Deriving a Single Interpretable Model by Merging Tree-Based Classifiers

Valerio Bonsignori, Riccardo Guidotti, and Anna Monreale(✉)

University of Pisa, Pisa, Italy
v.bonsignori@studenti.unipi.it,
{riccardo.guidotti,anna.monreale}@unipi.it

Abstract. Decision tree classifiers have been proved to be among the most interpretable models due to their intuitive structure that illustrates decision processes in form of logical rules. Unfortunately, more complex tree-based classifiers such as oblique trees and random forests overcome the accuracy of decision trees at the cost of becoming non interpretable. In this paper, we propose a method that takes as input any tree-based classifier and returns a single decision tree able to approximate its behavior. Our proposal merges tree-based classifiers by an intensional and extensional approach and applies a post-hoc explanation strategy. Our experiments shows that the retrieved single decision tree is at least as accurate as the original tree-based model, faithful, and more interpretable.

Keywords: Interpretable machine learning · Decision tree · Oblique tree · Model transparency · Merging decision trees

1 Introduction

Decision tree (DT) classifiers are very popular models still widely adopted in various business domains because their tree-like representation of knowledge is intuitive and because generally makes the decision logic employed interpretable by humans. The drawback of DTs is that their greedy training procedure returns models which are not remarkably accurate, especially for complex classification problems. To address this issue, DTs have been "empowered" either by using ensembles such for Random Forests [8] or by adopting multivariate and nonlinear splitting conditions such as in Oblique Trees [18]. Such models can reach higher levels of accuracy than regular DTs. Unfortunately, the high accuracy of these complex tree-based classifiers is paid by giving up interpretability.

In the literature, we can find two research lines to deal with the lack of interpretability of these complex tree-based classifiers. The first one relates to *tree merging procedures* [1,5,7] and the idea is to merge a set of DTs into a single one "summarizing" their behavior. Strategies for merging trees are different: joining DTs learned in parallel from disjoint subsets of data [7]; inducing a DT from the intersection of decision tables, each one representing a tree [1]; applying a recursive *lossless* merging procedure that makes the order of the merging not relevant [5]. The second research line is related to *eXplainable Artificial Intelligence* (XAI) approaches [6]. Starting from [2], in the literature we can find a set of works aiming at approximating the behavior of a classifier with a single DT for explaining the classification reasoning. To reach this goal different strategies have been proposed: inducing a DT from a set of selected "prototypes" [10]; using genetic programming to evolve DTs to mimic the behavior of a neural network [9]; building several ensembles on synthetically augmented data and then, learning a single DT on the enriched data [3]; constructing a DT by using the set of rule conjunctions that represent the original decision forest [13]; interpreting tree ensembles by finding tree prototypes in the tree space[16].

In this paper we combine these two research lines. We propose a Single-tree Approximation MEthod (SAME) that exploits a procedure for merging decision trees, a post-hoc explanation strategy, and a combination of them to turn any tree-based classifier into a single and interpretable DT.

The implementation of SAME required to adapt existing procedures for merging traditional decision trees to oblique trees by moving from an intensional approach to an extensional one. Our experiments on eight tabular datasets show that SAME is efficient and that the retrieved single decision tree is at least as accurate as the original non interpretable tree-based model.

2 Setting the Stage

We address the *single tree approximation of tree-based black box classifiers* [2,6]. Consider a classification dataset X, Y consisting of a set $X = \{x_1, x_2, \dots, x_n\}$ of instances with l labels (or classes) assigned to an instance in the vector $Y \in \mathbb{N}^n$. An instance of $x \in \mathbb{R}^m$ consist in a set of m pairs of attribute-value (a_i, v_i), where a_i and v_i is a value from the domain of a_i. We define a classifier as a function $f : \mathcal{X}^{(m)} \rightarrow \mathcal{Y}$ which maps data instances x from a feature space $\mathcal{X}^{(m)}$ with m input features to a decision y in a label space $\mathcal{Y}$, where y can take l different labels. We write $f(x) = y$ to denote the decision y given by f on x. We assume that any classifier can be queried at will.

