Università Ca' Foscari Venezia

Department of Environmental Sciences, Informatics and Statistics

Doctoral Degree
in *Computer Science*

36th cycle

Effective, Efficient, and Robust Learning Algorithms for Ranking and Classification

---

A thesis submitted in fulfilment of the requirements for
the degree of Doctor of Philosophy

Supervisor
Prof. Claudio Lucchese

Graduand
Federico Marcuzzi
Matriculation number: 853770

Academic Year
2022/2023

Contents

Abstract

Over the past decade, machine learning has gained significant traction and is now deployed across diverse domains, including information systems, finance, healthcare, cybersecurity, autonomous driving, and more. As machine learning finds applications in various sensitive scenarios, the demand for models that exhibit accuracy and robustness during the operational phase has grown exponentially.

One crucial factor that profoundly shapes the quality of machine learning models revolves around the training data they rely upon and the input data encountered at the operational phase. Therefore, the development of data-aware algorithms is of paramount importance in achieving high-quality machine-learning models.

This thesis contributes to this overarching objective by delving into the development of data-aware algorithms, emphasising the importance of this awareness during both the training and operational phases of machine learning models. The research presented in this thesis focuses on two primary domains. Firstly, it centres on information retrieval, with a particular emphasis on enhancing both the efficiency of learning-to-rank learning algorithms and the effectiveness of the learned models in solving ranking tasks. Secondly, it focused on adversarial machine learning scenarios, striving to enhance the robustness of binary classifiers against adversarial inputs at the operational phase while providing certifiable models to efficiently assess robustness against adversarial machine learning attacks.

Chapter 1

Introduction

Over the past decade, *Artificial Intelligence* (AI) has assumed an increasingly prominent role in various aspects of our society. Coined in 1956, the term "Artificial Intelligence" refers to the computer science discipline that focuses on developing systems and computers that perform tasks typically requiring human intelligence, such as learning, reasoning, decision-making, and problem-solving. This discipline has gained significant relevance in recent years, increasing across various domains due to its ability to automate complex processes and analyse vast amounts of data.

The methodology that has most significantly contributed to the rapid advancement of AI is *Machine Learning* (ML). Machine learning represents an innovative and dynamic approach to data analysis and problem-solving. It is centred on the automatic acquisition of knowledge from data to enhance computer performance in specific tasks. The adoption of ML has played a significant role in the rapid rise of AI in diverse scenarios. This remarkable progress has been driven by two fundamental traits: ease of use and the ability to extract valuable insights from data with little or no direct human intervention in the learning process. Research in ML aims to create algorithms that automatically enhance their performance based on data. In short, machine learning algorithms take data as input and produce a model that performs the desired task. The tasks performed by ML models primarily include classification, regression, clustering, and ranking. This thesis mainly focuses on the first and the last.

Nowadays, it is hard to find applications that do not apply ML models, especially in technological, financial, social, and medical scenarios. For example, ML is widely used in search engines, authentication systems, decision-making tools, and risk assessment systems. Given the extensive application of ML, especially in sensitive scenarios such as medicine and cybersecurity, developing reliable, robust, and secure systems is crucial.

The aspect that most likely compromises the achievement of these desirable

properties is the quality of the data involved in various stages of the model life cycle. The ability of an ML model to successfully execute its assigned task largely depends on the quality of the data used in the learning process, i.e., the training data. Consequently, the quality of the data is of fundamental importance. Unfortunately, the nature of the data and how it is collected often undermines its quality. Training data may contain inconsistencies, biases, or errors that can negatively impact the model's learning process and lead to expected outcomes.

Furthermore, due to various circumstances, an ML model can fail its task during the operational phase, i.e., when asked to perform the task it was trained for. For example, the model may fail to predict instances far from the distribution it encountered during the training phase, that is, instances that are very different from what it saw during the training phase. Additionally, the model may fail to correctly classify instances manipulated or artificially generated by a malicious entity, forcing it to take an unexpected behaviour, e.g., unauthorised system access or exposure of confidential information. Manipulating instances to compromise either the training or operational phase is called *Adversarial Machine Learning* (or Adversarial Learning). The vulnerability of machine learning models to these adversarial machine learning manipulations has given rise to a further branch of research in the field of ML, which focuses on creating models that are robust to such attacks.

This undesired characteristic of ML contradicts the literal definition of "Artificial Intelligence"; AI is not as intelligent as it might seem. As asserted by Goodfellow *et al.* in [75], an AI model, during its learning process, creates its own "*Potemkin village*"; ML models are not truly learning the semantics of data. This analogy suggests that the model performs well on instances that occur naturally but performs poorly when encountering instances with a low probability in the distribution. The inability of ML models to learn the semantics of data opens up possibilities for adversarial learning attacks, where malicious entities subtly modify input instances to cause unexpected behaviour in the model.

The discovery of these vulnerabilities has highlighted the need to develop rigorous data collection and pre-processing protocols, as well as advanced quality control methods to protect against adversarial machine learning attacks. In addition, due to the enormous volume of data generated daily and the massive amounts of data required to train complex systems such as ChatGPT, BERT [58], etc., it is often impossible to exercise complete control over the data, including all possible inputs that the model may encounter.

Therefore, this thesis addresses these issues by designing data-aware machine-learning algorithms that can autonomously intervene on the input to identify valuable and harmful data. Additionally, our focus extends to the development of tools that assess and guarantee the quality and security of learned models. We believe

these research directions are necessary to fully leverage the transformative potential of artificial intelligence across a wide range of applications. By prioritising these aspects, we can ensure that AI technology is used effectively and responsibly while minimising potential harm.

1.1 Contributions of This Thesis

This thesis encompasses the research effort I dedicated during my doctoral studies in Computer Science. The scientific contributions revolved around improving the data awareness of machine learning algorithms during both the training and operational phases. In particular, it focused on two specific domains: *Learning to Rank* (LtR) and *Adversarial Machine Learning* (AML) in binary classification.

In the Learning to Rank domain, the research focused on designing data-aware LtR learning algorithms to obtain more efficient learning algorithms and more effective and efficient ranking models. For what concerns adversarial machine learning in binary classification, the research concentrated on adversarial attacks at the operation phase. To counteract this phenomenon, data awareness was included in the learning process to obtain more robust learning algorithms.

Below, the main contribution and results for each domain area are provided.

Learning to Rank Learning to Rank represents a class of machine learning algorithms applied to solve the task of generating a ranking for a set of documents or items based on their relevance to a given query or request.

Our research in this domain is divided into two distinct directions. The first direction focuses on identifying the quality of documents in the training set to improve the effectiveness of the model's learning process. By distinguishing between "*good*" and "*bad*" documents, the model can avoid learning from non-relevant or misleading information. The second direction is to enhance the learning process using advanced document comparison strategies. By doing so, the model can better differentiate between relevant and non-relevant documents, thereby producing more accurate and effective rankings.

The research in this field is enclosed in three articles, each of which made significant strides in advancing our understanding and capabilities in this domain.

Filtering out Outliers in Learning to Rank [121] This work, in proceedings as a full paper at the *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval*, presents a novel concept in the field of Learning to Rank: the notion of *consistent outliers*. These consistent outliers refer to documents that are consistently mis-ranked during the training phase.

Our research unearthed a significant issue: the continuous presence of consistent outliers during the learning phase taints the final model, ultimately compromising its overall effectiveness. Building upon this outlier definition, we designed a new learning-to-rank learning algorithm called *Surrender on Outliers and Rank* (SOUR). The SOUR algorithm first identifies and isolates these persistent outlier documents, and then, it trains a model using a training dataset that has been cleansed of these noxious documents. This approach aims to enhance the model's effectiveness and robustness by mitigating the influence of these consistently mis-ranked documents.

On the Effect of Low-Ranked Documents: A New Sampling Function for Selective Gradient Boosting [110] In this work, we developed a new document selection function for the *Selective Gradient Boosting* (SelGB) framework proposed by Lucchese *et al.* in their work [115]. This selection function has been designed to make the most of the informativeness of the lowest-ranked non-relevant documents. This work is in proceeding as a full paper at the *SAC '23: The 2023 ACM SIGAPP Symposium on Applied Computing.*

SelGB is an ML framework that uses Gradient Boosting Decision Trees (GBDTs) for ranking purposes. As part of its training process, SelGB performs per-query document selection at regular intervals after a fixed number of training iterations.

The original selection function provided with SelGB completely discards the lowest-ranked non-relevant documents in favour of the highest-ranked non-relevant counterparts. We investigated the potential value of the lowest-ranked non-relevant documents in enriching the learning process. Our findings revealed that a blended approach, combining the highest-ranked and lowest-ranked non-relevant documents, effectively overcomes the limitations of the original algorithm, resulting in a more robust and stable learning process. Furthermore, removing a significant portion of the documents in the training set provides a significant speed-up in the training process, improving the efficiency of the learning algorithm.

LambdaRank Gradients are Incoherent [122] In our study, we discovered a notable issue with LambdaRank and its derivatives regarding gradient coherence to document relevances. Specifically, we observed that, during the learning phase, a mis-ranked document with high relevance could be pushed down in rankings more significantly than a document with lower relevance. This suggests that the learning algorithm failed to learn how to prioritise the most relevant documents during the learning phase.

We conducted an in-depth analysis of the occurrence of this phenomenon during truncated and un-truncated metric optimisation and discovered that these gradi-

ent incoherencies have a detrimental impact on the overall effectiveness of the final models. Furthermore, we discovered that this phenomenon is exacerbated by truncated metric optimisation, which provides a significant drop in performance compared to un-truncated optimisation. As a consequence, this discovery forces a choice between an efficient truncated training process and effective learned models provided by un-truncated optimisation.

To solve this exacerbation problem and avoid a trade-off between training efficiency and models' effectiveness, we devised three strategies for performing pairwise document comparisons. The aim is to improve the contribution of highly relevant documents that may be overlooked during the training phase. Through empirical testing, we proved that our proposed solutions successfully addressed gradient incoherencies, resulting in enhanced model effectiveness comparable to un-truncated metric optimisation algorithms while maintaining the training efficiency as truncated optimisation.

The work is in proceedings as a full paper at the *CIKM '23: The 2023 ACM International Conference on Information and Knowledge Management.*

Adversarial Machine Learning In the area of Adversarial Machine Learning, we placed a significant emphasis on classification models based on *Decision Trees* under *Evasion Attacks* (attacks at the operational phase).

Specifically, the contribution revolved around developing learning algorithms tailored to training robust models capable of withstanding evasion attacks. Additionally, we delved into the critical aspect of verifying and certifying the robustness of machine learning models to estimate the level of performance even when subjected to adversarial attacks. Moreover, we introduced a novel metric devised to assess the robustness of ML models, and we also designed an algorithm to efficiently verify this metric for tree-based models.

Also, in this domain, the contribution is enclosed in three articles aimed at improving and evaluating the robustness of machine learning models.

Feature Partitioning for Robust Tree Ensembles and Their Certification in Adversarial Scenarios [34] This article, published as a full paper at the *EURASIP Journal on Information Security, 2021*, introduces an innovative approach to training robust decision tree forests, particularly in the context of evasion attacks. The key concept involves training each tree within the forest on a distinct partition of the feature space. This strategic partitioning is meticulously designed to limit the impact of an evasion attack to less than half of the forest.

The ensemble structure resulting from this feature space partitioning lays the groundwork for developing two robustness certification algorithms. These certification algorithms efficiently and accurately compute the lower bound of the

model's robustness. Efficiency and accuracy are fundamental qualities for obtaining valuable information on the model's ability to resist adversary manipulation in a feasible time frame.

Beyond Robustness: Resilience Verification of Tree-Based Classifiers [31] This article, published as a full paper at the *Computers & Security, 2022*, introduces a new evaluation metric called *Resilience* to assess the model's security to AML attacks and compensate for the deficiencies in the well-known and widely used *Robustness* metric. The Resilience metric verifies the security (stability) of the model not only for instances in the evaluation set but also for every possible instance in a neighbourhood close to the original evaluation instance. A model is considered stable when it produces an equal prediction under attack for all possible instances sampled from this neighbourhood. Consequently, Resilience checks for any effective evasion attacks that can be generated from instances close to the original evaluation instance; , information not covered by the Robustness metric. This allows for better quantification of the model's ability to resist evasive attacks.

However, Resilience has some drawbacks: verifying model stability for all possible instances in a neighbourhood is not feasible since it is a continuous space. This makes calculating Resilience impossible. To overcome this problem, we provided a data-independent stability analyser for tree-based models that produces hyper-rectangles representing portions of the feature space where the model is stable. Through hyper-rectangles, it is possible to compute where the model is resilient.

Explainable Global Fairness Verification of Tree-Based Classifiers [30] This article differs from the previous two as it focuses on fairness in machine learning. However, we approached this issue from the perspective of an evasion attack scenario.

We expanded upon the model-stability analyser introduced in our previous work (i.e., article [31]) to create a Synthesis Algorithm that characterises model fairness; specifically, it looks for portions of the feature space where the model is guaranteed to exhibit a lack of causal discrimination. Causal discrimination occurs when individuals with comparable characteristics within the system but belonging to distinct protected groups, such as gender or race, receive disparate responses from the model: in such settings, the model is unfair.

We used the stability analyser to find portions of the feature space where the model may provide causal discrimination and the Synthesis Algorithm to characterise a portion of the feature space where the model is fair. The formulas generated by the Synthesis Algorithm are expressed as a set of traditional propositional logic formulas that pertain to the entire feature space rather than being confined to a specific evaluation set. This approach ensures global fairness guarantees.

Furthermore, our methodology is inherently explainable, meaning that human experts can easily understand it because it is based on conventional logical formulas. In particular, our empirical findings demonstrate that a concise set of simple logic formulas effectively encapsulates the classifier's fairness guarantees in practical scenarios.

The contribution of this research is published as a full paper at the *SaTML '23: The 2023 IEEE Conference on Secure and Trustworthy Machine Learning*.

The main contribution of this thesis is to make machine learning models more effective, efficient and robust. Specifically, for learning to rank algorithms, we aim to enhance the effectiveness of the ranking produced at the operational phase and efficiency in terms of training time. For binary classification algorithms, we focus on improving the security in terms of robustness to evasion attacks in adversarial scenarios and provide efficient and accurate robustness certification algorithms.

Notwithstanding the differences between the two research domains, it's worth noting that they share a common goal: the creation of learning algorithms that are aware of the input data. Therefore, in order to improve the state of the art in both research domains, we focused on developing data-aware learning algorithms to deal with potentially detrimental inputs encountered during training and operational phases. Harmful inputs encompass everything that undermines the model's quality, including noise, errors, or outliers within the training set, as well as malicious instances crafted by an attacker to cause unexpected model behaviour. Moreover, it's crucial to emphasise that input awareness serves not only to enhance the effectiveness and robustness of models but also their efficiency. Increased awareness of input data allows for strategic choices to minimise the computational costs of the training process.

1.2 Thesis Structure

The thesis is structured into the aforementioned macro areas: Learning to Rank in Part I and Adversarial Machine Learning in Part II. While these areas have distinct characteristics, they share fundamental concepts like machine learning, supervised learning, and ensemble methods. For this reason, the thesis begins with Chapter 2, "Basic Notions", consolidating the foundational knowledge relevant to both parts of the thesis.

Subsequently, the thesis delves into the heart of Part I, "Effective and Efficient Ranking Algorithms", dedicated to our contributions in the domain of learning to rank. The part initiates with Chapter 3, "Background and State of the Art", offering an introduction to the foundational concepts and the current state of the art in learning to rank. This chapter presents evaluation metrics, existing learning

algorithms for LtR, and benchmark datasets.

Part I dedicates a chapter to each article published within the learning to rank research field, providing an in-depth examination of the proposed solutions and the experiments conducted. In particular, Chapter 4 presents *Filtering out Outliers in Learning to Rank*, Chapter 5 introduces *On the Effect of Low-Ranked Documents: A New Sampling Function for Selective Gradient Boosting*, and Chapter 6 discusses *LambdaRank Gradients are Incoherent*. The articles are presented chronologically, highlighting the evolution of research and discoveries made in each work. These three chapters thoroughly examine the contributions and specific state of the art of each work and conduct in-depth analyses of the choices made and the results obtained.

The first part concludes with Chapter 7, "Discussion First Part", where the thesis gathers conclusions and considerations regarding the research in the LtR domain, the discoveries made and the relationships between the articles. Furthermore, it delineates potential future work in this domain and possible extensions of the published articles.

Subsequently, the thesis progresses into Part II, "Robust Learning Algorithms for Classification", dedicated to contributions in the Adversarial Machine Learning domain. This part starts with a detailed introduction to the concept of Adversarial Machine Learning in Chapter 8. This chapter includes the definition of attacker, types of attacks, evaluation metrics, state of the art, and datasets used in the experimental analysis.

In this part, two chapters are dedicated to articles published within the Adversarial Machine Learning domain. Chapter 9 presents *Feature Partitioning for Robust Tree Ensembles and their Certification in Adversarial Scenarios*, while Chapter 10 discusses *Beyond Robustness: Resilience Verification of Tree-Based Classifiers*. This chapter also includes a section on the follow-up work *Explainable Global Fairness Verification of Tree-Based Classifiers*. These articles are presented in chronological order, and each chapter provides a detailed analysis of the proposed solutions, their limitations, and the state of the art considered in each work.

The second part concludes with Chapter 11, "Discussion Second Part", where the thesis compiles conclusions and considerations regarding research in the Adversarial Machine Learning field, the contributions, and potential future work.

Ultimately, the thesis culminates with Chapter 12, "Conclusion". This chapter provides final insights regarding the research performed during the doctoral studies in improving the machine learning domain with data-aware learning algorithms to efficiently train effective, efficient and robust models. Moreover, this chapter provides interesting future works, bonding the distinct research areas covered in the two parts of the thesis.

Chapter 2

Basic Notions

This chapter introduces the fundamental concepts of machine learning, which are shared throughout both parts of the thesis. The chapter also introduces the notation used in this thesis and important concepts and algorithms that are essential to understanding the entire work, such as *Supervided Learning*, *Decision Trees*, *Random Forests*, and *Gradient Boosted Decision Trees*.

2.1 Machine Learning

Machine learning represents a transformative branch of artificial intelligence that endows computer systems with the ability to acquire knowledge, adapt to patterns, and make data-driven decisions without explicit programming. It involves a wide range of algorithms and methodologies designed to allow computers to learn from data. At its core, machine learning seeks to discover patterns, relationships, and insights within complex datasets. It offers invaluable solutions across various domains, from predictive analytics to natural language processing and autonomous systems. It embodies the pursuit of automated learning and decision-making, epitomising the intersection of data science, mathematics, and computer science in the quest to empower machines with cognitive capabilities. Machine learning, a core facet of artificial intelligence, comprises four fundamental paradigms, each with its unique characteristics:

- **Supervised Learning:** Supervised learning relies on data manually labelled by human experts to enable algorithms to learn from examples. One of the most common use cases is classification, where the model assigns labels to unseen input instances based on what it has previously learned.
- **Semi-Supervised Learning:** Semi-supervised learning involves using both labelled and unlabelled data, allowing for a balance between supervision and

data-driven discovery. This approach proves to be quite useful when it is challenging to acquire large labelled datasets.

- **Unsupervised Learning:** Unsupervised learning operates only on unlabeled data, with the objective of discovering hidden patterns or structures within the data. Common applications of unsupervised learning include clustering and dimensionality reduction.

- **Reinforcement Learning:** Reinforcement learning employs agents to interact with the environment in order to maximise cumulative rewards. This method proves to be pivotal in areas such as autonomous decision-making, where the agent learns to make decisions based on the rewards received.

These paradigms illustrate the varied landscape of machine learning, with each tailored to address specific objectives and domains.

This thesis exclusively concentrates on machine learning algorithms employing supervised learning. Consequently, the subsequent section provides a comprehensive examination of the supervised learning paradigm.

2.1.1 Supervised Learning

Supervised learning is a machine learning technique that, given an input instance, returns a score convertible into a label (in the case of classification) or a real value (in the case of regression). Informally, in classification tasks, given a set of correctly labelled objects, the supervised learning technique looks for a function, also known as *classifier*, that effectively discriminates them among the available classes. Subsequently, this function is used to assign labels to new objects. Similar to classification tasks, given a set of objects, each paired with a real value, in regression tasks, the supervised learning technique aims to identify a function named *regressor* that effectively captures the relationship between the input objects and their continuous variable. The regressor is used to assign a real value to new objects. Typically, the output of a supervised learning algorithm is called *learner* or *model*.

Finally, a more formal definition of the classification task is provided. Let $\mathcal{X} \subseteq \mathbb{R}^d$ represent a d-dimensional vector space of real-valued features named feature space, where each element $\boldsymbol{x} \in \mathcal{X}$ is referred to as *instance* and is represented as $\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})$. Each instance $\boldsymbol{x}$ is associated with a label $y \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the output space, also known as *ground truth*. This mapping between instances in $\mathcal{X}$ and labels in $\mathcal{Y}$ is defined by an unknown function $g : \mathcal{X} \to \mathcal{Y}$, named the *target* function. Given the hypothesis set $\mathcal{H}$, the supervised learning algorithm $\mathfrak{L}$ finds the function $h \in \mathcal{H}$ that best approximates the target function g, i.e., the function $\hat{h}$ [120]. The prediction of the function (model) $\hat{h}$ over the instance $\boldsymbol{x}$ is denoted as $\hat{h}(\boldsymbol{x})$.

To accomplish this objective, the learning algorithm necessitates a collection of exemplars that facilitate the acquisition of the correlation between $\mathcal{X}$ and $\mathcal{Y}$. In the domain of supervised learning, the set employed for this specific purpose is referred to as the *training set*. The learning algorithm exploits a training set, $\mathcal{D} = \{(\boldsymbol{x}_1, g(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_n, g(\boldsymbol{x}_n)\}$, with $n = |\mathcal{D}|$, which is a set of (*instance, true label*) pairs. In accordance with statistical learning theory, the function $\hat{h}$ that best approximates g can be found by means of *empirical risk minimisation* [168]. Given the sets $\mathcal{D}$ and $\mathcal{H}$, the empirical risk is defined as a loss function $\mathcal{L} : \mathcal{H} \times (\mathcal{Y} \times \mathcal{X})^n \rightarrow \mathbb{R}^+$. This particular loss function, $\mathcal{L}$, serves as a quantitative measure assessing the cost of an erroneous prediction performed by $\hat{h}(\boldsymbol{x})$ in comparison to the ground truth target function g. To find the hypothesis that minimises the empirical risk, the following optimisation problem must be solved:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \mathcal{L}(\mathcal{Y}, h(\mathcal{X})) \tag{2.1}$$

where $\hat{h}$ is the hypothesis that yields the lowest loss.

Finally, the $\mathcal{L}$ function can be derived by aggregating the *instance-level loss*, represented as $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$. Consequently, the $\mathcal{L}$ function can be explicitly defined as follows:

$$\mathcal{L} = \sum_{(\boldsymbol{x}, y) \in \mathcal{D}} \ell(y, h(\boldsymbol{x})). \tag{2.2}$$

This approach comprehensively assesses the cumulative loss incurred across the entire training set.

2.1.2 Decision Trees

A Decision Tree (DT) is one of the most well-known machine learning models, deriving its nomenclature from its tree-like structure. A DT comprises two principal components: internal nodes and leaves. Each node within the tree conducts a feature test, commonly referred to as a split, on the input data. Typically, every node is associated with a feature f and a threshold value v, upon which the test $f \leq v$ is performed.

The data traversing a node undergoes division into subsets based on the outcome of the test. Each leaf in the tree contains either a label in the case of classification or a numerical value in the context of regression. In the prediction phase, instances are assigned to the label or value corresponding to the leaf they ultimately reach. The splitting process is applicable to both categorical and numerical features.

For categorical features, the chosen category serves as the discriminator for the elements, dividing the set into those belonging to the selected category and

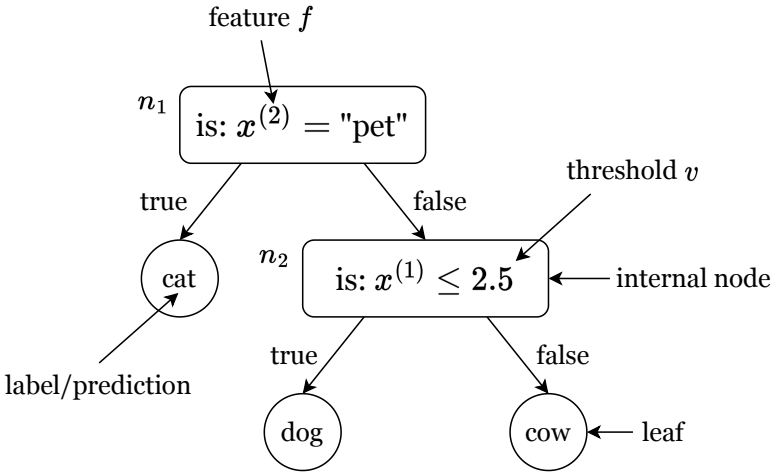


Figure 2.1: Decision tree example, with numeric and categorical features.

those that do not. When dealing with numeric features, the division occurs based on whether an element's value for the tested feature is lower or higher than a predefined threshold. In Figure 2.1, an illustrative example of a decision tree is presented, showcasing both categorical split (n_1) and numeric split (n_2).

So far, only the prediction phase of the model has been mentioned; however, it is equally important to provide an overview of the tree training phase. The creation of a Decision Tree (DT) typically involves an iterative process where the best-split point is selected at each step. This split divides the input dataset into subsets, which then become the input for subsequent steps, progressively reducing the size of the dataset. The training process continues until a specific termination condition, known as stop criteria, is met. Various stop criteria are employed to control the tree's growth, including *max depth*, which halts tree growth upon reaching a specified height, *max leaves*, which limits the number of leaves, and *purity*, which concludes growth when the majority class of a leaf exceeds a predefined threshold compared to the others.

During the learning phase, the primary objective is to identify the most effective way to divide the data into two groups, which is known as the optimal split. This optimal split indicates the feature f and value v that provide the best partition for the set $\mathcal{D}$. As detailed by Zhou in [190], several methods have been proposed to determine the best split. The most common criteria are Information Gain, Gain Ratio, and Gini Index for classification tasks, as well as Mean Square Error, Mean Absolute Error, and R-squared for regression tasks.

It's essential to note that merely pursuing the best split does not guarantee the creation of a well-generating decision tree. Overfitting, a phenomenon where the model adapts too closely to the training set, often learning from irrelevant

noise, can lead to suboptimal performance on unseen data. Techniques like node pruning, bootstrap sampling, and feature sampling have been employed to address overfitting. Further insights into these strategies can be found later in this thesis or at the following sources: [190, 63].

This thesis mainly employs binary decision trees, specifically designed for handling numeric features and performing binary classification tasks. Therefore, we provide an inductive definition of decision tree as defined by Calzavara *et al.* in [35]. A decision tree, t, can assume one of two roles: it can either be a leaf or a non-leaf (internal node). A leaf is represented as $\lambda(\hat{y})$, where $\hat{y} \in \mathcal{Y}$ is its class label. On the other hand, a non-leaf is characterised as $\sigma(f, v, t_l, t_r)$, with the following components:

- $f \in \{1, \ldots, d\} = \mathcal{F}$ designates a feature from the feature space $\mathcal{X}$.
- $v \in \mathbb{R}$ serves as the threshold to test the value of feature f.
- t_l and t_r correspond to the decision trees situated in the left and right branches of t, respectively.

During the classification phase, an instance $\boldsymbol{x}$ traverses the decision tree t, starting from the root and ending at a leaf $\lambda(\hat{y})$. The leaf reached by $\boldsymbol{x}$ contains the prediction $\hat{y} = t(\boldsymbol{x})$. The traversal from the root to a leaf is made through the internal nodes. Each internmal note $\sigma(f, v, t_l, t_r)$ encounteredd by the isntance $\boldsymbol{x}$ performs the feature test $x^{(f)} \leq v$. If the answer is positive, the instance proceeds along the left branch, otherwise toward the right branch. Consequently, at each feature test, the instance progresses through a single branch at a time, ultimately reaching a single leaf.

2.2 Ensemble Methods

This section provides an overview of ensemble methods, focusing on Random Forest (RF) [20] and Gradient Boosting Decision Trees (GBDTs) [69].

Usually, the standard approach to machine learning involves employing a single learner to address a specific problem. On the other hand, ensemble methods amalgamate multiple learners to accomplish the same task; this paradigm is called ensemble learning. These individual learners within the ensemble are often called *base learners* or *weak learners* and are trained through a base-learning algorithm. To get a good ensemble, individual learners should be as accurate as possible and as different as possible from one another.

Typically, creating an ensemble is divided into two parts: generating individual base learners and combining them through a combination strategy. These two

parts are not necessarily separate but can also occur simultaneously. Below is an example of this through two ensemble creation paradigms: sequential and parallel ensembling methods.

Parallel ensemble methods create base learners simultaneously, which makes their construction independent from each other. This independence can be exploited to reduce training time through parallelised model training. This paradigm leverages the capability to decrease ensemble error by combining independent learners (e.g., Bagging [19]). The combination occurs by aggregating the predictions of each base learner to create a single ensemble prediction. One of the most well-known combination strategies is *majority voting*.

In contrast, sequential ensemble methods generate base learners one after another. This allows the exploitation of dependencies between learners and an attempt to enhance model performance by introducing new learners (e.g., Gradient Boosting [69]). In general, this paradigm aims to correct the errors of the ensemble trained previously by adding a new base learner. In this case, the combination happens between the ensemble and the newly trained base learner. Consequently, the ensemble is formed as a chain of base learners. A well-known combination strategy is the weighted sum of the output from each base learner.

Overall, ensemble methods possess two crucial advantages. First, combining base learners usually results in better generalisation than individual learners. Second, learning multiple base learners and merging their predictions is often easier than creating a single powerful learner.

2.2.1 Random Forest

Random Forest [20] is among the most renowned ensemble methods, and it vividly exemplifies the concept of independence among base learners achieved through randomness. The *Forest* component in Random Forest is derived from its assembly of Decision Trees (forest), while the *Random* part stems from the randomisation that is performed with two strategies: bootstrap and feature sampling [63].

As previously mentioned, the fundamental combination of independent learners results in a more efficient ensemble. However, the size of the training set is finite, so it's not always feasible to divide the training set into equal parts and train a base learner for each of them. This limitation arises from the fact that each base learner would be trained on an excessively small subset.

Bootstrap sampling [63] is a strategy employed to establish independence among base learners through a trade-off between the diversity of instances within the samples and the maximum size of the samples. Precisely, with n instances in the training set, bootstrap sampling can generate samples of size n by selecting instances randomly with replacement. As a result, two different samples may be disjoint or may share only a small subset of instances. This approach facilitates

the creation of independent base learners by training on a distinct subset of the training set.

Feature sampling [20], on the other hand, covers the training phase of Decision Trees. Specifically, the selection of the optimal split is not based on all the features within the feature set $\mathcal{F}$; instead, it relies on a randomly chosen subset of features $\hat{\mathcal{F}} \subseteq \mathcal{F}$ typically of size k. This randomisation, introduced by feature sampling, enhances model generalisation since each node within the tree employs a distinct subset of features for data splitting. Note that while each base learner may encounter a reduction in performance due to the reduced search space, this effect is mitigated when they are aggregated in an ensemble.

Lastly, the aggregation of predictions involves majority voting for classification tasks and averaging for regression tasks. The majority voting strategy is divided into two categories: hard majority voting, where the most frequently predicted label among the base learners is selected as the final prediction, and soft majority voting, where the prediction of each base learner contributes to a weighted vote. Finally, for regression tasks, the result is computed by averaging the predictions generated by all the base learners. The algorithm for Random Forest is provided in Algorithm 1 [120].

2.2.2 Gradient Boosted Decision Trees

The Gradient Boosting Decision Tree (GBDT) [69] is an ensemble method that leverages the Gradient Boosting construction algorithm (GB), employing the decision tree learning algorithm as its base learner. The GBDT method is suitable for classification and regression tasks, particularly effective when dealing with noisy or imperfect data.

As mentioned earlier, the core element of GBDT is Gradient Boosting. GBDT is an iterative meta-algorithm that guides the combination of base learners to create more effective models. In each iteration of GBDT, a gradient descent step is taken by introducing a new base learner to the ensemble. For instance, when GBDT optimises a regression task with MSE, the new base learner corrects the error of the ensemble up to that point. This newly introduced base learner is then integrated into the ensemble to enhance its overall performance. These base learners essentially represent functions, and their combination allows the creation of a more powerful function that fits the data more effectively.

Even though Gradient Boosting Decision Trees is extensively used in Part I, a deep knowledge of how it works is not necessary. Therefore, we provide only a brief description of this learning algorithm below. Additional information regarding GBDT can be found in the references cited in the text below. Furthermore, the algorithm for Gradient Boosting Decision Trees is presented in Algorithm 2 [120].

Algorithm 1 Random Forest

function RandomForest($\mathcal{D}$, $\mathcal{F}$, N)
  Input
    $\mathcal{D}$: training set
    $\mathcal{F}$: feature set
    N : ensemble size
  Output
    F : final ensemble

  $F \leftarrow \emptyset$
  for $i = 1$ **to** N **do**
    $S_i \leftarrow$ BootstrapSampling($\mathcal{D}$)
    $t_i \leftarrow$ RandomForestTree($S_i, \mathcal{F}$)
    $F \leftarrow F \cup \{t_i\}$
  return F

function RandomForestTree($\mathcal{D}, \mathcal{F}$)
  Input
    $\mathcal{D}$: training set
    $\mathcal{F}$: feature set
  Output
    t : decision tree

  if *all instances in the same class* $\vee$ *stopping criteria has been reached* **then**
    return $\lambda(\hat{y}(\mathcal{D}))$
  $\hat{\mathcal{F}} \leftarrow$ RamdomFeatureSampling($\mathcal{F}, k$)
  $(f, v) \leftarrow$ BestSplit($\mathcal{D}, \hat{\mathcal{F}}$)
  $t_l \leftarrow$ RandomForestTree($\mathcal{D}^{(f)} \leq v, \mathcal{F}$)
  $t_r \leftarrow$ RandomForestTree($\mathcal{D}^{(f)} > v, \mathcal{F}$)
  $t \leftarrow \sigma(f, v, t_l, t_r)$
  return t

Friedman in [69, 70] formalised the GB and GBDT algorithms in a generalised manner with a minimal dependency on the specific loss function used. More formally, consider a learning algorithm $\mathfrak{L}$ responsible for training a decision tree with L leaves, tailored for regression tasks, known as the L-leaves_learner. At each iteration m of the gradient boosting algorithm, the L-leaves_learner constructs a tree that partitions the training set $\mathcal{D}$ into L distinct regions, denoted as $\{R_{lm}\}_{l=1}^{L}$.

Following the empirical risk minimisation principle, GB seeks the optimal γ value to minimise the ensemble error based on the loss function $\mathcal{L}$:

$$\gamma_{lm} = \arg\min_{\gamma} \sum_{\boldsymbol{x}_i \in R_{lm}} \mathcal{L}(y_i, h_{m-1}(\boldsymbol{x}_i) + \gamma). \tag{2.3}$$

This allows for the creation of the next strong learner, denoted as $h_m(\boldsymbol{x})$, by updating the outcomes of each leaf (region) in the learner $h_{m-1}(\boldsymbol{x})$ with the new values estimated at iteration m. Note that here, we use the notation h_m for ensembles of m trees instead of F_m to focus on the aggregation of functions of GDBT. Finally, the base learner is aggregated into the ensemble using the following aggregation formula:

$$h_m(\boldsymbol{x}) = h_{m-1}(\boldsymbol{x}) + \nu \cdot \gamma_{lm} 1(\boldsymbol{x} \in R_{lm}), \tag{2.4}$$

where $0 < \nu \leq 1$ is the *shrinkage* parameter designed to prevent overfitting.

Algorithm 2 Gradient Boosting Decision Trees

function GradientBoostingDecisionTrees($\mathcal{D}$, N)
  Input
    $\mathcal{D}$: training set
    N : ensemble size
  Output
    $h_m(\boldsymbol{x})$: final ensemble

  $h_0(\boldsymbol{x}) \leftarrow \arg\min_\gamma \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}(y_i, \gamma)$
  for $m = 1$ **to** N **do**
    $\tilde{y}_{im} \leftarrow -\left[\frac{\partial \mathcal{L}(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=h_{m-1}(\boldsymbol{x})}, i = 1, \ldots, |\mathcal{D}|$
    $\{R_{lm}\}_{l=1}^{L} \leftarrow$ L-leaves_learner($\{\tilde{y}_{im}, \boldsymbol{x}_i\}_{i=1}^{|\mathcal{D}|}$)
    $\gamma_{lm} \leftarrow \arg\min_\gamma \sum_{\boldsymbol{x}_i \in R_{lm}} \mathcal{L}(y_i, h_{m-1}(\boldsymbol{x}_i) + \gamma)$
    $h_m(\boldsymbol{x}) = h_{m-1}(\boldsymbol{x}) + \nu \cdot \gamma_{lm} 1(\boldsymbol{x} \in R_{lm})$
  return $h_m(\boldsymbol{x})$

2.3 Summary

In this first chapter, we introduced some fundamental Machine learning concepts that lay the foundation for the two parts of this thesis. Below, we provide a summary of the main concepts covered in this chapter:

- **Supervived Learning:** The first part of the chapter focused on supervised learning, which is the primary learning technique underlying the models used in this thesis. Supervised learning, utilises empirical risk minimisation to find the hypothesis $\hat{h}$ that best approximates the target function g when provided with a training dataset.

- **Decision Trees:** Next, we delved into the Decision Trees learning algorithm. We explained the structure of a decision tree, consisting of internal nodes and leaves, and explained how instances are partitioned within the tree's internal nodes, leading to predictions at the leaves. Furthermore, we presented a formal definition of the Decision Tree, which is extensively employed in the second part of the thesis.

- **Ensemble methods:** This section introduced the concept of ensemble learning and base learners. It outlined the two-step process of creating ensembles, including base-learner generation through a **base-learning algorithm** and their combination through a **combination strategy**. Finally, we explained how exploiting dependence and independence between learners is possible through a sequential and parallel construction, respectively.

- **Random Forest:** The "Random Forest" section holds fundamental importance within this chapter, as Random Forest is extensively employed in the second part of the thesis. A comprehensive understanding of Random Forest's internal structure is essential for understanding the experiments and providing an explanation of the results. For this reason, this section introduced key concepts such as **bootstrap sampling** and **feature sampling**. Furthermore, it is crucial to comprehend the mechanics of Random Forest's prediction, where individual tree's predictions are aggregated through **majority voting**.

- **Gradient Boosting Decision Trees:** Lastly, in the concluding section of this chapter, we introduced Gradient Boosting and its adaptation to decision trees, known as Gradient Boosted Decision Trees. In essence, GBDT trains each tree to correct the error made by the prediction of the ensemble trained until that moment. GBDT models are widely utilised in the first part of the thesis, as they are considered the state of the art in Learning to Rank.

Part I

Effective and Efficient Ranking Algorithms

Chapter 3

Background and State of the Art

This chapter introduces the first part of this thesis, i.e., Part I: *Effective and Efficient Ranking Algorithms*. This first part covers the contribution made to the Information Retrieval research area, focusing on learning-to-rank techniques, specifically on their effectiveness and efficiency. Instead, this chapter provides the background and the state of the art of Information Retrieval and Learning to Rank research areas to fully understand the contribution of this part of the thesis. The chapter encompasses the definition of Learning to Rank, the metrics to evaluate the model's effectiveness, the key solutions and the research within the LtR domain, and finally, an exploration of the most common benchmark datasets.

3.1 Information Retrieval

Information Retrieval (IR) is a fundamental discipline in the field of computer science and information science. IR is the science of searching for information within documents, searching for documents relevant to a query (queries are formal statements of information needs), or searching for metadata describing data. It plays a pivotal role in solving the problem of *information overload*, i.e., the difficulty of making decisions or extracting useful information from a large amount of data. In our modern, data-driven world, the exponential growth of digital information necessitates efficient methods for finding, accessing, and retrieving relevant data. This discipline is not only a cornerstone of web search engines but also underlies numerous applications, including document management systems, recommendation engines, and intelligent personal assistants.

In [119], Information Retrieval is defined as "finding material of an unstructured nature that satisfies an information need from within large collections." This succinctly captures the essence of IR, which revolves around the processes of searching for, assessing, and retrieving information that aligns with a user's spe-

cific information needs. The information can be in various formats, such as text documents, multimedia content, or structured data.

IR systems bridge the gap between users and vast repositories of information by employing a variety of techniques and algorithms. These systems employ methods from fields like natural language processing, machine learning, and data mining to index, rank, and retrieve relevant content efficiently.

Key components of an IR system include:

- **Document Collection:** A vast collection of documents, which could range from web pages and scientific articles to books and multimedia content.
- **Query:** A user's information need, expressed as a query, typically consisting of keywords or a more complex search string.
- **Indexing:** The process of analysing and cataloguing documents to facilitate efficient retrieval.
- **Ranking:** The algorithmic process of scoring and ordering documents based on their relevance to a given query.
- **Evaluation:** Methods to assess the performance of an IR system, typically using metrics like Precision [119], Recall [119], and F1-score [50].

Over the years, IR has evolved significantly, adapting to the changing landscape of information and user expectations. Today, the development of IR systems is driven by ongoing research in the field, with a focus on enhancing precision, personalisation, and scalability.

In conclusion, Information Retrieval is a critical discipline that empowers individuals and organisations to access the vast ocean of digital information efficiently. It combines the power of advanced technologies with a deep understanding of user needs to deliver timely, relevant, and valuable information.

3.2 Learning to Rank

One of the most classic examples of Information Retrieval (IR) applications is Web Search Engines. Web Search Engines take as input a user needs (a query) and retrieve a narrowed selection of documents from a pool of web pages, sorted by their relevance [101, 180, 61]. A widely used strategy in this domain is Learning to Rank (LtR). LtR is a class of machine learning techniques that apply supervised learning to solve ranking problems.

The main difference between LtR and traditional supervised learning techniques such as classification and regression is the type of problem they solve.

Traditional ML algorithms solve a prediction problem on independent instances, and the prediction result does not influence the outcome/performance on the other instances. In a classification task, given an instance $\boldsymbol{x}$, a classifier aims to predict a label $\hat{y}$ that best approximates the ground truth y. On the other hand, LtR solves a ranking problem on a list of items. In LtR, a ranker receives a query and a list of items as input and produces an order over the input list such that the most relevant items for the query are placed at the top of the list. Consequently, the position of the items in the list also influences the position of the others.

3.2.1 Formal Definition

Typically, in Information Retrieval applications, the Learning to Rank technique is implemented in the re-ranking stage of the query processing pipeline (i.e., first-stage retrieval and then re-ranking).

The query processing pipeline involves a first-stage retrieval, also known as sparse retrieval, where, given a user query q, a set of *candidate documents* D is extracted from a vast collection of documents. This step is fast and efficient and is carried out using strategies such as inverted index on tokens (words) within the documents or through more sophisticated metrics like BM25 [149]. Despite being very efficient, the sparse retrieval stage is not sufficiently accurate; hence, a second step, known as re-ranking or dense retrieval, is needed to refine the results. For each query q and associates documents in D, a dense representation is created using hand-crafted, manually-engineered features (we delve into details later in this section). This dense representation is given as input to a learning-to-rank model that produces a ranking over the set of candidate documents in D. Therefore, Learning to Rank focuses solely on the re-ranking stage, ignoring the first part of the pipeline.

In the context of Learning to Rank, an instance is defined as the triple (q, D, Y), where q is the query, D is the set of candidate documents for q, and Y is the set of relevance labels that each document in D has with respect to the query q. The set $D = \{d_1, \ldots, d_n\}$ is a subset of the available document collection in the training set. The set $Y = \{y_1, \ldots, y_n\}$ contains relevances drawn from the set $\mathcal{Y}$. Typically, $\mathcal{Y} = \{0, 1, 2, 3, 4\}$, with 0 denoting a non-relevant document and 4 representing the maximum relevance level with respect to query q.

The purpose of LtR is to train a ranker h that, given a query q and a set of candidate documents D, returns a ranking π over the set of documents, according to which documents are sorted and presented to the user. To generate the ranking π, the ranker predicts a score $s_i = h(d_i)$, with $s_i \in \mathbb{R}$, for each $d_i \in D$, and then sorts the documents in descending score order. As a result, $\pi[i]$ denotes the position of document d_i in the ranking; therefore, $\pi[i] = j$ when d_i appears at the top j-th position in the ranked list. Ideally, the ranker aims to produce a ranking

that aligns closely with the ideal ranking based on the ground-truth relevance labels in Y. The better the scores s_i generated by the ranker approximate this ideal ranking, the more effective (accurate) the ranker h is.

In LtR, the training set is defined as: $\mathcal{D} = \{(q_1, D_1, Y_1), \ldots, (q_n, D_n, Y_n)\}$, with $n = |\mathcal{D}|$. Note that a document d_i can be a candidate document for multiple queries so that it can belong to different sets of candidate documents with a different relevance label for each query. Furthermore, each query can have a different size of the candidate documents set.

Note that, in practice, LtR instances have the information of the query q combined (embedded) with the information of each document d_i to create a single *query-document* feature vector $\boldsymbol{x}_i$. The features in the feature vector $\boldsymbol{x}_i$ are query-dependent, document-dependent, and query-document-dependent. Formally, $\boldsymbol{x}_i = \phi(q, d_i)$, where ϕ is a function that combines information from the query q and the document d_i together. As mentioned by Lin *et al.* in [103], the resulting vector $\boldsymbol{x}_i$ consists of hand-crafted, manually-engineered features such as the number of query words appearing in the document corpus or the result of metrics like BM25 [149], Tf-Idf [151], PageRank score [21], etc. For the sake of clarity, in this thesis, we refer to LtR instances in the form of a triple (*query*, *documents*, *relevance labels*); this simplifies the distinction between queries and documents.

3.2.2 Evaluation Metrics

Another difference in LtR is the metrics to be optimised, which take into account the rank of the documents generated by the model. The most well-known IR metrics are briefly summarised below:

Normalized Discounted Cumulative Gain (NDCG) [88] measures the quality of a ranked list by considering both the relevance of the documents and their positions in the list. It performs a greater discount to the relevance scores of documents that are ranked lower on the list compared to those placed at the top, thus giving higher importance to high-ranked documents. *Precision* [119] measures the proportion of relevant documents among the top-ranked ones, while *Recall* [119] calculates the proportion of relevant documents that are retrieved in the top-ranked list. These metrics provide insights into how well a model balances relevance and coverage in its ranking. *Average Precision* (AP) [188] calculates the average precision for a single query. It sums the Precision values at each relevant document's position and divides it by the total number of relevant documents. *Mean Average Precision* (MAP) [106] is similar to AP but takes the mean of the AP for each query. *Expected Reciprocal Rank* (ERR) [44] is a metric that takes into account the graded relevance of documents. It focuses on the first relevant document and penalises the model more if it's ranked lower in the list. *Rank-Biased Precision* (RBP) [127] is a metric that considers document relevance as well as the likelihood of a user viewing an

item and the probability of stopping the list inspection. It's particularly useful for modelling user behaviour, where users often examine top-ranked results, making it valuable in search engine evaluation.

3.2.2.1 Normalized Discounted Cumulative Gain

The Normalized Discounted Cumulative Gain is one of the most well-known and widely used IR metrics for measuring the quality of a ranked list in Learning to Rank. For this reason, in the experimental part of Part I, we extensively use NDCG to evaluate the effectiveness of the models. The metric is examined in detail below.

The Normalized Discounted Cumulative Gain is a metric defined within the $[0, 1]$ range, where 1 indicates a perfect ranking and 0 represents the worst. It is used to assess the performance of a ranker for a specific query. The NDCG for a query q, i.e., $\text{NDCG}(\pi, Y)$, is computed as the ratio between the *Discounted Cumulative Gain* (DCG) and the *Ideal Discounted Cumulative Gain* (IDCG), expressed by the formula:

$$\text{NDCG}(\pi, Y) = \frac{\text{DCG}(\pi, Y)}{\text{IDCG}(\pi, Y)} = \frac{\text{DiscountedCumulativeGain}}{\text{IdealDiscountedCumulativeGain}}.$$

where π is the ranking produced by the ranker over the query q and Y is the set of relevances that documents in D have with respect to q.

The DCG accumulates a gain for each document based on its position inside the ranked list. The idea behind DCG is intuitive; a relevant document positioned at the top of the list is more likely to be seen by a user (high gain) than when it is located further away (low gain). DCG comprises two essential components: the gain G and the discount D. Typically, the two components are defined as follows: $\text{G}_i = 2^{y_i} - 1$ is the gain that document d_i, with relevance y_i, brings to the final ranking, and $\text{D}_i = \log_2(1 + \pi[i])$ is the discount of the document's gain according to its position $\pi[i]$ in the list. As it can be easily seen, due to this definition of discount, a very relevant document in the last position contributes much less to DCG than in the first position. The IDCG, instead, is the ideal DCG computed exactly like the DCG but on the ideal ranking produced through the ground truth. The $\text{DCG}(\pi, Y)$ is defined as follows.

$$\text{DCG}(\pi, Y) = \sum_{i=1}^{|D|} \frac{\text{G}_i}{\text{D}_i} = \sum_{i=1}^{|D|} \frac{2^{y_i} - 1}{\log_2(1 + \pi[i])}. \tag{3.1}$$

Note that several other ranking metrics can be similarly formulated with different gains G and discounts D and with proper normalisation.

The normalisation through the IDCG serves to prevent the model from favouring long queries since queries with a lot of candidate documents are likely to have a higher DCG. Furthermore, this allows for the comparison of queries of different lengths.

Real-world applications of information retrieval systems mostly focus on optimising the effectiveness of the top-k results, aligning with observed user behaviour [62]. Users tend to concentrate their interest on the initial k (e.g., 5/10) results when scanning a list of items rather than examining the entire extensive list. IR metrics naturally accommodate this behaviour by introducing a *cutoff threshold* k, leading to truncated versions of metrics. For example, NDCG@k evaluates the goodness of the ranking only for the top-k ranked documents. Truncated metrics are particularly relevant in evaluating rankers applied in real scenarios. So, due to the empirical risk minimisation principle, optimising a truncated metric is expected to be more effective than optimising its un-truncated counterpart.

The truncated version of NDCG can be easily computed by adding to the DCG formula in Equation 3.1 an indicator faction $\mathbb{1}[\pi[i] \leq k]$ that returns 1 for documents ranked within the cutoff k and 0 otherwise. Straightforward, this applies also to the IDCG. The NDCG@$k(\pi, Y)$ is defined as follows:

$$\text{NDCG@}k(\pi, Y) = \frac{\text{DCG@}k(\pi, Y)}{\text{IDCG@}k(\pi, Y)} = \frac{1}{\text{IDCG@}k(\pi, Y)} \sum_{i=1}^{|D|} \frac{\text{G}_i}{\text{D}_i} \mathbb{1}[\pi[i] \leq k]. \quad (3.2)$$

Note that the truncated version of NDCG maintains the same characteristics as its un-truncated counterpart, e.g., the definition in $[0, 1]$ and the possibility to compare queries of different lengths.

Finally, the overall effectiveness of a ranker on a set $\mathcal{D}$ can be computed as the average NDCG (or NDCG@k) across all queries in $\mathcal{D}$.

$$\text{NDCG@}k = \sum_{i=1}^{|\mathcal{D}|} \text{NDCG@}k(\pi_i, Y_i), \quad (3.3)$$

For the sake of readability, hereinafter, we use the notation NDCG@k instead of NDCG@$k(\pi, Y)$ when there is no ambiguity with the average NDCG, and the query and its ranking are easily drawn from the context.

