International Journal on Software Tools for Technology Transfer (2023) 25:287–299
https://doi.org/10.1007/s10009-023-00702-5

EXPLANATION PARADIGMS LEVERAGING ANALYTIC INTUITION

Special Section: Introducing Explanation Paradigms Leveraging Analytic Intuition

Forest GUMP: a tool for verification and explanation

Alnis Murtovi[1] · Alexander Bainczyk[1] · Gerrit Nolte[1] · Maximilian Schlüter[1] · Bernhard Steffen[1]

Accepted: 10 March 2023 / Published online: 30 May 2023

Abstract

In this paper, we present Forest GUMP (for Generalized, Unifying Merge Process) a tool for verification and precise explanation of Random forests. Besides pre/post-condition-based verification and equivalence checking, Forest GUMP also supports three concepts of explanation, the well-known *model explanation* and *outcome explanation*, as well as *class characterization*, i.e., the precise characterization of all samples that are equally classified. Key technology to achieve these results is algebraic aggregation, i.e., the transformation of a Random Forest into a semantically equivalent, concise white-box representation in terms of Algebraic Decision Diagrams (ADDs). The paper sketches the method and demonstrates the use of Forest GUMP along illustrative examples. This way readers should acquire an intuition about the tool, and the way how it should be used to increase the understanding not only of the considered dataset, but also of the character of Random Forests and the ADD technology, here enriched to comprise infeasible path elimination. As Forest GUMP is publicly available all experiments can be reproduced, modified, and complemented using any dataset that is available in the ARFF format.

Keywords Random forest · Binary/algebraic decision diagram · Aggregation · Infeasible paths · Verification · Equivalence checking · Explainability · Random seed

1 Introduction

Random Forests are one of the most widely known classifiers in machine learning [3, 19]. The method is easy to understand and implement, and at the same time achieves impressive classification accuracies in many applications [10]. Compared to other methods, Random Forests are fast to train and they are clearly more suitable for smaller datasets. In contrast to a single decision tree, Random Forests, a collection of many trees, do not overfit as easily on a dataset and their variance decreases with their size. On the other hand, Random Forests are considered black-box models because of their highly parallel nature: following the execution of Random Forests means, in particular, following the execution in all the involved trees. Such black-box executions are hard to explain to a human user even for very small examples.

In contrast, individual decision trees are considered white-box models because of their sequential evaluation nature. Even if a tree is large in size, a human can easily follow its computation step-by-step by evaluating (simple) decisions at each node from the root to a leaf. Indeed, the set of decisions along such an execution path precisely explains why a certain choice has been taken.

Popular methods towards explainability try to establish some user intuition. For example, they may hint at the most influential input data, like highlighting or framing the area of a picture where a face has been identified [24, 25]. Such information is very helpful, in particular to reveal some of the "popular" drastic mismatches incurred by neural networks: if the framed area of the image does not contain the "tagged" object, the identification is clearly questionable [31]. However, even in a correct classification, the tag by itself gives no reason why the identification is indeed correct.

More ambitious are methods that try to turn black-box model into white-box models, ideally preserving the semantics of the classification function. For Random Forests this has been achieved for the first time in [12, 14] using the 'aggregating power' of Algebraic Decision Diagrams (ADDs) and Binary Decision Diagrams (BDDs). ADDs are essen-

✉ A. Murtovi
alnis.murtovi@tu-dortmund.de

A. Bainczyk
alexander.bainczyk@tu-dortmund.de

G. Nolte
gerrit.nolte@tu-dortmund.de

M. Schlüter
maximilian.schlueter@tu-dortmund.de

B. Steffen
bernhard.steffen@tu-dortmund.de

[1] Dortmund, Germany

tially decision trees whose leaves are labeled with elements of some algebra, whereas BBDs are the special case for the algebra of Boolean values. Lifting the algebraic operations from the leaves to the entire ADDs/BDDs allows one to aggregate entire Random Forests into single semantically equivalent ADDs, the precondition for solving three explainability problems:

- The *Model Explanation Problem* [17], i.e. the problem of making the model as a whole interpretable, is solved in terms of an ADD that specifies precisely the same classification function as the original Random Forest (cf. Sect. 8.2).
- The *Class Characterization Problem* [12, 14], i.e. the problem, given a class c, characterizing the set of all samples that are classified by the Random Forest as c. This problem is solved in terms of a BDD which precisely characterizes this set of samples (cf. Sect. 8.3).
- The *Outcome Explanation Problem* [17], i.e. the problem of explaining a concrete classification, is solved in terms of a minimal conjunction of (negated) decisions that are sufficient to guide the sample into the considered class (cf. Sect. 8.4).

This paper is an extended version of [27] where Forest GUMP (for Generalized, Unifying Merge Process) was presented as a tool for providing a tangible experience with the three described concepts of (precise) explanation. Novel concepts in this paper are the verification of pre/post-conditions and equivalence checking. Technically, both heavily rely on the complete elimination of infeasible paths which is therefore treated in detail in this paper. We treat verification as special cases of model explanation:

- For pre-post-condition-based verification we simply project the model explanation of the considered ADD onto the part that is consistent with the precondition and check whether all remaining leaves satisfy the postcondition. The required projection can efficiently be implemented as part of infeasibility elimination.
- Equivalence checking is done by introducing an equality relation on ADDs that produces a BDD whose true leaves characterize all paths where the classification agree, while the false leaves characterizes all cases where the classification disagrees. The required equality relation is, in fact, simply the lifted version of the ordinary equality relation on the set of classes.

As Forest GUMP is publicly available all experiments can be reproduced, modified, and complemented using any data set that is available in the ARFF format [38].

Our implementation relies on the standard Random Forest implementation in Weka [39] and on the ADD implementation of the ADD-Lib [13, 16, 35]. Whereas the verification part in this paper is entirely new, the reader may find a more detailed description of the transformations, the three concepts of explanation, and a quantitative analysis in [12, 14, 15].

Related work In recent years, several approaches for verifying decision tree ensembles, including Random Forests, have been developed. They mostly propose solutions to answer robustness queries such as whether it is possible to change the output of a classifier for a given input x by slightly modifying the input or finding the nearest instance x' such that the classifier's prediction is changed. These methods work by e.g. encoding the problem as a mixed integer linear program [21], as a max-clique searching problem in k-partite graphs [5] or as a SMT formula [9, 33]. There are also approaches based on abstract interpretation [30, 36]. To our knowledge, we are the first to propose verification methods for Random Forests based on Algebraic Decision Diagrams.

There exist various methods for improving the understandability of Random Forests such as extracting decision rules from the considered black-box model [7], methods that are agnostic to the black-box model under consideration [23, 32] or by deriving a single decision tree from the black-box model [6, 8, 18, 37, 40]. In this context, single decision trees are considered key to a solution of both, the model explanation and outcome explanation problem. State-of-the-art solutions to derive a single decision tree from a Random Forest are approximative [6, 8, 18, 37, 40]. Thus, their derived explanations are not fully faithful to the original semantics of the considered Random Forest. This is in contrast to our ADD-based aggregation, which precisely reflects the semantics of the original Random Forest.

After a short introduction to Random Forests in Sect. 2, we present our approach to their aggregation in Sect. 3, followed by a detailed discussion concerning the elimination of *all* redundant predicates (which is essential for verification) in Sect. 4. Subsequently, we present a powerful, but non-compositional abstraction in Sect. 5, before we describe both, pre/post-condition-based verification and equivalence checking in Sect. 6, Forest GUMP in Sect. 7, and comprehensive use cases of Forest GUMP in Sect. 8. The paper closes with conclusions and directions to future work in Sect. 9.

2 Random Forests

Learning Random Forests is a quite popular, and algorithmically relatively simple classification technique that yields good results for many real-world applications. Its decision model generalizes a training dataset that holds examples of input data labeled with the desired output, also called *class*.

