# Building more accurate decision trees with the additive tree

José Marcio Luna[a,1,2], Efstathios D. Gennatas[b,1], Lyle H. Ungar[c], Eric Eaton[c], Eric S. Diffenderfer[a], Shane T. Jensen[d], Charles B. Simone II[e], Jerome H. Friedman[f,2], Timothy D. Solberg[b], and Gilmer Valdes[b,2]

[a]Department of Radiation Oncology, University of Pennsylvania, Philadelphia, PA 19104; [b]Department of Radiation Oncology, University of California, San Francisco, CA 94115; [c]Department of Computing and Information Science, University of Pennsylvania, Philadelphia, PA 19104; [d]Department of Statistics, University of Pennsylvania, Philadelphia, PA 19104; [e]Department of Radiation Oncology, New York Proton Center, New York, NY 10035; and [f]Department of Statistics, Stanford University, Stanford, CA 94305

Contributed by Jerome H. Friedman, August 8, 2019 (sent for review October 10, 2018; reviewed by Adele Cutler and Giles Hooker)

**The expansion of machine learning to high-stakes application domains such as medicine, finance, and criminal justice, where making informed decisions requires clear understanding of the model, has increased the interest in interpretable machine learning. The widely used Classification and Regression Trees (CART) have played a major role in health sciences, due to their simple and intuitive explanation of predictions. Ensemble methods like gradient boosting can improve the accuracy of decision trees, but at the expense of the interpretability of the generated model. Additive models, such as those produced by gradient boosting, and full interaction models, such as CART, have been investigated largely in isolation. We show that these models exist along a spectrum, revealing previously unseen connections between these approaches. This paper introduces a rigorous formalization for the additive tree, an empirically validated learning technique for creating a single decision tree, and shows that this method can produce models equivalent to CART or gradient boosted stumps at the extremes by varying a single parameter. Although the additive tree is designed primarily to provide both the model interpretability and predictive performance needed for high-stakes applications like medicine, it also can produce decision trees represented by hybrid models between CART and boosted stumps that can outperform either of these approaches.**

additive tree | decision tree | interpretable machine learning | CART | gradient boosting

The increasing application of machine learning to high-stakes domains such as criminal justice (1, 2) and medicine (3–5) has led to a surge of interest in understanding the generated models. Mispredictions in these domains can incur serious risks in scenarios such as technical debt (6, 7), nondiscrimination (8), medical outcomes (9, 10), and, recently, the right to explanation (11), thus motivating the need for users to be able to examine why the model made a particular prediction. Despite recent efforts to formalize the concept of interpretability in machine learning, there is considerable disagreement on what such a concept means and how to measure it (12, 13). Lately, 2 broad categories of approaches for interpretability have been proposed (14), namely, post hoc simple explanations for potentially complex models (e.g., visual and textual explanations) (15, 16) and intuitively simple models (e.g., additive models, decision trees). This paper focuses on intuitively simple models, specifically decision trees, which are used widely in fields such as healthcare, yet have lower predictive power than more sophisticated models.

Classification and Regression Tree (CART) analysis (17) is a well-established statistical learning technique that has been adopted by numerous fields for its model interpretability, scalability to large datasets, and connection to rule-based decision-making (18). Specifically, in fields like medicine (19–21), the aforementioned traits are considered a requirement for clinical decision support systems. CART builds a model by recursively partitioning the instance space, and labeling each partition with either a predicted category (in the case of classification) or real value (in the case of regression). Despite widespread use, CART models are often less accurate than other statistical learning models, such as kernel methods and ensemble techniques (22). Among the latter, boosting methods were developed as a means to train an ensemble that iteratively combines multiple weak learners (often CART models) into a high-performance predictive model, albeit with an increase of the number of nodes, therefore, harming model interpretability. In particular, gradient boosting methods (23) iteratively optimize an ensemble's prediction to increasingly match the labeled training data. In addition, some ad hoc approaches (24, 25) have been successful at improving the accuracy of decision trees, but at the expense of altering their topology, therefore affecting their capacity of explanation.

Decision tree learning and gradient boosting have been connected primarily through CART models used as the weak learners in boosting. However, a rigorous analysis in ref. 26 proves that decision tree algorithms, specifically CART and C4.5 (27), are,

## Significance

As machine learning applications expand to high-stakes areas such as criminal justice, finance, and medicine, legitimate concerns emerge about high-impact effects of individual mispredictions on people's lives. As a result, there has been increasing interest in understanding general machine learning models to overcome possible serious risks. Current decision trees, such as Classification and Regression Trees (CART), have played a predominant role in fields such as medicine, due to their simplicity and intuitive interpretation. However, such trees suffer from intrinsic limitations in predictive power. We developed the additive tree, a theoretical approach to generate a more accurate and interpretable decision tree, which reveals connections between CART and gradient boosting. The additive tree exhibits superior predictive performance to CART, as validated on 83 classification tasks.

Author contributions: J.M.L., E.D.G., L.H.U., E.E., E.S.D., C.B.S., J.H.F., T.D.S., and G.V. designed research; J.M.L., E.D.G., and G.V. performed research; J.M.L., E.D.G., L.H.U., E.E., S.T.J., J.H.F., T.D.S., and G.V. contributed new reagents/analytic tools; J.M.L., E.D.G., L.H.U., E.E., E.S.D., S.T.J., C.B.S., T.D.S., and G.V. analyzed data; and J.M.L., E.D.G., L.H.U., E.E., E.S.D., S.T.J., C.B.S., J.H.F., T.D.S., and G.V. wrote the paper.

Reviewers: A.C., Utah State University; and G.H., Cornell University.

Conflict of interest statement: J.M.L., E.E., L.H.U., C.B.S., T.D.S., and G.V. have a patent titled "Systems and methods for generating improved decision trees," pending status.



[1]J.M.L. and E.D.G. contributed equally to this work.

[2]To whom correspondence may be addressed. Email: gilmer.valdes@ucsf.edu, jose.luna@pennmedicine.upenn.edu, or jhf@stanford.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1816748116/-/DCSupplemental.

First published September 16, 2019.





STATISTICS

in fact, boosting algorithms. Based on this approach, AdaTree, a tree-growing method based on AdaBoost (28), was proposed in ref. 29. A sequence of weak classifiers on each branch of the decision tree was trained recursively using AdaBoost; therefore, rendering a decision tree where each branch conforms to a strong classifier. The weak classifiers at each internal node were implemented either as linear classifiers composed with a sigmoid or as Haar-type filters with threshold functions at their output. Another approach is the Probabilistic Boosting-Tree (PBT) algorithm introduced in ref. 30, which also uses AdaBoost to build decision trees. PBT trains ensembles of strong classifiers on each branch for image classification. The strong classifiers are based on up to third-order moment histogram features extracted from Gabor and intensity filter responses. PBT also carries out a divide-and-conquer strategy in order to perform data augmentation to estimate the posterior probability of each class. Recently, MediBoost, a tree-growing method based on the more general gradient boosting approach, was proposed in ref. 31. In contrast to AdaTree and PBT, MediBoost emphasizes interpretability, since it was conceived to support medical applications while exploiting the accuracy of boosting. It takes advantage of the shrinkage factor inherent to gradient boosting, and introduces an acceleration parameter that controls the observation weights during training to leverage predictive performance. Although the presented experimental results showed that MediBoost exhibits better predictive performance than CART, no theoretical analysis was provided.

