CA' FOSCARI UNIVERSITY OF VENICE
DEPARTMENT OF ENVIRONMENTAL SCIENCES, INFORMATICS AND STATISTICS
PHD IN COMPUTER SCIENCE, XXXIV CYCLE

MITIGATING UNFAIRNESS AND ADVERSARIAL ATTACKS IN MACHINE LEARNING

Doctoral Dissertation of:
Seyum Assefa Abebe

Supervisor:
Prof. Claudio Lucchese

Co-supervisor:
Prof. Salvatore Orlando

Università
Ca'Foscari
Venezia

Corso di Dottorato di ricerca
in Informatica

ciclo XXXIV

Tesi di Ricerca

Mitigating Unfairness and Adversarial Attacks in Machine Learning

SSD:INF/01

Coordinatore del Dottorato:
ch. prof. Agostino Cortesi

Supervisore:
ch.Prof. Claudio Lucchese

Co-Supervisore:
ch.Prof. Salvatore Orlando

Dottorando:
Seyum Assefa Abebe
Matricola: 854877

To Bella,
My love, my life, my light.
This thesis is affectionately dedicated.

Abstract

Machine learning (ML) advanced and increased processing and storage capacity have changed the technological environment. Algorithms such as decision trees and decision tree ensembles are widely used and achieve state-of-the-art performance in many application areas, ranging from financial systems to computer security. Despite the high level of performance frequently achieved by these systems, there is widespread agreement that their dependability should be carefully evaluated, especially when used in vital applications, such as medical, criminal justice, financial markets, system security, or self-driving cars. The risk of these algorithms unwittingly making incorrect conclusions while misdirected by artefacts or misleading correlations, during training or after training of a model, is not negligible, especially in the era of big data and extensive use of machine learning models. Accordingly, to increase users' trust and identify potential vulnerabilities of machine learning approaches, a research area of *adversarial machine learning* has been started to explore vulnerabilities of algorithms in adversarial settings and develop techniques to make models robust to attacks. Thus, many works were introduced by considering adversarial environments in machine learning, by studying potential weaknesses, evaluating models' performance under adversarial attacks, and designing defensive mechanisms against attacks.

In parallel, addressing the trustworthiness of systems in decision making, many scientists also explored the research field of *fair machine learning*, with the goal of designing systems that use machine learning that is not only able to perform tasks accurately but also bias-free towards a particular

group or individual which might originally come from biased datasets or algorithm design.

In this thesis, after giving a thorough background on adversarial and fair machine learning and motivating the simultaneous desirability of robustness and fairness properties along with notable works, we first present a literature review of adversarial learning, focused mainly on decision tree and tree ensembles by putting together attacks, defences, robustness verification, and performance evaluations to better understand the existing works and techniques. Additionally, we summarized unfairness mitigation mechanisms designed for tree-based models and grouped recent research works of literature into pre-processing, in-processing, and post-processing strategies.

In this work, we propose a novel decision tree learning algorithm, called TREANT, that on the basis of a formal threat model, minimizes an evasion-aware loss function at each step of the tree construction based on two technical concepts: *robust splitting and attack invariance*, which jointly guarantee the soundness of the learning process. The algorithm employs a formal threat model to generate attacks with a more flexible approach to specifying attacker capabilities. As a result, it can generate decision tree ensembles that are simultaneously accurate and nearly insensitive to evasion attacks, outperforming state-of-the-art adversarial learning techniques.

Finally, we present EiFFFeL: Enforcing Fairness in Forests by Flipping Leaves, a novel post-processing algorithm that, given a forest of decision trees for a binary classification task, modifies the prediction of a carefully chosen set of leaves to reduce the forest's discrimination degree.

This research calls for future efforts to investigate the convergence of the robustness and fairness of models, which is critical in achieving trustworthy and responsible ML. This also allows exploring and connecting with other areas like interpretability in ML, which seeks to explain the model prediction to a human.

Acknowledgments

IN THE NAME OF GOD, THE MOST GRACIOUS, THE MOST MERCIFUL

My at most gratitude to Professor Claudio Lucchese and Professor Salvatore Orlando, especially to Professor Lucchese for his patience, scientific guidance, encouragement and fruitful discussion during my Ph.D. program. I am so grateful to you both.

I thank my external reviewers: Prof. Giuseppe Manco, and Prof. Panagiotis Papapetrou for the time they spent on carefully reading the thesis and for their insightful comments and suggestions.

My friends Tinsu, Teddy,Leueel, Surafel, Josh, and Yoni. Please accept my sincere gratitude for being good friends, and for the unforgettable moments in Venice.

To my wife *Bella* i can't repay you back for everything you have done for me. Thank you for your forgiveness, patience, perseverance, seemingly inexhaustible well of support, and love. I owe you everything.

My boy, Adeel, you are my beacon of light in the uncharted hazy quest of mine.

Extremely grateful to my sister *Rawdi* and my brother in law Mohammed. My brothers Alex and Abdulkerim, my little sister Zubeyda thank you for being there for me.

Finally, to my Mom and Dad i thank you from the bottom of my heart for your unconditional love and for everything that you did in my life. Your prayer kept me safe from all the troubles; and the love i have for you is beyond expression.

Preface

This dissertation is submitted in partial fulfillment of the requirement for the degree of doctor of philosophy at the Ca'Foscari University of Venice. The thesis presents novel works in adversarial and fair machine learning. The first chapter begins with a brief introduction; the second chapter provides background on machine learning, concepts of adversarial learning, and fair machine learning. The third chapter presents a survey of adversarial learning targeting tree-based models, submitted to ACM Computing Surveys(CSUR) journal and is being reviewed. The Fourth chapter summarizes recent research advances on fair machine learning implemented using tree-based models. Chapter five discusses an evasion-aware decision tree learning algorithm published in Data Mining and Knowledge Discovery (DAMI). Finally, chapter six presents a work that ensures group fairness in forests by flipping leaves, which is published at ACM Symposium on Applied Computing (ACM SAC).

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Machine learning (ML) is becoming a prominent and integral part of many applications and computing mainstreams. Deep neural networks(DNN) and decision tree ensemble models demonstrate superior performance on many real-world problems, for example, in speech recognition [74], facial recognition [134], autonomous cars [34], and spam filtering [108, 174]. In addition, learning algorithms are finding their way into health informatics, fraud detection, computer vision, machine translation, natural language understanding, and system security as a vital tool to data analysis and decision making. Most ML algorithms are designed to approximate an unknown mapping from an input domain to an output domain by observing sample pairs of inputs and outputs from these domains, which is a key distinguishing characteristic of ML. Rather than explicitly describing the solution to a problem through code, the programmer describes how to discover this solution through examples of solved problem instances or training data. The result of learning is thus a model representing the approximate solution learned through a principled exploration of this data.

Highly important applications such as search, financial trading, data analytics, autonomous vehicles, and malware detection are all critically dependent on the underlying ML algorithms that interpret their respective domain

inputs to provide intelligent outputs that facilitate users' decision-making process or automated systems [65]. ML has traditionally been developed with an assumption that the environment is benign and bias-free during both training and evaluation of the model. Specifically, it is assumed that the statistical properties of the data on which the model is deployed to make predictions are identical to the ones of the data it was trained on. This assumption has been useful for designing effective ML algorithms, but they implicitly rule out the possibility that an adversary could alter some of these statistical properties. As ML is used in more contexts where malicious adversaries have an incentive to interfere with the operation of the ML system, it is increasingly important to protect against such adversarial manipulations.

A growing number of scientific works shows that machine learning systems can be attacked in different application domains, such as spam filtering [108, 174] and intrusion detection [60, 157]. Hence, It is critical to ensure that machine learning systems are secured and successful despite these attacks. So, to assess vulnerabilities, understand attacks, and make machine learning methods effective, a research field called *adversarial machine learning* was born [17]; to model and investigate attacks on machine learning and develop learning techniques robust to adversarial manipulation [170]. Thus, addressing the aims of adversarial machine learning has become an open scientific challenge and huge research field.

Another challenge that comes along with the prominence of machine learning and its deployment in critical areas of social, economic, and political importance such as financial sector [26,113], criminal justice [56], child welfare [166], and medicine [51,95] is: its far reaching implications in real life. The fact that traditional machine learning algorithms aim to maximize predictive performance only with regard to the historical training data may force the resultant model to make unfair and undesired predictions, e.g., some individuals are unfavourably treated due to some sensitive information. Due to this, serious concerns are coming from the general public regarding the reliability and fairness of machine learning in modelling human behaviours and characteristics. To counter this problem, another research field called *fair machine learning* started to rise.

1.1 Thesis Statement

Both adversarial and fair machine learning research areas gained attention in classification and regression tasks and were well researched separately using neural networks and linear models in recent years. Hence, Providing

a formal definition for an attack and defence methodologies in an adversarial setting and defining fairness and its metrics relative to a group or individual. Applications that use perceptual data saw remarkable advances in both fields, especially using deep neural networks. Researchers propose algorithmic solutions once it is known to suffers from adversarial attack and bias in decision making. In contrast, tree-based models which achieve state-of-the-art performance in the domain of non-perceptual data have not been given much attention and recently discovered they are vulnerable to adversarial attacks and prone to generating bias. Their interpretability compared to other models [158], generating decisions that are easy to be understood by humans in terms of syntactic checks over domain features, which is particularly appealing in the security setting. Despite this success, there is limited attention to tree-based models in adversarial and fair machine learning research communities, resulting in a sub-optimal state of the art in adversarial and fair machine learning techniques.

In this thesis, inspired by the previously mentioned challenges and the appeal to have a robust and fair model, we explore adversarial and fair machine learning techniques, propose an evasion attack aware robust tree learning method, and also implement group fairness optimization which ensures non-discriminate decision in a post-process approach using decision tree and tree ensembles. We hope our work contributes to the formulation of more robust and fair learning algorithms in the future.

1.2 Outline

We start in chapter 2 by providing background and overview on machine learning and its categories. In this chapter, we emphasise and explain, particularly the decision tree and tree ensembles, since our contributions on both adversarial and fair machine learning are implemented using them. We also broadly describe the main concepts of adversarial machine learning and fairness in learning algorithms.

Chapter 3 explores state-of-the-art adversarial machine learning techniques targeting tree-based models. This chapter extensively studies the current adversarial attack, defence, and verification and robustness evaluation methodologies detailing recent contributions.

In Chapter 4, we summarize state-of-the-art unfairness mitigation strategies applied to decision trees and tree-based models. The chapter details recent contributions on works on pre-processing, in-processing, and post-processing mitigation strategies.

Then, chapter 5 presents a novel learning algorithm designed to build

decision trees that are resilient against evasion attacks at test time.

In chapter 6, we study discrimination and bias in learning algorithms, give a formal definition of discrimination, and present our post-process group fairness enforcing approach.

Finally, in chapters 7 and **??** discuss future research directives and conclusive remarks.

Related Publications

This thesis includes research work of the following publications:

- **Seyum Assefa Abebe**, Claudio Lucchese, and Salvatore Orlando. EiFFFeL: enforcing fairness in forests by flipping leaves. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, 429-436.
- **Seyum Assefa Abebe**, Claudio Lucchese, Salvatore Orlando, and Stefano Calzavara. Adversarial Machine Learning Targeting Tree-Based Models:A Survey[Under Review]
- Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, **Seyum Assefa Abebe**, and Salvatore Orlando. Treant: training evasion-aware decision trees. Data Mining and Knowledge Discovery, 34(5):1390-1420, 2020.

CHAPTER 2

Background

In this chapter, we introduce the basic foundation of machine learning that this thesis builds upon, including a description of the most used algorithms and techniques, particularly giving emphasis to decision tree and tree ensemble learning algorithms. We also discuss *adversarial machine learning* by giving a formal definition, illustrating the state of the art, and describing how an adversary can attack learning algorithms using different approaches. By explaining a threat model, we review attacks and defence techniques proposed. In addition, we explain the problem of bias in machine learning, the sources of unfairness, define fairness from a group and individual point of view, and list unfairness mitigation strategies.

2.1 Machine Learning

Machine learning is a broad area that includes techniques for extracting knowledge from data, as well as the theory and analysis around these algorithms [84]. So, a machine learning algorithm is an algorithm that learns from data. The learning aspect is described in [118] as: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured

by P, improves with experience E." Many learning problems falls to this definition, taking image classification as an example: task T is recognizing the object presented in the image, experience E is a dataset of images with the target labels, and performance P is the percentage of correctly classified or identified images. A Machine learning system comprises a number of steps and can be represented schematically as the pipeline shown in Figure 2.1. To tackle a task using a machine learning system, as depicted in the figure, it starts by collecting raw data. The row data is then processed to get features for each instance and create a feature vector, which becomes processed data. The learning algorithm is applied to this data to have a mathematical function called a model, which predicts the labels of future instances.

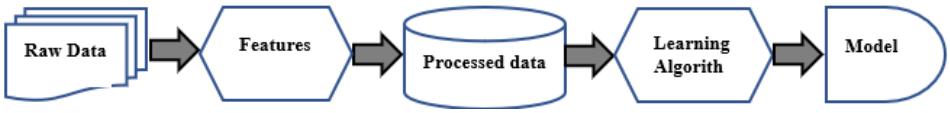

Figure 2.1: *Schematic representation of Machine learning process*

Machine learning can be broadly categorized into three major areas: supervised learning, unsupervised learning, and reinforcement learning. Since our work builds upon decision tree and decision tree ensembles, we discuss supervised learning, decision tree, and decision tree ensembles in the following sections.

2.1.1 Supervised Learning

Methods that are given training examples in the form of inputs labeled with corresponding outputs are supervised learning techniques. Formally, for a given d-dimensional vector space of real-valued features $\mathcal{X} \subseteq \mathbb{R}^d$, an instance $x \in \mathcal{X}$ is a d-dimensional feature vector $(x_1, x_2, \ldots, x_d)$ representing an object. Each instance $x \in \mathcal{X}$ is assigned a label $y \in \mathcal{Y}$ by unknown target function $g : \mathcal{X} \mapsto \mathcal{Y}$. Given a set of hypotheses $\mathcal{H}$, supervised learning algorithms search for a function $\hat{h} \in \mathcal{H}$ that best approximates the target g. The error that $\hat{h}$ makes in predicting the true label y is measured through some *loss function*. By means of empirical risk minimization [168], machine learning algorithms search for the model $\hat{h} \in \mathcal{H}$ that minimizes the given loss function on a given set of training instances. Supervised learning is subdivided in to two subcategories: regression, where labels are real-valued, i.e., $\mathcal{Y} = \mathbb{R}$, and classification, where $\mathcal{Y}$ is a finite set of labels.

Classic examples of supervised learning are spam filtering [108, 174], Image classification [96], and machine translation [155].

There are a number of algorithms which are grouped under supervised learning. Some of them are: Support Vector Machines (SVMs) [48], Deep Neural Network(DNN), Recurrent Neural Network(RNN), Decision Trees, and Decision Tree Ensembles(for example,Random Forest).

2.1.2 Decision Trees

An effective hypothesis space is that of decision trees [22, 139] which consists of a set of tree structured decision tests that predict the value of a target variable by learning simple decision rules inferred from the data. The prediction is generated by traversing the tree starting at its root. Each internal node of the tree represent a feature test which determines further split not yet defined; according to this test data arriving on this node splits into subsets according to their different values on the feature test. A leaf node is associated with a label, which will be assigned to instances falling into this node. For example let us take dummy single decision tree in Figure 2.2 which can be used for credit card fraud detection. The tree trained on credit card transactions dataset where the goal is to distinguish between valid and fraudulent transactions. By just using two features of the dataset,i.e. number of different cities the card has been used in the last 24hrs and the number of transactions, the tree learned how to predict fraudulent transactions.

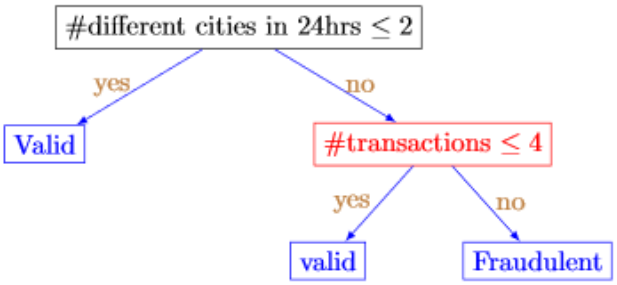

Figure 2.2: *An example of a single decision tree that might be used for credit card fraud detection. Given an input we can follow a path through the decision nodes to reach a final prediction.*

Decision tree learning algorithms are recursive processes. A given data is split at a node and this split is used to divide the data into subsets, and each subset is used as a given data in the next split. Thus, the key in decision tree algorithms is the spitting criterion which determine how to select the

split.

The most known splitting criterion are *information gain* criterion for ID3 algorithm [139], *gain ratio* C4.5 algorithm [140], and *gini index* criterion for CART algorithm [22].

Binary Decision Tree

A binary decision tree has internal nodes that perform thresholding over feature values. Such trees can be inductively defined as follows: a decision tree t is either a leaf $\lambda(\hat{y})$ for some label $\hat{y} \in \mathcal{Y}$ or a non-leaf node $\sigma(f, v, t_l, t_r)$, 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 decision trees (left and right respectively). At test time, an instance $\boldsymbol{x}$ traverses the tree 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 traversed tree 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. We just write λ or σ to refer to some leaf or node of the decision tree when its actual content is irrelevant. The problem of learning an optimal decision tree is known to be NP-complete [81, 121]; as such, a top-down greedy approach is usually adopted [80], as shown in Algorithm 1.

Algorithm 1 BUILDTREE

Input: training data $\mathcal{D}$
$\hat{y} \leftarrow \operatorname{argmin}_y \mathcal{L}(\lambda(y), \mathcal{D})$
$\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r)), \mathcal{D}_l, \mathcal{D}_r \leftarrow$ BESTSPLIT$(\mathcal{D})$
if $\mathcal{L}(\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r)), \mathcal{D}) < \mathcal{L}(\lambda(\hat{y}), \mathcal{D})$ **then**
    $t_l \leftarrow$ BUILDTREE$(\mathcal{D}_l)$
    $t_r \leftarrow$ BUILDTREE$(\mathcal{D}_r)$
    return $\sigma(f, v, t_l, t_r)$
else
    return $\lambda(\hat{y})$
end if

Algorithm 2 BESTSPLIT

Input: training data $\mathcal{D}$
▷ Build a set of candidate tree nodes $\mathcal{N}$ via exhaustive search over f and v
$\mathcal{N} \leftarrow \{\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r)) \mid \hat{y}_l, \hat{y}_r = \operatorname{argmin}_{y_l, y_r} \mathcal{L}(\sigma(f, v, \lambda(y_l), \lambda(y_r)), \mathcal{D})\}$
▷ Select the candidate node $\hat{t} \in \mathcal{N}$ which minimizes the loss $\mathcal{L}$ on the training data $\mathcal{D}$
$\hat{t} = \operatorname{argmin}_{t \in \mathcal{N}} \mathcal{L}(t, \mathcal{D}) = \sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$
▷ Split the training data $\mathcal{D}$ based on the best candidate node $\hat{t} = \sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$
$\mathcal{D}_l \leftarrow \{(\boldsymbol{x}, y) \in \mathcal{D} \mid x_f \leq v\}$
$\mathcal{D}_r \leftarrow \mathcal{D} \setminus \mathcal{D}_l$
return $\hat{t}, \mathcal{D}_l, \mathcal{D}_r$

The function BUILDTREE takes as input a dataset $\mathcal{D}$ and initially computes the label $\hat{y}$ which minimizes the loss on $\mathcal{D}$ for a decision tree composed of just a single leaf; for instance, when the loss is the Sum of Squared Errors (SSE), such label just amounts to the mean of the labels in $\mathcal{D}$. The function then checks if it is possible to grow the tree to further reduce the loss by calling a *splitting* function BESTSPLIT (Algorithm 2), which attempts to replace the leaf $\lambda(\hat{y})$ with a new sub-tree $\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$. This sub-tree is greedily identified by choosing f and v from an exhaustive exploration of the search space consisting of all the possible features and thresholds, and with the predictions $\hat{y}_l$ and $\hat{y}_r$ chosen so as to minimize the global loss on $\mathcal{D}$. If it is possible to reduce the loss on $\mathcal{D}$ by growing the new sub-tree, the tree construction is recursively performed over the subsets $\mathcal{D}_l = \{(\boldsymbol{x}, y) \in \mathcal{D} \mid x_f \leq v\}$ and $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_l$, otherwise the original leaf $\lambda(\hat{y})$ is returned. Real-world implementations of the algorithm typically use multiple stopping criteria to prevent overfitting, e.g., by bounding the tree depth, or by requiring a minimum number of instances in the recursive calls.

2.1.3 Decision Tree Ensembles

Ensemble methods combine several learners in order to have a better performance than a single learner. The main goal of having an ensemble model is to create a strong learner from many weak ones. The main techniques used to build ensembles of decision trees are *bagging* [21] and *boosting* [61,62]. These methods improve the performance over a decision tree by reducing its bias or variance [71].

Bagging. Bootstrap aggregating (Bagging) is a technique proposed by Leo Breiman [21] which reduces the variance of decision trees by training several decision trees on different *bags* of the given training dataset. A bag is achieved by bootstrap sample, i.e., random samples with replacement with the same size of the given dataset. The results from individual trees are eventually aggregated into the final prediction (outcome) for the tree ensemble.

Boosting. Boosting [61, 62, 149] is an ensemble method which iteratively trains individual decision trees, where each tree focuses on the misclassifications of the previously learned ones. Improving the new treeâs performance by reweighting the training data, so that data that is misclassified gets a higher weight.

Random Forest (RF) and Gradient Boosting Decision Trees (GBDT) are popular ensemble learning methods for decision trees [20, 63]. RFs are ob-

tained by independently training a set of trees $\mathcal{T}$, which are combined into the *ensemble predictor* $\hat{h}$, e.g., by using majority voting to assign the class label. Each $t_i \in \mathcal{T}$ is typically built by using bagging and per-node feature sampling over the training set. In GBDTs, instead, each tree approximates a gradient descent step along the direction of loss minimization. Both methods are very effective, where RF is able to train models with low variance, while GDBTs are models of high accuracy yet possibly prone to overfitting.

2.2 Adversarial Machine Learning

Adversarial machine learning has received a great deal of attention recently, with much attention being given to a phenomenon called *adversarial examples*. Commonly defined, an adversarial example takes an input and adds a small, imperceptible amount of distortion, which changes the original to create a new one [156]. Thus, adversarial machine learning embraces necessary techniques to evaluate ML algorithms to identify between benign and malicious samples(adversarial samples). In this context, an adversary which adopts a method to modify benign samples with the aim of misguiding a model with malicious input, this process is called *adversarial attack*. For example, consider an ML model which is used by a bank to grant loans to inquiring customers: a malicious customer may try to fool the model into illicitly qualifying him for a loan. Unfortunately, traditional ML algorithms proved vulnerable to a wide range of attacks.

This existence of attacks against machine learning systems prompts the scientific community to design machine learning systems under the consideration of an adversarial environment. In this case, studying vulnerabilities in the pipeline of a machine learning process for a possible weakness, identifying the potential knowledge of an adversary, and what possible goals can be achieved by an attacker helps to assess the threat posed by an attacker and devise a solution, called *adversarial defence*. Thus, the primary objective in designing a secure machine learning system is to model threats and evaluate their robustness against the corresponding attacks.

2.2.1 Threat Model

The security of the machine learning system is measured with respect to the adversarial goals and capabilities that it is designed to defend against the system's threat model. In this section, we go through the framework and scope of threat models in ML systems and illustrate the space of security models. We begin by identifying the attack surface of ML systems to

inform where and how an adversary will attempt to subvert the system.

The possibility of an adversary devising an attack against a machine learning-based system can happen at any stage of the process. To understand and identify where and how an adversary can attempt to subvert its target, we study the attack surface of a machine learning system. To this end, our work takes the insights and foundation from previous works and build upon [10, 79, 132].

2.2.2 The Machine Learning Attack Surface

Taking a machine learning-based system as a generalized data processing pipeline, an adversary can devise an attack at any stage of this pipeline shown below in Figure 2.3. Hence, an attacker can compromise the training of the machine learning model by injecting carefully crafted instances at the time of raw data collection [136]. An attacker can also affect the feature extraction to subvert the learning process and induce targeted model behaviours by forcing corrupted or noisy training data. For example, such kinds of attacks are used by adversaries to compromise anomaly detectors in spam and network intrusion [93].

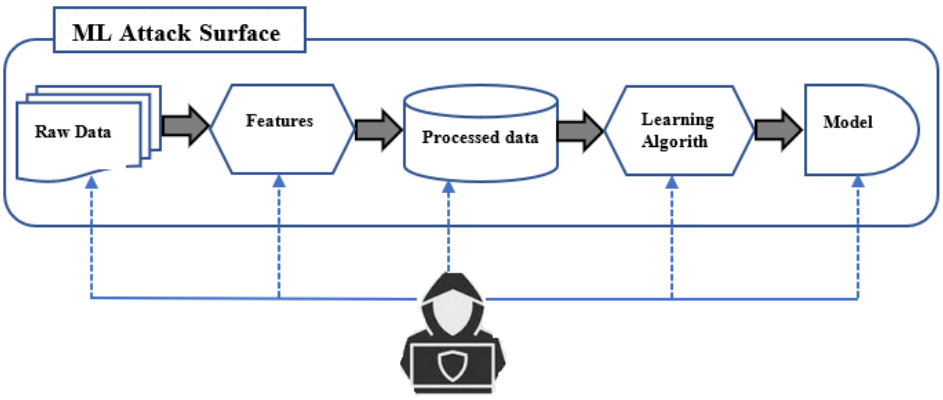

Figure 2.3: *Schematic representation of attack surface for machine learning based system*

Attacks can also happen against the ML system at decision time in which an adversary tries to subvert the learned model. A famous example for this kind of attack is in spam email traffic when spammer replace the letter "i" with a number "1" or a letter "l" in "Viagra" which becomes "V1agra".

All the above scenarios show that machine learning techniques and algorithms are not designed to be robust under an adversarial environment. To

avoid any attack or protect any ML-based system, an ML designer should take into consideration adversarial consideration. To this end, the first step will be modelling a threat.

A threat model can be defined from the perspective of an attacker in terms of the objective an adversary want to achieve, the knowledge of the adversary about the ML system, and the timing of an attack (when and where an attack happens in the ML system pipeline) [18].

2.2.3 Information Available to the Attacker

The knowledge of the adversary has a big impact in the quest of attacking a targeted system. The success and method of attack depends on the amount of knowledge the attacker possesses regarding the model parameters, training data set, and the learning algorithm. Depending on this, there are two categories: *white-box attacks*, in which the attacker know everything there is to know, and *black-box attacks*, in which the attackers' knowledge is limited about the targeted system.

White-box attacks. This setting assumes all information is available to the attacker and the adversary has full knowledge about the system or the model it is attacking. This setting allows the adversary to conduct particularly devastating attacks.

Black-box attacks. Contrary to white-box attacks, the adversary does not have specific information about either the model or algorithm used by the learner. Without substantial knowledge of the feature space, the learning algorithm, or the training data, the attacker can query the system and get an output. The output can be in the form of predicted labels or classification scores [38, 50, 131, 161].

Table 2.1 shows a brief difference between white-box and black-box attacks. In a white-box attack, the attacker has access to the model's parameters, whereas in a black-box attack, the attacker does not have access to these parameters, therefore it generates adversarial instances with a different model or no model at all in the hopes that they will transfer to the target model. Furthermore white-box attack can based its attack on the gradient of the ML model, or heuristic approaches to find smallest perturbation, while black-box attacks uses numerical estimation,boundary attack, or tranferability strategies to produce adversarial instances.

2.2.4 Adversarial Attack Timing

Based on when and where the attack happens [170], i.e., before or after the learning process(training), which can have a causative influence, that

		Scenario	
		White-Box	Black-Box
Attacker	Knowledge	• adversary has Knowledge about learned model,or learning algorithm • access to parameters, hyper-parameters, and data distribution	• adversary does not have knowledge about either learning algorithm or learned model • use the model as a prediction oracle
	Strategy	• based on gradient for continuous ML models • use Linear programming to find smallest perturbation • heuristic approaches	• numerical estimation • boundary attack • transferability

Table 2.1: *Summary of white-box and black-box attack settings*

introduces vulnerability at the beginning in which the adversary is capable of manipulating both training and test data, or exploratory influence which exploits vulnerabilities after training the model by only manipulating test data [9], we can generalize attacks into two groups *evasion* [11, 32, 66, 156] and *poisoning* [16, 83, 115, 175].

Evasion attacks. Evasion attacks are the most known type of attacks where adversaries craft *adversarial samples* that look like normal data instances yet force wrong predictions (wrong label), resulting in misclassification [11, 12, 14]. Adversarial samples are crafted at test time to evade detection and exploit the vulnerabilities of the trained model [104]. Given an instance x, the goal of an evasion attack is to add the smallest perturbation δ, which gives adversarial sample $\bar{x} = x + \delta$ that is able to evade the model into inaccurate prediction. A typical example is changing some pixels of an image so that the model fails to label it correctly.

