Timber! Poisoning Decision Trees

Stefano Calzavara
Università Ca' Foscari Venezia
Venice, Italy
stefano.calzavara@unive.it

Lorenzo Cazzaro
Università Ca' Foscari Venezia
Venice, Italy
lorenzo.cazzaro@unive.it

Massimo Vettori
Università Ca' Foscari Venezia
Venice, Italy
884477@stud.unive.it

***Abstract*—We present Timber, the first white-box poisoning attack targeting decision trees. Timber is based on a greedy attack strategy that leverages sub-tree retraining to efficiently estimate the damage caused by poisoning a given training instance. The attack relies on a tree annotation procedure, which enables the sorting of training instances so that they are processed in increasing order of the computational cost of sub-tree retraining. This sorting yields a variant of Timber that supports an early stopping criterion, designed to make poisoning attacks more efficient and feasible on larger datasets. We also discuss an extension of Timber to traditional random forest models, which is valuable since decision trees are typically combined into ensembles to improve their predictive power. Our experimental evaluation on public datasets demonstrates that our attacks outperform existing baselines in terms of effectiveness, efficiency, or both. Moreover, we show that two representative defenses can mitigate the effect of our attacks, but fail to effectively thwart them.**

***Index Terms*—adversarial machine learning, decision trees, poisoning attacks, tree ensembles.**

I. Introduction

Our daily activities are becoming increasingly more reliant on machine learning, yet the trustworthiness of machine learning has been questioned from different points of views. A prominent class of threats against machine learning is represented by *poisoning attacks* [1]. Poisoning attacks break the implicit assumption that data used to train machine learning models are representative of actual test data that will be seen upon model deployment. In particular, if the attacker can compromise the integrity of training data, e.g., by crafting incorrectly labeled instances, the training algorithm may operate on low-quality data yielding models with poor accuracy. The ultimate goal of a poisoning attack is to determine an effective way to pollute training data so as to force wrong model predictions at test time.

Poisoning attacks have been extensively investigated in the research literature. Multiple papers proposed poisoning attacks against different types of machine learning models, including support vector machines [2], [3], [4], [5], linear classifiers [6] and neural networks [7], [8]. Unfortunately, prior research largely neglected the investigation of poisoning attacks against *decision trees*, which are still one of the most effective machine learning models operating on tabular data [9]. Decision trees are peculiar because they are non-differentiable models, meaning that the loss function that they optimize does not have a gradient. This implies that poisoning attacks against decision trees cannot be formalized in terms of a traditional bilevel optimization problem in the style of [1] and new custom attack algorithms must be designed. A possible way around this limitation is using *black-box* attack strategies, which are model-agnostic and proved effective in some practical settings [10], [11], [12], [13], [14], [15]. Unfortunately, black-box attack strategies assume that the attacker does not know anything about the training process, hence they may underestimate the attacker's capabilities. In-depth understanding of poisoning attacks against decision trees requires the careful design and evaluation of *white-box* attack strategies, where the attacker abuses the inner workings of the tree learning algorithm to their advantage. This way, we may be able to perform a conservative security analysis which takes into account more powerful attackers with additional information about their target.

Contributions: Our contributions are as follows:

1) We propose Timber, the first white-box poisoning attack designed to target decision trees. Timber is based on a greedy attack strategy that leverages sub-tree retraining to efficiently estimate the damage caused by poisoning a given training instance. Timber relies on a tree annotation procedure which enables sorting training instances so that they are processed in increasing order of computational cost of sub-tree retraining. This sorting yields a variant of Timber that supports an early stopping criterion designed to make poisoning attacks more efficient and feasible on larger datasets.
2) We discuss how to generalize Timber from individual decision trees to decision tree ensembles, in

particular traditional random forest models based on independently trained trees [16]. This generalization is useful because decision trees are rarely used in isolation and ensembles of decision trees are normally used to solve challenging classification problems.

3) We experimentally assess the performance of our attacks on public datasets, showing that they outperform existing baselines in terms of effectiveness, efficiency or both. We also show that two representative defenses can mitigate the effect of our attacks, but fail to effectively thwart them.

Code availability: To support reproducible research, we make our code available on GitHub [17].

II. Background

We here present the main technical ingredients required to understand the rest of the paper.

A. Supervised Learning

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimensional vector space of real-valued *features*. An *instance* $\vec{x} \in \mathcal{X}$ is a d-dimensional feature vector $\langle x_1, x_2, \ldots, x_d \rangle$ representing an object in the feature space $\mathcal{X}$. Each instance is assigned a class label $y \in \mathcal{Y}$ by an unknown *target* function $f : \mathcal{X} \to \mathcal{Y}$. Supervised learning algorithms automatically learn a *classifier* $g : \mathcal{X} \to \mathcal{Y}$ from a *training set* of correctly labeled instances $\mathcal{D}_{\mathit{train}} = \{(\vec{x}_i, f(\vec{x}_i))\}_i$, with the goal of approximating the unknown target function f as accurately as possible.

The performance of classifiers is empirically estimated on a *test set* of correctly labeled instances $\mathcal{D}_{\mathit{test}} = \{(\vec{z}_i, f(\vec{z}_i))\}_i$, disjoint from the training set, yet drawn from the same data distribution. A traditional measure to assess the performance of classifiers is called *accuracy*, defined as the percentage of instances of the test set where the classifier performs a correct prediction. For simplicity, we here focus on binary classification tasks, i.e., we let $\mathcal{Y} = \{+1, -1\}$. This is a convenient setting to study poisoning attacks, because it allows us to represent poisoning in terms of *label flipping* attacks [1], where the attacker replaces a correctly labelled instance $(\vec{x}_i, f(\vec{x}_i)) \in \mathcal{D}_{\mathit{train}}$ with the mislabelled instance $(\vec{x}_i, -f(\vec{x}_i))$.

B. Decision Trees

We focus on traditional *binary decision trees* for classification [18]. Decision trees can be inductively defined as follows: a decision tree t is either a leaf $\lambda(y)$ for some label $y \in \mathcal{Y}$ or an internal node $\sigma(f, v, t_l, t_r)$, where $f \in \{1, \ldots, d\}$ identifies a feature, $v \in \mathbb{R}$ is a threshold for the feature, and t_l, t_r are decision trees (left and right child). At test time, the instance $\vec{x}$ traverses the tree t until it reaches a leaf $\lambda(y)$, which returns the prediction y, denoted by $t(\vec{x}) = y$. Specifically, for each traversed tree node $\sigma(f, v, t_l, t_r)$, $\vec{x}$ falls into the left sub-tree t_l if $x_f \leq v$, and into the right sub-tree t_r otherwise. Figure 1 represents an example decision tree of depth 2, which assigns label $+1$ to the instance $\langle 12, 7 \rangle$ and label -1 to the instance $\langle 8, 6 \rangle$.

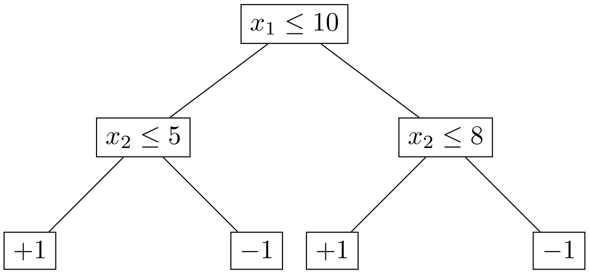


Fig. 1: Example of decision tree.

Decision trees are learned by an iterative process starting from a single leaf, which is grown into a full-fledged tree with the goal of minimizing the *entropy* of the leaves.[1] For any $\mathcal{D} \subseteq \mathcal{D}_{\mathit{train}}$, we define its entropy $H(\mathcal{D})$ as follows:

$$\begin{aligned} H(\mathcal{D}) = -(&|\mathcal{D}_{+1}|/|\mathcal{D}| \cdot \log_2(|\mathcal{D}_{+1}|/|\mathcal{D}|) \\ &+ |\mathcal{D}_{-1}|/|\mathcal{D}| \cdot \log_2(|\mathcal{D}_{-1}|/|\mathcal{D}|)), \end{aligned}$$

where $\mathcal{D}_y = \{(\vec{x}, y') \in \mathcal{D} \mid y' = y\}$ is the restriction of $\mathcal{D}$ to the instances with label y.

The training algorithm Tree-Train($\mathcal{D}$) is presented in Algorithm 1 and is invoked with input $\mathcal{D} = \mathcal{D}_{\mathit{train}}$. The algorithm splits a leaf including the data $\mathcal{D}$ by extracting a set of candidates splits, noted *splits*($\mathcal{D}$), which may be used to grow the tree by replacing the leaf with a new decision tree of depth one. The simplest definition of *splits*($\mathcal{D}$) is *splits*($\mathcal{D}$) $= \{(f, v) \mid \exists (\vec{x}, y) \in \mathcal{D} : x_f = v\}$, but implementations may vary and we do not make any assumption on how the candidate splits are computed. The training algorithm computes, for each $(f, v) \in$ *splits*($\mathcal{D}$), how the entropy would change if the leaf was grown into a tree of the form $\sigma(f, v, \lambda(y_l), \lambda(y_r))$ for some y_l, y_r minimizing the prediction errors in the leaves. This is done by computing the *information gain*

[1] Decision trees can also be trained to minimize other measures, such as *Gini impurity*. Our proposal can be readily generalized to other measures with limited effort.

