Improving the robustness of decision trees in security-sensitive setting

S.J.M. (Cas) Buijs

Improving the robustness of decision trees in security-sensitive setting

by

S.J.M. (Cas) Buijs

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 13, 2020 at 10:00 AM.

Student number:	4345185	
Project duration:	October 1, 2019 – August 13, 2020	
Thesis committee:	Prof. dr. ir. I. Lagendijk,	TU Delft, chair
	Dr. S. Verwer,	TU Delft, supervisor
	Dr. D. Tax,	TU Delft, committee member

TU Delft

Abstract

Machine learning is used for security purposes, to differ between the benign and the malicious. Where decision trees can lead to understandable and explainable classifications, an adversary could manipulate the model input to evade detection, e.g. the malicious been classified as the benign. State-of-the-art techniques improve the robustness by taking these adversarial attacks into account when building the model. In this work, I identify three factors contributing to the robustness of a decision tree: feature frequency, shortest distance between malicious leaves and benign prediction space, and impurity of benign prediction space. I propose two splitting criteria to improve these factors and suggest a combination with two trade-off approaches to balance the use of these splitting criteria with a common splitting criterion, Gini Impurity, in order to balance accuracy and robustness. These combinations allow building robuster models against adversaries manipulating the malicious data without considering adversarial attacks. The approaches are evaluated in a white-box setting against a decision tree and random forest, considering an unbounded adversary where robustness is measured using a L1-distance norm and the false negative rate. All combinations lead to robuster models at different costs in terms of accuracy, showing that adversarial attacks do not need to be taken into account to improve robustness. Compared to state-of-the-art work, the best approach achieves on average 3.17% better accuracy with an on average lower robustness of 5.5% on the used datasets for a single decision tree. In a random forest the best approach achieves on average 2.87% better robustness with a 2.37% better accuracy on the used datasets compared to the state-of-the-art work. The state-of-the-art work does not seem to affect all of the identified factors, which leaves room for even robuster models than currently existing.

Preface

In this thesis, I investigate whether and how decision trees can be made more robust without considering adversarial attacks to build the model. This research was performed to fulfil the requirements of the Computer Science master programme, Cyber Security specialisation, at Delft University of Technology. The evaluation of the approaches proposed in this work shows that robustness can be improved without considering adversarial attacks and, depending on the approach, performance close to that of state-of-the-art work can be achieved or even better as shown for a random forest.

There are several people I would like to thank in this preface, starting with my supervisor Sicco Verwer for accepting the responsibility of guiding me through this process, our weekly discussions in which ”change” was a recurring theme and the motivation he gave me to continue this research. Also, I would like to thank Chris Hammerschmidt for our discussions in the hallway and his guidance during the weekly group meetings.

Next, I would like to thank my family for their continuous support, prayers and their belief in me these last two years. There were many obstacles along the road but you always supported me to the best of your abilities and I am grateful for that. I am also grateful to my girlfriend, Taisiia. We have spent many evenings discussing my ideas and progress as well as the obstacles that came along. Your support was invaluable to me and has certainly contributed to me staying sane and motivated.

Thirdly, I would like to thank the many people, friends and colleagues, for their talks, discussions and their contribution to making this research an enjoyable process. Special thanks go to Daniël Vos, our in-depth discussions about decision trees and adversarial examples have certainly helped me, David Alderliesten and Marie Kegeleers, our regular meetings, walks and discussions often gave me the distraction that I needed and made the weeks shine a little brighter, Simone van Veen, our meetings and talks kept me motivated during the previous research and the lessons shared still guided me through this one, and Clinton Cao, our discussions led to new insights and were always a pleasant break from research.

Last but not least, I would like to thank Sandra Wolff and the other thesis students from the research group. It would not have been an optimal working environment on the 6th floor without all of our talks, sharing of ideas and your encouragement.

I sincerely hope you will enjoy reading this thesis and ”may the odds be ever in your favor”[1].

Cas Buijs
Delft, August 2020

[1] From the film *The Hunger Games*, 2012

Contents

1 Introduction

Nearly 20% of the user computers faced at least one malware-class web attack in 2019 [4]. In the same year, an increase of 33% in Internet-of-Things (IoT) malware attacks was discovered by Sonicwall [17], a 14% increase in uniquely detected malicious objects by Kaspersky [2, 4] and a 13% increase in threat detections against businesses by Malwarebytes [7]. On top of that, 62% respondents of a survey by ISACA stated their cybersecurity team to be understaffed up to some degree [6]. An ever-developing world, more-and-more dependent on technology where the threats are growing alongside their economical costs [8]. Home computers, smartphones and the rise of IoT brought new possibilities but an increasing number of machines, the data flowing between them and understaffed cybersecurity teams make it harder to defend against these potential threats.

A cyberattack does not happen in an instant but takes place in phases, which come together in the Intrusion Kill Chain (IKC) model [39] as can be seen in figure 1.1. In order to prevent a cyberattack from succeeding, measures can be taken to reduce the probability of an adversary reaching the next phase in the kill chain. For example, before a computer gets infected by malware the adversary goes through the delivery phase. In this phase the adversary tries to get the malware at its destination, e.g. through e-mail attachments or USB devices. To lower the likelihood of this attack to succeed, each e-mail attachment could be checked before forwarded to the recipient or a policy could be put in place that prohibits the use of USB devices. By taking measures at each phase of the IKC model, the likelihood for an adversary to pull-off a successful cyberattack decreases. It would not be possible for cybersecurity teams to manually perform all these tasks, e.g. checking all e-mail attachments that flow in and out of organisations , but algorithms, e.g. deep learning, could take over part of the workload for these repetitive tasks and there exists an increasing willingness in industry to make use of and invest in these technologies [43].

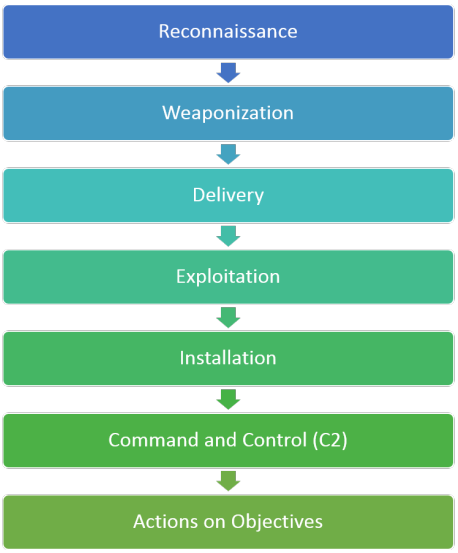


Figure 1.1: Phases of the Intrusion Kill Chain from [3].

The identification of cyberattacks could, at least in some cases, be translated into a machine learning (ML) classification problem. Algorithms, known as classifiers, learn characteristics of classes from training data, which they can later use to identify the corresponding classes in data not seen before. For example, consider a cyberattack where an adversary attempts to break into an organisation's network, phase 3 of the IKC model. This could be translated into a classification problem, where a classifier has to separate benign network traffic, the benign class, from malicious network traffic, the malicious class. A classifier could learn the characteristics of the benign class from traffic collected in an earlier

stage and those of the malicious class from traffic shared by others in the industry. Using the learned characteristics from the training data, it could classify traffic currently flowing through the network in order to detect the adverary's attempts. Figure 1.2 shows an overview of this process.

Deep learning is a ML concept referring to a group of techniques using Artificial Neural Networks to solve classification problems, e.g. Deep Neural Network (DNN). The idea stems from the working of animal brains, where a large number of cells communicate with each other in order to learn. These neural networks consist of layers of cells that learn to perform certain tasks, e.g. identifying different classes, by considering examples from training data. There are several applications for deep learning in traditional cybersecurity scenarios, like classifying malicious network traffic and botnet detection, but also in more modern situations, like detection of electricity theft [9]. Even supermarkets started experimenting with deep learning to detect known shoplifters in order to prevent further theft [5, 26]. It is clear that deep learning is applicable in a large number of fields to take over part of the cybersecurity team's workload. However these techniques are too complex to be well-understood, an aspect that goes against current societal trends.

Nowadays, society demands a better understanding of the processes that affect their lives, especially those used in automated decision-making. In the European Union, this wish of society was translated into a new regulation, known as the General Data Protection Regulation (GDPR). It provides European citizens with the right to obtain "meaningful information about the logic involved" in automated decision-making [1], e.g. how the process works, which variables were of significant influence. The consequence following from this regulation is that whenever a process is used in automated decision-making, it has to be well-understood in order to provide citizens with meaningful information about its logic upon request. Where deep learning techniques perform well in a wide range of areas, they are too complex in order to meet the aforementioned requirement necessary to be in line with law. This might seem applicable to all classifiers but there exists those widely known for their interpretability thus allowing its use while abiding the law.

Decision tree learning is a method that uses tree-like models to solve classification problems. Where a DNN classifies data through communication between layers of cells, which makes the decision hard to be understood, a decision tree allows a conclusion to be reached by following a path through the tree. A tree consists of conditions and possible answers to these conditions, leading to its leaves representing different classes. By starting at the root and traveling through the tree up to a leaf, its decision process can be well-understood in order to meet the requirements by law. When the tree gets too large, there might be too many conditions to consider in order to properly understand the decision process. However, in that situation it would still be possible to answer other questions regarding the logic involved in the process, e.g. which decisions were of significant influence.

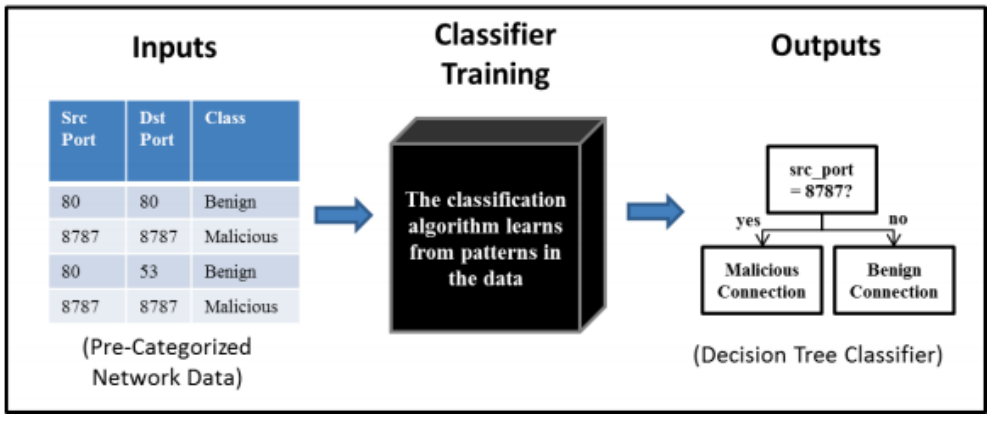


Figure 1.2: Overview decision tree learning from [44].

1.1. Motivation

Decision tree learning could support cybersecurity teams in current society, e.g. with identifying cyberattacks. However, in the traditional ML setting the data to be classified is assumed to come from the same distribution as the data used to train the classifier. This means that if a classifier is trained to recognise apples, it should not be used to recognise pears. In security-sensitive situations, e.g. the supermarkets using facial recognition to detect shoplifters [5, 26], this assumption cannot be guaranteed to hold as the data (and thus their underlying distribution) can be manipulated. A shoplifter could start using different shopping bags or, if applicable, grow a beard to fool the classifier. As a result the classifier's performance could decrease.

For the last two decades, researchers have been actively looking into this problem, the robustness of ML against malicious perturbations of samples [12, 16, 20, 24, 31]. Robustness can be understood as how well the model can withstand adversarial perturbations. Recently, research has also started into the robustness of decision tree learning [35, 40] which differs fundamentally from the current well-researched techniques as there exists no gradient to use. Chen et al. [22] reason that the robustness of classical decision trees and boosted tree systems can be improved, in case of a L_∞ norm bounded adversary, by optimising the worst-case performance of a split. The objective function can be formulated as a 0-1 integer optimisation problem but it is NP-hard in general and the number of minimisation problems to solve make it computationally intractable. They propose approximation algorithms in order to optimise the worst-case performance. Calzavera et al. [21] introduce a more flexible attack model consisting of pre- and post-conditions and numerically approximate the best split under worst-case performance. Where Chen et al. approximated the solution, in case of boosted tree systems, by considering four attack configurations, Calzavera et al. reformulate the objective function and numerically approximate a solution using convexity of the loss function. They also place constraints on tree growth to limit the number of sub-tree candidates in order to prevent the worst-case performance from declining. Vos takes a similar approach, optimising the worst-case performance, but also considers the case where only malicious samples are perturbed [57]. He introduced a different attack model by removing the concepts of budget and cost, as used by Calzavera et al., as well as the pre- and post-conditions. To improve robustness, the impact of an adversary is included by rewriting the splitting criterion. The problem can then be analytically solved to optimise worst-case performance, instead of computing the effect of different attack configurations. To allow an accuracy-robustness trade-off, the fraction of samples that can be perturbed during a split can be varied.

The state-of-the-art research on improving the robustness of tree-based systems focuses on improving the worst-case performance of the splitting criterion, which can be a costly, timewise, effort. Few works attempt to improve robustness without considering the worst-case performance, by better understanding what makes the classifier vulnerable, e.g. either through hypothesis testing [35] or weighting features and adding randomness in case of a random forest [59]. To the best of my knowledge, no approach has been considered that improves the robustness of both a single decision tree and an ensemble of trees without optimising the worst-case performance of the splitting criterion. Leaving a gap in the scientific understanding of the problem and the scope of possible solutions.

1.2. Research Questions

As earlier stated, no approach considered improving robustness of both a decision tree and an ensemble of these trees by a means other than optimising the splitting criterion's worst-case performance. The state-of-the-art research, by both Chen et al. and Calzavera et al., also left aside adversaries that only attack malicious samples, as considered by Vos. The combination is of interest for this work and the research objective of this thesis can be phrased as:

"How can tree-based models be made more robust against adversarial attacks perturbing malicious samples without considering adversarial examples in the splitting process?"

The research objective is broad and thus can better be divided into multiple research questions (RQ) that shape this thesis:

RQ1: Which factors of a tree-based model contribute to its robustness?
To address a decision tree's robustness, without considering adversarial examples during learning, requires understanding of what causes the classifier to become easier to fool. The answer to this question can be used to address the tree's robustness during its learning phase.

RQ2: How can robustness of a decision tree classifier be optimised during learning?
The previous research question answers what factors affect the classifier's robustness. To learn a more robust decision tree, approaches are explored that aim to optimise metrics related to the previously identified factors.

RQ3: What is the effect on a random forest when built using robust decision tree classifiers?
Addressing this research question means determining the effect of the different suggested approaches from the previous research question on an ensemble of trees. This allows to assess whether an ensemble of trees can be made more robust by consisting of individual robuster decision trees.

RQ4: How effective are the approaches in this work compared with state-of-the-art research?
The previous research questions analysed the effectiveness, in terms of robustness, of the proposed approaches with a baseline. This research question leads to a better understanding of the difference between state-of-the-art techniques using adversarial attacks in the learning phase and approaches which don't.

1.3. Contributions

The research performed and written down in this thesis report lead to several contributions in the field of adversarial machine learning focused on decision tree learning. The contributions are outlined in their particular paragraphs. The code for this research is available online[1] or can be found in the faculty's internal Gitlab.

Factors contributing to robustness
In this report, three factors are introduced that are believed to contribute to the robustness of decision tree classifiers. These factors are: *feature frequency*, whether features are used to split the data or not, *distance between leaves*, the distance between malicious leaves and benign prediction space, and *impurity of benign prediction space*, the fraction of malicious training samples falling in benign prediction space. Numerical data was obtained and analysed determining the strength of correlation with robustness for each of the factors and to determine the effects of a state-of-the-art approach on them.

Metrics to improve robustness
Two metrics, *minimising malicious samples* and *maximising distance*, are proposed to improve the identified factors and thus the classifier's robustness. Drawbacks from the integration of these metrics in the learning algorithm and their optimisation are investigated and addressed by trade-off approaches. Two trade-off approaches, *minimum threshold* and *greedy improvement*, are proposed to counter these

[1] Repository containing the code for this project: https://github.com/Cas-B/irdtss

drawbacks either by manually setting a parameter or allowing the algorithm to find a solution automatically. The evaluation of the approaches (and combinations of them) are presented in this report.

Evaluation of comparison with state-of-the-art research
After verification and validation of the proposed combinations of optimisation and trade-off approaches, they are compared with state-of-the-art work by Vos [57]. The evaluation is presented in this report and based on it a recommendation for the use of the approaches is given.

1.4. Outline

Chapter 2 provides background information necessary to understand the research and brief descriptions of research within the field. Chapter 3 continues by discussing related state-of-the-art work and provides identified research gaps. Chapter 4 describes the threat model used within this thesis, where Chapter 5 elaborates on the methodology and evaluation setup. Chapter 6 proceeds with discussions of the results following from the experiments. Chapter 7 discusses possible shortcomings related to the evaluation process and results, and Chapter 8 concludes this thesis while also providing recommendations for future work.

2

Background

This chapter covers the concepts necessary to understand the performed research. It starts with an explanation on decision trees, followed by one on random forests. After the classifiers are understood, adversarial examples are introduced. To perform a security assessment, two taxonomies are explained that are used to construct a threat model. At last, to get a better understanding of this research field, an overview of several works related to adversarial examples and decision trees is provided. This excludes state-of-the-art research, which is discussed in chapter 3. Already comfortable with the necessary concepts? Feel free to skip this chapter or skim through for a quick reminder.

2.1. Decision Trees

A Decision Tree (DT) is a predictive model used in different fields, e.g. data mining and machine learning. The internal structure of the model is tree-shaped consisting of nodes, where each node is either a *decision node*, representing a decision used to split samples based on their characteristics (attributes/features), or a *leaf node*, representing a class. Fig. 2.1 shows an example of a tree consisting of three nodes, a single decision node splitting into two leaf nodes. A DT returning a discrete prediction is called a *classification tree* and when it returns continuous predictions it is called a *regression tree*. There exist several algorithms to construct a DT, e.g. ID3 [50] or C4.5 [49], but all DTs used as part of this research are constructed using CART [58]. It is part of well-known machine learning software [48] also used in industry and thus whenever spoken of a DT in this work, it refers to a classification tree built with CART.

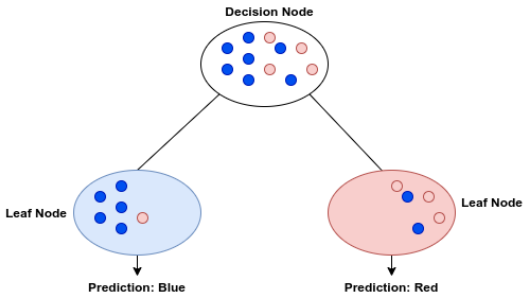


Figure 2.1: Illustration of a tree-structure.

2.1.1. Construction

A DT is constructed by splitting samples from a *training dataset* in order to separate the present classes as best as possible. Each decision node in a DT splits the samples, according to a splitting rule. In CART, the splitting rules have the following form [58]:

if CONDITION then left, otherwise go right

where CONDITION is of the form: *attribute X* <= *threshold*. A sample goes either left or right, resulting in a binary classification tree where each decision node has two child nodes. The most optimal split is found by an exhaustive search over all potential splitting points and attributes in order to optimise the splitting objective. There exist several metrics to evaluate a split, e.g. entropy [18] or twoing [18], but all DTs used in this work are evaluated using the *Gini impurity* [18] metric as it has several advantages over the alternatives [13, 53]. To fully grasp this metric it is best to go through some examples. Assume there are *C* classes and f_i is the fraction of samples labelled as class *i* within the node being evaluated.

The *Gini impurity* (GI) of this node can then be computed with the following equation [18]:

$$GI : \sum_{i=0}^{C-1} f_i(1 - f_i) \tag{2.1}$$

Example:
Assume a 2-class problem and the current node containing samples belonging to the same class:

$$GI : \sum_{i=0}^{C-1} f_i(1 - f_i) = 1(1 - 1) + 0(1 - 0) = 0 \tag{2.2}$$

Example:
Assume a 2-class problem and the current node containing samples of both classes in equal proportion:

$$GI : \sum_{i=0}^{C-1} f_i(1 - f_i) = 0.5(1 - 0.5) + 0.5(1 - 0.5) = 0.5 \tag{2.3}$$

As aforementioned, a decision node splits into two children. In CART, the best possible split is the one where the children lower the decision node's impurity the most, thus having the highest impurity improvement. Mathematically this comes down to the following. Assume *q* is the fraction of samples ending up in the left child node as a result of the split. The impurity improvement, *I*, can then be computed as follows [58]:

$$I : GI(\textit{current}) - qGI(\textit{left child}) - (1 - q)GI(\textit{right child}) \tag{2.4}$$

Example:
Assume a 2-class problem, with GI(current) = 0.4, GI(left child) = 0.2, GI(right child) = 0.1 and 80% of the samples from the current node moving to the left child node:

$$I : 0.4 - qGI(left) - (1 - q)GI(right) = 0.4 - 0.8 * 0.2 - (1 - 0.8) * 0.1 = 0.4 - 0.16 - 0.02 = 0.22$$

The splitting process continues until a certain *stopping criterion* has been reached, e.g. maximum number of leaf nodes, minimum number of samples in a leaf or the obtained impurity improvement becomes lower than a certain threshold. These criteria are important as they help preventing the tree from overfitting on the training data, which could cause the model's performance to drop.

2.1.2. Classification

After a model has been constructed, it can use the learned patterns to predict a sample's class. The process of predicting a sample's class is referred to as *classification* and the set of samples to be classified by the model is called the *test dataset*. Classification of a sample by the model comes down to following the path from root node to leaf node. Starting from the root node, branches are taken in accordance with the decision node's splitting rule and the sample's feature value, until a leaf node is reached. This path from root node to leaf node is called the sample's *decision path*. The resulting prediction depends on the training samples ending up in the particular leaf node. The class with most samples in the leaf node is the one predicted. The prediction is not influenced by the difference in the number of samples of each class but the *purer* the class, the better its prediction confidence.

This prediction process can be understood by taking a look at figure 2.1. The root node has split into two children, the left child and the right child. The left child consists of 6 samples, 5 belonging to the blue class and 1 belonging to the red class. The right child consists of 5 samples, 3 belonging to the red class and 2 belonging to the blue class. Following the aforementioned explanation, the left child predicts the blue class and the right child predicts the red class.

2.1.3. Evaluation

Understanding how the model classifies a sample, it is time to go into evaluating its classification performance, usually performed using the *test dataset*. A commonly used metric for this evaluation is the *accuracy*, the fraction of correctly predicted samples.

