Information Sciences 572 (2021) 522–542

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Approximating XGBoost with an interpretable decision tree

Omer Sagi, Lior Rokach

Department of Information Systems and Software Engineering, Ben-Gurion University of the Negev, P.O.B. 653, Beer-Sheva 8410501, Israel

A R T I C L E I N F O

Article history:
Received 8 July 2020
Received in revised form 18 May 2021
Accepted 25 May 2021
Available online 27 May 2021

Keywords:
Classification trees
Decision forest
Ensemble learning

A B S T R A C T

The increasing usage of machine-learning models in critical domains has recently stressed the necessity of interpretable machine-learning models. In areas like healthcare, finary – the model consumer must understand the rationale behind the model output in order to use it when making a decision. For this reason, it is impossible to use black-box models in these scenarios, regardless of their high predictive performance. Decision forests, and in particular Gradient Boosting Decision Trees (GBDT), are examples of this kind of model. GBDT models are considered the state-of-the-art in many classification challenges, reflected by the fact that the majority of Kaggle's recent winners used GBDT methods as a part of their solution (such as XGBoost). But despite their superior predictive performance, they cannot be used in tasks that require transparency. This paper presents a novel method for transforming a decision forest of any kind into an interpretable decision tree. The method extends the tool-set available for machine learning practitioners, who want to exploit the interpretability of decision trees without significantly impairing the predictive performance gained by GBDT models like XGBoost. We show in an empirical evaluation that in some cases the generated tree is able to approximate the predictive performance of a XGBoost model while enabling better transparency of the outputs.

© 2021 Elsevier Inc. All rights reserved.

1. Introduction

The increased deployment of machine-learning models into new domains has recently accelerated a broad discussion around the importance of model interoperability [5]. In some domains, experts are required to either understand the mechanism by which the model works or to be able to justify decisions that are based on model outputs. It is not sufficient, for example, for a medical diagnosis model to be accurate. It also needs to be transparent to the expert that uses its output when making a decision about a certain patient. But even in scenarios that allow a larger margin of error, humans are less likely to accept models that they cannot comprehend. Interpretable machine-learning models address these issues. They are defined as models that can be clearly visualized or explained using plain text to the end-user [29]. A decision tree is a broadly used example of an interpretable model. Every classification made by a decision tree can be associated with a corresponding decision path. In addition, the hierarchical structure of the model as a whole can be easily visualized or described to users of any level of expertise [4]. But despite their high level of interpretability, decision trees have limited predictive performance due to the myopic nature of their induction algorithms. These algorithms usually fail to capture complex interactions among the input features, leading to fundamental biases in cases where such interactions exist. This issue can be addressed by training an ensemble of decision trees, also known as a decision forest.

Decision forests combine multiple decision trees towards providing a single output in supervised machine-learning tasks. The ability of decision forests to integrate different hypotheses in a single model and their robustness to any type of relational dataset have driven their popularity within the data science community [37]. Gradient Boosting Decision Tree (GBDT)

https://doi.org/10.1016/j.ins.2021.05.055
0020-0255/© 2021 Elsevier Inc. All rights reserved.

is a sub-group of decision forests that includes models like XGBoost, CatBoost, and LightGBM. These models have recently found to be highly effective in numerous tasks, as reflected by the fact that most of Kaggle's recent winners used these methods in their solutions. However, decision forests as a whole, and GBDT models, in particular, are considered to be black-box models. Each classification that is made by a decision forest is required to go over numerous different trees. Therefore, the end-user cannot have a clear justification of the predictions made by the model. Furthermore, it is impossible for the end-user to grasp the model structure as it practically composed of numerous single models. A plethora of studies addressed the complexity of decision forests by presenting ensemble pruning approaches. These approaches aim at filtering a subset of base trees that performs at least as well as the original decision forest [28]. While these methods reduce the complexity without impairing the predictive performance, the end-result cannot be considered as an interpretable model. Several studies, mainly from the past few years presented methods for transforming a decision forest into a single decision tree. Some of these methods require the synthesis of a large set of unlabelled data [16,50] while others propose algorithms that manipulate the ensemble nodes into a new decision tree [42,43,45].

This paper presents a novel method for generating a single decision tree that is based on a previously trained decision forest. The generated tree aims at approximating the predictive performance of the decision forest. At the same time, it provides an explanatory mechanism for its classifications, enabling the end-user to understand its structure. This work can be viewed as an extension of the Forest-Based Tree (FBT) method that was developed and evaluated for independently induced decision forests, i.e. – decision forests in which the base-trees are trained independently [39] and are usually consist of a small number of deep trees rather than a large number of shallow trees. The main contribution of the new method is its applicability for both independently induced decision forests (e.g., random forest) and dependently induced decision forests (e.g. gradient boosting machines and XGBoost). It is done by refining the algorithm to be more focused on extracting information that is relevant to the original training set rather than considering only the explicit characteristics of the base trees. Another important contribution of the developed method is that it enables the configuration of the tree complexity by determining its maximum depth. Consequently, this configurable method allows its users to better control and address the trade-off between interpretability and predictive performance. The method includes three main stages. First, we conduct an ensemble pruning to the pre-trained ensemble. Then we extract a representative set of conjunctions from the pruned ensemble and finally, we build a decision tree that organizes the conjunction set in a tree structure. The remainder of the paper is structured as follows: In Section 2 we lay the scientific background and present related studies. In Section 3, we present the developed method. Section 4 presents an experimental evaluation for binary classification challenges and discusses its results. Section 5 concludes and suggests future research directions.

2. Background

Decision forests, and in particular gradient boosting decision trees (GBDT), are considered the best practice in many classification challenges [11,38]. However, interpretable models like decision trees usually opt over decision forests when either the model or its predictions are required to be transparent to the end-user. Building a decision tree that approximates the predictive performance of a given decision forest, with the focus on GBDT models, is the subject of this paper and in the following section, we provide its relevant scientific background. We review GBDT methods, interpretable machine learning, and the usage of decision trees in driving interpretability in machine-learning tasks.

2.1. Decision forests and gradient boosting decision trees

Ensemble learning is a powerful machine-learning technique that combines outputs of several machine-learning models to make a single decision, usually in classification or regression problems. The effectiveness of ensemble models is evident in cases where single model errors are likely to be compensated by other models towards improving the overall predictive performance [15]. These methods are considered the best practice for challenges involved with relational datasets, where each sample (e.g., person) consists of meaningful features of different nature (e.g., age, gender) [46]. Evaluated against deep neural networks, ensemble learning is suitable for small datasets but is also likely to produce inferior results for image-like or sequence-like data entities [37].

Among the ensemble methods used in practice, gradient boosted decision trees (GBDT) is probably the most frequently preferred method in many real-world applications [32,33]. In GBDT, multiple decision trees are constructed where the training of each tree hinges on previously trained trees [18]. In practice, each tree is trained to predict the pseudo-residuals of the previous trees, given a predefined objective function. Inferencing for new instances is applied in an additive manner where each tree adds its residual to the aggregated result. The usage of GBDT has become broader in recent years due to several new developments. The most notable one is Extreme Gradient Boosting (XGBoost) [9], a scalable machine learning system for tree boosting. The main novelty of XGBoost is that it adds a regularization component to the loss function so the complexity of the resulting ensemble is considered along with the predictability at each split. Furthermore, XGBoost enables its users to mitigate model overfitting by tuning multiple hyper-parameters such as tree single complexity, forest complexity, learning rate, regularization terms, column subspaces, dropouts, etc. XGBoost presents additional novelties such as handling missing data with nodes' default directions, enumerating efficiently over potential splitting thresholds during node split, and supporting distributed platforms such as Apache Hadoop. LightGBM is another recently developed GBDT method [27]. It includes a procedure that deals with a large number of sparse features by combining features that never take non-zero values simultaneously into a new single feature. In addition, LightGBM deals with a large number of instances by favoring instances with large gradients when moving from one tree to another. From an implementation standpoint, LightGBM uses histogram-based algorithms and network communication algorithms to reduce resource consumption and enable parallel learning at training time. CatBoost is a new GBDT toolkit developed at Yandex [32]. It replaces high-dimensional categorical features with single numerical values using target statistics.

The importance of the above-mentioned methods and of GBDT, in general, can be reflected by the fact that the majority of Kaggle's recent winners used GBDT methods as a part of their solution [9]. Another evidence is the increased number of professional online discussions evolved around GBDT as depicted in Fig. 1. Furthermore, a recent study that compared XGBoost with other established models, using 71 datasets, found XGBoost to be superior in terms of predictive performance and prediction efficiency[46]. From an applicability perspective, there are multiple pieces of evidence to the effectiveness of XGBoost in various domains such as healthcare [10], drug discovery [49], risk assessment [22] and more.

Despite its high predictive performance, there are some cases that do not allow the usage GBDT due to the limited interpretability that is enabled by the model. For example, in classification challenges that require a clear rational explanation for each classification (e.g. medicine, insurance, etc.). There are other practices where the black-box nature of GBDT models may deter users from using its predictions. Users may feel uncomfortable relying their decisions on GBDT's classifications as they cannot have a rational explanation for them [12]. In other cases, users would prefer to gain domain insights from the inner structure of the model; something that cannot be provided with GBDT [41].

2.2. Interpretable machine-learning models

The tension between the predictive performance and the level of interpretability of machine learning models has recently gained broader attention in a plethora of studies [29,5]. The main reason for this emerged discussion is the penetration of machine-learning into traditional domains, like health-care or finance, that involved critical decision making (e.g., medical diagnosis, determining the risk of a loan, etc.) [2]. For this kind of domain, models that merely optimize predictive performance may not be usable if they do not provide a certain level of transparency to the end-user. Standing as the opposite of black-box models, transparent models enable a direct understanding of the mechanism by which a model works [5]. Transparency can also be referred to in equivalent terms like interpretability, understandability, and intelligibility. In practical terms, a model may be regarded as interpretable if it enables the manual calculation of an outcome, given the model parameters and input data, in a reasonable amount of time [29]. Alternatively, a model is considered interpretable if it can be visualized or described in plain text to the end-user [34]. Another practical aspect of model interpretability stems from the ability to extract and communicate its underlying knowledge to its consumers. In contrast to interpretable models, black-box models are models that do not disclose any meaningful information about their outputs or their internal structure. GBDT and

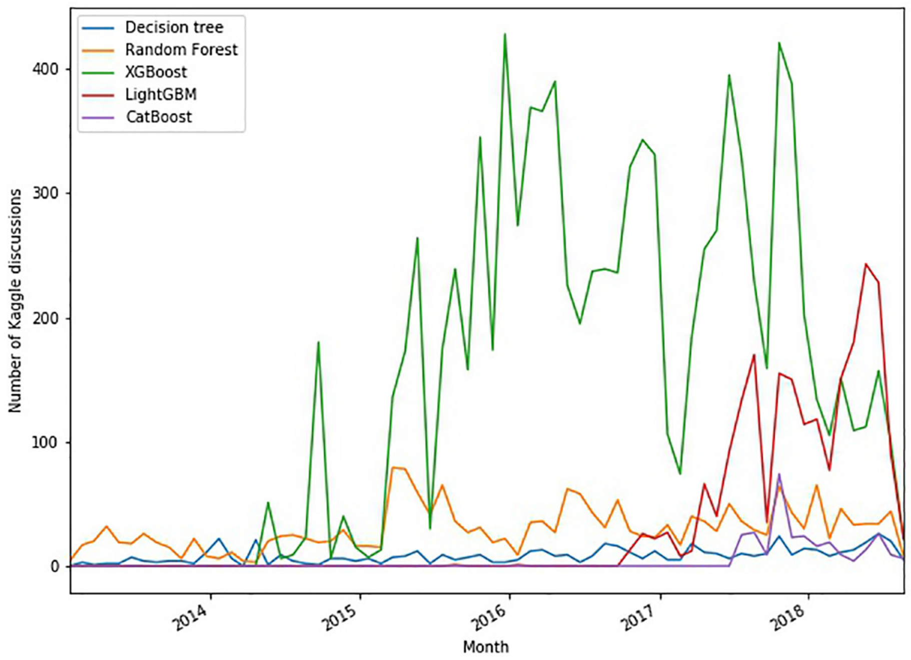


Fig. 1. Number of Kaggle discussions per tree-based algorithm over time (https://www.kaggle.com/shivamb/data-science-trends-on-kaggle).

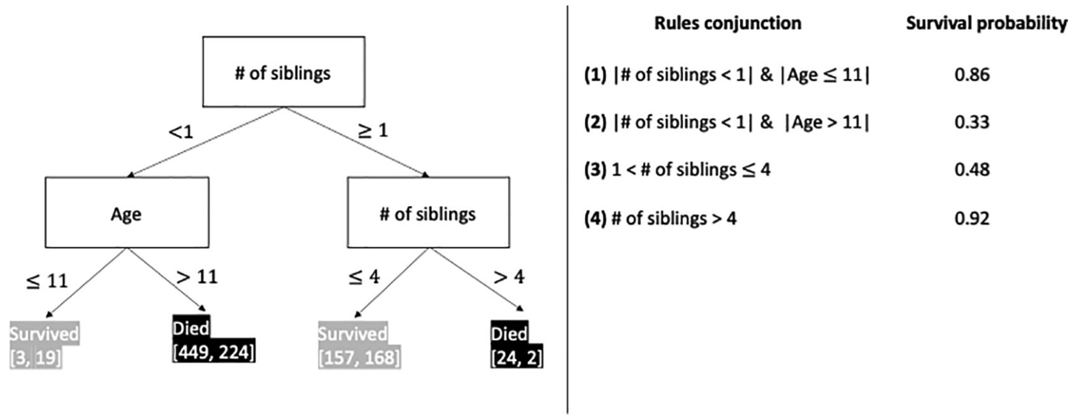
deep neural networks are examples of black-box models; despite their superior predictive performance in many tasks, it is impossible to use these models for problems that require interpretability.