More concretely, a *Random Forest* is a collection of decision trees that are typically themselves classifiers that were

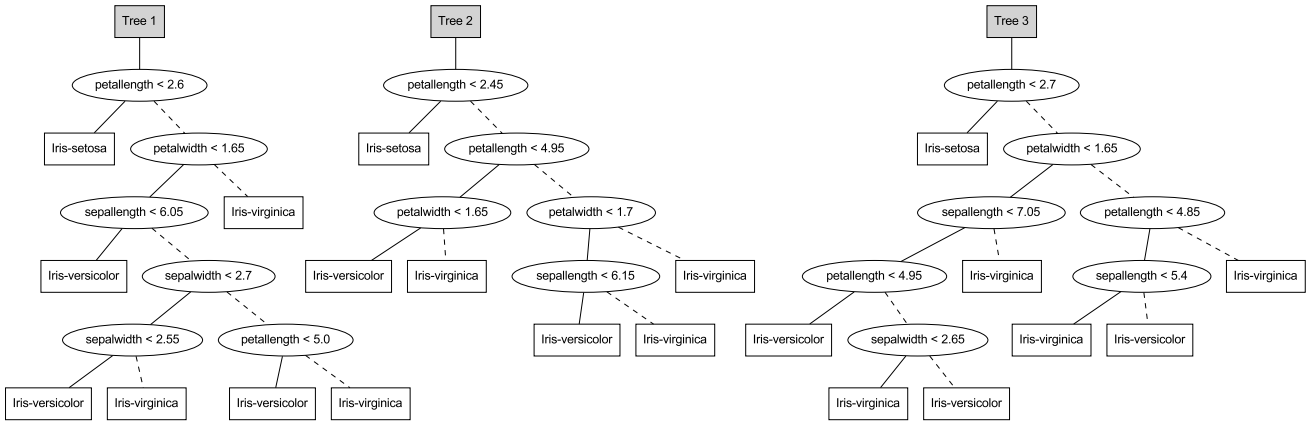

Fig. 1 Random Forest learned from the Iris dataset [11] (39 nodes)

each learned from a random sample of a given training dataset. Figure 1 shows a Random Forest with three trees. In practice, this number is typically one or two orders of magnitude higher. Section 8, e.g., discusses an example with 20 decision trees.

The point of Random Forests is to turn classification into a 'democratic' process which has much better statistical properties than approaches based on individual trees. This process proceeds in two steps:

– First, each decision tree of the forest is evaluated individually for the considered input data by tracing from their roots down to one of the leaves which yields one decision per tree, i.e. the predicted class.
– Second, the individual decisions are aggregated via *majority vote* which selects the most frequently chosen class.

The key advantage of this approach is, compared to single decision trees, the reduced variance. A detailed introduction to Random Forests, decision trees, and their learning procedures can be found in [3, 19, 29].

In this paper, we use Weka [39] as our reference implementation of Random Forests. However, our approach does not depend on implementation details and can be easily adapted to other implementations.

Figure 1 shows a small Random Forest that was learned from the popular Iris dataset [11]. The dataset lists dimensions of Iris flowers' sepals and petals for three different species. Using this forest to decide the species on the basis of given measurements requires to first evaluate the three trees individually and to subsequently determine the majority vote. This effort clearly grows linearly with the size of the forest. In the following we use this example to illustrate our approach of forest aggregation for explainability.

Key idea behind our approach is to partially evaluate the Random Forests at construction time which, in particular, eliminates redundancies between the individual trees of a Random Forest. E.g., in our accompanying Iris flower example (cf. Fig. 1) the predicate $petalwidth < 1.65$ is used in all three trees. This can easily lead to cases where the same predicate is evaluated many times in the classification process. The partial evaluation proposed in this paper transforms Random Forests into decision structures where such redundancies are totally eliminated.

An adequate data structure to achieve this goal for binary decisions are Binary Decision Diagrams [1, 4, 22] (BDDs): For a given predicate ordering, they constitute a normal form where each predicate is evaluated at most once, and only if required to determine the final outcome.

Algebraic Decision Diagrams (ADDs) [2] generalize BDDs to capture functions of the type $\mathbb{B}^{\mathcal{P}} \to C^n$ which are exactly what we need to specify the semantics of Random Forests for a classification domain C over a set of predicates $\mathcal{P}$. Moreover, in analogy to BDDs, which inherit the algebraic structure of their co-domain $\mathbb{B}$, ADDs also inherit the algebraic structure of their co-domains if available.

We exploit this property during the partial evaluation of Random Forests by considering the class vector co-domain (cf. Sect. 3). The aggregation to achieve the corresponding optimized decision structures is then a straightforward consequence of the used ADD technology.

3 Class vector aggregation

Class vectors faithfully represent the information about how many trees of the original Random Forest voted for a certain outcome. Obviously, this information is sufficient to obtain the precise results of a corresponding majority vote. Formally, the domain of *class vectors* forms a monoid

$$V := (\mathbb{N}^{|C|}, +, \mathbf{0})$$

where addition $+$ is defined component-wise and $\mathbf{0}$ is the neutral element.

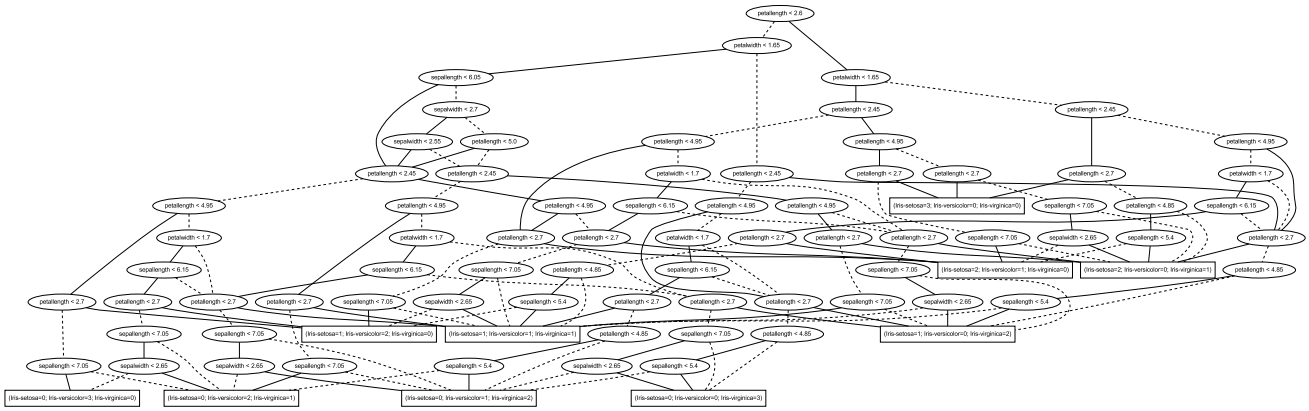

Fig. 2 Class vector aggregation of the Random Forest (83 nodes)

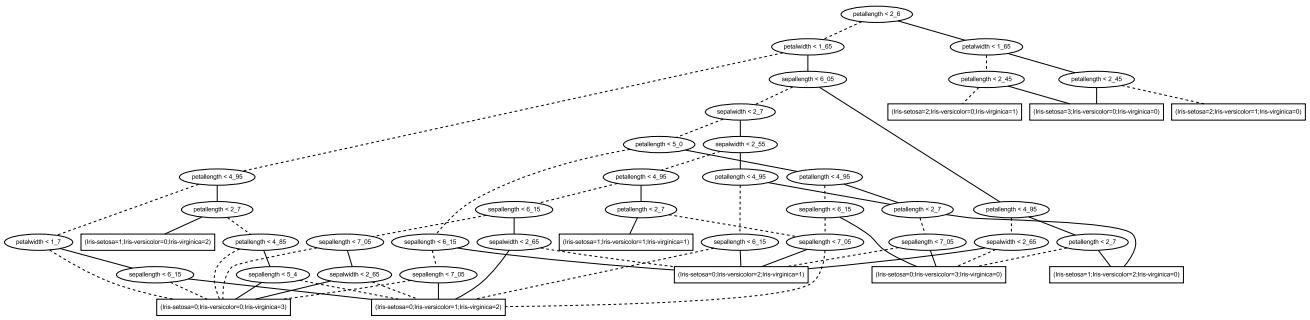

Fig. 3 Class vector aggregation of the Random Forest without semantically redundant nodes (43 nodes)

With the compositionality of the algebraic structure V and the corresponding ADDs $\mathcal{D}_V$, we can transform any Random Forest incrementally into a semantically equivalent ADD. Starting with the empty Random Forest, i.e. the neutral element $\mathbf{0}$, we consider one tree after the other, aggregating a growing sequence of decision trees until the entire forest is entailed in the new decision diagram. The details of this transformation are described in [12]. Figure 2 shows the result of this transformation for our running example.