Different approaches have been used to model evasion attacks in white-box setting. We adopt the abstraction of evasion attack from Vorobeychik *et al.* [170] in which evasion attacks on a binary classifier (also can be adopted to multi-class classifiers) can be represented in three conceptual elements. Given a classifier $f(x)$ and a scoring function $g(x)$ in which a classifier $f(x) = sgn\,(g(x))$, adversarial instance $\bar{x} = x + \delta$, and a cost to perturb x in to $\bar{x}$ represented by a function $c\,(x, \bar{x})$. So, an evasion attack is an attack that changes the sign of the classifier function by using adversarial instances generated by a minimum cost. The most familiar attack

model is using cost measured by norm based distances, $c(x, \bar{x}) = \|\bar{x} - x\|_p$. This cost can also be represented as an optimization problem for finding the smallest distortion; adopted from [49] it can be formulated as: $\underset{\bar{x}}{\text{argmin}} \quad c(x, \bar{x}), \quad \text{s.t.} \quad g(x) \neq g(\bar{x})$. In Chapter 5 we provide an evasion attack strategy with axis-aligned rule based adversarial sample generation for decision trees and tree ensembles along with its defensive mechanism.

Poisoning attacks. poisoning is a training time attack in which the adversary uses direct or indirect methods to deliberately manipulate the training data before training and causes the learning algorithm to make poor choices, and can be used to achieve a specific goal. Influencing the training data has major consequences. For example, in online learning, the model always retrains using live data periodically; if an adversary can change that data robustness of the model will be compromised immensely. Another effect of poisoning is degrading model performance [3, 102] and injecting backdoors [69] into the model, inducing errors when triggered.

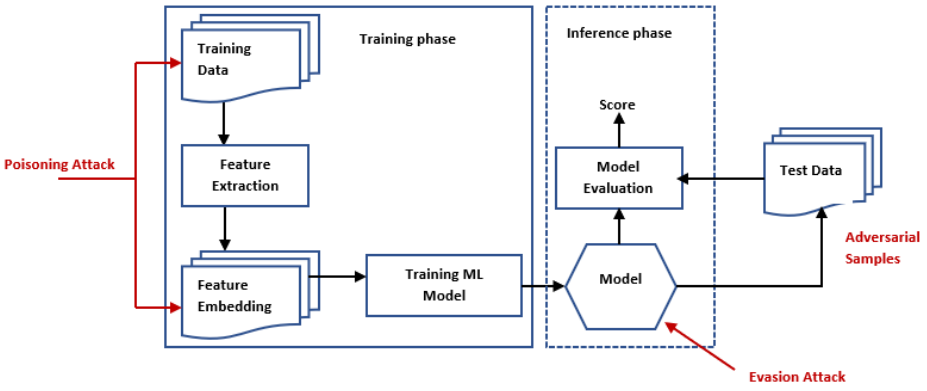

Figure 2.4: *Schematic representation of the distinction between evasion and poisoning attack on the surface of machine learning based system [52]*

2.2.5 Adversarial Goals

While attackers may have a variety of motivations and objectives for attacking machine learning systems, attacks can be defined in terms of *security violation, attack specificity,* and *error specificity*. We describe these attacks based on previous works in [18, 84, 128].

Security violation. The attacker may aim to cause: an integrity violation, to penetrate the system without compromising its normal operation

(minimizing the chance of being detected); and availability violation,to impair the typical system functionality accessible to authorized users ; or a privacy violation, to obtain private information about the system or its users, usually by reverse-engineering the underlying machine learning model.

Attack specificity. We differentiate between targeted and indiscriminate assaults based on whether the attacker intends to misclassify a specified group of samples or any accessible data. The former attacks are usually performed to target a specific system user or protected service.

Error specificity. If the adversary aims to have a sample misclassified as a precise class, we consider the attack as specific; or generic if the attacker aims to have a sample misclassified as any of the available classes excluding its true one. Other works [132] mix error and attack specificity by defining targeted and indiscriminate attacks depending on whether the attacker aims to cause specific or generic errors.

2.3 Learning Robust Machine Learning Models

2.3.1 General Adversarial Attack Defense Methods

Defending adversarial attacks have been studied along with the techniques of attacks. Methods which have been proposed for models such as deep neural network [66, 68, 133], support vector machine [15, 176] can be mentioned. Though the state-of-the-art defence mechanisms for decision trees and tree ensembles are a work in progress, recently, works in [31, 36, 90] show defensive mechanisms for tree ensembles against evasion attacks. A summary of selected recent advancements in adversarial defence mechanisms is presented in Table 2.2.

- **Adversarial Training:** The main objective is to increase model robustness by including adversarial instances into training dataset [**?**, 156]. The defender simply generates adversarial samples and augment these instances while training the model.
- **Gradient Hiding:** Implemented with the aim of hiding the gradient of the model from the adversary [160]. It is ineffective for non-continuous models and can be circumvented by learning a surrogate Black-Box model having gradient and crafting examples using it [131].
- **Defensive Distillation:** Distillation introduced by Hinton et al. [75] is a training procedure in which a model is trained using knowledge from a previously trained model. Papernot et al. [133] proposed a variant of the distillation method as a defensive mechanism in which it uses the

Defense Mechanism	Notable Works	Remark
Adversarial Training	[29,66,90,97,110,145, 150,156]	• adds adversarial samples into the training data • increases model robustness
Gradient Hiding	[46]	• defense against gradient-based attacks • hiding information about the model's gradient
Defensive Distillation	[129,133]	• mitigates adversarial samples crafted by FGSM but week to defend against JSMA
Feature Squeezing	[177]	• reduces the search space available to an adversary
Transferability Blocking	[77]	• prevent the adversarial examples to transfer from one model to another
Reject On Negative Impact	[124]	• removes adversarial samples that are injected into training data

Table 2.2: *Summary of Adversarial defense Mechanisms with notable works for each methods*

knowledge extracted from the model to improve its resilience against adversarial samples. This method smooths the model's decision surface in adversarial directions that could be exploited by the adversary.

- **Feature Squeezing:** This is a model hardening technique with the aim of limiting the complexity of data representation so that the adversarial perturbations disappear because of low sensitivity.
- **Transferability Blocking:** The main objective of this defensive method is to block the transferability of adversarial samples to a model so that a black-box attack will be ineffective. One of the proposed methods called null labelling [77] augments with a new null label the dataset and train the classifier to exclude adversarial cases by marking them as null . Null labelling has three basic steps (1) initial Training of the target classifier on the clean dataset to derive the decision boundaries for the classification task, (2) computing the null probabilities to determine generated adversarial instances belong to a null class and (3) adversarial Training to train the model again using clean training samples and adversarially generated ones.
- **Reject On Negative Impact:** By measuring the incremental effect of each individual suspicious training instance and discarding the ones

with a relatively significant negative impact on the overall performance, it tries to minimize the impact of adversarial instances. This method has been implemented to defend against poisoning attacks in spam filtering [9]. However, it is computationally ineffective since the number of trained classifiers scales linearly with the training set.

2.4 Fairness in Machine learning

The usage of machine learning techniques and algorithms in systems and applications affects our lives in many ways, from widespread algorithms we use every day with small interactions, e.g., search, recommendation, and social media, or specialized algorithms with fewer but higher-stakes interactions that are deployed to make important and life-changing decisions in critical areas, e.g., medicine, criminal justice, and finances. Thus, with this level of impact, systems and applications which use machine learning can have unintended consequences. As such, when using machine learning models or systems which are powered by it, accomplishing or aiming low prediction error is not enough. Therefore, it is crucial to ensure that decisions do not reflect discriminatory behaviour toward certain groups or populations based on certain characteristics such as age, race, gender, or political affiliation. A most known example of this problem is a case management and decision support tool used in the United States courts to assess the probability of a defendant committing a crime again, which helps to make pretrial detention and release decisions. The software called Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) helps judges to decide whether to release or keep a defendant in prison. It is revealed in an investigative article [1] that this tool has a bias against African-American defendants [24]. Another example is a facial recognition tool in a digital camera that over predicts Asians as blinking [2].

Recent studies [127, 148] showed the unintentional bias coming from machine learning algorithms. A list of examples discovered where artificial intelligence systems and applications have unintentionally encoded biases and introduced systematic discrimination in AI chatbots, loan application [111], dating, flight routing, immigration algorithms, and hiring processes. The problem also extends to other areas like face recognition applications, voice recognition, and search engines [78].

Hence, to uncover and rectify the likes of biased predictions and decisions mentioned above and address the problem,*fairness in machine learn-*

[1] https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-senten
[2] http://content.time.com/time/business/article/0,8599,1954643,00.html

ing become a prominent research field addressing the cause of bias and discrimination, mitigation methods, and unfairness evaluation metrics.

2.4.1 Sources of Unfairness

Bias is frequently linked to inequities and prejudice in machine learning systems. The word bias is frequently used in this context to refer to societal problems with demographic discrepancies. Defining disparities and the learning problem from a statistical point of view, at different stages of the machine learning pipeline discrepancies can happen and become a cause for unfairness [7]. Thus, potentially discrimination can come from the training data, learning algorithm, and predictive evaluation metrics.

2.4.1.1 The Data

The foundation of the machine learning model is the data on which it is trained. The data available for training a predictive model, especially measuring and categorizing tasks, When used as a basis for decision making, can lead to unwanted properties. These kind of data with undesirable properties are called "biased data" [8]. The notion of bias in data can be grouped into two depending on where the bias comes from. The first is *statistical bias*, i.e., concerns about non-representative sampling and measurement error, and the second reason for data bias is *societal bias*, i.e., concerns about objectionable social structures that are represented in the data.

Statistical bias– It is a systematic mismatch between the sample used to train a model and the world as it currently is. Specifically, a statistical bias comes from sampling bias and measurement error which induce fairness implications that are usually unmeasured by fairness definitions. *Sampling bias* occurs when a dataset is not representative of the full demographics to which the ML model will be applied. Incorrectly assuming the sample is representative can lead to a number of incorrect assumptions such as biased estimation of conditional probabilities, biased estimation of utility, and inadequate fairness adjustments [85]. Similarly, when an error is greater for some groups than others, *differential measurement error* [167] can occur and will result in an outcome with a profound consequence. For example, using historical loan repayment to predict future payments might be disadvantageous for groups that do not participate in a formal loan system in the past.

Societal bias– Even though the training data has an accurate demographic representation, it may still have objectionable social structure data points, if considered in a policy-making, will result in an outcome counter

to the decision maker's goals, and it is a non-statistical notion called societal bias [154]. For example, Using arrests as a metric of crime rate may create statistical bias due to measurement inaccuracy that varies by race due to a racist policing system. Moreover, even though we manage to measure crime perfectly, the data might still contain bias from the normative sense because of reasons like how crime is defined.

2.4.1.2 The Learning Algorithm

Even though there is no bias in the training data, there still be a bias that comes from the learning algorithm [6]. The choices in algorithm design, such as to use optimization functions, to use regularization, to apply regression models on the data as a whole or to consider subgroups, and to use statistically biased estimators in general, can contribute to a biased and unfair decision.

2.4.1.3 Evaluation

The evaluation metrics used to select a trained model can be a source of bias. Most measurements assume that final decisions as an aggregation of separately evaluated individual decisions; in which it is assumed aggregated decisions are similar are made simultaneously. Thus, deciding the extent to which prediction measurements assuming those conditions will have a potential impact in fair decision making.

2.4.2 Measuring Fairness

Considering philosophical and psychological points of view, fighting discrimination has a long history, and in recent years in machine learning. However, in order to achieve fairness and be able to remove discrimination in decision making, one should first define fairness and evaluate according to defined measurements.

2.4.2.1 Notion of Statistical Fairness

Though it is intuitive to understand and develop a sense of discrimination in decision making, it is not clear what it means to be fair for systems that use machine learning. In an algorithmic decision system, two families of fairness notions are proposed: *individual fairness* and *group fairness*. Individual fairness deals with fairness guarantee at an individual level while group fairness, on the other hand, require some statistic of a classifier to equally hold across some defined protected subgroups; the subgroups defined by *sensitive feature*(we will use a feature and attribute interchangeably) such

as gender, ethnicity, age, and political affiliation etc.; defining *privileged* and *unprivileged* groups depending on whether the subgroup is favoured to be positively classified.

Taking the notion introduced previously, let an individual instance i of a dataset $\mathcal{D}$ with d-dimensional feature vector represented as x_i and its corresponding label y_i, and a feature $S \in \{0, 1\}$ representing a *sensitive feature* which define the subgroup of an individual belongs to. In this regard, many fairness definitions have been introduced [169]. In this work, we focus on statistical fairness definitions in the binary classification setting. We presented well-known group and individual fairness definitions below and explained each with an example. Throughout the definitions below, we use $\hat{Y}$ as model prediction, Y true label, and S sensitive attribute.

Demographic Parity– Sometimes called statistical parity, which the prediction to be independent of a sensitive attribute, i.e., its objective is that subgroups defined by sensitive attribute must have the same positive prediction rates [58]. Demographic parity is used in applications such as predicting crime, hiring and giving loans. To explain more, let's take an example of predicting recidivism, assuming race as a sensitive feature, predicting the rate of getting no recidivism for White-American and African-American should be equal. In Chapter 6 we look into an approach to achieve the demographic parity when we actually build a classifier.

$$P(\hat{Y} = 1|S = 0) = P(\hat{Y} = 1|S = 0) \tag{2.1}$$

Equalized Odds– This measure ensures the criteria of demographic parity and extends that subgroups defined by sensitive feature to have the same positive prediction rates but for two cases: when the true label is 0 and when it is 1. In other words, it ensures the false positive rate (FPR) and false negative rate (FNR) are equal across the subgroup [70].

$$P(\hat{Y} = 1|S = 0, Y = A) = P(\hat{Y} = 1|S = 0, Y = A), A \in \{0, 1\} \tag{2.2}$$

Equality of opportunity– Also called predictive parity and a relaxed version of equalized odds where only the true positive rate (TPR) of both subgroups to be the same [44].

$$P(Y = 1|S = 0, \hat{Y} = A) = P(Y = 1|S = 0, \hat{Y} = A), A \in \{0, 1\} \tag{2.3}$$

Individual fairness– In another definition for fairness, Dework *et al.* [54] introduced individual fairness criteria in which it ensures predictions for

similar people are similar as well. This metric can be defined using a distance metric in which one can measure the degree of similarity between individuals. Individuals with the same condition must have the same prediction regardless of other criteria.

2.4.3 Unfairness Mitigation

There have been many attempts addressing fairness in machine learning; most approaches addressing the statistical notion of fairness are categorized into three: pre-processing, in-processing, and post-processing techniques. Though these techniques decrease the discrimination level of a system, they come with the cost of reducing predictive performance. A pictorial representation of the three approaches is depicted in Figure 2.5.

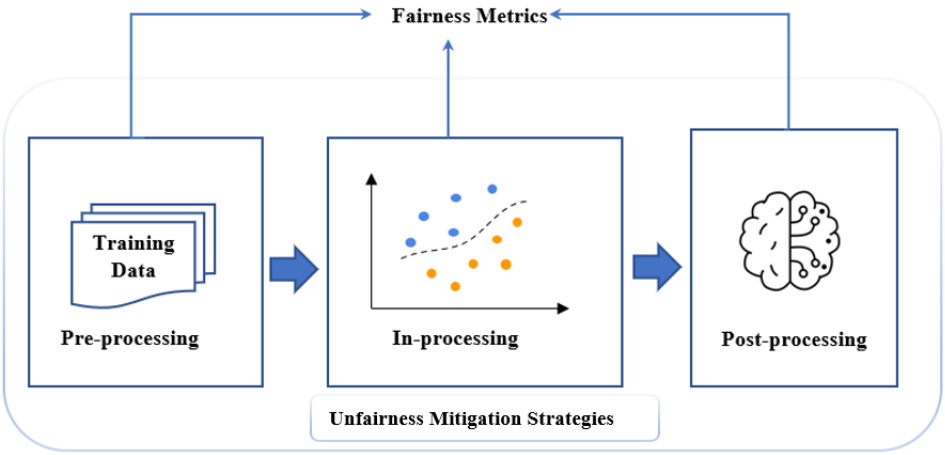

Figure 2.5: *Schematic representation of algorithmic interventions to achieve statistical notions of fairness in machine learning.*

Pre-processing– In this family of approaches, a dataset is pre-processed to decrease discrimination before the learning algorithm is trained. In most of the pre-processing approaches, fairness is achieved by transforming the feature space independent of the sensitive attribute. The most known pre-processing techniques are listed below.

- *Suppression.* This is a simple and straightforward approach that removes features that are correlated with the sensitive feature [87]. If the relation between the sensitive feature and other features is not linear, this approach might not be effective.
- *Massaging.* This approach changes the labels of some instances in the

dataset to remove discrimination in the input data [86]. The instances chosen to be their labels changed are from both subgroups defined by the sensitive attribute. To achieve a minimal effect on the predictive performance, this method picks candidates that are close to the decision boundary using a ranker.

- *Reweighing.* In reweighing approach, instead of changing labels, it assigns different weights in each subgroup, labels combination to ensure the training dataset is discrimination-free [87].

- *Optimized preprocessing.*This technique learns a probabilistic transformation that transforms the features and labels in the data with group fairness, individual distortion, and data fidelity constraints and objectives [27].

In-processing– This technique is used for fairness intervention by mitigating disparities and modifying the training process. Generally, approaches in this category can be grouped into two: Adversarial debiasing and prejudice removers.

Adversarial Debiasing. This technique learns two different classifiers at the same time in which the first classifier tries to maximize accuracy, and the second classifier an adversary tries to determine the sensitive attribute from the prediction of the first classifier. In other words, the predictor tries to reduce the adversary's ability to determine the sensitive attribute while the adversary tries to accomplish guessing the sensitive feature [181]. This approach leads to a fair classifier where the predictions cannot carry any group discrimination information that the adversary can exploit [67, 106, 172].

*Prejudice Removers.*This approach alters the training of the learning algorithm to remove discrimination. By introducing fairness criteria as a constraint, this method decreases discrimination while training. One such algorithm proposed by Zafer *et al.* [180] formulate an optimization problem in which, given a decision boundary-based classifier (e.g. logistic regression or linear support vector machines), it minimizes the loss subject to a fairness constraint. Another approach takes fairness metric as part of input in the meta-algorithm and returns an optimized classifier subject to the fairness metric [1]. Kamiran *et al.* [88] included a discrimination factor into the information gain splitting criterion of a single decision tree classifier by considering the split of a node under the influence of a sensitive feature, i.e., before a node split happens, not only the usual purity w.r.t. to the target label is calculated, but also the purity of the split w.r.t. the sensitive feature.

A cost-sensitive constrained optimization problem with fairness constraints for a classification problem of two players is proposed in [1].In this work, while one player optimizes accuracy, the other imposes a particular fairness constraint, yielding a randomized classifier with the lowest predictive error while satisfying fairness definitions such as equal odds, equal opportunity, and demographic parity.

Post-processing– The general technique in this category is to select a subset of samples and modify their predicted labels to meet fairness requirements. By choosing samples randomly works in [70, 138] solves a linear program to find probabilities with which to change output labels to optimize equalized odds. In another approach which gives favourable outcomes to unprivileged groups and unfavourable outcomes to privileged groups by choosing the most uncertain samples in the reject option band around the decision boundary [89]. Another work proposed in this category called *relabeling* which identifies the most discriminate leaves of a tree to relabel [88]. The authors combine this approach with a discrimination aware in-processing approach to get a better accuracy-discrimination trade-off. In Chapter 6 we extend the relabeling approach to a binary forest classifier that flips leaves to satisfy group fairness.

2.5 Summary

In this chapter, we cover the basic foundation of machine learning and its techniques. We started by discussing the basic concept of machine learning and delved into supervised learning, giving an in-depth understanding of decision trees and tree ensembles. In Section 2.2, by explaining the security vulnerability of machine learning, we explain the study of adversarial machine learning in-depth, define a threat model in which we cover the attack surface, information available to an attacker, attack timing, and attacker goals. We also provide general adversarial defence mechanisms and how to learn robust machine learning in Section 2.3.1.

In Section 2.4, we gave an explanation of the need for fair machine learning by listing and expanding sources of unfairness, fairness measurements, and mitigation methods to unfairness.

CHAPTER 3

Adversarial Machine Learning Targeting Tree-Based Models

Alternative to Deep Neural Networks(DNNs), tree-based models achieve state-of-the-art performance in the domain of non-perceptual data. However, research in the field of adversarial learning has been mainly focused on linear models and neural network models. This has changed recently, and the effect of an adversarial attack on tree-based models started to be studied and showed that they are vulnerable to adversarial attacks and exploited by deceptive input samples. In this regard, we provide a survey of adversarial machine learning targeting tree-based Models.

3.1 Introduction

Machine learning has been applied in a wide range of fields and improves performance in any area of applications [17]. Applications in computer vision [64, 91, 151], autonomous driving, and in applications of system security [45, 59, 76, 92, 94, 98, 123, 142] shows success and effectiveness of machine learning. The emergence of big data from the internet of things integrating the world increases the importance and use of machine learn-

ing [104]. Due to this, an attack on systems leveraging machine learning has potential damage that leads to a compromised situation. These attacks can damage the efficiency of a system, exploit any vulnerability or compromise predictive results [104]. Even though the development and design of machine learning models increasingly advanced to accommodate many shortcomings and vulnerabilities, there still exist methods of exploiting weaknesses to devise attacks against systems that employ machine learning models. Thus, crafting attacks in addition to defensive mechanisms of those attacks, called adversarial learning, has become an active research area in the past decade [10, 17, 117]. Thus, research [156] on the vulnerability of machine learning algorithms to adversarial samples, i.e., carefully-designed and imperceptible samples to fool a model [66], significantly increased and led to extensive studies [119, 126, 156].

Mainly, researchers focused on linear models and deep neural networks [32, 103, 109, 116, 132, 133] to understand the effect of adversarial attacks and devise defensive mechanisms. For example, Deep Neural Networks (DNNs) become the big focus area of adversarial learning research once it is known that they are vulnerable to deceptive samples [156]. Defence methods are also proposed to counter proposed attacks. Since most of the models are assumed to be differentiable and use gradient-based optimizers, the proposed attacks and defensive mechanisms emanate from it. But, it is difficult to generalize those methods of attacks and defences to non-differentiable models, such as decision trees. Due to this, in recent years, studies of tree-based models gained increasing interest in both attack and defence perspectives [90].

Tree-based models are competitive and give state-of-the-art performance [35], and in addition to their easiness to interpret and efficiency, it is known that tree-based models generate understandable rules in processing and exploring datasets. Such behaviours and advantages set them as an alternative to DNNs, which appeals to many application areas, especially in the security domain.

In recent years surveys on adversarial learning have been done [17, 104, 117], giving a general overview of adversarial machine learning. In those works, a comprehensive literature review on machine learning, adversarial learning and recent works on attacks and defensive techniques are discussed. Besides an in-depth analysis and history of Adversarial machine learning, Biggio *et al.* [17] explained a well-defined adversarial model consisting of four dimensions: goal, knowledge, capability, and attack strategy. Q.Liu *et al.* [104] review literature that covers threats in the training and testing phase of machine learning and their defensive mechanisms from a

data-driven point of view. Miller *et al.* [117] focused on defensive mechanisms for test-time evasion, data poisoning, backdoor, and reverse engineering attacks targeting deep neural network classifiers. Most of the reviews mentioned are focused on deep neural networks, giving small attention to those researches focused on decision trees and tree ensemble-based models. Hence, in this chapter, we provide a comprehensive review of the research efforts of adversarial machine learning by giving particular attention to decision trees and tree ensemble models.

3.2 Distortion

In most of the attacks, distortion using adversarial perturbation can be calculated as a distance, that is, the distance between the original instance and its' generated malicious version. It also can be modelled as a cost to be minimized, capturing the difficulty of modifying the benign instance, with the adversary incurring no cost for leaving the instance unperturbed. The following three metrics are well known in the adversarial learning research community for measuring distortion.

- $\ell_0 - norm$: captures localized perturbations with arbitrary magnitude. Given an instance $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a possible perturbation $\bar{x}$, we have that $\|\bar{x} - \boldsymbol{x}\|_0 = |\{f \in [1, d] \mid \bar{x}_f \neq x_f\}|$, and thus the ℓ_0-norm simply counts the dimensions of $\boldsymbol{x}$ that were actually perturbed. For example, attacks that use $\ell_0 - norm$ on image counts the number of modified pixels.

- $\ell_2 - norm$: attacks that calculate their cost using $\ell_2 - norm$ calculates the sum of squared differences between original and perturbed features of an instance. Given an instance $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a possible perturbation $\bar{x}$, we have $\|\bar{x} - \boldsymbol{x}\|_2 = \sum_f (\bar{x}_f - x_f)^2$.

- $\ell_\infty - norm$: this measurement is the simplest one and aims to minimize the amount of perturbation that can be applied to generate adversarial samples. It encourages uniformly spread perturbations with small magnitude. Given an instance $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a possible perturbation $\bar{x}$, we have that $\|\bar{x} - \boldsymbol{x}\|_\infty = \max_f |\bar{x}_f - x_f|$.

Though the above cost measurements are the most well known, there are works of literature that use different approaches for perturbation cost as in the works of Calzavara *et al.* [31], which we will discuss in detail in Chapter 5.

3.3 Review of Attacks Against Decision tree and Tree ensembles

In an adversarial attack, we look for the smallest perturbation that will lead to the wrong classification of the instance we perturb. Several adversarial attack strategies were proposed for neural networks and other continuous models. Some of the attacks are gradient-based methods, formulating the attack into an optimization problem based on a specially developed loss function for attacks, with the gradient obtained using either back-propagation in a white-box scenario [32,112] or numerical estimation in the soft-label black-box setting [38, 82, 163]. In hard label black-box settings, where the only access is the output label, decision-based attack methods can be applied, which usually starts with an initial adversarial example and minimizes the perturbation along the decision boundary [23, 25, 37, 41, 42].

Because tree ensemble models are non-continuous models, the above-mentioned gradient-based attacking methods can not be used. Decision-based attacking methods in black-box settings can be used against decision tree and tree ensemble models, but it may require a large number of queries to find the smallest perturbation to generate an adversarial instance. Due to this, in recent years, new approaches have been proposed for attacking decision trees and tree ensembles. Next, we will see those attack strategies by grouping them into white-box and black-box attacks depending on their settings.

3.3.1 White-box Attacks

In a white-box setting, it is assumed that an attacker has full access to the features, thresholds, and structure of a single tree or an ensemble. Attacks are proposed to show the weakness of tree-based models in this approach. Early work is a greedy search approach algorithm for attacking a single tree proposed by [130]. The algorithm exploits the structure of a tree and uses a leaf with a true label to find another leaf in its neighbourhood with a different label, then changing the feature values according to the splitting conditions along this path, forcing the same sample to be misclassified. The algorithm does not limit the amount of perturbation to produce adversarial samples. In later work, a variant of this algorithm with $l_\infty - norm$ perturbation is implemented by [35], with the minimum perturbation size it takes to attack a robust single decision tree.

An Evasion attack against tree ensembles has been proposed by [90]. This algorithm, by relying on Mixed Integer Linear Programming (MILP)

finds the smallest perturbation measured by $l_p - norm(p = 0, 1, 2, \infty)$. Constructing the attack using mixed-integer linear program, where the variables are tree nodes, and the objective is to minimize a distance between the evasive sample and the attacked data instance, and the constraints of the program are based on the model structure. The constraints include model mislabel requirement logical consistency among leaves and predicates. But, relying on MILP when attacking large scale tree ensembles makes it too expensive and time-consuming.

Finding the exact solution of minimal adversarial perturbation requires exponential time. Due to this, Kantchelian *et al.* [90] introduce a faster approximation method to generate adversarial samples using symbolic prediction for minimizing $l_0 - norm$. The algorithm iteratively changes feature values until the label of the instance being perturbed changes. In a later implementation of this algorithm by Chen *et al.* [35] use (RobustTrees) $l_p(p = 1, 2, \infty)$-norm minimization to check its effectiveness as evaluation of robustness in the adversarial training setting, and the authors indicated that the fast approximation method is not effective for norms other than l_0.

A faster algorithm Zhang *et al.* [183], which simplified the optimal evasion attack in [90] to a reduced and less costly strategy of $0-1$ Integer Linear programming (ILP), has been introduced for a particular type of model which restricts both input and prediction of every tree of the ensemble to binary values. This algorithm depends on the input and output analysis of the targeted system. An optimal evasion problem as the minimization of l_0 distance is formulated as $0-1$ ILP with constraints. The constraints are (1) ensuring any successful attack will change the label(sign) of the output (2) ensure that there is only one decision making path (3). Even with the introduction of attacks, it preserves the semantics of the binary decision tree. As a complement to their optimal evasion problem solution, Zhang *et al.* introduced a heuristic evasion algorithm that finds the most important features which appears most frequently and closer to the root node in a decision tree and greedily modify the single best feature at each iteration until the attack become successful in producing adversarial samples. This approach is also extended in black-box settings, which will be discussed in the next section.