Algorithm 1 Training algorithm for decision trees.

```
function TREE-TRAIN($\mathcal{D}$)
    $best\text{-}split \leftarrow \bot$
    $best\text{-}gain \leftarrow 0$
    for $(f, v) \in splits(\mathcal{D})$ do
        if $G(\mathcal{D}, f, v) > best\text{-}gain$ then
            $best\text{-}split \leftarrow (f, v)$
            $best\text{-}gain \leftarrow G(\mathcal{D}, f, v)$
    if $best\text{-}split = (f^*, v^*)$ then
        $t_l \leftarrow$ TREE-TRAIN($\mathcal{D}^{f^* \leq v^*}$)
        $t_r \leftarrow$ TREE-TRAIN($\mathcal{D}^{f^* > v^*}$)
        return $\sigma(f^*, v^*, t_l, t_r)$
    else
        if $|\mathcal{D}_{+1}| \geq |\mathcal{D}_{-1}|$ then return $\lambda(+1)$
        else return $\lambda(-1)$
```

$G(\mathcal{D}, f, v)$ resulting from partitioning $\mathcal{D}$ by using the split (f, v), which is defined as follows:

$$G(\mathcal{D}, f, v) = H(\mathcal{D}) - (|\mathcal{D}^{f \leq v}|/|\mathcal{D}| \cdot H(\mathcal{D}^{f \leq v}) + |\mathcal{D}^{f > v}|/|\mathcal{D}| \cdot H(\mathcal{D}^{f > v})),$$

where $\mathcal{D}^{f \leq v} = \{(\vec{x}, y) \in \mathcal{D} \mid x_f \leq v\}$ and $\mathcal{D}^{f > v} = \{(\vec{x}, y) \in \mathcal{D} \mid x_f > v\}$. Once the best split (f^*, v^*) has been found, the original leaf is replaced by the decision tree $\sigma(f^*, v^*, t_l, t_r)$, where t_l and t_r are the decision trees recursively trained over $\mathcal{D}^{f^* \leq v^*}$ and $\mathcal{D}^{f^* > v^*}$ respectively. The tree construction terminates when none of the possible splits enables some information gain or some other termination criterion is met, e.g., the tree exceeds a maximum depth (for simplicity, alternative termination criteria are not shown in the pseudo-code).

The computational complexity of the tree training algorithm is $O(d \cdot n^2 \log(n))$, where d is the number of features and n is the size of the training set [18], [19]. This complexity assumes that, for each feature f, the information gain is computed by ordering the training data based on the value of f, which simplifies the computation of the partitioning induced by each candidate split (f, v). In particular, each of the d features requires a sorting operation of cost $O(n \cdot \log(n))$ to find the best split. This must be repeated for each node in the decision tree, whose number is bounded above by $O(n)$.

C. Tree Ensembles

Decision trees are effective models for small datasets, but they may offer suboptimal performance on large and complicated datasets. The predictive power of tree-based classifiers can be increased by training *ensembles* of multiple decision trees, using algorithms like Random Forest (RF [16]) and Gradient Boosted Decision Trees (GBDT [20]). RF is based on the training of multiple independent trees, each trained on a subset of the training set and a subset of the features. The ensemble prediction is then performed by aggregating individual tree predictions, e.g., using hard majority voting. GBDT instead is a more sophisticated approach in which trees are iteratively trained, with each tree t_i being trained with the goal of reducing the prediction errors made by the previously trained trees $t_1, \ldots, t_{i-1}$.

III. Poisoning Decision Trees

We here introduce our threat model, we explain the key challenges of our research and we propose Timber, our poisoning attack operating against decision trees. We also discuss how Timber can be extended to decision tree ensembles, in particular based on the RF algorithm.

A. Threat Model

In a poisoning attack, the attacker targets the training data or the training algorithm to compromise the performance of the classifier at test time. To define our threat model for poisoning attacks, we start from a recent survey systematizing research in the field, which defines a clear attack framework and introduces terminology [1].

We focus on *availability* violations, i.e., the attacker's goal is to decrease the accuracy of the classifier that is trained by the learning algorithm: the more the accuracy is downgraded, the more the attack is considered effective. Moreover, we focus on *white-box* attacks, i.e., the attacker has complete knowledge of the training data, the training algorithm, and the model hyperparameters. In this way, we identify insights about decision tree construction that the attacker might abuse and we estimate security under the conservative assumption that the attacker has full knowledge of the training process. Finally, we assume that the attacker alters a subset of the training data collected by the target. The attacker can only modify the training labels, thus it does not perturb the features of any training sample, which is often referred to as a *label flipping* attack. The attacker can flip the labels of up to k arbitrarily chosen instances of the training set, leading to a *poisoned* dataset which is used to train the classifier. Label flipping is an appropriate threat model for scenarios where the labeling process is adversarial. For instance, in product rating systems, an attacker may assign low scores to targeted products in a public catalog to manipulate a recommender system. Similarly, an attacker may create a rogue mailbox to mislabel spam messages as ham with the goal of fooling a remote classifier trained over user reports.

The objective of our research is to find an algorithm to effectively identify the k instances to attack out of the n training instances, with the goal of compromising the accuracy of the trained classifier.

B. Baselines and Challenges

We are not aware of any poisoning attack in the literature that specifically targets decision trees. A few research papers present experiments targeting decision trees (among other models) by means of model-agnostic *black-box* poisoning techniques, e.g., based on the distribution of different features of the training data [21]. Similar approaches are useful to empirically assess the dangers posed by poisoning attacks, but they make the assumption that the attacker knows nothing about the training process and do not offer any guarantees about their practical effectiveness. This motivates the importance of white-box attacks abusing the inner workings of the tree learning algorithm to magnify the advantage of the attacker and enable a conservative security analysis. There are a few white-box poisoning attacks in the literature that work for entire classes of machine learning models, such as differentiable models [3], [5], [6], [7], [8]. Unfortunately, these attacks do not generalize to decision trees, because decision trees are not differentiable.

We here explain why poisoning decision trees is challenging by presenting a few baseline attack methods. The first observation we make is that finding the best k instances to flip by exhaustive enumeration of the subsets of instances is impossible, because there are $\binom{n}{k}$ subsets to test. Even for a small dataset of $n = 1{,}000$ instances and a tiny $k = 10$, there are around 2.634×10^{23} available combinations, which is intractable. A possible solution is then to use a heuristic *greedy* approach. We first train a decision tree t over $\mathcal{D}_{train}$ and we then try to flip each instance of $\mathcal{D}_{train}$ before training a new tree t'. After trying all the instances, we flip the one leading to the tree with the lowest accuracy and we iterate the process for k rounds, leading to $n \cdot k$ trees being grown. This complexity may be acceptable for small datasets, as shown in the experimental evaluation of label flipping attacks by Paudice et al. [5]. Unfortunately, for a medium dataset of $n = 5{,}000$ instances and $k = 500$, the greedy attack may already construct up to 2.5M trees.

To further speed up the attack, one might revise the proposed greedy approach to include an *early stopping* criterion, e.g., when the attack finds any instance leading to some accuracy loss, the attack flips its label and moves to the next round. Of course, this does not change the worst-case complexity of the algorithm, but in practice this variant of the attack is expected to be much faster. To estimate the benefits of early stopping, assume that on average just 10% of the training set must be analyzed to identify an instance leading to some accuracy loss. For a dataset of $n = 5{,}000$ instances and $k = 500$, this variant of the attack may construct around 250k trees.

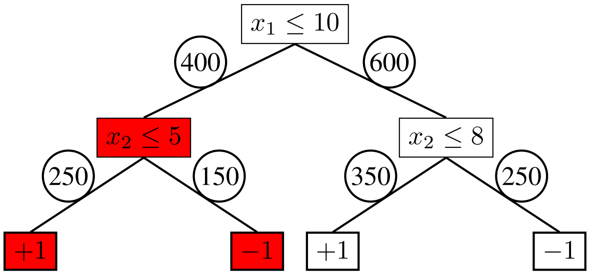


Fig. 2: Intuition of the Timber attack. If flipping the label of the instance $(\vec{x}, y)$ does not invalidate the best split of the root and $\vec{x}$ falls in its left child, only the sub-tree in red (including 400 instances) may need retraining.

C. Timber: Attack Overview

Our attack called Timber extends a traditional greedy attack strategy (possibly with early stopping) to improve its efficiency and make it usable in practice. Greedy attack strategies require the construction of a significant number of decision trees, as discussed in the previous section. The main intuition of our attack is that, if we can make decision tree construction itself more efficient, we can make greedy attack strategies scale to larger datasets. Recall that training a decision tree has a computational complexity of $O(d \cdot n^2 \log(n))$, because we split $O(n)$ nodes by paying a cost of $O(d \cdot n \log(n))$ for each node. A key insight of our attack is that the worst-case complexity of node splitting $O(d \cdot n \log(n))$ is very pessimistic, because the training set is partitioned across nodes when the tree is grown and becomes increasingly smaller, e.g., if the best split of the root is (f, v), the recursive calls of the training algorithm operate over the smaller datasets $\mathcal{D}^{f \leq v}$ and $\mathcal{D}^{f > v}$ respectively. For example, if (f, v) evenly splits the training set, the two recursive calls operate on $n/2$ instances, meaning that splitting each of the new nodes costs significantly less than splitting the root. Hence, nodes deeper in the tree are much cheaper to split than nodes higher in the tree, with the root being the most expensive node to split.