4 Infeasible path elimination

When aggregating the trees of a Random Forest they all use varying sets of predicates. In contrast to simple Boolean variables, predicates are not independent on one another, i.e. the evaluation of one predicate may yield some degree of knowledge about the outcome concerning other predicates. E.g., the predicate $petallength < 2.45$ induces knowledge about other predicates that reason about $petallength$: When the petal length is smaller than 2.45 it cannot possibly be greater or equal to 2.7 at the same time. This is not taken care of by the symbolic treatment of predicates we followed until now. In fact, predicates are typically considered independent in the ADD/BDD community.

Infeasible path elimination, as illustrated by the difference between Fig. 2 and Fig. 3 for our running example, leverages the potential of a semantic treatment of predicates with significant effect on the size of the resulting ADDs. In fact, the experiments with thousands of trees reported in [12] would not have been successful without infeasible path elimination.

While [12] already briefly describes the functionality of the infeasible path reduction, we will here specify the way the elimination works by means of a recursive function $\lambda\colon \mathcal{D}_A \to \mathcal{D}_A$, where $\mathcal{D}_A$ is the set of ADDs for an algebra A. Let $\Pi(\mathcal{P})$ be the set of all possible path conditions, i.e. the set of all conjunctions over, possibly negated, predicates of $\mathcal{P}$. We first define a function $\lambda' : \mathcal{D}_A \times \Pi(\mathcal{P}) \to \mathcal{D}_A$ as follows

$$\lambda'((p,t,u),pc) := \begin{cases} \lambda'(t,pc) & \text{if } pc \implies p \\ \lambda'(u,pc) & \text{if } pc \implies \neg p \\ (p,\lambda'(t,pc \wedge p), & \\ \quad \lambda'(u,pc \wedge \neg p)) & \text{otherwise} \end{cases}$$

$$\lambda'(a,pc) := a$$

Here, the path condition pc is the conjunction of predicates seen along the path from the root to a node. If we are at an inner node (p,t,u), there are three different cases:

1. If the path condition pc implies the predicate p, then the node (p,t,u) is redundant as the predicate p is always true and we know that we will always follow the then-successor t in this case. Thus, $\lambda'(t,pc)$ is returned.
2. The second case, where the path condition pc implies $\neg p$ can be handled analogously.
3. If the path condition pc implies neither p nor $\neg p$, we do not know whether the predicate p holds and the node is not redundant. When we make the recursive calls to the then- and else-successor we add p, respectively $\neg p$ to the path condition.

We can then define λ as

$$\lambda(x) \coloneqq \lambda'(x, true)$$

Please note that infeasible path elimination

- is only required after aggregation: The trees in the original Random Forest have no infeasible paths by construction. They are introduced in the course of our *symbolic* aggregation, which is insensitive to semantic properties.
- is compositional and can therefore be applied during the step-wise transformation, before the final most frequent label abstraction (cf. Sect. 5), and at the very end.
- does not support normal forms: Whereas class vector abstraction is canonical for a given variable ordering, infeasible path elimination is not! Thus our approach may yield different decision diagrams depending on the order of tree aggregation. It is guaranteed, however, that the resulting decision diagrams are minimal.

Infeasible path elimination is a hard problem in general.[1] For the theory of linear algebra considered in this paper, however, LP-solvers are powerful enough to identify infeasible paths in polynomial time.

Class vector aggregation and infeasible path elimination are both compositional and can therefore be applied in arbitrary order without changing the semantics. The majority vote at compile time described in the next section is not compositional and must therefore be applied at the very end.

5 Majority vote at compile time

As mentioned above, maintaining the information about the result of the majority votes is not compositional. In fact, knowing the result of the majority votes for two Random Forest gives no guarantees about the majority vote of the combined forest. Thus the majority vote abstraction can only be applied at the very end, after the entire aggregation has been computed compositionally.

The result of the compositional aggregation process, including infeasible path elimination, is a decision diagram $d \in \mathcal{D}_V$ with class vectors in its terminal nodes. The majority vote abstraction $\Delta_C : \mathcal{D}_V \to \mathcal{D}_C$ can now be defined as the lifted version of the majority vote abstraction on class vectors $\mathbf{v} \in \mathbb{N}^{|C|}$ (cf. [12]):

$$\delta_C(\mathbf{v}) \coloneqq \arg\max_{c \in C} \mathbf{v}_c.$$

Note that δ_C does not project into the same carrier set but rather from one algebraic structure V into another C.[2] However, these transformations can be applied to the corresponding decision diagrams in the very same way. Figure 4 shows the result of the most-frequent-class abstraction for our running example.

6 Random Forest verification

In this section, we present solutions of two verification problems, pre-post-condition-based verification and equivalence checking, as direct applications of model explanation [12].

6.1 Pre/post-condition-based verification

Given a precondition ϕ and a postcondition ψ and an ADD, we can either verify that for all inputs that satisfy ϕ the ADD's output satisfies ψ or we can provide a counter example that shows that the specification does not hold.

We can perform this procedure using the infeasible path elimination described in Sect. 4. One can easily incorporate the precondition to the infeasible path reduction by calling λ' with ϕ instead of $true$, i.e.

$$\lambda(x) \coloneqq \lambda'(x, \phi)$$

As we are concerned with classification and the output of a Random Forest is a class $c \in C$, postconditions can be specified as subsets $C \subset C$, e.g. C can be a subset of "good" classes. We can then define $\delta_{post}(C)\colon C \to \mathbb{B}$ as

$$\delta_{post}(C)(c) \coloneqq \begin{cases} 1 & \text{if } c \in C \\ 0 & \text{otherwise.} \end{cases}$$

The function $\delta_{post}(C)$ can be lifted to operate on ADDs, yielding $\Delta_{post}(C)\colon \mathcal{D}_C \to \mathcal{D}_{\mathbb{B}}$. If we apply the function $\Delta_{post}(C)$ to an ADD to which the infeasible path elimination with the precondition ϕ has been applied and the resulting

[1] For the cases considered here it is polynomial, but there are of course theories for which it becomes exponentially hard or even undecidable.

[2] Here, $(C, \emptyset)$ is a trivial algebra with no operations.

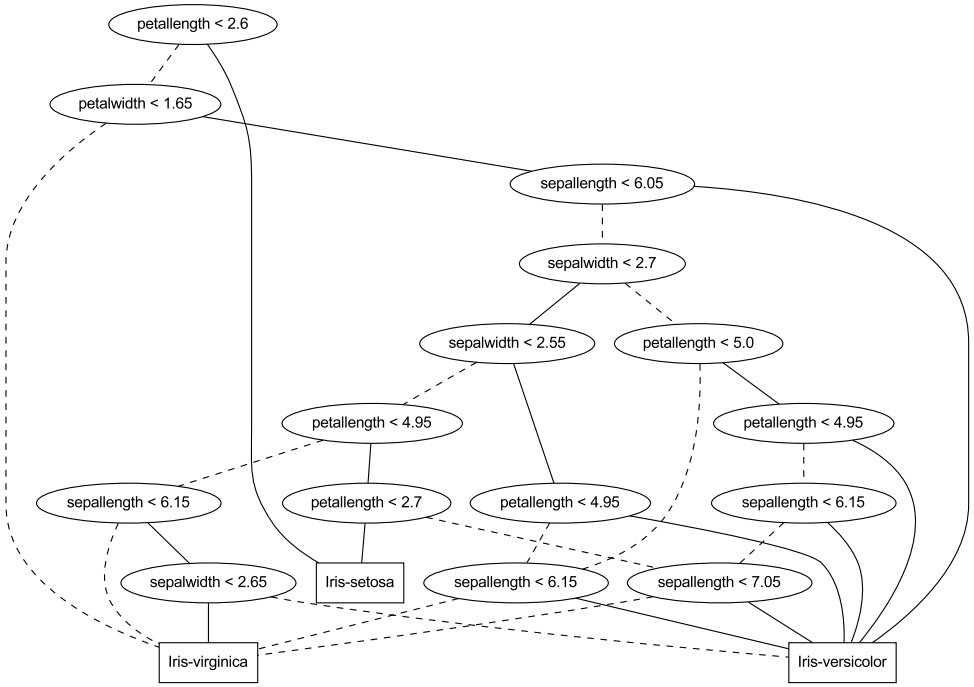

Fig. 4 Most frequent label abstraction of the aggregated Random Forest (majority vote) without semantically redundant nodes (18 nodes)

ADD is the function *true*, then the ADD satisfies the postcondition C. Otherwise, we can take one path from the root to the *false* leaf to obtain a concrete counterexample.

