Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
KR meets Machine Learning and Explanation

A Rectification-Based Approach for Distilling Boosted Trees into Decision Trees

Gilles Audemard[1] , **Sylvie Coste-Marquis**[1] , **Pierre Marquis**[1,2] , **Mehdi Sabiri**[1] , **Nicolas Szczepanski**[1]

[1]Univ. Artois, CNRS, CRIL, Lens, France
[2]Institut Universitaire de France, France
name@cril.fr

Abstract

We present a new approach for distilling boosted trees into decision trees, in the objective of generating an ML model offering an acceptable compromise in terms of predictive performance and interpretability. We explain how the correction approach called rectification can be used to implement such a distillation process. We show empirically that this approach provides interesting results, in comparison with an approach to distillation achieved by retraining the model.

1 Introduction

Applications of machine learning (ML) have expanded rapidly over the past decade, driven by the rise of AI systems based on ML that deliver increasingly strong predictive performance. However, in many high-stakes application domains — such as healthcare and law — predictive accuracy alone is not sufficient. Users must also be able to interpret the results produced, understand the reasoning behind the system's predictions through appropriate explanations, and, whenever possible, correct prediction errors.

Unfortunately, most accurate ML models are typically hard to interpret, and vice versa (Rudin 2019; Arrieta et al. 2020). Therefore, when both accuracy and interpretability are expected (as it is the case when an ML-based AI system is designed for a critical application), a compromise must be looked for. Model *distillation* (Hinton, Vinyals, and Dean 2015; Frosst and Hinton 2017; Gou et al. 2021), a method for building a "simpler" target ML model than the source ML model considered as input, can be used to achieve such a compromise (Asadulaev, Kuznetsov, and Filchenkov 2019; Wood-Doughty, Cachola, and Dredze 2022).

Following this research line, we present in this paper an *incremental distillation process*, where the aim is to *correct* an initial ML model I for binary classification, that is quite interpretable but not very accurate (here, a decision tree), using another ML model P for binary classification, that is quite precise but not interpretable (here, a boosted tree). Our objective is to correct I in an incremental way to make it logically closer to P at each correction step (i.e., to increase the number of instances $\boldsymbol{x}$ such that $I(\boldsymbol{x}) = P(\boldsymbol{x})$).

A key motivation for such a distillation process is that many XAI queries are tractable for decision trees, while they are intractable (NP-hard) for boosted trees (Audemard et al. 2021). In practice, only a few XAI algorithms have been implemented and are available online for boosted trees. Furthermore, such XAI algorithms for boosted trees do not scale up well. Especially, this is the case for the XAI query that consists in deriving a subset-minimal abductive explanation (aka a sufficient reason) (Ignatiev, Narodytska, and Marques-Silva 2019; Darwiche and Hirth 2020) for a given instance. Notably, the computation in a reasonable amount of time of a sufficient reason for an instance $\boldsymbol{x}$ given P may depend heavily on the instance $\boldsymbol{x}$ one starts with. In contrast, the critical part in the computation of a sufficient reason for $\boldsymbol{x}$ given P through the distillation of P into I is the distillation process itself: once I has been computed from P, even when I is large enough, computing a sufficient reason for $\boldsymbol{x}$ given I (and answering other XAI queries) becomes highly efficient for any instance.

In this paper, we show that *rectification* (Coste-Marquis and Marquis 2021), a belief change approach suited to the correction of binary classifiers, can be considered with profit for the distillation of a boosted tree P into a decision tree I. More precisely, we show how rectification can be specialized to the case when P is a boosted tree and I a decision tree, so that the rectification of P by I can be achieved incrementally and in a much more efficient way.

In our work, we compared our rectification-based approach to distillation with a simple, yet pure ML approach based on *retraining* the model (Zhou 2019). For each of the two approaches, we performed some experiments to assess the quality of the distillation produced, evaluated by measuring the predictive performance of the corrected decision tree relative to P, i.e., viewing P as a ground-truth oracle. The experimental results obtained have shown the interest of the distillation approach based on rectification compared to retraining.

Because the distillation of P into I may lead to a significant increase in the size of the decision tree (this increase being unavoidable in the worst case), it was also important to point out some empirical evidence to support the claim that the distillation of P into I is computationally useful. To do so, we focused on a specific XAI query, namely computing a sufficient reason for a given instance $\boldsymbol{x}$. We took advantage of state-of-the-art algorithms for deriving sufficient reasons to compare the time needed to derive a sufficient reason for $\boldsymbol{x}$ from I with the time needed to derive a sufficient reason for $\boldsymbol{x}$ from P, i.e., without distilling first P into I. As soon

as the former computation time is strictly smaller than the latter, the time spent in distilling P into I can be balanced over sufficiently many instances $\boldsymbol{x}$. Our empirical results (based on average computation times and numbers of time-out) clearly show that distilling P into I is advantageous despite the significant growth of I it may lead to.

For space reasons, the proofs of the propositions presented in the paper, as well as the datasets, the code used in our experiments and additional empirical results are furnished as a supplementary material available from (Audemard et al. 2025).

2 Formal Preliminaries

Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables and let y be a Boolean variable not appearing in X. Literals y and $\overline{y}$ are used to denote the classes of positive and negative instances (respectively). X corresponds to the set of Boolean conditions appearing in the boosted tree P and it can be used to describe the instances. The set of all instances over X is denoted by $\boldsymbol{X}$. The elements of X do not primarily represent independent conditions because they can come from the same numerical or categorical primitive attributes (for example, we can find in P the condition $x_1 = (S > 30)$ relating to the numerical attribute S but also the condition $x_2 = (S > 20)$ which is logically linked to it: x_1 cannot be true whereas x_2 would be false). A *domain theory*, in the form of a logical formula Th on X, specifies the links between non-independent Boolean conditions (for example, $Th = x_1 \Rightarrow x_2$). Each instance $\boldsymbol{x}$ of $\boldsymbol{X}$ can be considered as an interpretation on X that satisfies Th. $\boldsymbol{x}$ is then viewed as a mapping associating each Boolean variable x_i ($i \in [n]$) with 1 if and only if the i^{th} coordinate $\boldsymbol{x}_i$ of $\boldsymbol{x}$ is equal to 1. This interpretation can be represented by a (canonical) term $t_{\boldsymbol{x}}$ on X, formed by the set (interpreted as a conjunction) of the positive literals x_i ($i \in [n]$), such that $\boldsymbol{x}_i = 1$ and by the negative literals $\overline{x_i}$ ($i \in [n]$) such that $\boldsymbol{x}_i = 0$.

Definition 1. *A* binary classifier *on X is a mapping C from $\boldsymbol{X}$ to the set of Boolean values $\{0, 1\}$. $\boldsymbol{x} \in \boldsymbol{X}$ is a positive instance if $C(\boldsymbol{x}) = 1$ and a negative one if $C(\boldsymbol{x}) = 0$.*

Any binary classifier can be represented by a classification circuit in the sense of (Coste-Marquis and Marquis 2021):

Definition 2. *A* classification circuit Σ *on $X \cup \{y\}$ is a circuit equivalent to a formula of the form $\Sigma_X \Leftrightarrow y$ where Σ_X is a Boolean formula on X.*

Indeed, any binary classifier C based on Boolean attributes X (including decision trees and boosted trees) can be viewed as a Boolean formula C_X on X whose models are precisely the instances classified positively by C, i.e., satisfying $C(\boldsymbol{x}) = 1$. In general, there is no polynomial-time algorithm to get C_X from C, but associated with C, we can always define a classification circuit $\Sigma = C_X \Leftrightarrow y$.

In the following, when Φ is a Boolean circuit or a formula on $X \cup \{y\}$ and z is any variable from $X \cup \{y\}$, $\Phi(z)$ (resp. $\Phi(\overline{z})$) denotes the *conditioning* of Φ by z (resp. by $\overline{z}$). $\Phi(z)$ (resp. $\Phi(\overline{z})$) is the circuit (or the formula) obtained by replacing in Φ any occurrence of z by the Boolean constant $\top$ representing the truth value 1 (resp. $\bot$ representing the truth value 0). When $\Sigma = \Sigma_X \Leftrightarrow y$ is a classification circuit on $X \cup \{y\}$, the set of models of $\Sigma(y)$ consists precisely of the models of Σ_X. Finally, when $\boldsymbol{x} \in \boldsymbol{X}$ is an instance, $\Phi(\boldsymbol{x})$ denotes the iterative conditioning of Φ by each literal of $t_{\boldsymbol{x}}$. Thus, $\boldsymbol{x} \in \boldsymbol{X}$ is classified positively (resp. negatively) by Σ when $\Sigma(\boldsymbol{x})$ is equivalent to y (resp. $\overline{y}$).