Given a not interpretable tree-based classifier b, such as Random Forests [8] or Oblique Trees [18], our aim is to define a function taking as input b, X, and Y and returns a single DT classifier d which should guarantee the following properties. First, d must be able to mime the behavior of b, i.e., $d(x) = b(x)$ for as many instances $x \in X$ as possible. Second, the accuracy of d on unseen instances should be comparable with the accuracy of b. Third, d should not be a complex and deep tree to guarantee high levels of interpretability. The single decision tree d is intrinsically transparent because it is humanly possible to understand the reasons for the decision process of every instance $d(x) = y$.

In the following we summarize some key concepts important for our proposal.

Merging Decision Trees. Merging DT approaches are accurately described in [14]. Four phases are identified to merge a set of trees $T_1, T_2, \dots, T_k$ trained on various subsets of a given dataset into a unique decision tree T.

In the *first phase*, a decision tree T_i is *transformed* into a set of rules or rule tables (also named decision regions or decision tables[1]). In the *second phase*, the decision regions of two models T_1, T_2 are *merged* using a specific approach. The most intuitive idea to merge two rule tables is to compute the *intersection* of all the combinations of the rules from each region and use the results as merged table model. If the regions of the rules that are being intersected are dis-joined, the resulting rule will be empty. The intersection of two overlapping regions is added to the final table model. The class label associated with it is obvious when the two initial regions have the same outcome, otherwise it is necessary to solve the class conflict by employing a specific strategy such as using the class of the rule the highest confidence or probability [1]; or associating to each region a weight and selecting the class with highest weight [15]. The approach presented in [5] allows for simultaneously merging the decision regions of every tree. It uses the notion of *condition tree*. Given a tree T and a condition C, let S_j denote the condition set of node j in T, which is composed of conditions from root to node j, then a condition tree $T^{(C)}$ is composed of those nodes in T such that all the conditions in S_j satisfies C. Hence, if an inner node in T is not included in $T^{(C)}$, then all its branches are not included in $T^{(C)}$. Once that two models are merged, the *third phase*, named "pruning", tries to reduce the number of regions. In [1] the regions with the lowest relative number of training samples are filtered out, while in [7] redundant rules are removed during the resolution of class conflicts. Another strategy joins adjacent regions with the same predicted class [1,15]. In [5] the final tree T is pruned by removing inner nodes having as leaves the same class. Finally, the *fourth phase* consists in *growing the final DT* from the decision regions. In [1,15] the final DT is obtained by using the same procedure used to create the initial trees on the values in the regions in the final decision table. In [5] the final DT is directly obtained from the merging procedure.

We adopt the recursive merging procedure described in [5] because *(i)* it is more efficient and requires less memory than others, *(ii)* it does not require to re-train a DT at the end of the computation, *(iii)* it produces a DT with multi-way splits that is theoretically less deep than a binary DT, *(iv)* the merging method is *lossless* as it maintains for every instance the class label assigned by the tree ensembles with the same majority voting.

Impact of Attribute Types on Tree-based Classifiers. The aforementioned procedure for merging DTs suffers in presence of many attributes, and also in presence of large (potentially infinite) domains for each attribute. Indeed, in [5] is shown that the size of T merged from the trees learned from data with categorical attributes are much smaller than the trees learned from data with numerical attributes. Therefore, in [5] numerical attributes are discretized using the *Recursive Minimal Entropy Partitioning* (RMEP) method described in [4]. RMEP recursively divides the numerical values minimizing the entropy of the target class. The obtained splits are used to define regions represented by a single representative value. In [5] Fan et al. show that there is a negligible effect on

[1] We use T_i for DT, rule tables, decision tables, and decision regions.

the classification accuracy when using discretization w.r.t. not using it. Finally, we turn categorical attributes into numbers through *target encoding* [11].

