Learning Robust XGBoost Ensembles for Regression Tasks

Atri Vivek Sharma[1, 2] **Panagiotis Kouvaros**[*2] **Alessio Lomuscio**[2,1]

[1]Department of Computing, Imperial College London, UK
[2]Safe Intelligence, UK

Abstract

Methods to improve the adversarial robustness of tree-based ensemble models for classification tasks have received significant attention in recent years. In this work, we propose a novel method for training robust tree-based boosted ensembles. The method leverages the XGBoost framework and is applicable to any task that employs a differentiable loss function. It particularly introduces an analytical solution to the upper-bound of the robust loss function, that can be computed in constant time, enabling the construction of robust splits without sacrificing computational efficiency. Although our method is general, we focus its application on regression tasks, extending conventional regression metrics to better quantify model robustness. An extensive evaluation on 19 regression datasets from a widely-used tabular data benchmark demonstrates that in the face of adversarial perturbations in the input space, our proposed method results in ensembles that are up to 44% more robust compared to the present SoA and 113% more robust than the conventional XGBoost model when considering norm bounded attacks of radius 0.05.

1 INTRODUCTION

Adversarial examples, small perturbations in the vicinity of correctly classified inputs that result in misclassification, have been widely documented in the literature [Szegedy et al., 2013, Goodfellow et al., 2014]. Although mostly studied in the context of neural networks, research has demonstrated that decision-tree ensembles are also susceptible to adversarial perturbations [Papernot et al., 2016]. Such vulnerabilities are particularly concerning in safety-critical applications, where robust model performance is essential for deployment [Li and Li, 2020].

Several methods have been put forward to mitigate the impact of adversarial examples in the context of neural networks. Some approximate the worst-case loss under adversarial perturbations to be used under training [Goodfellow et al., 2014, Madry et al., 2018]. Others aim to identify a provable upper bound to the loss function under adversarial perturbations [Salman et al., 2019, Zhang et al., 2019, Huang et al., 2021, Müller et al., 2023, De Palma et al., 2024]

Similar approaches have been proposed for decision-tree ensembles, particularly with respect to the derivation of robust splits and the minimisation of the worst-case adversarial loss within the ensemble building process [Kantchelian et al., 2016, Chen et al., 2019, Andriushchenko and Hein, 2019, Vos and Verwer, 2021, Guo et al., 2022]. A common theme among these works is the utilisation of specific properties of specific loss functions, such as the gini-impurity [Vos and Verwer, 2021], margin-based classification loss functions [Andriushchenko and Hein, 2019], and the binary loss function [Guo et al., 2022]. The applicability of these approaches is limited to classification tasks only, thereby failing to address other tasks such as regression prevalent in domains such as finance. To the best of our knowledge, only the method described in [Chen et al., 2019] supports robust training for general loss functions, however, this method relies on a heuristic to estimate the loss under an adversarial perturbation, which can lead to suboptimal robustness. Therefore, the derivation of approaches for robust learning of ensembles with particular applicability to regression tasks remains largely unexplored.

To overcome these shortcomings, we present a novel approach to construct robust ensembles in the XGBoost framework that can be generally applied across various tasks and loss functions, with particular applicability to regression tasks. Our contributions can be summarised as follows:

[*]This work was conducted between November 2024 and April 2025 while Panagiotis Kouvaros was also affiliated with the University of Limassol.

- We introduce an efficient analytical solution to the upper bound of the robust loss when building XGBoost trees. This incorporates the impact of worst-case adversarial perturbations in the recursive node splitting procedure in $\mathcal{O}(1)$ time, leading to an overall complexity of $\mathcal{O}(n \log n)$. Our solution is general and can be applied to XGBoost ensembles with any loss function, and adversarial attack model.
- We study the robustness of XGBoost ensembles in the regression task, and highlight how conventionally trained models are extremely sensitive to input perturbations. We demonstrate that our proposed robust-splitting criterion significantly improves the robustness of the model, particularly on attacks with large perturbation magnitudes.

The rest of this work is organised as follows. We begin with a review of related work in Section 2. Section 3 provides an overview of the XGBoost algorithm and the adversarial attack model used in this work. Section 4 describes our proposed robust-splitting criterion; Section 5 evaluates the approach and compares it with related work on a variety of datasets. Finally, Section 6 concludes the work and discusses potential future directions.

2 RELATED WORK

Early seminal work in the robust training of tree-based models considered the augmentation of adversarial examples into the training process in successive boosting rounds [Kantchelian et al., 2016]. The work also demonstrated that identifying optimal adversarial examples under input constraints is NP-hard for an ensemble of trees, and provides a MILP formulation solution to exactly compute the minimal adversarial distance for a boosted ensemble. Although their training method is computationally efficient, it relies on sampling a finite number of adversarial examples when constructing trees, which is equivalent to optimizing a lower bound of the robust loss and ultimately results in suboptimal certified robustness. In contrast, our approach optimizes a certified upper bound of the robust loss, thereby achieving superior robustness.

More recent contributions aimed to increase the robustness of tree-based models by considering the worst-case adversarial loss when evaluating the quality of a candidate split. In particular, Chen et al. [2019] presents a framework to determine the worst-case splitting score for a candidate split with inputs perturbed with an L_∞ perturbation. It specifically gives a gini-impurity scoring function that can be computed exactly in $\mathcal{O}(n)$ time, and further presents a heuristic to estimate the worst-case adversarial loss for any scoring function in $\mathcal{O}(1)$ time. The method is integrated within the XGBoost algorithm, and is used to train robust decision trees and boosted ensembles. However, since the heuristic effectively provides only a lower bound on the robust loss, it can lead to suboptimal robustness during tree construction. An extension to the framework is provided by Calzavara et al. [2020], which considers an asymmetrical, non-uniform attack model characterized by axis-aligned perturbations and introduces the concept of an attack budget, limiting the number of points that can be perturbed. In this work, the attack model is integrated into a robust-loss function that is computed exactly per split. The complexity of this method is thus very high and is not scalable to large datasets. In contrast to these approaches, our method computes an upper bound to the robust loss in $\mathcal{O}(1)$ time which leads to improved robustness without compromising computational efficiency.

Similarly, Vos and Verwer [2021] presents an exact analytical solution for the worst-case gini-impurity score under an adversarial perturbation that can be computed in $\mathcal{O}(1)$ time. The work proves in particular that the robust loss function used in the split construction process is concave and can thus be solved analytically. However, the solution is specific to the gini-impurity score and the construction of independent classification trees. Therefore, it cannot be used to build boosted ensembles or trees for other tasks.

Other methods have been proposed to determine an upper bound on the robust loss over boosted ensembles and to train successive trees that minimize this loss. In particular, Andriushchenko and Hein [2019] introduces a method for obtaining an upper bound on the robust loss for binary classification tasks using a margin-based loss function over the ensemble, with a computational complexity of $\mathcal{O}(n)$. This approach is further optimized in Guo et al. [2022], where a 0-1 loss function is employed for AdaBoost ensembles [Freund and Schapire, 1997], resulting in a robust loss computation per split in $\mathcal{O}(1)$ time. However, these formulations are tailored specifically for classification tasks and are not easily extended to other domains. Moreover, the optimization in Guo et al. [2022] exploits unique properties of the 0-1 loss function, rendering the approach inapplicable to other convex loss functions. In contrast, our method provides an analytical solution for an upper bound on the robust loss that can be applied to any loss function, thereby offering greater flexibility and broader applicability across various tasks.

Similarly to Chen et al. [2019], our work targets the computation of robust splitting scores. However, instead of considering an approximate heuristic to estimate the robust loss, we propose an efficient analytical solution that computes an upper bound on the worst-case loss in $\mathcal{O}(1)$ time using a linear relaxation formulation of the splitting score used in an XGBoost tree. As the XGBoost algorithm constructs successive trees to minimise a second order Taylor approximation of the loss function, our method supports all differentiable loss functions and can be applied across various tasks.

Other related works propose more computationally intensive

approaches, such as MILP formulations for robust optimal trees [Vos and Verwer, 2022a], and post-training methods to prune and relabel tree leaves [Vos and Verwer, 2022b], to achieve robustness. However, due to their exponential or high polynomial time complexity, these methods are impractical for large-scale datasets compared to our approach.

3 BACKGROUND

In this section we fix the notation of the paper, recall gradient boosted ensembles and the XGBoost algorithm, and outline adversarial robustness and robust training for tree-based models.

Gradient Boosted Ensembles. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}\,(|\mathcal{D}| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, be a dataset with m features and n datapoints. Gradient boosting [Friedman, 2001] is the process of sequentially adding weak learners to an ensemble of learners to minimise a certain loss function on $\mathcal{D}$. The prediction $\hat{y}_i$ of an ensemble F with K weak learners is

$$\hat{y}_i = F(\mathbf{x}_i) = \sum_{k=1}^{K} f_k\left(\mathbf{x}_i\right), \tag{1}$$

where each f_k is an independent weak learner. The total loss of the ensemble at iteration t is

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(\mathbf{x}_i\right)\right), \tag{2}$$

where l is an arbitrary, differentiable loss function. In general, while any model can be used as a weak learner, decision trees are often chosen because of their expressivity and ease of training. In this work, we consider greedily trained binary trees with coordinate aligned splits of the form $f_t(\mathbf{x_i}) = w_{q(\mathbf{x_i})}$, where $q(\mathbf{x_i})$ is the tree traversal function that maps an input $\mathbf{x_i}$ to a leaf node with value $w_{q(\mathbf{x_i})}$. The tree traversal function $q_{\mathbf{x_i}}$ is learned using a greedy algorithm that recursively splits the input space to minimise the loss function.

