn et al. [16]).
- **SAINT** (Somepalli et al. [3]). In SAINT, the attention mechanism (Vaswani et al. [1]) is used in the contextual embedding of features. In addition, batch inter-sample attention is used to get better representations.

In the following section, we will introduce the proposed attention based solution as a part of inherently interpretable models.

3 Propositions: TabSRAs

In this section, we describe the TabSRAs inherently interpretable solution initially introduced in (Amekoe et al. [17, 18]). We also describe the proposed improvement of expressiveness as well as a way to interpret its results and incorporate human knowledge during its training phase.

TabSRAs (Fig. 1) are based on the Self-Reinforcement Attention (SRA) mechanism. In a nutshell, given the raw input $\mathbf{x} \in \mathbb{R}^p$, the SRA block (Fig. 2) denoted as a function $a(.) : \mathbb{R}^p \longrightarrow \mathbb{R}^p$ produces an attention vector $\mathbf{a} = (a_1, ..., a_i, ...a_p)$ which is further used to produce a reinforced input $\mathbf{o} = (o_1, ..., o_i, ..., o_p)$ as follows:

$$\mathbf{o} = \mathbf{a} \odot \mathbf{x} \tag{4}$$

where $\odot$ is the element-wise multiplication.

The learned reinforced vector $\mathbf{o}$ represents a new feature basis, where each component is guided by the raw input, that is, $o_i = a_i x_i$, helping to maintain the semantics of each dimension. In this space, some components may be shrunk, for instance, to zero using the attention weights $a_i \geq 0$. In other words, the attention vector acts like a soft instance-wise feature selector or mask. We provide some visual illustrations of how raw data are reinforced in Section 3.1.

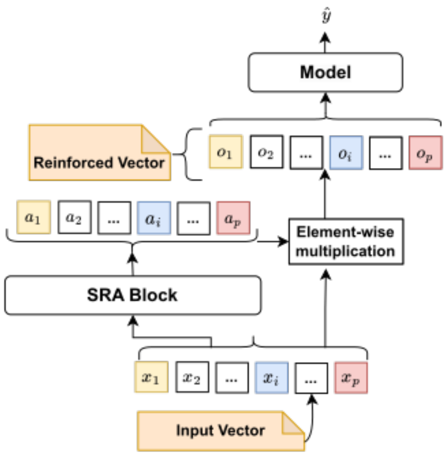


Fig. 1: TabSRAs architecture. The attention vector $\mathbf{a} = (a_1, ..., a_p) \in \mathbb{R}^p$ provided by the SRA block is used to produce a *reinforced* vector $\mathbf{o} = (o_1, ..., o_p) \in \mathbb{R}^p$.

Finally, this reinforced vector is aggregated using a highly **transparent** downstream model (e.g., linear models, decision trees, or rules) to produce the final output. Because we are interested in end-to-end training and differentiable architecture, in this work, we consider a linear downstream model resulting in the so-called TabSRALinear architecture (Section 3.2).

3.1 SRA block

In the SRA block, the input vector $\mathbf{x} = (x_1, ...x_i, ..., x_p) \in \mathbb{R}^p$, is encoded into p keys in $K = [\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_i, ..., \mathbf{k}_p]^T$ with $\mathbf{k}_i = (k_i^1, ..., k_i^{d_k}) \in \mathbb{R}^{d_k}$ using a key encoder and queries matrix $Q = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_i, ..., \mathbf{q}_p]^T$ with $\mathbf{q}_i = (q_i^1, ..., q_i^{d_k}) \in \mathbb{R}^{d_k}$ using a query encoder (Fig. 2).

Table 1: Some differences between the SRA block and the classical Transformer block. $K \in \mathbb{R}^{p\times d_k}$ the matrix of keys and $Q \in \mathbb{R}^{p\times d_k}$ the matrix of queries

Point	Classical Transformer block	SRA block
Attention weight	$A = softmax(\frac{QK^T}{\sqrt{d_k}}) \in \mathbb{R}^{p\times p}$	$\mathbf{a} = \frac{\sum_{d_k} Q\odot K}{d_k} \in \mathbb{R}^p$
Value encoding	Yes	No
Additional processing (residual connection, LayerNorm)	Yes	No

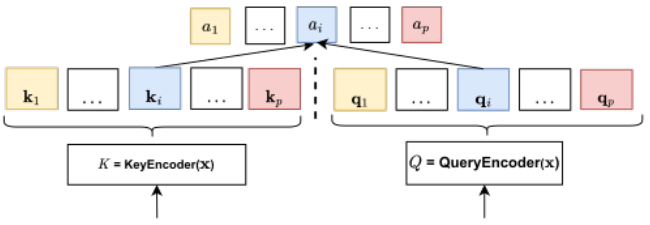


Fig. 2: SRA Block. The $KeyEncoder$ (resp. $QueryEncoder$) produces directly p keys (resp. queries)

Furthermore, with a Sigmoid activation function, all elements k_i^j of K (resp. q_i^j of Q) are scalar numbers bounded in $[0, 1]$.

The keys in K are compared to the queries Q component by component using the scalar product as in Vaswani et al. [1]. This allows quantifying the alignment of different transformations of the same input calculating the attention weights $\mathbf{a} = (a_1, .., a_i, ..., a_p)$ as follows :

$$a_i = \frac{\mathbf{q}_i \cdot \mathbf{k}_i}{d_k} \quad \text{for} \quad i \in 1, \cdots, p \tag{5}$$

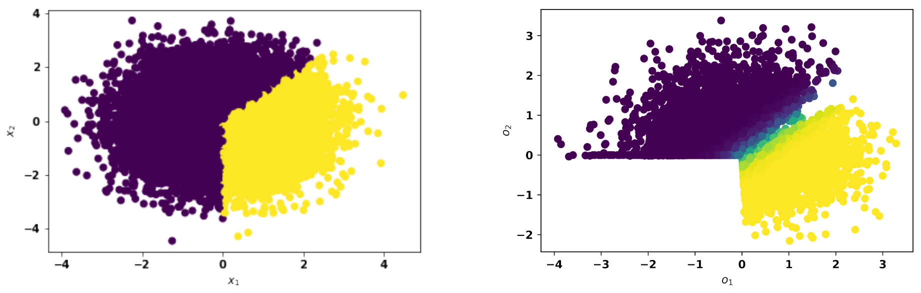
We further use the scaling by d_k in order to reduce the magnitude of the dot-product and to obtain dimension-free attention coefficients $a_i \in [0, 1]$. Some important differences between the proposed SRA block and the vanilla Transformer block (Vaswani et al. [1], Somepalli et al. [3], Kossen et al. [4], Huang et al. [5], Gorishniy et al. [6]) are summarized in Table 1.

How raw data are reinforced using the SRA block.
To illustrate how the raw data is reinforced in practice using the SRA block, we use a 2D toy dataset with the objective of facilitating the visualization. We consider the following function:

$$F_1(x) = 5x_1 - 5x_2\mathbb{1}_{x_1>0} \quad \text{and} \quad y = \{\mathbb{1}_{p>0.5} \quad \text{with} \quad p = 1/(1+e^{-F_1(x)}\} \tag{6}$$

As simple as it may seem, this function cannot be directly modeled with a linear model due to the term $\mathbb{1}_{x_1>0}$, which forces x_2 to have no effect on the output when $x_1 < 0$. Using the reinforced version of the raw inputs helps to alleviate this problem;

as shown in Fig. 3. Fig. 3a shows the original data distribution, with the yellow color indicating the class of interest. In Fig. 3b, we show the representation learned by multiplying the raw inputs with SRA coefficients. The green color represents a decision boundary to separate the two classes. Through multiplication, values of x_2 are significantly reduced, for instance, to 0 when needed (i.e., $o_2 \sim 0$ when $x_1 < 0$, $x_2 < 0$), which makes the classes easy to separate with the downstream model (e.g. a simple linear or logistic function). Moreover, this representation helps to understand the global behavior of the SRA based model, where it is confident in predicting class 1 (in yellow color) and where it is less confident, as highlighted by the green color (Fig. 3b).



(a) $\mathbf{x}$ (b) $\mathbf{o} = \mathbf{a} \odot \mathbf{x}$

Fig. 3: Illustration of the reinforcement process on 7500 synthetic data points with 0 mean, unit variance Gaussian distribution. The yellow color is used for the class of interest.

3.2 TabSRALinear: SRA Block and Linear downstream model

We investigate in this work a linear combination of the reinforced features (Equation 4) resulting in the additive TabSRA model (Fig. 1). This instantiation called TabSRALinear can be formalized as follows:

$$\begin{aligned}\sigma(\hat{y}) &= \beta_0 + \boldsymbol{\beta} \cdot \mathbf{o} \\ &= \beta_0 + \beta_1 o_1 + ... + \beta_i o_i + ... + \beta_p o_p \\ &= \beta_0 + \beta_1 a_1 x_1 + ... + \beta_i a_i x_i + ... + \beta_p a_p x_p\end{aligned} \tag{7}$$

$\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)$ is the linear regression coefficient vector, β_0 the bias term and $\mathbf{a} = (a_1, .., a_i, ..., a_p)$ the attention weights.

Interpretability with TabSRALinear. Using linear models to model data or phenomena that exhibit feature interactions results in internal conflicts between input components, poor accuracy, or even misleading interpretations. Considering these conflicts, the attention weight vector **a** may enhance or reduce some components (of the input vector) at strategic and specific positions depending on the context (or the whole information in x), resulting in an internal equilibrium. Therefore, it is natural to interpret:

- a_i as the correction that the feature x_i received from other features or itself (due to the interactions) before influencing the output. On the instance level, $a_i = 0$ corresponds to the particular case where x_i has no effect on the output.
- $\beta_i a_i x_i$ as the contribution (the prediction importance) of the feature x_i to the output.

Furthermore, visualizing or computing the gradient $\beta_i a_i x_i$ vs. x_j will help to understand the global contribution of the feature x_i as well as interactions (for $i \neq j$). Considering the important question of simulating the absence of features that arises with some well-known XAI tools (Chen et al. [28]) as highlighted in Section 2.1, it is worth nothing to point out that for TabSRALinear, the absence of the $i-$th feature corresponds to the case where $x_i = 0$. In addition, owing to interactions, an input feature can take a zero value (i.e., $x_i \approx 0$) but still influence the output through other features.

Human knowledge incorporation in TabSRALinear. For some problems (e.g., credit scoring), it is crucial to incorporate domain expert knowledge such as positive effects or monotonic constraints with respect to some features during or after training.

In TabSRALinear, the regression coefficient β_i (Equation 7) controls the overall "sense" (positive or not) of the effect of the $i-$th feature on the output. Therefore, a positive (resp. negative) sense constraint (knowledge) is added by setting $\beta_i \geq 0$ (resp. $\beta_i \leq 0$) during the training. Furthermore, a monotonic increasing (resp. decreasing) constraint on the $i-$th feature is added by setting at the same time $\beta_i \geq 0$ (resp. $\beta_i \leq 0$) and the attention weight a_i to a constant value, typically one. The latter constraint is equivalent to assuming that the effect of feature i on the output is not influenced by other features, i.e., linear. A practical use case is demonstrated in Section 4.4.1.

3.3 On the robustness of TabSRALinear's explanations

In this section, we provide a theoretical analysis and justify why TabSRALinear is a viable solution for robust self-explainability in tabular learning settings.

Robustness remains an important topic for feature attribution based explanation systems, with the goal of designing a convenient metric to assess the similarity of explanations provided for similar inputs (Alvarez-Melis and Jaakkola [33], Alvarez Melis and Jaakkola [37], Agarwal et al. [38]). This is mainly because many state-of-the-art interpretability tools (Lundberg and Lee [7], Ribeiro et al. [8], Lundberg et al. [9]) operate on a single data point and the use of point-wise explanations to understand complex models is perhaps too optimistic or can lead to a false sense of understanding (Alvarez-Melis and Jaakkola [33]). To address this limitation, one might want to go beyond individual points and examine the behavior of the models in the neighborhood of certain target points. Therefore, interpretability methods or inherently interpretable models must produce explanations that are stable or robust to local perturbations (Alvarez-Melis and Jaakkola [33]).

The TabSRALinear model is designed to produce relatively robust explanations considering the following theorem:

Theorem 1. *The feature attributions produced by TabSRALinear (Equation 7) is locally stable in the sense of Lipschitz, that is, for every* $\mathbf{x} \in \mathbb{R}^p$*, there exists* $\delta > 0$ *and* $L_{\mathbf{x}} \geq 0$ *finite such that:*

$$\|\mathbf{x} - \mathbf{x}'\|_1 < \delta \implies \|\boldsymbol{\beta} \odot a(\mathbf{x}) \odot \mathbf{x} - \boldsymbol{\beta} \odot a(\mathbf{x}') \odot \mathbf{x}'\|_1 \leq L_{\mathbf{x}} \|\mathbf{x} - \mathbf{x}'\|_1 \tag{8}$$

With $L_{\mathbf{x}} = \|\boldsymbol{\beta}\|_\infty[\|\mathbf{a}\|_\infty + L_{\mathbf{a}}(\|\mathbf{x}\|_\infty + \delta)]$ *and* $L_{\mathbf{a}} \geq 0$ *the Lipschitz constant of the SRA block.*

The objective of Theorem 1 is not to provide the tightest bound of the Lipschitz constant, but to provide some justification for the attention computation.