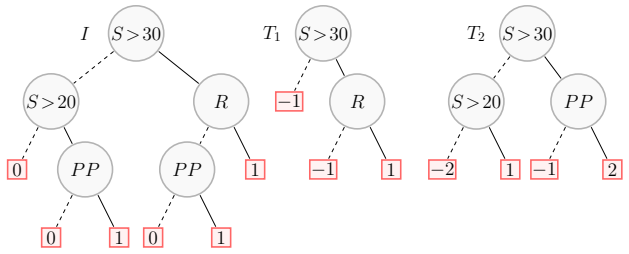

Figure 1: A decision tree I and a tree ensemble (alias a boosted tree) P. P is formed by two "regression" trees T_1 and T_2. For each tree, the dotted arc (resp. the solid arc) from a node labeled with a condition $cond$ corresponds to the assignment where $cond$ is false (resp. true).

Decision trees (Breiman et al. 1984; Quinlan 1986) and boosted trees (Freund and Schapire 1997) are two ML models that can be used to represent binary classifiers:

Definition 3. *A* decision tree *(resp.* "regression" tree*) T on X is a binary tree such that internal nodes are decision nodes labeled by elements x of X and leaves by Boolean constants (resp. real numbers). By convention, when following the left (resp. right) child of a decision node labelled by x, x is set to false (0) (resp. true (1)). An instance $\boldsymbol{x} \in \boldsymbol{X}$ satisfies $T(\boldsymbol{x}) = v$ if and only if the unique path from the root of T to a leaf which is compatible with $\boldsymbol{x}$ is a leaf labeled by v.*

Definition 4. *An* additive tree ensemble *P on X is a set $\{T_1, \ldots, T_m\}$ of "regression" trees on X. An instance $\boldsymbol{x} \in \boldsymbol{X}$ satisfies $P(\boldsymbol{x}) = 1$ iff $\sum_{i=1}^{m} T_i(\boldsymbol{x}) > 0$.*

In our experiments, the tree ensembles P under consideration were generated using the XGBoost algorithm (hence, slightly abusing words, we use the term "boosted tree" in this paper instead of "tree ensemble"). However, our rectification method applies equally to any other additive tree ensembles, such as (weighted) random forests or ensembles generated using Adaboost (Freund and Schapire 1997; Schapire and Freund 2014).

Example 1. *As an illustrative example, let us consider a problem of credit allocation to bank customers. Each customer is characterized by an annual salary (S a numerical attribute), a fact of having already reimbursed a previous loan (R a Boolean attribute) and, whether or not, they have a permanent position (PP a Boolean attribute). Two binary classifiers are considered. On the one hand, a boosted tree P consisting of two regression trees (T_1 and T_2). On the other hand, a decision tree I. P and I are described in Figure 1. X corresponds to the four Boolean conditions considered in this order $x_1 = (S > 30)$, $x_2 = (S > 20)$,*

$x_3 = R$, $x_4 = PP$. *Note that* x_1 *and* x_2 *are not independent. Indeed, an instance over* $X = \{x_1, x_2, x_3, x_4\}$ *is feasible only if it satisfies* $Th = x_1 \Rightarrow x_2$. *We can easily check that* I *is associated with a classification circuit on* $X \cup \{y\}$ *that is equivalent to* $((x_1 \wedge x_3) \vee (x_2 \wedge x_4)) \Leftrightarrow y$ *and that* P *is associated with a classification circuit on* $X \cup \{y\}$ *that is equivalent to* $(x_1 \wedge x_4) \Leftrightarrow y$ *(indeed,* $T_1 + T_2 > 0$ *holds precisely when* $S > 30$ *and* PP *are both true). We can also check that* I *and* P *classify the instances of* $\boldsymbol{X}$ *in the same way, except for the instances* $(0, 1, 1, 1)$, $(0, 1, 0, 1)$ *and* $(1, 1, 1, 0)$. *Indeed, these three instances are classified positively by* I *and negatively by* P.

A *classification rule* is a rule that indicates thanks to its conclusion part (the right-hand side of the rule) how to classify any instance matching its premises part (the left-hand side of the rule).

Definition 5. *A* classification rule *over* y *(resp.* $\overline{y}$*) is a formula of the form* $R = \varphi_X \Rightarrow y$ *(resp.* $R = \varphi_X \Rightarrow \overline{y}$*) where* φ_X *is a formula on* X.

Such a rule R can be *deduced* from a classification circuit Σ on $X \cup \{y\}$ precisely when $\Sigma \models R$ holds.

Example 2 (Example 1, cont'ed)**.** *The formula* $R = \overline{x_4} \Rightarrow \overline{y}$ *is a classification rule over* $\overline{y}$. *This rule can be deduced from the classification circuit* $(x_1 \wedge x_4) \Leftrightarrow y$ *associated with* P.

Classification rules do not state how to classify instances that are not covered by their left-hand side. For this reason, a classification rule (and more generally a set of such rules) does not represent a *complete classifier*. Observe that when an instance $\boldsymbol{x}$ is covered by the left-hand side φ_X of a rule R (so that R indicates how $\boldsymbol{x}$ must be classified), the iterative conditioning $R(\boldsymbol{x})$ of R by $t_{\boldsymbol{x}}$ is equivalent to the right-hand side of R. Thus, $R(\boldsymbol{x})$ can be interpreted as the application of the (partial) classifier R to instance $\boldsymbol{x}$, giving the corresponding class. Furthermore, classification rules can be conflicting: $R_1 = \varphi^1_X \Rightarrow y$ and $R_2 = \varphi^2_X \Rightarrow \overline{y}$ are said to be *conflicting* if and only if $Th \wedge \varphi^1_X \wedge \varphi^2_X$ is consistent. Given two conflicting classification rules $R_1 = \varphi^1_X \Rightarrow y$ and $R_2 = \varphi^2_X \Rightarrow \overline{y}$, one does not know how to classify an instance $\boldsymbol{x}$ satisfying $\varphi^1_X \wedge \varphi^2_X$ as the two rules give contradictory conclusions about the class of $\boldsymbol{x}$.

Finally, one needs to make precise the notion of abductive explanation for an instance given a binary classifier (Ignatiev, Narodytska, and Marques-Silva 2019):

Definition 6. *An* abductive explanation *for* $\boldsymbol{x} \in \boldsymbol{X}$ *given a binary classifier* C *on* X *is a term* t *on* X *such that* t *covers* $\boldsymbol{x}$ *(i.e.,* $t \subseteq t_{\boldsymbol{x}}$*) and for all* $\boldsymbol{x}' \in \boldsymbol{X}$ *such that* t *covers* $\boldsymbol{x}'$*, one has* $C(\boldsymbol{x}') = C(\boldsymbol{x})$.

Such an abductive explanation t provides a set of conditions (literals) corresponding to characteristics of the input instance $\boldsymbol{x}$ and explaining why the instance $\boldsymbol{x}$ is classified by C in the way it has been classified.

Example 3 (Example 1, cont'ed)**.** $t = \overline{x_1} = \overline{(S > 30)}$ *is an abductive explanation for* $(0, 1, 1, 1)$ *given* P. *The fact that the salary of the incomer is less than or equal to 30k$ is sufficient to explain why, according to* P, *the loan requested should not be granted.*

3 Rectifying a Classification Circuit

Let us now show how to use the rectification operation to iteratively correct a decision tree using classification rules extracted from a boosted tree. In the general case, when a classification circuit $\Sigma = \Sigma_X \Leftrightarrow y$ is rectified by a formula F on $X \cup \{y\}$, the result of the rectification process (Coste-Marquis and Marquis 2023) is the "corrected" classification circuit $\Sigma \star F$ defined by

$$\Sigma \star F = \Sigma^F_X \Leftrightarrow y, \text{ where}$$
$$\Sigma^F_X \equiv (\Sigma_X \wedge \neg(F(\overline{y}) \wedge \neg F(y))) \vee (F(y) \wedge \neg F(\overline{y})).$$

Σ^F_X characterizes the instances to be classified as positive after the rectification of Σ by F. Those instances can be gathered into two sets: the instances that are consistently asked to be classified as positive by F (they are characterized by the subformula $F(y) \wedge \neg F(\overline{y})$ of Σ^F_X), and the instances that were already classified as positive by Σ provided that F did not consistently ask them to be classified as negative (those instances are characterized by the subformula $\Sigma_X \wedge \neg(F(\overline{y}) \wedge \neg F(y))$ of Σ^F_X).