While predictive performance is an obviously desired property of a machine-learning model, the importance of interpretability was only recently highlighted in multiple studies. For example, the theory-guided data-science paradigm [26] mentions a lack of interpretability as a substantial obstacle in extracting scientific knowledge from an accurate model. The authors describe Google Flu Trends as an example of a model that achieved satisfactory accuracy on training and evaluation datasets but could not be used for gaining insights about its related scientific domain [25]. In the judicial domain, a black-box model that was used by several justice institutes to predict crime recidivism has appeared to be demographically biased, leading to prejudicial decisions that could have been prevented with a better understanding of the model mechanism [7]. Another example is Amazon's same-day delivery service that was found in a report written in 2016 to be unintentionally less likely to be offered in minority neighbourhoods [20]. The importance of interpretability in improving the fairness of machine-learning algorithms is also stressed in the Explainable AI (XAI) program, initiated by DRAPA, to address ethical issues in the military domain. There has been a surge of interest in explainable artificial intelligence recently following DRAPA's initiative [1]. FICO has launched a similar program in the financial domain where its objective is to *"create models and techniques that both accurate and provide good trustworthy explanations*" [2]. Finally, in the General Data Protection Regulation (GDPR), published by the European Union, organizations are required to address the "*right to explanation*" when using automated algorithms for decision making.

Different techniques that tackle the challenge of improving the interpretability of machine-learning models were presented in recent studies. These techniques can be categorized into local interpretability techniques and globally interpretable models [19]. The former type of technique aims at providing digestible explanations for individual predictions without necessarily interpret the model mechanism as a whole. Notable examples are LIME and SHAP that learn an interpretable local model around individual predictions [34]. Global interpretability is achieved by generating an interpretable model from a given opaque model. It is broadly agreed that a small set of interpretable models are recognized: decision trees, decision rules, nearest-neighbors, and linear models [34]. In the context of GBDT, approaches that were developed with the aim of simplifying ensemble models have focused on ensemble pruning for a long time [38]. Ensemble pruning methods search for a subset of representing members that has at least the same level of accuracy as the original ensemble [47]. One option to apply ensemble pruning is to leverage ranking methods that conduct greedy selection of trees, usually based on complexity or accuracy metrics [23]. Another option is to apply search heuristics [24]. Ensemble pruning methods have been shown to reduce the complexity of ensemble models. However, the new pruned model cannot be considered an interpretable model. Local interpretability of decisions made by tree ensembles is enabled by a mechanism for sensitivity analysis of individual predictions [30] or by using feature contributions to present the connection between an instance and its corresponding prediction [48]. On the other hand, Global interpretability is suggested to be obtained by applying ensemble derived models. These models are created by leveraging the properties of a given ensemble model to generate an interpretable model. It can be manifested by extracting meaningful rules from a given ensemble model or by building a single interpretable model that learns the classifications of a trained ensemble [6]. A common approach to derive an interpretable model from a given ensemble is the single-tree approximation, as described in detail at the next section.

2.3. Decision tree as an interpretable model

A decision tree is a classification model composed of test nodes that connect attributes into decision nodes, also known as leaves. Starting from the root node, each test node determines the division of the dataset into disjoint subsets. This process reoccurs in a recursive manner until no further division is required. A simple decision tree is illustrated in Fig. 2 (a) where the goal is to predict whether a passenger will survive the sinking Titanic, based on attributes like age, gender, number of sib-



Rules conjunction	Survival probability
(1) |# of siblings < 1| & |Age ≤ 11|	0.86
(2) |# of siblings < 1| & |Age > 11|	0.33
(3) 1 < # of siblings ≤ 4	0.48
(4) # of siblings > 4	0.92

Fig. 2. a. An example of a decision tree that predicts whether a Titanic passenger is likely to be survived based on its attributes. b. a breakdown of the tree into conjunctions of tests that are mapped to a decision).

lings, etc. In the constructed tree, nodes contain the class distribution (the first dimension refers to "died" and the second dimension refers to "survived") and the percentage of training instances that were routed to the node. Explanations of specific decisions can be easily provided since each leaf is translated into clear conjunction of attributes [4], as depicted in Fig. 2 (b). Due to this capability, decision trees are broadly used for machine-learning problems that require the understanding of the model structure as well as its outputs [45]. Furthermore, predictions made by decision trees are often faster, compared to ensemble models as only a few attributes are usually tested when classifying a given instance [38]. There are multiple induction algorithms for generating decision trees [8]. But in general, most induction algorithms search for the best splitting attribute following each partition of the tree in a greedy manner, relying on a predefined splitting criterion (e.g., Gini coefficient, entropy, etc.) [8]. The main drawback of this approach is that considering only one level ahead when selecting a splitting test limits the effectiveness of decision trees in modeling complex interactions among several attributes. In addition, the induction algorithms of decision trees are over-sensitive to slight changes in the training set which make them relatively unstable.

There were several attempts to generate an interpretable decision tree that approximates the predictive performance of a trained black-box model. One of the early works presented *TREPAN* [13], a method that uses a neural network as an oracle that classifies synthesized data that is then fed into an induction algorithm for training a decision tree. CMM [16] is another example in which artificial data is generated and then being classified by a black-box model so it can be later used as input data for a tree inducer. This method has recently shown to be effective in the adaptive shortening of medical questionnaires [50]. Deriving a single tree from a given ensemble model can also be conducted by including a post-processing stage that manipulates the ensemble's inner structure to create the new tree. An early work that manifested this approach has presented an algorithm that extracts if-then rules from a given ensemble. Rules are then converted into binary vectors that serve as training data of a new decision tree [3]. In interpretable single model (ISM) [42] the new tree is constructed by running an iterative selection of the splitting feature according to its importance within the original ensemble. This method requires the evaluation of each of the ensemble's test nodes at each split. Genetic extraction of a single interpretable model (GENESIM) [43] is a technique that applies genetic algorithms towards building a new tree from a given ensemble. The complexity of both ISM and GENESIM is exponential with the ensemble size as well as with the dimensionality of the input data. As a result, they can be applied only on small ensembles and small datasets. Global interpretation via recursive partitioning (GIRP) [45] is a method that builds a binary decision tree from a wide range of trained models by conducting recursive partitioning of the input space. The partitioning relies on the relative contribution of the features in the classifications of the original model. The method has been claimed to be useful in interpreting a random forest that was built for text classification. However, it was neither evaluated in solving other problems nor has an available implementation for testing its effectiveness.

3. Converting gradient boosting decision tree into a single tree

This section presents a method for generating a decision tree from a given decision forest. The presented method includes extensions and refinements to the forest-based tree (FBT) method presented in [39]. The main refinement is in adapting the conjunction set generation stage that breaks the decision forest into numerous building blocks. This stage was refined to consider the properties of the training data (e.g., class distribution, conjunctions that are relevant to the training instances) in a way that is parallel to the dependency of each base-tree previously trained trees. As the previous version of the method has focused on processing ensembles composed of independent base models (e.g., random-forest and extra-trees), the presented method was designed to be applicable for ensembles of dependent base models (like GBM and XGBoost). It is done by adapting the method to consider the connection between the source decision forest and the original training set rather than solely consider the inner structure of the decision forest. Moreover, the method proposed here enables the user to fine-tune the trade-off between predictive performance and tree complexity by configuring the maximum depth of the generated tree.

3.1. Method overview

Let (x, y) be an observation in the set D such that x is an m-dimensional vector of m features and $y \in [0, C]$ is a target variable where C is the number of classes. We train a decision forest $\hat{f}$ using D by constructing K base trees $\{T_1, \ldots, T_K\}$ where $\{h_{1(x)}, \ldots h_{K(x)}\}$ are the outputs of the K base trees for a given feature vector x and g is a function that aggregates the base-tree outputs to generate class probabilities vector. The trained decision forest is then used for inferencing new instances as follows:

$$f_x = g(\{h_{1(x)}, \ldots h_{K(x)}\}) \quad (1)$$

where the functionality of g depends on the forest type. For example: in the case of random forest g will be an average over the class probabilities derived from each base-tree while in XGBoost, g will be a softmax over the sum of base trees log-odds. The objective of the presented algorithm is to use f as an input for generating a new tree t that approximates the predictive performance of f as follows:

$$\forall x, t_x \approx f_x \quad (2)$$

The developed method extends the tool-set available for machine learning practitioners who want to exploit the interpretability of decision trees without significantly impairing the predictive performance gained by decision forests. its main premise is that decision trees and decision forests can both be represented as a finite set of conjunctive rules. It extends the previous version presented in [39] by being adapted to GBDT and by enabling the user to better control the trade-off between predictive performance and complexity. The method consists of three main stages:

1. Conducting ensemble-pruning in order to reduce the number of leaves considered when training the new tree.
2. Extracting a finite set of conjunctive rules that are likely to be applied by the pruned forest.
3. Organizing the extracted conjunctive rules as a hierarchical decision tree.

3.2. Decision forest pruning

Forest pruning is conducted as a preliminary stage of the developed method. The objective here is to obtain a smaller ensemble that includes only the most relevant base trees. It is driven by the assumption that a more parsimonious ensemble will decrease the likelihood of extracting irrelevant components when generating the new tree in the next stages. The pruning is conducted using a greedy algorithm that adds base trees to generate a new ensemble in an iterative manner, as formulated in Algorithm 1. Starting with an empty ensemble, the algorithm searches within the given ensemble for the base-tree that provides the highest ROC AUC when classifying a pruning-set (can also be the training set). This base tree is then added to the empty ensemble. In the next stage, the algorithm searches for the next base tree to be added out of the remaining trees. It adds the base tree that its combination with the already added tree will result in a two-tree ensemble with the maximum ROC AUC, compared to all the other combinations. This process is repeated until there is no base tree that its addition to the ensemble improves the ROC AUC.

A potential limitation of the described procedure is that in some cases, the composition of the new ensemble may stop too early. For example: in cases where adding the second tree does not improve the ROC AUC of the first tree as the first tree overfits the training set. In order to prevent a scenario as such, we define the minimum size of the final pruned ensemble as a hyper-parameter so the procedure stops if there is no base-tree that its addition improves the training set ROC AUC and when the number of included base trees is larger than the minimum size provided by the user as a hyper-parameter. It is also to be noted that while the training set can be used as the default pruning set, it is possible to prune the ensemble using a designated pruning set that was not used for training the model and is not expected to be used for evaluating the performance of the final tree.

The selection of this benefit-driven greedy approach was made for two main reasons. First, past studies showed that this kind of method can dramatically reduce the number of required trees [17]. This attribute is particularly important as the objective of the developed algorithm is to exclude as many decision rules as possible without adequately impairing the predictive performance of the original decision forest. The second reason for selecting this pruning approach is its ease of use. As opposed to other pruning methods, this method does not require the specification of complex hyper-parameters other than the number of minimum base models and therefore, the functionality of the algorithm is more likely to be transparent to the end-user. Nonetheless, this greedy approach can be easily replaced with other types of pruning methods if needed and the selection of the most suitable pruning approach can be further explored in future researches.

Algorithm 1: Generating a pruned forest from the given decision forest

```
  Input: $f$ (A decision forest composed of $K$ trees), $MS$ (minimum size of the pruned ensemble), $D$ (pruning set)
  Output: $\hat{f}$ (A pruned ensemble, a sub-set of $N$ base-trees from $f$ where $MS \leqslant N \leqslant K$)
1: $\hat{f} = \varnothing$
2: while $|\hat{f}| < MS$ and $AUC(\hat{f}, D)$ was improved in the previous iteration do
3:    if $|\hat{f}| > 0$ then
4:       $MIN\_AUC = AUC(\hat{f}, D)$
5:    else
6:       $MIN\_AUC = 0$
7:    end
8:    for $t_k$ in $f$ do
9:       set $\hat{f_k}$ to include all the base-trees in $\hat{f}$ and $t_k$
10:      if $AUC(\hat{f_k}, D) > MIN\_AUC$ then
11:         $MIN\_AUC = AUC(\hat{f_k}, D)$
12:         $\hat{t} = t_k$
13:      end
14:   end
15:   Add $\hat{t}$ to $\hat{f}$
16: end
```

3.3. Extraction of rule conjunctions

Following the ensemble pruning stage, we apply a procedure that aims at decomposing the pruned ensemble and extracting a set of conjunctive rules that can be used for classification. These conjunctive rules serve as the building blocks for generating the final decision tree. The generation of the conjunction set that represents the pruned forest is done by gradually merging the conjunction sets of its base trees into a single set that represents the entire ensemble. It is done by defining an empty set of conjunctive rules and then gradually merging new conjunctions from the different base trees.

The execution of this stage ignores the hierarchical organizations of the base trees that compose the pruned decision forest. Instead, each tree is viewed as a set of conjunctive rules mapped into class probabilities (a single probability of the positive class in binary classification challenges). Virtually, single trees are extracted as lists of leaves so the tree is decomposed from its hierarchical structure. The procedure starts with breaking each tree into a set of conjunctive rules that are mapped into a vector of class probabilities as illustrated in Fig. 2. The conjunction set that represents the pruned forest is then initialized to be the conjunction set of the first tree that was added to the pruned forest at the previous stage, namely, the base-tree with the best performance for the pruning set. Following the initialization, we gradually merge the conjunctions of the other base-trees into the final set. The order of the base-trees that are merged into the final set is determined by the order of their addition into the pruned forest. The approach is driven by the idea that merging the conjunctions of two trees will result in the conjunction set of a single decomposed tree. We provide an explanation to this notion through an illustrative example.

