NCART: Neural Classification and Regression Tree for Tabular Data

Jiaqi Luo^a, Shixin Xu^{a,*}

^aData Science Research Center, Duke Kunshan University, No.8 Duke Ave., Kunshan, 215000, Jiangsu Province, China

Abstract

Deep learning models have become popular in the analysis of tabular data, as they address the limitations of decision trees and enable valuable applications like semi-supervised learning, online learning, and transfer learning. However, these deep-learning approaches often encounter a trade-off. On one hand, they can be computationally demanding when dealing with large-scale or high-dimensional datasets. On the other hand, they may lack interpretability and may not be suitable for small-scale datasets. In this study, we propose a novel interpretable neural network called Neural Classification and Regression Tree (NCART) to overcome these challenges. NCART is a modified version of Residual Networks that replaces fully-connected layers with multiple differentiable oblivious decision trees. By integrating decision trees into the architecture, NCART maintains its interpretability while benefiting from the end-to-end capabilities of neural networks. The simplicity of the NCART architecture makes it well-suited for datasets of varying sizes and reduces computational costs compared to state-of-the-art deep learning models. Extensive numerical experiments demonstrate the superior performance of NCART compared to existing deep learning models, establishing it as a strong competitor to tree-based models.

Keywords:

Tabular data, Neural Networks, Classification and Regression Tree

Email addresses: jiaqi.luo@dukekunshan.edu.cn (Jiaqi Luo), shixin.xu@dukekunshan.edu.cn (Shixin Xu)

^{*}Corresponding author

1. Introduction

Tabular data is ubiquitous across various fields and industries, representing a wide range of valuable information, including financial records, customer information, and scientific measurements. Its pervasive presence underscores the importance of accurate prediction in tabular data analysis. Such predictions empower businesses, researchers, and analysts to fully leverage the potential of these datasets and drive meaningful advancements in their respective domains.

Decision tree models are widely recognized as an effective machine learning technique, particularly well-suited for analyzing tabular data. They offer several advantages, including interpretability, handling both numerical and categorical data, ease of implementation, and scalability. Decision trees are the basis for more advanced ensemble methods such as random forests [1] and gradient boost decision trees (GBDT) [2, 3, 4]. Nevertheless, they suffer from a lack of flexibility, discontinuity and the inability to pre-trained models, online learning, and transfer learning, which limits their use.

The capabilities of deep learning [5] in extracting high-level features, its end-to-end training capability, and overall flexibility have led to remarkable achievements in various domains, including image analysis [6], natural language processing [7], and generative modeling [8]. In the context of modeling tabular data, deep learning has emerged as a promising approach to address the limitations of decision trees. Researchers have been actively exploring innovative techniques [9] with the aim of overcoming the drawbacks of traditional methods and expanding the range of applications suitable for deep learning.

Recent studies have highlighted that deep learning models may not always outperform other machine learning methods, such as gradient boosting when applied to tabular data [9, 10, 11]. The relatively inferior performance of deep learning models in this context can be attributed to various factors. One reason is the limited availability of labeled training data. Deep learning models typically require large amounts of labeled data to effectively learn meaningful representations and generalize well. When training data is limited, the performance of deep learning models may be compromised. Additionally, many deep learning models often contain a large number of parameters. When handling high-dimensional or large-scale datasets, they can be time-consuming and require more resources. Interpretability remains a significant challenge in deep learning due to the complexity of these models. With their numerous layers and millions of parameters, deep learning models often act as black boxes, making it difficult to understand the decision-making process behind their predictions. This lack of interpretability

can be a significant drawback, especially in clinical or financial applications where limited data sets and interpretability are crucial for informed decision-making and interventions.

To tackle these challenges, we propose a novel tree-induced deep learning model called Neural Classification and Regression Tree (NCART). Drawing inspiration from the success of differentiable decision trees [12, 13] and ResNet [10] in handling tabular data, our model combines the strengths of decision trees and deep learning. Instead of using fully-connected layers, our model replaces them with a sum of several differentiable decision trees. This allows our model to capture interpretability similar to traditional tree models while benefiting from the end-to-end training capabilities of deep learning. Furthermore, the simplicity of the ResNet architecture ensures that our model achieves desirable results in a limited time. Our proposed model is designed to work effectively with datasets of varying sizes and demonstrates competitive performance compared to other stateof-the-art machine learning methods. By integrating the superiority of decision trees with the computational power of deep learning, our model not only achieves good performance but also provides insights into the underlying patterns and relationships within tabular data. Through this innovative combination, our NCRAT model bridges the gap between interpretability and performance in the analysis of tabular data, making it a promising approach for a wide range of applications.

The main contributions of this paper are summarized as follows:

- Proposal of NCART: We introduce NCART, a deep learning model that
 integrates differentiable decision trees into a fully-connected ResNet. The
 model is applicable to datasets of varying scales while maintaining lower
 computational costs compared to existing deep learning-based methods.
- Interpretable Model: NCART exhibits interpretability, generating feature importance similar to those produced by traditional decision tree models. This information is valuable for identifying the most influential features in the decision-making process and understanding the underlying patterns in the data.
- Superior Performance: Through extensive experimentation, NCART demonstrates superior performance compared to state-of-the-art deep learning models. Furthermore, it remains competitive with decision tree models, showcasing its efficacy in tackling complex tasks.

2. Related Work

In this section, we provide an overview of the concepts from previous research that are pertinent to deep learning for tabular data. We divide the deep learning modes into three categories according to the network structure, namely tree-induced networks, transformer-based networks, and other specialized models.

2.1. Tree-induced Neural Networks

Given the effectiveness of tree-based models for tabular data, researchers have made efforts to incorporate differentiable trees that combine the advantages of tree ensembles with gradient-based optimization of neural networks. Several notable approaches in this direction are summarized as follows:

- Deep Neural Decision Trees (DNDT): Proposed by Yang et al. [12], DNDT is a neural network-based tree model that is straightforward to implement. It incorporates a soft binning function for split decisions, enhancing interpretability. However, its scalability is limited in high-dimensional feature spaces due to the Kronecker product computation for binned features.
- Neural Oblivious Decision Ensembles (NODE): Inspired by CatBoost, NODE
 [13] employs oblivious decision trees to construct an ensemble model. It is
 a fully differentiable model that can handle both numerical and categorical
 features without requiring preprocessing. NODE has demonstrated superior
 performance compared to GBDT models across multiple datasets, albeit
 with higher computational costs.
- Net-DNF: Net-DNF [14] leverages the representation of decision trees as Boolean formulas, specifically disjunctive normal forms (DNF). By utilizing this insight, Net-DNF emulates the characteristics of GBDT and achieves comparable results to XGBoost on certain classification tasks.
- Binary Tree Learning: The authors propose a novel method [15] that combines Argmin differentiation and a mixed-integer programming model to simultaneously learn discrete and continuous parameters of the tree, which allows the users to leverage the generalization capabilities of stochastic gradient optimization for decision splits and principled tree pruning.

These approaches aim to combine the interpretability of decision trees with the optimization capabilities of deep learning models, providing promising alternatives for tackling tabular data challenges.

2.2. Neural Networks based on Transformer

Inspired by the recent successes of transformer-based models in natural language processing (NLP) [7], researchers have proposed multiple approaches using attention mechanisms for heterogeneous tabular data. TabNet [16] is one of the pioneering transformer-based models designed for tabular data. Its architecture consists of multiple subnetworks processed hierarchically in a sequential manner. Each subnetwork represents a decision step that performs soft instance-wise feature selection. The authors demonstrate that TabNet surpasses other variants on various large-scale tabular datasets. TabTransformer [17] employs self-attentionbased transformers to map categorical features to contextual embeddings. This approach enhances robustness to missing or noisy data and enables interpretability. Extensive experiments show that TabTransformer achieves performance comparable to tree-based ensemble techniques, demonstrating success even in the presence of missing or noisy data. FT-Transformer [10] is a straightforward adaptation of the Transformer architecture for tabular data. It considers the relationship between numerical and categorical features by feeding them together to transformer blocks, which TabTransformer has not considered. Self-Attention and Intersample Attention Transformer (SAINT) [18] is a hybrid attention approach that combines self-attention with inter-sample attention over multiple rows. Additionally, the researchers leverage self-supervised contrastive pre-training to enhance performance for semi-supervised problems. In the original experiments, SAINT consistently outperforms other methods on supervised and semi-supervised tasks. Non-Parametric Transformer (NPT) [19] utilizes the entire dataset as input. By employing attention between data points, the model can model and leverage relations between arbitrary samples for classifying test samples. However, these transformer-based methods have been observed to incur significant computational costs, including running time and hardware. TabPFN [20] is a new transformerbased model that can handle small-scale tabular data in less than a second with no hyperparameters tuning and can get competitive results with GBDT. However, its effectiveness is restricted to classification tasks involving numerical features. Furthermore, performing inference on larger-scale datasets poses a challenge on current consumer GPUs. Recently, inspired by the investigation of inherent characteristics of tabular data pointed out by [11], researchers have identified that proper handling of feature interactions and feature embedding can enhance the success of neural networks on tabular data. ExcelFormer [21] alternates between two attention modules for feature interactions and embedding updates, employing a specialized training method that gradually optimizes these modules through specific regularization techniques. What sets ExcelFormer apart is its remarkable performance, as it can achieve results on par with finely tuned models like XG-boost and Catboost, all without the need for hyperparameter tuning. T2G-Former [22] introduces a novel Graph Estimator to organize independent tabular features into a feature relation graph, enabling the development of a specialized tabular learning Transformer designed to compete effectively with GBDT.