Another specially designed attack method, called LT-Attack, proposed in [182], transforms an attack into a discrete search problem for generating adversarial examples for tree ensembles by exploiting the structure of the ensemble. By Transforming the input space into a tuple consisting of the number of leaves in a single tree for each tree of the ensemble, they create a new input space called "leaf tuple". After selecting a random adversarial sample from potential candidates to start the attack, their algorithm greedily

updates the initially chosen adversarial samples until it is "close" to the target (in terms of l_p norm). This is done by performing an iterative search over the leaf tuples space and stopping when no more adversarial examples with a smaller l_p norm exist.

Young *et al.* [178], exploring the theoretical aspects of search space for adversarial samples, pointed out that not only the non-continuity behaviour of tree-based models that makes it difficult to craft an attack, there is also a challenging aspect of no change in prediction within a sub-group of input space. If the input sub-space is large enough, continuous perturbation of instance to change the label within the region is inefficient. On the other hand, well-separated data points of input space lead to classification regions that are robust to adversarial attack, which comes from low-dimensionality. The authors analyzed the fore-mentioned theoretical concepts and proposed dividing the input space into a smaller subset of N^K (where N is the number of leaves of a single tree and K is the number of trees in the forest) convex polyhedrons containing training instances. Perturbing an instance is finding the closest polyhedron to the selected instance where the classifier predicts a different label and output the closest point in this region. The authors solve convex problems and get the optimal solution or upper bound. Acknowledging the exact solution might be computationally expensive, they provide a heuristic method for an approximate solution, called approximate region-based attack (RBA-Appr).

3.3.2 Black-box Attacks

In a real-world scenario, a white-box setting is not realistic due to the fact that the possibility that underlying information about the machine learning model will be revealed is slim, and thus white-box attacks cannot be applied. A valid attacking strategy is to make queries to the model and get the corresponding output without having the architecture or smoothness assumptions of the model. A number of algorithms and methods are proposed for a black-box attack using either numerical estimation in the soft-label black-box settings where the attacker has full access to the output probabilities of the model or decision-based black-box attacks where the attacker only have access to the output label. Comparing the soft-label and hard-label black-box attacks, queries in the soft-label setting will have more information for each query, which makes the strategy more open to a variety of attack strategies.

An early work on black-box attack is using transfer attack by Papernot *et al.* [130]. Instead of attacking the original model, attackers try to con-

struct a substitute model to mimic the original and then attack the substitute model using white-box attack methods. Adversarial examples crafted using a substitute model can be used on the original model to which the attacker has no direct access. The authors showed an intra and cross technique transferability attack for decision trees. Transfer attacks between two decision trees (intra-transferability) are more robust to such an attempt, but this work shows that decision trees and tree ensembles are more vulnerable to adversarial samples, which are crafted using other models such as DNN, SVM, and others.

In [23] Brendel *et al.* a boundary attack in which attacks are generated via random walks along the decision boundary with rejection sampling and only rely on the final model decision. The algorithm starts from a large adversarial perturbation and then seeks to reduce the perturbation while staying in the adversarial region. In other words, the algorithm is initialized with an input of the desired class and then takes small steps along the decision boundary to minimizing the distance between the original input and the adversarial sample. To reach to the smallest perturbation to the benign class, random sampling is used at each step to find the direction that leads to the region which has the smallest distance and still remains adversarial. The algorithm does not guarantee convergence though it aims to generate a high-quality adversarial sample.

Brunner *et al.* [25] pointed out that the boundary attack by Brendel *et al.* uses unbiased sampling for perturbation and makes it easy to implement and avoid this attack using either detection or filtering method of defensive mechanisms. To avoid this, they introduce constraints to the boundary attack to narrow the perturbation search space to samples which give a higher probability of getting a successful attack.

Using random walk around the boundary to find an upper bound solution needs a lot of queries to explore, even in a constrained setting that avoids samples with less probability of leading to a successful perturbation. To avoid computing too many queries and non-convergence issues Cheng *et al.* [41] propose an algorithm that formulates a hard-label black-box attack as a real-valued optimization problem, where the objective function can be evaluated by binary search with additional model queries. Though primarily devised for continuously differentiable models, the authors showed an untargeted attack against gradient boosting tree successfully find the smallest distortion to attack the model. Even though it shows how effectively attacks the model, the implementation of this method to high dimensional spaces will be computationally expensive since it uses many optimization solvers [183], hence requiring a large number of queries [42].

To reduce the number of queries required, authors of [42] reformulate [41] algorithm for calculating the sign of the directional derivative instead of the magnitude using a single query using an optimization algorithm based on zeroth-order optimization called, Sign-OPT and leads to fast convergence. In [37] Chen *et al.* proposed an unbiased estimate of the gradient direction at the decision boundary to improve the Boundary Attack. In each iteration, the adversarial example first approaches the boundary via a binary search, then moves along the estimated gradient direction to deviate from the decision boundary.

Even though the above mentioned attacking strategies are highly effective on continuous models and can be used on non-continuous models, their effectiveness does not scale to the decision tree, and tree ensembles due to their discrete nature need a large number of queries and have poor performance under the ℓ_∞ setting [39].

Another black-box attack proposed by Andriushchenko and Hein [4] is a query-efficient ℓ_∞ attack on boosted trees called cube-attack. The algorithm is based on an evolutionary algorithm, and on every iteration, a random subset of samples are selected and perturbed, success as a minimum perturbation that changes the final output. The new instance is kept if there are no potential other adversarial examples that are better than the old value. The adversarial sample is always positioned at a corner of the feasible set of ℓ_∞ ball. The algorithms' stochastic updating along the ℓ_∞ boundary usually achieves a better result than decision-based attacks.

3.4 Review of Defenses Proposed for Tree Ensembles

3.4.1 Adversarial Training

Adversarial training is an approach of creating adversarial samples and including them into the training process so that the learned model is going to be aware of attacks and become more hardened to an adversary [66, 97]. Unlike other defence strategies, the augmentation of training data with adversarial examples in the training phase is done intrinsically with the aim of enhancing robustness. Thus, it makes models which are trained adversarially to behave normally when encountered adversarial samples than models which are trained in standard settings.

The first adversarial training method for tree ensembles is introduced by Kantchelian *et al.* with the concept of including adversarial samples into training data, called adversarial boosting [90], focusing on l_0 perturbations. This method at each iteration of the boosting generates perturbations

greedily until it gets to the smallest, which have the largest impact on the model trained in that boosting round. Though this method provides a way to coerce the learning algorithm to minimize error, it does not necessarily guarantee an effective robustness performance under attack.

Prominently known adversarial learning methods assuming models to be differentiable makes it difficult to apply those techniques to decision trees and tree ensembles. by addressing this, Calzavara *et al.* [29] generalize adversarial training for gradient-boosted decision trees. By taking advantage of the nature of growing a decision tree and its use of thresholds, the authors pre-compute all possible thresholds which can be used during the tree growth and reduce the number of possible instances to be perturbed into a finite set of instances. Since adversarial training has to assure the minimization of loss of a model under attack, available optimization methods can not be employed on tree-based models. To address this, the authors implement classifier independent approximation function *LogSumExp (LSE)* function. In addition to this, their work includes a method that specifies the attacker's capabilities, which are described as rewriting rules and subjected to a budget limitation.

A genetic adversarial training algorithm is applied to decision trees and tree ensembles by Ranzato *et al.* [145] called Meta-Silvae to make the model more robust. The algorithm performs an abstract interpretation based on static analysis of a decision tree classifier which abstractly computes the exact set of leaves for a decision tree that is reachable from an adversarial region. Using this information, the Meta-Silvae maximizes a function of accuracy and robustness. This genetic algorithm uses elitist selection strategy, roulette wheel selection, single-point, crossover, and offspring mutation strategies. In addition, formal verification of robustness of a decision tree is also included by taking advantage of abstract interpretation.

3.4.2 Robust Optimization

The first defence mechanism, which accounts for the attacker in the process of decision tree training and optimally finding the best splitting feature and the threshold at the node level, is proposed by Chen *et al.* [35] as a robust optimization problem, which is called Robust Trees– considering ℓ_∞ norm bounded ball with a radius of ϵ to be used as a budget, the distance between instances is taken into consideration (along the boundaries of the ℓ_∞ ball) to find robust feature and threshold, then evaluate the worst-case performance under the smallest perturbation distance and worst-case accuracy under adversarial attack.

Given an instance x, perturbation within ℓ_∞ ball of radius ϵ is expressed as, $B_\epsilon^\infty(x) := [x-\epsilon, x+\epsilon]$. Chen *et al.* consider the worst-case perturbation, which results in the worst tree splitting function score with feature j and threshold η. Then they present the max-min optimization problem (we use the name robsut_score as presented in the referenced work). An instance x with feature j and threshold η, the best feature j^* and best threshold η^* for x which yields the split under robust training that uses all the perturbations of x in ℓ_∞ ball of radius ϵ is:

$$
\begin{aligned}
& j^*, \eta^* = \underset{j,\eta}{\operatorname{argmax}} \quad \min_{\bar{x}} \quad \text{score}(j, \eta, \bar{x}), \\
& \text{s.t:} \quad \bar{x} \in B_\epsilon^\infty
\end{aligned}
$$

The authors pointed out that the optimal tree construction will constitute an exponential number of attacks. Under adversarial attacks, instances will fall into the right or left side of the decision tree. But, there are instances that will fall to the left or right depending on the perturbation, which they put those unknown instances in a set and called ambiguity set. The authors state that those unknown instances under maximum attack needed to be optimal in tree construction, and the optimal tree construction on the elements of this set constitute an exponential number of attack and formulate it as a $0-1$ optimization problem, which is nonlinear. To approximate this optimization problem, the authors propose four cases 1)No perturbation(original split), 2) perturb all instances to the left, 3) perturb all instances to the right and 4) swap left and right instances. Even though this method can increase the robustness compared to regular training but does not scale.

A cost-aware and security domain constrained algorithm is presented in [40], in which a robust tree ensemble is trained with a robust split. This greedy algorithm takes inspiration from [35] and approximates the worst quality of node split into a best worst-case split. The cost constraint function allows change to feature value in a specified interval by using the security domain knowledge, which can be used to specify the cost constraints. The authors show the algorithms' application in the security domain using spam URL detection on Twitter.

A recent work called, GROOT [171] takes inspiration from TREANT Chapter 5 and [35], and come up with a faster implementation with flexibility that allows users to specify attacks in terms of axis-aligned perturbations. It fits binary classification decision trees such that they are robust against user-specified adversarial examples.

Another approach of robustness via robust optimization problem, which

finds the minimal perturbation with respect $\ell_\infty - distance$ and results provably guaranteed robust decision trees and optimal robust decision stumps, called robust loss bound. Andriushchenko and Hein [4] introduce element-wise bounding of an ensemble for boosted trees which minimize an upper bound on the worst-case loss not just of a new weak learner but also of the whole ensemble. In this way, every new tree learned in the ensemble is able to focus on the undecided instances that the previous ensemble couldn't classify robustly well according to the upper bound on the worst-case loss. For decision stumps, they come with a method of finding the exact robust boosted decision stumps by solving the attack model and optimization objective function with respect to the input dimensions separately, which results in solving simple one-dimensional optimization problems.

Algorithm	Run Time	Threat Model
TREANT [31]	$\mathcal{O}(n^2)$	Re-writting Rule based
GROOT [171]	$\mathcal{O}(n \log n)$	Norm (ℓ_∞) and Re-writting Rule based
Provably robust boosting [4]	$\mathcal{O}(n^2)$	Norm based (ℓ_∞)
Heuristic Chen *et al.* [35]	$\mathcal{O}(n \log n)$	Norm based(ℓ_∞)
Exact Chen *et al.* [35]	$\mathcal{O}(n^2)$	Norm Based(ℓ_∞)

Table 3.1: *Summary of selected Algorithms,their running time and threat model*

3.5 Robustness Verification and Evaluation

The idea of robustness is that a model's prediction is stable in the face of small changes in the input, ideally because the prediction is based on reliable abstractions of the real task that reflect how a human would perform the task. Given an input x and a model f, we want the model's prediction to stay the same for all inputs x' in the neighbourhood of x, where the neighbourhood is defined by some distance function c and some maximum distance Δ [65]. Therefore, the robustness of the model can be defined as follows:

$$\forall x', c(x, x') \leq \Delta => f(x) = f(x').$$

Robustness verification for deep neural network has been given much attention due to the fast-evolving adversarial ML studies in the area. This can be achieved by searching for adversarial instances in the neighbourhood of a particular example or testing a model with a large number of data points within a given bound. Those strategies are not effective on tree-based models, and recently, literature on verification of tree-based models' robustness started to emerge. In the next sections, we present state-of-the-

art works on methods and strategies to verify and evaluate robustness on decision tree and tree-based models.

3.5.1 Verification

Verifying robustness requires either finding the exact smallest adversarial perturbation or a guaranteed lower bound of the minimal adversarial perturbation. It is shown in [90] finding the exact smallest perturbation for tree ensemble is NP-hard, and finding minimal perturbation is proven to be NP-complete and they proposed Mixed integer linear programming approach which is used to achieve a complete verification that requires exponential time. This makes the algorithm unrealistic to implement for large tree ensembles.

Representing ℓ_∞ perturbation ball and the decision boundary of a leaf node as boxes, Chen *et al.* [36] formulate robustness verification problem of tree-based models as a graph problem by checking the intersection between those boxes. For an ensemble with K trees, it is possible to use the boxicity property to obtain intersections, and verifying robustness is equivalent to finding maximum cliques in $K - partite$ graph. The max-clique enumeration problem on a multi-partite graph with bounded boxicity gives lower bound robustness verification. The number of features is considered equal to the boxicity of the graph. The authors state that by level hierarchical enumeration of max-cliques, it is possible to find the lower bound verification of tree ensemble faster and also for input which has low feature dimension verification is in order of polynomial time using max-clique algorithm.

A domain agnostic algorithm framework based on an abstract interpretation that uses hyperrectangles of real intervals for formal verification of robustness of decision tree ensembles is introduced by [144]. The algorithm approximates functions in decision tree growing and can produce complete robustness verification against adversarial perturbations under certain conditions. Another notable work related to this to formal verification method for tree ensembles that leverage an abstraction-refinement approach is also introduced in [159].

The interpretability of decision trees comes as an advantage to verify the security of decision tree models against attacks. Representing an attacker and transforming decision trees into the imperative program [28] implement an abstract implementation framework to certify decision tree security guarantees against evasion attacks.

All the above mentioned works considered ℓ_∞ perturbation model, which

makes verification easier since the perturbation of features is independent of each other. In this regard, a generalized ℓ_p robustness verification method is introduced in [173]. While they prove complete verification is NP-complete, this work dynamic programming based algorithm for decision stumps and extend a previous work [36] to include ℓ_p norm robustness verification. In addition, this work includes ℓ_p norm certification for robust training algorithms for ensemble stumps and trees.

A general summary of robustness verification algorithms for decision tree, decision tree ensembles and stumps are presented in table 3.2.

ML Model	Paper	Verification	Bounded norm	Complexity
Single Tree	[36, 173]	Complete	ℓ_∞, ℓ_0	Linear
Tree Ensemble	[36]	Incomplete	ℓ_∞	Multi-level
	[90]	Complete	ℓ_∞	NP-complete
	[173]	Incomplete	$\ell_P, P \in (0.\infty)$	Extended Multi-level
	[144]	Complete	ℓ_∞	–
Stump Ensemble	[4]	Complete	ℓ_∞	Polynomial
	[173]	Incomplete	$\ell_P, P \in (0.\infty)$	Approximate Knapsack

Table 3.2: *Summary of robustness verification algorithms for decision tree, decision tree ensembles and stumps*

3.5.2 Evaluation

Evaluating a model for how well it resists attack has become an integral part of adversarial learning research. Evaluating robustness indicates how powerful an attack is to be successful and how sufficient enough the defence mechanism is integrated with the model. Most attacks and defence mechanisms related to decision trees and tree ensembles are based ℓ_p norm bounded perturbation [35,41,90]. Chen *et al.* indicated attacking strategies based on greedy search needs a larger perturbation of instances to attack a decision tree than other attacking methods. For instance, compared to [35] Papernot's method [130], generating adversarial samples need a larger ℓ_∞ norm for attacking decision trees. Kantchelian's attack, though it is not scalable, can find the minimum perturbation needed for generating adversarial samples, giving the true norm distortion value and robustness of the model.

To evaluate robustness is related to how large the perturbation has to be to attack a model; the larger the magnitude of distortion needed to attack, the more the model is robust. The average ℓ_∞ perturbation is used to show how strong a model is against an attack [35]. Since it is preferred to have adversarial samples with the smallest perturbation for the sake of not be-

ing detected, models which are attacked by adversarial samples with larger perturbation are more resistant to attacks. In relation to evaluation, other methods are also used in works of literature, Such as test error, robust test error, accuracy, macro F1, and ROC AUC on attacked data.

3.6 Datasets

Datasets are useful in evaluating and validating new methodologies since they can be utilized for experimentation. In this section we will list commonly used datasets that are used in most adversarial learning literature related to decision trees and tree ensembles. The main statistics of all the datasets are shown in Table 5.2.

- ***MNIST:*** The MNIST dataset comprising of 10-class handwritten digits, was first introduced by LeCun et al [100] is a benchmark dataset used for classification tasks in many research works.This is because of the reason that the dataset has a manageable size, allowing researchers to quickly check and prototype their algorithms. In addition, machine learning libraries (e.g. scikit-learn) and deep learning frameworks (e.g. Tensorflow, Pytorch) provide helper functions and convenient examples that use MNIST out of the box that makes it more familiar and easy to use. It is represented in binary; each feature represents a gray level on each pixel.
- ***MNIST 2-6:*** The binary mnist dataset [100] contains hand-written digits of "2" and "6". The attributes represent the gray levels on each pixel location. It is used for identifying between the two digits in classification tasks.
- ***FMNIST:*** This dataset is a representation of unique fashion products represented in 28×28 gray scale associated with a label from 10 classes which can be used as a replacement for MNIST dataset for machine learning algorithms [1].
- ***Breast-cancer:*** The breast cancer dataset [53] contains 2 classes of samples, each representing benign and malignant cells.The attributes represent different measurements of the cell's physical properties [2].
- ***Diabetes:*** The objective of the dataset [152] is to diagnostically predict whether or not a patient has diabetes based on certain diagnostic

[1] https://github.com/zalandoresearch/fashion-mnist
[2] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

measurements included in the dataset [3].

- ***cod-rna:*** The cod-rna dataset [165] contains 2 classes of samples representing sequenced genomes, categorized by the existence of non-coding RNAs. The attributes contain information onthe genomes, including total free-energy change, sequencelength, and nucleotide frequencies [4].
- ***Census Income:*** The objective of this dataset [53] is to determine whether a person makes over $50K a year based on given features, such as age, education, occupation, gender, race, etc. [5]
- ***Wine Quality:*** Predicting quality of wine given 11 physiochemical attributes and designed as multi-class classification problem. This task is turned into a binary classification in [31], where the positive class identifies good-quality wines (i.e., those whose quality is at least 6, on a 0-10 scale) and the negative class contains the remaining instance [6]
- ***Credit Cards:*** Credit cards dataset [179] aimed at the case of customers' default payments as a response variable using 23 variables. Default Payment as a response attribute (Yes = 1, No = 0) and other attributes include amount of given credit, gender, education, marital status, age, and etc. [7]

3.7 Summary

In this chapter, we presented a thorough overview of works related to adversarial machine learning focusing on decision trees and tree ensembles with the aim of providing a systematic survey on attack and defensive mechanisms for tree-based models. Specifically, we have revisited existing works of attacks from white-box and black-box strategy approaches and also reviewed proposed defences considering adversarial training and robust optimization as a broad categorization for proposed methods. In addition, we include methodologies used for robust verification and evaluation.

Finally, we observe that even though more works are being done recently on adversarial machine learning focusing on tree-based models, there needs to be much focus on this area since tree-based models are utilized in real-world system more often.

[3] https://www.kaggle.com/uciml/pima-indians-diabetes-database
[4] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html#cod-rna.
[5] https://archive.ics.uci.edu/ml/datasets/census+income
[6] https://www.kaggle.com/c/uci-wine-quality-dataset/data
[7] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

CHAPTER 4

Unfairness Mitigation Algorithms for tree-based models

Artificial intelligence and machine learning (ML) algorithms control an increasing number of choices affecting people's everyday lives, ranging from the criminal justice system, healthcare, transportation, and education to college admissions, recruiting, loan provision, and many others. Therefore, it is critical to create machine learning algorithms that are not just accurate but also objective and fair. Unfortunately, recent research suggests that algorithmic decision-making is intrinsically prone to bias, even when the intention is not to be such. In this chapter, we present a recent algorithmic solutions for fairness focusing on tree-based models.

4.1 Introduction

In recent years automation of the decision-making process has become more and more common in many application areas. In this decision process, machine learning models are used increasingly. However, despite delivering superior performance, those models are difficult to interpret since most are black boxes. Hence, it causes concern regarding the consequences

on people, potentially producing discrimination against a group or individual. Moreover, algorithms trained on biased data, in particular, are prone to learning, perpetuating, or even reinforcing these discrimination [19]. In this context, many organizations and governments introduced rules and legalization frameworks; for example, the EU introduced the General Data Protection Regulation (GDPR) security and privacy law in 2018, regulating the use and collection of sensitive personal data.

In addition to introducing laws and regulations combating bias, many fairness methodologies have been proposed. One of the methods is removing the sensitive attribute from the dataset, which is called "fairness through unawareness" [135], which does not solve algorithmic discrimination coming from datasets that have a complex correlation with sensitive attribute. Hence, bias mitigation strategies are introduced for machine learning. Recently, algorithmic solutions for decision trees and tree ensembles to improve their fairness have been proposed.

In this chapter, building on the foundations and main concepts of *fair machine learning* we presented in section 2.4, we summarize state-of-the-art works targeting tree-based models on fairness.

4.2 Fairness-enhancing Mechanisms for Tree-based Models

There are three different unfairness mitigation ways in machine learning described in section 2.4.3: data pre-processing, optimization approach during in-processing , and post-processing results of the algorithm. The following three sections review studies in each one of these categories targeting decision trees and tree ensembles. The below discussed works are summarized in Table 4.2.

4.2.1 Pre-processing Mechanisms

As mentioned in section 2.4.3 pre-processing approach involves modifying the training data before it is fed into an ML algorithm. Mechanisms in this category involve modifying the training data before it is fed into an ML algorithm. Preliminary mechanisms, such as Kamiran and Calders [86, 87] proposed changing or reweighing the labels of some instances before training to improve classification fairness. Changed labels are typically associated with samples closer to the decision boundary, as they are more likely to be discriminated. Recent mechanisms propose modifying feature representations. One of the methods [58] changes the dataset's features to make the distributions for both privileged and unprivileged groups become similar, making it more difficult for the algorithm to distinguish between the

two groups(defined by sensitive attribute), while in [27] it transforms the data to a new mapping that improves group fairness and individual fairness.

4.2.2 In-processing Mechanisms

The mechanisms in this category modifies the algorithm to account for fairness [2, 57, 67, 88, 141, 143, 184].

For example, Kamiran *et al.* [88] included a discrimination factor into the information gain splitting criterion of a single decision tree classifier by considering the split of a node under the influence of a sensitive feature, i.e., before a node split happens, not only the usual purity w.r.t. to the target label is calculated, but also the purity of the split w.r.t. the sensitive feature. Three alternative splitting criteria are given based on the way discrimination is accounted. The first option is subtracting discrimination gain from accuracy gain, which allows for a split if it is non-discriminatory; the second option is an accuracy-discrimination trade-off split where the accuracy gain is divided by discrimination gain to have the final gain value. The third option is adding the accuracy and discrimination gain to decide the best feature to split a node. The authors claim the additive information gain criterion with a post-process relabeling approach produces lower discrimination. [184] uses the same concept of adjusting split criterion of a decision tree to maximize information gain between the splitting attribute and the class label for online stream based decision-making. With closest to [88], we implement this method in Chapter 6 for the base trees of our forest and evaluate its impact on the overall forest discrimination value.

Aghaei *et al.* [2] proposed a framework for training fair decision trees by adding regularization terms to the Mixed-Integer Programming to penalize discrimination. The approach mitigates unfairness with the high computational cost. [141] put forward regularization-based approach to train fair decision tree and fair random forest.

Fantin *et al.* [57] exploits randomly generated decision trees and filter them by their fairness before adding them to the forest. This is achieved through a hyper-parameter fairness constraint, which forces to accept only decision trees with statistical parity below the given threshold. Each tree's generation and fairness thresholding can be done in a distributed framework that optimizes the trade of between discrimination and accuracy before being added to the forest. Furthermore, this algorithm uses randomness constraints to train base trees in which one feature is randomly selected to split a node for building a randomized decision tree.

Grari *et al.* [67] proposed the adversarial debiasing for decision trees.

They use gradient boosting the output while minimizing the ability of adversarial neural network predicting the sensitive feature.

4.2.3 Post-processing Mechanisms

Post-processing mitigation approaches focus on adjusting the final output of the trained model rather than the underline loss function or training data. The algorithms discussed in [70, 138] aim at achieving same error rates between privileged and unprivileged groups, [70] uses equalized odd and equalized opportunity to promote features which are more dependent on the target label than the sensitive attribute. While in [138] the proposed algorithm aims to achieve both privileged and unprivileged groups to have the same false negative rate and false positive rate by taking into account a calibrated probability estimates. Both proposed techniques flips some decisions of a classifier to enhance equalized odds or equalized opportunity. Another post-processing algorithm called Reject Option based Classification (ROC) [89] takes in to consideration the decision boundary of classifiers; in a region where uncertainty is high, it gives favorable outcomes to the unprivileged group and unfavorable outcomes to the privileged group to reduce discrimination. In [47] minimize discrimination by selecting different thresholds for each group in a manner which maximizes accuracy.

Dwork *et al.* [55] learn different classifier for each group by using decoupling method. Lohia*et al.* [105] used individual and group debiasing method to balance accuracy with both individual and group fairness by detecting samples that are prone to individual bias and consider them for prediction change for enhancing the disparate impact.

4.2.4 Hybrid Mechanisms

Methods mentioned in the previous sections might not be enough to achieve the desired fairness level. Hence, a hybrid mechanism combining mitigation strategies can be used. Foe example, [88] uses in-processing to optimize node splitting criterion and post-processing for changing the class label of some leaves in a decision tree classifier.

Table 4.1: *A summary of pre-process, in-process, and post-process mechanisms targeting tree based models*

Mitigation Strategy	Paper	Description
Pre-processing	[86, 87]	uses methods like massaging, suppression, sampling, and reweighing (section 2.4.3 for explanation)
	[58]	Pre-process data to decrease the distance between distributions of both groups.
	[27]	transforms the data to a new mapping that improves group fairness and individual fairness
In-processing	[88, 184]	Discrimination aware node splitting
	[2, 141]	Regularization based approach
	[143]	Use abstract interpretation and adversarial training to train fair decision trees
	[67]	adversarial debiasing
	[57]	filter base trees of a forest using fairness parameter to add only fair trees
Post-processing	[47, 70, 138]	selecting different decision thresholds for different groups to enhance equalized odds and equalized opportunity
	[89]	Reject Option based Classification (ROC),takes in to consideration the decision boundary of classifiers
	[55]	decoupling technique to learn a different classifier for each group
	[105]	detecting samples that are prone to individual bias and consider them for prediction change

Table 4.2: *A summary of pre-process, in-process, and post-process mechanisms targeting tree based models*

4.3 Summary

In this chapter, we presented an overview of works related to fair machine learning focusing on decision trees and tree ensembles with the aim of providing systematically summarized survey. Existing works on mitigation strategies are presented grouped as pre, in, and post-processing approaches.

CHAPTER 5

Treant: Training Evasion-Aware Decision Trees

Despite its success and popularity, machine learning is now recognized as vulnerable to *evasion attacks*, i.e., carefully crafted perturbations of test inputs designed to force prediction errors. In this paper we focus on evasion attacks against decision tree ensembles, which are among the most successful predictive models for dealing with non-perceptual problems. Even though they are powerful and interpretable, decision tree ensembles have received only limited attention by the security and machine learning communities so far, leading to a sub-optimal state of the art for adversarial learning techniques. Thus, in this chapter we propose TREANT, a novel decision tree learning algorithm that, on the basis of a formal threat model, minimizes an evasion-aware loss function at each step of the tree construction. TREANT is based on two key technical ingredients: *robust splitting* and *attack invariance*, which jointly guarantee the soundness of the learning process. Experimental results on publicly available datasets show that TREANT is able to generate decision tree ensembles that are at the same time accurate and nearly insensitive to evasion attacks, outperforming state-of-the-art adversarial learning techniques.