3.2.3 Learning Algorithms

In this section, we delve into the state-of-the-art in the domain of Learning to Rank. Specifically, this section is divided into three major research areas within the field of Learning to Rank: effectiveness, efficiency, and fairness.

3.2.3.1 Effectiveness

As mentioned above, most IR metrics consider the ranking generated on the predicted document scores. This inherently ties the result of the metric with the order of the documents; consequently, the ranking metrics are non-differentiable and flat-everywhere [22, 122].

Unlike traditional machine learning methods, an objective function that includes IR metrics cannot be optimised directly using gradient descent techniques. This makes it incompatible with gradient-based optimisation, making the optimisation of IR metrics a significant challenge in the field of Learning to Rank. Nevertheless, proficient ranking models hold paramount importance across a wide spectrum of Information Retrieval systems. As a result, there has been a competitive pursuit to tackle this problem from various perspectives.

Several solutions have been devised to solve the LtR paradigm with ML. The main approaches to solve this problem are categorised into three classes: *pointwise*, *pairwise*, and *listwise* approaches [101].

Pointwise Pointwise approaches treat each document in the training set independently of others and without considering the query's context. Consequently, they disregard the inherent ranking structure. In the Pointwise approach, the training process typically resembles supervised classification or regression. The main objective of the loss function is to predict each document individually and minimise the error between the predicted outcome and the document's ground truth relevance [153, 102]. Some examples of pointwise algorithms include McRank [102], which employs Gradient Boosting Trees to address ranking problems as multiclass classification tasks. It leverages a classification model capable of assigning a probability to each document, indicating its membership to different grades. These expected grades are then used for ranking. Another algorithm is Subset Ranking [53], which minimises a surrogate loss function defined on regression errors, which is bounded for DCG.

Pairwise Pairwise approaches reformulate the ranking problem into pairwise classification or pairwise regression tasks. While these approaches still disregard the ranking structure, they maintain the separation by queries. The input is transformed into *preference pairs* of documents within the same query. If document d_i is more relevant than d_j (i.e., $y_i > y_j$), then the pair (d_i, d_j) is a preference pair, indicating that document d_i should be ranked ahead of d_j in the output list. These preference pairs can be treated as instances, with the intra-pair order serving as their labels in a new classification problem. The optimisation problem aims to minimise the number of errors committed in ranking pairs of documents. Exiting classification methods can be used to train a classifier.

Among the pairwise algorithms, one notable approach is RankNet [26], which is a pairwise strategy that optimises a probabilistic loss function by mapping the model's output to a learned probability. There's also a version of RankNet that employs gradient boosting, known as GBRank [68]. Another well-known approach is AdaRank [182], which utilises boosting to learn weak rankers that minimise the pairwise misranking error. These weak rankers are then combined linearly to make predictions. Another example of a pairwise approach is Support Vector Machines for Ranking (Ranking SVM) [80], which constructs pairs of documents and trains a model to order these pairs through the SVM [52] algorithm.

Pairwise approaches typically optimise a convex upper-bounds of the pair misranking error. However, this optimisation does not directly imply an improvement in the ranking metric, thus leading to a mismatch between model optimisation and effectiveness on the desired metric.

Listwise The listwise approach tackles the ranking problem more naturally by incorporating information about the entire ranking list into the optimisation process. A listwise approach finds the optimal permutation of all available documents for a given query. This approach maintains the group structure of ranking, allowing for a more direct integration of ranking evaluation measures into the loss functions. In this scenario, conventional Machine Learning strategies cannot be directly applied.

Listwise approaches can be broadly categorised into two groups. The first group approximates the ranking metric using a smooth surrogate, as seen in SoftRank [161] and ApproxNDCG [142]. The second group employs heuristics to construct a smooth surrogate loss function as did for ListNET [37], ListMLE [178], XE_{NDCG}[22], and LambdaRank [25]. ListNET minimises the cross-entropy between the ground truth and the model's score distribution. On the other hand, ListMLE formulates LtR as a problem of minimising the likelihood loss function, which is equivalent to maximising the likelihood function of a probability model. In contrast, XE_{NDCG} [22] is similar to ListNET but defines an alternative cross-entropy loss function that guarantees strong theoretical properties, such as optimising a convex bound on mean NDCG.

Furthermore, LambdaRank is an approach initially designed for artificial neural networks that does not attempt to optimise a loss function but heuristically defines the loss gradient λ with respect to the model's score. LambdaRank is a pairwise algorithm that aims to order pairs of documents correctly; however, it embeds the information about the entire list's status within the optimisation process. For this reason, it can also considered a listwise approach. A variant of LambdaRank, called LambdaMART [27, 176], combines LambdaRank with MART (Multiple Additive Regression Trees) [69]. LambdaMART is widely regarded as the state

of the art in Learning to Rank. In a related work [173], the authors proposed a probabilistic framework for ranking metric optimisation called LambdaLoss. They demonstrated how LambdaRank is a special configuration with a well-defined loss in their framework. This work also provided specific loss functions with theoretical guarantees, bounding the NDCG and ARP [89].

Moreover, the Plackett-Luce model [116, 140] has been widely used to model a probabilistic distribution over rankings [23, 83, 179]. It defines a probability distribution over permutations of items based on their scores or relevances.

Typically, listwise and pairwise approaches outperform pointwise approaches due to their ability to consider and leverage the inherent structure and relationships within the ranking, which is crucial for accurately modelling the complex nature of ranking tasks. As mentioned above, listwise approaches directly optimise ranking metrics, incorporating the status of the entire ranking list into the optimisation process. While pairwise approaches focus on capturing relative preferences between pairs of items, which can be more informative than the absolute judgments used in pointwise approaches. This inherent consideration of ranking dynamics and relationships provides listwise and pairwise approaches with an advantage in effectively modelling and optimising for ranking tasks [37, 178].

3.2.3.2 Efficiency

Research in the Learning to Rank (LtR) domain is not solely focused on enhancing the effectiveness of rankers but also on improving the efficiency of algorithms during training and operational phases. This becomes particularly crucial when the system interacts with users, who seek faster response times to maintain their engagement with the platform [3, 129, 38].

In this regard, several strategies have been developed to expedite the model's operational phase. For instance, in [114, 28], documents unlikely to reach top positions are halted early during the evaluation to reduce the overall prediction time. Furthermore, various forest pruning strategies aim to reduce the number of trees to traverse during the prediction phase. In [172], the authors devised a strategy to identify and safely remove trees while preserving the accuracy of the original model. Additionally, in [113], trees that contribute minimally to the ranking are removed, significantly reducing the ensemble size.

Finally, some research focuses on making the most of the available hardware. For example, Ye *et al.* in [184] developed RapidScorer in which the particular representation of the trees has allowed them to exploit the SIMD instructions of modern CPUs (parallel computation). Dato *et al.* in [56] and Lucchese *et al.* in [112] proposed QuickScorer with different optimisations to make the sequential tree evaluation more cache-aware.

3.2.3.3 Fairness

Although this thesis does not focus on fairness in LtR, it is important to mention some solutions within the LtR domain regarding this issue. In fact, fairness in LtR is a burgeoning field of research that is gaining traction, addressing the equity of models [71, 59]. Traditional ranking models generate a static list of items ordered by their relevance to the user information need. However, within this list, there could be items equally relevant to the user's query. Users typically scan the list from the first to the last item, stopping when they find a relevant one. Consequently, items placed at the top of the list receive more exposure. This static ordering inherently creates some disparity among equally relevant elements. Once the model defines an order for the elements in the list, it remains fixed. Various research studies have emerged to tackle this issue, attempting to create a stochastic ordering based on document relevance [95, 155]. Thus, documents with equal relevance should have an equal probability of appearing before each other. Consequently, each time the model is queried, a new order is sampled based on the learned probabilities. For instance, in [135, 136], an efficient solution using Plackett-Luce models is proposed to address fairness and relevance problems in LtR. Additionally, in [23], a stochastic framework is presented that samples a list order from a distribution defined by raw scores.

Additionally, the datasets used to train machine learning models, including those in Learning to Rank, may contain biases related to gender, race, nationality, and so on. Unfortunately, the model can learn these biases, leading to discriminatory rankings. Addressing and mitigating these biases is a critical concern in LtR to ensure fair and unbiased outcomes in item ranking. Various fairness-aware approaches and techniques are being developed to tackle this important challenge in the field of information retrieval and machine learning. Solutions like DELTR [187] and Fair-PG-Rank [155] introduce the concept of fairness during the optimisation phase, aiming to create fairness models with respect to the notion of exposure. DELTR optimises for equal exposure, defining exposure as a document's probability of appearing in the top position of a ranking for a specific query. On the other hand, Fair-PG-Rank defines exposure as expected attention, which the authors equate to the expected position bias. These approaches incorporate the concept of fairness to ensure that items have a balanced chance of appearing at top positions in the ranking, promoting a fair and equitable ranking outcome.

3.2.4 Benchmark Datasets

In this final section of the thesis, we introduce the LtR datasets used to evaluate the effectiveness of the proposed solutions. The datasets mentioned in this section are the only ones used in the experimental part; other LtR datasets are not men-

Table 3.1: Datasets properties.

Dataset	#feat.	#queries	#doc.	$\overline{\text{query len.}}$	%non-rel.
Istella-X [115]	220	10,000	26,791,447	2,679.14	99.83
Istella-S [111]	220	33,018	3,408,630	103.24	88.61
Istella-F [56]	220	33,018	10,454,629	316.63	96.29
Yahoo! Set 1 [42]	519	29,921	709,877	23.73	26.09
MSLR-30K [141]	136	31,531	3,771,125	119.60	51.47

Table 3.2: Relevance distribution among datasets. The table reports the number of documents for each integer relevance label in the set $\mathcal{Y}$.

	Relevance distribution				
Dataset	0	1	2	3	4
Istella-X [115]	26,745,076	26,604	5,108	9,619	5,040
Istella-S [111]	3,020,406	83,167	135,989	93,957	75,111
Istella-F [56]	10,066,405	83,167	135,989	93,957	75,111
Yahoo! Set 1 [42]	185,192	254,110	202,700	54,473	13,402
MSLR-30K [141]	1,940,952	1,225,770	504,958	69,010	30,435

tioned in this thesis. All the datasets used are publicly available, and each dataset provides graded relevance labels, ranging from 0 (indicating irrelevance) to 4 (indicating perfect relevance), associated with the vectors representing query-document pairs.

Five LtR datasets were used in the experiments presented in this thesis, and they are detailed below. Furthermore, Table 3.1 and Table 3.2 summarise the fundamental details of the datasets.

- Istella-F [56]: The Istella LETOR full dataset, abbreviated as Istella-F for simplicity, is provided by TISCALI ITALIA S.p.A., an Italian company that operates the ISTELLA Web search engine (http://www.istella.it). The dataset comprises 33,018 queries and 220 features representing each query-document pair, resulting in a total of 10,454,629 documents. The substantial number of documents in the dataset makes it particularly useful for large-scale experiments on the efficiency and scalability of Learning to Rank solutions. The dataset is provided with a split between training and test sets based on an 80%-20% scheme. However, during our experimental phase, we also needed a validation set. For this reason, we extracted the validation set from the training set, resulting in a split in train, validation, and test

sets based on a 60%-20%-20% scheme. The dataset can be downloaded from http://quickrank.isti.cnr.it/istella-dataset.

- Istella-S [111]: The Istella-S LETOR dataset, referred to as Istella-S, is the smallest within the Istella dataset family. Istella-S is a smaller sample of the Istella-F dataset. Like Istella-F, it comprises 33,018 queries and 220 features representing each query-document pair. Istella-S is made up of 3,408,630 documents generated by sampling non-relevant documents to an average of 103 docuemnts per query. It has been divided into train, validation, and test sets following a 60%-20%-20% scheme. The dataset can be downloaded from http://quickrank.isti.cnr.it/istella-dataset.

- Istella-X [115]: The Istella-X (eXtended) LETOR dataset, for the sake of brevity, named Istella-X, is the largest within the Istella dataset family, comprising 10,000 queries and 220 features representing each query-document pair. Istella-X consists of 26,791,447 documents, holding the highest proportion of non-relevant documents, accounting for 99.83% of the total. This characteristic was designed to evaluate the effectiveness of LtR models in highly unbalanced queries, such as those found on the World Wide Web. The dataset was constructed using 10,000 queries sampled from a real-world Web search engine log. For each query, up to $5,000$ results were retrieved from a collection of 44,830,467 Web documents and ranked based on a BM25F [186] scoring function. The dataset has been split into train, validation, and test sets following a 60%-20%-20% scheme. The dataset us is made available at http://quickrank.isti.cnr.it/istella-dataset.

- MSLR-30K [141]: The Microsoft Learning to Rank Datasets consists of two sets, one containing approximately 30,000 queries and the other about 10,000 queries. In the experiments conducted in this thesis, we utilised the larger one and referred to this set as MSLR-30K. This dataset is the most balanced in terms of the number of relevant documents, representing approximately half of the dataset. MSLR-30K is provided in five folds, each containing a different permutation of five splits of the dataset. In each fold, three splits are allocated for the training set, one for the validation set, and the remaining one for the test set. In the experiments presented in this thesis, we specifically used Fold 1. Each fold provides a split in training, validation, and test sets that follow the 60%-20%-20% scheme. The MSLR Web dataset is available at http://research.microsoft.com/en-us/projects/mslr.

- Yahoo! Set 1 [42]: The Yahoo! Learn to Rank Challenge version 2.0 dataset comprises two distinct datasets: Set 1 and Set 2. In the experiments

conducted in this thesis, we utilised Set 1 and referred to this set as Yahoo! Set 1. This dataset is the smallest among those used in this thesis, consisting of only 709,877 documents with an average of approximately 24 documents per query. It also contains the highest proportion of relevant documents among those mentioned here, with about 74% of the documents being relevant. The dataset has been divided into training, validation, and test sets following a 60%-20%-20% scheme. The dataset can be downloaded at https://webscope.sandbox.yahoo.com/catalog.php?datatype=c.

3.3 Summary

In this chapter, we introduced the fundamental concepts and knowledge necessary to understand our contribution to the domain of Learning to Rank. Below is a brief summary of the key concepts discussed in this chapter.

- **Information Retrieval:** In this section, we explored Information Retrieval, a research field at the intersection of computer science and information science. IR addresses the challenge of **retrieving useful information** from **cluttered** and **unstructured** vast collections of data. Finally, IR widely employs machine learning techniques to address this challenge.

- **Learning to Rank:** In this section, we provided a detailed definition of the Learning to Rank problem and elucidated its distinctions from classification and regression. Specifically, in LtR, each instance comprises a user's **query** and a set of **candidate documents** assigned to it. The aim of the model is to generate a ranking over these documents such that the most relevant ones are ranked at the top of the list.

- **Evaluation Metrics:** Evaluation metrics in the domain of Learning to Rank must take into account the order of documents within the list. Among the presented metrics, there is the **Normalized Discounted Cumulative Gain**, which assesses the contribution (**gain**) of each document based on its relevance to the query and **discounts** based on its position within the ranked list.

- **Learning Algorithms:** In this section, we introduced one of the main challenges of Learning to Rank, namely the impossibility of optimising LtR metrics directly due to their intrinsic connection with item ordering. We presented the state of the art in LtR, categorised into **pointwise**, **pairwise**, and **listwise** approaches. The state of the art covered in this section encompasses algorithms to improve **effectiveness**, **efficiency**, and **fairness** in the domain of learning to rank.

- **Benchmark Datasets:** In the last section of this chapter, we introduced five benchmark datasets used to evaluate the effectiveness and efficiency of learning-to-rank algorithms. Each dataset possesses distinct characteristics, varying in the number of documents, the proportion of relevant and non-relevant documents in the dataset, and the number of queries.

Chapter 4

Surrender on Outliers and Rank

In this chapter, we discuss the work titled "*Filtering out Outliers in Learning to Rank*", in proceedings as a full paper at the *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval*. Further details can be found in the reference [121].

Outlier data points are recognised to have a negative impact on the learning process of regression or classification models, yet their influence in the learning-to-rank scenario has not been thoroughly investigated thus far. Traditional Learning to Rank learning algorithms assume that training sets do not contain noise and outliers [60]. However, this assumption is not always valid, especially in a context where human-labeled datasets are used. Additionally, the features of query-document vectors may be insufficient to distinguish relevant documents from non-relevant ones [40].

For instance, in a pairwise context, which still constitutes a significant part of the state of the art in LtR, the presence of outliers, noise, or erroneously labelled documents leads to a quadratic growth in the number of incorrect pairs. Consequently, it becomes necessary to consider the potential presence of issues/errors during training.

This work primarily consists of two contributions. First, we discovered the presence of documents within the training set that the model consistently struggles to rank correctly. Building upon this discovery, we coined the term "consistent outliers" in LtR and provided a precise definition. Second, we designed *Surrender on Outliers and Rank* (SOUR), a learning-to-rank method that detects and removes consistent outliers before constructing an effective ranking model. Through extensive experiments, we demonstrate that removing these consistent outliers before retraining a new model leads to statistically significant improvements in terms of NDCG. Furthermore, SOUR outperforms state-of-the-art denoising and outlier detection methods.

Additionally, we present a thorough experimental analysis, investigating the

behaviour of the proposed algorithm across different query types and assessing the impact of outlier removal on the model's raw predictions. Furthermore, we substantiate the effectiveness of our algorithm by comparing it to several variants inspired by related but less close works.

The chapter is structured as follows: Section 4.1 introduces the related work specific to this study. Section 4.2 provides the definition of consistent outliers in LtR, while Section 4.3 presents our algorithm called SOUR for detecting consistent outliers and training effective models on clean datasets. In Section 4.4, we define the experimental setting and the baselines used for comparison. Moreover, in Section 4.5, we provide the main results of our contribution: the performance of SOUR in terms of NDCG compared to the baselines. Section 4.6 includes a series of detailed analyses we performed to better understand the capabilities of the proposed algorithm. Finally, Section 4.7 summarises the contents covered in the chapter and outlines some future works.

4.1 Related Work: Outliers

The presence of outliers in the context of Learning to Rank poses a significant problem, yet it has received little attention from the research community so far. This is surprising considering the potential harmfulness of outliers and the high probability of having them due to the nature of the labels used to generate the ground truth. The labels themselves, representing the relevance of instances, can vary considerably based on various subjective considerations, thus introducing a potential source of noise and ambiguity into the learning process.

Despite the lack of thorough research into robustness against outliers in the field of Learning to Rank, it is essential to examine the current state of the art in this direction. Some relevant studies provide approaches and methodologies that can serve as baselines for the solution proposed in this work. Exploring these baselines is crucial to understand better what strategies have been proposed to address this issue.

In literature, one of the most common strategies for managing outliers in machine learning is *sample selection* [104, 132], where outliers are found and treated differently than legitimate samples (removed [90], learned separately [79], reweighed [60], etc.). A sample selection strategy that removes outliers is designed in [104]. The authors introduce a sample selection algorithm named Isolation Forest, which employs a forest of decision trees to decide whether an instance is an outlier. Isolation Forest trains each decision tree in the forest on a random subset of the training set. The randomisation helps create a diversified set of decision trees that collectively form the Isolation Forest. Then, for each instance in the training set, the depth of the leaf reached in each individual tree is calculated, i.e.,

how many nodes the instance passes through before reaching the leaf. Finally, the average depth across all trees is computed. Instances with a lower average depth are deemed outliers as it indicates that they were isolated more rapidly and are potential outliers. Consequently, instances that exhibit, on average, a shorter path from the root to the leaves within the forest are classified as outliers. The Isolation Forest algorithm effectively and efficiently identifies outliers, particularly in larger datasets, because it uses random decision trees and efficient depth calculation methods.

An early notable work is Robust C4.5 [90], a variant of the C4.5 algorithm [143], designed to be robust against outliers. The algorithm iteratively trains decision trees, pruning the dataset at each phase by removing misclassified instances from the training set. This iteration continues until no classification errors occur in the training set. While applied to a classification task, this technique shares some similarities with our method but is more aggressive and specific to classification.

In [177], the authors argue that strategies based on selective sampling are sensitive to changes in noise distribution. Furthermore, the difficulty in identifying outlier instances may lead to the removal of several legitimate instances. Another approach to handling noisy labels within the dataset is robust learning. Many of these strategies attempt to define a robust loss function in the presence of noise in the training set [189, 64, 148, 75, 177]. An attempt in the LtR context was made in [40], utilising a pairwise meta-learning approach that takes a linear model as input and generates an outlier-robust model. The input model hypothesis is refined through a non-linear sigmoid function. The sigmoid meta-ranker performs a non-convex optimisation that converges to a local optimum close to the original hypothesis. The algorithm leverages the non-linearity of the sigmoid function to suppress the effect of outliers, essentially softening the large negative loss generated by incorrectly ordered pairs.

A more recent attempt at achieving noise-robust training in Learning to Rank is PeerRank [177], leveraging the PeerLoss [107] loss function. In this article, they defined an instance-level loss function ℓ_{peer} employing the PeerLoss loss function to address ranking problems. They defined an instance-level loss function for both pointwise and pairwise settings. Since, as mentioned in Section 3.2.3.1, pairwise and listwise methods are more effective than pointwise ones, this work primarily focused on their pairwise definition.

For a pairwise instance $\boldsymbol{x}_i = (d_a, d_b)$ and the pairwise label $y_j = \mathbb{1}[y_a > y_b]$, the pairwise instance-level loss function is defined as $\ell(y_i, h(\boldsymbol{x}_i))$. In their PeerRank method, the pairwise instance-level loss function ℓ_{peer} introduces synthetic noise instances by randomly selecting pairs of documents from the training set and pairing them with random labels.

The ℓ_{peer} loss function can be summarised with the following definition:

$$\ell_{peer}(y_i, h(\boldsymbol{x}_i)) = \ell(y_i, h(\boldsymbol{x}_i)) - \ell(y_j, h(\boldsymbol{x}_k)), \tag{4.1}$$

where $\boldsymbol{x}_k$ is a pair of documents randomly selected from the set of candidate documents D such that $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ and $y_j = \text{Bernoulli}[p = 0.5]$.

PeerRank minimises this loss function with respect to both the original training instances and the synthetic noisy ones. Thereby, the ℓ_{peer} loss function punishes models that correctly rank such noisy instances.

4.2 Contribution 1: Consistent Outliers in LtR

In this section, we introduce the first contribution of our work, which encompasses the discovery of consistent outliers and the formal definition we provided. Specifically, we observed that given a learning algorithm $\mathfrak{L}$ that trains a model F through an iterative process, for each iteration i, there are documents consistently ranked incorrectly by the model F_i, updated at iteration i. This suggests that the learning algorithm is not able to correct the model's error in the subsequent iterations of the training process, maintaining the error for the entire training and compromising the model's final effectiveness.

Based on this discovery, we provided a formal and precise definition to better characterise this phenomenon; we formulated the definition of *consistent positive/negative outliers*.

However, before providing the definition of consistent outliers, we need to distinguish between *positive outliers* and *negative outliers*. To do this, we recall that the most commonly used quality metrics use a cutoff threshold k to manage the users' behaviour of focusing only on the top results of the ranking. This effectively constraints the metric evaluation to the contribution of the top-k ranked documents. This allows us to define an area of interest in the top-k positions of the ranking.

Intuitively, we define *positive outliers* as those **relevant** documents that are "pushed out" of the top-k scored documents, and *negative outliers* as those **non-relevant** documents that are "pushed in" to the top-k ranks. Relevant documents are documents with a relevance label greater than zero, and non-relevant documents have a relevance label equal to zero. Below, we provide a formal definition of positive and negative outliers:

Definition 1 (Positive Outlier)**.** Given a set of candidate documents D, the ranking π produced by a ranker, and a threshold k, a document $d_i \in D$ is said to be a *Positive Outlier* if *a)* it is relevant, i.e., it has a relevance label greater than 0, *b)* it is ranked below rank k, i.e., $\pi[i] > k$ and *c)* there is at least one document with a relevance label equal to 0 within rank k.

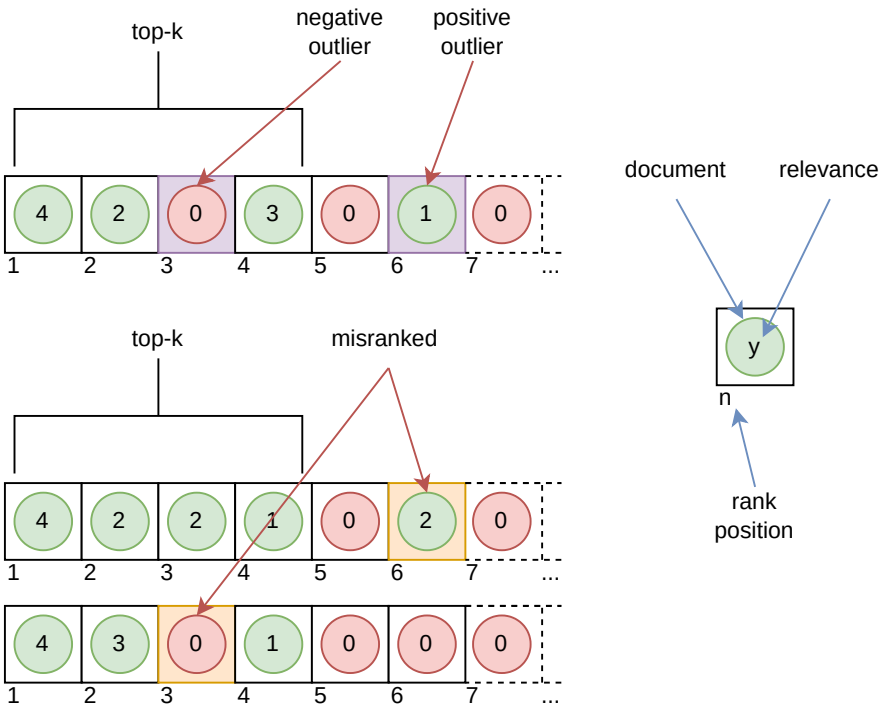


Figure 4.1: Outlier vs. misranked documents. Top: the document at rank 3 occupies a position in the top-k ($k = 4$) that might be filled by the relevant document at rank 6 and *vice versa*. Bottom: even if the ranking is incorrect, the top-k are either full of relevant documents or all possible relevant documents.

Definition 2 (Negative Outlier)**.** Given a set of candidate documents D, the ranking π produced by a ranker, and a threshold k, a document $d_i \in D$ is said to be a *Negative Outlier* if *a)* it is not relevant, i.e., with a relevance label equal to 0, *b)* it is ranked within rank k, i.e., $\pi[i] \leq k$ and *c)* there is at least one document below rank k with a relevance label greater than 0.

It is also crucial to distinguish between *outlier* and *misranked* documents. We refer to *misranked documents* for all other ranking errors that do not fit either of the above two definitions. Figure 4.1 illustrates the distinction between outlier documents (top) and misranked documents (bottom). Note that no (negative) outliers exist when the top-k positions only include relevant documents, and no (positive) outliers exist when all the relevant documents fall in the top-k positions.

As mentioned above, we discovered the presence of outliers consistently ranked wrongly during the training phase, therefore, we have to take into account all the training iterations. Thus, given a learning algorithm $\mathfrak{L}$, we are interested in every ranking produced by the intermedial model (inter-model) F_i after i iterations of the training process. Let F_i be an intermedial model learnt until iteration i, i.e.,

an ensemble of size i; we look at all the positive/negative outliers produced by F_i for each iteration of $\mathfrak{L}$ in a given iteration interval. Thus, we look for documents that are consistently considered outliers across the iteration interval, as defined below.

Definition 3 (Consistent Positive/Negative Outliers)**.** Given a list of candidate documents D, a learning algorithm $\mathfrak{L}$, two iteration-interval bounds s and e, and let F_i be the inter-model learnt until iteration i by the learning algorithm $\mathfrak{L}$. We define as *Consistent Positive/Negative Outliers* those documents that are positive/negative outliers in the ranking generated by a F_i, for every iteration $i \in [s, e]$.

We argue that a model finds it more difficult to properly rank these documents due to the outlier nature of these instances rather than the learning algorithm's limitations.

4.3 Contribution 2: SOUR Learning Algorithm

This section encapsulates the main contribution we brought with the research presented in this chapter. Here, we present our Learning to Rank algorithm, which leverages outlier removal during the training process. The main idea is to remove outlier instances before training rather than utilising the entire dataset, giving rise to our method named *Surrender on Outliers and Rank* (SOUR). The algorithm, outlined in Algorithm 3, comprises three main steps: *i)* base model training, *ii)* outlier detection and removal, and *iii)* final model training.

The SOUR algorithm takes as input three hyper-parameters: s, e, and t. The hyper-parameters s-tart and e-nd define the iterations of the learning algorithm during which the search for consistent outliers occurs. The hyper-parameter t specifies the type of outlier to remove from the training set and assumes the values $t \in \{\text{POS}, \text{NEG}, \text{ALL}\}$, representing positive, negative, or their union, respectively. A detailed explanation of how the algorithm works follows.

During the first step (line 13), an inter-model F_i is trained until iteration i on the complete training set $\mathcal{D}$ using a specified LtR learning algorithm $\mathfrak{L}$ and a metric Z. In an ideal scenario, F_i represents the optimal model achievable with $\mathfrak{L}$ at iteration i.

The second and core step of the algorithm (lines 14–23) consists of using the inter-model F_i to identify outliers in the training set. For each iteration $i \in [s, e]$, the algorithm uses the inter-model F_i to score and rank every set of candidate documents $D_j \in \mathcal{D}$, and for candidate set SOUR compute the positive and negative outliers according to Definitions 1, 2, respectively denoted with $\mathcal{O}_i^{\text{POS}}$ and $\mathcal{O}_i^{\text{NEG}}$. Then, it computes the set of consistent outliers by intersecting all the positive or negative outliers found so far, i.e., $\mathcal{O}^{\text{POS}} = \bigcap_i \mathcal{O}_i^{\text{POS}}$ and $\mathcal{O}^{\text{NEG}} = \bigcap_i \mathcal{O}_i^{\text{NEG}}$. It also

Algorithm 3 SOUR Algorithm.

1: **function** SOUR($\mathcal{D}, \mathfrak{L}, t, s, e, Z$)
2: **Input**
3: $\mathcal{D}$: training set
4: $\mathfrak{L}$: learning algorithm
5: t : type of consistent outliers to remove
6: s : iteration to start collecting outliers
7: e : iteration to end collecting outliers
8: Z : evaluation metric
9: **Output**
10: F : final model

11: $F_0 \leftarrow \emptyset$
12: **for** $i = 1$ to e **do**
13: $F_i \leftarrow \mathfrak{L}(F_{i-1}, \mathcal{D}, Z)$ ▷ Train inter-model at iteration i

14: **if** $i \geq s$ **then** ▷ Start to look for outliers after s iteration
15: $k \leftarrow Z.\text{CUTOFF}$ ▷ Get metric cutoff
16: $\mathcal{O}_i^{\text{NEG}} \leftarrow \{$not relevant documents with rank (by F_i) $\leq k$
17: above a relevant document with rank $> k\}$
18: $\mathcal{O}_i^{\text{POS}} \leftarrow \{$relevant documents with rank (by F_i) $> k$
19: below a not relevant document with rank $\leq k\}$

20: $\mathcal{O}^{\text{NEG}} \leftarrow \bigcap_{i \in [s,\ldots,e]} \mathcal{O}_i^{\text{NEG}}$ ▷ Compute consistent outliers
21: $\mathcal{O}^{\text{POS}} \leftarrow \bigcap_{i \in [s,\ldots,e]} \mathcal{O}_i^{\text{POS}}$
22: $\mathcal{O}^{\text{ALL}} \leftarrow \mathcal{O}^{\text{POS}} \cup \mathcal{O}^{\text{NEG}}$
23: $\mathcal{O} \leftarrow \textsc{Pick}(\mathcal{O}^{\text{POS}}, \mathcal{O}^{\text{NEG}}, \mathcal{O}^{\text{ALL}}, t)$ ▷ Select outliers

24: $F \leftarrow \mathfrak{L}(\emptyset, \mathcal{D} \setminus \mathcal{O}, Z)$ ▷ Train on the clean dataset
25: **return** F

considers the set $\mathcal{O}^{\text{ALL}} = \mathcal{O}^{\text{POS}} \cup \mathcal{O}^{\text{NEG}}$. Depending on $t \in \{\text{NEG}, \text{POS}, \text{ALL}\}$, the function PICK (line 23) selectes one of $\mathcal{O}^{\text{NEG}}$, $\mathcal{O}^{\text{POS}}$ or $\mathcal{O}^{\text{ALL}}$ to be the set of outliers removed by SOUR.

During the last step (line 24), a new model is trained after removing the outliers $\mathcal{O}$ from the input training set $\mathcal{D}$, and the resulting model is eventually returned.

The SOUR algorithm incurs an additional cost of running $\mathfrak{L}$ a second time. The cost of this step is primarily linked to the choice of the learning algorithm used. For instance, with tree-based learning algorithms like Gradient Boosting Decision

Table 4.1: Datasets properties.

	Istella-X	MSLR-30K	Yahoo! Set 1
#queries	10,000	31,531	29,921
avg. query length	2,679.14	119.60	23.72
#documents	26,791,447	3,771,125	709,877
relevance label distribution			
0	26,745,076	1,940,952	185,192
1	26,604	1,225,770	254,110
2	5,108	504,958	202,700
3	9,619	69,010	54,473
4	5,040	30,435	13,402
%non-relevant documents	99.83%	51.47%	26.09%

Trees, this additional cost is usually not a concern due to their inherent efficiency. It's also worth noting that the cost of outlier detection during the second step is negligible compared to the cost of running $\mathfrak{L}$. In our experimental section, we demonstrate that the size of the outlier set $\mathcal{O}$ is relatively small. Consequently, running $\mathfrak{L}$ on the cleaned dataset is nearly as efficient as with the original dataset. However, the resulting model yields notable performance improvements.

Finally, it's important to note that SOUR primarily focuses on metrics utilising a cutoff, like NDCG, ERR, Precision, etc., since they align with our defined notions of positive and negative outliers. However, SOUR can also effectively accommodate other metrics that don't rely on a threshold k by appropriately adding and tuning a dummy threshold k through the hyper-parameters.

4.4 Experimental Setup

We conduct experiments on three publicly available datasets: Istella-X, MSLR-30K, and Yahoo! Set 1, previously introduced in Section 3.2.4. To avoid jumping back to Section 3.2.4, the main information about the datasets, useful for this chapter, is summarised in Table 4.1.

We highlight that they have very different average query lengths. The limited number of documents in Yahoo! Set 1 allowed us to test the effectiveness of the proposed algorithm in extreme scenarios. All datasets include graded relevance labels with a different fraction of non-relevant documents: Yahoo! Set 1 having 26.09% out of 709,877, MSLR-30K with 51.47% out of 3,771,125 and Istella-X with 99.83% out of 26,791,447.

4.4.1 Baselines and Implementation

In the experimental phase, we compared the effectiveness of the SOUR algorithm with five different algorithms: the baselines λLGBM and λMART, two algorithms robust to outliers in the training set IsoLGBM and PeerLGBM, and an alternative variant of SOUR named SOUR*last*. All algorithms were built on top of the open-source LightGBM library [93]. The implementation of SOUR is available on GitHub[1]

Note that LightGBM stands for Light Gradient Boosting Machine, and the trained models are ensembles of decision trees trained with GBDTs. Consequently, all the models used in the experiments are tree ensembles, implying that the inter-model F_i is an ensemble trained up to iteration i containing i trees. Below are the implementation details of each of these algorithms:

- λMART: It is the exact implementation of the LambdaMART algorithm proposed in the article [27]. To train a model with the original implementation of LambdaMART, it is necessary to tell the LightGBM library not to execute gradient normalisation. Gradient normalisation is used to normalise the lambdas for different queries and improve the performance for unbalanced data.

- λLGBM: It implements the LambdaMART algorithm proposed in the article but with gradient normalisation designed in the LightGBM library.

- IsoLGBM: It is a combination of the Isolation Forest algorithm and λLGBM. Specifically, the Isolation Forest algorithm finds the outliers according to its definition and then creates a clean training set. Finally, this new training set is used by λLGBM to train the final model. Similar to what SOUR does, based on the relevance label of the outlier documents found by Isolation Forest, there are positive/negative outliers according to its outlier definition. In this case, using a hyper-parameter $t = \{\text{NEG}, \text{POS}, \text{ALL}\}$, IsoLGBM trains a model by removing negative, positive, and both kinds of outlier documents. For the isolation forest, we adopted the parameters suggested by the authors [104].

- PeerLGBM: It is a robust training algorithm that uses a data augmentation mechanism. The rationale is to inject the instance-level loss function with synthetic outlier instances so that the trained algorithm can learn to rank them poorly. We implemented the loss function in Equation 4.1 defined in [177] on top of LambdaMART. It should be noted that the PeerRank implementation proposed in the original article was done on LambdaRank,

[1]https://github.com/FedericoMarcuzzi/Filtering-out-Outliers-in-Learning-to-Rank

which shares the same objective function as LambdaMART; consequently, the implementation was straightforward.

- SOUR*last*: It is the same as SOUR, but only the outliers found in the last iteration of the training phase are removed. Trivially, it is equal to SOUR with $s = e =$*last iteration*.

4.5 Main Results

This work aimed to identify and eliminate outliers within the training set to enhance the effectiveness of learning-to-rank models. Consequently, the primary outcomes of this study revolve around the effectiveness of our solution compared to the state of the art.

4.5.1 Effectiveness

In this section, we evaluate the model's performance in terms of the average percentage of NDCG@5 and NDCG@10. It is important to note that λMART and λLGBM are not designed to be robust to outliers, whereas SOUR, SOUR*last*, PeerLGBM, and IsoLGBM each implement a different strategy to handle the presence of outliers in the training set.

Table 4.2 presents the results of each model on the test set alongside the best hyperparameters for each model. The table includes the size of each model, denoted as $|F|$, expressed in terms of the number of trees (i.e., at which iteration of GBDTs the training phase has terminated) and the hyperparameters s and t introduced earlier. Additionally, it includes a column $|\mathcal{O}|$ as the size of the removed outlier set.

Each model was trained to a maximum of 1,000 trees with early stopping criteria computed on the validation set. The best hyperparameters were determined through hyperparameter tuning on the validation set. Further detailed analysis of models' hyperparameters is discussed in Section 4.6.1.

The first conclusion that can be drawn from Table 4.2 is that SOUR outperforms other algorithms on every dataset and for both NDCG@5 and NDCG@10. As expected, the smallest improvement is achieved in the Yahoo! Set 1 dataset due to its small size. However, it is well known how small NDCG improvements are indeed very relevant [43]. Interesting gains are visible for Istella-X and MSLR-30K. The improvements brought by SOUR are statistically significant in most cases. This confirms that the proposed SOUR algorithm is effective and that further investigation is worthwhile. To measure the statistically significant improvement of our solution, we used a Fisher's randomisation test [65] with a one-

sided p-value. The statistically significant improvements with respect to SOUR are marked with *italic* * for $\mathsf{p} = 0.05$ and **bold** ** for $\mathsf{p} = 0.01$.

What the results reveal is that, for SOUR, there is no definitive best choice between consistent negative or positive outliers. The choice depends on the dataset used but remains consistent across different thresholds of the metric cutoff. However, as expected, SOUR prefers to remove consistent positive outliers from the training set of Yahoo! Set 1, given that, as mentioned in Table 4.1, it contains approximately 74% of positive documents in the dataset. However, due to the very limited number of documents per query in Yahoo! Set 1, removing a few of such a small set of documents does not provide significant benefits. Finally, a notable observation is that SOUR and SOUR*last* have opposing preferences regarding the optimal set of outliers to remove.

Regarding IsoLGBM, removing both types of outliers (positive and negative) is almost always more effective. It's important to note that Isolation Forest does not consider the relevance label when looking for outlier documents but relies solely on the documents' average path length. Consequently, removing all the outliers it identifies proves to be more effective than removing only a portion. However, the outliers found by Isolation Forest are not contextualised to the ranking problem, and therefore, legitimate documents may also be removed. This translates to lower performance compared to SOUR.

The PeerLGBM algorithm is the most robust model among the state-of-the-art competitors. However, it does not achieve statistically superior performance to SOUR. Although incorporating synthetic outliers during training makes PeerLGBM the most robust among competitors, it compromises the model's learning process by deviating it from the original data distribution, resulting in lower performance than SOUR on the test set.

The parameter s also appears to follow a non-predictable distribution and needs to be tuned for each dataset. However, even in this case, it remains similar for various metric cutoffs k for SOUR and SOUR*last*. We emphasise that the number of documents removed by SOUR is quite small; for example, only 1% of the non-relevant documents are removed from the MSLR-30K dataset. This low number of removed documents, significantly smaller than IsoLGBM, contributes to SOUR's higher effectiveness. Furthermore, the performance of SOUR*last* confirms that the effort made by SOUR in identifying consistent outliers is crucial for achieving better performance. Additionally, as seen in Table 4.2, searching for outliers only in the last training iteration leads to the removal of many documents. However, SOUR*last* allows for two interesting observations. First, the training phase is faster with significant removal of documents. For instance, when optimising NDCG@5, SOUR*last* removes approximately 42% of the documents from MSLR-30K while maintaining good performance. The second observation is that even by removing

Table 4.2: Effectiveness in terms of NDCG (percentage). Statistically significant improvements w.r.t. SOUR according to Fisher's randomisation test [65] (with a one-sided p-value) are marked with *italic** ($\mathsf{p} = 0.05$) and **bold**** ($\mathsf{p} = 0.01$).

Model	$\|F\|$	s	t	$\|\mathcal{O}\|$	%NDCG@5	$\|F\|$	s	t	$\|\mathcal{O}\|$	%NDCG@10
					Istella-X					
λMART	206				**67.11****	223				**72.72****
λLGBM	535				**72.82****	585				**77.23****
PeerLGBM	883				73.32	841				**77.49****
IsoLGBM	533		ALL	0.78%	*73.11**	745		ALL	0.02%	**77.39****
SOUR*last*	670	535	NEG	0.03%	73.38	811	585	NEG	0.08%	77.69
SOUR	617	0	POS	0.01%	73.62	939	800	POS	0.01%	78.04
					MSLR-30K					
λMART	998				**49.97****	979				**52.16****
λLGBM	802				**50.48****	775				**52.46****
PeerLGBM	1000				*50.53**	991				**52.48****
IsoLGBM	771		ALL	6.81%	**50.45****	725		ALL	5.92%	**52.44****
SOUR*last*	822	802	POS	42.87%	*50.42**	859	775	NEG	1.55%	**52.84****
SOUR	876	700	NEG	1.01%	50.81	928	700	NEG	1.01%	53.04
					Yahoo! Set 1					
λMART	975				**74.95****	829				**79.04****
λLGBM	840				*75.24**	701				*79.46**
PeerLGBM	987				75.36	662				79.50
IsoLGBM	989		NEG	6.08%	**74.88****	648		ALL	24.95%	**79.03****
SOUR*last*	997	840	NEG	0.76%	75.28	862	701	NEG	2.73%	**79.35****
SOUR	999	400	POS	23.26%	75.44	941	300	POS	18.20%	79.56

only the outliers found in the last iteration, it is possible to achieve equal or superior performance compared to the baselines and achieve the second-best result in many situations, such as in Istella-X.

Finally, the results in Table 4.2 highlight the impact of gradient normalisation, allowing λLGBM to achieve statistically superior performance to λMART.

4.6 In-Depth Analysis

In this section, we thoroughly explore different hyperparameter values to understand how they affect the model's behaviour. We then dive into a detailed analysis of how SOUR performs under various scenarios and deeply analyse the different strategies used to handle outliers in the training set. Finally, we tried to understand what provides the performance improvement of our approach, and then we

Table 4.3: SOUR hyper-parameter analysis. The average percentage of NDCG@k is computed on the ISTELLA-X validation set. The highest values are in **bold**.

		t = NEG		t = POS		t = ALL	
s	e	$\|F\|$	%NDCG@10	$\|F\|$	%NDCG@10	$\|F\|$	%NDCG@10
0	1000	940	77.02	523	76.86	724	**76.96**
100	1000	693	76.69	893	76.98	766	76.77
200	1000	870	**77.17**	451	76.59	587	76.79
300	1000	566	76.88	997	76.84	687	76.84
400	1000	537	76.85	579	76.79	824	76.84
500	1000	971	76.99	626	76.89	718	76.70
600	1000	635	76.94	566	76.67	949	76.84
700	1000	568	76.84	690	76.72	671	76.82
800	1000	624	76.80	939	***77.28***	679	76.84
900	1000	982	77.17	783	76.83	960	76.85
0	100	762	76.79	998	76.53	936	76.51
0	200	837	77.15	692	76.76	775	76.41
0	300	724	76.92	998	76.85	797	76.83
0	400	912	76.86	709	76.43	692	76.88
0	500	833	77.01	437	76.14	690	76.68
0	600	438	76.74	689	76.96	517	76.28
0	700	627	76.93	824	76.86	507	76.60
0	800	607	76.70	563	76.94	884	76.60
0	900	996	77.07	1000	77.01	973	76.98

experimentally justified our choices by comparing them against alternative variants.

4.6.1 Hyperparameter Analysis

In this section, we present the analysis for the ISTELLA-X dataset; however, similar behaviour was also observed for the other datasets. Although the other datasets exhibited analogous behaviour, they did not provide additional insights into the functioning of SOUR, so we did not include analyses on hyperparameters for those datasets. It's important to note that in this work, SOUR is trained using a GBDT learning algorithm, where each iteration of the learning algorithm corresponds to a tree. For the sake of clarity, hereinafter, we referred to the learning algorithm's iterations as "trees".

Table 4.3 includes the NDCG@10 of SOUR on the ISTELLA-X validation set

while varying the hyperparameters s (start) and e (end), and removing one of three different sets of consistent outliers at a time: $\mathcal{O}^{\text{NEG}}$, $\mathcal{O}^{\text{POS}}$, or $\mathcal{O}^{\text{ALL}}$, based on the parameter $t \in \{\text{NEG}, \text{POS}, \text{ALL}\}$.

For every set of consistent outliers identified for each (s, e) pair as presented in Table 4.3, we trained an ensemble F using a maximum of $1,000$ trees with early stopping criteria on the validation set. The resulting ensemble size is indicated in the table as $|F|$.

We report results on the two most interesting scenarios we found. In the first scenario, we looked for consistent outliers in outliers found by all the latest trees in the forest ($e = 1000$). In the second scenario, we considered consistent outlier documents that were found outliers in the initial trees ($s = 0$). Additionally, we provide results for the scenario with $s = 0$ and $e = 1000$, which involves removing each document that resulted in being an outlier in every single tree of the forest.

In general, the results exhibit close performance, but notable differences can be observed, particularly with the ALL strategy consistently yielding inferior results. On average, the most favourable outcomes are attained when $e = 1000$. This aligns with our expectations; the initial trees are still on an unstable path along the gradient descent. Consequently, the outlier documents at the initial trees might significantly differ from those identified by the final model. On the contrary, when fixing $e = 1000$ and adjusting s, we can overlook errors in the initial trees and concentrate on outlier documents identified by the final trees of the ranking algorithm.

In Figure 4.2, we reported the number of documents removed while varying the hyperparameters of SOUR. Generally, the number of outliers is notably small, ranging from approximately 2,000 to a maximum of around 12,000. Maintaining $e = 1000$ and adjusting s results in a larger number of document removals. As discussed earlier, this is primarily attributed to the significant instability of the initial trees. This instability tends to reduce the intersection among the outlier sets $\mathcal{O}_i$ found in the window $i \in [s, e]$. Hence, as the results in Table 4.3 suggest, when s is not small, not only a greater quantity but also a greater quality of the consistent outliers is expected.

Referring to Table 4.3, the optimal outcomes were observed with $t =$ POS, $s = 800$, and $e = 1000$, resulting in the removal of approximately 2,400 documents from a dataset containing over 26 million instances. This study demonstrates that removing a limited number of documents suffices to attain statistically significant improvements. In the subsequent analysis, we maintain $e = 1000$ and select the remaining parameters based on the performance observed on the validation set.

Concluding the investigation of the t hyperparameter, Figure 4.3 showcases the behaviour on the ISTELLA-X validation set for the optimal configurations of SOUR, denoted in boldface in Table 4.3. This comparison includes the baseline

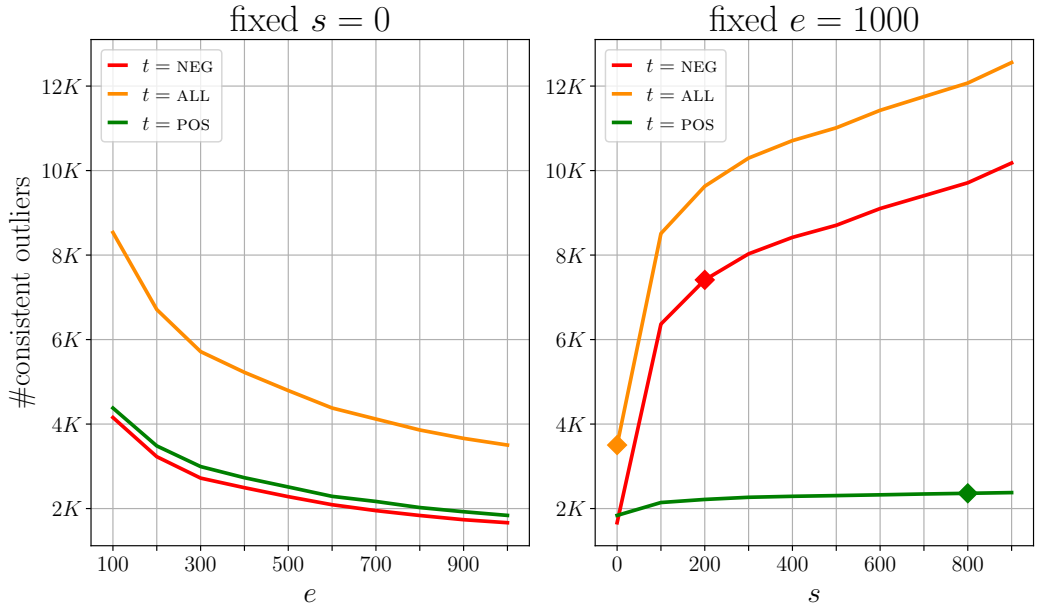


Figure 4.2: Number of consistent outlier documents removed by SOUR from ISTELLA-X dataset when varying s, e and t. Hyperparameters s and e vary with a step of 100 iterations. Diamonds are the best s values in correspondence to the highest NDCGs in Table 4.5 for each t.

λLGBM model. Notably, SOUR consistently outperforms the baseline across all t configurations. Each SOUR variant exhibits a steeper NDCG curve, achieving superior accuracy even at the earlier stage of the training process. The plot, encompassing 100 early stopping iterations, illustrates how SOUR exhibits continued accuracy improvement as additional trees are utilised compared to λLGBM.

