Calzavara *et al. EURASIP Journal on Information Security* (2021) 2021:12
https://doi.org/10.1186/s13635-021-00127-0

EURASIP Journal on Information Security

RESEARCH Open Access

Feature partitioning for robust tree ensembles and their certification in adversarial scenarios

Stefano Calzavara, Claudio Lucchese, Federico Marcuzzi* and Salvatore Orlando

Abstract

Machine learning algorithms, however effective, are known to be vulnerable in adversarial scenarios where a malicious user may inject manipulated instances. In this work, we focus on evasion attacks, where a model is trained in a safe environment and exposed to attacks at inference time. The attacker aims at finding a perturbation of an instance that changes the model outcome.

We propose a model-agnostic strategy that builds a robust ensemble by training its basic models on feature-based partitions of the given dataset. Our algorithm guarantees that the majority of the models in the ensemble cannot be affected by the attacker. We apply the proposed strategy to decision tree ensembles, and we also propose an approximate certification method for tree ensembles that efficiently provides a lower bound of the accuracy of a forest in the presence of attacks on a given dataset avoiding the costly computation of evasion attacks.

Experimental evaluation on publicly available datasets shows that the proposed feature partitioning strategy provides a significant accuracy improvement with respect to competitor algorithms and that the proposed certification method allows ones to accurately estimate the effectiveness of a classifier where the brute-force approach would be unfeasible.

Keywords: Adversarial machine learning, Evasion attack, Forests of decision trees

1 Introduction

Machine learning (ML) algorithms are currently used to train models that are then deployed to ensure system security and to control critical processes [1, 2]. Unfortunately, traditional ML algorithms proved vulnerable to a wide range of attacks, and in particular to *evasion attacks*, where an attacker carefully crafts perturbations of an input to force prediction errors [3–6].

While there is a large body of research on evasion attacks in linear classifiers [7, 8] and, more recently, on deep neural networks [9, 10], there are just a few works dealing with tree-based models. Decision trees are *interpretable* models [11], yielding predictions which are human-understandable in terms of syntactic checks over domain features, which is particularly appealing in the security setting. Moreover, *decision trees ensembles* are nowadays one of the best methods for dealing with non-perceptual problems and are one of the most commonly used techniques in Kaggle competitions [12].

In this paper, we present a meta-learning algorithm, called *Feature Partitioned Forest* (FPF), that builds an ensemble of decision trees aimed to be robust against evasion attacks. By focusing on binary classifiers, we show that the proposed algorithm is able to build models that are *robust by construction*. In fact, the algorithm benefits of the theoretical property that the majority of the learners in the ensemble is not fooled by an injected perturbation over a given instance, i.e., they still provide the same prediction as for the original instance. This means that the proposed algorithm limits by design the drop in accuracy an attacker may generate. Our method is based on a sampling of the features at training time. Specifically, we randomly equi-partition the set of features and train

*Correspondence: federico.marcuzzi@unive.it
Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venice, Italy

each tree of the ensemble on a distinct feature partition. Such sampling limits the number of features considered by a single tree, and therefore, it limits the number of trees affected by corrupting a given feature. Even if we focus our analysis on ensembles of decision trees, the proposed learning strategy generalizes to any ensemble method.

We also propose a *certification method* for tree ensembles that efficiently computes a lower bound of accuracy under attack on a given dataset. Finding a successful attack requires to find a set of features of a given instance that, if corrupted, affects the prediction of the model. In general, this requires an exhaustive search of all the possible feature subsets and their possible modifications, which becomes quickly unfeasible for powerful attacks or for large feature sets. We show that this problem can be reduced to a *partial set cover problem*, and we use efficient set cover algorithms to assess the non-existence of harmful attacks. By certifying the non-existence of attacks for some instances of a given dataset, we can provide an accuracy lower bound on the full dataset. Indeed, we show that the proposed lower bounds are efficient to be computed and quite accurate. Furthermore, we devise a *cascading* strategy where only those instances that are not certified as non-attackable by the proposed lower bounds are eventually evaluated with a slow exact method. The *cascading* strategy can reduce the total running time by one or two orders of magnitude.

In summary, the contributions of this work are as follows:

- We propose a novel *meta tree ensemble learning algorithm* named *Feature Partitioned Forest* that is robust against adversarial attacks by construction
- We experimentally validate FPF on three public datasets and we show that FPF can provide an improvement up to 16 points in terms of accuracy in presence of attacks compared to competitor algorithms
- We provide a novel *certification method* that allows ones to quickly compute a lower bound of the accuracy of a forest on a given dataset
- We devise an exact *cascading* strategy for exactly computing the accuracy in presence of an attacker by exploiting the proposed lower bounds as a fast preprocessing filter

2 Background and related work

Most of the work in adversarial learning regards classifiers, in particular binary ones. The attacker starts from a positive instance that is classified correctly by the deployed ML model and is interested in introducing minimal perturbations on the instance to modify the prediction from positive to negative, thus "evading" the classifier [3, 10, 13–18]. To prevent these attacks, different techniques have been proposed for different models, including support vector machines [8, 19], deep neural networks [10, 20, 21], and decision tree ensembles [16, 22–24].

2.1 Background and notation

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimensional vector space of real-valued *features*. An *instance* $\boldsymbol{x} \in \mathcal{X}$ is a d-dimensional feature vector $(x_1, x_2, \ldots, x_d)$, where we denote with $\mathcal{F}$ the set of features. Each instance $\boldsymbol{x} \in \mathcal{X}$ is assigned a label $y \in \mathcal{Y}$ by some unknown *target* function $g : \mathcal{X} \mapsto \mathcal{Y}$. The goal of a *supervised learning* algorithm that induces a *forest of decision trees* is to find the forest $\mathcal{T}$ that best approximates the target g. At training time, learning algorithms exploit a training set $\mathcal{D}_{train} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, then model accuracy is evaluated on a test set $\mathcal{D}_{test}$ where instance labels are unknown to the trained model.

In this paper, we discuss binary classification, where $\mathcal{Y} = \{-1, +1\}$, and focus on binary decision trees. Each tree $t \in \mathcal{T}$ can be inductively defined as follows: t is either a leaf $\lambda(\hat{y})$ for some label $\hat{y} \in \mathcal{Y}$, or an internal node $\sigma(f, v, t_l, t_r)$ (a.k.a, split), where $f \in [1, d]$ identifies a feature, $v \in \mathbb{R}$ is the threshold for the feature f, and t_l, t_r are left/right decision trees. At test time, an instance $\boldsymbol{x}$ traverses each tree $t \in \mathcal{T}$ until it reaches a leaf $\lambda(\hat{y})$, which returns the *prediction* $\hat{y}$, denoted by $t(\boldsymbol{x}) = \hat{y}$. Specifically, for each internal node $\sigma(f, v, t_l, t_r)$, $\boldsymbol{x}$ falls into the left tree t_l if $x_f \leq v$, and into the right tree t_r otherwise. Given a forest $\mathcal{T}$, the global prediction is defined as $\mathcal{T}(\boldsymbol{x}) = +1$ if $\sum_{t \in \mathcal{T}} t(\boldsymbol{x}) > 0$, and $\mathcal{T}(\boldsymbol{x}) = -1$ otherwise.

Finally, given a test set $\mathcal{D}_{test}$, we denote by $\overline{\mathcal{D}}_{test}$, $\overline{\mathcal{D}}_{test} \subseteq \mathcal{D}_{test}$, the set of test instances that are erroneously classified by $\mathcal{T}$, i.e., $\overline{\mathcal{D}}_{test} = \{(\boldsymbol{x}, y) \in \mathcal{D}_{test} \mid y \cdot \mathcal{T}(\boldsymbol{x}) < 0\}$. We define *accuracy* as:

$$ACC(\mathcal{T}, \mathcal{D}_{test}) = 1 - \frac{|\overline{\mathcal{D}}_{test}|}{|\mathcal{D}_{test}|}.$$

2.2 Threat model

We focus on the *evasion attack* scenario, where an attacker aims at fooling an already trained classifier by maliciously modifying a given instance before submitting it to the classification model. As in Kantchelian et al. [25], we assume an attacker A_b that is capable of modifying a given instance $\boldsymbol{x}$ into a perturbed instance $\boldsymbol{x}'$ such that the L_0-norm of the perturbation is smaller than the attacker's budget b, i.e., $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq b$. Therefore, attacker A_b can perturb the instance $\boldsymbol{x}$ by modifying at most b features, without any constraint on how much a given feature can be altered. Additional threat models have been investigated. Threat models based on L_p-norm constraints are common [9], and recently, models based on rewriting rules have been proposed [23]. We restrict attention to L_0-norm attacks because of their simplicity and effectiveness.

Indeed, a very small b is sufficient to achieve successful attacks. Su et al. [26] show that with a one-pixel attack, i.e., with $b = 1$, it is possible to fool a complex deep neural network as VGG16 [27] and decrease its accuracy to a poor 16%.

Given an instance $\boldsymbol{x} \in \mathcal{X}$, we denote by $A_b(\boldsymbol{x})$ the set of all the perturbed instances the attacker may generate:

$$A_b(\boldsymbol{x}) = \left\{\boldsymbol{x}' \mid \boldsymbol{x}' \in \mathcal{X} \ \wedge \ \|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq b\right\}.$$

Finally, we can formalize the *accuracy under attack* that an attacker A_b aims to *minimize*. Given the test set $\mathcal{D}_{test}$, let $\widehat{\mathcal{D}}_{test}$ be the set of the test instances that can be successfully attacked by A_b, i.e., $\widehat{\mathcal{D}}_{test} = \{(\boldsymbol{x}, y) \in \mathcal{D}_{test} \setminus \overline{\mathcal{D}}_{test} \mid \exists \boldsymbol{x}' \in A_b(\boldsymbol{x}),\ y \cdot \mathcal{T}(\boldsymbol{x}') < 0\}$. We thus define the *accuracy under attack* as:

$$ACC_{A_b}(\mathcal{T}, \mathcal{D}_{test}) = 1 - \frac{|\overline{\mathcal{D}}_{test}| + |\widehat{\mathcal{D}}_{test}|}{|\mathcal{D}_{test}|}. \qquad (1)$$

While $\overline{\mathcal{D}}_{test}$ includes the instances that are misclassified due to classifier errors, the set $\widehat{\mathcal{D}}_{test}$ include those instances that happen to be misclassified because of the attacker perturbations. The goal of a robust training strategy is to generate models that minimize $|\widehat{\mathcal{D}}_{test}|$ without hindering the accuracy on unattacked instances, i.e., without increasing $|\overline{\mathcal{D}}_{test}|$. In the following, we use $\mathcal{D}$, thus omitting the subscripts from $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$, when it is clear from the context the dataset we are referring to.

Note that the attack and the evaluation processes need to solve the same problem: to find a successful attack in $A_b(\boldsymbol{x})$. This is a difficult and computationally expensive task due to the possibly large size of $A_b(\boldsymbol{x})$ and to the number of interactions among trees in a forest. Kantchelian et al. [25] prove that the problem of finding a successful attack is NP-complete, regardless of the L_p-norm adopted. Authors provide an exact but expensive solution based on Mixed Integer Linear Programming, and an approximate solution which may fail in discovering successful attacks. Chen et al. [28] show that verifying the robustness of a forest $\mathcal{T}$ with at most l leaves per tree has cost $\min\{O(l^{|\mathcal{T}|}), O((2|\mathcal{T}|l)^{|\mathcal{F}|})\}$ assuming $d = L_\infty$. Also, in this case, authors propose an approximate strategy to deal with such exponential cost. Unfortunately, verifying the accuracy under attack is feasible only for weaker forests of decision stumps, i.e., trees with two leaves only [24], although recent work shows that the verification problem for L_∞-attackers can be solved both effectively and efficiently using abstract interpretation [29].