Since greedy attack strategies operate by flipping one instance at a time, the impact of this single instance on the trained tree is expected to be small in practice. In particular, flipping the label of a single training instance is unlikely to affect the best split of the root, because the best split is identified by considering $n \gg 1$ instances. However, the lower we descend in the decision tree, the higher the odds that the flipped instance affects the best split. In our example where the root evenly splits the training instances, the best split of a child of the root is found by processing just $n/2$ instances, meaning that a single label flip has a higher chance of changing the best split. This means that a single label flip normally preserves most of the structure of the decision tree and just a small, deep sub-tree where the best split has been invalidated needs to be retrained. If the root of this sub-tree includes just a small part of the training data, sub-tree retraining enables a significant speedup compared to retraining the entire tree from scratch. This intuition is shown in Figure 2, where the numbers in the circles show how the 1,000 instances of the training set are split across the nodes upon tree construction. If flipping the label of the instance $\langle 8, 6 \rangle$ does not invalidate the best split of the root $x_1 \leq 10$, we may recursively focus on its left child (the right child can be ignored, because the poisoned instance falls on the left). Then, if the label flip invalidates the best split of the left child $x_2 \leq 5$, we only need to retrain the sub-tree in red, whose construction only involves 400 instances (40% of the training set).

Our attack operates by annotating each node of the decision tree with the set of the training instances which would not change the current node best split, even if their label was flipped. We refer to such instances as the *stable instances* of the node. By leveraging this information, we can determine the portion of the decision tree impacted by a poisoning attack and estimate the attack effectiveness by retraining a single sub-tree, rather than the entire tree. To identify the stable instances within decision tree nodes, we leverage a compact representation of possible attacker's actions and their corresponding impact on the information gain computed during tree learning. The intuition is discussed for the dataset $\mathcal{D}$ in Figure 3, where the sun represents instances of the positive class, the moon represents instances of the negative class, and the dotted line shows the best split (f, v) of a tree node. The attacker has four possible options: (i) flip a positive instance on the left of the split, (ii) flip a negative instance on the left of the split, (iii) flip a positive instance on the right of the split, or (iv) flip a negative instance on the right of the split. In all four cases, the identity of the chosen instance is irrelevant, because the information gain depends just on the number of positive and negative instances on each side of the split.

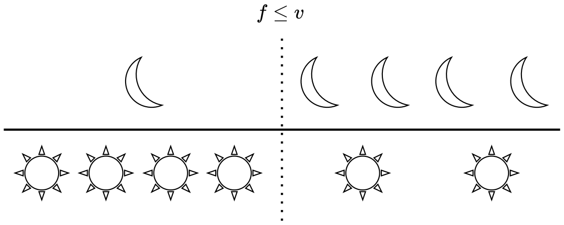


Fig. 3: Splitting the dataset $\mathcal{D}$ based on the split (f, v). Poisoning attacks can target positive or negative instances on the left or on the right of the split, leading to four attack possibilities that we must account for.

In our example, the initial entropy is 0.99 and the best split (f, v) has an information gain of 0.16. We then observe that, if the attacker flipped a positive instance on the left of the split, the left side of the split would include three positive instances and two negative instances. In this case the entropy of the data would stay the same, but the new information gain of the split would become 0.05. We compactly represent this information with the triple $(0.05, f \leq v, +1)$, meaning that flipping a positive instance on the left of the split (f, v) would lead to a new information gain of 0.05. We can similarly compute the other three triples $(0.44, f \leq v, -1), (0.31, f > v, +1), (0.07, f > v, -1)$, thus capturing the effect on the split (f, v) of all the possible attacker's actions in terms of a set of four triples, denoted by $G^*(\mathcal{D}, f, v)$. Observe that $G^*(\mathcal{D}, f, v)$ includes *at most* four elements, because positive or negative instances may not be present on the left or on the right of the split, meaning that some flips may be impossible. By computing $G^*(\mathcal{D}, f', v')$ for each other possible split (f', v'), it is possible to determine whether any attacker's action might lead to the identification of a new best split, i.e., a split with a higher information gain than (f, v).

Our attack algorithm trains a decision tree over the clean training data and then operates in two steps. The first step is *tree annotation* (described in Section III-D): we annotate each node of the decision tree with the set of its stable instances. The annotation process is efficient because it boils down to checking the information available in G^*, which can be directly computed during decision tree construction, because the training algorithm computes the information gain G of all the possible splits anyways. Computing G^* requires a simple adaptation of the formula used to compute G. The second step of the attack is *label flipping* (described in Section III-E): we use the computed stability information to identify the instances to prioritize in the poisoning attack to improve its efficiency. For each such instance, we flip its label and we retrain just the sub-tree of the decision tree that may be affected by this change to identify the accuracy loss. After choosing the instance to attack, e.g., the one leading to the highest accuracy loss or the first instance introducing some loss, we train a new decision tree over the poisoned dataset and we start the attack again until the maximum number of label flips has been reached.

D. Tree Annotation

We extend each node of the decision tree t with some auxiliary information: (i) the set of the training instances $t.\mathit{train}$ used in the node construction, which can be readily identified by instrumenting the training algorithm, and (ii) the set of the stable instances $t.\mathit{stable}$, which is computed by the tree annotation function ANNOTATE in Algorithm 2. The function takes as input a decision tree and returns its annotated version. The algorithm initially assumes all the training instances to be stable and prunes the set of stable instances whenever it finds evidence that flipping a label may invalidate the best split (f, v). This can only happen if there exists another split (f', v') leading to a higher information gain than (f, v) after label flipping, or when the information gain is the same but (f', v') is processed before (f, v) during tree construction. Assuming $\mathcal{D} = t.\mathit{train}$, this can be determined by checking each $(g, \phi, y) \in G^*(\mathcal{D}, f, v)$ against each $(g', \phi', y') \in G^*(\mathcal{D}, f', v')$: if $g' > g$, or $g' = g$ and the split (f', v') is processed before (f, v) in the lexicographic order, then all the instances in the intersection $I = \mathcal{D}_y^\phi \cap \mathcal{D}_{y'}^{\phi'}$ must be removed from V. To understand the definition of I, observe that $I \neq \emptyset$ when $y = y'$ and there exist instances satisfying the predicate $\phi \wedge \phi'$, i.e., there exists a class including instances falling in the portion of the feature space common to ϕ and ϕ'. For any such instance, a label flip would make (f', v') the new best split in place of (f, v).

An important point to note is that the identification of the stable instances can be directly embedded within the tree construction at training time. Indeed, the tree construction algorithm (Algorithm 1) must compute the information gain G for all the possible splits anyway. We can then modify the algorithm to compute the set G^* for all the possible splits, meaning that for each split we do not compute just a single information gain, but five (at most). This computation is very efficient, because it suffices to update the number of the positive and negative instances falling on the left and on the right of the split after label flipping, without any need to scan the entire dataset again. Our implementation directly integrates the computation of the stable instances in the tree construction algorithm of scikit-learn [22].

Algorithm 2 Tree annotation algorithm

```
function ANNOTATE($t$)
    $t.stable \leftarrow t.train$
    if $t = \sigma(f, v, t_l, t_r)$ then
        $\mathcal{D} \leftarrow t.train$
        for $(g, \phi, y) \in G^*(\mathcal{D}, f, v)$ do
            for $(f', v') \in splits(\mathcal{D}) \setminus \{(f, v)\}$ do
                for $(g', \phi', y') \in G^*(\mathcal{D}, f', v')$ do
                    if $g' > g \vee (g' = g \wedge (f', v') \prec (f, v))$ then
                        $I \leftarrow \mathcal{D}_y^\phi \cap \mathcal{D}_{y'}^{\phi'}$
                        $t.stable \leftarrow t.stable \setminus I$
        $t'_l \leftarrow$ ANNOTATE($t_l$)
        $t'_r \leftarrow$ ANNOTATE($t_r$)
        return $\sigma(f, v, t'_l, t'_r)$
    else
        return $t$
```

E. Label Flipping

The FLIP-RETRAIN function in Algorithm 3 takes as input an already annotated decision tree t and an instance $(\vec{x}, y) \in t.\mathit{train}$ to return the new decision tree t' obtained by replacing $(\vec{x}, y)$ with $(\vec{x}, -y)$ in the training data. The key insight of the function is that, since we precomputed stability information for all training instances, we can retrain just a specific sub-tree of t to construct the new tree t', hence t' does not need to be trained from scratch. This improves efficiency because retraining operates just over a subset of the training data rather than on the entire training set. The function recursively traverses t until it finds the first node where $(\vec{x}, y)$ is not stable, which identifies the sub-tree of t where retraining is required. Note that the retrained sub-tree must be annotated again, because its structure may have changed.