Definition 2.1 *The accuracy of a model is the fraction of a population classified correctly:*

$$accuracy = \frac{Number\ of\ correctly\ predicted\ samples}{Total\ number\ of\ samples}$$

It is important to understand that the *accuracy* metric does have its own flaws regarding evaluation of a model's classification performance. For example, an accuracy of 95% could hide the fact that the model only correctly predicts a single class making up 95% of the data and misclassifies all other classes. One way to discover these hidden facts is by using a *confusion matrix*, figure 2.2, which is made up of four categories: *True Positives* (TP), *False Positives* (TP), *True Negatives* (TN) and *False Negatives* (FN).

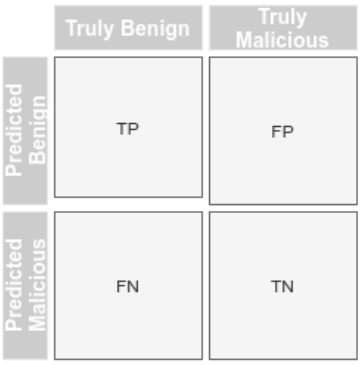


Figure 2.2: Confusion Matrix

Classifiers with hyperparameters, e.g. the maximum depth of a DT, are often first tuned using a *validation dataset* consisting of samples not part of the training- and test dataset, to improve the final model's evaluation performance. This tuning is not performed using the test dataset as it would introduce bias in later evaluations.

2.2. Random Forest

A random forest (RF) is an ensemble learning method in which several decision tree classifiers are combined into a single classifier. As long as these individual DTs are both accurate and diverse, it results in an RF that often has better accuracy than the individual classifiers themselves [29]. The RF, being an ensemble, has several differences regarding its construction and the classification of samples, compared to a DT.

2.2.1. Construction

As mentioned earlier, a RF needs to be made up out of a diverse set of DTs to become more accurate than each individual one. There are several approaches that can be used to diversify the individual DTs but the two relevant for this research are using a (1) different set of training samples and a (2) different set of features representing the training samples, randomness playing a role in each [19]. Bootstrap aggregation, also known as *bagging*, is used to construct different training datasets for the individual DTs. It works by taking the original training dataset, D, and creating m smaller datasets through uniform sampling with replacement from D [29]. To diversify the individual DTs even more, only a randomly selected subset of the original features is used during training. The set of constructed DTs together are then called a random forest.

2.2.2. Classification

Consisting out of a multiple DTs, a RF classifies samples slightly different by considering the predictions of the individual DTs and combining them into a single prediction. Different methods for combining the individual predictions exist, like performing a majority vote where the most predicted class is predicted by the RF. However, the RFs used in this research combine the individual classifiers by averaging their prediction probabilities to come to a prediction, which could result in better performance[1].

[1] Scikit-learn (2020). Ensemble methods. *Scikit-learn documentation*. Retrieved May 14, 2020, from https://scikit-learn.org/stable/modules/ensemble.html#forest

2.3. Adversarial Examples

Szegedy et al. introduced the term *adversarial example* in 2014, referring to a perturbed sample that would maximise a classifier's prediction error [54]. Nowadays, this term can be understood in the more general sense as referring to a sample that is perturbed with an adversarial goal in mind. These adversarial examples are used to evaluate the security of classifiers, e.g. sensitivity of learning algorithms to perturbations.

To better understand the concept of *adversarial example*, lets take the case where pictures are sorted, using ML, in stacks belonging to panda's and those belonging to gibbons. In a normal situation, a classifier correctly classifies the picture in figure 2.3, with 57.7% confidence, as a panda and places it on the correct stack. However, an adversary decides to cause failures in this process and perturbs the picture by adding a certain amount of noise. As a result, the picture is interpretable by a human as a panda but the classifier, which correctly predicted the picture before, classifies the perturbed picture as a gibbon with 99.3% confidence. The adversary has thus successfully caused the process to place the picture of a panda on the stack intended for pictures of gibbons.

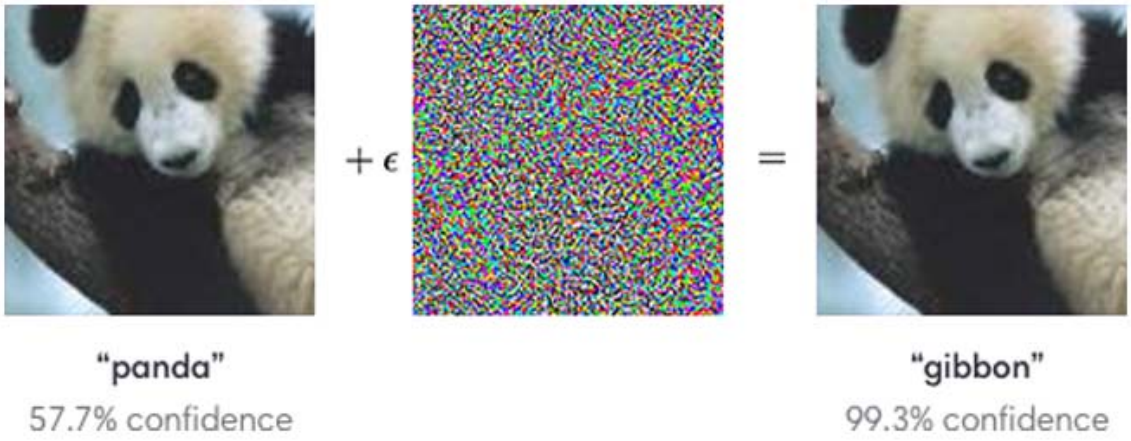


Figure 2.3: Adversarial Example of a misclassified panda from [32].

2.3.1. Crafting

Adversarial examples are used to assess the security of a classifier and they differ in the way they are crafted to fulfil their adversarial objective. Usually the objective is to craft adversarial examples while minimising the perturbations to the original sample, resulting in *minimal adversarial examples*. Arguments exist in favour of maximising the misclassification confidence [15] , however that has not become common practice so-far. A considerable number of algorithms have been proposed to construct adversarial examples, either based on gradients [27, 33, 45, 46], scores [24] or decisions [20]. Where the capabilities of the adversary decides whether the adversarial examples are constructed against the original model or a surrogate learner. However, in case of DTs it is possible to construct adversarial examples by making use of its tree-structure.

Crafting a minimal adversarial example for a DT comes down to finding the leaf node belonging to the targeted class, which requires the least amount of perturbations to be reached. Thus requiring knowledge of the prediction values of the leaf nodes and the splitting rules along their decision paths. Considering the capabilities of the adversary, this information can either be accessed directly or obtained through a number of queries. Crafting the adversarial example comes then down to evaluating each leaf node according to a distance metric and choosing the one with the best outcome. Algorithm 1 specifies the explanation of this *perfect attack* in pseudo-code.

Algorithm 1 Perfect Attack against Decision Trees

P ←find the leaf prediction values and store them based on their node id.
B ←find the feature requirements to reach each leaf node (breadth-first search) and store them based on their node id.
D ←∅, E ←∅, s ←Sample, l ←Target label

for each: *nodeid* ∈ P **do**
  if $P_{nodeid} = l$ **then**
    E_{nodeid} ←Perturb s with B_{nodeid}
    D_{nodeid} ←Compute distance between s and E_{nodeid}
  end if
end for

nodeid ←find nodeid with the smallest distance in D
return E_{nodeid}

A similar approach can be taken to craft minimal adversarial examples for a RF, which consists of several DTs. Instead of finding a single leaf node belonging to the targeted class, a combination of leaf nodes that results in the desired prediction has to be found. To compute the distance metric, splitting rules along a path in each DT have to be considered in order to minimise the objective function.

2.4. Threat Model

Adversarial examples are a tool used to evaluate the security of a learning algorithm and that what it should protect against is described by a threat model. Adversarial examples differ based on the objective of the adversary and its capabilities, which means that a threat model should consists of both an attack model and an adversary model. Section 2.4.1 describes the attack taxonomy by Barreno et al. [11] that is used to model an attack and section 2.4.2 explains the taxonomy introduced by Biggio et al. [14, 16] that can be used to model an adversary.

2.4.1. Attack

To perform a security assessment of a learning algorithm, it is important to understand what an adversary attempts to achieve with the adversarial examples it crafts. Barreno et al. introduced a taxonomy to model attacks against supervised learning systems [11], which has been used in several other works [12, 15, 42]. The taxonomy aids in specifying the objective of an attack and consists of three components: the *influence* it has over the classifier, the *specificity* of the attack and *security violation* it causes, as shown in table 2.1.

Influence

The influence of an attack can be understood in terms of how much control can be exercised over the learning system. A **causative** attack undermines the learning phase, when constructing a model, by manipulating the training data and/or the test data. A well-known kind of causative attack is a poisoning attack, in which altered samples are injected into the training data with which the model is built. The other type of attack only allows the test data to be manipulated in order to undermine the classification by a learned model, which is called an **exploratory attack**. A known kind of exploratory attack is an evasion attack, in which a sample is modified to be misclassified by the learned model.

Specificity

The specificity determines what the focus of the attack shall be. Two kinds of focus are used within this taxonomy, which are **targeted** and **indiscriminate**. There are different interpretations of the targeted focus, where Barreno described it as focusing on the target input [12], current research looks at the target output [41]. A targeted attack attempts to modify a sample in order to have it misclassified as a specific output class. An indiscriminate attack is broader and focuses on misclassifying a sample to any other output class than the true class.

Security Violation
The security violation concerns itself with the kind of error the classifier is expected to make as a result of the attack. Violations on the following parts of the learning system are identified: **integrity**, **availability** and **privacy**. A violation of the system's integrity results in an adversarial example to be misclassified. Violating the system's availability aims at producing enough false positives and false negatives in order to render the system unusable. The last kind of violation, privacy, aims at obtaining sensitive information from the system, e.g. about its users or its data.

Summary
By combining the three components described above, it is possible to clearly describe the objective of an attack a classifier should protect against. Table 2.1 summarises the different attack objectives as a result of the chosen taxonomy components.

		Integrity	Availability	Privacy
Causative	Targeted	Enable a specific misclassification by the system	Sufficient mistakes are introduced to render the system unusable for a specific person or service	Allow specific information about the system to be exfiltrated
	Indiscriminate	Enable a misclassification by the system	Sufficient mistakes are introduced to render the system unusable	Allow information about the system to be exfiltrated
Exploratory	Targeted	Find a sample that misclassifies to a specific class	Find a number of samples misclassified to a specific class	
	Indiscriminate	Find a sample that misclassifies to any other class	Find a number of samples misclassified to any class	

Table 2.1: Overview of attack taxonomy

2.4.2. Adversary

Specifying the objective of an attack using the taxonomy by Barreno et al. [11] tells what kind of attack is expected but it does not aid in determining the attack's effectiveness. Adversaries can differ in their approach and capabilities, resulting in different outcomes even if they share the same attack objective. Biggio et al. described a taxonomy to model an adversary [14, 16], which is defined over four dimensions: the *goal* of the adversary, the *capabilities* it has in order to manipulate data, the *knowledge* it is is able to utilise in building its attack and the *strategy* the adversary uses for its attack.

Goal
The goal can be formulated as an objective function that can be optimised by the adversary. The objective function considers the components defined in the attack model, e.g. targeted integrity violation to maximise the number of misclassified malicious samples.

Knowledge
The adversary might have certain knowledge of the learning system, which it can utilise when building an attack against it. The two extreme categories are: **white-box** and **black-box**. Applying Shannon's maxim: "*the enemy knows the system*", in a white-box setting the adversary knows everything about the learning system. This setting allows to take a closer look at worst-case scenarios and to compute a lower bound for the attack. In contrast, the black-box setting assumes the adversary has no knowledge about the learning system besides the results (predictions) it obtains through querying the system.

The mentioned categories cover the extreme situations but reality might actually lie in between. A third category, which might be called **grey-box**, would allow to specify which knowledge an adversary has

access to and which not. Biggio et al. propose a more schematic scheme to define the knowledge an adversary has about the system [14], which consists of: *(Part of) training data*, *Feature representation of each sample*, *Learning algorithm and decision function*, *Learned classifier's parameters* and *Classifier feedback (labelled results)*. The knowledge an adversary has about the system can be any combination of these parts, where the white-box and black-box define the possible extremes.

Capabilities
The capability of the adversary determines the control it has over the training and test data. Considering the *influence* component of the attack taxonomy, e.g. whether it has access to the training and/or test data, also the number of influenced samples, its ability to manipulate the class priors, which features can be manipulated and to which extent are a matter for thought. There could be a valid reason to limit an adversary's capabilities, e.g. prohibit an adversary from manipulating a feature it is certain to have no control over.

Strategy
After the *goal*, *knowledge* and *capabilities* of an adversary are defined, it is possible to define the optimal strategy for an attack, e.g. how adversarial examples should be constructed to optimise the objective function. The strategy considers how class priors can be manipulated, the number of samples affected by the attack, and which and how samples are to be perturbed. The combination of this information results in an attack strategy, completing the adversary model.

2.5. Relevant Approaches

The previous sections described the necessary background information to understand the technical parts of this research. Here the research is placed into context by briefly discussing other works related to adversarial examples and DTs, excluding state-of-the-art research. This provides a better understanding of the research performed within the field. A note to be made is that it might seem a work should instead be discussed in chapter 3, relevant state-of-the-art research. However, these works have been selected based on their status as well as obtainability of their source code.

2.5.1. Cyber Security

The research of adversarial examples focuses on the security of ML classifiers. However, most works are evaluated through image classification problems, using the MNIST dataset, instead in the more traditional security settings, e.g. malware detection. It is important to note that works tested in the more traditional settings do exist and have been increasing in number over the last few years. The main focus of these works is strengthening ML in the context of spam filtering [31], malware classification [28, 34, 37, 51, 52] and intrusion detection [38].

2.5.2. Crafting Adversarial Examples

Numerous algorithms for crafting adversarial examples have been proposed, which can generally be grouped into the three categories mentioned in section 2.3.1. Most of the approaches are developed for linear classifiers and DNNs but they can be relevant for non-linear classifiers, like DTs, due to phenomenon that adversarial examples are not always limited to the classifier they were crafted for [47]. Two algorithms have been proposed to craft adversarial examples, specifically for classifiers based on decision trees.

The first algorithm to craft adversarial examples for DTs was introduced in 2016 by Papernot et al. [47]. Constructed adversarial examples are classified by leaf nodes located as close as possible to the original leaf node in the tree. The method takes the decision path of the original samples and considers alternative paths by reversing the travel through the tree from leaf to root. It should be noted that this method does not attempt to minimise the perturbations necessary to craft the adversarial example.

Kantchelian et al. were the first, to the best of my knowledge, to introduce an algorithm that crafts adversarial examples against ensembles of DTs [40]. They prove that the problem of finding an instance belonging to a specific class for a tree ensemble is NP-complete. Finding the minimum adversarial example thus requires solving a NP-complete subproblem. A problem translation to a Mixed-Integer

Linear Programming (MILP) representation is suggested, which can be solved by specialised solvers. The translation is linear in size of the model but is not scalable in terms of the running time. As an alternative solution they propose an approximation algorithm, which greedily modifies the best feature until the sample is misclassified.

2.5.3. Defences

The considerable number of algorithms to craft adversarial examples shows the importance of improving the robustness of ML. A wide variety of defences have been proposed to achieve this goal, some with a focus on specific classifiers and others applicable to all of them, like adversarial training [54] and distillation [36]. There have also been defensive approaches proposed for tree-structured classifiers, which are described below.

Kantchelian et al. demonstrated improving the robustness of boosted DTs through *adversarial boosting* [40]. Their approximation algorithm was used to craft adversarial examples for each sample in the original training data, based on the current model, and appended to construct the training dataset for the next boosting round. This shows a similar idea as *adversarial training*, where adversarial examples are used to train more robust classifiers. They noted that training using L_0 norm bounded evasions reduced the robustness against other type of evasions, e.g. L_1, L_2 or L_∞ norm bounded.

Grosse et al. take a different approach and argue that adversarial examples, as they are drawn from a different distribution, could be detected by statistical tests [35], e.g. Maximum Mean Discrepancy (MMD) and Energy Distance (specific case of MMD without kernel). Through hypothesis testing, *"samples from the test distribution are statistically close to samples from the training distribution"*, they attempt to identify adversarial examples. However, this method requires the evaluation of several samples at once and cannot be used to identify whether a single sample is adversarial or not. They extended their work by including an outlier detection through training based on a constructed outlier class. It should be noted that training on an outlier class might not include all possible adversarial examples, which could leave the classifier vulnerable.

A different approach was taken by Yang and Chen, who improve the robustness of random forests [59]. Their idea is that vulnerability is caused by the frequency of features present in the RF, which can be addressed by preferring the less frequent ones. Their contribution is threefold and starts by introducing weights to prefer less frequent features. For each feature, this weight is computed using their average maximum information gain over all the trees in the forest and allows a robustness-accuracy trade-off by means of a parameter. The second contribution is an alternative for the information gain metric, which they call *differential ratio*. It uses the difference in the number of positive instances between two child nodes with respect to the number of instances in the node. Their last contribution consists of adding randomness in the classification process. Trees are clustered based on common features and a random number of trees from each cluster are selected to classify a sample, which should result in a more diverse set of involved features.

2.5.4. Others

It might seem that the focus lies primarily on discovering novel ways to craft adversarial examples and improving robustness but progress has also been made in other areas. The works here focused on model-extraction, robustness verification and guaranteeing the upper bounds on robustness.

Tramèr et al. developed model-extraction attacks against ML-as-a-service applications, with only access to a public querying interface [55]. They are able to, using either confidence values or class label predictions, construct a (near) perfect model similar to the original one. Their attacks are applicable to a wide range of classifiers, including a specific algorithm to extract the model of DTs.

Chen et al. focused on the problem of robustness verification for tree-based models [23], either verifying the robustness or providing a lower bound. To verify the robustness, they reduce the problem of finding a minimum adversarial example to finding maximum cliques in a multi-partite graph. By using a multi-level framework to compute (approximations of) the lower bound, they demonstrate that considerable speedups can be achieved in comparison to the MILP approach by Kantchelian et al. using L_∞

as distance metric.

The work by Andriushchenko et al. is the last to be mentioned. In their work they improve the robustness of boosted decision stumps and trees through optimisation of an upper bound on the robust loss [10]. They claim to introduce the ”first algorithms directly optimizing provably robustness guarantees in the area of boosting” [10].

3

Related Work

This chapter discusses relevant state-of-the-art research on improving the robustness of tree-like classifiers. The works have been chosen, not only as they are state-of-the-art work, but also based on the level of reproducibility as a result of detailed descriptions and accessible source code. First the work by Chen et al. [22] is discussed, which can be seen as a source of inspiration for the second work by Calzavara et al. [21]. The last work discussed is by Vos [57], which uses inspiration from both earlier works.

3.1. Robust Trees

Chen et al. propose an algorithm to improve the robustness of DTs in their work: "*Robust Decision Trees Against Adversarial Examples*" [22]. Their approach depends on performing the best split to optimise the worst-case performance against a L_∞ norm bounded adversary. The idea is visualised in figure 3.1, starting with the best possible separation of the training samples at the top. An adversary is capable of perturbing samples, causing them to move and potentially switch sides. To prevent this from happening, a different split can be made taking into account the potential perturbations of samples by an adversary. Making it harder for an adversary to cause misclassification of samples.

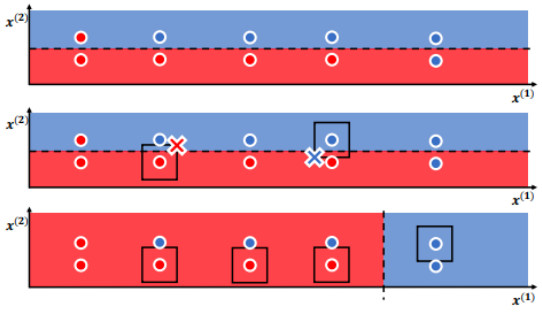


Figure 3.1: A simple illustration of the idea behind the robust splitting from the work by Chen et al. [22].

Defensive Approach

The approach of the authors would demand impact evaluations of a considerable number of combinations of attacks. To limit this number they divide the samples within a node into three groups, based on the impact of a potential attack: those samples that remain in the left or right child node (unaffected by the attack) and those that would switch child node as a result of the attack. They formulate the objective function, to optimise the worst-case performance, as a 0-1 integer optimisation problem using the samples affected by a potential attack as variables. However, the problem to be solved is NP-hard and the number of optimisation problems to solve for a single split makes it computational intractable [22]. They propose two approximation algorithms, one for DTs and another for tree boosting systems, to circumvent this issue.

The authors prove, for binary classification, that, considering the left child node, the information gain decreases when the fraction of 0-labelled samples and 1-labelled samples approach each other numerically. Their approximation algorithm comes down to finding a perturbation that minimises the difference between the two fractions. They prove a similar theorem for gini impurity and provide an algorithm to

find such a perturbation. This approach adds a computational complexity of $O(mN^2)$ where *m* is the number of features and *N* the number of samples belonging to the group affected by an attack.

They study ensembles built with regression trees instead of classification trees towards their approximation algorithm for tree boosting systems. However, their procedure seems also applicable to ensembles of DTs. To approximate the optimal solution for tree boosted systems, they consider the impact of four cases on the objective function and choose the one that minimises the worst-case performance. These four cases are: (1) no perturbation occurs, (2) all samples move to the left child node, (3) all samples move to the right child node or (4) all samples switch from child node, where the samples refer to those impacted by a potential attack.

Evaluation

The authors evaluated their approach by comparing natural models with their robustness improved versions in the context of *untargeted evasion attacks*. In terms of the attack taxonomy by Barreno et al. [11]: exploratory indiscriminate attacks violating the model's integrity, meaning that adversarial examples are allowed to be misclassified as any other class. They evaluate DTs using information gain as score function and Gradient-Boosted Decision Trees (GDBT) using a modified xgboost implementation to which they add their own split function.

Three algorithms for crafting adversarial examples are used (when applicable), by Papernot et al. [47], Kantchelian et al. [40] and Cheng et al. [25], resulting in both white-box (Kantchelian et al.) and black-box (Cheng et al.) settings. The number of adversarial examples crafted differ between the datasets, ranging from 100, for datasets categorised as small, to 5000 for those categorised as medium/large size. An exception is the MNIST 2 vs. 6 dataset, where the same parameters as used by Kantchelian et al. are used in order to compare with the *adversarial boosting* method. All adversarial examples are crafted against correctly classified samples, which are the only relevant ones for this evaluation.