First, we notice the quantity $\|\mathbf{x}\|_\infty$ in the expression of $L_{\mathbf{x}}$, which shows that the raw input data should be **bounded**. This is a common situation when using NNs. That is, the data scaling technique (using the minimum and maximum or the mean and standard deviation) or quantile transformation is used to speed up the convergence.

We also have the term $\|\mathbf{a}\|_\infty$ which proves that the smaller the attention weights, the more stable the explanations. Using the scaling of Equation 5, we have $\|\mathbf{a}\|_\infty = 1$ and we can identify in the first term of $L_{\mathbf{x}}$ the Lipschitz constant of linear models, which is simply $\|\boldsymbol{\beta}\|_\infty$. Moreover, in situations where there are no interactions (and nonlinear effects) between features, almost all attention weights are expected to be constant (i.e., $L_{\mathbf{a}} \approx 0$), and TabSRALinear is reduced to classical linear models.

We present the empirical evidence of the stability of TabSRALinear's feature attribution compared with that of state-of-the-art interpretability tools such as (Lundberg and Lee [7], Lundberg et al. [9]) in Section 4.2.3.

We kindly invite the reader to refer to Section A.2 for a complete proof of Theorem 1.

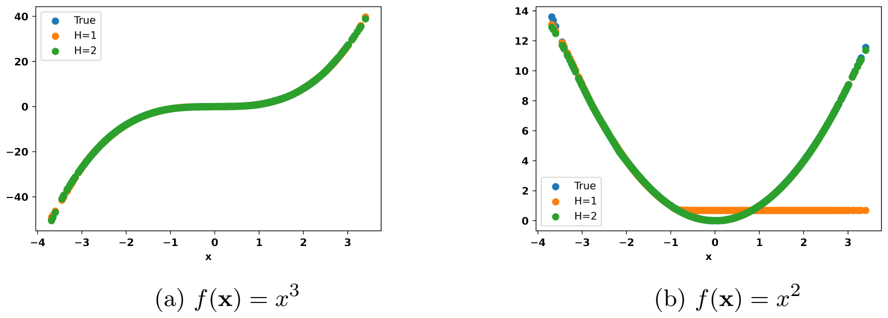
3.4 Improving TabSRALinear using model ensemble

With the formulation of TabSRALinear (Equation 7), we assume that the learned function is monotonic or linear around the origin, that is, $\mathbf{x} \approx 0$. In other words, the Lipschitz constant (Theorem 1) is $L_{\mathbf{x}} = \|\boldsymbol{\beta}\|_\infty(1 + \epsilon)$ where $\epsilon = L_{\mathbf{a}}\delta \longrightarrow 0$.
However, for some problems, the learned phenomena might not be monotonic around the origin (or around a given reference value), as illustrated by the quadratic function in Fig. 4b. For such a problem, the TabSRALinear may provide a poor modeling performance.

To mitigate this problem, we propose the use of model ensembling (Section 2.2.2), in which TabSRALinear is the base learner. To preserve the intelligibility of the overall architecture/ensemble, we found it more convenient to use the sum aggregation of the decisions of individual TabSRALinear (Equation 7) as follows:

$$\begin{aligned}\sigma(\hat{y}) &= \sigma(\hat{y}_1) + \cdots + \sigma(\hat{y}_h) + \cdots + \sigma(\hat{y}_H) \\ &= \beta_0^1 + \boldsymbol{\beta}^1 \cdot (\mathbf{a}^1 \odot \mathbf{x}) + \cdots + \beta_0^h + \boldsymbol{\beta}^h \cdot (\mathbf{a}^h \odot \mathbf{x}) + \cdots + \beta_0^H + \boldsymbol{\beta}^H \cdot (\mathbf{a}^H \odot \mathbf{x}) \\ &= \sum_{h=1}^{H} \beta_0^h + \left(\sum_{h=1}^{H} \beta_1^h a_1^h \right) x_1 + \cdots + \left(\sum_{h=1}^{H} \beta_i^h a_i^h \right) x_i + \cdots + \left(\sum_{h=1}^{H} \beta_p^h a_p^h \right) x_p\end{aligned} \tag{9}$$

with $\boldsymbol{\beta}^h$ (resp. $\mathbf{a}^h$) is the h-th linear model's coefficients (resp. the attention vector from the SRA block h) and β_0^h is the corresponding bias term. H represents the size or the number of learners in the ensemble. The TabSRALinear ensemble is equivalent to the Multi-head attention idea [1, 3–5] where context representations provided by different attention heads are first concatenated and thereafter transformed by a linear layer producing a new context representation or of the final output.



(a) $f(\mathbf{x}) = x^3$ (b) $f(\mathbf{x}) = x^2$

Fig. 4: Illustration of the importance of the TabSRALinear ensemble. A single TabSRALinear (H=1) perfectly fits the monotonic function shown in Fig. 4a. However, it struggles to fit the quadratic example (orange in Fig. 4b). Using an ensemble of two (H=2) TabSRALinear models alleviates this problem: one TabSRALinear is specialized in fitting the first branch of the parabolic, while the second model of the ensemble fits the other branch.

In general ensembling increases the number of parameters (compare to one single base learner), consequently affects transparency. However due to the additive nature, TabSRALinear ensemble (Equation 9) preserves the interpretable aspect of TabSRALinear that is, $\left(\sum_{h=1}^{H} \beta_i^h a_i^h\right) x_i$ represents the contribution or the effect of the feature x_i to the output. Nonetheless, a high number of H may result in less robust explanations (please refer to Section A.1 for the theoretical proof). Therefore, we recommend considering the ensemble size H from the set {1,2}, and for all results presented in the paper, including the case study, we optimized the ensemble size $H \in \{1, 2\}$. Furthermore, our ablation study (Table 6, Section 4.3) indicates that increasing H beyond 2 does not significantly enhance performance but instead increases running time.

We will refer to the TabSRALinear ensemble as TabSRALinear, and we will use the terms ensemble or head interchangeably.

4 Experiments

In this section, we report the key findings of the empirical comparison of inherently interpretable models to each other, as well as to full-complexity counterparts. Please note that the supplementary materials contain (1) a complete results analysis regarding the stability of TabSRALinear's explanations (Section B.4); (2) additional visualizations and results (Section B).

4.1 Experimental setup

4.1.1 Evaluation metrics

Our goal is to provide concrete numerical results to help practitioners and researchers choose between inherently interpretable solutions and explanations for full-complexity models. Given that the main justification for explaining blackbox models is to retain predictive performance while achieving interpretability, we believe it is crucial to compare both approaches (inherently interpretable models versus black-box models with XAI tools) based on the following criteria:

- **Predictive performance** (Section 4.2.1). To assess predictive performance, we utilize the coefficient of determination (R^2) for regression tasks and accuracy for classification tasks in the *middle-scale benchmark* (please refer to Section 4.1.2 for a description of the datasets). For datasets in the *Default benchmark* that are imbalanced, we evaluate the Area Under the Receiver Operating Characteristic curve (AUCROC), and particularly for highly imbalanced datasets such as the Credit Card Fraud dataset (Table 2), we measure the Area Under the Precision-Recall curve (AUCPR) instead of AUCROC (as discussed in Davis and Goadrich [39]).
- **Faithfulness** (Section 4.2.2). Given that the faithfulness of explanations provided by blackbox models with XAI tools is often questioned in the literature (Amoukou et al. [10], Huang and Marques-Silva [25, 40]), we also compare the two approaches based on this criterion. Specifically, we measure the precision in identifying the truly relevant features.
- **Stability/Robustness** (Section 4.2.3). Which option can provide more robust/coherent explanations, meaning similar explanations for similar inputs?

We further illustrate the unique capabilities of our proposed solution, TabSRALinear, and knowledge incorporation through two real-world applications (Section 4.4).

4.1.2 Datasets

To evaluate the models based on the criteria listed above, we considered three types of datasets:

Table 2: Benchmark datasets

Datasets	# Datapoints	# features	# Categorical features	Positive Class (%)
Bank Churn	10,000	9	2	20.37
Credit Default	30,000	22	3	22.16
Credit Card Fraud	284,807	29	0	0.17
Heloc Fico	10,459	23	0	47.81

- *Middle-scale benchmark*: Recently introduced in Grinsztajn et al. [16], it contains 45 real-world datasets (59 tasks precisely) of both numerical and heterogeneous binary classification or regression problems. We mainly use these datasets to assess the predictive performance of benchmark models. Note that the authors truncated the training set to 10,000 and the test set to 50,000 in order to ease the assessment of inductive biases of models (have homogeneous benchmarks). Also, cross-validation is used with a number of folds ranging from 1 to 5 depending on the test data size. Finally, in each fold, a Train/Validation/Test split is used (please refer to Section B.1.1 for more details).
- *Default benchmark*: we also consider four widely used datasets in tabular learning settings (we provide the summary in Table 2). We used two of these datasets to evaluate the robustness of explanations (Credit Card Fraud, Heloc Fico) and the remaining for the applicative case study.
- *Synthetic benchmark*: we finally consider synthetic datasets to assess explanations' faithfulness. As the ground truth for real-world datasets are generally unavailable for this purpose, we generate three synthetic datasets based on (1) additivity: we consider only additive functions. Recall that most feature attribution methods are used to explain the additive structure in modeled phenomena (Kumar et al. [11]) and (2) unicity: there is only one optimal explanation or feature attribution for every observation. Therefore, models that perfectly fit the data in terms of accuracy should converge with this explanation. In this concern, we consider datasets with five features $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$ of size 30,000 based on the Gaussian distribution (of mean 0 and variance 1) as follows:

$$\textit{Synthetic 1}: \quad y = 5x_1 - 5x_2 \tag{10}$$

$$\textit{Synthetic 2}: \quad y = x_1^2 \tag{11}$$

$$\textit{Synthetic 3}: \quad y = (5x_1 - 5x_2)\mathbb{1}_{x_5 \leq 0} + (5x_3 - 5x_4)\mathbb{1}_{x_5 > 0} \tag{12}$$

4.1.3 Experimental details on the predictive performance evalution

Tuning. For the *Middle-scale benchmark*, we conducted random hyperparameter tuning for each inherently interpretable algorithm (described in Section 2.2.1, including TabSRALinear) and CatBoost (Section 2.2.2) over a period of 9 days (216 hours) on a 64-core processor CPU machine.

For the remaining full-complexity models (Section 2.2.2), the tuning results are reported from the previous benchmarking (Grinsztajn et al. [16]) where a 64-Core Processor CPU machine is used for tree-based models and GPUs for NNs (please refer to [16] for more details). In this study, we make our best effort to ensure the convergence of algorithms, particularly NNs that have an important number of hyperparameters to tune and, at the same time, have the fairest comparison possible. Therefore, we consider $\approx$ 2x iterations per dataset for MLP, RF and $\approx$ 3x for XGBoost, CatBoost where x is the number of iterations for TabSRALinear ($\approx$ 95 per dataset). For ResNet, FT-Transformer and SAINT which are more resource consuming (see the training time Table 3), we used the same number of iterations as TabSRALinear.

Moreover, in order to have a bootstrap-like estimate of the test score (Grinsztajn et al. [16]), for each algorithm and dataset we sample 50% of iterations and repeat this process 10 times with different random seeds. Note that using all iterations once does not change our conclusions (Section B.6.1).

Results aggregation accross datasets. We consider the test accuracy for binary classification tasks and R^2 score for regressions. To aggregate the results across datasets, we use a metric similar to the Average Distance to the Minimum (ADTM) (Wistuba et al. [41]). More precisely, for each algorithm, we divide the achieved score with the one of the best performing for a given dataset. In this way, we get a unitless score in $[0, 1]$ which is further aggregated to produce the relative average score to best. We also consider the rank and the running time (training+testing).

Data processing. Unless otherwise specified, categorical inputs are one-hot encoded for models which do not handle them natively[1], and numerical inputs are scaled[2] using mean and standard deviation to accelerate the convergence of NNs models.

4.2 Benchmark results

4.2.1 Prediction performance

In Table 3, we report the predictive performance results of inherently interpretable and full-complexity models.
The main findings are:

- The relative average predictive performance gap between the overall best-performing inherently interpretable (EBM) and the full-complexity counterpart (CatBoost) is less than 4%.
- Regarding NNs models, the inherently interpretable solution TabSRALinear provides a very competitive performance compared with MLP-like architectures (MLP, ResNet) with a gap of less than 2.5%.

For tasks where predictive performance is the only consideration, decision tree ensemble models (such as CatBoost, XGBoost, Random Forest), are clearly preferable solutions, especially when run time is considered. Compared with tree based models, NNs have a longer general training time when using gradient descent optimization.

[1]The neural networks (NNs) described in Section 2.2.2 include an embedding layer designed to manage categorical features, as outlined in the work by Gorishniy et al. [6]. NNs (Section 2.2.2) are equipped with an embedding layer for handling categorical feature (Gorishniy et al. [6]) and CatBoost also offers a native approach for handling categorical features through a combination of target encoding (referred to as 'Borders') and Frequency Encoding (referred to as 'Counter')

[2]Gaussian quantile transformation was used in Grinsztajn et al. [16]; however, as this transformation is not bijective/linear, we only use scaling to preserve interpretability in the initial feature space.

Table 3: Predictive performance of models across 59 tasks (45 datasets). We report the rank over all tasks, the relative test score (Accuracy/R^2) and running time (training+inference) in seconds.