4.6.2 Model-based vs. Data-based Outliers

There are two common methods for detecting outliers, which we refer to as data-based and model-based. The former consists of finding instances that do not fit the data distribution; in contrast, the latter strategy is to spot instances that are hard to classify correctly by a given model. SOUR falls into the second category. Figure 4.4 shows the effect of Isolation Forest [104] as a method to detect outliers. As mentioned before, we implemented a variant named IsoLGBM where an Isolation Forest is used to remove outliers from the dataset before training the final λLGBM model. As for SOUR, based on parameter t, we filter out one of the three consistent outliers sets: positive, negative, or both kinds of consistent outliers. For the Isolation Forest, we adopted the hyperparameters suggested by

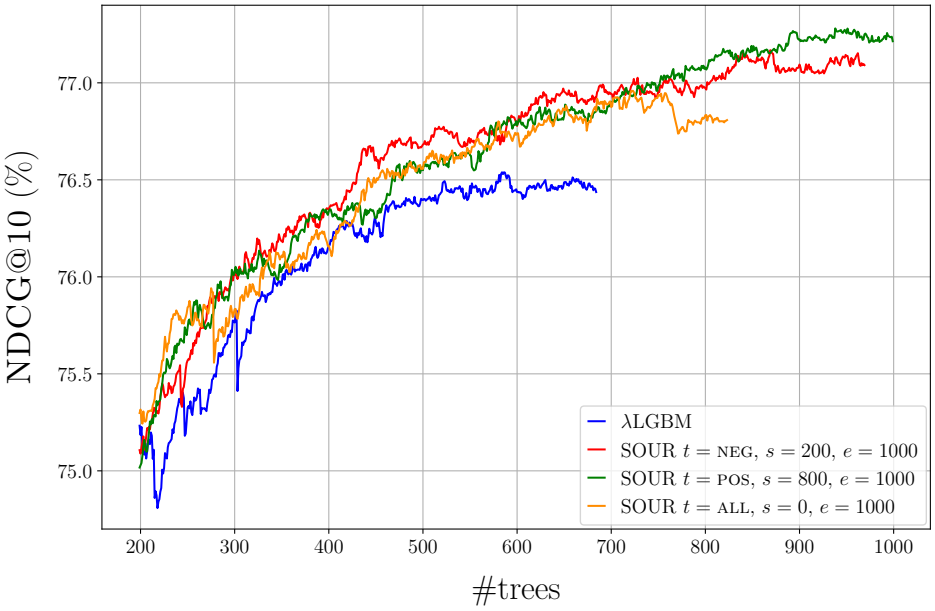


Figure 4.3: Performance in terms of NDCG@10 achieved by SOUR when varying t on Istella-X validation set.

the authors [104].

Figure 4.4 shows the performance of IsoLGBM on varying t compared to SOUR and the λLGBM baseline. The model's performance is measured in terms of NDCG@10 on the validation set of Istella-X. The NDCG curves of IsoLGBM suggest that t = ALL is the best setting for this strategy, with interesting improvements over λLGBM. However, the performance achieved by SOUR is better than those of IsoLGBM, thus allowing us to conclude that the proposed model-based approach is more effective than a data-based strategy based on Isolation Forests.

4.6.3 Data Removal vs. Data Augmentation

Amongst the models introduced in this study, PeerRank [177] is the only learning algorithm implementing robust training through a data augmentation approach. This approach enriches the training dataset with synthetic outlier instances, enabling the trained algorithm to learn to rank them properly. Consequently, we present PeerLGBM, our tailored implementation of PeerRank through the LightGBM library.

Figure 4.4 shows the performance of PeerLGBM compared to SOUR and the λLGBM baseline. As illustrated in Figure 4.5, PeerLGBM showcases superior performance compared to λLGBM, aligning closely with the behaviour observed

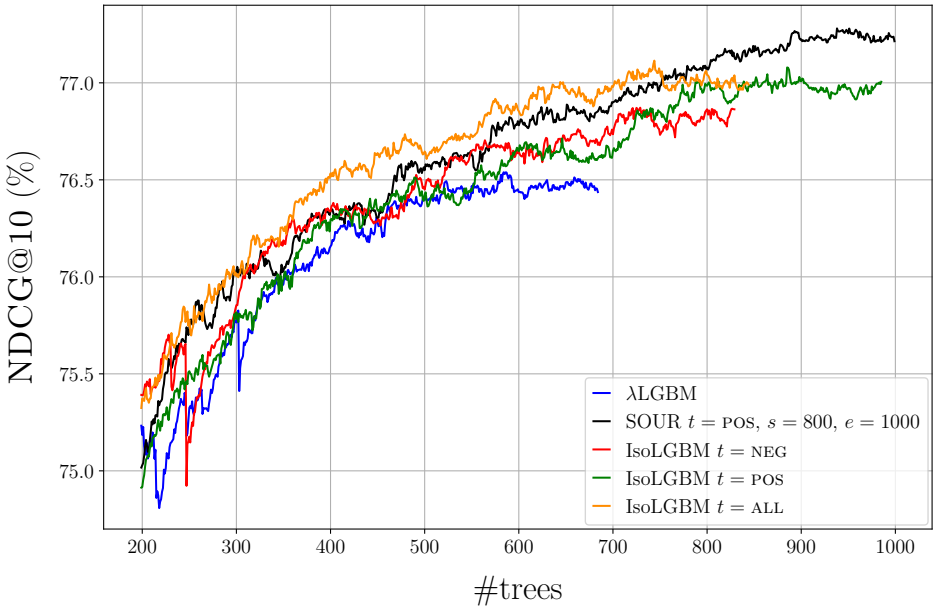


Figure 4.4: Performance in terms of NDCG@10 achieved by IsoLGBM when varying t on ISTELLA-X validation set.

in SOUR; however, SOUR retains the ability to yield better performance.

However, it is noteworthy that two completely opposing strategies (data removal for SOUR vs. data augmentation for PeerLGBM) yield performances so closely aligned, while IsoLGBM, which employs the same paradigm of data removal as SOUR, achieves lower performance. We attribute this phenomenon to the fact that the outlier search in IsoLGBM does not take into account the ranking problem, significantly impacting the quality of the outliers being removed. Conversely, PeerLGBM implements the PeerLoss function defined in Equation 4.1 directly within a learning algorithm designed for ranking (LambdaRank).

4.6.4 Per Query Class Performance

In this section, we thoroughly investigated the behaviour of SOUR and its competitors in two scenarios, comparing their performance in terms of NDCG@10. Table 4.4 presents a detailed analysis of the effectiveness of SOUR and its competitors across two distinct sets of queries.

Specifically, we divided the MSLR-30K test set into two classes. The set Q_o includes queries with at least one consistent negative outlier, while the set Q_c contains the remaining "clean" queries without consistent negative outlier. So, being Q_{test} the set of the query in the test set $\mathcal{D}_{test}$, the clean-queries set is defined

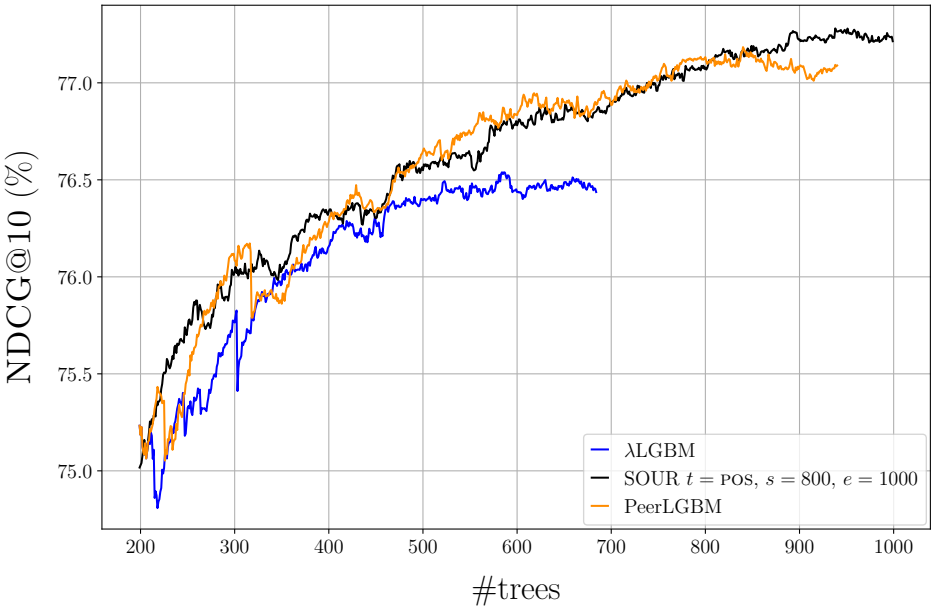


Figure 4.5: Performance in terms of NDCG@10 achieved by PeerLGBM on ISTELLA-X validation set.

as $Q_c = Q_{test} \setminus Q_o$.

We used SOUR to compute the consistent negative outlier sets on the MSLR-30K test set during the training of λLGBM. We computed a set of consistent negative outliers for different values of the hyperparameter s while keeping fixed $e = 1000$. The goal is to measure how the different algorithms improve or worsen their performance compared to λLGBM on the two different kinds of queries. Intuitively, an improvement in the queries with outliers Q_o and negligible changes in the remaining queries Q_c is expected.

For all algorithms, we use the best configuration reported in Table 4.2, except for IsoLGBM for which we use the best model trained with t = NEG that has 725 trees; to fit the experimental setting that makes use of consistent *negative* outliers.

The first row of Table 4.4 reports the results on the entire test set $Q_o \cup Q_c$ (intuitively the whole Q_{test} set), which correspond with the best results reported in Table 4.2. We observe an interesting behaviour when we vary s to consider different sets of outliers. All algorithms perform better than λLGBM on the set Q_o. Intuitively, each model deals correctly with outlier documents, and queries containing them are ranked more accurately. On the other hand, the accuracy over Q_c slightly drops for IsoLGBM and PeerLGBM. This means that these learning algorithms hinder the performance in the absence of outliers. Indeed, SOUR is the only algorithm that always improves over Q_o and Q_c. This confirms what

was mentioned earlier: the PeerLoss utilised in PeerLGBM deviates the model too much from the distribution of correct instances; consequently, it tends to overspecialise on outlier documents.

The last couple of rows of Table 4.4 includes the special case where negative outliers are computed by looking at the *best* validation iteration of λLGBM, i.e., only at the iteration $s = e = 775$. Even in this simplified setting, SOUR is the only one able to improve on both kinds of queries.

Through this analysis, we can conclude that SOUR effectively handles queries without consistent outliers. Moreover, it stands out as the only learning algorithm that can maintain its ability to provide effective ranking to queries containing consistent outliers, outperforming other learning algorithms.

4.6.5 Outliers Effect on Model Weights

To understand the source of the performance improvement provided by SOUR, we further investigated the score assigned to outlier documents and compared it with positive and negative documents both in λLGBM and in SOUR. We used the best hyperparameter reported in Table 4.2 for both models.

Table 4.5 reports the average score of the 30 leaves of the forest that are reached by the largest number of negative outliers $\mathcal{O}^{\text{NEG}}$. Recall that removing the documents in $\mathcal{O}^{\text{NEG}}$ from the training set provides the highest effectiveness for MSLR-30K). Similarly, we collected the same information for the leaves where the largest number of positive documents in the top-k (P_k) and negative documents (N) fall.

The results are very interesting and explain why SOUR can achieve such an increase in effectiveness. We can observe that in both λLGBM and SOUR, the negative outlier documents, on average, receive scores that are closer to those of the positive documents rather than the negative ones (see the last table's row: *avg.* $1 - 50$). In SOUR, the average score of negative outlier documents slightly increases, but the gap between positive documents in P_k and negative documents in N is much larger.

This unexpected phenomenon can be explained by the fact that outlier documents will always remain outliers, and the model will always struggle to rank them correctly. However, thanks to SOUR, by removing outlier documents from the training set, the model is learnt with better weights (in the case of the GBDT-based model with purer leaves). Consequently, the model can better distinguish between positive and negative documents (i.e., relevant and non-relevant). In other words, positive documents that end up in the leaves of a SOUR-trained model receive a higher score (a stronger upward push to the top positions in the ranking) than those trained with λLGBM. Conversely, negative documents receive a lower score (a stronger downward push to the bottom positions in the ranking).

Table 4.4: Effectiveness breakdown in terms of NDCG@10 (percentage) over Q_o and Q_c on the MSLR-30K test set.

s	query type	$\lvert Q_* \rvert$	λLGBM	IsoLGBM	PeerLGBM	**SOUR**
	$Q_o \cup Q_c$	6306	0.5246	0.5244	0.5248	0.5304
0	Q_o	3265	0.4555	0.4558	0.4574	0.4623
	Q_c	3041	0.5988	0.5982	0.5970	0.6036
100	Q_o	4306	0.4647	0.4658	0.4669	0.4717
	Q_c	2000	0.6537	0.6508	0.6493	0.6569
200	Q_o	4504	0.4683	0.4697	0.4706	0.4752
	Q_c	1802	0.6655	0.6614	0.6601	0.6685
300	Q_o	4618	0.4710	0.4726	0.4734	0.4779
	Q_c	1688	0.6712	0.6664	0.6651	0.6740
400	Q_o	4690	0.4725	0.4741	0.4749	0.4794
	Q_c	1616	0.6758	0.6706	0.6694	0.6786
500	Q_o	4742	0.4740	0.4755	0.4764	0.4809
	Q_c	1564	0.6780	0.6728	0.6712	0.6806
600	Q_o	4799	0.4756	0.4768	0.4779	0.4822
	Q_c	1507	0.6809	0.6761	0.6740	0.6841
700	Q_o	4851	0.4764	0.4777	0.4787	0.4830
	Q_c	1455	0.6855	0.6805	0.6783	0.6885
800	Q_o	4906	0.4780	0.4792	0.4803	0.4846
	Q_c	1400	0.6882	0.6832	0.6805	0.6912
900	Q_o	4957	0.4794	0.4806	0.4816	0.4859
	Q_c	1349	0.6908	0.6858	0.6831	0.6940
best	Q_o	5051	0.4809	0.4822	0.4833	0.4877
	Q_c	1255	0.7008	0.6945	0.6917	0.7026

Through this analysis, we can conclude that SOUR does not improve its ability to detect outliers (which is expected, as SOUR did not see them at training time), but SOUR has a better capability of distinguishing between positive and negative documents.

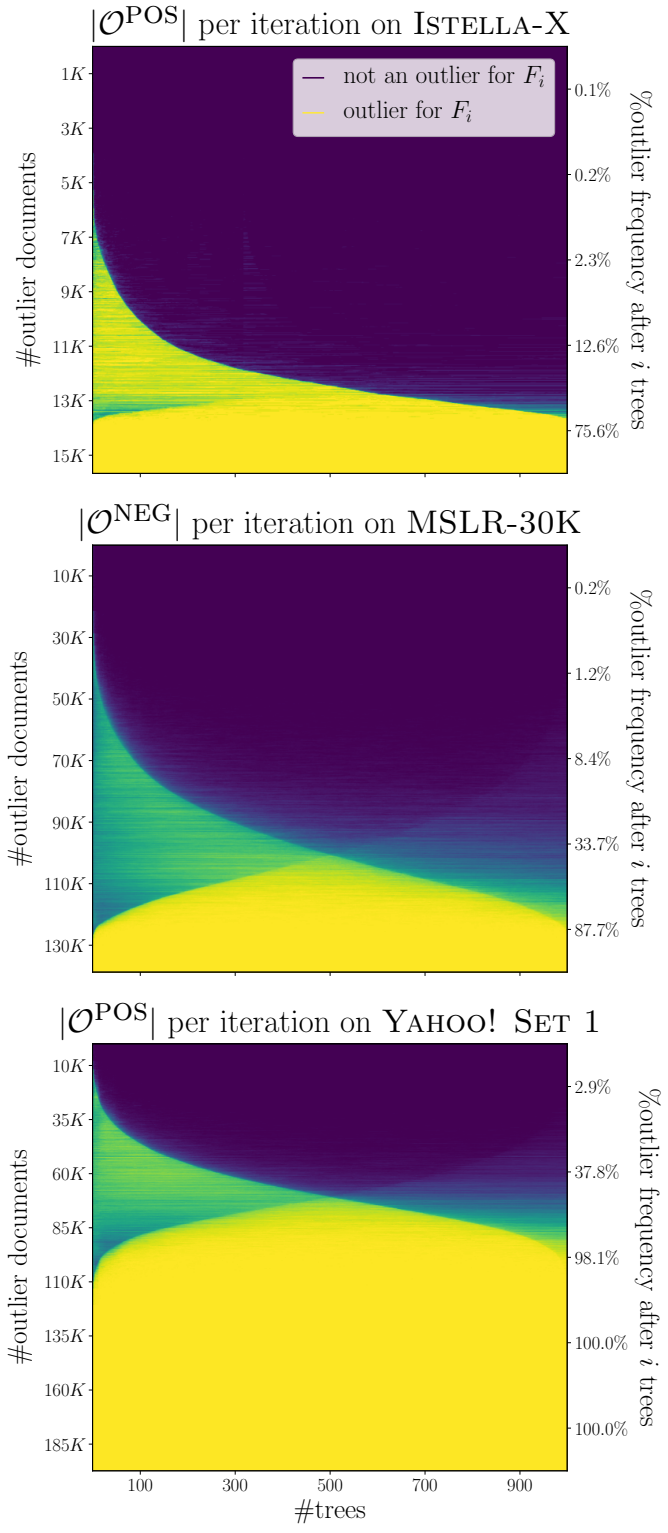


Figure 4.6: Number of outlier documents per iteration found in each training set while training with λLGBM and optimising NDCG@10. The results are limited to those documents found outliers for at least one iteration of the learning process. The documents are sorted by their outlier classification frequency.

Table 4.5: Exit leaves average score on MSLR-30K test set at different trees of the λLGBM and SOUR forests.

	λLGBM			SOUR		
tree	P_k	$\mathcal{O}^{\text{NEG}}$	N	P_k	$\mathcal{O}^{\text{NEG}}$	N
1	0.038	0.031	-0.022	0.044	0.036	-0.022
10	0.170	0.111	-0.174	0.177	0.144	-0.183
20	0.269	0.176	-0.293	0.283	0.208	-0.366
30	0.284	0.182	-0.413	0.371	0.277	-0.470
40	0.353	0.274	-0.532	0.392	0.315	-0.622
50	0.425	0.197	-0.645	0.437	0.260	-0.663
avg. 1–50	0.257	0.161	-0.358	0.280	0.207	-0.395

4.6.6 Consistent vs. Frequent Outliers

In this section, we analysed if the strict consistent property of consistent outliers defined in Definition 3 is fundamental or if it is possible to relax our definition with a *frequent* version.

To do so, we designed pSOUR, a variant of SOUR that removes frequent outliers rather than consistent outliers. In detail, the original SOUR algorithm removes from the dataset, all documents that are consistently outliers in an interval $[s, e]$ of the training iterations. Instead, pSOUR removes documents found outliers during the whole training phase with frequency larger than p%, i.e., at least a p% of the trees during training. Therefore, pSOUR is a less restrictive variant of SOUR as it does not require documents to be outliers in all the iterations between s and e.

By comparing pSOUR and SOUR, we want to answer whether it is effective to consider documents that are outliers in all interactions from the s-th to the e-th or whether it is sufficient to detect frequent outliers.

Figure 4.6 shows for each training iteration which document is considered an outlier based on λLGBM. For the sake of clarity, in Figure 4.6, we only included the documents being outliers in at least one iteration. Moreover, the y axis is sorted by the number of times a document resulted in an outlier during the entire training process. The hourglass-shaped plots show that the number of outlier documents is typically reduced iteration after iteration, except for a bunch of documents that are almost always and consistently considered outliers. Fixing the p parameter of pSOUR corresponds to tuning how many documents starting from the bottom of the plots should be removed from the training.

In Table 4.6, we summarise the performance of pSOUR in terms of NDCG@10 on the test set of MSLR-30K. For each value of p in Table 4.6, we collected the negative frequent outliers $\mathcal{O}_p^{\text{NEG}}$, then we removed $\mathcal{O}_p^{\text{NEG}}$ from the training set, and

Table 4.6: Effectiveness in terms of NDCG (percentage) achived by pSOUR with $t = \text{NEG}$, on MSLR-30K test set by varying p. The highest value is in bold.

$\|F\|$	NDCG@10	p	$\|F\|$	NDCG@10	p
676	0.5169	10%	799	0.5274	91%
869	0.5208	20%	836	0.5268	92%
994	0.5226	30%	817	0.5269	93%
865	0.5219	40%	926	0.5278	94%
754	0.5233	50%	839	0.5268	95%
896	0.5258	60%	859	0.5274	96%
964	0.5271	70%	885	0.5276	97%
750	0.5260	80%	976	**0.5289**	98%
924	0.5279	90%	932	0.5275	99%
SOUR	$t = \text{NEG}$	$s = 700$	928	0.5304	

finally we trained a model. What this analysis shows is that as p increases, the performance of pSOUR also increases up to $p = 90\%$. With small values of p, the strategy removes many negative documents, including those documents that are marked as outliers for a few iterations. Those documents may not be outliers but rather good negative documents to keep in the dataset; removing them is harmful to the model. Beyond $p = 90\%$, results are stable, with the best performance achieved at $p = 98\%$.

Finally, we can see that the highest NDCG value obtained by pSOUR in the test set (in boldface) is lower than the best performance of SOUR. Thus, although pSOUR can remove instances that are not useful for training, the SOUR strategy can produce more effective models. This highlights that removing documents consistently found as outliers is better than removing frequent outlier documents.

4.6.7 Removing vs. Exploiting Outliers

So far, we observed that consistent outliers are hard to rank correctly; therefore, we decided to remove them from the training. The following question arises: could we resort to the *curriculum learning* technique proposed by Bengio *et al.* in [9] to properly exploit those hard instances? According to curriculum learning, a delayed injection of difficult documents into the training set results in more accurate models. In this section, we provide an in-depth analysis of the possibility of exploiting the outliers through a curriculum learning strategy with a delayed injection of outliers.

To answer this question, we subdivide the consistent negative outliers found in

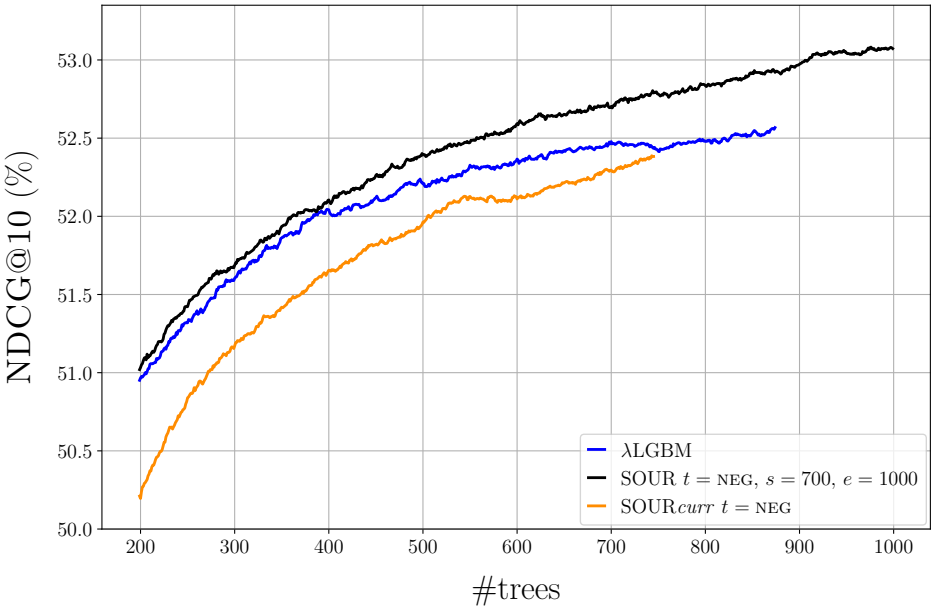


Figure 4.7: Performance in terms of NDCG@10 achieved by SOUR*curr* on MSLR-30K test set.

different forest stages into five sets: consistent negative outliers that appear only in the intervals $[0, 50]$, $[0, 200]$, $[0, 500]$, $[0, 800]$ and $[0, 900]$. During the training phase, we inject each set in the training set when the number of boosting iterations exceeds the right end of their interval. We call this strategy SOUR*curr*.

Figure 4.7 shows the effectiveness in terms of NDCG@10 of the learning algorithms on the MSLR-30K test set. The results show how using a curriculum learning strategy to gradually learn the hard instances (i.e., the consistent outliers) does not fit well in this scenario. As a result, the effectiveness of SOUR*curr* is far from SOUR and close to λLGBM only on the last iterations. We can not treat these outliers as instances of different difficulty, but they must be removed entirely from the beginning of the training phase.

4.6.8 Robustness to Outlier Frequency

In this section, we analyse the behaviour of SOUR and λLGBM on MSLR-30K dataset with different amounts of outliers. The analysis focuses on testing SOUR capability to detect and remove outliers during the training phase and how this affects the effectiveness at the prediction phase, compared to a baseline λLGBM model trained on the whole synthetically-noised dataset.

In particular, we manually added synthetic outliers in MSLR-30K training

set and validation set by flipping each document relevance label equal to 0 into a 4 with a probability $n = \{0.05, 0.075, 0.1\}$. Let $\mathcal{D}^n_{\text{TRAIN}}$ and $\mathcal{D}^n_{\text{VALID}}$ be respectively the synthetic training set and synthetic validation set, we perform training on $\mathcal{D}^n_{\text{TRAIN}}$ and early stopping on $\mathcal{D}^n_{\text{VALID}}$. Model evaluation is done on the original test set, and the results are summarised in Figure 4.8.

Since we introduce synthetic outliers with labels greater than 0, we train SOUR with $t = \text{POS}$ to remove consistent positive outliers. To study the behaviour of SOUR on different search intervals $[s, e]$, we kept fixed $e = 1000$ and varied the value of s from 0 to 900 with a step of 100.

From Figure 4.8, it can be seen that as probability n increases, the gap between the models trained with SOUR and λLGBM increases. This phenomenon appears since the first boosting interactions of the learning algorithms. All models trained with SOUR have superior performance with respect to the baseline. In particular, SOUR with $t = \text{POS}$, $s = 0$ and $e = 1000$, achieves the higler effectiveness. In this setting, SOUR removes documents detected as outliers in every training iteration, suggesting that most synthetic outliers are always considered outliers from the very beginning of the learning process.

Furthermore, due to the presence of synthetic positive outliers, using smaller $[s, e]$ intervals leads to the removal of good positive documents that are being pushed below the cutoff k for a sufficient number of iterations.

For the sake of truth, SOUR is not able to detect and remove all the synthetic outliers. In fact, as n increases, both SOUR and λLGBM lose part of their effectiveness.

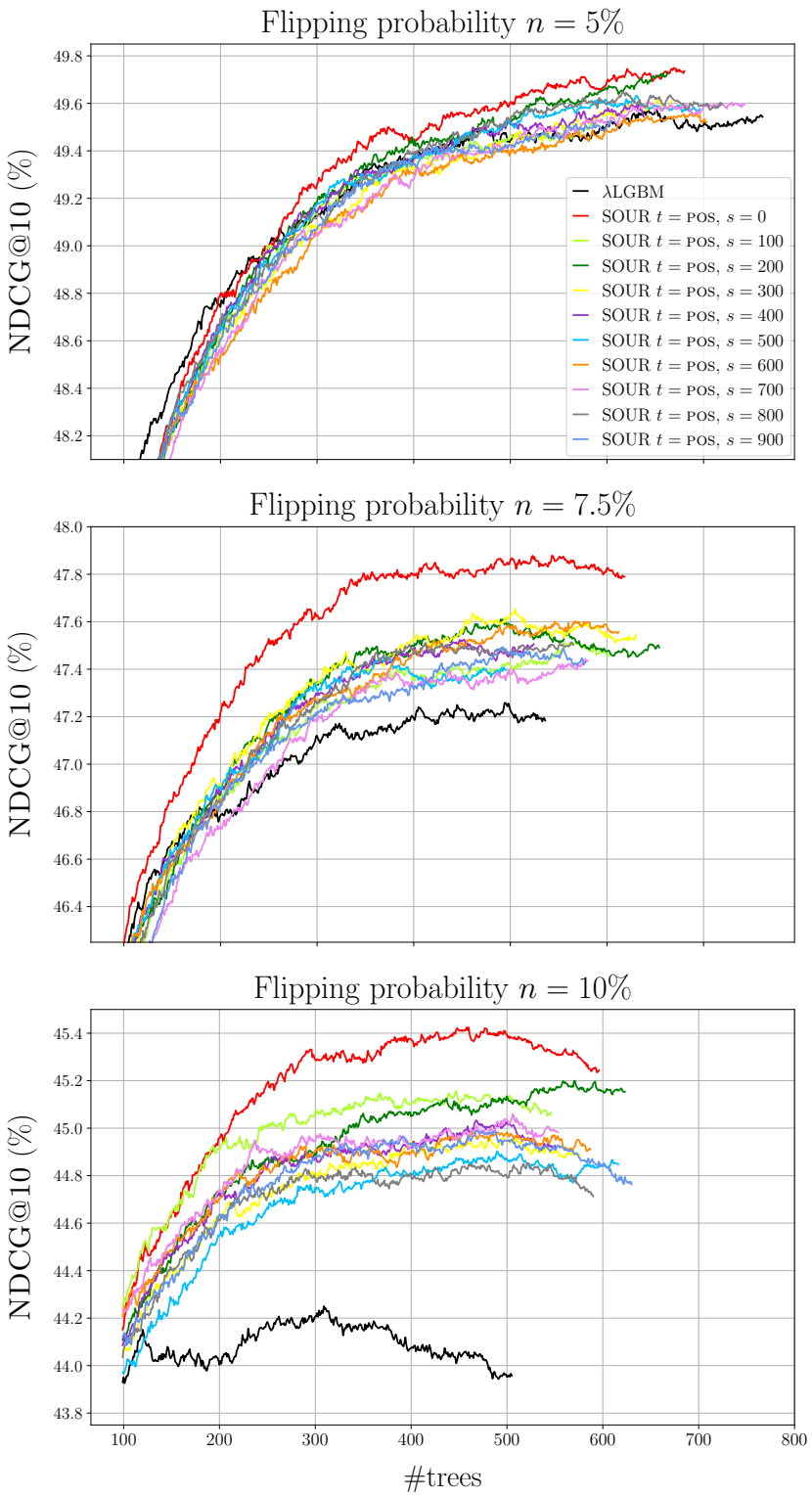


Figure 4.8: Performance in terms of NDCG@10 achieved by SOUR and λLGBM on MSLR-30K test set, by varying the flipping probability n of changing a relevance label from 0 to 4 in the training and validation set.

4.7 Summary

This last section of the Chapter, "Surrender on Outliers and Rank" summarises the main contribution of this work, its results, and potential future directions starting from this research.

- **Consistent Outliers:** In this work, we discovered and defined consistent positive/negative outliers in Learning to Rank. These outliers are documents consistently ranked incorrectly during a specific training window.
- **Contribution:** The main contribution of this work is the design of a new sample selection strategy named *Surrender on Outliers and Rank* (SOUR) for outlier detection and removal of consistent positive/negative outliers found in multiple iterations of the learning algorithm. SOUR identify and remove outliers/noise/errors in the training set to ensure higher data quality for model learning. This strategy ultimately aims to enhance the model's effectiveness during the operational phase.
- **Main Results:** SOUR aims to remove outlier documents from the training set that can be detrimental to the model learning phase, thus producing more effective models. Through three publicly available datasets, we demonstrated how SOUR achieves a statistically significant increase in effectiveness compared to the state of the art.
- **In-Depth Analysis:** Through in-depth analyses, we investigated various aspects of SOUR. We found that incorporating ranking information within the definition of outliers to look for provides higher data quality and, thus, more effective models (Section 4.6.2). Removing consistent outliers from the training set is more effective than using outlier data augmentation strategies. This is because a data augmentation strategy deviates the model from the original data distribution it might encounter during the operational phase (Section 4.6.3). We analysed SOUR 's behaviour in two queries (with or without consistent outliers). We observed that models trained by SOUR are the only ones to achieve both higher effectiveness on instances with outliers and not to reduce performance on queries with legitimate documents (Section 4.6.4). Training with SOUR produces models with cleaner weights (in this case, purer leaves), significantly impacting the model's ability to distinguish between legitimate positive and negative documents (Section 4.6.5). We demonstrated the constraint of removing consistent outliers rather than frequent outliers provides higher effectiveness in the trained model and, thus, a higher data quality in the training set (Section 4.6.6). We showed that consistent outliers cannot be considered difficult instances to use in a curriculum

learning setting, but rather, it is better to remove them permanently from the training set (Section 4.6.7). Lastly, we demonstrated how SOUR is more robust than the baselines in the presence of strongly noised training set scenarios (Section 4.6.8).

4.7.1 Future Work

Building upon the insights gained from SOUR, potential future work could revolve around enhancing and refining the existing approach. Here are some directions for future research:

- **Integrating Domain Knowledge:** It would be worth exploring new methods to integrate a finer domain-specific knowledge into SOUR. More knowledge could lead to more refined outlier identification, aligning better with the specific context or objectives of the application, e.g., enriching the definition of consistent outliers with documents' predicted score and feature vector.
- **Algorithm Combination:** An interesting direction is to exploit the outliers removal part of SOUR as a data-cleaning strategy for various learning algorithms. For example, combining SOUR with PeerLGBM to remove documents that the model consistently fails to rank correctly, then train models robust to other types of noise on a consistent-outliers free training set.
- **In Training SOUR:** Another direction for this research involves dynamic identification and removal of consistent outliers during the training process. This approach could enhance SOUR's efficiency by circumventing the need for model retraining on clean data.
- **Adaptive Hyperparameter:** Develop techniques to dynamically adapt the hyperparameters of SOUR during the training process. An adaptive approach could ensure optimal outlier removal and improve the efficiency of the training process.
- **SOUR for Other Tasks:** Extend the application of SOUR to tasks beyond learning to rank. Investigate how the concepts and methodologies in SOUR can be adapted and applied to outlier detection in different domains, such as classification or regression tasks.
- **Outlier Visualisation and Interpretability:** Develop techniques to visualise and interpret the outliers identified by SOUR. Providing clear and interpretable insights into the nature and impact of outliers can aid practitioners in understanding the model's behaviour and making informed decisions.

Chapter 5

On the Effect of Low-Ranked Documents

In this Chapter, we illustrate the work titled "*On the Effect of Low-Ranked Documents: A New Sampling Function for Selective Gradient Boosting*", in proceedings as a full paper at the *SAC '23: The 2023 ACM SIGAPP Symposium on Applied Computing*. Further details can be found in the reference [110].

As mentioned in Section 3.2, the primary application of Learning to Rank is in Web Search, where, given a user query, the model returns a list of documents ordered by relevance to the query. The LtR algorithms aim to optimise a ranking metric such as NDCG or ERR by assigning scores to each query-document vector based on their relevance by training on gold standard datasets. These datasets contain up to thousands of queries and millions of documents. They aim to provide the learning algorithm with positive documents of different relevance grades and negative documents non-relevant to the user's needs. However, not all documents within these datasets are useful examples, and including them in the training set can compromise the learning phase and the generalisation of the resulting model. Moreover, large training sets increase training time, a significant concern in contexts like online learning.

Current research trends in Learning to Rank are predominantly focused on improving model effectiveness by refining objective functions to better approximate ranking metrics or designing more efficient models through cache-aware algorithms and model-pruning strategies. However, limited attention has been given to improving the quality of the training set, and in particular, to understanding which documents are useful during the learning phase. Document selection strategies to enhance model effectiveness remain underexplored.

To address this gap, the main contribution of this work lies in the definition of a new document selection strategy called HIGH_LOW_SAMPL, for Selective Gradient Boosting (SelGB) framework. SelGB is a framework that makes use of GBDTs to

generate decision tree forests to solve LtR tasks. This framework uses a document selection strategy called SEL_SAMPL to select the most informative negative documents from the training set to learn models from a smaller but more informative subset of examples.

Our newly devised selection strategy focuses on selecting a percentage of documents likely to be misranked by the model and a percentage of documents that the model ranks perfectly. So, it extracts only the most useful documents from the training set to achieve a higher data quality and a faster training process due to a smaller training set size. This strategy is crucial in minimising the probability of misranking between positive and negative documents and preventing the model from overfitting difficult examples while allowing generalisation over simpler ones.

In detail, we designed HIGH_LOW_SAMPL, a new document selection strategy built on top of SelGB framework, but easily adaptable to other learning algorithms. This strategy allows SelGB to select from the training set: *i)* all the positive documents. *ii)* the most informative negative documents, i.e., the non-relevant documents ranked highest in the ranking, to highlight the differences between the positive ones. *iii)* the less informative negative documents, i.e., the non-relevant documents ranked lowest in the ranking, to avoid the model from overfitting on difficult examples and achieving poor generalisation on unseen documents. We perform an extensive experimental evaluation to show that SelGB combined with our HIGH_LOW_SAMPL outperforms its previous version and the baseline LambdaMART algorithm. Moreover, we prove how SelGB equipped with HIGH_LOW_SAMPL obtains a speed-up in the training process compared to LambdaMART without compromising the model effectiveness. We show how the lowest-ranked negative documents selected by HIGH_LOW_SAMPL allow the models to achieve a higher stability and a lower variance. Finally, the reduction in training time further underscores the practical benefits of our approach in large-scale applications, especially in scenarios like online learning where efficiency is crucial.

The remainder of this Chapter is structured as follows: Section 5.1 delves into the related work pertinent to this study. Section 5.2 introduces Selective Gradient Boosting, the framework for which our new selection strategy has been designed. Moreover, in Section 5.3, we present our novel document selection strategy named HIGH_LOW_SAMPL, developed for the Selective Gradient Boosting framework. Section 5.4 defines the experimental setup and the baselines employed for comparison. Furthermore, Section 5.5 provides the primary outcomes of our research, focusing on performance improvement in terms of NDCG (effectiveness) and training time duration (efficiency). Section 5.6 provides an in-depth analysis conducted to gain a comprehensive understanding of the capabilities of Selective Gradient Boosting when equipped with our selection strategy HIGH_LOW_SAMPL. Finally, Section 5.7 encapsulates the Chapter's content and outlines potential fu-

ture research directions.

5.1 Related Work: Sampling Strategies

As mentioned earlier, this work is an improvement over the work proposed by Lucchese *et al.* in [115]. Their work introduced a framework called SelGB, based on the gradient-boosted algorithm. It can be considered state-of-the-art in LtR algorithms for the re-ranking stage of the IR pipeline in a highly unbalanced scenario. SelGB implements a dynamic document selection strategy called SEL_SAMPL that, at each stage of the gradient boosting, aims to deal with highly unbalanced datasets. Since this algorithm is strictly related to our contribution, we provide a detailed introduction in the subsequent section. Note that SelGB uses GBDTs, so the related work in this Chapter mainly focuses on learning algorithms that employ them.

Gradient One-Sided Sampling (GOSS) is a technique utilised in Gradient Boosting to enhance the efficiency of the gradient-boosting model by reducing computational costs and memory consumption during the training phase. This is achieved by selectively subsampling the documents in the dataset, focusing on those with larger gradients, as they contribute more to the learning process. GOSS operates in two stages: first, it identifies and retains documents with gradients exceeding a predefined threshold, ensuring crucial data points are not discarded. Subsequently, it randomly samples a portion of the remaining documents with smaller gradients, effectively preserving the overall data distribution. This two-stage approach allows the model to be trained on a reduced dataset while preserving the informative documents crucial for accurate model training. These two steps allow GOSS to facilitate expedited training, making gradient boosting more scalable and efficient for handling massive datasets.

Another well-known strategy is Stochastic Gradient Boosting (SGB) [70]. SGB is a prominent machine learning technique widely employed in predictive modelling and regression tasks. It can be considered as a combination of Random Forest (Section 2.2.1) and Gradient Boosting Decision Trees (Section 2.2.2). As GBDT, Stochastic Gradient Boosting iteratively constructs an ensemble comprising multiple weak learners, typically decision trees, through an adaptive boosting process. Each tree is trained to predict the residual errors of the previous tree, refining the overall prediction with each subsequent iteration. What sets SGB apart from GBDT is the utilisation of stochasticity for both features and training instances sampling. In the sampling stage, a random subset of the training data is chosen for training each tree, introducing diversity into the models and reducing overfitting risks. Moreover, a random subset of features is considered for determining the best split at each tree node. This stochasticity enhances the model's robustness and

generalisation by reducing the correlation between trees and allowing for improved feature space exploration.

It is worth mentioning that Lucchese *et al.* in [115] has widely proven SelGB to be more effective than related sampling strategies such as GOSS and SGB.

Different works propose selection strategies that statically pick a set of informative documents to feed the training process [5, 86]. Among these selection strategies, there are *undersampling techniques* that play a vital role in addressing class unbalance issues within training datasets, especially in large-scale LtR applications, where the number of relevant and non-relevant instances can significantly differ. Undersampling techniques aim to mitigate this unbalance by reducing the number of instances from the overrepresented class (often non-relevant instances). This rebalancing enhances the model's ability to learn from both classes more equitably, resulting in a more accurate and unbiased prediction.

Finally, negative sampling is now a common data augmentation approach for improving the robustness [107], for dealing with scarcity of negative examples [171], or for managing unlabelled instances in a semi-supervised scenario [47], and many other tasks. This is not the case in this work as we do not propose a data generation method but a novel strategy to select existing non-relevant documents among the pool of available examples in the given dataset.

5.2 Selective Gradient Boosting

Selective Gradient Boosting (SelGB) is a framework developed by Lucchese *et al.* [115] to learn effective models from a highly unbalanced training set in the LtR re-ranking stage. SelGB implements a dynamic document selection strategy called SEL_SAMPL that, at each iteration of gradient boosting, is aimed at dealing with highly unbalanced datasets. Only the positive and the currently highest-ranked negative documents are considered for growing the next trees of the gradient-boosted forest. The authors showed how this selection strategy selects only useful examples from a very unbalanced set of documents (with a prevalence of negative examples), resulting in more effective models and reduced training time due to a reduced number of documents being processed.

To better position our contribution within this context, we must distinguish the SelGB framework from its selection strategy, denoted as SEL_SAMPL. In Algorithm 4, we present the pseudo-code for the Selective Gradient Boosting framework, utilising a generic selection strategy referred to as SAMPL. The function NEXTGRADIENTBOOSTEDDECISIONTREEITERATION trains and adds the next gradient-boosted decision trees to a given ensemble.

The algorithm takes as input a training set $\mathcal{D}$, the maximum size of the ensemble N, the hyperparameter n that manages the number of learning algorithm

iterations between two calls of the selection strategy, and the selection strategy SAMP that perform the selection of a subset of the candidates documents D_i for each query q_i. The algorithm iterates over a fixed number of iterations N and, after every n iterations, creates a new training set $\mathcal{D}^*$ through the document selection function SAMP given as input (line 13). The new dataset $\mathcal{D}^*$ is used to train the model in subsequent n iterations (line 14).

From the above, it can be deduced that the core of SelGB is the document selection function; different selection faction provides completely different effective learned models.

The selection function SEL_SAMPL takes as input the training set $\mathcal{D}$, the ensemble F_m, learnt until iteration m, and the values of the hyperparameter $p \in [0, 1]$. Given a query $q_i \in \mathcal{D}$ and its set of candidates documents D_i, let $D_i^- \subseteq D_i$ the set of negative (non-relevant) documents in the candidate set and $D_i^+ \subseteq D_i$ the set of positive (relevant) documents in the candidates set, where $D_i = D_i^+ \cup D_i^-$ and $D_i^+ \cap D_i^- = \emptyset$. For every query q_i, SEL_SAMPL creates a new set of candidates document D_i^* by selecting all the documents within the positive set D_i^+ and by selecting from D_i^- the $p\% \cdot |D_i^-|$ documents with the highest score estimated by the current model F_m. As a result, the final cardinality of the newly candidate set D_i^* for the query q_i is $|D_i^*| = |D_i^+| + p \cdot |D_i^-|$.

Finally, by combining all the D_i^*, the new training set $\mathcal{D}^*$ contains only the documents $\bigcup_{q_i \in \mathcal{D}} D_i^*$. The newly created training set $\mathcal{D}^*$ is used to perform the subsequent n iterations of the learning algorithm, and then a new selection of documents is performed.

The $p\%$ of the highest-ranked negative documents selected by SEL_SAMPL are the most informative negative examples [115]. These documents possess the highest scores among negative documents, placing them closer to positive documents than closer to the top-k positions in the ranking, which are more attractive to users. As a result, these documents are essential examples for learning effective rankers that push these documents away from the top positions and from the positive documents. On the other hand, the remaining $(1 - p)\%$ of negative documents have lower scores, making them relatively "easier" for the model to rank. Hence, SEL_SAMPL discards them from the training set as they are not informative enough.

Finally, the pseudo-code in Algorithm 4 is designed to work on GBDTs but can be easily generalised to handle other learning algorithms. Therefore, any learning algorithm that processes instances iteratively during the training phase can be applied (e.g., boosting iteration in GBDTs and epoch in NNs).

Algorithm 4 Selective Gradient Boosting

function SELECTIVEGRADIENTBOOSTING($\mathcal{D}$, N, n, SAMPL)
  Input
    $\mathcal{D}$: training dataset
    N : ensemble size
    n : # iterations between consecutive selection steps
    SAMPL : document selection strategy
  Output
    F_N : trained ensemble

  $F_0 \leftarrow \emptyset$
  $\mathcal{D}^* \leftarrow \mathcal{D}$
  for $m = 1$ to N **do**
    if (m mod n) $= 0$ **then**
      $\mathcal{D}^* \leftarrow$ SAMPL($\mathcal{D}$, F_m)
    $F_m \leftarrow$ NEXTGRADIENTBOOSTEDDECISIONTREEITERATION($F_{m-1}, \mathcal{D}^*$)
  return F_m

5.3 Contribution: A New Sampling Function

To prove the performance of SelGB framework and SEL_SAMPL selection strategy, Lucchese *et al.* released alongside SelGB the ISTELLA-X dataset; the most extensive public LtR dataset ever released in terms of the number of documents per query, with 99.83% negative examples. However, as we show in the experimental section, the algorithm does not bring significant improvements in smaller and less negatively unbalanced datasets like MSLR-30K, or it may even perform worse than well-known baselines like LambdaMART as in more positively unbalanced datasets like YAHOO! SET 1.

In this work, we designed a new document selection strategy that overcomes the limitation of SEL_SAMPL by improving the selection of the most informative negative documents to learn more effective models from smaller and higher-quality training sets. The main contribution of this work is the definition of a new document selection function called HIGH_LOW_SAMPL that can be given as input to SelGB. HIGH_LOW_SAMPL shares many similarities with SEL_SAMPL, but despite the last, it does not discard all the lowest-scored (lowest-ranked) negative examples. Our document selection function takes as input the training set $\mathcal{D}$, the ensemble F_m, learnt until iteration m, and the values of the two hyperparameters $p_1 \in [0, 1]$ and $p_2 \in [0, (1 - p_1))$.

In detail, for each query, q_i HIGH_LOW_SAMPL creates D_i^* by selecting from D_i all the positive examples in D_i^+ and a subset of the negative examples in D_i^-.

The main difference with SEL_SAMPL is how the negative examples are selected from D_i^-. SEL_SAMPL selects from D_i^- the $p \cdot |D_i^-|$ examples with the highest score estimated by the current model F_m. Instead HIGH_LOW_SAMPL selects the $p_1 \cdot |D_i^-|$ examples with the highest score and the $p_2 \cdot |D_i^-|$ examples with the lowest score. Even in this case, the current model F_m estimates the score for each candidate document. Trivially, if $p_1 = p$ and $p_2 = 0\%$, the two selection strategies, HIGH_LOW_SAMPL and SEL_SAMPL, coincide. The final cardinality of the subset D_i^* is $|D_i^*| = |D_i^+| + (p_1 + p_2) \cdot |D_i^-|$. Figure 5.1 graphically depicts how HIGH_LOW_SAMPL creates the new set of candidates document D_i^* for a query q_i.

Finally, each D_i^* set is combined to obtain the new training set $\mathcal{D}^*$ for the subsequent n iterations.

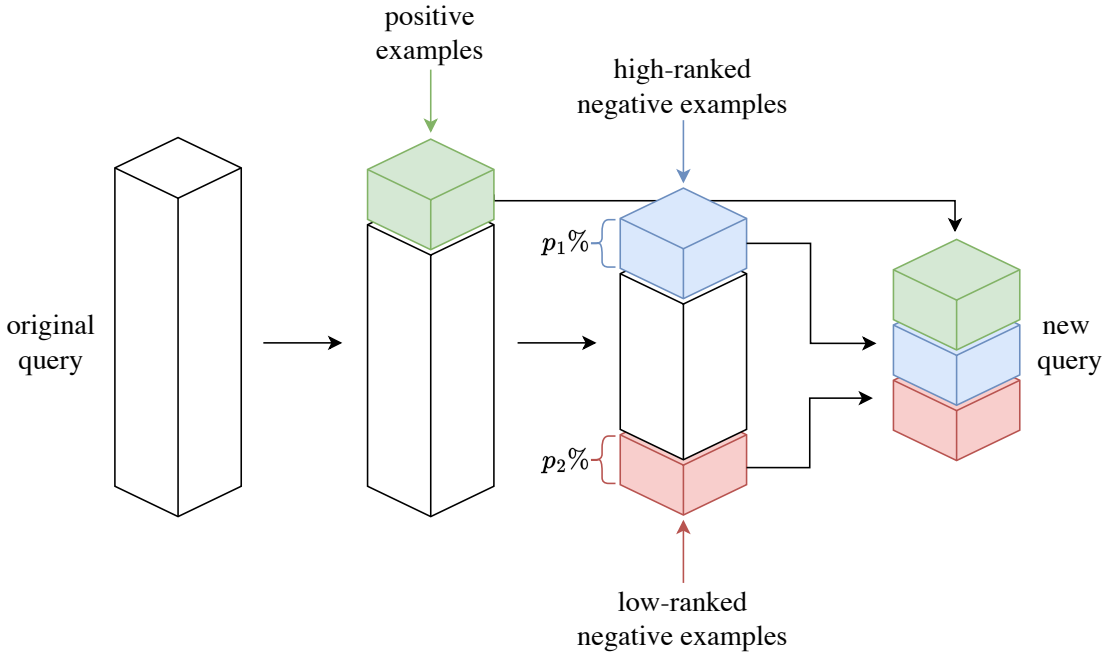


Figure 5.1: The figure shows how HIGH_LOW_SAMPL composes the new set of candidates document D^* for the query q. Given the set D ordered in descending order of (relevance label, predicted score), HIGH_LOW_SAMPL selects from D three document classes: the positive examples (in green), the highest-ranked negative examples (in blue), and the lowest-ranked negative examples (in red).

SEL_SAMPL selects the highest-ranked negative examples to enhance the discrimination between relevant and non-relevant documents and to minimise misranking risk. Nevertheless, avoiding the model to see the lowest-ranked negative documents during training can lead to overfitting the highest-ranked negative documents and lead to models with reduced generalisation capability. In

HIGH_LOW_SAMPL, we extended this idea by introducing negative documents with the lowest scores to balance the attention of the learning algorithm and prevent it from focusing only on the highest-ranked documents.

5.4 Experimental Setup

We performed our experiments on the three publicly available datasets ISTELLA-X, MSLR-30K, and YAHOO! SET 1 summarised for the sake of simplicity in Table 5.1. For each dataset, documents are labelled with graded relevance labels ranging from 0 to 4, where 0 refers to negative documents and a value greater than 1 refers to positive documents. All datasets have a different percentage of negative examples: ISTELLA-X with 99.81% out of 26,791,447, MSLR-30K with 51.47% out of 3,771,125 and YAHOO! SET 1 with 26.09% out of 709,877.

Table 5.1: Datasets properties.

Dataset	#queries	#doc.	$\overline{\text{query len.}}$	%non-rel.
ISTELLA-X	10,000	26,791,447	2,679.14	99.83
YAHOO! SET 1	29,921	709,877	23.73	26.09
MSLR-30K	31,531	3,771,125	119.60	51.47

5.4.1 Baselines and Implementation

In the experimental phase, we conducted a comparative analysis of the effectiveness of SelGB equipped with our document selection strategy HIGH_LOW_SAMPL against two distinct models: the baseline LambdaMART and the original definition of the SelGB framework that makes use of SEL_SAMPL selection strategy. The implementation of SELGB$_{HL}$ is available on GitHub[1]

Below, we outline the implementation specifics for each of these algorithms. Note that all the algorithms were developed using the open-source LightGBM library [93].