The evaluation metrics used are L_1, L_2 and L_∞, for which the robustness improved versions consistently achieve better results with a maximum of 8.5% reduction in accuracy compared to the normal models. Although not explicitly stated as a comparison, they do mention their method to improve the robustness where *adversarial boosting* could not on the aforementioned metrics.

Analysis

There are some remarks to be made about the research performed by Chen et al, regarding their approach and evaluation.

First, the approximation algorithms used to avoid solving an intractable computation. The authors constructed a, proven, approximation algorithm for single DTs but resorted to the use of four "representative" attack cases for tree boosting systems. This might indicate that the proven method is not translatable to ensembles or not scalable in terms of the running time, while the alternative of four cases is not enough to act as substitute for the range of possible attacks.

Second is the limitation on the number of crafted adversarial examples for the evaluation. The authors might have chosen for this option to reduce the runing time of the evaluation, as the attack by Kantchelian et al. is not scalable, in terms of the running time, to larger datasets. However, this limitation also results in a loss of information, which makes the results less exact in terms of a worst-case assessment.

Third concerns the indirect comparison with the *adversarial boosting* approach by Kantchelian et al. The authors mention their method to improve the robustness, where the other approach fails but they overlook the (potential) differences in adversarial examples used and implementation. The authors train their method using L_∞ perturbed adversarial examples, while Kantchelian et al. trained their model using L_0 perturbed adversarial examples. Next to that, it is unclear whether differences in implementations between the two classifiers exist, aside the differences resulting from their robust algorithms.

3.2. TREANT

Calzavera et al. take an approach similar to Chen et al. improving the robustness of DTs in their work: "*TREANT: Training Evasion-Aware Decision Trees*" [21]. They approach the problem also by optimising the worst-case performance but learned from their predecessors and address shortcomings existing in their work. Their contributions include a different approach to craft adversarial examples, which should be discussed before diving into their defensive approach.

Attack Generation

The authors argue that distance-based constraints on the adversary's capabilities, e.g. L_∞ bounded perturbations, are not a fit for all possible scenarios, e.g. there might be categorical attributes, a need for asymmetric perturbations or features may not be alterable at all. To address this issue, they describe the adversary's capabilities in terms of *rewriting rules* of the following form:

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

where $[a, b]$ and $[\delta_l, \delta_u]$ are intervals on real numbers between, and including, minus infinity and positive infinity. The former defines the precondition for applying the rule and the latter the magnitude of the perturbation with f the feature to affect and k the cost of the operation. At the start of an attack, a *budget* is given and rewriting rules can be recursively applied until the total cost exceeds this budget. Resulting in a set of possible attacks that can be performed against a given sample. The authors argue that this algorithm allows for better fine-tuning of the adversary's capabilities and can also be used to define the standard distance-based constraints.

The result of this algorithm could be a large set of attacks, which might not all be necessary in order to evaluate the robustness. The authors refer to the optimisation of the worst-case performance, which can be written as a min-max optimisation problem, by minimising the *loss under attack*. To reduce the number of evaluations of the *loss under attack*, the authors note that a decision tree ensemble creates a finite partitioning of the input vector space. As instances falling in the same partition share the same prediction, they take a single representative instance per partition for evaluation and by that reduce the number of attacks to be considered.

Defensive Approach

To limit the number of impact evaluations to consider, the same approach is taken as by Chen et al. splitting the number of samples in three groups and only working with the group that a possible attack might impact. Where Chen et al. chose to approximate the worst-case performance by using four attack cases, here the authors pursue a numerical optimisation. Rewriting the min-max problem, given a feature and threshold, in terms of the leaf predictions, they observe that numerical optimisation can be used as long as the instance-level loss function is a convex function. Samples are then placed in the child nodes corresponding with the strongest possible attack.

The authors observed that splitting the tree by optimising the *loss under attack* for a given split does not prevent this *loss under attack* to increase as a result of added sub-trees when growing the tree. To preserve the locally optimised *loss under attack*, the authors put restrictions on the growth of a tree by limiting the set of possible sub-trees through introduction of constraints in the optimisation problem. These constraints ensure that when a leaf turns into a decision node, the original path to this leaf still represents the most effective attack strategy against the decision tree.

Evaluation

The authors evaluated their approach by comparing natural RF and GBDT, as well as models built with *adversarial boosting* and the approach by Chen et al. with a RF built using their own approach. The approach is also evaluated in the context of *untargeted evasion attacks*. The natural models, as well as their own approach, use the implementation provided by the LightGBM framework.

The evaluation takes place in a white-box setting, using the algorithm introduced by the authors for crafting adversarial examples. There are three datasets, accompanied with specific *rewriting rules* described by the authors. These datasets are split into a training-, validation- and test set, in a 60-20-20

distribution. To measure the effectiveness of the approach, *accuracy*, *macro F1* and *ROC AUC* metrics are used alongside training and test budgets, where the ROC AUC metric is also used to fine-tune hyper parameters.

The evaluation results state that the approach proposed by the authors outperforms each model it was compared with, irrespective of the training budget used, on all datasets. The approach also achieves similar or better accuracy than both *adversarial boosting* and the approach by Chen et al.

Analysis

There are several remarks to make about the research performed by Calzavera et al. specifically regarding their evaluation.

First, the implementation of their approach used in the evaluation differs slightly in that they enforce features to only appear once in a leaf's decision path. This is a necessary consequence resulting from the use of their attack algorithm in the evaluation but differs from the standard behaviour of growing ensembles.

Second, the attack algorithm generates only a subset of the potential attacks during evaluation. Instead of computing all possible attacks, which could result in a large number of attacks and thus slowing down the process, they focus on attacks maximising the perturbation along a feature. They cause only the cost of rewriting rules to be relevant and not their magnitude, which they argue is acceptable as long as features are not tested more than once.

Third, their approach to assess the robustness of a classifier differs from those used in earlier works. The authors stated in their work that: "The easiness of crafting successful evasion attacks defines the robustness of a given ML model at test time" [21]. They evaluate this easiness by giving an adversary different budgets, defining the combination of attack rules that can be used, and measuring accuracy, F1 and ROC AUC. However, earlier works asses the robustness of a classifier through L_x distance norms [22, 40] that could explain the difference in perturbations to be made by an adversary as a result of the robust approach. Thus even when the evaluation metrics allow comparisons based on the successfulness of an adversary, no statements can be made about the exact differences in robustness between the standard and robust classifier.

Fourth, the authors state to outperform the competition but implemented their algorithms in a different classifier than their own. Both competitor approaches, *Adversarial Boosting* and *Robust Trees*, were implemented in a GBDT classifier, while the authors implemented their own robust approach in a RF classifier. To compare the effects of the different approaches in a fair manner, all of them should have been implemented using the same classifier as different classifiers might have different performances of themselves. The authors also observe in their evaluation that "RF typically behaves better than GBDT on all the validity measures" [21], which should also have disqualified a direct comparison with the performances of the competitor approaches.

3.3. Robust and Fair Decision Trees

Vos takes an approach similar to Chen et al. and Calzavera et al. by optimising the worst-case performance [57]. His contributions include a flexible framework to specify attacks, a decision tree learning approach that analytically solves the robust split optimisation and an approach to construct both robust and fair trees. However, this analysis focuses on the relevant parts related to the robustness of trees.

Attack Framework

The author argues that it is unrealistic to assume an adversary can only perturb samples within a L_x radius. In this he takes a similar approach as Calzavera et al., allowing the user to define the threat-model by describing the kind of possible perturbations per feature. Where Calzavera et al. introduced *rewriting rules* consisting of pre- and post-conditions that can be used at a certain cost, Vos takes a more flexible approach without any cost related concept. Perturbations can be described in tuple notation, e.g. (ϵ, ϵ), defining the amount an adversary is allowed to decrease or increase a numerical's feature value. For categorical values, the user can describe a set of alternative values the adversary can use. As no concept of cost is used, adversaries with different strengths require different threat-models to be defined.

Defensive Approach

To improve the classifier's robustness, the author takes an approach very similar to the ones by Chen et al. and Calzavera et al. by optimising the worst-case performance of a split. He also splits the samples into three groups, where the third group (I) consists only of samples whose outcome can be impacted by an attack. He rewrites the splitting criterion, Gini impurity, to include the impact of the adversary. Where Chen et al. chose to approximate the worst-case performance and Calzavera et al. pursued a numerical optimisation for their score functions, Vos uses concavity of its function to analytically solve it in constant time. To provide the user with a certain degree of control, it is possible to set a ρ parameter. This parameter sets the fraction of malicious samples an adversary can impact, which means that the third group is replaced by $\rho|I|$ samples from I chosen through random sampling.

Evaluation

The author evaluated the effects of his approach both on a single DT and a RF. The single DT is compared with a standard DT and one trained using TREANT in the context of *targeted evasion attacks*, samples misclassified as a specific class. The approaches are evaluated both in a scenario where only malicious samples are perturbed and where both benign and malicious samples can be perturbed as TREANT was designed for this latter threat-model. The standard DT uses the implementation by scikit-learn [48], where TREANT and the author's own approach are implemented on top of their own DT implementations. The effects on the RF is evaluated in context where an adversary only perturbs malicious samples and compared with a standard RF implementation by scikit-learn.

A white-box setting is used to compare the performances of the different approaches, allowing an adversary to craft adversarial examples using all available knowledge about the models. Different datasets, including both numerical as categorical features are used, split in 80% training, 10% validation and 10% test data. Robustness is measured in terms of *adversarial accuracy* and *F1 score*, where adversarial accuracy is defined as the accuracy on an optimally perturbed test set. The author defines an attack model for each dataset, which is used to compare the performances of the different models. The evaluation results state that the author's approach outperforms both the standard DT and (in most cases) TREANT in terms of robustness, while matching the accuracy of the standard DT. In terms of the running time, the algorithm is orders of magnitude faster than TREANT. Compared with a standard RF, the accuracy slightly drops while improving its robustness.

Analysis

There are certain remarks that can be made about the researched performed by Vos, specifically the evaluation process.

First, the attack framework introduced might have let go of the concept of budgets and costs but can also be considered less specific than the one by Calzavera et al. The framework of the author differs from theirs in that no pre-conditions can be specified to allow even more tight grained control over the kind of attacks an adversary can perform.

Second, the author compares different algorithms which are integrated in different implementations of DTs. To increase the reliability of the results, the different algorithms should have used the same DT implementation to outrule the possibility of differences between them.

Third, the author measures robustness in terms of *adversarial accuracy*. As explained in the analysis of TREANT, accuracy allows comparisons based on the successfulness of an adversary but it does not shed light on the exact differences in robustness between the different models.

Fourth, the author considers only trees of low depths. By only considering low depths, the author (partly) negates the effect of TREANT's concept of *attack invariance*, where robustness cannot decrease as the tree grows. Thus it remains uncertain whether the attack invariance would have allowed TREANT to outperform the author's approach at higher depths. Besides, datasets where it is hard to differentiate between classes might require deeper trees to achieve good performance, an aspect the author has not taken into account.

3.4. Research Gap

The analyses of the state-of-the-art research and other works related to adversarial decision tree learning allows the identification of several research gaps which led to the research questions described in section 1.2:

- Most works attempt to improve robustness of tree-based models by optimising the worst-case performance of the splitting criterion. However, a question addressed by only few works is whether robustness can be improved without optimising the splitting criterion's performance under adversarial attacks.
- Most works focused on improving the robustness of a DT consider a threat-model in which both classes, in binary classification, are attacked by an adversary. Few works, to the best of my knowledge only the one by Vos, consider the case in which a single class is attacked, called the malicious class.

4

Threat Model

This chapter specifies the threat model used within this work. To develop effective defensive measures, it is important to understand the type of attacks and adversaries you have to protect against. Both an attack model and a model of the adversary are described in, respectively, section 4.1 and section 4.2, using the taxonomies by Barreno [11] and Biggio et al. [16].

4.1. Attack Model

There exist different kinds of attacks and to develop an effective defense, it is important to know what you want to protect the classifier against. Barreno's taxonomy allows an attack to be specified over three dimensions [11], as explained in section 2.4.1.

The *influence* specifies whether the classifier is attacked during its construction or the classification phase. State-of-the-art have focused on the latter, which can be understood as society starts monetising classification models. Cloud-based ML-as-a-service providers, e.g. BigML, allow parties to monetise their models through payments per query a user makes. This not only increases the reliance on a smaller set of models, but also means that a successful attack against the classification phase affects a large group of people. In this work the same approach is taken and thus the focus lies on attacks with ***exploratory*** influence, those undermining the classification phase.

The *specificity* determines the focus of the attack, which can either be causing a misclassification towards a specific class or any class. Problems faced by cybersecurity teams can often be translated into ML classification problems, where a difference has to be made between benign and malicious. This results in binary classification problems, where the choice of specificity becomes irrelevant. For completeness, a ***targeted*** specificity of attacks is considered within this work where samples are misclassified as a specific class.

The third component concerns the *security violation* the attack causes against the classifier. Availability and privacy are important aspects of a classifier that is used for cybersecurity purposes but I believe it is of a higher priority to guarantee the classifier's integrity first. An adversary that can violate a classifier's integrity has no need to disrupt its availability and attacks against the privacy of a model are not developed enough yet. The attacks used within this work are thus aiming to violate the ***integrity*** of the classifier, causing samples to be misclassified.

To summarise, the attacks used within this work can be described as *targeted, exploratory attacks that violate the integrity of the classifier*. In a binary classification problem this comes down to either causing benign samples to be misclassified or malicious ones. Where the former would likely result in obstruction of daily activities, e.g. blocking benign network traffic, it is more profitable for an adversary to hide its activities and thus have malicious samples been misclassified as benign ones. This shall also be the focus of the attacks within this research.

4.2. Adversary Model

Having specified the attack, its effectiveness depends on the kind of adversary executing it. The taxonomy by Biggio et al. [16], as described in section 2.4.2, helps modelling an adversary by describing its goal, knowledge, capabilities and its strategy in using them to increase the effectiveness of the attack.

Goal. As mentioned in the specification of the attack model, an adversary benefits from having (part of) its activities remaining unnoticed. A certain time and effort are required to accomplish this, which constrain an adversary in further developing its business. Thus an adversary would like to spend as little time and effort necessary in order to have more left for other, more profitable, activities. The workload of the adversary can be described in terms of the distance between the sample and adversarial example as a larger distance means more modifications have taken place. Following the definition by Kantchelian et al. [40], let $f : X \rightarrow Y$ be a classifier for a given instance $x \in X$ and $d : X \text{ x } X \rightarrow \mathbb{R}$ a distance function, e.g. L_1-norm. The goal of the adversary can then be described as minimising the distance necessary for causing a misclassification:

$$\text{minimise } d(x, x') \text{ where } f(x) \neq f(x') \tag{4.1}$$

Knowledge. Within this work the interest lies on improving the robustness of the classifier, measured by a certain distance function. The difference between models stems from the use of different data and parameters, while the algorithm to build the classifier is public knowledge. To perform a worst-case evaluation, the adversary has perfect knowledge within this work. This includes knowledge of: 1) the feature set, 2) the decision function, 3) its parameters, 4) the training data and 5) feedback from the classifier. It should be noted that 3 and 4 do not add additional information as the goal is to cause misclassification and access to the trained model is available.

Capability. State-of-the-art differ in their description of the capabilities an adversary has. Where Chen et al. [22] determine the capability through a maximum norm for the perturbation per feature, Calzavera et al. [21] and Vos [57] use a more flexible model which allows the specific capabilities of an adversary to differ per feature. The latter approach allows for more realistic attacks to be modelled but specific domain knowledge is required to describe the possible perturbations. Within this work the same approach as taken by Chen et al. and others, e.g. Kantchelian et al., is used. The adversary is capable of: 1) manipulating the test dataset but only malicious samples and 2) perturbing samples across every feature up to a pre-determined ϵ per feature.

Strategy. With the adversary's goal, knowledge and capabilities specified, it is possible to define its strategy used to increase the effectiveness of its attacks. The adversary's goal is to cause malicious samples to be misclassified at the least possible cost. As it has complete knowledge of the classifier, algorithm 1 can be used to construct minimum adversarial examples against DTs. For the same reason, it can use the MILP approach by Kantchelian et al. [40] or the maximum cliques approach by Chen et al. [23] to construct minimum adversarial examples against ensembles of DTs. The adversary profits from having its activities remaining unnoticed, which means that it wants to have as many malicious samples being misclassified as possible. The strategy of the adversary thus comes down to using the aforementioned adversarial example crafting algorithms on each correctly predicted, by the classifier, malicious sample.

5 Methodology

Within this chapter, the methodology and procedure for the work is presented. Factors contributing to the robustness of a tree are determined and investigated in section 5.1. This is followed by section 5.2, describing approaches to optimise the classifier's robustness by addressing the identified factors. Section 5.3 proposes solutions to make a trade-off between accuracy and robustness to negate possible drawbacks from optimising robustness. The chapter ends with section 5.4 describing the evaluation process of the research performed.

5.1. Factors contributing to the robustness

State-of-the-art techniques improve robustness by approximating the best node split that optimises the worst-case performance of the splitting criterion, e.g. gini impurity. This might lead to the desired effect but a better understanding of robustness and its contributing factors may lead to different solutions and even better outcomes. Analysing a DT led to the identification of three potentially contributing factors to the DT's robustness: (1) *feature frequency*, described in section 5.1.1, (2) *distance between leaves*, described in section 5.1.2 and (3) *impurity of benign prediction space*, described in section 5.1.3.

5.1.1. Feature frequency

Yang and Chen, in their work: "*Using Randomness to Improve Robustness of Tree-based Models Against Evasion Attacks*" [59], aim to improve the robustness of a random forest by addressing the frequency of features. They observed that features appeared with different frequencies among the individual DTs and addressed the issue by adding weights to the splitting process. This forces a more equal distribution of the frequencies. The different frequencies for features are relevant as an adversary could target the most frequently appearing features to alter decisions in several DTs at once. An unequal distribution of frequencies thus could make it easier for an adversary to craft adversarial examples.

The frequency of features can also be understood in terms of a single DT. Not all of the available features, depending on the training data and max tree depth, might be used to separate the classes during tree growth. To craft an adversarial example, an adversary only has to modify the sample to meet the requirements of a decision path leading to a leaf predicting the target class. The frequency of a feature can thus be understood as its frequency of appearing in the paths leading to the target class. An unbalanced distribution of frequencies then means that fewer features are present in these paths. An example is shown in figure 5.1 where the left tree uses only a single feature and the right tree uses two features to separate the data. To have a sample falling in the right-most leaf be classified as benign, an adversary not only has to adjust the amount of transactions but also the amount of money withdrawn. The distribution of features became a bit more equal, making it harder for an adversary to craft an adversarial example. The first factor contributing to the robustness of a DT is thus its *feature frequency*.

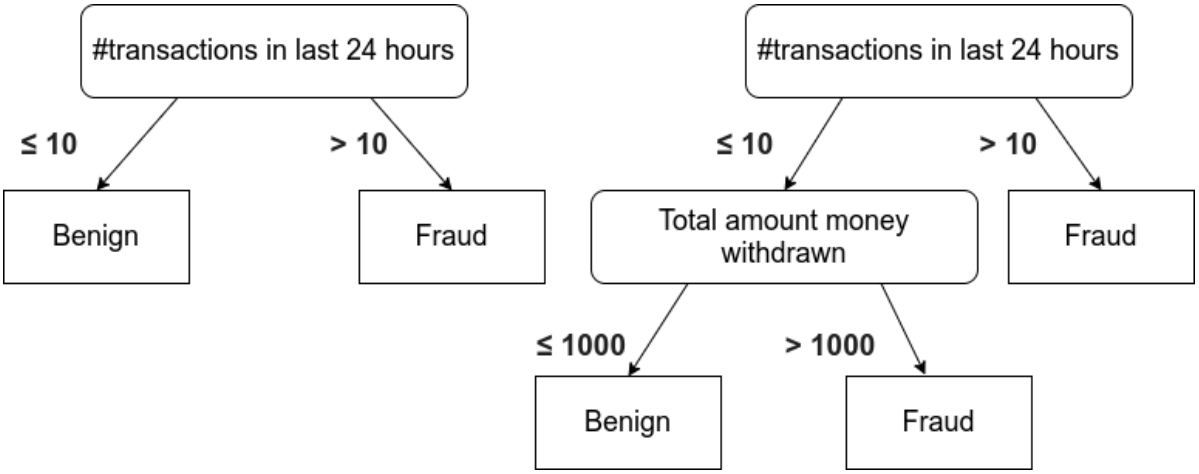


Figure 5.1: Example trees with different *feature frequency* for benign decision paths.

To test whether a correlation does exist between feature frequency and robustness, the number of modified features and the average amount of perturbations (L_1 distance) are plotted against each other for two datasets, see appendix figure A.1. The metrics are computed for 100 trees per dataset, trained using different training sets to create variety among the trees. To support the interpretation of these results, partial correlation with spearman's rank correlation coefficient, r_s, is computed using pingouin [56], keeping the other identified factors constant. Spearman's method is chosen instead of the Pearson correlation as the latter is more prone to outliers affecting the covariance and standard deviations. The figures indicate a positive correlation between the metrics for the diabetes dataset with r_s = 0.566. For the wine dataset the r_s = -0.248 but the figure indicates that no relation exists between the two metrics. It could be the result of an unknown variable, which interferes with this analysis.

5.1.2. Distance between leaves

To cause a misclassification to occur, an adversary perturbs a sample based on the relevant features and thresholds to reach a leaf predicting the target class, e.g. a benign predicting leaf. The total amount of perturbations required depends on the distance between the current leaf and target leaf, where a smaller distance means a lower amount of perturbations. The shortest distance between two leaves is the sum of distances based on features appearing in the decision paths of both leaves. This assumes that the sample already meets the requirements for all other decisions in the decision path of the target leaf. For example, suppose a sample ended in the right-most tree of the right tree in figure 5.1. In the worst-case situation, an adversary only has to adjust the amount of transactions as the sample to perturb already satisfies the other requirement. An adversary only has to cross the shortest distance between its current leaf and any leaf belonging to the target class. The second contributing factor to the robustness of a DT is thus the *the shortest distance between any malicious and benign leaf*.

Plotting the factor against robustness for two datasets did not indicate any correlation as the average shortest distance remained the same for all trees. For this reason, no additional support for the relevance of this factor is included.