2.3. Models with other specialized structures

To explain the underlying relationships between features, Zheng et al. introduced Dual-Route Structure-Adaptive Graph Networks (DRSA-Net) [23]. DRSA-Net dynamically learns a sparse graph structure between variables and characterizes interactions through dual-route message passing. Extensive experiments in various application scenarios demonstrate that DRSA-Net is competitive with classical algorithms and deep models. Regularization Learning Networks (RLN) [24] incorporate trainable regularization coefficients for individual weights in a neural network to reduce sensitivity. These coefficients promote sparsity in the network and can outperform deep neural networks, achieving results comparable to those of GBDT. However, the evaluation is based on a dataset primarily composed of numerical data and cannot handle categorical features. Wide&Deep [25] and DeepFM [26] are designed for online prediction where the input space is tabular data. They both use embedding techniques to handle the categorical features, but the difference is that Wide&Deep employs a linear model and a wide selection of hand-crafted logical expressions on features to improve the generalization capabilities while DeepFM applies a learned Factorization Machines to replace the hand-crafted features to improve the model performance. The DeepGBM model [27] is also for online prediction, it consists of two components: CatNN and GBDT2NN. The former is to handle the categorical features and has the same structure as DeepFM, the latter is to distill knowledge from a trained GBDT. By integrating the advantages of GBDT on tabular data and the flexibility of deep neural networks, DeepGBM can outperform other models on various online prediction problems. The DCN-V2 [28] framework is introduced as an improved approach to address the limitations of these existing feature interaction learning methods, enhancing practicality in large-scale industrial applications while outperforming state-of-the-art algorithms on benchmark datasets.

The lack of a consistent structure in general tabular data, unlike image and language data, poses a limitation on the application of deep learning. In order to address this challenge, researchers have devised several strategies. One prominent approach involves the transformation of tabular data into images or text. By utilizing more efficient models in CV and NLP, these efforts aim to enhance the perfor-

mance of deep learning in the domain of tabular data [29, 30]. Another solution is designing effective neural network architectures that aim to generate high-level features to boost performance, including TabNN [31], DANets [32], and TabCaps [33]. This innovative approach empowers deep learning to work better with tabular data, making it more flexible and efficient in handling the unique patterns that tabular data poses. Furthermore, Value Imputation and Mask Estimation (VIME) [34] extends the success of self- and semi-supervised learning in CV and NLP to tabular data. AutoGluon Tabular [35] builds ensembles of basic neural networks together with other traditional ML techniques, with its key contribution being a strong stacking approach.

Most deep neural networks are trained using back-propagation (BP), which may yield suboptimal performance or limited generalization [36]. To overcome this drawback, researchers have utilized Random Vector Functional Link (RVFL) networks [37] to design models for tabular data [38, 39]. Experimental results demonstrate that the proposed networks are competitive with other state-of-the-art neural networks. However, the method lacks comparisons with GBDT, which is usually considered the preferred choice for tabular data.

3. Neural Classification and Regression Tree

In this section, detailed descriptions of Neural Classification and Regression Tree (NCART) are presented. We first introduce the differentiable decision trees. Next, we present the architecture of NCART. Finally, we conclude this section with a discussion on the interpretability of NCART.

3.1. Approximation of a Decision Tree

The standard decision tree is a recursive subdivision of the feature space, but this process cannot be made differentiable, and of course, it cannot be expressed by a neural network. Instead of using the traditional decision tree, we have chosen to utilize the Oblivious Decision Tree (ODT) [40]. The idea is inspired by the successful endeavor to make the ODT differentiable, as previously mentioned [13].

An oblivious decision tree is a type of binary decision tree where each level tests the same feature. It splits the data along each feature and compares each feature to a learned threshold. The leaf value of an ODT is defined as

$$c(\mathbf{x}) = g(H(x_1 - s_1), \dots, H(x_n - s_n)),$$
 (1)

where $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ is the input feature vector, $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{R}^n$ is the learned threshold, $g : \mathbb{R}^n \to \mathbb{R}$ is a function mapping each split region to its corresponding leaf value and

$$H(x) = \begin{cases} 0, & \text{if } x < 0\\ 1, & \text{if } x \ge 0 \end{cases}$$
 (2)

is the *Heaviside step function* that indicates the decision routes.

However, an ODT does not consider the interaction between different features when making splits, which makes it less expressive compared to standard trees. To address the issue, we combine multiple ODTs together to approximate the representation ability of a standard decision tree.

Figure 1 presents an example to illustrate how to approximate a decision tree with two ODTs. On the left, there is a decision tree that partitions the feature space into a mesh comprising 5 distinct subregions. Within this mesh, there are nodes characterized by three connected edges, resulting in a grid-like structure resembling the letter "T". To further refine the representation, we extend the connections for each of these "T" nodes, creating a grid mesh. It's important to note that this extension does not impact the value of each individual region. Subsequently, we can approximate the refined feature space by considering it as the sum of two separate spaces. Each space has 4 distinct subregions and it can be effectively expressed as ODTs. The leaf values of the ODTs can be determined by solving the following objective:

$$\min |ODT_1 + ODT_2 - DT|$$

$$= \min_{a_i, b_j} \{ |a_1 + b_1 - c_1| + |a_2 + b_1 - c_1| + |a_2 + b_2 - c_2|$$

$$+ |a_3 + b_1 - c_3| + |a_3 + b_3 - c_3| + |a_4 + b_1 - c_4|$$

$$+ |a_4 + b_2 - c_4| + |a_4 + b_3 - c_5| + |a_4 + b_4 - c_5| \},$$

where ODT_i and DT are the vectors formed by the leaf values of ODT and the decision tree, respectively. By solving this optimization problem, we can effectively approximate a decision tree using two distinct oblivious trees.

It is important to highlight that ensemble tree models can be approximated using ODTs as they essentially consist of models composed of multiple trees. However, to achieve similar performance, ODTs often require a larger number of trees compared to standard decision trees. Furthermore, in an ODT, while nodes at the same level share the same test feature, the split values may vary. Since the

results are based on an ensemble of ODTs, we can focus on a specific formulation where each level of ODTs has identical split values. This approach helps reduce the dimensionality of the function and simplifies the computation process.

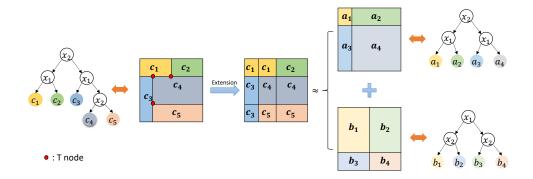


Figure 1: Illustration of approximation of a decision tree using two oblivious trees. The leaf values have the following relationships: $a_1 + b_1 \approx c_1$, $a_2 + b_1 \approx c_1$, $a_2 + b_2 \approx c_2$, $a_3 + b_1 \approx c_3$, $a_3 + b_3 \approx c_3$, $a_4 + b_1 \approx c_4$, $a_4 + b_2 \approx c_4$, $a_4 + b_3 \approx c_5$, $a_4 + b_4 \approx c_5$.

3.2. NCART Architecture

The architecture of an NCART block, which is illustrated in Figure 2, consists of 4 components: Data preprocessing, Feature Selection, Differentiable Oblivious Trees, and Ensemble.

Data preprocessing. Suppose the input is \mathbf{x} , we do not do any special processing on the input vector (including categorical features), and the vector is directly sent to the first layer which is a simple *Batch Normalization*:

$$\mathbf{x}^{B} = BatchNormalization(\mathbf{x}). \tag{3}$$

Feature selection. Feature selection plays a critical role in enhancing model performance by identifying and incorporating the most influential features. In an ideal scenario, our objective is to ascertain a projection matrix, denoted as $\mathbf{P}_{d,n}$, with the stipulation that each of its rows corresponds to a one-hot vector:

$$\widetilde{\mathbf{x}} = \mathbf{P}\mathbf{x}^B. \tag{4}$$

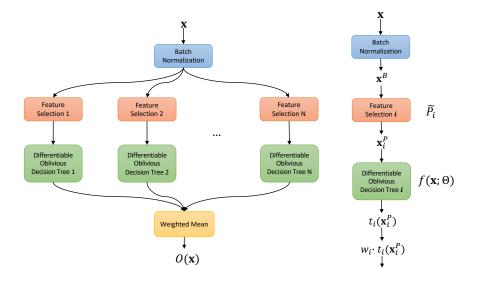


Figure 2: Structure of an NCART block with 4 components: Batch Normalization (blue block) for data preprocessing, Feature Selection (red block, Eq. (4)), Differentiable Oblivious Tree (green block, Eq. (8)), and Weight Mean for ensemble (yellow block, Eq. (9)).