5.1 Introduction

To date, research on evasion attacks has mostly focused on linear classifiers [15, 107] and, more recently, on deep neural networks [66, 156]. Whereas deep learning obtained remarkable and revolutionary results on many perceptual problems, such as those related to computer vision and natural language understanding, *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 [43]. Decision trees are also considered *interpretable* compared to other models [158], yielding predictions which are human-understandable in terms of syntactic checks over domain features, which is particularly appealing in the security setting. Unfortunately, despite their success, decision tree ensembles have received only limited attention by the security and machine learning communities so far, leading to a sub-optimal state of the art for adversarial learning techniques (see Section 5.2).

In this chapter, we thus propose TREANT,[1] a novel learning algorithm designed to build decision trees which are resilient against evasion attacks at test time. Based on a formal threat model, TREANT optimizes an evasion-aware loss function at each step of the tree construction [112]. This is particularly challenging to enforce correctly, considered the greedy nature of traditional decision tree learning [80]. In particular, TREANT has to ensure that the local greedy choices performed upon tree construction are not short-sighted with respect to the capabilities of the attacker, who has the advantage of choosing the best attack strategy based on the fully built tree. TREANT is based on the combination of two key technical ingredients: a *robust splitting* strategy for decision tree nodes, which reliably takes into account at training time the attacker's capability of perturbing instances at test time, and an *attack invariance* property, which preserves the correctness of the greedy construction by generating and propagating constraints along the decision tree, so as to discard splitting choices which might be vulnerable to attacks.

We finally deploy our learning algorithm within a traditional random forest framework [20] and show its predictive power on real-world datasets. Notice that, although there have been various proposals that tried to improve robustness against evasion attacks by using ensemble methods [13, 73,137,162], it was shown that ensembles of weak models are not necessarily strong [72]. We avoid this shortcoming by employing TREANT to train

[1]The name comes from the role playing game "Dungeons & Dragons", where it identifies giant tree-like creatures.

an ensemble of decision trees which are individually resilient to evasion attempts.

5.1.1 Roadmap

To show how TREANT improves over the state of the art, we proceed as follows:

1. We introduce our formal threat model, discussing an exhaustive white-box attack generation method, which allows for an accurate evaluation of the performance of decision trees under attack and proves scalable enough for our experimental analysis (Section 5.3).

2. We present TREANT, the first tree learning algorithm which greedily, yet soundly, minimizes an evasion-aware loss function upon tree construction (Section 5.4).

3. We experimentally show that TREANT outperforms existing adversarial learning techniques on four publicly available datasets (Section 5.5).

Our analysis shows that TREANT is able to build decision tree ensembles that are at the same time accurate and nearly insensitive to evasion attacks, providing a significant improvement over the state of the art.

5.2 Related Work

Adversarial learning, which investigates the safe adoption of ML in adversarial settings [79], is a research field that has been consistently increasing of importance in the last few years. In this paper we deal with *evasion attacks*, a research sub-field of adversarial learning, where deployed ML models are targeted by attackers who craft adversarial examples that resemble normal data instances, but force wrong predictions. Most of the work in this field 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 [11, 14, 32, 50, 66, 90, 125, 153, 183]. Contrary to robust machine learning [164] where some form of probabilistic random noise is assumed, in the adversarial setting even a single perturbation which is able to fool the classifier is assumed to be adopted by the attacker with 100% probability. To prevent evasion attacks, different techniques have been proposed for

different models, including support vector machines [15, 176], deep neural networks [66, 68, 133], and decision tree ensembles [35, 90]. Unfortunately, the state of the art for decision tree ensembles is far from satisfactory.

The first adversarial learning technique for decision tree ensembles is due to [90] and is called *adversarial boosting*. It is an empirical data augmentation technique, borrowing from the *adversarial training* approach [156], where a number of evading instances are included among the training data to make the learned model aware of the attacks and, thereby, possibly more resilient to them. Specifically, at each boosting round, the training set is extended by crafting a set of possible perturbations for each original instance and by picking the one with the smallest margin, i.e., the largest misprediction risk, for the model trained so far. Adding perturbed instances to the training set forces the learning algorithm to minimize the *average* error over both the original instances and the chosen sample of evading ones, but this does not provide clear performance guarantees under attack. This is both because evading instances exploited at training time might not be representative of test-time attacks, and because optimizing the average case might not defend against the *worst-case* attack. Indeed, the experiments in Section 5.5 show that the performance of ensembles trained via adversarial boosting can be severely downgraded by evasion attacks.

The second adversarial learning technique for decision tree ensembles was proposed in a very recent work by Chen *et al.*, who introduced the first tree learning algorithm embedding the attacker directly in the optimization problem solved upon tree construction [35]. The key idea of their approach, called *robust trees*, is to redefine the splitting strategy of the training examples at a tree node. They first identify the so-called *unknown* instances of $\mathcal{D}$, which may fall in either in $\mathcal{D}_l$ or in $\mathcal{D}_r$, depending on adversarial perturbations. The authors thus claim that the optimal tree construction strategy would need to account for an exponential number of attack configurations over these unknown instances. To tame such algorithmic complexity, they propose a sub-optimal heuristic approach based on four "representative" attack cases. Though the key idea of this algorithm is certainly interesting and shares some similarities with our own proposal, it also suffers from significant shortcomings. First, representative attack cases are not such anymore when the attacker is aware of the defense mechanism, and they are not anyway sufficient to subsume the spectrum of possible attacks: our algorithm takes into account all the possible attack cases, while being efficient enough for practical adoption. Moreover, the approach in [35] does not implement safeguards against the incremental greedy nature of decision tree learning: there is no guarantee that, once the best splitting has been identified, the

attacker cannot adapt his strategy to achieve better results on the full tree. Indeed, the experimental evaluation in Section 5.5 shows that it is very easy to evade the trained models, which turn out to be even more fragile than those trained through adversarial boosting in some cases.

5.3 Threat Model

The possibility to craft adversarial examples was popularized by [156] in the image classification domain: their seminal work showed that it is possible to introduce minimal perturbations into an image so as to modify the prediction of its class by a deep neural network.

5.3.1 Loss Under Attack and Adversarial Learning

At an abstract level, we can see the attacker A as a function mapping each instance to a set of possible perturbations, which might be able to evade the ML model. Depending on the specific application scenario, not every attack is plausible, e.g., A cannot force some perturbations or behaves surreptitiously to avoid detection. For instance, in the typical image classification scenario, A is usually assumed to introduce just slight modifications that are perceptually undetectable to humans. This simple similarity constraint between the original instance $\boldsymbol{x}$ and its perturbed variant $\boldsymbol{z}$ is well captured by a distance [66], e.g., one could have $A(\boldsymbol{x}) = \{\boldsymbol{z} \mid \|\boldsymbol{z} - \boldsymbol{x}\|_\infty \leq \varepsilon\}$.

Similarly, assuming that the attacker can run independent attacks on every instance of a given dataset $\mathcal{D}$, we can define $A(\mathcal{D})$ as the set of the datasets $\mathcal{D}'$ obtained by replacing each $(\boldsymbol{x}, y) \in \mathcal{D}$ with any $(\boldsymbol{z}, y)$ such that $\boldsymbol{z} \in A(\boldsymbol{x})$.

The hardness of crafting successful evasion attacks defines the *robustness* of a given ML model at test time. The goal of learning a robust model is therefore to minimize the harm an attacker may cause via perturbations. This learning goal was formalized as a min-max problem by [112]:

$$\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}} \quad \underbrace{\max_{\mathcal{D}' \in A(\mathcal{D})} \mathcal{L}(h, \mathcal{D}')}_{\mathcal{L}^A(h, \mathcal{D})} . \tag{5.1}$$

The inner maximization problem models the attacker A replacing all the given instances with an adversarial example aimed at maximizing the loss. We call *loss under attack*, noted $\mathcal{L}^A(h, \mathcal{D})$, the solution to the inner maximization problem. The outer minimization resorts to the empirical risk minimization principle, aiming to find the hypothesis that minimizes the loss under attack on the training set.

5.3.2 Attacker Model

Distance-based constraints for defining the attacker's capabilities are very flexible for perceptual problems and proved amenable for heuristic algorithms for solving the inner maximization problem of Equation 5.1 [112]. However, they cannot be easily generalized to other realistic application scenarios, e.g., where perturbations are not symmetric, where the attacker may not be able to alter some of the features, or where categorical attributes are present. To overcome such limitations, we model the attacker A as a pair (R, K), where R is a set of *rewriting rules*, defining how instances can be corrupted, and $K \in \mathbb{R}^+$ is a *budget*, limiting the amount of alteration the attacker can apply to each instance. Each rule $r \in R$ has form:

$$[a, b] \xrightarrow{f}_k [\delta_l, \delta_u],$$

where $[a, b]$ and $[\delta_l, \delta_u]$ are intervals on $\mathbb{R} \cup \{-\infty, +\infty\}$, with the former defining the *precondition* for the application of the rule and the latter defining the *magnitude* of the perturbation enabled by the rule; $f \in [1, d]$ is the index of the feature to corrupt; and $k \in \mathbb{R}^+$ is the *cost* of the rule. The semantics of the rewriting rule can be explained as follows: if an instance $\boldsymbol{x}$ satisfies the condition $x_f \in [a, b]$, then the attacker can corrupt it by adding any $v \in [\delta_l, \delta_u]$ to x_f and spending k from the available budget. Note that v can possibly be negative, leading to a subtraction. The attacker can corrupt each instance by using as many rewriting rules as desired in whatever order, up to budget exhaustion.

According to this attacker model, we define $A(\boldsymbol{x})$, the set of the attacks against an instance $\boldsymbol{x}$, as follows.

Definition 5.3.1 (Attacks)**.** *Given an instance $\boldsymbol{x}$ and an attacker $A = (R, K)$, we let $A(\boldsymbol{x})$ be the set of the* attacks *that can be obtained from $\boldsymbol{x}$, i.e., the set of the instances $\boldsymbol{z}$ such that there exists a sequence of rewriting rules $r_1, \ldots, r_n \in R$ and a sequence of instances $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n$ where:*

1. *$\boldsymbol{x}_0 = \boldsymbol{x}$ and $\boldsymbol{x}_n = \boldsymbol{z}$;*
2. *for all $i \in [1, n]$, the instance $\boldsymbol{x}_{i-1}$ can be corrupted into the instance $\boldsymbol{x}_i$ by using the rewriting rule r_i;*
3. *the sum of the costs of $r_1, \ldots, r_n$ is not greater than K.*

Notice that $\boldsymbol{x} \in A(\boldsymbol{x})$ for any A by picking an empty sequence of rewriting rules.

We highlight that this rule-based attacker model includes novel attack capabilities like asymmetric perturbations, easily generalizes to categorical variables, and still covers or approximates standard distanced-based models. For instance, L_0-distance attacker models where the attacker can corrupt at will a limited number of features can be easily represented [90]. The use of a budget is convenient to fine-tune the power of the attacker and enables the adoption of standard evaluation techniques for ML models under attack, like *security evaluation curves* [18].

Example 5.3.1 (L_0-Distance). The L_0-distance captures localized perturbations with arbitrary magnitude. Specifically, given an instance $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a possible perturbation $\boldsymbol{z}$, we have that $\|\boldsymbol{z} - \boldsymbol{x}\|_0 = |\{f \in [1, d] \mid z_f \neq x_f\}|$, and thus the L_0-distance simply counts the dimensions of $\boldsymbol{x}$ that were actually perturbed.

In our framework, we can model this by means of an attacker $A = (R, K)$, where the budget K stands for the largest L_0-distance allowed on adversarial perturbations and R includes, for all features f, a rewriting rule of the form:

$$[-\infty, +\infty] \xrightarrow{f}_1 [-\infty, +\infty].$$

It is easy to show that for all $\boldsymbol{z}$ we have $\boldsymbol{z} \in A(\boldsymbol{x})$ if and only if $\|\boldsymbol{z} - \boldsymbol{x}\|_0 \leq K$. In particular, the largest perturbation is obtained from the original $\boldsymbol{x}$ by applying exactly K distinct rules, each perturbing a different dimension.

Example 5.3.2 (L_1-Distance). The L_1-distance also captures localized perturbations, but constrains their magnitude. Specifically, given an instance $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a possible perturbation $\boldsymbol{z}$, we have that $\|\boldsymbol{z} - \boldsymbol{x}\|_1 = \sum_f |z_f - x_f|$.

In our framework, we can model this by means of an attacker $A = (R, K)$, where the budget K stands for the largest L_1-distance allowed on adversarial perturbations and R includes, for all features f, a rewriting rule of the form:

$$[-\infty, +\infty] \xrightarrow{f}_\varepsilon [-\varepsilon, +\varepsilon],$$

where $\varepsilon \in \mathbb{R}^+$ models a maximum *discrete step* of perturbation (and its cost).

It is easy to show that the set $A(\boldsymbol{x})$ can approximate $\{\boldsymbol{z} \mid \|\boldsymbol{z} - \boldsymbol{x}\|_1 \leq K\}$ with arbitrarily large accuracy by choosing appropriately small values of ε. Note that the largest perturbation is obtained from the original $\boldsymbol{x}$ by applying exactly $\lfloor K/\varepsilon \rfloor$ rules, always choosing the maximum or minimum magnitude $\pm\varepsilon$.

Example 5.3.3 (L_∞-Distance). The L_∞-distance encourages uniformly spread perturbations with small magnitude. Specifically, given an instance $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a possible perturbation $\boldsymbol{z}$, we have that $\|\boldsymbol{z}-\boldsymbol{x}\|_\infty = \max_f |z_f - x_f|$.

We observe that this form of non-localized perturbations is not currently supported by our threat model, since, once a rewriting rule is defined for a given feature, it can always be (locally) applied up to budget exhaustion. However, a straightforward solution to this issue would be to transform our current global budget into a set of *per-feature* budgets $\{K_1, \ldots, K_d\}$. Then, if K is the largest L_∞-distance allowed on adversarial perturbations, one could let $K_i = K$ for all i and just reuse the rewriting rules defined for the case of the L_1-distance. We do not implement this extension of the model for the sake of simplicity.

5.3.3 Attack Generation

Computing the loss under attack $\mathcal{L}^A$ is useful to evaluate the resilience of ML models to evasion attacks at test time; yet this might be intractable, since it assumes the ability to identify the most effective attack for all the test instances. This issue is thus typically dealt with by using a heuristic attack generation algorithm, e.g., the fast gradient sign method [66] or any of its variants, to craft adversarial examples which empirically work well. However, our focus on decision trees and the adoption of a rule-based attacker model enables an exhaustive attack generation strategy for the test set which, though computationally expensive, proves scalable enough for our experimental analysis and allows the actual identification of the most effective attacks. This enables the most accurate security assessment in terms of the actual value of $\mathcal{L}^A$.

We consider a *white-box* attacker model, where the attacker has the complete knowledge of the trained decision tree ensemble. We thus assume that the attacker exploits the knowledge of the structure of the trees in the targeted ensemble and, most importantly, of the features and thresholds which are actually used in the prediction process.

Note that a decision tree ensemble induces a finite partition of the input vector space $\mathcal{X}$, defined by the features and thresholds used in the internal nodes of the trees in the ensemble, where instances falling in the same element of the partition share the same prediction. This partition of the vector space makes it possible to significantly reduce the set of attacks that are relevant to compute $\mathcal{L}^A$ by considering at most one *representative* attack for each element of the partition [30]. Once this is done, one can feed all

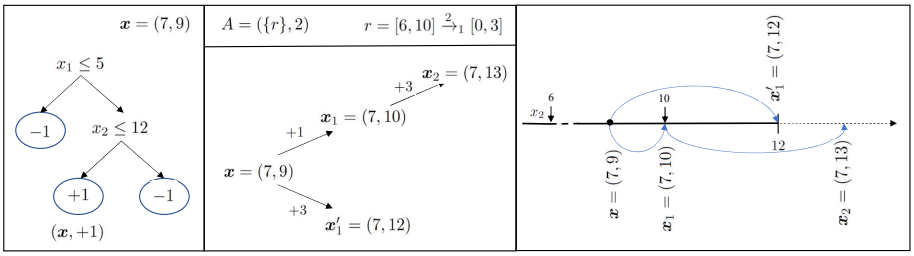

Figure 5.1: *A decision tree and an instance $\boldsymbol{x}$ that can be attacked by perturbing its feature 2 (by adding any value in interval $[0, 3]$). Note that since the maximum budget of A is 2, and the cost of applying the rule r is 1, the rule can be applied only twice provided that the precondition holds.*

the attack representatives to the tree ensemble and identify the one that maximizes the loss.

For the sake of simplicity, we just sketch the algorithm that generates the attack representatives. For any given instance $\boldsymbol{x}$, we first identify the set of applicable rules: if a rule targets the feature f with magnitude $[\delta_l, \delta_r]$, the interval of possible perturbations $[x_f + \delta_l, x_f + \delta_u]$ is split into the sub-intervals induced by (i) the ensemble's thresholds relative to feature f, since this might change the prediction of the tree ensemble, and (ii) the extremes of the pre-conditions of a rewriting rule operating on feature f, since this might enable further perturbations which will eventually lead to prediction changes. We then generate a single attack for each of the identified sub-intervals by applying maximal perturbations therein and recursively apply the algorithm up to budget exhaustion. Finally, we return just the attacks which actually crossed some threshold of the tree ensemble, since only those could lead to changes in predictions.

Example 5.3.4 (Attack Generation). Consider the instance $\boldsymbol{x} = (7, 9)$ with label $+1$ and the decision tree in Figure 5.1, which classifies the instance correctly. Pick then the attacker $A = (\{r\}, 2)$, where r is a rewriting rule of cost 1 which allows the corruption of the feature 2 by adding any value in $[0, 3]$, provided that the feature value is in the interval $[6, 10]$. In our formalism, this is represented as follows:

$$r = [6, 10] \xrightarrow{2}_1 [0, 3].$$

Only three values of the feature 2 are relevant in our setting to generate representative attacks: besides 12 (the threshold used by the decision tree), which actually partitions the second dimension into the intervals $(-\infty, 12]$ and $(12, +\infty)$, also 6 (the lower bound of the pre-condition of rule r) and 10 (the upper bound of the pre-condition of rule r). We include these bounds because rule r might be applied again, as long as the perturbations fall within the interval $[6, 10]$, and this might be useful to eventually cross a threshold of the decision tree.

Our algorithm thus applies r multiple times to perturb the second feature of $\boldsymbol{x} = (7, 9)$: in particular, the value 9 can initially be perturbed into any value from $[9, 12]$ after an application of the rule. Perturbations in this range can only cross one of the previously identified thresholds, i.e., 10. This induces a partitioning of $[9, 12]$ in the sub-intervals $[9, 10]$ and $(10, 12]$. The first attack $\boldsymbol{x}_1 = (7, 10)$ does not lead to a change of the prediction outcome of the decision tree, yet moved towards the decision threshold and can still be corrupted by rule r. The alternative attack $\boldsymbol{x}'_1 = (7, 12)$ also does not lead to any prediction change and cannot be corrupted any further due to the pre-condition of rule r. However, the attacker can target the second feature of $\boldsymbol{x}_1 = (7, 10)$ to corrupt it into any value from $[10, 13]$. Perturbations in this range can cross the decision threshold 12, inducing the sub-intervals $[10, 12]$ and $(12, 13]$. In particular, the attack $\boldsymbol{x}_2 = (7, 13)$ is generated by the algorithm and it is the only returned attack, since it is representative of all the attacks causing the instance $\boldsymbol{x}$ to fall into the partition $(12, +\infty)$ of the second dimension of the feature space.

5.4 TREANT: Key Ideas & Design

In this section, we present a novel decision tree learning algorithm that, by minimizing the loss under attack $\mathcal{L}^A$ at training time, enforces resilience to evasion attacks at test time. We call TREANT the proposed algorithm.

5.4.1 Overview

Compared to Algorithm 1, TREANT replaces the BESTSPLIT function by revising: (i) the computation of the predictions on the new leaves, (ii) the selection of the best split and (iii) the dataset partition along the recursion.

Before discussing the technical details, we build on the toy example in Figure 5.2 to illustrate the non-trivial issues arising when optimizing $\mathcal{L}^A$. Figure 5.2.(a) shows a dataset $\mathcal{D}$ for which we assume the attacker $A = (\{r\}, 1)$, where r is a rewriting rule of cost 1 which allows the corruption

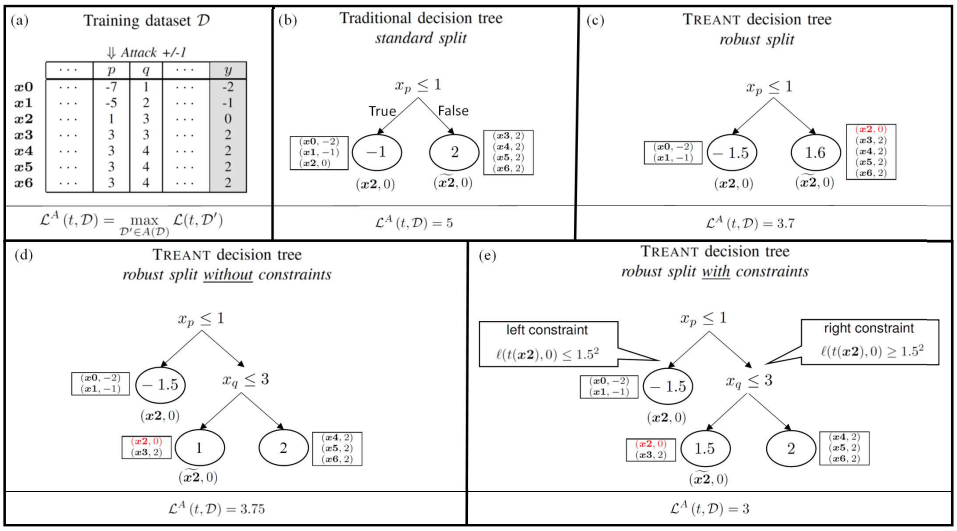

Figure 5.2: *Overview of the* TREANT *construction and its key challenges.*

of the feature p by adding any value in the interval $[-1, +1]$.

Assuming SSE is used as the underlying loss function $\mathcal{L}$, the decision stump initially generated by Algorithm 1 is shown in Figure 5.2.(b) along with the result of the splitting. Note that while the loss $\mathcal{L} = 2$ is small,[2] the loss under attack $\mathcal{L}^A = 5$ is much larger.[3] This is because the attacker may alter $\boldsymbol{x2}$ into a perturbed instance $\widetilde{\boldsymbol{x2}}$ so as to reverse the outcome of the test $x_p \leq 1$, i.e., the original instance $\boldsymbol{x2}$ falls into the left leaf of the stump, but the perturbed instance $\widetilde{\boldsymbol{x2}}$ falls into the right leaf. The first issue of Algorithm 1 is thus that the estimated loss $\mathcal{L}$ on the training set, computed when building the decision stump, is smaller than the loss under attack $\mathcal{L}^A$ we would like to minimize. We solve this issue by designing a novel *robust splitting* strategy to identify the best split of $\mathcal{D}$, which directly minimizes $\mathcal{L}^A$ when computing the leaves predictions and leads to the generation of a tree that is more robust to attacks. In particular, the decision stump learnt by using our robust splitting strategy is shown in Figure 5.2.(c), where the leaves predictions have been found by assuming that $\boldsymbol{x2}$ actually falls into the right leaf (according to the best attack strategy). For this new decision stump, the best move for the attacker is still to corrupt $\boldsymbol{x2}$, but the resulting $\mathcal{L}^A = 3.7$ is much smaller than that of the previous stump.[4] The figure also

[2] $\mathcal{L}(t,\mathcal{D}) = (-2+1)^2 + (-1+1)^2 + (-1-0)^2 + 4 \cdot (2-2)^2 = 2$.
[3] $\mathcal{L}^A(t,\mathcal{D}) = (-2+1)^2 + (-1+1)^2 + (2-0)^2 + 4 \cdot (2-2)^2 = 5$.
[4] $\mathcal{L}^A(t,\mathcal{D}) = (-2+1.5)^2 + (-1+1.5)^2 + (0-1.6)^2 + 4 \cdot (2-1.6)^2 = 3.7$.

shows the outcome of the robust splitting.

However, a second significant issue arises when the decision stump is recursively grown into a full decision tree. Suppose to further split the right leaf of Figure 5.2.(c), therefore considering only the instances falling therein, including the instance $\boldsymbol{x2}$ put there by the robust splitting. We would find that the best split is given by $x_q \leq 3$, where the feature q cannot be modified by the attacker. The resulting tree is shown in Figure 5.2.(d). Note however that, by creating the new sub-tree, new attacking opportunities show up, because the attacker now finds more convenient to just leave $\boldsymbol{x2}$ unaltered and let it fall directly into the left child of the root. As a consequence, by adding the new sub-tree, we observe an increased loss under attack $\mathcal{L}^A = 3.75$.[5] This second issue can be solved by ensuring that any new sub-tree does not create new attacking opportunities that generate a larger loss. We call this property *attack invariance*. The proposed algorithm grows the sub-tree on the right leaf by carefully adjusting its predictions as shown in Figure 5.2.(e), still decreasing the loss under attack to $\mathcal{L}^A = 3$ with respect to the tree in Figure 5.2.(c).[6] This is enforced by including constraints along the tree construction, as shown in the figure.

To sum up, the key technical ingredients of TREANT are:

1. *Robust splitting*: given a candidate feature f and threshold v, the robust splitting strategy evaluates the quality of the corresponding node split on the basis of a *ternary* partitioning of the instances falling into the node. It identifies those instances for which the outcome of the node predicate $x_f \leq v$ depends on the attacker's moves, and those that cannot be affected by the attacker, thus always traversing the left or the right branch of the new node. In particular, the $\mathcal{L}^A$ minimization problem is reformulated on the basis of left, right and unknown instances, i.e., instances which might fall either left or right depending on the attacker. Finally, the recursion on the left and right child of the node is performed by separating the instances in a binary partition based on the effects of the most harmful attack (Section 5.4.2).

2. *Attack invariance*: a security property requiring that the addition of a new sub-tree does not allow the attacker to find better attack strategies that increase $\mathcal{L}^A$. Attack invariance is achieved by imposing an appropriate set of constraints upon node splitting. New constraints are generated for each of the attacked instances present in the split node and are propagated to the child nodes upon recursion (Section 5.4.3).

[5] $\mathcal{L}^A(t, \mathcal{D}) = (-2 + 1.5)^2 + (-1 + 1.5)^2 + (0 - 1.5)^2 + (2 - 1)^2 + 3 \cdot (2 - 2)^2 = 3.75$.
[6] $\mathcal{L}^A(t, \mathcal{D}) = (-2 + 1.5)^2 + (-1 + 1.5)^2 + (0 - 1.5)^2 + (2 - 1.5)^2 + 3 \cdot (2 - 2)^2 = 3$.

Table 5.1: *Notation Summary*

Symbol	Meaning
$\mathcal{D}$	Training dataset
$\mathcal{D}^\lambda$	Local projection of $\mathcal{D}$ on the leaf λ
$A(\boldsymbol{x})$	Set of all the attacks A can generate from $\boldsymbol{x}$
$A(\mathcal{D})$	Set of all the attacks A can generate from $\mathcal{D}$
$\lambda(\hat{y})$	Leaf node with prediction $\hat{y}$
$\sigma(f, v, t_l, t_r)$	Node testing $x_f \leq v$ and having sub-trees t_l, t_r
$\mathcal{D}_l(f, v, A)$	Left elements of ternary partitioning on (f, v)
$\mathcal{D}_r(f, v, A)$	Right elements of ternary partitioning on (f, v)
$\mathcal{D}_u(f, v, A)$	Unknown elements of ternary partitioning on (f, v)
$\mathcal{D}_L(\hat{t}, A)$	Left elements of robust splitting on $\hat{t}$
$\mathcal{D}_R(\hat{t}, A)$	Right elements of robust splitting on $\hat{t}$
$\mathcal{C}_L(\hat{t}, A)$	Set of constraints for the left child of $\hat{t}$
$\mathcal{C}_R(\hat{t}, A)$	Set of constraints for the right child of $\hat{t}$

The pseudo-code of the algorithm is given in Section 5.4.4. To assist the reader, the notation used in the present section is summarized in Table 6.1.

5.4.2 Robust Splitting

We present our novel *robust splitting* strategy that grows the current tree t by replacing a leaf λ with a new sub-tree so as to minimize the loss under attack $\mathcal{L}^A$. For the sake of clarity, we discuss it as if the splitting was employed on the root node of a new tree, i.e., to learn the decision stump that provides the best loss reduction on the full input dataset $\mathcal{D}$. The next subsection discusses the application of the proposed strategy during the recursive steps of the tree-growing process.