Example 4 (Example 1, cont'ed)**.** *Suppose that the rectification formula* F *is the classification rule* $R = \overline{x_4} \Rightarrow \overline{y}$ *that can be deduced from the classification circuit* $(x_1 \wedge x_4) \Leftrightarrow y$ *associated with* P. *We have* $F(y) \equiv x_4$ *and* $F(\overline{y}) \equiv \top$, *so that* $F(y) \wedge \neg F(\overline{y}) \equiv \bot$ *and* $\neg(F(\overline{y}) \wedge \neg F(y)) \equiv x_4$. *Thus, rectifying the classification circuit* $((x_1 \wedge x_3) \vee (x_2 \wedge x_4)) \Leftrightarrow y$ *associated with* I *by the formula* F *leads to a classification circuit that is equivalent to* $(((x_1 \wedge x_3) \vee (x_2 \wedge x_4)) \wedge x_4) \Leftrightarrow y$, *thus equivalent to* $(((x_1 \wedge x_3) \vee x_2) \wedge x_4) \Leftrightarrow y$.

Interestingly, in the specific context considered in this paper, i.e., Σ_X is a decision tree on X and $F = P_X \Leftrightarrow y$ where P_X is a boosted tree on X, the previous definition of $\Sigma \star F$ can be simplified significantly, leading to improved computations. The next three propositions are the key results on which our distillation method is based. Proposition 1 shows that the general definition of a rectified classification circuit can be simplified when the rectification formula is a classification rule. Proposition 2 shows that the rectification of a classification circuit by a conjunction of classification rules can be achieved on a rule-per-rule basis. Finally, Proposition 3 indicates how classification rules can be generated from abductive explanations.

Proposition 1. *Let* $\Sigma = \Sigma_X \Leftrightarrow y$ *be a classification circuit on* $X \cup \{y\}$. *Let* $R = \varphi_X \Rightarrow y$ *(resp.* $R = \varphi_X \Rightarrow \overline{y}$*) be a classification rule over* y *(resp.* $\overline{y}$*). We have* $\Sigma \star R \equiv (\Sigma_X \vee \varphi_X) \Leftrightarrow y$ *(resp.* $\Sigma \star R \equiv (\Sigma_X \wedge \neg\varphi_X) \Leftrightarrow y$*).*

The previous example illustrates this proposition.

The next result shows that rectifying a classification circuit Σ by a (conjunctively-interpreted) *set* F *of classification rules deduced from another classification circuit* amounts to rectifying Σ by each rule from F in an iterative fashion (the order according to which the rules are considered does not matter because those rules are never conflicting):

Proposition 2. *Let* $\Sigma = \Sigma_X \Leftrightarrow y$ *be a classification circuit on* $X \cup \{y\}$. *Let* $\Phi = \Phi_X \Leftrightarrow y$ *be another classification circuit on* $X \cup \{y\}$. *Let* $\{R_1, \ldots, R_k\}$ *be a set of classification rules that can be deduced from* Φ. *We have*

$$\Sigma \star (R_1 \wedge \ldots \wedge R_k) \equiv (\Sigma \star R_1) \star \ldots \star R_k.$$

Finally, the next proposition shows how to deduce classification rules from a circuit $\Phi_X \Leftrightarrow y$, using abductive explanations for instances given Φ_X:

Proposition 3. *Let $C_X \Leftrightarrow y$ be a classification circuit on $X \cup \{y\}$ associated with a binary classifier C. Let $\boldsymbol{x} \in \boldsymbol{X}$ be an instance such that $C(\boldsymbol{x}) = 1$ (resp. $C(\boldsymbol{x}) = 0$). Let t be an abductive explanation for an instance $\boldsymbol{x} \in \boldsymbol{X}$ given C. $R = t \Rightarrow y$ (resp. $R = t \Rightarrow \overline{y}$) is a classification rule over y (resp. $\overline{y}$) that is implied by $C_X \Leftrightarrow y$.*

4 Distilling Boosted Trees into Decision Trees

By combining Propositions 1, 2 and 3, one can define an incremental (and possibly partial) distillation process of boosted trees P into decision trees I. The idea is to consider only instances $\boldsymbol{x}$ that are misclassified by I (i.e., those classified differently by P) as soon as they appear at inference time, and whenever such an instance $\boldsymbol{x}$ is encountered, to correct the classification circuit $I_X \Leftrightarrow y$ associated with I by a classification rule R that covers $\boldsymbol{x}$ and is implied by the classification circuit $P_X \Leftrightarrow y$ associated with P. At each correction step, the set $\boldsymbol{X}_I^{\pm} = \{\boldsymbol{x} \in \boldsymbol{X} \mid I(\boldsymbol{x}) \neq P(\boldsymbol{x})\}$ of instances from $\boldsymbol{X}$ that are, according to P, misclassified by I is reduced.

The choice for such a *lazy but opportunistic* approach to distillation comes from spatial complexity results showing that the full rectification of $I_X \Leftrightarrow y$ by $P_X \Leftrightarrow y$ would be out of reach in general. Indeed, a combinatorial blow-up is unavoidable in the worst case, whatever the way the distillation is achieved: it is known that, in the worst case, turning a boosted tree P_X into an equivalent decision tree I_X requires exponential space (de Colnet and Marquis 2023).

Algorithm 1 makes precise how one step of the incremental distillation process of P into I (the step triggered by $\boldsymbol{x}$) is achieved. An abductive explanation t for $\boldsymbol{x}$ given P is first computed. A classification rule R that is implied $P_X \Leftrightarrow y$ (cf. Proposition 3) is formed using t. Then, using Proposition 1, the classification circuit $\Sigma_X \Leftrightarrow y$ where $\Sigma_X = I$ is rectified by R. In detail, a decision tree I^R equivalent to $\Sigma_X \vee \varphi_X$ (or to $\Sigma_X \wedge \neg\varphi_X$ depending on the right-hand side of R) can be generated efficiently by looking at the root-to-leaf paths p of Σ_X and *rectifying each of them separately, in parallel*.[1] Basically, every root-to-leaf path p of a decision tree can be associated with a term t, which can be defined by induction as follows: let t be the empty term; if p reduces to a leaf node, then return t; otherwise, p starts with a decision node (the root of the tree) which is labelled by a Boolean variable x; then add to t literal x (resp. literal $\overline{x}$) if p goes right (resp. left) from the decision node and resume from the child node that has been reached. When rectifying Σ_X by R, only those paths p of Σ_X associated with terms t such that $t \wedge \varphi_X \wedge Th$ is consistent need to be considered. More precisely, among them only those paths leading to a 0-leaf (resp. a 1-leaf) need to be updated when the right-hand side of R is y (resp. $\overline{y}$). Thus, when the right-hand side of R is y (resp. $\overline{y}$) updating p simply consists in replacing its 0-leaf node (resp. its 1-leaf node) by a decision tree representing the conjunction of the literals from $t \setminus \varphi_X$ (resp. the negation of the conjunction of the literals from $t \setminus \varphi_X$). Finally, the domain theory Th associated with I can be leveraged to simplify the resulting tree I^R. In a bottom-up way, starting from the leaf of a branch p of I^R up to the root of I^R, an arc of p can be removed when the literal ℓ labelling it is a logical consequence of $(p \setminus \{\ell\}) \wedge Th$. Furthermore, any internal node of I^R with a left subtree identical to its right subtree can be replaced by one of its two subtrees. This simplification step is fundamental in practice to limit the growth of the tree.

[1] This approach can be easily extended to the multi-class and even to the multi-label classification setting.

Algorithm 1 The incremental distillation of a boosted tree into a decision tree.