Fig. 3 shows a simple decision forest of two base trees, where each tree is decomposed into a set of conjunctions. Each conjunction represents a single leaf in the given tree. The forest classifies the Titanic passengers as died or survived while the end result of each leaf is the survival probability, calculated as the proportion of survived passengers out of all of the passengers that were routed to the given leaf. Merging the two conjunction sets into a new conjunction set, representing a new tree, is illustrated in Fig. 4. Taking as an example the merge of leaf A from tree 1 with leaf A from tree 2 (1A-2A), rules of leaf 1A ($\#ofsiblings < 1$ and $age \leqslant 11$) are merged with the single rule of 2A ($Fareprice < 10$) and the resulting survival probability (0.54) is determined as the average of the two leaves' probabilities (0.86 and 0.22). In this example, the probabilities of the leaves were calculated as the proportion of training instances with positive class values that were routed into the leaf when training the base-trees. However, in other cases this value may be derived in different ways (e.g., log-odds in gradient boosting trees) Notice that although there are four conjunctions in each set, the resulting conjunction set has only 12 conjunctions rather than 16 (4 X 4). This is the result of excluding conjunctions that contain incompatible rules. For example conjunctions 1A and 2D cannot be merged, as the resulting conjunction will include incompatible ($\#ofsiblings$) rules. Another point that should be stressed is the reduction of rules that occur naturally when merging some of the leaves. For

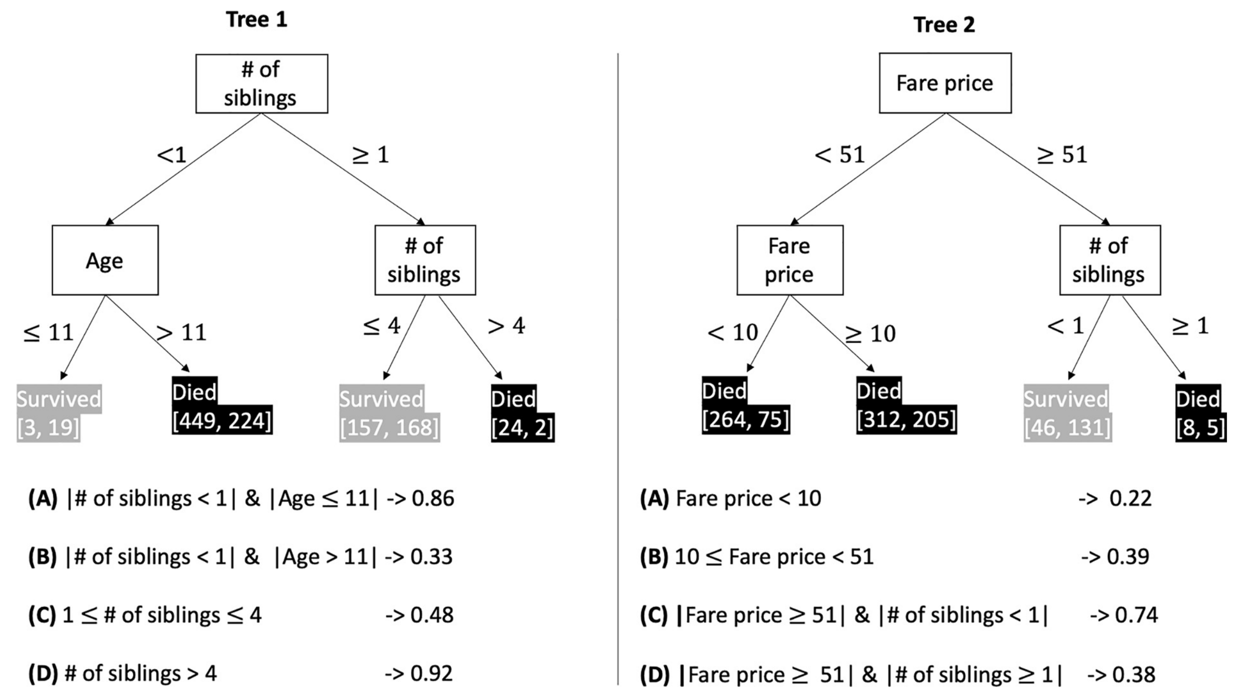
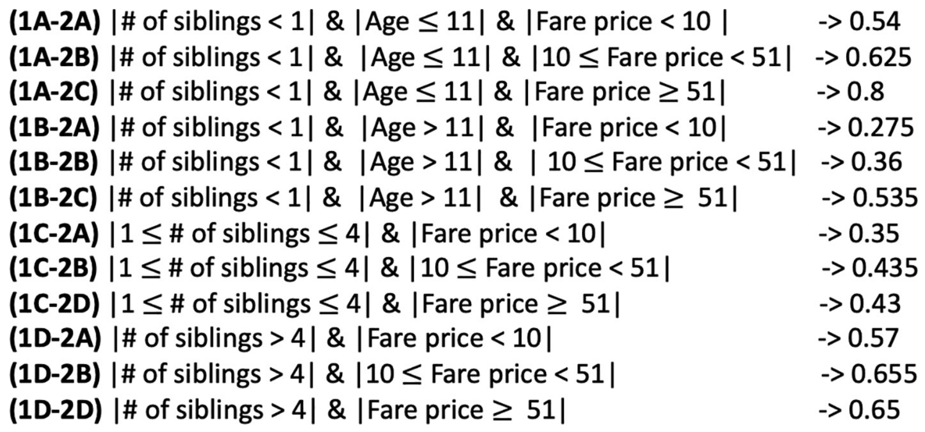

Tree 1
# of siblings
<1
≥ 1
Age
# of siblings
≤ 11
> 11
≤ 4
> 4
Survived [3, 19]
Died [449, 224]
Survived [157, 168]
Died [24, 2]
(A) |# of siblings < 1| & |Age ≤ 11| -> 0.86
(B) |# of siblings < 1| & |Age > 11| -> 0.33
(C) 1 ≤ # of siblings ≤ 4 -> 0.48
(D) # of siblings > 4 -> 0.92
Tree 2
Fare price
< 51
≥ 51
Fare price
# of siblings
< 10
≥ 10
< 1
≥ 1
Died [264, 75]
Died [312, 205]
Survived [46, 131]
Died [8, 5]
(A) Fare price < 10 -> 0.22
(B) 10 ≤ Fare price < 51 -> 0.39
(C) |Fare price ≥ 51| & |# of siblings < 1| -> 0.74
(D) |Fare price ≥ 51| & |# of siblings ≥ 1| -> 0.38

Fig. 3. An example of a decision forest of two trees and the corresponding conjunction sets. The number that each conjunction points at is the probability of a passenger that meet this leaf to survive the sinking Titanic.



(1A-2A)	|# of siblings < 1| & |Age ≤ 11| & |Fare price < 10 |	-> 0.54
(1A-2B)	|# of siblings < 1| & |Age ≤ 11| & |10 ≤ Fare price < 51|	-> 0.625
(1A-2C)	|# of siblings < 1| & |Age ≤ 11| & |Fare price ≥ 51|	-> 0.8
(1B-2A)	|# of siblings < 1| & |Age > 11| & |Fare price < 10|	-> 0.275
(1B-2B)	|# of siblings < 1| & |Age > 11| & | 10 ≤ Fare price < 51|	-> 0.36
(1B-2C)	|# of siblings < 1| & |Age > 11| & |Fare price ≥ 51|	-> 0.535
(1C-2A)	|1 ≤ # of siblings ≤ 4| & |Fare price < 10|	-> 0.35
(1C-2B)	|1 ≤ # of siblings ≤ 4| & |10 ≤ Fare price < 51|	-> 0.435
(1C-2D)	|1 ≤ # of siblings ≤ 4| & |Fare price ≥ 51|	-> 0.43
(1D-2A)	|# of siblings > 4| & |Fare price < 10|	-> 0.57
(1D-2B)	|# of siblings > 4| & |10 ≤ Fare price < 51|	-> 0.655
(1D-2D)	|# of siblings > 4| & |Fare price ≥ 51|	-> 0.65

Fig. 4. A merge of the two conjunction sets presented in Fig. 3. The rules are concatenated with an and operation and the end probability is the average of the two leaves' probabilities. The indices of the merged leaves are in Parenthesis at the beginn.ing of the line.

example: Merging 1D with 2D is involved with the exclusion of ($\#ofsiblings \geqslant 1$) as this rule is already included in 1D where ($\#ofsiblings > 4$).

Theoretically, the described procedure can be simply extended to forests of any size in an iterative manner. However, the complexity of this stage makes it impossible to be run with large ensembles, as the runtime grows exponentially with the ensemble size. In order to address this limitation, we present a heuristic, with linear complexity, for filtering a pre-determined number of conjunctions at each iteration. This heuristic enables the execution of this stage on decision forests of any size. its main premise is that despite the existence of numerous combinations of leaves, in practice, there are many combinations that are very unlikely to happen in reality.

3.3.1. Construction and filtering of conjunction sets

In order to include only the relevant conjunctions that have the highest probability to be used when predicting a new instance, we first include all the conjunctions that were assigned to training instances. These conjunctions are extracted by concatenating the rules that were applied to each training instance. The process of generating conjunction for a given training instance is formalized in Algorithm 2. The complexity of this stage is linear with the number of training instances. Following the addition of the conjunctions that were applied to training instances, we generate the final set of conjunctions by conducting the iterative merge and filter process. This process gets as an input the number of maximum conjunctions to be included after each iteration, i.e. – the number of conjunctions out of the generated conjunction set that will continue to be merged in the next iteration. Increasing the value of this hyper-parameter will increase the runtime but will also extract more information from the given pruned ensemble. An additional input is the original training data. The training data is required for calculating the class ratios and the empirical cumulative distribution functions (ECDF) of the different features. The calculated ECDFs enable the later calculation of conjunction probability; namely, the multiplication of the assumed independent probabilities of the feature values. For example: If a conjunction C_i consists of two rules r_1 and r_2 while $P(r_1) = 0.2$ and $P(r_2) = 0.3$ (i.e., 20% and 30% of the training samples comply r_1 and r_2 respectively) then $P(C_i) = P(r_1) * \ P(r_2) = 0.06$. We define a class ratio as the proportion of instances that have the class value. At each iteration in the merge process, if the number of conjunctions that were included in the new set does not exceed the maximum allowed size (N), the algorithm moves to the next iteration. Otherwise, it filters the conjunction set by including only the top $N \times classRatio$ conjunctions in terms of conjunction probability from each of the conjunction subsets that map to the relevant class. For example: if 20% of the training instances have a positive class value and the value of N is 100, then at each iteration only 20 conjunctions that map to the positive class will continue to the next iteration and the rest would be filtered out. The consideration of training set properties, like class probabilities and the pre-inclusion of conjunctions that are known to be useful for training instances, stems from the difference between a dependent and independent tree ensembles. Independent tree ensembles like random forest are induced using many local instances of the data and do not necessarily fit the model to the training set as a whole. On the other hand, dependent trees ensembles like GBDT consider the quality of the model against the training data at each iteration so the resulting rules are more tailored to the training data as a whole rather than to distributed subsets.

Algorithm 2: Generating a conjunction for a given data instance

```
Input $\hat{f}$ (A pruned decision forest composed of $K$ trees), $x$ (a training instance)
Output $C_x$ (A conjunction of rules that were applied by $\hat{f}$ on $x$ mapped into a classification output $g(h_{1x},..h_{Kx})$ where $g$ is the forest aggregation function and $h_{kx}$ is a given leaf output vector)
1 $C_x = \varnothing$
2 initialize $H_x$ to be a list of based-tree output vectors
3 for each base-tree $t_k$ in $\hat{f}$ do
4     Find $C_{kx}$ the conjunction in tree $k$ that should be assigned to $x$
5     $C_x = C_x \wedge C_{kx}$,
6     Add the output vector $h_{kx}$ of $C_{kx}$ corresponding leaf to $H_x$
7 end
```

3.4. Organizing the conjunctions in a tree structure

This section presents the final stage of the developed method. In this stage the extracted conjunctions are organized hierarchically into a tree structure. Theoretically, the set of conjunctions can be used as a stand-alone rule classifier. But it will be neither an efficient nor interpretable model as inferencing a new instance will be involved in iterating over numerous conjunctions and as the model could not be described or visualized to its users. Hence, the objective of this stage is to enable efficient inference and to represent the set of conjunctions as a globally interpretable model.

Similar to broadly used decision trees, the decision nodes of the developed tree test the values of certain attributes while leaves provide classification outputs. However, the induction of the tree substantially differs from traditional tree-inducing approaches. Rather than dividing the training instances, each decision node is induced by finding a decision rule (i.e., a selected attribute) that creates the best separation among the different rule conjunctions that are included in the conjunction set. As a preliminary stage, we count the number of times that each decision rule has been included in the splitting nodes of the pruned ensemble. For each feature, we create an ordered list of values accordingly. The motivation for creating this ordered list comes from the fact that the number of splitting candidates may grow exponentially with the ensemble size and the feature space. Therefore, we leverage the information about important features from the already trained ensemble. This approach stands in contrast to traditional decision trees that use different types of heuristics to find splitting values (mostly for numerical features) when generating a new decision node. In addition to the ordered list of values per feature, the tree generation algorithm receives the set of rule conjunctions and the maximum depth of the resulting tree as inputs. We start building the tree by defining a root node that contains the entire conjunction set which was created at the previous stage. The predicted class of the root node, as well as of any other node, is calculated by aggregating the output vector across the different node conjunctions (e.g., the sum of logits or average probability). Algorithm 3 formulates the process of splitting a node into descending nodes. The entropy of the root node, as well as any other node, is calculated by applying a simple entropy function on each dimension of the class output vector and then apply an average function on these entropies. For example: Assuming a classification task with three classes; if a given node in this task has a set of two conjunctions with the following outputs: $(0.3, 0.6, 0.1), (0.2, 0.6, 0.2)$ then the output of the conjunction set entropy function, in this case, will be:

$$\frac{entropy(0.3, 0.2) + entropy(0.6, 0.6) + entropy(0.1, 0.2)}{3} \tag{3}$$

Splitting a node into its descending nodes is done by selecting the feature that its most frequent value (given in the list of ordered splitting values per feature) result with the highest information gain, calculated as follows:

$$IG(CS_i, R) = \frac{|CS_{i1}| \cdot entropy(CS_{i1}) + |CS_{i2}| \cdot entropy(CS_{i2})}{|CS_{i1}| + |CS_{i2}|} - entropy(CS_i) \tag{4}$$

where CS_i is the set of conjunctions included in the splitted node i, R is the splitting rule, and CS_{i1} and CS_{i2} are the conjunction sets in the new descending nodes. Splitting CS_i into CS_{i1} and CS_{i2} is done in the following manner: Conjunctions that might fit data instances that comply the testing rule are routed to the left node while conjunctions that might fit data instances that contradict the testing rule are routed to the right node. Fig. 5 illustrates the split of the conjunction set presented in Fig. 4 by selecting $Age > 11$ as a splitting feature. We can see that the left node contains all the conjunctions that meet these rules as well as conjunctions that do not necessarily contradict this rule while the right node contains conjunctions that do not meet this rule with conjunctions that do not necessarily meet this rule. The tree generation is stopped in one of the following conditions:

1. There is only one conjunction that has been routed to the given node.
2. Node depth is equal to the maximum allowed depth.