Aiming at greedily optimizing the *min-max* problem in Equation 5.1, we have to find the best decision stump $\hat{t} = \sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$ such that:

$$\begin{aligned}
\hat{t} &= \operatorname*{argmin}_t \mathcal{L}^A(t, \mathcal{D}) = \\
&= \operatorname*{argmin}_t \max_{\mathcal{D}' \in A(\mathcal{D})} \mathcal{L}(t, \mathcal{D}') = \\
&= \operatorname*{argmin}_t \sum_{(\boldsymbol{x}, y) \in \mathcal{D}} \max_{\boldsymbol{z} \in A(\boldsymbol{x})} \ell(t(\boldsymbol{z}), y).
\end{aligned}$$

However, the equation shows that this is not trivial, because the loss incurred by an instance $(\boldsymbol{x}, y)$ may depend on the attacks it is possibly subject to. Similarly to [35], we thus define a ternary partitioning of the training dataset as follows.

Definition 5.4.1 (Ternary Partitioning)**.** *For a feature f, a threshold v and an attacker A, the* ternary partitioning *of the dataset $\mathcal{D} = \mathcal{D}_l(f, v, A) \cup \mathcal{D}_r(f, v, A) \cup \mathcal{D}_u(f, v, A)$ is defined by:*

$$\mathcal{D}_l(f, v, A) = \{(\boldsymbol{x}, y) \in \mathcal{D} \mid \forall \boldsymbol{z} \in A(\boldsymbol{x}) : z_f \leq v\}$$
$$\mathcal{D}_r(f, v, A) = \{(\boldsymbol{x}, y) \in \mathcal{D} \mid \forall \boldsymbol{z} \in A(\boldsymbol{x}) : z_f > v\}$$
$$\mathcal{D}_u(f, v, A) = (\mathcal{D} \setminus \mathcal{D}_l(f, v, A)) \setminus \mathcal{D}_r(f, v, A).$$

In words, $\mathcal{D}_l(f, v, A)$ includes those instances $(\boldsymbol{x}, y)$ falling into the left branch regardless of the attack, hence the attacker has no gain in perturbing x_f. A symmetric reasoning applies to $\mathcal{D}_r(f, v, A)$, containing those instances which fall into the right branch for all the possible attacks. The instances that the attacker may actually want to target are those falling into $\mathcal{D}_u(f, v, A)$, thus aiming at the largest loss. By altering those instances, the attacker may force each $(\boldsymbol{x}, y) \in \mathcal{D}_u(f, v, A)$ to fall into the left branch with a loss of $\ell(\hat{y}_l, y)$, or into the right branch, with a loss of $\ell(\hat{y}_r, y)$.

Example 5.4.1 (Ternary Partitioning). The test node $x_p \leq 1$ and the attacker considered in Figure 5.2.(c) determine the following ternary partitioning of $\mathcal{D}$:

- $\mathcal{D}_l(p, 1, A) = \{(\boldsymbol{x0}, -2), (\boldsymbol{x1}, -1)\}$
- $\mathcal{D}_r(p, 1, A) = \{(\boldsymbol{x3}, 2), (\boldsymbol{x4}, 2), (\boldsymbol{x5}, 2), (\boldsymbol{x6}, 2)\}$
- $\mathcal{D}_u(p, 1, A) = \{(\boldsymbol{x2}, 0)\}$

In other words, the instance $\boldsymbol{x2}$ is the only instance for which the branch taken at test time is unknown, as it depends on the attacker A.

By construction, given (f, v), the loss $\mathcal{L}^A$ can be affected by the presence of the attacker A only for the instances in $\mathcal{D}_u(f, v, A)$, while for all the remaining instances it holds that $\mathcal{L}^A = \mathcal{L}$. Since the attacker may force each instance of $\mathcal{D}_u(f, v, A)$ to fall into either the left or the right branch, the authors of [35] acknowledge a combinatorial explosion in the computation of $\mathcal{L}^A$. Rather than evaluating all the possible configurations, they thus propose a heuristic approach evaluating four "representative" attack cases: *i)* no attack, *ii)* all the unknown instances are forced in the left child, *iii)* all the unknown instances are forced in the right child, and *iv)* all the unknown instances are swapped by the attacker, i.e., they are forced in the left/right child when they would normally fall in the right/left child. Then, the loss $\mathcal{L}$ is evaluated for these four split configurations and the maximum value is used to estimate $\mathcal{L}^A$, so as to find the best stump $\hat{t}$ to grow. Note that

$\mathcal{L}$ is computed as in a standard decision tree learning algorithm. Unfortunately, this heuristic strategy does not offer soundness guarantees, because the above four configurations leave potentially harmful attacks out of sight and do not induce an upper-bound of $\mathcal{L}^A$.

To avoid this soundness issue, while keeping the tree construction tractable, we pursue a *numerical optimization* as follows. For a given (f, v), we highlight that finding the best attack configuration and finding the best left/right leaves predictions $\hat{y}_l, \hat{y}_r$ are two inter-dependent problems, yet the strategy adopted in [35] is to first evaluate a few different attack configurations, and then find the leaves predictions. We instead solve these two problems simultaneously via a formulation of the min-max problem that, fixed (f, v), is expressed solely in terms of $\hat{y}_l, \hat{y}_r$:

$$(\hat{y}_l, \hat{y}_r) = \operatorname*{argmin}_{y_l, y_r} \mathcal{L}^A(\sigma(f, v, \lambda(y_l), \lambda(y_r)), \mathcal{D}), \tag{5.2}$$

where $\mathcal{L}^A$ is decomposed via the ternary partitioning as:

$$\begin{aligned}\mathcal{L}^A(\sigma(f, v, \lambda(y_l), \lambda(y_r)), \mathcal{D}) \quad &= \\ = \quad \mathcal{L}(\lambda(y_l), \mathcal{D}_l(f, v, A)) &+ \mathcal{L}(\lambda(y_r), \mathcal{D}_r(f, v, A)) + \\ + \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_u(f, v, A)} &\max\{\ell(y_l, y), \ell(y_r, y)\}.\end{aligned}$$

Observe that if the instance-level loss ℓ is convex, then $\mathcal{L}^A$ is also convex[7] and it can be efficiently optimized numerically. Convexity is indeed a property enjoyed by many loss functions such as SSE (for regression) and Log-Loss (for classification). This allows one to overcome the exploration of the exponential number of attack configurations, still finding the optimal solution (up to numerical approximation).

Given the best predictions $\hat{y}_l, \hat{y}_r$, we can finally produce a binary split of $\mathcal{D}$ (as in Algorithm 1). To do this, we split the instances by applying the best adversarial moves, i.e., by assuming that every $(\boldsymbol{x}, y) \in \mathcal{D}_u(f, v, A)$ is pushed into the left or right child so as to generate the largest loss. If the two children induce the same loss, then we assume the instance is not attacked.

Definition 5.4.2 (Robust Splitting)**.** *For a sub-tree to be grown* $\hat{t} = \sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$ *and an attacker* A*, the* robust split *of* $\mathcal{D} = \mathcal{D}_L(\hat{t}, A) \cup \mathcal{D}_R(\hat{t}, A)$ *is defined as follows:*

[7] The pointwise maximum and the sum of convex functions preserve convexity.

- $\mathcal{D}_L(\hat{t}, A)$ *contains all the instances of* $\mathcal{D}_l(f, v, A)$ *and* $\mathcal{D}_R(\hat{t}, A)$ *contains all the instances of* $\mathcal{D}_r(f, v, A)$*;*
- *for each* $(\boldsymbol{x}, y) \in \mathcal{D}_u(f, v, A)$*, the following rules apply:*
  - *if* $\ell(\hat{y}_l, y) > \ell(\hat{y}_r, y)$*, then* $(\boldsymbol{x}, y)$ *goes to* $\mathcal{D}_L(\hat{t}, A)$*;*
  - *if* $\ell(\hat{y}_l, y) < \ell(\hat{y}_r, y)$*, then* $(\boldsymbol{x}, y)$ *goes to* $\mathcal{D}_R(\hat{t}, A)$*;*
  - *if* $\ell(\hat{y}_l, y) = \ell(\hat{y}_r, y)$*, then* $(\boldsymbol{x}, y)$ *goes to* $\mathcal{D}_L(\hat{t}, A)$ *if* $x_f \leq v$ *and to* $\mathcal{D}_R(\hat{t}, A)$ *otherwise.*

Example 5.4.2 (Robust Splitting). Once identified $\hat{y}_l$ and $\hat{y}_r$ for the decision stump $\hat{t} = (p, 1, \lambda(-1.5), \lambda(1.6))$ in Figure 5.2.(c), the datasets obtained for the leaves by robust splitting are:

- $\mathcal{D}_L(\hat{t}, A) = \{(\boldsymbol{x}\mathbf{0}, -2), (\boldsymbol{x}\mathbf{1}, -1)\}$
- $\mathcal{D}_R(\hat{t}, A) = \{(\boldsymbol{x}\mathbf{2}, 0), (\boldsymbol{x}\mathbf{3}, 2), (\boldsymbol{x}\mathbf{4}, 2), (\boldsymbol{x}\mathbf{5}, 2), (\boldsymbol{x}\mathbf{6}, 2)\}$

Notice that, unlike a standard decision tree learning algorithm, the right partition contains the instance $\boldsymbol{x}\mathbf{2}$ due to the presence of the attacker, even though such instance normally satisfies the root node test.

To summarize, the ternary partitioning allows $\mathcal{L}^A$ to be optimized for a given (f, v) and dataset $\mathcal{D}$, hence it can be used to find the best tree-growing step by an exhaustive search over f and v. Once this is done, the robust splitting allows the dataset $\mathcal{D}$ to be partitioned in order to feed the algorithm recursion on the left and right children of the newly created sub-tree. Ultimately, the goal of the proposed construction is solving the min-max problem of Equation 5.1 for a single tree-growing step and pushing the attacked instances into the partition induced by the most harmful attack.

5.4.3 Attack Invariance

The optimization strategy described in Section 5.4.2 needs some additional refinement to provide a sound optimization of $\mathcal{L}^A$ on the full dataset $\mathcal{D}$. When growing a new sub-tree at a leaf λ, we denote with $\mathcal{D}^\lambda$ the *local projection* of the full dataset at λ, i.e., the subset of the instances in $\mathcal{D}$ falling in λ along the tree construction by applying the robust splitting strategy. The key observation now is that the robust splitting operates by assuming that the attacker behaves *greedily*, i.e., by locally maximizing the generated loss, but as new nodes are added to the tree, new attack opportunities arise and different traversal paths towards different leaves may become more fruitful to the attacker. If this is the case, the robust splitting becomes unrepresentative of the possible attacker's moves and any learning decision made on

the basis of such splitting turns out to be unsound, i.e., with no guarantee of minimizing $\mathcal{L}^A$. Notice that this is a major design problem of the algorithm proposed in [35], and experimental evidence shows how the attacker can easily craft adversarial examples in some cases (see Section 5.5).

In the end, the computation of the best split for a given leaf λ cannot be done just based on the local projection $\mathcal{D}^\lambda$, unless additional guarantees are provided. We thus enforce a security property called *attack invariance*, which ensures that the tree construction steps preserve the correctness of the greedy assumptions made on the attacker's behavior. Given a decision tree t and an instance $(\boldsymbol{x}, y) \in \mathcal{D}$, we let $\Lambda^A(t, (\boldsymbol{x}, y))$ stand for the set of leaves of t which are reachable by some attack $\boldsymbol{z} \in A(\boldsymbol{x})$ that generates the largest loss among $A(\boldsymbol{x})$.

Attack invariance requires that the tree construction steps preserves Λ^A, in that the attacker has no advantage in changing the attack strategy which was optimal up to the previous step, thus recovering the soundness of the greedy construction. We define attack invariance during tree construction as follows.

Definition 5.4.3 (Attack Invariance)**.** *Let t be a decision tree and let t' be the decision tree obtained by replacing a leaf λ of t with the new sub-tree $\sigma(f, v, \lambda_l, \lambda_r)$. We say that t' satisfies* attack invariance *for the dataset $\mathcal{D}$ and the attacker A iff:*

$$\forall (\boldsymbol{x}, y) \in \mathcal{D}^\lambda : \Lambda^A(t', (\boldsymbol{x}, y)) \cap \{\lambda_l, \lambda_r\} \neq \emptyset.$$

The above definition states that, after growing a new sub-tree from λ, the set of the best options for the attacker against the instances in λ must include the newly created leaves, so that the path originally leading to λ still represents the most effective attack strategy against those instances.

Example 5.4.3 (Attack Invariance). Let t be the decision tree of Figure 5.2.(c). Figure 5.2.(d) shows an example where adding a new sub-tree to t leads to a decision tree t' which breaks the attack invariance property. Indeed, we have $\Lambda^A(t', (\boldsymbol{x}\mathbf{2}, 0)) = \{\lambda(-1.5)\}$, which contains neither $\lambda(1)$, nor $\lambda(2)$. Notice that the best attack strategy has indeed changed with respect to t, as leaving $\boldsymbol{x}\mathbf{2}$ unaltered now produces a larger loss (2.25) than the originally strongest attack (1.0).

We enforce attack invariance by introducing a set of *constraints* into the optimization problem of Equation 5.2. Suppose that the new sub-tree $\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$ replaces the leaf λ and that an instance $(\boldsymbol{x}, y) \in \mathcal{D}^\lambda$ is placed in the right child by robust splitting, because one of its corruptions traverses the threshold v and $\ell(\hat{y}_r, y) \geq \ell(\hat{y}_l, y)$. Then, attack invariance is

granted if, whenever the leaves $\lambda(\hat{y}_l)$ and $\lambda(\hat{y}_r)$ are later replaced by sub-trees t_l and t_r, there exists an attack $\boldsymbol{z} \in A(\boldsymbol{x})$ that falls into a leaf of t_r generating a loss larger than (or equal to) the loss of any other attack falling in t_l. We enforce such constraint during the recursive tree building process as follows. The requirement $\ell(\hat{y}_r, y) \geq \ell(\hat{y}_l, y)$ is transformed in the pair of constraints $\ell(t_r(\boldsymbol{x}), y) \geq \gamma$ and $\ell(t_l(\boldsymbol{x}), y) \leq \gamma$, where $\gamma = \min\{\ell(\hat{y}_r, y), \ell(\hat{y}_l, y)\}$. These two constraints are respectively propagated into the recursion on the right and left children. As long as any sub-tree t_r replacing $\lambda(\hat{y}_r)$ satisfies the constraint $\ell(t_r(\boldsymbol{x}), y) \geq \gamma$ and any sub-tree t_l replacing $\lambda(\hat{y}_l)$ satisfies the constraint $\ell(t_l(\boldsymbol{x}), y) \leq \gamma$, the attacker has no advantage in changing the original attack strategy, hence attack invariance is enforced.

To implement this mechanism, each leaf λ is extended with a set of constraints, which is initially empty for the root of the tree. When λ is then split upon tree growing, the constraints therein are included in the optimization problem of Equation 5.2 to determine the best predictions $\hat{y}_l, \hat{y}_r$ for the new leaves. These constraints are then propagated to the new leaves and new constraints are generated for them based on the following definition, which formalizes the previous intuition.

Definition 5.4.4 (Constraints Propagation and Generation)**.** *Let λ be a leaf to be replaced with sub-tree $\hat{t} = \sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$ and let $\mathcal{C}$ be its set of constraints. The sets of constraints $\mathcal{C}_L(\hat{t}, A)$ and $\mathcal{C}_R(\hat{t}, A)$ for the two new leaves are defined as follows:*[8]

- *if $\ell(t(\boldsymbol{x}), y) \lesseqgtr \gamma \in \mathcal{C}$ and there exists $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $z_f \leq v$, then $\ell(t_l(\boldsymbol{x}), y) \lesseqgtr \gamma$ is added to $\mathcal{C}_L(\hat{t}, A)$;*
- *if $\ell(t(\boldsymbol{x}), y) \lesseqgtr \gamma \in \mathcal{C}$ and there exists $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $z_f > v$, then $\ell(t_r(\boldsymbol{x}), y) \lesseqgtr \gamma$ is added to $\mathcal{C}_R(\hat{t}, A)$;*
- *if $(\boldsymbol{x}, y) \in \mathcal{D}_u^{\lambda}(f, v, A) \cap \mathcal{D}_L^{\lambda}(\hat{t}, A)$, then $\ell(t_l(\boldsymbol{x}), y) \geq \ell(\hat{y}_r, y)$ is added to $\mathcal{C}_L(\hat{t}, A)$ and $\ell(t_r(\boldsymbol{x}), y) \leq \ell(\hat{y}_r, y)$ is added to $\mathcal{C}_R(\hat{t}, A)$;*
- *if $(\boldsymbol{x}, y) \in \mathcal{D}_u^{\lambda}(f, v, A) \cap \mathcal{D}_R^{\lambda}(\hat{t}, A)$, then $\ell(t_l(\boldsymbol{x}), y) \leq \ell(\hat{y}_l, y)$ is added to $\mathcal{C}_L(\hat{t}, A)$ and $\ell(t_r(\boldsymbol{x}), y) \geq \ell(\hat{y}_l, y)$ is added to $\mathcal{C}_R(\hat{t}, A)$.*

Example 5.4.4 (Enforcing Constraints). The tree in Fig. 5.2.(e) is generated by enforcing a constraint on the loss of $\boldsymbol{x2}$. After splitting the root, the constraint $\ell(t_r(\boldsymbol{x2}), 0) \geq \ell(\hat{y}_l, 0)$ is generated for the right leaf of the tree in Fig. 5.2.(c), where $\ell(\hat{y}_l, 0) = (-1.5 - 0)^2 = 2.25$. The solution of the *constrained* optimization problem on the right child of the tree in

[8] We use the symbol $\lesseqgtr$ to stand for either $\leq$ or $\geq$ when the distinction is unimportant.

Fig. 5.2.(c) finally grows two new leaves, generating the tree in Fig. 5.2.(e). The difference from the tree in Fig. 5.2.(d) is that the prediction on the left leaf of the right child of the root has been enforced to satisfy the required constraint. For this tree, the attacker has no gain in changing attack strategy over the previous step of the tree construction, shown in Figure 5.2.(c).

More formally, after growing the tree in Fig. 5.2.(c) with suitable constraints we obtain the tree t' in Fig. 5.2.(e), where the leaf $\lambda(1.6)$ has been substituted with a decision stump with the two new leaves $\{\lambda(1.5), \lambda(2)\}$. This gives $\Lambda^A(t', (\boldsymbol{x}\mathbf{2}, 0)) = \{\lambda(-1.5), \lambda(1.5)\}$, where $\Lambda^A(t', (\boldsymbol{x}\mathbf{2}, 0)) \cap \{\lambda(1.5), \lambda(2)\} = \{\lambda(1.5)\} \neq \emptyset$, thus satisfying the attack invariance property.

Note that constraints grant attack invariance at the cost of reducing the space of the possible solutions for tree-growing. Nevertheless, in the experimental section we show that this property does not prevent the construction of robust decision trees that are also accurate in absence of attacks.

5.4.4 Tree Learning Algorithm

Our TREANT construction is summarized in Algorithm 3. The core of the logic is in the call to the TREANTSPLIT function (line 3), which takes as input a dataset $\mathcal{D}$, an attacker A and a set of constraints $\mathcal{C}$ initially empty, and implements the construction detailed along the present section. The construction terminates when it is not possible to further reduce $\mathcal{L}^A$ (line 4).

Function TREANTSPLIT is summarized in Algorithm 4. Specifically, the function returns the sub-tree minimizing the loss under attack $\mathcal{L}^A$ on $\mathcal{D}$ subject to the constraints $\mathcal{C}$, based on the ternary partitioning (lines 2-3). It then splits $\mathcal{D}$ by means of the robust splitting strategy (lines 4-5) and returns new sets of constraints (lines 6-7), which are used to recursively build the left and right sub-trees. The optimization problem (line 2) can be numerically solved via the `scipy` implementation of the SLSQP (Sequential Least SQuares Programming) method, which allows the minimization of a function subject to inequality constraints, like the constraint set $\mathcal{C}$ generated/propagated by TREANT during tree growing.

There is an important point worth discussing about the implementation of the algorithm. As careful readers may have noticed, the TREANTSPLIT function splits each leaf λ by relying on the set of attacks $A(\boldsymbol{x})$ for all instances $(\boldsymbol{x}, y) \in \mathcal{D}^\lambda$. Though one could theoretically pre-compute all the possible attacks against the instances in $\mathcal{D}$, this would be very inefficient both in time and space, given the potentially huge number of instances and

Algorithm 3 TREANT

Input: training data $\mathcal{D}$, attacker A, constraints $\mathcal{C}$
$\hat{y} \leftarrow \text{argmin}_y \mathcal{L}^A(\lambda(y), \mathcal{D})$ subject to $\mathcal{C}$
$\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r)), \mathcal{D}_l, \mathcal{D}_r, \mathcal{C}_l, \mathcal{C}_r \leftarrow$ TREANTSPLIT$(\mathcal{D}, A, \mathcal{C})$
if $\mathcal{L}^A(\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r)), \mathcal{D}) < \mathcal{L}^A(\lambda(\hat{y}), \mathcal{D})$ **then**
    $t_l \leftarrow$ TREANT$(\mathcal{D}_l, A, \mathcal{C}_l)$
    $t_r \leftarrow$ TREANT$(\mathcal{D}_r, A, \mathcal{C}_r)$
    return $\sigma(f, v, t_l, t_r)$
else
    return $\lambda(\hat{y})$
end if

Algorithm 4 TREANTSPLIT

Input: training data $\mathcal{D}$, attacker A, constraints $\mathcal{C}$
▷ Build a set of candidate tree nodes $\mathcal{N}$ using Ternary Partitioning to optimize $\mathcal{L}^A$
$\mathcal{N} \leftarrow \{\sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r)) \mid \hat{y}_l, \hat{y}_r = \text{argmin}_{y_l, y_r} \mathcal{L}^A(\sigma(f, v, \lambda(y_l), \lambda(y_r)), \mathcal{D})$ subject to $\mathcal{C}\}$
▷ Select the candidate node $\hat{t} \in \mathcal{N}$ which minimizes the loss $\mathcal{L}^A$ on the training data $\mathcal{D}$
$\hat{t} = \text{argmin}_{t \in \mathcal{N}} \mathcal{L}^A(t, \mathcal{D}) = \sigma(f, v, \lambda(\hat{y}_l), \lambda(\hat{y}_r))$
▷ Robust Splitting (see Definition 5.4.2)
$\mathcal{D}_l \leftarrow \mathcal{D}_L(\hat{t}, A)$
$\mathcal{D}_r \leftarrow \mathcal{D}_R(\hat{t}, A)$
▷ Constraint Propagation and Generation (see Definition 5.4.4)
$\mathcal{C}_l \leftarrow \mathcal{C}_L(\hat{t}, A)$
$\mathcal{C}_r \leftarrow \mathcal{C}_R(\hat{t}, A)$
return $\hat{t}, \mathcal{D}_l, \mathcal{D}_r, \mathcal{C}_l, \mathcal{C}_r$

attacks. Our implementation, instead, incrementally computes a *sufficient* subset of $A(\boldsymbol{x})$ along the tree construction. This makes the construction computationally feasible by exploiting the observation that the ternary partitioning used for node splitting only requires the identification of a single attack against the feature which is tested in the node predicate, hence the computation of the full set of attacks is not actually needed.

More specifically, each instance $(\boldsymbol{x}, y)$ is enriched with a cost annotation k, denoted by $(\boldsymbol{x}, y)^k$, initially set to 0 on the root. Such annotation keeps track of the cost of the adversarial manipulations performed to push $(\boldsymbol{x}, y)$ into λ during the tree construction. When splitting the leaf λ on (f, v), the algorithm generates only the attacks against the feature f and assumes that k was already spent from the attacker's budget to further reduce the number of possible attacks. When the instance $(\boldsymbol{x}, y)^k$ is pushed into the left or right partition of $\mathcal{D}^\lambda$ by robust splitting, the label k is updated to $k + k'$, where

k' is the *minimum cost* the attacker must spend to achieve the desired node outcome. The same idea is applied when propagating constraints, which are also associated with specific instances $(\boldsymbol{x}, y)$ for which the computation of $A(\boldsymbol{x})$ is required.

Observe that this implementation assumes that only the cost of adversarial manipulations is relevant, not their magnitude, which is still sound when none of the corrupted features is tested multiple times on the same path of the tree. We enforce such restriction during the tree construction, which further regularizes the growing of the tree. Since we are eventually interested in decision tree ensembles, this does not impact on the performance of whole trained models.

5.4.5 Complexity Analysis

In a standard decision tree construction algorithm, the cost of splitting a node λ is $O(d \cdot |\mathcal{D}^\lambda|)$, since this requires a scan of the instances in λ to find the best feature and threshold for tree growing [114]. Similarly, TREANT splits each decision tree node by means of an exhaustive search over all possible features and thresholds. The key difference lies in the node splitting procedure (Algorithm 4). In particular, given a fixed feature and threshold, TREANT pays an extra cost over traditional tree constructions coming from two factors: the ternary partitioning and the corresponding $\mathcal{L}^A$ optimization problem (see Eq. 5.2). As we anticipated, the optimization problem can be solved by the SLSQP method, which has cubic complexity in the number of variables slsqp. Our problem only has two variables, corresponding to the predictions on the left and right leaf respectively.

Regarding the computational complexity of the ternary partitioning, we analyse below the cost of deciding whether or not an instance $\boldsymbol{x}$ belongs to $\mathcal{D}_u^\lambda(f, v, A)$. This has to be paid for every instance in the dataset and it is a multiplicative factor with respect to the standard decision tree growing algorithm.

Proposition 1. Given a split candidate pair (f, v), an instance $(\boldsymbol{x}, y) \in \mathcal{D}$ and an attacker $A = (R, K)$, the computational complexity of deciding whether $\boldsymbol{x}$ belongs to $\mathcal{D}_u^\lambda(f, v, A)$ is $O\left(\left(\sqrt{2}|R| + 1\right)^{\frac{2K}{k^*}}\right)$, where k^* is the cost of the cheapest rule in R.

To assess whether $\boldsymbol{x}$ can be attacked, we are interested in finding (if it exists) a sequence of perturbations with minimum cost which leads to crossing the threshold v. Being of minimum cost, we can assume that each rule r in such chain is maximally exploited so as to perturb the instance

$\boldsymbol{x}$ to the extremes of the interval $[x_f + \delta_l, x_f + \delta_u]$ or to the ends of a precondition interval enabling some other rule in R. Therefore, in the worst case, each rule $r \in R$ can modify $\boldsymbol{x}$ into $2 + 2|R|$ different ways, and this process is repeated up to budget exhaustion, i.e., at most K/k^* times where k^* is the cost of the cheapest rule in R. We finally get a total cost of $O\left((|R|(2+2|R|))^{\frac{K}{k^*}}\right)$, or equivalently $O\left(\left(\sqrt{2}|R|+1\right)^{\frac{2K}{k^*}}\right)$.

Note that this bound is not tight as not all rules are always applicable, k^* might be far from the cost of other rules, and it might not always be possible to generate $2 + 2|R|$ perturbed instances. In fact, we may not need to enumerate all the possible perturbations.

Below we consider a setting where the set of rules R encodes a L_1-distance attacker, which is one of the most commonly used models in literature. As discussed in Section 5.3.2, the above attacker can be encoded with one single rule per feature, which leads to a significantly lower computational complexity.

Proposition 2. Given a split candidate pair (f, v), a dataset $\mathcal{D}$ and an attacker A such that there is only one rule per feature of the form $r : [-\infty, +\infty] \xrightarrow{f}_{\varepsilon} [-\varepsilon, +\varepsilon]$, the computational complexity of deciding whether $\boldsymbol{x}$ belongs to $\mathcal{D}_u^{\lambda}(f, v, A)$ is $O(1)$.

Supposing $x_f \leq v$ (a similar reasoning holds for $x_f > v$), an attacked instance $\boldsymbol{z}$ with $z_f > v$ can be crafted with minimum cost by applying the rule r to $\boldsymbol{x}$ a total of $\lfloor (v - x_f)/\varepsilon \rfloor + 1$ times, where each application bears a cost equal to ε. If the attacker's budget K is sufficient to cover such cost, then the instance $\boldsymbol{x}$ belongs to $\mathcal{D}_u^{\lambda}(f, v, A)$, and this check can be performed in constant time $O(1)$. By repeating this for every instance in $\mathcal{D}$, we have a total cost of $O(|\mathcal{D}|)$.

The above strategy can be easily generalized to L_0-distance attackers and to every other rule set R where only one rule per feature is given. Specifically, an analogous yet slightly more involved argument proves the following result.