5.1.3. Impurity of benign prediction space

Improving the robustness of a DT has been understood as making it more difficult for an adversary to craft adversarial examples, e.g. increasing the amount of required perturbations. However, it might require a lot of modifications to cause a misclassification by an adversary but this won't make a difference when the adversary does not need to perturb a sample for it to be misclassified. For a DT to be considered robust, it should thus not only make it difficult for an adversary to cause a misclassification but also ensure that an adversary actually needs to perform an action.

Leaf nodes can contain both benign and malicious training samples, with the prediction of the leaf representing the class with the majority of training samples within. As data to be classified should come from the same distribution as the training data, the fraction of a class' samples in a leaf acts as an indicator for the fraction of samples in the test data to be classified by this leaf. This means that the impurity of the benign classifying leaves influences the misclassification rate of malicious samples. The third contributing factor to the robustness of a DT is thus *the fraction of malicious training samples within benign prediction space*.

To test whether there exists a correlation between the fraction of malicious training samples within benign prediction space and the robustness of a DT, the former metric is plotted against the False Negative Rate for two datasets, see appendix figure A.2. The figures indicate a positive correlation for both datasets, with r_s = 0.412 for diabetes and r_s = 0.731 for the wine dataset. This supports the aforementioned argument that the factor does indeed contribute to the robustness of the classifier.

5.2. Optimising Robustness

In section 5.1, three factors were described that influence a DT's robustness. To improve the robustness of a DT without considering adversarial examples, different approaches are covered in this section. They aim to improve the identified factors, which should then, in turn, lead to a more robust DT. In section 5.2.1 an approach is described to trade the classifier's generalisation ability to increase its feature frequency and the distance between malicious and benign prediction spaces. Section 5.2.2 covers an approach to be combined with the previously suggested approach. A metric is added to integrate the method in the learning algorithm of the DT to construct *purer* benign prediction space. Last, in section 5.2.3 an approach is covered, also to be combined with the first, to further increase the distance between leaves.

5.2.1. Limiting benign prediction space

As explained in the previous section, a higher feature frequency can lead to a robuster DT. More unique features in the decision paths for benign classifying leaves means that an adversary must consider more relevant features. This also means that an adversary needs more capabilities in order to modify the other features with the necessary magnitude. To increase the feature frequency, multivariate DTs could be used where a split decision consists of a combination of variables. However, this is a different type of classifier and thus a different problem. To the best of my knowledge, there also does not exist an algorithm to compute minimum adversarial examples for these type of DTs that could be used to measure its robustness. An alternative could be to force the DT to select different features in decision paths but this would reduce the quality of the splits.

The aim is thus to increase the feature frequency while keeping the classifier's ability to effectively separate classes. Inspiration can be taken from one-class classification, where classifiers try to learn the representative characteristics of a single class from training data. Among the different methods to achieve this, there is one called: the *boundary method*. The one-class classifier learns to place boundaries around the training samples belonging to a specific class and optimises these boundaries using an objective function, e.g. minimising volume. An example of such a boundary can be seen in figure 5.2 (A). To increase the feature frequency in benign decision paths, the boundary method could be applied in leaves to limit the benign prediction space, see figure 5.2 (B). As benign samples come from a certain distribution they might not use all of the prediction space assigned by the classifier. This would change the working of benign classifying leaves as they could lead to both benign and malicious predictions, figure 5.2 (C). An additional advantage would be that a user could investigate classified samples by using the classification area to focus on the samples more likely belonging to the class of interest.

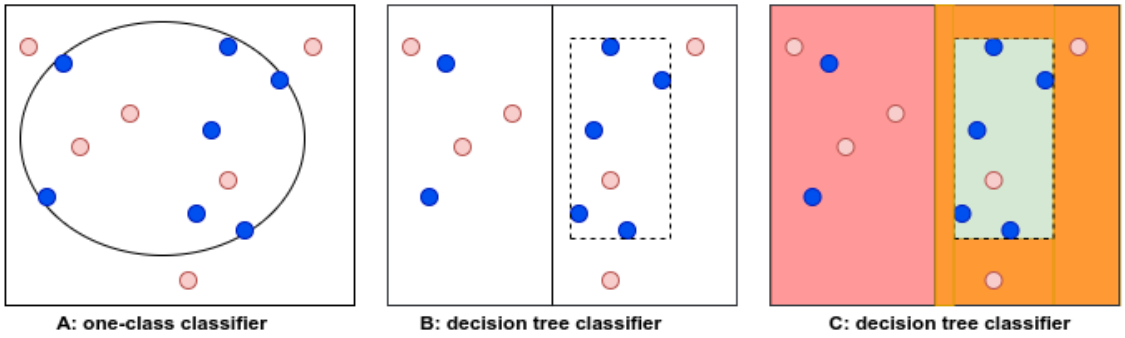


Figure 5.2: A: one-class classifier using the boundary method, B: the boundary method applied within a decision tree node, C: the different classification areas after applying the boundary method in a decision tree node.

To increase the feature frequency the most, the boundary method can be applied within benign classifying leaves using all features for the classification process. A consequence of limiting the benign prediction space is that benign samples might fall within the leaf but outside these boundaries. To limit the impact, boundaries are placed around all benign samples. The better the training data represents the class distributions, the likelier that benign samples do fall within the learned boundaries. Another measure to limit the impact is that boundaries are learned only when more than three benign training samples, an empirically chosen number, are present to avoid too tight prediction space. This approach is described in pseudo-code in Algorithm 2. A sample is thus classified as benign when it falls in a benign predicting leaf and within the boundaries when present. As the benign-classifying leaf is essentially split in two components, the classification probability should also reflect this. The probability within the boundaries is as in the original case, the proportion of benign samples from all training samples, in the node, falling within the boundaries. The area outside the boundaries contains no benign training samples and thus has no ability of predicting a benign sample.

Algorithm 2 Constructing boundaries around benign training samples

```
T ←Decision Tree Classifier
F ←Features
for each: leaf ∈ 𝒯 do
  P ←Prediction label of leaf
  if P is not benign then
    continue
  end if
  Cl ←Current leaf
  Bs ←Benign training samples in the leaf
  while count(Bs) < 3 do
    Cl ←parent node of Cl
    Bs ←Benign training samples belonging to Cl
  end while
  for each: f ∈ ℱ do
    S ←Sorted Bs along f
    Update max-min values for f
  end for
end for
```

Applying boundaries to each leaf in the trained DT has an impact on the classifier's training, specifically its running time. For each of the benign predicting leaves boundaries are constructed. As the construction time is dominated by sorting of the benign training samples, this adds $\mathcal{O}(n \log n)$ time using efficient sorting algorithms for each leaf. In the worst-case, there are 2^d, with d the depth of the tree, benign predicting leaves. Resulting in an additional $\mathcal{O}(2^d n \log n)$ time complexity.

5.2.2. Minimising malicious samples

The previously described approach reduces the benign prediction space within benign classifying leaves by placing boundaries, over all features, around the benign training samples. This approach increases feature frequency but also the distance between leaves as an additional distance between the split threshold and boundary has to be overcome. However, the approach does not directly affect the impurity of the benign prediction space, which might contain all the malicious samples in the leaf node. To address this last factor, a different approach is necessary to split the training data that aims to construct *purer* prediction space.

To construct *purer* benign prediction space, the splitting criterion can be replaced by an objective function minimising the number of malicious samples within. Let B_x be the set of boundaries of split x, bf the boundary feature, b_L the lower bound, b_H the upper bound, and M_x the set of malicious samples in split x with $m \in M_x$. The objective function can then be written as follows:

$$\text{minimise } (\sum_{i=0}^{M_L}(m|\forall b \in B_L, b_L \leq m_{bf} \leq b_H) + \sum_{i=0}^{M_R}(m|\forall b \in B_R, b_L \leq m_{bf} \leq b_H)) \tag{5.1}$$

The splitting procedure might find several optimal splits, which would differ in the division of samples over the child nodes. To choose between these optimal splits, it should be understood that the objective function does not aim to construct purer children, only purer benign prediction space. The difference lies in the effect it has on the distance an adversary has to overcome. The former increases the distance by forcing the adversary to cross the threshold, while the latter increases the distance within the leaf node itself. A combination of both might improve the robustness further, which is why the original splitting criterion is used as tie-breaker for the optimal splits found.

The integration of the previous approach in the learning algorithm changes the construction process of the boundaries. The new procedure has to be considered for each node split and is described, in pseudo-code, in algorithm 3.

Algorithm 3 Constructing boundaries while minimising the malicious samples within benign prediction space

```
SR ←Feature-indexed list of sorted samples along that feature, SL ←∅
Nbr ←All benign samples, Nbl ←∅
for each: f ∈ SR do
  for each: sample, label ∈ (Nbr ∪ Nmr) do
    if label is benign then
      Nbr ←Nbr - sample
      Nbl ←Nbl ∪ sample
      for each: f ∈ SR do
        Update the max-min values on both sides
        SR_f ←SR_f - sample_f
        SL_f ←SL_f ∪ sample_f
      end for
    end if
  end for
  Reset SR, SL, Nbr, Nbl
end for
```

The initialisation process has to be performed at the start of a split and is dominated by the sorting of all samples along each feature, which takes $\mathcal{O}(mn \log n)$ time using efficient sorting algorithms. The double for-loop exists in the original splitting procedure but additional operations are to be performed within. The time complexity of these operations are dominated by those for benign samples, which requires additions and removals from sorted lists as well as updating the maximum and minimum values for each feature. This takes $\mathcal{O}(m^2 n \log n)$ time by adding $\mathcal{O}(m \log n)$ operations to the original procedure. Minimising the number of malicious samples in the prediction space requires computing this

number for each potential split. This adds $\mathcal{O}(2mn_m)$, with n_m the number of malicious samples in the node, operations for each potential split. This results in $\mathcal{O}(m \log n + 2mn_m)$ additional operations per node split. A potential costly approach, depending on the number of malicious samples, but a model only has to be trained once. As for the classification phase, it should be noted that the space between boundaries monotonically decreases when the tree grows. Benign samples are split over child nodes, allowing the boundaries only to become tighter. This also means that only the boundaries closest to the relevant leaf node have to be considered, which only requires looping over $2m$ values.

5.2.3. Maximising distance

The previous approach complements the limited benign prediction space by addressing its impurity. However, by not focusing on its impurity there might be room to further increase the amount of perturbation necessary to craft adversarial examples. This could be achieved by further increasing the distance between leaves.

There are two ways to affect the distance an adversary has to overcome: by the distance necessary to cross the split threshold and by the distance between the split threshold and the benign prediction space. Instead of minimising the probability of malicious samples appearing in the benign prediction space, it might be more cost-effective to directly increase the distance between malicious samples within malicious classifying leaves and benign prediction space. The idea is illustrated in figure 5.3. On the left side samples are divided based on the best separation of classes, while on the right side the distance between the threshold and the corresponding boundary is optimised. It can be noted that in the first case, an adversary has to merely cross the threshold in order to craft an adversarial example as the boundary is aligned with the split threshold itself. To avoid such situations, the distance between the boundary and the threshold can be optimised as shown in the second case. As a result, the adversary needs to overcome a larger distance to craft adversarial examples for malicious samples in the left child node.

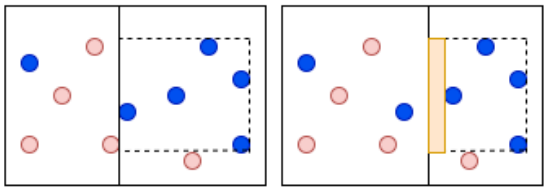


Figure 5.3: Different possible splits with the resulting threshold distance coloured yellow.

This approach thus comes down to maximising the distance between the splitting threshold and the corresponding boundary, which can be written as an objective function. Let B_x be the set of boundaries belonging to child node x, with bf the boundary feature corresponding with the split feature, b_L the lowerbound, b_H the upperbound and s the splitting threshold. The objective function can then be written as follows:

$$\text{maximise } ((|s - b_H||b \in B_L) + (|b_L - S||b \in B_R)) \tag{5.2}$$

The optimisation of the objective function could result in undesired behaviour. When only one side of the split contains boundaries, it is possible for the largest distance to exist between a single (or few) sample(s) and the corresponding boundary. This results in almost all samples in the parent node to end in only one of the child nodes, causing the tree to require more splits to separate the classes well. A solution would be to require boundaries on both sides of the split threshold but this would force malicious leaves to contain benign samples which is also undesired. As a solution, a parameter ρ is introduced with $0 \leq \rho \leq 1$. This parameter defines the fraction of samples that should at least end up in both child nodes, preventing splits separating only a single (or few) sample(s) using this criterion. As in the case of the previous approach, the splitting process might find several optimal splits. Using the same reasoning, the original splitting criterion acts as tie-breaker.

Maximising the distance between boundary and threshold comes at the cost of additional operations per split, however, it is less expensive than the previous approach. For each potential split, only the difference between two values has to be computed and this operation can be performed in $\mathcal{O}(1)$ time.

Compared with the previous approach, no considerable cost-wise penalty, on top of the boundary construction, is taken when optimising the robustness by maximising the boundary-threshold distance.

5.3. Integration

In the previous section different approaches were described to optimise the DTs robustness by addressing the identified factors. The last two approaches replace the original splitting criterion by one that aims to optimise a specific factor. However, these criteria do not aim to best separate the different classes, which could negatively impact the classifier's performance. To negate the negative impact on the classifier's performance, two different approaches are explored that allow a trade-off between accuracy and robustness. Section 5.3.1 describes a manual approach, allowing the user to decide what is most important in her/his situation. On the other hand, an automatic approach is described in section 5.3.2, allowing an algorithm to decide on a best trade-off. These approaches are meant to be used in combination with those described in the previous section, constructing a complete robustness improving procedure for DTs.

5.3.1. Minimum threshold

The original splitting criterion aims at minimising the probability of a random misclassification by the DT. This criterion is replaced by those introduced in the previous section. These new criteria optimise a specific factor contributing to the DTs robustness but do not necessarily lead to a good separation of classes. This could lead to a negative impact on the classifier's performance by an increase in the fraction of benign samples ending up in malicious classifying leaves. To guard the usability of the classifier, robustness could be traded for accuracy.

To control the trade-off between classification performance and robustness, a *minimum threshold* could be set to decide when a split should be made using the original splitting criterion or the newly introduced criterion that optimises robustness. Integrating this parameter in the algorithm can be achieved in several ways but focusing on a single criterion could lead to sub-optimal solutions. For example, let the minimum threshold be defined as the minimum gini improvement (in percentage) for the best split using the original splitting criterion (original split). This metric takes into account the quality of the separation of classes but ignores potentially better splits. There could be a split, using the optimisation criterion (robust split), that also has a gini improvement above the minimum threshold but with a higher robustness improvement than the original split. The combination of improvements could be more desired than only improving the quality of class separation. To include both criteria in the equation, the difference in gini improvement between the original split and robust split is used as a metric. In terms of gini improvement, this *split difference* can be defined as:

$$\text{Split difference: } \frac{|GI_{bs}| - |GI_{rs}|}{\max(|GI_{bs}|, |GI_{rs}|)}$$

where bs refers to the best split using the original splitting criterion and rs refers to the robust split. It should be noted that, although the *split difference* has been defined for gini impurity, it is trivial to construct a similar equation for other splitting criteria, e.g. information gain.

The parameter the user sets in terms of the *minimum threshold*, is thus the minimum difference in gini improvement necessary to perform a split based on the original splitting criterion. When the split difference becomes smaller than the minimum threshold, the user accepts that the improvement of robustness has to take priority over classification performance. The selection between the two types of splits can be written in the following form:

if *minimum threshold* < *split difference* then perform original split, otherwise perform robust split

5.3.2. Greedy Improvement

The trade-off issue can be solved by setting a minimum threshold but this introduces another parameter to be fine-tuned. An alternative solution could be to allow the algorithm itself to choose whether to use the original splitting criterion or one of the metrics introduced in the previous section. To this end, I suggest a heuristic that can be used by the algorithm to make this decision.

To choose between an original split and a robust split, the effects of both have to be understood. An original split leads to an improvement in the separation of classes, bringing the tree closer to the optimal state of separation. An original split could also result in purer benign prediction space as a result of moving malicious samples to a different child node. A robust split improves the robustness, either through purer benign prediction space or a larger distance between the split threshold and corresponding boundary of the benign prediction space. It does not necessarily lead to a better separation of classes and might even result in worsening it. Thus depending on the robustness metric used, the original split could optimise both where it is likelier for the robust split to focus on robustness alone.

To understand the trade-off between the splits, improvements for both metrics should be considered. The *split difference*, as defined for the *minimum threshold* approach, is the difference in gini improvement between the original split and robust split. On the other hand, the potential gain in robustness can be defined as the difference in robustness improvement between the robust split and original split, in the same manner as the *split difference* is defined. The split can then be made by greedily choosing the criterion matching the largest gain. In case the difference in gini improvement between the two splits is larger than the difference in robustness improvement, the original splitting criterion is used. When the difference in robustness improvement is larger, the robust split is performed. The more splits are performed, thus the deeper the tree, the likelier that both metrics have been improved. Effects of choices made using this heuristic might be larger for trees with smaller depths, resulting in larger differences between classification performance and robustness that could be observed.

It would be natural, in case of *minimising malicious samples*, to define the robustness improvement in terms of the decrease in malicious samples that fall in the benign prediction spaces. Let *M* be the number of malicious samples within the benign prediction space, the robustness improvement, *RI* can then be written as:

$$RI = \frac{M_{\text{parent}} - M_{\text{left child}} - M_{\text{right child}}}{M_{\text{parent}}} \tag{5.3}$$

where M_{parent} in case of the root node is equal to the number of malicious samples within when boundaries would be placed around all benign samples within that node. If a parent node was split using the original splitting criterion, the number of malicious samples that would have ended up within the corresponding boundaries of the split (those belonging to that division of samples) could be used. A similar construction does not make sense for the other optimisation approach, because of possible different involved features. To compare robust split candidates, distances can be measured against the total feature length. The total feature length is the difference between the maximum and minimum values for a feature over all samples in the root node. Let t be the threshold value, b the corresponding boundary value and tfl the corresponding total feature length. The robust ”improvement” can then be written as:

$$RI = \frac{|t - b|}{tfl} \tag{5.4}$$

The *coëfficient of variation* is a unitless metric showing the extent of variability, in terms of the improvements this would mean that a higher value is equal to a larger gain. It can be used to compare the differences in the respective gains and is defined as follows:

$$c_v = \frac{\sigma}{\mu} = \frac{\sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}}{\frac{1}{N}\sum_{i=1}^{N}(x_i)} \rightarrow \frac{\sqrt{(x_1 - \overline{x})^2 + (x_2 - \overline{x})^2}}{\frac{x_1 + x_2}{2}} \tag{5.5}$$

where x_1 and x_2 are the respective improvements for either gini improvement or robustness improvement. The automatic approach thus greedily chooses the highest obtainable gain to decide the type of splitting criterion to use.

5.4. Evaluation Setup

With the approaches explained, the next step is describing the method of evaluation to answer the research questions as described in section 1.2. First, general choices that affect all evaluations are explained in subsection 5.4.1, after which each following subsection describes the evaluation process to answer the relevant research questions in order as formulated earlier in this work.

5.4.1. Tools, datasets and robustness

Three choices impact each evaluation in this work, they are: the implementation to construct a DT, the datasets to evaluate the approaches and the metric used to measure robustness. Each of them affects further comparisons with other works in the field, among which state-of-the-art research.

There are two options regarding the implementation for constructing a DT: (1) write an own implementation or (2) modify an existing implementation, e.g. in Python or R. An advantage of the former is that it simplifies the implementation of the approaches as no other dependencies with existing code have to be taken into account. However, this could result in different implementations between this work and (future) state-of-the-art research, making eventual comparisons less reliable. The second choice requires modifying an existing codebase but would allow evaluation of the effects of the approaches using an implementation currently used in both academia and industry. As using the same implementation improves the reliability of a comparison between works, an existing implementation will be modified in this work. Two widely known implementations of a DT are written in Python and R. In this work the Python implementation by Scikit-learn[1] is used. The reasoning for this is more personal experience with programming in Python, which could help the process go smoother.

To evaluate the approaches introduced in this work, it would be best to test it against a benchmark. This would make it easier to compare results with other work that has been evaluated against it. However, currently there exists no such (binary classification) benchmark and defences for DTs are evaluated on different datasets of choice by their authors. As a result, the datasets chosen for evaluation in this work were hand-picked based on their applicability, binary classification, and usage for evaluation by state-of-the-art works. As the *minimising malicious samples* approach' run-time complexity is largely affected by the size of the dataset and the number of features, smaller datasets with fewer features are preferred. An overview of the datasets used for evaluation in this work can be found in table 5.1, which includes the category (based on size), number of samples, number of features and sources (including state-of-the-art works).

Name	Category (Size)	#Samples	#Features	Source
banknote	small	1.372	4	[30]
breast-cancer	small	683	10	[22]
diabetes	small	768	8	[22, 57]
ionosphere	small	351	32	[22, 57]
wine	medium	5.847	12	[21, 57]

Table 5.1: Overview of datasets used within this work.

There are several metrics used to measure robustness in other works, including: L_0, L_1, L_2 and L_∞ distance norms [22, 40], (adversarial) accuracy, F1 scores [21, 57] and ROC curves [21]. The choice of metric depends on the adversary, e.g. measuring accuracy on a maximally perturbed dataset when dealing with an unbounded adversary does not provide any insights. In the threat-model, chapter 4, is stated that in this work an adversary attempts to minimise the distance between adversarial examples and the original samples. This adversary is also not limited by its capabilities in order to measure the worst-case effect on the classifier. The combination of these two aspects led to the choice of measuring robustness in terms of the L_1 distance norm, which is the total amount of perturbations required to craft an adversarial example. This metric can be affected by the range of features, which is why all features are normalised between 0 and 1. An assumption is made that no difference in difficulty to perturb

[1] This refers to version 0.22.2, which can be found at https://github.com/scikit-learn/scikit-learn/

features exist to maintain simplicity. This metric might indicate the difficulty for an adversary to craft an adversarial example but it does not measure whether an adversary has to craft an attack. A classifier might already misclassify all relevant samples, resulting in an adversary not having to take any action at all. To call such a classifier robust, based on the difficulty in crafting an attack, would be partly misleading. This is why also the False Negative Rate (FNR) is considered in this work, to give a better insight in to the robustness of the classifier.