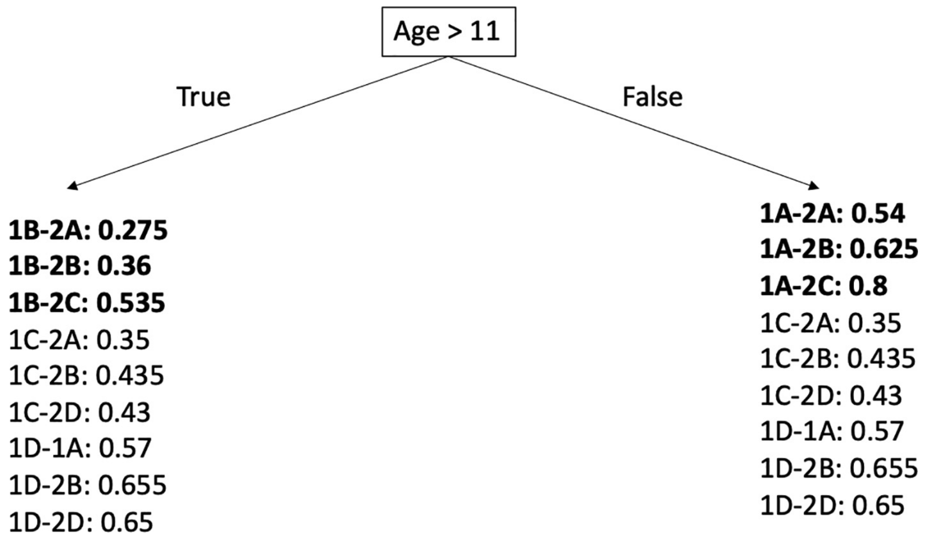


Fig. 5. An example of splitting the root node that contains the conjunction set of Fig. 4. The marked conjunctions are unique to each node while unmarked conjunctions are mutua.l to both sides.

3. The entropy has not been decreased.

Algorithm 3: A recursive splitting algorithm. The recursive call is referred as *split*

```
Input: $CS$ (Set of conjunctions in the corresponding node. The index $i$ that appears in Eq. 3 was removed to improve clarity), $FI$ (Hash-map in which keys are features $fe_j$ and values are ordered lists of splitting values $v_j$), $maxDepth$ (A parameter that determines the maximum depth of the generated tree)
Output: $t$ (A decision tree)
1: if $maxDepth = 0$ or $|CS| = 1$ then
2:     stop
3: end
4: $IG = 0$ information gain
5: $R = \varnothing$ selected splitting rule
6: for $fe_j, v_j$ in $FI$ do
7:     $R_j$ = splitting rule derived from $f_j$. namely, the first item in $v_j$
8:     $CS_(1)$ = all the conjunctions in $CS$ that may classify instances that meet $R$
9:     $CS_(2)$ = all the conjunctions in $CS$ that may classify instances that do not meet $R$
10:    $IG_j = \frac{|CS_1| \cdot entropy(CS_1) + |CS_2| \cdot entropy(CS_2)}{|CS_1| + |CS_2|} - entropy(CS)$
11:    if $IG_j > IG$ then
12:        $IG = IG_j$
13:        $CS_l = CS_1, CS_R = CS_2, R = R_j$
14:    end
15: end
16: remove the selected rule $R$ from its corresponding value set in $FI$
17: $Split(CS_l, FI, maxDepth - 1)$
18: $Split(CS_r, FI, maxDepth - 1)$
```

3.5. Differences from the independent forest-based tree model

The presented method is an evolvement of the previously developed forest-based tree [39]. It presents some novel parts that extend its usability to decision forests consist of dependently induced trees (e.g., XGBoost). This section highlights the extensions that were added in the presented work and discusses the theoretical motivations for these extensions. One adaptation that was added to the presented model is the consideration of the base-trees order in building the conjunction set. While in random forest and other independently-induced trees the order of the base tree does not relate to its importance (i.e., if a certain base tree was trained first or last), it may actually have a significant effect for dependently-induced trees. Connecting to the fact that the generation of the conjunction set is more likely to be biased towards conjunctions that were added at early iterations (as they are less likely to be filtered), the order of the added base trees may have a significant effect on the resulting conjunction set and hence, on the resulting decision tree. In order to address this issue, the presented method considers the order of the base tree in the pruning stage when building the conjunction set.

The proposed method adds an additional substantial refinement to the stage that generates the conjunction set by first including all the conjunctions that were used for classifying training instances prior to adding the conjunctions of the different base trees. This extension can be considered as another measure for reducing the likelihood of missing relevant information when constructing the conjunction set. Another important adaptation presented in the current method is that it is flexible to different types of conjunction outputs rather than being applicable solely for probability vectors. More specifically, instead of just averaging the probability vectors, this method supports the ability to sum the output vectors and then apply a softmax function so the aggregation of conjunction set outputs is aligned with the aggregation mechanism of GBDT. Other types of aggregation functions can be added easily with the generic formulation and structure of the presented method. The extended model also differs from its ancestor as it considers the label distribution when filtering the conjunction set. Whenever the size of the conjunction set exceeds the maximum allowed size, the filtering process ensures that the proportion of conjunctions that map into every class is proportional to the class ratio within the training set. Considering the training class balance when generating the set is motivated by studies that found that XGBoost may require a very careful hyperparameter tuning when applied for tasks with class imbalance and may not work well in such cases [44].

This work also adds several refinements to the final stage of building the tree from the constructed conjunctions. First and foremost, it allows the user to specify the maximum depth of the generated tree. This property enables users to address the trade-off between predictive performance and explainability by affecting the complexity of the trained tree. Finally, the developed method provides an additional refinement to the tree generation stage by changing the stopping criterion for splitting a given node. Instead of stopping when all the conjunction class vectors have the same classification (i.e., the most probable class in all the conjunctions is the same), the tree is continued to be generated towards maximizing the ROC AUC.

3.6. Explainability of the generated model and comparison with related methods

The goal of the developed method is to generate a global explainable model for decision forests and to enable the justification of individual decisions based on the conjunctive rule-set that was applied to a classified instance. The following text is an example of an explanation generated from an XGBoost based tree that was applied on an Iris dataset's instance, aiming at classifying attributes of sepals and petals into one of three Iris species:

$\mathbf{sepalwidth(cm) > 2.95\ AND\ sepallength(cm) < 5.45\ AND\ petallength(cm) < 2.7.}$

$\mathbf{AND\ petalwidth(cm) < 1.15, labels : [setosa : 0.992, versicolor : 0.006, virginica : 0.002]}$.

This conjunction is hierarchically ordered along with other conjunctions to compose the generated decision tree. New instances are assigned to their corresponding conjunctions while their classifications are given by the output vector. In the above example, Setosa will be the predicted class with a probability of 0.992. The model as a whole can also be visualized (as depicted in Fig. 2) for better transparency and for knowledge extraction. The explainability aspects of the presented model differ from other popular and relevant approaches for driving explainability into machine learning models. LIME [34] and shapely values [31] are examples of popular methods for making the outputs of machine-learning models more interpretable. Unlike the presented method, LIME and shapely values are model-agnostic methods that are able to be applied to any type of model. Both models provide explanations as numeric values that indicate the effect of each feature on the final output rather than explainable conjunction of rules. LIME trains a local linear model, based on the given instance, and provides the explanation in terms of the linear connection between each of the features to the decision made. In shapely value, on the other hand, the values are based on practices of coalitional game theory.

Anchors [35] is an example of an approach that generates explanations in the form of conjunctive rules by utilizing reinforcement learning along with graph search algorithms. Collection of High Importance Random Path Snippets (CHIRPS) is yet another method for extracting local conjunction explanations for random forest predictions that [21]. It is done by using frequent pattern mining for finding highly frequent splits within the forest. Both Anchors and CHIRPS strive to search for generative rules rather than generating a globally explainable model that is then used for generating local explanations. In addition, authors of both Anchors and CHIRPS mention the methods as highly dependent in the configuration of hyperparameters in addition to their tendency to yield different results for the same inputs. Methods like GENESIM, ISM and CMM [42,43,16] have the same goal as the method presented in this work; i.e., to generate a single explainable tree from a given decision forest. As mentioned, GENESIM and ISM were found to have a training runtime that is exponential with the feature space and the ensemble size. Therefore, these are not applicable for complex tasks [43,39]. An empirical evaluation that compares the developed method with the above mentioned approaches for tree generation is presented in the empirical evaluation section.

3.7. Implementation implications for different types of decision forests

From a theoretical perspective, the proposed algorithm can work with decision forests of any type to create a single tree. However, there are several points that should be considered when coming to implement the algorithm in certain scenarios.

First, it is important to note that the algorithm should not be run with decision forests that are built upon incomprehensible features (i.e., mathematical transformations of the original features). In these cases, there is usually a preprocessing stage that transforms the original features into other mathematical representations. Running the algorithm with such models will result in a single tree that its rules would mean nothing to the end-user. Rotation forest [36] is one example of this kind of method. In rotation forest, a principal component analysis (PCA) procedure is applied to the features before the training of each base tree. CatBoost [32] is an example of a GBDT model for which the proposed algorithm is not suitable as categorical features are processed to have numerical representations using target statistics. LightGBM is another GBDT model that cannot be fed into the algorithm as exclusive feature bundling is applied to group multiple categorical features. Another factor that might affect the applicability of the model is the inner structure of the transformed decision forest. In XGBoost, for example, each classification tree is virtually composed of *N* binary trees where *N* is the number of classes that the target variable can take. As a result, processing a given tree involved merging the conjunction sets of the *N* trees into a conjunction set that represents a single tree. This procedure is computationally expansive in cases where *N* is a large number. Due to this issue, this paper presents an empirical evaluation only for the case of binary classification challenges and the applicability of this method in multinomial classification tasks can be a subject for future research.

4. Experimental evaluation

Here we present an empirical evaluation of the developed algorithm by testing its ability to convert GBDT into a decision tree without impairing the predictive performance of the original forest. The main objective is to examine GBDT methods as the previous version of this algorithm was already found to be effective for ensembles of independently induced base trees [39]. It also enables us to analyze whether the algorithm performs differently for different ensemble types. The evaluation focuses on binary classification tasks since running the developed algorithm with XGBoost of numerous classes may take a substantial amount of time as described in the previous section. Although it is feasible to run the algorithm in challenges with a few classes, we preferred to focus on a well-defined scope. In addition to testing the effectiveness for GBDT, we also study here for the first time the trade-off between the tree complexity, in terms of maximum depth, and its predictive performance.

We tested different trees, derived by the algorithm from different types of decision forests, and compared their performance against other benchmark classifiers. Two main evaluation criteria were considered: the predictive performance and the complexity of the derived model. The predictive performance was measured using the ROC AUC, motivated by the notion that this measure is invariant to class imbalance and as it is independent to decision thresholds. Despite being less indicative than ROC AUC, we also measured generalized accuracy as this measure is more understandable to the non-expert consumer of the model. Evaluating the complexity of the derived model was done using the average prediction depth. In practical terms, it is the average number of rules that each instance went through during its inference. It is worthwhile mentioning that the training time was not measured as a part of the evaluation as it highly dependent on external factors such as code efficiency, computing resource utilization, etc.

4.1. Implementation and experiment settings

The experiment and the implementation of the algorithm were written in Python. The entire code is publicly available for usage and reproduction at the following link: https://github.com/sagyome/XGBoostTreeApproximator. The provided software package enables running the algorithm on any decision forest that implements the API of Sklearn's ensemble class. In addition, it includes an implementation of the algorithm for XGBoost [9]. It should be mentioned here that the implementation of the algorithm for XGBoost required a preliminary stage of transforming the textual representation of each of the base trees into a designated tree object – similar to the tree object that is implemented in SKlearn. In future extensions of this implementation or in other implementations, it might be useful to test different ways to extract the underlying base trees. In the current experiment, we fitted the forest-based tree by using three types of decision forests as inputs: XGBoost based tree (XGBBT) was fitted for XGBoost (XGB), Random forest-based tree (RFBT) was fitted for SKlearn's random forest (RF), and gradient boosting machine-based tree (GBMBT) was fitted for SKlearn's gradient boosting machines (GBM). We conducted an exhaustive grid search with 10-fold cross-validation to find the most suitable hyper-parameters for each of the decision forests. When training the FBT models we defined fixed values for the following parameters: The minimum size of the pruned forest was set to be 10 and the maximum number of conjunctions per iteration was set to be 1000. We defined three different levels as the maximum depth of the generated tree – four, eight, and twenty. In the evaluation, we present the option with the highest ROC AUC.