Model	Rank				Mean Test Score			Mean Running Time [3]	
	min	max	mean	median	mean	median	std	mean	median
DT	2	12	10.476	11	0.868	0.907	0.163	0.294	0.032
EBM_S	1	11	7.692	8	0.931	0.955	0.087	23.997	5.144
EBM	1	10	5.477	5	0.959	0.982	0.067	97.837	19.737
LR	7	12	11.701	12	0.760	0.839	0.232	21.124	19.716
TabSRALinear	1	12	8.225	9	0.901	0.971	0.197	47.576	38.073
MLP	1	12	6.992	8	0.924	0.973	0.159	24.165	19.256
ResNet	1	12	7.120	8	0.909	0.975	0.195	95.123	53.212
SAINT	1	12	5.625	6	0.946	0.982	0.093	216.053	126.841
FT-Transformer	1	11	5.203	5	0.944	0.984	0.109	126.589	77.465
Random Forest	1	10	4.214	4	0.985	0.992	0.021	39.030	8.252
XGBoost	1	11	2.728	2	0.988	0.998	0.029	18.254	12.561
CatBoost	1	10	2.545	2	0.991	0.999	0.021	12.176	4.025

[2]For full-complexity models, optimization results are taken from [16] where GPUs are used for NNs models. For TabSRALinear and LR we only use CPUs therefore their running time can be reduced when considering GPUs.

This fact is highlighted by the running time of Linear models (LR) optimized by gradient descent compared to the one of Decision Tree (DT); both are models known to be very shallow.

We can notice (Table 3) that the average relative test score between CatBoost and TabSRALinear is up to 9% while the median (i.e., on 50% of datasets) is less than 3%. This is due to the poor performance of the latter on some decision tree friendly datasets (see the comparison results on delays_zurich_transport Table B14 and yprop_4_1 Table B15). On such datasets, a simple decision tree may have superior predictive performance over MLP-like architectures or shallow NNs. FT-Transformer and SAINT, owing to their feature tokenization process, can still provide acceptable performance in such situations (see analysis on when the FT-Transformer is better than ResNet [6]). However, this comes with the cost of an important number of learnable parameters. We believe that combining numerical discretization (Gorishniy et al. [42]) and TabSRALinar or simply using piecewise constant approximators such as EBM could be a good predictive inherently interpretable solution in such situations.

For this *middle-scale benchmark*, CatBoost significantly outperformed EBM (with a gap above 5%) for 9/59 tasks. We argue that this is because of the difficulty of GAM based inherently interpretable solutions to conveniently handle higher-order interactions (above 3). For such cases, TabSRALinear appears to be a good solution, particularly when the dataset does not exhibit strong discontinuities or irregularities (see the results for covertype Table B20, pol, and sulfur Table B15).

Table 4: Relevant feature discovery capacity. Precision is used as a metric. R^2 (the higher, the better) is used to evaluate the test's predictive performance. The bold numbers denote the best for each dataset and metric.

Datasets	Models	Test performance	Precision
Synthetic 1	LR	1.00	1.00
	TabSRALinear	1.00	1.00
	EBM_S	1.00	1.00
	EBM	1.00	1.00
	XGBoost+TreeSHAP	1.00	1.00
Synthetic 2	LR	0.00	0.00
	TabSRALinear	**1.00**	1.00
	EBM_S	0.99	0.99
	EBM	0.99	0.99
	XGBoost+TreeSHAP	0.99	1.00
Synthetic 3	LR	0.50	0.52
	TabSRALinear	1.00	**0.99**
	EBM_S	0.49	0.51
	EBM	0.98	0.97
	XGBoost+TreeSHAP	1.00	0.78

4.2.2 Faithfulness of explanations

In this part, we try to answer the following question: *"Is it true that inherently interpretable models produce more reliable explanations?"* For this purpose, we consider the XAI solution, XGBoost (one of the best predictive model in Table 3), explained by the tree path-dependent SHAP algorithm (TreeSHAP [9]). As justified in Section 2.2.2, this approach, i.e., XGBoost+TreeSHAP, is arguably one of the most used solutions by practitioners. As inherently interpretable solutions, we consider all the listed models in Section 2.2.1 except Decision Tree (DT) as it does not produce direct feature attributions. For the datasets, we consider the additive *Synthetic benchmark* (Section 4.1.2) with 80/20% Train/Test split.

The example called *Synthetic 1* is linear regression friendly and only x_1 and x_2 are relevant. The example *Synthetic 2* represents a parabolic function; therefore linear regression cannot accurately model it unless identifying and adding the convenient quadratic terms of the raw input features. Only x_1 should have non-zero importance/attribution for this example. Finally, the *Synthetic 3* borrowed from (Amoukou et al. [10]) highlights interactions between the features. For this example, a perfect modeling algorithm should use, depending on the sign of x_5, the features x_1 and x_2 only, or alternatively, the features x_3 and x_4. We restrict our analysis to those data points with $x_5 \leq 0$, which comprise $\approx 3,300$ instances. Therefore, only x_1 and x_2 are relevant among (x_1, x_2, x_3, x_4). We use precision in finding the most relevant features as an evaluation metric, while the R^2 is used to assess the test's predictive performance.

Here are the main takeaways from Table 4:

- An inherently interpretable model with poor predictive performance or inductive bias can produce incorrect feature attributions or misleading discoveries. This is highlighted by the results of LR on *Synthetic 2* and EBM_S (EBM without interaction terms) on *Synthetic 3*.
- Using post hoc interpretability tools can lead to incorrect feature attribution, although the perfect predictive performance of the underlining model. It is the case of XGBoost+TreeSHAP on the *Synthetic 3*. This dataset highlights the feature interactions that are well-known situations in which most post hoc explanation tools struggle to find 'truly' relevant features of the underlying model (Amoukou et al. [10], Kumar et al. [11], Chen et al. [28], Huang and Marques-Silva [40]).

Overall, we can notice from these synthetic examples that an inherently interpretable model with "good" predictive performance/inductive bias usually produces reliable feature attribution. This is the case of TabSRALinear or EBM, which, with a predictive performance 2% lower than that of XGBoost on *Synthetic 3*, manages to discover the most important features with a precision of 97%, compared with 78% for XGboost. It is important to point out that for EBM, we move the value of pairwise interaction terms containing x_5 to the remaining main effect. In other words, we add for example the value of the interaction $f_{15}(x_1, x_5)$ to $f_1(x_1)$, producing the contribution of x_1. This is possible because we know the ground truth; otherwise, heuristic methods such as interaction purification (Lengerich et al. [43]) are used to solve identification ambiguity problems. Recall that TabSRALinear, thanks to its formulation, does not require any interaction purification.

4.2.3 Stability of explanations

We also evaluate the interpretable solutions (used in Section 4.2.2) based on the robustness/stability of explanations which involves answering the following question: *"Are the produced explanations similar for similar inputs?"* For this purpose, we use two real-world well-known numerical[4] datasets: Credit Card Fraud and Heloc Fico (Table 2). We consider the continuous notion of stability (Alvarez-Melis and Jaakkola [33]) and we estimate the local Lipschitz constant as follows:

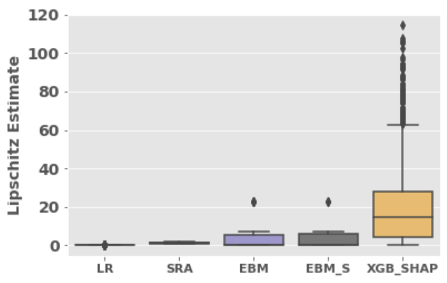
$$\hat{L}(x) = \arg\max_{\mathbf{x}' \in \mathcal{N}_\epsilon(\mathbf{x})} \| f_{expl}(\mathbf{x}) - f_{expl}(\mathbf{x}')\|_2 / \|\mathbf{x} - \mathbf{x}'\|_2 \tag{13}$$

where the vector $f_{expl}(\mathbf{x})$ is the feature attribution (the explanation) for the given observation $\mathbf{x}$. For linear models, $\hat{L}(\mathbf{x})$ is equivalent to $\|\boldsymbol{\beta}\|_\infty$ and for TabSRALinear it is proportional to $L_\mathbf{x}$ (Theorem 1).

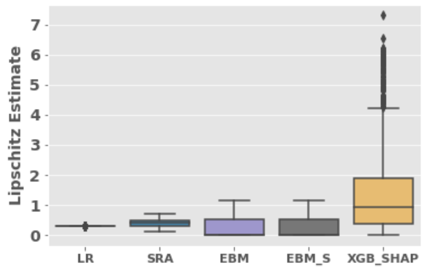
We use 80/20% Train/Test split and we generate 100 neighbors $\mathcal{N}_\epsilon(\mathbf{x})$ by adding a random Gaussian noise $\mathcal{N}(0, \epsilon \times I)$ to every target point $\mathbf{x}$ on the test data. The used perturbation is small enough to avoid excessively changing the predictions (Section B5). Thus, we use $\epsilon = 0.001$ for the Credit Card data and $\epsilon = 0.01$ for the Heloc Fico

[4]For the sake of brevity, we used datasets containing only numerical features, as the notion of neighborhood and Lipschtitz estimate is not trivial for categorical and discrete features. For more details, see Alvarez-Melis and Jaakkola [33]

dataset.



(a) Credit Card Fraud dataset:
2533 random test points



(b) Heloc Fico dataset:
1076 random test points

Fig. 5: Estimation of Lipschitz constant on real word datasets (the lower the better). LR = Logistic Regression, SRA=TabSRALinear, XGB_SHAP=XGBoost+TreeSHAP

We can see from Fig. 5 that the considered inherently interpretable models can produce more robust explanations than XGBoost+TreeSHAP. Due to their piecewise constant structure, Explainable Boosting Machines (EBMs) tend to produce usually similar explanations for similar inputs; however, they may also produce some outliers (see the example of the Credit Card Fraud dataset Fig. 5a) that arguably are due to some abrupt change points in the learned shape functions. Regarding XGBoost+TreeSHAP, the important variability in explanations can be either explained by the XGBoost's flexibility to capture local discontinuities or the incapacity of the interpretability tool to reproduce the correct local variability (Alvarez-Melis and Jaakkola [33]).

Apart from the Linear models (LR), which are known to be robust and conservative, TabSRALinear seems to be an encouraging tradeoff for robust interpretability, especially when adding predictive performance: test AUCROC for Heloc Fico (LR = 0.781; TabSRALinear= 0.795; EBM_S=0.795; EBM=0.798; XGboost= 0.798) and test AUCPR for Credit Card Fraud[5] (LR = 0.734; TabSRALinear= 0.844; EBM_S=0.831; EBM=0.834; XGboost=0.848).

4.3 Ablation study for TabSRALinear

To have more robust and intelligible explanations, we mentioned in Section 3.4 and Section A.1 that it is recommended to consider the size or number of ensemble/head $H \in \{1, 2\}$. In this section, we empirically show how varying some hyperparameters, such as the number of the ensemble, can impact the predictive performance of the TabSRALinear model. First, we provide some details on the implementation.

[5]This dataset is highly imbalanced therefore AUCPR is a more appropriate than AUCROC [39]

Table 5: Influence of the dimension of the query/key encoder d_k using the *Middle-scale benchmark*. H is set to 1. Each value of d_k is tuned with 17 random iterations.

d_k	Rank				Mean Test Score			Mean Running Time	
	min	max	mean	median	mean	median	std	mean	median
4	1	3	2.203	2	0.933	0.994	0.183	35.946	29.875
8	1	3	1.992	2	0.961	0.997	0.143	39.429	37.529
12	1	3	1.805	2	0.958	0.999	0.168	40.719	34.767

Table 6: Influence of the number of ensemble H using the *Middle-scale benchmark*. Each value of H is tuned with 30 random iterations.

H	Rank				Mean Test Score			Mean Running Time	
	min	max	mean	median	mean	median	std	mean	median
1	1	6	4.169	5	0.954	0.994	0.142	40.758	36.330
2	1	6	3.542	4	0.961	0.996	0.154	48.419	38.817
3	1	6	3.458	3	0.953	0.995	0.162	63.703	57.671
4	1	6	3.127	3	0.963	0.996	0.160	61.878	53.618
5	1	6	3.203	3	0.977	0.997	0.103	91.683	62.889
6	1	6	3.500	4	0.969	0.996	0.116	99.672	73.969

We use the same architecture for the key and query encoders (Section 3.1) which is M hidden layers fully connected neural network with ReLU activation. The dimension of $i-$th hidden layer is $m_i = pd_k//2^{(M-i)}$ for $i = 0, ..., M-1$ where $M \geq 1$, $d_k//2^M \geq 1$, p is the number of features and d_k the dimension of the query/key veectors (Section 3.1). With such implementation, the total number of learnable parameters in TabSRALinear is:

$$N = H\left[(p+1) + 2(2-(\frac{1}{2})^M)pd_k + \frac{1}{2}(M-1)p^2d_k + \frac{4}{3}(1-(\frac{1}{4})^M)p^2d_k^2\right] \qquad (14)$$

with H the number or size of the ensemble/head.

Dimension of the query/key encoder d_k. In Equation 14, we can notice that the number of parameters is quadratic in d_k. Therefore, we recommend not setting the value d_k too high (typically consider $d_k \in [4, 8, 12]$). In our experiments, $d_k = 8$ is a good default value (as shown in Table 5). Overall, it remains a tunable parameter depending on the case study.

Number of ensemble H. Increasing the size of the ensemble/head H leads to a rise in the number of parameters, consequently extending the processing time, as demonstrated in Table 6. However the change in the predictive performance does not change significant for the *Middle-scale benchmark*. As for trade-off between the running time and intelligibility, we recommend to optimize $H \in \{1, 2\}$.