The aforementioned approaches fall into the *white-box* scenario, where the attacker has complete knowledge of the attacked model. Conversely, in the *black-box* scenario, we assume the attacker can only issue classification queries to the model. But even in this case, only approximate strategies can be devised to discover possible attacks [30].

We are particularly interested in the evaluation process with the goal of determining the size of $\widehat{\mathcal{D}}$. To this end, in Section 4, we introduce some efficient *lower bounds* of ACC_{A_b}, by *upper bounding* the size of $\widehat{\mathcal{D}}$, and a *cascading* certification algorithm that resorts to a brute-force evaluation only when the lower bounds cannot guarantee the absence of successful attacks.

2.3 Robust training

In order to build robust models, two main research directions have been investigated: enriching the training dataset and modifying the optimized loss function.

Adversarial boosting by Kantchelian et al. [16] is the first adversarial learning technique for decision tree ensembles. It is an empirical data augmentation technique, borrowing from the *adversarial training* approach [9]. At each boosting round, a greedy algorithm is exploited to craft an adversarial counter-part of every instance in the training set. The next tree in the ensemble is trained on both original and perturbed instances. The need to craft new adversarial instances at each round makes this algorithm infeasible for large datasets. In addition, as for any *adversarial training* approach, adversarial instances produced at training time may not be representative of the adversarial instances occurring at test time, possibly leading to a poor accuracy.

Another adversarial learning technique for decision tree ensembles was proposed by Chen et al., who introduced the first tree learning algorithm embedding the attacker directly in the optimization problem solved upon tree construction [22]. The key idea of their approach, called *robust trees*, is to redefine the splitting strategy of the training examples at a tree node, so as to include the attacker impact. Also, in this case, the attacker behavior is only approximated. A similar solution is proposed in TREANT [23]. While building a single tree, the best split is chosen by minimizing the loss under attack without introducing any heuristic approximation. TREANT ensures that every node added to the tree does not increase the loss under attack of such tree and, eventually, multiple trees are grouped in a forest. Another step forward was made by Andriushchenko et al. [24], where they proposed an exact solution to optimize loss under attack for an ensemble of decision stumps, i.e., trees having only a root and two leaves. While the proposed approach is the first able to take into consideration the full ensemble, the use of decision stumps may limit the overall accuracy of the forest. Calzavara et al. [31] showed how it is possible to generalize the adversarial training to gradient-boosted decision trees by exploiting the knowledge of tree thresholds, thus reducing the set of possible perturbations without loss of generality. Thanks to this, they take advantage of differentiable approximations and make their optimization problem tractable for their experimental setting. More recently, Vos et al. [32] presented GROOT, an efficient algorithm for training robust decision trees. GROOT analytically calculates the adversarial impurity significantly reducing the training time, resulting in a training algorithm faster than the state of the art and with better performance in terms of adversarial accuracy on structured data. Finally, Ranzato et al. [33] proposed a genetic adversarial training algorithm called Meta-Silvae to train decision trees in order to maximize both accuracy and robustness to adversarial perturbation. The algorithm relies on a complete formal verification based on abstract interpretation.

The algorithm that is most similar to our proposal is the Random Subspace method (RSM) [34], which was successfully exploited by Biggio et al. [35] to build robust ensembles. The approach relies on the training of each single learner on a projection of the original dataset holding a random subset of its features. The rationale is given by the reduced variance of such randomized model. The algorithm FPF proposed in this work improves over RSM by exploiting projections of the dataset that provide *robustness by construction* for the majority of the trees in the ensemble, rather than random projections. In short, our approach divides the feature space in such a way as to ensure by construction that the majority of the ensemble is never involved in an attack. For example, if the attacker can modify at most any 3 features, and a model has been trained with FPF to resist this type of attacks, each combination of 3 features appears in less than half of the ensemble, thus leaving the majority of weak learners unaltered in their predictions. This is not guaranteed with RSM where each weak learner is trained on a random sub-sample of the feature space, with possible repetition among weak learners and so in the worst case, an attack on a feature can involve the entire ensemble. We highlight that FPF, similarly to RSM, is complementary to any other algorithm that can be used to train a weak learner of the final ensemble. Not only FPF can potentially be used to ensemble together decision trees, SVMs, or other models, but FPF can be exploited to create ensembles of robust learners trained with the approaches mentioned above such as *adversarial boosting* or *robust trees*. In this work, we focus on ensembles of decision trees, and we leave the investigation of other types of weak learners to future work.

3 Robust forest training

We aim to design a machine learning algorithm able to train ensembles, namely forests of binary decision trees, which are resilient to *evasion attacks*, which occur when an adversary manipulates test data to force prediction errors. Specifically, the robust forest is resilient by design in strong adversarial environments, where attackers can perturb at most b features of a test instance to deceive the learnt model and force a prediction error. In the following, we propose a novel training strategy that produces a forest $\mathcal{T}$ where the majority of its trees are not affected by the attacker A_b.

3.1 Feature partitioning

Given a partition $\mathcal{P}$ of the feature set $\mathcal{F}$ and an attacker A_b, we call $\mathcal{P}$ *robust* if the majority of its sets cannot be impacted by the attacker A_b, i.e., if the following property holds[1]:

$$\forall B \subseteq \mathcal{F}, |B| \leq b \qquad \sum_{P \in \mathcal{P}} \mathbb{1}[\, B \cap P \neq \emptyset] < \frac{|\mathcal{P}|}{2}.$$

In words, the majority of the sets in $\mathcal{P}$ do not contain any feature in B, i.e., the features perturbed by A_b, for whatever choice of B.

When $|B| \leq b$, it is straightforward to show that this property is surely satisfied if $|\mathcal{P}| \geq 2b + 1$. Consider the worst case: at most b distinct subsets of $\mathcal{P}$ can have an overlap with B, leaving the remaining $b + 1$ subset of $\mathcal{P}$ unaffected. Hereinafter, we consider only *robust* feature partitions $\mathcal{P}$ where $|\mathcal{P}| = 2b + 1$.

3.2 Robust forest

Let us consider a forest $\mathcal{T}$ that, given an attacker A_b, is built by exploiting a robust feature partition $\mathcal{P}$ as follows.

Let $\mathcal{D}$ be a training set and let $\mathcal{P}$ be a robust partition of its feature space. Given $P \in \mathcal{P}$, we call $\pi_P(\mathcal{D})$ the projection of $\mathcal{D}$ on the feature set P, i.e., the dataset obtained from $\mathcal{D}$ by discarding those features not included in P. Given a robust feature partitioning $\mathcal{P}$, it is thus possible to build a forest by training $2b + 1$ trees independently on the $2b + 1$ projections $\pi_P(\mathcal{D})$ for all $P \in \mathcal{P}$. We denote by $\mathcal{T}_\mathcal{P}$ such forest, having a number of trees equal to $|\mathcal{T}_\mathcal{P}| = |\mathcal{P}| = 2b + 1$.

The algorithm sketched above is able to build an ensemble of trees that we formally define as *Robust Forest*.

Definition 1 (Robust Forest) *Given an attacker A_b, we say that a forest $\mathcal{T}$ is robust if the majority of its trees is not affected by A_b for any of its attacks:*

$$\forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{x}' \in A_b(\boldsymbol{x}) \quad \sum_{t \in \mathcal{T}} \left(\mathbb{1}[\, t(\boldsymbol{x}) = t(\boldsymbol{x}')] \right) > \frac{|\mathcal{T}|}{2}.$$

We highlight that the above definition is very general as it is dataset independent. We define robustness for any given instance $\boldsymbol{x}$ in the feature space, and not for a specific train or test set. It is straightforward to show that in a forest $\mathcal{T}_\mathcal{P}$ at most b of its $2b + 1$ trees can be affected by the attacker, and thus, the *robustness* property is guaranteed by design.

[1] $\mathbb{1}[\, e]$ equals 1 if expression e is true and 0 otherwise.

In the best-case scenario where each $t \in \mathcal{T}_{\mathcal{P}}$ is perfectly accurate, the above robustness property ensures that, in the presence of attacks, only a minority of trees provides an incorrect prediction, and therefore, the forest is perfectly accurate even under attack. Clearly, this scenario is unlikely, and therefore, we discuss below how to strengthen the accuracy of $\mathcal{T}_{\mathcal{P}}$.

Note that the above definition and training strategy trivially generalize to any ensemble learning algorithm besides tree-based classifiers.

3.3 Increasing the accuracy of a robust forest

The above definition guarantees that the majority of trees in a robust forest is not affected by the attacker but it does not provide an estimate of the accuracy under attack (see Eq. 1) of a robust forest $\mathcal{T}_{\mathcal{P}}$, which clearly depends on the accuracy of its single trees. Yet, the more accurate the trees $t \in \mathcal{T}_{\mathcal{P}}$, the more likely the forest $\mathcal{T}_{\mathcal{P}}$ is accurate under attack.

The accuracy of single trees in a forest $\mathcal{T}_{\mathcal{P}}$ depends on the feature partition $\mathcal{P}$. The larger $|\mathcal{P}|$, the smaller is the number of features each tree can be trained on. To increase the accuracy of a robust forest $\mathcal{T}_{\mathcal{P}}$, we equipartition $\mathcal{F}$ across $\mathcal{P}$ so as to have $|P| \geq \lfloor |\mathcal{F}|/(2b+1) \rfloor$ for all $P \in \mathcal{P}$. Clearly, as the attacker's budget b increases, we need to partition $\mathcal{F}$ into a larger number of subsets, and for these to be effective, we need the dataset to have a larger number of high quality features. Note that this is true for every learning algorithm: if the attacker can perturb at will up to b features, it is necessary to have more than b high-quality features to train an accurate model.

In addition, a specific partition $\mathcal{P}$ may only be suboptimal as there may be multiple ways of partitioning $\mathcal{F}$ so as to achieve feature subsets with high predictive power. Along the lines of ensemble training, we use multiple feature partitions to train a distinct $\mathcal{T}_{\mathcal{P}}$ for each partition $\mathcal{P}$, and join together the resulting robust forests $\mathcal{T}_{\mathcal{P}}$ in a single decision tree ensemble. We can finally discuss our algorithm to train a *Feature Partitioned Forest* $\mathcal{T}$ aimed at being robust against an attacker A_b. The algorithm FPF, shown in Algorithm 1, iterates a number r of user-defined rounds. During each round i, the algorithm generates a random partition $\mathcal{P}_i$ of features $\mathcal{F}$ present in the given training dataset $\mathcal{D}$. At each round, $\mathcal{F}$ is randomly and evenly split into $k = 2b+1$ disjoint subsets ($|\mathcal{P}_i| = 2b+1$), and a new decision tree is trained on each of the dataset projections $\pi_{P_j}(\mathcal{D})$ for every feature subset $P_j \in \mathcal{P}_i$. The resulting $2b+1$ trees form a robust forest $\mathcal{T}_{\mathcal{P}_i}$.

We use an accept condition to filter out those $\mathcal{T}_{\mathcal{P}_i}$ that would not strengthen the final ensemble. For instance, it might be the case that some partitions in $\mathcal{P}_i$ do not contain sufficiently predictive features to train accurate trees. In this work, we use a simple acceptance criterion according to which a $\mathcal{T}_{\mathcal{P}_i}$ is accepted if and only if its training accuracy is larger than naively predicting the dataset's majority class. In this case, $\mathcal{T}_{\mathcal{P}_i}$ is added to $\mathcal{T}$. Eventually, an ensemble $\mathcal{T}$ of r robust forests is returned, including a total of $r(2b+1)$ trees.