For a concrete verification example let

$$\phi = (petallength < 5.3) \wedge (petalwidth < 1.67)$$

be the precondition and

$$C = \{Iris - setosa, Iris - versicolor\}$$

the postcondition.

By applying the infeasible path elimination with ϕ as the precondition we obtain the ADD in Fig. 13. If we apply the function $\Delta_{post}(C)$ to this ADD, the resulting ADD consists only of the *true* node as all classes in Fig. 13 are contained in the postcondition C. Therefore one can conclude that, given the precondition ϕ, the ADD satisfies the postcondition C.

6.2 Equivalence checking

In this section we provide a method for checking the equivalence of ADDs. As noted in [12] applying the infeasible path reduction does not preserve canonicity. Thus, we cannot check the semantic equivalence of two ADDs by checking whether they are structurally identical. But, similar to the class characterization, we can create an ADD that *characterizes* all paths in which two ADDs agree respectively disagree. We define the function $\delta_{\text{eq}} : C^2 \to \mathbb{B}$ as follows

$$\delta_{\text{eq}}(c, c') := \begin{cases} true, & \text{if } c = c' \\ false, & \text{otherwise} \end{cases}$$

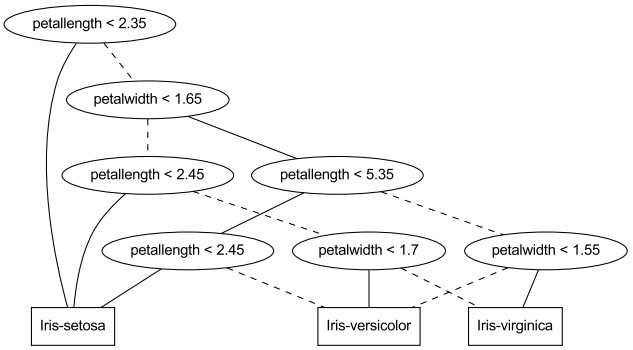

Fig. 5 The model explanation for a Random Forest with two decision trees

The function δ_{eq} can be lifted to operate on ADDs, yielding $\Delta_{\text{eq}} : \mathcal{D}_C^2 \to \mathcal{D}_{\mathbb{B}}$. The application of δ_{eq} produces a BDD which precisely characterizes the differences between two ADDs.

If we remove all infeasible paths of this BDD using the infeasible path elimination from Sect. 4 and the BDD consists of a single leaf, the *true* node, the ADDs are equivalent. Otherwise, we can follow one path from the root to the false leaf to construct a concrete input for which the two ADDs disagree.

For a concrete example consider Fig. 5 showing the model explanation for a Random Forest consisting of two decision trees and Fig. 6 which results from extending this ADD by adding the effect of a third decision tree. The infeasible path elimination for the ADD in Fig. 6 has been applied after each aggregation step. More concretely, after the aggregation of

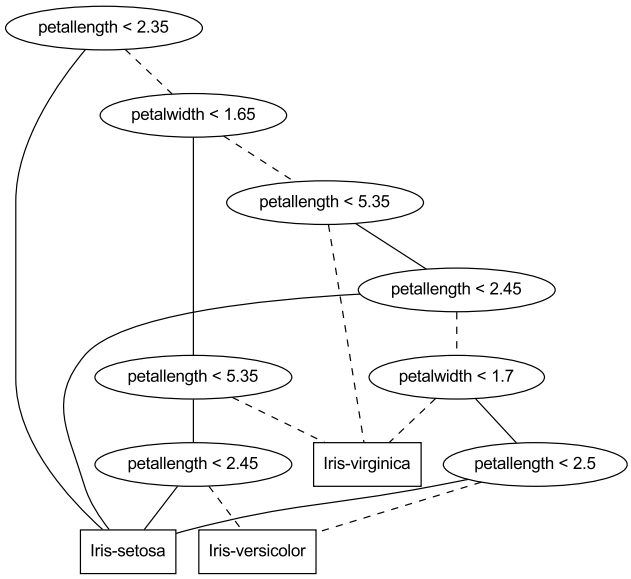

Fig. 6 The model explanation for a Random Forest with three decision trees

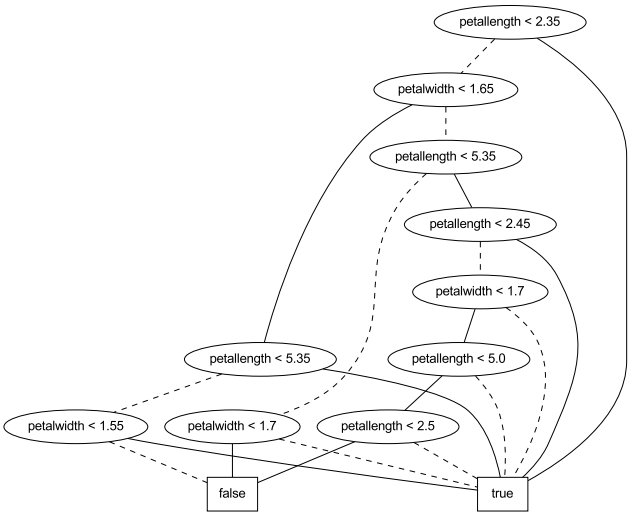

Fig. 7 The equivalence ADD for the ADDs of Fig. 5 and 6 after infeasibility elimination

the first two ADDs, corresponding to the first two decision trees, the infeasible path elimination is applied and the result is then aggregated with the third ADD, corresponding to the third decision tree, where the infeasible paths are again eliminated. After infeasibility elimination the equivalence ADD of these two ADDs precisely identifies the impact of the third decision tree (cf. Fig. 7): all paths that end in the *true* leaf are paths that the two Random Forests agree on and correspondingly, all paths that end in the *false* leaf are paths the Random Forests disagree on.

7 Forest GUMP: structural overview

Forest GUMP[3] (Generalized Unifying Merge Process) is a tool we developed to illustrate the power of algebraic aggregation for the optimization, verification and explanation of Random Forests. It is designed to allow everyone, in particular people without IT or machine learning knowledge, to experience the nature of Random Forests. To avoid unnecessary entry hurdles, we decided to implement Forest GUMP as a simple-to-use web application. It allows the user to experience the methods described in the previous sections and the proposed solutions to verification and explainability problems which will be illustrated in the following section.

