On Equivalences of Decision Trees and tree-based Ensemble Learners

T. Florian Kauffeldt[*], Tobias Brock[†], Paul Stiller[‡] and Carsten Lanquillon[**]

Abstract

This article aims to improve the interpretability of tree-based ensemble learners. For this purpose, we define a series combination, which combines the underlying trees of an ensemble. Applying this method, the existence of prediction equivalent decision trees is proven. However, the resulting composed decision tree may easily demonstrate high complexity, therefore we prove the reduction of its order (number of vertices) by eliminating empty vertices and non-empty redundant vertices. Furthermore, general algorithms for iteratively composing decision trees and the reduction of their order are provided, as well as an algorithm, which combines both concepts. In the appendix, a practical implementation of prediction equivalent decision trees is demonstrated in Python.

Keywords: Decision Trees, Ensemble Learners, Prediction Equivalence, Interpretability

[*]Corresponding author. Faculty of International Business, Heilbronn University, Heilbronn, Germany. Email: florian.kauffeldt@hs-heilbronn.de

[†]Faculty of International Business, Heilbronn University, Heilbronn, Germany. Email: t.brock335@googlemail.com email

[‡]Faculty of Economics and Transport, Heilbronn University, Heilbronn, Germany. Email: paul.niklas.stiller@googlemail.com

[**]Faculty of Economics and Transport, Heilbronn University, Heilbronn, Germany. Email: carsten.lanquillon@hs-heilbronn.de

1. Preliminaries

A *rooted directed graph* (V, E, v_0, T) consists of a set V (*the vertices or nodes*), a set $E \subseteq V \times V$ (*the edges*), a node $v_0 \in V$ (*the initial node or root*), and a set $T \subset V$ (*the terminal nodes or leaves*).

Notice that the set E defines predecessors $p(v)$ and successors $s(v)$ for each vertex $v \in V$:

$$p(v) = \{v' : (v', v) \in E\} \text{ and } s(v) = \{v' : (v, v') \in E\},$$

where the root has no predecessor ($p(v_0) = \varnothing$) and the terminal nodes have no successors ($s(t) = \varnothing$ for $t \in T$).

We call a rooted directed graph *tree* τ if for every pair $v', v'' \in V$, there is a unique path from v' to v''. This implies that each vertex has at most one predecessor ($p(v)$ is a function), there are no loops ($p(v) \neq v$ for all $v \in V$), there are no cycles, and $|E| = |V| - 1$, where $|E|$ and $|V|$ denote the size and order respectively.

The following operation with trees will be needed in the sequel:

Definition 1 (Series Combinations of Trees)**.** Given two trees τ_1, τ_2, let $T_{\tau_1} = \{t_1, \ldots, t_{|T_{\tau_1}|}\}$ be the set of terminal nodes of τ_1. Let $\mathcal{M}(\tau_2) = \{\gamma_1, \ldots, \gamma_{|T_{\tau_1}|}\}$ be a set of $|T_{\tau_1}|$ trees with pairwise disjoint vertices ($V_{\gamma_i} \cap V_{\gamma_j} = \varnothing$ for all i, j) that are edge-preserving isomorphic to τ_2: $\gamma_i \cong \tau_2$ for all i such that each root in $\mathcal{M}$ corresponds to a unique terminal node of τ_1: $v_0^{\gamma_i} = t_i$ for $i = 1, \ldots, |T_{\tau_1}|$.

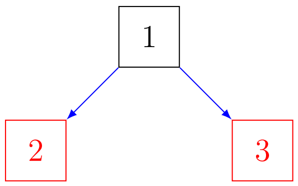
A *series combination* $\tau_1 \otimes \tau_2$ is then a tree defined by the union of τ_1 and the trees in $\mathcal{M}(\tau_2)$:

$$\tau_1 \otimes \tau_2 = \tau_1 \cup \bigcup_{\gamma \in \mathcal{M}(\tau_2)} \gamma.$$

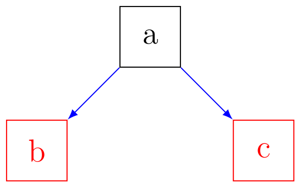
Remark 1. In general, series combinations are not commutative: $\tau_1 \otimes \tau_2 \neq \tau_2 \otimes \tau_1$.

Remark 2. Series combinations of several trees are performed sequentially. For instance, $\tau_1 \otimes \tau_2 \otimes \tau_3$ corresponds to a series combination $\tau' \otimes \tau_3$, where $\tau' = \tau_1 \otimes \tau_2$.

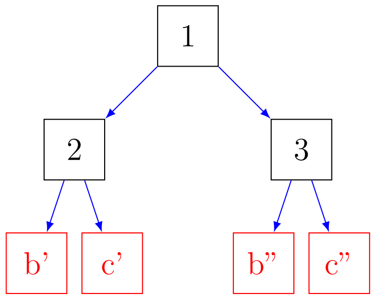
Intuitively, we can think of a series combination as appending the second tree to the first tree. The following figure illustrates a series combination:



(a) Tree τ_1



(b) Tree τ_2



(c) $\tau_1 \otimes \tau_2$

Figure 1: A series combination

2. Classification and Regression Trees and Ensemble Learning

A prediction problem is the task of estimating a target variable $y \in \mathcal{Y}$ based on K features or inputs $(x_1, \dots, x_K) \in \mathcal{X} = \bigtimes_{k=1}^{K} X_k$. A prediction rule $r : \mathcal{X} \to \mathcal{Y}$ solves this task. A prediction problem $(\mathcal{X}, \mathcal{Y})$ where the target variable takes on a finite number of values is called *classification problem*. The problem is called *regression problem* if the target variable is continuous.