Here, $\mathbf{x}^B = (x_1^B, \cdots, x_n^B)^{\mathsf{T}}$, $\widetilde{\mathbf{x}} = (x_{k_1}^B, \cdots, x_{k_d}^B)^{\mathsf{T}}$, and k_1, k_2, \cdots, k_d represents a subsequence of $1, \cdots, n$. The elements of matrix \mathbf{P} , denoted as P_{ij} , are binary, taking values in the set 0, 1, and they adhere to the constraint $\sum_{j=1}^n P_{ij} = 1$ for all i.

However, it is hard to learn **P** due to its discrete nature, existing in a discrete integer space. In our research, we introduce an innovative approach that involves the application of a sparse transformation to each row of a learnable matrix. This process leads to the creation of a sparse projection, which is mathematically represented as:

$$\widetilde{\mathbf{P}} = h(\mathbf{A}) = \begin{bmatrix} h(\mathbf{A}_1) \\ h(\mathbf{A}_2) \\ \dots \\ h(\mathbf{A}_d) \end{bmatrix}.$$
 (5)

Specifically, **A** is a learnable matrix in the neural network. $\mathbf{A}_i = (A_{i1}, \dots, A_{in})$ is the i_{th} row of **A**. h is a sparse function similar to the softmax, but able to output sparse probabilities, ensuring that the sum of each row remains equal to 1. One popular choice for the sparse function is the *Sparsemax* [41] or *entmax* [42], which is considered a hyperparameter in our network configuration. Further details regarding the mathematical expressions defining the sparse function can be

found in Appendix A.

Subsequently, we apply this sparse projection to the feature vector \mathbf{x} , resulting in the projected feature vector \mathbf{x}_P :

$$\mathbf{x}^P = \widetilde{\mathbf{P}}\mathbf{x}^B. \tag{6}$$

Differentiable Oblivious Decision Tree. In order to make (1) differentiable, we undertake the following two modifications:

- We replace the non-differentiable function H(x) with the sigmoid function, denoted as $\sigma(x) = \frac{1}{1+e^{-x}}$.
- We change g to be a two-layer fully-connected network f with ReLU activation, which is expressed as:

$$f(\mathbf{x}; \mathbf{\Theta}) = \mathbf{W_2}(ReLU(\mathbf{W_1}\mathbf{x} + \mathbf{b_1})) + \mathbf{b_2},\tag{7}$$

where Θ are the parameters in the network.

Subsequently, these modifications yield a differentiable version of the ODT, which is defined as:

$$t(\mathbf{x}^P) = f(\sigma(\mathbf{x}^P - \mathbf{s}; \Theta))$$

= $f(\sigma(x_1^P - s_1), \sigma(x_2^P - s_2), \dots, \sigma(x_n^P - s_n); \Theta).$ (8)

The key advantage of these modifications is that the differentiability of both the sigmoid function σ and the function f enables the tree model t to also be differentiable, thereby enhancing its capacity for learning.

Ensemble. All the individual outputs are collectively directed to the final layer, where a weighted vote is conducted to produce the output, defined as:

$$O(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} w_i \cdot t_i(\mathbf{x}_i^P)$$
(9)

Here, $\mathbf{x}_i^P = \widetilde{\mathbf{P}}_i \mathbf{x}^B$ is the output of i_{th} projection (see Fig. 2). The parameters w_i , with i ranging from 1 to N, are considered to be learnable, allowing the model to adapt and optimize these parameters during the training process.

NCART Network Architecture. In a GBDT algorithm, each new tree is employed to learn the residual errors or gradients of the loss function. A similar concept is adopted in the ResNet, where each new block represents the residual mapping to be learned, effectively behaving like an ensemble of shallow networks [43].

Drawing inspiration from these techniques, we propose to combine multiple NCART blocks to enhance the overall model's performance. The architecture of our ensemble model is illustrated in Fig. 3, resembling a modified feed-forward ResNet in which each fully-connected block is replaced with an NCART block. In our default configuration, each NCART block consists of an equal number of differentiable trees, and the final NCART block, highlighted in red, serves the purpose of dimensionality reduction.

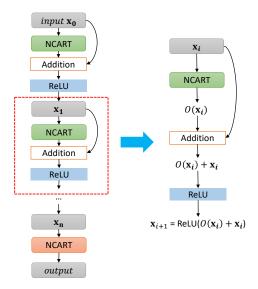


Figure 3: NCART Network architecture. A green-filled block indicates the utilization of all features for feature splitting, while a red-filled block signifies the presence of a feature selection layer.

3.3. Feature Importance

Calculating feature importance in NCART involves a process similar to that of traditional decision trees. Since our model consists of multiple tree structures, the methodology for determining feature importance remains akin to conventional decision tree analysis.

For a differentiable ODT, we can easily derive the decision rule for a feature from Eq.(8). For a given feature index j, if $\sigma(x_j - s_j) > 0.5$, it indicates that

 $x_j > s_j$ and the corresponding samples will be assigned to the right leaf. Conversely, if $\sigma(x_j - s_j) \le 0.5$, the samples will be assigned to the left leaf. With this approach, we can directly apply the Gini coefficient, commonly used in traditional decision trees, to measure the impurity of feature x_j . The formulation for the Gini coefficient is as follows:

$$I_{j} = 1 - \left(\frac{m_{j}^{left}}{M}\right)^{2} - \left(\frac{m_{j}^{right}}{M}\right)^{2},\tag{10}$$

where M is the number of samples. m_j^{left} is the number of samples divided into the left leaf, and m_j^{right} is the number of samples that are divided into the right leaf.

If the NCART block includes a feature selection layer, the Gini coefficient can be formulated as follows:

$$I_j^{sel} = \sum_{k=1}^d \left(1 - \left(\frac{m_k^{left}}{M} \right)^2 - \left(\frac{m_k^{right}}{M} \right)^2 \right) h(A)_{kj}. \tag{11}$$

where *d* is the dimensionality of feature selection.

In an NCART with L blocks (Fig. 3), where the last block contains a feature selection layer, each differentiable tree is independent, and every feature in a differentiable tree only splits once. Thus, to calculate the Gini coefficient of a feature x_j in an NCART, we can obtain it by summing up the results from all the individual trees:

Importance_j =
$$\sum_{l=1}^{L-1} \sum_{k=1}^{N} I_{lkj} + \sum_{k=1}^{N} I_{kj}^{sel}$$
, (12)

where I_{lkj} represents the Gini coefficient of feature x_j at the k_{th} tree of the l_{th} layer.

4. Experiments

4.1. Experiments Setup

Datasets. We conduct comparisons on 20 datasets, which are all from OpenML¹. These datasets serve diverse purposes, including both classification and regression tasks. They encompass a range of sample sizes and feature types, combining

¹https://www.openml.org/

categorical and numerical variables. For all datasets, we didn't do any special preprocessing, except apply the ordinal encoding to the categorical features and specify the corresponding indices for CatBoost and Transformer-based models. More details about the datasets can be found in Appendix B.

Benchmark models. Our comparative analysis includes the proposed models and benchmarking against three GBDT models and seven widely-used deep learning baselines: XGBoost [2], CatBoost [4], LightGBM [3], NODE [13], TabNet [16], SAINT [18], FT-Transformer [10], TabCaps [33], RLN [24], and ResNet [10]. It's important to note that TabCaps is exclusively designed for classification tasks and may not be suitable for regression tasks.

Evaluation metrics. For classification tasks, both binary and multi-class, we assess model performance using the **AUC** (Area Under the Receiver Operating Characteristic Curve). Additionally, we employ the **F1-score** as an evaluation metric for classification tasks, considering potential dataset imbalances. In the case of regression tasks, we utilize the **MSE** (Mean Squared Error) to evaluate the performance of the various methods.

Training procedure. Our training procedure follows the guidelines outlined in [9] for all baseline models. We evaluate the performance of our models using their open-source code². To tune hyperparameters, we utilize the Optuna library³ with 10 trials for each model. Each trial configuration undergoes 5-fold cross-validation. Detailed descriptions of the adjusted hyperparameters can be found in Appendix C, while other parameters remain at their default values. For neural network models, we conduct training for 1000 epochs using the Adam optimizer[44] with a learning rate of 0.001. Our training batches consist of 1024 samples, with validation batches comprising 256 samples. For each algorithm and each 5-fold cross-validation, we run the algorithm for up to 50000s to prevent excessively long runtimes. All experiments are conducted on a workstation equipped with an Intel Core i9-10900X CPU, 128GB RAM, and an NVIDIA-3080 GPU.