Forest GUMP's user interface (see Fig. 8) is essentially divided into two parts. On the left side the user can input the necessary data to learn a Random Forest and subsequently visualize it while the currently chosen representation will be visualized on the right side. First, the user has to upload a dataset or choose one of six datasets that we provide (cf. (1) in Fig. 8) on which the Random Forest will be learned. Next, the hyperparameters necessary for the learning procedure have to be selected, such as the number of trees to be learned (cf. (2) in Fig. 8). Then, one can choose different aggregation methods, i.e. the ones mentioned in the previous sections and further ones which will be explained in the following Sections (cf. (3) in Fig. 8). Further, there is a checkbox to enable the elimination of infeasible paths for the selected aggregation method and another checkbox to specify when elimination takes place, i.e. stepwise or after merging all ADDs. It is also possible to input a sample, classify it with the ADD and highlight the path from the root to the leaf (satisfied predicates are highlighted in green, unsatisfied predicates are highlighted in red). Moreover, the area provides two further buttons that allow users to check the equivalence between two ADDs and then offer them a pre-post-condition-based verification. In the end, the currently visualized ADD can be exported as an executable classifier since Forest GUMP provides code generators for Java, C++, Python and GraphViz's dot format (cf. (4) in Fig. 8). Additionally, the currently visualized ADD can be exported as an SVG to be viewed locally (cf. (4) in Fig. 8).

The grey rectangle (cf. (5) in Fig. 8) points to the root of the currently visualized ADD. One can zoom into/out which can be helpful when the ADDs are rather large (cf. (6) in Fig. 8). On the top left the number of nodes and the length of the currently highlighted path are displayed (cf. (7) in Fig. 8). On the bottom right, one can open a history of all the representations one chose to visualize (cf. (8) in Fig. 8).

[3] A link to a running instance of Forest GUMP is available at https://gitlab.com/scce/forest-gump.

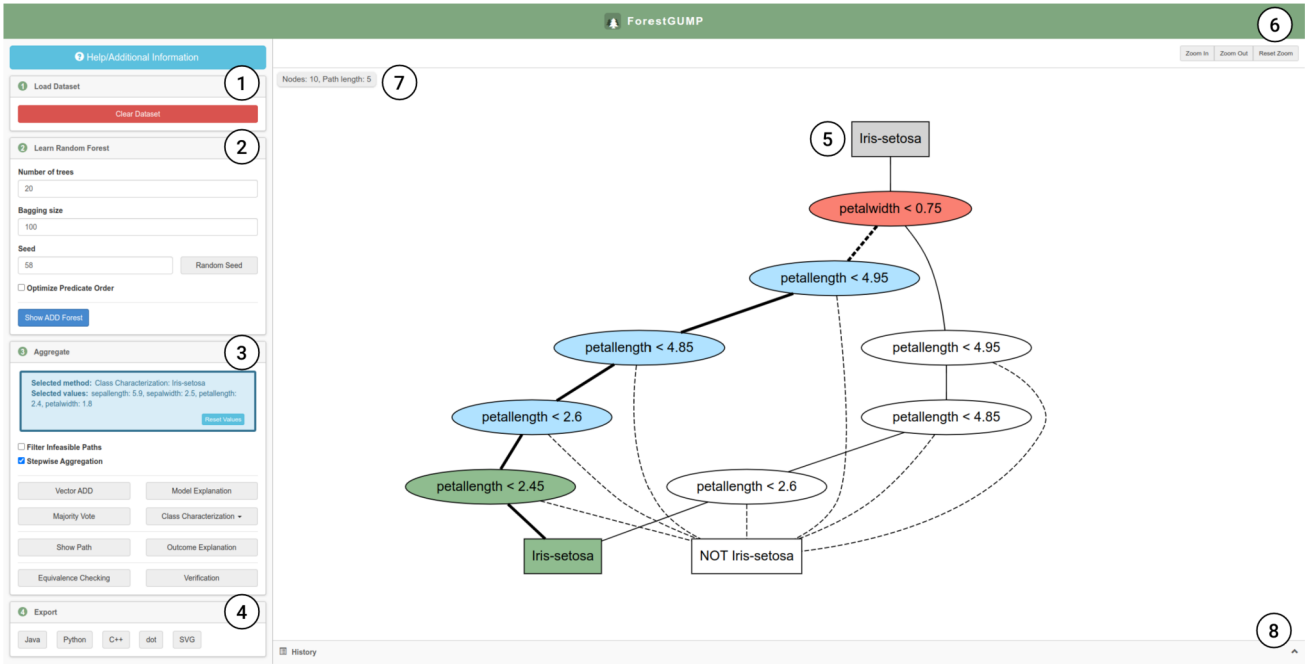

Fig. 8 Overview of Forest GUMP. The visualized ADD is our solution to the class characterization problem (cf. Sect. 8.3) for the class Iris-Setosa (10 nodes, highlighted path of length 5) (Color figure online)

Fig. 9 The execution history in Forest GUMP: The user can re-execute previous setups and export the history as CSV

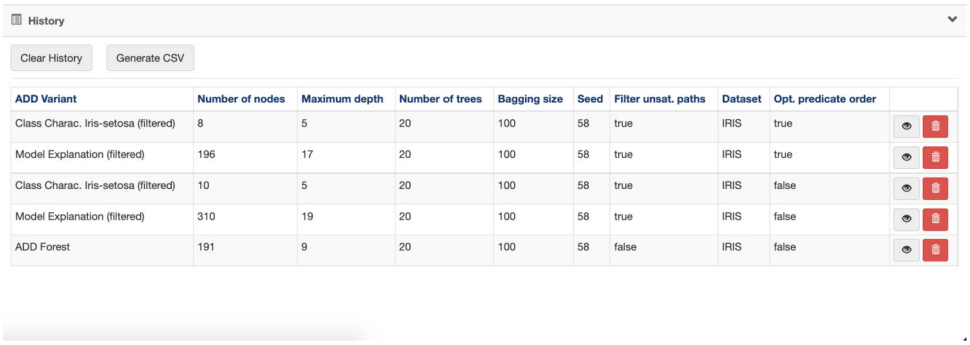

Figure 9 shows the expanded execution history. For each visualized ADD, the execution history lists the aggregation variant, the hyperparameters used to learn the Random Forest and the size (i.e. the number of nodes) and the maximum depth which is the longest path from root to leaf. The execution history also allows one to replay an experiment by clicking on the button on the right side of a row which allows one to compare different ADD variants. One can also delete the individual entries or the whole history and export the history to a CSV.

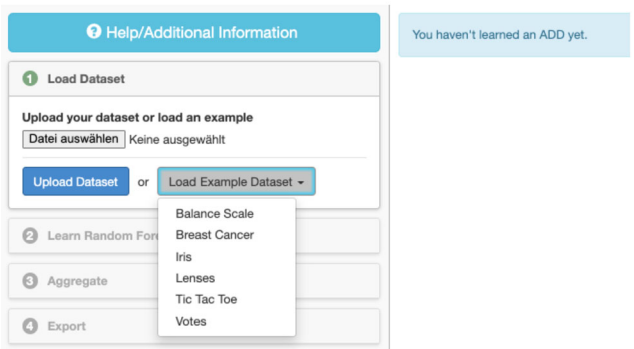

Fig. 10 Users can choose to upload their own dataset or select one of six exemplary datasets

8 Forest GUMP in action

In the following, we will see how complicated it is to understand how a Random Forest comes to its decision and illustrate the effects of the three explainability problems and the introduced verification methods.