Proposition 3. Given a split candidate pair (f, v), a dataset $\mathcal{D}$ and an attacker A such that there is only one rule per feature of the form $r : [a, b] \xrightarrow{f}_{k} [\delta_l, \delta_u]$, the computational complexity of deciding whether $\boldsymbol{x}$ belongs to $\mathcal{D}_u^{\lambda}(f, v, A)$ is $O(1)$.

Our experimental evaluation builds on the threat model supported by the proposition above, which shows that the ternary partitioning can be efficiently performed in practical use cases.

5.4.6 From Decision Trees to Tree Ensembles

In this section we introduce a new tree learning algorithm, yet individual decision trees are rarely used in practice and ensemble methods are generally preferred for real-world tasks. As anticipated in Section 2.1.1, the most popular ensemble methods for decision trees are Random Forest (RF) and Gradient Boosting Decision Trees (GBDT). Extending TREANT to these ensemble methods is straightforward, because both methods can be seen as *meta-algorithms* which build on top of existing tree learning algorithms. RF builds multiple independent trees $t_1, \ldots, t_n$ by using bagging and per-node feature sampling in each t_j, while in GBDT each tree t_i adds a gradient descent step to minimize the cumulative loss incurred by the previous trees. Hence, both methods eventually apply an underlying tree learning algorithm multiple times to different training data.

The only delicate point to notice is that, since the TREANT algorithm is parametric over an attacker $A = (R, K)$, using the same attacker in the individual tree constructions is a *conservative* approach to ensemble learning. This comes from two factors: first, the construction of each tree t_j relies on a robust splitting procedure which only accounts for attacks against t_j, yet other trees in the ensemble might contribute to make such attacks ineffective; second, the attacker's budget K is essentially refreshed along each tree construction. This conservative approach overestimates the power of the attacker and cannot harm security, though it might unnecessarily downgrade performance in the unattacked setting. That said, our experimental evaluation in the next section shows that ensembles built using TREANT are very accurate also in such setting. We leave the design of more sophisticated ensemble learning techniques to future work.

5.5 Experimental Evaluation

5.5.1 Methodology

We compare the performance of classifiers trained by different learning algorithms: two standard approaches, i.e., Random Forest [20] (RF) and Gradient Boosting Decision Trees [63] (GBDT) as provided by the LightGBM[9] framework; two state-of-the-art adversarial learning techniques, i.e., Adversarial Boosting [90] (AB) and Robust Trees [35] (RT); and a Random Forest of trees trained using the proposed TREANT algorithm (RF-TREANT). Notice that the original implementation of AB exploited a heuristic algorithm to quickly find effective adversarial examples, which does not

[9] https://github.com/microsoft/LightGBM

Dataset	census	wine	credit	malware
n. of instances	45,222	6,497	30,000	47,580
n. of features	13	12	24	1,000
class distribution (pos.÷neg. %)	25÷75	63÷37	22÷78	96÷4

Table 5.2: *Main statistics of the datasets used in our experiments.*

guarantee to find the most damaging attack. Our own implementation of AB, which is built on top of LightGBM, exploits the white-box attack generation method described in Section 5.3.3 to identify the *best* adversarial examples. In this regard, our implementation of AB is more effective than the original algorithm.

We perform our experimental evaluation on four publicly available data sets, using three standard validity measures: accuracy, macro F1 and ROC AUC. We compute all measures both in absence of attacks and under attack, using our white-box attack generation method. We used a 60-20-20 train-validation-test split through stratified sampling. Hyper-parameter tuning on the validation data was conducted to set the number of trees (≤ 100), number of leaves ($\{8, 32, 256\}$) and learning rate ($\{0.01, 0.05, 0.1\}$) of the various ensembles so as to maximize ROC AUC. All the results reported below were measured on the test data. Observe that all the compared adversarial learning techniques are parametric with respect to the budget granted to the attacker, modeling his power: we consider multiple instances of such budget both for training (*train budget*) and for testing (*test budget*).

5.5.2 Datasets and Threat Models

We perform our experimental evaluation on four datasets: Census Income,[10] Wine Quality,[11] Default of Credit Cards,[12] and Malware Analysis.[13] We refer to such datasets as `census`, `wine`, `credit`, and `malware`, respectively. Their main statistics are shown in Table 5.2; each dataset is associated with a binary classification task.[14]

We therefore design different threat models by means of sets of rewriting rules indicating the attacker capabilities, with each set tailored to a given

[10] `https://archive.ics.uci.edu/ml/datasets/census+income`
[11] `https://www.kaggle.com/c/uci-wine-quality-dataset/data`
[12] `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`
[13] `https://ieee-dataport.org/open-access/malware-analysis-datasets-top-1000-pe-imports`
[14] The `wine` dataset was originally conceived for a multiclass classification problem; we turned that into a binary one, where the positive class identifies good-quality wines (i.e., those whose quality is at least 6, on a 0-10 scale) and the negative class contains the remaining instances.

dataset. The features targeted by those rules have been selected after a preliminary data exploration stage, where we investigated the importance and the data distribution of all the features, e.g., to identify the magnitude of adversarial perturbations. Of course, in a real-world deployment the definition of the appropriate threat model would depend on the specific application scenarios of the trained classifiers: the definitions here considered are evocative of plausible attack scenarios possibly anticipated by domain experts, yet they are primarily intended as a way to test the robustness of the trained models against evasion attacks.

In the case of `census`, we define six rewriting rules: (i) if a citizen never worked, he can pretend that he actually works without pay; (ii) if a citizen is divorced or separated, he can pretend that he never got married; (iii) a citizen can present his occupation as a generic "other service"; (iv) a citizen can cheat on his education level by lowering it by 1; (v) a citizen can add up to \$2,000 to his capital gain; (vi) a citizen can add up to 4 hrs per week to his working hours. We let (i),(ii), and (iii) cost 1, (iv) cost 20, (v) cost 50, and finally (vi) cost 100 budget units. We consider 30, 60, 90, and 120 as possible values for the budget.

In the case of `wine`, we specify four rewriting rules: (i) the alcohol level can be increased by 0.5% if its original value is less than 11%; (ii) the residual sugar can be decreased by 0.25 g/L if it is already greater than 2 g/L; (iii) the volatile acidity can be reduced by 0.1 g/L if it is already greater than 0.25 g/L; (iv) free sulfur dioxide can be reduced by -2 g/L if it is already greater than 25 g/L. We let (i) cost 20, (ii) and (iii) cost 30, and (iv) cost 50 budget units. We consider 20, 40, 60, 80 and 100 as possible values for the budget.

For `credit`, the attacker is represented by three rewriting rules: (i) the repayment status of August and September can be reduced by 1 month if the payment is delayed up to 5 months; (ii) the amount of bill statement in September can be decreased by 4,000 NT dollars if it is between 20,000 and 500,000; and (iii) the amount of given credit can be increased by 20,000 NT dollars if it is below 200,000. For each rule, a cost of 10 budget units is required. We consider 10, 30, 40, and 60 as possible budget values.

Finally, the attacker targeting the `malware` dataset is modelled by three rewriting rules, which allow one to flip three binary features from 0 to 1. These features represent invocations to the following functions: `_cexit`, `SearchPathW`, `exit`. Each rule has cost 20 and we pick three possible values of the budget: 20, 40 and 60. Note that since the `malware` dataset is imbalanced, at training time we oversampled the minority class so as to increase ten times the corresponding instances, yet the oversampling was

not applied at test time. For such imbalanced datasets, accuracy values are less relevant than macro F1 and ROC measures, for which measures a trivial classifier always predicting the majority class would barely score 0.50.

5.5.3 Experimental Evaluation

The primary goal of our experiments is answering the following research questions:

1. What is the robustness of standard decision tree ensembles like RF and GBDT against evasion attacks?
2. Can adversarial learning techniques improve robustness against evasion attacks and which technique is the most effective?
3. What is the impact of the training budget on the effectiveness of adversarial learning techniques?
4. What are the key structural properties of the decision trees trained by TREANT, i.e., why do they provide appropriate robustness guarantees?
5. What is the performance overhead of TREANT compared to other adversarial learning techniques?

We report on the answer to each research question in a separate sub-section.

5.5.3.1 Robustness of Standard Decision Tree Ensembles

In Figure 5.3, we show how the accuracy, F1 and ROC AUC of standard ensembles of decision trees trained by RF and GBDT change in presence of attacks. The x-axis indicates the testing budget of the attacker, normalized in the range $[0, 1]$, with a value of 0 denoting the unattacked scenario.

Two main findings appear clear from the plots. First, both GBDT and RF are severely impacted when they are attacked, and their performance deteriorates to the point of turning them into almost random classifiers already when the attacker spends just half of the maximum budget, e.g., in the case of the `wine` dataset. On that dataset, the drop of ROC AUC ranges from -25.8% to -40.6% for GBDT and from -15.5% to -28.4% for RF, when the attacker is supplied just half of the budget. Second, RF typically behaves better than GBDT on our validity measures, with a few cases where the improvement is very significant. A possible explanation of this phenomenon is that RF usually exhibits better generalization performance, while GBDT is known to be more susceptible to jiggling data, therefore more likely to overfit [122]. Since robustness to adversarial examples in a way resembles

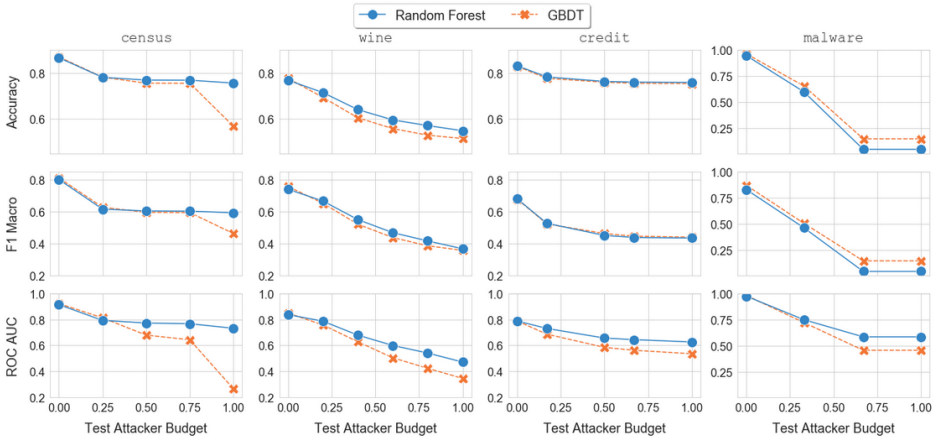

Figure 5.3: *The impact of the attacker on RF and GBDT.*

the ability of a model to generalize, RF is less affected by the attacker than GBDT. Still, the performance drop under attack is so massive even for RF that none of the traditional methods can be reliably adopted in an adversarial setting.

The higher resiliency of RF to adversarial examples motivated our choice to deploy TREANT on top of such ensemble method in our implementation. It is worth remarking though that TREANT is still general enough to be plugged into other frameworks for ensemble tree learning.

5.5.3.2 Robustness of Adversarial Learning Techniques

We now measure the benefit of using adversarial learning techniques to contrast the impact of evasion attacks at test time. More specifically, we validate the robustness of our method in comparison with the two state-of-the-art adversarial learning methods: Adversarial Boosting (AB) and Robust Trees (RT). Note that the authors of RT did not experimentally compare RT against AB in their original work [35].

We first investigate how robust a model is when it is targeted by an attacker with a test budget exactly matching the training budget. This simulates the desirable scenario where the threat model was accurately defined, i.e., each model is trained knowing the actual attacker capabilities. Table 5.3 shows the results obtained by the different adversarial learning techniques for the different training/test budgets. It is clear how our method improves over its competitors, basically for all measures and datasets. Most importantly, the superiority of our approach often becomes more apparent

			AB			RT			RF-TREANT		
			Acc.	F_1	ROC	Acc.	F_1	ROC	Acc.	F_1	ROC
census	Budget	30	**0.85**	**0.78**	**0.90**	0.81	0.69	0.88	**0.85**	0.77	**0.90**
		60	0.78	0.69	0.83	0.81	0.70	0.87	**0.85***	**0.77**	**0.89**
		90	0.80	0.71	0.83	0.78	0.61	0.86	**0.85***	**0.77**	**0.89**
		120	0.79	0.69	0.79	0.74	0.56	0.53	**0.84***	**0.76**	**0.89**
wine	Budget	20	**0.76**	**0.74**	**0.82**	0.73	0.70	0.80	**0.76**	**0.74**	**0.82**
		40	0.72	**0.69**	0.79	0.63	0.57	0.67	**0.73***	**0.69**	**0.80**
		60	**0.72**	**0.69**	0.79	0.59	0.49	0.55	**0.72**	0.68	**0.80**
		80	0.72	0.68	0.77	0.62	0.54	0.63	**0.73***	**0.69**	**0.80**
		100	0.70	0.67	0.76	0.65	0.57	0.71	**0.73***	**0.69**	**0.80**
credit	Budget	10	0.81	0.64	0.75	0.81	0.63	0.75	**0.82***	**0.66**	**0.77**
		30	0.79	0.54	0.66	**0.81**	0.61	0.73	**0.81**	**0.62**	**0.75**
		40	0.78	0.55	0.66	**0.81**	**0.62**	0.73	**0.81**	**0.62**	**0.74**
		60	0.78	0.53	0.62	**0.81**	**0.62**	0.72	**0.81**	**0.62**	**0.74**
malware	Budget	20	**0.94**	0.78	0.95	**0.94**	**0.83**	**0.97**	**0.94**	**0.83**	**0.97**
		40	**0.94**	0.79	0.95	**0.94**	**0.83**	**0.97**	**0.94**	**0.83**	**0.97**
		60	0.88	0.69	0.94	0.94	**0.83**	**0.97**	**0.95**	**0.83**	**0.97**

Table 5.3: *Comparison of adversarial learning techniques (training and test budget coincide). The asterisk denotes statistically significant difference against the best competitor with p value 0.01 under McNemar test.*

as the strength of the attacker grows. For example, the percentage improvement in ROC AUC over AB on the `credit` dataset amounts to 2.1% for budget 10, while this improvement grows to 19.6% for budget 60. It is also interesting to show that the heuristic approach implemented in RT is not always representative of all the possible attacks which might occur at test time: RT behaves similarly to our method on the `credit` and `malware` datasets, but it performs way worse on the `census` and `wine` datasets. In particular, the heuristic approach exploited by RT is most effective on the `malware` dataset, likely due to the presence of simple binary features. Indeed, this proves the effectiveness of our proposed strategy in presence of more complex datasets where the attacker behaviour cannot be approximated through simple heuristic approaches.

The second analysis we carry out considers the case of adversarial learning techniques trained with the maximum available budget. We use security evaluation curves to measure how the performance of the compared

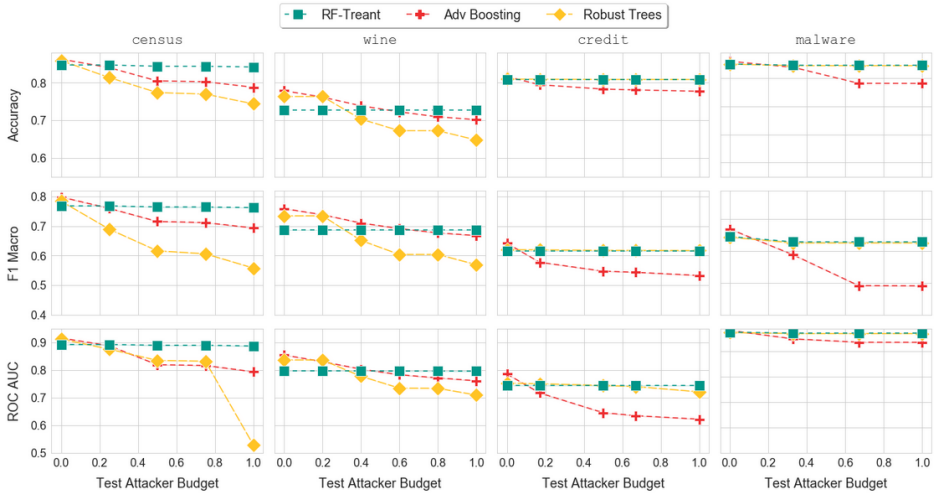

Figure 5.4: *Comparison of adversarial learning techniques for different test budgets and maximum train budget.*

methods changes when the test budget given to the attacker increases up to the maximum available. The results are shown in Figure 5.4, where we normalized the test budget in $[0, 1]$. The security evaluation curves show that our method constantly outperforms its competitors on all datasets and measures, especially when the attacker gets stronger. The price to pay for this increased protection is just a slight performance degradation in the unattacked setting, which is always compensated under attack. Indeed, the performance of our method is nearly constant and insensitive to variations in the attacker's budget, which is extremely useful when such information is hard to exactly quantify for security experts. We observe again that RT cannot always cope with strong attackers: this is particularly apparent in the case of the `census` dataset, where the model trained by RT is completely fooled when the test budget reaches its maximum.

5.5.3.3 Impact of the Training Budget

Another intriguing aspect to consider is how much adversarial learning techniques are affected by the assumptions made on the attacker's capabilities upon learning, i.e., the value of the training budget. Figure 5.5 is essentially the "dual" of Figure 5.4, where we consider the strongest possible attacker (with the largest test budget) and we analyze how much models learned with different training budgets are able to respond to evasion attempts.

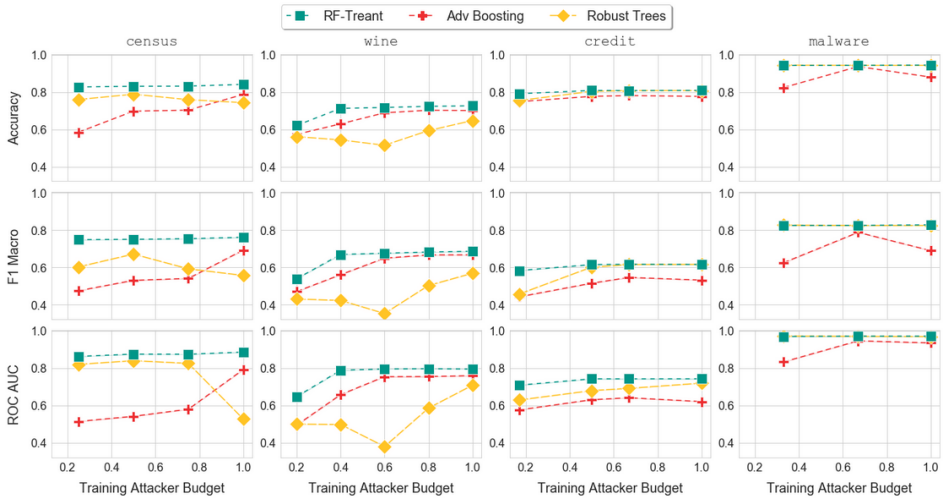

Figure 5.5: *Comparison of adversarial learning techniques for different train budgets and maximum test budget.*

We draw the following observations. First, our method leads to the most robust models for all measures and datasets, irrespective of the budget used for training. Moreover, our method is the one which most evidently presents a healthy, expected trend: the greater the training budget used to learn the model, the better its performance under attack. This trend eventually reaches its peak when the training budget matches the test budget. AB and RT show a more unpredictable behavior, as their performance fluctuates up and down, and sometimes suddenly drops. This is likely due to the fact that these approaches are heuristics and eventually shortsighted with respect to the set of all the attacks which might occur at test time. Finally, we remark a last appealing, distinctive aspect of our method: even when the training uses a significantly smaller budget than the one used by the attacker at test time, it already achieves nearly optimal performance. The same is not true for its competitors, which complicates their deployment in real-world settings, since it requires security experts to be very precise in their budget estimates.

5.5.3.4 Structural Properties of Decision Trees

To better understand why our approach provides improved robustness against attacks, we studied the feature importance of the trained adversarial learning models on the different datasets (considering the highest training budget). Figure 5.6 shows the plots built for the `wine` and `credit` datasets,

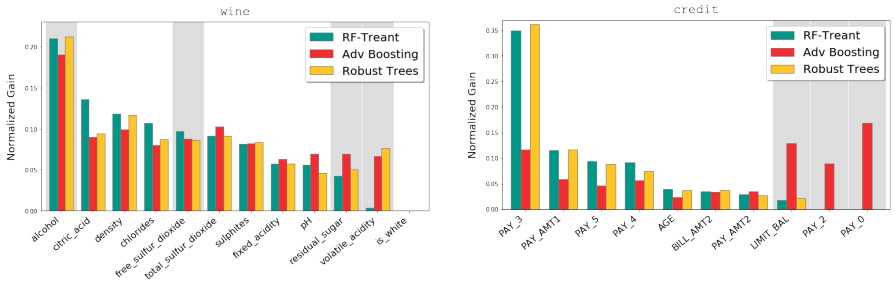

Figure 5.6: *Feature importance for* `wine` *and* `credit` *datasets. The grey background denotes attacked features.*

where we use a grey background to denote attacked features.

In the case of the `wine` dataset, we observe that the alcohol level (though attacked) is a very useful feature for all the models. The importance of this feature seems inherent to the training data, however the figure is very different for other attacked features, like residual sugar and volatile acidity. These features are quite useful overall for AB and RT, while they are essentially not used by our model: this justifies the improved robustness of our method over its competitors. As to the case of the `credit` dataset, the feature importance of RT follows the same lines of our model: attacked features are essentially not used, while AB is fooled into giving a lot of importance to them. This motivates why AB performs quite worse than the other two methods there.

5.5.3.5 Efficiency Analysis

We conclude our experiments with an efficiency evaluation of our algorithm. We note that code optimization was not the main goal of our prototype implementation, i.e., we were more concerned about robustness to attacks than about efficiency. However, it is still possible to draw interesting conclusions about efficiency as well. Figure 5.7 compares the training times of our models against those of the models trained by our implementation of RT on the `wine` and `credit` datasets. This analysis is insightful, because RT is essentially a simplified version of our approach, where the ternary partitioning is heuristically approximated and attack invariance is not enforced upon tree construction.

The figure reports the normalized running times with respect to the fastest training time, i.e., RT with the lowest train budget, as a function of the attacker budget. The figure also plots the amount of generated attacks in terms of the multiplicative increase of the original training dataset,

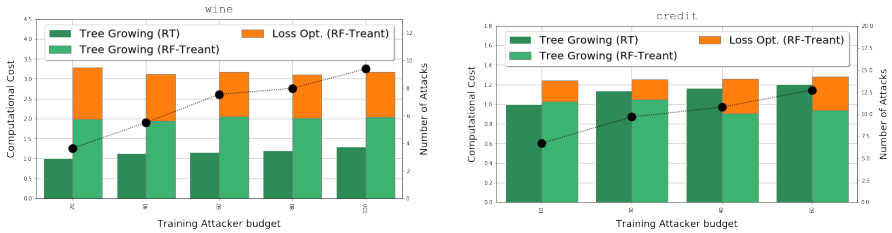

Figure 5.7: *Normalized training times for the* `wine` *and* `credit` *datasets. The black line shows the amount of attacks generated during training in terms of a multiplicative factor of the original dataset size.*

due to the corrupted instances an attacker can generate for each budget.

Regarding RF-TREANT, we report a breakdown of its cost into loss optimization and tree growing. The impact of the loss optimization ranges between 20% and 30% of the total time. Fluctuations in the loss optimization cost are due to the use of a numerical solver, whose inner workings and resolution strategies may change on the basis of the number of variables, constraints, etc., of the optimization problem. The remaining cost of the tree growing phase, which includes the ternary partitioning, dominates the overall RF-TREANT training time. Overall, the cost of RF-TREANT is pretty stable when varying the attacker budget.

The figure confirms that RT is indeed faster than our approach, for all budgets. For the `wine` dataset, the overhead is essentially of a 3x factor, while for `credit` the cost of the two algorithms is actually very close. The overhead on `wine` however is largely justified by the complete coverage of all the possible attack strategies, which greatly improves the robustness guarantees provided by our approach (see Table 5.3). An interesting trend is shown on the `credit` dataset, where with an increased budget the cost of solving the optimization problem increases, while the tree growing itself becomes cheaper than that of RT. This is likely due to the larger number of attacks that induces more complex optimization problems to be solved, i.e., with more constraints required to enforce attack invariance. At the same time, such constraints reduce the tree growing opportunities, thus reducing the cost of the tree growing.

It is also interesting to note that the number of generated attacks grows much faster than the training times: for example, when moving from budget 20 to budget 100, the number of generated attacks for the `wine` dataset increases from around 4x to around 9x the dataset size, yet the overall training time does not significantly increase. This means that the attack generation takes only a limited fraction of the training time, because each feature just

needs to be attacked independently of all the others. Similar considerations apply to the `credit` dataset, though it is worth noticing that the overhead of our approach over RT is much more limited there for all budgets.

5.6 Summary

This chapter proposes TREANT, a new adversarial learning algorithm that is able to grow decision trees that are resilient against evasion attacks. TREANT is the first algorithm which greedily, yet soundly, minimizes an evasion-aware loss function, capturing the attacker's goal of maximizing prediction errors. Our experiments, conducted on four publicly available datasets, confirm that TREANT produces accurate tree ensembles, which are extremely robust against evasion attacks. Compared to the state of the art, TREANT exhibits a significant improvement.

CHAPTER 6

EiFFFeL: Enforcing Fairness in Forests by Flipping Leaves

Nowadays, Machine Learning (ML) techniques are extensively adopted in many socially sensitive systems, thus requiring to carefully study the fairness of the decisions taken by such systems. Many approaches have been proposed to address and to make sure there is no bias against individuals or specific groups which might originally come from biased training datasets or algorithm design.

In this chapter, we propose a fairness enforcing approach called EiFFFeL –Enforcing Fairness in Forests by Flipping Leaves– which exploits tree-based or leaf-based post-processing strategies to relabel leaves of selected decision trees of a given forest. Experimental results show that our approach achieves a user-defined group fairness degree without losing a significant amount of accuracy.

6.1 Introduction

Algorithms proposed recently in bias mitigation has focused on neural networks. However, the efficiency and explainability of tree ensembles for

many applications makes them preferable to be implemented in many areas. Even though there are few works focused on studying fairness for trees and tree ensembles, notably [67, 88, 141, 184], most of them are focused on single decision tree classifiers and in-processing approaches. Our interest mainly lies in developing fair random forest classifiers with post-processing approaches designed to relabel leaves with accuracy and discrimination constraints. We take advantage of implementing a post-process approach, in which we do not require to know the training process.

Contributions. We focus on decision tree ensembles for binary classification tasks susceptible to group discrimination with respect to attributes sensitive classes such as age, gender, race, etc. We propose a post-processing approach named EiFFFeL –Enforcing Fairness in Forests by Flipping Leaves– that given a forest, however trained, selects a subset of its leaves and changes their predictions so as to reduce the discrimination degree of the forest.

We summarize the main contributions of our work as follows.

1. We propose an iterative leaf flipping post-processing algorithm to ensure group fairness .
2. We devise tree-based and leaf-based flipping methodologies on top of random forest classifier to enforce fairness.
3. We report experimental evaluations of group fairness on three different datasets, aiming to empirically show the effectiveness of our method.

6.2 Fairness in Machine Learning

Without loss of generality, we consider a binary classifier $g : \mathcal{X} \to \mathcal{Y}$ that maps an input feature vector $\boldsymbol{x} \in \mathcal{X}$ to a binary class label $y \in \mathcal{Y} = \{0, 1\}$. Among the attributes in the feature space $\mathcal{X}$, a binary attribute called *sensitive feature* $S \in \{0, 1\}$ identifies the aspects of data which are socio-culturally precarious for the application of machine learning. Specifically, given $x \in \mathcal{X}$ and $x.S$ the value of the sensitive attribute S for the given instance, if $x.S = 0$ then we say that x belongs to the *unprivileged group* that could possibly be discriminated.

6.2.1 Fairness and Discrimination Definitions

To achieve non-discriminatory and fair machine learning model, it is essential to first define *fairness*. In a broad context, fairness can be seen from an individual or a group point of view. *Individual fairness* requires that similar

individuals being treated similarly. *Group fairness* requires fairness of the classification model to apply on the two groups, defined through the binary sensitive feature S [54]. Our work focuses on group fairness, in which a group of individuals identified by S risks for experiencing discrimination.

We define the discrimination of a classifier measured by *group fairness* as follows. Recall that attribute $S = 0$ identifies the unprivileged group, while $S = 1$ corresponds to the privileged one, whose members are not discriminated but rather favoured by a learnt ML model. Moreover, we assume that the values 1 and 0 of class label Y represent *favorable* and *unfavorable* outcomes, respectively. For example, $Y = 1$ might correspond to the decision of granting a loan, thus favouring a bank customer.

A classifier g applied over $x \in \mathcal{X}$ is non-discriminatory if its prediction $g(\boldsymbol{x})$ is statistically independent of the sensitive attribute S. Hence, a classifier is fair if both groups have equal probability of being classified as belonging to the favorable class, which is the desirable outcome.