- LambdaMART: This implementation adheres faithfully to the LambdaMART algorithm as presented in the original publication [27], but incorporating gradient normalisation as designed in the LightGBM library.

- SELGB$_S$: This is an implementation of the Selective Gradient Boosting framework exactly as proposed in [115]; i.e., the one using SEL_SAMPL.

[1]https://github.com/FedericoMarcuzzi/On-the-Effect-of-Low-Ranked-Documents

- SELGB$_{HL}$: This implementation represents SelGB enhanced with our document selection strategy HIGH_LOW_SAMPL.

We applied the gradient normalisation implemented in the LightGBM library for all the learning strategies. Gradient normalisation is crucial for normalising lambdas across various queries, thereby enhancing performance for unbalanced data. It's worth noting that in the original work by Lucchese *et al.*, gradient normalisation was not used. Consequently, the performances reported in this study differ from those presented in [115].

5.5 Main Results

This work aims to design a new document selection strategy for SelGB to construct higher-quality training sets, aiming for more effective models in the operational phase. However, this work's contribution extends to enhancing the efficiency of the training process for these more effective models. Consequently, the main results of this work are divided into two sections: the effectiveness of the models during the operational phase and the efficiency during the training phase.

In the experiments, we compare our solution SELGB$_{HL}$ (i.e., SelGB equipped with our selection strategy, HIGH_LOW_SAMPL) with the baseline LambdaRank, and the state-of-the-art SELGB$_S$ (i.e., the original SelGB equipped with the selection strategy, SEL_SAMPL).

5.5.1 Effectiveness

Table 5.2 summarises the performance in terms of NDCG@10 achieved by each learning algorithm on the dataset test set splits. The optimal hyperparameters were selected through model selection based on the models' performance attained on the validation sets. A detailed analysis of the selection of the best hyperparameters p for SEL_SAMPL and p_1 and p_2 for HIGH_LOW_SAMPL is in Section 5.6.1. Additionally, to demonstrate that the higher quality of the training set produced by our selection strategy leads to more effective models even in the initial learning iterations, we also included NDCG@10 values obtained on the test set after 150, 350, and 550 iterations. The term "Full" refers to the best number of trees in which the training concluded due to the early stopping criterion; this value may vary for each model. Note that where the value N/A is reported, the training terminated early, resulting in an ensemble size smaller than the number of trees reported in the table.

We used a Fisher's randomisation test [65] with a one-sided p-value ($\mathsf{p} = 0.01$) to measure the statistical significance improvement, brought by HIGH_LOW_SAMPL

Table 5.2: Effectiveness in terms of NDCG (percentage). Statistically significant improvements w.r.t. SELGB$_{HL}$ according to Fisher's randomisation test [65] (with a one-sided p-value) are marked with **bold** ** ($\mathsf{p} = 0.01$).

Algorithm	$\lvert F\rvert$ 150	350	550	Full
		MSLR-30K		
LambdaMART	**50.38** **	**51.86** **	**52.31** **	**52.46** **
SELGB$_S$, $p = 40\%$	50.89	51.97	**52.31** **	**52.62** **
SELGB$_{HL}$, $p_1 = 20\%$, $p_2 = 40\%$	50.98	52.09	52.59	52.97
		YAHOO! SET 1		
LambdaMART	**78.04** **	78.89	79.29	**79.46** **
SELGB$_S$, $p = 30\%$	78.19	78.92	**79.15** **	**79.37** **
SELGB$_{HL}$, $p_1 = 40\%$ $p_2 = 30\%$	78.29	78.96	79.25	79.58
		ISTELLA-X		
LambdaMART	**75.01** **	**76.64** **	77.22	**77.23** **
SELGB$_S$, $p = 1\%$	77.32	78.58	N/A	78.65
SELGB$_{HL}$, $p_1 = 1\%$ $p_2 = 2\%$	77.15	78.39	N/A	78.47

to SelGB compared to the competitors. In Table 5.2, when SELGB$_{HL}$ achieves statistically superior performance to a competitor, we denote it with **bold** ** alongside the performance of the weaker model.

The results reveal that the dataset characteristics heavily influence the effect of the employed selection strategies. Indeed, we observe distinct behaviours for each of the three datasets used. Let's examine the conclusions that can be drawn for each dataset individually.

In MSLR-30K, the model trained with our approach, SELGB$_{HL}$ with $p_1 = 20\%$ and $p_2 = 40\%$, emerges as the overall best model, showcasing a statistically significant improvement over both the baseline LambdaMART and SELGB$_S$. SELGB$_{HL}$ proves to be statistically superior to LambdaMART in all ensemble sizes reported in table 5.2 and also outperforms SELGB$_S$ in the final part of the training process. It is worth noting that it never achieved an inferior performance to the two competitors. In this dataset, SELGB$_S$ is the second-best, consistently achieving performance equal to or better than the baseline LambdaMART. Therefore, selecting only the highest-ranked negative documents did not lead SELGB$_S$ to overfitting on MSLR-30K; however, this definitely resulted in the loss of useful information. In fact, SELGB$_{HL}$, which leverages both the highest and the lowest-ranked negative documents, was able to attain superior performance. Additionally, SELGB$_{HL}$ includes 40% of the lowest-ranked negative documents and only 20%

of the highest-ranked ones, in contrast to SELGB$_S$, which includes 40% of the highest-ranked negative documents. This highlights that SELGB$_{HL}$ can achieve the highest effectiveness by prioritising the lowest-ranked negative documents over the highest-ranked ones.

For the YAHOO! SET 1 dataset, the results are slightly different. Once again, SELGB$_{HL}$ is the superior model, demonstrating statistically significant improvements over its competitors. As concern for SELGB$_S$, in the latter part of the training, it loses the advantage initially achieved over LambdaMART. This underscores that in a positively unbalanced dataset like YAHOO! SET 1, where negative documents constitute only 26.09% of the dataset, leveraging information from the lowest-ranked negative documents is essential to train effective models.

Finally, we observe different results also on the ISTELLA-X dataset. In this scenario, both SELGB$_{HL}$ and SELGB$_S$ exhibit the most evident statistically significant increments compared to the baseline LambdaMART. However, there's no statistical evidence to assert that SELGB$_{HL}$ is superior to SELGB$_S$, and *vice versa*. Given that ISTELLA-X comprises 99.83% negative documents, even if only the 1% of the highest-ranked negative documents is selected, the set of negative documents is still consistently more prominent than the 0.17% of positive documents. Consequently, this sample of highest-ranked negative documents is sufficiently larger to encapsulate all the necessary information to distinguish between positive and negative documents, including the lowest ones. Additional negative documents in the training set do not yield statistically better results.

We can conclude from these results that using HIGH_LOW_SAMPL as the selection strategy allows for creating training sets of higher quality compared to SEL_SAMPL. Specifically, it enables achieving better performance in positively unbalanced datasets (such as YAHOO! SET 1) or balanced datasets (like MSLR-30K). However, it does not yield improvements, compared to the state of the art, in strongly negatively unbalanced datasets (like ISTELLA-X), delivering performance comparable to SEL_SAMPL.

5.5.2 Efficiency

In this section, we assessed the models' performance in terms of training efficiency, where a more efficient training phase leads to reduced training time. To conduct this experiment, we utilised a machine equipped with an Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz and an operating system Ubuntu 20.04.4 LTS. The experiments were executed without parallelisation in a single thread.

The results presented in Table 5.3 represent the training times required to train the best models identified through model selection on the validation set concerning models' effectiveness. The specific hyperparameters yielding the best performance are referred to as *best*, which can be found in Table 5.2.

Table 5.3: Efficiency in terms of training time (milliseconds). The label *best* means the hyperparameters found through model selection with respect to effectiveness.

Algorithm	Dataset		
	Istella-X	MSLR-30K	Yahoo! Set 1
	training time per tree		
LambdaMART	13.2 sec	2.8 sec	1.8 sec
SelGB$_S$, *best*	3.7 sec	2.2 sec	1.6 sec
SelGB$_{HL}$, *best*	3.9 sec	2.3 sec	1.6 sec
	total training time		
LambdaMART	128.7 min	35.7 min	21.3 min
SelGB$_S$, *best*	25.3 min	30.0 min	18.3 min
SelGB$_{HL}$, *best*	28.5 min	34.3 min	22.7 min
	percentage of training set		
LambdaMART	100.0%	100.0%	100.0%
SelGB$_S$, *best*	1.1%	58.4%	81.6%
SelGB$_{HL}$, *best*	3.1%	79.0%	91.2%

Each model used for the result in Table 5.3 has a different number of trees at which the training process terminates. Therefore, alongside the total training time for each model, we provide the time spent to train a single tree (training time per tree). This result allows us to understand the actual impact of the training set size on the training time. Lastly, Table 5.3 also displays the size of the training set used to train the respective model.

Note that we can fairly compare the efficiency of LambdaMART with SelGB$_{HL}$ and SelGB$_S$ since the cost of executing the selection strategy is negligible and can be ignored. The reason for this is that the asymptotic complexity of both High_Low_Sampl and Sel_Sampl is $O(n \log n)$, where n is the number of negative documents within the query. In detail, the selection strategies' cost lies in the sorting algorithm used to rank the negative documents to retrieve the lowest-ranked and highest-ranked negative documents. However, this cost can be amortised since, at each iteration of the learning algorithm, the algorithm orders the examples to evaluate the model's performance. Consequently, High_Low_Sampl and Sel_Sampl have no overhead on the training time. Moreover, due to the lower number of documents being processed at training time, with the same number of iterations, the cost of training SelGB$_S$ and SelGB$_{HL}$ is theoretically less than or equal to that of LambdaMART.

The training times reported in Table 5.3 demonstrate that, in general, both SelGB$_{HL}$ and SelGB$_S$ are more efficient than LambdaMART as they operate

on smaller training sets. However, SELGB$_{HL}$ tends to be slightly slower than SELGB$_S$ due to the additional cost of processing the lowest-ranked negative examples. Nevertheless, this cost is relatively small, especially considering the significant improvement in model effectiveness. Consequently, we can assert that the proposed selection strategy enables statistically superior performance in terms of effectiveness compared to the state of the art (i.e., SELGB$_S$) while maintaining comparable efficiency. Moreover, it is consistently more efficient and effective than the baseline (i.e., LambdaMART).

Note that there are minor discrepancies where LambdaMART takes less time to train than the other two strategies. This discrepancy arises from the different termination points of the training process. Since early stopping on the validation set was used as the training stopping criterion, it is possible that the training with LambdaMART terminated earlier than those of SELGB$_S$ or SELGB$_{HL}$, resulting in a lower total training time even though it used the entire training set. Moreover, the training times per tree show that SELGB$_{HL}$ and SELGB$_S$ strategies are consistently faster in training a single tree, confirming this motivation.

In conclusion, we can infer that the reduction in training time is not directly proportional to the size of the training set. For instance, when examining the performance on ISTELLA-X where LambdaMART utilises 100% of the training set and SELGB$_S$ only 1%, the reduction in training time is not a factor of 100 but rather approximately a factor of 5.

5.6 In-Depth Analysis

The in-depth analysis we conducted is divided into understanding the effect of the hyperparameter p of SEL_SAMPL used in SELGB$_S$ and two hyperparameters p_1 and p_2 of HIGH_LOW_SAMPL used in SELGB$_{HL}$. Finally, we thoroughly examined the effect of the lowest-ranked negative documents on the model, understanding why their inclusion leads to increased model effectiveness. These detailed analyses were entirely conducted on the MSLR-30K dataset.

5.6.1 Hyperparameter Analysis

In this section, we detail the effect of the selection strategies SEL_SAMPL and HIGH_LOW_SAMPL by varying the values of their hyperparameters.

Each model is trained with up to 1,000 trees, using early stopping criteria based on the model's performance on the validation set. In the extensive study conducted by Lucchese *et al.* [115], they proved that the best value for the SEL_SAMPL's hyperparameter n is 1. For this reason, we kept hyperparameter $n = 1$ for both SELGB$_S$ and SELGB$_{HL}$ also in this work. With this preliminary setting, we

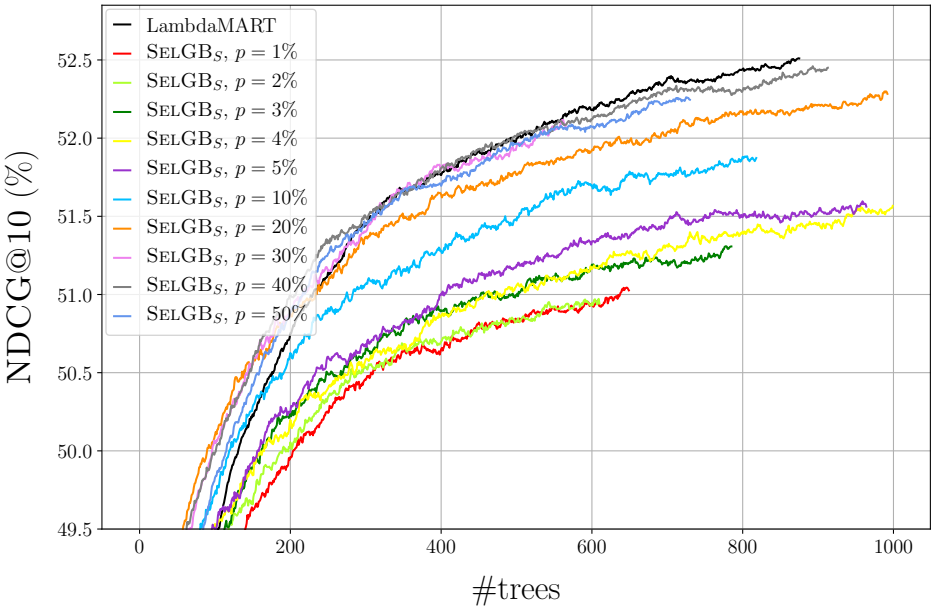


Figure 5.2: Performance in terms of NDCG@10 achieved by SelGB$_S$ when varying hyperparameter p on MSLR-30K validation set

performed hyperparameter tuning on the remaining selection strategy hyperparameters, selecting the best models through model selection on the validation sets.

Firstly, we analysed the behaviour of SelGB$_S$ by varying the hyperparameter p of Sel_Sampl. Figure 5.2 displays the NDCG@10 over training for each model learned by SelGB$_S$ on the MSLR-30K validation set as we varied the parameter p. For completeness, the figure also includes the LambdaMART baseline.

As depicted in Figure 5.2, the performance of SelGB$_S$ deteriorates as p decreases, indicating that an overly aggressive removal of negative examples compromises the model's performance. Consequently, SelGB$_S$ is unable to enhance the NDCG@10 performance compared to the baseline on the MSLR-30K dataset. This behaviour is also observed in the Yahoo! Set 1 dataset. Conversely, the performances on Istella-X align with those reported in the original work [115], where the best performance is achieved for very small values of p.

Subsequently, we analysed the effect of the hyperparameters p_1 and p_2 used in our selection strategy SelGB$_{HL}$. In order to find the best values for the hyperparameters p_1 and p_2 and avoid an exhaustive quadratic grid search, we started the analysis by selecting the best three p values from the hyperparameter tuning of SelGB$_S$ as candidate best values for hyperparameter p_1 of High_Low_Sampl. Specifically, the chosen p_1 values are 20%, 30%, and 40% for MSLR-30K and Yahoo! Set 1, and 0.25%, 0.5%, and 1.0% for Istella-X. We then fine-tuned

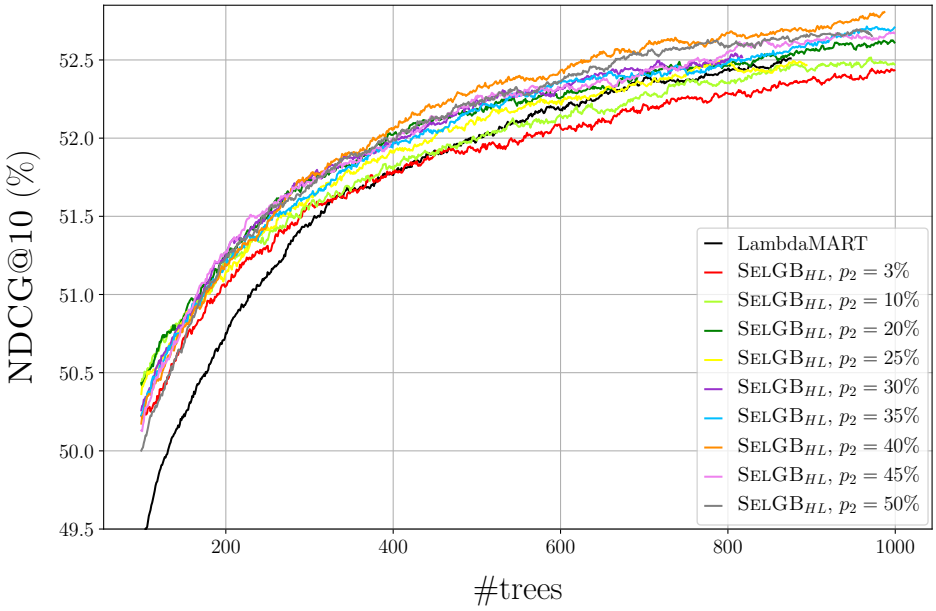


Figure 5.3: Performance in terms of NDCG@10 achieved by SELGB$_{HL}$ with $p_1 = 20\%$, when varying hyperparameter p_2 on MSLR-30K validation set

the value of p_2 for each candidate p_1.

In Figure 5.3, we illustrated the performance of SELGB$_{HL}$ by varying p_2 while keeping the hyperparameter p_1 fixed at 20%. Even in this case, the figure includes the LambdaMART baseline. For clarity, Figure 5.3 only displays models that achieve NDCG@10 higher than LambdaMART on the validation set. Through the model selection on the validation set, we discovered that $p_1 = 20\%$ is the optimal value among the three p_1 values tested on MSLR-30K. As concerns the other datasets, the best values for the hyperparameters p_1 and p_2 are the ones reported in table 5.2.

In Figure 5.3, it is evident that, unlike the case with SELGB$_S$, models trained with SELGB$_{HL}$ consistently attain superior NDCG@10 performance throughout the training phase compared to LambdaMART. These findings confirm that including the lowest-ranked negative examples is crucial for training effective models.

Furthermore, Figure 5.3 highlights how HIGH_LOW_SAMPL improves the models' effectiveness even in the early training iterations. This further validates the impact that HIGH_LOW_SAMPL has on efficiency during the training phase. For instance, the model trained with $p_2 = 40\%$ achieves the same performance as LambdaMART, but with 600 trees instead of 800, saving 200 iterations. Alternatively, a more aggressive pruning of the training set can be applied, for example, with $p_2 = 3\%$ or 10% to achieve similar performance to LambdaMART while sig-

nificantly reducing the size of the training set and, consequently, a significative reduction of the training time.

5.6.2 The Impact of the Lowest Ranked Documents

In this last segment of the experimental section, we conducted a thorough analysis to evaluate the influence of the lowest-ranked negative examples on model predictions. This analysis aimed to underline why HIGH_LOW_SAMPL is superior to SEL_SAMPL. Specifically, we sought to emphasise how disregarding the lowest-ranked negative examples can detrimentally affect the model's effectiveness.

We carried out this examination on models trained using both SELGB$_S$ and SELGB$_{HL}$, clearly demonstrating that the models trained with SELGB$_{HL}$ are more stable and effective due to the inclusion of the lowest-ranked negative examples.

To accomplish this, we selected a subset of queries from the MSLR-30K training set and observed how documents' ranks vary after each tree in both SELGB$_S$ and SELGB$_{HL}$. We used the best hyperparameters identified through model selection, as detailed in Table 5.2; specifically, $p = 40\%$ for SEL_SAMPL and $p_1 = 20\%$ and $p_2 = 40\%$ for HIGH_LOW_SAMPL. Figure 5.4 depicts the result of this analysis for one specific query (query ID: 1349), but we highlight that the same behaviour was observed in all queries in the sample.

Figure 5.4 shows with a colour the label of the document ranked at the i-th position (on the x axis) after the t-th tree of the forest (on the y axis). The green colour represents positive documents (i.e., documents with a relevance label greater than 0). For the sake of simplicity, we sampled only queries where positive documents have the same relevance label. The red colour stands for those documents that are consistently scored the lowest, i.e., those that occur in the bottom p_2 portion of the rank positions after each of the 1,000 trees. We computed this set for SELGB$_{HL}$ (on the right-hand side), and we report their rank positions also for SELGB$_S$ (left-hand side). We denote this set with $D_{HL}^{-\cap}$. The light grey colour is used for the remaining non-relevant documents. The width of each rank position is proportional to the logarithmic discount factor of NDCG to highlight the top-ranked positions that contribute the most to the ranking quality.

There are several interesting insights we can provide with this fine-grained analysis (see Figure 5.4).

- *i)* the documents within the set $D_{HL}^{-\cap}$ are roughly the same across boosting iterations. This might be expected from its definition. In fact, after 1,000 trees, $D_{HL}^{-\cap}$ always cover the 50% of the lowest p_2 portion of the ranking. Computing the same statistic on SELGB$_S$ leads to a poor 10%, and just 15% at iteration 66.

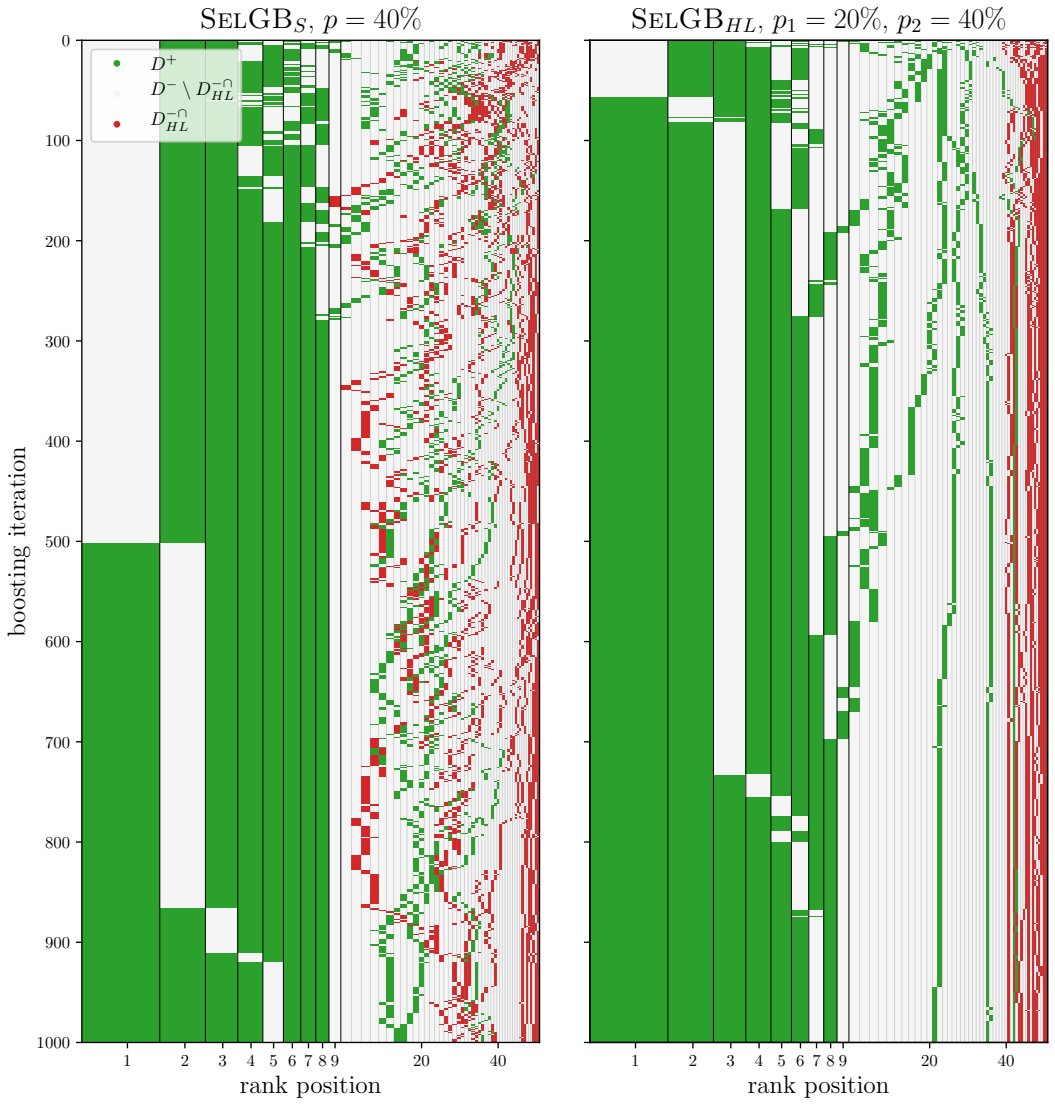


Figure 5.4: Document rankings (x axis) at each boosting iteration (y axis) for both SELGB$_S$ (left) and SELGB$_{HL}$ (right) for query 1349 sampled from MSLR-30K training set. The rank position widths are proportional to the logarithmic discount factor of NDCG. In green colour are query documents with relevance > 0. In red colour are documents consistently scored low by HIGH_LOW_SAMPL in 1,000 boosting iterations. The remaining non-relevant documents are in light grey colour.

- *ii)* the negative examples in $D^{-\cap}_{HL}$ have a very stable rank in SELGB$_{HL}$ (red squares, right-hand side), but the same documents have a cluttered behaviour during the training with SELGB$_S$ (red squares, left-hand side).

- *iii)* also positive examples not placed in the top-k position (rightmost green traces) have a very stable behaviour in SELGB$_{HL}$ (right chart); this does not hold for SELGB$_S$ (left chart), where their ranking varies significantly, tree after tree.

- *iv)* in SELGB$_{HL}$, some positive examples require a few more iterations to reach the top-k position of the list, but in the end, they reach higher positions than in SELGB$_S$. As a result, equipping Selective Gradient Boosting with the proposed selection strategy HIGH_LOW_SAMPL produces more effective models.

All of the above confirms that the ranking provided by SELGB$_{HL}$ is of higher accuracy than SELGB$_S$. Some of the reasons for this accuracy might be found in the stability provided by the lowest-ranked negative document. The set of lowest-ranked negative documents provides the model with interesting information about the training process that translated into higher stability, reduced variance, and the opportunity to better refine the predicted document scores.

5.7 Summary

In this section, we provide a comprehensive overview of the main results and contributions of the study introduced in this Chapter titled "On the Effect of Low-Ranked Documents". The summary delves into the implications of the proposed HIGH_LOW_SAMPL selection strategy within the SelGB framework.

- **Selective Gradient Boosting:** The framework SelGB utilises GBDTs to generate decision tree forests for solving the learning-to-rank task. Every n iterations round, it employs a selection strategy, SEL_SAMPL, to select from the training set $\mathcal{D}$ all the positive documents and the p percentage of the highest-ranked negative examples (the most informative ones) to create a higher qualitative training set $\mathcal{D}^*$. Then, the training proceeds from a smaller but more informative subset of the training set.

- **Contribution:** The main contribution of this work lies in the definition of a new documents selection function, HIGH_LOW_SAMPL, for SelGB framework. HIGH_LOW_SAMPL enriches the selection made by SEL_SAMPL by selecting not only the p_1 percentage of the highest-ranked negative documents but also the p_2 percentage of the lowest-ranked negative documents to achieve a higher qualitative dataset and without overfitting to difficult documents.

- **Main Results:** Through extensive experiments, we empirically proved that SelGB combined with the new selection strategy HIGH_LOW_SAMPL has a statistically significant increase in performance, in terms of NDCG, compared to the state-of-the-art SELGB$_S$ (i.e., the previous version of SelGB combined with SEL_SAMPL) and the baseline LambdaMART. Moreover, SelGB equipped with HIGH_LOW_SAMPL achieves a speed-up in the training process compared to LambdaMART without compromising model effectiveness.

- **In-Depth Analysis:** Finally, we provided an in-depth analysis to underscore the impact of the lowest-ranked negative documents on model predictions. The analysis proved that the presence of the lowest-ranked negative documents is fundamental to improving stability and effectiveness. In fact, the ranking produced by the model trained considering the lowest-ranked negative documents is shown to be more stable and with lower variance. So we concluded that HIGH_LOW_SAMPL can make the most from the training set by retrieving useful information that enriches the learning process and brings models with higher stability and less variance.

5.7.1 Future Work

In this Chapter, we proposed HIGH_LOW_SAMPL, a novel document selection strategy tailored for the SelGB framework, significantly enhancing the effectiveness of learning to rank models. Building upon this research, there are several promising directions for future work:

- **Dynamic Hyperparameters Tuning:** Exploration of adaptive strategies to dynamically adjust hyperparameters, such as p_1 and p_2, during the training process based on the model's performance or dataset characteristics. This could improve the quality of each training set created by the document selection strategy, making them more tailored for the training moment the learning algorithm faces.

- **Generalisation to Deep Learning Models:** Extending the application of HIGH_LOW_SAMPL beyond gradient boosting-based models to deep learning architectures for Learning to Rank. Investigating how HIGH_LOW_SAMPL can be adapted and integrated into neural network-based ranking models could open new avenues for research.

- **Move Focus on Positive Documents:** In both Lucchese *et al.* [115] and this study, the document selection functions have predominantly focused on negative documents (i.e., documents with a relevance label equal to 0). However, positive documents can also potentially diminish the quality of the training set by diverting the model's attention from particularly relevant positive documents. For example, in positively unbalanced datasets where the number of positive documents is very high, a small size of the top relevant ranking positions (i.e., the ones most viewed by the users) may generate competition between the positive documents. A prospective avenue of research involves defining a selection strategy that includes identifying the most significant positive documents.

- **Integration with Online Learning:** Investigating the possibility of applying HIGH_LOW_SAMPL into online learning settings where models require continuous updates. From the results reported in this Chapter, we demonstrated that HIGH_LOW_SAMPL can selectively choose a small subset of high-quality documents from the available dataset based on the model scores in a precise training moment. This capability can be leveraged in online LtR contexts where a reduced quantity of high-quality data is needed to expedite the online training process.

Chapter 6

LambdaRank Gradients are Incoherent

In this Chapter, we discuss the work titled "*LambdaRank Gradients are Incoherent*", in proceedings as a full paper at the *CIKM '23: The 2023 ACM International Conference on Information and Knowledge Management*. Further details can be found in the reference [122].

As mentioned in Section 3.2.3.1, a significant limitation in advancing research on Learning to Rank models' effectiveness is the non-differentiability of Information Retrieval metrics. Typically, IR metrics depend on the sorted list of documents, rendering an objective function utilising IR metrics not directly optimisable by gradient descent-based methods due to the non-differentiability or flatness of these metrics concerning the model parameters [25, 22, 62].

Despite the non-differentiability of IR metrics, numerous LTR algorithms rely on gradient-based approaches. They either optimise an approximate version of the ranking metric or construct gradients based on heuristic approximations, such as LambdaRank [25]. LambdaRank bypasses the need to define an approximate loss function; instead, it heuristically defines gradients indicating whether a document score should be increased or decreased to enhance the ranking quality. Initially designed for artificial neural networks, LambdaRank stood as a state-of-the-art algorithm in LTR until it was surpassed by LambdaMART [27], an analogous version based on gradient-boosted decision trees. Given that both LambdaRank and LambdaMART rely on heuristics, their gradients do not precisely compute the derivative of an IR metric.

In this work, we show that LambdaRank heuristics (and its derivatives like LambdaLoss Framework [173], LambdaMART, etc.) have an inherent flaw and they can generate incoherent gradients. Later in Figure 6.1, we show a few examples where the most relevant document in the result list does not get the largest gradient, and therefore, how it is impossible for the learned model to rank it in

the top position. We call *gradient incoherency* such phenomena where a relevant document receives a smaller gradient than a less relevant one. We are aware that gradients are approximate, and, therefore, trade-offs need to be made to optimise non-differentiable functions. However, such incoherency may undermine the learning process. Moreover, this phenomenon is more prominent with the use of truncated metrics optimisation, where we would like the model to focus on the top positions. Due to the gradient incoherency, the gradients are unable to push the most relevant documents upward.

The contributions of this work are as follows. *i)* We bring to light the issue of gradient incoherencies affecting LambdaRank, which has not been previously shown in the literature. *ii)* We show how truncated metric optimisation exacerbates the phenomenon of gradient incoherencies and undermines the aim of truncation, which is to ensure that the user encounters the most relevant documents among the first positions. *iii)* We propose an improvement over the LambdaRank gradient computation to optimise truncated ranking metrics. Specifically, we propose Lambda-eX, which *extends* the set of document pairs considered by LambdaRank when computing gradients.

We validate experimentally our results on five publicly available datasets. We show how Lambda-eX can reduce the number of queries affected by gradient incoherencies introduced by truncated metric optimisation. This reduction is significantly large during the early stage of the training, which allows Lambda-eX to achieve high-quality performance after a few trees. Finally, we show that optimising truncated metrics can accelerate training time due to a lower number of partial derivatives to be computed and that Lambda-eX can achieve statistically significant improvements while maintaining the same train efficiency as truncated LambdaRank.

The Chapter is structured as follows. In Section 6.1, we introduce the related work specific to research aiming to optimise IR metrics. In Section 6.2, we delve into the details of LambdaRank and its derivatives. Furthermore, we provide a detailed introduction to the optimisation of truncated metrics and how it is defined within the objective functions of LambdaRank and its derivatives. In Section 6.3, we delve into the core contribution of this work, introducing the concept of *gradient inconsistency* and how LambdaRank and its derivatives are affected by it. Moreover, we explain how these gradient inconsistencies negatively influence model learning and demonstrate how the truncated metric optimisation exacerbates this gradient inconsistency problem. In Section 6.4, we introduce the main contribution of this work, Lambda-eX, a learning algorithm for reducing gradient inconsistencies while optimising truncated metrics. Moreover, we demonstrated that Lambda-eX can achieve statistically significant improvement in effectiveness without sacrificing the training efficiency brought by the truncated metric opti-

misation. Section 6.5 details the experimental setup used to perform experiments and the learning algorithms utilised to compare our solution. Section 6.6 showcases the main results of this work, focusing on both the effectiveness and efficacy of the proposed solution. Section 6.7 provides in-depth analyses to show how the proposed strategy effectively reduces the exacerbation of gradient inconsistencies introduced by truncated metric optimisation and how, even in the early training iterations, the proposed solution succeeds in training models that are statistically more effective than the reference baselines. Finally, in Section 6.8, we summarise the contributions, results, and potential future work of this study.

6.1 Related Work: IR Metrics Optimisation

One of the biggest challenges in Learning to Rank is metrics optimisation. Ranking metrics are non-differentiable since they are inherently tied to the order of the documents. Therefore, unlike most machine learning techniques, an objective function that makes use of IR metrics cannot be directly optimised by gradient descent methods. However, effective ranking models are essential in a huge variety of applications in IR systems. This began an arms race to address the problem from multiple directions. Among those attempts are ranking metrics approximation or loss function that indirectly aligns with the desired metric. However, most LTR approaches optimise a loss function that is loosely related to a ranking metric or is its upper bound.

A well-known class of solutions that aims to minimise the number of errors committed in ranking pairs of documents is that of pairwise approaches. These include RankNet [26], a pairwise approach that optimises a probabilistic loss function by mapping the model output to a learned probability. Another well-known approach is AdaRank [182], which learns weak rankers that minimise the pairwise misranking error and then linearly combines them for prediction. Pairwise approaches typically optimise a convex upper-bounds of the pair misranking error. However, this optimisation does not directly imply an improvement in the ranking metric, thus leading to a mismatch between model optimisation and effectiveness on the desired IR metric.

To fill the gap, listwise approaches embed the information on the status of the entire ranking list into the optimisation process. Listwise approaches fall mainly into two macro-categories, those that approximate the ranking metric through a smooth surrogate of it, such as SoftRank [161] and ApproxNDCG [142], and those that use heuristics to construct a smooth surrogate loss function such as ListNET [37], XE_{NDCG} [22], and LambdaRank [25]. ListNET minimises the cross-entropy between ground truth and the model's score distribution. XE_{NDCG} is a cross-entropy loss function that guarantees strong theoretical properties like

optimising a convex bound on mean NDCG. Then, LambdaRank is a learning algorithm which does not try to optimise a loss function but heuristically defines the loss gradient. The way LambdaRank defines its gradients is of fundamental importance to this work; hence, the next section will be entirely devoted to this aspect.

Worth mentioning is also the study conducted in [62], where the authors investigated whether training a model on the same truncated metric used for the evaluation (e.g., NDCG@k) is better than training it on the un-truncated metric (e.g., NDCG). What emerged is in line with the discovery made with our research: training on un-truncated metrics returns better-performing models. However, the reason given by [62] only tells half the story. They claim that optimising truncated metrics reduces the number of contributions λ_{ij} each gradient λ_i receives. They also showed that the same performance as the un-truncated metric can be achieved with an equal number of document pairs (of λs) during training. However, in this work, we show that with the same order of magnitude in the number of pairs as in the truncated metric optimisation, we can obtain the same performance as the un-truncated metric optimisation. This confirms that the reason behind the drop in performance lies in the exacerbation of the gradient incoherencies rather than in a general lack of pairwise document comparisons. So, selecting the right pairs of documents makes it possible to obtain the same performance as the un-truncated metric.

6.2 LambdaRank

Gradient-based learning algorithms, such as artificial neural networks or gradient-boosted decision trees, run iterative updates to build a ranker that minimises a given cost function C. For instance, artificial neural networks compute the gradient direction $\partial C/\partial w_j$ to update each network weight w_j at each batch processed. Similarly, gradient-boosted decision trees iteratively learn a new tree that approximates $\partial C/\partial s_i$ for each document d_i in the training set. In both cases, directly or indirectly, a key step is the computation of $\partial C/\partial s_i$. Unfortunately, most IR metrics are rank-based: they depend on documents ranking π rather than s_i. This makes the cost function either flat, i.e., modifications of s_i do not change π and therefore do not change the cost C, or non-differentiable, i.e., modifications of s_i change π causing a non-smooth change of the cost C.

Most approaches drive the learning process by means of a proxy cost function that is differentiable. One of the most relevant approaches is LambdaRank [25]. LambdaRank's cost function stems from the RankNet cost [26], which is further enhanced by considering the impact on the IR metric at hand. The gradient $\partial C/\partial s_i$ is computed on the basis of pairwise *lambdas* λ_{ij}. Given a document pair

d_i and d_j such that d_i is more relevant than d_j, i.e., $y_i > y_j$, we have that:

$$\lambda_{ij} = \frac{\partial C(s_i - s_j)}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} \left|\Delta \mathrm{Z}_{ij}\right|, \tag{6.1}$$

where $\sigma = 1$, and $|\Delta \mathrm{Z}_{ij}|$ is the amount of change in the IR metric Z generated by swapping the rank positions of d_i and d_j while leaving the rank positions of all other documents unchanged. The value of λ_{ij} estimates the change on the cost function C when the distance between the two scores s_i and s_j is modified. Note that if two documents have the same relevance label, then $\lambda_{ij} = 0$ due to the fact that $\Delta \mathrm{Z}_{ij} = 0$. We recall that λ_{ij} implements the derivative of the RankNet cost function multiplied by $|\Delta \mathrm{Z}_{ij}|$. The RankNet cost increases if the two documents are not in the correct order and converges asymptotically to 0 if the documents are in the correct order with a large gap in their scores. The $|\Delta \mathrm{Z}_{ij}|$ component boosts the error when this has a significant impact on the specific IR metric.

The gradient of the single document is finally computed as:

$$\lambda_i = \sum_{j:(i,j)\in I} \lambda_{ij} - \sum_{k:(k,i)\in I} \lambda_{ki}, \tag{6.2}$$

where I is the set of ordered pairs (i, j) such that $y_i > y_j$, i.e., $I = \{(i, j) \mid d_i, d_j \in D \ \wedge \ y_i > y_j\}$. In regard to the asymptotic complexity of computing λ_{ij}, Equation 6.2 requires to evaluate $O(n^2)$ document pairs with n being the number of candidate documents in D for the given query.

According to [25], when maximising an IR metric such as NDCG, the lambdas are formulated as $\partial U / \partial s_i$ where U is the utility function (metric) being maximised, rather than a cost to be minimised. Moreover, the sign of the various λ_{ij} is set so that the most relevant document d_i receives a positive gradient update, while d_j receives a negative update and is *pushed* down through the ranks π.

Before going through the next section, let's focus on the $|\Delta \mathrm{Z}_{ij}|$ in Equation 6.1. Indeed, Z could be any ranking metric such as NDCG, ERR, etc. In this work, we focus on NDCG, although the results and conclusions we made also generalise to other metrics. For this reason, we briefly reintroduce the NDCG metric, previously introduced in Section 3.2.2.1. NDCG is calculated by the ratio of DCG to IDCG (i.e., the ideal DCG of the ground truth ranking).

The DCG consists of two components: the gain $\mathrm{G}_i = 2^{y_i} - 1$ that document d_i, with relevance y_i, contributes to the final ranking, and the discounting $\mathrm{D}_i = \log_2(1 + \pi[i])$ that discounts the document's gain based on its position $\pi[i]$ in the list. In fact, a very relevant document in the last position contributes much less to NDCG than in the first position.

Below, we provide a concise definition of NDCG:

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}} = \frac{1}{\mathrm{IDCG}} \sum_{i=1}^{n} \frac{\mathrm{G}_i}{\mathrm{D}_i} = \frac{1}{\mathrm{IDCG}} \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log_2(1 + \pi[i])},$$

Therefore $|\Delta\text{NDCG}_{ij}|$ is defined as the difference between the NDCG computed on the current ranking π and the NDCG computed on the ranking that results from swapping the two documents at ranks $\pi[i]$ and $\pi[j]$. The $|\Delta\text{NDCG}_{ij}|$ can be efficiently computed as follows:

$$\Delta\text{NDCG}_{ij} = |\text{G}_i - \text{G}_j| \left| \frac{1}{\text{D}_i} - \frac{1}{\text{D}_j} \right| \tag{6.3}$$

By combining Equation 6.1 and 6.3, each λ_{ij} optimising NDCG can be computed with the following equation:

$$\lambda_{ij} = \frac{1}{1 + e^{(s_i - s_j)}} \frac{1}{\text{IDCG}} |\text{G}_i - \text{G}_j| \left| \frac{1}{\text{D}_i} - \frac{1}{\text{D}_j} \right|. \tag{6.4}$$

Therefore, the gradient λ_{ij} has three components: the RankNet cost, the gain difference and the difference of the inverse discount.

6.2.1 Lambda Loss Framework

About a decade later, in [173], the authors introduced LambdaLoss, a probabilistic framework for optimising ranking metrics. This framework aims to provide theoretical justifications for empirically effective learning algorithms like LambdaRank. Despite LambdaRank's effectiveness, the underlying loss it optimises for remained unknown until the introduction of this framework. In [173], it was demonstrated that LambdaRank is a special configuration with a well-defined loss in the LambdaLoss framework, thus providing theoretical justification for it. Moreover, they showed that LambdaRank optimises a coarse upper bound of NDCG.

More importantly, this framework allows the definition of metric-driven loss functions that have clear connections to different ranking metrics. The article defined various metric-driven loss functions based on NDCG and ARP. Among those are the metric-driven loss function NDCG-Loss2 and the hybrid loss NDCG-Loss2++, optimising the NDCG metric and yielding the most statistically significant results. Furthermore, NDCG-Loss2 has been proved to be an upper bound of the NDCG metric.

The metric-driven loss function NDCG-Loss2 shares many similarities with LambdaRank. The main difference lies in the definition of the discount used in ΔNDCG_{ij}. Specifically, the discount $\rho_{ij} = |1/\text{D}_i - 1/\text{D}_j|$ used by LambdaRank in Equation 6.4 becomes $\delta ij = |1/\text{D}_{|i-j|} - 1/\text{D}_{|i-j|+1}|$ in the metric-driven loss function NDCG-Loss2.

Finally, in [173], they also proposed a hybrid loss function named NDCG-Loss2++, which is a linear combination of the two discountings, $\rho_{ij} + \mu\delta_{ij}$, with μ being a weight coefficient managing the trade-off between the two. Consequently,

the gradients calculated through the hybrid loss function NDCG-Loss2 are as follows:

$$\lambda_{ij} = \frac{1}{1 + e^{(s_i - s_j)}} \frac{1}{\text{IDCG}} |\text{G}_i - \text{G}_j| \, (\rho_{ij} + \mu\delta_{ij}). \tag{6.5}$$

6.2.2 Truncated Metric Optimisation

Real-world applications of information retrieval systems mostly try to optimise the effectiveness for only the first k results. This manner is strictly related to user behaviour [62]. When users scan a list of results, they focus more on the first k (i.e., 5/10) results and do not look at all the thousands of results in the list. IR metrics naturally provide a truncated version with a *cutoff threshold* k. For instance, NDCG@k is computed by considering only the contribution of the top-k ranked documents. Truncated metrics are the IR metrics of interest to evaluate the goodness of a ranker in most application scenarios. Therefore, according to the empirical risk minimisation principle, optimising a truncated metric is expected to be more effective than its un-truncated variant.

Optimising a truncated metric at training time also brings a straightforward efficiency improvement. In Equation 6.2, the computation of all λ_i requires computing the pairwise gradients λ_{ij} for every pair of documents $d_i, d_j \in D$. Under a truncated metric, documents ranked beyond k are not considered, and therefore, a pair of such documents has a value of $|\Delta\text{Z}_{ij}@k|$ equal to 0. Thus, the pairs to be considered are limited to those that contain at least one document in the top-k. To do so, the gradients λ_i in Equation 6.2 are computed by replacing the set I with a I_τ defined as follows:

$$I_\tau = \{(i, j) \mid d_i, d_j \in D \wedge y_i > y_j \wedge \min(\pi[i], \pi[j]) \leq \tau\}. \tag{6.6}$$

From now on, we denote by k the *cutoff* threshold used by the evaluation metric and by τ the *truncation level* [62] optimised during the training. Note that k and τ do not need to be the same. When $\tau = k$, we are maximising the truncated metric with cutoff k, while when $\tau = +\infty$, we are maximising the un-truncated metric.

Last but not least, the introduction of τ reduces the asymptotic complexity of the objective function from $O(n^2)$ to $O(\tau n)$. A significant reduction of the training time is achieved when $\tau \ll n$.

Despite being a tiny detail, we remark that we assume the use of Equation 6.1 (and Equation 6.4) without modification even in the presence of a truncation level, i.e., ΔZ_{ij} is used rather than $\Delta\text{Z}_{ij}@k$, as this is common practice in the most popular implementations, e.g., LightGBM [93].

6.3 Contribution 1: Gradient Incoherency

LambdaRank and its variants provide a smooth approximation of commonly used IR metrics. Yet, we would like this approximation to provide some basic guarantees. We thus introduce a *coherency* property defined as follows.

Definition 4 (Gradient Coherency)**.** Given two documents d_i and d_j such that $y_i > y_j$ and $\pi[i] > \pi[j]$, i.e., d_i is more relevant than d_j, but it is ranked at a worse position, we say that the gradients of the utility function U are *coherent* at d_i, d_j if it holds that:

$$\frac{\partial U}{\partial s_i} \geq \frac{\partial U}{\partial s_j}.$$

The above definition states that if two documents d_i and d_j are misranked, we would like the computed gradient to be larger at the most relevant document d_i. This is because pushing up d_i more than d_j may restore the ideal ordering. Conversely, when the gradient coherency does not hold, pushing up d_j more than d_i may only worsen the current ranking. Despite its simplicity, the gradient coherency property is not easy to satisfy, and, indeed, it is not enjoyed by LambdaRank gradients.

6.3.1 On Truncated Optimisation

In Figure 6.1(a), we show an example of LambdaRank gradients computation when maximising NDCG@1 with a truncation level $\tau = 1$. Suppose this is the ranking after a given number of iterations, epochs, or boosting rounds of a gradient-based optimisation algorithm. The most relevant document $d_\triangle$ is currently ranked second, while a less relevant document $d_\star$ is ranked first. Then, there are three other non-relevant documents. The arrows depict the computed gradients, and, as expected, relevant documents get an upward push, while non-relevant documents get a downward push. However, we have that document $d_\triangle$ gets a smaller gradient than $d_\star$, i.e., $\lambda_\triangle < \lambda_\star$, meaning that the most relevant document $d_\triangle$ will not be able to reach the top position in the next iteration. On the contrary, the gap between $d_\star$ and $d_\triangle$ is meant to increase in favour of the least relevant $d_\star$. This is a gradient incoherency according to Definition 4.

To clarify this behaviour, in Table 6.1, we report the computation of the document gradients λ_i as a function of the pairwise λ_{ij} according to Equation 6.4. Recall that λ_{ij} is considered *if and only if* $(i, j) \in I_\tau$, i.e., at least one of the two documents is ranked above the truncation level τ. Let's focus on the relevant documents. Document $d_\star$ receives a negative contribution $-\lambda_{\triangle\star}$ from the most relevant document $d_\triangle$, and three positive contributions from the non-relevant documents. Document $d_\triangle$ is below the truncation level, and therefore its gradient is simply

$\lambda_\triangle = +\lambda_{\triangle\star}$. Therefore, the reason for the gradient incoherency is due to the contribution of the non-relevant documents that significantly contribute to $d_\star$ but not to $d_\triangle$.

The reader may immediately recognise that setting a truncation level $\tau = +\infty$ would solve this issue at the cost of a higher computational cost. While this holds true in this case, as we explained in the next section, it is not always the case that a truncation level $\tau = +\infty$ resolves the issue of gradient incoherencies.

The contribution of this work pursues the following direction: to widen the set of the λ_{ij} considered to reduce the exacerbation of gradient incoherencies introduced by a small truncation level while maintaining a computational cost comparable to that of a small truncation level.

Before moving forward, let's investigate a few similar examples to further understand the behaviour of LambdaRank. Figure 6.1(b) shows a similar scenario to that of Figure 6.1(a), with the most relevant document $d_\triangle$ in the last position. As $d_\triangle$ moves downwards, both the RankNet cost and the discounting component of $\Delta\text{NDCG}_{\triangle\star}$ (i.e., $|1/\text{D}_\triangle - 1/\text{D}_\star|$) increase generating a large upward gradient update for $d_\triangle$ and a symmetrical downward update for $d_\star$. In the setting depicted in Figure 6.1(b), the gradient $\lambda_\triangle$ is larger than $\lambda_\star$ thus complying with the *Gradient Coherency* property.

Table 6.1: Detailed computation of LambdaRank gradients for the example illustrated in Figure 6.1(a).

d_i	y_i	s_i	$\pi[i]$	λ_i
$d_\star$	1	0.04	0	$\lambda_\star = -\lambda_{\triangle\star} + \lambda_{\star 3} + \lambda_{\star 4} + \lambda_{\star 5}$ $\approx -0.124 + 0.083 + 0.093 + 0.100 \approx 0.152$
$d_\triangle$	2	0.03	1	$\lambda_\triangle = +\lambda_{\triangle\star} \approx 0.124$
d_3	0	0.02	2	$\lambda_3 = -\lambda_{\star 3} \approx -0.083$
d_4	0	0.01	3	$\lambda_4 = -\lambda_{\star 4} \approx -0.093$
d_5	0	0.00	4	$\lambda_5 = -\lambda_{\star 5} \approx -0.100$

However, as the rank distance between $d_\star$ and $d_\triangle$ increases, the value of $|1/\text{D}_\triangle - 1/\text{D}_\star|$ increases only marginally. If more non-relevant documents are placed in between $d_\star$ and $d_\triangle$ as in Figure 6.1(c), the contributions of such non-relevant documents provide additional increments to $\lambda_\star$ but do not affect $\lambda_\triangle$. Eventually, $d_\star$ gets a larger gradient than $d_\triangle$, breaking the *Gradient Coherency* property again. This happens because the discounting factor difference between $d_\star$ and $d_\triangle$, which should push $d_\triangle$ upwards stronger than $d_\star$, is too small to overcome the lack of gradient contribution of the pairs removed by the truncation level.