In this paper, we propose a rigorous mathematical approach for building single decision trees that are more accurate than traditional CART trees. In this context, we use the total number of decision nodes (NDN) as a quantification of interpretability, due to their direct association with the number of decision rules of the model and the depth of the tree. This paper addresses a number of fundamental shortcomings: 1) We introduce the additive tree (AddTree) as a mechanism for creating decision trees. Each path from the root node to a leaf represents the outcome of a gradient boosted ensemble for a partition of the instance space. 2) We theoretically prove that AddTree generates a continuum of single-tree models between CART and gradient boosted stumps (GBS), controlled via a single tunable parameter of the algorithm. In effect, AddTree bridges between CART and gradient boosting, identifying previously unknown connections between additive and full interaction models. 3) We provide empirical results showing that this hybrid combination of CART and GBS outperforms CART in terms of accuracy and interpretability. Our experiments also provide further insight into the continuum of models revealed by AddTree.

## Background on CART and Boosting

Assume we are given a training dataset $(\boldsymbol{X},\boldsymbol{y})=\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where each $d$-dimensional data instance $\boldsymbol{x}_i \in \boldsymbol{X} \subseteq \mathcal{X}$ has a corresponding label $y_i \in \mathcal{Y}$, drawn independently and from an identical and unknown distribution $\mathcal{D}$. In a binary classification setting, $\mathcal{Y}=\{\pm 1\}$; in regression, $\mathcal{Y}=\mathbb{R}$. The goal is to learn a function $F:\mathcal{X}\mapsto\mathcal{Y}$ that will perform well in predicting the label on new examples drawn from $\mathcal{D}$. CART analysis recursively partitions $\mathcal{X}$, with $F$ assigning a single label in $\mathcal{Y}$ to each terminal node; in this manner, there is full interaction. Different branches of the tree are trained with disjoint subsets of the data, as shown in Fig. 1.

In contrast, boosting iteratively trains an ensemble of $T$ weak learners $\{h_t:\mathcal{X}\mapsto\mathcal{Y}\}_{t=1}^{T}$, such that the resulting model $F(\boldsymbol{x})$ is a weighted sum of the weak learners' predictions: $F(\boldsymbol{x})=\sum_{t=1}^{T}\rho_t h_t(\boldsymbol{x})$ with $\boldsymbol{\rho}\in\mathbb{R}^T$.* Each boosted weak learner

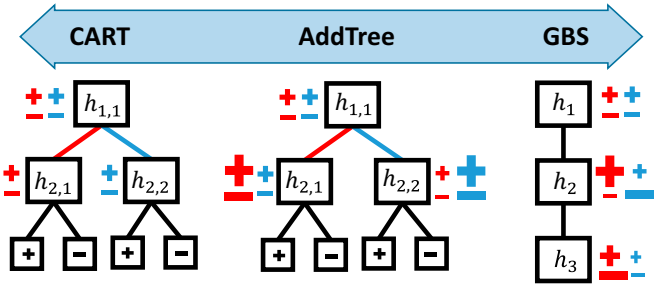


**Fig. 1.** A depiction of the continuum relating CART, GBS, and our AddTree. Each algorithm has been given the same 4 training instances (blue and red symbols); the symbol's size depicts its weight when used to train the adjacent node.

is trained with a different weighting of the entire dataset, unlike CART, repeatedly emphasizing mispredicted instances at every round (Fig. 1). GBS or simple regression creates a pure additive model in which each new ensemble member reduces the residual of previous members (32). Interaction terms can be included in the ensemble by using more complex learners, such as deeper trees.

Classifier ensembles with decision stumps as the weak learners, $h_t(\boldsymbol{x})$, can be trivially rewritten (31) as a complete binary tree of depth $T$, where the decision made at each internal node at depth $t-1$ is given by $h_t(\boldsymbol{x})$, and the prediction at each leaf is given by $F(\boldsymbol{x})$. Intuitively, each path through the tree represents the same ensemble, but one that tracks the unique combination of predictions made by each member.

## The Additive Tree

This interpretation of boosting lends itself to the creation of a tree-structured ensemble learner that bridges CART and GBS: the AddTree. At each node, a weak learner is found using gradient boosting on the entire dataset. Rather than using only the data in the current node to decide the next split, the algorithm allows the remaining data to also influence this split, albeit with a potentially differing weight. Critically, this process results in an interpretable model that is simply a decision tree of the weak learner's predictions, but with higher predictive power due to its growth via boosting and the resulting effect of regularizing the tree splits. This process is carried out recursively until the depth limit is reached. By varying the weighting scheme, this approach interpolates between CART trees at one extreme and boosted regression stumps at the other. Notably, the resulting tuning of the weighting scheme allows AddTree to improve in predictive performance over CART and GBS, while allowing direct interpretation.

AddTree maintains a perfect binary tree of depth $n$, with $2^n-1$ internal nodes, each of which corresponds to a weak learner. The $k$th weak learner, denoted $h_{k,l}$, along the path from the root node to a leaf prediction node $l$ induces 2 disjoint partitions of the feature space $\mathcal{X}$, namely $P_{k,l}$ and its complement $P_{k,l}^c$. Let $\{R_{1,l},\ldots,R_{n,l}\}$ be the corresponding set of partitions along that path to $l$, where the $k$th term in the tuple, namely $R_{k,l}$, can be either $P_{k,l}$ or $P_{k,l}^c$. We can then define the partition of $\mathcal{X}$ associated with the leaf node $l$ as the intersection of all of the set of partitions along the path from the root node to the leaf node $l$; that is, $\boldsymbol{\mathcal{R}}_{n,l}=\bigcap_{k=1}^{n}R_{k,l}$. AddTree predicts a label for each $\boldsymbol{x}\in\boldsymbol{\mathcal{R}}_{n,l}$ via the ensemble consisting of all weak learners along the path to $l$ so that the resulting model is given by $F(\boldsymbol{x}\in\boldsymbol{\mathcal{R}}_{n,l})=\sum_{k=1}^{n}\rho_{k,l}h_{k,l}(\boldsymbol{x})$. To focus each branch of the tree on corresponding instances, thereby constructing diverse ensembles, AddTree maintains a set of weights denoted $\boldsymbol{w}_{n,l}\in\mathbb{R}^N$ over all training data. Such weights are associated with training a weak learner $h_{n,l}$ at the leaf node $l$ at depth $n$.

*In classification, $F$ gives the sign of the prediction. CART models are often used as the $h_t$s.