Using the problem formalization by [88], the discrimination of a model g with respect to a sensitive attribute S and a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ can be computed as follows:

$$disc_{\mathcal{D},S,g} := \frac{|\{(\boldsymbol{x}, y) \in \mathcal{D} \mid \boldsymbol{x}.S = 1 \wedge g(\boldsymbol{x}) = 1\}|}{|\{(\boldsymbol{x}, y) \in \mathcal{D} \mid \boldsymbol{x}.S = 1\}|} - \frac{|\{(\boldsymbol{x}, y) \in \mathcal{D} \mid \boldsymbol{x}.S = 0 \wedge g(\boldsymbol{x}) = 1\}|}{|\{(\boldsymbol{x}, y) \in \mathcal{D} \mid \boldsymbol{x}.S = 0\}|},$$

where $\boldsymbol{x}.S$ refers to the sensitive attribute of the instance $\boldsymbol{x}$. When S and $\mathcal{D}$ are clear from the context we simply use the notation $disc_g$.

To clarify the above definition, let's consider the case of a classifier g used by the HR staff of a company. The classifier g suggests hiring when $g(\boldsymbol{x}) = 1$ vs. not hiring when $g(\boldsymbol{x}) = 0$. We may wonder whether the classifier favours *men* ($S = 1$) over *women* ($S = 0$). The value of $disc_g$ is large if the ratio of men with a favorable hiring prediction is larger than the ratio of women with a favorable hiring prediction. By minimizing $disc_g$ we can provide a fairer classifier w.r.t. the gender attribute.

Table 6.1: *Notation Summary*

Symbol	Meaning
$\mathcal{D}$	Dataset
S	Sensitive feature
λ	leaf
Λ	Set of Leaves to be flipped
$disc_{\mathcal{F}}$	Forest discrimination
$disc_t$	Tree discrimination
$accu_{\mathcal{F}}$	Forest accuracy
$\Delta disc_\lambda$	change in discrimination after flipping
$\Delta accu_\lambda$	change on accuracy after flipping
γ	Ratio of change in accuracy and discrimination

6.3 The EiFFFel Algorithm

We propose a novel post-processing algorithm named EiFFFeL that, given a forest of decision trees for a binary classification task, modifies the prediction of a carefully chosen set of leaves so as to reduce the forest's discrimination degree. This process is named leaf relabeling, or, since we are focusing on a binary prediction task, *leaf flipping*.

The rationale is to flip the prediction of the leaves that contribute the most to the model discrimination degree so as to make them fair. Recall that the score $disc_{\mathcal{D},S,g}$ adopted to evaluate the model's discrimination depends on the number of privileged/unprivileged instances with a favorable prediction. Therefore, by flipping a leaf label we can increase or decrease the number of instances that contribute to the discrimination score. Note that, while leaf relabeling can be done judiciously so as to reduce discrimination, modifying the leaf predictions determined at training time may reduce the accuracy of the whole forest.

Therefore the goal of EiFFFeL is to find a sweet-spot in the accuracy vs. discrimination trade-off. While leaf relabeling was introduced by [88] for a single tree, we improve such strategy and extend it to a forest of decision trees.

In this work we focus on Random Forests ensembles, which, for their high accuracy and limited bias, are an optimal candidate for building a fair classifier. The approach is however general and we leave to future work the application to other tree ensembles, such as those obtained by bagging and boosting approaches.

The proposed EiFFFel algorithm accepts a user-defined maximum discrimination constraint ϵ and a minimum relative accuracy drop constraint α. Given a forest $\mathcal{F}$, it iteratively modifies the prediction associated with a subset of the leaves of $\mathcal{F}$, until either the desired discrimination ϵ is achieved,

or the maximum required accuracy drop α is hit.

Below we first illustrate the *Leaf Scoring* strategy used to find the most discriminative leaves of a tree, and then we illustrate two variants of the EiFFFeL algorithm.

Algorithm 5 SCORE_LEAVES

Input: Decision Tree t, Dataset $\mathcal{D}$, Sensitive feature S
Output: Candidate flipping leaves Λ

1: $\Lambda \leftarrow \emptyset$
2: **for all** $\lambda \in t \mid \neg\lambda.flipped$ **do**
3: $\quad \Delta accu_\lambda \leftarrow -abs\left(\frac{|\mathcal{D}^\lambda_{y=1}|-|\mathcal{D}^\lambda_{y=0}|}{|\mathcal{D}|}\right)$
4: $\quad \Delta disc_\lambda \leftarrow sign\left(|\mathcal{D}^\lambda_{y=1}| - |\mathcal{D}^\lambda_{y=0}|\right) \cdot \left(\frac{|\mathcal{D}^\lambda_{S=1}|}{|\mathcal{D}_{S=1}|} - \frac{|\mathcal{D}^\lambda_{S=0}|}{|\mathcal{D}_{S=0}|}\right)$
5: $\quad \gamma \leftarrow \frac{\Delta disc_\lambda}{\Delta accu_\lambda}$
6: $\quad$ **if** $\gamma \geq 0$ **then**
7: $\quad\quad \lambda.score \leftarrow \gamma$
8: $\quad\quad \Lambda \leftarrow \Lambda \cup \{\lambda\}$
9: $\quad$ **end if**
10: **end for**
11: **return** Λ

6.3.1 Leaf Scoring

EiFFFeL borrows from [88] a simple strategy for scoring leaves according to their impact $\Delta accu_\lambda$ and $\Delta disc_\lambda$ on accuracy and discrimination respectively. Then, the ratio γ between the two is used as a score to greedily select the best leaves to be *flipped*.

We proceed as described in Alg. 5. We consider only leaves of the tree that were not flipped during previous iteration of the EiFFFeL algorithm (see subsection below). For those leaves we compute the accuracy and discrimination variation in the case of flipping the leaf prediction. We illustrate shortly the computations below, please refer to [88] for a more detailed description.

The change in accuracy $\Delta accu_\lambda$ clearly depends on the number of instances of $\mathcal{D}$ that fall into the leaf λ denoted with $\mathcal{D}^\lambda$. The training process sets the leaf prediction to the majority class among such instances. Therefore, when flipping the leaf prediction the accuracy may only decrease depending on the instances with label 1 and 0, denoted by $\mathcal{D}^\lambda_{y=1}$ and $\mathcal{D}^\lambda_{y=0}$ respectively. The difference between the size of these two sets results in the accuracy loss as computed in line 3.

The change in discrimination $\Delta disc_\lambda$ depends on the number of privileged and unpriviledged instances that fall in the leaf λ respectively denoted

by $\mathcal{D}^{\lambda}_{S=1}$ and $\mathcal{D}^{\lambda}_{S=0}$, and on their analogous on the whole dataset $\mathcal{D}_{S=1}$ and $\mathcal{D}_{S=0}$. If the leaf prediction equals 1 (favourable class), then increasing $\mathcal{D}^{\lambda}_{S=1}$ would increase the discrimination, while increasing $\mathcal{D}^{\lambda}_{S=0}$ would decrease it. The opposite holds if the prediction of the leaf equals 0 (unfavourable class). As the original leaf prediction depends on the majority of the instances between $\mathcal{D}^{\lambda}_{y=1}$ and $\mathcal{D}^{\lambda}_{y=0}$, the sign of their difference is used to correct the above contributions as computed in line 4.

The ratio $\gamma = \Delta disc_{\lambda}/\Delta accu_{\lambda}$ is positive if the flipping generates a discrimination drop, and it is large if the benefit to discrimination is larger than the harm to accuracy. If the value of γ is positive, then this is stored with the leaf λ, and λ is recorded into the set of candidate leaves Λ. The set Λ is eventually returned and exploited during the iterations of EiFFFeL.

Algorithm 6 EiFFFeL-TF (Tree-based Flipping)

Input: Random Forest classifier $\mathcal{F}$, Discrimination Constraint $\epsilon \in [0, 1]$,Accuracy Constraint $\alpha \in [0, 1]$,Training Dataset $\mathcal{D}$,Sensitive feature S
Output: Fair Random Forest $\mathcal{F}$

1: **for all** $t \in \mathcal{F}$ **do**
2: $t.flipped \leftarrow false$
3: **for all** $\lambda \in t$ **do**
4: $\lambda.flipped \leftarrow false$
5: **end for**
6: **end for**
7: $accu^{*}_{\mathcal{F}} \leftarrow \frac{|\mathcal{D}_{y=1} \wedge \mathcal{F}(\boldsymbol{x})=1| + |\mathcal{D}_{y=0} \wedge \mathcal{F}(\boldsymbol{x})=0|}{|\mathcal{D}|}$
8: $\Delta accu_{\mathcal{F}} \leftarrow 0$
9: **while** $|\{t \in \mathcal{F} \mid \neg t.flipped\}| > 0 \wedge disc_{\mathcal{D},S,\mathcal{F}} > \epsilon \wedge \Delta accu_{\mathcal{F}} < \alpha$ **do**
10: $t^{\dagger} \leftarrow \text{argmax}_{t \in \mathcal{F}}\ disc_{\mathcal{D},S,t}$
11: $\Lambda \leftarrow$ SCORE_LEAVES$(t^{\dagger}, \mathcal{D}, S)$
12: **if** $\Lambda \neq \emptyset$ **then**
13: **for all** $\lambda \in \Lambda$ **do**
14: $\lambda.pred = 1 - \lambda.pred$
15: **end for**
16: **end if**
17: $t^{\dagger}.flipped = true$
18: $accu_{\mathcal{F}} \leftarrow \frac{|\mathcal{D}_{y=1} \wedge \mathcal{F}(\boldsymbol{x})=1| + |\mathcal{D}_{y=0} \wedge \mathcal{F}(\boldsymbol{x})=0|}{|\mathcal{D}|}$
19: $\Delta accu_{\mathcal{F}} \leftarrow accu^{*}_{\mathcal{F}} - accu_{\mathcal{F}}$
20: **end while**
21: **return** $\mathcal{F}$

6.3.2 EiFFFeL Leaf Flipping Strategies

By exploiting the scoring technique discussed before, we propose two strategies to choose which trees and which leaves in those trees to flip.

The first strategy, named *Tree-based Flipping*, is illustrated in Alg. 6. During each iteration of EiFFFeL, the tree $t^{\dagger}$ with the largest discrimina-

tion degree is greedily selected: this is the best tree to be attacked in order to significantly reduce the discrimination of the full forest. Then, we use the previous scoring technique to find the set of leaves Λ in $t^\dagger$ that should be relabeled. If Λ is not empty, the predictions $\lambda.pred$ of such leaves will be flipped. Then, the whole tree is marked as already flipped. The selection is repeated by considering only the remaining non-flipped trees. The algorithm ends when all trees have been flipped, or when the desired discrimination ϵ is achieved, or when tolerated accuracy drop α is met. Note that the accuracy drop is computed by comparing the accuracy of the original forest with the accuracy of the current forest after the flipping step.

Algorithm 7 EiFFFeL-LF (Leaf-based Flipping)

Input: Random Forest classifier $\mathcal{F}$, Discrimination Constraint $\epsilon \in [0, 1]$, Accuracy Constraint $\alpha \in [0, 1]$, Training Dataset $\mathcal{D}$, Sensitive feature S
Output: Fair Random Forest $\mathcal{F}$

for all $t \in \mathcal{F}$ **do**
  $t.flipped \leftarrow false$
  for all $\lambda \in \mathcal{T}$ **do**
    $\lambda.flipped \leftarrow false$
  end for
end for
$accu^*_{\mathcal{F}} \leftarrow \frac{|\mathcal{D}_{y=1} \wedge \mathcal{F}(\boldsymbol{x})=1| + |\mathcal{D}_{y=0} \wedge \mathcal{F}(\boldsymbol{x})=0|}{|\mathcal{D}|}$
$\Delta accu_{\mathcal{F}} \leftarrow 0$
while $|\{t \in \mathcal{F} \mid \neg t.flipped\}| > 0 \wedge disc_{\mathcal{D},S,\mathcal{F}} > \epsilon \ \wedge \ \Delta accu_{\mathcal{F}} < \alpha$ **do**
  $t^\dagger \leftarrow \text{argmax}_{t \in \mathcal{F}} \ disc_{\mathcal{D},S,t}$
  $\Lambda \leftarrow$ SCORE_LEAVES$(t^\dagger, \mathcal{D}, S)$
  if $\Lambda = \emptyset$ **then**
    $t.flipped \leftarrow true$
  else
    $\lambda^\dagger \leftarrow \text{argmax}_{\lambda \in \Lambda} \ \lambda.score$
    $\lambda^\dagger.flipped = true$
    $\lambda^\dagger.pred = 1 - \lambda.pred$
    $accu_{\mathcal{F}} \leftarrow \frac{|\mathcal{D}_{y=1} \wedge \mathcal{F}(\boldsymbol{x})=1| + |\mathcal{D}_{y=0} \wedge \mathcal{F}(\boldsymbol{x})=0|}{|\mathcal{D}|}$
    $\Delta accu_{\mathcal{F}} \leftarrow accu^*_{\mathcal{F}} - accu_{\mathcal{F}}$
  end if
end while
return $\mathcal{F}$

Such tree-based strategy might be too aggressive, as it immediately flips all the candidate leaves of the selected tree. Indeed, only a few leaves may be sufficient to meet our discrimination and accuracy requirements. Therefore we propose a second strategy, named *Leaf-Based Flipping*, illustrated in Alg 7. As in the former strategy, we first select the tree $t^\dagger$ with the largest discrimination. Then we use the leaf scoring technique to find a set of candidate leaves from $t^\dagger$. If such set is empty, e.g., because they were already

flipped or they cannot improve the discrimination, the full tree is marked as flipped and the procedure is repeated on the remaining non-flipped trees. Otherwise, the leaf with the largest score $\lambda^{\dagger}$ is selected, marked as flipped, while its prediction is inverted. The process is repeated until all trees have been flipped, or the desired discrimination ϵ is achieved, or the tolerated accuracy drop α is met.

We argued that the Leaf-based approach exploits a more fine-grained tuning of the given forest, and therefore it can achieve the desired accuracy with a smaller set of alterations. Indeed, reducing the flips applied to the forest provides a larger accuracy.

6.4 Experimental Evaluation

6.4.1 Datasets.

We use datasets publicly available, widely used in fairness literature, concerning binary classification. We pre-process them using one-hot encoding for categorical features, binary encoding of sensitive feature, and removing of instances containing missing values. Moreover, we use an 80/20 training/test split.

- *Adult*: The Adult UCI income dataset [53] contains 14 demographic attributes of more than 45,000 individuals, together with class labels which states whether their income is higher than $50K or not. As sensitive attribute, we use the *gender* encoded as a binary attribute 1/0 for male/female respectively.

- *COMPAS*: The COMPAS dataset [5] contains data collected on the use of the COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) risk assessment tool. It contains 13 attributes of more than 7,000 convicted criminals, with class labels that state whether or not the individual reoffend within 2 years of her/his most recent crime. We use *race* as sensitive attribute encoded as a binary attribute 1/0 for Others/African-American, respectively.

- *Bank*: Bank marketing dataset [120] contains 16 features about 45,211 clients of direct marketing campaigns of a Portuguese banking institution. The goal is to predict whether the client will subscribe or not to a term deposit. We consider the *age* as sensitive attribute, encoded as a binary attribute 1/0, indicating whether the clientâs age is $\geq$25 or $<$25, respectively.

6.4.2 Experimental Setup.

We apply our proposed EiFFFeL algorithm over a Random Forest classifier with/without the fair splitting of nodes for individual base trees, and evaluate the performance of the algorithms in terms of model accuracy and discrimination over the three datasets mentioned above.

We compare our results against a DFRF classifier (*Distributed fair random forest*) [57], which only includes fair decision trees within the forest. The setting of hyper-parameters of DFRF are the same as the one described in the original work. We use fair split and sensitive feature as hyper-parameters, along with tree number and maximum tree depth. Additionally, we also compare our results against EOP (*Equalized Odds Post-processing*) [70, 138], a random forest classifier with the same number of base estimators and maximum depth as ours. After training and achieving the desired equalized odd we score the discrimination in the same approach we used for our experiments.

In conclusion, the comparisons of accuracy and discrimination values are among the following methods:

- DFRF [57],[1] which adds base trees to the forest only if they are fair;
- EOP [70, 138],[2] which adopts a post-processing method based on achieving equalized odds requiring the privileged and unprivileged groups to have the same false negative rate and same false positive rate;
- our implementations of EiFFFeL-TF and EiFFFeL-LF algorithms, whose post-processing is applied to a plain Random Forest of trees;
- the same post-processing techniques of EiFFFeL-TF and EiFFFeL-LF applied on top of a random forest with discrimination aware base trees [88]. These versions are denoted by EiFFFeL-TF* and EiFFFeL-LF*.

Finally, the *baseline accuracy* and *discrimination* used to compare the various methods are the ones obtained by a plain Random Forest of trees, trained on the three datasets through the scikit-learn algorithm Random Forest Classifier[3]. The various EiFFFeL methods are applied to the same baseline Random Forest.

[1] https://github.com/pjlake98/Distributed-Fair-Random-Forest

[2] https://github.com/Trusted-AI/AIF360/blob/master/aif360/algorithms/postprocessing/calibrated_eq_odds_postprocessing.py

[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

6.4.3 Results.

Tables 6.2), 6.3), and 6.4) compare the decreases in accuracy and discrimination, obtained by the different algorithms, on the three datasets with respect to the baseline results obtained by plain Random Forest models.

			DFRF		EOP		EiFFFeL-TF		EiFFFeL-LF		EiFFFeL-TF*		EiFFFeL-LF*	
			ΔAccu↓	ΔDisc ↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑
Adult	ϵ	0.01	7(0.78)	18(0.02)	2(0.83)	7(0.13)	4(0.81)	19(0.01)	4(0.81)	20(0)	6(0.79)	15(0.05)	3(0.82)	17(0.03)
		0.05	3(0.82)	13(0.07)			3(0.82)	16(0.04)	2(0.83)	15(0.05)	6(0.79)	16(0.04)	3(0.82)	16(0.04)
		0.10	4(0.81)	15(0.05)			2(0.83)	12(0.08)	1(0.84)	12(0.08)	1(0.84)	12(0.08)	2(0.83)	10(0.1)
		0.15	2(0.83)	10(0.1)			0(0.85)	8(0.12)	0(0.85)	9(0.11)	0(0.85)	7(0.13)	0(0.85)	7(0.13)

Table 6.2: *Comparison of accuracy reduction and discrimination decrease on Adult dataset with respect to baseline accuracy of 0.85 and discrimination 0.2. Along with ΔAccu and ΔDisc, we also report (within parentheses) the final accuracy and discrimination values obtained.*

			DFRF		EOP		EiFFFeL-TF		EiFFFeL-LF		EiFFFeL-TF*		EiFFFeL-LF*	
			ΔAccu↓	ΔDisc ↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑
Bank	ϵ	0.01	9(0.73)	13(0.05)	0(0.82)	14(0.04)	7(0.75)	17(0.01)	10(0.72)	15(0.03)	8(0.74)	14(0.04)	5(0.77)	10(0.08)
		0.05	4(0.78)	11(0.07)			3(0.79)	13(0.05)	8(0.74)	14(0.04)	8(0.74)	13(0.05)	5(0.77)	13(0.05)
		0.10	4(0.78)	6(0.12)			2(0.80)	10(0.08)	1(0.81)	7(0.11)	7(0.75)	9(0.09)	4(0.78)	8(0.10)
		0.15	4(0.78)	9(0.09)			0(0.82)	4(0.14)	0(0.82)	4(0.14)	6(0.76)	5(0.13)	2(0.80)	5(0.13)

Table 6.3: *Comparison of accuracy reduction and discrimination decrease on Bank dataset with respect to baseline accuracy of 0.82 and discrimination 0.18. Along with ΔAccu and ΔDisc, we also report (within parentheses) the final accuracy and discrimination values obtained.*

Recall that increasing ϵ, we reduce the space for improving discrimination, and as a side effect, we preserves the baseline accuracy. Indeed, in these experiments the accuracy constraint α was set to 1, so that there are no limits in the possible accuracy reduction ΔAccu. This allows us to compare our methods against DFRF and EOP, which do not have this α constraint. Indeed, EOP is completely parameter free, and does not support neither α nor ϵ.

In more details, Tables 6.2), 6.3), and 6.4) report, for different values of ϵ in the set $\{0.01, 0.05, 0.10, 0.15\}$, the ΔAccu and ΔDisc values obtained by the different algorithms, where ΔAccu and ΔDisc indicate the *absolute difference* in accuracy and discrimination w.r.t. the baselines. Indeed, we express these Δ absolute differences in points/hundredths (each point corresponds to $1/100$). Note that while greater values of ΔDisc are better, greater values of ΔAccu are worse, so a trade-off is needed. In addition,

			DFRF		EOP		EiFFFeL-TF		EiFFFeL-LF		EiFFFeL-TF*		EiFFFeL-LF*	
			ΔAccu↓	ΔDisc ↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑	ΔAccu↓	ΔDisc↑
COMPAS	ϵ	0.01	11(0.58)	28(0.02)	4(0.65)	5(0.25)	25(0.44)	29(0.01)	5(0.64)	26(0.04)	9(0.60)	29(0.01)	1(0.68)	7(0.23)
		0.05	5(0.64)	13(0.17)			12(0.57)	28(0.02)	5(0.64)	22(0.08)	9(0.60)	28(0.02)	1(0.68)	7(0.23)
		0.10	4(0.65)	7(0.23)			7(0.62)	21(0.09)	5(0.64)	21(0.09)	1(0.68)	21(0.09)	1(0.68)	7(0.23)
		0.15	2(0.67)	6(0.24)			1(0.68)	19(0.11)	2(0.67)	15(0.15)	0(0.69)	16(0.14)	1(0.68)	7(0.23)

Table 6.4: *Comparison of accuracy reduction and discrimination decrease on Compas dataset with respect to baseline accuracy of 0.69 and discrimination 0.3. Along with ΔAccu and ΔDisc, we also report (within parentheses) the final accuracy and discrimination values obtained.*

besides the absolute Δ values, we also report (within parentheses) the final values for accuracy and discrimination score obtained by the various techniques.

For example, for the *Adult* dataset (Table 6.2) and $\epsilon = 0.01$, EiFFFeL-TF can reach a very low discrimination score of 0.01, by only losing 4 points in accuracy (from 0.85 of the baseline to 0.81). In comparison, the best results we can obtain with DFRF in terms of discrimination is a score of 0.02, by losing 7 points in accuracy (from 0.85 of the baseline to 0.78). Overall, our algorithms are capable of reducing discrimination better than DFRF while maintaining the same accuracy. Also EOP does not work well, as the best discrimination score is only 0.13, by losing 2 points in accuracy. In addition, using $\epsilon = 0.15$ for EiFFFeL-TF and EiFFFeL-LF (also EiFFFeL-TF* and EiFFFeL-LF*), we can decrease the baseline discrimination of about $7 - 9$ points, by keeping the same accuracy of the baseline.

Results for the Bank dataset (Table 6.3) shows that EiFFFeL-TF can reach for $\epsilon = 0.01$ the desired discrimination score, but losing 7 points in accuracy (from 0.82 to 0.75), whereas DFRF has worse discrimination score of 0.05 and a worse accuracy of 0.73. EOP does not lose any accuracy for lowering the discrimination score by 14 points to 0.04.

Finally, considering the results obtained for the COMPAS dataset (Table 6.4), we observe in some cases DFRF works pretty well, but always one of our algorithms gets better results. For example, for $\epsilon = 0.01$, the best discrimination score of 0.01 is obtained by EiFFFeL-TF*, by only losing 9 points in accuracy, against the 11 points lost by DFRF with a discrimination score of 0.02.

Figures 6.1, 6.2, and 6.3 report the same data of the above tables, where we varied the discrimination constraint $\epsilon = \{0.01, 0.05, 0.1, 0.5\}$, with no constraints on accuracy. The results obtained by EOP are not plotted, as its results are always worse than the competitors and do not vary with ϵ.

Specifically, Figure 6.1 reports results for the three datasets, and aims at showing the tradeoff of accuracy vs. discrimination when we vary ϵ. Recall that we are interested in achieving low discrimination and high accuracy, and thus the best tradeoff corresponds to points of curves falling in the top-left quadrant.

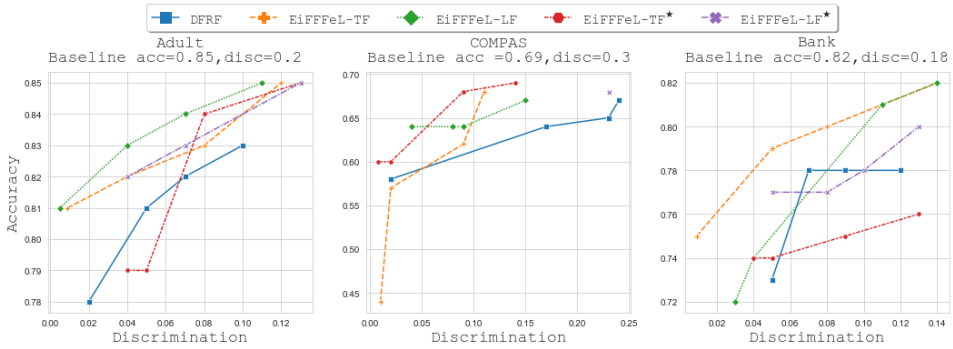

Figure 6.1: *Accuracy vs. discrimination scores after relabeling for constraints* $\epsilon = 0.01, 0.05, 0.1, 0.15$.

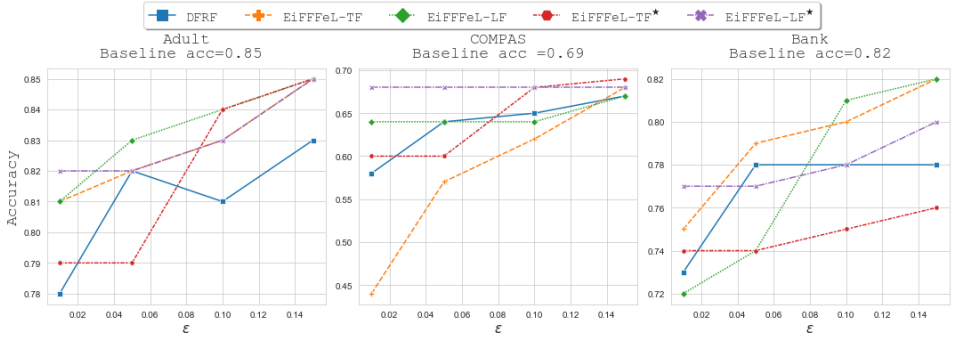

Figure 6.2: *Accuracy of the model as a function the* ϵ *constraint.*

First, we highlight that DRFR performs poorly on most settings compared to the proposed EIFFFeL variants. On the Adult dataset, EIFFFeL-LF dominates the other algorithms for all values of ϵ and achieves the desired or better discrimination with the largest accuracy. To appreciate the strict relationships between of the setting of ϵ and the discrimination/accuracy obtained, the reader can refer to the other two Figures 6.1 and 6.2.

Returning to Figure 6.1, the COMPAS EIFFFeL-LF provides the best performance together with EIFFFeL-TF*. This is the only dataset where EIFFFeL-TF* provides interesting performance, and thus the discrimina-

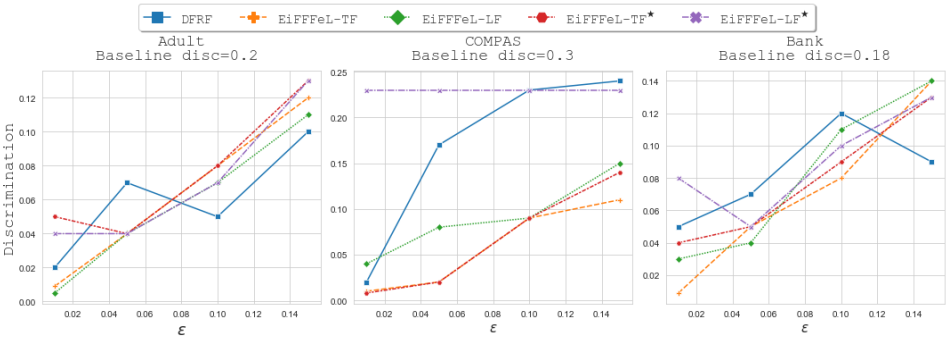

Figure 6.3: *Discrimination scores as a function of the ϵ constraint.*

				EiFFFeL-TF		EiFFFeL-LF		EiFFFeL-TF⋆		EiFFFeL-LF⋆	
				Accu	Disc	Accu	Disc	Accu	Disc	Accu	Disc
Adult	ϵ=0.01	α	0.01	0.83	0.09	0.84	**0.08**	0.84	0.10	0.84	0.11
			0.02	0.83	0.09	0.83	**0.06**	0.83	0.10	0.83	0.07
			0.03	0.82	**0.04**	0.82	0.05	0.82	0.07	0.82	**0.04**
			0.05	0.81	0.01	0.81	**0.00**	0.80	0.08	0.82	0.03

Table 6.5: *Accuracy and discrimination scores on the Adult dataset for $\epsilon = 0.01$ and $\alpha = 0.01, 0.02, 0.03, 0.05$. The baseline accuracy and discrimination score are 0.85 and 0.2, respectively.*

tion aware splitting at training time provides some benefits. We also highlight that when using $\epsilon = 0.15$ (see Figure 6.3) the algorithm DFRF only gets a discrimination score of 0.25. Note that EIFFFeL-LF⋆ is not able to provide better performance when varying ϵ, thus resulting in a constant curve.