5.4.2. Verification of approaches

The proposed optimisation approaches in this work are meant to improve the identified factors that contribute to the classifier's robustness. Before evaluating their effects in terms of classification performance and accuracy, their effects based on the factors are to be verified.

To verify the effects of the approaches, results of a standard DT (sklearn) are compared with those from DTs trained using: *Limited Benign Prediction Space* (lbps), *Minimising Malicious Samples* (mms) and *Maximising Distance* (md). The effects are analysed for three datasets: *breast-cancer*, *diabetes* and *wine*, where all correctly classified malicious samples are attacked by an adversary using algorithm 1. The dataset is split in a training and test set, respectively 80% and 20% of the data, using stratified sampling and each algorithm is run for depths 1 to 10 on each dataset. The ρ parameter for *maximising distance* is set to 0.1. To verify the working of the approaches, the following metrics are measured: *Mean number of features modified for an attack*, *Mean fraction of malicious training samples in benign prediction space* and *Mean shortest distance from malicious leaves to any benign leaf*.

5.4.3. Optimisation & Trade-Off Approaches

The integration into the learning algorithm consists of both an *optimisation* approach and a *trade-off* approach. Both the optimisation approaches independently and in combination with the trade-off approaches are evaluated.

To evaluate the effects of the optimisation approaches, results of a standard DT (sklearn) are compared with those from DTs trained using: *Minimising Malicious Samples* (mms) and *Maximising Distance* (md). The effects are analysed for all datasets in table 5.1: *banknote*, *breast-cancer*, *diabetes*, *ionosphere* and *wine*, where all correctly classified malicious samples are attacked by an adversary using algorithm 1. The dataset is split in a training, validation and test set, respectively 80%, 10% and 10% of the data, using stratified sampling. Hyper-parameters are tuned for accuracy using the validation set, where only the best performing model is run against the test set. The depth is varied between 1 to 10 and the ρ parameter for *maximising distance* is varied from 0.05 to 0.25 in steps of 0.05. The accuracy, L_1 distance, False Negative Rate (FNR) and False Positive Rate (FPR) are measured.

To evaluate the effects of the combination of approaches, results of a standard DT (sklearn) are compared with those from DTs trained using: *Minimising Malicious Samples with Greedy Improvement* (mms-GI), *Minimising Malicious Samples with Minimum Threshold* (mms-MT), *Maximising Distance with Greedy Improvement* (md-GI) and *Maximising Distance with Minimum Threshold* (md-MT). The experiment follows the same procedure as for the optimisation approaches with the threshold value, during validation, been varied from 0.1 to 0.5 in steps of 0.1.

5.4.4. Random Forest

The proposed approaches are meant to be applied in a combination of optimisation approach with trade-off approach, which is why only the combinations are evaluated for a Random Forest (RF).

To evaluate the effects of the combination of approaches, results of a standard RF (sklearn) are compared with those from RFs trained using: *Minimising Malicious Samples with Greedy Improvement* (mms-GI), *Minimising Malicious Samples with Minimum Threshold* (mms-MT), *Maximising Distance with Greedy Improvement* (md-GI) and *Maximising Distance with Minimum Threshold* (md-MT). The effects are analysed for three datasets: *breast-cancer*, *diabetes* and *wine*, where all correctly classified malicious samples are attacked by an adversary.

To compute minimum adversarial examples, an approach based on Chen et al's [23] algorithm 2 is

used. The algorithm considers each leaf node in a tree as a node in a multi-partite graph with edges between the nodes when their prediction space, restricted by the boundaries and splitting rules, overlap. As the prediction space of leaf nodes within a tree cannot overlap, a maximum-clique in the multi-partite graph represents a prediction by the tree ensemble. These maximum cliques can be found by iterating over all the trees and merging overlapping nodes, where the merged nodes over all trees in the end represent the prediction space of an ensemble's leaf. The number of maximum cliques depends on the number of leaves in the individual DTs and the number of trees in the RF, where an adversary has to iterate over all maximum cliques in order to find the minimum adversarial example. The number of cliques put constraints on both running time and space, limiting the experiment to the aforementioned datasets and only considering a RF consisting of 5 trees. There only exists the attack by Kantchelian et al. [40] as an alternative for computing exact minimum adversarial examples against a tree ensemble but the additional boundaries within the leaf prevent this algorithm from being used.

The dataset is split in a training, validation and test set, respectively 80%, 10% and 10% of the data, using stratified sampling. Hyper-parameters are tuned for accuracy using the validation set, where only the best performing model is run against the test set. The depth is varied between 1 to 5 and the ρ parameter for *maximising distance* is varied from 0.05 to 0.15 in steps of 0.05. The variety is less compared to previous experiments due to the constraints introduced by the attack algorithm. The accuracy, L_1 distance, False Negative Rate (FNR) and False Positive Rate (FPR) are measured.

5.4.5. Comparison with state-of-the-art research

The last experiment in this work is the comparison with state-of-the-art work by Vos [57], further referred to as rfdt (Robust and Fair Decision Tree). The reason for this is that other state-of-the-art works are implemented for different types of classifiers, e.g. GBDT, had not made their source-code available or were outperformed by his work. The experiment compares the approaches for both a single DT and a RF.

To compare the effects of the approaches in case of a single DT, results of a standard DT (sklearn) are compared with those from DTs trained using: *Minimising Malicious Samples with Greedy Improvement* (mms-GI), *Minimising Malicious Samples with Minimum Threshold* (mms-MT), *Maximising Distance with Greedy Improvement* (md-GI), *Maximising Distance with Minimum Threshold* (md-MT) and Vos' approach (rfdt). The effects are analysed for three datasets: *diabetes*, *ionosphere* and *wine*. All correctly classified malicious samples are attacked by an adversary using threat-models defined by Vos per dataset to limit the adversary's capabilities. The dataset is split in a training, validation and test set, respectively 80%, 10% and 10% of the data, using stratified sampling. Hyper-parameters are tuned for *adversarial accuracy*, the accuracy on a successfully attacked dataset, using the validation set, where only the best performing model is run against the test set. Parameters are varied as in Vos' work, with depths from 1 to 5, fairness (for rfdt) set to 0 and the ρ parameter for rfdt taking values from 0.0 to 1.0 in steps of 0.25. The ρ parameter for *maximising distance* is varied from 0.05 to 0.25 in steps of 0.05 and the minimum threshold is varied from 0.1 to 0.9 in steps of 0.1. The accuracy and *adversarial accuracy* are measured.

To better understand the relation between the state-of-the-art approach and the identified factors contributing to robustness, its impact is measured on the *diabetes* and *wine* dataset. The baseline (sklearn), state-of-the-art technique (rfdt) and best performing approach in this work are compared in their best performing settings using the validation set to measure the same metrics as in the verification experiment.

To compare the effects of the approaches in case of a RF, results of RFs trained with the same approaches as aforementioned are compared. The same datasets are used and RFs consisting of 5 and 10 trees are considered. The rest of the evaluation follows the same procedure as mentioned above. In this experiment it is possible to consider a tree ensemble consisting of 10 trees as the *adversarial accuracy* is measured instead of the L_1 distance. This does not require iterating over all the maximum cliques but only over those till one is found that leads to misclassification of the malicious sample.

6

Results

This chapter discusses the results of the previously described experiments. First, the hypotheses for each of the experiments are stated. This is followed by the results of the verification experiments, which tests the effects of the optimisation approaches against the identified factors. Third, the results of the optimisation approaches, with and without trade-off approaches, are shown and elaborated upon. After discussing the experiments related to a single DT, as fourth the results of the approaches used to build a RF are discussed. This chapter ends discussing the results of the comparison of the approaches with state-of-the-art work by Vos [57].

6.1. Hypotheses

At the start of the experiments, several hypotheses were stated regarding the behaviour of several approaches and their effects on either the identified factors, the accuracy or robustness.

Verification of approaches

In the verification experiment, four algorithms are compared with each other: Standard Sklearn gini impurity (sklearn), Limited Benign Prediction Space (lbps), Minimising Malicious Samples (mms) and Maximising Distance (md). The effect that each of these algorithms has on the identified factors is computed and discussed later on. As the different approaches (lbps, mms, md) are designed to improve a certain factor, different hypotheses can be stated.

The lbps approach limits the benign prediction space in an attempt to cause an increase in the number of features an adversary has to consider when crafting adversarial examples. Depending on the features, whether they distinguish strongly between the classes or not, this could lead to an increase in the robustness of the classifier. This can be stated as follows:

Hypothesis *A DT trained with lbps requires more features to be modified by an adversary to craft adversarial examples than a DT trained with sklearn.*

The mms approach combines the lbps approach, which limits the benign prediction space, with an optimisation to reduce the impurity of this space. This means that the optimisation should lead to a lower fraction of malicious training samples falling in benign prediction space. On top of the previous hypothesis, this also leads to the following hypothesis:

Hypothesis *A DT trained with mms has a smaller fraction of malicious training samples in benign prediction space than a DT trained with sklearn or lbps.*

The md approach combines the lbps approach with an optimisation to increase the distance between leaves. In turn this should increase the distance from a malicious leaf to benign prediction space. Next to the hypothesis stated for lbps, the following hypothesis applies to md:

Hypothesis *A DT trained with md has a larger, average, minimum distance from a malicious leaf to benign prediction space than a DT trained with sklearn or lbps.*

Optimisation & Trade-Off Approaches

In this experiment, the sklearn, mms and md algorithms from the previous experiment are compared with each other based on their accuracy and robustness. The mms and md approach are also combined with two trade-off approaches: Greedy Improvement (GI) and Minimum Threshold (MT). The combination with a trade-off approach is indicated by a tag consisting of a dash and the latter's abbreviation behind the name, e.g. mms-GI.

The mms approach should lead to fewer false negatives, while increasing the number of features an adversary has to modify and their distance by the reduced benign prediction space. This leads to the following hypothesis:

Hypothesis *A DT trained with mms achieves the following performances compared with a DT trained with sklearn:*

- *A smaller number of false negatives*
- *A larger number of false positives*
- *A higher average L1 robustness*

The md approach should lead to a larger distance between malicious leaves and benign prediction space, resulting in a larger robustness. The following hypothesis applies:

Hypothesis *A DT trained with md achieves the following performances compared with a DT trained with sklearn:*

- *A larger number of false positives*
- *A higher average L1 robustness*

The trade-off approaches, both GI and MT, attempt to increase the accuracy of a DT trained using mms or md by trading it for robustness. This leads to the following hypothesis:

Hypothesis *A DT trained using mms or md in combination with GI or MT achieves the following performances compared with a DT trained using the relevant optimisation approach without GI or MT:*

- *A higher accuracy*
- *A lower average L1 robustness*

Random Forest

In this experiment, a RF consisting of DTs trained with sklearn are compared with RFs consisting of DTs trained with mms-GI, mms-MT, md-GI and md-MT. Similar hypothesis can be stated as for the previous experiment, in which the subject ”a DT” should be exchanged for ”a RF”. For conciseness, the hypotheses are not repeated.

Comparison State-of-the-art

In the comparison with state-of-the-art algorithm by Vos [57] a difference should be expected by their different optimisation approaches. Where the state-of-the-art (sota) technique optimises the worst-case performance of sklearn, the approaches proposed in this work optimise identified factors. As their robustness metric differs from the one used in this work by focusing on *adversarial accuracy*, only hypotheses for the combinations of optimisation and trade-off approaches are given.

State-of-the-art optimises the original gini impurities worst-case performance but does not necessarily affect the fraction of malicious training samples in benign prediction space. This leads to the following hypothesis:

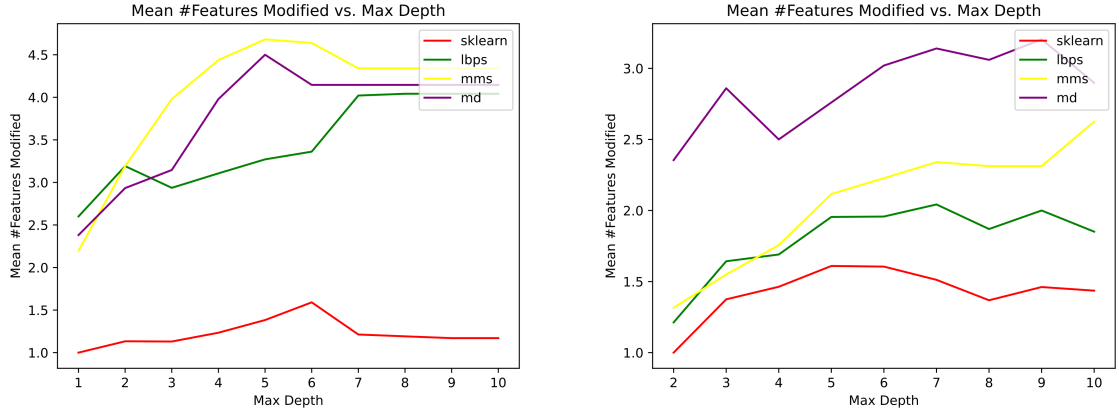
Hypothesis *A DT/RF trained using mms in combination with GI or MT has a smaller number of false negatives than a DT trained using sota.*

Hypothesis *A DT/RF trained using mms or md in combination with GI or MT has a lower or equal adversarial accuracy than a DT trained using sota.*

6.2. Verification of approaches

Several approaches were described in section 5.2. These approaches should improve the robustness of a DT by addressing the earlier identified factors contributing to its robustness. Discussing the verification results for these approaches, similar abbreviations are used as stated in the previous section: baseline (sklearn), *limited benign prediction space* (lbps), *minimising malicious samples* (mms) and *maximising distance* (md).

As all approaches include the lbps approach, the first hypothesis to test is whether this causes the adversary to modify more features to craft adversarial examples. The effect of the approaches on the average number of features modified is shown for two datasets in figure 6.1. The results show a clear difference in the average number of modified features between the baseline and methods using lbps for all tested tree depths. These results confirm the hypothesis that an adversary needs to modify more features after applying this approach.

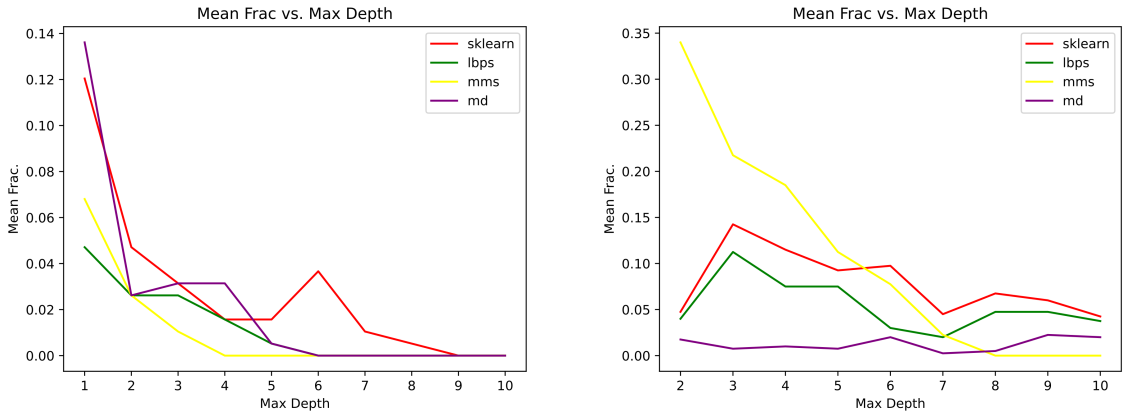


(a) Dataset: breast-cancer

(b) Dataset: diabetes

Figure 6.1: Average number of modified features to craft adversarial examples. Sklearn = standard tree, lbps = limited benign prediction space, mms = minimising malicious samples, md = maximising distance.

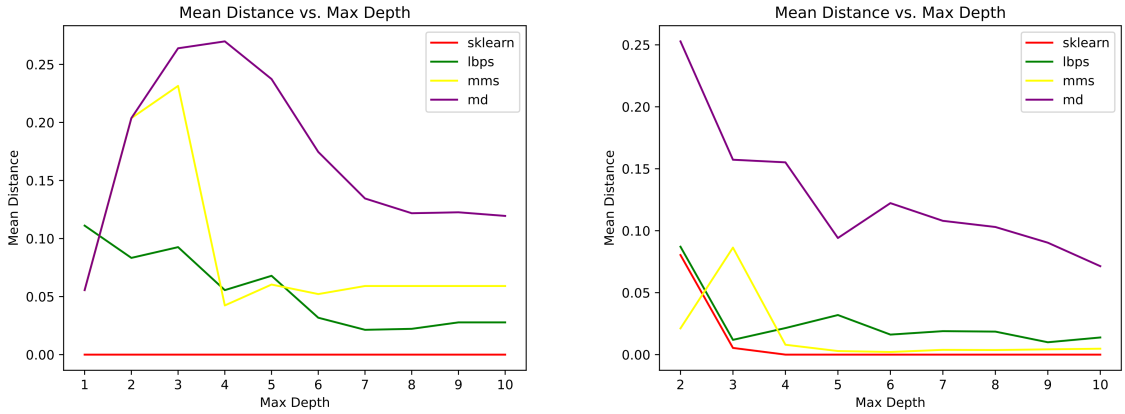
The second hypothesis states that a DT trained with mms has a smaller fraction of malicious training samples in benign prediction space compared to sklearn. Limiting the benign prediction space might cause malicious samples that fall within the benign classifying leaf to be left outside of this carved out space. This is expected to increase the purity of the benign prediction space, where *mms* attempts to further improve this. Figure 6.2 shows the average fraction of malicious training samples that falls within benign prediction space (average frac mal.). The results show that applying lbps leads to a faster decrease in the average frac mal. compared to the baseline for all tree depths. The mms approach should lead to a further decrease but for the diabetes dataset this only occurs from depth 7 onwards, with a higher average frac mal. than the baseline and lbps for lower depths. This might be the result of the criterion not aiming to separate the classes, keeping malicious samples in leaves dominated by benign training samples. It indicates that it might be necessary to also separate the classes better in order to decrease the average frac mal. for lower depths too. As the approach does achieve to decrease the average frac mal. further at higher tree depths compared to both baseline and lbps, the hypothesis can be considered as confirmed.



(a) Dataset: breast-cancer (b) Dataset: diabetes

Figure 6.2: Average fraction of malicious training samples within benign prediction space. Sklearn = standard tree, lbps = limited benign prediction space, mms = minimising malicious samples, md = maximising distance.

The last hypothesis states that training a DT with md results in a larger minimum distance between malicious leaves and benign prediction space. Figure 6.3 shows the average minimum distance for the different approaches for two datasets. The baseline shows a distance close to zero, indicating that the closest benign leaves share the same parent node as the malicious leaves. This leads them to only be a single threshold value apart from each other. The md approach shows an increase in the average minimum distance for all tree depths by more than tripling it. Thus the results confirm the hypothesis that the md approach further increases this distance compared to the baseline and lbps.



(a) Dataset: breast-cancer (b) Dataset: diabetes

Figure 6.3: Average minimum distance between malicious leaves and benign prediction space. Sklearn = standard tree, lbps = limited benign prediction space, mms = minimising malicious samples, md = maximising distance.

6.3. Optimisation & Trade-Off Approaches

The verification experiment confirmed that the proposed approaches improve the factor they were designed to improve. As the factors contribute to the robustness of a classifier, the approaches should improve the classifier's robustness as a result. Discussing the results for the optimisation and trade-off approaches, similar abbreviations for the approaches are used as in the previous experiment.

Optimisation

The accuracy and L1 robustness for the different approaches are shown in table 6.1. For all datasets, the proposed approaches achieve a higher, average, L1 robustness compared with the baseline. The cost in terms of accuracy are either minimal, as for mms, or up to ≈10% for md. In the latter case, this decrease in accuracy goes hand-in-hand with at least a doubling (close to tripling for wine) of the robustness. The md approach consistently increases the robustness compared with lbps, whereas mms could achieve a slightly lower L1 distance than lbps. On the other hand, mms leads to better accuracy than md which can be attributed to its focus on further decreasing the false negative rate.

Dataset	Sklearn		lbps		mms		md	
	Acc.	L1	Acc.	L1	Acc.	L1	Acc.	L1
banknote	0.986	0.108	0.891	0.207	0.942	0.161	0.899	0.223
breast-cancer	0.957	0.042	0.942	0.788	0.957	1.074	0.913	1.330
diabetes	0.753	0.057	0.727	0.091	0.792	0.130	0.649	0.250
ionosphere	0.861	0.180	0.944	0.726	0.889	1.314	0.917	0.855
wine	0.744	0.050	0.723	0.079	0.723	0.074	0.689	0.139

Table 6.1: Accuracy and L1 robustness (distance) scores for a Decision Tree with the malicious class attacked. Classifiers were optimised for (1) accuracy and if equal (2) L1 robustness on a validation set and tested on a separate test set. Baseline (Sklearn), Limited benign prediction space (lbps), Minimising malicious samples (mms) and Maximise Distance (md).

The effects of the approaches can be better understood when taking a closer look at the False Negative Rate (FNR), figure 6.4 and False Positve Rate (FPR), figure 6.5, for diabetes and wine. The results for the other datasets can be found in appendix figures B.1 and B.2. As a result of making the benign prediction space smaller, the FPR increases for the proposed approaches as more benign samples are misclassified as malicious. The effect of the increasing FPR on the accuracy seems to be (partly) negated by the decrease in FNR. This decrease is not visible for mms at lower depths for the wine dataset, as observed and explained in the verification experiment, but the only slight increase in FPR prevents the accuracy to drop.

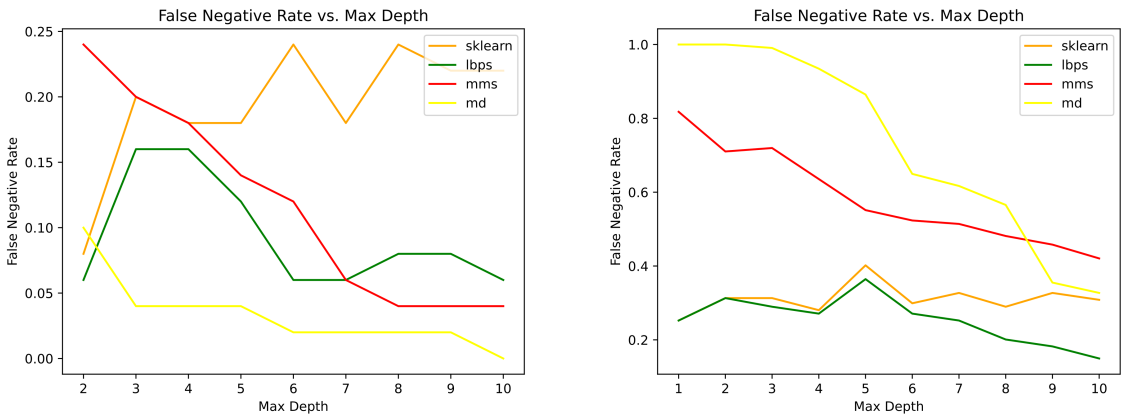


Figure 6.4: False negative rate for diabetes (left) and wine (right) on the validation set with md frac 0.2 (diabetes) and 0.05 (wine).