We train the tree as follows. At each boosting step, we have a current estimate of the function $F_{n-1,l}(\boldsymbol{x})$ at the leaf $l$ of a perfect binary tree of depth $n-1$. We seek to improve this estimate by replacing each of the $2^{n-1}$ leaf prediction nodes with additional weak learners $\{h_{n,l}\}_{l=1}^{2^{n-1}}$ with corresponding scaling factor $\{\rho_{n,l}\}_{l=1}^{2^{n-1}}$, growing the tree by one level. This yields a revised estimate of the function at each terminal node $l$ equivalent to $2^{n-1}$ separate functions,

$$F_{n,l}(\boldsymbol{x}) = F_{n-1,l}(\boldsymbol{x}) + \rho_{n,l} h_{n,l}(\boldsymbol{x}) \quad \forall l \in \{1,\dots,2^{n-1}\},$$

one for each leaf's corresponding ensemble. The goal is to minimize the loss over the data

$$L_n(\boldsymbol{X},\boldsymbol{y}) = \sum_{l=1}^{2^{n-1}} \sum_{i=1}^{N} w_{n,l,i} \times \ell(y_i, F_{n-1,l}(\boldsymbol{x}_i) + \rho_{n,l} h_{n,l}(\boldsymbol{x}_i)), \quad [1]$$

by choosing the appropriate values of $\rho_{n,l}$ and $h_{n,l}$ at each leaf. Our proposed approach aims at breaking the connection between the global loss in Eq. **1** and the loss used for each weak learner by independently minimizing the inner summation for each $l \in \{1,\dots,2^{n-1}\}$; that is,

$$L_{n,l}(\boldsymbol{X},\boldsymbol{y}) = \sum_{i=1}^{N} w_{n,l,i}\, \ell(y_i, F_{n-1,l}(\boldsymbol{x}_i) + \rho_{n,l} h_{n,l}(\boldsymbol{x}_i)). \quad [2]$$

Eq. **2** can be solved efficiently via gradient boosting of each $L_{n,l}(\cdot)$ in a level-wise manner through the tree.

Next, we focus on deriving AddTree where the weak learners are binary regression trees with least squares as the loss function $\ell(\cdot)$. Following ref. 23, we first estimate the negative unconstrained gradients at each data instance $\left\{\tilde{y}_i = -\frac{\partial \ell(y_i, F_{n-1,l}(\boldsymbol{x}_i))}{\partial F_{n-1,l}(\boldsymbol{x}_i)}\right\}_{i=1}^{N}$, which are equivalent to the residuals [i.e., $\tilde{y}_i = y_i - F_{n-1,l}(\boldsymbol{x}_i)$]. Then, we can determine the optimal parameters for $L_{n,l}(\cdot)$ by solving

$$\arg\min_{\rho_{n,l},\, h_{n,l}} \sum_{i=1}^{N} w_{n,l,i}\ (\tilde{y}_i - \rho_{n,l}\, h_{n,l}(\boldsymbol{x}_i))^2. \quad [3]$$

Gradient boosting solves Eq. **3** by first fitting decision stumps $h_{n,l}$ to the residuals $(\boldsymbol{X}, \tilde{\boldsymbol{y}})$,

$$h_{n,l}(\boldsymbol{x}_i) = \arg\min_{h} \sum_{i=1}^{N} w_{n,l,i} (\tilde{y}_i - h)^2,$$

then solving for the optimal scaling factor $\rho_{n,l}$, whose main function is to scale the weak learners obtained from pseudoresiduals to fit the original data,

$$\rho_{n,l} = \arg\min_{\rho} \sum_{i=1}^{N} w_{n,l,i}\, (y_i - F_{n-1,l}(\boldsymbol{x}_i) - \rho h_{n,l}(\boldsymbol{x}_i))^2,$$

or, equivalently, solving for the simple regressor $\gamma_{n,l}(\boldsymbol{x}_i) = \rho_{n,l} h_{n,l}(\boldsymbol{x}_i)$ as follows:

$$\gamma_{n,l}(\boldsymbol{x}_i) = \arg\min_{\gamma} \sum_{i=1}^{N} w_{n,l,i}\, (y_i - F_{n-1,l}(\boldsymbol{x}_i) - \gamma)^2.$$

If all instance weights $\boldsymbol{w}_{n,l}$ remain constant, this approach would build a perfect binary tree of depth $T$, where each path from the root to a leaf represents the same ensemble, and so would be exactly equivalent to gradient boosting of $(\boldsymbol{X},\boldsymbol{y})$. Instead, AddTree constructs diverse ensembles by focusing each branch of the tree on corresponding instances that belong to that partition. To accomplish this, the weights are updated separately for each of $h_{n,l}$'s 2 children: Instances in the corresponding partition have their weights multiplied by a factor of $1+\lambda$, and instances outside the partition have their weights multiplied by a factor $\lambda \in [0,\infty)$ denoted "interpretability factor." The update rule for the weight $w_{n,l}(\boldsymbol{x}_i)$ of $\boldsymbol{x}_i$ for $R_{n,l} \in \{P_{n,l}, P_{n,l}^c\}$ is given by

$$w_{n,l}(\boldsymbol{x}_i) = w_{n-1,l}(\boldsymbol{x}_i)\, (\lambda + \mathbb{1}[\boldsymbol{x}_i \in R_{n,l}]), \quad [4]$$

where $\mathbb{1}[p]$ is a binary indicator function that is 1 if predicate $p$ is true, and 0 otherwise, and the initial weights $\boldsymbol{w}_0$ are typically uniformly distributed. Therefore, we can also think of AddTree as growing a collection of interrelated ensembles, where each is tuned to yield optimal predictions for one partition of the instance space. The complete AddTree approach is detailed as Algorithm 1. Notice that a learning rate or shrinkage parameter $\nu$ is in place in order to apply regularization by shrinkage in step 8 of the algorithm. The learning rate balances the contributions of each node to the running function estimate of its ensemble. Larger learning rates allow each weak learner to contribute more to the function estimate, resulting in shorter (and consequently more interpretable trees); smaller learning rates slow the running function estimate to yield larger but potentially more accurate trees as shown in *SI Appendix*, Figs. S5 and S6.

## Results

We provide the following results: 1) a theoretical analysis establishing the connections between CART, GBS, and AddTree; 2) empirical results demonstrating that AddTree yields a decision tree that outperforms CART and performs equivalently to GBS;

**Algorithm 1:** AddTree($\boldsymbol{X}$, $\boldsymbol{y}$, $\boldsymbol{w}_{n,l}$, $\lambda$, $n$, $T$, $\mathcal{R}_{n,l}$, $F_{n-1,\,l}$, $\nu$)

**Inputs:** Training data $(\boldsymbol{X},\boldsymbol{y}) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$,
instance weights $\boldsymbol{w}_{n,l} \in \mathbb{R}^N$ (default: $w_{n,l,i} = \frac{1}{N}$),
$\lambda \in [0, +\infty)$, node depth $n$ (default: 1),
max depth $T$, node domain $\mathcal{R}_{n,l}$ (default: $\mathcal{X}$),
prediction function $F_{n-1,l}(\boldsymbol{x})$ (default: $F_0(\boldsymbol{x}) = \bar{y}_0$),
learning rate $\nu$.