SKlearn's decision tree (DT) was trained to serve as a benchmark to the forest-based trees and similar to the decision forests, it was also done following a grid-search of the most suitable hyper-parameters. We also trained additional benchmarks of decision trees that were derived from a decision forest. One example is CMM [16], that was trained by using the implementation that was published by the authors of [50]. More specifically, we generated unlabelled synthetic data that is ten times larger than the training data, using a multivariate gaussian distribution. We then labeled the synthetic data with an XGBoost and trained a decision tree following a grid search of hyper-parameters. We also trained GENESIM and ISM as additional benchmarks of decision trees that were derived from decision forests. These models were trained using the opensource code that was provided by the authors of GENESIM. As GENESIM and ISM are not scalable for problems with many features or many instances, there were several datasets for which these algorithms did not manage to converge. We filled the result tables with NA values in cells that are corresponding to these datasets. In addition, the statistical tests for testing the performance of the different models across the different datasets do not consider these two models. The above-mentioned evaluation criteria were compared using 20 binary classification tasks from the UCI repository. Table 1 presents the properties of the datasets that were included in the experiment It is important to note that since training time is dependent on code optimization, it was not measured and compared across the different methods. For each of the 20 datasets, we randomly divided the dataset into train and test sets in an 80/20 ratio respectively. The pruning of each of the forest models was conducted by using the training data as the pruning set. We measured the evaluation criteria (ROC AUC, generalized accuracy, and prediction depth) based on the test set predictions. The procedure was repeated 10 times for each of the datasets.

4.2. Results

Table 2 and Table 3 summarize the average ROC AUC of the tested models per dataset. Table 2 shows these results for the different ensemble models and the forest-based tree models while Table 3 shows the results for the different benchmarks. Each cell contains the average and the standard deviation values of the related model for the given dataset. We mark with bold cells of models that their metric value for the corresponding dataset was superior to all the other models or that were not significantly inferior to the best model (in terms of metric value). The significance was measured using a paired t-test with $p\_value \geqslant 0.05$ (i.e., if the p_value of the paired test is larger than 0.05 then the model is considered equivalent to the best model). Table 8 presents a comparison of the ROC AUC across different levels of maximum tree depth. There are three levels for each forest-based tree (4, 8, 20) and the comparison of these levels are presented for the random-forest-based tree (RFBT) and for XGBoost-based tree (XGBBT). Generalized accuracy is an additional metric that was measured in the experiment, summarized in Table 4 for decision forests and their derived trees, and in Table 5 for the other benchmarks. Table 6 and Table 7 summarize the prediction complexity in terms of the number of rules that were applied to test instances. We follow the procedure described in [14] towards comparing the performance of the different models across the different datasets where ISM and GENESIM were not included since they have not been converged when trained for some of the datasets. As a first stage, we conducted an adjusted Friedman test for each evaluation criterion. The test rejected the null hypothesis that all models performed the same in terms of AUC ($F_f = 95.6, p \approx 0$), generalized accuracy ($F_f = 33.33, p \approx 0$) and average prediction depth test instance ($F_f = 127.8, p \approx 0$). We then conducted a post hoc Nemenyi test of the AUC, the generalized accuracy, and the average prediction depth for the different models, depicted in Table 9, Tables 10 and 11 respectively.

According to the gained results, in 5 out of the 20 datasets, at least one of the forest-based trees was not significantly less accurate than a decision forest, in terms of ROC AUC. Furthermore, the results presented in Table 9 suggests that the XGBoost based tree (XGBBT) does not significantly less accurate than gradient boosting machines ($p \approx 0.16$) and significantly more accurate than a CART decision tree that was trained following an exhaustive grid-search of hyper-parameters ($p \approx 0.078$). The average depth of all of the forest-based trees was significantly smaller compared to the average depth of all of the decision forests as each instance went through no more than 20 rules rather than hundreds of rules during its classification. Therefore, the forest-based tree yields a Pareto improvement for the five datasets that were mentioned above.

Table 1
Datasets characteristics. # catg – Number of categorical features, # Real – Number of continuous features.

Dataset	# Catg	# Real	# Instances	Majority label proportion
Tic_tac_toe	9	0	958	0.64
Credit	9	6	666	0.55
Kohkiloyeh	5	0	100	0.66
Banknote	0	4	1372	0.55
Breast_cancer	0	30	569	0.64
Occupancy	0	5	2665	0.64
Pima	0	8	768	0.65
Mamographic	0	5	830	0.51
Internet	4	0	322	0.7
Thyroid	23	6	2791	0.95
Seismic	4	14	2584	0.94
Liver	1	9	579	0.72
Ctherapy	0	6	90	0.6
Aust	0	14	690	0.56
Spambase	0	57	4601	0.6
German	13	7	1000	0.7
Ionosphere	0	34	351	0.65
Divorce	0	54	170	0.52
Haberman	0	3	306	0.72
Biodeg	0	41	1055	0.66

Table 2
AUC comparison (1).

Dataset	XGB	RF	GBM	XGBBT	RFBT
Aust	92.91 ± 2.1	92.37 ± 1.89	$\mathbf{93.15} \pm \mathbf{2.21}$	92.02 ± 3.07	90.77 ± 2.91
Banknote	$\mathbf{100.0} \pm \mathbf{0.0}$	99.98 ± 0.01	99.99 ± 0.02	99.94 ± 0.1	99.56 ± 0.3
Biodeg	$\mathbf{94.37} \pm \mathbf{1.74}$	$\mathbf{94.36} \pm \mathbf{1.65}$	94.2 ± 1.79	88.91 ± 3.18	88.78 ± 2.84
Breast	$\mathbf{99.21} \pm \mathbf{0.39}$	98.97 ± 0.61	$\mathbf{99.27} \pm \mathbf{0.4}$	97.9 ± 0.5	98.31 ± 1.58
Credit	$\mathbf{94.0} \pm \mathbf{1.56}$	$\mathbf{93.74} \pm \mathbf{1.3}$	$\mathbf{94.47} \pm \mathbf{1.03}$	92.34 ± 2.68	91.61 ± 1.8
Ctherapy	96.11 ± 2.82	97.35 ± 2.33	$\mathbf{97.96} \pm \mathbf{2.96}$	$\mathbf{98.02} \pm \mathbf{2.71}$	95.31 ± 7.27
Divorce	$\mathbf{99.79} \pm \mathbf{0.32}$	$\mathbf{99.76} \pm \mathbf{0.31}$	$\mathbf{99.78} \pm \mathbf{0.28}$	$\mathbf{99.78} \pm \mathbf{0.23}$	97.91 ± 1.76
German	$\mathbf{84.48} \pm \mathbf{1.75}$	$\mathbf{85.08} \pm \mathbf{2.08}$	$\mathbf{84.91} \pm \mathbf{1.95}$	80.49 ± 1.66	78.65 ± 3.66
Haberman	$\mathbf{80.97} \pm \mathbf{7.02}$	$\mathbf{81.11} \pm \mathbf{6.25}$	$\mathbf{80.62} \pm \mathbf{6.2}$	80.45 ± 6.18	$\mathbf{80.8} \pm \mathbf{6.05}$
Internet	$\mathbf{100.0} \pm \mathbf{0.0}$	$\mathbf{100.0} \pm \mathbf{0.0}$	99.98 ± 0.04	$\mathbf{100.0} \pm \mathbf{0.0}$	$\mathbf{100.0} \pm \mathbf{0.0}$
Ionosphere	97.28 ± 1.16	$\mathbf{97.92} \pm \mathbf{1.07}$	97.2 ± 1.31	93.58 ± 1.12	93.82 ± 1.98
Kohkiloyeh	81.25 ± 9.72	$\mathbf{87.02} \pm \mathbf{5.41}$	81.83 ± 9.67	80.42 ± 9.36	$\mathbf{84.95} \pm \mathbf{6.6}$
Liver	81.2 ± 3.83	$\mathbf{83.57} \pm \mathbf{3.23}$	$\mathbf{82.73} \pm \mathbf{2.61}$	78.36 ± 5.06	81.86 ± 3.44
Mamographic	$\mathbf{89.65} \pm \mathbf{1.72}$	$\mathbf{89.79} \pm \mathbf{1.67}$	88.14 ± 2.06	88.56 ± 2.19	89.39 ± 1.6
Occupancy	$\mathbf{99.85} \pm \mathbf{0.07}$	$\mathbf{99.91} \pm \mathbf{0.1}$	$\mathbf{99.88} \pm \mathbf{0.07}$	99.76 ± 0.13	99.56 ± 0.32
Pima	$\mathbf{85.27} \pm \mathbf{1.83}$	84.37 ± 2.3	84.92 ± 2.15	83.73 ± 1.57	83.07 ± 1.79
Seismic	$\mathbf{96.45} \pm \mathbf{0.6}$	$\mathbf{96.25} \pm \mathbf{0.4}$	$\mathbf{96.27} \pm \mathbf{0.65}$	95.9 ± 0.37	95.72 ± 0.56
Spambase	$\mathbf{99.0} \pm \mathbf{0.28}$	98.91 ± 0.27	98.84 ± 0.35	96.04 ± 0.44	91.98 ± 1.26
Thyroid	$\mathbf{99.82} \pm \mathbf{0.06}$	$\mathbf{99.81} \pm \mathbf{0.06}$	$\mathbf{99.78} \pm \mathbf{0.08}$	99.64 ± 0.18	99.55 ± 0.17
Tic-tac-toe	$\mathbf{99.95} \pm \mathbf{0.07}$	99.84 ± 0.14	$\mathbf{99.95} \pm \mathbf{0.05}$	96.35 ± 2.48	97.1 ± 0.61
Average	93.58 ± 1.85	94.01 ± 1.55	93.69 ± 1.79	92.11 ± 2.16	91.93 ± 2.33

Comparison of average ROC AUC per dataset. Each cell contains the average and standard deviation of the different runs.
Marked with bold – The best score and its equivalent scores ($p\_value \geqslant 0.05$).

Table 3
AUC comparison (2).

Dataset	GBMBT	DT	CMM	GENESIM	ISM
Aust	91.11 ± 2.46	90.81 ± 3.24	85.81 ± 2.95	89.33 ± 3.37	89.54 ± 3.58
Banknote	99.94 ± 0.07	98.24 ± 1.3	99.49 ± 0.33	99.16 ± 0.74	98.68 ± 0.5
Biodeg	88.36 ± 4.36	85.78 ± 2.56	86.03 ± 2.58	*NA*	*NA*
Breast	98.76 ± 1.08	96.22 ± 3.62	96.96 ± 1.66	96.44 ± 1.72	95.32 ± 1.53
Credit	91.92 ± 2.73	90.78 ± 3.66	91.74 ± 1.82	78.53 ± 5.19	78.6 ± 4.52
Ctherapy	95.68 ± 5.97	93.27 ± 8.9	88.64 ± 7.51	90.92 ± 4.55	83.74 ± 6.35
Divorce	99.36 ± 0.58	97.06 ± 2.08	97.34 ± 1.52	98.68 ± 0.6	97.35 ± 1.89
German	77.2 ± 5.03	76.99 ± 3.32	78.45 ± 2.52	76.31 ± 2.39	75.92 ± 3.65
Haberman	71.14 ± 5.45	80.03 ± 6.33	72.62 ± 9.31	77.26 ± 4.93	74.57 ± 8.01
Internet	$\mathbf{100.0} \pm \mathbf{0.0}$	99.38 ± 1.4	$\mathbf{100.0} \pm \mathbf{0.0}$	$\mathbf{100.0} \pm \mathbf{0.0}$	94.86 ± 2.03
Ionosphere	94.58 ± 1.78	92.37 ± 2.58	91.16 ± 6.54	*NA*	*NA*
Kohkiloyeh	81.05 ± 8.67	78.45 ± 5.84	74.28 ± 7.41	71.71 ± 11.04	79.82 ± 8.04
Liver	75.26 ± 3.8	79.05 ± 4.43	74.88 ± 6.76	74.49 ± 1.08	72.37 ± 3.61
Mamographic	86.73 ± 3.83	87.11 ± 2.3	82.17 ± 3.03	86.62 ± 2.45	85.2 ± 2.02
Occupancy	99.77 ± 0.09	99.11 ± 0.69	98.55 ± 0.64	99.38 ± 0.2	99.24 ± 0.19
Pima	80.51 ± 4.26	83.81 ± 2.51	78.69 ± 3.05	77.1 ± 2.48	73.14 ± 3.43
Seismic	92.47 ± 1.08	$\mathbf{96.17} \pm \mathbf{0.69}$	76.2 ± 23.93	95.99 ± 0.37	93.39 ± 0.62
Spambase	96.46 ± 0.45	93.89 ± 2.26	89.46 ± 3.81	*NA*	*NA*
Thyroid	99.22 ± 0.44	98.98 ± 0.51	98.07 ± 0.59	98.98 ± 0.71	95.47 ± 0.84
Tic-tac-toe	96.99 ± 1.59	95.99 ± 1.09	97.3 ± 1.68	97.46 ± 1.2	96.49 ± 1.63
Average	90.83 ± 2.69	90.67 ± 2.96	87.89 ± 4.38	88.73 ± 2.53	87.28 ± 3.08

Comparison of average ROC AUC per dataset. Each cell contains the average and standard deviation of the different runs.
Marked with bold – The best score and its equivalent scores ($p\_value \geqslant 0.05$).

4.3. Results discussion

The obtained results suggest that both random forest-based tree and XGBoost based tree significantly outperform other single tree models. Fig. 6 illustrates this finding by visualizing the results of a Wilcoxon–Holm post hoc test. On the other hand, GBM based tree did not significantly outperform the CART decision tree (DT) but did outperform all of the other models that generate tree from a pre-trained decision forest. In general, the ROC AUC of these benchmarks (i.e. – CMM, ISM and GENESIM) has failed behind all of the other models, including the CART decision tree. Nevertheless, the generalized accuracy of CMM was not significantly inferior to all of the other forest-derived tree, as depicted in Fig. 7. The reason for the gap between the AUC and the generalized accuracy is the instability of CMM's data synthesis processes that tends to over-generate instances of the majority class, resulting in excessive label imbalance as already shown in previous evaluations.

Table 4
Generalized accuracy comparison (1).