Algorithm 1 FPF Training

```
Input: dataset $\mathcal{D}$, attacker $A_b$, training rounds $r$
$\mathcal{T} \leftarrow \emptyset$
$k \leftarrow 2b+1$
for $i = 1$ to $r$ do
    repeat
        $\mathcal{P}_i \leftarrow$ RandomPartition$(\mathcal{F}, k)$
        $\mathcal{T}_{\mathcal{P}_i} \leftarrow \emptyset$
        for all $P_j \in \mathcal{P}_i$, $j = 1, \ldots, k$ do
            $t_{ij} \leftarrow$ DecisionTree$(\pi_{P_j}(\mathcal{D}))$
            $\mathcal{T}_{\mathcal{P}_i} \leftarrow \mathcal{T}_{\mathcal{P}_i} \cup \{t_{ij}\}$
        end for
    until AcceptCondition$(\mathcal{T}_{\mathcal{P}_i})$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{T}_{\mathcal{P}_i}\}$
end for
return $\mathcal{T}$.
```

We propose two methods for exploiting the so built *Feature Partitioned Forest* $\mathcal{T}$ at prediction time. The *hierarchical* method h-FPF (see Algorithm 2) exploits the hierarchical nature of $\mathcal{T}$ in the way that a given instance $\boldsymbol{x}$ is independently classified by each sub-forest $\mathcal{T}_{\mathcal{P}_i} \in \mathcal{T}$, and then the resulting r predictions are merged through a majority voting scheme. The *flat* method f-FPF (see Algorithm 3) considers $\mathcal{T}$ as an ensemble of $r(2b+1)$ independent trees, and the predictions of all single trees are eventually aggregated via majority voting.

Algorithm 2 h-FPF Prediction

```
Input: instance $\boldsymbol{x} \in \mathcal{X}$, FPF forest $\mathcal{T}$
for $\mathcal{T}_{\mathcal{P}_i} \in \mathcal{T}$, with $i = 1 \ldots r$ do
    $y_i = \mathcal{T}_{\mathcal{P}_i}(\boldsymbol{x})$
end for
if $\sum_{i=1}^{r} y_i > 0$ then
    return $+1$
else
    return $-1$
end if
```

Proposition 1 (Robustness of FPF) *The forest* $\mathcal{T}$ *built by the algorithm* FPF *is robust against an attacker* A_b*, as the majority of its trees is not affected by* A_b *for any of its attacks.*

The above proposition is trivially true when considering the h-FPF method, as each sub-forest $\mathcal{T}_{\mathcal{P}_i}$ is robust by construction. When considering the f-FPF method, we recall

Algorithm 3 f-FPF Prediction

```
Input: instance $x \in \mathcal{X}$, FPF forest $\mathcal{T}$
for $\mathcal{T}_{\mathcal{P}_i} \in \mathcal{T}$, with $i = 1 \dots r$ do
    for $t_{ij} \in \mathcal{T}_{\mathcal{P}_i}$, with $j = 1 \dots 2b+1$ do
        $y_{ij} = t_{ij}(x)$
    end for
end for
if $\sum_{i=1}^{r} \sum_{j=1}^{2b+1} y_{ij} > 0$ then
    return $+1$
else
    return $-1$
end if
```

that during each round FPF trains a set of $2b + 1$ trees, where at most b trees can be affected by A_b. After the same reasoning applied to r rounds, FPF builds a forest $\mathcal{T}$ of $r(2b + 1)$ trees of which at most rb can be affected by A_b, leaving a majority of $r(b + 1)$ trees unaffected.

4 Evaluation and certification of tree-based models

Evaluating the accuracy of a model in the presence of an attacker is a difficult and computationally expensive task. This is due to the possibly large size of $A_b(\boldsymbol{x})$ and to the number of interactions among trees in a forest. Below, we first discuss an expensive brute-force strategy for certifying the accuracy under L_0-norm attacks of a tree ensemble FPF on a given test dataset. Then, we show that the existence of an attack over a given dataset instance can be reduced to the existence of a solution for the *partial set coverage problem* [36]. We use this result to devise a strategy aimed at reducing the cost of the brute-force strategy and providing an efficient *lower bound certification* of the accuracy under attack.

4.1 Brute-force evaluation

Given an instance $\boldsymbol{x} \in \mathcal{X}$, the brute-force evaluation of a forest $\mathcal{T}$ consists in generating all the possible perturbations an attacker A_b can operate to find whether there exists $\boldsymbol{x}' \in A_b(\boldsymbol{x})$ such that $\mathcal{T}(\boldsymbol{x}) \neq \mathcal{T}(\boldsymbol{x}')$. Specifically, for a given test dataset $\mathcal{D}$ and a given tree ensemble $\mathcal{T}$, the brute-force evaluation can exactly compute the accuracy under attack according to Eq. 1.

The size of $A_b(\boldsymbol{x})$ is infinite, but we can limit its enumeration to the set of attacks that are *relevant* for the given forest $\mathcal{T}$, i.e., those attacks that can invert the outcome of a test in some internal nodes of trees in $\mathcal{T}$. This set of *relevant attacks*, denoted with $\hat{A}_b(\boldsymbol{x}|\mathcal{T})$, can be computed as follows.

Recall that nodes in a tree are in the form $x_f \leq v$ for some threshold v. Indeed, the thresholds used in the tree nodes induce a discretization of the input space $\mathcal{X}$ that we exploit as follows. For any given feature $f \in \mathcal{F}$, we define with $\mathcal{V}_f$ the set of relevant thresholds as follows:

$$\mathcal{V}_f = \{v \mid \exists \sigma(f, v, t_l, t_r) \in t, t \in \mathcal{T}\} \cup \{\infty\}$$

The set $\mathcal{V}_f$ includes all the thresholds that are associated with f in any node $\sigma(f, v, t_l, t_r)$ of any tree in $\mathcal{T}$, plus the infinity value that allows the algorithm to also include the attack that traverses the right branch of the node with the largest threshold.

An attacker A_b can perturb any subset of features $F \subseteq \mathcal{F}$ such that $|F| \leq b$, and therefore, the set of relevant perturbations the attacker may operate is described by the Cartesian product $\mathcal{V}_{f_1} \times \ldots \times \mathcal{V}_{f_b}$, with $f_i \in F$. We denote by $\hat{A}_b(\boldsymbol{x}|\mathcal{T}, F)$ the set of relevant attacks on the given set of features F, i.e., each perturbed vector $\boldsymbol{x}' \in \hat{A}_b(\boldsymbol{x}|\mathcal{T}, F)$ satisfies the following:

$$x'_f = \begin{cases} x_f & \text{if } f \notin F, \\ v \in \mathcal{V}_f & \text{if } f \in F. \end{cases}$$

In conclusion, the set of relevant attacks is:

$$\hat{A}_b(\boldsymbol{x}|\mathcal{T}) = \bigcup_{\substack{\forall F \subseteq \mathcal{F} \\ |F| = b}} \hat{A}_b(\boldsymbol{x}|\mathcal{T}, F)$$

An attacker A_b can successfully perturb an instance $(\boldsymbol{x}, y) \in \mathcal{D}$ against a forest $\mathcal{T}$ if there exists at least one $\boldsymbol{x}' \in \hat{A}_b(\boldsymbol{x}|\mathcal{T})$ that induces $\mathcal{T}$ to misclassify. We can thus exactly identify the portion of the test dataset that A_b can successfully perturb by using the discretized attacks in $\hat{A}_b(\boldsymbol{x}|\mathcal{T})$, i.e., $\widehat{\mathcal{D}} = \left\{(\boldsymbol{x}, y) \in \mathcal{D} \setminus \overline{\mathcal{D}} \mid \exists \boldsymbol{x}' \in \hat{A}_b(\boldsymbol{x}|\mathcal{T}),\ y \cdot \mathcal{T}(\boldsymbol{x}') < 0\right\}$, where $\overline{\mathcal{D}}$ includes the test instances misclassified by $\mathcal{T}$ in the absence of attack. Finally, we can exactly compute the accuracy under attack according to Eq. 1. This brute-force approach is very expensive, due to three factors: (i) as b increases, the number of possible feature combinations $F \subset \mathcal{F}$ with $|F| = b$ increases; (ii) as the number of trees and nodes grows, the number of threshold values associated with each feature increases; and (iii) for each perturbed instance $\boldsymbol{x}'$, the prediction $\mathcal{T}(\boldsymbol{x}')$ must be computed by traversing the given forest.

4.2 Attacking forest $\mathcal{T}$ as a partial set coverage problem

We now introduce some simplifying worst-case assumptions and then show that an effective attack exists if it can be reduced to a solution for the partial set coverage problem. First, we assume that if a tree in $\mathcal{T}$ provides a wrong prediction before the attack, then its prediction will be incorrect also after the attack. Second, we assume that if a tree uses a feature f for its prediction over $\boldsymbol{x}$, then attacking f causes the tree to generate a wrong prediction. Note that these assumptions are pessimistic from the point of view of a defender. Indeed, modifying a feature f does not necessarily flip the test performed on every node using that feature and, even if it was the case, tests over other features may suffice to avoid a wrong prediction.

Given an instance $(\boldsymbol{x}, y) \in \mathcal{D}$ and a forest $\mathcal{T}$, let C be the set of all *correct* trees $t \in \mathcal{T}$ over $\boldsymbol{x}$, i.e., $C = \{t \in \mathcal{T} \mid t(\boldsymbol{x}) \cdot y > 0\}$. Let $\overline{C} = \mathcal{T} \setminus C$ be the set of all trees providing a wrong prediction over $\boldsymbol{x}$. The goal of the attacker is to force a sufficient number of trees to misclassify $\boldsymbol{x}$ such that the majority of trees are incorrect. The minimum number of trees the attacker must fool is δ such that $|\overline{C}| + \delta = \lceil |\mathcal{T}|/2 \rceil$. It turns out that in a robust forest of $r(2b+1)$ trees, trained with FPF, where the attacker can affect at most rb trees, it is impossible for the attacker to fool the forest if $|\overline{C}| < \lceil r/2 \rceil$. This means that a forest can be robust even if some of its trees are not correct in the absence of an attacker.

Let $S_f \subseteq C$ be the set of all the *correct* trees that use feature f and let $\Sigma = \{S_f\}_{f \in \mathcal{F}}$ be the collection of all S_f. In order for A_b to successfully attack $\mathcal{T}$ over $\boldsymbol{x}$, there must exist a subset $S^* \subseteq \Sigma$, with $|S^*| \le b$ since A_b can perturb a maximum of b features, such that $|\overline{C}| + |\bigcup_{S_f \in S^*} S_f| \ge \lceil |\mathcal{T}|/2 \rceil$, or, equivalently, such that $|\bigcup_{S_f \in S^*} S_f| \ge \delta$ with $\delta = \lceil |\mathcal{T}|/2 \rceil - |\overline{C}|$. The thoughtful reader has surely recognized that this formulation of our problem is nothing else that an instance of the *partial set coverage problem*, where given the set of trees C and the collection $\Sigma \subseteq 2^C$, we have to select up to b sets in Σ that cover at least δ trees.

Before attacking the partial set coverage problem, we make a few improvements to provide a stricter definition of sets S_f in relation to our scenario. First, we note that a tree may include a feature f in some of its nodes, but these nodes may never be traversed during the evaluation of an instance $\boldsymbol{x}$. Therefore, we say that a *correct* tree t belongs to S_f for an instance $\boldsymbol{x}$ only if the traversal path of $\boldsymbol{x}$ in t includes a node with a test on feature f. This already reduces the size of each S_f.