4.2. Performances

To ensure reproducibility and mitigate model variance, for each approach and dataset, we report the mean and standard deviation out of 5-fold runs for the best

²https://github.com/kathrinse/TabSurvey

³https://optuna.org/

Dataset Parameters diabetes F	Metric				4	TahNet	Model	FTTrans	TabCaps	RLN	RecNet	NCART
		XGBoost	CatBoost	LightGBM	NODE	Taurion		1.1.114115	. 1		INCOLINCE	INCOME
					B	Binary Classification	tion					
	AUC F1-score	82.44 ± 4.03 59.69 ± 4.25	83.53 ± 3.24 60.9 ± 6.46	82.63 ± 3.5 44.19 ± 23.19	82.98 ± 3.11 62.51 ± 6.62	54.06 ± 1.79 39.17 ± 2.01	62.30 ± 5.63 12.17 ± 17.34	81.64 ± 3.39 57.71 ± 4.83	74.55 ± 5.39 50.04 ± 9.33	63.59± 6.21 47.65±5.56	82.66 ± 3.5 62.06 ± 4.47	$\frac{83.72 \pm 3.39}{64.61 \pm 5.39}$
credit-g	AUC F1-score	78.14 ± 4.19 83.62 ± 2.4	78.45 ± 6.03 84.53 ± 2.03	79.13±4.31 83.22±2.56	76.66 ±4.63 82.69±1.67	57.34 ± 1.98 81.89 ± 0.28	77.54 ± 5.0 83.43±2.47	77.38±6.2 82.95±2.98	74.79±0.94 83.09±1.00	54.22 ± 6.31 82.53 ± 0.35	74.67±4.03 80.35±2.77	76.86±2.74 84.2±1.79
qsar-biodeg F	AUC F1-score	93.43 ±1.52 80.1±3.74	92.87±2.02 79.42±3.00	93.54 ±1.15 80.42±1.52	92.48±2.04 79.89±1.82	57.34 ±5.89 42.18±4.75	93.01±1.09 80.56±3.18	92.88±2.14 78.43±3.18	93.01±2.19 81.96±2.81	91.33±1.71 75.3±1.38	92.93±1.38 81.13±2.8	93.64±1.14 82.14±1.71
scene	AUC F1-score	99.14±0.19 94.76±1.69	99.21±0.39 96.88±1.20	99.30±0.26 94.36±1.74	98.93±0.20 95.1±1.64	81.26±2.79 70.01±4.58	ООМ	ООМ	98.96±0.54 97.00±1.34	99.30±0.16 96.88±1.20	98.06±1.63 94.99±1.33	98.65±0.36 95.27±0.62
ozone-level F	AUC F1-score	91.58±2.51 36.99±10.15	91.87±2.94 28.71±11.61	91.94±2.80 36.94±14.65	89.97±3.40 0.0±0.00	67.13±5.93 19.11±10.23	89.18±1.28 26.0±8.82	85.94±2.59 21.93±15.38	89.80±9.28 26.38±17.24	64.14±13.39 4.03±5.61	89.35±4.02 13.14±2.02	93.10±2.04 27.15±15.12
delta_ailerons F	AUC F1-score	97.91±0.36 94.88 ± 0.20	97.85±0.34 94.68±0.29	97.98±0.22 94.78±0.29	97.38±0.29 94.42±0.33	97.03±1.14 89.73±10.22	96.38±0.39 93.88±0.46	97.48±0.42 94.52±0.38	96.86±0.64 94.16±0.54	60.67 ± 18.47 69.34 ± 0.02	97.18±0.48 92.52±1.90	97.56±0.38 94.69±0.46
MagicTelescop F	AUC F1-score	93.46±0.56 85.55±0.86	$\frac{93.57 \pm 0.75}{85.72 \pm 1.16}$	93.47±0.67 85.54±1.08	92.05±0.60 84.08±0.40	92.71±0.86 84.14±0.57	90.23±1.66 81.52±1.51	93.16±0.57 84.63±0.41	92.69±0.60 84.63±1.14	84.96±1.7 76.65±1.15	90.80±1.99 81.29±3.07	92.98±0.86 84.73±1.51
jannis	AUC F1-score	87.34±0.14 80.09±0.11	$\frac{87.52\pm0.22}{80.38\pm0.25}$	87.13±0.23 79.93±0.18	84.65±0.29 77.92±0.40	86.17±0.38 79.03±0.57	78.39±1.29 72.66±1.51	87.02±0.24 80.17±0.41	85.58±0.27 78.62±0.30	80.38±1.36 72.04±1.61	85.58±0.30 78.48±0.39	86.74±0.16 79.54±0.33
road-safety F	AUC F1-score	$\frac{90.24 {\pm} 0.29}{81.86 {\pm} 0.33}$	89.29 ± 0.27 80.55 ± 0.32	90.11±0.28 81.76±0.34	87.43±0.87 78.75±0.40	88.54±0.24 79.42±0.29	86.62±0.87 77.44±0.42	76.14 \pm 0.24 69.78 \pm 0.92	84.56±1.89 75.16±1.15	54.78±2.99 27.58±32.89	79.53±0.49 72.42±0.37	88.23±0.18 79.17±0.23
Higgs	AUC F1-score	82.80±0.05 74.67±0.08	82.79±0.03 74.68±0.04	82.46±0.10 74.45±0.05	82.34±0.57 74.56±0.22	83.26±0.43 75.04±0.50	83.62±0.06 75.36±0.18	82.82±0.17 72.41±1.02	82.42±0.23 74.61±0.22	69.9±2.28 63.87±1.09	$\frac{84.35\pm0.05}{75.85\pm0.19}$	83.45±0.09 75.15±0.20
autoUniv-au7	AUC F1-score	71.43 ± 2.61 53.38 \pm 2.61	69.56±2.80 47.65±2.80	72.34±3.04 51.68±3.04	Muli 66.12±2.79 44.85±2.79	Multi-class Classification 79 51.07±3.54 56.7 79 31.32±3.54 36.2	cation 56.71±4.55 36.22±4.55	62.20±2.57 38.64±2.57	63.72±3.33 42.69±3.33	50.07±0.14 17.52±0.14	58.90±4.43 39.93±4.43	65.40±2.30 47.45±2.30
plants-margin F	AUC F1-score	99.41±0.10 73.6±0.10	$\frac{99.82 \pm 0.02}{84.43 \pm 0.02}$	99.06±0.18 74.53±0.18	99.53±0.08 78.82±0.08	98.37±0.27 56.59±0.27	99.47±0.08 69.96±0.08	99.46±0.16 72.27±0.16	99.60±0.10 83.64±0.10	91.64±0.65 4.45±0.65	99.68±0.06 80.0±0.06	99.54±0.05 76.04±0.05
waveform	AUC F1-score	96.87 ± 0.22 85.42 ± 0.22	97.09 ± 0.28 86.03 ± 0.28	96.78±0.22 84.86±0.22	97.26 ± 0.20 86.48 ± 0.20	96.14±0.45 84.38±0.45	97.09±0.24 86.15±0.24	$\frac{97.42 \pm 0.28}{86.89 \pm 0.28}$	97.25±0.21 86.27±0.21	97.33 ± 0.20 87.00±0.20	97.42±0.34 86.68±0.24	97.17±0.25 86.34±0.25
gas-drift F	AUC F1-score	99.95±0.04 99.44±0.04	99.96±0.03 99.42±0.03	99.96 ± 0.04 99.41 ± 0.04	99.86±0.06 98.43±0.06	99.88±0.05 98.4±0.05	99.96±0.02 99.37±0.02	99.74±0.09 96.16±0.09	99.76±0.09 98.86±0.09	83.20±14.58 59.05±14.58	99.88±0.02 98.83±0.02	99.96±0.03
EMNIST	AUC F1-score	99.49±0.01 83.53±0.01	99.56 ± 0.01 84.09 ± 0.01	99.44 ± 0.01 83.18 ± 0.01	99.42 ± 0.02 80.99 ± 0.02	99.32±0.03 80.36±0.03	оом ООМ	ООМ	99.00±0.03 81.02±0.03	98.51 ± 0.22 69.9 ± 0.22	$\frac{99.65 \pm 0.01}{85.59 \pm 0.01}$	99.56±0.02 84.23±0.02
Average Rank F	AUC F1-score	3.67 3.73	2.93 3.27	3.33 4.93	5.80 6.33	7.73 8.27	7.27 7.40	6.53 7.33	6.27 5.33	8.87	5.33 5.80	3.60 3.20
Best/Worst F	AUC F1-score	1/0 <u>4/0</u>	3/0	0/0	0/0	0/1	1/2 0/3	1/2 0/2	0/0	1/8	2/0	3/0

Table 1: Mean \pm std. results of 11 models on different classification datasets. The <u>bold</u> indicates the top result; *OOM* represents there exists GPU overflow.

configuration. The results of the comparison are summarized in Table. 1, Table. 2, and Fig. 4.