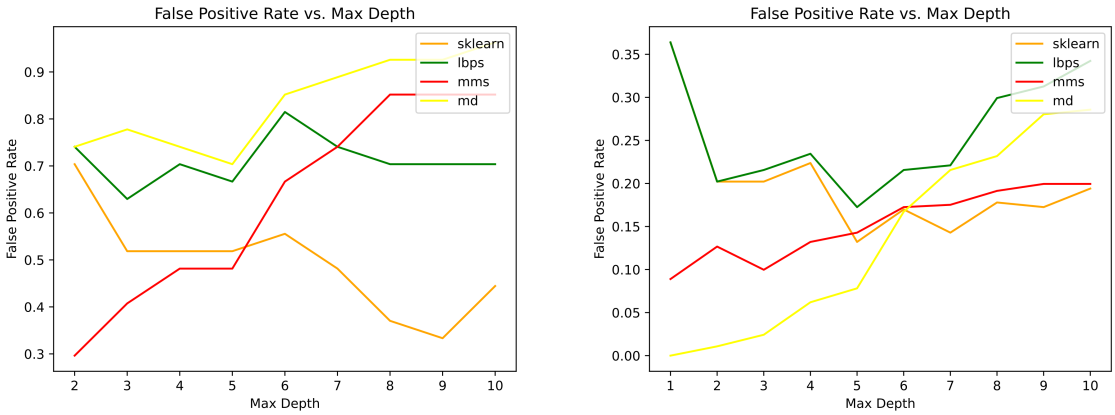


Figure 6.5: False positive rate for diabetes (left) and wine (right) on the validation set with md frac 0.2 (diabetes) and 0.05 (wine).

The hypotheses stated for this experiment can be confirmed with the observed results. The average L1 robustness is in all cases higher when training a DT using mms or md. The FNR and FPR show that the effects of the methods lead to a lower FNR for mms, dependent on the tree depth, and a higher FPR for both mms and md, as was expected. Thus this concludes that the optimisation approaches function as designed and lead to robuster models.

Trade-Off

To negate the negative effect an optimisation approach might have on the classifier's classification performance, trade-off approaches can be used. The optimisation approaches can be combined with a trade-off approach, which allows both using the optimisation approach and the original Gini Impurity criterion to grow the tree. This should not only lead to a decrease in FNR but also limit the increase of FPR.

The results for the combination of approaches are shown in table 6.2. The results indicate that combining mms with a trade-off approach does not improve the classifier's accuracy and even lowers the classifier's robustness. On average, mms-GI results in a 3.78% drop in accuracy and mms-MT in a 2.64% drop in accuracy compared with just mms. This is unexpected as the trade-off approaches attempt to improve the accuracy by balancing mms with gini impurity. On the other hand, combining md with a trade-off approach does show improvement in the classifier's accuracy at the cost of its robustness. On average, md-GI leads to a 2.56% higher accuracy and md-MT to a 5.06% higher accuracy. Combining md with *greedy improvement* leads to higher robustness, while a combination with *minimum threshold* achieves higher accuracy. It should be noted that the robustness does stay at least equal to the baseline or remains higher for all combination of approaches.

Dataset	mms-GI		mms-MT		md-GI		md-MT	
	Acc.	L1	Acc.	L1	Acc.	L1	Acc.	L1
banknote	0.950	0.141	0.928	0.164	0.928	0.167	0.993	0.127
breast-cancer	0.942	0.938	0.942	1.350	0.942	1.250	0.942	1.214
diabetes	0.714	0.102	0.740	0.103	0.675	0.169	0.779	0.060
ionosphere	0.889	1.314	0.833	0.150	0.917	0.903	0.917	0.883
wine	0.709	0.078	0.728	0.064	0.733	0.096	0.689	0.139

Table 6.2: Accuracy and L1 robustness (distance) scores for a Decision Tree with the malicious class attacked. Classifiers were optimised for (1) accuracy and if equal (2) L1 robustness on a validation set and tested on a separate test set. Minimising malicious samples (mms), Maximise Distance (md), Greedy Improvement (GI) and Minimum Threshold (MT).

The trade-off approaches should offer a more diverse set of accuracy-robustness combinations but unexpectedly this did not occur for mms. As the accuracy metric does not give further insights into the cause, the FNR and FPR for these approaches on the wine dataset are shown in figure 6.7. The results for the other datasets can be found in appendix figure B.3. The figures show that the trade-off approaches manage to further reduce the FNR but that the accuracy does not improve as a result of an even further increasing FPR. An explanation for this is that the trade-off approaches might cause the robust splits to occur closer to the leaves of the trees. As the original split attempts to best separate the classes, the limited benign prediction space might be restraining the classifier's generalisation ability more compared to the benefit of minimising the malicious samples in the benign prediction space. Further support for this explanation is shown in figure 6.6, which shows a Pareto frontier plot for all possible splits of a node for the wine dataset and their gini improvement and robustness improvement. The plot indicates that the best original split does not necessarily lead to the worst improvement in robustness. Remembering the criteria used for the trade-off approaches, this would lead to a lower gain in robustness compared to the gain in gini and thus to an original split taking place in case of *greedy improvement*. The large difference in gini improvement between the best original split and the best robust split would also cause the difference to remain above the *minimum threshold* and thus again leading to an original split taking place. This result indicates that it might be necessary to make the boundaries less restrictive in order to avoid large penalties in terms of FPR.

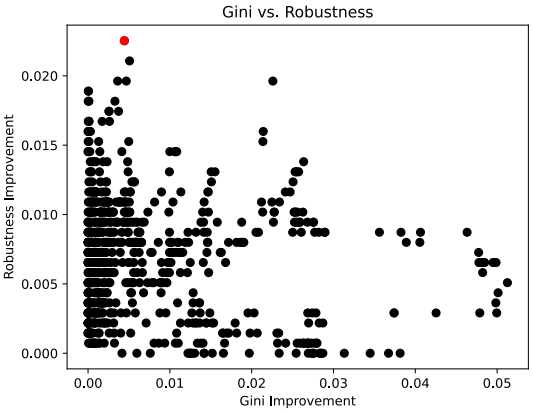


Figure 6.6: Pareto frontier plot of the robustness improvement and gini improvement for all possible splits of a node. The red-coloured dot is the possible robust split.

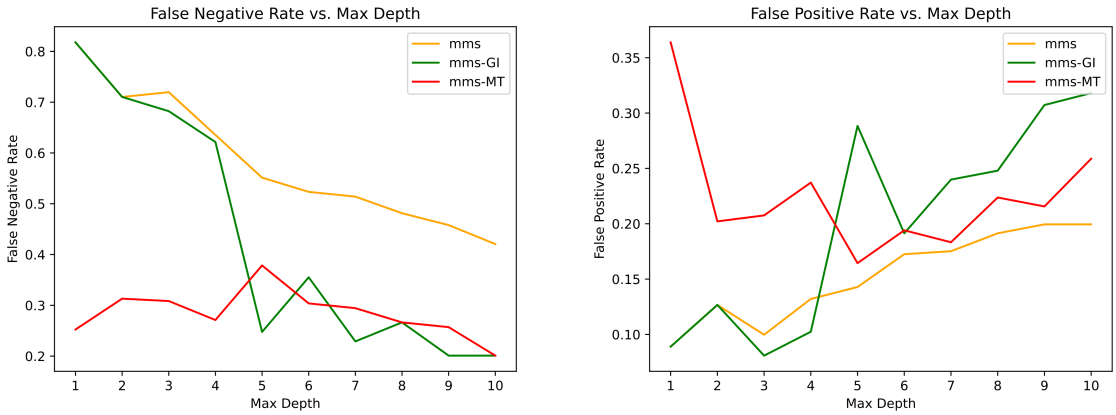


Figure 6.7: False Negative Rate (left) and False Positive Rate (right) for wine dataset.

The earlier stated hypothesis for the combination approaches can only be confirmed for md, whereas the accuracy does not improve for mms. Although the FNR decreases faster when using trade-off approaches, it also causes a faster increase in FPR as a result of original splits taking place earlier in the tree growing process. Thus this concludes that the trade-off approaches might not work as thought in all cases.

6.4. Random Forest

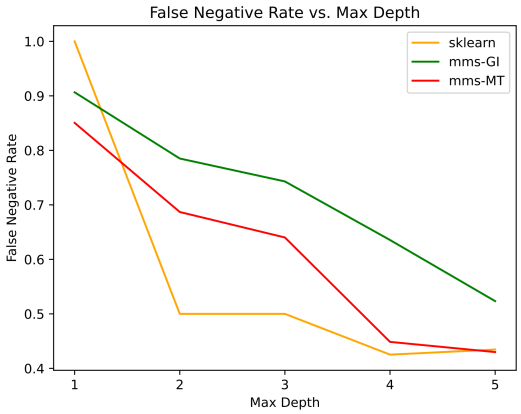
The previous experiment showed the effects of the different proposed approaches on the classifier's accuracy and robustness. To understand whether these effects also apply when combining several trained DTs in an ensemble, the proposed approaches are used to train a RF. Only the combination of approaches is used as they are supposed to achieve a better trade-off, even though results showed that this does not have to be the case for mms.

The results of applying the approaches for the construction of DTs used in a RF are shown in table 6.3. The L1 robustness is at least 1.5 times the baseline robustness for the combinations including GI with accuracy not dropping beyond 3%. The mms-GI approach outperforms mms-MT on both accuracy and robustness for all datasets. The md-GI approach outperforms md-MT on both breast-cancer and wine datasets but achieves lower robustness for the diabetes dataset. These results imply that combining an optimisation approach with *greedy improvement* might lead to better overall performance than using *minimum threshold* for a RF. However, these results could be a consequence of limiting the possible thresholds, although the specific threshold criterion might also be of influence.

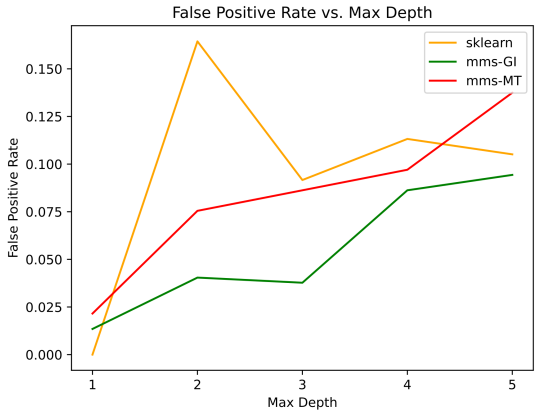
Dataset	sklearn		mms-GI		mms-MT		md-GI		md-MT	
	Acc.	L1	Acc.	L1	Acc.	L1	Acc.	L1	Acc.	L1
breast-cancer	0.957	0.298	0.957	1.380	0.942	1.894	0.971	1.901	0.928	1.443
diabetes	0.727	0.147	0.701	0.226	0.688	0.184	0.714	0.174	0.701	0.200
wine	0.745	0.036	0.732	0.058	0.730	0.040	0.725	0.053	0.742	0.048

Table 6.3: Accuracy and L1 robustness (distance) scores for a Random Forest with the malicious class attacked. Algorithms were optimised for (1) accuracy and if equal (2) L1 robustness on a validation set and tested on a separate test set. Baseline (sklearn), Minimising malicious samples (mms), Maximise Distance (md), Greedy Improvement (GI) and Minimum Threshold (MT).

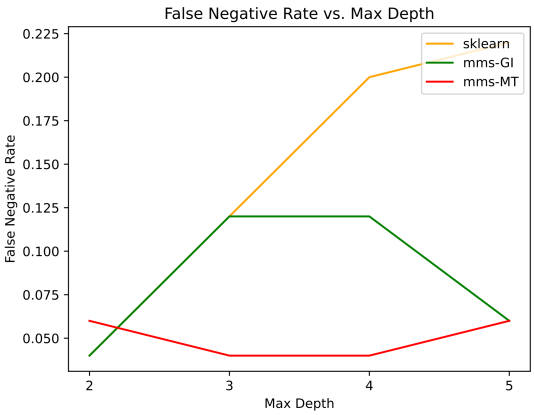
The trade-off approaches led to a faster decrease in FNR and increase in FPR for mms in the previous experiment. The same metrics are shown for an RF but then compared with the baseline, as shown in figure 6.8 for wine and diabetes. The results for the other dataset can be found in appendix figure C.1. The approaches lead to an increase in FNR compared with the baseline but at the same time also to a decrease in FPR for wine. This is contradictory to the expected effect of the approach but can be explained by the low tree depth for this dataset. A similar effect as in the previous experiment is shown for diabetes, where the FNR decreases and FPR increases as expected. However, the accuracy for mms-GI and md-GI decreases less for all datasets compared to the previous experiment. This indicates that using the combination of approaches in a RF can lead to better results for low tree depths.



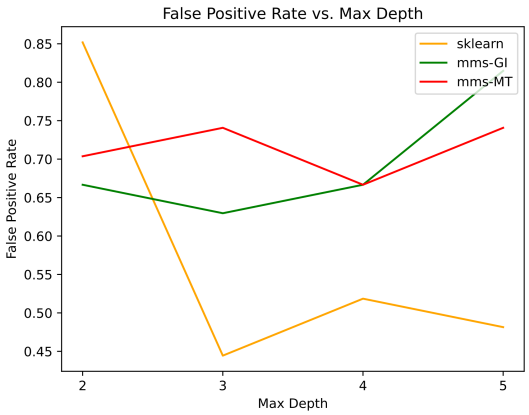
(a) Dataset: wine



(b) Dataset: wine



(c) Dataset: diabetes



(d) Dataset: diabetes

Figure 6.8: False Negative Rate (left) and False Positive Rate (right) for wine and diabetes datasets in case of a random forest with 5 trees.

The approaches consistently improve the L1 robustness for the tested datasets when compared with the baseline but do not necessarily increase the robustness compared to a single DT. For the wine dataset, the approaches used in a RF lead to a slightly lower robustness compared to their DT counterpart but this could be attributed to the depth parameter in this experiment. In the verification experiment it was shown that approaches might require deeper trees, depending on the dataset, before their effect is shown. Thus, given this understanding, it can be concluded that the approaches do result in more robust RFs.

6.5. Comparing with state-of-the-art research

The preceding experiments were meant to validate the effect of the proposed approaches, both for a single DT and a RF. As it is clear that the approaches improve the classifier's robustness, their effect can be compared with the state-of-the-art work. In this experiment the combination of approaches are compared with both the baseline (sklearn) and state-of-the-art work by Vos [57] (rfdt) for a single DT, a RF consisting of 5 DTs and a RF consisting of 10 DTs. As explained in the description of this experiment, section 5.4.5, the robustness is measured using *adversarial accuracy* for this experiment. This is the accuracy on a dataset that is maximally perturbed given the related threat model. The best performance of both accuracy and adversarial accuracy are made bold for each dataset.

Single Tree

The results for a single DT are shown in tables 6.4, approaches combined with GI, and 6.5, approaches combined with MT. The results indicate that mms-GI achieves on average a 3.17% higher regular accuracy than rfdt but rfdt achieves on average a 5.5% higher adversarial accuracy. The md-GI approach scores either equal to the baseline or worse, as shown for ionosphere and wine. As the approach did lead to more robust DTs in preceding experiments, this effect might be related to the threat-model specified by Vos and the low tree depths considered in this experiment. The approach forces a certain amount of samples on both sides of the threshold, which in turn could require deeper trees to separate the classes well.

The results for combinations including MT show worse performance than when GI is included. For diabetes, mms-MT only performs splits using the original splitting criterion while a better trade-off was found in a preceding experiment on the same dataset in the same setting but with different depths. This emphasises the importance of deeper trees for the proposed approaches to optimally work. This also indicates that including GI reduces the importance of the depth of the tree. The md-MT approach only shows improved performance on the ionosphere dataset but a closer look shows that the tree is of only 1 depth. Such a shallow tree might not perform well when more unknown data is present as the leaves are expected to be rather impure, leading to misclassifications.

Dataset	sklearn		rfdt		mms-GI		md-GI	
	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc
diabetes	0.610	0.571	0.662	**0.649**	**0.688**	0.610	0.610	0.571
ionosphere	0.833	0.778	0.833	**0.806**	**0.861**	**0.806**	0.750	0.694
wine	0.735	0.631	0.697	**0.689**	**0.738**	0.618	0.671	0.62

Table 6.4: Accuracy (R. Acc) and Adversarial Accuracy (A. Acc) for a Decision Tree. Algorithms were optimised for Adversarial Accuracy with the best scores for each category made bold. Robust and Fair Decision Trees (rfdt), Greedy Improvement (GI), Minimising Malicious Samples (mms), Maximise Distance (md).

Dataset	sklearn		rfdt		mms-MT		md-MT	
	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc
diabetes	0.610	0.571	**0.662**	**0.649**	0.610	0.571	0.610	0.571
ionosphere	0.833	0.778	0.833	0.806	0.778	0.694	**0.917**	**0.861**
wine	**0.735**	0.631	0.697	0.689	0.728	**0.702**	0.671	0.62

Table 6.5: Accuracy (R. Acc) and Adversarial Accuracy (A. Acc) for a Decision Tree. Algorithms were optimised for adversarial accuracy with the best scores for each category made bold. Robust and Fair Decision Trees (rfdt), Minimum Threshold (MT), Minimising Malicious Samples (mms), Maximise Distance (md).

This comparison experiment shows that mms-GI is able to compete with the state-of-the-art work on the tested datasets, where it is also able to achieve better regular accuracies under the assumed threat-model. However, the state-of-the-art work outperforms the proposed approaches on adversarial accuracy. The performance for mms-MT and the md combinations indicate that larger tree depths are required for these combinations to achieve performance as shown in earlier experiments.

The hypothesis for this experiment states that mms in combination with a trade-off approach achieves a lower FNR than rfdt. In figure 6.9 the FNR for both approaches, in their best performing settings, are plot for diabetes and wine. These figures confirm the hypothesis where the FNR is indeed lower for mms-GI than for rfdt. To better understand the total impact on the regular accuracy, the FPR for both approaches on the same datasets are shown in figure 6.10. These figures show that rfdt achieves a lower FPR compared to mms-GI and that both approaches achieve their accuracies based on minimising different metrics. It emphasises again that a robust model should not only decrease the FNR but also FPR in order to better compete with state-of-the-art research.

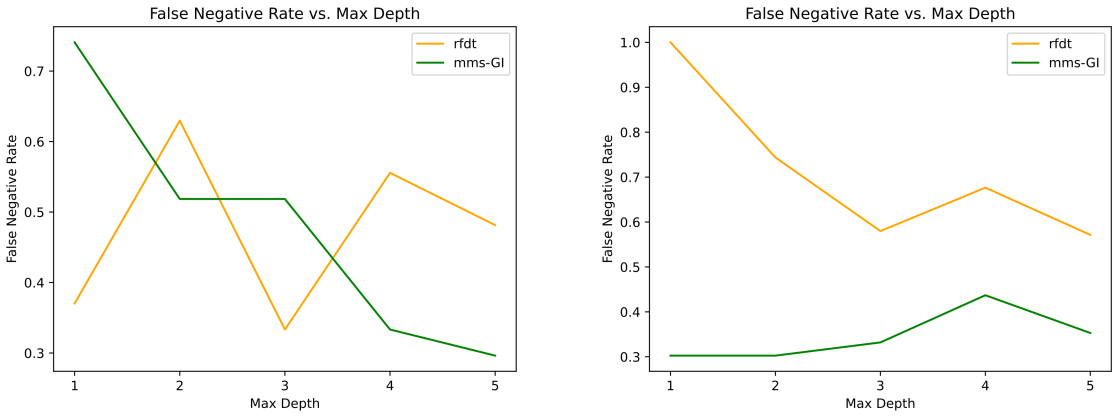


Figure 6.9: False negative rate for diabetes (left) and wine (right) on the validation set for the best achieving settings.

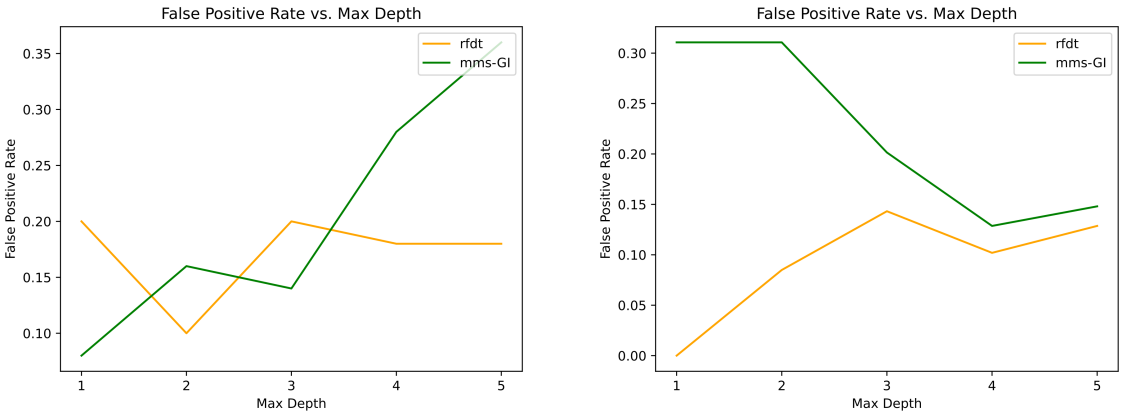


Figure 6.10: False positive rate for diabetes (left) and wine (right) on the validation set for the best achieving settings.

In the verification experiment, the effects of the proposed approaches on the identified factors were discussed. To better understand the effect that the state-of-the-art work has on these factors, its effects and those of mms-GI are plotted for diabetes and wine. The results are shown in figures 6.11, 6.12 and 6.13, where the approaches' best performing settings were used. These results indicate that rfdt does increase the average number of features an adversary has to modify, which should contribute to a higher L1 robustness. It does not lead to a faster decreasing FNR but instead leads to a slower decreasing FNR, which could result in more false negatives during the classification phase. There seems to be no difference from the baseline regarding the average distance between malicious leaves and benign prediction space. This is understandable as the state-of-the-art work still uses the same algorithm as the baseline where a single threshold value could separate a benign and malicious leaf. The state-of-the-art work seems to be able to improve the classifier's robustness without addressing all of the contributing factors, which leaves an opportunity to further improve its robustness by actually addressing the identified factors.