Then, among the nodes along the traversal path of instance $\boldsymbol{x}$ before the attack, we can further distinguish between nodes where the test $x_f \le v$ is true, and nodes where the test is false. In the former case, the attacker must increase the value of x_f to affect the traversal path, while in the latter case the attacker must decrease x_f. Clearly, these two attacks cannot coexist.

Therefore, we define sets S_f^+ and S_f^- as follows. Given a *correct* tree $t \in C$, we include t in S_f^+ if the traversal path of $\boldsymbol{x}$ in t includes a node with a test $x_f \le v$ on feature f and this test gives a true outcome. Otherwise, if the outcome of this test turns out to be false, we include t in S_f^- . This method allows us to achieve a more accurate modeling of when an attack can actually affect the final prediction. This also reduces the size of sets in Σ and decreases the risk of overestimating the effect of an attack. We can finally conclude the relation with the *partial set cover* problem as follows.

Proposition 2 (Partial set coverage as a necessary condition for successful attacks.) *Given $(\boldsymbol{x}, y) \in \mathcal{D}$, where $\mathcal{T}(\boldsymbol{x}) \cdot y > 0$, a necessary condition for the existence of a successful attack $\boldsymbol{x}' \in A_b(\boldsymbol{x})$ such that $\mathcal{T}(\boldsymbol{x}') \cdot y < 0$, is that there exists a solution for the partial set coverage problem, stated as follows:*

Given the set system (C, Σ), where C is the finite set of correct trees for $\boldsymbol{x}$, where $\Sigma \subseteq 2^C$ with $\Sigma = \{S_f^+\}_{f \in \mathcal{F}} \cup \{S_f^-\}_{f \in \mathcal{F}}$, and given integer b and a constant $\delta = \lceil |\mathcal{T}|/2 \rceil - |\overline{C}|$, the goal is to find a sub-collection $S^* \subseteq \Sigma$, where $|\bigcup_{S \in S^*} S| \ge \delta$, with the constraints that $|S^*| \le b$ and, $\forall f \in \mathcal{F}$, if $S_f^+ \in S^*$ ($S_f^- \in S^*$) then $S_f^- \notin S^*$ ($S_f^+ \notin S^*$).

Proof We show that if there exists $\boldsymbol{x}' \in A_b(\boldsymbol{x})$ such that $\mathcal{T}(\boldsymbol{x}') \cdot y < 0$, then there exists $S^* \subseteq \Sigma$, where $|\bigcup_{S \in S^*} S| \ge \delta$, $|S^*| \le b$, and S_f^+ and S_f^- are mutually exclusive in S^*. Given $\boldsymbol{x}' \in A_b(\boldsymbol{x})$, we say that for any attacked feature f either the corresponding set S_f^+ belongs to S^* if $x'_f - x_f > 0$ (corrupted by increment) or S_f^- belongs to S^* if $x'_f - x_f < 0$ (corrupted by decrement). Clearly, it holds that $|S^*| \le b$, and the sets S_f^+ and S_f^- are mutually exclusive in S^*. Let $\overline{C'}$ be the set of (formerly correct) trees corrupted by the successful attack $\boldsymbol{x}'$, then it holds that $|\overline{C'}| \ge \delta$. By construction, any tree $t \in \overline{C'}$ belongs to either S_f^+ or S_f^- included in S^*. Therefore, it holds that $|\overline{C'}| \le |\bigcup_{S \in S^*} S|$, which implies $|\bigcup_{S \in S^*} S| \ge \delta$. □

Note that Proposition 2 states that the existence of a solution S^* for our partial set cover problem is only a necessary (not sufficient) condition for the attack. Thus, if S^* exists, we cannot say that $\mathcal{T}$ can be fooled for sure, as the attacker might modify all the features identified by the cover without being able to affect the final forest prediction. However, we know that if a solution S^* does not exist, then $\mathcal{T}$ is robust on the given instance $\boldsymbol{x}$.

In the following, we use this result to compute an upper bound of the size of $\widehat{\mathcal{D}}$, the set of all instances in the test dataset that can be attacked. On the one hand, this allows us to lower bound the accuracy under attack ACC_{A_b} (see Eq. 1); on the other hand, this result makes it possible to speed up the exact computation of ACC_{A_b} by employing the brute-force approach only for those instances for which a sufficiently large set cover does not exist.

4.3 Fast accuracy lower bound

The method discussed in this section computes an overestimate of the size of the partial set cover $S^* \subseteq \Sigma$, for the problem stated in Proposition 2.

If we consider the b largest sets in Σ, it is clear that the cardinality of the union of the sets within S^* is smaller or equal to the sum of the cardinalities of such b largest sets (inclusion-exclusion principle). We improve this trivial upper bound by considering that the two sets S_f^+ and S_f^- cannot be included together in a potential cover.

We thus define the *fast lower bound* set S_{FLB} to be the set of the b largest sets in Σ after enforcing the constraint that for a given feature f only the largest between S_f^+ and S_f^- is considered.

We can conclude that if $\sum_{S\in S_{FLB}} |S| < \delta$, then a suitable partial cover cannot exist and therefore the forest $\mathcal{T}$ cannot be attacked on $\boldsymbol{x}$.

Therefore, we define the set $\widehat{\mathcal{D}}_{FLB}$ of attackable instances according to the *fast lower bound* method as follows. For each correctly classified instance $(\boldsymbol{x}, y) \in \mathcal{D}$, we build the partial coverage problem according to Proposition 2, and *iff* $\sum_{S\in S_{FLB}} |S| \geq \delta$, then we include the instance $(\boldsymbol{x}, y)$ in $\widehat{\mathcal{D}}_{FLB}$. Since it holds that $|\widehat{\mathcal{D}}_{FLB}| \geq |\widehat{\mathcal{D}}|$, we define the *fast lower bound accuracy* as:

$$ACC_{A_b}^{FLB} = 1 - \frac{|\overline{\mathcal{D}}| + |\widehat{\mathcal{D}}_{FLB}|}{|\mathcal{D}|} \leq ACC_{A_b}.$$

4.4 Exhaustive accuracy lower bound

In order to improve over the fast lower bound, we also consider a more expensive option, where all the possible covers are considered, still respecting the constraint that any S_f^+ and S_f^- are mutually exclusive. We evaluate all the possible covers $S^\dagger \subset 2^\Sigma$, $|S^\dagger| \leq b$, and we call *exhaustive lower bound* cover, denoted with S_{ELB}, the first cover found such that $|\bigcup_{A\in S_{ELB}} A| \geq \delta$.

By applying the same procedure to every correctly classified instance $(\boldsymbol{x}, y)$ in the dataset, we identify the set of instances $\widehat{\mathcal{D}}_{ELB}$ for which there exists an *exhaustive lower bound* cover S_{ELB} that solves the problem in Proposition 2.

Note that $|\widehat{\mathcal{D}}| \leq |\widehat{\mathcal{D}}_{ELB}| \leq |\widehat{\mathcal{D}}_{FLB}|$, and thus, we use this method to compute another lower bound for the accuracy of $\mathcal{T}$ on the test dataset $\mathcal{D}$:

$$ACC_{A_b}^{ELB} = 1 - \frac{|\overline{\mathcal{D}}| + |\widehat{\mathcal{D}}_{ELB}|}{|\mathcal{D}|} \leq ACC_{A_b},$$

where the following relationship trivially holds:

$$ACC_{A_b}^{FLB} \leq ACC_{A_b}^{ELB} \leq ACC_{A_b}.$$

This *exhaustive lower bound* search incurs into the exponential cost of enumerating the possible covers in 2^Σ, but it improves over the brute-force attack, thanks to the cover-based formulation, by ignoring the relevant threshold values $\mathcal{V}_f$ each feature can be attacked to.

We recall that while it is true that $|\widehat{\mathcal{D}}| \leq |\widehat{\mathcal{D}}_{ELB}| \leq |\widehat{\mathcal{D}}_{FLB}|$, we cannot claim that $\widehat{\mathcal{D}} \subseteq \widehat{\mathcal{D}}_{ELB} \subseteq \widehat{\mathcal{D}}_{FLB}$. The above bounds can prove the non-existence of an actual cover, but they may not be used to find a successful attack strategy.

4.5 Cascading evaluation

Above, we presented two algorithms, FLB and ELB, that efficiently find an overapproximation of a cover of attacked trees which allows us to estimate the upper bound of the most harmful attack. Strategies have different costs: FLB requires to sort the candidate sets of the cover, while ELB performs an exhaustive search of all the possible subsets of Σ. Both methods are however much cheaper than brute-force evaluation.

When the lower bound information is not considered sufficient, in order to compute the actual accuracy under attack ACC_{A_b}, we propose to exploit the following CASCADING strategy.

Given an instance $\boldsymbol{x}$ and an attacker A_b, we build the collection of sets of trees $\Sigma = \{S_f^+\}_{f\in\mathcal{F}} \cup \{S_f^-\}_{f\in\mathcal{F}}$ and proceed as follows:

1 Compute $S_{FLB} \subseteq \Sigma$: if $\sum_{S\in S_{FLB}} |S| < \delta$, then no sufficiently large set cover exists, and therefore, the instance $\boldsymbol{x}$ cannot be attacked; otherwise
2 Search for a suitable cover $S_{ELB} \subseteq \Sigma$: if there is no S_{ELB} such that $|\bigcup_{S\in S_{ELB}} S| \geq \delta$, then the instance $\boldsymbol{x}$ cannot be attacked; otherwise
3 Use the brute-force method to check the existence of a successful attack on $\boldsymbol{x}$.

Experimental results show that the above cascading strategy is able to strongly reduce the number of instances in a given dataset $\mathcal{D}$ for which the brute-force approach is required.

4.6 Non-binary classification

While we leave to future work the design and evaluation of an algorithm for non-binary classification, we highlight that the proposed methodology can be easily generalized to a multi-class scenario, and we sketch below a basic certification methodology. The algorithms proposed so far aim at certifying the impossibility of the attacker of modifying a number of correct trees δ such that $\delta \geq \lceil |\mathcal{T}|/2 \rceil - |\overline{C}|$, where $\overline{C}$ is the set of trees wrongly classifying the given instance. In regard to a multi-class classification problem, given classes $\mathcal{Y}$ with $|\mathcal{Y}| > 2$, it is possible to verify robustness by running $|\mathcal{Y}| - 1$ certifications analogous to the binary case. Let us denote with C_c the set of trees classifying, before the attack, the given instance $(\boldsymbol{x}, y)$ as class c, $c \in \mathcal{Y}$, thus being C_y the trees predicting the correct label y. For a given class $c \in \mathcal{Y}$, the attacker aims at attacking the trees in $\bigcup_{i\neq c} C_i$ so as to make c the new majority class. The best-case scenario from the point of the attacker is given by modifying the predictions of the trees in C_y, as in this case it is sufficient to attack $\delta = \lceil (|C_y| - |C_c|)/2 \rceil$ trees. If the attacker is not able to alter at least δ trees of the forest, then no successful attack is possible. To this end, we can exploit any of the set-cover base techniques proposed so far to verify the absence of

Table 1 Dataset description

Dataset	$\lvert\mathcal{F}\rvert$	#top $\mathcal{F}$	maj. class	$\lvert\mathcal{D}\rvert$
WINE	13	7	73.0%	178
BREAST_CANCER	30	15	62.7%	569
SPAM_BASE	57	26	60.6%	4600
MNIST 0/1	784	54	53.3%	14780
MNIST 5/6	784	173	52.1%	13189
MNIST 1/7	784	79	51.9%	15170

any cover of size at least δ among the trees $\bigcup_{i\neq c} C_i$. Such verification is to be repeated for each $c \in \mathcal{Y}$.