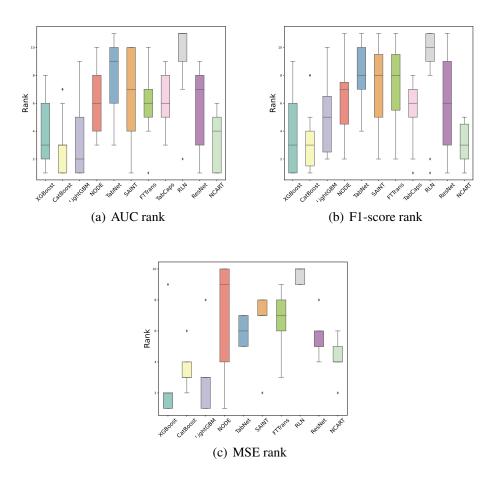


Figure 4: Rank values of different models on 20 datasets. Fig. (a) and (b) are for classification tasks and Fig. (c) is for regression tasks.

Referring to Table 1, Table 2, and Figure 4, it is evident that tree-based models retain their advantages. When we look at the AUC, CatBoost stands out with the best average performance. LightGBM and NCART also perform well, achieving the same number of top results, although their average ranks are slightly higher than CatBoost's. In contrast, a shift in focus towards the MSE highlights XG-Boost's superiority, both in terms of best score and average performance. Furthermore, when considering performance through F1-score, NCART emerges as the

top-performing model in terms of average performance. In addition to its elevated average performance, NCART competes strongly with XGBoost in achieving optimal results based on the F1-score.

NODE and TabCaps demonstrate competitive or superior performance on the majority of datasets. NODE excels in terms of the AUC, while TabCaps performs better in F1-score. Nevertheless, it's important to note that NODE's performance diminishes notably when confronted with high-dimensional regression datasets, like the superconduct dataset. Additionally, when handling the year dataset, which is large-scale and high-dimensional, NODE necessitates significantly longer training time. TabNet has proven to be a competitive model in cases where the sampling number is large, which matches the highlight in the original paper. However, its performance decreases considerably when the sample size is small. Therefore, its effectiveness can be limited based on the size of the dataset. SAINT struggles to effectively handle small-scale datasets and requires additional GPU memory when confronted with high-dimensional features, as observed in the context of datasets such as scene and EMNIST. Simultaneously, FT-Transformer demonstrates competitive performance, yet it also requires more resources to deal with high-dimensional data. Although RLN is suitable for all kinds of datasets and can obtain the two best results, it also produces some of the worst scores, indicating a higher variability in their results. ResNet proves to be a highly effective baseline, particularly when dealing with large-scale datasets. NCART, a variant of fully-connected ResNet, surpasses ResNet's performance across a wide range of datasets and delivers superior results comparable to those achieved by other deep learning models.

Additionally, our experiments indicate that deep learning models can achieve comparable or competitive results in classification tasks across datasets of varying sizes. However, when it comes to regression tasks, tree-based methods consistently demonstrate superior performance. This discrepancy can be attributed to the discrete nature of tabular datasets and the non-smooth characteristics of the target function. On the contrary, deep learning models tend to be biased towards excessively smooth solutions, which may hinder their performance in regression tasks [11].

To further substantiate this conclusion, we employ two distinct datasets. The first, analcatdata, comprises categorical features and features only 10 distinct target values, rendering its target distribution discrete. This discrete nature makes it more amenable to tree models, as they excel at approximating piecewise constant functions. In contrast, kin8nm, a dataset stemming from a realistic simulation of forward dynamics, features a continuous target distribution due to the continu-

ous nature of dynamics. Consequently, deep learning models tend to outperform tree-based models on this dataset.

Model	analcatdata	kin8nm	superconduct	house_sales	year	Average Rank	Best/Worst
XGBoost	0.0051±0.0018	0.0137±0.0003	86.19±1.71	0.0281±0.0012	74.76±0.60	3.0	2/0
CatBoost	0.0052±0.0017	0.0092 ± 0.0002	105.70 ± 7.61	$0.0286{\pm}0.0015$	76.74 ± 0.65	3.6	0/0
LightGBM	0.0054±0.0018	0.0111 ± 0.0004	85.17±1.72	$\underline{0.0281 {\pm} 0.0008}$	76.60 ± 0.55	3.2	<u>2/0</u>
NODE	0.0091±0.0017	$\underline{0.0042 {\pm} 0.0001}$	1350.31±39.79	$0.1327 {\pm} 0.0050$	TimeOut	6.8	1/2
TabNet	0.0273±0.0161	0.0100 ± 0.0014	141.46 ± 10.60	$0.0556 {\pm} 0.0119$	$82.03{\pm}4.49$	6.0	0/0
SAINT	0.2988±0.0129	$0.0044 {\pm} 0.0001$	199.16±4.32	$0.1248 {\pm} 0.0038$	94.12±0.99	6.4	0/0
FTTrans	0.3034±0.0121	0.0049 ± 0.0004	225.70 ± 12.77	$0.1058 {\pm} 0.0041$	86.37±2.12	6.6	0/0
RLN	0.3219±0.0434	$0.0347 {\pm} 0.0050$	315.75 ± 44.06	$0.3676 {\pm} 0.1020$	$125.81\!\pm\!1.04$	9.6	0/3
ResNet	0.0126±0.0060	0.0063 ± 0.0004	157.02 ± 3.93	$0.1295\; {\pm} 0.0042$	77.09 ± 1.06	5.6	0/0
NCART	0.0140±0.0026	0.0080 ± 0.0007	$125.81\!\pm\!10.66$	0.0371 ± 0.0020	75.06 ± 0.84	4.2	0/0

Table 2: Mean \pm std. results of 10 models on different regression datasets. The **bold** indicates the top result; *TimeOut* means the running time exceeds the time limit.

4.3. Inference time

We also analyze the average inference time, which is given in Fig. 5, Fig. 6, Fig. 7, and Table. 3, of the 5-fold cross-validation with the best parameters of different models in comparison to their performance. Detailed results are presented in Appendix D. Besides, we also present the training time in Appendix E

Compared to evaluation scores, tree models have bigger advantages in inference time. When working with a small dataset, most deep learning models produce similar results, except for the SAINT model. And they can outperform XG-Boost and CatBoost. However, as the dataset size increases, the computational cost of various models, including NODE, TabNet, SAINT, and RLN, escalates rapidly. Among deep learning models, TabCaps is the most efficient model, boasting the fastest total computational time for classification tasks. Although NCART demonstrates slightly lower performance compared to TabCaps, it maintains the smallest average rank among deep learning models for both classification and regression tasks. ResNet and FT-Transformer also deliver exceptional results in terms of total running time, but their average ranks fall short of competitiveness.

It is indeed surprising that deep learning models exhibit better inference times compared to tree-based methods on two classification datasets, plants-margin and EMNIST, which each contain more than 40 object classes. This is because current GBDT methods are designed for single output, if they want to handle multiple

outputs, they need to construct multiple decision trees first and then concatenate the predictions of all trees to obtain multiple outputs. When the output dimension is high, this strategy is not efficient [45]. Instead, neural networks are more flexible and they just need to adjust the number of neurons in the last layer to adapt to any output dimension. For networks with numerous parameters, such small modification has little impact on the inference time. The computation time of deep learning is more affected by the network structure and dataset size. The structure determines the speed of convergence, while the amount of data influences the number of iterations needed in the training process. These two examples highlight the potential of deep learning models to efficiently handle complex classification tasks with large numbers of classes.

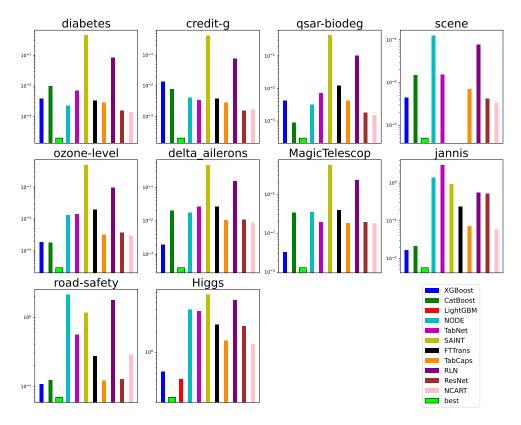


Figure 5: Average inference time(s) of a five cross-validation with the best hyperparameters for binary classification tasks. Subfigures are sorted by dataset size. The value on the y-axis represents the inference time. Missing areas indicate that the model has GPU memory overflow.

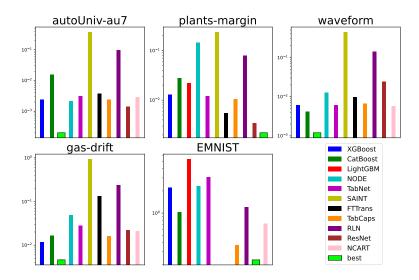


Figure 6: Average inference time(s) of a five cross-validation with the best hyperparameters for multi-class classification tasks. Subfigures are sorted by dataset size. The value on the y-axis represents the inference time. Missing areas indicate that the model has GPU memory overflow.