The XGBoost algorithm. The XGBoost algorithm [Chen and Guestrin, 2016] is a popular implementation of gradient boosting that is widely used in various tasks because of its efficiency and scalability. The algorithm constructs new weak learners by optimising a second-order Taylor approximation of the loss function. For their construction, it introduces regularisation to penalise complex tree structures and large leaf node values, thereby weakening overfitting. Concretely, the loss function from Equation 2 is approximated as

$$\begin{aligned}\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} \Big[& l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(\mathbf{x}_i\right) \\ & + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\Big] + \Omega\left(f_t\right),\end{aligned} \tag{3}$$

where g_i and h_i are the first and second-order gradient statistics on the loss function, and $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$ is the regularisation term with hyperparameters γ and λ used to respectively control the penalisation of the number of leaf nodes T and leaf node values w. Using this approximate loss formulation, the optimal value of a leaf z can be computed as:

$$w_z^* = -\frac{\sum_{i \in I_z} g_i}{\sum_{i \in I_z} h_i + \lambda}, \tag{4}$$

where I_z is the set of indices of datapoints that reach leaf z. The algorithm recursively identifies the best series of splits to minimise the loss function. In the exact greedy algorithm, each feature value is considered as a potential split. The split with the greatest loss reduction, or split score $\mathcal{S}$, is selected. The split score is a function of the threshold value η and the feature index j. When splitting a parent node containing the set of training data points $\mathcal{I}$ into left and right child nodes, containing the set of points $\mathcal{I}_L$ and $\mathcal{I}_R$ respectively, we express the split score as a function of $\mathcal{I}_L$ and $\mathcal{I}_R$ as:

$$\begin{aligned}\mathcal{S}(\mathcal{I}_L, \mathcal{I}_R) = \frac{1}{2}\Bigg[& \frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} \\ & - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda}\Bigg] - \gamma.\end{aligned} \tag{5}$$

Robust Learning for Boosted Ensembles. Adversarial examples are imperceptible perturbations to a correctly predicted input that cause incorrect predictions by the model in question. Consider an input region $\Delta_\epsilon(\mathbf{x})$ centered around a datapoint $\mathbf{x}$ with a radius of ϵ under the L_∞ norm:

$$\Delta_\epsilon(\mathbf{x}) = \left\{\mathbf{x}' \in \mathbb{R}^m \mid \|\mathbf{x}' - \mathbf{x}\|_\infty \leq \epsilon\right\}. \tag{6}$$

Given an input $\mathbf{x}$, an adversarial example $\mathbf{x}'$ can be computed as:

$$\mathbf{x}' = \underset{\mathbf{x}' \in \Delta_\epsilon(\mathbf{x})}{\operatorname{argmax}} \{L\left(F\left(\mathbf{x}'\right), y\right)\}. \tag{7}$$

Thus, an ensemble can be trained to be robust against adversarial examples by minimising the loss under the worst-case adversarial perturbation for each training example, as formulated by Madry et al. [2018]:

$$\min_F \sum_{i=1}^{n} \max_{\mathbf{x}' \in \Delta_\epsilon(\mathbf{x_i})} L\left(F\left(\mathbf{x}'\right), y_i\right). \tag{8}$$

In tandem with the minimisation of this adversarial loss, the construction of a weak learner at iteration t of gradient boosting requires the optimisation of the following loss function:

$$\mathcal{L}_{rob}^{(t)} = \sum_{i=1}^{n} \max_{\mathbf{x}' \in \Delta_\epsilon(\mathbf{x}_i)} l\left(y_i, \sum_{k=1}^{(t-1)} f_k\left(\mathbf{x}'\right) + f_t\left(\mathbf{x}'\right)\right). \tag{9}$$

4 ROBUST XGBOOST TREES

In this section we introduce a robust splitting criterion which we incorporate into the XGBoost algorithm. Differently from conventional approaches, which target the minimisation of the loss function with respect to the training data, our method targets the robustness of the derived trees to adversarial perturbations. This is achieved by integrating an analytical upper bound of the adversarial loss within the recursive splitting procedure of the individual decision trees.

4.1 ROBUST SPLITTING IN XGBOOST TREES

At the core of our robust training procedure is a splitting criterion that modifies Equation 5 to incorporate the worst-case adversarial loss within the recursive splitting procedure of individual decision trees. Instead of simply considering static sets of points $\mathcal{I}_L$ and $\mathcal{I}_R$ for the left and right child nodes, this new formulation additionally contains the ambiguity set $\Delta\mathcal{I}$, which contains all data points that could change child nodes under an adversarial perturbation. We here consider perturbations within an L_∞ ball centred around a training data point (7). When using axis-aligned splits, the computation of the worst-case robust loss function with respect to these perturbations can be computed by treating each feature independently. In particular, when splitting on feature j, perturbations on other features have no impact on which child node a data point ends up in; therefore only perturbations of $\pm\epsilon$ along feature j need to be considered.

To define and analyse the robust splitting criterion, we borrow the following notation from Chen et al. [2019] on various sets of data points for a given split with a threshold η on feature j.

Table 1: Definitions of all sets used in the robust splitting criterion, when considering a split on feature j with threshold η, under an L_∞ perturbation of radius ϵ.

Notation	Definition
$\mathcal{I}$	Set of examples in the node being split
$\mathcal{I}_L$	$\mathcal{I} \cap \{(\mathbf{x_i}, y_i) \mid x_i^{(j)} \leq \eta\}$
$\mathcal{I}_R$	$\mathcal{I} \cap \{(\mathbf{x_i}, y_i) \mid x_i^{(j)} > \eta\}$
$\Delta\mathcal{I}$	$\mathcal{I} \cap \{(\mathbf{x_i}, y_i) \mid \eta - \epsilon < x_i^{(j)} \leq \eta + \epsilon\}$
$\mathcal{I}_L^0$	$\mathcal{I}_L \setminus \Delta\mathcal{I}$
$\mathcal{I}_R^0$	$\mathcal{I}_R \setminus \Delta\mathcal{I}$

Intuitively, the ambiguity set $\Delta\mathcal{I}$ contains all points that could switch child nodes under an adversarial perturbation. In the context of an L_∞ attack model, this essentially corresponds to all points that are within a distance of ϵ from the threshold η. The sets $\mathcal{I}_L^0$ and $\mathcal{I}_R^0$ contain all points that are further than ϵ from the threshold η and are thus guaranteed to remain in the left and right child nodes respectively under any perturbation.

The robust splitting criterion can then be defined as:

$$\begin{aligned}\mathcal{S}_{\text{rob}}(\mathcal{I}_L^0, \mathcal{I}_R^0, \Delta\mathcal{I}) = \min_{r_i} \frac{1}{2}\Bigg[& \frac{\left(\sum_{i\in\mathcal{I}_L^0} g_i + \sum_{i\in\Delta\mathcal{I}} r_i g_i\right)^2}{\sum_{i\in\mathcal{I}_L^0} h_i + \sum_{i\in\Delta\mathcal{I}} r_i h_i + \lambda} \\ & + \frac{\left(\sum_{i\in\mathcal{I}_R^0} g_i + \sum_{i\in\Delta\mathcal{I}} (1-r_i) g_i\right)^2}{\sum_{i\in\mathcal{I}_R^0} h_i + \sum_{i\in\Delta\mathcal{I}} (1-r_i) h_i + \lambda} \\ & - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\Bigg] - \gamma,\end{aligned} \tag{10}$$

where r_i is a binary variable that indicates whether a datapoint i in the ambiguity set $\Delta\mathcal{I}$ moves to the left child node. Computing the optimal value of r_i is a combinatorial optimisation problem with exponential complexity, and thus computationally intractable to solve for $\mathcal{O}(\|\mathcal{I}\|m)$ candidate splits.

We can instead derive a lower bound to the problem by considering a linear relaxation on the binary variables r_i:

$$\begin{aligned}\mathcal{S}_{\text{rob}}(\mathcal{I}_L^0, \mathcal{I}_R^0, \Delta\mathcal{I}) \geq \mathcal{S}_{\text{rob}}^{\text{lb}} = \min_{p,q} \frac{1}{2}\Bigg[& \frac{\left(\sum_{i\in\mathcal{I}_L^0} g_i + p\right)^2}{\sum_{i\in\mathcal{I}_L^0} h_i + q + \lambda} \\ & + \frac{\left(\sum_{i\in\mathcal{I}_R^0} g_i + \sum_{i\in\Delta\mathcal{I}} g_i - p\right)^2}{\sum_{i\in\mathcal{I}_R^0} h_i + \sum_{i\in\Delta\mathcal{I}} h_i + \lambda} \\ & - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\Bigg] - \gamma,\end{aligned} \tag{11}$$

where p and q are continuous variables that represent the sum of the first and second derivatives of the points from the ambiguity set that move to the left child node. This relaxation results in a continuous optimisation problem with an analytical solution that can be computed in constant time. The lower bound for the robust split score given by this solution can be used to upper bound the robust loss function and can therefore be used to evaluate candidate splits towards optimising the latter upper bound.

4.2 TIGHTENING THE LINEAR RELAXATION

The previously described linear relaxation of the robust splitting criterion is generally quite loose because the optimal values for p and q may not satisfy the binary constraints imposed on r_i. This discrepancy can lead to substantial under-approximation errors in the robust split scores, effectively diminishing the distinction between high-quality and low-quality candidate splits. As a result, the decision tree may fail to identify effective splits, ultimately compromising its predictive performance.