5 Experiments

5.1 Experimental settings

In Table 1, we report the main characteristics of the datasets used in the experimental evaluation, including the number of features, the number of top relevant features measured as those contributing to 90% of the feature importance in a Random Forest, and the relative size of the majority class. Datasets, ranging from small to mid-sized, are associated to a binary classification task, and they are commonly used in adversarial learning literature[2].

We compare our proposed algorithm FPF against the following tree-ensemble competitors:

- Random Forest (RF) [37], which, although attacker-unaware, is known to have some good level of robustness thanks to the ensembling of several decision trees. As in the original algorithm, each tree is trained on a bootstrap sample of the dataset, with no constraints on the number of leaves, and with feature sampling of size $\sqrt{|\mathcal{F}|}$ at each node.
- Random Subspace method (RSM) [34], which was successfully exploited in the adversarial setting by Biggio et al. [35]. In this case, each tree is trained on a projection of the original dataset on a subset of its features. Validation experiments showed best results with 20% feature sampling.
- Robust Trees (RT) [22] is an adversarial learning algorithm that targets robustness through optimization of the model performance under attack. We created an ensemble by training each individual tree with the algorithm proposed in the original paper.

Hereinafter, we use b to represent the expected attacker's budget exploited as training parameter of FPF and RT, and we use k to stand for the attacker's budget used at test time to generate attacks. We conducted a fair comparison by allowing each method to grow 300 trees, which was observed experimentally to be an optimal configuration. By leveraging the usual 60-20-20 train, validation and test set split, for all the considered algorithms, we fine-tuned their hyper-parameters on the validation set by optimizing the accuracy under attack ACC_{A_k}. The number of leaves was fine-tuned in the set $\{4, 8, 16\}$. The RSM algorithm requires a sampling parameter p chosen in the set $\{0.1, 0.2, 0.4, 0.6\}$. Finally, regarding FPF, we fine-tuned its b parameter in the range $[1, 5]$ while the number of rounds r was set on the basis of b so as to have no more than 300 trees ($r = \lfloor\frac{300}{2b+1}\rfloor$). All reported results were computed on the test set of each dataset.

The following experimental evaluation aims at answering the following research questions:

- Is FPF able to train robust models in terms of accuracy under attack?
- How is FPF affected by the number of rounds r and the expected attacker's budget b used at training time?
- How accurate are the proposed lower bounds FLB and ELB?
- How efficient is the computation of the proposed lower bounds FLB and ELB, and can we exploit them to analyze models on larger attacker's budgets k?

5.2 Robustness analysis

In Tables 2, 3, and 4, we report the accuracy of FPF, RF, RSM, and RT against an attacker that can modify up to 3 features. To generate all the attacks, we used the exhaustive brute-force algorithm defined in Section 4.1, through which we can calculate the exact robustness of the models. For the SPAM_BASE dataset, we limit the analysis to two attacked features, due to the cost of computing ACC_{A_3}. We also report the value of ACC_{A_0}, which captures the

Table 2 Accuracy on the WINE dataset (in boldface best results for each k)

Model	Parameters					Accuracy			
	b	r	p	ml	$\lvert\mathcal{T}\rvert$	ACC_{A_0}	k	ACC_{A_k}	Δ f-FPF
f-FPF	2	60		8	300	0.972	1	**0.861**	0.000
	5	27		8	297	0.889	2	**0.750**	0.000
	5	27		8	297	0.889	3	0.611	0.000
h-FPF	2	60		8	300	0.972	1	**0.861**	0.000
	5	27		8	297	0.917	2	0.722	−0.028
	5	27		8	297	0.917	3	0.611	0.000
RSM			0.2	8	300	0.944	1	0.833	−0.028
			0.2	4	300	0.972	2	0.722	−0.028
			0.2	8	300	0.944	3	0.444	−0.167
RF				8	300	0.944	1	0.833	−0.028
				4	300	1.0	2	0.389	−0.361
				4	300	1.0	3	0.000	−0.611
RT	1			8	300	0.738	1	0.738	−0.123
	2			8	300	0.738	2	0.738	−0.012
	3			8	300	0.738	3	**0.738**	+0.127

[2] All datasets are available at the UCI Machine Learning Repository.

Table 3 Accuracy on the BREAST_CANCER dataset (in boldface best results for each k)

Model	Parameters					Accuracy			
	b	r	p	ml	$\lvert\mathcal{T}\rvert$	ACC_{A_0}	k	ACC_{A_k}	Δ f-FPF
f-FPF	3	42		8	300	0.939	1	**0.912**	0.000
	4	33		4	297	0.930	2	**0.851**	0.000
	5	27		8	300	0.930	3	**0.816**	0.000
h-FPF	3	42		8	300	0.939	1	**0.912**	0.000
	4	33		4	297	0.930	2	0.842	−0.009
	5	27		8	300	0.930	3	**0.816**	0.000
RSM			0.2	16	300	0.956	1	**0.912**	0.000
			0.2	16	300	0.956	2	0.833	−0.018
			0.2	16	300	0.956	3	0.658	−0.158
RF				4	300	0.930	1	0.877	−0.035
				8	300	0.947	2	0.842	−0.009
				4	300	0.930	3	0.614	−0.202
RT	1			4	300	0.923	1	0.562	−0.350
	1			4	300	0.923	2	0.505	−0.346
	1			4	300	0.923	3	0.454	−0.362

model accuracy in the absence of attacks. For FPF, we evaluate its robustness on varying the number of rounds r and the defense parameter b, still keeping the total number of trees about constant and comparable with the size of the forests generated by competitor algorithms. The rightmost column in these tables reports the difference w.r.t. the f-FPF algorithm for the same value of k.

In all datasets, RF performs best or second best in the absence of attacks, but its accuracy significantly drops under attack. The perturbation of one single feature is sufficient to harm the model, with a loss of 5 to 15 points in accuracy, and when attacking two features, the accuracy drops well under 50% for WINE and SPAM_BASE datasets. As previously discussed, the L_0-norm attack we are tackling in this work is indeed very powerful and sufficient to fool a very accurate and effective random forest model which is not adversarially trained.

Table 4 Accuracy on the SPAM_BASE dataset (in boldface best results for each k)

Model	Parameters					Accuracy			
	b	r	p	ml	$\lvert\mathcal{T}\rvert$	ACC_{A_0}	k	ACC_{A_k}	Δ f-FPF
f-FPF	2	60		16	300	0.918	1	**0.837**	0.000
	4	33		16	297	0.866	2	**0.782**	0.000
h-FPF	2	60		16	300	0.913	1	0.834	−0.003
	4	33		8	297	0.865	2	0.775	−0.007
RSM			0.2	16	300	0.909	1	**0.837**	0.000
			0.2	16	300	0.909	2	0.736	−0.046
RF				8	300	0.923	1	0.766	−0.071
				4	300	0.902	2	0.449	−0.333
RT	3			8	300	0.715	1	0.433	−0.422
	3			8	300	0.715	2	0.201	−0.655

The RSM model provides good performance in the absence of attacks, meaning that the dataset projection is not disadvantageous, and it is much more robust than RF in the presence of attacks. However, when attacking two features, RSM exhibits a drop of about 10 to 20 points in accuracy on all datasets. When attacking three features, the accuracy of the model is very small: below 45% on WINE and below 66% on BREAST_CANCER.

The results obtained with RT show how this strategy for training robust models against adversarial attacks does not perform well in case of L_0-norm attacks. This can be explained by observing how the learning algorithm trains a robust model. During the training phase, the model has to choose the feature and threshold pair which maximizes the minimum accuracy under attack. However, an L_0-norm attacker can always modify a feature as much as he wants and therefore can always cross the threshold. Consequently, it is difficult for the training algorithm to choose which pair is the best, because each pair generates the same information gain under attack. As a consequence, the algorithm cannot guarantee robustness for this type of attack. Furthermore, it can be seen that for the WINE dataset, the algorithm always returns the same accuracy and robustness and that these exactly coincide with the majority class percentage of the dataset. This happens because in the training phase the algorithm realizes that always returning the majority class is the best solution for maximizing robustness.

The proposed FPF algorithm can provide the best robustness in virtually all attack scenarios. We highlight that the best defensive b found is always larger than the attacker's budget k, meaning that increasing the number of feature partitions provides even better robustness. The scenario where the attacker may attack $k = 3$ features is especially relevant: the proposed FPF algorithm achieves a 15–20% better accuracy than the competitor RSM. Furthermore, the results of the experiments on WINE reported in Table 2 also highlight the ability of FPF to guarantee greater robustness even with few features in the dataset. As reported in Table 1, WINE has 13 features of which only 7 are relevant. Attacks executed with budget 1, 2, and 3 compromise the 7.7%, 15.4%, and 23.1% of the features respectively or 14.3%, 28.6%, and 42.8% if we consider only the relevant features. For each budget value, FPF showed a greater (or equal) robustness than the other models.

Experiments do not show any significant difference between f-FPF and h-FPF. They both provide the same accuracy with $k = 1$ and $k = 3$, with f-FPF exhibiting slightly better figures on $k = 2$. While h-FPF has better theoretical guarantees in a black-box scenario (see

Section 6), the two variants f-FPF and h-FPF do not differ much in practice.

We conclude that FPF is able to outperform state-of-the-art competitors, especially when considering stronger attackers.

5.3 Sensitivity analysis

In Table 5, we evaluate the sensitivity of FPF w.r.t. the number of rounds r on the BREAST_CANCER dataset for different values of b. Similar results were observed for the other datasets. With a few exceptions, the ensembling strategy improves the accuracy of the resulting model, and accuracy increases when increasing the number of rounds until a plateau is reached. We can conclude that using a large number of rounds r improves the robustness of the trained model.

Table 5 Sensitivity analysis with respect to the number of rounds r on the BREAST_CANCER dataset

b	A_k	Number of rounds r		
		1	**15**	**30**
f-FPF algorithm				
1	0	**0.956**	0.939	0.939
	1	0.842	**0.921**	**0.921**
	2	0.000	0.675	**0.719**
	3	0.000	0.114	**0.158**
2	0	0.93	**0.939**	**0.939**
	1	0.904	**0.930**	**0.930**
	2	0.754	**0.842**	**0.842**
	3	0.123	0.404	**0.474**
3	0	**0.947**	0.939	0.939
	1	0.912	**0.921**	**0.921**
	2	0.833	**0.86**	**0.86**
	3	0.675	**0.772**	0.763
4	0	**0.956**	0.939	0.939
	1	0.895	**0.904**	**0.904**
	2	0.825	**0.868**	**0.868**
	3	0.728	**0.807**	**0.807**
5	0	0.930	**0.939**	0.930
	1	**0.912**	0.895	0.895
	2	0.842	0.860	**0.868**
	3	0.746	**0.816**	**0.816**
h-FPF Algorithm				
1	0	**0.956**	**0.956**	0.947
	1	0.842	0.912	**0.921**
	2	0.000	**0.658**	0.421
	3	0.000	0.07	**0.079**
2	0	0.93	**0.939**	**0.939**
	1	0.904	**0.930**	**0.930**
	2	0.754	0.833	**0.851**
	3	0.123	0.351	**0.377**
3	0	**0.947**	0.939	0.939
	1	0.912	0.921	**0.930**
	2	0.833	0.860	**0.877**
	3	0.675	**0.772**	**0.772**
4	0	**0.956**	0.930	0.930
	1	0.895	**0.904**	**0.904**
	2	0.825	0.860	**0.868**
	3	0.728	0.798	**0.816**
5	0	0.930	**0.939**	0.921
	1	**0.912**	0.895	0.895
	2	0.842	0.868	**0.877**
	3	0.746	0.816	**0.825**