Post-hoc Explanation Strategy. Research on XAI has flourished over the last few years [6,12]. Explanation methods can be categorized as: *intrinsic* or *post-hoc*, depending on whether the machine learning approach is transparent or if the explanation is retrieved by querying the model after the training; and *local* or *global*, depending on whether the explainer reveals the reasons for the prediction of a specific instance, or the overall logic of obscure model. We mention this categorization because in our work we rely on *global post-hoc* explanation. Thus, given a black box classifier b trained on a dataset X, Y, a global post-hoc explainer f applied on b and X aims at finding an interpretable classifier c, i.e., $c = f(b, X)$, such that the behavior of c on X is adherent with the behavior of b on X, i.e., $b(X) \sim c(X)$. For instance, in [2] a particular DT c is trained on $X, \hat{Y} = b(X)$ and the global interpretable model c is returned as explanation.

3 Single-Tree Approximation Method

Our proposal to tackle the problem formulated in Sect. 2 consists of reducing any tree-based classifier to a single tree approximating its behavior. We name it SAME, standing for Single-tree Approximation MEthod, and we illustrate its pseudo-code in Algorithm 1. The main idea of SAME is to exploit procedures for merging DTs, a post-hoc explanation strategy, and a combination of them to turn any tree-based classifier into a single interpretable DT.

SAME takes as input a known dataset X, a tree-based classifier b, a flag μ indicating if oblique trees have to be merged, and a flag ν indicating if the post-hoc explanation approach is applied separately to each oblique tree of the forest. The algorithm returns a single DT classifier T. We assume that X has statistical properties similar to the training set used by b. It works in different ways depending on the type of b.

- *Case 1.* If b is a single DT it directly returns it (lines 9–10).
- *Case 2.* If b is a forest of DT (lines 11–12), then SAME runs the *forest2single* function (lines 1–4) that exploits the *mergeTrees* procedure described in [5].
- *Case 3.* If b is a single oblique tree, SAME runs the *b2forest* function to derive a random forest from b and then, from the forest it merges the various trees with *forest2single* like in Case 2 (lines 13–15). The *b2forest* function (lines 5–8) classifies X using the single oblique tree and then trains on $X, \hat{Y}$ a random forest classifier, i.e., it approximates the behavior of an oblique tree with a random forest.
- *Case 4.* If b is a forest of oblique trees and μ is false, SAME applies the same procedure described for *Case 3*, i.e., it runs the *b2forest* function that in this case derives a random forest mimicking the forest of oblique trees and from it merges the various trees with *forest2single* (lines 16–18).
- *Case 5.* If b is a forest of oblique trees and μ is true, SAME first runs the *oforest2osingle*, that as described in following subsection derives an oblique

```
Algorithm 1: SAME
Input  : $X$ - known data, $b$ - tree-based classifier, $\mu$ - merge oblique trees flag,
         $\nu$ - disjoint post-hoc explanation flag
Output: $T$ - single decision tree
function forest2single($b$):
    $\mathcal{T} = \{T_1, \ldots, T_k\} \leftarrow getTrees(b)$;
    return $mergeTrees(\mathcal{T})$;
function b2forest($b, X$):
    $\hat{Y} \leftarrow b(X)$;
    return $trainRandomForest(X, \hat{Y})$;
if $b$ is Single Decision Tree then
    $T \leftarrow b$;
else if $b$ is Forest of Decision Trees then
    $T \leftarrow forest2single(b)$;
else if $b$ is Single Oblique Trees then
    $RF \leftarrow b2forest(b, X)$;
    $T \leftarrow forest2single(RF)$;
else if $b$ is Forest Oblique Trees $\wedge \neg\mu$ then
    $RF \leftarrow b2forest(b, X)$;
    $T \leftarrow forest2single(RF)$;
else if $b$ is Forest of Oblique Trees $\wedge\ \mu \wedge\ \neg\nu$ then
    $OT \leftarrow oforest2osingle(b, X)$;
    $RF \leftarrow b2forest(OT, X)$;
    $T \leftarrow forest2single(RF)$;
else if $b$ is Forest Oblique Trees $\wedge\ \mu \wedge\ \nu$ then
    $\mathcal{T} \leftarrow \emptyset$;
    for $OT_i \in b$ do
        $RF_i \leftarrow b2forest(OT_i, X)$;
        $\mathcal{T} \leftarrow \mathcal{T} \cup getTrees(RF_i)$;
    $T \leftarrow forest2single(\mathcal{T})$;
return $T$;
```

trees from a forest of oblique trees and, then it turns the oblique tree OT into a single DT as in *Case 3* (lines 19–22).