Require: a decision tree I and a boosted tree P over X, an instance $\boldsymbol{x} \in \boldsymbol{X}$ such that $\boldsymbol{x} \in \boldsymbol{X}_I^{\pm}$.
Ensure: a decision tree I^R such that $\boldsymbol{X}_{I^R}^{\pm} \subset \boldsymbol{X}_I^{\pm}$.
$t \leftarrow abductive - expl(P, \boldsymbol{x})$
if $P(\boldsymbol{x}) = 1$ **then** $R \leftarrow (t \Rightarrow y)$ **else** $R \leftarrow (t \Rightarrow \overline{y})$
$I^R \leftarrow simplify(rectify(I, R))$
return (I^R)

Example 5 (Example 1 cont'ed)**.** *Consider the instance $\boldsymbol{x} = (0, 1, 1, 1)$. As $I(\boldsymbol{x}) = 1$ and $P(\boldsymbol{x}) = 0$, we have $\boldsymbol{x} \in \boldsymbol{X}_I^{\pm}$, so one needs to correct the misclassification done by I. Using the approach introduced in (Audemard et al. 2023), the abductive explanation $t = \overline{x_1} = \overline{(S > 30)}$ for $\boldsymbol{x}$ given P is first computed. Since $P(\boldsymbol{x}) = 0$, from Proposition 3, we know that the classification rule $R = \overline{x_1} \Rightarrow \overline{y}$ is implied by $P_X \Leftrightarrow y$.*

Then, one rectifies the classification circuit $\Sigma = I_X \Leftrightarrow y$ by the rule R and produces a circuit $\Sigma \star R$ which is equivalent to $(I_X \wedge \neg t) \Leftrightarrow y$, following Proposition 1. To do it, we generate a decision tree I^R such that I_X^R is equivalent to $I_X \wedge \neg t$. I^R is obtained by updating every path of I corresponding to a term that is consistent with the premises $t = \overline{x_1} = \overline{(S > 30)}$ of R but with a 1-leaf (since the conclusion $\overline{y}$ of R asks for a 0-leaf). A single path of I needs to be updated, the one associated with the term $\overline{(S > 30)} \wedge (S > 20) \wedge PP$ (see Figure 1). Updating it simply consists in replacing its 1-leaf by a 0-leaf, resulting in the tree shown in Figure 2 (left sub-figure), in which the replaced leaf appears in red. This tree can then be simplified (the resulting tree is shown in the sub-figure on the right).

Note that the correction step achieved by rectifying $I_X \Leftrightarrow y$ by R also corrects the classification error made by $I \Leftrightarrow y$ on the instance $(0, 1, 0, 1)$. Indeed, the decision tree I^R classifies all instances in the same way as P, except for $(1, 1, 1, 0)$, which will eventually be corrected if the instance $(1, 1, 1, 0)$ is considered in the future.

In our approach, each correction step is thus triggered by *a single instance $\boldsymbol{x}$*. Considering a population instead (i.e., a batch of instances classified in the same way by P) would not be possible in general because two instances can be classified in the same way by P for incompatible reasons (i.e., the two sets of abductive explanations can be disjoint).

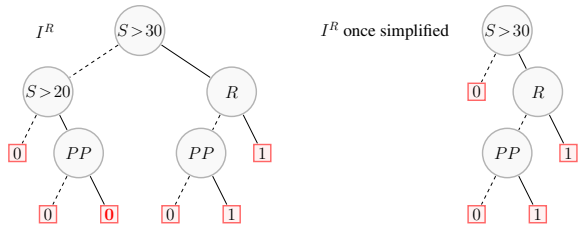

Figure 2: A decision tree I^R such that $I^R_X \Leftrightarrow y$ is equivalent to $(I_X \Leftrightarrow y) \star R$ (left). An equivalent decision tree, obtained by simplification (right).

Logical guarantees Notably, our rectification-based approach ensures that I is *made logically closer* to P at each correction step (i.e., the number of instances $\boldsymbol{x}$ such that $I(\boldsymbol{x}) = P(\boldsymbol{x})$ increases at each step). Stated differently, Algorithm 1 is correct w.r.t. its specification:

Proposition 4. *The decision tree I^R computed by Algorithm 1 from a decision tree I on X, a boosted tree P on X, and an instance $\boldsymbol{x} \in X^{\pm}_I$ is such that $X^{\pm}_{I^R} \subset X^{\pm}_I$.*

This guarantee is offered whatever the decision tree I one starts with, especially in the restricted case when the initial tree I consists of a single node (e.g., a leaf labelled with the majority class) or when I already is at start equivalent to P. Actually, the choice of the initial decision tree I impacts only the number of correction steps that will be needed to achieve a full distillation.

The guarantee that $X^{\pm}_{I^R} \subset X^{\pm}_I$ is also ensured whatever the rule R that is computed by Algorithm 1. When forming classification rules R from abductive explanations t, subset-minimal abductive explanations appear as the best candidates since they lead to the most general (i.e., logically strongest) classification rules R. Indeed, since such rules cover more instances than rules that are less general, using them may lead to diminishing the number of correction steps of the distillation process. However, deriving subset-minimal abductive explanations given boosted trees is intractable. This explains why in our implementation we focused on tree-specific explanations (Audemard et al. 2023).

The order in which rectifications have been made also has no impact on the resulting tree once all the misclassified instances $\boldsymbol{x}$ have been handled, provided that the same rule has been extracted whatever the step at which $\boldsymbol{x}$ is considered (this is a direct consequence of Proposition 2). Indeed, in this case, the resulting tree represents the same binary classifier whatever the order with which the misclassified instances have been encountered.

On the contrary, at each step, the misclassified instance $\boldsymbol{x}$ that is encountered has a big impact on the abductive explanation t that is computed (since t must be an abductive explanation for $\boldsymbol{x}$), thus on the classification rule R that is derived from this explanation, and, as a consequence, on the misclassified instances that are covered by this rule. Similarly, the abductive explanation t for $\boldsymbol{x}$ given P that is chosen to form R matters. Indeed, the number of instances that remain misclassified after a preset number k of rectification steps can greatly vary depending on the instance considered at each step and its chosen explanation. Especially, if the correction process is interrupted before exhaustion (i.e., the distillation of P into I is partial), significantly different decision trees can be generated after k rectification steps depending on the choices made.

Complexity analysis A key advantage of Algorithm 1 is its computational efficiency. Indeed, this algorithm runs in strictly polynomial time with respect to the size of its inputs (the boosted tree P and the current decision tree I). Specifically:

- *Explanation extraction:* While deriving subset-minimal abductive explanations for boosted trees is NP-hard, our algorithm circumvents this by deriving *tree-specific* explanations. The derivation of a tree-specific explanation t is guaranteed to run in polynomial time (Audemard et al. 2023).
- *Rectification and simplification:* Rectifying the decision tree I involves identifying and updating the root-to-leaf paths of I that are consistent with the classification rule R associated with t. The subsequent bottom-up simplification traverses the generated nodes to remove redundant arcs. Both operations are performed in polynomial time relative to the number of nodes in the current tree I (Coste-Marquis and Marquis 2023).

Accordingly, what makes the full distillation of P into I computationally demanding is the repetition of the rectification steps. Our lazy, instance-triggered approach offers a way to keep the control of the complexity of the distillation process.

5 Experiments

Empirical protocol In our experiments, we considered various datasets for binary classification, coming from the open repositories UCI (https://archive.ics.uci.edu/ml/index.php) and openML (https://www.openml.org/). Each instance is associated with a class c, which is equal to 1 or to 0 depending on whether the instance is positive or negative.

Each dataset was partitioned into a 70% split used *only* to train the oracle P, and a 30% split to trigger corrections and test performance. This simple evaluation protocol proves enough given that our goal is not to improve P (which is viewed as an oracle), but to logically correct I to mimic P's behavior.