Dataset	XGB	RF	GBM	XGBBT	RFBT
Aust	85.22 ± 1.1	$\mathbf{86.81 \pm 3.18}$	$\mathbf{86.52 \pm 1.75}$	85.65 ± 2.26	84.49 ± 2.69
Banknote	$\mathbf{99.78 \pm 0.33}$	99.34 ± 0.6	$\mathbf{99.78 \pm 0.33}$	99.42 ± 0.42	99.12 ± 0.55
Biodeg	87.2 ± 2.25	$\mathbf{87.49 \pm 1.52}$	$\mathbf{87.87 \pm 2.38}$	82.56 ± 3.44	84.45 ± 3.08
Breast	$\mathbf{97.37 \pm 1.89}$	96.05 ± 0.95	$\mathbf{97.72 \pm 1.11}$	95.44 ± 2.51	$\mathbf{97.11 \pm 4.38}$
Credit	$\mathbf{87.67 \pm 3.3}$	$\mathbf{87.22 \pm 1.5}$	86.47 ± 2.19	85.11 ± 3.38	$\mathbf{87.82 \pm 2.63}$
Ctherapy	90.0 ± 7.24	88.89 ± 8.78	$\mathbf{94.44 \pm 7.86}$	$\mathbf{93.33 \pm 9.13}$	$\mathbf{93.33 \pm 7.24}$
Divorce	$\mathbf{97.65 \pm 1.32}$	$\mathbf{97.65 \pm 1.32}$	97.06 ± 2.08	$\mathbf{97.65 \pm 1.32}$	97.06 ± 0.0
German	75.9 ± 1.47	$\mathbf{77.0 \pm 2.18}$	$\mathbf{77.4 \pm 2.33}$	74.7 ± 2.54	71.9 ± 3.34
Haberman	$\mathbf{74.26 \pm 5.94}$	$\mathbf{75.08 \pm 5.77}$	72.95 ± 5.42	73.61 ± 6.97	$\mathbf{75.08 \pm 4.49}$
Internet	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.06 ± 1.4	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
Ionosphere	$\mathbf{93.43 \pm 1.92}$	$\mathbf{94.29 \pm 2.86}$	$\mathbf{94.0 \pm 2.93}$	88.86 ± 1.86	90.86 ± 1.63
Kohkiloyeh	74.5 ± 7.25	$\mathbf{78.0 \pm 7.15}$	$\mathbf{76.5 \pm 6.69}$	$\mathbf{76.0 \pm 6.58}$	$\mathbf{77.5 \pm 5.4}$
Liver	71.21 ± 5.19	$\mathbf{72.59 \pm 3.07}$	72.41 ± 4.99	70.69 ± 8.22	$\mathbf{73.97 \pm 2.68}$
Mamographic	$\mathbf{82.17 \pm 2.65}$	$\mathbf{82.89 \pm 2.03}$	81.81 ± 2.1	$\mathbf{82.53 \pm 1.13}$	$\mathbf{82.77 \pm 1.84}$
Occupancy	98.57 ± 0.31	$\mathbf{98.87 \pm 0.42}$	98.5 ± 0.55	98.54 ± 0.5	98.46 ± 0.16
Pima	75.19 ± 2.88	$\mathbf{76.49 \pm 2.36}$	74.68 ± 1.78	75.06 ± 2.66	74.81 ± 2.31
Seismic	93.08 ± 1.01	93.04 ± 1.0	93.0 ± 0.88	92.96 ± 0.91	93.15 ± 1.1
Spambase	$\mathbf{95.42 \pm 0.94}$	$\mathbf{95.76 \pm 0.55}$	$\mathbf{95.13 \pm 0.96}$	91.43 ± 0.58	87.43 ± 2.32
Thyroid	$\mathbf{97.53 \pm 0.51}$	97.22 ± 0.67	$\mathbf{97.48 \pm 0.67}$	97.17 ± 0.61	$\mathbf{97.51 \pm 0.56}$
Tic-tac-toe	$\mathbf{99.06 \pm 0.77}$	98.23 ± 1.31	98.96 ± 0.82	94.06 ± 2.54	94.38 ± 1.13
Average	88.76 ± 2.41	89.15 ± 2.36	89.09 ± 2.46	87.74 ± 2.88	88.06 ± 2.38

Comparison of average generalized accuracy per dataset. Each cell contains the average and standard deviation of the different runs.
Marked with bold – The best score and its equivalent scores ($p\_value \geqslant 0.05$).

Table 5
Generalized accuracy comparison (2).

Dataset	GBMBT	DT	CMM	GENESIM	ISM
Aust	83.62 ± 3.01	83.48 ± 2.14	84.49 ± 0.97	84.01 ± 2.34	85.41 ± 2.8
Banknote	99.42 ± 0.55	97.96 ± 1.17	99.49 ± 0.33	98.5 ± 0.83	97.52 ± 0.58
Biodeg	82.18 ± 3.43	83.51 ± 2.63	81.04 ± 3.4	*NA*	*NA*
Breast	97.19 ± 2.77	90.61 ± 1.31	96.05 ± 2.46	94.45 ± 1.93	93.85 ± 1.39
Credit	87.07 ± 3.42	86.62 ± 4.03	$\mathbf{87.82 \pm 2.68}$	77.1 ± 2.85	76.79 ± 3.39
Ctherapy	91.11 ± 7.45	$\mathbf{93.33 \pm 7.24}$	86.67 ± 8.43	86.67 ± 4.22	84.44 ± 4.83
Divorce	96.47 ± 3.22	97.06 ± 2.08	97.06 ± 0.0	$\mathbf{97.65 \pm 0.88}$	96.08 ± 1.96
German	71.5 ± 3.06	68.0 ± 3.32	74.3 ± 2.73	68.47 ± 2.56	71.2 ± 2.41
Haberman	66.72 ± 5.24	73.61 ± 5.65	$\mathbf{74.59 \pm 5.02}$	73.48 ± 3.63	71.41 ± 5.15
Internet	$\mathbf{100.0 \pm 0.0}$	99.38 ± 1.4	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	91.75 ± 0.73
Ionosphere	90.57 ± 2.17	87.43 ± 1.2	88.0 ± 6.28	*NA*	*NA*
Kohkiloyeh	$\mathbf{78.0 \pm 6.75}$	74.5 ± 7.25	72.5 ± 7.17	68.67 ± 9.32	$\mathbf{76.33 \pm 7.61}$
Liver	70.17 ± 6.32	70.69 ± 2.92	70.52 ± 4.97	68.05 ± 2.1	66.67 ± 4.53
Mamographic	81.93 ± 3.04	$\mathbf{82.77 \pm 2.39}$	$\mathbf{82.17 \pm 3.03}$	81.04 ± 2.45	81.61 ± 2.62
Occupancy	98.65 ± 0.36	98.39 ± 0.56	98.09 ± 0.52	98.28 ± 0.38	98.38 ± 0.41
Pima	73.25 ± 3.6	76.23 ± 3.51	$\mathbf{76.62 \pm 2.72}$	70.87 ± 2.36	72.09 ± 2.07
Seismic	90.64 ± 4.49	93.0 ± 1.12	93.11 ± 1.08	$\mathbf{93.32 \pm 0.54}$	$\mathbf{93.39 \pm 0.62}$
Spambase	91.89 ± 0.77	91.49 ± 0.9	83.1 ± 4.92	*NA*	*NA*
Thyroid	97.24 ± 0.73	97.35 ± 0.55	97.03 ± 0.54	97.07 ± 1.08	95.47 ± 0.84
Tic-tac-toe	94.38 ± 1.93	93.85 ± 1.66	94.27 ± 3.79	91.08 ± 1.43	94.01 ± 2.94
Average	87.1 ± 3.12	86.96 ± 2.65	86.85 ± 3.05	85.22 ± 2.29	85.08 ± 2.64

Comparison of average generalized accuracy per dataset. Each cell contains the average and standard deviation of the different runs.
Marked with bold – The best score and its equivalent scores ($p\_value \geqslant 0.05$).

Table 9 and Fig. 6 also show that there is no significant gap among the predictive performance of the different decision forests (i.e. – XGB, RF, and GBM), although XGBoost is ranked higher than the rest of the models.

Another important finding that emerged from the results is that the ability of the model to approximate the performance of the given decision forest may vary across forest types. Fig. 9 presents the distribution of the ROC AUC ratio between the generated tree and its source forest across the different datasets. It implies that the approximation of random forest is usually more consistent while GBM is more likely to be poorly approximated by the algorithm in some cases. In addition, it shows that in some cases, the approximation of XGBoost yields higher ROC AUC, compared to the source model. In order to analyze the approximation error, we measured the fidelity [19] between XGB and XGBBT and compared it to the fidelity between XGB and DT. The fidelity is calculated as the percentage of matched classifications of XGB in respect to XGBBT and DT. Fig. 10 illustrates the gap between the distribution of average fidelities across the different datasets. In Fig. 10 (a) we can see that there is a significant difference between DT and XGBBT in terms of their ability to approximate XGB. A paired t-test

Table 6
Average depth comparison (1).

Dataset	XGB	RF	GBM	XGBBT	RFBT
Aust	498.96 ± 282.43	391.27 ± 154.73	417.89 ± 164.44	12.69 ± 1.48	15.41 ± 0.85
Banknote	371.33 ± 80.13	298.61 ± 119.1	594.52 ± 6.44	11.12 ± 0.81	12.19 ± 0.64
Biodeg	485.41 ± 128.14	422.42 ± 159.2	477.3 ± 162.93	15.58 ± 1.47	17.37 ± 1.02
Breast	442.52 ± 230.65	343.53 ± 62.2	293.25 ± 2.09	13.68 ± 0.73	13.69 ± 0.68
Credit	296.32 ± 140.72	561.43 ± 164.73	354.09 ± 137.49	13.04 ± 0.95	14.66 ± 2.1
Ctherapy	207.86 ± 66.01	260.56 ± 90.12	461.02 ± 151.71	10.3 ± 0.88	11.19 ± 0.62
Divorce	134.81 ± 7.17	90.42 ± 3.53	263.99 ± 42.17	11.05 ± 0.83	10.85 ± 1.02
German	286.32 ± 146.48	786.77 ± 273.61	473.45 ± 159.23	14.98 ± 1.29	16.74 ± 1.38
Haberman	168.9 ± 96.03	228.04 ± 156.18	337.94 ± 128.16	9.13 ± 1.53	8.66 ± 4.57
Internet	305.3 ± 198.09	183.9 ± 8.95	286.05 ± 6.96	10.77 ± 1.46	12.06 ± 0.32
Ionosphere	389.83 ± 176.56	383.98 ± 149.51	410.27 ± 157.86	13.15 ± 1.06	14.4 ± 1.61
Kohkiloyeh	603.8 ± 341.75	316.88 ± 109.17	430.79 ± 155.66	10.63 ± 0.91	12.2 ± 0.68
Liver	568.8 ± 464.07	399.83 ± 265.52	353.92 ± 137.45	13.43 ± 0.83	16.07 ± 3.36
Mamographic	214.15 ± 79.51	404.39 ± 100.7	348.25 ± 126.48	10.17 ± 0.72	10.64 ± 0.35
Occupancy	601.74 ± 378.65	274.23 ± 88.65	452.62 ± 178.15	11.34 ± 0.87	12.75 ± 0.45
Pima	226.49 ± 74.55	378.72 ± 92.27	356.84 ± 135.81	12.41 ± 0.44	14.59 ± 1.61
Seismic	251.38 ± 173.2	438.87 ± 146.56	419.04 ± 163.03	13.91 ± 2.13	15.15 ± 3.81
Spambase	1026.15 ± 448.31	833.01 ± 297.37	599.3 ± 0.94	18.2 ± 0.73	18.74 ± 0.54
Thyroid	317.51 ± 123.14	552.65 ± 132.14	535.99 ± 124.45	14.61 ± 0.84	15.75 ± 0.98
Tic	648.55 ± 67.36	700.13 ± 16.29	600.0 ± 0.0	11.44 ± 0.44	10.79 ± 0.11
Average	402.31 ± 185.15	412.48 ± 129.53	423.33 ± 107.07	12.58 ± 1.02	13.7 ± 1.34

Comparison of average depth per dataset. Each cell stores the average and standard deviation of the average number of rules that were applied to a test instance.

Table 7
Average depth comparison (2).