Of course, the use of sub-tree retraining alone does not necessarily suffice to yield an efficient poisoning attack algorithm. Indeed, although we can efficiently estimate the impact of poisoning a given instance and retrain just a sub-tree, we may still have many instances in the training set. We may then want to restrict the number of instances to consider in our poisoning attack to further speed up the process. A relevant insight here is that the stability information computed by the annotation procedure allows us to identify those instances leading to a particularly efficient sub-tree retraining, hence we may prioritize such instances in our attack strategy. The intuition is that we can assign a *score* to each training instance $(\vec{x}, y)$ based on the percentage of training instances included in the first node of the prediction path where $(\vec{x}, y)$ is

Algorithm 3 Retraining algorithm

```
Require: $(\vec{x}, y) \in t.train$
function FLIP-RETRAIN($t, (\vec{x}, y)$)
    if $t = \sigma(f, v, t_l, t_r) \wedge (\vec{x}, y) \in t.stable$ then
        if $x_f \leq v$ then
            $t'_l \leftarrow$ FLIP-RETRAIN($t_l, (\vec{x}, y)$)
            return $\sigma(f, v, t'_l, t_r)$
        else
            $t'_r \leftarrow$ FLIP-RETRAIN($t_r, (\vec{x}, y)$)
            return $\sigma(f, v, t_l, t'_r)$
    else
        $\mathcal{D} \leftarrow (t.train \setminus \{(\vec{x}, y)\}) \cup \{(\vec{x}, -y)\}$
        $t \leftarrow$ TREE-TRAIN($\mathcal{D}$)
        return ANNOTATE($t$)
```

not stable. This number $s \in [0, 1]$ estimates the cost of sub-tree retraining when $(\vec{x}, y)$ is subject to label flipping. The score information can be used to speed up the attack by improving the efficiency of early stopping. In particular, one may sort instances based on increasing order of score, so that the attack starts from instances supporting efficient sub-tree retraining and may quickly hit the early stopping condition.

F. Extension to Tree Ensembles

Our poisoning attack was designed and presented for traditional decision trees, however decision trees are seldom used in isolation for classification tasks due to their limited predictive power. Better classifiers can be built by training ensembles of decision trees, using algorithms like RF and GBDT. Poisoning decision tree ensembles using attack strategies like the proposed greedy approach is even more computationally expensive than targeting a single decision tree, because an ensemble may include tens or hundreds of trees to retrain, meaning that effective speedup strategies are even more important. Luckily, our proposed attack can be readily generalized to decision tree ensembles of independently trained trees like RF classifiers, because our annotation procedure can be directly applied to the individual trees constituting the ensemble. Once all the trees in the ensemble have been independently trained and annotated, we may identify the candidate instances to attack just by redefining the notion of score of an instance in terms of the mean of the scores computed for the individual trees. Intuitively, this updated notion of score estimates the aggregate cost of sub-tree retraining for all the trees in the ensemble, i.e., an instance with a small score ensures efficient sub-tree retraining in all the trees. Observe that, if the training algorithm is embarrassingly parallel like RF and there are at least as many threads as the number of trees to train, it is possible to retrain all the sub-trees in parallel, hence it might be more appropriate to replace the mean of the scores with their maximum, because the execution time of the slowest thread determines the actual execution time of the attack. The effectiveness of each label flip is estimated as the accuracy loss forced on the entire forest.

We observe that our poisoning attack cannot be readily generalized to ensembles based on interdependent trees, like GBDT models. The reason is that trees composing such models are trained sequentially, because the next tree in the ensemble is trained to minimize the prediction errors produced by the previously trained trees. Assume then that our poisoning attack is also performed sequentially and let t_i be the tree under attack. Flipping the label of a training instance of t_i may also affect the construction of some tree t_j with $j < i$, meaning that the prediction errors performed by the previously trained trees may change, leading to the training of a different tree t'_i in place of t_i. This means that it would be difficult to make sub-tree retraining an effective way to optimize the efficiency of the attack. We consider the generalization of our techniques to GBDT models to be an intriguing yet challenging direction for future work.

IV. Experimental Evaluation

We perform our experimental evaluation on four public datasets: Musk2 [23], Wine [24], Spambase [25] and Breast-Cancer [26] (abbreviated as Breast). The key characteristics of the chosen datasets are reported in the appendix. All the datasets are tabular and related to binary classification tasks, thus well-suited for decision tree learning and inference. Moreover, Musk2, Spambase and Breast-Cancer have been adopted as benchmarks in related work [5], [27]. Datasets are split as 80/20 for training/testing using stratified random sampling.

A. Methodology

We assume the attacker can poison k training instances, ranging from 1% to 10% of the training set. We consider 10% to be an upper bound for realistic attacks. Moreover, we assume that the attacker operates over one of the two classes (the positive class). This is a realistic assumption because the attacker may be more interested in disrupting the detection of a specific class, e.g., the classification of spam emails as spam. Additionally, targeting a specific class may also harm the classification of instances of the other class.

Recall that we are not aware of any poisoning attacks designed to target decision trees, except those proposed in this paper. Our baselines are then general attack strategies that may be applied to any type of classifier. We consider three different groups of attack strategies:

1) *Greedy* [5] and our new attack *Timber* always iterate over all the training instances for k rounds and pick every time the one leading to the highest accuracy loss upon label flipping at each round. Timber exploits sub-tree retraining to improve efficiency. Timber is guaranteed to produce the same accuracy loss as the Greedy attack strategy, but it is expected to be faster in practice.
2) *Greedy with early stopping (GES)* and our *Timber with early stopping (TES)* iterate over all the training instances for k rounds and, at each round, terminate as soon as they encounter an instance leading to some accuracy loss upon label flipping. TES exploits sub-tree retraining and processes instances in increasing order of score, prioritizing those where sub-tree retraining is more efficient.
3) *Entropy* [10] and *K-Medoids* [10] are model-agnostic, black-box poisoning attacks operating in a single round. Entropy chooses the k instances to flip according to a score based on the entropy measure, while K-Medoids separates the training instances into two clusters and chooses the k instances to flip based on a mathematical distance.

For all the strategies in the groups 1 and 2, we assume that if none of the processed instances introduces some accuracy loss, then the attacker flips the one leading to the smallest increase in accuracy. Although this choice goes against the attacker's goal in the short term, it may lead to model changes, enabling new attacks. Moreover, this choice forces all the attack strategies to always flip k instances, leading to a fair comparison. We also experimented with a simple baseline based on random label flips [28], [29], [30], but we chose to omit it due to its poor performance for the budget we consider, especially in comparison to our proposed attacks.

We measure the effectiveness of different attack strategies in a white-box setting. We first use grid search to find the best model trained on the clean data. After the attack, we train a new model over the poisoned dataset using the same hyperparameters to estimate the accuracy loss. This setting represents a pessimistic scenario where the attacker has perfect knowledge of the training data and hyperparameters of the target model. We compare the effectiveness of the poisoning attacks in terms of F1 score and accuracy loss. We also consider the F1 score since the distribution of the class labels of the considered datasets is unbalanced. Even though Timber optimizes the loss of accuracy of the model, it also affects the F1 score by inducing misclassifications.

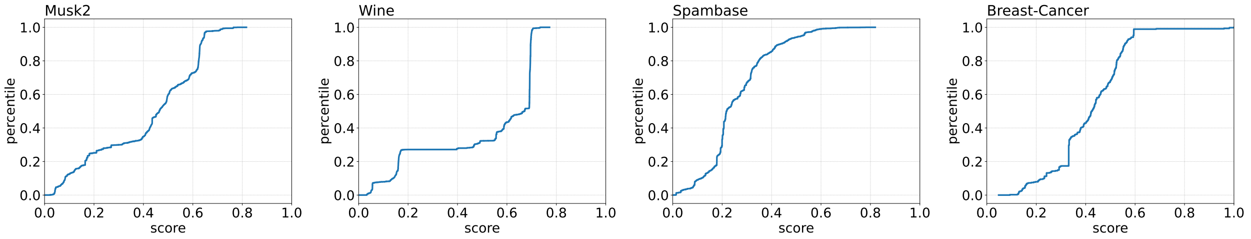
We focus on attacking decision tree ensembles, in particular RFs (without bootstrap sampling), since they are normally employed for tabular data classification in place of individual decision trees. The attacker tunes the number of trees from 2 to 15 and the maximum depth of the trees from 2 to 25. More details on the best RF model for each dataset are provided in the appendix.

B. Attack Efficiency

We first assess the efficiency of the considered attacks by measuring the running time required to poison 10% of the training set. This serves as an upper bound for the time needed to poison smaller subsets. In our experiments, we set a timeout of ten hours for each attack. The experiments have been performed on a virtual machine with 98 GB of RAM and Ubuntu 20.04.6 LTS, running on a server with an Intel Xeon Gold 6348 2.60GHz. To reduce running times, we rely on parallel implementations of the different attack strategies, using 16 threads. Training instances to poison are allocated to the different threads in a round-robin fashion.