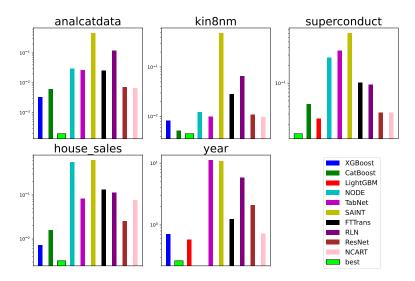


Figure 7: Average inference time(s) of a five cross-validation with the best hyperparameters for regression tasks. Subfigures are sorted by dataset size. The value on the y-axis represents the inference time. Missing areas indicate that the model's running time exceeds the time limit.

Dataset	XGBoost	CatBoost	LightGBM	NODE	TabNet	SAINT	FTTrans	TabCaps	RLN	ResNet	NCART
				C	Classificatio	on					
Average Rank	4.27	5.07	2.07	7.47	7.27	10.4	7.6	4.4	9.67	4.4	3.47
Best/Worst	0/0	1/0	12/0	0/0	0/1	0/8	0/2	0/0	0/5	1/0	1/0
Total time	2.9278	<u>1.5584</u>	5.8944	10.2912	10.6376	≫14.3193	≫4.0526	2.0984	6.0161	3.3365	2.4406
Regression											
Average Rank	2.2	2.8	<u>1.4</u>	8.4	6.6	9.6	7.0	-	7.6	5.2	4.2
Best/Worst	<u>8/0</u>	1/0	7/0	0/9	3/3	0/8	0/2	-	1/0	0/0	0/0
Total time	0.7450	0.3352	0.6097	≫0.8593	11.6860	13.0290	1.5171	-	6.1394	2.1789	0.8495

Table 3: Average inference time(s) of a five cross-validation with the best hyperparameters. The **bold** indicates the top result; - indicates that the model can not be applied to the task type; \gg means the model can not produce results, due to issues such as GPU overflow or reaching a time limit, when dealing with some datasets.

4.4. Interpretability

In both practical applications and research, understanding the importance of individual features in machine learning is paramount, particularly when dealing with tabular data. This knowledge empowers decision-makers to enhance model performance and allocate resources judiciously. Significantly, in fields like finance and healthcare, where marginal improvements in predictions hold substantial real-world implications.

In this subsection, we utilize four interpretable models, namely XGBoost, LightGBM, CatBoost, and NCART, to explore the feature importance in a medical dataset diabetes. Fig. 8 illustrates the importance scores obtained from these models, revealing variations in their assessment of feature importance.

These differences in feature importance primarily stem from the distinct structures and methodologies employed by each model. XGBoost, LightGBM, and CatBoost are ensemble methods based on GBDT, while the neural network operates on a different paradigm. The ensemble methods construct decision trees using diverse subsets of data and apply various randomization techniques, leading to discrepancies in their individual feature importance rankings. Additionally, the models' unique hyperparameters, such as learning rates, tree depths, and the number of trees, significantly influence the final importance scores for each feature. On the other hand, the neural network, with its layers, neurons, and activation functions, approaches feature importance from a different perspective.

Despite these differences in feature importance, all four models provide varying degrees of interpretability. This interpretability enables us to gain valuable insights into the factors driving their predictions, making them useful tools for un-

derstanding the relationships between features and target variables in the dataset.

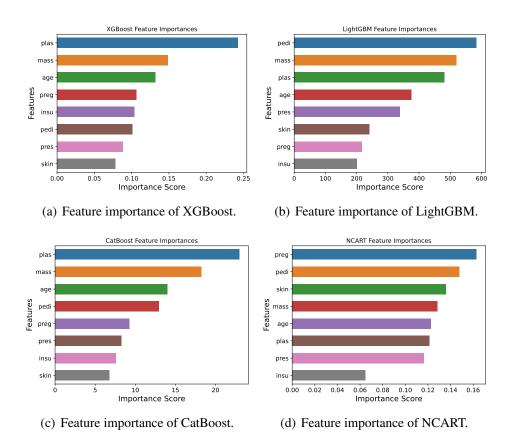


Figure 8: Feature importance of three boosting models and NCART on diabetes dataset.

4.5. Limitations

While NCART exhibits competitive performance in terms of numerical results compared to GBDT, there are several limitations to be considered:

- NCART can achieve results that are competitive with GBDT, but there is still a noticeable performance gap, particularly when it comes to regression tasks.
- NCART can compete with GBDT in terms of inference speed on small datasets, but its performance declines noticeably when working with larger datasets.

- One notable limitation of NCART is its inability to handle missing values. This can be a significant drawback in real-world datasets, where missing data is common, as it necessitates pre-processing steps to impute missing values before using NCART.
- NCART is more sensitive to hyperparameters in comparison to traditional tree models. Tuning hyperparameters becomes crucial to achieving optimal performance, but this process can be time-consuming and require expertise.

5. Conclusions

The performance of deep learning models in handling tabular data has been relatively restricted when compared to other machine learning techniques. Challenges such as limited data availability, high computational costs, and interpretability issues have posed significant obstacles to the widespread adoption of deep learning in real-life applications.

In this paper, we introduce NCART, a novel interpretable neural network that overcomes these challenges by incorporating differentiable decision trees within the ResNet architecture. By fusing decision trees with deep learning, our model combines the interpretability of decision trees with the powerful capabilities of deep neural networks.

NCART's simplicity in architecture makes it well-suited for datasets of varying sizes, while its efficiency leads to reduced computational expenses in comparison to state-of-the-art deep learning models. Our extensive numerical experiments reveal that NCART achieves competitive performance across datasets of different sizes, offering valuable insights into underlying patterns and relationships.

The introduction of NCART represents a significant step forward in addressing the limitations of deep learning in handling tabular data, opening up new possibilities for valuable applications in diverse scenarios. With its interpretability, efficiency, and performance, NCART paves the way for enhanced utilization of deep learning techniques in practical data analysis and decision-making tasks.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China no. 12071190.

Appendix A. Sparse Function

We denote the d-probability simplex (the set of vectors representing probability distributions over d choices) by $\Delta^d := \{\mathbf{p} \in \mathbb{R}^d : \mathbf{p} \geq \mathbf{0}, \|\mathbf{p}\|_1 = 1\}.$

sparsemax is an alternative to softmax which tends to yield sparse probability distributions:

$$sparsemax(\mathbf{z}) := argmin_{\mathbf{p} \in \Delta^d} \|\mathbf{p} - \mathbf{z}\|^2.$$

entmax is a more general sparse function that has the following expression:

$$\alpha - entmax(\mathbf{z}) := argmin_{\mathbf{p} \in \Delta^d} \mathbf{p}^\mathsf{T} \mathbf{z} + H_{\alpha}^T(\mathbf{p}).$$

Here, $H_{\alpha}^{T}(\mathbf{p})$ is called $Tsallis\alpha - entropie$ which is defined as:

$$H_{\alpha}^{T}(\mathbf{p}) = \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_{j=1}^{d} (p_{j} - p_{j}^{\alpha}) & \alpha \neq 1 \\ -\sum_{j=1}^{d} p_{j} \log p_{j} & \alpha = 1, \end{cases}$$

where p_j is the j_{th} component of **p**. It's worth noting that 1 - entmax is equivalent to the *softmax* function, and 2 - entmax is equal to the *sparsemax* function.

Appendix B. Datasets Description

Table. B.4 are the datasets used in this paper, the column #Target means the number of distinct values in the label, and the column Objective gives a brief introduction to the objectives of each dataset. The selection of data follows the following principles: 1. Diversity of feature types; 2. Diversity of feature dimensions; 3. Diversity of sample numbers.

Appendix C. Optimization of hyperparameters

Table. C.5 lists the search range of hyperparameters, which refers to the survey [9] and some adjustments are made on this basis to try to avoid GPU memory overflow and to lower the risk of overfitting on small-scale datasets. We implement the NCART model ⁴ using PyTorch and employ the official open-source

⁴The codes will be released to GitHub after acceptance.

Dataset	#Samples	#Num.Feat.	#Cat.Feat.	#Target	Data id	Objective	
			Bina	ry Classificat	ion		
diabetes	768	8	0	2	37	Patient's diabetes status recognition	
credit-g	1000	20	13	2	31	People's credit risks prediction	
qsar-biodeg	1055	41	0	2	1494	Analyze molecule biodegradation and structure	
scene	2407	299	5	2	312	Scene recognition	
ozone-level	2534	72	0	2	1487	Ozone level alarm forecasting	
delta_ailerons	7129	5	0	2	803	Aileron control prediction for aircraft	
MagicTelescop	13376	10	0	2	44125	Gamma signal recognition	
jannis	57580	54	0	2	44131	For challenging machine learning competitions	
road-safety	111762	32	3	2	44161	Personal injury road accidents recognition	
Higgs	940160	24	0	2	44129	Higgs boson signals recognition	
			Multic	lass Classific	ation		
autoUniv-au7	700	12	4	3	1553	An advanced dataset generator for classification	
plants-margin	1600	64	0	100	1491	Plant leaf classification	
waveform	5000	40	0	3	60	Waves prediction	
gas-drift	13910	128	0	6	1476	Gas sensor drift prediction	
EMNIST	131600	784	0	47	41039	Handwritten character digits recognition	
				Regression			
analcatdata	4052	7	5	10	44055	For analyzing categorical data	
kin8nm	8192	8	0	8188	189	Predict the dynamics of a robot arm	
superconduct	21263	79	0	3007	44148	The critical temperature prediction	
house_sales	21613	17	2	4028	44066	House sale prices prediction	
year	515345	90	0	89	44027	Song release year prediction	

Table B.4: Details of datasets (sorted by dataset size in each task type). #Target in regression task means the number of unique values.

implementations for other models ⁵ 6 7 8 9 10 11 12 13.