For each dataset, one started by learning a precise model P, here, a boosted tree, using the algorithm provided in the XGBoost library (Chen and Guestrin 2016), adjusting hyperparameter values using grid search in an attempt to get a predictive performance that is good enough. The hyperparameterization of P (i.e., the values of max depth, n_estimators, and learning rate) was achieved using only the training set. Once P has been learned, every instance of each dataset has been translated into an instance represented in the space of the n Boolean conditions used in P (thus, the resulting instances are described solely using Boolean attributes). L (resp. T) denotes the resulting set of instances

Dataset	$\|E\|$	$\|F\|$	$\|B\|$	$\%I_o$	$\%I_d$	$\%P$	Repository	$\|T^{\pm}_{I_o}\|$	$\|T^{\pm}_{I_d}\|$
bank	4521	48	521	87.50%	85.46%	88.0%	UCI	125	153
biodegradation	1054	41	242	81.75%	81.08%	84.68%	openML	42	48
australian	690	38	174	81.63%	80.39%	86.20%	openML	26	32
bupa	345	5	108	89.58%	89.58%	97.26%	UCI	8	8
german	1000	58	105	94.28%	93.57%	95.71%	UCI	13	20
contraceptive	1473	21	68	68.44%	62.13%	73.87%	UCI	77	126
cleveland	303	23	42	78.57%	76.19%	87.5%	openML	12	13
compas	6172	11	33	68.28%	65.85%	69.29%	openML	125	254
cnae	1080	856	15	96.13%	95.97%	96.47%	UCI	5	5
breast-tumor	286	37	58	52.7%	52.5%	53.5%	UCI	29	30
balance_0_vs_1	625	4	17	88.37%	84.88%	90.90%	UCI	6	18
balance_0_vs_2	625	4	17	82.97%	82.60%	90.14%	UCI	7	8
balance_1_vs_2	625	4	17	88.90%	80.00%	94.21%	UCI	16	31

Table 1: Description of the datasets and accuracies of the models at start, before distillation.

used to learn decision trees (resp. to trigger corrections and to test the predictive performance of the trees).

Then, decision trees I have been learned from L using the algorithm provided in the scikit-learn library (Pedregosa et al. 2011). Two configurations have been tested: one for which hyperparameters have been set to their *default values* and one for which hyperparameters have been *optimized*. The hyperparameterization of I (here, adjusting the value of max depth) was based on the training set L, only. In the default configuration, the depth of I is not bounded a priori (whatever the step): any internal node N of I is decomposed whenever the subset of the training set verifying all the conditions of the path going from the root of the tree to N only contains instances of the same class (the node N is said to be pure). In the optimized configuration, the depth of I has been tuned in order to achieve a better predictive performance and avoid overfitting. It has been set, for each dataset, to the value given in Table 3, step 0, column D. The depth of retrained trees has been limited to this value as well.

The default configuration typically leads to constructing decision trees that overfit, offering in general a lower accuracy when assessed on T. Nevertheless, it makes sense to consider this default configuration for two reasons. On the one hand, because of the limited precision of the trees considered initially under this configuration, the correction steps to be carried out can be numerous. On the other hand, it is particularly favorable for ensuring correction guarantees, even when retraining is used for the correction purpose. Indeed, in practice, the choices made at each decision node under the default configuration ensure that the instances of the training set are associated with their expected class (the one given in the training set). Indeed, every time an instance is added to the training set so as to correct its classification, the new decision tree learned after this addition classifies the instance correctly: the desired correction is thus achieved. To ensure robustness when learning the initial decision trees I from L (the training set translated into Boolean conditions), we did not rely on a single split. Instead, we utilized a repeated random sub-sampling cross-validation process: we learned 10 decision trees I from L, retaining 70% of instances from L for training each tree. The median accuracy $\%I_o$ (resp. $\%I_d$) obtained for the 10 decision trees has been measured on the corresponding test set T for optimized (resp. default) configuration.

The next step was to correct the decision tree I at hand whenever necessary and whatever the configuration used to learn it. For each $(\boldsymbol{x}, c)$ in T, $P(\boldsymbol{x})$ is considered as the "true" class (ground truth) of $\boldsymbol{x}$. $T^{\pm}_I = \{(\boldsymbol{x}, c) \in T : I(\boldsymbol{x}) \neq P(\boldsymbol{x})\}$ is the subset of instances in T that, according to P, are not classified correctly by I. The accuracy I_P of I *relative to* P, empirically measured on T, is given by $1 - \frac{|T^{\pm}_I|}{|T|}$. So, if $I(\boldsymbol{x}) = P(\boldsymbol{x})$ for every $(\boldsymbol{x}, c) \in T$, I_P is 100%. Thus, $1 - I_P$ indicates the proportion of instances in T that still need correction. Of course, the value of I_P only indicates the extent to which I classifies instances in the same way as P, but does not give any information about the actual predictive performance of P.

For each $\boldsymbol{x} \in T^{\pm}_I$, we computed a tree-specific explanation t for $\boldsymbol{x}$ given P using the code furnished in the PyXAI library (Audemard et al. 2024). This explanation t gives rise to the classification rule $R = t \Rightarrow y$ when $P(\boldsymbol{x}) = 1$ and to the classification rule $R = t \Rightarrow \overline{y}$ when $P(\boldsymbol{x}) = 0$.

- As to the retraining approach, at each step, a small sample of classified instances $(\boldsymbol{x}', P(\boldsymbol{x}))$ containing $(\boldsymbol{x}, P(\boldsymbol{x}))$ and such that t covers $\boldsymbol{x}'$ is added to L. More precisely, a limited ratio r ($r = 1\%$ in the experiments) of the total number of instances that can be produced using the n Boolean conditions occurring in P is generated and the number of instances in the sample is limited to a preset bound b (equal to 100 in the experiments). This restriction is necessary to prevent an unmanageable growth of the training set L at each step since the number of instances covered by t is exponential in $n - |t|$. The sample is obtained by randomly choosing, according to a uniform distribution, conditions from P not present in t until $max(r \times 2^{n-|t|}, b)$ instances have been generated. Only those instances that are feasible given the underlying domain theory Th are retained. Then, every instance $(\boldsymbol{x}', c)$ from the resulting training set such that the premises t of R cover $\boldsymbol{x}'$ and the conclusion of R is different from c is removed from the training set. Finally, a new training of

the model I using the updated training set is achieved.

- As to the rectification approach, the current decision tree I is corrected with the classification rule R, resulting in a new decision tree.

After each correction, $T_I^{\pm}$ has been recalculated before moving on to the next correction (this is necessary as I has been modified).

For each of the two correction approaches and for each of the two configurations tested, we measured after each correction (for each resulting decision tree I) the accuracy I_P of the decision tree I relative to P (this accuracy is estimated on the test set T), and the size of the decision tree I (the number N of its nodes) and the depth D of I.

In order to be sure that computational benefits may result from the distillation process, it was also important *to check that the time spent in distilling P into a decision tree can be balanced*. We made some experiments to test whether this is actually the case: using the rectification approach, P has been distilled into a decision tree in an incremental way, up to the step f from which $I_P = 100\%$ (i.e., when the resulting tree classifies all the instances of T as the boosted tree P) and the overall compilation / distillation time required to reach such a tree I has been measured. This has been achieved for each of the 10 decision trees learned from L at start, using the default configuration (i.e., the depth of the initial tree was not bounded a priori). Then, for each dataset, 100 instances $\boldsymbol{x}$ were picked up uniformly at random. A sufficient reason for each $\boldsymbol{x}$ given I has been computed for each I using a deletion-based algorithm (Ignatiev, Narodytska, and Marques-Silva 2019) and a sufficient reason for each $\boldsymbol{x}$ given P has been computed using the state-of-the-art algorithm from (Ignatiev et al. 2022). For the two algorithms, a timeout of 180 seconds was considered per instance.

Datasets used in the experiments are described in Table 1. We kept in the experiments only those datasets from UCI and openML leading to boosted trees P achieving higher accuracies than the corresponding decision trees I considered at start, i.e., at step 0. From left to right, the table indicates the number $|E|$ of examples (instances) in the dataset, the number $|F|$ of features (attributes) used to describe the instances initially, the number $|B|$ of Boolean conditions used in the dataset once binarized using P, the median accuracy $\%I_o$ (resp. $\%I_d$) obtained for the 10 decision trees considered under the optimized (resp. default) configuration, the accuracy $\%P$ of the boosted tree, the repository from which the dataset comes from, and finally the median number $|T^{\pm}_{I_o}|$ (resp. $|T^{\pm}_{I_d}|$) of instances of the test set T classified differently by I_o (resp. I_d) and P.

All the experiments have been conducted on computers with 2 quad-core Intel(R) Xeon(R) CPU E5-2643 0 @ 3.30GHz, each equipped with 32GiB of memory.

Empirical results Table 2 provides, for each dataset, the median values of I_P (in %) and the median number of nodes N of the corrected tree for each of the two correction methods (rec is rectification, ret is retraining). Table 3 provides the same type of information as Table 2 when the depths of the decision trees have been optimized.