Table 7: Statistics for the target feature (*Exited*) with respect to the feature *NumOfProducts* of the churn modeling data set: fraction (Frac), number (Nb).

NumOfProducts	Statistics		
	Frac. Exited (%)	Nb. Exited	Nb. of observations
1	28	1409	5084
2	8	348	4590
3	83	220	266
4	100	60	60

The number of hidden layers in query/key encoder M**.** So far, we have given the results of TabSRALinear with one hidden encoder $M = 1$, in order to avoid learning suspicious interactions.

We explore the predictive performance for $M = 2$ utilizing the *Middle-scale benchmark*. TabSRALinear, with $M = 1$, emerges victorious in 31 out of 59 tasks. This outcome is not surprising given our focus on middle-scale datasets. However, we anticipate that employing $M = 2$ could yield comparable or potentially superior performance for larger datasets characterized by potentially strong interactions.

4.4 Real world application of TabSRALinear

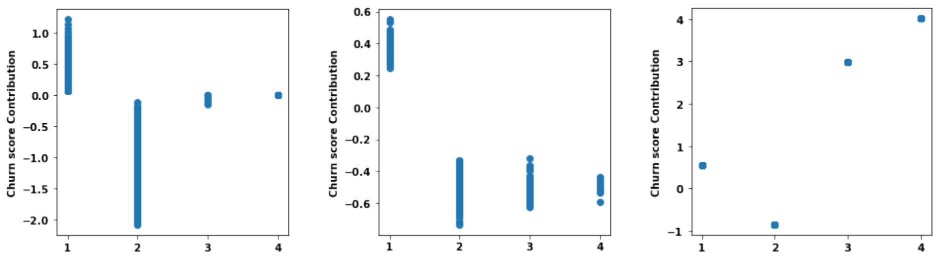
In this section, we show the value of the TabSRALinear inherently interpretable solution considering two real-world applications: bank churn modeling and credit default prediction (80/20% Train/Test split).

4.4.1 Application 1: Bank churn modeling

This first example illustrates how one can leverage TabSRALinear's knowledge incorporation capabilities more specifically the negative sense constraint (described in Section 3.2) to mitigate identified bias in data collection.
We consider the Bank churn modeling dataset where the objective is to predict if a customer is going to churn. For this type of problem, it is important to explain, or find out for customers of interest, the characteristics that may lead them to want to close their accounts. This enables customer services to accurately target the profiles to be contacted.

More importantly, there is one feature called *NumOfProducts*, which refers to the number of products purchased by a customer through the bank (Table 7). Intuitively, a customer with a high *NumOfProducts* is less likely to leave the bank. However, according to the exploration of the collected data of 10000 recordings (Table 7), 100% of customers with 4 products have left the bank. This probably indicates a bias in the data, especially since only 60/10000 observations have 4 products. Similarly, customers with 3 products (266/10000) have $\approx$ 83% as churn rate whereas for the majority of customers (with 1 and 2 products), the risk of churn decreases with the number of products.

Logistic Regression (LR) is in accordance with human intuition when producing a negative coefficient for the feature *NumOfProducts*. Nevertheless, its overall predictive



(a) TabSRALinear (b) XGBoost+TreeSHAP (c) EBMs

Fig. 6: Bank churn modeling: Effect contribution of the feature *NumOfProducts* ranging from 1 to 4

performance (i.e., test AUCROC) is 0.781 compared to 0.857 and 0.874 for EBM_S and EBM respectively. This performance gap is arguably due to nonlinear effects and interactions present in this data (see Section B.3.1 for more details). Although their test AUCROC is good, Explainable Boosting Machines (EBMs) exactly reproduce the bias in the data, as shown in Fig. 6c.

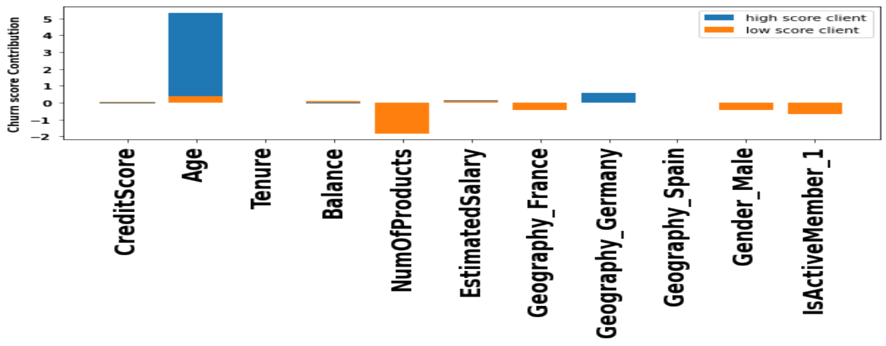
Regarding XGBoost, there is a monotonic option in which one can typically constrain the effect of a given feature to grow monotonically (decreasing). Doing so for XGBoost, as shown in Fig. 6b results in a drop of the test AUCROC from 0.872 to 0.821[6][7]

In this application, the uniqueness of TabSRALinear lies in its negative weight (sense) constraint (Section 3.2) which helps maintain a test AUCROC of 0.854 in the addition of the inherent intelligibility. That is, instead of using global monotonic constraint (as for LR and XGBoost), the negative weight constraint helps TabSRALinear to consider a decreasing effect for a *NumOfProducts* from 1 to 2. At the same time, this constraint forces the TabSRALinear to not directly use *NumOfProducts* to assign a high churn score for customers with more than 3 products, highlighted by a nearly zero effect contribution (Fig. 6a). This is achieved by producing an attention weight close to zero for *NumOfProducts*. For these customers, the churn risk, according to TabSRALinear, is usually due to their *Age* and the fact that their *Geography=Germany*, as shown in Fig. 7. These explanations are confirmed by our data exploration because the churn rate of customers in Germany is twice that of other countries, and customers with *Age* around 60 are very likely to churn, especially when they are not active members ($\approx 90\%$ of churn rate). We argue that this category includes customers who are close to or at retirement age and who are, therefore, likely to be looking for a bank that can advise them on tax, pension, and wealth management issues. So they are apt to change banks, especially when they are less active (interacting less with their

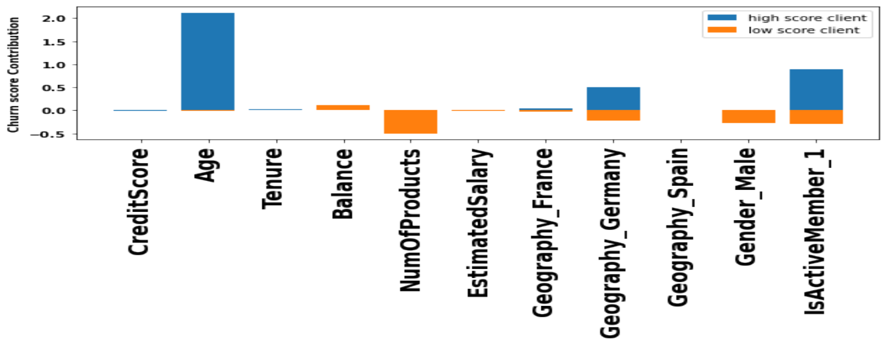
[6]One may want to process the *NumOfProducts* in binary feature, i.e., 1 product versus more than 1 product. This will help maintain the test AUCROC of XGBoost at approximately 0.828 for this dataset.

[7]The obtained explanations using TreeSHAP are not actually monotonic decreasing as they are supposed to be Fig. 6b. That is, the effect of the *NumOfProducts* for some customers with 3 products is higher than the one of some customers with 2 products.

current bank). Additional results on the interactions of the *Age* and *ActiveMember* indicators are provided in Section B.3.1.



(a) TabSRALinear



(b) XGBoost+TreeSHAP

Fig. 7: Individual prediction understanding for the bank churn modeling. We consider a high churn risk customer with *NumOfProducts=3* (meaning that *NumOfProducts* should not be a churn risk factor for this customer after the bias correction). Both TabSRALinear and XBoost+TreeSHAP indicates that *Age=63* and *Geography=Germany* are the two most important features, which confirms our data knowledge. On the other hand, the low churn risk costumer has 2 products, i.e.,*NumOfProducts=2*, is an active member (*IsActiveMember_1=1*) and is a male (the churn rate is 16% for *Gender=Male* and 25%, remaining).

4.4.2 Application 2: Credit Card Default prediction

Another criterion for evaluating model interpretability is the human-friendliness of explanations, meaning they should be concise and easily understood. In this section, we demonstrate that TabSRALinear provides concise explanations compared to existing solutions using the credit default dataset. This dataset predicts the default probability of credit card clients in Taiwan and includes a group of correlated features with:

- 6 related to the repayment status (ranging from -2, paid two months in advance, to 8, eight months overdue);
- 6 related to the bill statement amount;
- 6 related to the previous payment amount.

In the case of a high default risk client shown in Fig. 8, TabSRALinear focuses mainly on PAY_0, which represents the repayment status in the last month before the prediction. Our exploratory analysis confirms that PAY_0 is the most important risque factor for this dataset. Unlike TabSRALinear, XGBoost+TreeSHAP spreads the contribution among the set of correlated resulting in less sparse feature attribution (Amoukou et al. [10], Kumar et al. [11], Chen et al. [28]).

To confirm this observation, we illustrate (in Fig. 9) using all the test points (6000 precisely), the capacity of PAY_0 to separate high default score customers (in yellow). According to both TabSRALinear and XGBoost, the default risk becomes important from 2 months of delay in the repayment status PAY_0. In addition, TabSRALinear came up to isolate almost its high default score (in yellow) based on PAY_0 (with the simple rule contribution PAY_0$>$ 3). For XGBoost+TreeSHAP, this separation is less clear, demonstrating the need to add information or visualization regarding the remaining features (which are already correlated to PAY_0). We recall that test AUCROC is very close for these two models: 0.794 for XGBoost and 0.791 for TabSRALinear.

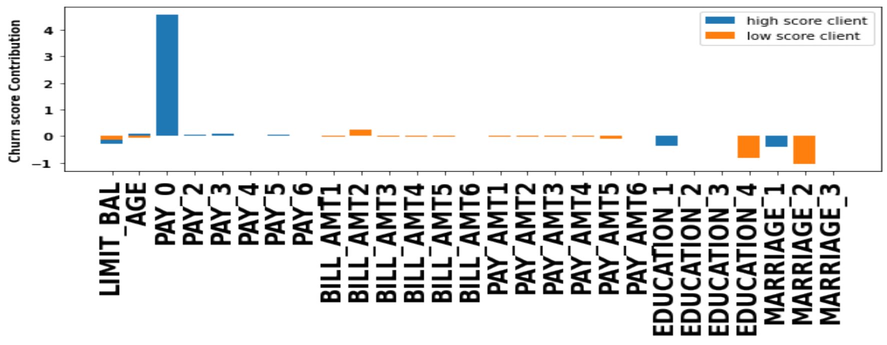
To produce concise explanations (feature attribution), TabSRALinear shows its superiority over other solutions such as XGBoost+TreeSHAP (the conclusion is the same for EBMs, Section B.3.2). We recall that the feature attribution of TabSRAlinear becomes very sparse for problems when one hot encoding is used for a categorical feature. That is, all reference features (with value 0) have exactly 0 effect contribution as for classic linear models.

We summarize the **main takeaways** from the case studies in two points:

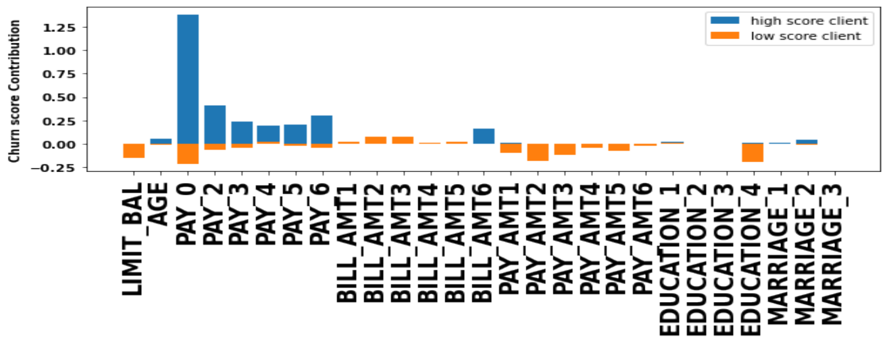
- Full-complexity models can provide superior predictive performance. However, after performing important model diagnostics (e.g., bias correction, adding monotonic constraints), their performance can drop significantly, even under an inherently interpretable solution that is more flexible for knowledge incorporation (such as TabSRALinear).
- When the sparsity of the explanations is an important consideration, it is worth nothing to test an inherently interpretable model such as TabSRALinear, especially for datasets with an important number of one-hot encoded features.