Definition 2 (Decision Tree)**.** Given a prediction problem $(\mathcal{X}, \mathcal{Y})$, a *decision tree* $\delta_{(\mathcal{X},\mathcal{Y})}$ is a structure (τ, f, σ, d) consisting of

(i) a tree τ

(ii) a function $f : V \setminus T \to \{1, \dots, K\}$ that assigns a feature to each non-terminal node

(iii) a split of the feature space $X_{f(v)}$ at each vertex v represented by a set-valued function σ, formally, let $proj_1(e)$ be the first component of $e \in E$, then
$$\sigma : E \to \{S : \ S \subset X_{f(proj_1(e))}\}$$
such that, for every vertex v, the set $\{\sigma(e) : proj_1(e) = v\}$ is a partition of $X_{f(v)}$.

(iv) a function $d : T \to \mathcal{Y}$ that assigns a prediction to each terminal node.

A decision tree is called *classification tree* if the underlying estimation problem is a classification problem. Otherwise, it is called *regression tree*. The umbrella term "classification and regression trees (CART)" is introduced in Breiman et al. (1984).

Definition 3 (Ensemble Learner)**.** Given a prediction problem $(\mathcal{X}, \mathcal{Y})$, a (tree-based) *ensemble learner* $\eta_{(\mathcal{X},\mathcal{Y})}$ is a structure (δ, w, h) consisting of

(i) a vector of decision trees $\delta = (\delta_1, \ldots, \delta_M)$

(ii) a vector of weights $w = (w_1, \ldots, w_M)$

(iii) a rule $h_\eta : \mathcal{X} \to \mathbb{R}$ that aggregates the prediction rules of the trees $(h_{\delta_1}, \ldots, h_{\delta_M})$ and their weights $(w_1, \ldots, w_M)$

For our main theorems, we need the following notion of equivalence between learners:

Definition 4. Given a prediction problem $(\mathcal{X}, \mathcal{Y})$, two learners γ_1, γ_2 are called *prediction equivalent* if their prediction rules are equivalent: $h_{\gamma_1}(x) = h_{\gamma_2}(x)$ for all $x \in \mathcal{X}$.

Remark 3. Notice, if $\mathcal{Y}$ is a continuous space, then prediction equivalence implies the same regression prediction for both learners.

Remark 4. Given a classification problem $(\mathcal{X}, \mathcal{Y})$ with $\mathcal{Y} = \{1, ..., g\}$. An ensemble classifier is an ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$ that outputs a vector of prediction rules $\boldsymbol{h}_\eta = (h_{\eta,1}, ..., h_{\eta,g})$ together with a classification rule r:

$$r(\boldsymbol{h}_\eta(x)) = \underset{l \in \{1,...,g\}}{\arg\max}\, h_{\eta,l}(x).$$

In a binary classification problem, where $\mathcal{Y} = \{0, 1\}$, a single prediction rule $h_\eta = h_1$ suffices. A classification rule can then be determined by setting some threshold value c such that

$$r(h_\eta(x)) = \begin{cases} 1 \text{ if } h_\eta(x) > c \\ 0 \text{ else} \end{cases} .$$

3. Theorems

In our main theorems, we show that for every ensemble learner, there exists a prediction equivalent decision tree and the reduction of the tree order.

Lemma 1. *Each decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$ implies a prediction rule $h : \mathcal{X} \to \mathcal{Y}$.*

Proof. Given a decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$. Let $E(t)$ be the set of edges along the path from the root to some terminal node t. Define the condition $C_k(t)$ for feature k as follows: if there exists an $e \in E(t)$ such that $\sigma(e) \subseteq X_k$, then $C_k(t) = \bigcap_{e \in E(t), \sigma(e) \subseteq X_k} \sigma(e)$, otherwise, $C_k(t) = X_k$. Notice that $C(t) = \bigtimes_{k=1}^{K} C_k(t)$ is a subset of $\mathcal{X}$. At each vertex the splits of the space of one feature are a partition. Hence, for each input vector $(x_1, \ldots, x_k) \in \mathcal{X}$, there is exactly one terminal node t such that $(x_1, \ldots, x_k) \in C(t)$. Now, define a function that maps each input vector to the prediction of its unique terminal node $d(t)$, which is a prediction rule. $\square$

Theorem 1. *For any ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$, there exists a prediction equivalent decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$.*

Proof. Given an ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})} = (\delta_1, \ldots, \delta_M, w_1, \ldots, w_M, h_\eta)$, let τ_m be the underlying tree of δ_m for $m = 1, \ldots, M$. Define a tree τ^* by the series combination of all trees τ_m: $\tau^* = \bigotimes_{m=1}^{M} \tau_m$.

For $m > 1$, denote by $\mathcal{M}(\tau_m)$ the set of trees that are edge-preserving isomorphic to τ_m and let $V_{\mathcal{M}(\tau_m)} = \bigcup_{\gamma \in \mathcal{M}(\tau_m)} V_\gamma$ be the union of their vertices. Let $\zeta_{\tau_m} : V_{\mathcal{M}(\tau_m)} \to V_{\tau_m}$ be the function that maps the vertices of each tree in $\mathcal{M}(\tau_m)$ to the vertices of τ_m under the isomorphisms between each tree in $\mathcal{M}(\tau_m)$ and τ_m.

Denote by f_m the function of δ_m that assigns features to vertices and define a function f^* by $f^*(v) = f_1(v)$ for $v \in V_{\tau_1}$ and $f^*(v) = f_m(\zeta_{\tau_m}(v))$ for $v \in V_{\mathcal{M}(\tau_m)}$.

Denote by σ_m the function of δ_m that assigns feature splits to edges and define a function σ^* by $\sigma^*(v', v'') = \sigma_1(v', v'')$ for $(v', v'') \in E_1$ and $\sigma^*(v', v'') = \sigma_m(\zeta_{\tau_m}(v'), \zeta_{\tau_m}(v''))$ for $v', v'' \in V_{\mathcal{M}(\tau_m)}$.