**Outputs:** The root node of a hierarchical ensemble

1: If $n \geq T$, return a prediction node $l_n$ that predicts the weighted average of $\boldsymbol{y}$ with weights $\boldsymbol{w}_{n,l}$
2: Create a new subtree root $l_n$ to hold a weak learner
3: Compute negative gradients:
$\left\{\tilde{y}_i = -\frac{\partial \ell(y_i, F_{n-1,l}(\boldsymbol{x}_i))}{\partial F_{n-1,l}(\boldsymbol{x}_i)}\right\}_{i=1}^{N}$
4: Fit weak classifier $h_{n,l}(\boldsymbol{x}) : \mathcal{X} \mapsto \mathcal{Y}$ by solving:
$h_{n,l} \leftarrow \arg\min_h \sum_{i=1}^{N} w_{n,l}(\boldsymbol{x}_i)(\tilde{y}_i - h)^2$
5: Let $\{P_{n,l}, P_{n,l}^c\}$ be the partitions induced by $h_{n,l}$.
6: $\rho_{n,l} \leftarrow \arg\min_{\rho} \sum_{i=1}^{N} w_{n,l}(\boldsymbol{x}_i)\, (y_i - F_{n-1,l}(\boldsymbol{x}_i) - \rho h_{n,l})^2$
7: Define the simple regressor:
$\gamma_{n,l} \leftarrow \rho_{n,l} h_{n,l}$
8: Update the current function estimation:
$F_{n,l}(\boldsymbol{x}) = F_{n-1,l}(\boldsymbol{x}) + \nu \cdot \gamma_{n,l}$
9: Update the left and right subtree instance weights:
$w_{n,l}^{(\text{left})}(\boldsymbol{x}_i) \leftarrow w_{n,l}(\boldsymbol{x}_i)\, (\lambda + \mathbb{1}[\boldsymbol{x}_i \in P_{n,l}])$
$w_{n,l}^{(\text{right})}(\boldsymbol{x}_i) \leftarrow w_{n,l}(\boldsymbol{x}_i)\, (\lambda + \mathbb{1}[\boldsymbol{x}_i \in P_{n,l}^c])$
10: If $\mathcal{R}_{n,l} \bigcap P_{n,l} \neq \emptyset$, compute left subtree recursively:
$l_n.\text{left} \leftarrow \text{AddTree}\left(\boldsymbol{X}, \boldsymbol{y}, \lambda, \boldsymbol{w}_{n,l}^{(\text{left})}, \mathcal{R}_{n,l} \bigcap P_{n,l}, n+1, T, F_{n,l}, \nu\right)$
11: If $\mathcal{R}_{n,l} \bigcap P_{n,l}^c \neq \emptyset$, compute right subtree recursively:
$l_n.\text{right} \leftarrow \text{AddTree}\left(\boldsymbol{X}, \boldsymbol{y}, \lambda, \boldsymbol{w}_{n,l}^{(\text{right})}, \mathcal{R}_{n,l} \bigcap P_{n,l}^c, n+1, T, F_{n,l}, \nu\right)$

and 3) an empirical analysis of the behavior of AddTree under various parameter settings.

**Theoretical Analysis.** This section proves that AddTree is equivalent to CART when $\lambda = 0$ and converges to GBS as $\lambda \to \infty$, thus establishing a continuum between CART and GBS. We include proof sketches for the 3 lemmas used to prove our main result in Theorem 1; full proofs are included in *SI Appendix*.

**Lemma 1** *The weight of $\boldsymbol{x}_i$ at leaf $l \in \{1, \dots, 2^n\}$ at depth $n$ of the tree ($\forall n = 1, 2, \dots$) is given by*

$$w_{n,l}(\boldsymbol{x}_i) = w_0(\boldsymbol{x}_i)\,(\lambda+1)^{\sum_{k=1}^{n} \mathbb{1}[\boldsymbol{x}_i \in R_{k,l}]}\lambda^{\sum_{k=1}^{n} \mathbb{1}[\boldsymbol{x}_i \in R^c_{k,l}]}, \quad [5]$$

*where $\{R_{1,l}, \dots, R_{n,l}\}$ is the sequence of partitions along the path from the root to $l$.*

***Proof Sketch:*** *Shown by induction based on Eq. **4**.* □

**Lemma 2** *The optimal simple regressor $\gamma^*_{n,l}(\boldsymbol{x})$ that minimizes the squared error loss function Eq. **2** at leaf $l \in \{1, \dots, 2^n\}$ at depth $n$ of the tree is given by*

$$\gamma^*_{n,l}(\boldsymbol{x}) = \begin{cases} \frac{\sum_{i:\boldsymbol{x}_i \in R_{n,l}} w_{n,l}(\boldsymbol{x}_i)\left(y_i - F_{n-1,l}(\boldsymbol{x}_i)\right)}{\sum_{i:\boldsymbol{x}_i \in R_{n,l}} w_{n,l}(\boldsymbol{x}_i)} & \text{if } \boldsymbol{x}_i \in R_{n,l} \\ \frac{\sum_{i:\boldsymbol{x}_i \in R^c_{n,l}} w_{n,l}(\boldsymbol{x}_i)(y_i - F_{n-1,l}(\boldsymbol{x}_i))}{\sum_{i:\boldsymbol{x}_i \in R^c_{n,l}} w_{n,l}(\boldsymbol{x}_i)} & \text{otherwise} \end{cases}. \quad [6]$$

***Proof Sketch:*** *For a given region $R_{n,l} \subset \mathcal{X}$ at the depth $n$ of the tree, the simple regressor has the form*

$$\gamma_{n,l}(\boldsymbol{x}) = \begin{cases} \gamma_{n_1,l} & \text{if } \boldsymbol{x} \in R_{n,l} \\ \gamma_{n_2,l} & \text{otherwise} \end{cases}, \quad [7]$$

*with constants $\gamma_{n_1,l}, \gamma_{n_2,l} \in \mathbb{R}$. We take the derivative of the loss function (Eq. **2**) in each of the 2 regions $R_{n,l}$ and $R^c_{n,l}$, and solve for where the derivative is equal to zero, obtaining Eq. **6**.* □

**Lemma 3** *The AddTree update rule is given by $F_{n,l}(\boldsymbol{x}) = F_{n-1,l}(\boldsymbol{x}) + \gamma_{n,l}(\boldsymbol{x})$. If $\gamma_{n,l}(\boldsymbol{x})$ is defined as*

$$\gamma_{n,l}(\boldsymbol{x}) = \frac{\sum_{i:\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}_{n,l}} w_{n,l}(\boldsymbol{x}_i)(y_i - F_{n-1,l}(\boldsymbol{x}_i))}{\sum_{i:\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}_{n,l}} w_{n,l}(\boldsymbol{x}_i)},$$

*with constant $F_0(\boldsymbol{x}) = \bar{y}_0$, then $F_{n,l}(\boldsymbol{x}) = \bar{y}_{n,l}$ is constant, with*

$$\bar{y}_{n,l} = \frac{\sum_{i:\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}_{n,l}} w_{n,l}(\boldsymbol{x}_i) y_i}{\sum_{i:\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}_{n,l}} w_{n,l}(\boldsymbol{x}_i)}, \quad \forall n = 1, 2, \dots. \quad [8]$$

***Proof Sketch:*** *The proof is by induction on $n$, building upon Eq. **7**. Each $\gamma_{n,l}(\boldsymbol{x}_i)$ is constant and so $\bar{y}_{n,l}$ is constant; therefore, the lemma holds under the given update rule.* □

Building upon these 3 lemmas, our main theoretical result is presented in the following theorem, and explained in the subsequent 2 remarks:

**Theorem 1** *Given the AddTree optimal simple regressor (Eq. **6**) that minimizes the loss (Eq. **2**), the following limits hold for the parameter $\lambda$ of the weight update rule (Eq. **4**):*

$$\lim_{\lambda \to 0} \gamma^*_{n,l}(\boldsymbol{x}) = \frac{\sum_{i:\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}_{n,l}} w_0(\boldsymbol{x}_i) y_i}{\sum_{i:\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}_{n,l}} w_0(\boldsymbol{x}_i)} - \bar{y}_{n-1,l}, \quad [9]$$

$$\lim_{\lambda \to \infty} \gamma^*_{n,l}(\boldsymbol{x}) = \begin{cases} \frac{\sum_{\boldsymbol{x}_i \in R_{n,l}} w_0(\boldsymbol{x}_i)(y_i - F_{n-1,l}(\boldsymbol{x}_i))}{\sum_{\boldsymbol{x}_i \in R_{n,l}} w_0(\boldsymbol{x}_i)} & \text{if } \boldsymbol{x}_i \in R_{n,l} \\ \frac{\sum_{\boldsymbol{x}_i \in R^c_{n,l}} w_0(\boldsymbol{x}_i)(y_i - F_{n-1,l}(\boldsymbol{x}_i))}{\sum_{\boldsymbol{x}_i \in R^c_{n,l}} w_0(\boldsymbol{x}_i)} & \text{otherwise}, \end{cases} \quad [10]$$

*where $w_0(\boldsymbol{x}_i)$ is the initial weight for the $i$th training sample.*

**Proof** *Eq. **9** follows from applying Eq. **5** in Lemma 1 to Eq. **6** in Lemma 2 and taking the limit when $\lambda \to 0$, regarding the result $F_n(\boldsymbol{x}) = \bar{y}_{n,l}$, with constant $\bar{y}_{n,l}$ defined by Eq. **8**. Similarly, Eq. **10** follows from applying Eq. **5** in Lemma 1 to Eq. **6** in Lemma 2 and taking the limit when $\lambda \to \infty$.* □

**Remark 1** *The simple regressor given by Eq. **9** calculates a weighted average of the difference between the random output variables $y_i$ and the previous estimate $\bar{y}_{n-1,l}$ of the function $F^*(\boldsymbol{x})$ in the disjoint regions defined by $\boldsymbol{\mathcal{R}}_{n,l}$. This formally defines the behavior of the CART algorithm.*

**Remark 2** *The simple regressor given by Eq. **10** calculates a weighted average of the difference between the random output variables $y_i$ and the previous estimate of $F^*(\boldsymbol{x})$ given by the piece-wise constant function $F_{n-1,l}(\boldsymbol{x}_i)$. $F_{n-1,l}(\boldsymbol{x}_i)$ is defined in the overlapping region determined by the latest stump, namely $R_{n,l}$. This defines the behavior of the GBS algorithm.*

Based on Remarks 1 and 2, we can conclude that AddTree is equivalent to CART when $\lambda \to 0$ and GBS as $\lambda \to \infty$. In addition to identifying connections between these 2 algorithms, AddTree provides the flexibility to train a model that lies between CART and GBS, with potentially improved performance over either, as we show empirically in the next section. Finally, notice that, although the introduction of the $\lambda$ parameter might be reminiscent of decision trees trained with fuzzy logic (33), in the fuzzy logic approach, the continuity is between a simple constant model and CART, while AddTree produces a continuous mapping between CART and GBS.

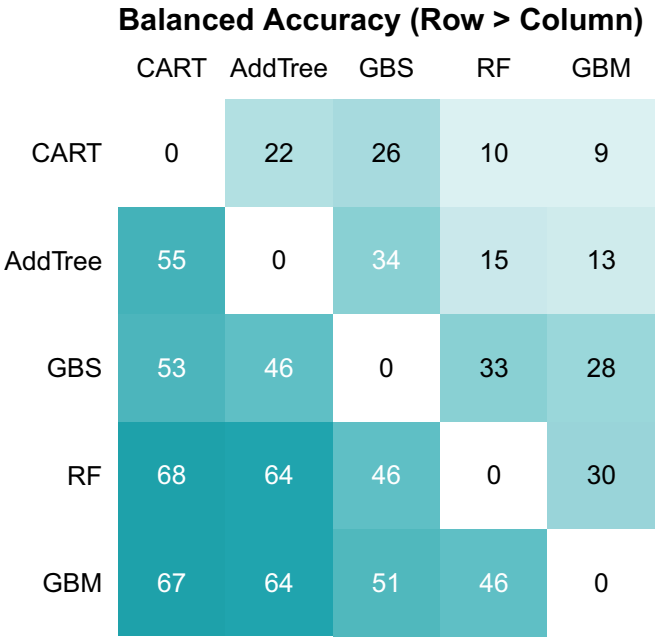


**Fig. 2.** Frequency that an algorithm (rows) has higher average BACC compared to each one of the remaining algorithms (columns) under study across 83 binary classification tasks. Each of the learning algorithms has been tuned to maximize BACC. Ties have been ruled out for the sake of clarity.

**Table 1. Performance indexes for all of the learning algorithms under comparison across 83 PMLB classification tasks**

| Performance measure | CART | AddTree | GBS | RF | GBM |
|---|---|---|---|---|---|
| Mean BACC | $811.5 \times 10^{-3}$ | $815.5 \times 10^{-3}$ | $804.1 \times 10^{-3}$ | $837.5 \times 10^{-3}$ | $843.4 \times 10^{-3}$ |
| SD BACC | $146.1 \times 10^{-3}$ | $144.8 \times 10^{-3}$ | $156.4 \times 10^{-3}$ | $140.9 \times 10^{-3}$ | $138.2 \times 10^{-3}$ |
| Mean F1 score | $796.7 \times 10^{-3}$ | $799.7 \times 10^{-3}$ | $776.9 \times 10^{-3}$ | $831.9 \times 10^{-3}$ | $833.3 \times 10^{-3}$ |
| SD F1-score | $177.0 \times 10^{-3}$ | $174.8 \times 10^{-3}$ | $193.4 \times 10^{-3}$ | $173.4 \times 10^{-3}$ | $170.0 \times 10^{-3}$ |
| Mean no. of nodes | 55.4 | 50.7 | 4,692.0 | 412,458.4 | 316,728.8 |
| SD no. of nodes | 86.2 | 84.1 | 1,614.1 | 752,654.8 | 700,981.6 |

**Empirical Evaluation of Predictive Performance.** In this section, the predictive performance based on balanced accuracy (BACC) (34) of a set of optimized machine learning models, namely, AddTree, CART, GBS, Random Forest (RF) and Gradient Boosting Machines (GBM), was calculated in 83 classification tasks selected from the Penn Machine Learning Benchmark (PMLB) (35). The frequency of the classification tasks for which one of the aforementioned models outperformed the BACC of all of the others is illustrated in Fig. 2, where ties are not included, for the sake of clarity. AddTree outperformed CART in 55 (66.3%) classification tasks with high statistical significance ($P = 3.08 \times 10^{-7}$), while AddTree was outperformed by GBS in 46 (55.4%) tasks, but with no statistical significance ($P = 0.06$). In fact, AddTree exhibits the highest mean BACC (improvement rate 1.4%) and F1 score (improvement rate 2.9%) across the PMLB tasks, as shown in Table 1. Moreover, RF and GBM displayed superior performance to GBS and to the single trees, that is, CART and AddTree. These results are also consistent with the F1-score comparison shown in *SI Appendix*, Fig. S1.