To alleviate this shortcoming, we now tighten the relaxation by introducing box constraints around the minimum and maximum values of p and q. In particular, we observe the following:

- The first derivative of the loss function can be positive or negative, therefore the minimum value of p is the sum of all negative elements in the set $\{g_i \mid i \in \Delta\mathcal{I}\}$, and the maximum value of p is the sum of all positive elements in the set $\{g_i \mid i \in \Delta\mathcal{I}\}$. Hence, $p \in \left[\sum_{i\in\Delta\mathcal{I}} \min(0, g_i), \sum_{i\in\Delta\mathcal{I}} \max(0, g_i)\right]$.
- The second derivative of any convex loss function is always positive, therefore $q \in \left[0, \sum_{i\in\Delta\mathcal{I}} h_i\right]$.

While the box constraints greatly tighten the linear relaxation, they do not capture the combinatorial nature of the binary variables r_i. We can further tighten the relaxation by approximating the feasible region of the values p and q that are consistent with the points in the ambiguity set moving between the left and right child nodes. In particular, we aim to capture the constraint that if a point i moves to the left node, then *both* the first and the second derivatives of the point must contribute to the sums p and q, and vice versa. This can be achieved by considering the maximum and minimum values of the first and second derivatives of the points in the ambiguity set.

To define the constraints we introduce some preliminary notation. Let u be an auxiliary variable that denotes the number of points in the ambiguity set that move to the left child node. Let $g_{\text{min}}^{(\Delta\mathcal{I})}$, $g_{\text{max}}^{(\Delta\mathcal{I})}$, $h_{\text{min}}^{(\Delta\mathcal{I})}$ and $h_{\text{max}}^{(\Delta\mathcal{I})}$ be the minimum and maximum values of the first and second derivatives of the points in the ambiguity set, respectively. Furthermore, let $G^{(\Delta\mathcal{I})}$ and $H^{(\Delta\mathcal{I})}$ be the sums of the first and second derivatives of the points in the ambiguity set. These can be used to construct linear constraints between p, q and u as follows:

$$\begin{aligned} g_{\text{min}}^{(\Delta\mathcal{I})} \cdot u \leq p \leq g_{\text{max}}^{(\Delta\mathcal{I})} \cdot u, \\ h_{\text{min}}^{(\Delta\mathcal{I})} \cdot u \leq q \leq h_{\text{max}}^{(\Delta\mathcal{I})} \cdot u, \\ g_{\text{min}}^{(\Delta\mathcal{I})} \left(|\Delta\mathcal{I}| - u\right) \leq G^{(\Delta\mathcal{I})} - p \leq g_{\text{max}}^{(\Delta\mathcal{I})} \left(|\Delta\mathcal{I}| - u\right), \\ h_{\text{min}}^{(\Delta\mathcal{I})} \left(|\Delta\mathcal{I}| - u\right) \leq H^{(\Delta\mathcal{I})} - q \leq h_{\text{max}}^{(\Delta\mathcal{I})} \left(|\Delta\mathcal{I}| - u\right). \end{aligned} \tag{12}$$

Solving for the auxiliary variable in these inequalities results in the following linear constraints between p and q:

$$\begin{aligned} p \leq q \cdot c_1, \\ p \geq q \cdot c_2, \\ p \geq G^{(\Delta\mathcal{I})} - c_1(H^{(\Delta\mathcal{I})} - q), \\ p \leq G^{(\Delta\mathcal{I})} - c_2(H^{(\Delta\mathcal{I})} - q), \end{aligned} \tag{13}$$

where the parameters c_1 and c_2 are defined as follows:

$$c_1 = \begin{cases} \frac{g_{\text{max}}^{(\Delta\mathcal{I})}}{h_{\text{min}}^{(\Delta\mathcal{I})}}, & g_{\text{max}}^{(\Delta\mathcal{I})} \geq 0, \\ \frac{g_{\text{max}}^{(\Delta\mathcal{I})}}{h_{\text{max}}^{(\Delta\mathcal{I})}}, & g_{\text{max}}^{(\Delta\mathcal{I})} < 0, \end{cases} \quad c_2 = \begin{cases} \frac{g_{\text{min}}^{(\Delta\mathcal{I})}}{h_{\text{max}}^{(\Delta\mathcal{I})}}, & g_{\text{min}}^{(\Delta\mathcal{I})} \geq 0, \\ \frac{g_{\text{min}}^{(\Delta\mathcal{I})}}{h_{\text{min}}^{(\Delta\mathcal{I})}}, & g_{\text{min}}^{(\Delta\mathcal{I})} < 0. \end{cases} \tag{14}$$

The optimal values of p and q can be computed analytically in constant time by solving the constrained optimisation problem. The detailed analytical solution is given in Appendix B.

We have thus derived a tight formulation for the robust splitting criterion that can be used to evaluate candidate splits in the tree building procedure. The key advantages of this formulation are threefold: (i) it can be solved analytically in constant time, (ii) it can be integrated within the XGBoost algorithm to build robust gradient boosted ensembles with limited computational overhead, and (iii) it is agnostic to the choice of loss function, and can thus be used with any differentiable loss function for various tasks such as regression, classification, and ranking.

4.3 CONSTRUCTING ROBUST TREES

We now integrate the robust splitting criterion with the XGBoost algorithm. The underlying tree building procedure is modified to evaluate candidate splits using the robust splitting criterion, and select the split that maximises the robust score.

Similarly to previous work [Guo et al., 2022] [Vos and Verwer, 2021], we consider all features $j \in [m]$ and thresholds $\eta \in W_j$ as candidate splits, where

$$W_j = \bigcup_{i\in\mathcal{I}} \{x_i^{(j)} - \epsilon, x_i^{(j)}, x_i^{(j)} + \epsilon\}. \tag{15}$$

To evaluate the robust score for each candidate split using Equation 11, the values of $\sum_{i\in\mathcal{I}_L^0} g_i$, $\sum_{i\in\mathcal{I}_R^0} g_i$, $\sum_{i\in\Delta\mathcal{I}} g_i$, $\sum_{i\in\mathcal{I}_L^0} h_i$, $\sum_{i\in\mathcal{I}_R^0} h_i$, $\sum_{i\in\Delta\mathcal{I}} h_i$, $g_{\text{min}}^{(\Delta\mathcal{I})}$, $g_{\text{max}}^{(\Delta\mathcal{I})}$, $h_{\text{min}}^{(\Delta\mathcal{I})}$ and $h_{\text{max}}^{(\Delta\mathcal{I})}$ for each feature j and threshold $\eta \in W_j$ need to be computed. All thresholds are efficiently evaluated by considering a sorted W_j and maintaining running sums, minimums and maximums of the first and second derivatives for the fixed and ambiguous sets respectively. This enables the robust split score to be computed in constant time for each candidate split. The proposed algorithm iterates over a larger set of candidate splits compared to the exact XGBoost algorithm, however this presents a constant time overhead for the overall training procedure, and the exploration of the additional candidate splits leads to more robust trees in practice.

The sorting operation has a time complexity of $\mathcal{O}(n \log n)$, and the evaluation of the robust score for each candidate

split has a time complexity of $\mathcal{O}(1)$. Thus, the overall time complexity of finding an optimal split over m features is $\mathcal{O}(mn \log n)$. This is the same as the time complexity of the exact XGBoost tree building procedure. The detailed algorithm for constructing robust trees is provided in Algorithm 1.

Once an optimal split is identified, we assign leaf weights to the left and right child nodes based on the optimal values of p_η and q_η of the threshold that maximises the robust splitting criterion:

$$\begin{aligned} w_L^* &= -\frac{\sum_{i \in \mathcal{I}_L^0} g_i + p_\eta}{\sum_{i \in \mathcal{I}_L^0} h_i + q_\eta + \lambda} \\ w_R^* &= -\frac{\sum_{i \in \mathcal{I}_R^0} g_i + \sum_{i \in \Delta\mathcal{I}} g_i - p_\eta}{\sum_{i \in \mathcal{I}_R^0} h_i + \sum_{i \in \Delta\mathcal{I}} h_i - q_\eta + \lambda}. \end{aligned} \tag{16}$$

Finally, as in conventional greedy tree building procedures [Mingers, 1989] and in the XGBoost algorithm, we apply pruning to the tree once it is constructed. We start with the leaf nodes and iteratively prune the tree by removing nodes that have a robust splitting score below γ. This post-training pruning step accounts for the greediness of the building procedure which cannot guarantee that the robust loss is decreasing at successive splits. As opposed to alternative methods of early stopping [Guo et al., 2022], by pruning after training, we allow certain splits to be made that may not lead to a local loss reduction, but may have a greater loss reduction in subsequent splits.

Thus, we propose a **greedy certified** training method, that is guaranteed to minimise the upper bound of the robust loss function is minimised at each split. In principle, this should lead to significantly more robust trees than a heuristic approach to estimate the robust loss function per split.