Dataset	GBMBT	DT	CMM	GENESIM	ISM
Aust	12.88 ± 0.68	3.62 ± 0.85	4.47 ± 1.15	4.5 ± 1.64	**2.59** ± **0.76**
Banknote	11.41 ± 0.83	4.15 ± 0.51	5.49 ± 0.61	**3.5** ± **0.18**	7.03 ± 0.69
Biodeg	15.13 ± 1.43	**6.31** ± **1.18**	7.23 ± 0.73	*NA*	*NA*
Breast	14.22 ± 0.68	**2.99** ± **0.68**	5.16 ± 0.66	**2.86** ± **0.7**	3.09 ± 0.53
Credit	13.77 ± 0.61	4.42 ± 1.01	3.71 ± 0.65	**2.78** ± **2.7**	3.39 ± 0.85
Ctherapy	10.4 ± 2.06	2.52 ± 0.36	3.64 ± 1.01	2.26 ± 0.41	**1.99** ± **0.75**
Divorce	9.93 ± 2.33	1.62 ± 0.22	3.63 ± 0.74	**1.0** ± **0.0**	**1.0** ± **0.0**
German	14.52 ± 0.83	**3.71** ± **0.98**	7.74 ± 1.07	5.87 ± 1.51	4.32 ± 1.5
Haberman	10.93 ± 0.97	3.68 ± 0.85	**2.59** ± **1.28**	3.57 ± 1.22	**2.61** ± **1.0**
Internet	7.38 ± 0.32	3.09 ± 0.08	4.5 ± 0.35	3.07 ± 0.1	**2.23** ± **0.3**
Ionosphere	14.1 ± 1.93	**3.63** ± **1.1**	6.62 ± 1.07	*NA*	*NA*
Kohkiloyeh	11.9 ± 0.6	4.77 ± 0.83	4.74 ± 0.61	**2.94** ± **1.47**	3.86 ± 0.95
Liver	14.79 ± 1.87	**2.96** ± **0.07**	5.6 ± 0.9	4.76 ± 2.49	3.31 ± 0.92
Mamographic	11.5 ± 0.41	3.36 ± 0.81	3.67 ± 0.66	4.82 ± 1.36	**3.03** ± **0.93**
Occupancy	10.88 ± 1.22	**2.4** ± **0.41**	3.34 ± 1.21	**2.22** ± **0.33**	2.54 ± 0.56
Pima	13.2 ± 1.01	**4.16** ± **1.09**	5.76 ± 0.27	4.83 ± 1.26	5.36 ± 1.29
Seismic	15.45 ± 4.7	3.0 ± 0.0	**1.64** ± **1.84**	**1.31** ± **0.69**	0.0 ± 0.0
Spambase	18.0 ± 0.6	8.53 ± 0.63	9.16 ± 0.41	*NA*	*NA*
Thyroid	15.7 ± 1.65	3.09 ± 0.36	3.39 ± 0.81	2.9 ± 0.87	**1.0** ± **0.0**
Tic	11.62 ± 0.48	4.89 ± 0.07	8.09 ± 0.62	**4.63** ± **0.14**	5.53 ± 0.29
Average	12.89 ± 1.26	3.85 ± 0.6	5.01 ± 0.83	3.4 ± 1.0	**3.05** ± **0.67**

Compariosn of average depth per dataset. Each cell stores the average and standard devistion of the average number of rules that were applied to a test instance.
Marked with bold – The best score and its equivalent scores ($p\_value \geqslant 0.05$).

conducted between these two fidelity vectors showed that this difference is significant (p-value < 0.08). Fig. 10 (b) implies that there is a slight difference between the fidelities of XGB with XGBBT and RF and RFBT. However, this difference is not significant according to a paired t-test (p-value > 0.15).

We also analyzed the effect of model complexity on the approximation level. Fig. 11 (a) shows that there is a negative correlation between the number of unique conjunctions that are applied by the decision forest (for training data) to the level of approximation obtained by the generated tree. GBM and XGBoost are more likely to yield a larger number of unique conjunctions, as depicted in Fig. 11 and hence, the algorithm is more likely to omit useful conjunctions when constructing the conjunction set. This finding stresses the necessity of refining and further developing better approaches for excluding irrelevant conjunctions without impairing the quality of the approximation. Regarding the tree complexity, the results imply a significant difference between the random forest-based tree and the XGBoost based tree. Fig. 8 visualizes the results of the

Table 8
ROC AUC as a function of the maximum depth of the trained tree.

Dataset	XGBBT4	XGBBT8	XGBBT20	RFBT4	RFBT8	RFBT20
Aust	91.69 ± 2.23	92.02 ± 3.07	91.6 ± 3.4	90.77 ± 2.91	88.63 ± 1.54	88.68 ± 2.61
Banknote	97.66 ± 2.65	99.75 ± 0.45	99.94 ± 0.1	98.07 ± 1.99	99.41 ± 0.44	99.56 ± 0.3
Biodeg	84.97 ± 1.55	88.21 ± 2.64	88.91 ± 3.18	80.3 ± 5.29	84.72 ± 5.43	88.78 ± 2.84
Breast	96.64 ± 3.75	97.02 ± 0.99	97.9 ± 0.5	97.21 ± 2.72	97.99 ± 2.22	98.31 ± 1.58
Credit	92.03 ± 2.09	92.34 ± 2.68	91.64 ± 3.75	90.98 ± 3.6	91.12 ± 2.56	91.61 ± 1.8
Ctherapy	89.69 ± 10.18	96.05 ± 6.11	98.02 ± 2.71	93.7 ± 8.68	95.31 ± 7.27	93.83 ± 4.79
Divorce	99.78 ± 0.23	99.62 ± 0.62	99.62 ± 0.71	96.02 ± 2.26	96.89 ± 2.63	97.91 ± 1.76
German	78.24 ± 2.54	80.49 ± 1.66	78.87 ± 3.77	77.54 ± 3.29	78.65 ± 3.66	77.77 ± 3.18
Haberman	80.45 ± 6.18	79.54 ± 8.22	79.49 ± 8.39	78.95 ± 4.54	79.95 ± 6.25	80.8 ± 6.05
Internet	99.2 ± 0.38	99.92 ± 0.17	100.0 ± 0.0	96.64 ± 3.29	99.84 ± 0.35	100.0 ± 0.0
Ionosphere	92.24 ± 3.79	92.59 ± 3.0	93.58 ± 1.12	92.31 ± 3.23	91.62 ± 2.55	93.82 ± 1.98
Kohkiloyeh	73.95 ± 13.01	78.38 ± 12.94	80.42 ± 9.36	74.15 ± 8.85	84.75 ± 5.92	84.95 ± 6.6
Liver	78.36 ± 5.06	76.95 ± 5.2	75.37 ± 6.62	79.09 ± 2.76	81.39 ± 4.69	81.86 ± 3.44
Mamographic	86.55 ± 2.02	88.56 ± 2.19	88.54 ± 1.14	87.95 ± 2.94	89.01 ± 1.85	89.39 ± 1.6
Occupancy	99.4 ± 0.31	99.76 ± 0.13	99.74 ± 0.15	99.28 ± 0.39	99.51 ± 0.19	99.56 ± 0.32
Pima	82.54 ± 2.36	83.7 ± 2.22	83.73 ± 1.57	79.5 ± 3.67	80.42 ± 2.15	83.07 ± 1.79
Seismic	95.54 ± 0.55	95.57 ± 0.83	95.9 ± 0.37	95.05 ± 1.0	95.48 ± 0.53	95.72 ± 0.56
Spambase	88.07 ± 2.32	92.42 ± 2.02	96.04 ± 0.44	88.21 ± 3.09	89.57 ± 3.52	91.98 ± 1.26
Thyroid	99.34 ± 0.36	99.51 ± 0.29	99.64 ± 0.18	98.96 ± 0.89	99.55 ± 0.17	99.26 ± 0.35
Tic	80.36 ± 2.89	95.98 ± 3.63	96.35 ± 2.48	84.22 ± 6.19	96.09 ± 1.55	97.1 ± 0.61
Average	89.34 ± 3.22	91.42 ± 2.95	91.77 ± 2.5	88.94 ± 3.58	91.0 ± 2.77	91.7 ± 2.17

A comparison between the different levels of maximum tree depth (there are 3 levels: 4, 8 and 20). The comparison was conducted for XGBoost based tree (XGBBT) and for random forest-based tree (RFBT).

Table 9
Nemenyi test for AUC comparison.

	XGB	RF	GBM	XGBBT	RFBT	GBMBT	DT
RF	1.000	–	–	–	–	–	–
GBM	0.999	1	–	–	–	–	–
XGBBT	0.034	0.05	0.162	–	–	–	–
RFBT	0.004	0.006	0.027	0.998	–	–	–
GBMBT	0.000	0.001	0.004	0.934	0.999	–	–
DT	0.000	0	0	0.078	0.335	0.701	–
CMM	0.000	0	0	0.022	0.139	0.416	1

Table 10
Nemenyi test for accuracy comparison

	XGB	RF	GBM	XGBBT	RFBT	GBMBT	DT
RF	1.000	–	–	–	–	–	–
GBM	0.993	0.89	–	–	–	–	–
XGBBT	0.216	0.06	0.722	–	–	–	–
RFBT	0.989	0.863	1	0.762	–	–	–
GBMBT	0.060	0.012	0.375	1	0.416	–	–
DT	0.012	0.002	0.129	0.97	0.151	1	–
CMM	0.085	0.018	0.459	1	0.503	1	0.998

Table 11
Nemenyi test for average prediction depth comparison.

	XGB	RF	GBM	XGBBT	RFBT	GBMBT	DT
RF	1.000	–	–	–	–	–	–
GBM	1.000	1	–	–	–	–	–
XGBBT	0.000	0	0	–	–	–	–
RFBT	0.041	0.05	0.034	0.816	–	–	–
GBMBT	0.002	0.003	0.002	0.998	0.991	–	–
DT	0.000	0	0	0.072	0	0.009	–
CMM	0.000	0	0	0.395	0.007	0.101	0.994

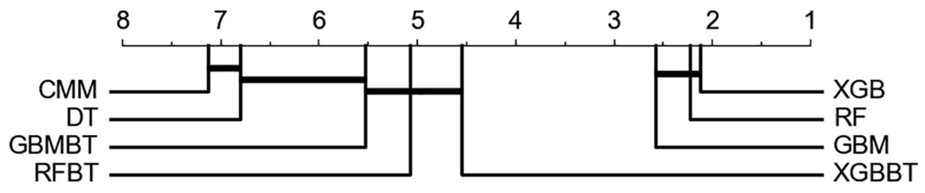


Fig. 6. Critical difference diagram of ROC AUC based on Wilcoxon–Holm post hoc test.

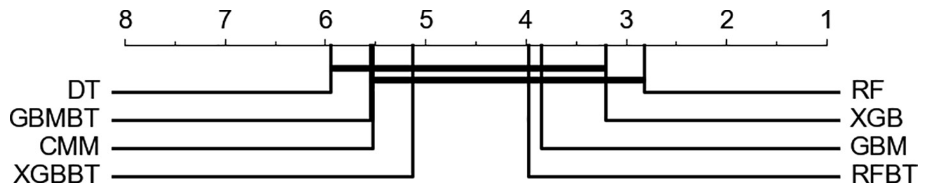


Fig. 7. Critical difference diagram of generalized accuracy based on Wilcoxon–Holm post hoc test.

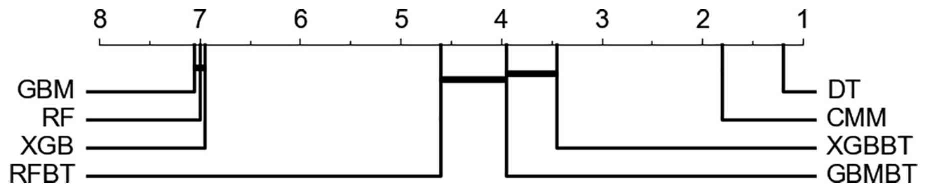


Fig. 8. Critical difference diagram of prediction average depth based on Wilcoxon–Holm post hoc test.

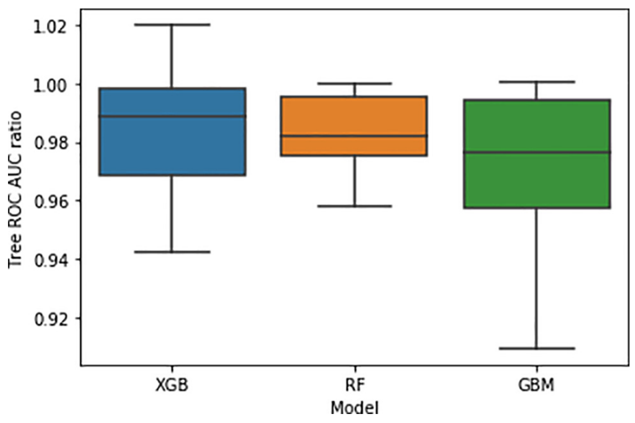


Fig. 9. Distribution of the ROC AUC ratio between the generated decision tree and its source decision forest over the different types of forests.

Wilcoxon–Holm post hoc test. It shows that the prediction depth of XGBBT is significantly lower than the prediction depth of RFBT. A reasonable explanation for this outcome is that the random forest-based tree is usually deeper than XGBoost based tree. In other words – it is more common that the algorithm will generate a shallower tree when applied on XGBoost rather than if it is applied on random forest, as visualized in Fig. 12. This figure on the other hand, also shows that the approximation of GBM can be done with shallow trees as well, and yet, the resulting tree is usually less accurate than either XGBBT or RFBT. This finding may imply for redundancy of information in the conjunction set extracted for GBMBT in relation to the other two models. Another potential explanation is that GBM is limited in its ability to consider many hypotheses as the inducing of each tree is highly affected by the trees that were trained before while in other models there is a higher level of randomization in the induction of each base tree. The optimal value of the tree depth is subject to the user preferences about the required complexity and accuracy and their trade-off. Table 8 shows that there is no rule-of-thumb for selecting the optimal depth. For some datasets (e.g., biodeg and spambase) increasing the maximum allowed depth can dramatically increase the predictive performance while for others there is a limit to the gained performance and even a decrease (e.g., credit and divorce). Fig. 13 shows how increasing the maximum allowed depth can affect the ROC AUC of the resulting mod-

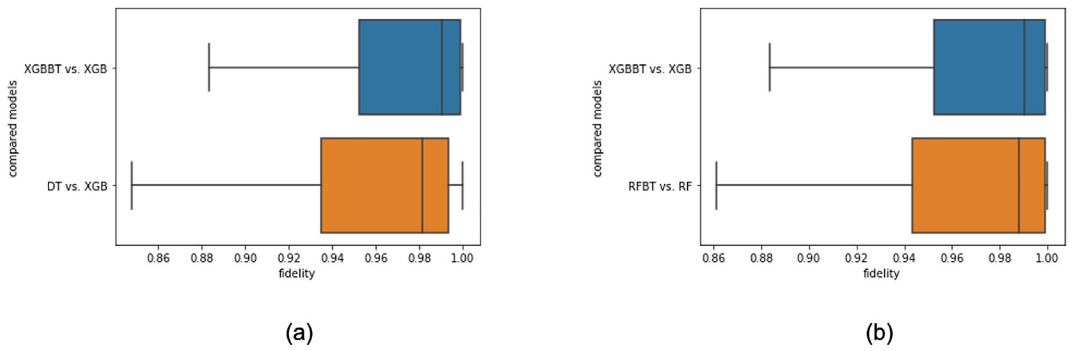


Fig. 10. Distribution of algorithms fidelity across the different datasets. The distribution is of the average fidelities across the different datasets. (a) compares the fidelities of DT and XGBBT with XGB while (b) compares the fidelity of RFBT and RF with the fidelity of XGBBT and XGB.