5 Limitations and Recommendations

In this study, we demonstrated through benchmarking that state-of-the-art inherently interpretable machine learning models generally achieve predictive performance very close to that of full-complexity counterparts (with less than 4% relative performance



(a) TabSRALinear

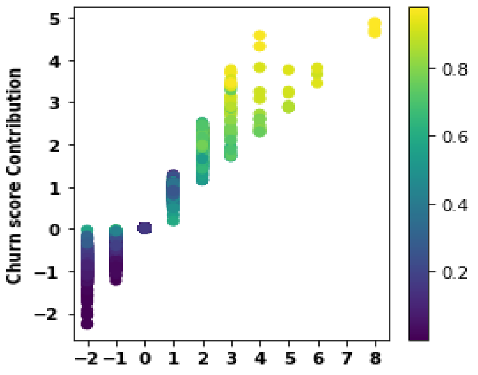


(b) XGBoost+TreeSHAP

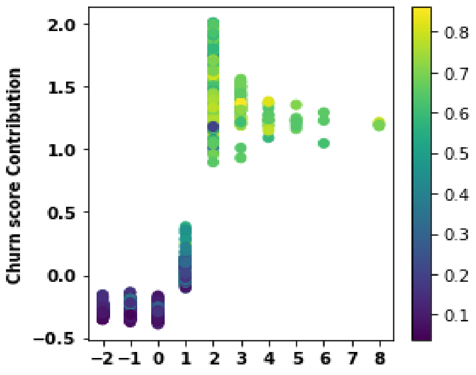
Fig. 8: Individual prediction understanding for the credit card default dataset.

gap on 59 tasks/45 datasets). However, to draw rigorous conclusions about the inductive biases of the algorithms, the datasets were selected to ensure no concept drift between the training and test data, balanced classes for binary classification problems, and a limited number of 10,000 observations for training data and 50,000 for test data following Grinsztajn et al. [16]. How would the inclusion of concept drift, class imbalance, and big data (datasets comprising millions of observations) impact results? With respect to the data size, our findings from the *Default benchmark* (Table 2), which includes up to 285,000 observations, and those from Chang et al. [15] (6 datasets with over 500,000), suggest that the performances are still comparable. However, further investigation is necessary for datasets that are (highly) imbalanced and problems involving concept drift, and we will address this in future work.

In addition, we listed some models in Section 2.2.1 as inherently interpretable solutions, but different regulators or stakeholders may have different requirements for the use case (the model must be very transparent, such as a small set of decision rules, linear models with few predictors or features). For certain classes of models, interpretability can be assessed based on the number of parameters or complexity. For instance, in statistical models like Logistic Regression, the model's interpretability



(a) TabSRALinear



(b) XGBoost+TreeSHAP

Fig. 9: From individual to global effect understanding for the PAY_0 feature of the credit card default dataset. On x-axis, we show the number of months of delay in the payment ranging from -2 to 8.

increases as the number of predictors or non-null parameters decreases. Similarly, in decision trees, interpretability can be assessed based on the depth or the number of nodes. However, in models such as the Explainable Boosting Machine (EBM_S) within Generalized Additive Models (GAMs), interpretability is not directly quantified by the number of trees or parameters (which can often be in the thousands), but also in terms of predictors, akin to Logistic Regression. This is because all these trees are aggregated into a one-dimensional piecewise constant function per predictor. Similarly, in the case of TabSRALinear, interpretability is not directly linked to the number of parameters or weights, but rather to the predictors themselves. Therefore, merely counting the number of nodes or weights does not necessarily enhance interpretability (e.g., for feature attribution), but rather serves the purpose of transparency. Overall, we recommend that interested readers/practitioners do not consider the term "inherently interpretable" as generic, but instead define it based on the specific use cases and requirements.

In general, it is challenging, if not impossible, to find a single solution that offers optimal performance across all metrics, including predictive performance, faithfulness, stability, and knowledge incorporation, irrespective of the dataset or problem at hand (a concept known as the "no free lunch theorem"). In some scenarios, accuracy may be the primary concern, while in others, interpretability takes precedence. For tabular learning tasks, where understanding the predictions or discovering hidden patterns is an important consideration, we recommend practitioners to begin by using both full-complexity models (typically tree ensembles, which, in addition to performance, are not greedy in terms of computational cost and can be easily explained using TreeSHAP [9]) and inherently interpretable solutions (typically GAMs: piecewise constant approach such as EBMs (Nori et al. [13]), piecewise linear approach such as GAMI-Net (Yang et al. [29]), and attention based solution such TabSRALinear). Depending on the available computational resources, one could begin by using default hyperparameters or a fixed set of hyperparameter configurations. One may opt to keep only the inherently interpretable models if their predictive performances are *comparable* to those of the full-complexity model under consideration, and discard the latter. Otherwise, it may be advisable to also explain the full-complexity model using post hoc tools. After conducting any necessary model diagnosis (e.g., bias correction), the final model can be chosen based on the number of explanations that align with the data knowledge, after taking advantage of the Rashomon effect (Müller et al. [44]) (where different "good" predictive models can offer different explanations[8]). Additionally, it is important to note that inherently interpretable solutions generally do not require additional computational cost to produce explanations.

Regarding TabSRALinear, we recommend using it with a good feature selection process for the following reasons: (i) As NNs based model, it is more sensitive to uninformative features compared to decision tree-based or piecewise constant approximators (Grinsztajn et al. [16]). (ii) with the current implementation, the number of learned parameters is quadratic with respect to the number of the input features (for more details, please refer to Section 4.3).

6 Conclusion

In many tabular learning use cases, researchers and practitioners justify the choice of full-complexity models based on their superior predictive performance over classical statistical models (e.g., linear models). The latter, on the other hand, are favored for their transparency, which usually leads to a dilemma when it comes to choosing a model for many real-life use cases. In this study, we investigated the consideration of machine learning (ML) based inherently interpretable models (IIM). Through a thorough benchmarking, we showed that IIM such as Explainable Boosting Machine, can produce a predictive performance that is usually very close to full-complexity ML models. Second, we proposed TabSRALinear, an attention based IML that demonstrates its superiority when the robustness of explanations and human knowledge incorporation are key considerations. The code and results are open-sourced[9] and we hope this work will encourage researchers and practitioners to conduct further investigations.

Supplementary information. All data supporting the findings of this study are available within the paper and its Supplementary Information.

Conflict of Interest. The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[8] In general, these explanations do not represent causality, but rather a potential description of the association between the observed feature and the target variable

[9] https://github.com/anselmeamekoe/TabSRA

[2] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)

[3] Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C.B., Goldstein, T.: Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342 (2021)

[4] Kossen, J., Band, N., Lyle, C., Gomez, A.N., Rainforth, T., Gal, Y.: Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. Advances in Neural Information Processing Systems **34**, 28742–28756 (2021)

[5] Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z.: Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020)

[6] Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. Advances in Neural Information Processing Systems **34**, 18932–18943 (2021)

[7] Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)

[8] Ribeiro, M.T., Singh, S., Guestrin, C.: " 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, pp. 1135–1144 (2016)

[9] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.-I.: From local explanations to global understanding with explainable ai for trees. Nature machine intelligence **2**(1), 56–67 (2020)

[10] Amoukou, S.I., Salaün, T., Brunel, N.: Accurate shapley values for explaining tree-based models. In: International Conference on Artificial Intelligence and Statistics, pp. 2448–2465 (2022). PMLR

[11] Kumar, I.E., Venkatasubramanian, S., Scheidegger, C., Friedler, S.: Problems with shapley-value-based explanations as feature importance measures. In: International Conference on Machine Learning, pp. 5491–5500 (2020). PMLR

[12] Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence **1**(5), 206–215 (2019)

[13] Nori, H., Jenkins, S., Koch, P., Caruana, R.: Interpretml: A unified framework for machine learning interpretability. arXiv preprint arXiv:1909.09223 (2019)

[14] Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., Hinton, G.E.: Neural additive models: Interpretable machine learning with neural nets. Advances in Neural Information Processing Systems **34**, 4699–4711 (2021)

[15] Chang, C.-H., Caruana, R., Goldenberg, A.: Node-gam: Neural generalized additive model for interpretable deep learning. arXiv preprint arXiv:2106.01613 (2021)

[16] Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems **35**, 507–520 (2022)

[17] Amekoe, K.M., Azzag, H., Lebbah, M., Dagdia, Z.C., Jaffre, G.: A new class of intelligible models for tabular learning. In: In The 5th International Workshop on eXplainable Knowledge Discovery in Data Mining (PKDD)-ECML-PKDD (2023)

[18] Amekoe, K.M., Dilmi, M.D., Azzag, H., Dagdia, Z.C., Lebbah, M., Jaffre, G.: Tabsra: An attention based self-explainable model for tabular learning. In: The 31th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) (2023)

[19] Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

[20] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.-Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30** (2017)

[21] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. Advances in neural information processing systems **31** (2018)

[22] Chen, K.-Y., Chiang, P.-H., Chou, H.-R., Chen, T.-W., Chang, T.-H.: Trompt: Towards a better deep neural network for tabular data. arXiv preprint arXiv:2305.18446 (2023)

[23] Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G.: Deep neural networks and tabular data: A survey. arXiv preprint arXiv:2110.01889 (2021)

[24] McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., White, C.: When do neural nets outperform boosted trees on tabular data? arXiv e-prints, 2305 (2023)

[25] Huang, X., Marques-Silva, J.: The inadequacy of shapley values for explainability. arXiv preprint arXiv:2302.08160 (2023)

[26] Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic

explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

[27] Marques-Silva, J., Ignatiev, A.: Delivering trustworthy ai through formal xai. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 12342–12350 (2022)

[28] Chen, H., Covert, I.C., Lundberg, S.M., Lee, S.-I.: Algorithms to estimate shapley value feature attributions. Nature Machine Intelligence, 1–12 (2023)

[29] Yang, Z., Zhang, A., Sudjianto, A.: Gami-net: An explainable neural network based on generalized additive models with structured interactions. Pattern Recognition **120**, 108192 (2021)

[30] Lou, Y., Caruana, R., Gehrke, J., Hooker, G.: Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 623–631 (2013)

[31] Popov, S., Morozov, S., Babenko, A.: Neural oblivious decision ensembles for deep learning on tabular data. arXiv preprint arXiv:1909.06312 (2019)

[32] Chen, Z., Tan, S., Nori, H., Inkpen, K., Lou, Y., Caruana, R.: Using explainable boosting machines (ebms) to detect common flaws in data. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 534–551 (2021). Springer

[33] Alvarez-Melis, D., Jaakkola, T.S.: On the robustness of interpretability methods. arXiv preprint arXiv:1806.08049 (2018)

[34] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., et al.: Api design for machine learning software: experiences from the scikit-learn project. arXiv preprint arXiv:1309.0238 (2013)

[35] Ignatiev, A., Izza, Y., Stuckey, P.J., Marques-Silva, J.: Using maxsat for efficient explanations of tree ensembles. In: AAAI (2022)

[36] Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: International Conference on Machine Learning, pp. 3145–3153 (2017). PMLR

[37] Alvarez Melis, D., Jaakkola, T.: Towards robust interpretability with self-explaining neural networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc., ??? (2018)

[38] Agarwal, C., Johnson, N., Pawelczyk, M., Krishna, S., Saxena, E., Zitnik, M.,

Lakkaraju, H.: Rethinking Stability for Attribution-based Explanations (2022)

[39] Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 233–240 (2006)

[40] Huang, X., Marques-Silva, J.: The inadequacy of shapley values for explainability. arXiv preprint arXiv:2302.08160 (2023)

[41] Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Learning hyperparameter optimization initializations. In: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10 (2015). IEEE

[42] Gorishniy, Y., Rubachev, I., Babenko, A.: On embeddings for numerical features in tabular deep learning. Advances in Neural Information Processing Systems **35**, 24991–25004 (2022)

[43] Lengerich, B., Tan, S., Chang, C.-H., Hooker, G., Caruana, R.: Purifying interaction effects with the functional anova: An efficient algorithm for recovering identifiable additive models. In: International Conference on Artificial Intelligence and Statistics, pp. 2402–2412 (2020). PMLR

[44] Müller, S., Toborek, V., Beckh, K., Jakobs, M., Bauckhage, C., Welke, P.: An Empirical Evaluation of the Rashomon Effect in Explainable Machine Learning (2023)

[45] Kim, H., Papamakarios, G., Mnih, A.: The lipschitz constant of self-attention. In: International Conference on Machine Learning, pp. 5562–5571 (2021). PMLR

[46] Ultsch, A.: Clustering wih som: U* c. Proc. Workshop on Self-Organizing Maps (2005)

[47] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)

[48] Biewald, L.: Experiment Tracking with Weights and Biases. Software available from wandb.com (2020). https://www.wandb.com/

Appendix A Additional theoretical results

A.1 On the Lipschitz estimate of TabSRALinear ensemble

Theorem 2. *The feature attributions produced by TabSRALinear ensemble (Equation 9) are locally stable in the sense of Lipschitz, that is, for all* $\mathbf{x} \in \mathbb{R}^p$, *there exists* $\delta > 0$ *and* $L_{\mathbf{x}} \geq 0$ *finite such that:*

$$\|\mathbf{x} - \mathbf{x}'\|_1 < \delta \implies \|\sum_{h=1}^{H} \boldsymbol{\beta}^h \odot a^h(\mathbf{x}) \odot \mathbf{x} - \sum_{h=1}^{H} \boldsymbol{\beta}^h \odot a^h(\mathbf{x}') \odot \mathbf{x}'\|_1 \leq L_{\mathbf{x}} \|\mathbf{x} - \mathbf{x}'\|_1 \tag{A1}$$

With $L_{\mathbf{x}} = \sum_{h=1}^{H} \|\boldsymbol{\beta}^h\|_\infty \left[\|\mathbf{a}^h\|_\infty + L_{\mathbf{a}}^h(\|\mathbf{x}\|_\infty + \delta)\right]$ *and* $L_{\mathbf{a}}^h \geq 0$ *the Lipschitz constant of the* h*-th SRA block.*

Theorem 2 demonstrates that the Lipschitz estimate of the TabSRALinear ensemble is theoretically additive with respect to the number of learners in the ensemble, denoted H. Hence, a large value of H may lead to less stable explanations. Our experimental results on *Default benchmark* and the ablation study (Table 6) demonstrate that $H = 2$ is typically sufficient to achieve strong predictive performance.
The proof of the Theorem 2 can be readily accomplished by using the fact that: (i) the feature attributions of each individual TabSRALinear in the ensemble are locally stable in the sense of Lipschitz in accordance with Theorem 1. (ii) A finite sum of Lipschitz continuous functions is Lipschitz continuous, as stated in Lemma 3).