Let T^* be the set of terminal nodes of τ^*. Let $C(t) \subset \mathcal{X}$ be the subset of the feature space associated with terminal node $t \in T^*$ (see proof of Lemma 1). Observe that $h_\eta(x') = h_\eta(x'')$ for all $x', x'' \in C(t)$. Now, define $d^*(t) = h_\eta(x)$ for $x \in C(t)$. Notice that the structure $\delta^*_{(\mathcal{X},\mathcal{Y})} = (\tau^*, f^*, \sigma^*, d^*)$

is a decision tree. Furthermore, by construction, it is prediction equivalent to $\eta_{(\mathcal{X},\mathcal{Y})}$. $\square$

Corollary 1. *For any ensemble classifier $\eta_{(\mathcal{X},\mathcal{Y})}$, there exists a prediction equivalent classification tree.*

Proof. Given an ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})} = (\delta_1, ..., \delta_m, w_1, ..., w_m, \boldsymbol{h})$. Apply a classification rule r to $\boldsymbol{h}_\eta : r(h_\eta(x))$. The result is an ensemble classifier. By Theorem 1, we have that for any ensemble learner, there exists a prediction equivalent decision tree $\delta^*_{(\mathcal{X},\mathcal{Y})}$. Let T^* be the set of terminal nodes in $\delta^*_{(\mathcal{X},\mathcal{Y})}$. Notice that $r(\boldsymbol{h}_\eta(x')) = r(\boldsymbol{h}_\eta(x''))$ for all $x', x'' \in C(t),\ t \in T^*$. Define $d^*(t) = r(\boldsymbol{h}_\eta(x))$ for all $x \in C(t)$ which is a classification rule. Therefore, we have that the structure $\delta^*_{(\mathcal{X},\mathcal{Y})} = (\tau^*, f^*, \sigma^*, d^*)$ is a classification tree that is by construction prediction equivalent to $\eta_{(\mathcal{X},\mathcal{Y})}$. $\square$

Before we prove our first reduction theorem, some more preliminary notation has to be discussed. Recall that the immediate predecessor of a vertex $v \in V$ is $p(v) = \{v' : (v', v) \in E\}$. Now, define the set of all predecessors for a vertex v as $v \uparrow = \bigcup_{i=0}^{i_0} p^i(v)$, where $p^i(v)$ denotes the ith predecessor of v, e.g., $p^0(v) = v$ and $p^2(v) = p(p(v))$. Notice that $p^{i_0}(v) = v_0$.

Definition 5. Given an ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$, a prediction equivalent decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$ has efficient paths if

(i) there exists no $v \in V$ such that $C_k(v) = \emptyset$ (empty vertices)

(ii) there exists no $t \in T$ such that $C_k(p^{\underline{i}}(t)) \cap C_k\left(p^{\bar{i}}(t)\right) = C_k\left(p^{\bar{i}}(t)\right)$, with $C_k\left(p^{\bar{i}}(t)\right) \neq \emptyset$ and $\underline{i} < \bar{i}$ (non-empty redundant vertices)

for all $k \in \{1, ..., K\}$.

Remark 5. Given a decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$ that is prediction equivalent to some ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$. The prediction at each terminal node $t \in T$ can be formulated as the inner product $d(t) = \langle \mathbf{h}, \mathbf{w} \rangle$, with $\mathbf{h} = (h_0, h_{\delta_1}, ..., h_{\delta_M})$ and $\mathbf{w} = (1, w_1, ..., w_M)$. Notice that for every $t'_m \in t \uparrow$, where t'_m denotes the terminal node of tree δ_m, we have that $d(t'_m) = \langle h_{\delta_m}, w_m \rangle$. $h_0 \in \mathbb{R}$ typically refers to an initial prediction or 0. Therefore, it holds that

$$d(t) = \langle \mathbf{h}, \mathbf{w} \rangle = h_0 + \sum_{m=1}^{M} \langle h_{\delta_m}, w_m \rangle = h_0 + \sum_{m=1}^{M} d(t'_m).$$

Theorem 2. *For every ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$, there exists a prediction equivalent decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$ such that $\delta_{(\mathcal{X},\mathcal{Y})}$ has efficient paths.*

Proof. Let $\delta^*_{(\mathcal{X},\mathcal{Y})} = (\tau^*, f^*, \sigma^*, d^*)$ be a prediction equivalent decision tree for an arbitrary ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$. Denote the set of vertices by V^* and the set of terminal nodes by T^*. Given a path $E^*(t)$ in V^*, with $t \in T^*$, the set of all vertices can be formulated as $t^* \uparrow$, which is a subset of V^*. If $C_k\left(p^{\underline{i}}(t)\right) \cap C_k\left(p^{\overline{i}}(t)\right) = C_k\left(p^{\overline{i}}(t)\right)$, where $C_k\left(p^{\overline{i}}(t)\right) \neq \emptyset$, then $p^{\underline{i}}(t), p^{\overline{i}}(t) \in t^* \uparrow$, are the two distinct bottom and top non-empty vertices respectively ($\underline{i} < \overline{i}$) for which an analogous condition of feature k holds. Formally, $\bigcap_{i \in \{\underline{i},...,\overline{i}\}} C_k\left(p^{i}(t)\right) = C_k\left(p^{\overline{i}}(t)\right) \neq \emptyset$, where $\{\underline{i},...,\overline{i}\}$ is a distinct set of indexes. Notice there can exist multiple subsets within a path $t^* \uparrow$ for which an intersection of an analogous condition of feature k exists. Denote by $I_k = \{\underline{i},...,\overline{i}\}$ any arbitrary subset of indexes for feature k under analogous conditions and let $\mathcal{I}_k = \bigcup_{I_k \subseteq \mathcal{I}_k} I_k$ be the (not necessarily disjoint) union of those subsets. Now, define a set for any $t \in T^*$ that contains the non-empty redundant vertices within $t^* \uparrow$ under analogous conditions of feature k as

$$R^k_{t^*\uparrow} = \bigcup_{\overline{i} \in \mathcal{I}_k} \left\{ p^{i}(t) : C_k\left(p^{i}(t)\right) \cap C_k\left(p^{\overline{i}}(t)\right) = C_k\left(p^{\overline{i}}(t)\right) \neq \emptyset \right\}.$$