- *Case 6.* If b is a forest of oblique trees μ is true and ν is true, SAME turns each oblique tree of the oblique forest into a forest of traditional DTs repetitively applying *b2forest*. Finally, it runs the *forest2single* on the union of the derived forests of DTs (lines 23–28).

SAME reduces any approximation problem with another one for which a solution is known in a sort of "cascade of approximations" making possible in this way to turn any classifier based on traditional or oblique trees into a single DT. The flags μ and ν controls the different type of approximation when the tree-based classifier is a forest of oblique trees: if μ is false, the post-hoc explanation strategy is directly employed for approximating the oblique forest; when μ is true and ν is false, the forest of oblique trees is approximated directly with the function *oforest2osingle* described in the following; when both are true, the post-hoc explanation approach is applied separately for each oblique tree.

Merging Oblique Trees. We define the *oforest2osingle* function used to merge a forest of oblique trees into a single oblique tree as an extension of the algorithm presented in [5]. The needs of adaptation comes from the higher complexity of the test in the nodes of oblique trees that can take the form of a multivariate test, and each multivariate test constitutes itself a meta-feature. A partition of the space using this higher level test changes the shape of the regions to be merged by the merging tree algorithm [5]. Thus, we define a different construction of the condition tree, and a more complex procedure for selecting the features for the final merge. We employ an available dataset X to model the relationship between two conditions exploiting the records in X satisfying those conditions. In other words, we turn the *relationship between two conditions* definition described in [5] from intentional to extensional. In [5] the relationship between two conditions is formally defined as:

Definition 1 (Relationships of Two Conditions). *Given two conditions C_1, C_2, where C_1 is a condition $s_i \in I_1$, and C_2 is a condition $s_j \in I_2$, with s_i, s_j being a split attribute, and I_1, I_2 being real value intervals. If $i = j$ and $I_1 \cap I_2 = \emptyset$, then $C_1 \cap C_2 = \emptyset$, in all the other cases $C_1 \cap C_2 \neq \emptyset$.*

That is, two conditions C_1 and C_2 have a relationship ($C_1 \cap C_2 \neq \emptyset$) if they identify a common region of the data. We define the *data-driven relationship between two multivariate conditions* as follows, exploiting the notion of *coverage* of a multivariate condition defined as the set of records in X satisfying (or covered by) the multivariate condition MC, i.e., $cov_X(MC) = \{x_i | \forall x_i \in X \text{ s.t. } MC(x_i)\}$, where $MC(x_i)$ is true if the record x_i satisfies the multivariate condition MC.

Definition 2 (Data-Driven Relationships of Two Multivariate Conditions). *Given a dataset X and two multivariate conditions MC_1, MC_2, we have that if $cov_X(MC_1) \cap cov_X(MC_1) = \emptyset$ then $MC_1 \cap MC_2 = \emptyset$, in all the other cases $MC_1 \cap MC_2 \neq \emptyset$.*

MC indicates a multivariate condition of a given oblique tree node, that can also involve a single variable. We define an oblique condition tree as follows:

Definition 3 (Oblique Condition Tree). *Given an oblique decision tree T, a multivariate condition MC, and a dataset X, let S_j denote the multivariate condition set of node j in T which is composed of the multivariate conditions from root to node j. An oblique condition tree $T^{(MC)}$ is composed of the nodes in the branch S_j satisfying $\{\forall\, MC' \in S_j, MC' \cap MC \neq \emptyset\}$. If an inner node in T is not included in $T^{(MC)}$, then all its branches are not included in $T^{(MC)}$.*