Algorithm 1: Robust Splits for XGBoost Trees

Input: Training set $\mathcal{D} = \{(x_i, y_i)\}|_{i=1}^n$, $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$, ϵ, the radius of the L_∞ ball
Input: Node indices $\mathcal{I}$, per-instance gradients $\{g_i\}_{i \in \mathcal{I}}$, Hessians $\{h_i\}_{i \in \mathcal{I}}$, regularization parameter λ, minimum gain γ

for $j \leftarrow 1$ **to** m **do**
  for i in sorted($\mathcal{I}$, ascending by x_i^j) **do**
    for $\eta \in \{x_i^j - \epsilon, x_i^j, x_i^j + \epsilon\}$ **do**
      $\mathcal{I}_L^0 = \{(\mathbf{x_i}, y_i) | x_i^j \leq \eta - \epsilon\}$;
      $\mathcal{I}_R^0 = \{(\mathbf{x_i}, y_i) | x_i^j > \eta + \epsilon\}$;
      $\Delta\mathcal{I} = \{(\mathbf{x_i}, y_i) | \eta - \epsilon < x_i^{(j)} \leq \eta + \epsilon\}$;
      Update sums $\sum_{i \in \mathcal{I}_L^0} g_i$, $\sum_{i \in \mathcal{I}_R^0} g_i$, $\sum_{i \in \Delta\mathcal{I}} g_i$, $\sum_{i \in \mathcal{I}_L^0} h_i$, $\sum_{i \in \mathcal{I}_R^0} h_i$, $\sum_{i \in \Delta\mathcal{I}} h_i$;
      Update values $g_{\text{min}}^{(\Delta\mathcal{I})}$, $g_{\text{max}}^{(\Delta\mathcal{I})}$, $h_{\text{min}}^{(\Delta\mathcal{I})}$ and $h_{\text{max}}^{(\Delta\mathcal{I})}$;
      $\mathcal{S}_{\text{rob}}^{\text{lb}} \leftarrow$ Lower bound of robust split score from Equation 11;
      $p_\eta, q_\eta \leftarrow \operatorname{argmin} \mathcal{S}_{\text{rob}}^{\text{lb}}$;
    end
  end
end
$m^*, \eta^*, p_\eta^*, q_\eta^* \leftarrow \operatorname{argmax} \mathcal{S}_{\text{rob}}^{\text{lb}}$;

Output: Best split: feature m^*, threshold η^*, optimal values p_η^*, q_η^*

5 EXPERIMENTAL EVALUATION

We evaluate the robust training procedure introduced in the previous section by comparing it with the state-of-the-art approaches on a variety of datasets and with respect to different performance metrics.

Performance metrics. Given a sample $(\mathbf{x}, y)$, let $\hat{y}_{\text{adv}}$ denote the output of the model F under a worst-case adversarial perturbation on the input $\mathbf{x}$. For regression tasks with a mean squared error loss, $\hat{y}_{\text{adv}}$ is equal to:

$$\hat{y}_{\text{adv}} = F(\mathbf{x}') \quad \text{where} \quad \mathbf{x}' = \operatorname*{argmax}_{\mathbf{x}' \in \Delta_\epsilon(\mathbf{x})} |F(\mathbf{x}') - y| \,. \tag{17}$$

On the basis of $\hat{y}_{\text{adv}}$ we define the robust R^2 score as:

$$R^2_{\text{rob}} = 1 - \frac{\sum_i^n (y_i - \hat{y}_{\text{adv},i})^2}{\sum_i^n (y_i - \bar{y})^2}, \tag{18}$$

where $\bar{y}$ is the mean of the labels on the test dataset. This can be used to effectively summarise the lower bound of the performance over a test set under adversarial attacks of radius ϵ. Furthermore, given $\hat{y} = F(\mathbf{x})$, let $\hat{y}_{\text{dev}}$ denote the output of the model that maximises the deviation from the true output, i.e.,

$$\hat{y}_{\text{dev}} = F(\mathbf{x}') \quad \text{where} \quad \mathbf{x}' = \operatorname*{argmax}_{\mathbf{x}' \in \Delta_\epsilon(\mathbf{x})} |F(\mathbf{x}') - \hat{y}| \,. \tag{19}$$

On the basis of $\hat{y}_{\text{dev}}$, we define the robust mean absolute deviation (MAD) as:

$$\text{MAD} = \frac{1}{n} \sum_i^n |\hat{y}_i - \hat{y}_{\text{dev},i}|. \tag{20}$$

The MAD metric gives an upper bound to the average deviation in the model's predictions under L_∞ norm-bounded adversarial attacks. In the following, we use the MILP-based method originally introduced in Kantchelian et al. [2016] to compute the outputs $\hat{y}_{\text{adv}}$ and $\hat{y}_{\text{dev}}$ for each test point.

In addition to R^2_{rob} and MAD, we introduce a robust accuracy metric, analogous to those used in the literature for classification tasks. In our case, we define robust accuracy as the proportion of test data points for which the model's

output remains within a specified threshold τ of the true label under adversarial perturbations, i.e., a point is deemed robust if $|\hat{y}_{\text{adv}} - y| \leq \tau$.

To ensure consistency in our evaluation across the various datasets (which have different label ranges), we vary the threshold τ, weighted by the range of output values in the test set, i.e., $\tau = w_\tau \cdot (\max\{y\} - \min\{y\})$.

We refer to the robust accuracy metric as $\text{acc}_{\text{rob}}^{(w_\tau)}$, where, for our experiments, we consider $w_\tau \in \{0.2, 0.4\}$. The metric allows us to evaluate the robustness of a model against perturbations that cause a change in the output by a specified fraction of the range of the output values.

Baselines. We consider two baseline methods for evaluating the performance of our robust training method:

- **XGBoost** [Chen and Guestrin, 2016]: The conventional XGBoost method with the mean squared error objective.
- **Robust-GBDT** [1] [Chen et al., 2019]: The robust XGBoost training method that uses heuristics to estimate the worst-case robust loss at a split. To our knowledge, this is the only method in the literature that directly supports robust training for general loss functions.

Table 2: Average results over 19 regression datasets for the proposed robust-splitting criterion, Robust-GBDT, and XGBoost under an L_∞ adversarial attack for various ϵ values. $\frac{\text{MAD}}{y_{\text{range}}}$ refers to the mean absolute deviation of the model normalised by the range of the output values in the test set.

ϵ	Method	R^2	R^2_{rob}	$\frac{\text{MAD}}{y_{\text{range}}}$	$\text{acc}_{\text{rob}}^{(0.2)}$	$\text{acc}_{\text{rob}}^{(0.4)}$
0.005	xgboost	0.678	0.220	0.062	0.794	0.967
	RGBDT	**0.701**	0.572	0.032	0.895	0.985
	ours	0.690	**0.636**	**0.015**	**0.910**	**0.987**
0.01	xgboost	0.678	-0.186	0.093	0.730	0.929
	RGBDT	**0.697**	0.421	0.050	0.855	0.976
	ours	0.669	**0.582**	**0.023**	**0.896**	**0.986**
0.05	xgboost	**0.678**	-3.828	0.278	0.382	0.683
	RGBDT	0.632	-1.268	0.166	0.564	0.856
	ours	0.521	**0.189**	**0.049**	**0.813**	**0.966**
0.1	xgboost	**0.678**	-7.191	0.396	0.190	0.505
	RGBDT	0.565	-4.440	0.255	0.371	0.719
	ours	0.301	**-0.049**	**0.055**	**0.742**	**0.945**

Datasets and Hyperparameters. We consider 19 regression datasets from a widely used tabular data benchmark introduced in Grinsztajn et al. [2022]. This benchmark is chosen as it provides a diverse collection of real-world regression tasks, facilitating a comprehensive evaluation of the robustness of the proposed method. As our approach uses the L_∞ attack model, which is only applicable on continuous features, we consider the datasets in the benchmark that only contain continuous features.

We tune the hyperparameters of the three approaches to maximise the conventional R^2 score on the validation set of each dataset, as this approach closely mirrors the application of such models in practice. The hyperparameters for the XGBoost baseline are obtained from the same benchmark, which presents the best grid-search hyperparameters on each dataset. For the two robust methods, we conduct a grid-search over the maximum tree depth, the L_2 regularisation parameter λ, and the minimum loss reduction γ, using the hyperparameters from the XGBoost baseline as a starting point. We further limit the number of trees in the ensemble to 100 to mitigate the scalability issues associated with certifying large models with the MILP solver.

We scale all feature values in the training data to $[0, 1]$ to ensure uniformity in the perturbations across features. We conduct experiments on the datasets with a range of adversarial perturbation radii $\epsilon \in \{0.005, 0.01, 0.05, 0.1\}$. All results are obtained by averaging over a 5-fold cross-validation.

The average performance of the proposed robust-splitting criterion, Robust-GBDT, and XGBoost over the 19 regression datasets for various ϵ values is outlined in Table 2. We present more granular results across different datasets for a perturbation radius of 0.05 in Table 3. Results for other values of perturbation radii can be found in Tables 5, 6 and 7 in the appendix. The results clearly demonstrate the improved robustness of the proposed method compared to the baselines, especially when observing larger perturbation radii. The results also highlight fragilities with the conventional XGBoost models, which appear unable to maintain their performance under adversarial perturbations. Indeed, even small input perturbations lead to large deviations in the output values, completely degrading its predictive performance, as indicated by the large negative R^2_{rob} values. This strongly underscores the need for considering robustness in regression tasks.

In contrast, the Robust-GBDT method obtains much higher robustness than the conventional XGBoost model, and exhibits good performance when considering small perturbations. However, it is significantly less effective than the proposed robust-splitting criterion at higher perturbation radii, where it obtains large negative values of R^2_{rob} in some cases. The table additionally shows that the Robust-GBDT method obtains higher standard R^2 scores than the proposed method. This can be intuitively explained by the fact that Robust-GBDT uses a simple heuristic to estimate the worst-case robust loss, while our method determines a provable upper bound to the robust loss at each candidate split. This ultimately makes the proposed approach more robust at the expense of a drop in predictive performance.