Finally, on the Bank Dataset, EIFFFeL-TF and EIFFFeL-LF achieve the best results, with an advantage for EIFFFeL-TF for smaller values of ϵ. Finally, the results show how we can obtain the desired discrimination degree with a limited drop in accuracy. Overall, the proposed EIFFFEL algorithm outperforms the competitor DFRF, and, on average, it is advisable to avoid the discrimination aware node splitting. We believe that working only at post-processing allows us to exploit a richer set of trees grown, by exploring a larger and unconstrained search space.

The effect of varying the discrimination constraint ϵ without constraining accuracy can be observed in Figure 6.2, where we discover that lower discrimination is achieved with large accuracy reduction. This is due to the fact that a small discrimination threshold allows our flipping strategies to force the change of many leaves, thus changing more the classification decision regions, with a final lower accuracy. However our approach of se-

lecting potential leaves to relabel seems better than training random forest with only fair trees. In addition, training and then rejecting trees (because they are not fair) makes longer the training of the forest, particularly when we fail often in finding fair trees.

Finally, Figure 6.3 contrasts the discrimination measured on the test set against the desired discrimination constraint ϵ. Clearly, the twos do not always match. In particular, DFRF has an unstable behaviour, meaning that filtering the tree to be added to the forest is not the best option. Conversely, EiFFFeL-TF and EiFFFeL-LF provide a much more stable behaviour.

We also discuss the results of other experiments, aiming to evaluate the effects of different values for the α constraints. Note that only the EiFFFeL algorithms support the α parameter, so we cannot reports any results for the competitors DFRF and EOP. Specifically, Table 6.5 reports results relative to the Adult dataset, where, for a fixed $\epsilon = 0.01$, we vary the α constraint over the expected accuracy, with values ranging in the set {0.01,0.02,0.03,0.05}. For each α value, we show in bold the best results in terms of discrimination score. We observe that the accuracy constraint α has an indirect impact on the final discrimination score obtained. Using EiFFFeL-LF with $\alpha = 0.01$, the loss in accuracy is 1 point as expected, while the baseline discrimination score decreases by more than half (from 0.2 to 0.08). Furthermore, as the α value increases, discrimination score decreases further. With $\alpha = 0.05$, EiFFFeL-LF is able to reduce by 4 points the final accuracy, by also achieving a discrimination score of 0, thus showing the power of our method in achieving a very good trade-off between accuracy and discrimination.

6.5 Summary

In this chapter, we deal with fairness in machine learning, and specifically in binary classifiers trained by a Random Forest algorithm. We are interested in group fairness, so as to mitigate the effect of bias against specific groups, which may comes from biased training datasets or algorithm design.

We develop EiFFFeL, a novel post-process approach, which maintains good predictive performance of the trained model with a low discrimination score. Our approaches flips the label of selected leaf (or leaves) of base trees in a random forest by using two algorithms: (i) an aggressive tree-based approach, which flips all candidate leaves of a tree, and (ii) a leaf-based strategy which only flips the label of the most discriminative leaf of a tree. Both strategies are implemented by considering accuracy and

discrimination constraints. Indeed, the constraints are used to control the minimum accuracy decrease we can tolerate in order to achieve the desired discrimination value. In addition, we have tested the impact of incorporating discrimination aware node split strategies for base trees of the forest, by adding discrimination gain value in their node splitting criterion [88].

By using three publicly available datasets, our experimental results show that effective non-discriminative models can be obtained, while keeping a strict control over both accuracy and discrimination level. Compared to state-of-the-art methods, which adopt both in-process and post-process bias mitigation approaches, EiFFFeL resulted to produce the most accurate models that also exhibit the best levels of fairness.

CHAPTER 7

Conclusion

Machine learning models are increasingly used in many areas such as healthcare, energy, transportation, and education, becoming essential element of decision-making processes–sometimes to the point where models replace humans and make decisions autonomously. Given the vast areas of applications and their input domain machine learning is applied to, it is expected models exhibit errors since it is impossible to have sample representative training data that capture the input domain. As a result, machine learning models are vulnerable to attackers that may methodically exploit the attack surface. In addition to security vulnerability, there is also a concern for unfair treatment and discrimination by machine learning models.

In recent years the debate over responsible Artificial Intelligence increases where arguments how to achieve *Robustness* and *Fairness* while using a machine learning model, and a framework how to satisfy them together. Robust and fair training are desired properties in many applications, such as fraud detection, combining and satisfying both properties has become challenging and a debating topic among researchers in both areas.

In this thesis, we focused on mitigating unfairness and adversarial attacks in machine learning, introducing the concepts in Chapter 1.

Chapter 2 provides background information on machine learning, par-

ticularly on decision tree and decision tree ensembles. In this chapter we also provide fundamentals of adversarial machine learning, covering attack surface, knowledge of an attacker, and attackers goal and timing; adding general defensive mechanisms for adversarial attacks. In addition, we go through the sources of unfairness, how to measure fairness, and unfairness mitigation methodologies while studying fair machine learning.

In Chapter 3 and Chapter 4, we explored state-of-the-art works on adversarial and fair machine learning targeting decision tree and tree ensembles. Chapter 3 study the vulnerabilities, attack strategies, and defense mechanisms proposed, and Chapter 4 summarize pre, in, and post-processing unfairness mitigation strategies.

In Chapter 5 focusing on evasion attacks, we proposed a novel decision tree learning algorithm,TREANT, which is resilient against evasion attacks. At each step of tree construction our algorithm minimizes evasion-aware loss function which captures the attacker's goal of maximizing prediction error allowing robust model. TREANT model an attacker with asymmetric changes and different constraints, with a "budget" to spend on changing data points, which allows robustness on a different level of attacker strength. Our experiments, conducted on publicly available datasets, confirm that TREANT produces accurate tree ensembles, which are extremely robust against evasion attacks, and compared to the state of the art, TREANT exhibits a significant improvement. TREANT is published on Data Mining and Knowledge Discovery (DAMI), and also presented in the Journal Track of ECML/PKDD 2020.

In Chapter 6 dealing with unfairness in machine learning, we presented EiFFFeL:Enforcing Fairness in Forests by Flipping Leaves, a post-processing unfairness mitigation approach in which we flip a selected leaves of a tree dealing with group fairness. Using a tree-based and leaf-based approach for selecting a potential leaves of a tree to flip, our algorithm finds a trade-off between user define discrimination level and accuracy drop. By using publicly available datasets, our experimental results show that effective non-discriminative models can be obtained, while keeping a strict control over both accuracy and discrimination level. Compared to state-of-the-art methods, which adopt both in-process and post-process bias mitigation approaches, EiFFFeL resulted to produce the most accurate models that also exhibit the best levels of fairness. EiFFFel is published on Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing.

Finally, we want to highlight promising and challenging research areas for future work. We summarize the main identified ones in the following sections. First, we point out the future works related to Chapter 5 and

6. Then we will discuss research directions in achieving robustness and fairness together: to make fairness more robust(*robust fair training*) and to integrate robust and fair training in equal terms.

Extension to Current Research works

In Chapter 5, we add evasion aware decision tree in to ensemble by training individual base trees. Revising this approach to make the decision tree construction aware of its deployment inside an ensemble would exploit the information that the currently grown ensemble is particularly strong or weak against some classes of attacks to guide the construction of the next member of the ensemble. At the same time, we want to explore ways to relax the restriction that each attacked feature is only tested once on each path of the decision trees, without sacrificing the soundness and scalability of the construction. It is also intriguing to explore further applications and extensions of our proposed threat model: for example, we consider to take advantage of work on inverse classification [99] to express dependencies between different features, e.g., features which cannot be manipulated, but are computed as a function of other corrupted features.

In chapter 6 we consider group fairness only, and how to extend it to individual fairness is understudied. In the future, we will extend this to other fairness definitions,e.g., individual fairness; this is also entails how to approach multiple fairness constraints while they may not be compatible to satisfy together. In addition, the effect of multiple sensitive features in relation to discrimination and accuracy needs to be explored, and we want to evaluate our approach to other tree ensemble learning methods. Hence, extending this work in both directions.

Convergence of Robustness and Fairness

Trustworthy algorithms has become a crucial component of modern machine learning frameworks [33, 147]. Making AI systems fair, robust, interpretable, and transparent have become a main research focus for big technology companies like Google[1], Microsoft[2], and IBM[3].

As fairness and robustness are two critical components of trustworthy AI, and addressing both together has become a focal point to it. Fairness entails learning an unbiased model, whereas robustness is learning a resilient

[1] https://ai.google/responsibilities/responsible-ai-practices

[2] https://www.microsoft.com/en-us/ai/responsible-ai?activetab=pivot1%3Aprimaryr6

[3] https://research.ibm.com/teams/trusted-ai

model from tainted data, and it is well recognized that addressing only one of these may have a negative impact on the other. Improving fairness in a model while ignoring robustness may led to unwanted fairness-accuracy tradeoff [146]; and focusing on robustness only may have an adverse affect on fairness. In this regard,two research directions has been proposed to combine robustness and fairness together [101]: Obtaining robustness and fairness objective together and make fairness more robust.

Research works in [88, 141] used fair-splitting criterion to improve discrimination metric in decision trees and decision tree forests. By taking the approach of achieving robustness and fairness together, we want to explore training a decision tree which is robust-using adversarial training, and fair-using fair-splitting criterion.

Robust and Fair split

One of the most known criterion for splitting decision tree node is Gini impurity. Decision trees greedily trained using this criterion are vulnerable to adversarial attacks and produce unfair decisions. To make this training robust [171] uses adversarial Gini impurity and propagate attacked samples in to child nodes for a single class, while [35] also uses the same strategy but move samples in both direction of two classes. Hence, finding the optimal split which more accurately represent a potential adversarial attack.

In the future, we want to combine the adversarial Gini impurity with a splitting criterion that encourages fair splits. For example, by subtracting the Gini gain of a protected attribute from adversarial Gini gain on a node, the score function will stimulate a more robust and fair split at the node. Hence, the convergence of fairness and robustness.

The above solution is one of the methods for converging fairness and robustness together for decision tree algorithm, but not the only one. Hence, we also want to investigate the direction of making fair training more robust and robust training more fairer. For example, we want to extend our evasion aware robust training TREANT in Chapter 5 and post-processing method EiFFFeL in Chapter 6 to include fairness and robustness, respectively; or pursue any other work we find on the way.

Bibliography

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69. PMLR, 2018.

[2] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1418–1426, California, USA, 2019. AAAI Press.

[3] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAIâ16, page 1452â1458, Phoenix, Arizona, 2016. AAAI Press.

[4] Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 12997–13008, Vancouver, Canada, 2019. NIPS.

[5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. propublica (2016), 2016.

[6] Ricardo Baeza-Yates. Bias on the web. *Communications of the ACM*, 61(6):54–61, 2018.

[7] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2, 2017.

[8] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

[9] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[10] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[11] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[12] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin IP Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Security evaluation of support vector machines in adversarial environments. In *Support Vector Machines Applications*, pages 105–153. Springer, 2014.

[13] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems for robust classifier design in adversarial environments. *Int. J. Machine Learning & Cybernetics*, 1(1-4):27–41, 2010.

[14] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, 2014.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *ACML Asian conference on machine learning*, pages 97–112, 2011.

[16] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1807–1814, 2012.

[17] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[18] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[19] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

[20] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[21] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

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

[23] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *International Conference on Learning Representations*, 2017.

[24] Tim Brennan, William Dieterich, and Beate Ehret. Evaluating the predictive validity of the compas risk and needs assessment system. *Criminal Justice and behavior*, 36(1):21–40, 2009.

[25] Thomas Brunner, Frederik Diehl, Michael Truong Le, and Alois Knoll. Guessing smart: Biased sampling for efficient black-box adversarial attacks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4957–4965, 2019.

[26] Ajay Byanjankar, Markku Heikkilä, and Jozsef Mezei. Predicting credit risk in peer-to-peer lending: A neural network approach. In *2015 IEEE symposium series on computational intelligence*, pages 719–725. IEEE, 2015.

[27] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. *Advances in neural information processing systems*, 30, 2017.

[28] Stefano Calzavara, Pietro Ferrara, and Claudio Lucchese. Certifying decision trees against evasion attacks by program analysis. In *European Symposium on Research in Computer Security*, pages 421–438. Springer, 2020.

[29] Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. Adversarial training of gradient-boosted decision trees. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2429–2432, 2019.

[30] Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. Adversarial training of gradient-boosted decision trees. In Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu, editors, *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 2429–2432. ACM, 2019.

[31] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, pages 1–31, 2020.

[32] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE symposium on security and privacy S&P*, pages 39–57, 2017.

[33] Hongyan Chang, Ta Duy Nguyen, Sasi Kumar Murakonda, Ehsan Kazemi, and Reza Shokri. On adversarial bias and the robustness of fair machine learning. *arXiv preprint arXiv:2006.08669*, 2020.

[34] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.

[35] Hongge Chen, Huan Zhang, Duane S. Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In *International Conference on Machine Learning ICML*, pages 1122–1131, 2019.

[36] Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane Boning, and Cho-Jui Hsieh. Robustness verification of tree-based models. In *Advances in Neural Information Processing Systems*, pages 12317–12328, Vancouver,Canada, 2019. Neural Information Processing Systems.

[37] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[38] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[39] Weilun Chen, Zhaoxiang Zhang, Xiaolin Hu, and Baoyuan Wu. Boosting decision-based black-box adversarial attacks with random sign flip. In *European Conference on Computer Vision*, pages 276–293. Springer, 2020.

[40] Yizheng Chen, Shiqi Wang, Weifan Jiang, Asaf Cidon, and Suman Jana. Cost-aware robust tree ensembles for security applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2291–2308. USENIX Association, August 2021.

[41] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *In International Conference on Learning Representations*, 2019.

[42] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. *International Conference on Learning Representations*, 2019.

[43] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2017.

[44] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[45] Asaf Cidon, Lior Gavish, Itay Bleier, Nadia Korshun, Marco Schweighauser, and Alexey Tsitkin. High precision detection of business email compromise. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1291–1307, Santa Clara, CA, USA, 2019. USENIX.

[46] T Ciodaro, D Deva, JM De Seixas, and D Damazio. Online particle detection with neural networks based on topological calorimetry information. In *Journal of physics: conference series*, volume 368, page 012030. IOP Publishing, 2012.

[47] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, pages 797–806, 2017.

[48] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[49] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

[50] Hung Dang, Yue Huang, and Ee-Chien Chang. Evading classifiers by morphing in the dark. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 119–133, 2017.

[51] Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan OâDonoghue, Daniel Visentin, et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9):1342–1350, 2018.

[52] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[53] Dheeru Dua and Casey Graff. Uci machine learning repository (2017). *URL http://archive. ics. uci. edu/ml*, 37, 2017.

[54] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[55] Cynthia Dwork, Nicole Immorlica, Adam Tauman Kalai, and Max Leiserson. Decoupled classifiers for group-fair and efficient machine learning. In *Conference on fairness, accountability and transparency*, pages 119–133. PMLR, 2018.

[56] Equivalent. Practitionerâs guide to compas core. *URL http://www.equivant.com/wp-content/uploads/Practitioners-Guide-to-COMPAS-Core-040419.pdf*, 2022.

[57] James Fantin. *A Distributed Fair Random Forest*. PhD thesis, University of Wyoming, 2020.

[58] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[59] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*, WWW â07, page 649â656, New York, NY, USA, 2007. Association for Computing Machinery.

[60] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68, 2006.

[61] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

[62] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer.

[63] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[64] Sunila Gollapudi. Deep learning for computer vision. In *Learn Computer Vision Using OpenCV*, pages 51–69. Springer, Berkeley, CA, 2019.

[65] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

[66] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations ICLR*, 2015.

[67] Vincent Grari, Boris Ruf, Sylvain Lamprier, and Marcin Detyniecki. Achieving fairness with decision trees: An adversarial approach. *Data Science and Engineering*, 5(2):99–110, 2020.

[68] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *International Conference on Learning Representations ICLR, Workshop Track Proceedings*, 2015.

[69] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[70] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[71] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, New York, 2009.

[72] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *USENIX Workshop on Offensive Technologies WOOT*, 2017.

[73] Shlomo Hershkop and Salvatore J. Stolfo. Combining email models for false positive reduction. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 98–107, 2005.

[74] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[75] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[76] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. Detecting and characterizing lateral phishing at scale. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1273–1290, Santa Clara, CA, USA, 2019. USENIX.

[77] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.

[78] Ayanna Howard and Jason Borenstein. The ugly truth about ourselves and our robot creations: the problem of bias and social inequity. *Science and engineering ethics*, 24(5):1521–1536, 2018.

[79] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *ACM Workshop on Security and Artificial Intelligence AISec*, pages 43–58, 2011.

[80] Earl B Hunt, Janet Marin, and Philip J Stone. Experiments in induction. 1966.

[81] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[82] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2137–2146. PMLR, 10–15 Jul 2018.

[83] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[84] Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. *Adversarial machine learning*. Cambridge University Press, 2018.

[85] Nathan Kallus and Angela Zhou. Residual unfairness in fair machine learning from prejudiced data. In *International Conference on Machine Learning*, pages 2439–2448. PMLR, 2018.

[86] Faisal Kamiran and Toon Calders. Classifying without discriminating. In *2009 2nd international conference on computer, control and communication*, pages 1–6. IEEE, 2009.

[87] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and information systems*, 33(1):1–33, 2012.

[88] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. Discrimination aware decision tree learning. In *2010 IEEE International Conference on Data Mining*, pages 869–874, 2010.

[89] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929. IEEE, 2012.

[90] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. Evasion and hardening of tree ensemble classifiers. In *International Conference on Machine Learning ICML*, pages 2387–2396, 2016.

[91] Asharul Islam Khan and Salim Al-Habsi. Machine learning in computer vision. *Procedia Computer Science*, 167:1444–1451, 2020.

[92] Amin Kharraz, William Robertson, and Engin Kirda. Surveylance: automatically detecting online survey scams. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 70–86, San Francisco, CA, 2018. IEEE.

[93] Marius Kloft and Pavel Laskov. Online anomaly detection under adversarial impact. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 405–412. JMLR Workshop and Conference Proceedings, 2010.

[94] Maria Konte, Roberto Perdisci, and Nick Feamster. Aswatch: An as reputation system to expose bulletproof hosting ases. *SIGCOMM Comput. Commun. Rev.*, 45(4):625â638, August 2015.

[95] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015.

[96] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[97] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *International Conference on Learning Representations*, 2017.

[98] Bum Jun Kwon, Jayanta Mondal, Jiyong Jang, Leyla Bilge, and Tudor Dumitraş. The dropper effect: Insights into malware distribution with downloader graph analytics. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS â15, page 1118â1129, New York, NY, USA, 2015. Association for Computing Machinery.

[99] Michael T. Lash, Qihang Lin, W. Nick Street, and Jennifer G. Robinson. A budget-constrained inverse classification framework for smooth classifiers. In *IEEE International Conference on Data Mining Workshops ICDMW*, pages 1184–1193, 2017.

[100] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[101] Jae-Gil Lee, Yuji Roh, Hwanjun Song, and Steven Euijong Whang. Machine learning robustness, fairness, and their convergence. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4046–4047, 2021.

[102] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, pages 1885–1893, Barcelona,Spain, 2016. NIPS.

[103] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772, Venice,Italy, 2017. IEEE.

[104] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor CM Leung. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE access*, 6:12103–12117, 2018.

[105] Pranay K Lohia, Karthikeyan Natesan Ramamurthy, Manish Bhide, Diptikalyan Saha, Kush R Varshney, and Ruchir Puri. Bias mitigation post-processing for individual and group fairness. In *Icassp 2019-2019 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 2847–2851. IEEE, 2019.

[106] Gilles Louppe, Michael Kagan, and Kyle Cranmer. Learning to pivot with adversarial networks. *Advances in neural information processing systems*, 30, 2017.

[107] Daniel Lowd and Christopher Meek. Adversarial learning. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 641–647, 2005.

[108] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *CEAS*, volume 2005, 2005.

[109] Jiajun Lu, Theerasit Issaranon, and David Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 446–454, Venice, Italy, 2017. IEEE.

[110] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE International Conference on Data Mining*, pages 301–309, 2015.

[111] Xiaojun Ma, Jinglan Sha, Dehua Wang, Yuanbo Yu, Qian Yang, and Xueqi Niu. Study on a prediction of p2p network loan default based on the machine learning lightgbm and xgboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, 31:24–39, 2018.

[112] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations ICLR*, 2018.

[113] Milad Malekipirbazari and Vural Aksakalli. Risk assessment in social lending via random forests. *Expert Systems with Applications*, 42(10):4621–4631, 2015.

[114] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. Sliq: A fast scalable classifier for data mining. In *International conference on extending database technology*, pages 18–32. Springer, 1996.

[115] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[116] Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS â17, page 135â147, New York, NY, USA, 2017. Association for Computing Machinery.

[117] David J Miller, Zhen Xiang, and George Kesidis. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108(3):402–433, 2020.

[118] Tom Mitchell. Machine learning. 1997.

[119] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 2574–2582, 2016.

[120] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[121] Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Minining and Knowledge Discovery*, 2(4):345–389, 1998.

[122] Said Nawar and Abdul Mouazen. Comparison between random forests, artificial neural networks and gradient boosted machines methods of on-line vis-nir spectroscopy measurements of soil total nitrogen and total carbon. *Sensors*, 17(10):2428, 2017.

[123] Terry Nelms, Roberto Perdisci, Manos Antonakakis, and Mustaque Ahamad. Towards measuring and mitigating social engineering software download attacks. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 773–789, Austin, TX, 2016. USENIX.

[124] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, JD Tygar, and Kai Xia. Misleading learners: Co-opting your spam filter. In *Machine learning in cyber trust*, pages 17–51. Springer, 2009.

[125] Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Steven J. Lee, Satish Rao, Anthony Tran, and J. Doug Tygar. Near-optimal evasion of convex-inducing classifiers. In *International Conference on Artificial Intelligence and Statistics AISTATS*, pages 549–556, 2010.

[126] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 427–436, 2015.

[127] Osonde A Osoba and William Welser IV. *An intelligence in our image: The risks of bias and errors in artificial intelligence*. Rand Corporation, 2017.

[128] Nicolas Papernot. *Characterizing the limits and defenses of machine learning in adversarial settings*. PhD dissertation, Penn State: The Pennsylvania State University, 2018.

[129] Nicolas Papernot and Patrick McDaniel. Extending defensive distillation. *arXiv preprint arXiv:1705.05264*, 2017.

[130] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[131] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[132] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European symposium on security and privacy EuroS&P*, pages 372–387, 2016.

[133] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE symposium on security and privacy S&P*, pages 582–597, 2016.

[134] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[135] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568, 2008.

[136] Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif. Misleading worm signature generators using deliberate noise injection. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[137] Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In *IEEE International Conference on Data Mining ICDM*, pages 488–498, 2006.

[138] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *Advances in neural information processing systems*, 30, 2017.

[139] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[140] JR Quinlan. C4. 5: Programs for machine learning morgan kaufmann san francisco. *CA, USA*, 1993.

[141] Edward Raff, Jared Sylvester, and Steven Mills. Fair forests: Regularized tree induction to minimize model bias. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 243–250, New Orleans, USA, 2018. ACM.

[142] M Zubair Rafique, Tom Van Goethem, Wouter Joosen, Christophe Huygens, and Nick Nikiforakis. It's free for a reason: Exploring the ecosystem of free live streaming services. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS 2016)*, pages 1–15, San Diego, California, 2016. Internet Society.

[143] Francesco Ranzato, Caterina Urban, and Marco Zanella. Fair training of decision tree classifiers. *arXiv preprint arXiv:2101.00909*, 2021.

[144] Francesco Ranzato and Marco Zanella. Robustness verification of decision tree ensembles. *OVERLAY@ AI* IA*, 2509:59–64, 2019.

[145] Francesco Ranzato and Marco Zanella. Genetic adversarial training of decision trees. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 358–367, 2021.

[146] Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. Fr-train: A mutual information-based approach to fair and robust training. In *International Conference on Machine Learning*, pages 8147–8157. PMLR, 2020.

[147] Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. Sample selection for fair and robust training. *Advances in Neural Information Processing Systems*, 34, 2021.

[148] Michael Roy. Cathy oâneil. weapons of math destruction: How big data increases inequality and threatens democracy. new york: Crown publishers, 2016. *College & Research Libraries*, 78(3):403, 2017.

[149] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[150] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[151] Himanshu Shekhar, Sujoy Seal, Saket Kedia, and Amartya Guha. Survey on applications of machine learning in the field of computer vision. In *Emerging Technology in Modelling and Graphics*, pages 667–678. Springer, 2020.

[152] Jack W Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 261, Washington, D.C, 1988. American Medical Informatics Association.

[153] Nedim Srndic and Pavel Laskov. Practical evasion of a learning-based classifier: A case study. In *IEEE symposium on security and privacy S&P*, pages 197–211, 2014.

[154] Harini Suresh and John V Guttag. A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002*, 2, 2019.

[155] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[156] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations ICLR*, 2014.

[157] Kymie MC Tan, Kevin S Killourhy, and Roy A Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *International Workshop on Recent Advances in Intrusion Detection*, pages 54–73. Springer, 2002.

[158] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 465–474, 2017.

[159] John Törnblom and Simin Nadjm-Tehrani. An abstraction-refinement approach to formal verification of tree ensembles. In *International Conference on Computer Safety, Reliability, and Security*, pages 301–313. Springer, 2019.

[160] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[161] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[162] Tich Phuoc Tran, Pohsiang Tsai, and Tony Jan. An adjustable combination of linear regression and modified probabilistic neural network for anti-spam filtering. In *International Conference on Pattern Recognition ICPR*, pages 1–4, 2008.

[163] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[164] David E Tyler. Robust statistics: Theory and methods, 2008.

[165] Andrew V Uzilov, Joshua M Keegan, and David H Mathews. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):173, 2006.

[166] Rhema Vaithianathan, Emily Putnam-Hornstein, Nan Jiang, Parma Nand, and Tim Maloney. Developing predictive models to support child maltreatment hotline screening decisions: Allegheny county methodology and implementation. *Center for Social data Analytics*, 2017.

[167] Tyler J VanderWeele and Miguel A Hernán. Results on differential and dependent measurement error of the exposure and the outcome using signed directed acyclic graphs. *American journal of epidemiology*, 175(12):1303–1310, 2012.

[168] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, Denver, Colorado, 1992. NIPS.

[169] Sahil Verma and Julia Rubin. Fairness definitions explained. In *2018 ieee/acm international workshop on software fairness (fairware)*, pages 1–7. IEEE, 2018.

[170] Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

[171] Daniël Vos and Sicco Verwer. Efficient training of robust decision trees against adversarial examples. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10586–10595. PMLR, 18–24 Jul 2021.

[172] Christina Wadsworth, Francesca Vera, and Chris Piech. Achieving fairness through adversarial learning: an application to recidivism prediction. *arXiv preprint arXiv:1807.00199*, 2018.

[173] Yihan Wang, Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. On lp-norm robustness of ensemble decision stumps and trees. In *International Conference on Machine Learning*, pages 10104–10114. PMLR, 2020.

[174] Gregory L Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *CEAS*. Citeseer, 2004.

[175] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *international conference on machine learning*, pages 1689–1698. PMLR, 2015.

[176] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.

[177] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[178] Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri. Robustness for non-parametric classification: A generic attack and defense. In *International Conference on Artificial Intelligence and Statistics*, pages 941–951. PMLR, 2020.

[179] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480, 2009.

[180] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970. PMLR, 2017.

[181] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. AIES '18, page 335â340, New York, NY, USA, 2018. Association for Computing Machinery.

[182] Chong Zhang, Huan Zhang, and Cho-Jui Hsieh. An efficient adversarial attack for tree ensembles. *Advances in Neural Information Processing Systems*, 33, 2020.

[183] Fuyong Zhang, Yi Wang, Shigang Liu, and Hua Wang. Decision-based evasion attacks on tree ensemble classifiers. *World Wide Web*, pages 1–21.

[184] Wenbin Zhang and Eirini Ntoutsi. Faht: An adaptive fairness-aware decision tree classifier. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1480–1486, Macao, 7 2019. International Joint Conferences on Artificial Intelligence Organization.