Appendix D. Inference Time

The average inference time(s) of a five cross-validation with the best hyperparameters are listed in Table D.6.

⁵XGBoost: https://xgboost.readthedocs.io/en/stable/

⁶CatBoost: https://catboost.ai/

 $^{^7} LightGBM: \verb|https://lightgbm.readthedocs.io/en/latest/|$

⁸NODE: https://github.com/Qwicen/node

⁹TabNet: https://github.com/dreamquark-ai/tabnet

¹⁰ SAINT: https://github.com/somepago/saint

¹¹FT-Transformer & ResNet: https://github.com/Yura52/rtdl

¹²TabCaps: https://github.com/WhatAShot/TabCaps

¹³RLN: https://github.com/irashavitt/regularization_learning_networks

HyperParameters	Range	HyperParameters	Range
		XGBoost	
num_boost_round	1000	early_stopping_rounds	10
max_depth	LogUniformInt [2, 10]	alpha	LogUniform [1e-8, 0.1]
lambda	LogUniform [0.5, 2]	eta	LogUniform [0.05, 0.3]
		CatBoost	
iterations	1000	od_wait	10
max_depth	LogUniformInt [2, 10]	l2_leaf_reg	LogUniform [0.1, 2]
learning_rate	LogUniform [0.05, 0.3]		
]	LightGBM	
iterations	1000	early_stopping_round	10
num_leaves	LogUniformInt [8, 64]	$lambda_l_1$	LogUniform [1e-8, 0.1]
$lambda_l_2$	LogUniform [1e-8, 0.1]	learning_rate	LogUniform [0.05, 0.3]
		NODE	
num_layers	[2, 4, 8]	total_tree_count	[128, 256]
tree_depth	[4, 6, 8]	tree_out put_dim	[2, 3]
		TabNet	
n_d	LogUniform [8, 16]	n_steps	UniformInt [1, 6]
gamma	Uniform [1, 2]	n_independent	UniformInt [1, 5]
n_shared	UniformInt [1, 5]	momentum	LogUniform [0.01, 0.4]
mask_type	[sparsemax, entmax]		
		SAINT	
dim	[16, 32, 64]	depth	[1, 2, 3, 6]
heads	[2, 4, 8]	dropout	[0, 0.1, 0.2, 0.3, 0.4, 0.5]
	FT	-Transformer	
num_blocks	UniformInt [1, 6]	num_tokens	[8, 16, 24, 32, 64, 128, 256, 512]
dropout_att	[0, 0.1, 0.2, 0.3, 0.4, 0.5]	dropout_ffn	[0, 0.1, 0.2, 0.3, 0.4, 0.5]
dropout_res	[0, 0.1, 0.2, 0.3, 0.4, 0.5]		
		TabCaps	
learning_rate	UniformInt [0.02, 0.1]	num_senior_capsules	UniformInt[1, 5]
primary_capsule_size	UniformInt[64, 128]	num_primary_capsules	UniformInt[4, 32]
senior_capsule_size	UniformInt[4, 32]	num_learnable_words	UniformInt[16, 64]
		RLN	
layers	UniformInt [2, 6]	theta	UniformInt [-12, -8]
norm	[1, 2]		
		ResNet	
n_blocks	UniformInt [1, 10]	d_hidden	[32, 64, 128, 256, 512]
dropout	[0, 0.1, 0.2, 0.3, 0.4, 0.5]		-
		NCART	
N in Eq. (9)	[8, 16, 32, 64]	d in Eq. (4)	UniformInt [2, 10]
		h in Eq. (5)	

Table C.5: Hyperparameters space.

Dataset	XGBoost	CatBoost	LightGBM	NODE	TabNet	SAINT	FTTrans	TabCaps	RLN	ResNet	NCART
				Binary	Classificat	ion					
diabetes	0.0039	0.0098	0.0002	0.0023	0.0070	0.4502	0.0033	0.0029	0.0854	0.0016	0.0014
credit-g	0.0140	0.0077	0.0002	0.0042	0.0034	0.4367	0.0039	0.0029	0.0756	0.0016	0.0017
qsar-biodeg	0.0042	0.0009	0.0003	0.0032	0.0071	0.4305	0.0121	0.0043	0.1015	0.0018	0.0015
scene	0.0045	0.0148	0.0005	0.1279	0.0156	OOM	OOM	0.0071	0.0785	0.0042	0.0034
ozone-level	0.0019	0.0018	0.0003	0.0131	0.0143	0.4957	0.0196	0.0032	0.0964	0.0037	0.0030
delta_ailerons	0.0020	0.0206	0.0004	0.0178	0.0266	0.4656	0.0266	0.0105	0.1564	0.0108	0.0086
MagicTelescop	0.0033	0.0336	0.0013	0.0354	0.0196	0.5677	0.0398	0.0183	0.2344	0.0192	0.0181
jannis	0.0165	0.0211	0.0057	1.3936	3.0272	0.9445	0.2409	0.0716	0.5579	0.5238	0.0584
road-safety	0.1075	0.1217	0.0693	2.1546	0.5648	1.1658	0.2741	0.1197	1.7843	0.1259	0.2867
Higgs	0.5332	0.2298	0.4163	4.0186	3.8084	6.5720	2.4771	1.4561	5.4842	2.3597	1.3039
				Multicla	ss Classific	ation					
autoUniv-au7	0.0024	0.0154	0.0002	0.0021	0.0031	0.3752	0.0037	0.0024	0.0983	0.0014	0.0029
plants-margin	0.0130	0.0276	0.0221	0.1441	0.0119	0.2342	0.0055	0.0105	0.0793	0.0034	0.0022
waveform	0.0061	0.0041	0.0012	0.0126	0.0061	0.4448	0.0098	0.0067	0.1406	0.0244	0.0057
gas-drift	0.0116	0.0166	<u>0.0046</u>	0.0492	0.0281	0.9341	0.1340	0.0160	0.2388	0.0219	0.0209
EMNIST	2.2039	1.0328	5.3717	2.3124	3.0943	OOM	OOM	0.3663	1.2044	0.2330	0.7222
				R	egrssion						
analcatdata	0.0033	0.0062	0.0002	0.0294	0.0269	0.4621	0.0256	-	0.1183	0.0071	0.0067
kin8nm	0.0083	0.0052	0.0045	0.0121	0.0099	0.4858	0.0282	-	0.0649	0.0110	0.0096
superconduct	0.0139	0.0445	0.0249	0.2725	0.3562	0.7072	0.1020	-	0.0938	0.0316	0.0316
house_sales	0.0072	0.0159	0.0032	0.5453	0.0812	0.6169	0.1300	-	0.1115	0.0255	0.0759
year	0.7122	0.2633	0.5768	TimeOut	11.2118	10.7570	1.2313	-	5.7785	2.1037	0.7257

Table D.6: Average inference time(s) of a five cross-validation with the best hyperparameters. The **bold** indicates the top result; - indicates that the model can not be applied to the task type; *OOM* represents there exists GPU overflow; *TimeOut* means the running time exceeds the time limit.

Appendix E. Training Time

The detailed training time(s) of five cross-validation with the best hyperparameters are shown in Table E.7.