Comparison with exact robust loss. In addition to the experiments conducted on the tabular data benchmark, we

[1] Code taken from https://github.com/chenhongge/RobustTrees

Table 3: Comparisons of standard and robust regression metrics for the proposed robust-splitting criterion, Robust-GBDT, and XGBoost over 19 regression benchmark datasets for an L_∞ adversarial attack of radius $\epsilon = 0.05$. The $\text{MAD}_{\text{ratio}}$ column describes the ratio of MAD obtained by the baseline compared to our approach.

Dataset	ours			RGBDT			xgboost			y_{range}	$\text{MAD}_{\text{ratio}}$	
	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD		RGBDT	XGBoost
Ailerons	0.742	**0.431**	0.000	**0.765**	0.360	0.000	0.758	0.365	0.000	0.003	-	-
Bike_Sharing_Demand	0.508	**0.049**	**31.434**	0.605	-0.422	52.997	**0.613**	-0.328	43.569	410.200	1.686	1.386
Brazilian_houses	0.745	**0.626**	**0.253**	0.838	-0.100	0.768	**0.986**	-18.806	3.507	4.776	3.036	13.862
MiamiHousing2016	0.722	**0.148**	**0.369**	0.830	-0.727	0.708	**0.865**	-0.991	0.743	3.533	1.919	2.014
abalone	0.323	**0.203**	**0.622**	**0.465**	-0.280	1.962	0.364	-0.815	2.741	22.400	3.154	4.407
cpu_act	0.979	**0.933**	**3.289**	**0.984**	0.882	5.475	0.975	-16.297	78.840	99.000	1.665	23.971
delays_zurich_transport	0.055	**0.009**	**0.120**	**0.108**	-0.154	0.619	0.101	-0.459	1.060	10.272	5.158	8.833
diamonds	0.939	**0.382**	**0.120**	0.957	-0.009	0.361	**0.966**	-2.341	0.731	2.264	3.008	6.092
elevators	0.597	**-0.090**	**0.004**	**0.823**	-5.457	0.016	0.822	-4.351	0.015	0.057	4.000	3.750
house_16H	0.255	**0.052**	**0.165**	**0.320**	-0.287	0.393	0.295	-2.199	1.037	11.112	2.382	6.285
house_sales	0.741	**0.440**	**0.193**	0.830	-0.184	0.474	**0.836**	-0.478	0.521	3.745	2.456	2.699
houses	0.750	**0.438**	**0.248**	0.788	-0.297	0.502	**0.853**	-1.740	0.794	2.671	2.024	3.202
medical_charges	-0.004	**-0.220**	**0.111**	0.412	-7.816	0.787	**0.908**	-5.596	0.748	1.852	7.090	6.739
nyc-taxi-green-dec-2016	0.003	**-0.003**	**0.005**	0.193	-3.068	1.029	**0.440**	-7.020	1.498	3.270	205.800	299.600
pol	0.943	**0.519**	**16.551**	0.969	0.376	21.964	**0.976**	0.152	27.053	100.000	1.327	1.635
sulfur	0.780	**-0.793**	**0.035**	**0.918**	-2.205	0.077	0.909	-3.372	0.090	0.861	2.200	2.571
superconduct	0.573	**0.402**	**5.242**	0.757	-1.855	40.668	**0.784**	-3.442	54.040	129.840	7.758	10.309
wine_quality	0.245	**0.063**	**0.156**	0.381	-1.305	0.830	**0.390**	-1.752	0.927	5.400	5.321	5.942
yprop_4_1	0.003	**-0.004**	**0.000**	**0.062**	-1.553	0.024	0.049	-3.253	0.041	0.149	inf	inf

evaluate the empirical performance and the tightness of the approximation of the proposed lower bound of the robust splitting criterion and the heuristic from Chen et al. [2019] against the exact robust loss, computed by exactly solving the mixed integer optimisation problem in Eq. 10 in Figure 1. We observe that our method indeed provides a principled lower bound to the exact solution, and with a much tighter approximation compared to the heuristic method.

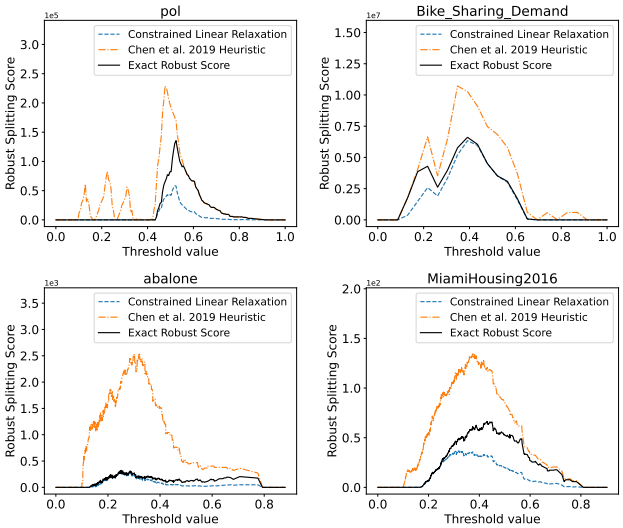

Figure 1: Comparison of robust splitting scores obtained by our method, the heuristic from Chen et al. [2019], and an exact solution to the robust splitting score from Equation 10 across threshold values, with $\epsilon = 0.1$. These scores are computed from the root node of the first tree of the ensemble, on the optimal feature for splitting.

Additionally, we compare the robustness profile of the proposed method, the exact splitting score, and the heuristic method from Chen et al. [2019] by computing and plotting the R^2_{rob} of models under varying perturbation radii ϵ. As the exact robust loss is extremely computationally expensive to compute, we limit our evaluation to 5 datasets from the tabular data benchmark. A subset of the results is presented in Figure 2 below, with the full results available at Figure 3 in the appendix. We additionally compare the performance of the heuristic method with a relaxed perturbation radius (2x the original radius) to evaluate its robustness-performance trade-off.

The empirical robustness results in Figure 2 and the tightness of the approximation shown in Figure 1 demonstrate that our proposed lower bound of the robust splitting score achieves performance closely aligned with the exact robust splitting score across diverse datasets and training radii. This suggests that the proposed lower bound yields empirically comparable models in terms of both performance and robustness. In some cases, the relaxation even leads to more robust and accurate models. Thus, employing the proposed lower bound of the robust splitting score closely approximates the models obtained by the exact robust splitting score, while being significantly more computationally efficient.

Furthermore, the proposed approach leads to models with greater robustness compared to the heuristic approach, even with a relaxed radius, obtaining a very different robustness profile with a more favourable trade-off between robustness and predictive performance, which becomes more pronounced at higher perturbation radii.

Indeed, we observe a robustness and predictive performance trade-off between the three methods evaluated. Nonethe-

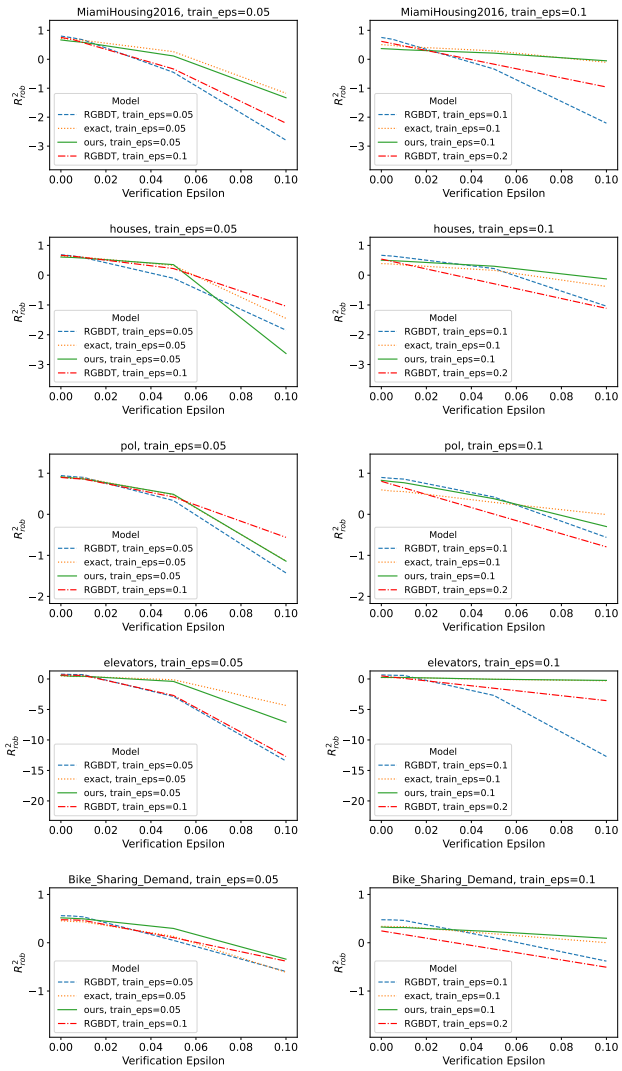

Figure 2: Comparison of the robustness profile of the models obtained by our method, the heuristic from Chen et al. [2019], the same heuristic set with a relaxed perturbation radius, and an exact solution to the robust splitting score from Equation 10.

less, we believe our proposed robust-splitting criterion is particularly compelling for applications where robustness is paramount. By delivering markedly improved empirical robustness against adversarial perturbations—especially at higher perturbation levels—our approach presents a valuable alternative to both conventional XGBoost and existing robust training methods.