Given an oblique decision tree T and a multivariate condition MC, a simple algorithm for computing $T^{(MC)}$ is to traverse T depth-first from the root. For each branch of multivariate condition MC' of inner node j, there are two cases: *(i)* if MC' satisfies $MC_1 \cap MC_2 \neq \emptyset$ keep the root of that branch and search that branch recursively; *(ii)* if MC' satisfies $MC_1 \cap MC_2 = \emptyset$ then the whole branch is not included in $T^{(MC)}$. The definition of pruned condition trees is directly applied to pruned oblique decision tree. Indeed, the inner node j is kept in the oblique condition tree if there are records in X satisfying MC in both partitions after the split determined by MC' in node j, i.e., $|cov_X(MC \wedge MC')| > 0$ and $|cov_X(MC \wedge \neg MC')| > 0$. If this is not the case and $|cov_X(MC \wedge MC')| = 0$ or $|cov_X(MC \wedge \neg MC')| = 0$, then the oblique condition tree maintains only the sub-branch covering at least one instance. If both $|cov_X(MC \wedge MC')| = 0$ and $|cov_X(MC \wedge \neg MC')| = 0$, then no sub-branches must be added to the tree.

Table 1. Datasets details (left). Tree-based classifiers performance (right).

Dataset	n	m	l	DT		RF		OT		OF	
				acc	*F1*	*acc*	*F1*	*acc*	*F1*	*acc*	*F1*
`iris`	150	4	3	.933	.933	.933	.933	.933	.933	.933	.933
`cancer`	569	30	2	.921	.918	.930	.926	.921	.916	.956	.953
`armchair`	1000	2	3	.920	.922	.902	.902	.920	.922	.922	.924
`german`	1000	19	2	.720	.678	.660	.534	.735	.677	.755	.704
`employee`	1470	29	2	.816	.551	.854	.554	.871	.676	.850	.566
`compas`	7214	9	3	.628	.535	.631	.538	.634	.538	.636	.531
`fico`	10459	23	2	.712	.710	.717	.717	.707	.706	.730	.728
`adult`	32561	12	2	.853	.778	.854	.767	.851	.772	.850	.770

Table 2. Fidelity in approximating RF, OT, RF. Best values are underlined.

Dataset	RF		OT		OF			
	SAME	PHDT	SAME	PHDT	SAME$_{\neg\mu}$	SAME$_{\mu\neg\nu}$	SAME$_{\mu\nu}$	PHDT
`iris`	1.00	1.00	.933	1.00	.733	.333	1.00	1.00
`cancer`	.991	.947	.860	.947	.912	.912	.932	.930
`armchair`	.892	1.00	.918	1.00	.980	.838	.972	1.00
`german`	.975	.975	.945	.925	.810	.820	.785	.880
`employee`	1.00	.959	.969	.956	.898	.969	.963	.966
`compas`	.897	.979	.880	.996	.916	.859	.858	.994
`fico`	.978	.962	.908	.962	.911	.894	.911	.900
`adult`	.995	.994	.988	.951	.964	.992	.970	.988

Therefore, at a high level, the function *oforest2osingle* can be implemented as in [5] but updating the definition of condition tree with the definition of oblique condition tree. However, practically it is worth to mention another important difference from [5]. *Step 1* of the recursive merging procedure described in [5] determines the split attribute to use for the root of T by selecting the *most frequent* split attribute: when dealing with multivariate conditions of oblique trees is not trivial to determine the most frequent attribute. Thus, we defined the following policies: *(i)* Aiming at interpretability, we prefer univariate splits, acting on a unique variable, to multivariate, splits[2]. Among traditional univariate conditions we select the most frequent one. *(ii)* Among multivariate conditions we prefer those leading to the highest information gain during the training of the oblique tree that generated that split. *(iii)* In case of multivariate conditions with the same number of splits and with the same gain, we randomly select one of them.

[2] We highlight that also oblique trees can adopt as best split a traditional split.