Dataset		Correction steps (default configuration)															
		0			1			2				Final step (rec)					
		I_P	N	D	I_P	N	D	I_P	N	D		I_P	N	D	f		
bank	rec	88.76	512	22	88.83	562	24	88.90	602	27		100.0	12273	39	153		
	ret	88.76	512	22	89.60	520	23	89.75	521	22		94.17	941	34	-		
biodegradation	rec	85.01	183	12	85.33	326	30	85.64	571	33		100.0	76331	50	47		
	ret	85.01	183	12	83.28	192	12	84.06	194	12		90.85	339	28	-		
australian	rec	84.54	118	10	85.02	225	18	85.50	302	19	⋯	100.0	2527	24	30		
	ret	84.54	118	10	89.13	94	10	89.85	94	9		97.10	221	15	-		
bupa	rec	92.30	52	9	93.26	81	12	94.23	90	13		100.0	242	16	8		
	ret	92.30	52	9	89.42	48	7	87.98	49	7		92.30	78	10	-		
german	rec	93.33	76	7	93.66	117	15	94.0	157	16		100.0	1099	21	20		
	ret	93.33	76	7	95.33	77	8	94.83	75	8		97.66	105	9	-		
contraceptive	rec	71.49	676	19	71.94	663	19	72.39	681	19	⋯	100.0	4543	25	113		
	ret	71.49	676	19	85.29	392	14	86.19	385	14		94.57	559	23	-		
cleveland	rec	85.71	72	7	87.36	101	10	89.01	113	10		100.0	336	13	12		
	ret	85.71	72	7	89.01	61	7	86.81	65	7		95.60	91	8	-		
compas	rec	86.28	1022	16	87.87	550	15	87.93	554	15		100.0	465	15	71		
	ret	86.28	1022	16	99.62	314	12	99.59	314	12		99.64	341	12	-		
cnae	rec	98.61	56	13	98.91	34	8	99.22	42	8		100.0	79	9	5		
	ret	98.61	56	13	99.69	45	13	99.38	51	13		99.69	53	13	-		
breast-tumor	rec	64.53	96	7	65.69	99	11	66.86	111	12		100.0	799	17	30		
	ret	64.53	96	7	73.25	88	7	72.67	86	7		77.90	103	7	-		
balance_0_vs_1	rec	90.69	164	12	91.22	168	12	92.02	174	12		100.0	201	14	18		
	ret	90.69	164	12	92.28	154	13	92.55	154	12		94.14	163	12	-		
balance_0_vs_2	rec	91.66	97	9	92.64	103	10	93.62	107	10		100.0	135	9	8		
	ret	91.66	97	9	92.64	96	9	93.13	97	9		97.05	113	11	-		
balance_1_vs_2	rec	81.79	15	4	82.36	17	4	84.68	23	5		100.0	111	9	25		
	ret	81.79	15	4	82.08	15	4	81.50	15	4		80.92	15	5	-		

Table 2: Empirical accuracy I_P relative to P (in %), median number of nodes N, and median depth D of decision trees for different datasets and each correction method when the decision trees are learned under the default configuration. f is the median number of steps at which $T_I^{\pm}$ becomes empty when correction is performed by rectification.

In Tables 2 and 3, a few correction steps are highlighted. Step 0 is about the initial decision tree, step 1 (resp. 2, f) is about the decision tree obtained after 1 (resp. 2, f) correction steps.

Whatever the configuration used for learning the trees, after each rectification, *it is guaranteed that I_P strictly increases* since at least the instance triggering the correction step has been corrected at each step. The maximum number f of steps required to correct every instance of $T_I^{\pm}$ by rectifying it is thus well-defined. Comparing this number f to the number $|T_I^{\pm}|$ of instances that were initially misclassified (see Table 1), one may observe in Tables 2 and 3 that rectification with rules covering a possibly large number of instances can have a very significant impact on the number of correction steps required for achieving a complete distillation over T (for example, for `compas` under optimized configuration, 38 steps have been sufficient while 125 instances from T were initially misclassified).

In contrast, the empirical results show that successive corrections by retraining can decrease the value of I_P, making a *complete correction impossible* in general. This holds under the optimized configuration, but also under the default configuration. In this last case, one knows that adding

Dataset		Correction steps (optimized configuration)														
		0			1			2				Final step (rec)			
		I_P	N	D	I_P	N	D	I_P	N	D		I_P	N	D	f
bank	rec	90.75	408	14	90.82	454	22	90.89	426	25		100.0	9322	36	125
	ret	90.75	408	14	90.82	401	14	91.04	403	14		94.03	560	14	-
biodegradation	rec	86.75	30	3	87.06	100	29	87.38	209	30		100.0	57826	47	42
	ret	86.75	30	3	85.80	31	3	83.28	31	3		76.34	29	3	-
australian	rec	87.43	45	4	87.92	69	14	88.40	130	17	⋯	100.0	1477	22	25
	ret	87.43	45	4	90.82	43	4	91.06	36	4		79.46	43	4	-
bupa	rec	92.30	46	7	93.26	73	13	94.23	79	13		100.0	266	14	8
	ret	92.30	46	7	87.01	46	7	86.53	44	7		86.53	59	7	-
german	rec	95.5	44	4	95.83	57	13	96.16	75	14		100.0	585	20	13
	ret	95.5	44	4	97.5	41	4	98.0	43	4		98.0	45	4	-
contraceptive	rec	82.57	24	3	82.80	39	11	83.48	64	13	⋯	100.0	1200	20	69
	ret	82.57	24	3	87.33	22	3	86.42	23	3		86.19	29	3	-
cleveland	rec	86.81	29	3	88.46	36	6	90.10	48	7		100.0	247	11	11
	ret	86.81	29	3	89.56	29	3	86.81	29	3		93.40	29	3	-
compas	rec	93.25	63	4	93.60	45	6	94.24	51	7		100.0	193	12	38
	ret	93.25	63	4	93.43	60	4	93.43	60	4		91.95	57	4	-
cnae	rec	98.76	41	7	99.07	30	7	99.38	37	7		100.0	65	8	5
	ret	98.76	41	7	99.38	33	7	99.69	35	7		99.69	41	7	-
breast-tumor	rec	66.27	148	14	67.44	159	14	68.60	171	15		100.0	603	16	29
	ret	66.27	148	14	74.41	141	14	73.83	141	14		84.88	153	14	-
balance_0_vs_1	rec	96.80	39	4	97.34	29	7	97.87	41	7		100.0	65	9	6
	ret	96.80	39	4	97.34	41	4	97.60	40	4		96.27	39	4	-
balance_0_vs_2	rec	93.62	24	3	94.60	31	7	95.58	39	7		100.0	74	9	7
	ret	93.62	24	3	94.11	25	3	94.11	25	3		90.68	25	3	-
balance_1_vs_2	rec	91.32	115	8	91.90	121	8	92.48	127	8		100.0	197	8	16
	ret	91.32	115	8	92.48	118	8	92.77	118	8		96.53	138	8	-

Table 3: Empirical accuracy I_P relative to P (in %), median number of nodes N, and median depth D of decision trees for different datasets and each correction method when the depth of the decision trees has been optimized. f is the median number of steps at which $T_I^{\pm}$ becomes empty when correction is performed by rectification.

$(\boldsymbol{x}, P(\boldsymbol{x}))$ to the training set before retraining I is enough to guarantee that $\boldsymbol{x}$ will be classified as required by P after retraining. However, this is not enough to ensure that the accuracy of the decision tree relative to P increases gradually with the correction steps. Indeed, a misclassified instance corrected by retraining at a certain step *may become misclassified again after subsequent retraining steps*.

Tables 2 and 3 also show that the growth of the size of the trees (and of their depth) is more significant when rectification is used than when retraining is used, and that the size of rectified trees I can become quite large, thus possibly questioning the benefits offered by the distillation process.