Let $F \subseteq \{1,...,K\}$ be the subset of features that exist in $t^* \uparrow$ and $\mathcal{I} = \bigcup_{k \in F} \mathcal{I}_k$ the indexes over all features. The set of non-empty redundant paths in V^* with respect to all $k \in F$ is

$$R^* = \bigcup_{t \in T^*} \left\{ \bigcup_{i=\underline{i}^*}^{\overline{i}^*} p^{i}(t) : p^{i}(t) \in \bigcup_{k \in F} R^k_{t^*\uparrow} \text{ and } i \neq \overline{i} \text{ for all } \overline{i} \in \mathcal{I} \right\}.$$

The indexes $\underline{i}^*$ and $\overline{i}^*$, with $\underline{i}^* \leq \overline{i}^*$ represent the bottom and top vertices with respect to all $k \in F$ features for which an analogous condition in a connected predecessor subset exists. Let the set of empty vertices in V^* be

$$R' = \bigcup_{k \in \{1,...,K\}} \left\{ v : C_k(v) = \emptyset \text{ and } v \in V^* \right\}.$$

Define the set of all redundant vertices as

$$R = \left(R^* \cup R'\right).$$

Now, let τ be a tree with vertices $V = V^*\backslash R$ and terminal nodes T. Define a set that contains the successors of reduced paths in R^* for $t \in T$ as $S_{t\uparrow} = \left\{s\left(p^{\underline{i}^*}(t)\right) : s\left(p^{\underline{i}^*}(t)\right) \in t\uparrow\right\}$. The connection between the non-empty reduced vertices is established by defining $p(v'') = p\left(p^{\bar{i}^*}(t)\right)$ for all $v'' \in S_{t\uparrow}, t \in T$. Moreover, let $f(v) = f^*(v)$ for any $v \in V$. The new partitions are

$$\sigma(v', v'') = \begin{cases} \sigma\left(p\left(p^{\bar{i}^*}(t)\right), v''\right) & \text{for } v'' \in S_{t\uparrow} \\ \sigma^*(v', v'') & \text{for } v'' \in t\uparrow\backslash S_{t\uparrow} \end{cases}$$

for all $t \in T$. Given any reduced subset $\bigcup_{i=\underline{i}^*}^{\bar{i}^*} p^i(t) \subseteq R^*_{t^*\uparrow}$ and its corresponding successor $v'' \in S_{t\uparrow}$ for a path $t^*\uparrow\subset V^*$, let

$$D = v'' \cup \bigcup_{i=\underline{i}^*}^{\bar{i}^*} p^i(t) = \bigcup_{i=\underline{i}^*-1}^{\bar{i}^*} p^i(t)$$

and define

$$\bar{d}(v'') = \begin{cases} \sum_{t'_m \in D} d(t'_m) & \text{if there exists } t'_m \in D \\ 0 & \text{else} \end{cases} .$$

Furthermore, the disjoint union is $\mathcal{D} = \bigcup_{D\subseteq t^*\uparrow} D$. Observe that all $t'_m \in \mathcal{D} \cup \mathcal{D}^C = t^*\uparrow$, where $t^*\uparrow\subset V^*$. Therefore, for any $t \in T$ it holds that

$$\begin{aligned} d(t) &= h_0 + \sum_{D\subseteq\mathcal{D}} \bar{d}(v'') + \sum_{t'_m \in \mathcal{D}^C} d(t'_m) \\ &= h_0 + \sum_{m=1}^{M} d(t'_m) \\ &= d^*(t). \end{aligned}$$

Notice that the structure (τ, f, σ, d) is a decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$ that is prediction equivalent to $\delta^*_{(\mathcal{X},\mathcal{Y})}$, which implies that $\delta_{(\mathcal{X},\mathcal{Y})}$ is also prediction equivalent to $\eta_{(\mathcal{X},\mathcal{Y})}$. Furthermore, there exists no $v \in V$ such that $C_k(v) = \emptyset$ and $t \in T$ such that $C_k\left(p^{\underline{i}}(t)\right) \cap C_k\left(p^{\bar{i}}(t)\right) = C_k\left(p^{\bar{i}}(t)\right)$ for all $k \in \{1, ..., K\}$. $\square$

Corollary 2. *Given an ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$ and a prediction equivalent decision tree $\delta^*_{(\mathcal{X},\mathcal{Y})}$ of order $|V^*|$. Then, there exists a prediction equivalent decision tree $\delta_{(\mathcal{X},\mathcal{Y})}$ of order $|V|$ such that $|V^*| \geq |V|$.*

Proof. Let $\eta_{(\mathcal{X},\mathcal{Y})}$ be an ensemble learner and let $\delta^*_{(\mathcal{X},\mathcal{Y})} = (\tau^*, f^*, \sigma^*, d^*)$ be a prediction equivalent decision tree to $\eta_{(\mathcal{X},\mathcal{Y})}$. By Theorem 2, we know that for every ensemble learner $\eta_{(\mathcal{X},Y)}$, there exists a prediction equivalent decision tree $\delta_{(\mathcal{X},\mathcal{Y})} = (\tau, f, \sigma, d)$ with efficient paths. Let $|V^*|$ and $|V|$ denote the order of $\delta^*_{(\mathcal{X},\mathcal{Y})}$ and $\delta_{(\mathcal{X},\mathcal{Y})}$ respectively and let T^* and T be their set of terminal nodes. Then, it is straightforward to see that

$$\begin{aligned} |V^*| &= \left| \bigcup_{t \in T^*} p^i(t) \right| \\ &\geq \left| \bigcup_{t \in T} p^i(t) \right| \\ &= |V| \end{aligned}$$