6 CONCLUSIONS

In this work we have proposed methods to improve and evaluate the adversarial robustness of tree-based ensemble models in the context of regression tasks. We introduce a novel method to construct splits that are robust to adversarial perturbations in the context of the XGBoost algorithm. Our method is based on an analytical solution to the upper-bound of the Taylor approximation of the loss function typically used within XGBoost, that can be computed in constant time. This enables us to construct robust splits while maintaining the computational efficiency of the original algorithm. Our formulation is generalisable to any differentiable loss function and can thus be extended to various use-cases, including regression.

Furthermore, we proposed a series of novel metrics to quantify the robustness of regression models and evaluate the robustness of the XGBoost algorithm. Our results show that the models are highly sensitive to adversarial perturbations in the input space, which leads to significant performance degradation. Extensive experiments highlight that our proposed robust XGBoost algorithm derives models that are more robust to perturbations in the input space. We additionally observe a trade-off between robustness and predictive performance in several experiments.

Limitations and Future Work: There are some limitations in the proposed method that provide several promising directions for future work:

- The procedure considers a simplification of the robust loss function, by considering the worst-case loss per split, thereby constructing robust trees in isolation. Perturbations in the previous trees in the ensemble are not currently considered. A future work in this direction is to consider the robust loss over the entire ensemble, and build a procedure that certifiably minimises the robust loss over the complete ensemble.

- The current work primarily focusses on L_∞ norm adversarial attacks on numerical features, which limits its applicability on mixed and categorical data which are common in tabular datasets. As the proposed linear relaxation approach is highly general (and only requires the creation of an ambiguity set), it is extensible to other types of data and adversarial attacks. This presents a promising direction for future work to explore a framework for creating ambiguity sets for categorical and mixed data types, as well as for other types of adversarial attacks.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive feedback and suggestions. This work was partly supported by UK Research and Innovation [grant number EP/S023356/1], in the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence. Alessio Lomuscio is partly supported by the Royal Academy of Engineering via a Chair of Emerging Technologies.

References

M. Andriushchenko and M. Hein. Provably robust boosted decision stumps and trees against adversarial attacks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS'19)*. Curran Associates, Inc., 2019.

S. Calzavara, C. Lucchese, G. Tolomei, S. A. Abebe, and S. Orlando. Treant: Training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, 34(5): 1390–1420, 2020.

H. Chen, H. Zhang, D. Boning, and C.-J. Hsieh. Robust decision trees against adversarial examples. In *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, pages 1122–1131. PMLR, 2019.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, pages 785–794. ACM, 2016.

A. De Palma, R. Bunel, K. Dvijotham, M. P. Kumar, R. Stanforth, and A. Lomuscio. Expressive losses for verified robustness via convex combinations. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*. Openreview.net, 2024.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS'22)*, pages 507–520. Curran Associates, Inc., 2022.

J.-Q. Guo, M.-Z. Teng, W. Gao, and Z.-H. Zhou. Fast provably robust decision trees and boosting. In *Proceedings of the International Conference on Machine Learning (ICML'22)*, pages 8127–8144. PMLR, 2022.

Y. Huang, H. Zhang, Y. Shi, Z. Kolter, and A. Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS'21)*, pages 22745–22757. Curran Associates, Inc., 2021.

A. Kantchelian, J. D. Tygar, and A. D. Joseph. Evasion and hardening of tree ensemble classifiers. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*, pages 2387–2396. JMLR.org, 2016.

D. Li and Q. Li. Adversarial deep ensemble: Evasion attacks and defenses for malware detection. *IEEE Transactions on Information Forensics and Security*, 15:3886–3900, 2020.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*, 2018.

J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

M. N. Müller, F. Eckert, M. Fischer, and M. Vechev. Certified training: Small boxes are all you need. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*, 2023.

N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS'19)*. Curran Associates, Inc., 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

D. Vos and S. Verwer. Efficient training of robust decision trees against adversarial examples. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, pages 10586–10595. PMLR, 2021.

D. Vos and S. Verwer. Robust optimal classification trees against adversarial examples. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*, volume 36, pages 8520–8528. AAAI Press, 2022a.

D. Vos and S. Verwer. Adversarially robust decision tree relabeling. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'22)*, pages 203–218. Springer, 2022b.

H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, pages 7472–7482. PMLR, 2019.

Learning Robust XGBoost Ensembles for Regression Tasks
Supplementary Material

Atri Vivek Sharma[1, 2] **Panagiotis Kouvaros**[†2] **Alessio Lomuscio**[2,1]

[1]Department of Computing, Imperial College London, UK
[2]Safe Intelligence, UK

A EXPERIMENTS

A.1 HYPERPARAMETERS USED

The hyperparameters for the conventional XGBoost models are detailed in Table 4, obtained from the grid-search conducted in the tabular data benchmark [Grinsztajn et al., 2022]. The hyperparameters for the robust XGBoost models were further tuned using these as a starting point.

Table 4: Details of the hyperparameters used in the experiments, obtained from the hyperparameter tuning conducted in the tabular data benchmark presented in Grinsztajn et al. [2022]. The number of trees was restricted to 100 for all experiments due to computational constraints of the MILP based solver used to compute the robust metrics.

dataset	No. of Trees	Max Depth	γ	Learning Rate	Minimum samples per Leaf	λ
cpu_act	100	9	0.0011	0.0272	5	2.978
pol	100	11	0.0000	0.0272	4	2.525
elevators	100	7	0.0000	0.0407	18	1.297
wine_quality	100	11	0.0000	0.0118	2	1.417
Ailerons	100	7	0.0000	0.0219	77	1.515
houses	100	10	0.0001	0.0319	2	3.303
house_16H	100	11	0.0002	0.0044	3	3.183
diamonds	100	5	0.0003	0.0635	4	2.056
Brazilian_houses	100	4	0.0000	0.0722	1	3.204
Bike_Sharing_Demand	100	9	0.0000	0.0143	3	1.660
nyc-taxi-green-dec-2016	100	3	0.1054	0.2163	2	2.335
house_sales	100	11	0.2247	0.0222	1	2.168
sulfur	100	6	0.0000	0.1523	2	1.469
medical_charges	100	4	0.0236	0.0289	14	1.516
MiamiHousing2016	100	9	0.0310	0.0314	2	3.226
superconduct	100	10	0.0066	0.0173	15	1.497
yprop_4_1	100	11	0.0007	0.0119	1	1.224
abalone	100	10	0.0000	0.0062	29	1.415
delays_zurich_transport	100	10	0.0130	0.0061	24	3.043

[†]This work was conducted between November 2024 and April 2025 while Panagiotis Kouvaros was also affiliated with the University of Limassol.

A.2 ADDITIONAL RESULTS

This section outlines the detailed results of the experiments described in Section 5 for the L_∞ adversarial attacks of radii $\epsilon = 0.005$, $\epsilon = 0.01$ and $\epsilon = 0.1$.

Table 5: Comparisons of standard and robust regression metrics for an L_∞ adversarial attack of radius $\epsilon = 0.005$.

Dataset	ours			RGBDT			xgboost			y_{range}	$\text{MAD}_{\text{ratio}}$	
	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD		RGBDT	XGBoost
Ailerons	0.800	0.791	0.000	0.786	0.779	0.000	0.758	0.751	0.000	0.003	-	-
Bike_Sharing_Demand	0.706	0.697	1.122	0.710	0.707	0.414	0.613	0.608	0.557	410.200	0.369	0.496
Brazilian_houses	0.935	0.876	0.154	0.944	0.668	0.396	0.986	-0.997	1.052	4.776	2.571	6.831
MiamiHousing2016	0.876	0.827	0.067	0.878	0.805	0.089	0.865	0.771	0.100	3.533	1.328	1.493
abalone	0.482	0.405	0.369	0.496	0.399	0.437	0.364	0.198	0.597	22.400	1.184	1.618
cpu_act	0.986	0.978	0.934	0.981	0.971	1.152	0.975	0.951	1.801	99.000	1.233	1.928
delays_zurich_transport	0.137	0.133	0.011	0.134	0.077	0.171	0.101	-0.047	0.388	10.272	15.545	35.273
diamonds	0.962	0.960	0.005	0.965	0.949	0.035	0.966	0.907	0.108	2.264	7.000	21.600
elevators	0.866	0.842	0.000	0.869	0.825	0.001	0.822	0.781	0.001	0.057	inf	inf
house_16H	0.382	0.283	0.105	0.401	0.248	0.128	0.295	-0.985	0.682	11.112	1.219	6.495
house_sales	0.848	0.821	0.035	0.853	0.806	0.056	0.836	0.749	0.087	3.745	1.600	2.486
houses	0.810	0.757	0.059	0.838	0.570	0.188	0.853	0.435	0.237	2.671	3.186	4.017
medical_charges	0.813	0.729	0.067	0.906	0.649	0.139	0.908	0.625	0.121	1.852	2.075	1.806
nyc-taxi-green-dec-2016	0.304	0.127	0.124	0.357	-0.174	0.265	0.440	-0.592	0.435	3.270	2.137	3.508
pol	0.986	0.974	1.784	0.985	0.961	2.786	0.976	0.945	2.968	100.000	1.562	1.664
sulfur	0.897	0.830	0.008	0.909	0.813	0.012	0.909	0.546	0.014	0.861	1.500	1.750
superconduct	0.815	0.682	5.573	0.817	0.542	9.779	0.784	-0.876	30.349	129.840	1.755	5.446
wine_quality	0.429	0.319	0.105	0.432	0.230	0.176	0.390	0.136	0.194	5.400	1.676	1.848
yprop_4_1	0.067	0.050	0.000	0.062	0.038	0.001	0.049	-0.719	0.015	0.149	inf	inf

Table 6: Comparisons of standard and robust regression metrics for an L_∞ adversarial attack of radius $\epsilon = 0.01$.