Table I presents the running times of Greedy, Timber, GES and TES across different datasets. The black-box attacks K-Medoids and Entropy are much faster and always complete within a few seconds, so their runtimes are not comparable to white-box strategies and are not reported in the table to improve readability. Despite their efficiency, K-Medoids and Entropy are significantly less effective than the other attack methods, as reported in Section IV-C. Our experiments confirm that Timber and TES are faster than their counterparts Greedy and GES. Remarkably, the Greedy attack strategy turned out to be infeasible on the Musk2 dataset, exceeding our timeout of ten hours. The computed speedup on the Musk2 and Spambase datasets ranges from 2x to 6x, while it is smaller (less than 2x) on the Wine dataset. This demonstrates the significant advantage in efficiency enabled by tree annotation and sub-tree retraining, that enable performing an attack over the entire training set in a reasonable amount of time. We finally observe that Timber and TES require around the same time as Greedy and GES to complete the attack on Breast-Cancer.

We investigate the reasons behind the different speedups achieved by Timber and TES over Greedy and GES. We here focus on TES and we refer to the appendix for a similar discussion on Timber. Figure 4 shows the empirical cumulative distribution function of the mean scores of each instance of the training set, averaged over the k rounds of the TES attack. On Musk2 and



(a)

(b)

(c)

(d)

Fig. 4: Empirical cumulative distribution function of the mean scores of the training instances averaged over the rounds of TES on the considered datasets. The scores range from 0 to 1.

TABLE I: Runtime of the poisoning attacks with budget k equal to 10% of the training set size. Bold represents the best results in the two groups of columns.

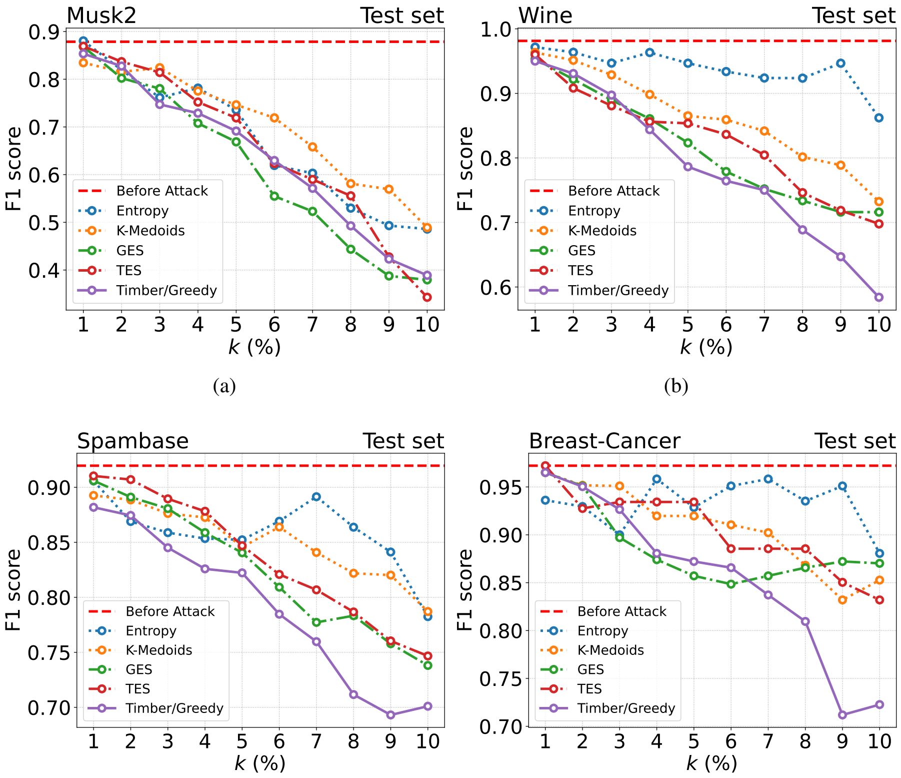
Dataset	Runtime			
	Greedy	**Timber**	**GES**	**TES**
Musk2	>10h	**1h41m**	3h8m	**29m17s**
Wine	3h21m	**2h59m**	1h24m	**1h15m**
Spambase	6h55m	**3h7m**	1h12m	**14m39s**
Breast	**2m38s**	3m23s	55s	**37s**

Spambase, more than 60% of the mean scores are below 0.5, meaning that most of the training instances that can be attacked are located at the root of sub-trees with few training instances on average, i.e., less than 50% of the number of instances in the training set, leading to high efficiency gains. This is particularly evident on the Musk2 dataset, the considered dataset with more instances and features, where TES is six times faster than GES. In contrast, the fact that less than 30% of the instances have a mean score smaller than 0.5 on Wine motivates the higher runtime of TES on the dataset, where the overhead induced by the annotate and sub-tree retraining is less effectively compensated. Finally, although the majority of the mean scores in Breast-Cancer fall between 0.3 and 0.6, the dataset is too small to observe a considerable speedup. This is reasonable, since Timber and TES are designed to enable greedy poisoning attacks on large datasets. When the dataset is small and Greedy terminates in a few minutes, the optimizations introduced by our attacks may be unneeded.

C. *Attack Effectiveness*

We now assess the effectiveness of the poisoning attacks in terms of F1 score loss on the test set. Figure 5 shows the F1 score loss induced by the considered attacks for each dataset and different values of k (we just report a single line for Timber and Greedy, because they always produce the same output). We observe that the black-box attacks Entropy and K-Medoids are consistently outperformed by the other attacks. For example, on the Musk2 dataset, the initial F1 score is 0.88 and TES reduces it to 0.34, while the most effective black-box attack Entropy reduces it just to 0.49 (+0.15 over TES). Additionally, Timber/Greedy, which always iterates over all the training instances, performs better than the early-stopping attacks TES and GES on Wine, Spambase, and Breast-Cancer. This is expected since early-stopping attacks explore only a subset of attack options during each round. For instance, Timber/Greedy reduces the F1 score on Wine to 0.58, while TES is less effective, reducing it to 0.70 (+0.12). It may occasionally happen that attacks relying on early stopping are more effective in reducing the F1 score than Timber/Greedy, because all the considered attacks are greedy. This happens for TES on the Musk2 dataset, which reduces the F1 score to 0.34, while Timber reduces the F1 score of the model to 0.39 (+0.05). Finally, note that TES is more effective than GES on three datasets out of four, and it is only one point less effective than GES on Spambase. Its enhanced efficacy is likely due to sorting the instances exploited by the attack to improve efficiency. Attacking instances with lower scores, i.e., retraining sub-trees in which few training instances fall, allows the attack to perform more local changes, inducing consistent losses in the performance of the target model. Ultimately, Timber and TES are the most effective attacks, with Timber generally being more effective than TES in reducing the F1 score but at a higher computational cost.

We can observe the same trends in the effectiveness of the poisoning attacks when considering the accuracy loss on the test set instead. For space reasons, we report the accuracy loss for each dataset and different values of k in the appendix.



(a)

(b)

(c)

(d)

Fig. 5: F1 score of the attacked model under different poisoning attacks for budget k equal to different percentages of poisoned training data, from 1% to 10%. A red horizontal line represent the F1 score of the model trained on the clean training set. Note that Timber is guaranteed to produce the F1 score loss as the Greedy attack strategy.

D. *Defenses*

Our previous evaluation showed that Timber and TES attacks perform best, as Timber is usually the most effective and TES is usually the most efficient. We now show that both Timber and TES are effective attacks even when applying a defense against poisoning before training. We recall that poisoning attacks against decision trees have been under-explored and the same applies to defenses. We then focus on two model-agnostic defenses:

1) *kNN-based defense* [5], a popular defense based on k-Nearest Neighbours (kNN) that performs *training data sanitization*. For each instance of the possibly poisoned training set, it computes its N nearest neighbors. If the fraction of the neighbors having the same label is greater than a threshold η, the algorithm assigns to the instance the label of these neighbors. The algorithm can be iterated for M times on the training set.
2) *Bagging-based defense* [31], a recent defense based on bagging that performs *robust training*. It uses the defended classifier as a base classifier. It employs a variant of bagging based on hash functions to generate G subsets of the possibly poisoned training set, each containing $K\%$ instances of the training set. Then, it trains G base classifiers on these subsets. The prediction for a test instances is obtained by aggregating the predictions of the base classifiers using hard majority voting.

In our evaluation, we perform grid search over the hyperparameters to select the values providing the highest F1 score on the validation set. This way, we estimate the effectiveness of the defense in the best possible setting from the defender's perspective. We consider $N \in \{4, 8, 12\}$, $\eta \in \{0.6, 0.75, 0.9\}$, $M \in \{1, 3, 5\}$, $G \in \{5, 10, \dots, 35, 40\}$ and $K \in \{20\%, 10\%, 5\%, 2.5\%\}$, including also the values used in the work presenting the defenses [5], [31].

TABLE II: F1 score on the test set of the best model trained on different training sets: F_1^c of the model trained on the clean training set, F_1^p of the model trained on the poisoned training set, F_1^n of the model trained on the training set sanitized by the kNN-based defense and F_1^b of the model trained by the bagging-based defense on the poisoned training set.