These three examples make us draw two interesting observations. First, the

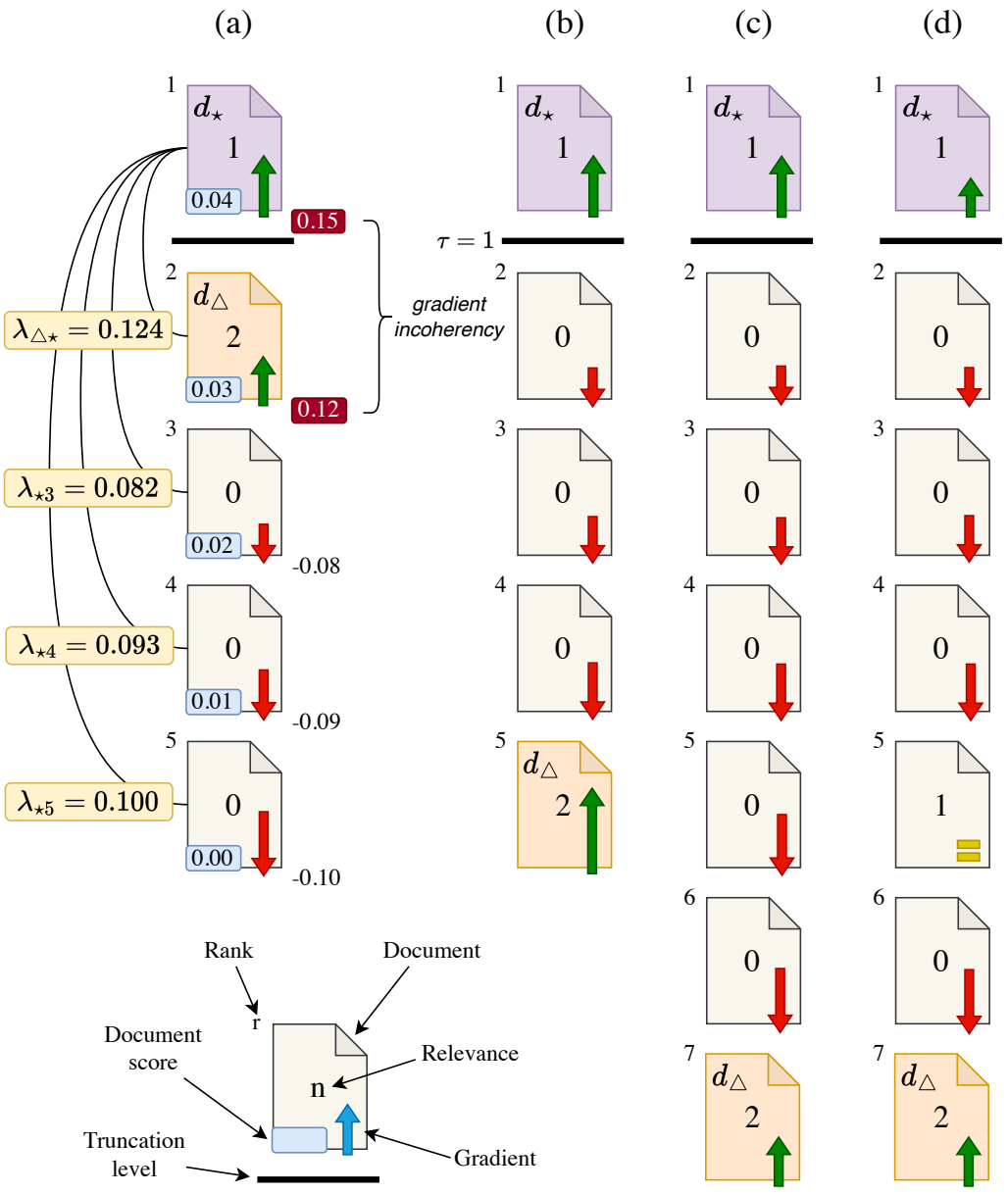


Figure 6.1: Examples of gradient incoherency.

more the dataset contains non-relevant documents (i.e., with relevance labels equal to 0), the more likely the problem will occur. We empirically prove this in Section 6.7.2, especially with the ISTELLA-X dataset. Second, the occurrence of gradient incoherencies is related to the model's error in a non-linear way. Suppose the model ranks document $d_\triangle$ very poorly. In that case, it is impossible for $d_\triangle$ to improve its ranking (Figure 6.1(c)); if the model error is not large, the model will correctly push

$d_\triangle$ upwards (Figure 6.1(b)), but, if document $d_\triangle$ gets just below the truncation level, it is pushed downwards again (Figure 6.1(a)). The only way document $d_\triangle$ can escape this problem is by a fortuitous update (e.g., neural network weight update (LambdaRank) or tree's leaf value (LambdaMART)), that may generate a completely different gradient since it is affected by other documents also from other queries, or by other implicit algorithm-dependent approximations.

A last scenario is illustrated in Figure 6.1(d), where an additional relevant document is added by replacing document d_5 with a document with label $y_5 = 1$. Being the label the same as $d_\star$'s, the value of $\Delta\text{NDCG}_{\star 5}$ is 0 for $\lambda_{\star 5}$, so d_5 reduces the number of gradient contributions to $d_\star$. However, note that the gradient of $d_\triangle$ is not affected by d_5 because they are both beyond the truncation level. The new document makes the gradient of document $d_\star$ smaller than the gradient of document $d_\triangle$, reversing the gradient computation outcome once more.

These examples show how difficult it is to model the *Gradient Coherence* analytically. We provided a few examples showing the impact of the label and the rank difference, which are computed by ΔZ_{ij}. Clearly, also the score difference is relevant and captured by the RankNet component of the gradient in Equation 6.1. So far, we focused on NDCG only. It is easy to show that the coherence property does not hold for other discount-based metrics, such as Expected Reciprocal Rank, Rank-Biased Precision, etc.

6.3.2 On Un-truncated Optimisation

The above discussion suggests that the truncation level τ is the cause of the incoherencies in the gradient computation. Indeed, this is not true.

Let's set $\tau = +\infty$, meaning that no truncation is used, and consider the example in Table 6.2. We have three documents with scores respectively 0.02, 0.01, and 0.00, and with relevance labels respectively 4, 0, and 1. Since there is no truncation level, all the pairwise gradients λ_{ij} are relevant. Document d_1 has a positive gradient λ_1 as it is ranked higher than documents with smaller relevance labels. This positive push allows gaining a desirable *margin* from the other documents. Document d_2 is non-relevant and receives a negative gradient contribution from both the other documents. Unexpectedly, document d_3, despite having an higher label than d_2, receives the strongest downward push, i.e., $\lambda_3 < \lambda_2$ with $y_3 > y_2$. This is a gradient incoherence. The reason is that swapping document d_1 with d_3 has a larger impact on the NDCG than swapping d_1 with d_2, resulting in $\lambda_{13} > \lambda_{12}$. LambdaRank prefers avoiding the risk of moving d_1 to the third position rather than pushing d_3 up to the second place. Indeed, this comes from the discount factor of the NDCG metric that demotes documents' contributions in the lower ranks. These gradients clearly push the ranking away from the ideal configuration.

Table 6.2: Example of computation of LambdaRank gradients with $\tau = +\infty$.

d_i	y_i	s_i	$\pi[i]$	λ_i		
d_1	4	0.02	0	$\lambda_1 = \lambda_{12} + \lambda_{13}$	$\approx 0.176 + 0.221$	≈ 0.397
d_2	0	0.01	1	$\lambda_2 = -\lambda_{12} - \lambda_{32}$	$\approx -0.176 - 0.004$	≈ -0.180
d_3	1	0.00	2	$\lambda_3 = -\lambda_{13} + \lambda_{32}$	$\approx -0.221 + 0.004$	≈ -0.217

This shows that LambdaRank gradients are incoherent independently of the truncation level. In this case, the major player is the discounting factor. Given the larger importance of truncated metrics, we leave the analysis of un-truncated metrics and gradients to future work. In this work, we focus on the cases that break the *Gradient Coherence* in the presence of a truncation level, aiming not to compromise the computational efficiency this provides.

6.4 Contribution 2: Lambda-*eX*

We put ourselves in the scenario of truncated IR metrics optimisation. From the above, we can say that the natural choice of using a truncation level at training time to maximise a truncated metric is very beneficial in terms of computational cost, but it suffers from incoherence in gradient computation. The main contribution of this work is Lambda-*eX*, a new approach to optimise truncated ranking metrics that *limits incoherencies* while preserving *training time efficiency*. Specifically, we propose heuristic methods to *extend* the set of document pairs considered by LambdaRank when computing gradients.

6.4.1 Main Idea

We claim that the gradient incoherencies exacerbate due to missing computations of certain λ_{ij} gradients. More specifically, relevant documents that are not ranked above the truncation level are not evaluated against all the other candidate documents in D but only against the top-k, and this discards some of the λ_{ij} and causes under-estimation of their gradient. One possible approach is to use $I_{\tau=+\infty}$, i.e., the set containing all the pairs of documents $d_i, d_j \in D$. However, this does not allow limiting the number of pairwise gradients computed to minimise the computational cost of the training process.

We thus define a *Full-Gradient Document Set* $X \subseteq D$ for which we compute a *complete* gradient estimation as in the un-truncated case $\tau = +\infty$. We compute λ_{ij} gradients as in Equation 6.2 but on the basis of the set I_X:

$$I_X = \{(i,j) \mid d_i, d_j \in D \wedge y_i > y_j \wedge (d_i \in X \vee d_j \in X)\}. \tag{6.7}$$

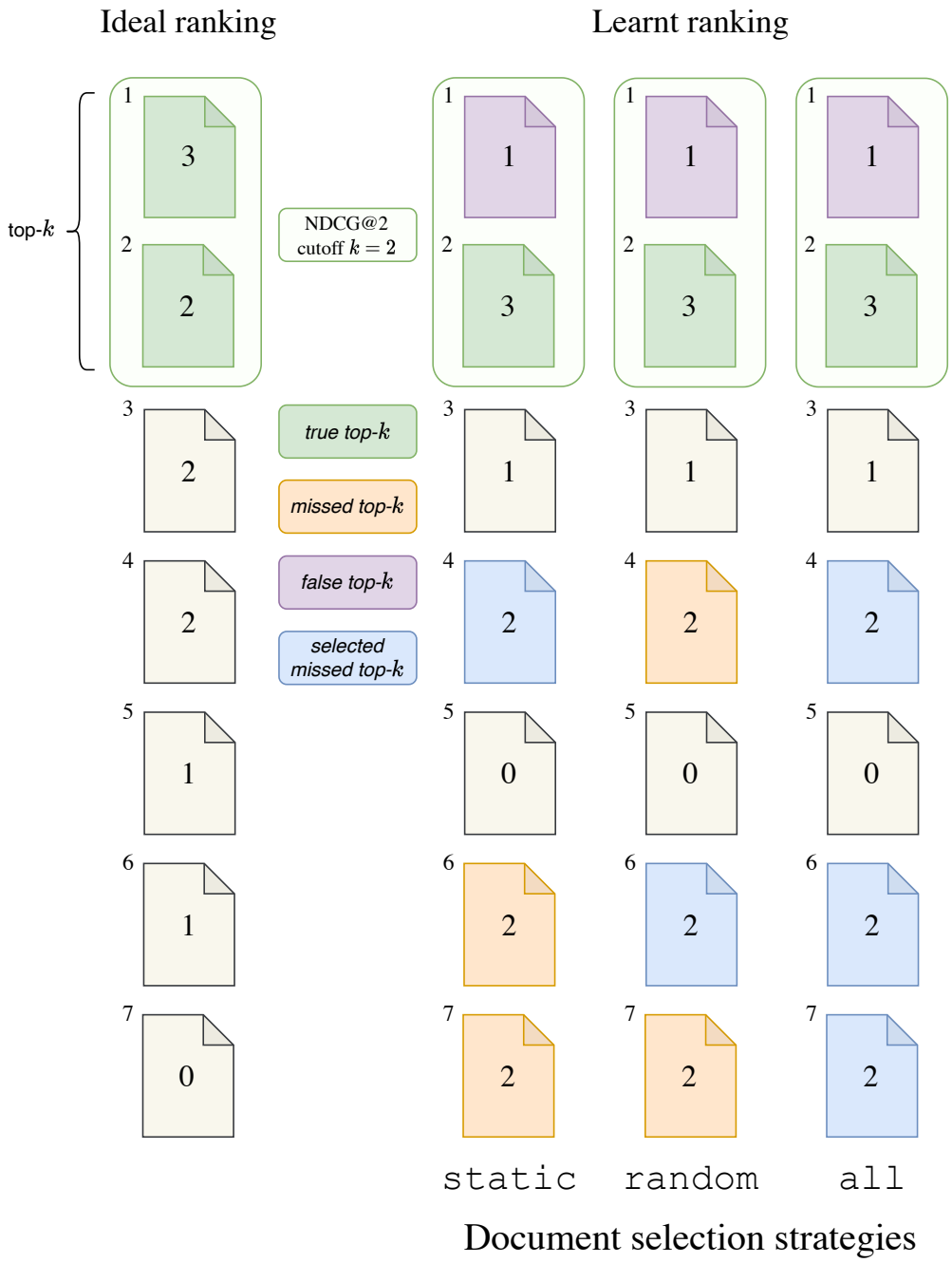


Figure 6.2: Lambda-*eX* *missed top-k* selection strategies.

The above means that the gradient λ_i of a document $d_i \in X$ is computed by considering the λ_{ij} (or λ_{ji}) for every other document $d_j \in D$. This provides a more accurate gradient estimation for the documents in X. We thus remark that

Lambda-eX does not exploit a fixed truncation level but rather selects the set X for each query dynamically. Indeed, the set X may not match any set I_τ for any value of τ. Also, by limiting X such that $|X| \ll |D|$, we have $|I_X| \ll |I_{\tau=+\infty}|$ and achieve a more efficient computation than in the un-truncated case.

To understand how the set X is built, let's first investigate an example from Figure 6.2. The leftmost example in Figure 6.2 shows a set of documents ranked according to their relevance labels. If we desire to maximise NDCG@2, the model must rank the document with a relevance label of 3 in the first position and a document with a relevance label of 2 in the second position. This represents the optimal ranking. Suppose a model produces the rightmost ranking depicted in the same figure; we distinguish documents into three categories. We call *true top-k* a *relevant* document ranked among the top-k and whose label occurs in the top-k of the ideal ranking. This is the case of the document in the second position of the ranking. We call *false top-k* a document ranked among the top-k but whose label is not among the top-k of the ideal ranking. This is the case of the top-ranked document with relevance 1. Finally, we call *missed top-k* a *relevant* document that is not ranked in the top-k but whose label is present among the top-k of the ideal ranking. This is the case of documents with label 2. Note that the above definition is not based on the document identities but rather on their relevance label.

The previous analysis leads us to state that *missed top-k* documents receive an under-estimated gradient, making it impossible for them to climb up to the top ranks. The proposed Lambda-eX aims at improving the learning process by providing a complete and more accurate gradient estimation for the *missed top-k* documents. To do so, Lambda-eX may include in X the documents ranked in the top-k positions by the current model and all the *missed top-k* documents. Since the number of *missed top-k* documents can be large, and we want to limit the size of X to about k, Lambda-eX uses some heuristic criteria to select a subset of the *missed top-k* documents to be included in X. Lambda-eX selection strategies are discussed in the next section.

Note that also Lambda-eX adopts the value $\Delta \mathrm{Z}_{ij}$ with respect to the un-truncated metric. This means that even for a pair of documents d_i and d_j below the cutoff, the value of $\Delta \mathrm{Z}_{ij}$ is not 0. Consequently, if $(i, j) \in I_X$ the partial derivative λ_{ij} will contribute to the gradients λ_i and λ_j.

6.4.2 Selection Strategies

The way Lambda-eX builds the set X is the core of the algorithm. We let k be the cutoff of the IR metric being optimised. First, we include in X all the top-k documents currently ranked by the model. Then, we propose three different ways to select the *missed top-k* documents ranked below the metric cutoff to be used to *extend* X. For the sake of efficiency, the first two strategies generate a set X of

size $|X| \leq 2k$, while the size of X for the third strategy depends on the number of relevant documents in the query. Note that $|X|$ is query-dependent. The three selection strategies are defined as follows.

- `static`: Let h be the number of *false top-k* documents, the `static` strategy includes in X a total of h *missed top-k* documents having the largest scores. In Figure 6.2, the example `static` shows how, among the possible *missed top-k* documents (in orange), it selects the best ($h = 1$) ranked documents below the cutoff among the *missed top-k* documents which have label 2. This strategy focuses on documents closest to the cutoff and thus most likely to fall into the problem explained in Section 6.3.1.

- `random`: Analogous to `static`, but documents are selected randomly instead of rank-based. In example `random`, it randomly selects the second *missed top-k* of relevance 2. A random selection allows the model to see all *missed top-k* documents during training and improves model generalisation.

- `all`: Analogous to `static`, but all the *missed top-k* documents are included in X even if their number is larger than h. In example `all`, every document of relevance label equal to 2 is placed in X. In this case, all *missed top-k* documents are compared simultaneously in favour of greater generalisation of the model at the expense of the efficiency of the gradient computation.

Finally, we also provide two hybrid variants: `all-static` and `all-random`. With efficiency in mind, the goal is to limit the size of X. Depending on the query, the `all` may potentially include all the relevant documents, i.e., with a label greater than 0. To avoid such blow-up of X, the hybrid strategies roll back to either `static` or `random` in this degenerate case, otherwise, they implement the `all` strategy.

In terms of the computational complexity of the gradient computation, the size of X is $k + h$ for the `static` and `random` strategies, and since $h \leq k$, it holds that $|X| \leq 2k$. Therefore, the asymptotic complexity of Lambda-eX with `static` or `random` in computing all the λ_i is $O(kn)$, with n the number of documents in the query. For `all`, `all-static`, and `all-random`, the *missed top-k* documents may correspond to the whole subset of relevant documents. Let n^+ be the number of such documents; the computational complexity is $O((k + n^+)n)$. Note that in general, it is expected that $n^+ \ll n$, meaning a smaller cost than with $I_{\tau=+\infty}$.

6.5 Experimental Setup

The experimental phase aims to verify whether the proposed solution, Lambda-eX, can counteract the exacerbation of gradient incoherencies introduced during the

optimisation of truncated metrics and whether this reduction translates into more effective models. Furthermore, it aims to verify whether the training process with Lambda-eX maintains the same efficiency in terms of training time as a learning algorithm that optimises truncated metrics.

We performed extensive evaluation analysis on five publicly available datasets summarised in Table 6.3. Istella-X has the highest number of documents per query and non-relevant documents, while Istella-F has the highest number of queries. MSLR Web30K Fold 1 is the most balanced since half of the documents are relevant. Instead, Yahoo! Learning to Rank Challenge Set 1 is the smallest one with about 700,000 documents and only an average of 23.73 per query. All datasets have graded relevance labels ranging from 0 to 4, where 0 stands for non-relevant and 4 for highly relevant.

Table 6.3: Datasets properties.

Dataset	#queries	#doc.	$\overline{\text{query len.}}$	%non-rel.
Istella-X	10,000	26,791,447	2,679.14	99.83
Istella-S	33,018	3,408,630	103.24	88.61
Istella-F	33,018	10,454,629	316.63	96.29
Yahoo! Set 1	29,921	709,877	23.73	26.09
MSLR-30K	31,531	3,771,125	119.60	51.47

6.5.1 Baselines and Implementation

It was shown empirically that embedding LambdaRank gradients within Gradient Boosting Decision Trees is more effective and efficient than using LambdaRank gradients in a feed-forward artificial neural network [27]. The implementation of LambdaRank with GBDTs is callded LambdaMART. Therefore, without loss of generality, we perform all experiments and analyses by means of the more effective and efficient LambdaMART.

Below, we provide an overview of the implementation details for each of these algorithms. Note that all algorithms were implemented through the LightGBM library and are available on GitHub[1]. Moreover, each model was trained with gradient normalisation.

Recall that Lambda-eX can be seen as a technique to improve the gradient estimation of already existing learning algorithms such as LambdaMART and the ones derived from the metric-driven loss function designed in LambdaLoss Framework. For this reason, we analyse the effectiveness and efficiency improvements brought

[1]github.com/FedericoMarcuzzi/LambdaRank-Gradients-are-Incoherent

by Lambda-eX by comparing a learning algorithm with and without Lambda-eX set expansion.

Following are the algorithms used in the experimental phase.

- LambdaMART: as a baseline we used the original LambdaMART algorithm optimising three different values of the truncation level τ.
  - LambdaMART$_{\tau=k}$ for which the truncation level τ is set equal to the metric cutoff k. This exactly optimises the truncated metric used for the evaluation.
  - LambdaMART$_{\tau=k+3}$ with $\tau = k + 3$ as suggested in [51]. A τ slightly larger than k provides a better gradient estimation for documents close to the metric cutoff at a minimal cost without deviating too much from the evaluation metric.
  - LambdaMART$_{\tau=+\infty}$ with $\tau = +\infty$ optimises the un-truncated metric, i.e., the IR metric for the whole ranking.
- LambdaMART-eX: to analyse the impact of Lambda-eX techniques we impelmented it on top of LambdaMART; this union is called LambdaMART-eX. Moreover, for each selection strategy suggested in Section 6.4.2, we implemented a different version of LambdaMART-eX.
  - LambdaMART-$eX_{\texttt{static}}$ makes use of the `static` selection stretegy.
  - LambdaMART-$eX_{\texttt{random}}$ makes use of the `random` selection stretegy.
  - LambdaMART-$eX_{\texttt{all}}$ makes use of the `all` selection stretegy.
  - LambdaMART-$eX_{\texttt{all-static}}$ makes use of the `all-static` hybrid selection stretegy.
  - LambdaMART-$eX_{\texttt{sll-random}}$ makes use of the `all-random` hybrid selection stretegy.

Moreover, as mentioned above, gradient incoherencies also affect LambdaRank derivatives, such as the metric-driven loss functions designed in LambdaLoss Framework, (e.g., NDCG-Loss1, NDCG-Loss2, and hybrid loss NDCG-Loss2++). To this end, we also investigate whether extending the set of document pairs is beneficial for these loss functions. We focus the analysis on NDCG-Loss2++, which is the loss function shown to achieve higher performance among the others in [173]. For the sake of clarity, hereinafter, we refer to the learning algorithm that makes use of NDCG-Loss2++ loss function as LambdaLoss. To assess the impact of Lambda-eX on the LambdaLoss algorithm, we also performed the same experiments designed for LambdaMART with LambdaLoss. We also implemented all the

algorithms defined above using the LambdaLoss algorithm. So, we compare the five versions of LambdaLoss-eX (i.e., one for each selection strategy) with the following three baselines: LambdaLoss$_{\tau=k}$, LambdaLoss$_{\tau=k+3}$, and LambdaLoss$_{\tau=+\infty}$

6.6 Main Results

The primary results of this work aim to address two questions: can the use of the Lambda-eX technique yield more effective models than those trained using the original learning algorithms? Can models trained with Lambda-eX maintain the training efficiency guaranteed by truncated metric optimisation? For this reason, we divided the main results into two sections, one on effectiveness and the other on efficiency.

6.6.1 Effectiveness

We evaluate the models' effectiveness in terms of NDCG@k for different cutoff values: 5, 10, and 15. For each model, we stop the training process after 1,000 trees and select the best iteration based on the performance achieved on the validation set. The best hyperparameters used to train each model are reported in Section 6.7.1 Statistically significant improvements with respect to the baselines LambdaMART$_{\tau=k+3}$ and LambdaLoss$_{\tau=k+3}$ were computed according to Fisher's randomisation test [65] with a two-sided p-value. Statistically significant improvement are marked with *italic** for $\mathsf{p} = 0.05$ and **bold**** for $\mathsf{p} = 0.01$.

We choose these as reference baselines since both perform mostly better than models trained with $\tau = k$ and slightly worse than $\tau = +\infty$ but with a much lower training cost of the latter. Results are summarised in Table 6.4 for LambdaMART-based algorithms and in Table 6.5 for LambdaLoss-based algorithms. In the following sub-section, we evaluate the computational cost of the discussed methods.

First of all, we highlight that the observations drawn from the results are mostly the same for both LambdaMART and LambdaLoss.

Interestingly, models trained with $\tau = +\infty$ achieve the best NDCG@k values across datasets, especially datasets from the Istella family. The only performance drops occur with LambdaLoss learning algorithm on MSLR-30K and Yahoo! Set 1. In these cases, the performance of LambdaLoss is clearly worse than that of LambdaMART. The NDCG scores obtained with $\tau = +\infty$ might seem surprising as the target metric is not optimised; however, this highlights the effect of the gradient incoherencies intruded by $\tau = k$ and $\tau = k + 3$.

These results are interesting as they demonstrate how gradient incoherencies conflict with the empirical risk minimisation principle, which suggests that optimising the evaluation metric should lead to higher effectiveness. In the presence

Table 6.4: Effectiveness in terms of NDCG (percentage). Statistically significant improvements w.r.t. $\tau = k + 3$ according to Fisher's randomisation test [65] (with a two-sided p-value) are marked with *italic* * ($\mathsf{p} = 0.05$) and **bold** ** ($\mathsf{p} = 0.01$).

	LambdaMART			LambdaMART-*eX*				
Dataset	$\tau = k$	$\tau = k+3$	$\tau = +\infty$	`static`	`random`	`all`	`all-stt`	`all-rnd`
				NDCG@5				
Ist-X	73.32	74.11	**75.35** **	**75.19** **	**75.17** **	**75.15** **	**75.19** **	**75.17** **
Ist-S	70.19	70.42	*70.64* *	**70.67** **	**70.71** **	70.55	*70.65* *	*70.64* *
Ist-F	67.02	67.24	**67.62** **	**67.55** **	**67.67** **	**67.50** **	**67.68** **	**67.71** **
Yah 1	75.35	75.59	**75.85** **	75.67	75.59	75.63	75.73	75.67
MS 30	50.66	51.15	51.22	50.95	50.96	51.24	*51.42* *	51.38
				NDCG@10				
Ist-X	77.53	78.55	78.61	78.61	78.61	78.61	78.61	78.61
Ist-S	76.35	76.48	**76.71** **	**76.66** **	**76.70** **	**76.69** **	**76.72** **	**76.70** **
Ist-F	71.85	72.07	**72.39** **	**72.42** **	**72.46** **	**72.42** **	**72.35** **	**72.46** **
Yah 1	79.62	79.78	79.84	79.66	79.75	79.78	79.81	79.80
MS 30	52.66	53.02	52.98	52.96	53.08	*53.23* *	53.19	53.14
				NDCG@15				
Ist-X	79.00	79.29	79.45	79.44	79.48	79.44	79.44	79.48
Ist-S	80.63	80.59	*80.73* *	80.69	*80.71* *	**80.75** **	**80.80** **	*80.73* *
Ist-F	75.46	75.56	**75.87** **	**75.94** **	**75.90** **	**75.92** **	**76.00** **	**76.00** **
Yah 1	82.01	81.96	82.03	81.94	82.07	82.04	82.07	82.04
MS 30	54.60	54.72	54.67	54.82	**54.93** **	54.84	54.75	54.83

of gradient incoherencies, considering all document pairs provides a more accurate gradient estimation rather than focusing solely on those maximising the truncated metric. However, as demonstrated in the next section, this comes at a non-trivial computational cost.

Models trained with Lambda-*eX* provide very interesting results. The scored NDCG@k values are most of the times statistically significantly better than those of the baselines with $\tau = k$ and $\tau = k + 3$, and never statistically worse. Furthermore, they often achieve the same performance as $\tau = +\infty$.

Overall, the different Lambda-*eX* variants implemented for each selection strategy achieve similar performance. The `random` variant seems to achieve statistical improvements in most of the experiments, while `all` and `all-static` in a few specific cases. However, the `random` strategy is preferable due to its lower computational complexity.

From the results in Tables 6.4 and 6.5, two interesting considerations can be drawn about our strategy. Lambda-*eX* is most effective in datasets like the Is-

Table 6.5: Effectiveness in terms of NDCG (percentage). Statistically significant improvements w.r.t. $\tau = k + 3$ according to Fisher's randomisation test [65] (with a two-sided p-value) are marked with *italic** ($\mathsf{p} = 0.05$) and **bold**** ($\mathsf{p} = 0.01$).

	LambdaLoss			LambdaLoss-eX				
Dataset	$\tau{=}k$	$\tau{=}k{+}3$	$\tau{=}{+}\infty$	`static`	`random`	`all`	`all-stt`	`all-rnd`
				NDCG@5				
Ist-X	74.08	74.22	**75.40****	**75.33****	**75.19****	**75.14****	**75.33****	**75.19****
Ist-S	69.97	70.50	**71.11****	**70.92****	**70.92****	**70.97****	**70.95****	**71.02****
Ist-F	66.97	67.56	**68.18****	**68.08****	**68.26****	**68.24****	**68.07****	**68.18****
Yah 1	75.44	75.69	74.96	75.74	75.81	75.84	75.74	75.86
MS 30	50.77	51.04	49.19	50.99	50.92	51.05	51.10	51.08
				NDCG@10				
Ist-X	77.91	78.35	78.74	**78.94****	*78.77**	**78.94****	**78.94****	*78.77**
Ist-S	76.63	76.89	**77.37****	**77.26****	**77.43****	**77.23****	**77.44****	**77.28****
Ist-F	72.18	72.53	**73.17****	**73.21****	**73.20****	**73.16****	**73.12****	**73.15****
Yah 1	79.63	79.68	79.19	**79.94****	**79.94****	**79.98****	**79.93****	*79.89**
MS 30	52.89	53.08	51.36	52.99	52.98	52.95	53.02	52.99
				NDCG@15				
Ist-X	78.81	79.06	**79.73****	*79.60**	**79.78****	*79.60**	*79.60**	**79.78****
Ist-S	80.96	81.15	81.29	*81.31**	**81.40****	**81.38****	**81.38****	**81.38****
Ist-F	75.97	76.24	**76.74****	**76.85****	**76.85****	**76.83****	**76.82****	**76.85****
Yah 1	81.88	81.95	81.50	*82.15**	**82.16****	82.09	82.09	82.09
MS 30	54.68	54.60	53.23	54.63	54.65	54.64	54.69	54.62

tellas which contain many non-relevant documents. As mentioned above, many non-relevant documents increase the chance of incoherencies, which are successfully managed by Lambda-eX. We can conclude that Lambda-eX finds its best application in datasets with many non-relevant documents. This is particularly appealing in realistic scenarios where there are far fewer documents relevant to a query than non-relevant ones.

The second interesting consideration is that as the metric cutoff k increases, the performance gap between Lambda-eX and the baselines decreases. The reason behind it is straightforward. Recall that in the experiments, we fixed τ equal to the cutoff k and to the slightly larger value $k + 3$. Thus, with a small cutoff, the probability of a relevant document being ranked below the truncation level is high; consequently, many of its λ_{ij} are discarded. As the cutoff increases, relevant documents are more likely to be ranked above a larger truncation level, and therefore, their gradient is fully computed.

Table 6.6: Efficiency in terms of training time (milliseconds). For truncated-optimisation $k = 5$, i.e., evaluation metric NDCG@5.

Dataset	LambdaMART $\tau=k$	LambdaMART $\tau=k+3$	LambdaMART $\tau=+\infty$	LambdaMART-eX stt	rnd	all	all-stt	all-rnd
	training time per objective function							
IST-X	89	100	2574	115	132	118	113	130
IST-S	12	16	31	17	20	18	18	22
IST-F	30	38	119	43	49	46	45	52
YAH 1	3	4	12	4	6	6	6	7
MS 30	4	5	34	5	7	9	8	10
	training time per tree							
IST-X	672	751	3253	792	815	795	796	813
IST-S	156	122	143	129	129	130	130	133
IST-F	263	288	380	306	308	301	299	307
YAH 1	207	205	215	208	294	287	209	212
MS 30	291	298	388	324	317	326	314	334

6.6.2 Efficiency

In the previous section, we showed how Lambda-eX is able to achieve statistically significant improvement in terms of NDCG both on LambdaMART and LambdaLoss when optimising truncated metrics. However, to be a valid alternative to the algorithms optimising the un-truncated metrics, shown to be the most effective, it needs to be more efficient.

To do so, we measure the average training time spent by the LightGBM library in training a tree and in executing the objective function (i.e., computing the gradient for each document for each query) while training a single tree. We run this analysis with fixed $k = 5$ (i.e., evaluation metric NDCG@5), affecting the training efficiency of LambdaMART$_{\tau=k}$, LambdaMART$_{\tau=k+3}$ and the three LambdaMART-eX variants. Results are reported in Table 6.6.

Note that since the difference between LambdaMART and LambdaLoss relies only on the discounting factor in the ΔNDCG_{ij}, the computational cost of the two algorithms is the same. For this reason, we performed the efficiency analysis only on LambdaMART, but the results generalise also for LambdaLoss.

The average execution time of an iteration of LambdaMART-eX's objective function aligns with the one of LambdaMART$_{\tau=k}$ and LambdaMART$_{\tau=k+3}$. The reason behind this relies on a similar asymptotic complexity. As expected, LambdaMART$_{\tau=+\infty}$ is the one that spends more time executing the objective function since it has to process $O(n^2)$ document pairs. Note that the cost of Lamb-

daMART$_{\tau=+\infty}$ can be up to 30 times larger than the other competitors.

The difference in execution time of the objective functions only affects the training of a tree with long results lists. This can be seen with ISTELLA-X which tree learning time increases from 792 milliseconds with LambdaMART-$eX_{\texttt{static}}$ to 3,253 milliseconds with LambdaMART$_{\tau=+\infty}$. This is a 4× slowdown that happens for each of the 1,000 trees of the model trained.

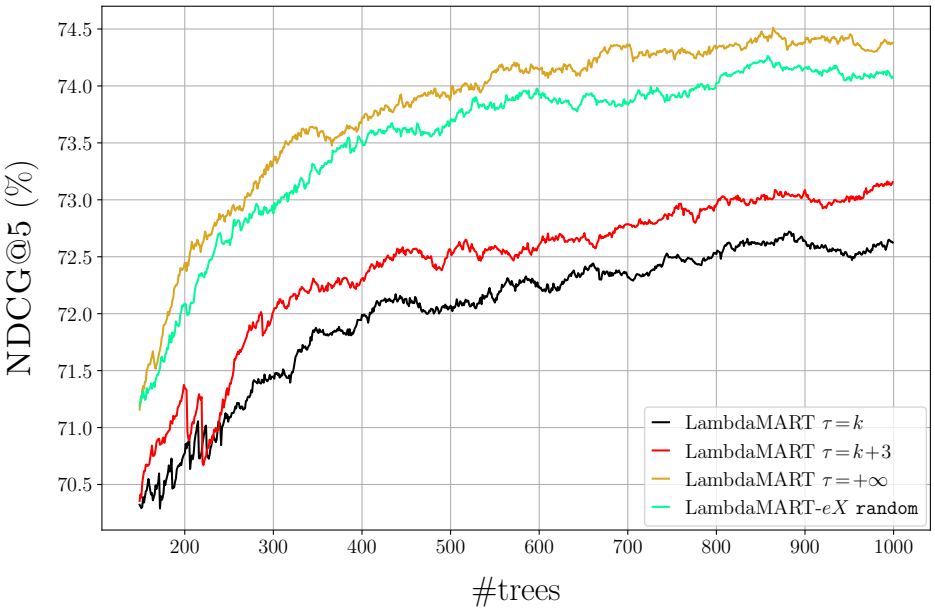


Figure 6.3: Models performance on ISTELLA-X validation set.

Another interesting observation in favour of LambdaMART-eX's efficiency is that it manages to achieve the same performance as LambdaMART$_{\tau=k}$ and LambdaMART$_{\tau=k+3}$ with far fewer trees. In Figure 6.3, the model trained with LambdaMART-$eX_{\texttt{random}}$ achieves the same performance as LambdaMART$_{\tau=k}$ with about 250 trees in ISTELLA-X, and the same performance as LambdaMART$_{\tau=k+3}$ with 300 trees. The difference in model size significantly reduces training time, even though the average time taken to train a single tree is similar. The same behaviour was observed for all the Lambda-eX variants. Furthermore, an interesting side effect of a smaller ensemble size translates into an increase in efficiency also at the operational phase, where an instance has to traverse a few trees.

In conclusion, LambdaMART-eX can reduce the training time compared to LambdaMART$_{\tau=k}$ and LambdaMART$_{\tau=k+3}$ by training equally effective models with far fewer trees, and compared to LambdaMART$_{\tau=+\infty}$ especially when training datasets with a high average number of documents per query. This result generalises to other learning algorithms such as LambdaLoss.

6.7 In-Depth Analysis

In the previous section, we demonstrated that Lambda-eX allows training effective and efficient models. The capability of Lambda-eX to learn more effective models compared to a learning algorithm that optimises truncated metrics is attributed to the reduction in the number of gradient incoherencies. However, does Lambda-eX effectively reduce these incoherencies? To answer this question, we conducted an in-depth analysis of the number of gradient incoherencies affecting the learning algorithms. However, before delving into the analysis, let's briefly describe the hyperparameters used to obtain the models whose results are reported in the previous tables.

6.7.1 Hyperparameter Analysis

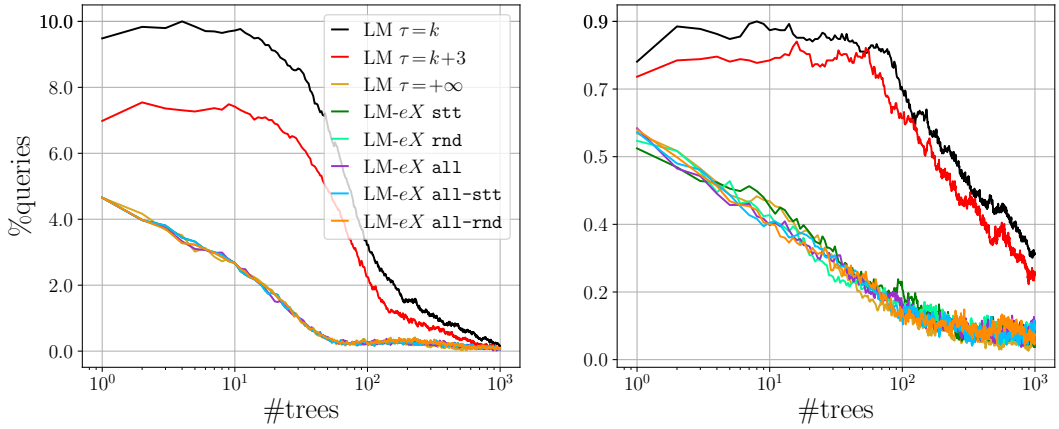
In Table 6.7 are summarised the hyperparameters used for both LambdaMART and LambdaLoss learning algorithms. The best hyperparameters were found through hyperparameter tuning on the validation set. Furthermore, for each dataset, the maximum value of max_bin is set to 255. The weight coefficient μ used in the hybrid loss function NDCG-Loss2++ is set to 5, while for models trained with Lambda-eX on MSLR-30K and Yahoo! Set 1, the best value is 0.5. Recall that the hyperparameter μ manages the trade-off between the discount ρ_{ij} of LambdaMART and δ_{ij} of NDCG-Loss2, i.e., $\rho_{ij} + \mu\delta_{ij}$.

Table 6.7: Hyperparameters per dataset.

Dataset	learning_rate	num_leaves	min_data	min_hessian
Istella-X/S/F	0.05	64	20	0.001
Yahoo! Set 1	0.02	200	100	0
MSLR-30K	0.02	400	50	0

6.7.2 Incoherency Reduction

Another important question we wanted to answer in this work is: Does Lambda-eX reduce the number of gradient incoherencies? To answer this question, we analyse the number of queries affected by gradient incoherencies during the training process. We performed this analysis for LambdaMART-based models (LM). In Figure 6.4, we report, for each tree of the trained forest, the number of queries encompassing at least one violation of Definition 4 when optimising NDCG@k with $k = 5$. We restrict our attention to the most harmful violations where a *false top-k* document d_i gets a larger gradient than a *missed top-k* document d_j ranked



(a) Istella-X (b) MSLR-30K

Figure 6.4: percentage of queries (y-axis) affected by at least one gradient incoherence during each tree learning (x-axis). The scale in the x-axis is logarithmic.

below k. We report the results of this analysis for both Istella-X (Figure 6.4a) and MSLR-30K (Figure 6.4b).

Results show that using a truncation level $\tau = k$ generates the largest amount of incoherencies in the early stage of the training process, involving about 10% of the queries of Istella-X after 10 trees and 4% after 100 trees. However, the number of incoherencies falls down significantly towards the end of the forest. The initial trees of the forest are strongly affected by gradient incoherencies, which are mostly solved afterwards. Similar behaviour is exhibited by LambdaMART$_{\tau=k+3}$, with fewer incoherencies overall. For Istella-X, this behaviour was expected since 99.83% of its documents have relevance labels equal to 0, and, as explained in Section 6.3.1, this increases the chance of incoherencies.

The best behaviour is obtained by LambdaMART$_{\tau=+\infty}$ with the number of queries affected by gradient incoherencies that quickly fall to about 2% after 10 trees. A similar trend is for MSLR-30K dataset. This confirms that discarding some of the λ_{ij} values generates a large number of incoherencies, both in LambdaMART$_{\tau=k}$ and LambdaMART$_{\tau=k+3}$.

Interestingly enough, all the five variants of LambdaMART-eX have the same behaviour of LambdaMART$_{\tau=+\infty}$. This confirms that the proposed Lambda-eX succeeds in limiting the number of incoherencies, as with LambdaMART$_{\tau=+\infty}$ where all the λ_{ij} are considered, but without performing all the pairwise comparison of documents.

A final interesting consideration can be made by observing again Figure 6.3 where the effect of having fewer incoherencies at the beginning of the training phase translates into more effective models already in the first trees of the ensemble.

The first 300 trees of LambdaMART-$eX_{\texttt{random}}$ perform as well as 1,000 trees of LambdaMART$_{\tau=k+3}$.

6.8 Summary

In this section, we provide a comprehensive overview of the main results and contributions of the study introduced in this Chapter titled "LambdaRank Gradients are Incoherent".

- **Gradient Incoherencies:** In this work, we discovered a notable issue concerning LambdaRank and its derivatives (e.g., LambdaLoss and LambdaMART); they are affected by gradient incoherencies. A gradient incoherency occurs when a document with higher relevance in the ground truth receives a smaller gradient push than a document with lower relevance. These gradient inconsistencies prevent the learning algorithm from effectively learning the optimal rankings for training queries. Moreover, we discovered that these incoherencies become more pronounced when optimising truncated metrics. As a result, the efficiency gains brought by truncated metric optimisation come at the expense of a reduction in model effectiveness.

- **Contribution:** In this work, we provided Lambda-eX, a technique to optimise the truncated metric while maintaining the training efficiency of truncated metric optimisation and the effectiveness of un-truncated optimisation. Lambda-eX succeeds in both aims by expanding the set of documents for which to compute a complete gradient estimation. Moreover, we provided five versions of Lambda-eX, each with a different paradigm in the document selection.

- **Main Results:** Through extensive experiments, we showed that Lambda-eX allows LambdaRank-based learning algorithms such as LambdaMART and LambdaLoss to achieve statistically significant improvement in terms of NDCG@k with respect to models trained to directly optimise the target metric while maintaining the same efficiency in terms of training time. Finally, when Lambda-eX is used to train models on datasets with a high average number of documents per query, it is able to achieve the same performance as the un-truncated optimisation while significantly reducing the training time, e.g., about 40 minutes difference in Istella-X.

- **In-Depth Analysis:** Through specific analysis, we have demonstrated that the use of Lambda-eX effectively mitigates the exacerbation of gradient incoherencies introduced by truncated optimisation, resulting in a number of incoherencies comparable to that achieved with non-truncated optimisations.

6.8.1 Future Work

In this final section, we present potential avenues for future research and extensions based on the results obtained in this study.

- **Eliminate All Gradient Incoherencies:** In this Chapter, we introduced Lambda-eX, a novel technique aimed at mitigating the exacerbation of gradient incoherency resulting from truncated metric optimisation while preserving training efficiency. However, the phenomenon of gradient incoherency is only attenuated and not entirely eliminated. So, the research problem left unsolved in this work is how to extinguish gradient incoherencies definitively. A possible path to reach this goal is to explore and provide more sophisticated heuristics to select the most effective sets of documents, aiming to eliminate incoherencies while preserving computational efficiency. Alternatively, defining a new ΔZ_{ij} to allow any LambdaRank-based algorithm to optimise the evaluation metric without generating incoherencies.

- **Other Learning Algorithms:** Additional pursuits of this research concern discovering the presence of gradient incoherencies in other learning algorithms not based on LambdaRank. Understanding how these incoherencies manifest and influence the learning phase of such algorithms is another critical goal. Additionally, it would be interesting to elucidate how Lambda-eX, despite its design for LambdaRank-based learning algorithms, can be an effective technique also for these algorithms, potentially alleviating the impact of gradient incoherencies and thus improving their overall performance.

- **Theoretical Formalisation:** The solution proposed in this work aims to resolve gradient incoherencies through a heuristic; moreover, the gradient incoherency phenomenon lacks a theoretical formalisation explaining exactly why they occur. A deeper theoretical analysis of gradient incoherencies and their impact on the learning process could provide valuable insights. Formal models and theoretical frameworks could be developed to understand better the conditions under which gradient incoherencies occur and how they affect the learning process.

- **Integration with Online Learning:** Lambda-eX's ability to be efficient during training time can be beneficial in an online learning scenario. Furthermore, as shown in Section 6.6.2 in Figure 6.3, the ability of Lambda-eX to reduce the number of gradient incoherencies allows it to learn very effective models already at the beginning of the training process, providing an even more appealing tool for online learning, where robust and fast training are crucial.

Chapter 7

Discussion First Part

Part I encompasses the research works in the domain of learning to rank achieved during my doctoral studies. This chapter summarises our main effort in designing data-aware learning algorithms to create more effective and efficient models and outlines interesting observations and future work directions.

In Chapter 4, we provided the definition of *consistent outliers* [121], which refers to documents consistently misranked by the model during the training process. We designed SOUR [121], a learning algorithm that detects and removes consistent outliers within the training set. Through this work, we found that removing these consistent outliers from the training set allows us to learn rankers with a statistically significant improvement in effectiveness compared to models trained on the entire training set without affecting the training efficiency.

In Chapter 5, we provided High_Low_Sampl [110], a novel document selection strategy for Selective Gradient Boosting framework. This strategy efficiently selects relevant and non-relevant documents that contribute the most to the learning process while eliminating superfluous or detrimental documents that could negatively impact learning quality. Our selection strategy addresses the limitations of the original Sel_Sampl selection strategy presented alongside the Selective Gradient Boosting framework. Specifically, High_Low_Sampl includes the lowest-ranked non-relevant documents in the training set, which were previously excluded by Sel_Sampl. Including the lowest-ranked non-relevant documents enhances training stability and reduces variance in document ranking positions during training, resulting in more effective rankers.

Furthermore, in Chapter 6, we discovered that algorithms based on the well-known LambdaRank learning algorithm are affected by what we refer to as *gradient incoherencies*. A gradient incoherency occurs when a misranked document receives an upward/downward push smaller than those with lower/higher relevance labels. The push the documents receive is the gradient force of the optimised loss function. Consequently, these incoherencies prevent the learning algorithm from learning the

optimal ranking. Moreover, we discovered that optimising truncated metrics to align with empirical risk minimisation and improve model efficiency exacerbates the problem of gradient incoherencies, resulting in less effective models than those optimising the entire metric. To address this exacerbation phenomenon, we introduced Lambda-eX, which eliminates the exacerbation of gradient incoherencies introduced by truncated metric optimisation while preserving its advantages.

As anticipated in the introductory section of this thesis, the contribution we aimed to bring to the LtR domain lies in algorithms that are more aware of the input data used for learning. To this aim, the algorithms presented in Chapter 4 (i.e., SOUR) and Chapter 5 (i.e., HIGH_LOW_SAMPL), it is clear how these algorithms achieve this goal. The former removes harmful documents (i.e., consistent outliers) from the training set; instead, the latter selectively chooses the most useful documents from the training set (i.e., positive documents, highest-ranked non-relevant documents, and lowest-ranked non-relevant documents). Both algorithms actively influence the learning process by modifying the training set, creating a higher-quality training set.

The work presented in Chapter 6 focuses on Gradient Incoherencies [122] and seems to deviate from the objective of this thesis, i.e., to make learning algorithms more data-aware regarding the input data. In fact, Lambda-eX [122] does not modify the training set and performs the training on the untouched input set. Despite that, Lambda-eX is a data-aware algorithm. The Lambda-eX data awareness regarding the training set shifts more toward a concept of greater efficiency than greater effectiveness. Experimental results demonstrated that achieving effective models does not necessarily require comparing every document within a query to all others; instead, only a small subset of documents requires a complete comparison. Lambda-eX aims to create a subset of documents within the training set for complete comparison to allow the learning algorithm to produce models that are as effective as those that perform complete comparison for each document but with a more efficient training phase. Thus, conscious interaction with the training data leads to an advantage in the LtR domain by producing models of equal effectiveness but with better efficiency in training time. Finally, we showed how Lambda-eX allows training models to be equally effective as those trained with truncated optimisations but with a considerable reduction in forest size (see Figure 6.3), resulting in an improvement in model efficiency in the operational phase.
We want to conclude this Chapter with an important consideration and a global analysis of the work we did in the field of LtR. The research discoveries we made in this first part of the thesis are presented chronologically. An idea we matured through the last published work, that of Chapter 6 (i.e., gradient incoherencies and Lambda-eX), could be the primary avenue for future research starting from this thesis.

The research question we pose is the following: Does the increase in effectiveness observed in rankers trained with learning algorithms employing selection, sampling, or training set cleaning strategies (such as Selective Gradient Boosting, SOUR, etc.) result from an incidental reduction in gradient incoherencies caused by the removal of problematic documents?

This question, for which we have not yet provided an answer, stands as a significant area for future work based on this thesis, and it is raised from the following two considerations:

First, when SOUR is used to remove consistent outliers, removing those misranked documents might equate to removing those documents that, due to gradient incoherencies, never achieve the correct ranking. Indeed, in Section 6.3.1 and in Figure 6.1, we already proved that removing, altering, or introducing documents might introduce or eliminate gradient incoherencies. For example, the consistent positive outliers in Figure 4.1 that can be seen as the document $d_{\triangle}$ in Figure 6.1(a), misranked by the optimisation process due to the contribution that a less relevant document receives from the documents with relevance label equal to 0. Removing it from the training set may lead to a more coherent optimisation process.