4 Experiments

In this section we show the effectiveness of SAME when approximating different types of tree-based classifiers on various datasets[3].

We experimented SAME on eight datasets[4]. Details are in Table 1 (left). We split each dataset into three partitions: X_{tr} used for training tree-based classifiers (56%), X_{ap} used by SAME for the post-hoc approximation strategies (24%), X_{ts} used to measure the performance of the resultant single trees (20%).

We trained and approximated with a single decision tree the following tree-based classifiers: Decision Tree (DT) and Random Forest (RF) as implemented by *scikit-learn*; Oblique Decision Tree (ODT) and Oblique Forest (OF) as implemented in [17][5]. We select the best parameter setting for DTs and OTs using a randomized search with cross-validation on X_{tr} analyzing different max depths and class weights[6]. For RFs and OFs we used ensembles composed by 20 estimators and with max depth equals to 4. For OTs we adopted the House Holder CART version [18]. Regarding the parameters of SAME, for *Case 3, 4, and 5* we adopt a RF with 20 base estimators and a 20 max depth $[4, 5, 6, 7]$, while for *Case 6* we adopt a RF with 20 base estimators and a 20 max depth $[4, 5, 6]$. These parameters are the result of an a randomized search to find the best settings[7].

The classification performance are reported in Table 1 (right) in terms of accuracy and F1-score. We immediately notice that the OFs ha generally the best performance among the various tree-based classifiers. However, there is a small but statistically significant discrepancy among the accuracy scores (and F1-score) of the classifiers (non-parametric Friedman test p-value < 0.1).

To the best of our knowledge the problem treated is somewhat novel and in the literature there are not competitors explicitly designed for this task. Concerning post-hoc explanations, in line with Trepan [2], we compare SAME with PHDT, a post-hoc decision tree approximating any tree-based classifier with a DT. When the tree-based classifier is an OF, we adopt the notation $\text{SAME}_{\neg\mu}$, $\text{SAME}_{\mu\neg\nu}$, $\text{SAME}_{\mu\nu}$ to indicate *Cases 4, 5, and 6* (Sect. 3), respectively.

All the tree-based classifiers are trained on X_{tr}, SAME and PHDT exploit the X_{ap} partition while the evaluation measures are computed on X_{ts}.

Evaluation Measures. We evaluate the performances under different perspectives on the partition X_{ts}. First, we check to which extent the single tree T is able to accurately mime the behavior of b. We define the *fidelity* as $fid(Y_b, Y_T) = eval(Y_b, Y_T)$ where $Y_b = b(X_{ts})$, $Y_T = T(X_{ts})$, and *eval* can be

[3] Python code and datasets available at: https://github.com/valevalerio/SAME. Experiments run on Ubuntu 20.04 LTS, 252 GB RAM, 3.30GHz x 36 Intel Core i9.

[4] The datasets are available on SAME Github, on the UCI repository https://archive.ics.uci.edu/ml/index.php, and on Kaggle https://www.kaggle.com/datasets.

[5] `scikit-learn`: https://scikit-learn.org/, [17] Github repository https://github.com/TorshaMajumder/Ensembles_of_Oblique_Decision_Trees.

[6] max depth $\in \{5, 6, 7, 8, 10, 12, 16, \textit{unbounded}\}$, class weight $\in \{\textit{normal}, \textit{balanced}\}$.

[7] Details of the parameters tested can be found in SAME repository.

the *accuracy* or the *F1-score*. Second, we test if T can replace b, i.e., how much is accurate T if compared with the b on unseen instances X_{ts}. We define the *accuracy deviation* Δ as $\Delta = eval(Y, Y_T) - eval(Y, Y_b)$ where Y being the vector of real class for the partition X_{ts}, $Y_b = b(X_{ts})$, $Y_T = T(X_{ts})$ and *eval* can be the *accuracy* or the *F1-score*. Δ is positive if T is better than b on unseen data, it is negative otherwise, it is zero if they have exactly the same performance[8]. Third, we measure characteristics describing a decision tree T such as: *number of leaves*, *number of nodes*, *tree depth*, and *average path length*. We aim at obtaining low values since a simple model is generally more interpretable.