Dataset	ours			RGBDT			xgboost			y_{range}	$\text{MAD}_{\text{ratio}}$	
	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD		RGBDT	XGBoost
Ailerons	0.800	0.753	0.000	0.785	0.723	0.000	0.758	0.703	0.000	0.003	-	-
Bike_Sharing_Demand	0.714	0.704	1.053	0.714	0.699	1.882	0.613	0.591	2.460	410.200	1.787	2.336
Brazilian_houses	0.906	0.843	0.174	0.936	0.711	0.323	0.986	-2.579	1.459	4.776	1.856	8.385
MiamiHousing2016	0.865	0.758	0.120	0.878	0.716	0.161	0.865	0.648	0.186	3.533	1.342	1.550
abalone	0.470	0.355	0.477	0.493	0.304	0.730	0.364	0.053	0.999	22.400	1.530	2.094
cpu_act	0.985	0.974	1.195	0.981	0.961	1.875	0.975	-1.275	12.555	99.000	1.569	10.506
delays_zurich_transport	0.129	0.125	0.013	0.133	0.070	0.176	0.101	-0.094	0.501	10.272	13.538	38.538
diamonds	0.960	0.941	0.016	0.965	0.922	0.068	0.966	0.843	0.156	2.264	4.250	9.750
elevators	0.858	0.814	0.001	0.871	0.817	0.001	0.822	0.765	0.001	0.057	1.000	1.000
house_16H	0.335	0.204	0.119	0.392	0.196	0.182	0.295	-1.161	0.738	11.112	1.529	6.202
house_sales	0.838	0.795	0.053	0.854	0.764	0.094	0.836	0.657	0.145	3.745	1.774	2.736
houses	0.802	0.701	0.099	0.827	0.375	0.270	0.853	0.058	0.370	2.671	2.727	3.737
medical_charges	0.682	0.547	0.088	0.899	0.257	0.226	0.908	0.204	0.215	1.852	2.568	2.443
nyc-taxi-green-dec-2016	0.241	0.044	0.130	0.354	-1.261	0.620	0.440	-1.396	0.639	3.270	4.769	4.915
pol	0.983	0.949	3.782	0.984	0.942	4.080	0.976	0.923	4.340	100.000	1.079	1.148
sulfur	0.888	0.724	0.012	0.876	0.589	0.019	0.909	0.099	0.026	0.861	1.583	2.167
superconduct	0.783	0.592	7.229	0.804	0.208	16.294	0.784	-1.458	37.067	129.840	2.254	5.128
wine_quality	0.395	0.187	0.175	0.426	-0.017	0.322	0.390	-0.164	0.356	5.400	1.840	2.034
yprop_4_1	0.070	0.042	0.001	0.071	0.018	0.001	0.049	-0.951	0.018	0.149	1.000	18.000

Table 7: Comparisons of standard and robust regression metrics for an L_∞ adversarial attack of radius $\epsilon = 0.1$.

Dataset	ours			RGBDT			xgboost			y_{range}	$\text{MAD}_{\text{ratio}}$	
	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD	R^2	R^2_{rob}	MAD		RGBDT	XGBoost
Ailerons	0.562	0.025	0.000	0.726	-0.399	0.000	0.758	-0.431	0.000	0.003	-	-
Bike_Sharing_Demand	0.274	-0.664	46.494	0.578	-1.781	86.108	0.613	-1.726	78.848	410.200	1.852	1.696
Brazilian_houses	0.615	0.307	0.375	0.718	-1.457	1.184	0.986	-20.624	3.674	4.776	3.157	9.797
MiamiHousing2016	0.283	-0.162	0.251	0.816	-3.269	1.202	0.865	-3.275	1.191	3.533	4.789	4.745
abalone	0.111	0.042	0.299	0.331	-0.241	1.555	0.364	-1.346	3.532	22.400	5.201	11.813
cpu_act	0.964	0.862	5.382	0.981	0.746	8.559	0.975	-16.530	79.775	99.000	1.590	14.823
delays_zurich_transport	-0.004	-0.004	0.001	0.072	-0.145	0.487	0.101	-0.649	1.315	10.272	487.000	1315.000
diamonds	0.002	-0.019	0.099	0.928	-7.332	1.628	0.966	-9.919	1.857	2.264	16.444	18.758
elevators	0.313	-0.222	0.003	0.800	-16.935	0.029	0.822	-16.461	0.028	0.057	9.667	9.333
house_16H	0.165	-0.036	0.101	0.271	-0.683	0.526	0.295	-3.748	1.371	11.112	5.208	13.574
house_sales	0.476	0.201	0.216	0.805	-1.201	0.705	0.836	-1.797	0.775	3.745	3.264	3.588
houses	0.614	0.153	0.303	0.789	-1.222	0.726	0.853	-3.219	1.039	2.671	2.396	3.429
medical_charges	-0.290	-0.297	0.021	-0.115	-13.729	0.942	0.908	-13.566	1.135	1.852	44.857	54.048
nyc-taxi-green-dec-2016	0.007	-0.056	0.055	0.155	-1.003	0.474	0.440	-13.888	2.193	3.270	8.618	39.873
pol	0.850	-0.164	29.583	0.946	-0.982	46.064	0.976	-1.444	54.829	100.000	1.557	1.853
sulfur	0.452	-0.904	0.038	0.878	-29.388	0.310	0.909	-16.217	0.219	0.861	8.158	5.763
superconduct	0.206	0.033	5.157	0.640	-2.533	43.676	0.784	-4.386	60.570	129.840	8.469	11.745
wine_quality	0.121	-0.027	0.140	0.361	-2.104	1.054	0.390	-3.173	1.288	5.400	7.529	9.200
yprop_4_1	0.000	-0.003	0.000	0.060	-0.703	0.014	0.049	-4.222	0.047	0.149	inf	inf

A.3 ROBUSTNESS PROFILE

This section outlines the complete results of the comparison between the robustness profiles of the proposed method, to the exact robust splitting criterion, and the heuristic presented in Chen et al. [2019].

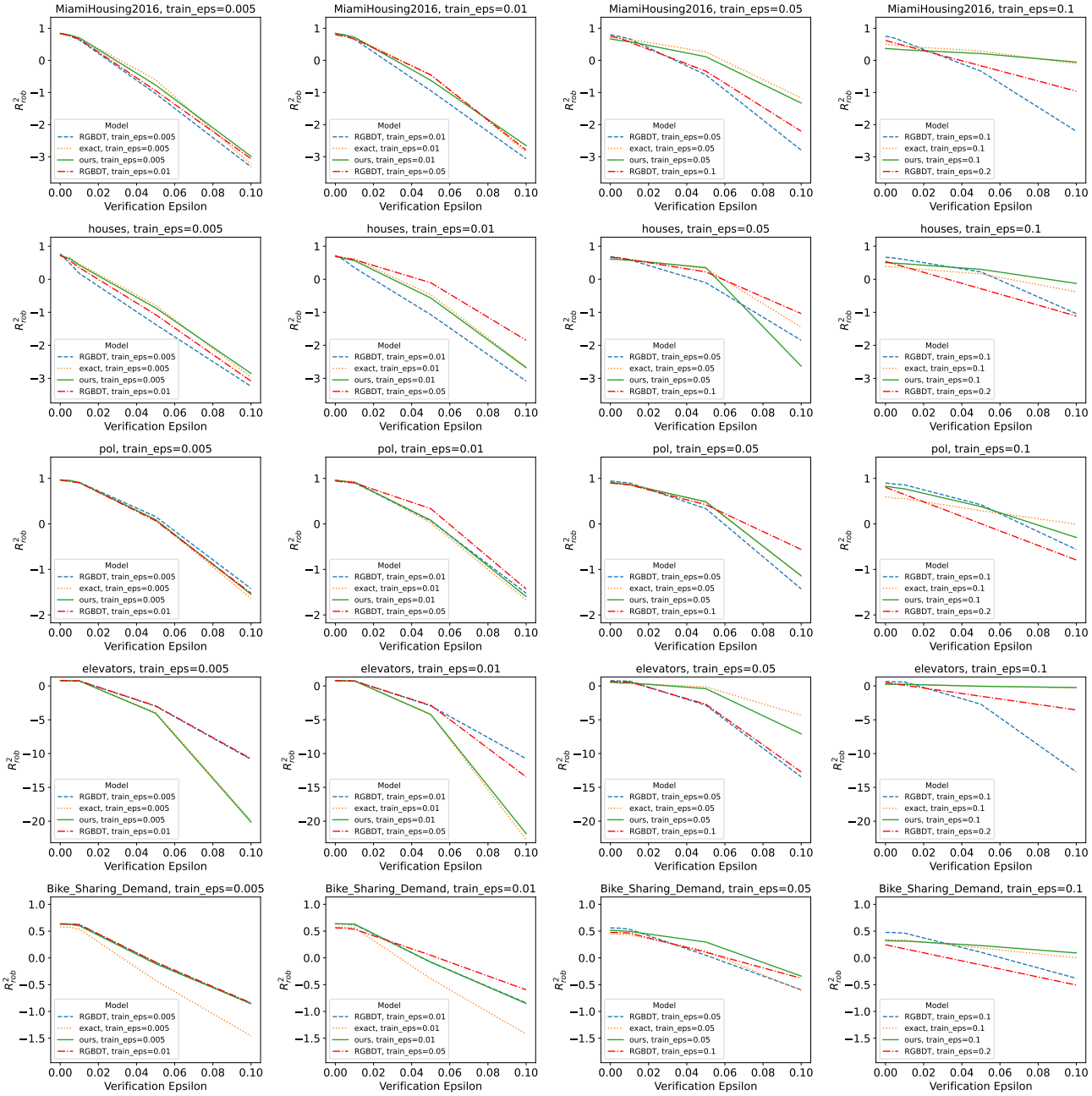

Figure 3: Comparison of the robustness profile of the models obtained by our method, the heuristic from Chen et al. [2019], the same heuristic set with a relaxed perturbation radius, and an exact solution to the robust splitting score from Equation 10.