In Table 1, the average BACC and F1 score of AddTree overcomes GBS, even exhibiting less SD, despite the fact that GBS overcomes AddTree in the counts of classification tasks with better BACC as illustrated in Fig. 2. Furthermore, comparison of the BACCs of AddTree and CART is illustrated on the bar chart in Fig. 3, where the 55 tasks where AddTree outperforms CART are indicated by the bars with $\Delta\text{BACC} > 0$. A similar result is shown in the scatter plot in *SI Appendix*, Fig. S2. Also, notice the meaningful reduction of NDNs of AddTree with respect to GBS. As expected, the ensemble methods RF and GBM are more accurate than the single-tree approaches, but at the expense of having larger NDNs.

**Number of Decision Nodes.** As shown in Table 1, the average NDN used to carry out the classification tasks was 55.4 (SD = 0.18) for CART trees and 50.7 (SD = 0.17) for AddTree trees, and this reduction was statistically significant ($P = 0.001$). Contrasting with the previous NDN values for AddTree and CART, the average number of stumps assembled by GBS to carry out the classification tasks was 4,692.0 (SD = 1,614.1), surpassed only by the large tree ensembles RF with 412,458.4 (SD = 752,654.8) and GBM with 316,728.8 (SD = 700,981.6). In contrast to GBS, AddTree exploits model interactions, which may explain the equivalence in performance with a smaller number of nodes.

**Empirical Variance of AddTree.** By using training data from PMLB, the variance of AddTree as a function of the interpretability parameter $\lambda$ is estimated and shown in Fig. 4. As observed in the plot, as $\lambda$ increases, AddTree reduces variance with respect to CART ($\lambda = 0$) without compromising bias since, as $\lambda \to \infty$, it converges to GBS, which is a consistent estimator. Therefore, the reduction of variance without compromising bias explains the superior performance of AddTree over CART.

## Discussion

The previous section provided a formal proof that AddTree establishes a previously unknown connection between CART and GBS. This connection motivates the hypothesis that the predictive performance of AddTree may surpass the performances of CART and GBS, while improving the limited interpretability that GBS provides. Preliminary results regarding performance of AddTree were previously presented in ref. 36, but using small datasets. The superior statistical power of PMLB allows a more rigorous assessment of the performance and interpretability of AddTree. Therefore, a set of experiments was carried out to validate this hypothesis.

The results presented in Fig. 2 validate the ability of AddTree to surpass the predictive performance of CART over a wide variety of classification tasks. This is further corroborated by *SI Appendix*, Fig. S3, which shows the distribution of the difference $\Delta\text{BACC} = \text{BACC}_{\text{AddTree}} - \text{BACC}_{\text{CART}}$, in which the mean reflects a bias toward positive values of $\Delta$BACC; therefore, $\mathbb{E}\{\text{BACC}_{\text{AddTree}}\} > \mathbb{E}\{\text{BACC}_{\text{CART}}\}$.[†] The superior performance of AddTree over CART has been shown to be statistically significant as well. Moreover, AddTree showed a significant reduction on the average NDN with respect to CART. The difference between the NDNs of AddTree and CART, that is, $\Delta\text{NDN} = \text{NDN}_{\text{AddTree}} - \text{NDN}_{\text{CART}}$, has the distribution shown in *SI Appendix*, Fig. S4, whose mean value is −4.66 (SD = 48.50). Therefore, we conclude that AddTree can generate better predictions than CART while building smaller trees.

Despite the apparent performance advantage of GBS over AddTree, the results are not statistically significant. Furthermore, GBS had an average NDN of 4,692.0 (SD = 1,614.1) over the classification tasks, whereas AddTree had an average NDN of 50.7 (SD = 0.17); however, GBS can be indirectly interpreted through partial dependence plots (23). Notice that limiting the GBS ensemble size to match the maximum depth of CART and AddTree would have provided a better comparison between GBS and AddTree in terms of interpretability based on NDNs. However, the number of stumps of GBS was adaptively added, and the hyperparameters were optimally tuned, to provide the best performance possible, and yet the predictive performance of GBS does not improve over AddTree significantly. Through these results, we conclude that AddTree provides an alternative paradigm for building decision trees with a predictive performance that surpasses the performance of CART and which is as accurate as optimized GBS. AddTree improvement in predictive performance over CART is a result not only of its connection to boosting, but of the effect of regularizing the tree splits through the passing of all of the training data to the descendant nodes and the variation of the weighting scheme. In fact, as illustrated in Fig. 4, AddTree reduces variance as $\lambda$ increases without compromising bias since, as $\lambda \to \infty$, it converges to GBS, a boosting-based algorithm that primarily reduces bias.

The tree ensembles RF and GBM outperformed CART, AddTree, and GBS with statistical significance, at the expense of obtaining less-interpretable models, by using the total NDN

[†]The symbol $\mathbb{E}\{\cdot\}$ denotes the expectation operator.

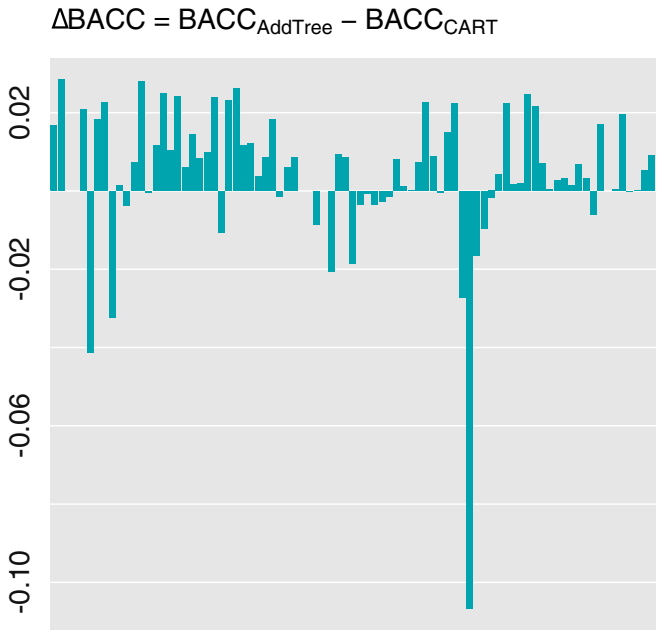


**Fig. 3.** Bar chart showing the difference $\Delta\text{BACC} = \text{BACC}_{\text{AddTree}} - \text{BACC}_{\text{CART}}$. AddTree exhibits significantly better BACC than CART ($P = 3.08 \times 10^{-7}$) in 55 tasks ($\Delta\text{BACC} > 0$) out of the 83 PMLB classification tasks. The one outlier task in favor of CART consists of a synthetic parity problem, which is ill-suited to be captured by any soft regression-like method such as AddTree, and it is better solved by the hard binary logic structure of CART (more details in *SI Appendix*).

as proxies to measure interpretability. However, even in the scenarios where understanding the model is not of concern, the superior predictive performance of AddTree provides possibilities for improving over the predictive performance of RF and GBM, namely, 1) the replacement of the stumps ($\gamma_{n,l}(\boldsymbol{x})$, Eq. **6**) by decision trees such as CART trees, 2) the implementation of bagged AddTree, and 3) the application of gradient boosting to AddTree. Furthermore, 4) formal extensions of AddTree to other losses, and 5) the analysis of connections of AddTree to the recently introduced Generalized Random Forest (37) are in our future research agenda.