Results. We present the results obtained by approximating single trees with SAME and PHDT from DT, RF, OT, and OF with the available variants.

Table 2 reports the fidelity using the accuracy as *eval* (similar results are recorded for F1-score). We observe that both SAME and PHDT have good performance in approximating the behavior of the various tree-based classifiers. Indeed, they are even in terms of times which overcomes the other method. For SAME the best approximations are those performed using *Case 2* on the RF.

Table 3. Accuracy deviation on test set for RF, OT, OF. Best values are underlined.

Dataset	RF		OT		OF			
	SAME	PHDT	SAME	PHDT	SAME$_{\neg\mu}$	SAME$_{\mu\neg\nu}$	SAME$_{\mu\nu}$	PHDT
iris	$\underline{0.000}$	$\underline{0.000}$	$\underline{0.067}$	0.000	-0.200	–0.600	$\underline{0.000}$	$\underline{0.000}$
cancer	–0.009	$\underline{0.000}$	–0.035	$\underline{-0.018}$	–0.053	–0.070	$\underline{-0.035}$	$\underline{-0.035}$
armchair	–0.065	$\underline{0.000}$	–0.047	$\underline{0.000}$	–0.010	–0.105	–0.012	$\underline{0.000}$
german	$\underline{-0.015}$	$\underline{-0.015}$	$\underline{-0.025}$	–0.035	–0.050	–0.060	$\underline{-0.025}$	-0.030
employee	$\underline{0.000}$	–0.014	$\underline{-0.010}$	–0.017	–0.061	$\underline{-0.001}$	–0.010	–0.007
compas	–0.023	$\underline{0.003}$	–0.034	$\underline{-0.001}$	–0.003	–0.024	-0.026	$\underline{-0.001}$
fico	$\underline{-0.002}$	–0.003	$\underline{0.006}$	0.001	–0.022	$\underline{-0.009}$	–0.021	–0.016
adult	$\underline{-0.001}$	–0.002	$\underline{-0.003}$	–0.010	–0.016	$\underline{0.000}$	–0.008	–0.002

Table 3 and Table 4 shows respectively *(i)* the accuracy deviation using the accuracy as *eval* (similar results for F1-score), and *(ii)* the accuracy of the decision trees obtained from the approximation of tree-based models trained on X_{tr} compared with DTs directly trained on X_{tr} and tested on X_{ts}. In Table 3 we observe that the deviation accuracy is only limitedly smaller than zero. This indicates that the approximated trees have a predictive power comparable to the original tree-based classifiers. SAME leads to a decision tree which is more accurate than the original model four times more than PHDT does. Table 4 highlights that in five cases out of eight, the decision tree approximated by SAME is a better model than a decision tree directly trained on the training data. Table 5

[8] We highlight that even though they can have the same performance there is no guarantee that the mis-classification errors are the same.

Table 4. Accuracy on test set for single trees approximating RF, OT, OF compared with the accuracy of the DT. Best values are underlined.

Dataset	DT	RF		OT		OF			
		SAME	PHDT	SAME	PHDT	SAME$_{\neg\mu}$	SAME$_{\mu\neg\nu}$	SAME$_{\mu\nu}$	PHDT
`iris`	.933	.933	.933	1.00	.933	.733	.333	.933	.933
`cancer`	.921	.912	.921	.895	.912	.868	.851	.921	.921
`armchair`	.920	.855	.920	.855	.902	.910	.815	.910	.922
`german`	.720	.705	.705	.635	.625	.685	.675	.730	.725
`employee`	.816	.816	.802	.844	.837	.810	.870	.840	.843
`compas`	.628	.605	.631	.597	.630	.631	.610	.610	.635
`fico`	.712	.710	.709	.723	.718	.685	.798	.709	.714
`adult`	.843	.852	.851	.851	.844	.835	.851	.842	.848

reports the tree depth. We omit the other characteristics describing decision trees for space reasons. We observe that in general the trees returned by PHDT are more compact than those returned by SAME. However in both cases they are nearly always deeper than the original DTs.