A.2 Proof Theorem 1.

Before providing a proof of the theorem, we consider the following lemmas:
Lemma 3. *(i) The sum of two Lipschitz continuous functions is Lipschitz continuous (ii) The product of two Lipschitz continuous and bounded functions is Lipschitz continuous*

Proof. We consider θ and ψ two functions from $\mathbb{R}^p \longrightarrow \mathbb{R}^p$, L_θ and $L_\psi-$ Lipschitz. For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$ we have:
(i)

$$\begin{aligned} \|(\theta + \psi)(\mathbf{x}) - (\theta + \psi)(\mathbf{x}')\|_1 &= \|(\theta(\mathbf{x}) - \theta(\mathbf{x}')) + (\psi(\mathbf{x}) - \psi(\mathbf{x}'))\|_1 \\ &\leq \|\theta(\mathbf{x}) - \theta(\mathbf{x}')\|_1 + \|\psi(\mathbf{x}) - \psi(\mathbf{x}')\|_1 \quad \text{Minkowski's inequality} \\ &\leq L_\theta \|\mathbf{x} - \mathbf{x}'\|_1 + L_\psi \|\mathbf{x} - \mathbf{x}'\|_1 \\ &= (L_\theta + L_\psi)\|\mathbf{x} - \mathbf{x}'\|_1 \end{aligned} \tag{A2}$$

(ii) Moreover let's assume that θ and ψ are bounded ie $\|\theta\|_\infty < \infty$ and $\|\psi\|_\infty < \infty$, then we have:

$$\begin{aligned}
\|(\theta \odot \psi)(\mathbf{x}) - (\theta \odot \psi)(\mathbf{x}')\|_1 &= \|\theta(\mathbf{x}) \odot (\psi(\mathbf{x}) - \psi(\mathbf{x}')) + \psi(\mathbf{x}') \odot ((\theta(\mathbf{x}) - \theta(\mathbf{x}'))\|_1 \\
&\leq \|\theta(\mathbf{x}) \odot (\psi(\mathbf{x}) - \psi(\mathbf{x}'))\|_1 + \|\psi(\mathbf{x}') \odot ((\theta(\mathbf{x}) - \theta(\mathbf{x}'))\|_1 \\
&\text{using Hölder's inequality:} \\
&\leq \|\theta(\mathbf{x})\|_\infty \|\psi(\mathbf{x}) - \psi(x')\|_1 + \|\psi(\mathbf{x})\|_\infty \|\theta(\mathbf{x}) - \theta(\mathbf{x}')\|_1 \\
&\leq L_\psi \|\theta(\mathbf{x})\|_\infty \|\mathbf{x} - \mathbf{x}'\|_1 + L_\theta \|\psi(\mathbf{x})\|_\infty \|\mathbf{x} - \mathbf{x}'\|_1 \\
&= (L_\theta \|\psi(\mathbf{x})\|_\infty + L_\psi \|\theta(\mathbf{x})\|_\infty) \|\mathbf{x} - \mathbf{x}'\|_1 \\
&\leq (L_\theta \|\psi\|_\infty + L_\psi \|\theta\|_\infty) \|\mathbf{x} - \mathbf{x}'\|_1
\end{aligned} \tag{A3}$$

□

Lemma 4. *The attention vector outputted using the SRA block is stable is sense of Lipschitz i.e., for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$, there exists a constant $L_{\mathbf{a}} \geq 0$ finite such that:*

$$\|a(\mathbf{x}) - a(\mathbf{x}')\|_1 \leq L_{\mathbf{a}} \|\mathbf{x} - \mathbf{x}'\|_1 \tag{A4}$$

Moreover $\|a\|_\infty = 1$.

Proof. Each component k_i^j of the keys matrix K (resp. q_i^j of Q) is stable in the sense of Lipschtiz as outputted by a fully connected layer [45] (linear transformations followed by common activation such as ReLU, Sigmoid) and is bounded in $[0, 1]$ using Sigmoid activation. Hence the for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$ there exists $\alpha_{k_i^j} > 0$, $\alpha_{q_i^j} > 0$, finite such that $|k_i^j(\mathbf{x}) - k_i^j(\mathbf{x}')| = \alpha_{k_i^j} \|\mathbf{x} - \mathbf{x}'\|_1$ and $|q_i^j(\mathbf{x}) - q_i^j(\mathbf{x}')| = \alpha_{q_i^j} \|\mathbf{x} - \mathbf{x}'\|_1$.
Then each component of the attention vector is also Lipschitz since:

$$\begin{aligned}
|a_i(\mathbf{x}) - a_i(\mathbf{x}')| &= |\frac{1}{d_k}(\sum_{j=1}^{d_k} k_i^j(\mathbf{x}) q_i^j(\mathbf{x}) - \sum_{j=1}^{d_k} k_i^j(\mathbf{x}') q_i^j(\mathbf{x}'))| \\
&= |\frac{1}{d_k} \sum_{j=1}^{d_k} (k_i^j(\mathbf{x}) q_i^j(\mathbf{x}) - k_i^j(\mathbf{x}') q_i^j(\mathbf{x}'))| \\
&\leq \frac{1}{d_k} \sum_{j=1}^{d_k} \alpha_i^j \|\mathbf{x} - \mathbf{x}'\|_1 \quad \text{product and sum of Lipsctiz function} \\
&= L_{a_i} \|\mathbf{x} - \mathbf{x}'\|_1
\end{aligned} \tag{A5}$$

with $L_{a_i} = \frac{1}{d_k}\sum_{j=1}^{d_k} \alpha_i^j$.
Finally we have:

$$\begin{aligned}\|a(\mathbf{x}) - a(\mathbf{x}')\|_1 &= \sum_{i=1}^{p} |a_i(\mathbf{x}) - a_i(\mathbf{x}')| \\ &\leq \sum_{i=1}^{p} L_{a_i}\|\mathbf{x}-\mathbf{x}'\|_1 \quad \text{using the Equation A5} \\ &= L_{\mathbf{a}}\|\mathbf{x}-\mathbf{x}'\|_1\end{aligned} \tag{A6}$$

with $L_{\mathbf{a}} = \sum_{i=1}^{p} L_{a_i}$.
Since every $a_i \in [0,1]$ we have $\|\mathbf{a}\|_\infty = 1$. □

Proof. of Theorem 1

$$\begin{aligned}|\boldsymbol{\beta}\cdot(a(\mathbf{x})\odot\mathbf{x}) - \boldsymbol{\beta}\cdot(a(\mathbf{x}')\odot\mathbf{x}')| &= |\boldsymbol{\beta}\cdot(a(\mathbf{x})\odot\mathbf{x} - a(\mathbf{x}')\odot\mathbf{x}')| \\ &\leq \|\boldsymbol{\beta}\|_\infty \|a(\mathbf{x})\odot\mathbf{x} - a(\mathbf{x}')\odot\mathbf{x}'\|_1\end{aligned} \tag{A7}$$

Using A3 and considering $\psi(\mathbf{x}) = a(\mathbf{x})$, $\theta(\mathbf{x}) = \mathbf{x}$ we have $\|\psi\|_\infty = \|a\|_\infty = 1$ and $\|\theta\|_\infty = \|\mathcal{D}\|_\infty = \max_{\mathbf{x}\in\mathcal{D}}\|\mathbf{x}\|_\infty$ which is the overall maximal observable feature value. Therefore:

$$|\boldsymbol{\beta}\cdot(a(\mathbf{x})\odot\mathbf{x}) - \boldsymbol{\beta}\cdot(a(\mathbf{x}')\odot\mathbf{x}')| \leq \|\boldsymbol{\beta}\|_\infty(\|\mathbf{a}\|_\infty + L_{\mathbf{a}}\|\mathcal{D}\|_\infty)\|\mathbf{x}-\mathbf{x}'\|_1 \tag{A8}$$

In the formulation of the TabSRALinear, we are not necessarily interested in global stability or a uniform Lipschitz constant, as in Equation A8 but rather in regional or local stability around a given target or anchor data point.
With this consideration, given the target data point $\mathbf{x}$, we can restrict $\mathcal{D}$ to its neighborhood i.e.,

$$\mathcal{D} = \{\mathbf{x}' \in \mathbb{R}^p / \|\mathbf{x}-\mathbf{x}'\|_1 < \delta\} \tag{A9}$$

therefore $\|\mathcal{D}\|_\infty \leq \|\mathbf{x}\|_\infty + \delta$.
Using the Equation A8, it results that for every $\mathbf{x} \in \mathbb{R}^p$, there exists a constant $\delta > 0$ such that $\|\mathbf{x}-\mathbf{x}'\|_1 < \delta$ implies:

$$|\boldsymbol{\beta}\cdot(a(\mathbf{x})\odot\mathbf{x}) - \boldsymbol{\beta}\cdot(a(\mathbf{x}')\odot\mathbf{x}')| \leq \|\boldsymbol{\beta}\|_\infty(1 + L_{\mathbf{a}}(\|\mathbf{x}\|_\infty + \delta))\,\|\mathbf{x}-\mathbf{x}'\|_1 \tag{A10}$$

□

Appendix B Additional empirical informations

B.1 Datasets

B.1.1 Middle-scale benchmark

This benchmark of 45 datasets (59 binary classification and regression and tasks) is introduced in a paper titled: *Why do tree-based models still outperform deep learning on typical tabular data?* (Grinsztajn et al. [16]). The main goal of this benchmark was to identify certain meta-features or inductive biases that explain the superior predictive performance of tree-based models over NNs in tabular learning. We take a step forward by incorporating inherently interpretable models in the assessment the predictive performance. We provide essential details about datasets and refer the interested reader to the original paper (Grinsztajn et al. [16]) for further information. The main criteria for selecting datasets are:

- The datasets contain heterogeneous features (this excludes images and signal datasets).
- The datasets are not high dimensional. That is, the ratio a p/n is below 1/10, $p < 500$ with p the number of features and n the number of observations.
- The data are I.I.D. Stream-like datasets or time series are removed.
- They are real-world data.
- The datasets are not too easy and not too small, i.e., $p \geq 4$ and $n \geq 3000$.
- They are not deterministic datasets. Datasets where the target is a deterministic function of the predictors of the predictors.

Furthermore, in order to keep the learning as homogeneous as possible, some subproblems that deserve their own analysis are excluded. That is:

- The size of the datasets. In the *Middle-scale* regime, the training set is truncated to 10,000 and the test set to 50,000.
- There is no missing data. All data points with missing values are removed.
- Balanced classes. For classification, the target is binarised if there are several classes by taking the two most numerous classes, and we keep half of the samples in each class.
- Categorical features with more than 20 items are removed.
- Numerical features with less than 10 unique values are removed.

Finally, every algorithm and hyperparameters combination is evaluated on the same random seed **Train/Validation/Test** split or fold. More precisely, 70% of samples for the train set (or the percentage which corresponds to a maximum of 10,000 samples if 70% is too high). Of the remaining 30%, 30% are used for the validation set (truncated to a maximum of 50,000 samples), and 70% for the test set (also truncated to a maximum of 50,000 samples). Depending on the size of the test set, several random seed splits/folds are used (cross-validation). That is:

- If the test set is more than 6000 samples, we evaluate our algorithms on 1 fold.
- If the test set is between 3000 and 6000 samples, we evaluate our algorithms in 2 folds.

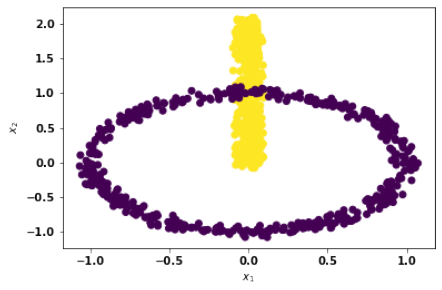
- If the test set is 1000 and 3000 samples, we evaluate our algorithms on 3 folds.
- If the test set is less than 1000 testing samples, we evaluate our algorithms on 5 folds.

B.1.2 Default benchmark

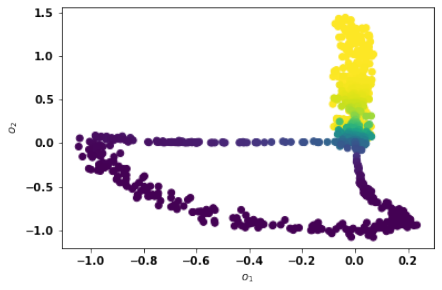
- **Bank Churn**. This dataset contains details of a bank's customers, and the target variable is a binary variable reflecting the fact whether the customer left the bank (closed his account) or continues to be a customer. We drop the CustomerId and Surname columns. We also drop the HasCrCard column, which is non really informative https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling.
- **Credit Default**. The goal is to predict the default probability (for the next month) of credit card clients in Taiwan using historical data. https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients. We drop the SEX column as generally, the gender information is not used in credit scoring.
- **Credit Card Fraud**. For this dataset, the aim is to predict whether credit card transactions are fraudulent or genuine. The original dataset is PCA transformed for privacy purpose. We drop the time information in our study. https://www.kaggle.com/mlg-ulb/creditcardfraud
- **Heloc Fico**. For this dataset, the target variable to predict is a binary variable called RiskPerformance. The value "Bad" indicates that a consumer was 90 days past due or worse at least once over a period of 24 months from when the credit account was opened. The value "Good" indicates that they have made their payments without ever being more than 90 days overdue. https://community.fico.com/s/explainable-machine-learning-challenge.