The empirical results presented in Table 4 concern, on the one hand, the computational benefits of distilling P into I when the goal is to derive sufficient reasons, and on the other hand, a comparison of the computation times required by the two correction approaches used (rectification and retraining). The results show that in spite of the growth of the size of the trees,[2] the distillation of P into a decision tree I is useful when the XAI objective that is pursued is to compute sufficient reasons. In Table 4, t_C is the average distillation time (in seconds) over the 10 trees. For each tree, the distillation time includes the time required to identify instances $\boldsymbol{x}$ that are classified differently by the current decision tree I and by P, the time required to compute an abductive explanation t for each such $\boldsymbol{x}$ given P, and the time needed to rectify the current I by the classification rule corresponding to t (see Proposition 3).[3] t_I and t_P are respectively the mean computation times (in seconds) achieved by the algorithms for computing a sufficient reason over the set of instances (only instances for which the computation of a sufficient reason terminated in due time have been considered when evaluating the average). to_I is the mean number of timeouts for the 10 trees and to_P is the number of timeouts reached by the algorithms for computing a sufficient reason over the 100 instances. The improvement ratio $\alpha = \frac{t_P}{t_I}$ measures how many times faster is, on average, the computation of a sufficient reason when based on I instead of P, while $\beta = \lceil \frac{t_C}{t_P - t_I} \rceil$, referred to as the break-even point, indicates (when positive) the number of explanation queries from which the compilation effort is compensated.

As Table 4 points it out, significant computational benefits about the derivation of sufficient reasons can be achieved by distilling P into I. On the one hand, no timeouts have been observed when the computation of sufficient reasons was based on I, whatever the dataset and the decision tree at start: for each instance, a sufficient reason has been computed in less than 180 seconds. In contrast, the number of timeouts reached when sufficient reasons were derived from P varies with the dataset and can be very significant (exceeding 50% of the instances tested for the datasets `bank` and `contraceptive`). On the other hand, the improvement ratio α was huge (more than two orders of magnitude for every dataset). The break-even point $\beta = 1$ was equal to its least possible value when $\alpha > 1$ whatever the dataset. This means that the distillation of P into I is valuable even when the computation of a sufficient reason is required for a single instance. For more details, we refer the reader to the supplementary material (Audemard et al. 2025). Of course, it must be kept in mind that the decision tree I obtained after f rectification steps is not equivalent to P in general (it is only equivalent to P on T), which limits the scope of the comparison. However, this is consistent with our objective (see Section 4), since our goal is not to fully distill P into a decision tree: in our approach, the distillation process is achieved incrementally, in a lazy and opportunistic fashion.

The proposed rectification-based approach to distillation thus appears as practical enough, despite the growth of the decision tree.

Table 4 also gives the cumulative average times (in seconds) required by the successive correction operations up to the final rectification step f. d_{rec} and d_{ret} represent respectively the cumulative times for rectification and retraining in the default case, while o_{rec} and o_{ret} represent respec-

[2] Notably, for each dataset, the 10 decision trees one started with correspond to the default configuration. As reflected by the values of columns f, N, and D in Tables 2 and 3, this choice was less favourable than the choice of the optimized configuration in terms of numbers of rectification step and of size and depth of the tree I obtained once all the rectifications have been achieved.

[3] Accordingly, it can be observed that t_C is larger than the average cumulated rectification time d_{rec} pointed out in Table 4.

Dataset	t_C	to_I	t_I	to_P	t_P	α	β	d_{rec}	d_{ret}	o_{rec}	o_{ret}
bank	39.74	0	0.62	54	102.25	164.91	1	13.22	7978.16	4.18	3141.64
biodegradation	22.84	0	0.55	20	87.95	156.27	1	8.01	367.46	3.59	196.69
australian	3.12	0	0.02	0	20.21	1010.5	1	0.65	80.67	0.047	44.69
bupa	0.25	0	0.001	0	1.50	1500	1	0.007	4.29	0.003	2.78
german	0.61	0	0.012	0	8.02	668.33	1	0.048	29.59	0.008	8.87
contraceptive	4.84	0	0.02	85	71.68	3634	1	1.37	860.04	0.09	132.35
cleveland	0.56	0	0.005	0	5.24	1048	1	0.007	9.44	0.002	6.73
compas	4.95	0	0.007	0	23.31	3330	1	0.09	142.01	0.012	35.65
cnae	0.019	0	0.0003	0	0.05	166.66	1	1.2e-3	0.54	5.5e-4	0.19
breast-tumor	0.79	0	0.004	19	78.26	19565	1	0.03	2.74	0.03	2.46
balance_0_vs_1	0.19	0	0.001	0	0.95	950	1	8.8e-3	1.06	1.6e-3	0.34
balance_0_vs_2	0.07	0	0.0009	0	0.20	222.22	1	4.2e-3	0.29	4.7e-3	0.59
balance_1_vs_2	0.13	0	0.001	0	0.30	300	1	5.5e-3	2.46	8.3e-3	1.04

Table 4: Evaluation of the computational benefits of distilling P into I in the objective of deriving sufficient reasons (compilation time t_C, numbers of timeout to_P and to_I, mean computation times t_P and t_I, improvement ratio α, and break-even point β) and comparison of the computation times required by each of the two methods (rectification and retraining), when both the default case and the optimized case are considered.

tively the cumulative times for rectification and retraining in the optimized case. The results show that the computation times required to achieve the rectification process are small enough (and often two orders of magnitude smaller than the corresponding times when retraining is used).

6 Other Related Work

In ML, a number of approaches have been designed so far to tackle the problem of explaining tree ensemble models to mitigate the trust risks associated with their lack of transparency, see e.g., (Friedman and Popescu 2008; Deng 2018; Obregon and Jung 2023). Most of the time, (possibly overlapping) sets of classification rules (i.e., decision sets) are targeted (in some approaches (Sagi and Rokach 2021; Sagi and Rokach 2020), those sets of rules are finally combined into decision trees). In decision sets, rule conflicts or incomplete coverage are practically handled using heuristics (e.g., ordering rules into a decision list, or adding a default rule), leading to models that can be visually simpler and more compact than decision trees. However, decision sets lack *computational intelligibility*: answering formal XAI queries from them is generally intractable.

Conversely, a decision tree intrinsically guarantees that rules are mutually exclusive and collectively exhaustive. While this strict structural property can lead to larger visual models, it is precisely this property that makes many XAI queries (e.g., deriving subset-minimal abductive explanations or testing verification queries) computationally tractable in polynomial time, whereas they are NP-hard for decision sets (and even, more specifically, for CNF or DNF classifiers) (Audemard et al. 2021).

Additionally, it is important to distinguish our logic-based approach from statistical approximation techniques like those furnished in the *imodels* package (https://github.com/csinva/imodels) (Singh et al. 2021). Methods in *imodels* train a surrogate rule-based model to globally mimic a black box, but offer no formal guarantees that the surrogate will perfectly match the black box on specific instances.

Alternatively, (Li et al. 2025) shows how to turn tree ensembles into decision trees without considering sets of rules. The resulting trees approximate the tree ensembles considered as input. Experiments have shown that the approximation achieved can be of good quality, but, as mentioned by the authors, the scalability of the approach to high-dimensional datasets remains a challenge.

Closer to our approach is the "Born-Again Tree Ensembles (BATE)" approach presented in (Vidal and Schiffer 2020), and more specifically, the heuristic approach reported in this paper. This heuristic approach aims to compute a decision tree that is *equivalent* to a given weighted random forest, thus offering the very same guarantees as our own approach. Beyond the fact that the inputs considered in the two approaches are not identical (weighted random forests vs. boosted trees), the methods at work to generate decision trees (dynamic programming vs. successive rectifications) in the two approaches are completely different. Furthermore, no data preprocessing is required in our approach (while, in BATE, continuous features are binned into 10 ordinal scales). Another significant difference is that our approach is lazy and opportunistic (which is not the case of BATE). This difference may explain the large discrepancy we observed empirically between the two approaches in terms of scalability. Indeed, we ran heuristic BATE (see https://github.com/vidalt/BA-Trees) on our datasets for random forests with a reduced number of trees (10 trees) of limited depth and parameter -obj set to 4, considering a timeout of 4 hours. heuristic BATE succeeded in computing in due time a decision tree equivalent to the input random forest only for two datasets (contraceptive - the resulting tree contains 281 752 nodes and it has been computed in 174 seconds - and compas - the resulting tree contains 7 806 968 nodes and it has been computed in 2248 seconds). A timeout occurred for all the remaining datasets. We also tried to increase the number of trees and/or the tree depth in order to increase the predictive performance of the forest one started with but under such conditions, heuristic BATE failed to return a decision tree within 4 hours.