Second, we can make an even stronger argument in the context of selection strategies like Sel_Sampl and High_Low_Sampl, where removing some negative documents from the training set can significantly improve effectiveness. Let's draw a parallel between Lambda-eX and High_Low_Sampl applied on the Istella-X dataset, which contains 99.83% of documents with relevance 0. In Chapter 6, we demonstrated how the presence of many non-relevant documents in the training set significantly increases the occurrence of gradient incoherencies. In the same Chapter, we showed that removing these incoherencies in such datasets leads to the highest effectiveness improvement. At the same time, we showed in Chapter 5 how the sole selection of 1% of non-relevant documents by Sel_Sampl from the training set of Istella-X produces the highest increase in effectiveness among all the datasets and learning algorithms used in those experiments. We can conclude that selecting only 1% of the non-relevant documents, i.e., discarding (removing) 99% of documents with relevance 0, is exactly like removing the contributions received by document $d_{\star}$ from the non-relevant document below k, that create gradient incoherencies and prevent document $d_{\triangle}$ from reaching the top position. So, document selection strategies like Sel_Sampl and High_Low_Sampl might reduce the likelihood of gradient incoherency occurrence.

Finally, note that not only the removal of documents but also the addition or modification of some documents and their relevance labels allows the removal of incoherencies. Therefore, similar considerations can be made to data augmentation or undersampling strategies.

Part II

Robust Learning Algorithms for Classification

Chapter 8

Background and State of the Art

This Chapter provides an overview of the background and the state of the art in the context of Robust Learning Algorithms for classification. It primarily focuses on Adversarial Machine Learning (AML), especially for learning algorithms based on decision trees or forests applied to classification tasks. The Chapter introduces the concepts of *Attack*, *Attacker*, and the type of attacks. Specifically, it focuses on the type of attacks known as *evasion attacks*. The Chapter provides the most common performance metrics for evaluating machine learning models in an adversarial scenario. Then, it explores state-of-the-art strategies to enhance the model's robustness against evasion attacks and verify or certify the robustness of machine learning models. Furthermore, the Chapter presents the datasets used in the experimental phase related to this part of the thesis.

8.1 Adversarial Machine Learning

Adversarial Machine Learning (AML) is a dynamic and evolving field that resides at the intersection of machine learning and cybersecurity. It explores the vulnerabilities of machine learning models and their susceptibility to adversarial attacks. In essence, AML investigates the ways in which malicious actors can manipulate or deceive machine learning systems, with potentially far-reaching consequences.

The first reference to Adversarial Machine Learning dates back to 2004, in the work by Dalvi *et al.* [54]. Nowadays, it is a crucial point to consider during the development of Artificial Intelligence systems. As machine learning algorithms increasingly find applications in critical domains such as finance, healthcare, and autonomous vehicles, the need for robust and secure models has never been more pressing. AML seeks to address the inherent weaknesses of these algorithms when confronted with adversarial inputs and to develop strategies that can safeguard against these threats.

Adversarial attacks can take various forms, but one common category is evasion attacks, where adversaries intentionally manipulate input data to mislead machine learning models into making incorrect predictions or classifications. The consequences of such attacks can be severe, impacting everything from spam filters to image recognition systems and autonomous vehicles.

To mitigate these risks, AML researchers delve into understanding adversarial attack strategies to develop robust machine learning models and devise methods to certify their security and robustness. The continuous struggle between those exploiting vulnerabilities and those strengthening machine learning systems makes adversarial machine learning an engaging and crucial field of study with significant implications for the future of AI and cybersecurity.

8.1.1 Attack Taxonomy

To better understand how AML poses a risk to machine learning models, it is essential to provide a taxonomy of attacks that can be perpetrated within the machine learning context. The literature offers a taxonomy of attacks on systems utilising machine learning based on three fundamental features: the *influence* that the attack has on the system, the type of *security violation* caused by the attack, and the *specificity* of the attack [85, 12, 14, 6].

The feature *influence* is divided into two types of attack: **causative** and **exploratory**.

- **causative:** A causative attack is an adversarial attack where the adversary interferes with the learning phase by altering the training data used to train a machine learning model. A causative attack aims to manipulate the model's behaviour by injecting malicious or perturbed data into the training set. This can decrease the model's effectiveness, making it more susceptible to misclassification or other security breaches. Causative attacks focus on influencing the model's training process.

- **exploratory:** An exploratory attack is an adversarial attack where the adversary does not alter the training data but instead attempts to gain insights or extract information from an already trained machine learning model. The attacker seeks to understand the model's vulnerabilities, decision boundaries, or internal parameters without modifying the learning algorithm or the training data. This information can then be used to craft adversarial examples or inputs that deceive the model without changing the model itself. Exploratory attacks focus on understanding and exploiting the existing model's weaknesses.

The feature *security violation* is divided into three types of violation: **integrity**, **availability** and **privacy** violation.

- **integrity violation:** Integrity violation refers to a type of security breach where an attacker attempts to deceive the machine learning model into an incorrect classification of adversarial instances as legitimate or benign. In other words, the attacker's goal is to compromise the integrity of the model's predictions by deceiving it into making incorrect or undesired classifications. This type of attack can lead to significant vulnerabilities in applications such as intrusion detection systems or malware detection, where the model's integrity is compromised.

- **availability violation:** Availability violation occurs when an attacker disrupts or compromises the normal functioning of a machine learning system by forcing it to make a large number of errors or misclassifications. This type of attack is similar to a denial-of-service (DoS) attack, where the primary objective is to render the machine learning service unavailable to legitimate users. Availability violation can be achieved by overwhelming the model with adversarial inputs, causing it to fail or produce unreliable results, thus affecting its availability for intended users.

- **privacy violation:** Privacy violation in adversarial machine learning pertains to attacks that allow an adversary to infer confidential or sensitive information about users or individuals from the model's responses. For example, in a privacy violation attack, the adversary might exploit the model's behaviour to glean sensitive data, such as biometric information, demographic details, or personal preferences, even when such data should remain confidential. This type of attack can have serious privacy implications and is a significant concern in applications involving personal data and sensitive information.

Finally, the last attack feature is *specificity*, which can be **targeted** or **indiscriminate**.

- **targeted:** In targeted attacks, adversaries have a specific goal in mind. They aim to manipulate the machine learning model's behaviour towards particular instances or a restricted set of instances. These attacks are customised to exploit vulnerabilities in the model, with the goal of achieving a particular outcome. Targeted attacks are often designed to mislead the model into misclassifying specific data points, and they require a deep understanding of the model's decision-making process.

- **indiscriminate:** Indiscriminate attacks are not tailored to specific instances. The objective of an indiscriminate attack is to disrupt or confuse the model's classification process more broadly. These attacks seek to create a general impact, such as increasing misclassifications across various classes or instances, without focusing on a particular set of targets. Indiscriminate attacks exploit generic vulnerabilities in the model's architecture or training data, making them a broader threat.

8.1.2 Adversary's Model

Biggio *et al.* in [16, 14] introduced an attacker model, which can be divided into four key facets. Firstly, the *Adversary's Goal*: what the attacker intends to target and the desired outcomes. Secondly, the *Adversary's Knowledge*: the knowledge the attacker has with the target system, encompassing their level of insight. Thirdly, the *Adversary's Capability*: the "weapons" the attacker has to perform the attack. Lastly, the *Adversary's Strategy*: how the attacker decides to attack the machine learning system.

8.1.2.1 Adversary's Goal

Based on the taxonomy provided in Section 8.1.1, the attacker's goal is defined upon the three features: influence, security violation, and specificity. Specifically, the first feature determines at which stage of the machine learning system's lifecycle the attack will be executed: during the learning phase (causative) or the operational phase (exploratory). The second feature identifies the nature of the violation the attacker intends to inflict on the system. Lastly, the third feature identifies the portion of the instance set on which to perform the attack (a restricted subset or the whole set).

The primary objective of the attacker's goal is to maximise the extent of damage inflicted on the system; therefore, it can be formulated as an optimisation problem for identifying the optimal attack strategy. Table 8.1 highlights the relationship between the influence, specificity, and security violation features [6, 120].

8.1.2.2 Adversary's Knowledge

The adversary's knowledge about the model is defined by the amount of information it has on how the model was trained. This information is the dataset used to perform the training, how instances are represented in features, which learning algorithm is used, which decision function is used, the model parameters and finally, the output (or feedback) returned by the model in the operational phase.

Table 8.1: Adversary's goal summary based on features influence, specificity, and security violation. Due to space constraints, the values of the specificity feature, Targeted and Indiscriminate, are referred to as T and I, respectively.

		Integrity	Availability	Privacy
Causative	T	Permit a specific intrusion	Create sufficient errors to make system unusable for one person or service	Sign into the system as a specific person
	I	Permit at least one intrusion	Create sufficient errors to make learner unusable	Sign into the system as an arbitrary person
Exploratory	T	Find a permitted intrusion from a small set of possibilities	Find a set of points misclassified by the learner	Access information about a specific person
	I	Find a permitted intrusion		Access information of an arbitrary person

Based on the amount of information the attacker has on the system, it is possible to define three main attack scenarios as reported by Biggio and Roli in [16]:

White-Box Scenario In a white-box scenario (also known as a perfect-knowledge scenario), the adversary has complete knowledge of the target machine-learning model. This means the attacker is aware of the model's architecture, parameters, training data, and internal workings. With this detailed insight, the adversary can carefully craft and execute attacks with a deep understanding of how the model functions. This scenario allows the adversary to perform highly effective and damaging attacks due to the comprehensive knowledge of the system the attacker possesses.

Grey-Box Scenario In a grey box scenario (also known as a limited-knowledge scenario), the information about the target model possessed by the attacker is partial. This partial knowledge may include some information about the training data distribution, specific features used by the model, and the model's internal details or architecture. For example, in such scenarios, the attacks may have access to only a portion of the model's weights or a high-level description of its structure. In this scenario, the knowledge of the attacker is not complete as in a "white-box scenario", and the attacker has to put more effort into producing damaging attacks.

Black-Box Scenario In a black box scenario (also known as a zero-knowledge scenario), the attacker has no access to the internal model architecture, parameters, or any other privileged information. The attacker typically interacts with the model by submitting input queries and observing the corresponding output predictions. With limited knowledge, black-box attackers aim to craft adversarial examples that can deceive the model without a deep understanding of its inner workings. Black-box attacks are generally more challenging due to the lack of information but are relevant in real-world scenarios where the attacker has very limited knowledge of the model it wants to attack.

8.1.2.3 Adversary's Capability

The adversary's capability encompasses the arsenal of tools and resources the attacker has to perform an attack. This not only includes determining which dataset the attacker can potentially compromise but also extends to defining the nature of the attack itself. Specifically, it may involve choosing between tampering with the training set or manipulating input instances during the operational phase (commonly referred to as the test set). Furthermore can indicate the classes of instances the attacker is able to target.

For instance, consider the attacker's flexibility to execute different types of attacks. One option is the causative attack, where the attacker interferes with both the training and test sets, thereby initiating a poisoning attack aimed at compromising the integrity of the machine-learning system. Alternatively, the attacker can employ an evasion attack, which exclusively targets the test set to deceive the model.

Furthermore, the adversary's capability also pertains to the allocation of resources, often quantified in terms of a *budget*. This encompasses the number of instances that can be injected or modified within each dataset and the effort required to modify the features of an instance. The careful management of these resources plays a critical role in crafting effective adversarial strategies while adhering to predefined attacker resource constraints.

8.1.2.4 Adversary's Strategy

The adversary's strategy specifies how the attacker should execute the attack to pursue the adversary's goal while leveraging on the adversary's knowledge and adhering to the adversary's capability. In essence, the adversary's strategy delineates the attacker's approach to maximise its goal by manipulating instances within the training and/or test set while staying within the constraints imposed by its capability and knowledge.

In the AML contest, in most cases, the problem of determining the most effective attack strategy translates into solving an optimisation problem. The objective function of this problem aligns with the adversary's goal. The attacker's capability imposes the problem's constraints, and how the optimisation is solved depends on the adversary's knowledge of the system.

8.1.3 Threat Model

Another important component in the context of adversarial machine learning is the threat model. The threat model specifies the upper bound constraints within which the machine learning system can be attacked, in other words, its degree of freedom of an attacker with unlimited resources. The threat model imposes constraints on which features an attacker can modify, how much each feature can be perturbed, and the cost of making modifications to each feature.

It is important to note that although the threat model is closely related to the adversary's model, they are two distinct entities. Specifically, the threat model outlines the full spectrum of potential attacks the system may encounter, whereas the adversary's model manages the stronger threat that an attacker can pose with its capability and resources. Consequently, the attacker might produce attacks with less strength than what the threat model permits, often due to resource limitations. Nonetheless, it's crucial to underscore that the attacker can never execute an attack surpassing the constraints imposed by the threat model.

8.1.4 Type of Attack

Adversarial machine learning attacks can be categorised into classes based on the attacker's goal. These attacks aim to undermine the integrity and reliability of machine learning models, often with harmful consequences. Below is a brief description of the most common attack types found in the literature [16, 137, 67]:

- **Evasion Attacks:** Evasion attacks, also known as adversarial perturbations, focus on manipulating input data to mislead a machine learning model during its operational phase. These attacks entail making subtle, imperceptible alterations to input instances in an attempt to deceive the model into generating incorrect predictions. For example, an image classifier might be fooled into misclassifying a stop sign as a yield sign with only slight, strategically engineered modifications.

- **Poisoning Attacks:** Poisoning attacks primarily target the training phase of machine learning systems. The attacker injects malicious instances into the training dataset, thereby influencing the learning process to yield a model

that is inherently flawed. Poisoning attacks can involve introducing perturbed data points or strategically selected instances meant to manipulate the model's decision boundaries. Once trained on this corrupted set of data, the model's integrity becomes compromised, resulting in incorrect classifications during the operational phase.

- **Model Inversion Attacks**: Model inversion attacks focus on breaching the privacy of the model's training data. The attacker employs this approach to reverse-engineer sensitive information that the model has been trained on, often by exploiting subtle information leaks or output observations. By deducing underlying patterns and sensitive features, model inversion attacks compromise the privacy and security of the individuals or organisations that contributed their data to train the model.

- **Transferability Attacks:** Transferability attacks leverage the intriguing property that adversarial examples designed to deceive one machine learning model can often be successfully employed to fool other models as well, even if they are dissimilar. These attacks enable adversaries to craft a single adversarial instance that, when misclassified by one model, can also deceive another model. Transferability underscores the universal challenges posed by adversarial inputs, emphasising the need for robust defences against such threats.

8.2 Evasion Attacks

The main contribution of this thesis to the domain of adversarial machine learning primarily focuses on making tree-based machine learning classifiers robust and certifiable when subjected to *evasion attacks*. In particular, following the attack taxonomy introduced above, we focus our discussion on evasion attacks: *exploratory* attacks, producing *integrity violations* with *indiscriminate* targets in a *white-box* scenario.

An evasion attack occurs during the operational phase of the machine learning system [16, 11], e.g., when the model is applied in production and interacts with users. Given an already trained model, the attacker manipulates the legitimate input instances to force the model to produce incorrect predictions. These carefully manipulated instances are referred to as adversarial examples or *evasion instances*.

A common example of a real-world scenario illustrating this type of attack is the one of an aggressive seller attempting to deceive an email spam detection system. The seller carefully obfuscates the advertising content of the email in a way that the anti-spam system classifies that email as "ham" instead of "spam".

However, the modification cannot be too extreme, or the advertising message the seller wants to convey becomes illegible even to humans.

In the creation of evasion attacks, the attacker is constrained by the adversary's model and the threat model. Consequently, an attacker executing an evasion attack must adhere to different constraints. These constraints may involve the maximum number of features allowed to be modified (attackable features) and the degree to which each feature can be perturbed (attack magnitude). Generally, the perturbations that the attacker can make are modelled using the notion of cost. The costs are treated as resources the attacker must expend to carry out the attack. The attacker has a limited *budget* to spend on modifying the instance. For example, the attacker can modify a certain feature only by spending a certain cost. If the cost of the modification exceeds the available budget, the modification cannot be made. Therefore, the attacker can modify the instance as long as it has a sufficient budget. The adversary's strategy is to find the perturbation that maximises the prediction error while staying within the budget and threat model constraints.

The discovery of the existence of evasion attacks has raised some interesting observations. Multiple times, it has been shown that the perturbations applied to instances to generate effective evasion attacks are imperceptible to the human eye [16, 126, 75, 96, 160], and humans can easily correctly classify these evasion instances. This has led to observations such as the one made by Goodfellow *et al.* in [75], where machine learning models do not learn the semantics of the class. Therefore, even a slight alteration of the instance can produce a completely different prediction.

8.2.1 Formal Definition

More formally, given a classifier h and let $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ be a *legitimate* instance represented as a d-dimensional real-valued vector $\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})$ with label $y \in \mathcal{Y}$. An attacker A aims to manipulate $\boldsymbol{x}$ to generate an *evasion instance* $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $h(\boldsymbol{x}) \neq h(\boldsymbol{z})$. The set $A(\boldsymbol{x})$ is the *adversarial perturbation* set and resembles the formalisation of the adversary's model. $A(\boldsymbol{x})$ contains all the possible evasion instances generated by the attacker under the constraints of the adversary's model and threat model. Different adversary's models and threat models generate different adversarial perturbation sets.

The following formalisation provides a common example of an adversarial perturbation set for evasion attacks:

$$A(\boldsymbol{x}) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathcal{X} \ \wedge \|\boldsymbol{x} - \boldsymbol{z}\|_\rho \leq \delta \ \wedge \ \boldsymbol{x}_{lb} \preceq \boldsymbol{z} \preceq \boldsymbol{x}_{ub} \ \wedge \ \textstyle\sum_{f \in B} c_f \leq b\} \quad (8.1)$$

where B is the set of attacked features, c is a d-dimensional vector in $\mathbb{N}^d$ indicating te *cost* c_f to perturb each feature f, and b the attack's budget. Then, $\boldsymbol{u} \preceq \boldsymbol{v}$ means

that each element of $\boldsymbol{u}$ has to be less or equal to the corresponding element in $\boldsymbol{v}$, and $\boldsymbol{x}_{lb}$ and $\boldsymbol{x}_{ub}$ are respectively the lower bound and upper bound that the evasion instance $\boldsymbol{z}$ can assume.

The three constraints in Equation 8.1 ensure consistency between $\boldsymbol{z}$ and the adversary's capability. The constraint $\|\boldsymbol{x} - \boldsymbol{z}\|_\rho \leq \delta$ implies that the difference between $\boldsymbol{x}$ and $\boldsymbol{z}$, with respect to the L_ρ-norm or any distance function must be at most $\delta \in \mathbb{R}$. The box constraint $\boldsymbol{x}_{lb} \preceq \boldsymbol{z} \preceq \boldsymbol{x}_{ub}$ has two roles. First, ensures that the instance $\boldsymbol{z}$ remains within a fixed bound of values, e.g., by setting $\boldsymbol{x}_{lb} = \mathbf{0}$ and $\boldsymbol{x}_{ub} = \mathbf{1}$, the instance $\boldsymbol{z}$ is bounded in $[0, 1]^d$. Second, it allows to model non-attackable features; i.e., to ensure f being a non-attackable feature, it is sufficient to impose $x_{lb}^{(f)} = x_{ub}^{(f)} = x^{(f)}$. Finally, $\sum_{f \in B} c_f \leq b$ ensures that the overall cost of perturbing the features in B does not exceed the attacker's budget b.

To achieve this goal, attackers typically solve an optimisation problem to find the minimal perturbation of $\boldsymbol{x}$ that adheres to the constraints of the adversary's model and the threat model, enabling the creation of the evasion instance $\boldsymbol{z}$ that deceives the model. However, there are two different types of evasion attacks: **binary evasion attacks** and **multiclass evasion attacks**, each with different goals and optimisation strategies.

8.2.1.1 Binary Evasion Attacks

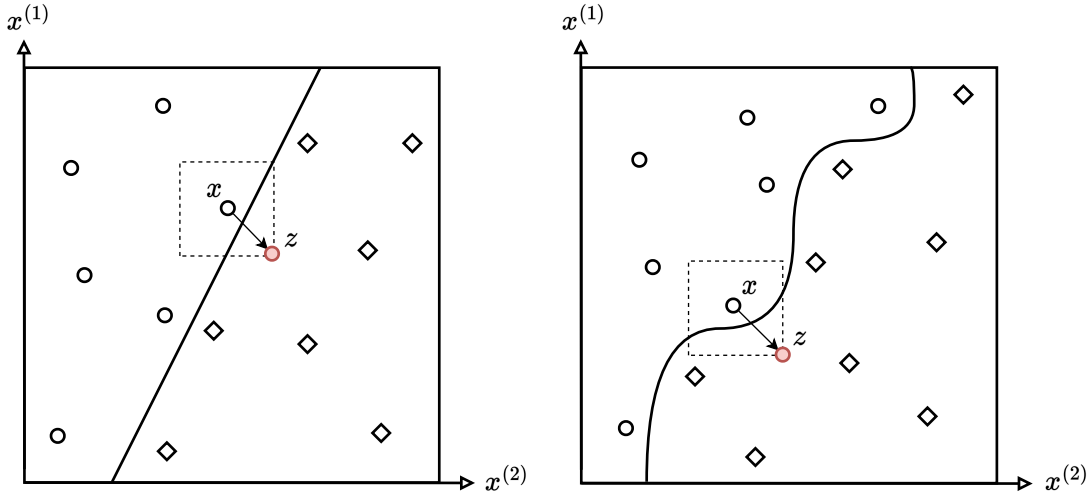
In the context of adversarial machine learning, a binary evasion attack is a type of attack in which an attacker seeks to manipulate a binary classifier to produce a desired binary response (often a "positive" or "negative" classification). In other words, the goal of the attack is to influence the model to make a specific decision, such as classifying an input as positive when it should actually be classified as negative, or *vice versa*.

This type of attack can have significant consequences in applications where binary decisions are critical, such as malware classification, financial fraud detection, or medical diagnosis. The adversary's goal is to push an instance as far as possible beyond the decision boundary that divides the two classes.

More formally, in a binary classification task with $\mathcal{Y} = \{-1, 1\}$, it is possible to generalise the intent of the attacker to look for an adversarial example to deceive the model h through the following optimisation problem [120]:

$$\arg\min_{\boldsymbol{z} \in A(\boldsymbol{x})} h(\boldsymbol{z}) y$$

where the prediction $h(\boldsymbol{z})$ denote the confidence score of the binary classifier h on the evasion instace $\boldsymbol{z}$. The smaller the $h(\boldsymbol{z})y$ value, the larger the error produced by h in predicting $\boldsymbol{z}$.



(a) Linear model (b) Non-linear model

Figure 8.1: Binary evasion attacks. The regions around the point $\boldsymbol{x}$ represent the constraint imposed by $\|\boldsymbol{x} - \boldsymbol{z}\|_\infty \leq \delta$, i.e., the L_∞-norm.

Figure 8.1 shows two examples of binary evasion attacks in the context of a linear and a non-linear model, respectively. Note that in the experimental section of this part of the thesis, we primarily focus on binary evasion attacks.

8.2.1.2 Multiclass Evasion Attacks

In a multiclass problem, the machine learning system must discriminate an instance belonging to a specific class among many others. In the context of adversarial machine learning, a multiclass evasion attack occurs when an attacker tries to manipulate a multiclass classifier to force it to classify the evasion instance in any (or in a specific class) of the available classes.

In this context, the attacker can perform two types of attack [126]:

- **Error-Generic Evasion Attacks:** With an error-generic evasion attack, the attacker attempts to misclassify an instance into one of the other classes.
- **Error-Specific Evasion Attacks:** With an error-specific evasion attack, the attacker attempts to misclassify an instance into a specific class.

In particular, in the error-generic scenario, the adversary has an interest in attacking the model, regardless of the class in which the adversarial example is misclassified. It is sufficient that the ending class is different from the original. For example, a well-known criminal has an interest in not being recognised by a

video surveillance system, but he is not interested in the identity with which he is mistakenly associated. Instead, in the error-specific scenario, the attacker wants to be classified in a specific class. This can be seen as a person attempting to authenticate as a specific user within a system. In this case, the attacker is interested in being misclassified as that person. Below, for each scenario, the optimisation problem that the adversary must solve to create an evasion instance is formalised.

Error-Generic Evasion Attacks To generate an error-generic evasion instance, the attacker must solve the following optimisation problem:

$$\arg\min_{\boldsymbol{z} \in A(\boldsymbol{x})} \Delta(\boldsymbol{z})$$

where $\Delta(\boldsymbol{z})$ is defined as:

$$\Delta(\boldsymbol{x}) = h_a(\boldsymbol{x}) - \max_{a \neq b} h_b(\boldsymbol{x}).$$

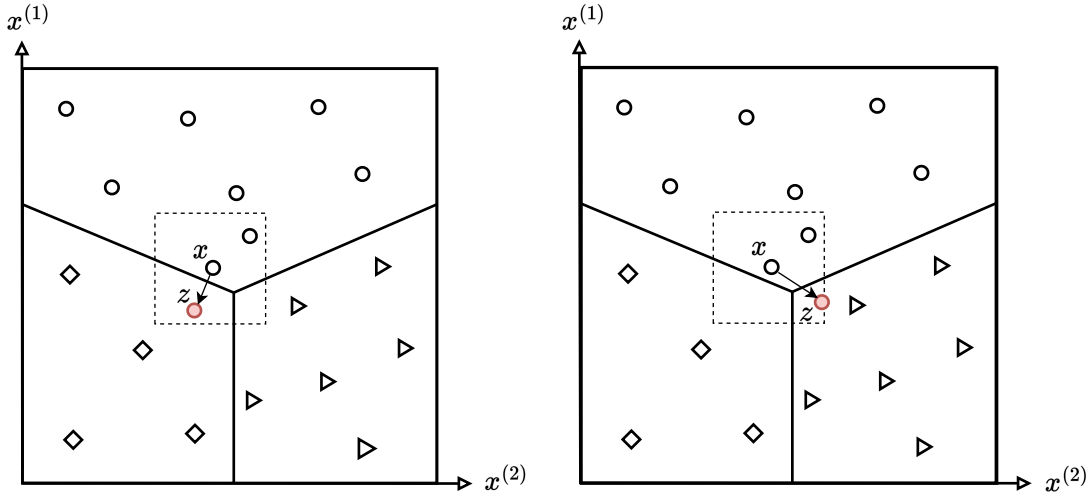
and where $a, b \in \mathcal{Y}$ and $h_y(\boldsymbol{x})$ denotes the confidence score of the classifier h on the instance $\boldsymbol{x}$ for any class $y \in \mathcal{Y}$. The label a represents the original class of the instance $\boldsymbol{x}$, while b represents any class different from the original one, in which the attacker tries to misclassify the evasion instance $\boldsymbol{z}$. The maximisation problem $\max_{a \neq b} h_b(\boldsymbol{x})$ looks for the wrong class $b \neq a$, which $h_b(\boldsymbol{z})$ produces the highest prediction score. Finally, the minimisation of $\Delta(\boldsymbol{z})$ forces the optimisation problem to look for the perturbed instance $\boldsymbol{z}$ that produces the minimum predicted score difference between the original class a and the class that generated the highest prediction score b. In other words, this means finding the evasion instance $\boldsymbol{z}$ misclassified into the closest class to $\boldsymbol{x}$.

Error-Specific evasion attacks To generate an error-specific evasion instance, the attacker must solve the following optimisation problem:

$$\arg\max_{\boldsymbol{z} \in A(\boldsymbol{x})} \Delta(\boldsymbol{z})$$

The difference with the error-generic scenario lies exclusively in the transformation of the minimisation problem into a maximisation problem. The label b represents the adversary's target class, the one in which the attacker wants its evasion instance $\boldsymbol{z}$ to be misclassified. On the other hand, label a represents the original class of the legitimate instance $\boldsymbol{x}$.

In this case, the attacker looks for the best perturbation of $\boldsymbol{x}$, which generates the highest score difference between the target class b and the original class a. The larger the difference, the greater the model's error when it misclassifies $\boldsymbol{z}$ as b instead of a.



(a) Error-generic evasion attack (b) Error-specific evasion attack

Figure 8.2: Multiclass evasion attacks. Figure (a): the attacker looks for the closest class to $\boldsymbol{x}$ to generate an error-generic evasion instance $\boldsymbol{z}$. Figure (b): the attacker looks for an error-specific evasion instance $\boldsymbol{z} \in A(\boldsymbol{x})$ with the highest score in the target class $\triangleright$ (triangle).

Finally, Figure 8.2 points out the differences between error-generic (Figure 8.2a) and error-specific (Figure 8.2b) evasion attacks.

8.2.2 Common Adversary's Constraints

Recall the definition of the adversarial perturbation set $A(\boldsymbol{x})$ provided in Equation 8.1. Commonly, the constraints employed to model the attacker's capability take the form of $\|\boldsymbol{x}-\boldsymbol{z}\|_\rho \leq \delta$. This constraint indicates that the evasion instance $\boldsymbol{z}$ can deviate from the original instance $\boldsymbol{x}$ by at most δ according to a certain constraint modelled by $\|\cdot\|_\rho$. These constraints are generally modelled using L_ρ-norm. The most common norms used in adversarial machine learning are as follows:

- The L_0-norm: $\|\boldsymbol{x}-\boldsymbol{z}\|_0 = \sum_{f=1}^{d} \mathbb{1}[x^{(f)} \neq z^{(f)}]$
- The L_1-norm: $\|\boldsymbol{x}-\boldsymbol{z}\|_1 = \sum_{f=1}^{d} |x^{(f)} - z^{(f)}|$
- The L_2-norm: $\|\boldsymbol{x}-\boldsymbol{z}\|_2 = \sqrt{\sum_{f=1}^{d} (x^{(f)} - z^{(f)})^2}$
- The L_∞-norm: $\|\boldsymbol{x}-\boldsymbol{z}\|_\infty = \max_f |x^{(f)} - z^{(f)}|$

During the experimental phase of this part of the thesis, the primary norms used are the L_0-norm and the L_∞-norm.

8.2.3 Evaluation Metrics

In the context of a machine learning model subjected to evasion attacks, there are generally two scenarios for performance evaluation: performance under normal conditions, i.e., when the model is not under attack, and performance under attack, i.e., the model's performance when exposed to malicious instances. Below are summarised the most common metrics for evaluating the performance of classification models under normal conditions:

The *Accuracy* (ACC) metric is probably the most well-known metric for assessing the performance of a classification model. It is calculated by measuring the ratio of correctly predicted instances to the total number of available instances. The *Precision* and *Recall* metrics, previously discussed for ranking in Section 3.2.2, calculate the fraction of positive instances predicted correctly compared to all instances predicted as positive and the fraction of positive instances predicted correctly compared to all actual positive instances. Furthermore, the *Area Under the Precision-Recall Curve* (AUC-PR) metric assesses the trade-off between Precision and Recall. This metric provides a nuanced perspective, particularly beneficial when dealing with imbalanced datasets where one class significantly outweighs the other. The *Receiver Operating Characteristic - Area Under the Curve* (ROC-AUC) [18] metric measures the model's ability to distinguish between positive and negative classes. A value of 1 indicates a perfect model, while 0.5 indicates a model that is no better than a random choice. The *Log Loss* is another critical metric, measuring the accuracy of predicted probabilities compared to actual probabilities. Minimising Log Loss indicates a model's proficiency in providing precise probability estimates. The *Confusion Matrix* offers a comprehensive overview of a model's predictions, detailing the counts of True Positives, True Negatives, False Positives, and False Negatives. This matrix serves as the foundation for deriving various performance metrics. *Matthews Correlation Coefficient* (MCC) [124] is a single-value metric that encapsulates the overall quality of a classifier, considering the balance between classes. A value of $+1$ denotes perfect predictions, 0 suggests random predictions, and -1 signifies perfect incorrect predictions.

Many of the just introduced metrics can be adapted to measure the model's performance under attack. For example, the *Robust Accuracy* (or Robustness) measures the percentage of correct model predictions on evasion instances compared to the total number of evasion instances. *Robust Precision* measures the fraction of adversarial instances correctly predicted as positive compared to the total number of adversarial instances classified as positive, while *Robust Recall* measures the fraction of adversarial instances correctly predicted as positive com-

pared to the total number of actual positive adversarial instances. The AUC-ROC metric can also be used to evaluate the model performance under adversarial attacks by measuring the model's ability to distinguish between positive and negative classes on evasion instances. Additionally, there is a less strict version of Robustness called *Stability*, which measures how many legitimate instances under attack do not cause a change in the model's prediction, regardless of whether the model's prediction on the legitimate instance is correct or not.

Below, we delve into the details of the metrics used to evaluate the model's accuracy and performance under attack in the studies covered in this part of the thesis.

8.2.3.1 Accuracy

Among the evaluation metrics under normal conditions listed above, the one we utilised for the experimental part of this thesis is Accuracy (ACC). Let $\mathcal{D}$ be a set of legitimate instances, i.e., instances not manipulated by the attacker, and let h be the classifier we want to evaluate. For each instance $(\boldsymbol{x}, y) \in \mathcal{D}$, Accuracy measures the ratio of instances correctly classified by h over the total number of instances in the set $\mathcal{D}$. Formally, we can define the Accuracy metric as follows:

$$\text{ACC} = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathbb{1}[h(\boldsymbol{x}) = y]}{|\mathcal{D}|} \tag{8.2}$$

8.2.3.2 Stability

Given an instance $\boldsymbol{x} \in \mathcal{X}$, the classifier h, and let $A(\boldsymbol{x})$ be the adversarial perturbation set of evasion instances $\boldsymbol{z}$, the classifier h is stable on $\boldsymbol{x}$ with respect to $A(\boldsymbol{x})$ if the following definition holds:

Definition 5 (Stability)**.** The classifier h is *stable* on the instance $\boldsymbol{x}$ if and only if, for every adversarial instance $\boldsymbol{z} \in A(\boldsymbol{x})$, we have $h(\boldsymbol{z}) = h(\boldsymbol{x})$.

Stability is useful for the security certification of classifiers: if the classifier h is stable on the instance $\boldsymbol{x}$, no adversarial instance $\boldsymbol{z} \in A(\boldsymbol{x})$ can be assigned a label different from the classifier prediction $h(\boldsymbol{x})$; hence no evasion attack is possible. However, stability does not capture whether a classifier is useful in practice: for example, a trivial classifier which always predicts a constant class is stable for any instance in $\mathcal{X}$.

8.2.3.3 Robustness

Due to the lack of accuracy concerns in the definition of stability, the actual property of interest for classifiers deployed in adversarial settings is robustness,

which additionally requires the classifier to perform correct predictions.

Given an instance $\boldsymbol{x} \in \mathcal{X}$ with label $y \in \mathcal{Y}$, the classifier h, and let $A(\boldsymbol{x})$ be the adversarial perturbation set of evasion instances $\boldsymbol{z}$, the classifier h is robust on $\boldsymbol{x}$ with respect to $A(\boldsymbol{x})$ if the following definition holds:

Definition 6 (Robustness)**.** The classifier h is *robust* on the instance $\boldsymbol{x}$ if and only if $h(\boldsymbol{x}) = y$ and h is stable on $\boldsymbol{x}$.

Regarding the metrics for assessing the model's performance under attack, we employed Robustness (or Robust Accuracy). This metric evaluates the model's performance by taking into account the evasion instances generated by the attacker. Given a set of instances $\mathcal{D}$, the Robustness ($\mathrm{R}(A)$) metric measures how many instances $\boldsymbol{x} \in \mathcal{D}$ are *robust* for classifier h under the attacks generated by the adversary's model A. In other words, the Robustness measures for how many instances $\boldsymbol{x} \in \mathcal{D}$ there not exists an evasion instance $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $h(\boldsymbol{z}) \neq y$, i.e., a successful attack. Formally, given an attacker A (i.e., an adversary's model), we define the Robustness metric as follows:

$$\mathrm{R}(A) = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathbb{1}[\forall \boldsymbol{z} \in A(\boldsymbol{x}) \,|\, h(\boldsymbol{z}) = y]}{|\mathcal{D}|} \tag{8.3}$$

Note that Robustness is computationally expensive to compute, as it requires the generation of all evasion instances $\boldsymbol{z} \in A(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{D}$.

8.2.4 Countermeasures to Evasion Attacks

The countermeasures that can be used to defend machine learning systems against evasion attacks can essentially be divided into two categories: creating more robust models against evasion attacks by sacrificing model accuracy in non-adversarial scenarios and designing algorithms that provide a fast robustness verification or certification of the machine learning system. This section is then divided into two parts. The first part focuses on the state of the art in developing more robust machine learning models. The second part covers the verification and certification of the robustness of the models.

For both sections, special attention is given to tree-based learning algorithms. The reason behind this is that, over the last years, research in this domain has primarily focused on linear classifiers [109, 15] and neural networks [160, 75], neglecting tree-based models. Decision trees are *interpretable* models [162], yielding human-understandable predictions regarding syntactic checks over domain features, which is particularly appealing in the security setting. Moreover, *decision trees ensembles* are nowadays one of the best methods for dealing with non-perceptual problems and Learning to Rank tasks.

8.2.4.1 Adversarial Robustness Enhancement

As the vulnerability of machine learning models to various types of attacks came to light, researchers in the domain of machine learning actively engaged in designing robust models, addressing both linear and non-linear models. For example, Support Vector Machine (SVM) [52], a well-known model in ML, was deeply examined by Biggio *et al.* in [12], revealing its susceptibility to evasion, poisoning, and privacy attacks. The same work provided countermeasures for each attack through an adversary-aware design of SVM models. In contrast, Chen *et al.* in [48] introduced Randomized-SVM, a version of SVM resistant to generalised adversarial attacks under uncertainty, achieved by training a distribution of classifiers instead of a single one.

An interesting and widely used technique for enhancing model robustness is *adversarial training*. Adversarial training involves incorporating evasion instances, alongside legitimate instances, into the training set during the model training phase. This approach allows the model to learn a decision boundary that is more robust against evasion attacks. The effectiveness of adversarial training is highly dependent on the strength of the generated attacks and the capability of the attack generator to distribute the attacks across different areas of the feature space, ensuring better model generalisation.

The vulnerability of neural networks (NN) to evasion attacks was extensively investigated by Goodfellow *et al.* in [75]. They attributed the susceptibility of NN models to evasion attacks to the linear nature of NN in high-dimensional spaces. In the same work, they provided a fast method for generating evasion instances against NNs, and these instances were used to train robust maxout networks [76] through adversarial training.

Kurakin *et al.* in [96] proposed a way to scale adversarial training efficiently to large models and datasets, along with a solution to the label leaking effect that causes adversarially trained models to be more robust on perturbed instances than on original instances. Madry *et al.* [117], instead, enhanced the robustness of neural networks through robust optimisation, utilising a natural saddle point formulation to capture security against adversarial attacks. Building upon this work, Cai *et al.* in [29] extended the concept of adversarial training. They advocated for a gradual injection of increasingly stronger evasion instances, leading to more robust models than those trained immediately with stronger attacks. They called this strategy *Curriculum Adversarial Training*; an optimised adversarial training approach. In addition, to prevent the model from *catastrophic forgetting* problem, i.e., forgetting the weakest attacks, they introduced the *batch mixing* technique during the training phase. This technique involves introducing lower-intensity attacks during the training of stronger attacks.

The literature also presents works aimed at enhancing the robustness of tree-

based models. Kantchelian *et al.* in [92] devised two new algorithms to generate evasion instances tailored for tree-based ensembles such as gradient boosted decision trees and random forests. The first algorithm relies on Mixed Integer Linear Programming (MILP), which identifies the optimal solution at a high computational cost. The second algorithm, named *symbolic prediction*, prioritises speed over optimality but still produces effective evasion instances. This last algorithm was used to train hardened GBDT models through *adversarial boosting* (i.e., adversarial training applied at each boosting iteration), with evasion instances generated by the symbolic prediction algorithm. These algorithms empirically demonstrated the vulnerability of both random forests and GBDTs to evasion attacks and highlighted how adversarial boosting can enhance the robustness of tree-based learning algorithms.

Chen *et al.* [45] designed an algorithm named *Robust Split*, aiming to create reliable models robust against evasion attacks by formulating the decision tree training process as an optimisation problem for determining the best splitting point. Robust Split considers the distance between data points and optimises worst-case performance under adversarial perturbations. Although models trained with Robust Split exhibit slightly lower accuracy on legitimate instances, they are more robust when facing evasion instances.

Subsequently, Calzavara *et al.* [36] developed a novel method for learning decision trees that are simultaneously accurate and nearly insensitive to evasion attacks. The learning algorithm, named Training Evasion-Aware Decision Trees (TREANT), serves as a base-learning algorithm to train the base learners of a random forest. In essence, the TREANT algorithm does not substantially differ from the learning algorithm of a normal decision tree. The model's robustness is guaranteed by how the dataset is partitioned during the tree's growth phase. During dataset division, the algorithm ensures that the loss resulting from the division is the lowest compared to the maximum loss generated by all possible attacks. To do so, the algorithm exploits the optimisation problem designed by Madry *et al.* in [117].

8.2.4.2 Certified and Verifiable Model Robustness

Alongside the research to develop more robust models against evasion attacks, a branch of research is dedicated to creating learning algorithms that are verifiable or certifiable by design or algorithms that provide robustness verification or certification for already trained models.

Providing robustness certification means finding the minimal perturbation for which no effective attack exists or finding a lower bound of robustness where false negatives, i.e., legitimate instances classified as attacks, may occur but not false positives, i.e., attacks classified as legitimate. On the other hand, robustness

verification focuses on checking whether, given a legitimate instance, there exists an effective adversarial example without actually generating all the evasion instances in the adversarial perturbation set $A(\mathbf{x})$.

It is noteworthy that in the literature, the distinction between robustness certification and verification is not always clear-cut, and the two terms are often used interchangeably. Additionally, sometimes, the former is called sound verification, while the latter is called complete verification.

These algorithms aim to compute model robustness more quickly, thereby circumventing the intractability of generating all possible attacks to verify the existence of an effective evasion attack. In fact, Kantchelian *et al.* in [92] demonstrated that verifying security against evasion attacks for decision tree ensembles is NP-complete when malicious perturbations are modelled with an arbitrary norm.

For this reason, several algorithms have been developed to make machine learning models verifiable or verify existing models. Chen *et al.* in [46] addressed the robustness verification problem for general tree-based models, including decision trees, random forests, and gradient boosted decision trees. They presented a simple linear-time algorithm for verifying a single tree. Additionally, they designed a solution for tree ensembles by formulating the verification problem as a max-clique problem on a multipartite graph with bounded boxicity, which, under certain conditions, can be executed with a polynomial-time algorithm.

Building on the work of Gehr *et al.* in [73] for neural networks, Ranzato and Zanella in [145] applied the concept of abstract interpretation to develop a tool called Silva for the formal verification of robustness and stability properties of tree-based ensembles. Silva utilises an abstract domain of not necessarily closed real hyperrectangles and is capable of conducting complete robustness checks of standard adversarial perturbations and outputting concrete adversarial attacks.

Leino *et al.* in [100] formalised a notion of global robustness, capturing the operational properties of online local robustness certification while providing a natural learning objective for robust training. Local robustness is the classical definition of robust and refers to the model's robustness computed locally at the instance level, i.e., the perturbations near individual instances. Global robustness refers to the model robustness with respect to the whole feature space. They achieved global robustness by introducing a special class representing instances falling within an area near the decision boundary, where a perturbation could lead to a class change. If an instance falls within these areas, the prediction is rejected. Finally, this work demonstrated how widely-used architectures can be easily adapted to this objective by incorporating efficient global Lipschitz bounds into the network, resulting in certifiably constructed models that achieve state-of-the-art verifiable and clean accuracy.

Yang *et al.* in [183] focused on providing certified robustness for ensemble

models, along with sufficient and necessary conditions for robustness for different ensemble methods. They discovered that diversified gradients and a large confidence margin are sufficient and necessary conditions for certifiably robust ensemble models under the model-smoothness assumption. Then, they provide a bounded model-smoothness analysis based on the proposed Ensemble-before-Smoothing strategy. Through these theoretical findings, they propose lightweight Diversity Regularized Training (DRT) to train certifiably robust ensemble ML models.

In a more recent research effort by Calzavara *et al.* in [32], they exploited a restricted class of decision tree ensembles called large-spread ensembles to design a verifiable learning algorithm for training polynomial time verifiable models. Moreover, they proved that large-spread ensembles are more robust than traditional ensembles against evasion attacks at the cost of an acceptable loss of accuracy in the non-adversarial setting.

8.2.5 Benchmark Datasets

In this section, we introduce the classification datasets employed in the experimental sections of the research encompassed within this part of the thesis. Table 8.2 summarises the fundamental details of these datasets. The list comprises a total of 8 publicly available datasets, each featuring a varying number of classes, ranging from 2 to 11.

As previously mentioned, the works presented in this part of the thesis only focus on binary classification tasks. Consequently, in the case of multiclass datasets (more than two classes), we selected and isolated two specific classes or merged classes with fewer instances to create a larger class suitable for binary classification. The details of each dataset transformation are addressed in the corresponding work's Chapters.

- Wine [66]: The Wine dataset is a well-known dataset for classification. The dataset contains information pertaining to the outcomes of a chemical examination conducted on wines cultivated in a specific Italian region. These analyses were conducted on wines originating from three distinct cultivars. The analysis focused on quantifying the presence of 13 different constituents within each variant of the wines. The classification task aims to identify the cultivar of origin based on the features representing 13 different constituents within the wine. The dataset contains 3 classes; however, in the experiments, we collapsed two classes in one class to perform binary classification tasks. In this scenario, an attacker alters the constituents of the wine coming from a cultivar to make it appear as if it comes from another cultivar. The dataset can be downloaded from https://archive.ics.uci.edu/dataset/109/wine.

Table 8.2: Datasets properties.

Dataset	#features	#instances	#classes
Wine [66]	13	178	3
Spam Base [84]	57	4,601	2
Breast Cancer D [158]	30	569	2
Breast Cancer O [10]	9	683	2
MNIST [99]	784	70,700	10
Diabetes [156]	8	768	2
Cod-RNA [167]	8	59,535	2
Sensorless [7]	48	58,509	11

- Spam Base [84]: The Spam Base dataset focuses on the "spam" concept, encompassing various forms such as advertisements for products/websites, get-rich-quick schemes, chain letters, etc. The objective of the classification task associated with this dataset is to correctly determine whether a given email is spam or not. The Spam Base dataset is well-known in the adversarial machine learning domain. In this scenario, the attacker's aim is to deceive the model to classify spam emails as ham (not spam). The dataset can be downloaded from https://archive.ics.uci.edu/dataset/94/spambase.

- Breast Cancer D [158]: The Breast Cancer D dataset is the Breast Cancer Wisconsin (Diagnostic) dataset, which includes features extracted from breast masses classified as "benign" or "malignant". This dataset is used to train binary classification models to determine whether a breast mass is benign or malignant. In detail, the dataset involves features computed from a digitised image of a fine needle aspirate (FNA) of a breast mass. These features specifically capture characteristics of the cell nuclei present in the image. The dataset can be downloaded from https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic.

- Breast Cancer O [10]: The Breast Cancer O dataset is the Breast Cancer Wisconsin (Original) dataset, which includes features extracted from breast masses classified as "benign" or "malignant". The dataset contains 16 instances with missing values, which we removed in the experiments. The dataset was acquired by the University of Wisconsin Hospitals, Madison. It comprises features such as clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, and others. This dataset is utilised for training binary classification models to determine whether a breast mass is benign or malignant. The dataset can be downloaded from

https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original.

- MNIST [99]: The MNIST dataset comprises a collection of 70,000 images, each intricately detailed at 28x28 pixels, resulting in a total of 784 pixels per image. These images encapsulate handwritten digits ranging from 0 to 9 (ten classes). Each image is represented with a two-dimensional pixel matrix of greyscale values. The greyscale values of each pixel span from 0 to 255, providing a spectrum of shades that collectively form the visual representations of numerical characters. MNIST stands as a fundamental benchmark in the realm of machine learning, commonly employed for training and testing image classification algorithms. The dataset contains 10 classes; however, in the experiments, we isolated two classes and performed binary classification tasks. The dataset can be downloaded from https://web.archive.org/web/20220331130319/https://yann.lecun.com/exdb/mnist/

- Diabetes [156]: The Diabetes dataset originates from the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset contains extracted features to predict whether a patient has diabetes or not. The predictions are based on specific diagnostic measurements contained in the dataset. The instances were selected under certain constraints from a larger database, with a focus on female patients who are at least 21 years old and of Pima Indian heritage. The dataset comprises various medical predictor variables, such as the number of pregnancies, BMI (Body Mass Index), insulin level, age, and others. Through this dataset, it is possible to train binary classifiers to predict whether a patient has diabetes or not. The dataset can be downloaded from https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

- Cod-RNA [167]: The Cod-RNA dataset provided by Uzilov *et al.* in [167] contains two classes and is used in binary classification tasks. The dataset is designed for detecting non-coding RNAs based on the predicted free energy change for secondary structure formation. The dataset can be downloaded from https://www.csie.ntu.edu.tw/c̃jlin/libsvmtools/datasets/binary.html#cod-rna.

- Sensorless [7]: The Sensorless dataset comprises features extracted from electric current drive signals, where the drive system includes both intact and defective components. The dataset encompasses 11 distinct classes, each representing a different condition. These conditions have been measured multiple times under 12 operating conditions, encompassing various speeds, load moments, and load forces. The dataset contains 11 classes, so it is

suitable for multiclass classification tasks. However, during the experiments, we isolated two classes and performed binary classification tasks. The dataset can be downloaded from https://archive.ics.uci.edu/dataset/325/dataset+for+sensorless+drive+diagnosis.

8.3 Summary

In this chapter, we introduced the fundamental concepts and knowledge necessary to understand our contribution to the domain of Adversarial Machine Learning. Below is a brief summary of the key concepts discussed in this chapter.

- **Adversarial Machine Learning:** In this section, we introduced the concept of adversarial machine learning affecting the machine learning systems. We provided a detailed description of the **adversary's model**, encompassing its goal, knowledge, capability and strategy, and the difference to the **threat model**. Lastly, we briefly described the **types of adversarial attacks**.
- **Positioning of This Work:** In this thesis, we focus on making tree-based binary classifiers robust and certifiable when subjected to **evasion attacks** characterised as **exploratory** attacks, producing **integrity violations** with **indiscriminate** targets in a **white-box** scenario.
- **Evasion Attacks:** Given that this part of the thesis primarily focuses on evasion attacks, we furnished a detailed definition of evasion attacks and made a distinction between binary and multiclass evasion attacks. We introduced the concept of the **adversarial perturbation set**, i.e., the collection of all evasion instances generable by the attacker starting from a legitimate instance. Finally, we presented the most common **constraints** used to model the adversary's model and threat model.
- **Evaluation Metrics:** We introduced the most common metrics used in machine learning to assess the effectiveness of classification models with or without the presence of evasion attacks. Among the presented metrics, there is the **Accuracy**, which measures the number of instances classified correctly and the **Robustness**, which measures how many instances do not have an evasion instance that produces an incorrect prediction.
- **Countermeasures:** In this section, we introduced the state of the art in the domain of adversarial machine learning in a binary classification scenario. We focused specifically on the literature concerning designing **robust learning** algorithms robust to evasion attacks and on **verify** or **certify** model robustness under evasion attacks.
- **Benchmark Datasets:** In the final section of this chapter, we introduced eight benchmark datasets used in the experiments for evaluating the effectiveness of the models, both with and without the presence of the attacker. Each dataset possesses distinct characteristics, varying in the number of instances and features.

Chapter 9

Feature Partitioned Forests

In this chapter, we illustrate the work titled "*Feature Partitioning for Robust Tree Ensembles and their Certification in Adversarial Scenarios*", in proceedings as a full paper at the *EURASIP Journal on Information Security, 2021.* Further details can be found in the reference [34].

As mentioned in Section 8.1, machine learning models are deployed in a large variety of contexts such as system security, health care, control critical processes, and other [85, 16]. Unfortunately, traditional machine learning algorithms are proved to be vulnerable to a wide range of attacks, and in particular to evasion attacks, where an attacker carefully manipulates an input instance to force model's prediction errors [11, 131, 138, 128].