The impact of b is evaluated in Table 6. Note that in the BREAST_CANCER dataset, we identified only 15 informa-

Table 6 Sensitivity analysis with respect to the training parameter b on the BREAST_CANCER dataset

$\|\mathcal{T}\|$	A_k	Parameter b				
		1	**2**	**3**	**4**	**5**
f-FPF algorithm						
50	0	0.939	0.939	0.947	0.939	**0.956**
	1	0.904	0.921	0.912	**0.930**	0.921
	2	0.737	0.851	**0.868**	**0.868**	0.860
	3	0.105	0.404	0.728	**0.825**	**0.825**
75	0	**0.939**	**0.939**	**0.939**	**0.939**	**0.939**
	1	0.912	**0.930**	0.912	0.904	0.904
	2	0.719	0.842	**0.868**	0.860	**0.868**
	3	0.167	0.404	0.754	**0.816**	**0.816**
100	0	**0.939**	**0.939**	**0.939**	**0.939**	**0.939**
	1	**0.93**	0.921	0.921	0.904	0.904
	2	0.465	0.842	0.860	**0.886**	0.868
	3	0.158	0.482	0.763	0.798	**0.816**
h-FPF Algorithm						
50	0	**0.947**	0.939	**0.947**	**0.947**	**0.947**
	1	0.904	**0.93**	0.921	0.921	0.921
	2	0.711	0.842	0.868	0.86	**0.877**
	3	0.026	0.351	0.737	**0.807**	0.789
75	0	**0.947**	0.939	0.939	0.939	0.939
	1	0.912	**0.93**	0.921	0.904	0.904
	2	0.404	0.833	0.86	0.86	**0.868**
	3	0.096	0.351	0.746	**0.798**	0.789
100	0	0.947	0.939	0.939	**0.947**	0.930
	1	**0.921**	**0.921**	**0.921**	0.904	0.904
	2	0.412	0.842	0.868	**0.877**	0.868
	3	0.096	0.439	0.754	**0.798**	**0.798**

Table 7 Accuracy lower bound analysis on BREAST_CANCER. f-FPF with $|\mathcal{T}| \approx 300$ and 8 leaves

	$k=1$			$k=2$			$k=3$		
b	ACC_{A_1}	$ACC^{ELB}_{A_1}$	$ACC^{FLB}_{A_1}$	ACC_{A_2}	$ACC^{ELB}_{A_2}$	$ACC^{FLB}_{A_2}$	ACC_{A_3}	$ACC^{ELB}_{A_3}$	$ACC^{FLB}_{A_3}$
1	0.912	0.886	0.886	0.763	0	0	0.184	0	0
2	0.912	0.904	0.904	0.842	0.833	0.825	0.649	0	0
3	0.912	0.912	0.912	0.860	0.851	0.842	0.781	0.763	0.737
4	0.904	0.904	0.904	0.868	0.868	0.868	0.798	0.798	0.798
5	0.895	0.895	0.895	0.868	0.868	0.868	0.816	0.816	0.807

tive features which are difficult to partition in $2b+1$ sets for large values of b. In this case, the results are mixed and we can identify two different trends. For attacks A_2 and A_3, increasing b brings some interesting benefits, and it is always worthwhile to use $b \geq 4$. When the attacker's budget is limited to A_1, then the best configuration also depends on the number of trees. When using a limited number of trees, accuracy under attack increases with b. But when the forest is sufficiently large, then the benefit of increasing the number of trees is larger than the benefit of increasing b.

We conclude that, while it is beneficial to increase the number of rounds r, it is not always a good strategy to increase b when the total number of trees is small and the number of informative features is limited.

5.4 Lower bound analysis

In Tables 7 and 8, we compare the accuracy under attack ACC_{A_k} with the lower bounds $ACC^{ELB}_{A_k}$ and $ACC^{FLB}_{A_k}$ on both f-FPF and h-FPF forests, built with different values of b on the BREAST_CANCER dataset and attacked by different attackers A_k.

We first highlight that with $k=1$, the estimate provided by the lower bounds is exact in the large majority of settings. With $k=2$, we observe a small difference between the two estimates $ACC^{ELB}_{A_k}$ and $ACC^{FLB}_{A_k}$. Still, with $b \geq 4$, both bounds provide an exact estimate, while with $b=1$ and $b=2$, the exhaustive lower bound introduces an error always smaller than 0.019 points and the fast lower bound smaller than 0.026 points. Finally, with $k=3$, the gap between the two lower bounds increases. The error exhibited by the exhaustive technique is within 0 and 0.052 points, while the fast lower bound shows an error between 0 and 0.088. Clearly, when $k > b$, the lower bound estimate is useless, as there always exists a sufficiently large partial cover.

We conclude by observing that both lower bounds are very close to the actual accuracy under attack. For instance, the exhaustive lower bound always provides the correct accuracy of the f-FPF model with $b \geq 4$ and for every value of k. This makes the provided lower bounds an efficient and accurate tool. In Figs. 1 and 2, we show the accuracy lower bound computed with FLB on different datasets, while varying b and the attacker power k. In this set of experiments, we also consider three binary classification datasets generated from MNIST by isolating instances of two digits. These datasets encompass a much larger number of features that makes brute-force approaches to security certification infeasible. Indeed, the computational efficiency of the proposed bound allows us to compute the minimum accuracy for large values of b and large attacker's budgets k. Figures show how larger values of b allow ones to sustain a larger attacker strength. Of course, when the attacker becomes too strong compared with the number of relevant features in the dataset, then the accuracy of FPF drops. Regarding the MNIST datasets, the lower bounds allow us to state that a reasonable accuracy can be achieved also when attacking more than 20 features. The weakest dataset is that of digits 5 vs. 6, where the attacker is clearly required to change fewer pixels to induce a misclassification.

5.5 Efficiency analysis

In Table 9, we report the per-instance average time required to run the brute-force certification method BF and the proposed CASCADING strategy. We observe how

Table 8 Accuracy lower bound analysis on BREAST_CANCER. h-FPF with $|\mathcal{T}| \approx 300$ and 8 leaves

	$k=1$			$k=2$			$k=3$		
b	ACC_{A_1}	$ACC^{ELB}_{A_1}$	$ACC^{FLB}_{A_1}$	ACC_{A_2}	$ACC^{ELB}_{A_2}$	$ACC^{FLB}_{A_2}$	ACC_{A_3}	$ACC^{ELB}_{A_3}$	$ACC^{FLB}_{A_3}$
1	0.912	0.895	0.868	0.746	0	0	0.175	0	0
2	0.904	0.904	0.904	0.842	0.833	0.816	0.632	0	0
3	0.912	0.912	0.904	0.868	0.851	0.842	0.789	0.737	0.701
4	0.904	0.904	0.895	0.860	0.860	0.860	0.807	0.798	0.781
5	0.895	0.895	0.895	0.868	0.868	0.860	0.816	0.807	0.798

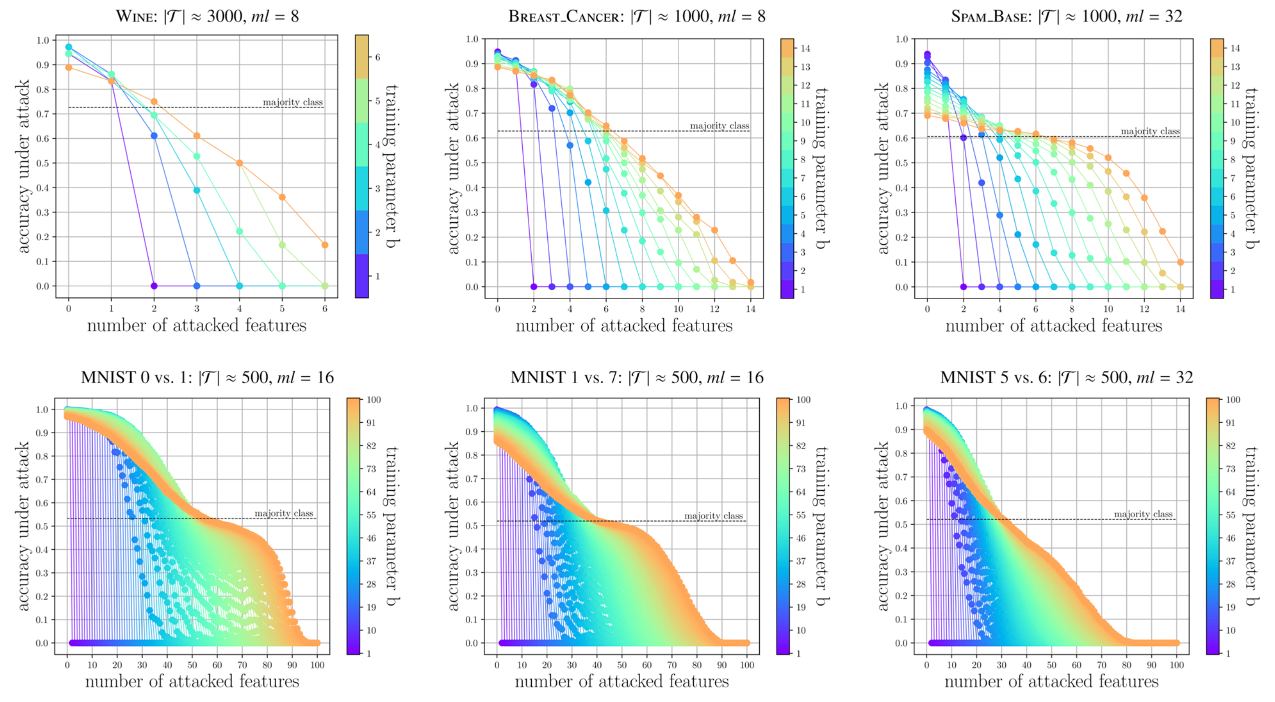

Fig. 1 Accuracy lower bounds for large b with f-FPF

the computational cost required by BF exponentially increases when increasing k. As expected, the BF approach quickly becomes infeasible.