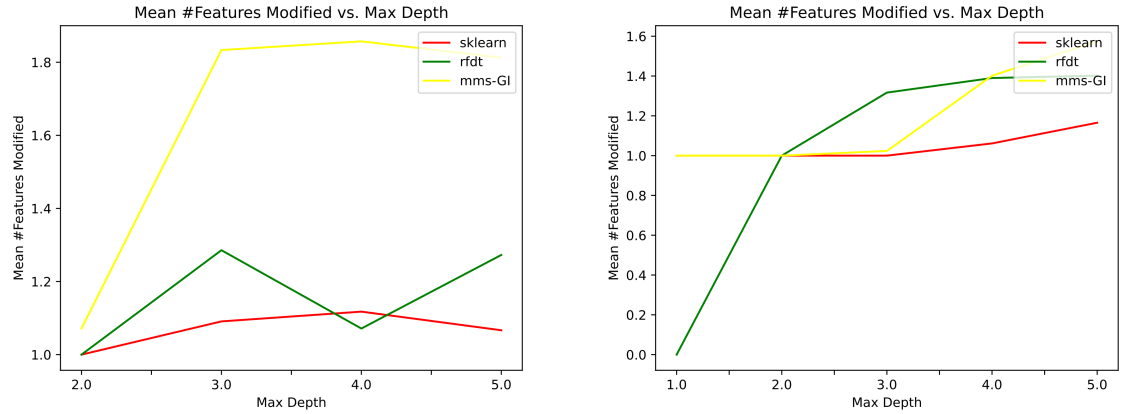


Figure 6.11: Average number of modified features to craft adversarial examples for diabetes (left) and wine (right) on the validation set for the best achieving settings.

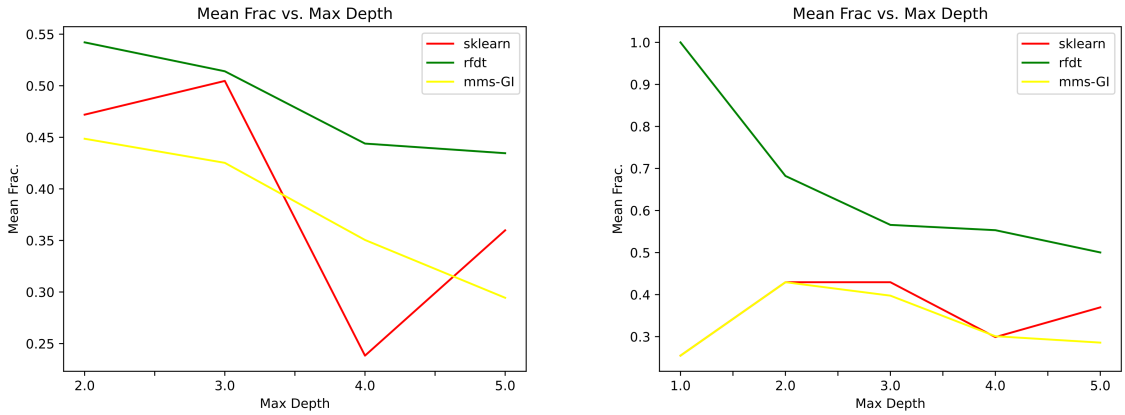


Figure 6.12: Average fraction of malicious training samples within benign prediction space for diabetes (left) and wine (right) on the validation set for the best achieving settings.

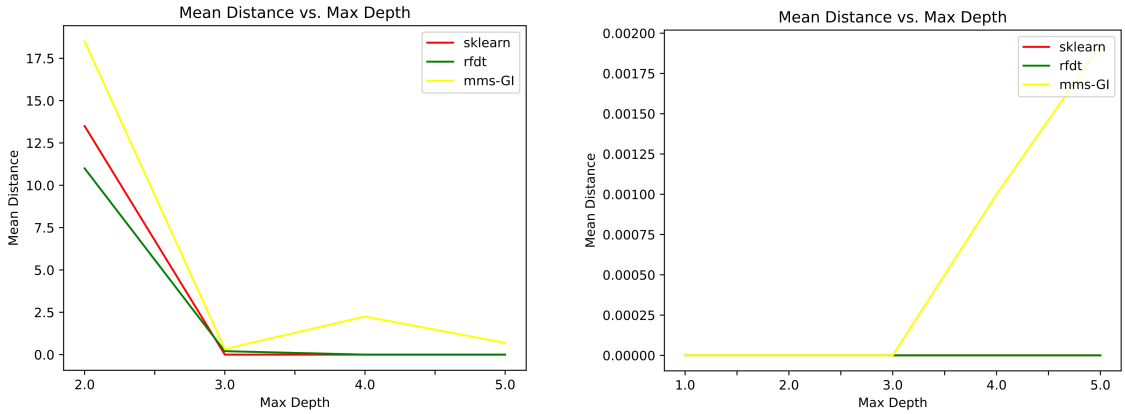


Figure 6.13: Average distance between malicious leaves and benign prediction space for diabetes (left) and wine (right) on the validation set for the best achieving settings.

Random Forest

The state-of-the-art work outperformed the proposed approaches on adversarial accuracy for a single DT. To understand whether this conclusion can also be drawn for ensembles, the same experiment is run for a RF consisting of 5 and 10 trees. The results for the RF consisting of 5 trees are shown in tables 6.6 and 6.7.

The results indicate that for the diabetes dataset, both the proposed approaches and state-of-the-art work perform worse regarding regular accuracy and adversarial accuracy than the baseline. An explanation for this could be that because only a few features are used to construct a DT, the number of trees in the RF is too low to allow the ensemble to perform well. For the other datasets, rfdt nor the proposed approaches outperform the others consistently on either of the metrics. This might indicate that a group of DTs trained with the proposed approaches might compete with state-of-the-art work for adversarial accuracy.

Dataset	sklearn		rfdt		mms-GI		md-GI	
	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc
diabetes	**0.753**	**0.701**	0.675	0.636	0.727	0.662	0.714	0.662
ionosphere	0.889	0.667	**0.944**	0.833	0.889	0.**861**	0.917	**0.861**
wine	0.735	0.658	**0.777**	**0.763**	0.697	0.658	0.669	0.618

Table 6.6: Accuracy (R. Acc) and Adversarial Accuracy (A. Acc) for a Random Forest with 5 estimators. Algorithms were optimised for adversarial accuracy with the best scores for each category made bold. Robust and Fair Decision Forest (rfdt), Greedy Improvement (GI), Minimising Malicious Samples (mms), Maximise Distance (md).

Dataset	sklearn		rfdt		mms-MT		md-MT	
	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc
diabetes	**0.753**	**0.701**	0.675	0.636	0.727	0.689	**0.753**	0.688
ionosphere	0.889	0.667	**0.944**	**0.833**	0.833	0.806	0.917	0.806
wine	0.735	0.658	**0.777**	**0.763**	0.708	0.631	0.669	0.618

Table 6.7: Accuracy (R. Acc) and Adversarial Accuracy (A. Acc) for a Random Forest with 5 estimators. Algorithms were optimised for adversarial accuracy with the best scores for each category made bold. Robust and Fair Decision Forest (rfdt), Minimum Threshold (MT), Minimising Malicious Samples (mms), Maximise Distance (md).

The results for a RF consisting of 10 trees are shown in tables 6.8 and 6.9. The increase in the number of trees seems to have positively affected the adversarial accuracy of the proposed approaches, while negatively affecting it for rfdt. The mms-GI approach has, on average, a 2.8% higher adversarial accuracy than rfdt and mms-MT a 2.87% higher adversarial accuracy. The latter one also has, on average, a 2.37% higher regular accuracy, thus performing better on the datasets than state-of-the-art work. However, the approaches including md still achieve adversarial accuracies lower than that of the baseline for the majority of the datasets. These results, in combination with those for a single DT, indicate that it is better to use mms instead of md, no matter the trade-off approach considered.

Dataset	Sklearn		rfdt		mms-GI		md-GI	
	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc
diabetes	**0.727**	0.662	0.688	0.649	**0.727**	**0.689**	0.662	0.623
ionosphere	**0.944**	0.722	**0.944**	0.778	0.861	0.833	0.917	**0.861**
wine	0.705	0.652	0.706	**0.683**	0.705	0.672	**0.713**	0.634

Table 6.8: Accuracy (R. Acc) and Adversarial Accuracy (A. Acc) for a Random Forest with 10 estimators. Algorithms were optimised for adversarial accuracy with the best scores for each category made bold. Robust and Fair Decision Forest (rfdt), Greedy Improvement (GI), Minimising Malicious Samples (mms), Maximise Distance (md).

Dataset	sklearn		rfdt		mms-MT		md-MT	
	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc	R. Acc	A. Acc
diabetes	0.727	0.662	0.688	0.649	**0.766**	**0.662**	0.675	0.610
ionosphere	**0.944**	0.722	**0.944**	0.778	0.917	**0.889**	0.917	0.861
wine	0.705	0.652	0.706	**0.683**	**0.725**	0.645	0.708	0.623

Table 6.9: Accuracy (R. Acc) and Adversarial Accuracy (A. Acc) for a Random Forest with 10 estimators. Algorithms were optimised for adversarial accuracy with the best scores for each category made bold. Robust and Fair Decision Forest (rfdt), Minimum Threshold (MT), Minimising Malicious Samples (mms), Maximise Distance (md).

The hypotheses for this experiment stated that the proposed approaches including mms achieve a lower FNR than state-of-the-art work, while the latter achieves better or equal adversarial accuracy than all of the proposed approaches for both a DT and RF. The first hypothesis can be confirmed by the results in this section, however the second hypothesis can only be confirmed for a single DT. The approaches including mms achieve a slightly better overall adversarial accuracy in a RF compared to rfdt. The results also indicate that combinations including mms perform better than those including md. The mms-GI approach comes closest to the adversarial accuracy achieved by the state-of-the-art technique in a single DT, a maximum difference less than 7%, while achieving better regular accuracy. The approaches including md performed even worse than the baseline, indicating that they might not lead to the desired model when using low tree depths. Applying the combinations including md in a RF does not lead to better results, as the approaches are mostly still outperformed by the baseline.

7

Discussion

In this chapter, possible shortcomings related to the evaluation process, section 5.4.1, and the results, Chapter 6, are identified and discussed. By analysing the shortcomings, a better idea can be obtained of their impact on the drawn conclusions and their validity.

7.1. Limitations related to identified factors

Three factors were described in section 5.1 that are believed to contribute to the robustness of a decision tree: *feature frequency*, *distance between leaves* and *impurity of benign prediction space*. To support the arguments made there, correlations between the factors and robustness were plotted for two datasets and the Spearman's rank correlation coefficient reported.

The results indicated that a correlation exists for two out of three factors. However, it is possible that the observed correlations are a result of an unknown interfering variable. It is difficult to control the other variables when growing a tree as certain observations can only be made in a varied population of trees. This requires training them using different training data or differing the depth of the tree, both of which also impact other variables in the tree. To account for unknown variables, supportive evidence was computed for two different datasets where a relation should appear in both. The use of more datasets should make it less likely that a correlation is concluded based on results interfered with by an unknown variable. However, there is no guarantee that this is the case and a decisive conclusion cannot be drawn from these results. This should be considered when drawing conclusions from the supportive evidence.

The results also indicated that there does not exist a correlation between the shortest distance between malicious and benign leaves in a tree and the tree's robustness. A brief analysis showed that this result could be attributed to most malicious leaves sharing a parent node with a benign leaf, which leads to the shortest distance of zero. To address this issue, the correlation was analysed for several tree depths. At lower tree depths (smaller tree), the training samples belonging to different classes might still be largely mixed. This could result in child nodes predicting the same class and thus malicious leaves not sharing a parent node with a benign leaf. However, a different conclusion could not be drawn as in most trees the malicious leaves still shared a parent node with a benign leaf. Taking into account the reason behind this observation, a better method to evaluate the correlation between this factor and a tree's robustness should lead to a decisive answer. In light of the situation, drawn conclusions should be adjusted accordingly.

7.2. Limitations related to the datasets

Results of the performed experiments indicate that the proposed approaches improve the robustness of a decision tree and random forest. This further supports the idea that robust tree-based models can be build without considering adversarial attacks during its learning phase.

To evaluate the effect of the approaches, five datasets were considered for a decision tree and three datasets for a random forest. As there does not exist a current benchmark, datasets were chosen based on whether they were used to evaluate state-of-the-art work and their effect on the run-time complexity of the proposed approaches. This led to a group of datasets, most consisting of a small number of samples (< 1000). The consequence of using small datasets is that the observed effects might not be generalisable. The datasets are already a sample of the total population and splitting these datasets in a training, validation and test set reduces the number of considered samples even further. To address this concern, datasets were split using stratified sampling to preserve the balance of classes and conclusions are drawn over all of the datasets. Observing an effect in several different datasets should reduce the uncertainty of it being a result of a poor population sample. However, conclusions drawn from these evaluations should be considered in the light of the sizes of the datasets used.

7.3. Limitations related to the threat model

In this work, a white-box scenario is considered with an unbounded adversary to assess the worst-case performance of a classifier in terms of the perturbation distance. Techniques exist to construct a (near) perfect surrogate model with only access to querying the original model, supporting the choice for a white-box scenario. However, the adversary's capabilities might be considered unrealistic.

An alternative to using an unbounded adversary is to constrains its capabilities by specifying which feature is modifiable and to what extent. This should lead to a more realistic scenario and thus a more realistic assessment of the model's robustness. Taking this into account, the comparison with state-of-the-art research was performed considering the threat models as used in the work by Vos [57]. However, these threat models are arbitrarily constructed and might not resemble an adversary's real capabilities. The specified capabilities could even lead to a better robustness score than might be achieved in real. This should be considered when drawing conclusions from these evaluations. A possible improvement for this subject could be to include a domain expert when specifying a threat model.

7.4. Limitations related to the evaluation

The evaluation of the proposed approaches indicated improvement in robustness of the tree-based models but not in all cases as good as the state-of-the-art technique. The procedure must be analysed to better understand the validity of these conclusions.

The results of the evaluation with a random forest indicated a robuster tree ensemble. The size of the random forest was limited to five trees, because of constraints introduced by the attack algorithm. To assess the model's worst-case performance, minimum adversarial examples were computed. This requires considering all distinct benign prediction spaces, which leads to an increase in running time and/or space usage. The evaluation of a random forest leaves uncertainty regarding the generalisability of the results, because of the consideration of a single setting. This issue was addressed in the comparison with state-of-the-art work, where a random forest of five trees and 10 trees was considered. This was possible as a different evaluation metric was used, which does not require the computation of minimum adversarial examples. However, tree ensembles could consists of hundreds of trees, still leaving uncertainty for the generalisability of the results. This should be considered when drawing conclusions regarding a random forest.

The comparison with state-of-the-art research was performed in a similar setting as used in their evaluation with other works, both in terms of threat models used and classifier settings considered. This means that the maximum tree depth was limited to a depth of five, a choice they based on the inter-

pretability of a tree model. However, earlier experiments showed that the proposed approaches might require larger trees to perform well. A consequence of using multiple criteria to learn the model. Limiting the tree depth might thus have lead to results not representative for the abilities of the approaches. The other experiments partly address this issue by considering larger trees but the difference in evaluation metric and threat model prevents a direct comparison. The possible effect on the results should thus be considered when drawing conclusions from this experiment. A future improvement could be to run the evaluation for larger trees too.

8

Conclusions and Future Work

This chapter concludes the work within this thesis by answering the research questions as stated in Chapter 1 using the results from Chapter 5 and 6. This research aimed at finding an alternative way to improve the robustness of decision tree classifiers where unbounded adversaries aim at causing misclassifications of malicious samples. The state-of-the-art techniques attempt to optimise the splitting process' worst-case performance under adversarial attacks but ignore other relevant aspects that could lead to new approaches improving the classifier's robustness. In this work, several of these approaches are introduced, which improve a decision tree's robustness through identified contributing factors.

RQ1: Which factors of a tree-based model contribute to its robustness?
Three factors contributing to a decision tree's robustness were identified and described in section 5.1. These are: *feature frequency*, the presence or absence of features in a benign decision path, *shortest distance between malicious leaves and benign prediction space* and the *impurity of benign prediction space*. The correlation of each of these factors with the decision tree's robustness was analysed for two datasets through analysis of plots and using Spearman's rank correlation coefficient. This resulted in supportive evidence that there exists a correlation for two out of the three factors. The third factor could not be shown to have a correlation with the decision tree's robustness; however, the effects of the approaches proposed in this work do hint at the existence of such correlation. These three factors do not make up all factors contributing to the robustness of a decision tree classifier but are the first to be identified as such.

RQ2: How can robustness of a decision tree classifier be optimised during learning?
Two splitting criteria are introduced in this work to improve a decision tree's robustness: *Minimising Malicious Samples* and *Maximising Distance*. Both criteria require multidimensional boundaries to be placed around the benign samples, limiting the benign prediction space in a decision tree and improving the feature frequency. The first criterion then improves robustness further by reducing the impurity of the benign prediction space. The latter criterion increases the distance between leaves by maximising the distance between boundaries on both sides of a split threshold to increase the shortest distance from a malicious leaf to benign prediction space. Results show that each of these criteria leads to a higher L1 robustness for the decision tree while keeping the drop in accuracy less than 10%. Results also indicate that these criteria could require deeper trees in order to achieve good performance as the criteria do not attempt to separate samples from different classes. This could lead to impure leaves, containing large amounts of training samples from both classes.

To address this issue, two trade-off approaches are proposed: *Greedy Improvement* and *Minimum Threshold*. Both methods attempt to make a decision per node split between using the criterion aforementioned or the original splitting criterion. The former greedily chooses the criterion leading to the largest gain between gini impurity improvement and robustness improvement, where the latter defines a threshold value for the gini impurity improvement below which robust splits are made. Results show

that a trade-off approach does not necessarily lead to a better performing model. When robust splits are only made closer to the leaves of the final tree, the original splitting criterion limits the effectiveness of the introduced criteria. The purer a benign leaf is by itself, the more constraining the limiting of benign prediction space becomes, leading to a larger increase in false positives.

The robustness of a classifier can be improved by both introduced criteria in combination with either of the suggested trade-off approaches. The difference depends on their impact on the classifier's classification performance. A combination of *minimising malicious samples* with any of the trade-off approaches would lead to a drop in accuracy between 2.5% and 4%.

RQ3: What is the effect on a random forest when built using robust decision tree classifiers?
A random forest consisting of decision trees trained with one of the aforementioned criteria in combination with a trade-off approach is more robust than when gini impurity is used. Combinations including *Greedy Improvement* showed a robustness score 1.5 times that of the baseline with accuracy dropping only slightly (less than 3%). Overall, criteria combined with *Greedy Improvement* achieved better performance for both accuracy and robustness than criteria combined with *Minimum Threshold*. Results also indicate that adding more trees to a random forest could lead to better performance for the *minimising malicious samples* criterion, independent of which trade-off approach is used.

RQ4: How effective are the approaches in this work compared with state-of-the-art research?
Based on the robustness criterion used, the accuracy on a maximally perturbed dataset, applying the *maximising distance* criterion in a decision tree leads to a decrease of the tree's robustness. This effect could be attributed to the tree depth, where the proposed approaches perform better in larger trees (higher tree depth). The *minimising malicious samples* criterion in combination with *greedy improvement* achieved on average a 3.17% higher accuracy but a 5.5% lower robustness compared to the state-of-the-art technique. This shows that the combination of this criterion and trade-off approach can compete with state-of-the-art work in small trees. Results also indicate that combining a criterion with *greedy improvement* leads to better performance in small trees than *minimum threshold*.

Applied in a random forest of 10 trees, combinations including the *minimising malicious samples* criterion achieve better adversarial accuracy than state-of-the-art work. In combination with *greedy improvement* this is, on average, a difference in 2.8% adversarial accuracy and for *minimum threshold* a difference of 2.87%. The latter combination also achieves a 2.37% higher accuracy, making it outperform the state-of-the-art technique.

The results show that of the proposed approaches, the combinations including *minimising malicious samples* can compete with state-of-the-art work. They do not perform as well in a decision tree but combining several of these robust trees in a random forest leads to a better performance than achieved by state-of-the-art work.

8.1. Future Work

Robustness contributing factors

In this work, several factors were identified that are believed to contribute to the robustness of a decision tree classifier. Although there are arguments for their relevance, not all factors could statistically be proven to be of significant influence for the robustness of the classifier. Further research might identify more related factors and/or be able to prove the significance of the already identified factors even further.

Run-time complexity improvements

The approaches proposed within this work made use of multidimensional boundaries, which introduce a run-time penalty during training. The individual approaches could also add their own penalty on top of this, which would make them less attractive outside academia. Further research might better understand the problem and avoid these heavy-cost constructions, resulting in a faster algorithm to train robust DTs.

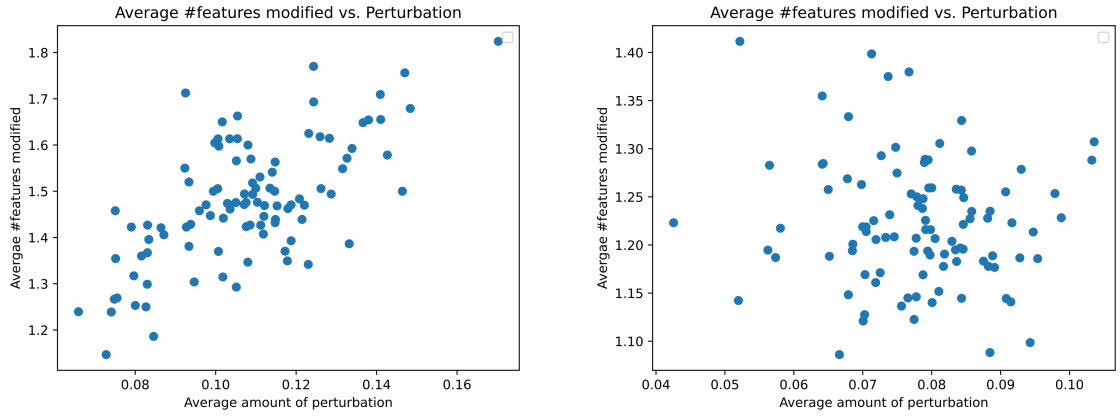
Benchmark
A proper benchmark is missing in the field, which leads to the situation that different proposals are evaluated on different datasets. This makes it harder to compare the techniques with one another. Another research option might be to design/develop a benchmark that can be used within this field to compare the different proposed techniques with each other.