B ANALYTICAL SOLUTION TO THE LINEAR RELAXATION OF ROBUST SPLITTING CRITERION

We need to compute the robust splitting criterion:

$$\begin{aligned}\mathcal{S}_{\text{rob}}(\mathcal{I}_L^0, \mathcal{I}_R^0, \Delta\mathcal{I}) \geq \mathcal{S}_{\text{rob}}^{\text{lb}} = \min_{p,q} \frac{1}{2}\Bigg[& \frac{\left(\sum_{i\in\mathcal{I}_L^0} g_i + p\right)^2}{\sum_{i\in\mathcal{I}_L^0} h_i + q + \lambda} \\ & + \frac{\left(\sum_{i\in\mathcal{I}_R^0} g_i + \sum_{i\in\Delta\mathcal{I}} g_i - p\right)^2}{\sum_{i\in\mathcal{I}_R^0} h_i + \sum_{i\in\Delta\mathcal{I}} h_i + \lambda} \\ & - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\Bigg] - \gamma,\end{aligned} \tag{21}$$

subject to the constraints:

$$\begin{aligned} p &\in \left[\sum_{i\in\Delta\mathcal{I}} \min(0, g_i), \sum_{i\in\Delta\mathcal{I}} \max(0, g_i)\right], \\ q &\in \left[0, \sum_{i\in\Delta\mathcal{I}} h_i\right], \\ p &\leq q \cdot c_1, \\ p &\geq q \cdot c_2, \\ p &\geq G^{(\Delta\mathcal{I})} - c_1(H^{(\Delta\mathcal{I})} - q), \\ p &\leq G^{(\Delta\mathcal{I})} - c_2(H^{(\Delta\mathcal{I})} - q), \end{aligned} \tag{22}$$

where the parameters c_1 and c_2 are defined as follows:

$$c_1 = \begin{cases} \frac{g_{\max}^{(\Delta\mathcal{I})}}{h_{\min}^{(\Delta\mathcal{I})}}, & g_{\max}^{(\Delta\mathcal{I})} \geq 0, \\ \frac{g_{\max}^{(\Delta\mathcal{I})}}{h_{\max}^{(\Delta\mathcal{I})}}, & g_{\max}^{(\Delta\mathcal{I})} < 0, \end{cases} \quad c_2 = \begin{cases} \frac{g_{\min}^{(\Delta\mathcal{I})}}{h_{\max}^{(\Delta\mathcal{I})}}, & g_{\min}^{(\Delta\mathcal{I})} \geq 0, \\ \frac{g_{\min}^{(\Delta\mathcal{I})}}{h_{\min}^{(\Delta\mathcal{I})}}, & g_{\min}^{(\Delta\mathcal{I})} < 0. \end{cases} \tag{23}$$

The minimisation problem in Eq. 21 can be simplified, by only considering the contributions of the left and right child nodes to determine optimal values of p and q, as the contribution of the parent node is constant and does not depend on the split.

$$f(p, q) = \frac{(A + p)^2}{C + q} + \frac{(B + T - p)^2}{D + Q - q} \tag{24}$$

where we consider the following simplified notation for the sums of the first and second derivatives:

$$\begin{aligned} A &= \sum_{i\in\mathcal{I}_L^0} g_i, & C &= \sum_{i\in\mathcal{I}_L^0} h_i, \\ B &= \sum_{i\in\mathcal{I}_R^0} g_i, & D &= \sum_{i\in\mathcal{I}_R^0} h_i, \\ T &= \sum_{i\in\Delta\mathcal{I}} g_i, & Q &= \sum_{i\in\Delta\mathcal{I}} h_i, \end{aligned}$$

Taking partial derivatives of Eq. 24 with respect to p and q gives us the following equations:

$$\frac{\partial f}{\partial p} = \frac{2(A+p)}{C+q} - \frac{2(B+T-q)}{D+Q-q} = 0, \tag{25}$$

$$\frac{\partial f}{\partial q} = -\frac{(A+p)^2}{(C+q)^2} + \frac{(B+T-p)^2}{(D+Q-q)^2} = 0 \tag{26}$$

Both equations reduce to the same proportionality condition

$$\frac{A+p}{C+q} = \frac{B+T-p}{D+Q-q} = \mu \tag{27}$$

Adding the two numerators and denominators shows

$$\mu = \frac{A+B+T}{C+D+Q} \tag{28}$$

Hence every stationary point must lie on the *central line*

$$p = \mu(C+q) - A. \tag{29}$$

Substituting Eq. 29 back into the objective function Eq. 24 yields a constant minimum for f:

$$f(p,q) = \frac{A^2+B^2+T^2+2(AB+AT+BT)}{C+D+Q}. \tag{30}$$

If any point on the line segment (Eq. 29) satisfies the constraints in Eq. 22, then it is a global minimum of the objective function. If not, the minimum must be attained at one of the extreme points of the feasible polygon defined by the constraints. This involves minimising the analytical value of f for each of the 8 constraint edges, as well as at their corners. We now enumerate the possible minimisation cases.

Case 1 $p = k\,q$ ($k \in \{c_1, c_2\}$):

$$q^\star = \frac{AD+AQ-BC-CT}{A+B-k(C+D+Q)+T}, \quad p^\star = kq^\star. \tag{31}$$

Case 2 $p = T - k\,(Q-q)$ ($k \in \{c_1, c_2\}$):

$$q^\star = \frac{AD+AQ-BC-kQ(C+D+Q)+DT+QT}{A+B-k(C+D+Q)+T}, \quad p^\star = T - k(Q-q^\star). \tag{32}$$

Case 3 fixed $p = \bar{p}$ ($\bar{p} \in \{\sum_{i\in\Delta\mathcal{T}} \min(0, g_i), \sum_{i\in\Delta\mathcal{T}} \max(0, g_i)\}$):

$$q^\star = \frac{AD+AQ-BC-CT+\bar{p}(C+D+Q)}{A+B+T}, \quad p^\star = \bar{p}. \tag{33}$$

Case 4 fixed $q \in \{0, Q\}$: similar formulas with roles of p and q swapped.

Case 5 Vertices of the bounding box defined by $p \in \{p_{min}, p_{max}\}$ and $q \in \{0, Q\}$:

$$(p^\star, q^\star) \in \{(p_{min}, 0), (p_{min}, Q), (p_{max}, 0), (p_{max}, Q)\}. \tag{34}$$

Case 6 Intersection of $p = k_a q$ and $p = T - k_b(Q-q)$, where $k_a, k_b \in \{c_1, c_2\}$ and $k_a \neq k_b$:

$$q^\star = \frac{T - k_b Q}{k_a - k_b}, \quad p^\star = k_a q^\star. \tag{35}$$

Case 7 Intersection of $p = c_1 q$ with $p = c_2 q$ (if $c_1 \neq c_2$), and intersection of $p = T - c_1(Q-q)$ with $p = T - c_2(Q-q)$ (if $c_1 \neq c_2$):

$$\text{For } p = c_1 q, p = c_2 q: \quad p^\star = 0, \quad q^\star = 0. \tag{36}$$

$$\text{For } p = T - c_1(Q-q), p = T - c_2(Q-q): \quad q^\star = Q, \quad p^\star = T. \tag{37}$$

Case 8 Intersection of $p = kq$ with $q = Q$, where $k \in \{c_1, c_2\}$ and intersection of $p = T - k(Q - q)$ with $q = 0$, where $k \in \{c_1, c_2\}$:

$$\text{For } p = kq \text{ and } q = Q: \quad p^\star = kQ, \quad q^\star = Q. \tag{38}$$

$$\text{For } p = T - k(Q - q) \text{ and } q = 0: \quad p^\star = T - kQ, \quad q^\star = 0. \tag{39}$$

Case 9 Intersection of $p = T - k(Q - q)$ with $p = \bar{p}$, where $k \in \{c_1, c_2\}$ and $\bar{p} \in \{p_{min}, p_{max}\}$:

$$p^\star = \bar{p}, \quad q^\star = \frac{\bar{p} + kQ - T}{k} \quad (\text{if } k \neq 0). \tag{40}$$

Case 10 Specific points $(p^\star, q^\star) = (\bar{p}, k/\bar{p})$, where $\bar{p} \in \{p_{min}, p_{max}\}$ and $k \in \{c_1, c_2\}$:

$$p^\star = \bar{p}, \quad q^\star = \frac{k}{\bar{p}} \quad (\text{if } \bar{p} \neq 0). \tag{41}$$

Thus, computing the global minimum corresponds to checking the feasibility of the unconstrained analytical solution, 8 edges, and 24 vertices of the bounding polygon. In practice, several of the intermediate values are cached, which leads to the minimum being computed efficiently, and in constant time.