Dataset	F_1^c	**Timber**			**TES**		
		F_1^p	F_1^n	F_1^b	F_1^p	F_1^n	F_1^b
Musk2	0.88	0.39	0.55	0.50	0.34	0.56	0.48
Wine	0.98	0.58	0.81	0.80	0.70	0.83	0.84
Spambase	0.92	0.70	0.78	0.86	0.75	0.78	0.81
Breast	0.97	0.72	0.94	0.97	0.83	0.93	0.95

To understand the effectiveness of the evaluated defenses, we compute four measures over the test set: the F1 score of the original model trained on the clean training set (denoted by F_1^c), the F1 score of the model trained over the poisoned dataset (denoted by F_1^p) and the F1 score of the model trained on the poisoned dataset after applying the defense (denoted by F_1^d, with $d \in \{n, b\}$ discriminating between the kNN-based defense and the bagging-based defense). This allows us to compute for each defense d the *estimated defense benefit* $F_1^d - F_1^p$, i.e., the increase in F1 score enabled by the application of the defense w.r.t. the undefended poisoned model, and the *estimated residual damage* $F_1^c - F_1^d$, i.e., the decrease in F1 score w.r.t. the original model

The computed results are reported in Table II. The numbers show that the analyzed defenses provide some mitigation against our attacks. In particular, the estimated defense benefit $F_1^n - F_1^p$ for the kNN-based defense ranges between 0.03 and 0.23, with an average value of 0.15, while the estimated defense benefit $F_1^b - F_1^p$ for the bagging-based defense ranges between 0.06 and 0.25, with an average value of 0.15. Nevertheless, the damage caused by our poisoning attacks despite the application of the defenses is significant. The estimated residual damage $F_1^c - F_1^n$ for the kNN-based defense ranges between 0.03 and 0.33, with an average value of 0.17, while the estimated residual damage $F_1^c - F_1^b$ for the bagging-based defense ranges between 0 and 0.40, with an average of 0.16. This implies that, on average, our attack reduces the F_1 score of the original model by between 0.16 and 0.17, even when one of the two defenses is applied. Thus, the analyzed defenses effectively mitigate the damage of our poisoning attacks, but they are far from being able to completely thwart them. The only dataset where the application of the defense yields a model with comparable performance to the original one is Breast-Cancer. This can be explained by the simplicity of this dataset, where a decision stump (i.e., a decision tree of depth 1) achieves an F_1 score of 0.95. This suggests that the classes are easily distinguishable, enhancing the effectiveness of the defenses. These observations are confirmed by looking at the effect of the defenses on the accuracy, that we report in the appendix.

V. Related Work

We here discuss poisoning attacks and defenses using the taxonomy provided in [1].

A. Poisoning Attacks

Availability poisoning attacks aim to degrade the accuracy of the target classifier to compromise its utility. Existing label flip attacks and our new attacks, Timber and TES, belong to this category, which assumes that the attacker can only modify the labels of the instances in the training set. The objective is to find the combination of flips that leads to the best accuracy loss of the target model. Label flip attacks have been deeply investigated for support vector machines [2], [3], [4], [5], linear regression models [6] and neural networks [7], [8]. To the best of our knowledge, no work in the literature has proposed poisoning attacks specifically for decision tree ensembles. Previous work [32], [28], [30], [13], [33], [29] evaluates the robustness of decision tree ensembles against the random label flip attack, a model-agnostic attack that selects the label to flip randomly. [10] proposes other two model-agnostic attacks, *Entropy* and *K-Medoids*, that are also evaluated on decision tree ensembles. Our work fills an important gap in the literature by proposing the first poisoning attack specifically tailored for decision tree ensembles, which is feasible on large datasets and clearly outperforms other attack approaches.

Other availability poisoning attacks are clean-label, i.e., they assume that the attacker can modify only features [34], [35], [36], [12], [11], and hybrid, in the sense that they target both features and labels [37], [38], [39]. Bilevel poisoning attacks are the most popular attacks of these two categories. They find the best perturbation to apply to the training data by solving a bilevel optimization problem [34]. Most of the attacks of this type target differentiable models since they exploit gradients extracted from the loss function of the target model to find the best perturbations to apply to the training instances. However, decision trees are non-differentiable models. Thus, previous work evaluated only model-agnostic clean-label and hybrid attacks [40], [12], [11], [28] on decision tree ensembles. Designing clean-label and hybrid poisoning attacks for decision tree ensembles is a relevant direction for future work.

Finally, another category of poisoning attacks aims at harming the *integrity* of ML models [36], [41], [42], [43], [44], [45], [46], [47], [48]. The objective is to preserve the general performance of the target model, while causing the misclassification of specific samples. The aim of these attacks is different from ours, since our two proposed attacks target the availability of decision tree ensembles, so we do not compare against this category. However, even integrity attacks against decision tree ensembles have not been deeply investigated in the literature. Designing efficient and effective integrity-poisoning attacks against decision tree ensembles is another interesting line of research for future work.

B. Defenses Against Poisoning

We focus on defenses against availability poisoning attacks, in particular label flip attacks, grouped into two classes: *training data sanitization* and *robust training* [1].

Training data sanitization defenses are model-agnostic approaches that remove poisoning samples from the training set before training by recognizing instances that are different from the other legitimate training points. Previously proposed techniques exploit k-Nearest Neighbours classifiers [5], clustering algorithms [49] and outlier detection algorithms [50], [51].

Robust training defenses aim instead at mitigating the effect of poisoning during training. In particular, the defenses consist of training algorithms that mitigate the effect of poisoned samples. Some model-agnostic defenses of this type exploit bagging, leveraging the observation that using small subsets of the training set for training ensembles can mitigate the effect of poisoning [2], [52], [53], [54], [31], [28]. These defenses and the Randomized Smoothing-based defense [55] can also provide certificates about the robustness of the model to availability poisoning attacks. Another defense proposed in [56] removes instances from the training set if they induce a significant loss in accuracy when used in training. Finally, specific defenses for differentiable models have been proposed and exploit robust optimization [57], [58], [59], regularization [27], [60] and loss correction [61].

To the best of our knowledge, no robust training defenses have been specifically designed to protect decision tree ensembles, even though all the model-agnostic defenses previously described can be applied. Algorithms that verify the robustness of decision trees against poisoning attacks have been proposed instead [62], [63]. [28] is the only work specifically evaluating the application of a defense to decision tree ensembles, in particular RFs, against availability poisoning attacks like random label flip. It employs the hash bagging defense inspired by previous work [53], [54], [31], and it observes a degradation of the performance of the RFs even when adopting the defense. However, it considers unrealistic attack budgets ranging from 10% to 30% of training instances. Our work is orthogonal to this work since we do not focus on defenses, but we propose new attacks specifically tailored for decision tree ensembles. We demonstrate that our attacks Timber and TES are still effective when one representative defense [5], [31] from each group is adopted, even when we consider more realistic attacker's capabilities, i.e., the attacker can flip at most 10% of the labels of the training set. The two defenses partially mitigate the effect of the attacks, but they are not able to thwart them. We leave the evaluation of other defenses as future work, as well as designing defenses specifically tailored to enhance the robustness of decision tree ensembles to poisoning attacks.

VI. Conclusion

We presented Timber, the first white-box poisoning attack for decision trees. Timber uses a greedy strategy, incorporating an annotation procedure for the tree and sub-tree retraining to efficiently assess the impact of poisoned instances, optimizing the computational cost of the attack. This allows Timber to scale to larger datasets than standard greedy methods. The Timber variant with early stopping offers faster runtimes, though with potentially reduced effectiveness. We also extended Timber to decision tree ensembles, particularly random forests, to demonstrate its relevance in real-world machine learning applications. Our experiments on public datasets show that Timber and its variant with early stopping outperform existing black-box strategies in terms of attack effectiveness and existing greedy attacks in terms of attack efficiency. Moreover, our two attacks are not thwarted by two representative defenses.

As future work, we would like to generalize our techniques to GBDT models and design more powerful defenses for poisoning attacks specific to decision tree ensembles. We also plan to study how to efficiently perform clean-label and integrity poisoning attacks against decision tree ensembles, to fill the gap in the literature.

Acknowledgements: This research was supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. Moreover, it acknowledges support from the European Union - Next-GenerationEU - PNRR – M.4 C.2, I.1.1 - PRIN 2022 WHAM!, 2022ZZX57L, H53D23003750006.