□

4. Algorithms

Algorithm 1: Decision Tree Composition

```
1  input ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})} = (\delta_1, \ldots, \delta_M, w_1, \ldots, w_M, h_\eta)$;
2  $\delta^*_1 = \delta_1$
3  for $m = 2$ to $M$ do
4      $\tau^*_m = \tau^*_{m-1} \otimes \tau_m$
5      $V_{\mathcal{M}(\tau_m)} = \bigcup_{\gamma \in \mathcal{M}(\tau_m)} V_\gamma$
6      $\zeta_{\tau_m} : V_{\mathcal{M}(\tau_m)} \to V_{\tau_m}$
7      $f'_m(v) = f_m(\zeta_{\tau_m}(v)), v \in V_{\mathcal{M}(\tau_m)}$
8      $\sigma'_m(v', v'') = \sigma_m(\zeta_{\tau_m}(v'), \zeta_{\tau_m}(v'')), v', v'' \in V_{\mathcal{M}(\tau_m)}$
9      $d^*_m(t) = h_0 + \sum_{n=1}^{m} d(t'_n), t \in T^*_m$
10     $\delta^*_m = (\tau^*_m, f^*_m, \sigma^*_m, d^*_m)$
11 end
12 return composed decision tree $\delta^*_M = (\tau^*_M, f^*_M, \sigma^*_M, d^*_M)$
```

Algorithm 1 constructs prediction equivalent decision trees by iteratively combining the trees of an ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})}$. The first decision tree δ^*_1 remains unchanged. For $m > 1$, a series combination constructs a new tree $\tau^*_m = \tau^*_{m-1} \otimes \tau_m$ where τ^*_{m-1} is the composed tree of the previous iteration. The function $\zeta_{\tau_m} : V_{\mathcal{M}(\tau_m)} \to V_{\tau_m}$ maps the nodes of tree τ_m under the isomorphism to the series combination. Afterward, features are allocated by the function $f'_m(v) = f_m(\zeta_{\tau_m}(v)), v \in V_{\mathcal{M}(\tau_m)}$ and edges are established by $\sigma'_m(v', v'') = \sigma_m(\zeta_{\tau_m}(v'), \zeta_{\tau_m}(v'')), v', v'' \in V_{\mathcal{M}(\tau_m)}$ for iteration m. Given an initial prediction h_0, we have $d^*_m(t) = h_0 + \sum_{n=1}^{m} d(t'_n), t \in T^*_m$, where T^*_m is the set of terminal nodes in τ^*_m. The result is a decision tree $\delta^*_m = (\tau^*_m, f^*_m, \sigma^*_m, d^*_m)$, where f^*_m is defined as $f^*_m(v) = f_1(v)$ for $v \in V_{\tau_1}$ and $f^*_m(v) = f_j(\zeta_{\tau_j}(v))$ for all $v \in V_{\mathcal{M}(\tau_j)}$ and $j \in \{2, ..., m\}$. Then, edges are defined analogously. After M iterations the algorithm is complete.

The general reduction in algorithm 2 iterates over all paths $t^* \uparrow$ in a composed decision tree δ^*_M. First, it has to be examined whether the path contains redundant vertices by $t^* \uparrow \cap R \neq \emptyset$, else the algorithm continues with the next iteration. Recall that $R = R^* \cup R'$, where R^* is the set of non-empty redundant vertices and R' the set of empty vertices in δ^*_M. If $t^* \uparrow \cap R' \neq \emptyset$ the algorithm should proceed. Otherwise, we have that $t^* \uparrow \cap R^* \neq \emptyset$. Then, a new upset $t \uparrow = t^* \uparrow \setminus R$ has to be defined and the connection for the successor nodes $v'' \in S_{t\uparrow}$, where $S_{t\uparrow} = \left\{ s\left(p^{\underline{i}^*}(t)\right) : s\left(p^{\underline{i}^*}(t)\right) \in t \uparrow \right\}$ for all $\bigcup_{i=\underline{i}^*}^{\overline{i}^*} p^i(t) \subseteq R^*_{t^*\uparrow}$ has to be established. The predecessor of a successor v'' is defined by letting $p(v'') = p\left(p^{\overline{i}^*}(t)\right)$ and $\sigma_M\left(p\left(p^{\overline{i}^*}(t), v''\right)\right)$. Else, $\sigma_M(v', v'') = \sigma^*_M(v', v'')$ for $v'' \in t \uparrow \setminus S_{t\uparrow}$. Reduced terminal nodes $t'_n, n \in \{1, ..., M\}$ in the set D, where $D = v'' \cup \bigcup_{i=\underline{i}^*}^{\overline{i}^*} p^i(t)$, have to be allocated to the corresponding successor by letting $\overline{d}\,(v) = \sum_{t'_n \in D} d(t'_n)$. When $\bigcup_{t'_n \in t\uparrow} t'_n \cap D = \emptyset$, the algorithm continues with the next successor. If all connections between the nodes in $t \uparrow$ are established, the decision at t can be expressed as $d_M(t) = h_0 + \sum_{D \subseteq \mathcal{D}} \overline{d}\,(v) + \sum_{t'_n \in \mathcal{D}^C} d(t'_n)$. The algorithm is completed after iterating over all $t \in T^*_M$. The output is a new set of terminal nodes $T_M = T^*_M \setminus R$ that defines the set of vertices for the underlying tree τ_M by $V_{\tau_M} = \bigcup_{t \in T_M} t \uparrow$. Only unchanged paths $t^* \uparrow$ in δ^*_M or non-empty terminal nodes that contain non-empty redundant paths in their upset are considered for the construction of δ_M. For unchanged paths $t \uparrow = t^* \uparrow$, it holds that $f_M(v) = f^*_M, \sigma_M(v) = \sigma^*_M(v)$ and $d_M(v) = d^*_M(v)$ for all $v \in t \uparrow$. Else, $\sigma_M(v)$ and $d_M(v)$ are updated as described in the algorithm for $v'' \in S_{t\uparrow}$. Also, let $f_M(v) = f^*_M(v)$ for all $t \uparrow$.