References

- [1] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.
- [2] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

Dataset	XGBoost	CatBoost	LightGBM	NODE	TabNet	SAINT	FTTrans	TabCaps	RLN	ResNet	NCART
				Bina	ary Classific	cation					
diabetes	0.08	1.08	0.11	32.85	0.13	156.10	1.09	0.74	24.22	1.99	5.79
credit-g	0.18	4.19	0.16	84.10	<u>0.07</u>	73.99	6.28	0.39	16.08	2.44	7.18
qsar-biodeg	0.13	3.21	0.18	20.84	0.14	199.24	29.71	2.02	46.89	1.39	13.34
scene	0.44	3.48	<u>0.21</u>	193.95	14.73	OOM	OOM	2.13	34.73	1.63	13.16
ozone-level	<u>0.11</u>	1.91	0.24	211.53	2.15	298.27	70.45	1.30	4.98	3.23	8.26
delta_ailerons	<u>0.06</u>	2.95	0.17	346.49	50.27	89.17	91.79	1.81	5.11	10.97	15.22
MagicTelescop	0.29	10.61	0.38	369.31	31.19	180.62	18.59	22.92	59.22	20.43	19.19
jannis	1.31	18.75	<u>0.86</u>	500.89	684.32	169.47	155.88	37.78	83.88	48.70	36.45
road-safety	3.87	5.85	<u>3.82</u>	1000.29	1306.03	443.30	805.73	135.45	25.47	192.86	132.75
Higgs	<u>6.63</u>	9.63	10.73	954.56	8209.51	1162.72	2914.61	1022.71	599.21	1015.46	1424.64
				Multi	class Classi	fication					
autoUniv-au7	0.20	1.01	0.18	30.11	<u>0.06</u>	136.29	7.06	0.33	41.37	1.11	4.13
plants-margin	6.53	10.05	6.66	405.50	81.59	103.81	39.45	27.11	4.60	3.94	4.85
waveform	1.19	2.30	<u>0.45</u>	28.52	9.57	129.35	4.24	2.38	46.18	4.34	9.48
gas-drift	<u>2.10</u>	6.68	6.81	1145.76	41.46	455.32	131.10	8.73	33.51	14.81	35.15
EMNIST	251.56	207.63	249.05	2649.74	1536.03	OOM	OOM	304.60	<u>98.74</u>	99.62	288.04
Average Rank	1.93	3.93	2.20	9.53	6.73	10.13	8.20	4.80	7.13	5.07	6.47
Best/Worst	<u>7/0</u>	0/0	4/0	0/4	2/3	0/9	0/3	0/0	1/0	1/0	0/0
Total time	<u>274.67</u>	289.33	280.00	7974.43	11967.27	≫4420.14	≫4276.10	1570.42	1164.22	1422.93	1991.12
					Regression	1					
analcatdata	0.31	1.10	0.15	97.77	7.66	42.17	9.39	-	30.32	6.41	11.49
kin8nm	1.69	4.13	1.43	210.32	108.63	323.10	29.08	-	39.24	15.71	14.98
superconduct	2.46	13.37	16.24	858.54	132.95	569.11	279.68	-	81.50	28.59	44.02
house_sales	0.81	5.07	0.64	1744.48	132.61	387.82	70.98	-	169.44	22.45	36.68
year	14.22	8.25	12.92	TimeOut	3036.32	3916.41	1805.23	-	120.02	1594.27	690.45
Average Rank	2.0	2.4	1.6	9.6	8.0	8.6	6.8	-	6.2	4.8	5.0
Best/Worst	1/0	1/0	<u>3/0</u>	0/3	0/1	0/1	0/0	-	0/0	0/0	0/0
Total time	<u>19.48</u>	31.91	31.37	≫2911.10	3580.22	5238.61	2686.11	-	401.14	1667.43	789.83

Table E.7: Average training time(s) of a five cross-validation with the best hyperparameters. The **bold** indicates the top result; - indicates that the model can not be applied to the task type; *OOM* represents there exists GPU overflow; *TimeOut* means the running time exceeds the time limit.

- [3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems 30 (2017).
- [4] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, Advances in neural information processing systems 31 (2018).
- [5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015)

436-444.

- [6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
- [8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (11) (2020) 139–144.
- [9] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, G. Kasneci, Deep neural networks and tabular data: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022).
- [10] Y. Gorishniy, I. Rubachev, V. Khrulkov, A. Babenko, Revisiting deep learning models for tabular data, Advances in Neural Information Processing Systems 34 (2021) 18932–18943.
- [11] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, in: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022.
- [12] Y. Yang, I. G. Morillo, T. M. Hospedales, Deep neural decision trees, arXiv preprint arXiv:1806.06988 (2018).
- [13] S. Popov, S. Morozov, A. Babenko, Neural oblivious decision ensembles for deep learning on tabular data, arXiv preprint arXiv:1909.06312 (2019).
- [14] L. Katzir, G. Elidan, R. El-Yaniv, Net-dnf: Effective deep modeling of tabular data, in: International Conference on Learning Representations, 2021.
- [15] V. Zantedeschi, M. Kusner, V. Niculae, Learning binary decision trees by argmin differentiation, in: International Conference on Machine Learning, PMLR, 2021, pp. 12298–12309.

- [16] S. Ö. Arik, T. Pfister, Tabnet: Attentive interpretable tabular learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 6679–6687.
- [17] X. Huang, A. Khetan, M. Cvitkovic, Z. Karnin, Tabtransformer: Tabular data modeling using contextual embeddings, arXiv preprint arXiv:2012.06678 (2020).
- [18] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, T. Goldstein, Saint: Improved neural networks for tabular data via row attention and contrastive pre-training, arXiv preprint arXiv:2106.01342 (2021).
- [19] J. Kossen, N. Band, C. Lyle, A. N. Gomez, T. Rainforth, Y. Gal, Self-attention between datapoints: Going beyond individual input-output pairs in deep learning, Advances in Neural Information Processing Systems 34 (2021) 28742–28756.
- [20] N. Hollmann, S. Müller, K. Eggensperger, F. Hutter, Tabpfn: A transformer that solves small tabular classification problems in a second, arXiv preprint arXiv:2207.01848 (2022).
- [21] J. Chen, J. Yan, D. Z. Chen, J. Wu, Excelformer: A neural network surpassing gbdts on tabular data, arXiv preprint arXiv:2301.02819 (2023).
- [22] J. Yan, J. Chen, Y. Wu, D. Z. Chen, J. Wu, T2g-former: Organizing tabular features into relation graphs promotes heterogeneous feature interaction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 10720–10728.
- [23] Q. Zheng, Z. Peng, Z. Dang, L. Zhu, Z. Liu, Z. Zhang, J. Zhou, Deep tabular data modeling with dual-route structure-adaptive graph networks, IEEE Transactions on Knowledge and Data Engineering 01 (2023) 1–13.
- [24] I. Shavitt, E. Segal, Regularization learning networks: deep learning for tabular datasets, Advances in Neural Information Processing Systems 31 (2018).
- [25] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st workshop on deep learning for recommender systems, 2016, pp. 7–10.

- [26] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: a factorization-machine based neural network for ctr prediction, arXiv preprint arXiv:1703.04247 (2017).
- [27] G. Ke, Z. Xu, J. Zhang, J. Bian, T.-Y. Liu, Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 384–394.
- [28] R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, E. Chi, Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems, in: Proceedings of the web conference 2021, 2021, pp. 1785–1797.
- [29] B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, J. Dong, Supertml: Two-dimensional word embedding for the precognition on structured tabular data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.
- [30] P. Yin, G. Neubig, W.-t. Yih, S. Riedel, Tabert: Pretraining for joint understanding of textual and tabular data, arXiv preprint arXiv:2005.08314 (2020).
- [31] G. Ke, J. Zhang, Z. Xu, J. Bian, T.-Y. Liu, Tabnn: A universal neural network solution for tabular data (2018).
- [32] J. Chen, K. Liao, Y. Wan, D. Z. Chen, J. Wu, Danets: Deep abstract networks for tabular data classification and regression, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, 2022, pp. 3930–3938.
- [33] J. Chen, K. Liao, Y. Fang, D. Chen, J. Wu, Tabcaps: A capsule neural network for tabular data classification with bow routing, in: The Eleventh International Conference on Learning Representations, 2022.
- [34] J. Yoon, Y. Zhang, J. Jordon, M. van der Schaar, Vime: Extending the success of self-and semi-supervised learning to tabular domain, Advances in Neural Information Processing Systems 33 (2020) 11033–11043.
- [35] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, Autogluon-tabular: Robust and accurate automl for structured data, arXiv preprint arXiv:2003.06505 (2020).

- [36] P. N. Suganthan, On non-iterative learning algorithms with closed-form solution, Applied Soft Computing 70 (2018) 1078–1082.
- [37] Y.-H. Pao, G.-H. Park, D. J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
- [38] Q. Shi, R. Katuwal, P. N. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, Pattern Recognition 117 (2021) 107978.
- [39] Q. Shi, M. Hu, P. N. Suganthan, R. Katuwal, Weighting and pruning based ensemble deep random vector functional link network for tabular data classification, Pattern Recognition 132 (2022) 108879.
- [40] L. Rokach, O. Maimon, Decision trees, Data mining and knowledge discovery handbook (2005) 165–192.
- [41] A. Martins, R. Astudillo, From softmax to sparsemax: A sparse model of attention and multi-label classification, in: International conference on machine learning, PMLR, 2016, pp. 1614–1623.
- [42] B. Peters, V. Niculae, A. F. T. Martins, Sparse sequence-to-sequence models, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2019, pp. 1504–1519.
- [43] A. Veit, M. J. Wilber, S. Belongie, Residual networks behave like ensembles of relatively shallow networks, Advances in neural information processing systems 29 (2016).
- [44] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
- [45] Z. Zhang, C. Jung, Gbdt-mo: gradient-boosted decision trees for multiple outputs, IEEE transactions on neural networks and learning systems 32 (7) (2020) 3156–3167.