Table 5. Decision trees depth. Bests values are underlined.

Dataset	DT	RF		OT		OF			
		SAME	PHDT	SAME	PHDT	SAME$_{\neg\mu}$	SAME$_{\mu\neg\nu}$	SAME$_{\mu\nu}$	PHDT
`iris`	3	4	3	5	3	4	1	4	3
`cancer`	6	19	6	5	6	1	13	10	4
`armchair`	7	4	6	3	6	4	4	5	5
`german`	4	3	3	21	8	13	13	14	7
`employee`	3	28	5	9	7	4	10	17	4
`compas`	8	13	10	12	10	5	10	10	11
`fico`	7	25	12	16	9	32	24	14	8
`adult`	8	15	10	12	13	13	10	8	11

5 Conclusion

We have presented SAME, a single-tree approximation method designed to effectively and efficiently turn every tree-based classifier into a single DT. Experimentation on various datasets reveals that SAME is competitive with baseline approaches or overcomes them. Moreover, the approximated tree can replace the original model as it can have better performance. Possible future research directions are: extending SAME to any type of tree-based and rule-based classifier and using SAME as post-hoc global explanation method for any black box.

Acknowledgment. Work partially supported by the European Community H2020 programme under the funding schemes: G.A. 871042 *SoBigData++*, G.A. 761758 *Humane AI*, G.A. 952215 *TAILOR* and the ERC-2018-ADG G.A. 834756 "XAI: Science and technology for the eXplanation of AI decision making".

References

1. Andrzejak, A., et al.: Interpretable models from distributed data via merging of decision trees. In: CIDM, pp. 1–9. IEEE (2013)
2. Craven, M.W., et al.: Extracting tree-structured representations of trained networks, pp. 24–30 (1995)
3. Domingos, P.M.: Knowledge discovery via multiple models. Intell. Data Anal. **2**(1–4), 187–202 (1998)
4. Dougherty, J., et al.: Supervised and unsupervised discretization of continuous features. In: ICML, pp. 194–202. Morgan Kaufmann (1995)
5. Fan, C., et al.: Classification acceleration via merging decision trees. In: FODS, pp. 13–22. ACM (2020)
6. Guidotti, R., et al.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 93:1–93:42 (2019)
7. Hall, L.O., et al.: Combining decision trees learned in parallel. In: Working Notes of the KDD-97 Workshop on Distributed Data Mining, pp. 10–15 (1998)
8. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1998)
9. Johansson, U., et al.: Evolving decision trees using oracle guides. In: CIDM, pp. 238–244. IEEE (2009)
10. Krishnan, R., et al.: Extracting decision trees from trained neural networks. Pattern Recognit. **32**(12), 1999–2009 (1999)
11. Micci, D.: A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. SIGKDD Explor. **3**(1), 27–32 (2001)
12. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artif. Intell. **267**, 1–38 (2019)
13. Sagi, O., Rokach, L.: Explainable decision forest: transforming a decision forest into an interpretable tree. Inf. Fusion **61**, 124–138 (2020)
14. Strecht, P.: A survey of merging decision trees data mining approaches. In Proceedings of10th Doctoral Symposium in Informatics Engineering, pp. 36–47 (2015)
15. Strecht, P., Mendes-Moreira, J., Soares, C.: Merging decision trees: a case study in predicting student performance. In: Luo, X., Yu, J.X., Li, Z. (eds.) ADMA 2014. LNCS (LNAI), vol. 8933, pp. 535–548. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14717-8_42
16. Tan, S., et al.: Tree space prototypes: another look at making tree ensembles interpretable. In: FODS, pp. 23–34. ACM (2020)
17. Torsha, M.: Ensembles of oblique decision trees (2020)
18. Wickramarachchi, D.C., et al.: HHCART: an oblique decision tree. Comput. Stat. Data Anal. **96**, 12–23 (2016)