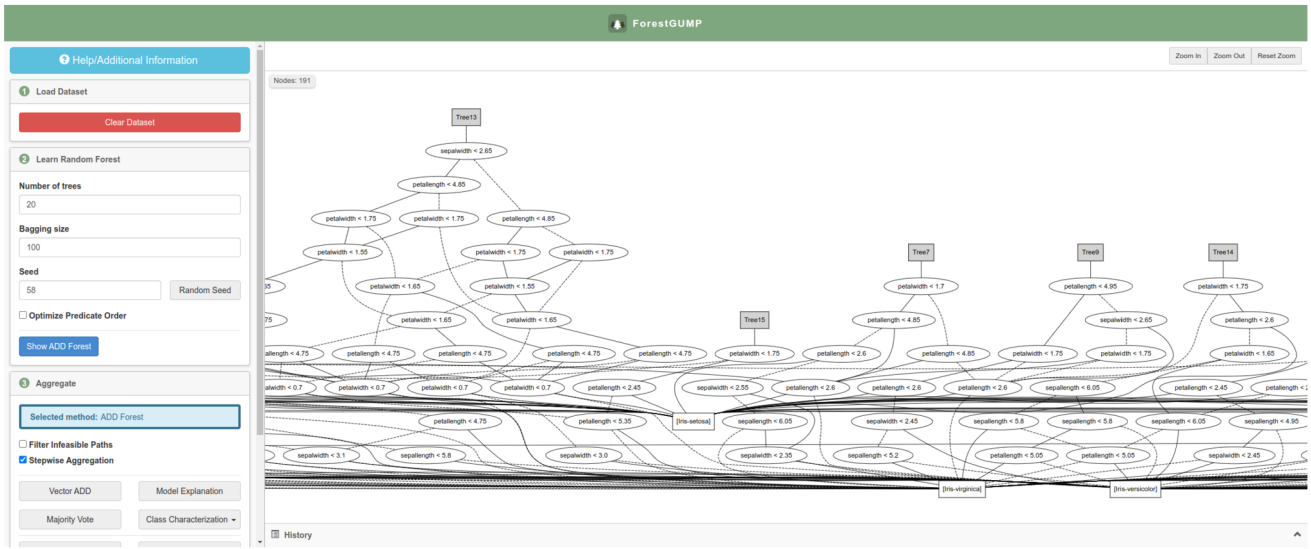

Fig. 11 A Random Forest consisting of 20 individual decision trees (191 number of nodes, the longest path consists of 9 nodes). Note that each decision tree is represented as an ADD and that all ADDs share common subfunctions, i.e. it is essentially a shared ADD forest. The actual Random Forest, where nothing is shared, contains 284 nodes

8.1 Learning a Random Forest

To begin, we need a Random Forest which requires a dataset on which it will be learned. In Forest GUMP, the user can upload their own dataset in the Attribute-Relation File Format (ARFF) [39]. Alternatively, we provide six exemplary datasets from which a user can select one to directly start using the tool. Figure 10 illustrates how this looks like in Forest GUMP.

Having chosen a dataset, next, the hyperparameters necessary for the learning procedure of the Random Forest have to be specified (see Fig. 11). The inputs are the following:

– the *number of trees* to be learned,
– the *bagging size*, i.e. the fraction of samples to be used to learn each tree and
– a *seed* to be able to reproduce the setting.

Additionally, the user can decide to eliminate the infeasible paths as this can strongly reduce the size of the ADDs (see Sect. 4). While the predicate order is fixed by default, the user can decide to let Forest GUMP optimize the predicate order as the order can also greatly impact the size of the ADDs. A more in depth discussion on the interplay between the infeasible path elimination and the predicate order will follow.

Figure 11 shows a Random Forest that was learned on the Iris dataset, consisting of 20 trees,[4] a bagging size of 100% and 58 as the seed.

If we now want to classify a given input, for each tree we would have to traverse from the root to the leaf and receive one predicted class per tree. The class which was predicted most often is the final result. Trying to understand why the Random Forest predicted this specific class is seemingly impossible. In the following we will show how we can do better.

8.2 Model explanation

A concise white-box model corresponding to the Random Forest of Fig. 11 can be constructed through the most frequent label abstraction (see Sect. 5) of the aggregated Random Forest (see Sect. 3), whose infeasible paths are eliminated (see Sect. 4). This solves the Model Explanation Problem.

Figure 12 sketches the result of this construction: A white-box model with 310 nodes. Admittedly, this model is still frightening, but given a sample, it allows one to easily follow the corresponding classification process, and in this case it may require at most 19 individual decisions based on the petal and sepal characteristics. This decision set is our set of predicates. The conjunction of these predicates is a solution to the Outcome Explanation Problem. However, more concise explanations are derived from the class characterization BDD discussed in the following section.

Given the sample $petallength = 2.4$, $petalwidth = 1.8$, $sepallength = 5.9$, $sepalwidth = 2.5$, the outcome explanation given by the model explanation consists of the following 9 predicates (in Fig. 12 satisfied predicates are highlighted

[4] Note that this is a shared representation where each decision tree is represented by its own root in the shared ADD structure.

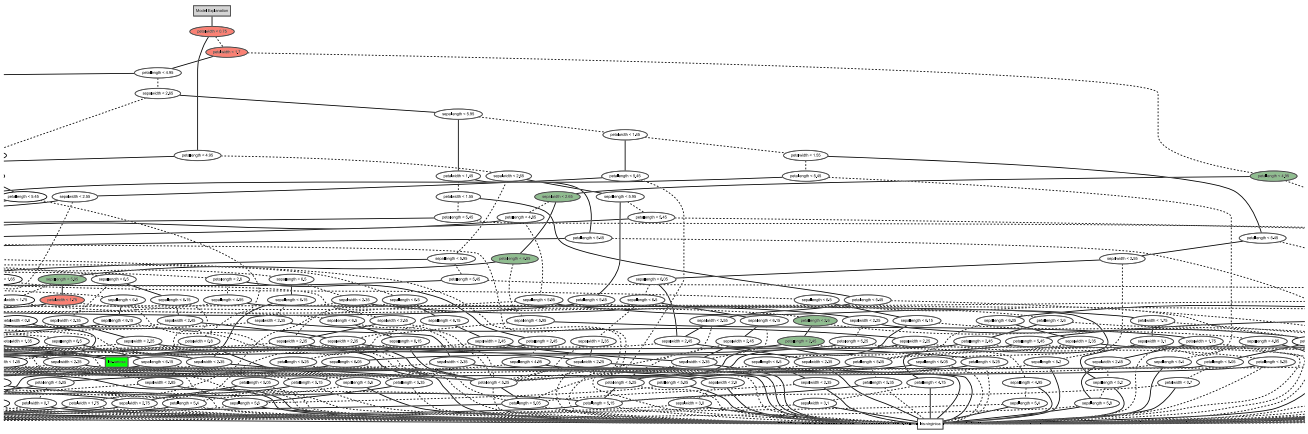

Fig. 12 An extract of the model explanation. The ADD is constructed from the most frequent label abstraction of the aggregated Random Forest following an elimination of all infeasible paths (310 nodes, the longest path with length 19, the highlighted path has a length of 9) (Color figure online)

in green, unsatisfied predicates are highlighted in red):

$$\begin{aligned}&\neg(petalwidth < 0.75) \wedge \neg(petalwidth < 1.7) \wedge\\ &(petallength < 4.95) \wedge (sepalwidth < 2.65) \wedge\\ &(petallength < 4.85) \wedge (sepallength < 5.95) \wedge\\ &\neg(petalwidth < 1.75) \wedge (petallength < 2.6) \wedge\\ &(petallength < 2.45)\end{aligned}$$

While this is already an improvement compared to the Random Forest, where you would have to traverse all 20 decision trees, we will see how we can improve even more in the following.

8.3 Class characterization

The class characterization problem is particularly interesting because it allows one to 'reverse' the classification process. While the direct problem is 'given a sample, provide its classification', the reverse problem sounds 'given a class, what are the characteristics of all the samples belonging to this class?'

BDD-based Class Characterization can be defined via the following simple transformation function: Given a class $c \in C$, we define a corresponding projection function $\delta_B(c) : C \to \mathbb{B}$ on the co-domain as

$$\delta_B(c)(c') := \begin{cases} 1 & \text{if } c' = c \\ 0 & \text{otherwise,} \end{cases}$$

for $c' \in C$. Again, the function $\delta_B(c)$ can be lifted to operate on ADDs, yielding $\Delta_B(c) : \mathcal{D}_C \to \mathcal{D}_{\mathbb{B}}$.

The BDD shown in Fig. 8 is a minimal characterization of the set of all the samples that are guaranteed to be classified as Iris-Setosa.

Remark: Being able to reverse a learned classification function has a major practical importance. Think, e.g., of a marketing research scenario where data have been collected with the aim to propose best fitting product offers to customers according to their user profile. This scenario can be considered as a classification problem where the offered product plays the role of the class. Now, being able to reverse the customer → product classification function provides the marketing team with a tailored product → customer promotion process: for a given product, it addresses all customers considered to favor this very product as in the corresponding patent [20].