## Materials and Methods

All experiments were carried out using the *rtemis* machine learning library (38) written in the R language (39). The statistical significance of all of the predictive performance measurements was evaluated using the Wilcoxon rank sum test (40).

**Predictive Performance Evaluation and NDN.** The PMLB repository (35) contains a selection of curated datasets, accessible through GitHub, for evaluating supervised classification. PMLB includes datasets from commonly used machine learning benchmarks such as the University of California, Irvine machine learning repository (41), the metalearning benchmark (42), and the Knowledge Extraction based on Evolutionary Learning tool (43). It includes real and synthetic datasets in order to assess and compare the performance of different learning algorithms. Out of the 95 available PMLB datasets, 11 datasets with less than 100 instances were ruled out to avoid overfitting in the analysis, and 1 dataset with 48,842 instances was discarded because of the extended execution times that the performance analyses demanded in this particular dataset. The remaining 83 datasets have a mean of 1,713.0 instances (SD = 2,900.3) with 36.3 features (SD = 111.4). The distributions of performance comparisons, including the tests for statistically significant differential performances, were based on viewing the 83 datasets as a random sample of potential classification tasks.

In this study, we evaluated the performance of AddTree against CART, GBS, RF, and GBM using BACC, a measurement commonly used to calculate performance in 2-class imbalanced domains (34). All experiments were performed using nested resampling. For each dataset, the hyperparameters of each algorithm were tuned using grid search to maximize BACC across 5 stratified subsamples (internal resampling). External resampling was performed using 25 stratified subsamples, fixed for each dataset to ensure a direct comparison across algorithms. In both internal and external resampling, the training set size was set to 75% of the available number of cases. The following hyperparameters were tuned: 1) The interpretability factor $\lambda$ and the learning rate were tuned for AddTree, 2) the complexity parameter used in cost complexity pruning (44) was tuned for CART, 3) the learning rate and the maximum number of stumps in the ensemble were tuned for GBS, 4) the number of predictors randomly sampled as candidates at each split was tuned for RF, and 5) the maximum tree depth as well as the learning rate were tuned for GBM. Detailed definitions of the hyperparameters are provided in *SI Appendix*, Table S1, and the hyperparameter space is shown in *SI Appendix*, Table S2.

The AddTree approach grows trees where a minimum of one observation per child node is required to make a node internal; otherwise, such a node becomes a leaf node (min.membership parameter). A maximum tree depth of 30 is also in place (maxdepth parameter); however, this tree depth was not reached in any of the experiments, mostly due to the stopping effect of min.membership. Furthermore, the number of nodes in AddTree is reduced based on the elimination of impossible and redundant paths (i.e., paths where subsequent child branches give the same class in both outputs); however, no other statistical pruning method (e.g., minimum-error or small-tree based pruning) was carried out. In contrast, CART trees were grown and pruned using the *rpart* package (44). The CART tree growth is limited by a minimum number of 2 observations for a split to be attempted (minsplit parameter) and, same as in AddTree, a maxdepth of 30 for fair comparison. In addition, any branch whose complexity measure is less than a value denoted prune.cp is trimmed. The prune.cp is optimally tuned using grid search. CART also relies on a variation of the complexity parameter that indicates that any split which does not decrease the overall lack of fit based on the cross-validated classification error by a factor of 0.0001 is not attempted. The GBS and GBM ensembles are grown and pruned using the *gbm* package (45). Individual trees' growth is limited by the minimum number of 1 observation in the terminal nodes (n.minobsinnode parameter), as well as the maxdepth parameter, which was also tuned using grid search. The number of GBS and GBM trees were tuned by early stopping based on cross-validated classification error. RF tree ensembles were grown and pruned using the *ranger* package (46). The trees were grown until the minimum number of 1 observation at the terminal nodes was achieved (nodesize

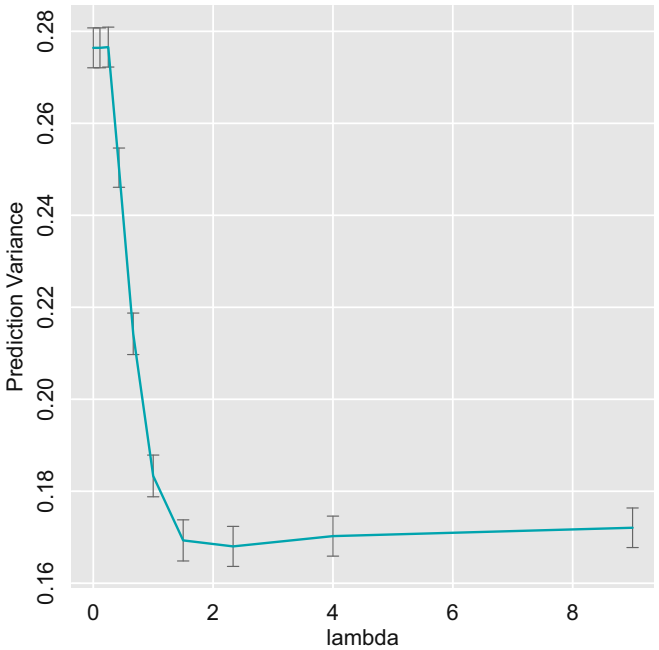


**Fig. 4.** Estimate of the variance of AddTree as a function of the interpretability parameter $\lambda$. Notice that AddTree is able to improve the variance with respect to CART. The error bars represent 1 SE.

parameter). Moreover, a default number of 1,000 ensembled trees is specified (ntree parameter) to try to ensure that every input row gets predicted at least a few times.

**Empirical Variance of AddTree.** The variance of AddTree was calculated as indicated in ref. 47 using the twonorm dataset of PMLB. We calculated the percent of times a test prediction was different from the mode of all test set predictions across the bootstraps where the case was left out (not part of training set); 500 bootstraps were drawn from 1,000 randomly subsampled cases. AddTree was trained on each bootstrap for a set of $\lambda$ values, namely, $\{0.00, 0.11, 0.25, 0.43, 0.67, 1.00, 1.50, 2.33, 4.00, 9.00\}$.

**ACKNOWLEDGMENTS.** This work was partially supported by an award granted by the Emerson Collective. Additionally, the research reported in this publication was supported by the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under Award K08EB026500. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