In this work, we present a meta-learning algorithm named *Feature-Partitioned Forest* (FPF) to train ensembles of decision trees robust against evasion attacks constrained by L_0-norm perturbations. In particular, we focus on binary classification tasks and show that the proposed algorithm can train models that are *robust by construction*. The algorithm benefits from the theoretical property that the majority of the base learners in the ensemble are never affected by the evasion instances attacking the model.

The idea behind FPF that provides the theoretical guarantees relies on how the feature set is partitioned at training time. Specifically, we randomly equi-partition the set of features and train each tree of the ensemble on a distinct feature partition. Such sampling limits the number of features considered by a single tree, and therefore, it limits the number of trees affected by the perturbations on a set of features.

Along with FPF, we provide two *certification methods* for our tree-based ensembles that efficiently compute a lower bound of the model's robustness under attack. Finding a successful attack requires finding a set of features of a given instance that, if corrupted, affects the prediction of the model. In general, this requires an exhaustive search of all the possible feature subsets and their possi-

ble modifications, which becomes quickly unfeasible for powerful attacks or large feature sets. We show that this problem can be reduced to a *partial set cover problem*, and we use efficient set cover algorithms to assess the non-existence of harmful attacks. By certifying the non-existence of attacks for some instances of a given set, we can provide an accurate model's robustness lower bound in polynomial time. Furthermore, we devise a *cascading* strategy where instances found by the proposed lower bounds as potentially attackable are eventually evaluated with a slow exact method. The *cascading* strategy can reduce the total running time up to two orders of magnitude and provide an exact local robustness computation.

In summary, the contributions of this work are as follows: *i)* We design a novel meta tree-ensemble learning algorithm named Feature-Partitioned Forest that is robust against adversarial attacks by construction. *ii)* We provide two novel *certification methods* that quickly compute a lower bound of the model's robustness on a given set of legitimate instances. *iii)* We devise an exact *cascading* strategy for robustness verification by exploiting the proposed lower bounds as a fast preprocessing filter.

We experimentally evaluate FPF on three public datasets and show that FPF trains tree-based ensembles that achieved an improvement of up to 16% in terms of robustness compared to the baselines. Moreover, we empirically demonstrated how the proposed robustness lower bound certifiers are accurate and can provide a 100× speed-up compared to an efficient brute force verifier.

The Chapter is structured as follows. In Section 9.1, we introduce the related work specific to decision-tree-based ensemble methods in the domain of evasion attacks. In Section 9.2, we delve into the definition of the threat model and the adversary's model. In Section 9.3, we delve into the main contribution of this work. We introduce Feature-Partitioned Forest and the theoretical guarantees behind it. In Section 9.4, we introduce the second contribution of this work: two certification algorithms to compute a lower bound of FPF-based models' robustness. Moreover, we provide an efficient cascade verification algorithm for exact robustness computation. Section 9.5 details the experimental setup used to perform experiments and the learning algorithms employed to assess the robustness of our solution. In Section 9.6, the primary outcomes are presented, emphasising the accuracy and robustness of the learning algorithms, as well as the accuracy and efficiency of the robustness certifiers and verifiers. Section 9.7 provides in-depth empirical and analytical studies of the behaviour of FPF on varying its hyperparameters. Finally, in Section 9.8, we summarise our contributions, results, and potential future work.

9.1 Related Work: Robust Training

Most of the work in adversarial learning regards classifiers, particularly binary ones. The attacker starts from a legitimate instance that is classified correctly by the machine learning system and tries to perturb the instance to force the model to change the predicted class [130, 11, 14, 157, 92, 39, 55, 75]. To prevent these attacks, different techniques have been proposed for different models, including support vector machines [15, 181, 48], deep neural networks [78, 75, 139], decision tree ensembles [92, 45, 36, 4], and others learning algorithms.

As mentioned in Seccion 8.2.4, the main research directions investigated entangling the models' robustness to evasion attacks divided into enriching the training dataset (e.g., adversarial training) [92, 160] and attack-aware loss function [45, 36].

Kantchelian *et al.* in [92] borrowed the idea of the adversarial training approach from Szegedy *et al.* [160] and applied it to GBDTs. A greedy algorithm named *symbolic prediction* is exploited at each boosting round to create an effective evasion instance for every legitimate instance in the training set. Then, each tree in the ensemble is trained on both original and perturbed instances. This approach has increased the models's robustness; however, it comes at the expense of training efficiency. Indeed, the necessity to generate new evasion instances at each boosting round makes this algorithm impractical for large datasets. Additionally, as for other adversarial training approaches, evasion instances generated from the training set may not accurately represent the evasion instances encountered during the operation phase, potentially resulting in lower accuracy.

Another adversarial learning technique for decision tree ensembles was proposed by Chen *et al.* in [45], introducing the first tree learning algorithm, here named Robust Tree (RT), that directly incorporates the attacker into the optimisation problem solved during tree construction. The central concept of their approach, named *robust split*, redefines the splitting strategy for training examples at a tree node to account for the impact of the attacker. To mitigate the complexity of computing all possible evasion attacks, the authors approximated the attacker's behaviour.

A similar solution is proposed by Calzavara *et al.* in [36], where they designed TREANT. TREANT is a learning algorithm aimed at training robust decision trees against evasion attacks. While constructing a single tree, the optimal split is chosen by minimising the loss under attack without introducing any heuristic approximation. TREANT ensures that every node added to the tree does not increase the loss under attack for that tree. Finally, multiple trees trained with TREANT are grouped in an ensemble to increase the model's accuracy and robustness.

Another step forward was made by Andriushchenko *et al.* in [4], where they proposed an exact solution to optimise loss under attack for an ensemble of decision

stumps, i.e., trees having only a root and two leaves. While the proposed approach is the first to take into consideration the full ensemble, the use of decision stumps may limit the overall accuracy of the forest.

Calzavara *et al.* [35] pointed out the problem that most of the adversarial training approach assumes the use of differentiable learning algorithms to create evasion attacks. Consequently, adversarial training cannot be directly applied to machine learning techniques such as ensembles of decision trees. To address this limitation, they developed an adversarial training approach for gradient boosted decision trees. Furthermore, to enhance the algorithm's efficiency, they exploited the knowledge of tree thresholds, thus reducing the set of possible perturbations without loss of generality. This allows them to take advantage of differentiable approximations and makes the optimisation problem tractable.

Vos *et al.* [170] introduced GROOT, an efficient algorithm for training accurate and robust decision trees. GROOT analytically calculates the adversarial Gini impurity, significantly reducing training time. It achieves a learning algorithm that is two orders of magnitude faster than competitors [36, 45], with superior performance in terms of robustness. Moreover, GROOT allows the definition of a more expressive threat model, not only constrained by norms but also permitting each feature to be perturbed with a user-specified parameter, either a maximum distance or constraints on the direction of perturbation. It also allows the specification of the target class of attack and a trade-off between accuracy and robustness.

Then, Ranzato *et al.* [147] proposed a genetic adversarial training algorithm called Meta-Silvae to train decision trees in order to maximise both accuracy and robustness to adversarial perturbation. The algorithm relies on a Silva, a complete formal verification based on abstract interpretation designed by Ranzato and Zanella in [145].

Finally, Tin Kam in [81] devised the Random Subspace method (RSM), a learning algorithm for training accurate tree-based ensemble models. RSM employs an autonomous, pseudorandom process to select a small number of dimensions from a given feature set. The new feature set created from the selection is called subspace. During the training phase, each instance in the training set is projected to this subspace, and each base learner of the ensemble is tainted with instances projected in a different subspace. In the prediction phase, an instance traverses each tree of the ensemble and is projected to the same subspace of the tree. The use of randomisation provides a convenient way to explore the feature space to provide a more heterogeneous ensemble.

The vast number of subspaces in high-dimensional feature spaces offers more choices than needed in practice. Hence, while most other classification methods suffer from the curse of dimensionality, this method can take advantage of high dimensionality. Contrary to Occam's Razor, the classifier improves on generalisation

accuracy as it grows in complexity.

RSM produces distributed feature weights, making it a strong candidate as a method to achieve a good trade-off between accuracy and robustness in adversarial classification scenarios. In fact, RSM was deeply studied by Biggio *et al.* in [13] as an ensemble method to train accurate and robust machine learning models.

RSM shares many similarities with FPF but does not provide any theoretical guarantees to the model's robustness. FPF divides the feature set to ensure that the majority of the ensemble is never involved in an attack by construction. For example, if the attacker can modify at most any combination of three features, FPF distributes the feature among the trees in the ensemble to guarantee that any combination of three features appears in less than half of the ensemble, thus leaving the majority of weak learners unaltered in their predictions. This robustness by construction is not guaranteed with RSM where each base learner is trained on a random subset of the feature set, with possible repetition among weak learners, and so, in the worst case, an attack on a feature can involve the entire ensemble.

Finally, research in this area has also focused on verifying the robustness of the models under attack. It is worth noting that while the attacker looks for an effective attack, the evaluation process must verify that no effective attack exists, and this process is much more costly, if not infeasible. This difficulty and the high computational cost are caused by the large number of potential adversary attacks.

Kantchelian *et al.* in [92] proved that the problem of providing a robustness verification for tree-based ensembles is already NP-hard regardless of the L_ρ-norm adopted. Chen *et al.* in [46] showed that verifying the robustness of a forest F with at most l leaves per tree has cost $\min\{O(l^{|F|}), O((2|F|l)^{|\mathcal{F}|})\}$ assuming a L_∞-norm attacker. Andriushchenko and Hein in [4] demonstrated the feasibility of robustness verification for simplified decision trees, known as decision stumps, i.e., trees with only two leaves. Additionally, recent research by Ranzato and Zanella in [145] used abstract interpretation to address verification of tree-based ensemble. Furthermore, Calzavara *et al.* in [32] utilised a specific category of decision tree ensembles termed large-spread ensembles to develop a verifiable learning approach, facilitating the training of polynomial time verifiable models.

In this research direction, the special ensemble structure provided by the FPF robust learning algorithm allowed us to develop two efficient robustness certifiers that provide accurate robustness lower bound. In particular, the most efficient certifier computes an accurate robustness lower bound in polynomial time.

9.2 Threat and Adversary's Model

The threat model of the machine learning system considered in this work assumes a binary classifier that is vulnerable to evasion attacks. In this scenario, the system

is vulnerable to an attacker that tries to deceive an already trained binary classifier h by maliciously perturbing a legitimate instance $\boldsymbol{x}$ to create an evasion instance $\boldsymbol{z}$. The vulnerability of the machine learning system allows an attacker to modify at most $\lceil |\mathcal{F}|/2 \rceil - 1$ features with any intensity.

Concerning the adversary's model, we adopted the definition proposed by Kantchelian *et al.* in [92] and characterised it as follows: We denoted an attacker with A_b, where b represents the budget (resources) available to modify a legitimate instance to create an evasion attack. The attacker A_b generates evasion instances under the constraints of the threat model. Furthermore, in this work, we assumed that the cost of perturbing each feature f is $c_f = 1$. Consequently, A_b can modify a given instance $\boldsymbol{x}$ into an evasion instance $\boldsymbol{z}$ such that the L_0-norm of the perturbation is smaller than the attacker's budget b, i.e., $\|\boldsymbol{x} - \boldsymbol{z}\|_0 \leq b$, and the budget $b < |\mathcal{F}|/2$. Therefore, attacker A_b can perturb the instance $\boldsymbol{x}$ by modifying at most b features without any intensity constraints.

Given the adversary's model A_b, and an instance $\boldsymbol{x} \in \mathcal{X}$, we defined the adversarial perturbation set $A_b(\boldsymbol{x})$, i.e., the set of all the evasion instances the attacker may generate starting from $\boldsymbol{x}$, as follow:

$$A_b(\boldsymbol{x}) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathcal{X} \ \wedge \ \|\boldsymbol{x} - \boldsymbol{z}\|_0 \leq b\}\,. \tag{9.1}$$

Additional adversary's models have been investigated in the literature, including adversary's models based on generic L_p-norm constraints, as seen in the work of Szegedy *et al.* [160], or adversary's models based on rewriting rules, as proposed by Calzavara *et al.* in [36]. We decided to restrict attention to L_0-norm attacks because of their simplicity and effectiveness. Indeed, a very small b is sufficient to achieve successful attacks. Su *et al.* in [159] showed that with a one-pixel attack, i.e., with $b = 1$, it is possible to fool a complex deep neural network as VGG16 [154] and decrease its accuracy to a poor 16%. This emphasises that such a simple adversary's model can still significantly impact the security and robustness of the machine learning system.

9.3 Contribution 1: Feature-Partitioned Forest

In this section, we introduce the main contribution of this work. We provided Feature-Partitioned Forest (FPF), an ensemble method designed to train forests of binary decision trees that exhibit robustness against evasion attacks. Specifically, the ensembles generated by FPF are robust by design to attackers capable of perturbing, at most, b features of legitimate instances to induce prediction errors. FPF guarantees such robustness by train forests where the majority of the trees remain not affected by those evasion attacks.

9.3.1 Robust Feature Partitioning

In this section, we provide our definition of robust feature partitioning in adversarial scenarios. However, we first need to clarify what feature partitioning is.

Definition 7 (Feature Partitioning)**.** Given the feature set $\mathcal{F}$, the partition set $\mathcal{P} = \{P_1, \ldots, P_n\}$ is a partition of $\mathcal{F}$, if the following property holds:

$$\bigcup_{P_i \in \mathcal{P}} P_i = \mathcal{F} \quad \wedge \quad \forall_{P_i, P_j \in \mathcal{P}} \, P_i \cap P_j = \emptyset.$$

In other words, $\mathcal{P}$ is a partition of the feature set $\mathcal{F}$ if all sets $P \in \mathcal{P}$ are disjoint and their union returns the entire feature set. Note that the number of set n can be any number in the interval $[1, |\mathcal{F}|]$.

Through Definition 7, we can define a robust feature partitioning as follows:

Definition 8 (Robust Feature Partitioning)**.** Given a partition $\mathcal{P}$ of the feature set $\mathcal{F}$, an attacker A_b, and the set of attacked features B, $\mathcal{P}$ is a **robust** partition of $\mathcal{F}$ if the majority of its sets do not contain the attacked features in B, i.e., if the following property holds:

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

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

When $|B| \leq b$, it is straightforward to show that this property is surely satisfied if $|\mathcal{P}| \geq 2b + 1$. In the worst case, at most b distinct subsets of $\mathcal{P}$ overlap with B, leaving the remaining $b + 1$ subset of $\mathcal{P}$ not affected. Hereinafter, we consider only *robust* feature partitions $\mathcal{P}$ where $|\mathcal{P}| = 2b + 1$.

9.3.2 Robust Forest Ensembling

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

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

From above, we can formally define a *Robust Forest*, robust to evasion attacks generated by A_b, as follows:

Definition 9 (Robust Forest)**.** Given an attacker A_b, a forest F is a **robust forest** if the majority of its trees are not affected by A_b for any of its attacks:

$$\sum_{t \in F_{\mathcal{P}}} (\mathbb{1}[t(\boldsymbol{x}) = t(\boldsymbol{z})]) > \frac{|F_{\mathcal{P}}|}{2}, \qquad \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{z} \in A_b(\boldsymbol{x}).$$

It is straightforward to show that in $F_{\mathcal{P}}$ at most b of its $2b + 1$ trees can be affected by the attacker, and thus the *robustness* property is guaranteed by design.

Note that, in the best case scenario where each $t \in F_{\mathcal{P}}$ is perfectly accurate, the above robustness property ensures that, in the presence of attacks, only a minority of trees provide an incorrect prediction, and therefore, the forest is perfectly accurate even under attack. Clearly, this scenario is unlikely to happen in reality; therefore, in the next section, we discuss how to increase the accuracy of $F_{\mathcal{P}}$.

Finally, the definition of robust forest given above holds for any instances $\boldsymbol{x} \in \mathcal{X}$ and not only for the instances in the given dataset; consequently, this definition is dataset independent. Moreover, the above definition and training strategy trivially generalises to any base-learning algorithm besides tree-based classifiers.

9.3.3 Improve Model Accuracy and Robustness

The Definition 9 guarantees that the majority of trees in a robust forest are not affected by an attack. However, it does not provide any guarantees of the robustness computed with Equation 8.3 since it also depends on the accuracy of each tree in the forest. In fact, in a FPF forest, the more accurate the trees $t \in F_{\mathcal{P}}$, the more likely the forest $F_{\mathcal{P}}$ is robust.

The accuracy of single trees in a forest $F_{\mathcal{P}}$ strongly depends on the feature partition $\mathcal{P}$. The larger the set $\mathcal{P}$, the smaller the number of features each tree is trained on. To increase the accuracy of a robust forest $F_{\mathcal{P}}$, we equi-partition $\mathcal{F}$ across $\mathcal{P}$ so as to have $|P| \geq \lfloor |\mathcal{F}|/(2b+1) \rfloor$ for all $P \in \mathcal{P}$. Clearly, as the attacker's budget b increases, we need to partition $\mathcal{F}$ into a greater number of subsets. However, to generate accurate trees, we also require the dataset to have a large number of high-quality features. Note that this is true for every learning algorithm: if the attacker can perturb up to b features, it is necessary to have more than b high-quality features to train an accurate model. In addition, a specific partition $\mathcal{P}$ may only be sub-optimal as there may be multiple ways of partitioning $\mathcal{F}$ so as to achieve feature subsets that provide trees with higher predictive power.

In light of these observations, we define the final Feature-Partitioned Forest algorithm to train robust and accurate forests. To enhance accuracy and, consequently, the robustness of the forests, we choose to exploit the set of possible robust partitions of $\mathcal{F}$ to train multiple distinct robust forests $F_{\mathcal{P}}$, for each partition $\mathcal{P}$, and then combine them into a single robust forest.

Algorithm 5 Feature-Partitioned Forest Training

function FEATUREPARTITIONEDFOREST($\mathcal{D}$, r, A_b)
  Input
    $\mathcal{D}$: training set
    r : number of robust training rounds
    A_b : adversary's model
  Output
    F : final ensemble ▷ A set of robust forests

  $F \leftarrow \emptyset$
  $b \leftarrow A_b$
  $k \leftarrow 2b + 1$
  for $i = 1$ **to** r **do**
    repeat
      $\mathcal{P}_i \leftarrow$ RANDOMPARTITION($\mathcal{F}, k$) ▷ Partition $\mathcal{F}$ into k disjoint set
      $F_{\mathcal{P}_i} \leftarrow \emptyset$
      for $P_j \in \mathcal{P}_i$ **do**
        $t_{ij} \leftarrow$ DECISIONTREE($\pi_{P_j}(\mathcal{D})$)
        $F_{\mathcal{P}_i} \leftarrow F_{\mathcal{P}_i} \cup \{t_{ij}\}$
    until ACCEPTCONDITION($F_{\mathcal{P}_i}$)
    $F \leftarrow F \cup F_{\mathcal{P}_i}$
  return F

The details of the final FPF learning algorithm are outlined in Algorithm 5. FPF iterates over a given number r of robust training rounds, where r is a hyper-parameter of the algorithm. During each round $i \in [1, r]$, the algorithm generates a random robust partition $\mathcal{P}_i$ of the feature set $\mathcal{F}$. Thus, according to Definition 8, at each round i, the feature set $\mathcal{F}$ is randomly and evenly split into $2b + 1$ disjoint subsets (i.e., $|\mathcal{P}_i| = 2b + 1$), and a new decision tree is trained on each of the dataset projections $\pi_{P_j^i}(\mathcal{D})$ for every feature subset $P_j^i \in \mathcal{P}_i$. Then, resulting $2b + 1$ trees are ensembled into a robust forest $F_{\mathcal{P}_i}$. Finally, after r iterations, all the robust forests $F_{\mathcal{P}_i}$ are ensembled into an even more accurate and robust forest $F_{\mathcal{P}} = F_{\mathcal{P}_1} \cup \cdots \cup F_{\mathcal{P}_r}$.

Note that the prediction of a FPF forest consists of the majority voting over the predicted classes returned by the trees in the ensemble.

The function ACCEPTCONDITION used in Algorithm 5 is used as an accept condition to filter out those $F_{\mathcal{P}_i}$ that would not strengthen the final ensemble. For instance, it might be the case that some partitions in $\mathcal{P}_i$ do not contain sufficiently predictive features to train accurate trees. In this work, we use a simple accep-

tance criterion. A forest $F_{\mathcal{P}_i}$ is accepted if training accuracy exceeds the dataset's majority class percentage of the most frequent class. If the accept condition is satisfied, $F_{\mathcal{P}_i}$ is included in $F_{\mathcal{P}}$, otherwise a new robust partition $\mathcal{P}_i$ for iteration i is generated.

It is crucial to note that, through the ensembling of robust forests, the resulting forest remains robust by construction. In other words, the majority of the trees within it are not affected by the evasion attacks.

Proposition 1 (Robustness of FPF)**.** The forest $F_{\mathcal{P}}$ trained by the FPF learning algorithm is **robust** against an attacker A_b, if the majority of its trees are not affected by A_b for any of its attacks.

The proof of the above proposition is trivial. We recall that during each robust training round i, FPF trains a set of $2b + 1$ trees, where at most b trees can be affected by A_b. After r rounds, FPF has built a forest of $r(2b+1)$ trees, of which at most rb can be affected by A_b, while the remaining $r(b + 1)$ trees are not affected; the majority of the trees are not affected by A_b.

Furthermore, it's important to recall that the model's robustness is closely tied to the accuracy of trees not affected by the attack. In fact, an attacker can exploit a non-attacked tree that is also inaccurate to enhance the strength of the attack. The use of multiple robust training rounds r to reinforce the robustness of the forest against evasion attacks generated by A_b is effective for the following proposition.

Proposition 2 (Inaccuracy Tolerance)**.** Given a FPF robust forest $F_{\mathcal{P}}$, with r robust training rounds, trained to be robust to an attacker A_b. The forest $F_{\mathcal{P}}$ is robust to any possible attack generated by A_b if at most ξ percentage of the trees in $F_{\mathcal{P}}$ are inaccurate, with ξ defined as follows:

$$\xi = \frac{\lceil r/2 \rceil - 1}{r(2b + 1)}$$

As a result, the larger the value of r, the higher the upper bound on the number of trees that can be inaccurate. This leads to a higher probability that at least the majority of the forest predicts the correct class.

9.4 Contribution 2: Robustness Certifiers

Evaluating the robustness of a model in the presence of an attacker is a difficult and computationally expensive task. This is due to the possibly large size of $A_b(\boldsymbol{x})$ and the number of interactions among trees in a forest. Below, we first discuss an

expensive brute-force strategy for robustness verification of any forest of binary-clarification trees. Then, we show for FPF forest how the existence of an attack over an instance can be reduced to the existence of a solution for the *partial set coverage problem* [72]. We exploit this result to design a strategy to reduce the cost of the brute-force strategy and provide two efficient *lower-bound* robustness certifications of FPF forests.

Before going into the definition of the brute-force robustness verifier and the efficient robustness verifiers, we recall the definition of the decision tree provided in Section 2.1.2 since the algorithm proposed in this section deeply relies on the tree structure.

Given a tree-based ensemble F, each tree $t \in F$ can be inductively defined as follows: t is either a leaf $\lambda(\hat{y})$ for some label $\hat{y} \in \mathcal{Y}$, or an internal node $\sigma(f, v, t_l, t_r)$ (a.k.a, split), where $f \in \{1, \ldots, d\}$ identifies a feature in $\mathcal{F}$, $v \in \mathbb{R}$ is the threshold for the feature f in that node, and t_l, t_r are left/right decision trees. At test time, an instance $\boldsymbol{x}$ traverses each tree $t \in F$ until it reaches a leaf $\lambda(\hat{y})$, which returns the *prediction* $\hat{y}$, denoted by $t(\boldsymbol{x}) = \hat{y}$. Specifically, for each internal node $\sigma(f, v, t_l, t_r)$, $\boldsymbol{x}$ falls into the left tree t_l if $x^{(f)} \leq v$, and into the right tree t_r otherwise. Given a forest F and an instance $\boldsymbol{x}$, the forest prediction $F(\boldsymbol{x})$ is defined as the most frequent predicted class $\hat{y} = t(\boldsymbol{x})$ by each $t \in F$, for $\boldsymbol{x}$.

9.4.1 Brute Force Robustness Verifier

Given an instance $\boldsymbol{x} \in \mathcal{X}$, the brute-force robustness verification of $\boldsymbol{x}$ for the forest F consists of finding an evasion attack $\boldsymbol{z} \in A_b(\boldsymbol{x})$, such that $F(\boldsymbol{x}) \neq F(\boldsymbol{z})$. With such an algorithm, we can exactly compute the robustness defined in Equation 8.3 of a tree ensemble F over a given set of instances $\mathcal{D}$.

Unfortunately, the size of $A_b(\boldsymbol{x})$ is infinite, so using a naive brute-force algorithm to generate all the possible evasion instances is infeasible. However, we can exploit the inner structure of the decision trees to limit the size of the adversarial perturbation set $A_b\boldsymbol{x}$ by considering only the attacks that are *relevant* for the given forest F, i.e., those attacks that can invert the outcome of a test in some internal nodes of trees in F. We denoted this set of *relevant attacks*, as $A_b(\boldsymbol{x}|F)$.

To understand how $A_b(\boldsymbol{x}|F)$ is generated, recall that nodes in a tree are in the form $x^{(f)} \leq v$ for some threshold v. Indeed, the thresholds used in the tree nodes induce a discretisation of the input space $\mathcal{X}$ that can be exploited for an efficient brute-force algorithm. For any given feature $f \in \mathcal{F}$, we define with $\mathcal{V}_f$ the set of relevant thresholds as follows:

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

The set $\mathcal{V}_f$ includes all the thresholds that are associated with f in any node $\sigma(f, v, t_l, t_r)$ of any tree in F, plus the infinity value that allows the algorithm also

to include the attack that traverses the right branch of the node with the largest threshold.

An attacker A_b can perturb any subset of features $B \subseteq \mathcal{F}$ such that $|B| \leq b$, and therefore the set of relevant perturbations the attacker may operate is described by the Cartesian product $\mathcal{V}_{f_1} \times \ldots \times \mathcal{V}_{f_b}$, with $f_i \in B$. We denote by $A_b(\boldsymbol{x}|F, B)$ the set of relevant attacks on the given set of features B, i.e., each perturbed vector $\boldsymbol{z} \in A_b(\boldsymbol{x}|F, B)$ satisfies the following:

$$z^{(f)} = \begin{cases} x^{(f)} & \text{if } f \notin B, \\ v \in \mathcal{V}_f & \text{if } f \in B. \end{cases}$$

In conclusion, the set of relevant attacks is:

$$A_b(\boldsymbol{x}|F) = \bigcup_{\substack{\forall B \subseteq \mathcal{F}, \\ |B|=b}} A_b(\boldsymbol{x}|F, B)$$

The discretisation introduced by the thresholds of the trees allows us to generate the exact finite set of possible attack $A_b(\boldsymbol{x}|F)$ without loss of generality. Through $A_b(\boldsymbol{x}|F)$, we can efficiently and exactly verify the robustness of F on an instance $\boldsymbol{x}$. Hereinafter, we refer to this efficient robustness verification algorithm as BF.

Given a set of instances $\mathcal{D}$, with BF, it is possible to efficiently compute the exact robustness as defined in Equation 8.3. To make the computation of the robustness even faster, we can focus the attention only on the correctly predicted instances since the wrongly predicted instances are not robust by definition. Thus we define the set of instances for which exists at least a successful evasion attack as $\widehat{\mathcal{D}} = \{(\boldsymbol{x}, y) \in \mathcal{D} \setminus \overline{\mathcal{D}} \mid \exists \boldsymbol{z} \in A_b(\boldsymbol{x}|F), y \neq F(\boldsymbol{z})\}$, where $\overline{\mathcal{D}}$ is the set of instances misclassified by F in absence of attack.

This brute-force approach is still very expensive due to three factors: *i)* as b increases, the number of possible feature combinations $B \subset \mathcal{F}$ with $|B| = b$ increases; *ii)* as the number of trees and nodes grows, the number of threshold values associated with each feature increases; *iii)* for each perturbed instance $\boldsymbol{z}$, the prediction $F(\boldsymbol{z})$ must be computed by traversing each trees of the forest.

9.4.2 Feature-Partitioned Forest Robustness Certifier

Even though the BF algorithm allows the reduction of the infinite adversarial perturbation set $A_b(\boldsymbol{x})$ into the finite relevant attacks set $A_b(\boldsymbol{x}|F)$, it doesn't change the fact that computing all relevant attacks is still infeasible, as outlined in the three points stated previously. Consequently, another significant contribution

of this work is the development of two robustness certifiers that efficiently provide an accurate lower bound of the robustness of FPF forests.

We now introduce some simplifying worst-case assumptions and then show that an effective attack exists if it can be reduced to a solution for the partial set coverage problem. First, we assume that if a tree in F provides a wrong prediction before the attack, then its prediction will be incorrect also after the attack. Second, we assume that if a tree uses a feature f for its prediction over $\boldsymbol{x}$, then attacking f causes the tree to generate a wrong prediction.

Note that these assumptions are pessimistic from the point of view of a defender. Indeed, modifying a feature f does not necessarily flip the test performed at every node of the tree that uses that feature, and even if it was the case, tests performed over other features may suffice to avoid a wrong prediction.

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

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

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

Then, among the nodes along the traversal path of instance $\boldsymbol{x}$ before the attack, we can further distinguish between nodes where the test $x^{(f)} \leq v$ is *true* and nodes

where the test is *false*. In the former case, the attacker must increase the value of $x^{(f)}$ to affect the traversal path, while in the latter case, the attacker must decrease $x^{(f)}$. Clearly, these two attacks cannot coexist.

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

Proposition 3 (Partial set coverage as a necessary condition for effective attacks)**.** Given $(\boldsymbol{x}, y) \in \mathcal{D}$, where $F(\boldsymbol{x}) = y$, a *necessary condition* for the existence of a successful attack $\boldsymbol{z} \in A_b(\boldsymbol{x})$ such that $F(\boldsymbol{z}) \neq y$, is that there exists a solution for the *partial set coverage problem*, stated as follows:

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

Proof of the correctness of Proposition 3. We show that if there exists $\boldsymbol{z} \in A_b(\boldsymbol{x})$ such that $F(\boldsymbol{z}) \neq y$, then there exists $S^* \subseteq \Xi$, where $|\bigcup_{S \in S^*} S| \geq \delta$, $|S^*| \leq b$, and S_f^+ and S_f^- are mutually exclusive in S^*. Given $\boldsymbol{z} \in A_b(\boldsymbol{x})$, we say that for any attacked feature f either the corresponding set S_f^+ belongs to S^* if $z^{(f)} - x^{(f)} > 0$ (corrupted by increment) or S_f^- belongs to S^* if $z^{(f)} - x^{(f)} < 0$ (corrupted by decrement). Clearly, it holds that $|S^*| \leq b$, and the sets S_f^+ and S_f^- are mutually exclusive in S^*. Let $\overline{C'}$ be the set of (formerly correct) trees corrupted by the successful attack $\boldsymbol{z}$; then it holds that $|\overline{C'}| \geq \delta$. By construction, any tree $t \in \overline{C'}$ belongs to either S_f^+ or S_f^- included in S^*. Therefore, it holds that $|\overline{C'}| \leq |\bigcup_{S \in S^*} S|$, which implies $|\bigcup_{S \in S^*} S| \geq \delta$. □

Note that Proposition 3 states that the existence of a solution S^* for our partial set cover problem is only a necessary (not sufficient) condition for the attack. Thus, if S^* exists, we cannot say that F can be fooled for sure, as the attacker might modify all the features identified by the cover without being able to affect the final forest prediction. However, we know that if a solution S^* does not exist, then F is robust on the given instance $\boldsymbol{x}$.

In the following, we use this method to compute an upper bound of the size of $\widehat{\mathcal{D}}$, i.e., the set of instances in the set $\mathcal{D}$ for which there exist a successful evasion

attack. On the one hand, this method allows us to efficiently compute a lower bound of the exact robustness $\mathrm{R}(A_b)$. On the other hand, it makes it possible to speed up the exact computation of $\mathrm{R}(A_b)$ by employing the BF algorithm only for those instances for which a sufficiently large set cover exists, i.e., a successful attack may exist.

9.4.2.1 Fast Robustness Lower Bound Certifier

The method discussed in this section, named Fast Robustness Lower Bound (FLB), computes an overestimate of the size of the partial set cover $S^* \subseteq \Xi$, for the problem stated in Proposition 3.

Consider the b sets in Ξ that provide the largest set cover. Trivially, the cardinality of the union of the b sets within S^* is smaller or equal to the sum of the cardinalities of the b largest sets in Ξ (inclusion-exclusion principle). Due to this approximation, it is possible to run FLB in *polynomial time*. Furthermore, we improve this trivial upper bound by considering that the two sets S_f^+ and S_f^- cannot be included together in a potential cover.

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

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

Therefore, we define the set $\widehat{\mathcal{D}}_{\mathrm{FLB}}$ of attackable instances according to the *fast lower bound* method as follows. For each correctly classified instance $(\boldsymbol{x}, y) \in \mathcal{D}$, we build the partial coverage problem according to Proposition 3, and *iff* $\sum_{S \in S_{\mathrm{FLB}}} |S| \geq \delta$, then we include the instance $(\boldsymbol{x}, y)$ in $\widehat{\mathcal{D}}_{\mathrm{FLB}}$. Since it holds that $|\widehat{\mathcal{D}}_{\mathrm{FLB}}| \geq |\widehat{\mathcal{D}}|$, we define the robustness computed with FLB as follows:

$$\mathrm{R}(A_b)_{\mathrm{FLB}} = 1 - \frac{|\overline{\mathcal{D}}| + |\widehat{\mathcal{D}}_{\mathrm{FLB}}|}{|\mathcal{D}|} \leq \mathrm{R}(A_b).$$

9.4.2.2 Exhaustive Robustness Lower Bound Certifier

In order to improve over FLB, we also consider a more expensive option named *Exhaustive Robustness Lower Bound* (ELB), where all the possible covers are considered, still respecting the constraint that any S_f^+ and S_f^- are mutually exclusive. We evaluate all the possible covers $S^\dagger \subset 2^\Xi$, $|S^\dagger| \leq b$, and we call *exhaustive lower bound cover*, denoted with S_{ELB}, the first cover found such that $|\bigcup_{S \in S_{\mathrm{ELB}}} S| \geq \delta$. If no exhaustive lower bound cover S_{ELB} can be found, then F is robust on the given instance $\boldsymbol{x}$.

By applying the same procedure to every correctly classified instance $(\boldsymbol{x}, y) \in \mathcal{D}$, we identify the set of instances $\widehat{\mathcal{D}}_{\text{ELB}}$ for which there exists an exhaustive lower bound cover S_{ELB} that solves the problem in Proposition 3.

Note that $|\widehat{\mathcal{D}}| \leq |\widehat{\mathcal{D}}_{\text{ELB}}| \leq |\widehat{\mathcal{D}}_{\text{FLB}}|$; thus we use this method to compute another lower bound of the robustness of F on a given set $\mathcal{D}$ as follows:

$$\text{R}(A_b)_{\text{ELB}} = 1 - \frac{|\overline{\mathcal{D}}| + |\widehat{\mathcal{D}}_{\text{ELB}}|}{|\mathcal{D}|} \leq \text{R}(A_b),$$

where the following relationship trivially holds:

$$\text{R}(A_b)_{\text{FLB}} \leq \text{R}(A_b)_{\text{ELB}} \leq \text{R}(A_b).$$

This exhaustive lower bound cover search incurs into the exponential cost of enumerating the possible covers in 2^{Ξ}, but it improves over the brute-force attack, thanks to the cover-based formulation, by ignoring the relevant threshold values $\mathcal{V}_f$ each feature can be attacked. Moreover, it is possible to make ELB even faster by applying it only on the instances in $\widehat{\mathcal{D}}_{\text{FLB}}$, i.e., the instances for which FLB found a set cover.

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

9.4.2.3 Non-Binary Classification Robustness Certifier

Even if this work is focused on binary classification tasks, we highlight that the proposed methodology can be easily generalised to a multi-class scenario, and we sketch below a basic certification methodology.

The algorithms proposed so far aim at certifying the impossibility of the attacker of modifying a number of correct trees δ such that $\delta \geq \lceil |F|/2 \rceil - |\overline{C}|$, where $\overline{C}$ is the set of trees wrongly classifying the given instance. In regard to a multi-class classification problem, given classes $\mathcal{Y}$ with $|\mathcal{Y}| > 2$, it is possible to verify robustness by running $|\mathcal{Y}| - 1$ certifications analogous to the binary case.

Let's denote with C_c the set of trees that, in the absence of attacks, classify a given instance $(\boldsymbol{x}, y) \in \mathcal{D}$ as belonging to class c, with $c \in \mathcal{Y}$; then let C_y be the set of trees predicting the correct label y. For a given class $c \in \mathcal{Y}$, the attacker aims at attacking the trees in $\bigcup_{i \neq c} C_i$ so as to make c the new majority class. The best-case scenario from the point of the attacker is given by modifying the predictions of the trees in C_y, as in this case, it is sufficient to attack $\delta = \lceil (|C_y| - |C_c|)/2 \rceil$ trees. If the attacker is not able to alter at least δ trees of the forest, then no successful attack is possible.

To this end, we can exploit any of the set-cover base techniques proposed so far to verify the absence of any cover of size at least δ among the trees $\bigcup_{i \neq c} C_i$. Such verification is to be repeated for each $c \in \mathcal{Y}$.

9.4.3 Cascade Robustness Verifier

Above, we presented two algorithms, FLB and ELB, that efficiently find an over-approximation of a cover of attacked trees, which allows us to estimate the upper bound of the most harmful attack. The two strategies have different costs: FLB requires sorting the candidate sets of the cover, while ELB performs an exhaustive search of all the possible subsets of Ξ. However, Both methods are much cheaper than the brute-force algorithm BF.

When the lower-bound information is not considered sufficient, in order to compute the exact robustness $\mathrm{R}(A_b)$, we exploit the following Cascading strategy. Given an instance $\boldsymbol{x}$ and an attacker A_b, we build the collection of sets of trees $\Xi = \{S_f^+\}_{f \in \mathcal{F}} \cup \{S_f^-\}_{f \in \mathcal{F}}$ and proceed as follows:

1. compute $S_{\mathrm{FLB}} \subseteq \Xi$: if $\sum_{S \in S_{\mathrm{FLB}}} |S| < \delta$, then no sufficiently large set cover exists, and therefore the instance $\boldsymbol{x}$ cannot be attacked; otherwise
2. search for a *exhaustive lower bound* cover $S_{\mathrm{ELB}} \subseteq \Xi$: if there is no S_{ELB} such that $|\bigcup_{S \in S_{\mathrm{ELB}}} S| \geq \delta$, then the instance $\boldsymbol{x}$ cannot be attacked; otherwise
3. use the BF algorithm to check the existence of a successful attack on $\boldsymbol{x}$.

In the experimental part of this work, we show how the above cascading strategy is able to strongly reduce the number of instances in a given dataset $\mathcal{D}$ for which the brute-force approach is required.

9.5 Experimental Setup

In the experimental part of this work, we verify whether the robustness by construction theoretically guaranteed by the FPF learning algorithm translates into empirically higher robustness in an adversarial scenario compared to the baselines. To achieve this, we used three publicly available datasets: Wine, Breast Cancer D, and Spam Base.

Table 9.1 provides the main characteristics of the datasets, including the number of features, the number of top relevant features measured as those contributing to 90% of the feature importance in a Random Forest, and the majority class percentage. Note that the Wine dataset is provided with three classes; however, to use this dataset, we binarise it by merging the cultivar classes 1 and 2 into one larger class.

Table 9.1: Datasets properties.

Dataset	#features	#top feat.	#instances	%maj. class
Wine	13	7	178	73.0%
Breast Cancer D	30	15	569	62.7%
Spam Base	57	26	4,601	60.6%
MNIST 0 vs. 1	784	54	14,780	53.3%
MNIST 1 vs. 7	784	79	15,170	51.9%
MNIST 5 vs. 6	784	173	13,189	52.1%

In this section, we also extensively analyse the performance of the certification algorithms FLB and ELB in terms of efficiency and accuracy. For these analyses, we also use the MNIST dataset. The MNIST dataset contains images of digits ranging from 0 to 9 for a total of 10 classes. To use this dataset in binary classification, we selected three pairs of visually similar classes, i.e., classes in which the attacker can modify a few pixels to transform one digit into another. The datasets we created are the following: MNIST 0 vs. 1, MNIST 1 vs. 7, and MNIST 5 vs. 6.

9.5.1 Baselines and Implementation

In the experimental phase, we compare our proposed algorithm, FPF, against three tree-based ensemble competitors: Random Forest, Random Subspace Method, and Robust Tree. The algorithms FPF, Random Forest, and Random Subspace Method were implemented using the open-source Scikit-learn library [24]. For Robust Tree, we used the implementation provided by Calzavara *et al.* [36], and the code can be found on GitHub[1]. Finally, the code for FPF with its FLB and ELB certificates, and the efficient brute-force algorithm BF, is also available on GitHub[2].

Below, we provide details of the implemented algorithms.

- Random Forest (RF): The RF algorithm defined by Breiman [20] is not designed to be robust to evasion attacks; however, it is known to have some good level of robustness thanks to the ensembling of several decision trees. As in the original algorithm, each tree is trained on a bootstrap sample of the dataset and with feature sampling of size $\sqrt{|\mathcal{F}|}$ at each node.

[1]https://github.com/FedericoMarcuzzi/TREANT
[2]https://github.com/FedericoMarcuzzi/Feature-Partitioning-for-Robust-Tree-Ensembles

- Random Subspace method (RSM): The RSM algorithm proposed by Tin Kam [81], which was successfully exploited in the adversarial setting by Biggio *et al.* in [13]. In this case, each tree is trained on a projection of the original dataset based on a subset (subspace) of the feature set. RSM has a hyperparameter $p \in [0, 1]$, which constrains the size of the subset of features used for each tree learning to $\lceil p|\mathcal{F}| \rceil$.

- Robust Tree (RT): The RT algorithm, proposed by Chen *et al.* in [45], trains individual decision trees to be robust against evasion attacks by optimising the performance under attack. We created a robust ensemble by training each individual tree with the algorithm proposed in the original work and by ensembling them together with a majority voting prediction.

To untie the model's resistance from the attacker's strength, hereinafter, we refer to b as the strengthening parameter exploited by FPF and RT at training time and to k as the attacker's budget used to generate the evasion attacks.

The ensemble size significantly influences the final robustness of the strategies. We conduct a fair comparison of the different learning strategies by allowing each method to grow approximately 300 trees, a configuration experimentally observed to be optimal.

Each dataset is divided into train, validation, and test sets according to a 60%-20%-20% scheme with a stratified sampling strategy in order to maintain the same distribution of classes in each set. For each learning algorithm, we fine-tune their hyperparameters on the validation set by optimising for robustness $\mathrm{R}(A_k)$. Specifically, we fine-tune the maximum number of leaves $\hat{l}$, selecting from the set $\{4, 8, 16, 32\}$ for all algorithms. Regarding FPF, we fine-tune the hyperparameter b selecting from the set $[1, 5]$, while the number of rounds r is constrained based on b to ensure no more than 300 trees in the ensemble, i.e., $r = \lfloor 300/(2b + 1) \rfloor$. Finally, the RSM algorithm requires a sampling parameter p to constrain the size of the subspaces, and we fine-tune it among the values $p \in \{0.1, 0.2, 0.4, 0.6\}$. We perform model selection in the validation set to find the best model for each attacker A_k, and we report the results computed on the test set.

9.6 Main Results

In this section, we summarise the results of the experiments conducted to address the following questions: Is the FPF learning algorithm capable of training robust tree-based ensembles compared to competitors? How accurate are the proposed robustness lower bounds FLB and ELB? How efficient is the Cascading strategy for exact robustness computation compared to the brute-force algorithm BF?

Table 9.2: Methods performance in terms of accuracy (percentage) ACC and robustness (percentage) R(A_k) on WINE dataset. The highest robustness R(A_k) for each k is marked with **bold**.

model	hyperparameters					performance			
	b	r	p	$\hat{l}$	$\lvert F\rvert$	ACC	k	R(A_k)	ΔFPF
RF				8	300	94.4	1	83.3	-2.8
				4	300	100.0	2	38.9	-36.1
				4	300	100.0	3	0.0	-61.1
RSM			0.2	8	300	94.4	1	83.3	-2.8
			0.2	4	300	97.2	2	72.2	-2.8
			0.2	8	300	94.4	3	44.4	-16.7
RT	1			8	300	73.8	1	73.8	-12.3
	2			8	300	73.8	2	73.8	-1.2
	3			8	300	73.8	3	**73.8**	+12.7
FPF	2	60		8	300	97.2	1	**86.1**	
	5	27		8	297	88.9	2	**75.0**	
	5	27		8	297	88.9	3	61.1	

The section is divided into two parts. The first part focuses on comparing the robustness and accuracy of the models produced by each algorithm. The second part concentrates on the accuracy of the certificates in computing the lower bound of the robustness of the forests trained with FPF and the efficiency introduced by the certificates for computing the exact robustness with the Cascading strategy.

9.6.1 Model Robustness

We measured the robustness of the models against an attacker A_k with three different values of the budget $k \in \{1, 2, 3\}$. Note that the robustness can be seen as the accuracy of the model under attack. So, for the sake of simplicity hereinafter, we will use the terms accuracy and robustness interchangeably when the model is under attack.

In tables 9.2, 9.3, and 9.4, are summarised the results for each dataset. The results are computed on the best models selected through model selection on the validation set, and the best hyperparameters are reported for each model.

Each table provides the exact robustness R(A_k) calculated as in Equation 8.3. We used the exact brute-force algorithm BF defined in Section 9.4.1 to generate all the attacks. Alongside the robustness, there is the accuracy ACC to compare the

Table 9.3: Methods performance in terms of accuracy (percentage) ACC and robustness (percentage) R(A_k) on Breast Cancer D dataset. The highest robustness R(A_k) for each k is marked with **bold**.

model	hyperparameters					performance			
	b	r	p	$\hat{l}$	$\lvert F\rvert$	ACC	k	R(A_k)	ΔFPF
RF				4	300	93.0	1	87.7	-3.5
				8	300	94.7	2	84.2	-0.9
				4	300	93.0	3	61.4	-20.2
RSM			0.2	16	300	95.6	1	**91.2**	0.0
			0.2	16	300	95.6	2	83.3	-1.8
			0.2	16	300	95.6	3	65.8	-15.8
RT	1			4	300	92.3	1	56.2	-35.0
	1			4	300	92.3	2	50.5	-34.6
	1			4	300	92.3	3	45.4	-36.2
FPF	3	42		8	300	93.9	1	**91.2**	
	4	33		4	297	93.0	2	**85.1**	
	5	27		8	300	93.0	3	**81.6**	

Table 9.4: Methods performance in terms of accuracy (percentage) ACC and robustness (percentage) R(A_k) on Spam Base dataset. The highest robustness R(A_k) for each k is marked with **bold**.

model	hyperparameters					performance			
	b	r	p	$\hat{l}$	$\lvert F\rvert$	ACC	k	R(A_k)	ΔFPF
RF				8	300	92.3	1	76.6	-7.1
				4	300	90.2	2	44.9	-33.3
RSM			0.2	16	300	90.9	1	**83.7**	0.0
			0.2	16	300	90.9	2	73.6	-4.6
RT	3			8	300	71.5	1	43.3	-42.2
	3			8	300	71.5	2	20.1	-65.5
FPF	2	60		16	300	91.8	1	**83.7**	
	4	33		16	297	86.6	2	**78.2**	

models' performance in the absence of attacks. Finally, the rightmost column in each table reports the difference in robustness (i.e., ΔFPF) between a FPF forest and a competitor under the same attack (i.e., the same value of k). Note that for the SPAM BASE dataset, we limit the analysis to a maximum of $k = 2$, i.e., at most two attacked features. The reason for that is the prohibitive cost of computing the adversarial perturbation set $A_3(\boldsymbol{x})$ for each instance of the test set.

In all datasets, Random Forest performs best or second best in the absence of attacks, providing the highest accuracy. Under attack, the accuracy (robustness) of Random Forest drops significantly. The perturbation of a single feature is enough to force the model to lose up to 15% of accuracy across all datasets. When attacking three features, the accuracy drops to zero for WINE and to about 60% for BREAST CANCER D. As previously discussed, the L_0-norm attacks can be successful even by modifying a few features. Consequently, it is easy to fool a very accurate random forest model which is not adversarially trained.

The RSM is the other learning algorithm that provides the most accurate model in the absence of attacks. The model trained with RSM is overall much more robust than Random Forest in the presence of attacks, suggesting that training each ensemble's tree on a different dataset projection is advantageous in adversarial scenarios. However, when attacking two features, RSM also exhibits a drop in accuracy up to 27%, with the most significant decrease in the SPAM BASE dataset. When attacking three features, the accuracy of the RSM models drops below 45% on WINE and 66% on BREAST CANCER D.

The results obtained by RT models show how this robust learning strategy does not perform well in the case of L_0-norm attacks. This can be explained by observing how the learning algorithm trains a robust model. During the training phase, the model has to choose the feature-threshold pair (f, v), which maximises the minimum accuracy under attack, i.e., the robustness. However, an attacker like A_k constrained by the L_0-norm and a budget k can always modify any combination of k features as much as it wants and, therefore, can always cross the algorithm's chosen threshold. Consequently, the evasion attacks generated by A_k make it extremely difficult for the training algorithm to choose the best pair since pairs with the same feature generate the same information gain under attack. The difficulty of the algorithm in dealing with this type of attack can be seen from the WINE dataset, where the algorithm always returns the same accuracy and robustness, and these exactly coincide with the majority class percentage of the test set. The reason behind that lies in the fact that during the training phase, the RT algorithm funds more advantages for maximising the robustness always to return the majority class rather than learning a more accurate tree.

Finally, looking at the results obtained by the models trained by FPF learning algorithm, we can observe that it provides the best robustness in the majority

Table 9.5: Robustness lower bound analysis on Breast Cancer D for a FPF forest with $|F| \approx 300$ and maximum number of leaves $\hat{l} = 8$.

	$R(A_1)_{algo}$			$R(A_2)_{algo}$			$R(A_3)_{algo}$		
hyperparameter b	BF	ELB	FLB	BF	ELB	FLB	BF	ELB	FLB
1	91.2	88.6	88.6	76.3	0.0	0.0	18.4	0.0	0.0
2	91.2	90.4	90.4	84.2	83.3	82.5	64.9	0.0	0.0
3	91.2	91.2	91.2	86.0	85.1	84.2	78.1	76.3	73.7
4	90.4	90.4	90.4	86.8	86.8	86.8	79.8	79.8	79.8
5	89.5	89.5	89.5	86.8	86.8	86.8	81.6	81.6	80.7

of the cases. We highlight that the best defensive b found is always larger than the attacker's budget k, meaning that increasing the number of feature partitions provides better robustness than exactly optimising the attacker's strength. The scenario where the attacker has a budget of $k = 3$ is particularly interesting. The FPF models have up to 20%, 35% and 60% higher robustness compared to the RSM, RT, and RF models, respectively.

Furthermore, the results on the Wine dataset highlight the ability of FPF to guarantee greater robustness even with few features in the dataset. As reported in Table 9.1, Wine has 13 features, of which only 7 are relevant. Attacks executed with budgets k equal to 1, 2, and 3 compromise the 7.7%, 15.4%, and 23.1% of the features of the dataset, respectively, and 14.3%, 28.6%, and 42.8% if we consider only the relevant features. For each budget value, FPF models showed a greater (or equal) robustness than the competitors.