Algorithm 2: General Reduction

```
1  input composed decision tree $\delta^*_M = (\tau^*_M, f^*_M, \sigma^*_M, d^*_M)$;
2  for $t$ in $T^*_M$ do
3      if $t^* \uparrow \cap R \neq \emptyset$ then
4          if $t^* \uparrow \cap R' \neq \emptyset$ then
5              continue
6          else
7              $t \uparrow = t^* \uparrow \setminus R$
8              for $v''$ in $S_{t\uparrow}$ do
9                  $p(v'') = p\left(p^{\bar{i}^*}(t)\right)$
10                 $\sigma_M\left(p\left(p^{\bar{i}^*}(t), v''\right)\right)$
11                 if $\bigcup_{t'_n \in t\uparrow} t'_n \cap D \neq \emptyset$ then
12                     $\overline{d}\,(v'') = \sum_{t'_n \in D} d(t'_n)$
13                 else
14                     continue
15                 end
16             end
17         end
18         $d_M(t) = h_0 + \sum_{D \subseteq \mathcal{D}} \overline{d}\,(v'') + \sum_{t'_n \in \mathcal{D}^C} d(t'_n)$
19     else
20         continue
21     end
22 end
23 output $T_M = T^*_M \setminus R, V_{\tau_M} = \bigcup_{t \in T_M} t \uparrow$
24 return reduced decision tree $\delta_M = (\tau_M, f_M, \sigma_M, d_M)$
```

Algorithm 3 describes a reduced decision tree composition. Let $\delta^r_1 = \delta_1$. For $m > 1$, the unreduced tree is given by $\tau^u_m = \tau^r_{m-1} \otimes \tau_m$. The iterative combination of trees is constructed as in algorithm 1. Let $f^r_m(v) = f^*_m(v)$ for $v \in V_{\tau^r_m}$ be the function that assigns features to the reduced tree structure. Then, the function f^u_m is defined as $f^r_{m-1}(v)$ for $v \in V_{\tau^r_{m-1}}$ and $f'_m(v) = f_m(\zeta_{\tau_m}(v)), v \in V_{\mathcal{M}(\tau_m)}$. σ^u_m and σ^r_m are constructed analogously. For the decision, let $d^u_m = d^r_{m-1} + d(t'_m)$ and define $S'_m = \{v'' : \overline{d}(v'') \neq 0 \text{ and } t \in T_m\}$ and $T'_m = \{t'_n : t'_n \in t \uparrow \setminus S'_m\}$. The set of all decision nodes in $t \uparrow$ is $S^*_m = S'_m \cup T'_m$. Let $\overline{d}\,(v'') = \overline{d}(v') + \sum_{t'_n \in D} d(t'_n), v' \in (S'_{m-1} \cap D)$ be the

Algorithm 3: Reduced Decision Tree Composition

```
input ensemble learner $\eta_{(\mathcal{X},\mathcal{Y})} = (\delta_1, \ldots, \delta_M, w_1, \ldots, w_M, h_\eta)$;
$\delta_1^r = \delta_1$
for $m = 2$ to $M$ do
    $\tau_m^u = \tau_{m-1}^r \otimes \tau_m$
    $\delta_m^u = (\tau_m^u, f_m^u, \sigma_m^u, d_m^u)$
    for $t$ in $T_m^*$ do
        if $t^* \uparrow \cap R_m \neq \emptyset$ then
            if $t^* \uparrow \cap R_m' \neq \emptyset$ then
                continue
            else
                $t \uparrow = t^* \uparrow \setminus R_m$
                for $v''$ in $S_{t\uparrow}^m$ do
                    $p(v'') = p\left(p^{\bar{i}^*}(t)\right)$
                    $\sigma_m^r\left(p\left(p^{\bar{i}^*}(t), v''\right)\right)$
                    if $(S_{m-1}^* \cup t_m') \cap D \neq \emptyset$ then
                        $\overline{d}(v'') = \overline{d}(v') + \sum_{t_n' \in D} d(t_n')$
                    else
                        continue
                    end
                end
                $d_m^r(t) = h_0 + \sum_{v'' \in S_m'} \overline{d}(v'') + \sum_{t_n' \in t\uparrow \setminus S_m'} d(t_n')$
            end
        else
            continue
        end
    end
    $T_m = T_m^* \setminus R_m, V_{\tau_m} = \bigcup_{t \in T_m} t \uparrow$
    $\delta_m^r = (\tau_m^r, f_m^r, \sigma_m^r, d_m^r)$
end
return composed decision tree $\delta_M^r = (\tau_M^r, f_M^r, \sigma_M^r, d_M^r)$
```

sum of decision in D, where e.g. $\overline{d}(v') = d(t_n') + \overline{d}(v), \overline{d}(v) = d(t_{n-1}') + d(t_{n-2}')$ or $\overline{d}(v') = 0$ if there is no $v' \in (S_{m-1}' \cap D)$. If $t \uparrow = t^* \uparrow$, then $f_m^r(v) = f_m^u(v), \sigma_m^r(v) = \sigma_m^u(v)$ and $d_m^r(t) = d_m^u(t)$ for all $v \in t^* \uparrow$.

5. Examples

Example 1 (Aggregate Prediction Rules of Ensemble Classifiers)**.** The following are the prediction rules of two prominent ensemble classifiers:

a) Random Forest (Ho, 1995; Breiman, 2001):

$$h_\eta(x) = \frac{h_{\delta_1}(x) + \cdots + h_{\delta_M}(x)}{M},$$

i.e., $w_1 = \cdots = w_M = 1/M$ and $h_0 = 0$.

b) Gradient Boost (Friedman, 2001):

$$h_\eta(x) = h_0 + \alpha \cdot h_{\delta_1}(x) + \cdots + \alpha \cdot h_{\delta_M}(x),$$

i.e., $w_1 = \cdots = w_M =$ learning rate $= \alpha$ and $h_0 \in \mathbb{R}$.