The path highlighted in Fig. 8 is the path from the root to the leaf for the same sample $petallength = 2.4$, $petalwidth = 1.8$, $sepallength = 5.9$, $sepalwidth = 2.5$. Here nodes are colored green in case the predicate holds and red otherwise. Compared to the path with length 9 in the model explanation, we now have a path of length 5 with the following predicates:

$$\begin{aligned}&\neg(petalwidth < 0.75) \wedge (petallength < 4.95) \wedge\\ &(petallength < 4.85) \wedge (petallength < 2.6) \wedge\\ &(petallength < 2.45)\end{aligned}$$

8.4 Outcome explanation problem

The previous classification formula expresses the collection of 'conditions' that this sample satisfies, and it provides therefore a precise justification why it is classified in this class. Despite the fact that class characterization BDDs are concise in a global context it is easy to see that there are some redundancies in the formula of this specific local path. For example, a $petallength < 2.45$ is also inherently smaller than 2.6, 4.85 and 4.95; therefore, for this specific sample those three predicates are redundant. This is the result of the imposed predicate ordering in BDDs: all the BDD predicates

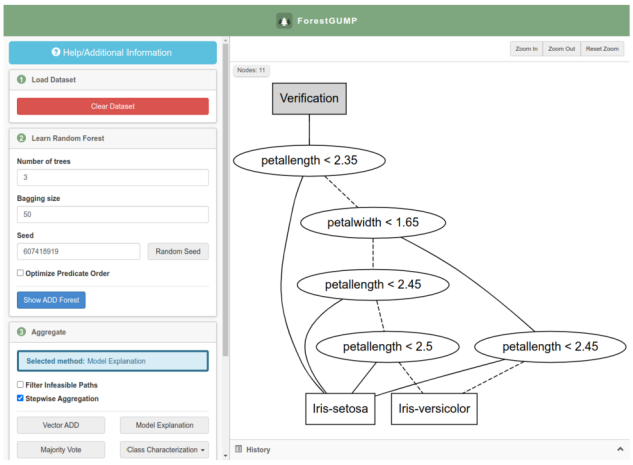

Fig. 13 ADD of Fig. 6 after infeasible path elimination with precondition $\phi = (petallength < 5.3) \wedge (petalwidth < 1.67)$ and postcondition $C = \{\text{Iris-setosa, Iris-versicolor}\}$

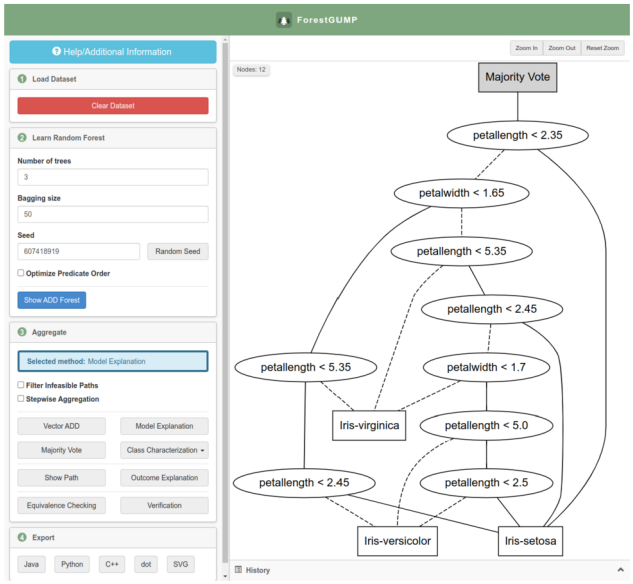

Fig. 14 The model explanation for the same Random Forest of Fig. 6 where infeasible paths are eliminated only after the aggregation of all three decision trees

are listed, and they are listed in a fixed order. After eliminating these redundancies, we are left with the following precise minimal outcome explanation: this sample is recognized as belonging to the class Iris-Setosa because it has the properties

$$\neg(petalwidth < 0.75) \wedge (petallength < 2.45).$$

In Forest GUMP we make these redundant predicates explicit by highlighting them in blue (see Fig. 8). From 9 predicates in the model explanation to 5 predicates in the class characterization, we have now arrived at an explanation that only consists of 2 predicates.

8.5 Verification

When looking at a certain classification result, one question is central: how stable is this result, i.e., would the classification be the same if the input values would marginally be changed? Pre/post-condition-based verification is an ideal tool to answer this question, and our model explanation-based approach is particularly well-suited: the model explanation projected onto a precondition that expresses the desired stability interval directly specifies all adversarial examples. This is the optimal information for judging whether the classification is critical. In the example ADD shown in Fig. 13 one immediately sees, e.g., that Iris-setosa is the right class unless the petallength is greater than 2,45.

Considering equivalence checking: In Sect. 6.2, we saw a model explanation characterizing the semantic difference between two given ADDs. In order to have an example for a successful equivalence check let us consider the ADD in Fig. 14. This ADD corresponds to the same Random Forest as the one shown in Fig. 6. The only difference is that, this time, the infeasible path elimination has been applied after the aggregation of all three decision trees. This is in contrast to the ADD in Fig. 6 where the infeasibility elimination has been applied iteratively, i.e. after merging the first two ADDs corresponding to the first two decision trees and again after merging the result with the third one.

As the infeasible path elimination is semantics preserving, the resulting ADDs of Fig. 6 and 14 are semantically equivalent. One the other hand, we can easily see that structurally, they are not equal. This illustrates our observation that infeasible path elimination is not canonical (cf. Sect. 4). We can, however, easily confirm their semantic equivalence by first constructing the equivalence ADD and a subsequently eliminating all infeasible paths (cf. Sect. 6.2) which, as desired, results in the ADD consists only of a single terminal node *true*.

9 Conclusion and perspectives

In this paper, which is an extended version of [27], we have presented Forest GUMP, a tool for verification and precise explanation of Random forests. Novel contributions are the verification of pre/post-conditions and equivalence checking. Both heavily rely on the complete elimination of infeasible paths which is therefore treated in detail in this paper. Technically, verification is treated as special cases of model explanation:

– Pre-post-condition-based verification is reduced to model explanation projected onto the part that is consistent with the precondition combined with a check whether all remaining leaves satisfy the postcondition.

– Equivalence checking is done by introducing an equality relation on ADDS that produces a BDD whose true leaves characterizes all paths where the classification agree, while the false leaves characterizes all cases where the classification disagrees.

Besides pre/post-condition-based verification and equivalence checking, Forest GUMP also supports three concepts of explanation, the well-known *model explanation* and *outcome explanation*, as well as *class characterization*, i.e., the precise characterization of all samples that are equally classified.

Forest GUMP is designed to provide even non-technical people with a tangible experience with the validation of machine learned models and its limitations. As Forest GUMP is publicly available all experiments can be reproduced, modified, and complemented using any data set that is available in the ARFF format [38].

Playing with Forest GUMP led to interesting observations about Random Forest Learning: Changing the random seed for the learning process had a significant impact on the size of the explanation models and the class characterizations. The observed sizes of the explanation models ranged from 138 to 519. Interesting was that the larger sizes did not necessarily imply a better prediction quality. The same also applied to the class characterizations. In fact, we observed a 100% prediction quality for a class characterization of only 3 nodes, while a class characterization for the same species with 40 nodes only scored 33% prediction.

The impact of infeasible path elimination is enormous. As reported in [12], forests with 10.000 trees can be handled when unsatisfiable path are eliminated. Otherwise, treating 100 trees is quite problematic. Our newest experience with a similar approach for Neural Networks [28, 34] confirms the importance of infeasible path.

Of course, these are first steps in a very ambitious new direction and it has to be seen how far the approach carries. Scalability will probably require decomposition methods, perhaps in a similar fashion as illustrated by the difference between model explanation and the considerably smaller class characterization. More work is needed also on techniques that aim at dealing with large numbers of predicates.

Funding Note Open Access funding enabled and organized by Projekt DEAL.

Data Availability The artifact is available in the Zenodo repository [26].