Attacking ensembles
Currently, only two algorithms exist to attack ensembles of decision trees. Both of them have their flaws, either not being scalable in running time or requiring too much memory for exact computations. It would be interesting to have an attack that is both scalable and does not require too much memory in order to attack the ensembles of trees.

Attacking both classes
The approaches within this work are designed to protect against an adversary perturbing malicious samples to cause misclassifications. However, there are scenarios in which an adversary might attempt to cause misclassifications of benign samples, e.g. to reduce the usefulness of the classifier in practice. It would be interesting to see if the techniques can be utilised in a manner that protects both classes without losing their individual advantages.

A

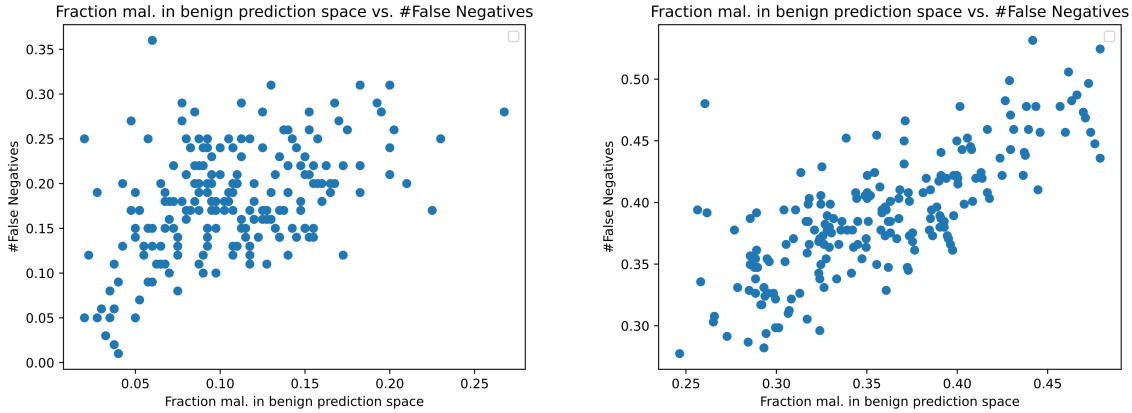
Correlation plots for contributing factors



(a) Dataset: diabetes

(b) Dataset: wine

Figure A.1: Average modified features and average amount of perturbation (L_1 distance) of 100 trees at tree depth 5



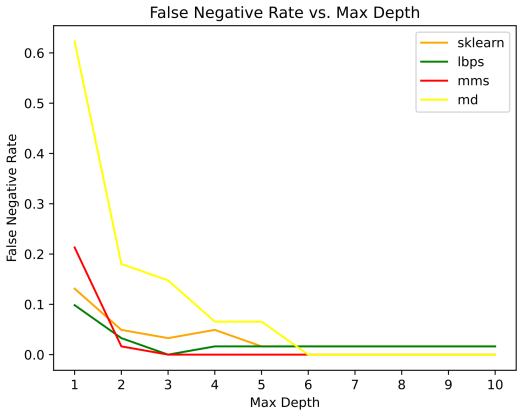
(a) Dataset: diabetes

(b) Dataset: wine

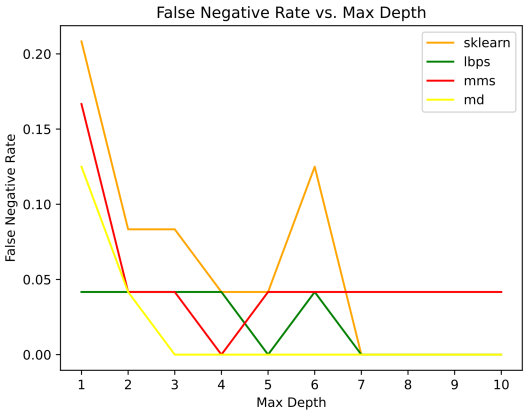
Figure A.2: Average fraction malicious training samples within benign prediction space and number of false negatives of 100 trees per depth for depths 3 to 7.

B

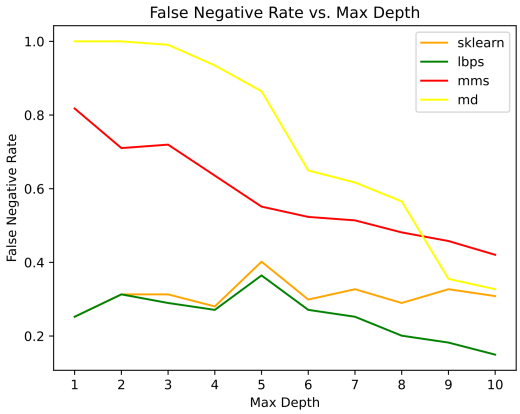
Additional results integration experiment



(a) Dataset: banknote

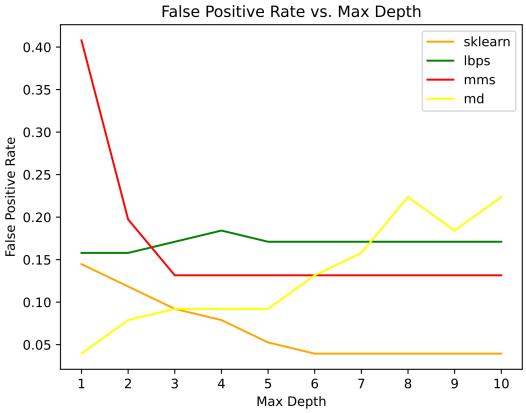


(b) Dataset: breast-cancer

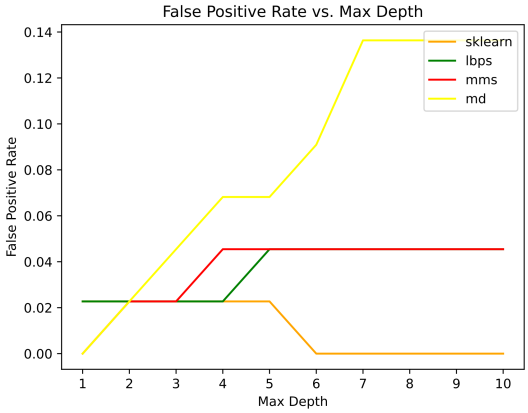


(c) Dataset: wine

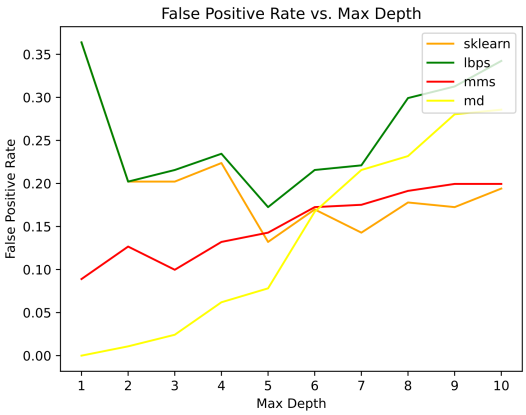
Figure B.1: False negative rates on the validation set.



(a) Dataset: banknote

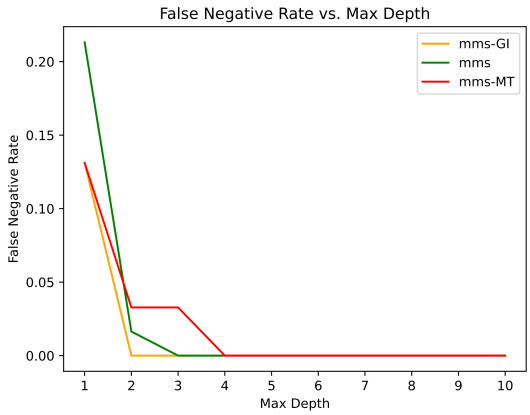


(b) Dataset: breast-cancer

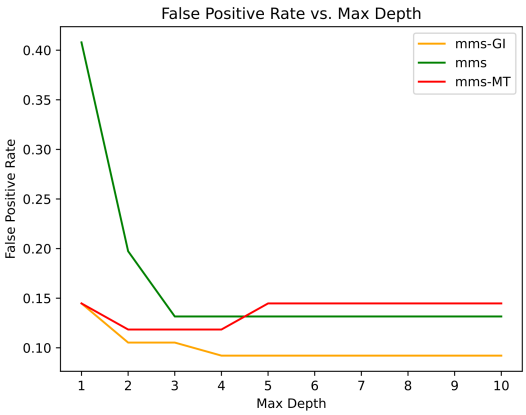


(c) Dataset: wine

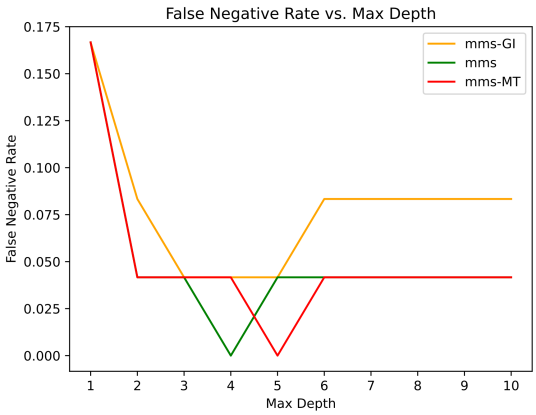
Figure B.2: False positive rates on the validation set.



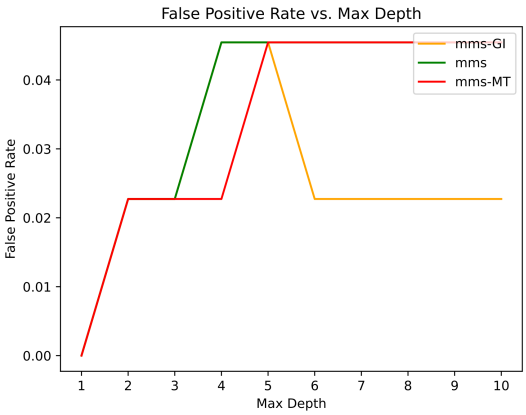
(a) Dataset: banknote



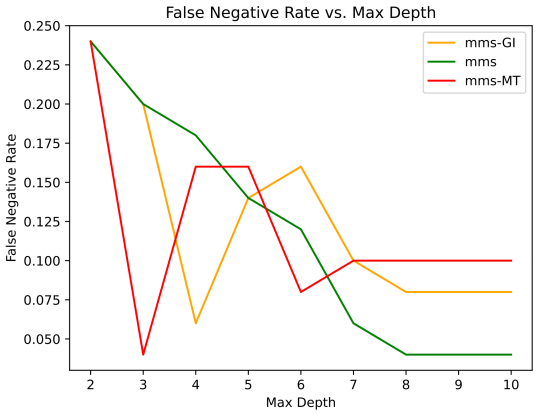
(b) Dataset: banknote



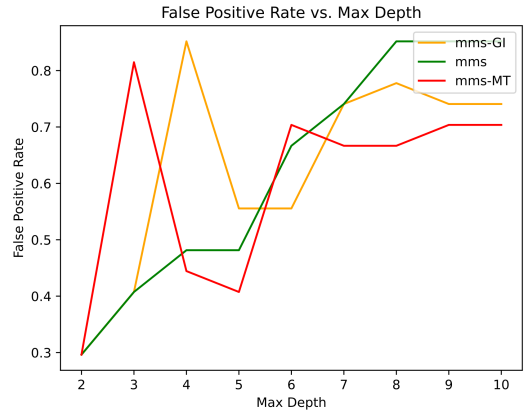
(c) Dataset: breast-cancer



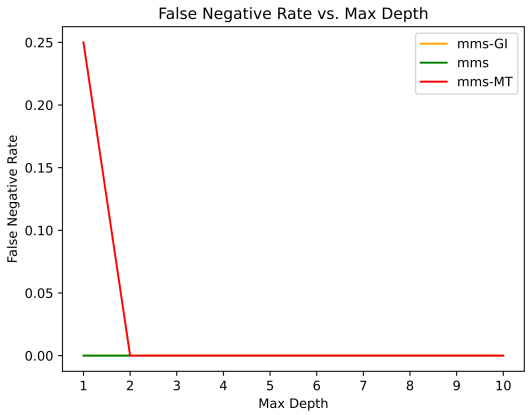
(d) Dataset: breast-cancer



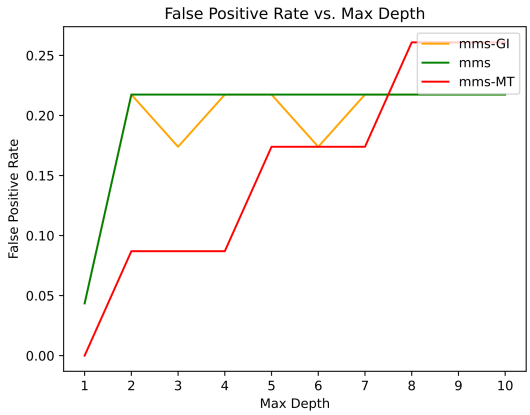
(e) Dataset: diabetes



(f) Dataset: diabetes



(g) Dataset: ionosphere

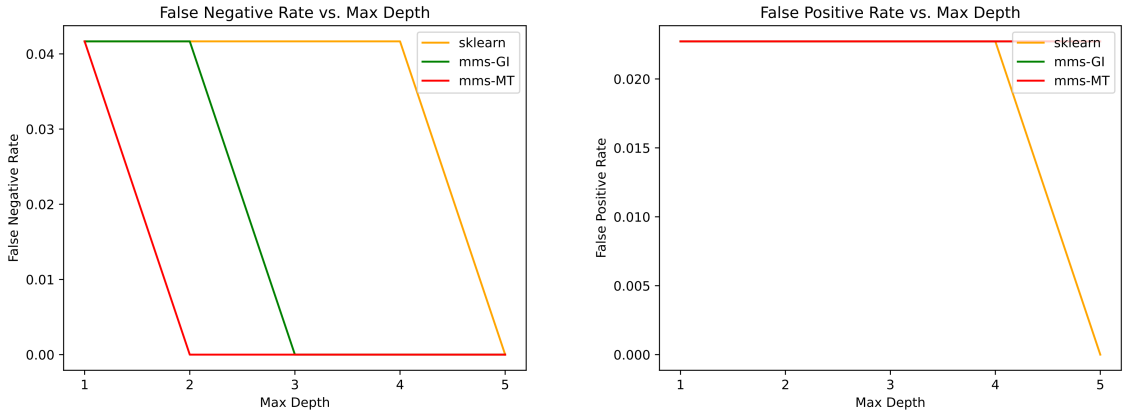


(h) Dataset: ionosphere

Figure B.3: False negative rates and false positive rates on the validation set for mms, mms-GI and mms-MT.

C

Additional results random forest experiment



(a) Dataset: breast-cancer

(b) Dataset: breast-cancer

Figure C.1: False negative rates and false positive rates on the validation set.

Bibliography

[1] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *OJ*, L 119:1–88, 05 2016.

[2] *Kaspersky Security Bulletin 2018. Statistics*. Kaspersky, Dec 2018. URL https://securelist.com/kaspersky-security-bulletin-2018-statistics/89145/.

[3] Cyber kill chain applied to ics, Jun 2019. URL https://www.incibe-cert.es/en/blog/cyber-kill-chain-applied-ics.

[4] *Kaspersky Security Bulletin 2019. Statistics*. Kaspersky, Dec 2019. URL https://securelist.com/kaspersky-security-bulletin-2019-statistics/95475/.

[5] Supermarkt wordt met hulp van panasonic de veiligste winkel van nederland, Feb 2019. URL https://nl.business.panasonic.be/security-oplossingen/panasonic-gezichtsherkenning-helpt-supermarkt-bij-het-winnen-van-retailaward.

[6] *State of Cybersecurity 2020: Global Update on Workforce Efforts and Resources*. ISACA, 2020. URL https://www.isaca.org/go/state-of-cybersecurity-2020.

[7] *2020 State of Malware Report*. Malwarebytes Labs, Feb 2020. URL https://blog.malwarebytes.com/reports/2020/02/malwarebytes-labs-releases-2020-state-of-malware-report/.

[8] Accenture. 2019 cost of cybercrime study: 9th annual, Mar 2019. URL https://www.accenture.com/us-en/insights/security/cost-cybercrime-study.

[9] Mamoun Alazab and MingJian Tang. *Deep learning applications for cyber security*. Springer, 2019. ISBN 978-3-030-13057-2. URL https://doi.org/10.1007/978-3-030-13057-2.

[10] Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. In *Advances in Neural Information Processing Systems 32*, pages 13017–13028. Curran Associates, Inc., 2019.

[11] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, page 16–25, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595932720. URL https://doi.org/10.1145/1128817.1128824.

[12] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, Nov 2010. ISSN 1573-0565. URL https://doi.org/10.1007/s10994-010-5188-5.

[13] Mridula Batra and Rashmi Agrawal. Comparative analysis of decision tree algorithms. In Bijaya Ketan Panigrahi, M. N. Hoda, Vinod Sharma, and Shivendra Goel, editors, *Nature Inspired Computing*, pages 31–36, Singapore, 2018. Springer Singapore.

[14] B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, April 2014. ISSN 2326-3865.

[15] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317 – 331, 2018. ISSN 0031-3203. URL https://doi.org/10.1016/j.patcog.2018.07.023.

[16] Battista Biggio, Giorgio Fumera, and Fabio Roli. Pattern recognition systems under attack: Design issues and research challenges. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(07):1460002, 2014. URL https://doi.org/10.1142/S0218001414600027.

[17] Geoff Blaine. Encrypted attacks, iot malware surge as global malware volume dips, Oct 2019. URL https://blog.sonicwall.com/en-us/2019/10/sonicwall-encrypted-attacks-iot-malware-surge-as-global-malware-volume-dips/.

[18] Leo Breiman. Technical note: Some properties of splitting criteria. *Machine Learning*, 24(1): 41–47, Jul 1996. ISSN 1573-0565. URL https://doi.org/10.1023/A:1018094028462.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. URL https://doi.org/10.1023/A:1010933404324.

[20] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2017.

[21] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, Jun 2020. URL https://doi.org/10.1007/s10618-020-00694-9.

[22] Hongge Chen, Huan Zhang, Duane S. Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1122–1131. PMLR, 2019. URL http://proceedings.mlr.press/v97/chen19m.html.

[23] Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane Boning, and Cho-Jui Hsieh. Robustness verification of tree-based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12317–12328. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9399-robustness-verification-of-tree-based-models.pdf.

[24] 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*, AISec '17, page 15–26, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352024. URL https://doi.org/10.1145/3128572.3140448.

[25] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. Paper presented at the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA., 2019.

[26] Tom Chivers. Facial recognition... coming to a supermarket near you, Aug 2019. URL https://www.theguardian.com/technology/2019/aug/04/facial-recognition-supermarket-facewatch-ai-artificial-intelligence-civil-liberties.

[27] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models, 2017.

[28] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable and Secure Computing*, 16(4):711–724, July 2019. ISSN 2160-9209. URL https://doi.org/10.1109/TDSC.2017.2700270.

[29] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6.

[30] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[31] Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 353–360, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. URL https://doi.org/10.1145/1143844.1143889.

[32] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Rocky Duan, Pieter Abbeel, and Jack Clark. Attacking machine learning with adversarial examples, Mar 2019. URL https://openai.com/blog/adversarial-example-research/.

[33] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[34] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification, 2016.

[35] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples, 2017.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[37] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan, 2017.

[38] Chi-Hsuan Huang, Tsung-Han Lee, Lin-huang Chang, Jhih-Ren Lin, and Gwoboa Horng. Adversarial attacks on sdn-based deep learning ids system. In Kuinam J. Kim and Hyuncheol Kim, editors, *Mobile and Wireless Technology 2018*, pages 181–191, Singapore, 2019. Springer Singapore. ISBN 978-981-13-1059-1.

[39] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, 2011. URL https://lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf.

[40] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. Evasion and hardening of tree ensemble classifiers. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2387–2396. JMLR.org, 2016.

[41] Guofu Li, Pengjia Zhu, Jin Li, Zhemin Yang, Ning Cao, and Zhiyi Chen. Security matters: A survey on adversarial machine learning, 2018.

[42] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE Access*, 6:12103–12117, 2018. ISSN 2169-3536. URL https://doi.org/10.1109/ACCESS.2018.2805680.

[43] Jeff Loucks, Tom Davenport, and David Schatsky. State of ai in the enterprise, 2nd edition, Oct 2018. URL https://www2.deloitte.com/content/dam/insights/us/articles/4780_State-of-AI-in-the-enterprise/DI_State-of-AI-in-the-enterprise-2nd-ed.pdf.

[44] Jeff Markey. Using decision tree analysis for intrusion detection: A how-to guide, Jun 2011. URL https://www.sans.org/reading-room/whitepapers/detection/paper/33678.

[45] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[46] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, March 2016. URL https://doi.org/10.1109/EuroSP.2016.36.

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

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

[49] Ross J Quinian. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[50] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986. ISSN 1573-0565. URL https://doi.org/10.1007/BF00116251.

[51] N. rndic and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE Symposium on Security and Privacy*, pages 197–211, May 2014. URL https://doi.org/10.1109/SP.2014.20.

[52] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Generic black-box end-to-end attack against state of the art api call based malware classifiers. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 490–510, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00470-5.

[53] Sonia Singh and Priyanka Gupta. Comparative study id3, cart and c4.5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, 27(27), Jul 2014.

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

[55] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 601–618, USA, 2016. USENIX Association. ISBN 9781931971324.

[56] Raphael Vallat. Pingouin: statistics in python. *Journal of Open Source Software*, 3(31):1026, 2018. URL https://doi.org/10.21105/joss.01026.

[57] Daniël Vos. Adversarially robust decision trees against user-specified threat models, 2020. URL http://resolver.tudelft.nl/uuid:c9d9cdc6-4f98-4730-8fb6-43e6e3444002.

[58] Xindong Wu. *The top ten algorithms in data mining*. CRC Press, 2009.

[59] Fan Yang, Zhiyuan Chen, and Aryya Gangopadhyay. Using randomness to improve robustness of tree-based models against evasion attacks. In *Proceedings of the ACM International Workshop on Security and Privacy Analytics*, IWSPA '19, page 25–35, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361781. URL https://doi.org/10.1145/3309182.3309186.