B.2 Additional results for TabSRAs: Vizualisation

How raw data are reinforced using the SRA block.
In addition to Section 3.1, we included two more examples, the 2D chainLink (Ultsch [46]), the Noisy two moon as depicted in Fig. B1 and Fig. B2. By applying SRA coefficients to this dataset, we acquired a new data representation that enables the easy separation of classes, as shown in Fig. B1b. Even without knowledge of the true data-generating process, it is apparent that all observations have been moved strategically so that a simple rule can effectively isolate nearly all yellow observations of interest.

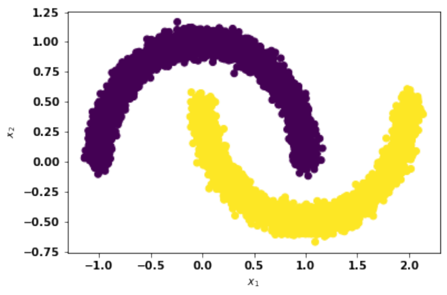


(a) $\mathbf{x}$

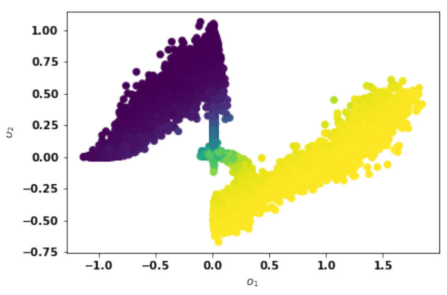


(b) $\mathbf{o} = \mathbf{a} \odot \mathbf{x}$

Fig. B1: ChainLink 2D: 1000 data points



(a) $\mathbf{x}$



(b) $\mathbf{o} = \mathbf{a} \odot \mathbf{x}$

Fig. B2: Noisy two moons: 10000 data points

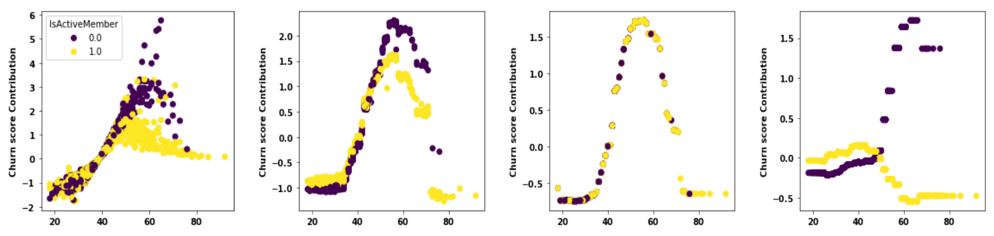
B.3 Additional results on the applicative case studies

B.3.1 Bank churn modeling

As shown in Fig. B3, the effect of the *Age* feature on the churn risk is bell-shaped according to TabSRALinear, XGBoost+TreeSHAP, and EBMs. Moreover there is a strong interaction between the *Age* and *IsActiveMember* feature highlighted par the important influence on the churn score of the *Age* (around 60) for non active member. To handle this interaction, the GAMs based inherently interpretable solution EBM needed to break it down into main effect and interaction effect, which is not necessary with TabSRALinear. Overall, these findings explain the poor predictive performance of the Logistic Regression (LR) as highlighted in Section 4.4.1.

B.3.2 Credit Card Default

In Section 4.4.2, we demonstrate using the credit card default dataset that TabSRALinear can generate more concise explanations compare to XGBoost+TreeSHAP. As



(a) TabSRALinear (b) XGBoost (c) EBMs (d) EBM

Fig. B3: Bank churn modeling: interaction between the *Age* and *IsActiveMember* feature. (B3b): XGBoost refers to the XGBoost+TreeSHAP feature attribution solution. (B3c): Indicates the main effect of the *Age* on the churn score for both EBM_S , EBM and (B3d) show the pairwise interaction for the latter.

depicted in Fig. B4, EBMs also spread contribution among correlated features. As a result, their feature attributions are less sparse compared to those of TabSRALinear, particularly with EBMs having pairwise interactions, which may assign non-zero feature attribution to interaction terms as well as main effects.

B.4 Additional results for the robustness study

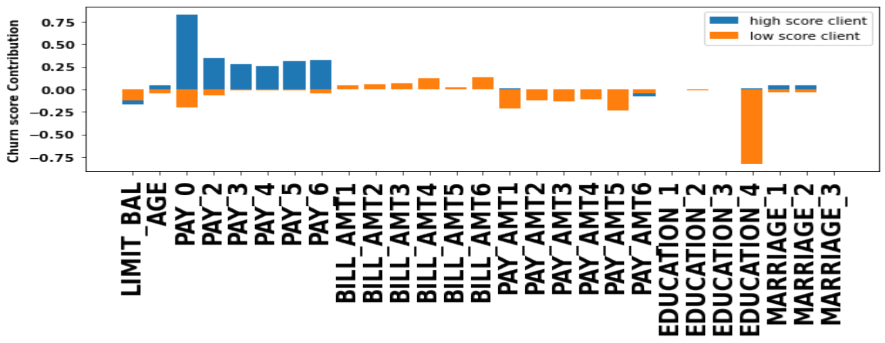
The output of piecewise constant approximators (for instance EBMs and XGBoost) are generally more sensitive to input perturbation compared to Linear models (LR) and TabSRALinear, as depicted in Fig B5. We argue that this is due to their flexibility in producing discontinuities or learning irregular functions (Grinsztajn et al. [16], McElfresh et al. [24]).

B.5 Implementation details for the predictive performance evaluation

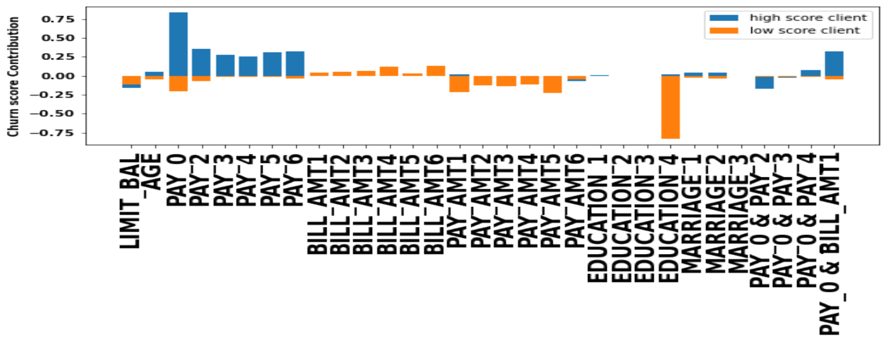
B.5.1 Search space for hyperparameters

For full-complexity models, the hyperparameter spaces are derived from the previous study [16]. For tree-based models, the number of estimators (trees) is not tuned but rather set to a high value: 250 for Random Forest (RF), 1000 for XGBoost, and CatBoost. For the latter, early stopping is employed with patience 20. Default parameters for tree-based models are ScikitLearn/XGBoost/CatBoost's defaults.

For Neural Nets (NNs), the maximal number of epochs is set to 300 with early stopping and checkpoint (the best model on the validation set is kept). The early stopping round is 40 for MLP, ResNet, TabSRALinear, Linear, 10 for SAINT and 50 for EBMs. Note that for the model using early stopping, 20% of the training dataset is used as validation set (different from the validation which is in the test part).
For NNs, we used the Pytorch library [47], the Weights and Biases platform [48] for hyperparameters optimization and experiment tracking.



(a) EBM_S



(b) EBM

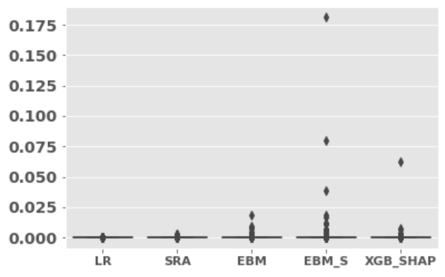
Fig. B4: Individual prediction understanding for the credit card default dataset.

Table B1: Decision Tree (DT)

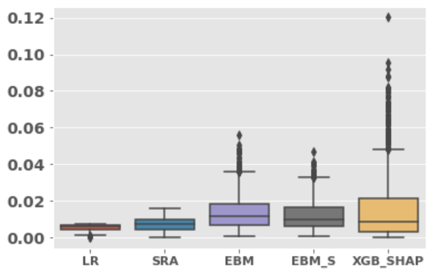
Parameter	Distribution	Default
Max depth	[2, 3, 4, 5, 6, 7]	-
Min sample split	[2, 3]	-
Min samples leaf	LogUniformInt [1.5, 50.5]	-

Table B2: Linear Models (LR)

Parameter	Distribution	Default
Learning rate	LogUniform [1e-5,1e-2]	-
Weight decay	LogUniform [1e-6,1e-4]	-
Learning rate scheduler	[True,False]	-
Use bias term	[True]	-
Batch size	[128, 256, 512, 1024]	-



(a) Credit Card Fraud dataset:
2533 random test points



(b) Heloc Fico dataset:
1076 random test points

Fig. B5: Change in predictions using input perturbationes (Section 4.2.3). LR = Logistic Regression, SRA=TabSRALinear, XGB_SHAP=XGBoost+TreeSHAP

Table B3: TabSRALinear

Parameter	Distribution	Default
Learning rate	LogUniform [1e-5,1e-2]	-
Weight decay	LogUniform [1e-6,1e-4]	-
Learning rate scheduler	[True,False]	-
Use bias term encoders	[True]	-
Use bias term classifier	[True]	-
Dropout	Uniform [0,0.5]	-
N Head	[1,2]	-
Dim Head	[4,8, 12]	-
Batch size	[128, 256, 512, 1024]	-

Table B4: EBM_S

Parameter	Distribution	Default
Learning rate	LogUniform [1e-4,0.7]	-
Max bins	[128,256, 512]	-
Number of interaction terms	[0]	-
Min samples lead	IntUniform [1,100]	-
Max rounds	[20000]	-
Inner bags	[0, 5, 10, 15]	-
Outer bags	[8, 16, 32, 64, 128]	-

B.6 Additional results about the predictive performance

B.6.1 Results of hyperpameters optimization using all iterations

In this part, we present the results of hyperparameter tuning using all iterations instead of the use bootstrapping as decribed in Section 4.1.3.

As indicated in Table B13, utilizing all iterations once instead of employing bootstrapping does not change the predictive ranking of algorithms and our conclusions

Table B5: EBM

Parameter	Distribution	Default
Learning rate	LogUniform [1e-4,0.7]	-
Max bins	[128,256, 512]	-
Number of interaction terms	[0, 5, 10, 15, 20, 25]	-
Min samples lead	IntUniform [1,100]	-
Max rounds	[20000]	-
Inner bags	[0, 5, 10, 15]	-
Outer bags	[8, 16, 32, 64, 128]	-

Table B6: Random Forest (RF)

Parameter	Distribution	Default
Max depth	[None, 2, 3, 4] ([0.7, 0.1, 0.1, 0.1])	-
Num estimators	250	
Criterion	[gini, entropy] (classif) [squared_error, absolute_error] (regression)	-
Max features	[sqrt, sqrt, log2, None, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]	-
Min samples split	[2, 3] ([0.95, 0.05]	-
Min samples leaf	LogUniformInt[1.5, 50.5]	-
Boostrap	[True, False]	-
Min impurity decrease	[0.0, 0.01, 0.02, 0.05] ([0.85, 0.05, 0.05, 0.05])	-

Table B7: XGBoost

Parameter	Distribution	Default
Max depth	UniformInt[1,11]	-
Num estimators	1000	-
Min child weight	LogUniformInt[1, 1e2]	-
Subsample	Uniform[0.5,1]	-
Learning rate	LogUniform[1e-5,0.7]	-
Col sample by level	Uniform[0.5,1]	-
Col sample by tree	Uniform[0.5, 1]	-
Gamma	LogUniform[1e-8,7]	-
Lambda	LogUniform[1,4]	-
Alpha	LogUniform[1e-8,1e2]	-

Table B8: CatBoost

Parameter	Distribution	Default
Max depth	UniformInt[3,11]	-
Num estimators	1000	-
Learning rate	LogUniform[1e-5,0.7]	-
L2 Leaf Reg	LogUniform[1,10]	-

(Section B.6.1).