Example 2. Predict a binary label $\mathcal{Y} = \{0, 1\}$ based on three features X_1, X_2, X_3 with data:

	X_1	X_2	X_3	$\mathcal{Y}$
1	1	12	0	1
2	1	87	0	1
3	0	44	0	0
4	1	19	1	0
5	0	32	0	1
6	0	14	0	1

Let $\eta_{(\mathcal{X},\mathcal{Y})}$ be a binary Gradient Boost Classifier with learning rate $\alpha = 0.8$, $M = 2$ trees, max$(leaves) = 3$, max$(depth) = 2$ and classification cutoff $c = 0.5$. The structure is given by $H_M(\mathbf{x}) = h_0 + \alpha \sum_{m=1}^{M} h_m(\mathbf{x})$. The initial prediction is $h_0 = \log \frac{4}{2} = 0.7$ and for the corresponding probability we have $P(h_0) = \frac{e^{h_0}}{1+e^{h_0}} = \frac{2}{3}$. The pseudo residuals r_{im} of the first regression tree δ_1 are given by

	X_1	X_2	X_3	$\mathcal{Y}$	$P(h_0)$	r_{i1}
1	1	12	0	1	0.67	0.33
2	1	87	0	1	0.67	0.33
3	0	44	0	0	0.67	-0.67
4	1	19	1	0	0.67	-0.67
5	0	2	0	1	0.67	0.33
6	0	14	0	1	0.67	0.33

and for the first tree we have:

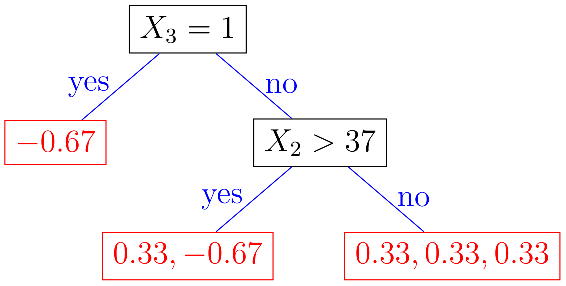


Figure 2: First regression tree δ_1

The output values of the terminal regions R_{jm} are calculated by

$$\gamma_{jm} = \frac{\sum_{\mathbf{x} \in R_{jm}} r_{im}}{\sum_{\mathbf{x} \in R_{jm}} (p_{i,m-1} \cdot (1 - p_{i,m-1}))}$$

where m is the index of the mth tree, $j \in \{1, ..., J\}$ denotes the jth terminal node and i are the data points in a specific terminal region R_{jm}. Therefore, we get the output values

	X_1	X_2	X_3	$\mathcal{Y}$	$P(h_0)$	r_{i1}	γ_{j1}
1	1	12	0	1	0.67	0.33	1.51
2	1	87	0	1	0.67	0.33	−0.77
3	0	44	0	0	0.67	−0.67	−0.77
4	1	19	1	0	0.67	−0.67	−3.03
5	0	32	0	1	0.67	0.33	1.51
6	0	14	0	1	0.67	0.33	1.51

The updated prediction is given by

$$H_m(\mathbf{x}) = H_{m-1} + \alpha \sum_{j=1}^{J_m} \gamma_{jm} I(\mathbf{x} \in R_{jm})$$

For the first observation we get $\log \frac{4}{2} + 0.8 \cdot 1.51 \approx 1.90$. Applying the sigmoid function to the initial prediction and the output values of the first tree yields a new probability $P(h_0 + h_1(\mathbf{x}))$. The new pseudo residuals for the second tree are

	X_1	X_2	X_3	$\mathcal{Y}$	$h_0 + h_1(\mathbf{x})$	$P(h_0 + h_1(\mathbf{x}))$	r_{i2}
1	1	12	0	1	1.90	0.87	0.13
2	1	87	0	1	0.08	0.52	0.48
3	0	44	0	0	0.08	0.52	-0.52
4	1	19	1	0	-1.73	0.15	-0.15
5	0	32	0	1	1.90	0.87	0.13
6	0	14	0	1	1.91	0.87	0.13

and the second regression tree is δ_2:

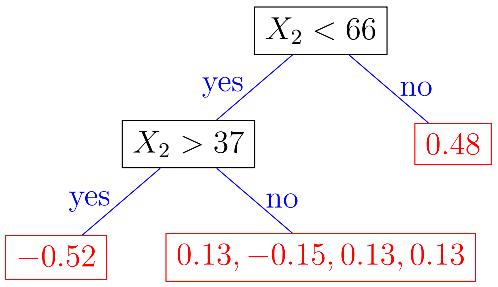


Figure 3: Second regression tree δ_2

The final predictions are given by $H_2(\mathbf{x}) = h_0 + \sum_{m=1}^{2} h_m(\mathbf{x})$. For the first observation we get $\log \frac{4}{2} + 0.8 \cdot (1.51 + 0.51) \approx 2.31$. The new probabilities and the classification decision D are

	X_1	X_2	X_3	$\mathcal{Y}$	$H_2(\mathbf{x})$	$P(H_2(\mathbf{x}))$	D
1	1	12	0	1	2.31	0.91	1
2	1	87	0	1	1.61	0.83	1
3	0	44	0	0	$-1,59$	0.17	0
4	1	19	1	0	-1.32	0.21	0
5	0	32	0	1	2.31	0.91	1
6	0	14	0	1	2.31	0.91	1

Composing the gradient boost outcomes by Theorem 1 yields a new decision tree $\delta^*_{(\mathcal{X},\mathcal{Y})}$. Applying the sigmoid function results in the same probabilities and therefore in a prediction equivalent classification decision.