References

[1] A. E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B. A. Moser, A. Oprea, B. Biggio, M. Pelillo, and F. Roli, "Wild patterns reloaded: A survey of machine learning security against training data poisoning," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 294:1–294:39, 2023. [Online]. Available: https://doi.org/10.1145/3585385

[2] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *Multiple Classifier Systems - 10th International Workshop, MCS 2011, Naples, Italy, June 15-17, 2011. Proceedings*, ser. Lecture Notes in Computer Science, C. Sansone, J. Kittler, and F. Roli, Eds., vol. 6713. Springer, 2011, pp. 350–359. [Online]. Available: https://doi.org/10.1007/978-3-642-21557-5_37

[3] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, ser. Frontiers in Artificial Intelligence and Applications, L. D. Raedt, C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, Eds., vol. 242. IOS Press, 2012, pp. 870–875. [Online]. Available: https://doi.org/10.3233/978-1-61499-098-7-870

[4] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53–62, 2015. [Online]. Available: https://doi.org/10.1016/j.neucom.2014.08.081

[5] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label sanitization against label flipping poisoning attacks," in *ECML PKDD 2018 Workshops - Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings*, ser. Lecture Notes in Computer Science, C. Alzate, A. Monreale, H. Assem, A. Bifet, T. S. Buda, B. Caglayan, B. Drury, E. García-Martín, R. Gavaldà, S. Kramer, N. Lavesson, M. Madden, I. M. Molloy, M. Nicolae, and M. Sinn, Eds., vol. 11329. Springer, 2018, pp. 5–15. [Online]. Available: https://doi.org/10.1007/978-3-030-13453-2_1

[6] P. Awasthi, M. Balcan, and P. M. Long, "The power of localization for efficiently learning linear separators with noise," *J. ACM*, vol. 63, no. 6, pp. 50:1–50:27, 2017. [Online]. Available: https://doi.org/10.1145/3006384

[7] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=Sy8gdB9xx

[8] M. Zhang, L. Hu, C. Shi, and X. Wang, "Adversarial label-flipping attack and defense for graph neural networks," in *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, C. Plant, H. Wang, A. Cuzzocrea, C. Zaniolo, and X. Wu, Eds. IEEE, 2020, pp. 791–800. [Online]. Available: https://doi.org/10.1109/ICDM50108.2020.00088

[9] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/0378c7692da36807bdec87ab043cdadc-Abstract-Datasets_and_Benchmarks.html

[10] H. Zhang, N. Cheng, Y. Zhang, and Z. Li, "Label flipping attacks against naive bayes on spam filtering systems," *Appl. Intell.*, vol. 51, no. 7, pp. 4503–4514, 2021. [Online]. Available: https://doi.org/10.1007/s10489-020-02086-4

[11] A. Prud'Homme and B. Kantarci, "Poisoning attack anticipation in mobile crowdsensing: A competitive learning-based study," in *WiseMLWiSec 2021: Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning, Abu Dhabi, United Arab Emirates, July 2, 2021*, C. Pöpper and M. Vanhoef, Eds. ACM, 2021, pp. 73–78. [Online]. Available: https://doi.org/10.1145/3468218.3469050

[12] K. Talty, J. Stockdale, and N. D. Bastian, "A sensitivity analysis of poisoning and evasion attacks in network intrusion detection system machine learning models," in *2021 IEEE Military Communications Conference, MILCOM 2021, San Diego, CA, USA, November 29 - Dec. 2, 2021*. IEEE, 2021, pp. 1011–1016. [Online]. Available: https://doi.org/10.1109/MILCOM52596.2021.9652959

[13] A. R. Shahid, A. Imteaj, P. Y. Wu, D. A. Igoche, and T. Alam, "Label flipping data poisoning attack against wearable human activity recognition system," in *IEEE Symposium Series on Computational Intelligence, SSCI 2022, Singapore, December 4-7, 2022*, H. Ishibuchi, C. Kwoh, A. Tan, D. Srinivasan, C. Miao, A. Trivedi, and K. A. Crockett, Eds. IEEE, 2022, pp. 908–914. [Online]. Available: https://doi.org/10.1109/SSCI51031.2022.10022015

[14] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017. [Online]. Available: http://arxiv.org/abs/1712.05526

[15] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You autocomplete me: Poisoning vulnerabilities in neural code completion," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. D. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 1559–1575. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/schuster

[16] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[17] "Timber GitHub Repository," https://github.com/massimo-vettori/timber, accessed: 2025-03-02.

[18] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[19] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[20] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451

[21] J. Y. Chang and E. G. Im, "Data Poisoning Attack on Random Forest Classification Model," 2020.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] "Musk2 Dataset," https://archive.ics.uci.edu/dataset/75/musk+version+2, accessed: 2024-09-24.

[24] "Wine Dataset," https://archive.ics.uci.edu/dataset/186/wine+quality, accessed: 2024-09-24.

[25] "Spambase Dataset," https://archive.ics.uci.edu/dataset/94/spambase, accessed: 2024-09-24.

[26] "Breast cancer Dataset," http://archive.ics.uci.edu/dataset/14/breast+cancer, accessed: 2024-09-24.

[27] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proceedings of the 3rd Asian Conference on Machine Learning, ACML 2011, Taoyuan, Taiwan, November 13-15, 2011*, ser. JMLR Proceedings, C. Hsu and W. S. Lee, Eds., vol. 20. JMLR.org, 2011, pp. 97–112. [Online]. Available: http://proceedings.mlr.press/v20/biggio11/biggio11.pdf

[28] M. Anisetti, C. A. Ardagna, A. Balestrucci, N. Bena, E. Damiani, and C. Y. Yeun, "On the robustness of random forest against untargeted data poisoning: An ensemble-based approach," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 4, pp. 540–554, 2023. [Online]. Available: https://doi.org/10.1109/TSUSC.2023.3293269

[29] F. A. Yerlikaya and S. Bahtiyar, "Data poisoning attacks against machine learning algorithms," *Expert Syst. Appl.*, vol. 208, p. 118101, 2022. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.118101

[30] C. Dunn, N. Moustafa, and B. Turnbull, "Robustness evaluations of sustainable machine learning models against data poisoning attacks in the internet of things," *Sustainability*, vol. 12, no. 16, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/16/6434

[31] R. Chen, Z. Li, J. Li, J. Yan, and C. Wu, "On collective robustness of bagging against data poisoning," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 3299–3319. [Online]. Available: https://proceedings.mlr.press/v162/chen22k.html

[32] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010. [Online]. Available: https://doi.org/10.1007/s10994-010-5188-5

[33] K. Aryal, M. Gupta, and M. Abdelsalam, "Analysis of label-flip poisoning attack on machine learning based malware detector," in *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*, S. Tsumoto, Y. Ohsawa, L. Chen, D. V. den Poel, X. Hu, Y. Motomura, T. Takagi, L. Wu, Y. Xie, A. Abe, and V. Raghavan, Eds. IEEE, 2022, pp. 4236–4245. [Online]. Available: https://doi.org/10.1109/BigData55660.2022.10020528

[34] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. [Online]. Available: http://icml.cc/2012/papers/880.pdf

[35] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 1689–1698. [Online]. Available: http://proceedings.mlr.press/v37/xiao15.html

[36] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, B. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, and A. Sinha, Eds. ACM, 2017, pp. 27–38. [Online]. Available: https://doi.org/10.1145/3128572.3140451

[37] J. Feng, Q. Cai, and Z. Zhou, "Learning to confuse: Generating training time adversarial data with auto-encoder," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 11 971–11 981. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/1ce83e5d4135b07c0b82afffbe2b3436-Abstract.html

[38] L. Fowl, P. Chiang, M. Goldblum, J. Geiping, A. Bansal, W. Czaja, and T. Goldstein, "Preventing unauthorized use of proprietary data: Poisoning for secure dataset release," *CoRR*, vol. abs/2103.02683, 2021. [Online]. Available: https://arxiv.org/abs/2103.02683

[39] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, B. Bonet and S. Koenig, Eds. AAAI Press, 2015, pp. 2871–2877. [Online]. Available: https://doi.org/10.1609/aaai.v29i1.9569

[40] L. Verde, F. Marulli, and S. Marrone, "Exploring the impact of data poisoning attacks on machine learning model reliability," in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES-2021, Virtual Event / Szczecin, Poland, 8-10 September 2021*, ser. Procedia Computer Science, J. Watróbski, W. Salabun, C. Toro, C. Zanni-Merk, R. J. Howlett, and L. C. Jain, Eds., vol. 192. Elsevier, 2021, pp. 2624–2632. [Online]. Available: https://doi.org/10.1016/j.procs.2021.09.032

[41] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 1885–1894. [Online]. Available: http://proceedings.mlr.press/v70/koh17a.html

[42] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 6106–6116. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/22722a343513ed45f14905eb07621686-Abstract.html

[43] J. Guo and C. Liu, "Practical poisoning attacks on neural networks," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXVII*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12372. Springer, 2020, pp. 142–158. [Online]. Available: https://doi.org/10.1007/978-3-030-58583-9_9

[44] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein, "Metapoison: Practical general-purpose clean-label data poisoning," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/8ce6fc704072e351679ac97d4a985574-Abstract.html

[45] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2909068

[46] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25th Annual Network and Distributed System Security*

Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018. The Internet Society, 2018. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_03A-5_Liu_paper.pdf

[47] T. A. Nguyen and A. T. Tran, "Wanet - imperceptible warping-based backdoor attack," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=eEn8KTtJOx

[48] E. Sarkar, H. Benkraouda, G. Krishnan, H. Gamil, and M. Maniatakos, "Facehack: Attacking facial recognition systems using malicious facial characteristics," *IEEE Trans. Biom. Behav. Identity Sci.*, vol. 4, no. 3, pp. 361–372, 2022. [Online]. Available: https://doi.org/10.1109/TBIOM.2021.3132132

[49] R. Laishram and V. V. Phoha, "Curie: A method for protecting SVM classifier from poisoning attack," *CoRR*, vol. abs/1606.01584, 2016. [Online]. Available: http://arxiv.org/abs/1606.01584

[50] C. Frederickson, M. Moore, G. Dawson, and R. Polikar, "Attack strength vs. detectability dilemma in adversarial machine learning," in *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. IEEE, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN.2018.8489495

[51] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 3517–3529. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/9d7311ba459f9e45ed746755a32dcd11-Abstract.html