References

1. Akers, S.B.: Binary decision diagrams. IEEE Trans. Comput. **27**(6), 509–516 (1978)
2. Bahar, R., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Pardo, A., Somenzi, F.: Algebraic decision diagrams and their applications. In: Proceedings of 1993 International Conference on Computer Aided Design (ICCAD), pp. 188–191 (1993). https://doi.org/10.1109/ICCAD.1993.580054
3. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
4. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. Comput. **35**(8), 677–691 (1986). https://doi.org/10.1109/TC.1986.1676819
5. Chen, H., Zhang, H., Si, S., Li, Y., Boning, D.S., Hsieh, C.: Robustness verification of tree-based models. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 12317–12328 (2019). https://papers.nips.cc/paper/2019/hash/cd9508fdaa5c1390e9cc329001cf1459-Abstract.html
6. Chipman, H.A., George, E.I., McCulloh, R.E.: Making sense of a forest of trees. In: Weisberg, S. (ed.) Proceedings of the 30th Symposium on the Interface, pp. 84–92. Interface Foundation of North America, Fairfax Station, VA (1998)
7. Deng, H.: Interpreting tree ensembles with inTrees. Int. J. Data Sci. Anal. **7**(4), 277–287 (2019). https://doi.org/10.1007/s41060-018-0144-8
8. Domingos, P.M.: Knowledge discovery via multiple models. Intell. Data Anal. **2**(1–4), 187–202 (1998). https://doi.org/10.1016/S1088-467X(98)00023-7
9. Einziger, G., Goldstein, M., Sa'ar, Y., Segall, I.: Verifying robustness of gradient boosted models. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 2446–2453. AAAI Press, New York (2019). https://doi.org/10.1609/aaai.v33i01.33012446
10. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(1), 3133–3181 (2014)
11. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**(2), 179–188 (1936)
12. Gossen, F., Steffen, B.: Algebraic aggregation of random forests: towards explainability and rapid evaluation. Int. J. Softw. Tools Technol. Transf. (2021). https://doi.org/10.1007/s10009-021-00635-x
13. Gossen, F., Margaria, T., Murtovi, A., Naujokat, S., Steffen, B.: Dsls for decision services: a tutorial introduction to language-driven engineering. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Modeling - 8th International Symposium, Proceedings, Part I, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018. Lecture

Notes in Computer Science, vol. 11244, pp. 546–564. Springer, Berlin (2018). https://doi.org/10.1007/978-3-030-03418-4_33
14. Gossen, F., Margaria, T., Steffen, B.: Towards explainability in machine learning: the formal methods way. IT Prof. **22**(4), 8–12 (2020). https://doi.org/10.1109/MITP.2020.3005640
15. Gossen, F., Margaria, T., Steffen, B.: Formal methods boost experimental performance for explainable AI. IT Prof. **23**(6), 8–12 (2021). https://doi.org/10.1109/MITP.2021.3123495.
16. Gossen, F., Murtovi, A., Linden, J., Steffen, B.: The java library for algebraic decision diagrams. https://add-lib.scce.info. Accessed 2023-02-22
17. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 93 (2019). https://doi.org/10.1145/3236009
18. Hara, S., Hayashi, K.: Making tree ensembles interpretable: a Bayesian model selection approach. In: Storkey, A.J., Pérez-Cruz, F. (eds.) International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain. PMLR Proceedings of Machine Learning Research, vol. 84, pp. 77–85. (2018). http://proceedings.mlr.press/v84/hara18a.html
19. Ho, T.K.: Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282 (1995). https://doi.org/10.1109/ICDAR.1995.598994
20. Hungar, H., Steffen, B., Margaria, T.: Methods for generating selection structures, for making selections according to selection structures and for creating selection descriptions. USPTO Patent number: 9141708 (Sep 2015). https://patents.justia.com/patent/9141708
21. Kantchelian, A., Tygar, J.D., Joseph, A.D.: Evasion and hardening of tree ensemble classifiers. In: Balcan, M., Weinberger, K.Q. (eds.) Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 JMLR Workshop and Conference Proceedings, vol. 48, pp. 2387–2396 (2016). http://proceedings.mlr.press/v48/kantchelian16.html
22. Lee, C.Y.: Representation of switching circuits by binary-decision programs. Bell Syst. Tech. J. **38**(4), 985–999 (1959)
23. Lou, Y., Caruana, R., Gehrke, J.: Intelligible models for classification and regression. In: Yang, Q., Agarwal, D., Pei, J. (eds.) The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012 pp. 150–158. ACM, New York (2012). https://doi.org/10.1145/2339530.2339556
24. Mangla, P., Singh, V., Balasubramanian, V.N.: On saliency maps and adversarial robustness. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 272–288. Springer, Berlin (2020)
25. Mundhenk, T.N., Chen, B.Y., Friedland, G.: Efficient saliency maps for explainable ai. Arxiv preprint (2019). arXiv:1911.11293
26. Murtovi, A., Bainczyk, A., Steffen, B.: Forest gump: a tool for explanation (tacas 2022 artifact) (Nov 2021). https://doi.org/10.5281/zenodo.5733107
27. Murtovi, A., Bainczyk, A., Steffen, B.: Forest GUMP: a tool for explanation. In: Fisman, D., Rosu, G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, Proceedings, Part II, ETAPS 2022, Munich, Germany, April 2-7, 2022. Lecture Notes in Computer Science, vol. 13244, pp. 314–331. Springer, Berlin (2022). https://doi.org/10.1007/978-3-030-99527-0_17.
28. Nolte, G., Schlüter, M., Murtovi, A., Bernhard, S.: The power of Typed Affine Decision Structures: a case study. Int. J. Softw. Tools Technol. Transf. (2023, in this issue). https://doi.org/10.1007/s10009-023-00701-6
29. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
30. Ranzato, F., Zanella, M.: Abstract interpretation of decision tree ensemble classifiers. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 5478–5486. AAAI Press, New York (2020). https://ojs.aaai.org/index.php/AAAI/article/view/5998
31. Ribeiro, M.T., Singh, S., Guestrin, C.: "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, pp. 1135–1144 (2016)
32. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": explaining the predictions of any classifier. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016 pp. 1135–1144. ACM, New York (2016). https://doi.org/10.1145/2939672.2939778
33. Sato, N., Kuruma, H., Nakagawa, Y., Ogawa, H.: Formal verification of decision-tree ensemble model and detection of its violating-input-value ranges. CoRR (2019). arXiv:1904.11753
34. Schlüter, M., Nolte, G., Murtovi, A., Bernhard, S.: Towards rigorous understanding of Neural Networks via semantics-preserving transformations. Int. J. Softw. Tools Technol. Transf. (2023, in this issue). https://doi.org/10.1007/s10009-023-00700-7
35. Steffen, B., Gossen, F., Naujokat, S., Margaria, T.: Language-Driven Engineering: From General-Purpose to Purpose-Specific Languages, pp. 311–344. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91908-9_17
36. Törnblom, J., Nadjm-Tehrani, S.: Formal verification of random forests in safety-critical applications. In: Artho, C., Ölveczky, P.C. (eds.) Formal Techniques for Safety-Critical Systems - 6th International Workshop, FTSCS 2018, Gold Coast, Australia, November 16, 2018, Revised Selected Papers. Communications in Computer and Information Science, vol. 1008, pp. 55–71. Springer, New York (2018). https://doi.org/10.1007/978-3-030-12988-0_4
37. Van Assche, A., Blockeel, H.: Seeing the forest through the trees: learning a comprehensible model from an ensemble. In: Kok, J.N., Koronacki, J., Mantaras, R.L.D., Matwin, S., Mladenič, D., Skowron, A. (eds.) Machine Learning: ECML 2007, pp. 418–429. Springer, Berlin (2007)
38. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: Practical Machine Learning Tools and Techniques, vol. 2 (2005)
39. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques 4th edn. Morgan Kaufmann, San Francisco (2016)
40. Zhou, Y., Hooker, G.: Interpreting Models via Single Tree Approximation (2016)