Table B9: MLP

Num layers	UniformInt [1, 8]	4
Layer size	UniformInt [16, 1024]	256
Dropout	[0, 0.5]	0.2
Learning rate	LogUniform [1e-5,1e-2]	1e-3
Category embedding size	UniformInt [64, 512]	128
Learning rate scheduler	[True, False]	True
Batch size	[256, 512, 1024]	512

Table B10: ResNet

Num layers	UniformInt [1, 16]	8
Layer size UniformInt	[64, 1024]	256
Hidden factor	Uniform [1, 4]	2
Hidden dropout	[0, 0.5]	0.2
Residual dropout	Uniform[0, 0.5]	0.2
Learning rate	LogUniform [1e-5, 1e-2]	1e-3
Weight decay	LogUniform [1e-8, 1e-3]	1e-7
Category embedding size	UniformInt [64, 512]	128
Normalization	[batchnorm, layernorm]	batchnorm
Learning rate scheduler	[True, False]	True
Batch size	[256, 512, 1024]	512

Table B11: FT Transformer

Num layers	UniformInt [1, 6]	3
Feature embedding size	UniformInt [64, 512]	192
Residual dropout	Uniform [0, 0.5]	0
Attention dropout	Uniform [0, 0.5]	0.2
FFN dropout	Uniform [0, 0.5]	0.1
FFN factor	Uniform [2/3, 8/3]	4/3
Learning rate	LogUniform [1e-5, 1e-3]	1e-4
Weight decay	LogUniform [1e-6, 1e-3]	1e-5
kv compression	[True, False]	True
kv compression sharing	[headwise, key-value]	headwise
Learning rate scheduler	[True, False]	False
Batch size	[256, 512, 1024]	512

Table B12: SAINT

Num layers	UniformInt [1, 2, 3, 6, 12]	3
Num heads	[2, 4, 8]	4
Dropout	[0, 0.5]	0.2
Layer size	UniformInt [32, 64, 128]	128
Dropout	[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]	0.1
Learning rate	LogUniform[1e-5, 1e-3]	3e-5
Batch size [128, 256, 512, 1024]	512	

Table B13: Predictive performance of models across 59 tasks (45 datasets) using all random hyperparameters search iterations. We report the rank over all tasks, the relative test score (Accuracy/R^2) and running time (training+inference) in seconds

Model	Rank				Mean Test Score			Mean Runing Time	
	min	max	mean	median	mean	median	std	mean	median
DT	4.000	12.000	10.525	11	0.869	0.909	0.169	0.282	0.032
EBM_S	1	11	7.686	8	0.931	0.955	0.086	21.101	5.269
EBM	1	10	5.525	5	0.959	0.982	0.067	70.597	18.139
LR	8	12	11.729	12	0.763	0.840	0.230	21.837	20.462
TabSRALinear	2	12	8.008	8	0.909	0.974	0.181	48.167	38.073
MLP	1	12	7.017	8	0.928	0.975	0.140	23.840	16.970
ResNet	2	12	7.102	8	0.911	0.976	0.185	118.734	70.919
SAINT	1	11	5.669	6	0.953	0.982	0.075	253.252	131.435
FT-Transformer	1	10	5.119	5	0.949	0.984	0.096	164.446	82.789
Random Forest	1	10	4.322	4	0.986	0.993	0.019	36.497	7.918
XGBoost	1	10	2.712	2	0.990	0.998	0.021	20.001	12.591
CatBoost	1	9	2.585	2	0.991	1.000	0.020	11.318	3.596

Table B14: Regression tasks with **numerical features only**. D1=Ailerons, D2=Bike_Sharing_Demand, D3=Brazilian_houses, D4=MiamiHousing2016, D5=abalone, D6=cpu_act, D7=delays_zurich_transport, D8=diamonds, D9=elevators, D10=house_16H

Model	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
DT	0.772	0.636	0.982	0.812	0.489	0.967	0.018	0.940	0.688	0.326
EBM_S	0.826	0.652	0.984	0.894	0.512	0.979	0.026	0.944	0.859	0.468
EBM	0.841	0.683	0.990	0.924	0.533	0.982	0.027	0.945	0.887	0.496
LR	0.819	0.280	0.803	0.720	0.476	0.666	0.005	0.929	0.815	0.229
TabSRALinear	0.838	0.644	0.983	0.922	0.514	0.964	0.012	0.942	0.902	0.484
MLP	0.836	0.674	0.994	0.909	**0.576**	0.977	0.013	0.942	0.915	0.481
ResNet	0.838	0.201	0.997	0.913	0.565	0.981	0.011	0.944	0.899	0.489
SAINT	0.571	0.685	0.994	0.921	0.564	0.985	0.021	0.943	**0.917**	0.489
FT-Transformer	0.842	0.679	**0.998**	0.921	0.568	0.980	0.019	0.943	0.915	0.470
Random Forest	0.839	0.687	0.993	0.924	0.551	0.983	**0.031**	0.945	0.837	0.502
XGBoost	0.845	0.692	**0.998**	**0.936**	0.545	**0.986**	0.030	**0.946**	0.904	**0.532**
CatBoost	**0.857**	**0.703**	0.996	**0.936**	0.546	**0.986**	0.028	**0.946**	0.913	0.488

B.6.2 Performance per dataset

In this section, we provide the detailed individual results for each task in the in *Middle-scale benchmark*. R^2 is reported as predictive performance metric for regression tasks and the Accuracy is reported for classification tasks.

Table B15: Regression tasks with **numerical features only**. D11=house_sales, D12=houses, D13=medical_charges, D14=nyc-taxi-green-dec-2016, D15= pol, D16=sulfur, D17=superconduct, D18=wine_quality, D19= yprop_4_1

Model Model	D11	D12	D13	D14	D15	D16	D17	D18	D19
DT	0.794	0.706	0.978	0.435	0.951	0.783	0.804	0.280	0.026
EBM_S	0.842	0.781	0.979	0.516	0.845	0.733	0.883	0.336	0.048
EBM	0.876	0.817	0.979	0.538	0.923	0.769	0.888	0.392	0.056
LR	0.743	0.674	0.819	0.287	0.706	0.523	0.742	0.240	0.043
TabSRALinear	0.865	0.811	0.979	0.469	0.991	0.817	0.894	0.346	0.023
MLP	0.865	0.815	**0.980**	0.454	0.963	0.843	0.893	0.390	0.014
ResNet	0.866	0.825	0.979	0.469	0.955	0.807	0.892	0.366	0.013
SAINT	0.877	0.825	0.979	0.490	0.995	0.787	0.895	0.371	0.057
FT-Transformer	0.880	0.832	0.979	0.451	0.994	0.859	0.878	0.362	0.045
Random Forest	0.870	0.829	0.979	**0.556**	0.989	0.844	0.908	**0.502**	**0.092**
XGBoost	**0.887**	0.847	0.979	0.551	0.990	0.863	**0.909**	0.490	0.080
CatBoost	**0.887**	**0.850**	0.979	0.536	0.991	**0.877**	0.908	0.495	0.089

Table B16: Regression tasks with **heterogeneous features**. D20=Airlines_DepDelay_1M, D21=Allstate_Claims_Severity, D22=Bike_Sharing_Demand, D23=Brazilian_houses, D24=Mercedes_Benz_Greener_Manufacturing, D25=SGEMM_GPU_kernel_performance, D26=abalone, D27=analcatdata_supreme, D28=delays_zurich_transport

Model	D20	D21	D22	D23	D24	D25	D26	D27	28
DT	0.037	0.384	0.789	0.984	0.574	**1.000**	0.497	0.980	0.056
EBM_S	0.046	0.510	0.744	0.983	0.549	**1.000**	0.514	0.981	0.069
EBM	0.048	0.517	0.924	0.991	0.563	**1.000**	0.537	**0.983**	0.073
LR	0.033	0.482	0.366	0.826	0.533	0.699	0.445	0.740	0.008
TabSRALinear	0.041	0.507	0.927	0.979	0.547	0.999	0.529	0.958	0.052
MLP	0.041	0.514	0.935	0.994	0.558	**1.000**	**0.577**	0.981	0.061
ResNet	0.040	0.512	0.934	0.996	0.567	**1.000**	0.575	0.978	0.057
SAINT	0.045	0.521	0.940	0.994	0.553	**1.000**	0.561	0.979	0.065
FT-Transformer	0.045	0.519	0.933	0.996	0.561	**1.000**	0.559	0.980	0.063
Random Forest	0.045	0.499	0.935	0.993	**0.577**	**1.000**	0.554	0.981	**0.076**
XGBoost	0.048	**0.535**	0.945	**0.997**	0.575	**1.000**	0.554	**0.983**	0.074
CatBoost	**0.050**	**0.535**	**0.949**	0.996	0.576	**1.000**	0.550	0.982	**0.076**

Table B17: Regression tasks with **heterogeneous features**. D29=diamonds, D30=house_sales, D31=medical_charges, D32=nyc-taxi-green-dec-2016, D33=particulate-matter-ukair-2017, D34=seattlecrime6, D35=topo_2_1, D36=visualizing_soil

Model	D29	D30	D31	D32	D33	D34	D35	D36
DT	0.964	0.795	0.978	0.441	0.636	0.180	0.009	**1.000**
EBM_S	0.987	0.853	0.979	0.521	0.670	0.180	0.049	0.935
EBM	0.989	0.885	0.979	0.563	0.679	**0.186**	0.053	0.993
LR	0.952	0.751	0.819	0.318	0.533	0.040	0.000	0.871
TabSRALinear	0.983	0.879	0.978	0.533	0.652	0.061	0.000	0.996
MLP	0.987	0.878	**0.980**	0.470	0.659	0.171	0.025	**1.000**
ResNet	0.987	0.882	0.979	0.486	0.662	0.176	0.019	0.998
SAINT	0.989	0.889	0.979	0.498	0.669	0.180	0.054	**1.000**
FT-Transformer	0.990	0.891	0.979	0.470	0.671	0.179	0.039	**1.000**
Random Forest	0.988	0.875	0.979	0.567	0.673	0.182	**0.070**	**1.000**
XGBoost	**0.991**	**0.897**	0.978	**0.575**	**0.691**	0.185	0.060	**1.000**
CatBoost	**0.991**	**0.897**	0.979	0.560	0.690	**0.186**	0.069	**1.000**

Table B18: Classification tasks with **numerical features only**. D37=Bioresponse, D38=Diabetes130US, D39=Higgs, D40=MagicTelescope, D41=MiniBooNE, D42=bank-marketing, D43=california, D44=covertype

Model	D37	D38	D39	D40	D41	D42	D43	D44
DT	0.680	0.601	0.649	0.788	0.869	0.771	0.839	0.745
EBM_S	0.772	0.605	0.686	0.828	0.915	0.799	0.879	0.752
EBM	0.775	**0.606**	0.707	0.850	0.924	0.803	0.890	0.777
LR	0.735	0.599	0.636	0.768	0.842	0.742	0.831	0.627
TabSRALinear	0.767	0.605	0.679	0.850	0.918	0.789	0.877	0.793
MLP	0.763	0.604	0.685	0.850	0.933	0.789	0.868	0.780
ResNet	0.766	0.604	0.689	0.858	0.936	0.786	0.870	0.790
SAINT	0.758	0.604	0.705	0.847	**0.937**	0.793	0.880	0.801
FT-Transformer	0.748	0.605	0.703	0.857	0.934	0.794	0.885	0.799
Random Forest	**0.794**	0.604	0.707	0.853	0.927	0.797	0.892	0.824
XGBoost	0.792	**0.606**	**0.714**	0.859	**0.937**	0.805	0.901	0.817
CatBoost	0.785	0.605	0.712	**0.860**	**0.937**	**0.806**	**0.904**	**0.830**

Table B19: Classification tasks with **numerical features only**. D45=credit, D46=default-of-credit-card-clients, D47=electricity, D48=eye_movements, D49=heloc, D50=house_16H, D51=jannis, D52=pol

Model	D45	D46	D47	D48	D49	D50	D51	D52
DT	0.752	0.699	0.775	0.566	0.693	0.823	0.715	0.915
EBM_S	0.767	0.713	0.821	0.590	**0.722**	0.870	0.747	0.948
EBM	0.770	0.716	0.829	0.614	0.721	0.877	0.763	0.978
LR	0.706	0.678	0.740	0.556	0.710	0.821	0.724	0.855
TabSRALinear	0.744	0.709	0.792	0.584	0.719	0.873	0.749	0.983
MLP	0.771	0.708	0.807	0.578	0.719	0.877	0.745	0.944
ResNet	0.772	0.706	0.808	0.581	0.718	0.873	0.744	0.948
SAINT	0.763	0.714	0.819	0.583	0.718	**0.888**	0.767	0.979
FT-Transformer	0.775	0.714	0.816	0.584	0.721	0.880	0.763	0.981
Random Forest	0.772	**0.718**	0.859	0.645	0.717	0.881	0.772	0.982
XGBoost	0.773	0.716	**0.868**	**0.662**	0.717	**0.888**	0.777	0.981
CatBoost	**0.776**	**0.718**	0.861	0.642	0.717	0.887	**0.778**	**0.984**

Table B20: Classification tasks with **heterogeneous features**. D53=albert, D54=compas-two-years, D55=covertype, D56=default-of-credit-card-clients, D57=electricity, D58=eye_movements, D59=road-safety

Model	D53	D54	D55	D56	D57	D58	D59
DT	0.637	0.660	0.760	0.698	0.767	0.565	0.727
EBM_S	0.652	0.675	0.776	0.712	0.827	0.596	0.732
EBM	**0.658**	0.672	0.799	0.717	0.838	0.616	0.749
LR	0.635	0.667	0.772	0.679	0.740	0.567	0.697
TabSRALinear	0.643	0.668	0.849	0.711	0.806	0.602	0.748
MLP	0.652	0.678	0.834	0.709	0.819	0.586	0.756
ResNet	0.650	0.676	0.833	0.704	0.824	0.589	0.761
SAINT	0.652	0.674	0.847	0.712	0.830	0.593	0.765
FT-Transformer	0.653	**0.682**	0.858	0.715	0.831	0.590	0.769
Random Forest	0.654	0.679	0.859	0.717	0.876	0.658	0.760
XGBoost	0.656	0.678	0.857	0.714	0.883	**0.660**	0.768
CatBoost	0.656	0.676	**0.873**	**0.719**	**0.884**	0.651	**0.770**