In other approaches to the distillation of tree ensembles into decision trees (see e.g., (Breiman and Shang 1996) (Zhou, Zhou, and Hooker 2023)), the transformation of the input tree ensemble relies on an *improved learning step*: the black box classifier P is simply used as an oracle for generating new instances in order *to complete the training set used to learn* I. No instances are removed from the training set. Thus, a decision tree I is built up once for all (no correction of it is to be done once it has been derived). The issues to be considered for computing I are the generation of new instances (connected to the splitting rule used), the stopping rule (deciding when the splitting process must be stopped), and the pruning rule. A common principle guiding the generation of new instances ("manufacturing data" (Breiman and Shang 1996)) is to ensure to have sufficiently many instances at each decision node to decide when to split and how to split. Those additional instances are useful to stabilize the greedy splitting strategy that is leveraged to con-

struct the decision tree I, thereby improving its accuracy (Zhou, Zhou, and Hooker 2023). In order to satisfy those conditions, the number of extra instances to be generated is typically huge, leading to high learning times. Despite it, there is in general no guarantee that the predictive performance of I is close to the one of P.

In contrast to all the approaches mentioned above to distilling a tree ensemble into a decision tree, our approach is *correction-guided*. Rectification is leveraged to correct the current decision tree in an opportunistic way. No new instance is to be generated but only one explanation for each instance $\boldsymbol{x}$ that triggers a correction. Furthermore, a key feature of rectification is that *it offers logical guarantees*: it ensures that the corrections that are targeted are effective, which is not the case of distillation approaches based on a completion of the training set and/or on (re)training. Especially, the trees resulting from such tree distillation approaches are not guaranteed to be equivalent to the black box considered at start, while this is ensured by our approach "in the limit", i.e., provided that all instances to be corrected are rectified at some step.

7 Conclusion

We have presented and evaluated a new approach for distilling a boosted tree P into a decision tree I. Our approach is based on the rectification operation that provides guarantees that the corrections that are requested are effective. In contrast to a distillation approach that would boil down to translating P into I as a whole, our approach is lazy and opportunistic: I is corrected only when an instance $\boldsymbol{x}$ encountered at inference time is such that $I(\boldsymbol{x}) \neq P(\boldsymbol{x})$. This makes it possible to control the size of the resulting (rectified) decision tree I, therefore enabling one to stop the rectification process as soon as desired. Experiments have shown that our approach to distillation may provide interesting results compared to previous approaches, especially distillation by retraining, completion of the training set or using the heuristic BATE approach.

Interestingly, algorithms for minimizing decision trees could be considered within the proposed approach. Basically, the idea would be to interleave minimization steps with correction steps so as to increase the number of correction steps that could be handled while ensuring that the size of the resulting decision tree remains "small enough". However, since the minimization problem for decision trees is NP-hard and even hard to approximate up to any constant factor (Sieling 2008), taking advantage of minimization steps could have a significant impact on the overall rectification time.

Many additional avenues for future research can be envisioned. One promising direction is the definition of a stopping criterion for the distillation process. In particular, monitoring the marginal improvement in agreement relative to the increase in tree size could help identify an optimal trade-off, thereby providing practical guidance to users on whether continuing the distillation process is worthwhile. Future work will also investigate the method's sensitivity to initial conditions, the impact of the correction order, and its scalability when applied to large-scale datasets.

Acknowledgements

Many thanks to the anonymous reviewers for their numerous comments and insights. This work has benefited from the support of the AI Chair EXPEKCTATION (ANR-19-CHIA-0005-01) and of the France 2030 MAIA Project (ANR-22-EXES-0009) of the French National Research Agency (ANR).

AI Declaration

The authors have not employed any Generative AI tools.

References

Arrieta, A. B.; Díaz, N.; Ser, J. D.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; Chatila, R.; and Herrera, F. 2020. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* 58:82–115.

Asadulaev, A.; Kuznetsov, I.; and Filchenkov, A. 2019. Interpretable few-shot learning via linear distillation. *CoRR* abs/1906.05431.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.-M.; and Marquis, P. 2021. On the computational intelligibility of boolean classifiers. In *Proc. of KR'21*, 74–86.

Audemard, G.; Lagniez, J.-M.; Marquis, P.; and Szczepanski, N. 2023. Computing abductive explanations for boosted trees. In *Proc. of AISTATS'23*, 4699–4711.

Audemard, G.; Lagniez, J.; Marquis, P.; and Szczepanski, N. 2024. Pyxai: An XAI library for tree-based models. In *Proc. of IJCAI'24*, 8601–8605.

Audemard, G.; Coste-Marquis, S.; Marquis, P.; Sabiri, M.; and Szczepanski, N. 2025. A rectification-based approach for distilling boosted trees into decision trees. https://archive.softwareheritage.org/swh:1:dir:a618d0829abc70acdd2a975745aa6a76f20a0efd.

Breiman, L., and Shang, N. 1996. Born again trees. Technical Report Technical Report (1(2), 4), University of California, Berkeley, Berkeley, CA.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Wadsworth.

Chen, T., and Guestrin, C. 2016. XGBoost: A scalable tree boosting system. In *Proc. of KDD'16*, 785–794.

Coste-Marquis, S., and Marquis, P. 2021. On belief change for multi-label classifier encodings. In *Proc. of IJCAI'21*, 1829–1836.

Coste-Marquis, S., and Marquis, P. 2023. Rectifying binary classifiers. In *Proc. of ECAI'23*, 485–492.

Darwiche, A., and Hirth, A. 2020. On the reasons behind decisions. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI'20)*, 712–720.

de Colnet, A., and Marquis, P. 2023. On translations between ML models for XAI purposes. In *Proc. of IJCAI'23*, 3158–3166.

Deng, H. 2018. Interpreting tree ensembles with intrees. *Int. J. Data Sci. Anal.* 7:277—-287.

Freund, Y., and Schapire, R. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55(1):119–139.

Friedman, J., and Popescu, B. 2008. Predictive learning via rule ensembles. *Ann. Appl. Stat.* 2(3):916—-954.

Frosst, N., and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *CoRR* abs/1711.09784.

Gou, J.; Yu, B.; Maybank, S.; and Tao, D. 2021. Knowledge distillation: A survey. *Int. J. Comput. Vis.* 129(6):1789–1819.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using maxsat for efficient explanations of tree ensembles. In *Proc. of AAAI'22*, 3776–3785.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-based explanations for machine learning models. In *Proc. of AAAI'19*, 1511–1519.

Li, Z.; Du, X.; Xu, A.; Wu, T.; and Cao, Y. 2025. Explaining tree ensembles through single decision trees. *Information Fusion* 123:103244.

Obregon, J., and Jung, J. 2023. Rulecosi+: Rule extraction for interpreting classification tree ensembles. *Inf. Fusion* 89:355–381.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* 1(5):206–215.

Sagi, O., and Rokach, L. 2020. Explainable decision forest: Transforming a decision forest into an interpretable tree. *Inf. Fusion* 61:124–138.

Sagi, O., and Rokach, L. 2021. Approximating xgboost with an interpretable decision tree. *Inf. Sci.* 572:522–542.

Schapire, R., and Freund, Y. 2014. *Boosting: Foundations and Algorithms*. MIT Press.

Sieling, D. 2008. Minimization of decision trees is hard to approximate. *J. Comput. Syst. Sci.* 74(3):394–403.

Singh, C.; Nasseri, K.; Tan, Y. S.; Tang, T. M.; and Yu, B. 2021. imodels: a python package for fitting interpretable models. *J. Open Source Softw.* 6(61):3192.

Vidal, T., and Schiffer, M. 2020. Born-again tree ensembles. In *Proc. of ICML'20*, volume 119 of *Proceedings of Machine Learning Research*, 9743–9753. PMLR.

Wood-Doughty, Z.; Cachola, I.; and Dredze, M. 2022. Model distillation for faithful explanations of medical code predictions. In *Proc. of BioNLP@ACL'22*, 412–425.

Zhou, Y.; Zhou, Z.; and Hooker, G. 2023. Approximation trees: statistical reproducibility in model distillation. *Data Mining and Knowledge Discovery*.

Zhou, Z.-H. 2019. Abductive learning: towards bridging machine learning and logical reasoning. *Science China Information Science* 62(7):76101:1–76101:3.