[52] J. Jia, X. Cao, and N. Z. Gong, "Intrinsic certified robustness of bagging against data poisoning attacks," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 7961–7969. [Online]. Available: https://doi.org/10.1609/aaai.v35i9.16971

[53] A. Levine and S. Feizi, "Deep partition aggregation: Provable defenses against general poisoning attacks," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=YUGG2tFuPM

[54] W. Wang, A. Levine, and S. Feizi, "Improved certified defenses against data poisoning with (deterministic) finite aggregation," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 22 769–22 783. [Online]. Available: https://proceedings.mlr.press/v162/wang22m.html

[55] E. Rosenfeld, E. Winston, P. Ravikumar, and J. Z. Kolter, "Certified robustness to label-flipping attacks via randomized smoothing," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 8230–8241. [Online]. Available: http://proceedings.mlr.press/v119/rosenfeld20b.html

[56] B. Nelson, M. Barreno, F. Jack Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, *Misleading Learners: Co-opting Your Spam Filter*. Boston, MA: Springer US, 2009, pp. 17–51. [Online]. Available: https://doi.org/10.1007/978-0-387-88735-7_2

[57] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A robust meta-algorithm for stochastic optimization," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 1596–1606. [Online]. Available: http://proceedings.mlr.press/v97/diakonikolas19a.html

[58] C. Liu, B. Li, Y. Vorobeychik, and A. Oprea, "Robust linear regression against training data poisoning," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, B. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, and A. Sinha, Eds. ACM, 2017, pp. 91–102. [Online]. Available: https://doi.org/10.1145/3128572.3140447

[59] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 19–35. [Online]. Available: https://doi.org/10.1109/SP.2018.00057

[60] A. Demontis, B. Biggio, G. Fumera, G. Giacinto, and F. Roli, "Infinity-norm support vector machines against adversarial label contamination," in *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017*, ser. CEUR Workshop Proceedings, A. Armando, R. Baldoni, and R. Focardi, Eds., vol. 1816. CEUR-WS.org, 2017, pp. 106–115. [Online]. Available: https://ceur-ws.org/Vol-1816/paper-11.pdf

[61] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 2233–2241. [Online]. Available: https://doi.org/10.1109/CVPR.2017.240

[62] S. Drews, A. Albarghouthi, and L. D'Antoni, "Proving data-poisoning robustness in decision trees," in *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, A. F. Donaldson and E. Torlak, Eds. ACM, 2020, pp. 1083–1097. [Online]. Available: https://doi.org/10.1145/3385412.3385975

[63] A. P. Meyer, A. Albarghouthi, and L. D'Antoni, "Certifying robustness to programmable data bias in decision trees," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 26 276–26 288. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/dcf531edc9b229acfe0f4b87e1e278dd-Abstract.html

Appendix

In this section, we present figures and tables that include information and experimental results omitted from the main body of the paper.

We first show in Table III the key characteristics of the datasets and in Table IV the details of the best RF model for each dataset.

We then show in Figure 6 the empirical cumulative distribution function of the mean scores of each instance in the training set, averaged over k rounds of the Timber attack, where k is set to 10% of the training set size. The distributions of the mean scores across the datasets are similar to those observed for the TES attack. Therefore, the reasoning behind the speed-up of Timber over Greedy follows the same rationale as the speed-up of TES over GES (see Section IV-B).

Figure 7 instead shows the accuracy loss on the best model for each dataset, induced by the pool of considered attacks across different values of k. The trends in the results are similar to those observed for the F1 score loss, confirming the insights into the effectiveness of the attacks derived from the discussion on the F1 score loss induced by each attack (see Section IV-C).

Finally, we show in Table V the effectiveness of the evaluated defenses, using accuracy as the performance metric. In particular, we compute four measures over the test set: the accuracy of the original model trained on the clean training set (denoted by a^c), the accuracy of the model trained over the poisoned dataset created by Timber or TES (denoted by a^p) and the accuracy of the model trained on the poisoned dataset after applying the defense (denoted by a^d, with $d \in \{n, b\}$ discriminating between the kNN-based defense and the bagging-based defense). The results align with those derived from the F1 score evaluations (see Section IV-D), showing that the defenses can mitigate the impact of both attacks but cannot actually thwart them.

TABLE III: Dataset statistics.

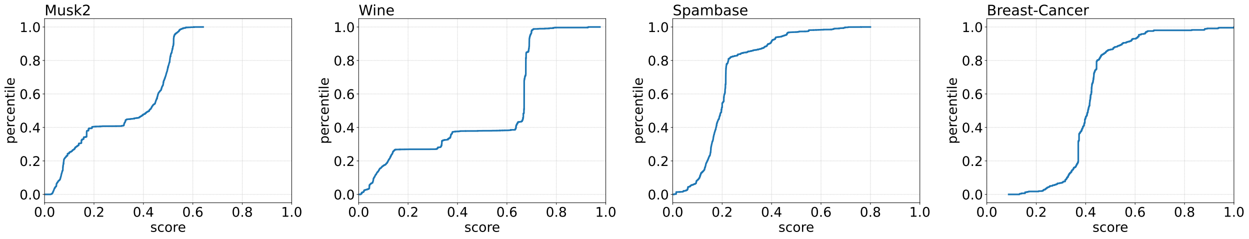
Dataset	**Instances**	**Features**	**Distribution**
Musk2	6,598	166	85%/15%
Wine	6,497	11	75%/25%
Spambase	4,601	57	61%/39%
Breast	569	30	63%/37%

TABLE IV: Number of trees, maximum depth, accuracy and F1 score on the test set of the RF.

Dataset	**# Trees**	**Max. Depth**	**Accuracy**	**F1 score**
Musk2	7	20	0.96	0.88
Wine	14	9	0.99	0.98
Spambase	15	20	0.94	0.92
Breast	14	7	0.97	0.97

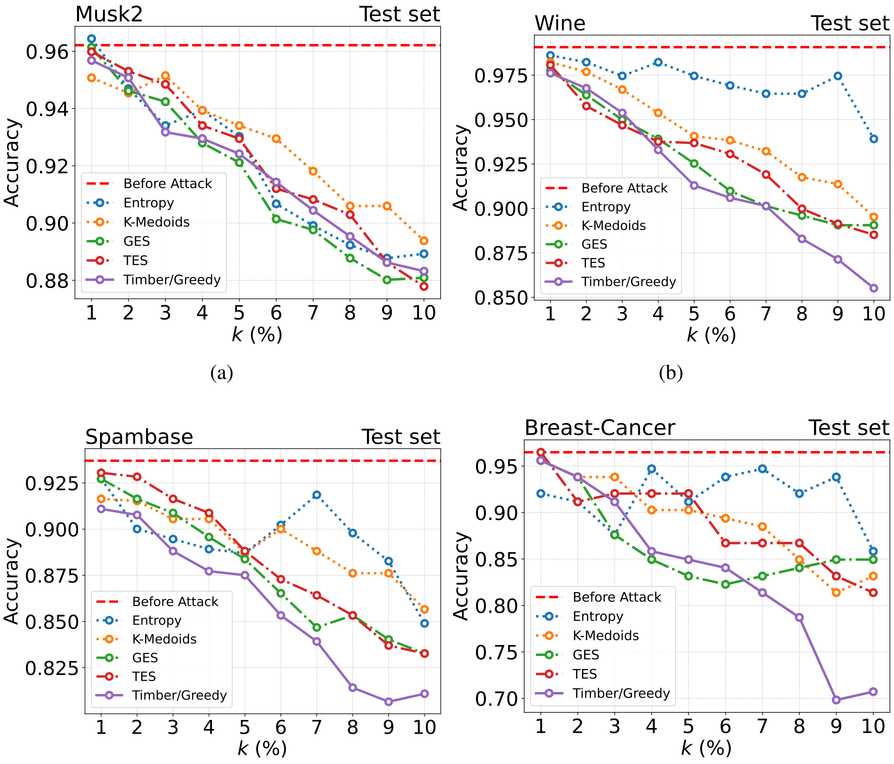
TABLE V: Accuracy on the test set of the best model trained on different training sets: a^c of the model trained on the clean training set, a^p of the model trained on the poisoned training set, a^n of the model trained on the training set sanitized by the kNN-based defense and a^b of the model trained by the bagging-based defense on the poisoned training set.

Dataset	a^c	**Timber**			**TES**		
		a^p	a^n	a^b	a^p	a^n	a^b
Musk2	0.96	0.88	0.90	0.89	0.88	0.90	0.88
Wine	0.99	0.86	0.92	0.91	0.89	0.93	0.93
Spambase	0.94	0.81	0.85	0.90	0.83	0.85	0.87
Breast	0.96	0.71	0.92	0.96	0.81	0.91	0.96



(a) (b) (c) (d)

Fig. 6: Empirical cumulative distribution function of the mean scores of the training instances over the iterations of Timber on the considered datasets. The scores range from 0 to 1.



(a) (b)

(c) (d)

Fig. 7: Accuracy of the attacked model under different poisoning attacks for budget k equal to different percentages of poisoned training data, from 1% to 10%. A red horizontal line represent the accuracy of the model trained on the clean training set. Note that Timber is guaranteed to produce the same accuracy loss as the Greedy attack strategy.