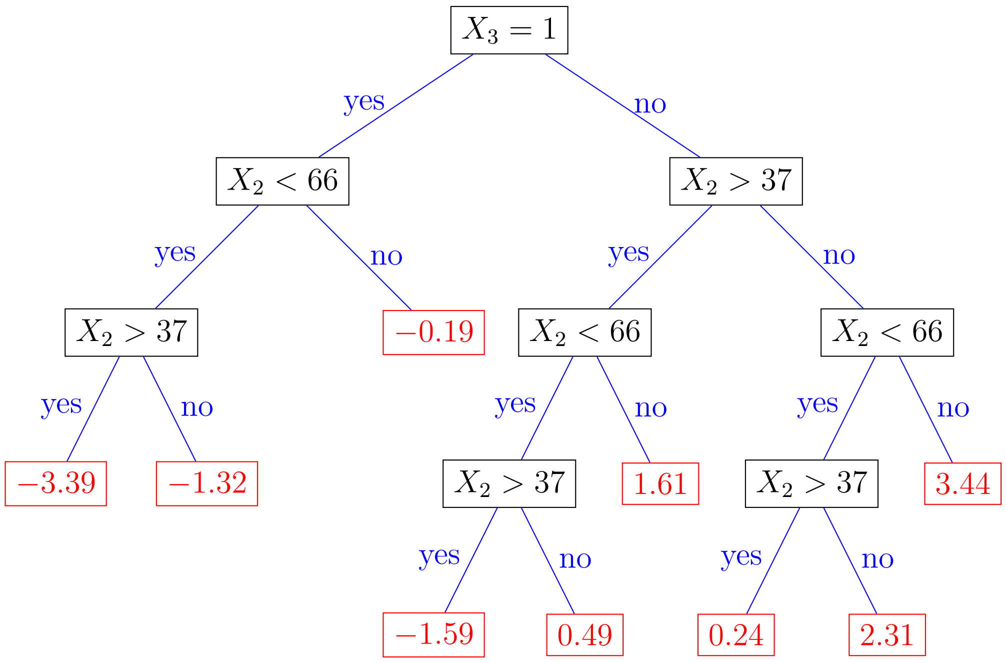


Figure 4: Prediction equivalent decision tree $\delta^*_{(\mathcal{X},\mathcal{Y})}$

Example 3. Applying Theorem 2 to $\delta^*_{(\mathcal{X},\mathcal{Y})}$ from example 2 yields a new decision tree with efficient paths. The first tree has order $|V^*| = 17$. For the new tree we get $|V| = 11$.

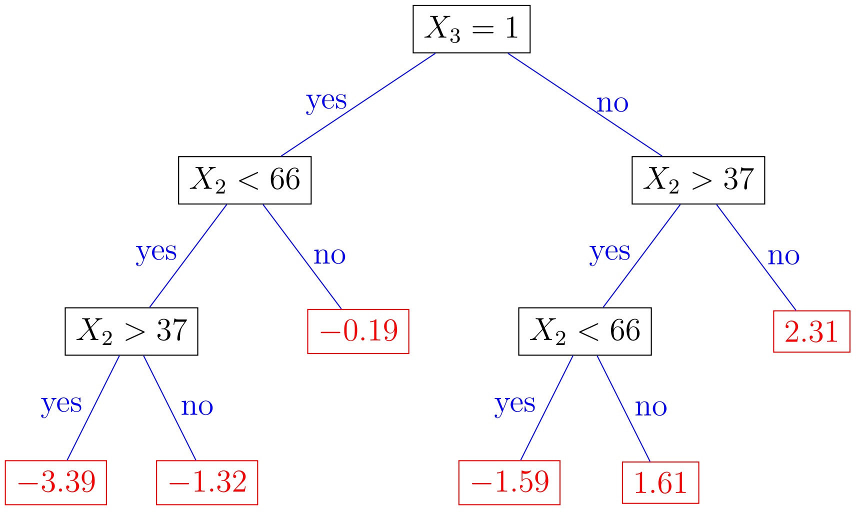


Figure 5: Reduced tree $\delta_{(\mathcal{X},Y)}$

6. Discussion

We have introduced the series combination, which combines the underlying trees of an ensemble and reallocates their features, edges and decision rules. This allowed us to prove the existence of prediction equivalent decision trees. In Algorithm 1, the procedure was described iteratively by the decision tree composition, also allowing the output of subtrees. In the appendix, the algorithm was demonstrated with a practical implementation in Python, where the underlying binary trees of a gradient boost or random forest can be composed. Furthermore, the resulting model structure can be evaluated as well as individual decisions of a given path. However, the composed subtrees may easily contain empty vertices or non-empty redundant vertices. Therefore, theorem 2 introduced a reduction procedure to decrease the order of the tree. In Algorithm 2, the reduction was described pathwise algorithmically. In addition, the two concepts are combined in Algorithm 3, where in each iteration two decision trees are composed and immediately reduced. Consequently, the reduced tree is used for the next composition step.

For the decision tree composition and the reduction of the composed tree, it should be evaluated up to which tree order the interpretability improves. Moreover, another theorem may be introduced to further ameliorate the interpretability, considering the reduction of the tree such that it performs approximately prediction equivalent. In addition, the prototype can be extended to Algorithm 2 and with particular consideration of the combined, iterative approach from Algorithm 3.

7. Conclusion

In this article, we have demonstrated a method that may improve the interpretability of tree-based ensemble learners such as gradient boosts or random forests. Our method yields a composed and reduced decision tree from all underlying trees of an ensemble. Therefore, it can be applied to improve the interpretability at a global level for the entire tree model structure and at a local level for individual decisions of a path.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Chapman and Hall.

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

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pages 278–282. IEEE Comput. Soc. Press.

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others.

The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Erklärung zur Bachelorarbeit

Hiermit versichere ich, dass ich die von mir vorgelegte Bachelorarbeit selbständig verfasst habe, ich keine anderen als die in der Bachelorarbeit angegebenen Quellen und Hilfsmittel benutzt habe, ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Inhalte als solche kenntlich gemacht habe.

Des Weiteren versichere ich, dass die von mir vorgelegte Bachelorarbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens war oder ist.

Ich versichere zudem, dass die von mir eingereichte elektronische Version in Form und Inhalt der gedruckten Version der Bachelorarbeit entspricht.

Ort, Datum Place, date	Unterschrift Signature