1. S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, A. Huq, "Algorithmic decision making and the cost of fairness" in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, S. Matwin, S. Yu, F. Farooq, Eds. (SIGKDD, 2017), pp. 797–806.
2. A. Caliskan, J. J. Bryson, A. Narayanan, Semantics derived automatically from language corpora contain human-like biases. *Science* **356**, 183–186 (2017).
3. F. Cabitza, G. Banfi, Machine learning in laboratory medicine: Waiting for the flood? *Clin. Chem. Lab. Med.* **56**, 516–524 (2018).
4. M. A.-M. Salem, A. Atef, A. Salah, M. Shams, "Recent survey on medical image segmentation" in *Computer Vision: Concepts, Methodologies, Tools, and Applications*, M. Khosrow-Pour, S. Clarke, M. E. Jennex, A. Becker, A.-V. Anttiroiko, Eds. (IGI Global, 2018), pp. 129–169.
5. P. Yadav, M. Steinbach, V. Kumar, G. Simon, Mining electronic health records (EHRs): A survey. *ACM Comput. Surv.* **50**, 1–40 (2018).
6. D. Sculley *et al.*, "Hidden technical debt in machine learning systems" in *Proceedings of International Conference on Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett, Eds. (NeurIPS, 2015), pp. 2503–2511.
7. F. Louzada, A. Ara, G. B. Fernandes, Classification methods applied to credit scoring: Systematic review and overall comparison. *Surv. Oper. Res. Manage. Sci.* **21**, 117–134 (2016).
8. J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks. *ProPublica*. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Accessed 1 October 2018.
9. Z. C. Lipton, "The doctor just won't accept that!" in *Proceedings of Symposium of Interpretable Machine Learning at the International Conference on Neural Information Processing Systems*, A. G. Wilson, J. Yosinski, P. Simard, R. Caruana, W. Herlands, Eds. (NeurIPS, 2017), pp. 1–3.
10. R. Caruana *et al.*, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission" in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, L. Cao *et al.*, Eds. (SIGKDD, 2015), pp. 1721–1730.
11. B. Goodman, S. Flaxman, "European Union regulations on algorithmic decision-making and a 'right to explanation'" in *Proceedings of Workshop on Human Interpretability in Machine Learning at the International Conference on Machine Learning*, B. Kim, D. M. Malioutov, K. R. Varshney, Eds. (International Conference on Machine Learning, 2016), pp. 1–9.
12. Z. C. Lipton, The mythos of model interpretability. arXiv:1606.03490. Deposited 6 March 2017.
13. F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning. arXiv:1702.08608. Deposited 2 March 2017.
14. F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. Wortman Vaughan, H. Wallach, "Manipulating and measuring model interpretability" in *Proceedings of Symposium of Interpretable Machine Learning at the International Conference on Neural Information Processing Systems*, A. G. Wilson, J. Yosinski, P. Simard, R. Caruana, W. Herlands, Eds. (NeurIPS, 2017), pp. 1–14.
15. M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier" in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, B. Krishnapuram *et al.*, Eds. (SIGKDD, 2016), pp. 1135–1144.
16. L. A. Hendricks *et al.*, "Generating visual explanations" in *European Conference On Computer Vision*, B. Leibe, J. Matas, N. Sebe, M. Welling, Eds. (Springer, 2016), pp. 3–19.
17. L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, *Classification and Regression Trees* (Wadsworth & Brooks, Monterey, CA, 1984).
18. W.-Y. Loh, Fifty years of classification and regression trees. *Int. Stat. Rev.* **82**, 329–348 (2014).
19. S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology. *IEEE Trans. Sys. Man Cybernetics* **21**, 660–674 (1991).
20. B. Kim, J. A. Shah, F. Doshi-Velez, "Mind the gap: A generative approach to interpretable feature selection and extraction" in *Proceedings of International Conference on Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett, Eds. (NeurIPS, 2015), pp. 2260–2268.
21. J. M. Luna *et al.*, Predicting radiation pneumonitis in locally advanced stage II–III non-small cell lung cancer using machine learning. *Radiother. Oncol.* **133**, 106–112 (2019).
22. R. Caruana, A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms" in *Proceedings of the International Conference on Machine Learning*, W. Cohen, A. Moore, Eds. (Association for Computing Machinery, 2006), pp. 161–168.
23. J. H. Friedman, Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **29**, 1189–1232 (2001).
24. W.-Y. Loh, Regression trees with unbiased variable selection and interaction detection. *Stat. Sin.* **12**, 361–386 (2002).
25. W.-Y. Loh, Improving the precision of classification trees. *Ann. Appl. Stat.* **3**, 1710–1737 (2009).
26. M. Kearns, Y. Mansour, On the boosting ability of top-down decision tree learning algorithms. *J. Comput. Syst. Sci.* **58**, 109–128 (1999).
27. J. R. Quinlan, *C4.5: Programs for Machine Learning* (Elsevier, 2014).
28. Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **55**, 119–139 (1997).
29. E. Grossmann, "AdaTree: Boosting a weak classifier into a decision tree" in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, Z. Zhu, R. Kumar, Y.-P. Hung, R. Haralick, A. Hanson, Eds. (IEEE, 2004), pp. 105–105.
30. Z. Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering" in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, S. Ma, H.-Y. Shum, W. T. Freeman, L. Van Gool, S. Chaudhuri, Eds. (IEEE, 2005), vol. 2, pp. 1589–1596.
31. G. Valdes *et al.*, MediBoost: A patient stratification tool for interpretable decision making in the era of precision medicine. *Sci. Rep.* **6**, 37854 (2016).
32. J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: A statistical view of boosting. *Ann. Stat.* **28**, 337–407 (2000).
33. Y. Yuan, M. J. Shaw, Induction of fuzzy decision trees. *Fuzzy Sets Syst.* **69**, 125–139 (1995).
34. K. H. Brodersen, C. S. Ong, K. E. Stephan, J. M. Buhmann, "The balanced accuracy and its posterior distribution" in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, A. Ercil, K. Boyer, M. Cetin, S.-W. Lee, Eds. (IEEE, 2010), pp. 3121–3124.
35. R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, J. H. Moore, PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Min.* **10**, 36 (2017).
36. J. M. Luna *et al.*, "Tree-structured boosting: Connections between gradient boosted stumps and full decision trees" in *Conference on Neural Information Processing Systems (NIPS 2017)*, A. G. Wilson, J. Yosinski, P. Simard, R. Caruana, W. Herlands, Eds. (NeurIPS, 2017)., pp. 1–8.
37. S. Athey, J. Tibshirani, S. Wager, Generalized random forests. *Ann. Stat.* **47**, 1148–1178 (2019).
38. E. D. Gennatas, "Towards precision psychiatry: Gray matter development and cognition in adolescence," PhD thesis, University of Pennsylvania, University Park, PA (2017).
39. R Core Team, *R: A Language and Environment for Statistical Computing* (R Foundation for Statistical Computing, Vienna, Austria, 2018).
40. M. Pagano, K. Gauvreau, *Principles of Biostatistics* (Chapman and Hall/CRC, 2018).
41. D. Dheeru, E. K. Taniskidou, UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 15 June 2017.
42. M. Reif, "A comprehensive dataset for evaluating approaches of various meta-learning tasks" in *Proceedings of the International Conference on Pattern Recognition and Methods (ICPRAM)*, J. S. Sánchez, P. L. Carmona, Eds. (IEEE, 2012), pp. 273–276.
43. J. Alcalá-Fdez *et al.*, Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Logic Soft Comput.* **17**, 255–287 (2011).
44. T. Therneau, B. Atkinson, B. Ripley, package 'rpart.' R package version 4.1-13. https://cran.r-project.org/web/packages/rpart/rpart.pdf. Accessed 1 July 2018.
45. G. Ridgeway *et al.*, package 'gbm.' R package version 2.1.3. https://cran.r-project.org/web/packages/gbm/gbm.pdf. Accessed 7 July 2018.
46. M. N. Wright, A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *J. Stat. Softw.* **77**, 1–17 (2017).
47. G. Valentini, T. G. Dietterich, "Low bias bagged support vector machines" in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, T. Fawcett, N. Mishra, Eds. (AAAI Press, 2003), pp. 752–759.