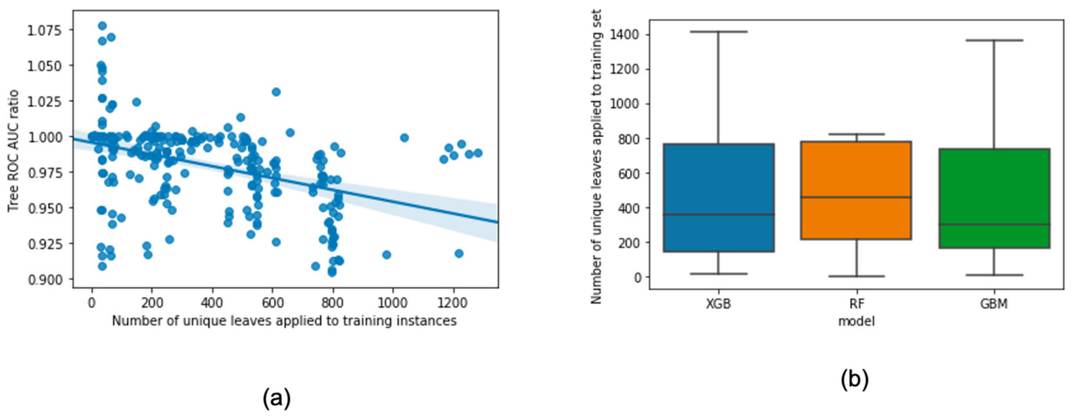


Fig. 11. (a) The correlation between the number of unique leaves (e.g., conjunction sets) that were applied to the training instances and the approximation ratio, calculated as the generated tree ROC AUC divided by the source forest ROC AUC (b) the distribution of the number of unique leaves across different types of decision forests.

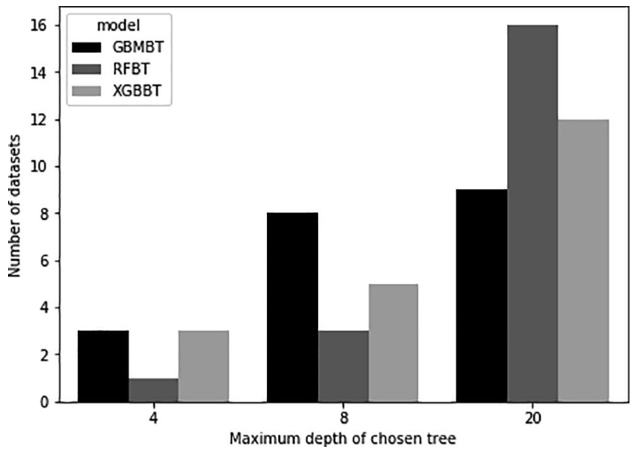


Fig. 12. Number of datasets per model for which the maximum depth of the trained tree was constrained in the corresponding level. For example: 3 of the GBMBT models had the best performance when the maximum tree depth was set to be 4.

els in two examples. For the kohkiloyeh dataset (a) – the optimal AUC is gained in maximum depth 8 while any further increase does not improve the predictive performance while for the haberman dataset (b) any value above 4 can actually impair the predictive performance so the user may end up with a tree that is more complex and less accurate.

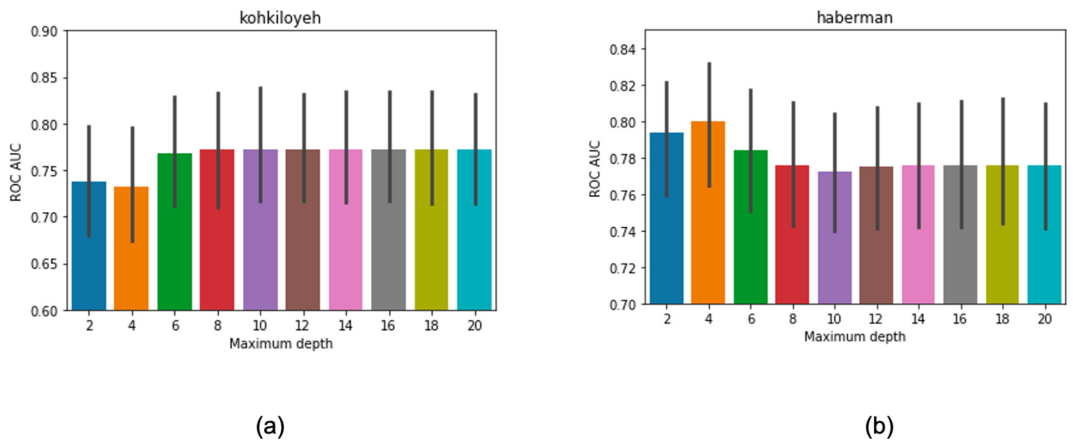


Fig. 13. The effect of tree maximum depth on the ROC AUC of the generated trees. Averages of 10 runs for 2 dataset examples: (a) Kohkiloyeh and (b) haberman.

5. Conclusion and future work

This paper presented a method that builds a decision tree that approximates the predictive performance of a pre-trained ensemble of trees (namely, decision forest). The developed method is an extension of the work presented in [39]. This method is compatible with both independently induced forests (e.g., random-forest) and dependently induced forests (e.g. XGBoost and GBM) and not only to the former type like the previous version of this work. Among the extensions that were added to the algorithm is the ability to restrict the maximum depth of the trained tree, the consideration of class distribution when generating the conjunction set that represents the source ensemble, the inclusion of all of the conjunctions that were applied to training instances and the usage of entropy over class probabilities as a splitting criterion when constructing the new tree. The method was implemented and evaluated for different types of ensembles like the random forest, gradient boosting machines, and XGBoost. The XGBoost implementation is currently available only for binary classification challenges. In the future, it is worthwhile to explore and develop an implementation that is suitable for multi-class XGBoost models. Another potential direction that can further improve this method is to refine the conjunction extraction stage. Today the method extract ensemble's information as a filtered set of conjunction set and some relevant conjunctions are inevitably omitted. Best practices from graph theory such as granular computing and normalized cuts [40]. As a side product, the generated decision tree can be also considered to be included in another black-box ensemble of complex models, as many of the black-box models often win in various machine-learning competitions. Finally, other types of pruning methods may be explored as potential alternatives to the greedy pruning method that was selected in this work.

CRediT authorship contribution statement

Omer Sagi: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing - original draft. **Lior Rokach:** Conceptualization, Methodology, Validation, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.ins.2021.05.055.

References

[1] A. Abdul, J. Vermeulen, D. Wang, B.Y. Lim, M. Kankanhalli, Trends and trajectories for explainable, accountable and intelligible systems: an hci research agenda, in: Proceedings of the 2018 CHI conference on human factors in computing systems, 2018, pp. 1–18.
[2] A. Adadi, M. Berrada, Peeking inside the black-box: a survey on explainable artificial intelligence (xai), IEEE Access 6 (2018) 52138–52160.
[3] Y. Akiba, S. Kaneda, H. Almuallim, Turning majority voting classifiers into a single decision tree, in: Tools with Artificial Intelligence, 1998. Proceedings. Tenth IEEE International Conference on, IEEE, 1998, pp. 224–230. .

[4] C. Apté, S. Weiss, Data mining with decision trees and decision rules, Future Gener. Comput. Syst. 13 (2–3) (1997) 197–210.
[5] A.B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Inf. Fusion 58 (2020) 82–115.
[6] O. Bastani, C. Kim, H. Bastani, Interpretability via model extraction. arXiv preprint arXiv:1706.09773, 2017. .
[7] R.K. Bellamy, K. Dey, M. Hind, S.C. Hoffman, S. Houde, K. Kannan, P. Lohia, S. Mehta, A. Mojsilovic, S. Nagar, et al, Think your artificial intelligence software is fair? think again, IEEE Softw. 36 (4) (2019) 76–80.
[8] L. Breiman, Classification and regression trees, Routledge, 2017.
[9] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016, pp. 785–794.
[10] X. Chen, L. Huang, D. Xie, Q. Zhao, Egbmmda: extreme gradient boosting machine for mirna-disease association prediction, Cell Death Disease 9 (1) (2018) 3.
[11] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, J. Peng, Xgboost classifier for ddos attack detection and analysis in sdn-based cloud, in: 2018 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, 2018, pp. 251–256.
[12] A. Chouldechova, D. Benavides-Prado, O. Fialko, R. Vaithianathan, A case study of algorithm-assisted decision making in child maltreatment hotline screening decisions, in: Conference on Fairness, Accountability and Transparency, 2018, pp. 134–148.
[13] M. Craven, J.W. Shavlik, Extracting tree-structured representations of trained networks, in: Advances in neural information processing systems, 1996, pp. 24–30. .
[14] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[15] T.G. Dietterich, Ensemble methods in machine learning, in: International workshop on multiple classifier systems, Springer, 2000, pp. 1–15. .
[16] P. Domingos, Knowledge discovery via multiple models, Intell. Data Anal. 2 (1–4) (1998) 187–202.
[17] W. Fan, F. Chu, H. Wang, P.S. Yu, Pruning and dynamic scheduling of cost-sensitive ensembles, in: AAAI/IAAI, 2002, pp. 146–151. .
[18] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. (2001) 1189–1232.
[19] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Comput. Surveys 51 (5) (2018) 93.
[20] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Comput. Surveys 51 (5) (2019) 93.
[21] J. Hatwell, M.M. Gaber, R. Azad, Chirps: explaining random forest classification, Artif. Intell. Rev. (2020).
[22] H. He, W. Zhang, S. Zhang, A novel ensemble method for credit scoring: adaption of different imbalance ratios, Expert Syst. Appl. 98 (2018) 105–117.
[23] Q. Hu, D. Yu, Z. Xie, X. Li, Eros: ensemble rough subspaces, Pattern Recogn. 40 (12) (2007) 3728–3739.
[24] X. Jiang, C.-A. Wu, H. Guo, Forest pruning based on branch importance, Comput. Intell. Neurosci. (2017). .
[25] S. Kandula, J. Shaman, Reappraising the utility of google flu trends, PLoS Comput. Biol. 15 (8) (2019) e1007258.
[26] A. Karpatne, G. Atluri, J.H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, V. Kumar, Theory-guided data science: a new paradigm for scientific discovery from data, IEEE Trans. Knowl. Data Eng. 29 (10) (2017) 2318–2331.
[27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: a highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems, 2017, pp. 3146–3154. .
[28] V.Y. Kulkarni, P.K. Sinha, Pruning of random forest classifiers: a survey and future directions, in: 2012 International Conference on Data Science & Engineering (ICDSE). IEEE, 2012, pp. 64–68.
[29] Z.C. Lipton, The mythos of model interpretability. arXiv preprint arXiv:1606.03490, 2016. .
[30] A. Lucic, H. Haned, M. de Rijke, Explaining predictions from tree-based boosting ensembles. arXiv preprint arXiv:1907.02582, 2019. .
[31] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.
[32] L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, in: Advances in Neural Information Processing Systems, 2018, pp. 6638–6648.
[33] S. Raschka, J. Patterson, C. Nolet, Machine learning in python: main developments and technology trends in data science, Mach. Learn. Artif. Intell. Inf. 11 (4) (2020) 193.
[34] M.T. Ribeiro, S. Singh, C. Guestrin, Why should i trust you?: Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, 2016, pp. 1135–1144.
[35] M.T. Ribeiro, S. Singh, C. Guestrin, Anchors: high-precision model-agnostic explanations, AAAI vol. 18 (2018) 1527–1535.
[36] J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (10) (2006) 1619–1630.
[37] L. Rokach, Decision forest: twenty years of research, Inf. Fusion 27 (2016) 111–125.
[38] O. Sagi, L. Rokach, Ensemble learning: a survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 8 (4) (2018) e1249.
[39] O. Sagi, L. Rokach, Explainable decision forest: transforming a decision forest into an interpretable tree, Inf. Fusion (2020).
[40] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[41] A. Stojić, N. Stanić, G. Vuković, S. Stanišić, M. Perišić, A. Šoštarić, L. Lazić, Explainable extreme gradient boosting tree-based prediction of toluene, ethylbenzene and xylene wet deposition, Sci. Total Environ. 653 (2019) 140–147.
[42] A. Van Assche, H. Blockeel, Seeing the forest through the trees: learning a comprehensible model from an ensemble, in: European Conference on Machine Learning. Springer, 2007, pp. 418–429.
[43] G. Vandewiele, O. Janssens, F. Ongenae, F. De Turck, S. Van Hoecke, Genesim: genetic extraction of a single, interpretable model, Stat 1050 (2016) 17. .
[44] C. Wang, C. Deng, S. Wang, Imbalance-xgboost: leveraging weighted and focal losses for binary label-imbalanced classification with xgboost, Pattern Recogn. Lett. (2020).
[45] C. Yang, A. Rangarajan, S. Ranka, Global model interpretation via recursive partitioning, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2018, pp. 1563–1570.
[46] C. Zhang, C. Liu, X. Zhang, G. Almpanidis, An up-to-date comparison of state-of-the-art classification algorithms, Expert Syst. Appl. 82 (2017) 128–150.
[47] Y. Zhang, S. Burer, W.N. Street, Ensemble pruning via semi-definite programming, J. Mach. Learn. Res. 7 (Jul) (2006) 1315–1338.
[48] Y.-L. Zhang, L. Li, Interpretable mtl from heterogeneous domains using boosted tree, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2053–2056.
[49] J. Zhong, Y. Sun, W. Peng, M. Xie, J. Yang, X. Tang, Xgbfemf: an xgboost-based framework for essential protein prediction, IEEE Trans. Nanobiosci. 17 (3) (2018) 243–250.
[50] Y. Zhou, G. Hooker, Interpreting models via single tree approximation. arXiv preprint arXiv:1610.09036, 2016. .