The proposed CASCADING strategy provides a 10× speed-up that increases to 100× when $k = 3$. This huge gap is due to the efficiency and accuracy of the proposed lower bounds. On varying k, the fraction of instances for which FLB cannot certify the non-attackability is respectively 2%, 8%, and 20%. These instances are processed during the ELB step, which leaves to the last BF step 2%,

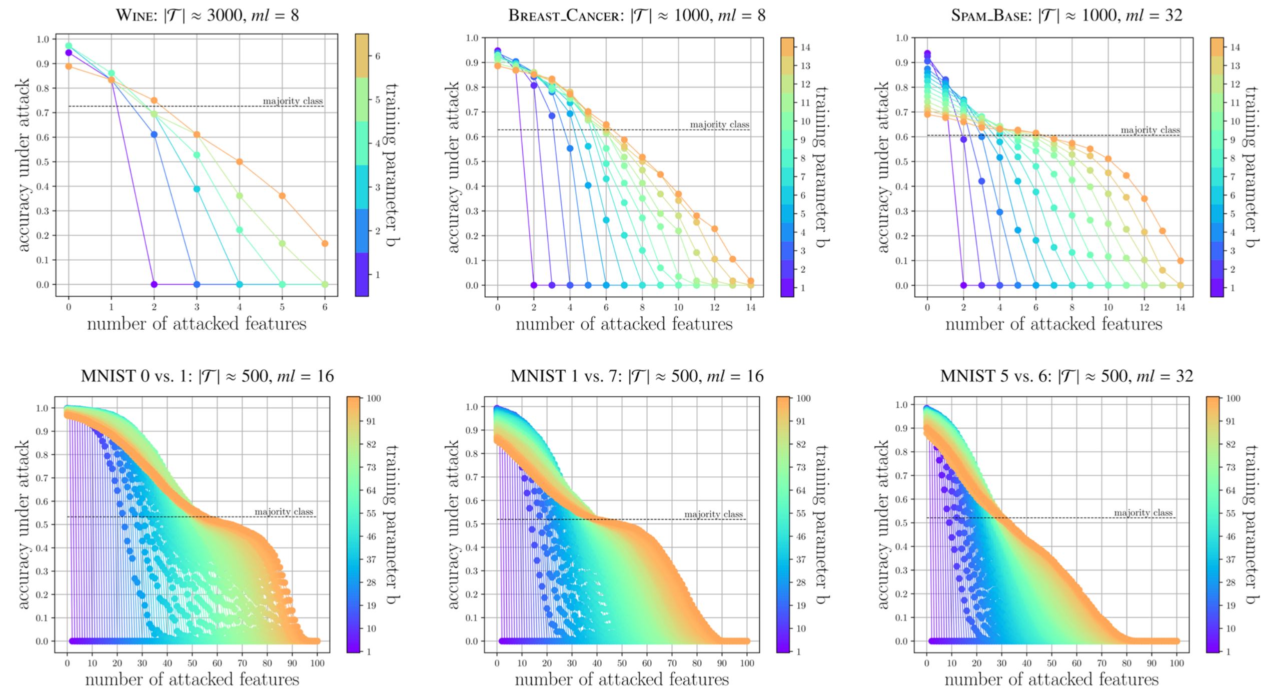

Fig. 2 Accuracy lower bounds for large b with h-FPF

Table 9 Certification per-instance average execution time on BREAST_CANCER

Method	$k = 1$		$k = 2$		$k = 3$	
BF	263.6 ms		1053.4 ms		3635 s	
CASCADING	27.1 ms		166.0 ms		36 s	
breakdown:						
–FLB	2.7 ms	(100%)	3.2 ms	(100%)	3.3 ms	(100%)
–ELB	0.8 ms	(2%)	2.0 ms	(8%)	29.7 ms	(20%)
–BF	23.6 ms	(2%)	160.8 ms	(7%)	36 s	(18%)

Between parentheses, the fraction of instances passing through the CASCADING stages. f-FPF with $|\mathcal{T}| \approx 300$ and 8 leaves

7%, and 18%, respectively, of the dataset to be analyzed. The BF certification of these last instances largely covers from 85% to about 100% of the total running time, while the FLB and ELB steps are two or more orders of magnitudes faster. We thus conclude that the proposed FLB and ELB are both sufficiently accurate, as discussed in the previous subsections, and they can be used in a CASCADING-like strategy to provide significant speed-ups to any other exact certification method.

6 Conclusion

This paper proposes FPF, a new algorithm to generate forests of decision trees, based on random equi-partitioning of the feature set, along with a projection of the dataset on these partitions before training each single decision tree. The method is proven to be resilient against evasion attacks, and, more importantly, we are able to certify in a very efficient way that, given a test dataset, some of the instances cannot be attacked at all, thus avoiding the costly computation of all the possible evasion attacks.

The experimental evaluation, carried out on publicly available datasets, is promising and outperforms the main direct competitor, based on ensembles built on random sampling of the features. Moreover, we show that the proposed certification methods provide a very close approximation of the actual accuracy and they can be used through a cascading approach to speed-up an exact accuracy under attack computation.

We finally highlight that the proposed feature partitioning methodology easily generalizes to ensembles of other machine learning algorithms, whose investigation is left as future work.

Appendix

Theoretical analysis

In this section, we study analytically the behavior of FPF on varying the number of rounds r, the number of features d, and the attacker's budget b, and we also take into consideration the probability of incorrect prediction by trees of the forest. The aim of this analysis is to investigate the impact of the above hyper-parameters to the robustness of a forest built by FPF. To do so, we resort to the common *black-box attack* scenario where the attacker has no access to the internal structure of the forest to choose which features to attack. We thus compute the accuracy of the forest under attack by estimating the probability that the attacker A_b may successfully fool the given forest by picking b features at random. Indeed, this probabilistic analysis is aimed at understanding the asymptotic behavior of the proposed algorithm.

We highlight that in the experimental section, we rather adopt a more severe *white-box attack* scenario where we consider an instance attacked if there is at least one successful attack.

Let d be the number of features in the input dataset, b the attacker's budget, $\mathcal{P}$ a robust equi-partition of features $\mathcal{F}$, and $s{=}d/(2b+1)$ the number of features in each of the $2b+1$ sets. For the sake of simplicity, to guarantee equi-partitioning, we assume that d is a multiple of $2b+1$[3]. Moreover, we let e be the probability of a tree $t \in \mathcal{T}$ of being erroneous, i.e., we do not assume that all trees are perfectly accurate. We further adopt the conservative and pessimistic assumption that if the attacker modifies a feature, then a tree using that feature will provide a wrong prediction.

We first compute the probability $\Pr(h)$ that b features, selected at random by A_b, overlap with exactly h partitions in $\mathcal{P}$. To do so, we first restrict our attention to attacks that are entirely included into a given subset $H \subset \mathcal{P}$, with $|H|{=}h$, and then we generalize to the full partition $\mathcal{P}$. When $|H|{=}h$, the set H includes sh features, and we denote by U_H the set of possible attacks over any b features in H, where $|U_H| = \binom{sh}{b}$ is the number of such attacks. Indeed, we are interested in computing $\widehat{U}_H \subseteq U_H$, i.e, the set of attacks to the features in H that exactly overlap all the h partitions in H (and not less).

Proposition 3 (Computing the number of attacks $\widehat{U}_H$) *Let K_i be set of all attacks in U_H that do not overlap with $P_i \in H$, and let $\bar{K}_i$ be the complement of K_i in U_H. The following equation holds:*

$$\widehat{U}_H = \bigcap_{i=1}^{h} \bar{K}_i.$$

Proof Let C be a possible attack, i.e., a tuple of b features chosen from the sh features of the h partitions in H. For every $C \in \widehat{U}_H$, by definition, C contains h features from h different partitions in H, and therefore, C has non-empty intersection with every $\bar{K}_i$. This proves

[3]Otherwise, to guarantee semi-equi-partitioning, some partitions will have size $s{=}\lfloor d/(2b+1)\rfloor$ and others $s+1$.

$\widehat{U}_H \subseteq \cap_{i=1}^h \bar{K}_i$. To also prove that $\cap_{i=1}^h \bar{K}_i \subseteq \widehat{U}_H$, let us suppose that $C \in \cap_{i=1}^h \bar{K}_i$, and that, by contradiction, C does not overlap with h partitions as C does not contain any feature in partition $P_j \in H$. This implies that $C \in K_j$, since K_j contains by construction all the tuples of b features that *do not* overlap with $P_j \in H$. Since $C \in K_j \Rightarrow C \notin \bar{K}_j$, which in turn implies that $C \notin \cap_{i=1}^h \bar{K}_i$. This contradicts our hypothesis and concludes the proof. □

We can finally write the probability $\Pr(h)$ as follows:

$$\Pr(h) = \frac{\sum_{H \subset \mathcal{P}, |H|=h} |\widehat{U}_H|}{\binom{d}{b}} = \frac{\binom{2b+1}{h} \, |\bigcap_{i=1}^h \bar{K}_i|}{\binom{d}{b}}, \tag{2}$$

where the factor $\binom{2b+1}{h}$ counts the number of ways of selecting $H \subset \mathcal{P}$, with $|H| = h$, while the denominator $\binom{d}{b}$ is the number of all the possible attacks on the full feature space.

To compute the cardinality of $\widehat{U}_H$, we resort to the complementary formulation of the inclusion-exclusion principle:

$$|\widehat{U}_H| = \left|\bigcap_{i=1}^h \bar{K}_i\right| = \left|U_H - \bigcup_{i=1}^h K_i\right| =$$
$$= |U_H| - \sum_{i=1}^h |K_i| + \sum_{1 \le i<j \le h} |K_i \cap K_j| + \ldots + (-1)^h |K_1 \cap \ldots \cap K_h|.$$

We can rewrite the above formula as follows:

$$|\widehat{U}_H| = \left|\bigcap_{i=1}^h \bar{K}_i\right| = \binom{sh}{b} - \sum_{i=1}^h \binom{s(h-1)}{b} + \sum_{1 \le i<j \le h} \binom{s(h-2)}{b} + \ldots + = \sum_{k=0}^h (-1)^k \binom{h}{k} \binom{s(h-k)}{b}$$

where the cardinality of the intersection of k distinct K_i sets is computed as $\binom{s(h-k)}{b}$, resulting by the attacks to b features limited to the remaining of $h-k$ partitions in $H \subset \mathcal{P}$, each of size s.

We can finally conclude by rewriting the formula in Eq. 2:

$$\Pr(h) = \frac{\binom{2b+1}{h} \sum_{k=0}^h (-1)^k \binom{h}{k} \binom{s(h-k)}{b}}{\binom{d}{b}} \tag{3}$$

We can now compute the probability $\Pr(\mathcal{T_P})$ that $\mathcal{T_P}$ is accurate. The forest $\mathcal{T_P}$ of $2b+1$ trees is accurate if the number ϵ of erroneous trees is at most b. There are several cases that lead to this outcome: we may have h trees affected by the attacker and the remaining $\epsilon - h$ trees being wrong independently of the attacker, each with probability e. Moreover, those $\epsilon - h$ can be selected at random among the $|\mathcal{P}| - h$ ones. We define as $\Pr(\epsilon|h)$ the probability that a total of ϵ trees in $\mathcal{T_P}$ provide a wrong prediction, given that h were already harmed by the attacker:

$$\Pr(\epsilon|h) = \binom{|\mathcal{P}|-h}{\epsilon-h} e^{\epsilon-h} (1-e)^{|\mathcal{P}|-\epsilon}. \tag{4}$$

Given that the attacker may negatively affect at least one and at most b trees, we have that the probability of $\mathcal{T_P}$ being correct is:

$$\Pr(\mathcal{T_P}) = \sum_{\epsilon=1}^b \sum_{h=1}^\epsilon \underbrace{\Pr(h)}_{attacked} \underbrace{\Pr(\epsilon|h)}_{inaccurate}. \tag{5}$$

Figure 3a shows the probability $\Pr(h)$ for different values of b. The number of features in the dataset was chosen so as to have the multiple of $2b+1$ closest to 100. We highlight that the probability of "hitting" only a few partitions is usually small: when $b = 5$, there are 11 partitions

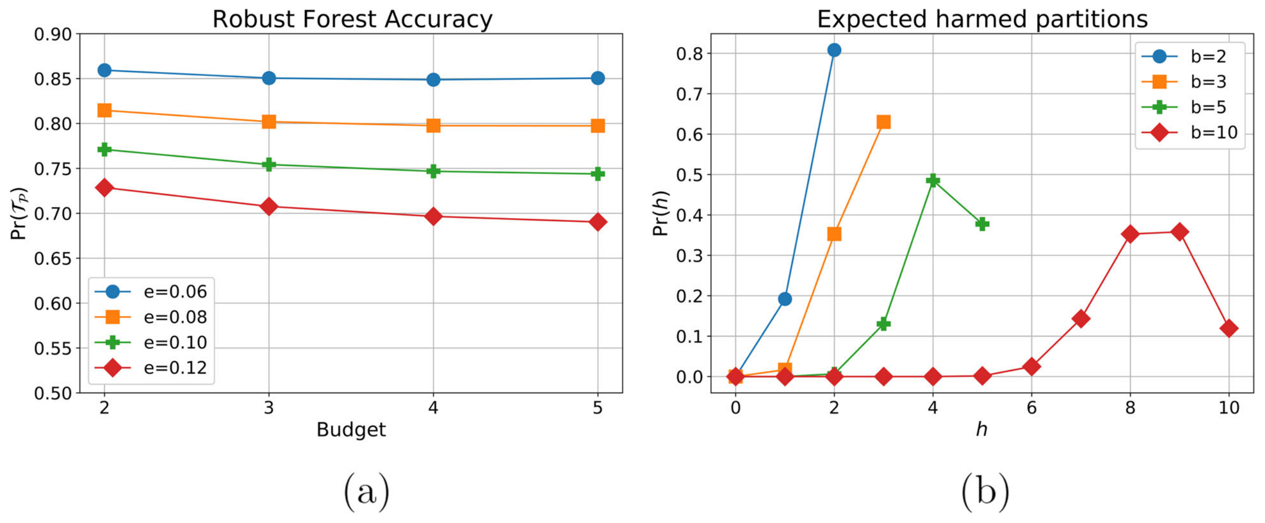

Fig. 3 Expected accuracy

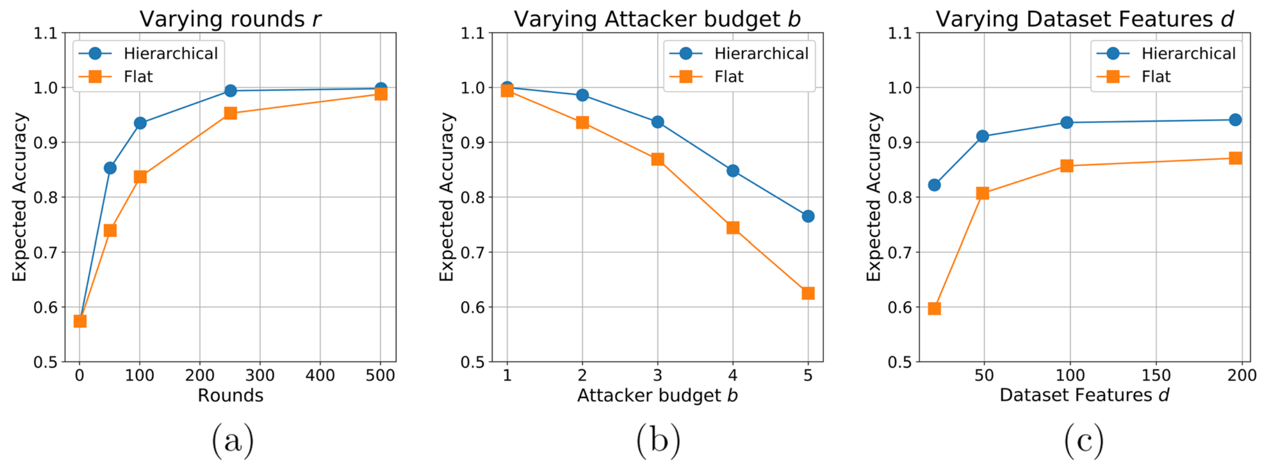

Fig. 4 Flat vs. hierarchical ensembling

and there is less than 15% probability of hitting 3 partitions and about 85% of hitting at least 4 partitions; when $b = 10$, there is nearly 0 probability of hitting less than 6 partitions out of 21, and there is about 80% probability of hitting at least 8 partitions. Interestingly, hitting $b = 10$ partitions is not the most probable event.

Clearly, we are hypothesizing that our attacker is very powerful, as it can likely impact on features used by a significant portion of trees in a forest $\mathcal{T}_\mathcal{P}$ by breaking these trees. This is partially confirmed in Fig. 3b where the error rate probability e of trees is also considered. The expected accuracy $\Pr(\mathcal{T}_\mathcal{P})$ is not large, but, interestingly, increasing the attacker's budget does not have a significant impact. It is true that the attacker can harm a good number of partitions, yet the majority of them are expected to provide a correct results, even considering e, eventually providing sufficient accuracy figures.

Finally, we report in Fig. 4 the expected accuracy comparing the flat f-FPF versus the hierarchical h-FPF ensemble prediction methods. Results were computed by simulation and, when not specified, considering a number of features $d = 99$, an attacker's budget $b = 3$, number of rounds $r = 101$, and a large tree error rate $e = 0.18$. Figure 4a shows how using multiple rounds quickly improves the ensemble accuracy. In the long run, hierarchical and flat solutions may converge, but the hierarchical approach provides better results earlier (up to 10% improvement). Increasing the attacker's budget has a significant impact on the accuracy, yet the hierarchical approach is more robust as shown in Fig. 4b. Clearly, the harm of the attack depends on the number of features available. In datasets with insufficient features to properly populate the partitions, the accuracy might decrease. Otherwise, the number of features is not very relevant, and the accuracy is stable once a good number of features is available as reported in Fig. 4c.

Abbreviations
ML: Machine learning; FPF: Feature Partitioned Forest; RSM: Random Subspace method; RT: Robust trees; h-FPF: Hierarchical-Feature Partitioned Forest; f-FPF: Flat-Feature Partitioned Forest; SVM: Support-vector machines; FLB: Fast accuracy lower bound; ELB: Exhaustive accuracy lower bound; ACC: Accuracy; RF: Random forest; BF: Brute force

Acknowledgements
Not applicable.

Authors' contributions
All authors have contributed equally. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The datasets generated and/or analyzed during the current study are available in the UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/spambase, https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) and https://archive.ics.uci.edu/ml/datasets/wine. The datasets generated and/or analyzed during the current study are available in the The MNIST Database repository, http://yann.lecun.com/exdb/mnist/.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 5 June 2021 Accepted: 13 November 2021
Published online: 11 December 2021

References
1. L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, J. D. Tygar, in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. Adversarial machine learning, (2011), pp. 43–58
2. B. Biggio, F. Roli, Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recognit. **84**, 317–331 (2018)
3. B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, F. Roli, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Evasion attacks against machine learning at test time, (2013), pp. 387–402
4. A. M. Nguyen, J. Yosinski, J. Clune, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Deep neural networks are easily

fooled: high confidence predictions for unrecognizable images, (2015), pp. 427–436
5. N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. The limitations of deep learning in adversarial settings, (2016), pp. 372–387
6. S. Moosavi-Dezfooli, A. Fawzi, P. Frossard, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Deepfool: a simple and accurate method to fool deep neural networks, (2016), pp. 2574–2582
7. D. Lowd, C. Meek, in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. Adversarial learning, (2005), pp. 641–647
8. B. Biggio, B. Nelson, P. Laskov, in *Asian Conference on Machine Learning*. Support vector machines under adversarial label noise, (2011), pp. 97–112
9. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus, in *ICLR*. Intriguing properties of neural networks, (2014)
10. I. J. Goodfellow, J. Shlens, C. Szegedy, in *ICLR*. Explaining and harnessing adversarial examples, (2015)
11. G. Tolomei, F. Silvestri, A. Haines, M. Lalmas, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Interpretable predictions of tree-based ensembles via actionable feature tweaking, (2017), pp. 465–474
12. F. Chollet, *Deep learning with Python*, 1st edn. (Manning Publications Co., USA, 2017)
13. B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. Lau, S. J. Lee, S. Rao, A. Tran, J. D. Tygar, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Near-optimal evasion of convex-inducing classifiers, (2010), pp. 549–556
14. B. Biggio, G. Fumera, F. Roli, Security evaluation of pattern classifiers under attack. IEEE Trans. Knowl. Data Eng. **26**(4), 984–996 (2014)
15. N. Srndic, P. Laskov, in *2014 IEEE Symposium on Security and Privacy*. Practical evasion of a learning-based classifier: a case study, (2014), pp. 197–211
16. A. Kantchelian, J. D. Tygar, A. D. Joseph, in *International Conference on Machine Learning*. Evasion and hardening of tree ensemble classifiers, (2016), pp. 2387–2396
17. N. Carlini, D. A. Wagner, in *2017 IEEE Symposium on Security and Privacy (SP)*. Towards evaluating the robustness of neural networks, (2017), pp. 39–57
18. H. Dang, Y. Huang, E. Chang, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Evading classifiers by morphing in the dark, (2017), pp. 119–133
19. H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, F. Roli, Support vector machines under adversarial label contamination. Neurocomputing. **160**, 53–62 (2015)
20. S. Gu, L. Rigazio, in *ICLR, Workshop Track Proceedings*. Towards deep neural network architectures robust to adversarial examples, (2015)
21. N. Papernot, P. D. McDaniel, X. Wu, S. Jha, A. Swami, in *2016 IEEE Symposium on Security and Privacy (SP)*. Distillation as a defense to adversarial perturbations against deep neural networks, (2016), pp. 582–597
22. H. Chen, H. Zhang, D. S. Boning, C. Hsieh, in *International Conference on Machine Learning*. Robust decision trees against adversarial examples, (2019), pp. 1122–1131
23. S. Calzavara, C. Lucchese, G. Tolomei, S. A. Abebe, S. Orlando, Treant: training evasion-aware decision trees. Data Min. Knowl. Discov. **34**(5), 1390–1420 (2020)
24. M. Andriushchenko, M. Hein, in *NeurIPS*. Provably robust boosted decision stumps and trees against adversarial attacks, (2019), pp. 12997–13008
25. A. Kantchelian, J. D. Tygar, A. Joseph, in *International Conference on Machine Learning*. Evasion and hardening of tree ensemble classifiers, (2016), pp. 2387–2396
26. J. Su, D. V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**(5), 828–841 (2019)
27. K. Simonyan, A. Zisserman, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, ed. by Y. Bengio, Y. LeCun. Very Deep Convolutional Networks for Large-Scale Image Recognition, (2015). http://arxiv.org/abs/1409.1556
28. H. Chen, H. Zhang, S. Si, Y. Li, D. Boning, C.-J. Hsieh, in *Advances in Neural Information Processing Systems*. Robustness verification of tree-based models, (2019), pp. 12317–12328
29. F. Ranzato, M. Zanella, 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*. Abstract Interpretation of Decision Tree Ensemble Classifiers (AAAI Press, 2020), pp. 5478–5486. https://aaai.org/ojs/index.php/AAAI/article/view/5998
30. M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, C.-J. Hsieh, in *International Conference on Learning Representation (ICLR)*. Query-efficient hard-label black-box attack: an optimization-based approach, (2019)
31. S. Calzavara, C. Lucchese, G. Tolomei, S. A. Abebe, S. Orlando. Treant: training evasion-aware decision trees, vol. 34, (2020), pp. 1390–1420. https://doi.org/10.1007/s10618-020-00694-9
32. D. Vos, S. Verwer, in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research), vol. 139*, ed. by M. Meila, T. Zhang. Efficient Training of Robust Decision Trees Against Adversarial Examples (PMLR, 2021), pp. 10586–10595
33. F. Ranzato, M. Zanella, in *GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021*, ed. by F. Chicano, K. Krawiec. Genetic adversarial training of decision trees (ACM, 2021), pp. 358–367. https://doi.org/10.1145/3449639.3459286
34. T. K. Ho, The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1998)
35. B. Biggio, G. Fumera, F. Roli, Multiple classifier systems for robust classifier design in adversarial environments. Int. J. Mach. Learn. Cybern. **1**(1-4), 27–41 (2010)
36. R. Gandhi, S. Khuller, A. Srinivasan, Approximation algorithms for partial covering problems. J. Algorithms. **53**(1), 55–84 (2004). https://doi.org/10.1016/j.jalgor.2004.04.002
37. L. Breiman, Random forests. Mach. Learn. **45**(1), 5–32 (2001)

Publisher's Note