From these results, we conclude that FPF outperforms the competitors, especially when considering stronger attackers.

9.6.2 Robustness Certifier

In this section, we analyse in detail the accuracy of the FLB and ELB certifiers and the efficiency of the Cascading strategy. To perform these analyses, we utilised the efficient brute-force algorithm BF as defined in Section 9.4.1.

9.6.2.1 Accuracy

To assess the reliability of the FLB and ELB certificates in estimating the robustness lower bound of FPF forests, we compare them with the BF algorithm on the Breast Cancer D dataset.

Specifically, we compare the robustness $R(A_k)$ calculated by each algorithm, BF, ELB, and FLB, on forests trained with FPF. To better explore the capability

Table 9.6: Robustness verification per-instance average execution time on Breast Cancer D. FPF forest with $|F| \approx 300$, hyperparameter $b = 3$, and maximum number of leaves $\hat{l} = 8$.

method	run time	R(A_1)	$\lvert\widehat{\mathcal{D}}\rvert$	run time	R(A_2)	$\lvert\widehat{\mathcal{D}}\rvert$	run time	R(A_3)	$\lvert\widehat{\mathcal{D}}\rvert$
BF	263.6 ms	91.2	100	1053.4 ms	84.2	100	3635 s	78.1	100
Cascading	27.1 ms			166.0 ms			36.1 s		
each:									
–FLB	2.7 ms	91.2	100	3.2 ms	84.2	100	3.3 ms	73.7	100
–ELB	0.8 ms	91.2	2	2.0 ms	85.1	8	29.7 ms	76.3	20
–BF	23.6 ms	91.2	2	160.8 ms	86.0	7	36.0 s	78.1	18

of the algorithms, we vary the training parameter b from 1 to 5 and the attacker's budget k from 1 to 3. The results of this analysis are reported in Table 9.5.

Firstly, we observe that with $k = 1$, the robustness estimations provided by FLB and ELB are the same and exact in most settings. With $k = 2$, a slight difference between ELB and FLB is observed, where ELB provides a slightly better robustness estimation. However, both remain accurate in the estimation of the robustness. For smaller values of b, i.e., $b \leq 3$, the gap between the two lower bounds increases with the attacker's budget k. Finally, ELB always provides the exact robustness with $b \geq 4$ for every value of k, whereas FLB when $k \leq 2$. Note that, by definition of the lower bound algorithms, when $k > b$, the certifiers estimation is useless, as there always exists a sufficiently large set cover to attack the majority of the forest; consequently, both FLB and ELB estimate robustness equal to zero.

From the results, we conclude that both certificates provide an accurate robustness lower bound of FPF forests. Moreover, while FLB has a slightly lower robustness estimation than ELB, it has a significant advantage in terms of efficiency, providing an accurate robustness estimation in polynomial time.

9.6.2.2 Efficiency

In this section, we analyse how the efficiency of the FLB and ELB certifiers can be effectively exploited in a Cascading strategy to efficiently estimate the exact robustness of a FPF forest.

In detail, given the test set $\mathcal{D}$ of the Breast Cancer D dataset, we collect the per-instance average time to compute the exact robustness of a FPF forest F with the BF algorithm and the Cascading strategy. The Cascading strategy executes in cascade, from the most efficient (i.e., FLB) to the most accurate (i.e., BF), the robustness certification/verification algorithms to efficiently estimate the

robustness of F over the set $\mathcal{D}$. As explained in Section 9.4, FLB is the most efficient certifier among the two provided in this work; however, it is also the one that mostly underestimates the robustness of the models. Therefore, all instances for which FLB cannot certify the absence of attacks, i.e., $\widehat{\mathcal{D}}_{\text{FLB}}$, are given as input to the slower but more accurate ELB certifier. However, ELB also provides an approximate robustness estimation, so the instances not certified as secure, i.e., $\widehat{\mathcal{D}}_{\text{ELB}}$, are eventually given as input to the exact brute-force BF algorithm. Finally, the results of all three algorithms are combined to compute the exact robustness of the forest on the test set $\mathcal{D}$.

The results of this analysis are reported in Table 9.6. The table reports the per-instance average time required to run the brute-force algorithm BF and the proposed Cascading strategy. For the Cascading strategy, we provide a breakdown of the individual contributions of each algorithm, including the average execution time per instance, the estimated robustness, and the percentage of instances processed from the test set $\widehat{\mathcal{D}}$. Note that $\widehat{\mathcal{D}}$ contains only the correctly classified instances by F in the absence of an attack.

The results show that the computational cost required by BF exponentially increases when increasing k. As expected, the robustness estimation with the BF algorithm quickly becomes infeasible. The proposed Cascading strategy provides a 10× speed-up that increases to 100× when $k = 3$. This huge gap is due to the efficiency and accuracy of the proposed lower bounds. On varying k, the fraction of instances for which FLB cannot certify the non-attackability is respectively 2%, 8%, and 20%. These instances are processed during the ELB step, which leaves to the last BF step the 2%, 7%, and 18%, respectively, of instances to be analysed. The execution of the BF algorithm on these last instances largely covers from 85% to about 100% of the total running time, while the FLB and ELB steps are two or more orders of magnitude faster. We thus conclude that the proposed FLB and ELB are both sufficiently accurate, as discussed in the previous section, and they can be used in a Cascading-like strategy to provide significant speed-ups to any other robustness verification method for an exact robustness estimation.

The last efficiency analysis we perform aims to highlight the power of the FLB in certifying the robustness lower bound of FPF forests in scenarios where an exact robustness computation with the BF algorithm is infeasible. Specifically, we compute the robustness lower bound using FLB on the Wine, Breast Cancer D, and Spam Base datasets, varying the FPF training hyperparameter b and the attacker's budget k.

In this analysis, we also consider three binarised classification datasets generated from MNIST, namely MNIST 0 vs. 1, MNIST 1 vs. 7, and MNIST 5 vs. 6. These datasets allow us to provide a more interpretable explanation of the effect of an evasion attack on input instances. While it is challenging to

understand the meaning of perturbing any combination of k features of a feature vector representing 13 different constituents, as in the case of the WINE dataset, it is much easier to visualise the effect of perturbing k pixels of an image representing a digit.

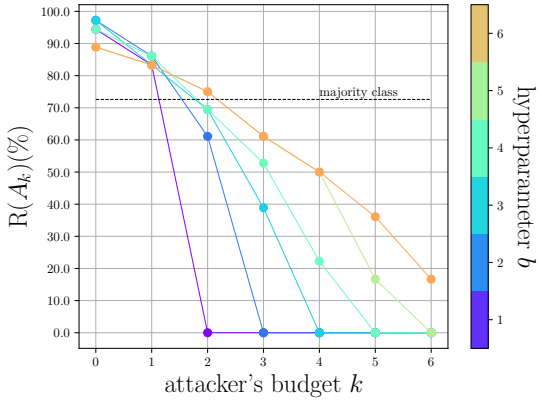
Furthermore, the MNIST datasets encompass a much larger number of features, making robustness computation with brute-force approaches infeasible. Indeed, the computational efficiency of the proposed certifiers allows us to compute the robustness lower bound for large values of the attacker's budgets k. We demonstrated the robustness of FPF forests against attacks executed with a budget up to $k = 100$, i.e., attacks over any combination of 100 features out of 784. This is an extremely high number that is intractable for the BF algorithm.

We summarise the results for each dataset in Figure 9.1. The x axis represents the robustness $\mathrm{R}(A_k)$, while the y axis the number of attacked features k, i.e., the attacker's budget. The colours of the curves and the colour map on the right side of each plot identify models trained with different values of the training parameter b. The black dashed line indicates the majority class percentage in the test set.

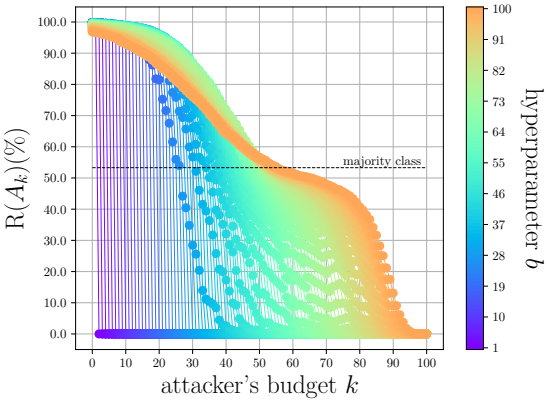
The figures show that for larger values of b, FPF forests can sustain a larger attacker strength. For example, in BREAST CANCER D dataset, it is possible to train a model with robustness 80% when 4 features out of 30 (or out of 15 relevant features) are attacked. However, when the attacker's strength becomes too significant compared to the number of relevant features in the dataset, the model's robustness significantly drops under the majority class line.

Intuitively, among the three MNIST datasets, MNIST 0 VS. 1 is the one where an attacker needs to modify the higher number of features, i.e., pixels of the image, to transform a 0 digit into a 1, and *vice versa*. As shown in Figure 9.1b, for this dataset, it is possible to train FPF forests that guarantee 95% accuracy (i.e., robustness) with 20 attacked features and 90% with 30 attacked features. In Figure 9.1d, for MNIST 1 VS. 7, where it is easier to transform a 1 into a 7, attacking 10 features is sufficient to reach 95% of robustness. Finally, the MNIST 5 VS. 6 dataset is the easiest to attack because, graphically, the digits 5 and 6 are very similar, and the attacker can modify fewer features to transform one digit into the other. This is evident in Figure 9.1f, where with just 10 attacked features (i.e., pixels), the robustness decreases to 90%.

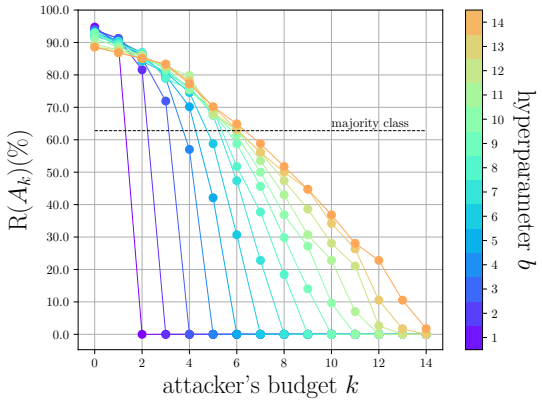
With this final analysis, we demonstrate how FLB can be used to efficiently analyse the behaviour and robustness of FPF forests in extreme adversary scenarios where using an exact robustness verifier would be prohibitive. Furthermore, the results align with our expectations of the ease or difficulty of performing an attack that makes an instance very similar to the instances belonging to the target class. This shows that FLB is also accurate in estimating the behaviour of the robustness of FPF forests under attack.



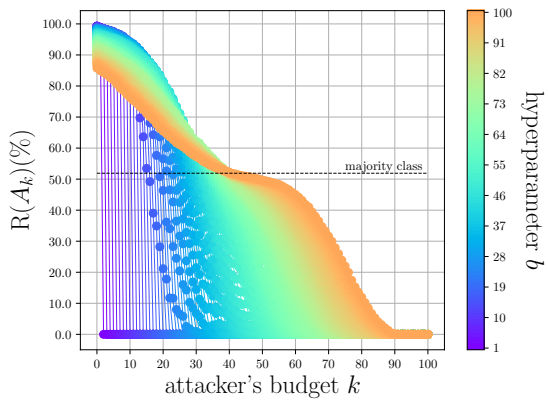
(a) WINE, $|F| \approx 3000$, $\hat{l} = 8$.



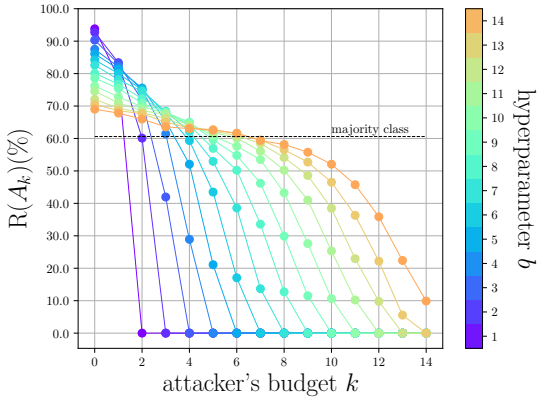
(b) MNIST 0 VS. 1, $|F| \approx 500$, $\hat{l} = 16$.



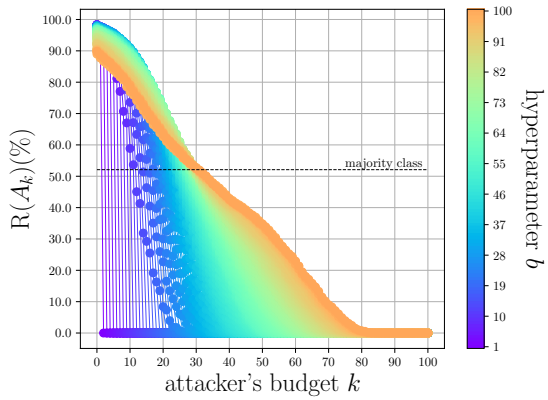
(c) BREAST CANCER D, $|F| \approx 1000$, $\hat{l} = 8$.



(d) MNIST 1 VS. 7, $|F| \approx 500$, $\hat{l} = 16$.



(e) SPAM BASE, $|F| \approx 1000$, $\hat{l} = 32$.



(f) MNIST 5 VS. 6, $|F| \approx 500$, $\hat{l} = 32$.

Figure 9.1: Robustness $\mathrm{R}(A_k)$ computed with FLB by varying b and k.

9.7 In-Depth Analysis

In this final section of the experimental part, we conduct an in-depth analysis of the hyperparameters b and r of the FPF learning algorithm. Additionally, we provided a theoretical analysis of the robustness of robust forests trained with a single round of FPF. Specifically, we formalised the probability of a robust forest $F_{\mathcal{P}}$ to predict the correct label for an input instance under attack while considering different values of the attacker's budget and correct-tree-prediction probability.

9.7.1 Hyperparameter Analysis

The FPF learning algorithm has two hyperparameters. The first is the hyperparameter b, representing the maximum number of features the FPF forest tolerates to be attacked while still ensuring theoretical robustness by construction. The second hyperparameter is r, indicating how many iterations of robust partitioning to perform and how many robust sub-forests to include in the final ensemble. Recall that each round r adds a robust forest $F_{\mathcal{P}}$ containing $2b+1$ trees to the final forest F. For each combination of hyperparameter values r and b, we train a model and collect the robustness on the validation set for different values of the attacker's budget k. Please note that, for both tables, A_0 corresponds to the model's accuracy in the absence of an attack (i.e., A_k with $k = 0$). For both hyperparameters, we conduct the analysis on the Breast Cancer D dataset, and the results are presented in two tables.

In Table 9.7, we evaluate the effect of varying the number of rounds r on FPF forests for different values of b and the attacker's budget k. For each b and k configuration, the highest robustness obtained among different r values is highlighted in bold to emphasise the effect of the r parameter in various attack scenarios. As seen in the results, in the majority of cases, executing more iterations r allows for the training of more robust models, with robustness increasing with the number of rounds until reaching a plateau. We conclude that using a large number of rounds r enhances the robustness of the trained model.

The results of the analysis on hyperparameter b are provided in Table 9.8. For each forest size $|F|$ and attacker's budget k, we highlight in bold the highest robustness provided by the corresponding b values. This emphasises the effect of the hyperparameter b as the intensity of the attack varies and helps to understand whether it is always better to choose the largest possible b or whether it should instead be chosen based on the attacker's budget k.

To better understand the results, recall that the Breast Cancer D dataset has only 15 informative features, which are challenging to partition into $2b + 1$ sets for large values of b. In this case, the results are mixed, showing two different trends. For attacks generated by A_2 and A_3, increasing b improves robustness, and

using $b \geq 4$ is always worthwhile. When the attacker's budget is limited to $k = 1$, the best configuration depends on the number of trees. When the number of trees is limited, the robustness increases with b. However, when the forest is sufficiently large, increasing the number of trees outweighs the benefit of increasing b.

In conclusion, while increasing the number of rounds r to increase the robustness is always beneficial, increasing b when the total number of trees is small, and the number of informative features is limited is not always a good strategy.

Table 9.7: Sensitivity analysis of FPF forests to the hyperparameter r on the Breast Cancer D dataset. Best results in bold for each value of A_k and b.

A_k	r	hyperparametr b				
		1	2	3	4	5
0	1	**95.6**	93.0	**94.7**	**95.6**	93.0
	15	93.9	**93.9**	93.9	93.9	**93.9**
	30	93.9	**93.9**	93.9	93.9	93.0
1	1	84.2	90.4	91.2	89.5	**91.2**
	15	**92.1**	**93.0**	**92.1**	**90.4**	89.5
	30	**92.1**	**93.0**	**92.1**	**90.4**	89.5
2	1	0.0	75.4	83.3	82.5	84.2
	15	67.5	84.2	**86.0**	**86.8**	86.0
	30	**71.9**	**93.9**	**86.0**	**86.8**	**86.8**
3	1	0.0	12.3	67.5	72.8	74.6
	15	11.4	40.4	**77.2**	**80.7**	**81.6**
	30	**15.8**	**47.4**	76.3	80.7	**81.6**

9.7.2 Theoretical Analysis

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

Table 9.8: Sensitivity analysis of FPF forests to the hyperparameter r on the Breast Cancer D dataset. Best results in bold for each value of k and $|F|$.

$\|F\|$	A_k	b 1	2	3	4	5
50	0	93.9	93.9	94.7	93.9	**95.6**
	1	90.4	92.1	91.2	**93.0**	92.1
	2	73.7	85.1	**86.8**	**86.8**	86.0
	3	10.5	40.4	72.8	**82.5**	**82.5**
75	0	**93.9**	**93.9**	**93.9**	**93.9**	**93.9**
	1	91.2	**93.0**	91.2	90.4	90.4
	2	71.9	84.2	**86.8**	86.0	**86.8**
	3	16.7	40.4	75.4	**81.6**	**81.6**
100	0	**93.9**	**93.9**	**93.9**	**93.9**	**93.9**
	1	**93.0**	92.1	92.1	90.4	90.4
	2	46.5	84.2	86.0	**88.6**	86.8
	3	15.8	48.2	76.3	79.8	**81.6**

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

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

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

Proposition 4 (Computing the number of attacks $\widehat{U}_H$)**.** Let K_i be the set of all attacks in U_H that *do not* overlap with $P_i \in H$, and let $\bar{K}_i$ be the complement of K_i in U_H. The following equation holds:

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

Proof of the correctness of Proposition 4. Let C be a possible attack, i.e., a tuple of b features chosen from the sh features of the h partitions in H. For every $C \in \widehat{U}_H$, by definition C contains h features from h different partitions in H, and therefore C has non empty intersection with every $\bar{K}_i$. This proves $\widehat{U}_H \subseteq \cap_{i=1}^{h} \bar{K}_i$. To also prove that $\cap_{i=1}^{h} \bar{K}_i \subseteq \widehat{U}_H$, let's suppose that $C \in \cap_{i=1}^{h} \bar{K}_i$, and that, by contradiction, C does not overlap with h partitions as C does not contain any feature in partition $P_j \in H$. This implies that $C \in K_j$ since K_j contains by construction all the tuples of b features that *do not* overlap with $P_j \in H$. Since $C \in K_j \Rightarrow C \notin \bar{K}_j$, which in turn implies that $C \notin \cap_{i=1}^{h} \bar{K}_i$. This contradicts our hypothesis and concludes the proof. □

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

$$\Pr(h) = \frac{\sum_{H \subset P, |H|=h} \left|\widehat{U}_H\right|}{\binom{d}{b}} = \frac{\binom{2b+1}{h} \left|\bigcap_{i=1}^{h} \bar{K}_i\right|}{\binom{d}{b}}, \tag{9.2}$$

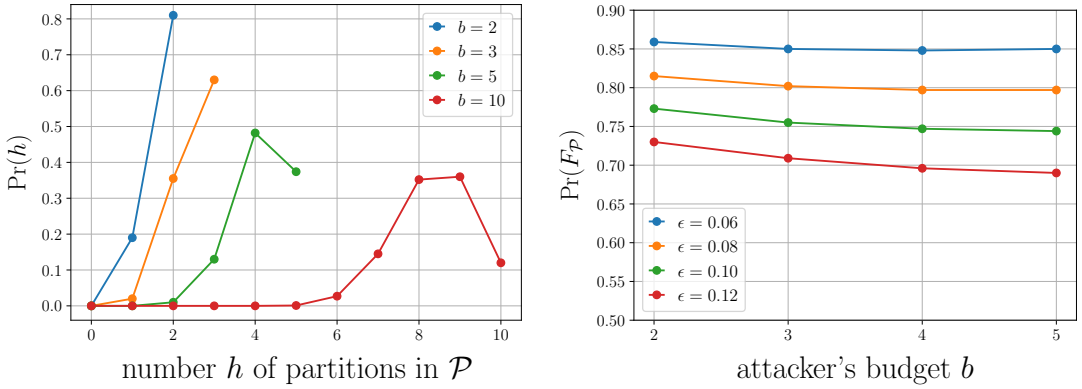
where the factor $\binom{2b+1}{h}$ counts the number of ways of selecting $H \subset \mathcal{P}$, with $|H| = h$, while the denominator $\binom{d}{b}$ is the number of all the possible attacks on the full feature set.

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

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

We can rewrite the above formula as follows:

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



(a) Expected harmed partitions.

(b) Expected ensemble's robustness.

Figure 9.2: Expected attacker effect.

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

We can finally conclude by rewriting the formula in Eq. 9.2:

$$\Pr(h) = \frac{\binom{2b+1}{h}\sum_{k=0}^{h}(-1)^k\binom{h}{k}\binom{s(h-k)}{b}}{\binom{d}{b}} \tag{9.3}$$

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

$$\Pr(e|h) = \binom{|\mathcal{P}|-h}{e-h}\epsilon^{e-h}(1-\epsilon)^{|\mathcal{P}|-e}. \tag{9.4}$$

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

$$\Pr(F_{\mathcal{P}}) = \sum_{e=1}^{b}\sum_{h=1}^{e}\underbrace{\Pr(h)}_{attacked}\underbrace{\Pr(e|h)}_{inaccurate}. \tag{9.5}$$

Figure 9.2a shows the probability $\Pr(h)$ for different values of b. In this study, the number of features in the dataset is chosen to have the multiple of $2b + 1$ closest to 100. We highlight that the probability of "hitting" only a few partitions is usually small. When $b = 5$, there are 11 partitions, and there is less than 15% probability of hitting 3 partitions and about 85% of hitting at least 4 partitions; when $b = 10$, there is nearly 0 probability of hitting less than 6 partitions out of 21, and there is about 80% probability of hitting at least 8 partitions. Interestingly, hitting $b = 10$ partitions is not the most probable event.

We are hypothesising that our attacker is very powerful, as it can likely impact features used by a significant portion of trees in a forest $F_{\mathcal{P}}$ by breaking these trees. This is partially confirmed in Figure 9.2b, where the error rate probability ϵ of trees is also considered. The expected accuracy $\Pr(F_{\mathcal{P}})$ is not large, but, interestingly, increasing the attacker's budget does not have a significant impact. It is true that the attacker can harm a good number of partitions, yet the majority of them are expected to provide correct results, even considering e.

9.8 Summary

In this section, we summarise the main contributions of the research presented in this chapter, titled "Feature Partitioned Forests".

- **Evasion Attaks:** Machine learning systems are affected by adversarial machine learning attacks, where a malicious entity attempts to force the system to exhibit erroneous behaviour. Evasion attacks are well-known attacks in the adversarial machine learning domain. The attacker performs an evasion attack by perturbing a legitimate instance to induce the model to make an incorrect prediction. Our contribution focussed on designing learning algorithms for training tree-based binary classification ensembles robust to evasion attacks. The adversary's model we used is modelled through the L_0-norm, where the legitimate instance and the evasion instance differ at most by k, i.e., the attacker's budget k.

- **Contribution:** The main contributions of this work can be divided into two parts. The first part involves the development of FPF, a learning algorithm designed to train ensembles robust by construction against evasion attacks. FPF trains forests of decision trees through random equi-partitioning of the feature set, along with a projection of the dataset onto these partitions before training each individual decision tree. This robust partitioning ensures that less than half of the forest is involved during an evasion attack. The second contribution comprises the development of two efficient and accurate certification algorithms for estimating a lower bound on the robustness of forests trained with FPF. These algorithms are Fast Robustness Lower Bound (FLB) and Exhaustive Robustness Lower Bound (ELB). Additionally, we have designed an efficient brute-force algorithm called BF for robustness verification, leveraging the internal structure of trees to discretise the set of all possible attacks. Lastly, we combined FLB, ELB, and BF into a cascading strategy for efficient robustness verification of FPF forests.

- **Main Results:** Through extensive experiments, we demonstrated that forests trained with FPF exhibit greater robustness against evasion attacks, outperforming the models trained by both robust and non-robust learning algorithms. Furthermore, we showed that FLB and ELB are accurate robustness certification algorithms, providing a tight lower bound on the exact model's robustness. Additionally, we demonstrated that the cascade strategy significantly reduces robustness verification times, even by two orders of magnitude, compared to solely executing an efficient brute-force algorithm like BF.

- **In-Depth Analysis:** Through an in-depth hyperparameter analysis, we empirically demonstrated that the robustness of FPF forests increases with

the number of robust sub-forests within the ensemble, i.e., the number of iterations of the FPF algorithm. On the other hand, we showed that the number of partitions used to divide the feature space must be proportional to the attack's budget. Excessive partitioning of the feature space leads to less accurate individual trees, compromising the accuracy and robustness of the final ensemble, especially when the dataset has a limited number of relevant features.

9.8.1 Future Work

In this concluding section, we delineate prospective paths for future research and extensions, building upon the results and ideas from this work.

- **Different Adversary Models's:** In this work, we employed adversary's models based on the L_0-norm, enabling the attacker to target any combination of a subset of features with any desired intensity. However, this type of attack may not always be realistic or feasible in real scenarios. A potential extension of this research involves exploring more complex adversary's models constrained by norms different from L_0-norm or exploiting rewriting rules [36].

- **Other Learning Algorithms:** One of the most straightforward extensions of this work is to exploit the main idea beyond robust feature partitioning and dataset projection provided by FPF to enhance the robustness of non-tree-based ensembles, such as ensembles of neural networks or other base learners. Additionally, investigate the combined effect of FPF's robust feature partitioning with learning algorithms to train robust base learners such as Adversarial Boosting [92] or Robust Split [45].

- **Multi-class classificaiton:** Although in Section 9.4, we provided a solution to exploit the main idea behind FLB and ELB certificates to create a certification method for multi-class classification, the proposed FPF algorithm still works exclusively on binary classification. Consequently, another promising extension of this work is to adapt the FPF learning algorithm to multi-class classification problems. In particular, it would be worth it to explore how robust feature partitioning and dataset projection can be effectively exploited to handle multi-class classification while maintaining the models' robustness by construction in adversarial scenarios.

Chapter 10

Beyond Robustness

In this chapter, we discuss the work titled "*Beyond robustness: Resilience verification of tree-based classifiers*", in proceedings as a full paper at the *Computers & Security, 2022*. Further details can be found in the reference [31].

In this study, we delve into the scope of quantifying the security of machine learning systems in adversarial scenarios where there is an attacker performing evasion attacks. As explained above, among the various methods to evaluate the security of machine learning models against evasion attacks, an important metric is Robustness (defined in Equation 8.3).

Robustness is certainly an intuitive and desirable property to estimate the performance of classifiers deployed in adversarial settings, yet it is sub-optimal because it is strongly dependent on the choice of a specific input $\boldsymbol{x}$. While the performance of classifiers must indeed be empirically estimated on a set of correctly labelled inputs (test set), such inputs are normally assumed to be sampled from an underlying data distribution, and robustness says nothing about unsampled data. In other words, a robustness proof for $\boldsymbol{x}$ does not provide any guarantee about any other input $\boldsymbol{x}'$ close to $\boldsymbol{x}$ which could have been sampled in place of it. This is concerning because the actual inputs of the classifier at operational phase time will be different samples drawn from the same distribution of $\boldsymbol{x}$, which are not covered by standard security assessments based on robustness. This problem has been independently acknowledged in very recent work on *global robustness* properties by Chen *et al.* in [49] and Leino *et al.* in [100], which advocates the need for verification techniques establishing robustness guarantees on all the possible inputs provided to the classifier. Unfortunately, these efforts are still at an early stage, and there is no uniform "one size fits all" definition of global robustness that can be readily used to verify the security of classifiers. The work introduced in this Chapter falls in the same research line of global robustness properties, yet it takes a different direction to better complement the extensive amount of work on robustness verification.

In particular, in this Chapter, we propose a generalisation of robustness, called *resilience*, designed to make the security assessment of classifiers more reliable. Resilience generalises traditional robustness guarantees from a specific test set to all the other possible test sets which could have been sampled in place of it, i.e., which are close to it given an appropriate definition of neighborhood. Resilience thus provides a more conservative account of the security of classifiers than robustness while retaining its intuitive flavour. Most importantly, the connection between resilience and robustness allows one to leverage traditional tools for robustness verification as the first step of a resilience verification pipeline, thus integrating with significant research efforts spent on robustness verification.

The main contributions of this work are two. First, we criticise the traditional Robustness measure used to estimate the security of classifiers against evasion attacks, and we propose an improved measure called Resilience. We then discuss how resilience can be estimated by combining an arbitrary robustness verification technique with a *data-independent stability analysis*, which identifies a subset of the feature space where the classifier does not change its predictions despite adversarial perturbations at test time. Then, we present a simple technique to turn any classifier into a globally robust classifier (in the sense of Chen *et al.* in [100]) by leveraging such data-independent stability analysis, thus clarifying the connections with recent work in the area.

Second, we propose a data-independent stability analysis for decision trees and tree-based ensembles. The stability analysis is based on *symbolic attacks*, i.e., symbolic representations of a set of instances, along with their (relevant) adversarial perturbation sets. These representations facilitate the analysis of tree-based classifiers independently of a specific test set. Our analysis is proved sound and can be readily leveraged to establish both robustness and resilience proofs for tree-based classifiers (Section 10.4).

In the experimental part, we show that resilience verification is both useful and feasible in practice, yielding a more reliable security assessment of classifiers deployed in adversarial settings. In particular, our experiments show that robustness can be significantly affected by the choice of a specific test set; hence, it may give a false sense of security. On the other hand, resilience is effective at discriminating between secure models and models that turned out to be robust just *by accident*, i.e., thanks to a lucky, specific sampling of the test set. Finally, our work showed why resilience is superior to robustness in terms of practical security guarantees and showed that even robust-trained models are not necessarily resilient.

The chapter is organised as follows. In Section 10.1, we provide an overview of related work in the field of adversarial machine learning and global robustness verifiers for tree-based classifiers. Section 10.2 defines the threat model and adversary's model used in this work. Section 10.3 introduces the first main contribution

of this work, namely the introduction of the metric *resilience*, highlighting its differences with robustness and stability. Section 10.4 presents the second main contribution, which is the definition of the data-independent stability analyser for tree-based classifiers. Section 10.5 provides a detailed discussion of the implementation of the data-independent stability analyser and the experimental setup used to verify the usefulness of the resilience metric. In Section 10.6, we delve into the main results, demonstrating that robustness offers only a local view of robustness, leading to a false sense of security, while resilience provides a better estimate of model security under attack. Section 10.7 includes an analysis of the efficiency of the data-independent stability analyser implementation and provides correctness proofs for the theorems on which the data-independent stability analyser is based. Finally, in Section 10.9, we summarise the contributions, results, and potential future directions of this study.

10.1 Related Work: Security Verification

Recent independent works in the domain of adversarial machine learning acknowledged the limitations of robustness for the security verification of classifiers [49, 100]. Chen *et al.* in [49] defined a set of five new *global robustness* properties, i.e., universally-quantified statements over one or more inputs to the classifier and its corresponding outputs. These properties also include a data-independent stability definition that requires any two inputs differing just for the value of a fixed set of features to lead to "close" predictions. The authors also proposed a technique to verify this property for a custom type of rule-based classifiers generalising decision tree ensembles. Although their work shares similarities with ours in terms of research goals, we note several important differences. First, our data-independent stability analysis allows one to identify a subset of the feature space where the classifier is stable rather than verifying stability over the entire feature space; in other words, we can effectively identify sub-spaces where stability is globally guaranteed. This is more useful in practice because stability over the entire feature space is often too strong, essentially requiring that the set of non-robust features is unused for classification. Indeed, contrary to the security notion proposed in this work, i.e., the resilience, their global robustness properties are entirely abstract from the data distribution, which is a sound yet overly conservative choice in the machine learning setting. This claim is confirmed by the experimental evaluation provided by Chen *et al.* in [49], which shows that stability cannot be verified for any model (standard or robust) besides those deliberately trained by the authors to enforce that property. Rather, we are able to use our resilience notion to perform practically useful security evaluations of existing ML models while still overcoming the limitations of robustness confirmed by our experiments.

Furthermore, Leino *et al.* in [100] introduced globally robust neural networks. They proposed a technique to train neural networks with a special output $\perp$, designed to signal predictions performed on a subset of the feature space that is too close to the decision boundary and, hence, potentially subject to evasion attacks. Their notion of global robustness requires that any two "close" inputs must lead to the same output unless $\perp$ is returned for at least one of the two inputs. Our data-independent stability analysis essentially captures the same notion because its output could also be used to return $\perp$ on all the instances that do not fall on a stable subset of the feature space. However, the global robustness defined by Leino *et al.* in [100] cannot be used to reason about the security of traditional classifiers, which do not use the $\perp$ label. Finally, their technical treatment is quite different from ours since they focus on neural networks while we care more about tree-based classifiers in adversarial scenarios.

The security certification of decision trees and decision tree ensembles has received an increasing amount of attention from the research community during the last few years. The first seminal work on the topic is due to Kantchelian *et al.* in [92]. They showed that computing minimal adversarial perturbations for tree ensembles is NP-complete in general and proposed a mixed-integer linear programming technique for the task. This motivated additional work in the area by Chen *et al.* in [46]. They investigated restricted fragments of the problem which are tractable in polynomial time and proposed an approximated, yet sound, approach to verify robustness against L_∞-norm attackers. In later work, Ranzato and Zanella in [145] proposed the use of abstract interpretation to mitigate the complexity of robustness verification by means of a sound over-approximation of the ensemble predictions, again assuming L_∞-norm attackers. Calzavara *et al.* in [33] showed that abstract interpretation could also be used to verify the robustness of decision trees against an expressive threat model, where the attacker is encoded as an arbitrary imperative program. All of these approaches only prove robustness and cannot be directly used to prove resilience without a data-independent stability analysis, like the one proposed in the present paper.

A different line of work which is more directly comparable to ours is related to the VoTE checker by Törnblom and Nadjm-Tehrani in [164]. Given a tree ensemble, VoTE computes the set of all the equivalence classes induced by the ensemble over the feature space. Once the equivalence classes have been computed, VoTE uses a property checker module to verify different properties on them. The idea of computing equivalence classes from the tree ensemble yields a data-independent analysis approach; however, there are important differences with respect to our work. First, their analysis is not adversary-aware, and the security implications of data independence are not explored by the authors since they just verify traditional robustness properties. We rather clarify the practical relevance of data

independence by introducing a new formal security notion called resilience, and we design experiments to show its empirical value on real datasets. Moreover, computing all the equivalence classes of an ensemble is infeasible in general due to their combinatorial explosion, as also observed by the authors of VoTE. In a follow-up work, the same authors proposed an abstraction-refinement approach to mitigate this complexity problem [163]. However, contrary to our analysis, their extension is not proved sound, which is an important requirement for the analysis of classifiers deployed in adversarial settings.

10.2 Threat and Adversary's Model

In this study, we consider a threat model for a machine learning system designed for binary classification, making it susceptible to evasion attacks. An attacker attempts to mislead an already trained binary classifier h by perturbing a legitimate instance $\boldsymbol{x}$ to generate an evasion instance $\boldsymbol{z}$. We assume each feature in the features space $\mathcal{F}$ denoted as $f \in \{1, \ldots, d\}$ is associated with a *cost* $c_f \in \mathbb{N}$, that the attacker has to pay in order to add to f a perturbation $\delta \in \mathbb{R}$ arbitrarily drawn from an interval I_f^{atk}. The interval I_f^{atk} can be different for each feature f and takes the form of $[\delta_l, \delta_r]$, where δ_l and δ_r are the minimal and the maximal perturbation values, respectively. We encode robust features that cannot be manipulated by setting $[0, 0]$ as their perturbation interval.

We characterise the attacker's model A_b by using the previous definition of the threat model and constraining the attacker with a *budget* b, which determines the maximum aggregate cost the attacker can spend to manipulate features. The budget b characterises the attacker's power. For the sake of clarity, we will refer to the attacker's model as A when b is known from the context. Given the attacker's model A_b, we formalise the adversarial perturbation set $A_b(\boldsymbol{x})$ as follows:

Definition 10 (Adversarial Perturbation Set)**.** Given an instance $\boldsymbol{x} \in \mathcal{X}$, we define the adversarial perturbation set $A_b(\boldsymbol{x})$, as the set of the adversarial instances $\boldsymbol{z}$ for which there exists a set of attacked features $B \subseteq \mathcal{F}$ such that:

1. For all $f \in B$, we have $z^{(f)} = x^{(f)} + \delta$ for some $\delta \in I_f^{atk}$.
2. For all $f \notin B$, we have $z^{(f)} = x^{(f)}$.
3. We have $\sum_{f \in B} c_f \leq b$.

This threat model is inspired by traditional distance-based attackers from the adversarial ML literature and it is expressive enough to model a wide range of attacks, including those based on the L_0-norm and the L_∞-norm which are traditionally used in prior work [34, 46]. In particular, note that:

- L_0-norm attackers can be modelled by assuming that each feature can be perturbed in the interval $[-\infty, +\infty]$ by paying cost 1 and by setting the attacker's budget to the maximum L_0-distance assumed for the attack.

- L_∞-norm attackers can be modelled by assuming that each feature can be perturbed in the interval $[-\delta, +\delta]$ by paying cost 1, where δ is the maximum L_∞-distance assumed for the attack, and by setting the attacker's budget to the total number of features.

By using this threat model, we are able to design a relatively general analysis technique that readily applies to at least these two popular classes of attackers from the literature. These attackers are popular because they are simple mathematical models of two representative classes of threats: strong corruptions of a small number of features (L_0-norm) and weak corruptions of all the features (L_∞-norm).

10.3 Contribution 1: The Resilience Metric

Before delving into one of the primary contributions of this work, we introduce the fundamental technical elements necessary for a comprehensive understanding of this study. Here, our attention is centred on conventional binary decision trees, which represent the most prevalent and widely used variant of such models.

Given that we extensively discussed the internal structure of decision trees in Section 2.1.2 and have also addressed it in the previous Chapter 9, we only repeat a few essential details. A decision tree t can be inductively defined as a leaf $\lambda(y)$ or an internal node $\sigma(f, v, t_l, t_r)$, where f identifies a feature, v is the node threshold, and t_l and t_r are the left and right decision trees, respectively. At test time, the instance $\boldsymbol{x}$ traverses the tree t until it reaches a leaf $\lambda(y)$, which returns the prediction, denoted by $t(\boldsymbol{x})$. Specifically, for each traversed tree node $\sigma(f, v, t_l, t_r)$, $\boldsymbol{x}$ falls into the left sub-tree t_l if $x^{(f)} \leq v$, and into the right sub-tree t_r otherwise.

In this work, ensemble decision tree in tree-based ensembles (i.e., forests) $F = \{t_1, \ldots, t_n\}$ where the ensemble prediction $F(\boldsymbol{x})$ is computed by performing majority voting on the individually tree-predicted classes.

10.3.1 Robustness vs. Resilience

We now discuss important shortcomings in the traditional robustness measure, and we propose a generalisation of robustness, called *resilience*, which is designed to mitigate those. We then explain how resilience can be verified in practice, and we further elaborate on its design by discussing its connections with a recent definition of global robustness. [100].

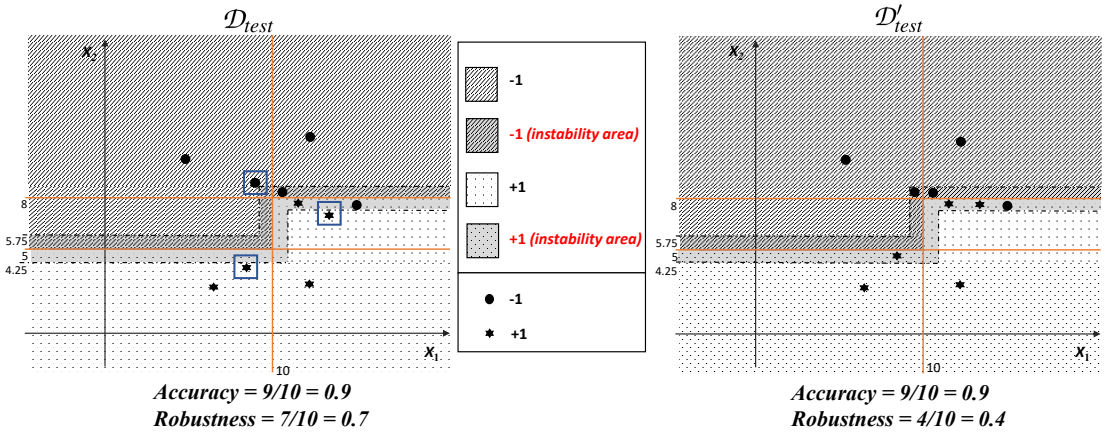


Figure 10.1: Robustness is not robust: slightly different test sets lead to very different values of robustness

A key problem of robustness is its strong *data-dependence*, i.e., robustness is quantified on a specific test set $\mathcal{D}_{test}$. Hence, it is possible that even tiny differences between two test sets $\mathcal{D}_{test}$ and $\mathcal{D}'_{test}$ might lead to quite different values of robustness. This might give a false sense of security. For example, a classifier having robustness 0.7 on $\mathcal{D}_{test}$ might only have robustness 0.4 on $\mathcal{D}'_{test}$, although both $\mathcal{D}_{test}$ and $\mathcal{D}'_{test}$ are representative of the same data distribution of the test instances. We show this problem of robustness in Figure 10.1, where we compute the accuracy and robustness of both $\mathcal{D}_{test}$ and $\mathcal{D}'_{test}$ with the same tree classifier. We assume here a two-dimensional feature space and an L_1-norm attacker such that $A(\boldsymbol{x}) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathcal{X} \wedge \|\boldsymbol{z} - \boldsymbol{x}\|_1 \leq 0.75\}$, leading to the highlighted *instability areas* around the decision boundaries of the tree. The figure shows that the same classifier provides very different robustness measures on the two close test sets $\mathcal{D}_{test}$ and $\mathcal{D}'_{test}$. The reason behind this drop in robustness lies in more instances of $\mathcal{D}'_{test}$ falling in the instability area. Hence, the adoption of $\mathcal{D}_{test}$ over $\mathcal{D}'_{test}$ for robustness computation might give a false sense of security.

To mitigate this problem of robustness, we propose *resilience*, a new security measure that explicitly assumes that test instances are sampled from a given data distribution; hence, each instance $\boldsymbol{x}$ is just a possible sample drawn from a set of neighbours $N(\boldsymbol{x})$. For example, $N(\boldsymbol{x})$ may contain all the instances that are within a maximum L_∞-distance from $\boldsymbol{x}$, as we assume in our experiments.

In Figure 10.1, we define $N(\boldsymbol{x}) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathcal{X} \ \wedge \ \|\boldsymbol{x} - \boldsymbol{z}\|_\infty \leq 0.50\}$ and the neighborhood of $\boldsymbol{x}$ is thus graphically represented by a small box around $\boldsymbol{x}$. Indeed, the figure shows these boxes only for the instances of $\mathcal{D}_{test}$ whose neighbourhood $N(\boldsymbol{x})$ intersects the instability areas of the tree and, therefore, other instances in their neighbourhood might suffer from evasion attacks.

Resilience avoids the shortcomings of robustness illustrated in Figure 10.1, as it generalises the idea of robustness to all the test sets which could have been sampled within neighbourhoods of the original test set. In other words, resilience requires the classifier to be robust on test instances while remaining stable in their neighbourhoods, for which the correct class labels are unknown.

Formally, given the attacker A, an instance $\boldsymbol{x} \in \mathcal{X}$, the classifier h, and let $N(\boldsymbol{x})$ be the neighbourhood set of the instance $\boldsymbol{x}$, the classifier h is resilient on $\boldsymbol{x}$ with respect to A if the following definition holds:

Definition 11 (Resilience)**.** The classifier h is *resilient* on the instance $\boldsymbol{x}$ if and only if h is robust on $\boldsymbol{x}$ and h is stable on all the instances $\boldsymbol{x}' \in N(\boldsymbol{x})$.

Back to our example, the resilience of both the test sets of Figure 10.1 turns out to be 0.4, i.e., the measured robustness of $\mathcal{D}'_{test}$, the unlucky test set shown in the right part of the figure.

Note that resilience generalises beyond the test set by means of the $N(\boldsymbol{x})$ component, which extends the stability guarantees of robustness to an uncountable set of neighbours not included in the test set. Still, similarly to robustness, we can quantify resilience on a test set of correctly labelled instances like in traditional ML pipelines by defining a resilience measure $\mathcal{R}(A, N)$ as follows:

$$\mathcal{R}(A, N) = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{test}} \mathbb{1}[\forall \boldsymbol{x}' \in N(\boldsymbol{x}), \forall \boldsymbol{z} \in A(\boldsymbol{x}') \mid h(\boldsymbol{z}) = y]}{|\mathcal{D}_{test}|} \tag{10.1}$$

Observe that resilience provides a lower bound to robustness as claimed, i.e., $\mathcal{R}(A, N) \leq \mathrm{R}(A, N)$ for every classifier h and test set $\mathcal{D}_{test}$.

Finally, we acknowledge that resilience is still a data-dependent property. However, the dependence from the test set is much weaker for resilience than for robustness, thanks to the introduction of the $N(\boldsymbol{x})$ component. Indeed, data dependence is not necessarily a problem per se because a high-quality test set is required to empirically assess the performance of ML models anyway; however, excessive dependence on a given test set might provide a false sense of security against evasion attacks.

10.3.2 Resilience Verification

Traditional robustness verification approaches cannot be readily applied to prove resilience. In particular, note that robustness verification takes as input an instance $\boldsymbol{x}$ and attempts to assess its stability, while resilience verification requires proving stability for an uncountable set of instances $N(\boldsymbol{x})$.

Nevertheless, it is possible to estimate resilience by combining robustness verification with a *data-independent* stability analysis. In particular, assume one has

a technique to identify the set $X_S = \{\boldsymbol{x} \in \mathcal{X} \mid h \text{ is stable on } \boldsymbol{x}\}$, then h is resilient on $\boldsymbol{x}$ if and only if h is robust on $\boldsymbol{x}$ and $N(\boldsymbol{x}) \subseteq X_S$. Note that X_S in Figure 10.1 corresponds to the whole area except the instability ones. Hence, resilience verification reduces to robustness verification with the additional condition $N(\boldsymbol{x}) \subseteq X_S$, i.e., we also have to check that $N(\boldsymbol{x})$ does not intersect the instability area. This allows one to take advantage of existing robustness verifiers also to quantify resilience, provided that the stable part of the feature space X_S has been computed. Note that computing X_S is not trivial, in particular for tree ensembles, because a given instance traverses the ensemble reaching a set of leaves, and any of such reachable set of leaves corresponds to a sub-space which should be evaluated for inclusion in X_S. Clearly, the number of these sub-spaces grows exponentially with the number of trees in the ensemble. In order to compute X_S, and therefore compute resilience, we resort to an approximated approach: computing a tractable under-approximation of X_S for decision trees and decision tree ensembles is a key contribution of this work.

In particular, the data-independent stability analysis introduced in Section 10.4 allows one to compute a set $X'_S \subseteq X_S$, i.e., a sound under-approximation of the portion of the feature space where a tree-based classifier is stable. The proposed computation of X'_S depends on the classifier alone and not on the test data at hand. In fact, we can exploit the generality of X'_S to efficiently compute lower bounds for the robustness and resilience measures as follows.

Given that if $\boldsymbol{x} \in X'_S$ and $h(\boldsymbol{x}) = y$, then h is robust on $\boldsymbol{x}$, we can define a lower bound $\hat{\mathrm{R}}$ on robustness as:

$$\hat{\mathrm{R}}(A, X'_S) = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{test}} \mathbb{1}[\boldsymbol{x} \in X'_S \wedge h(\boldsymbol{x}) = y]}{|\mathcal{D}_{test}|} \tag{10.2}$$

Similarly, given that if h is robust on $\boldsymbol{x}$ and $N(\boldsymbol{x}) \subseteq X'_S$, then h is also resilient on $\boldsymbol{x}$, we can introduce a lower bound $\hat{\mathcal{R}}$ on resilience as:

$$\hat{\mathcal{R}}(A, X'_S) = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{test}} \mathbb{1}[\forall \boldsymbol{z} \in A(\boldsymbol{x}) \mid h(\boldsymbol{z}) = y \wedge N(\boldsymbol{x}) \subseteq X'_S]}{|\mathcal{D}_{test}|} \tag{10.3}$$

Note that the pre-computation of X'_S makes robustness verification straightforward for all the instances falling in X'_S and that the computation of the resilience estimate $\hat{\mathcal{R}}(A, X'_S)$ can potentially exploit any robustness verification method, which allows one to leverage existing work in the area. Moreover, we may use the accuracy of $\hat{\mathrm{R}}(A, X'_S)$ with respect to the true robustness $\mathrm{R}(A)$ (established using an existing robustness verifier) as a proxy for the accuracy of X'_S, which allows us to assess the quality of the under-approximation $\hat{\mathcal{R}}(A, X'_S)$ and advocate its adoption to mitigate the false sense of security provided by $\mathrm{R}(A)$.

Note that, for the sake of simplicity, hereinafter, we refer to $\mathrm{R}(A)$, $\hat{\mathrm{R}}(A, X'_S)$, $\mathcal{R}(A, N)$, and $\hat{\mathcal{R}}(A, X'_S)$ as R, $\hat{\mathrm{R}}$, $\mathcal{R}$, and $\hat{\mathcal{R}}$, respectively, when the attacker A, the neighbour set N, and set X'_S can be easily derived by the context.

10.3.3 Resilience vs. Global Robustness

Recent work proposed a technique to train *globally robust* neural networks, which provide robustness guarantees for all the possible inputs rather than just for the inputs in the test set [100]. The idea of globally robust neural networks, generalisable to arbitrary classifiers, is that the set of labels $\mathcal{Y}$ is extended with a special class $\perp$, used to denote that no reliable prediction is possible because the instance is too close to the decision boundary of the classifier and thus potentially susceptible to evasion attacks. Global robustness requires that any two instances that are close enough to each other are either assigned the same prediction or at least one of them is flagged as $\perp$. This property is intuitive and desirable; however, contrary to resilience, it cannot be used to verify the security of existing classifiers that have not been trained to return the $\perp$ label.

Rather, note that the proposed approach to resilience verification based on a data-independent stability analysis can be readily applied to transform any classifier into a globally robust classifier. In particular, given any classifier h and a subset of the feature space $X'_S \subseteq \mathcal{X}$ where h is stable, one can define a globally robust classifier h' as follows:

$$h'(\boldsymbol{x}) = \begin{cases} h(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in X'_S \\ \perp & \text{otherwise} \end{cases}$$

Observe that the previously defined robustness estimate $\hat{\mathrm{R}}$ provides a (local) robustness measure of the globally robust classifier h' obtained through the previous construction.

10.4 Contribution 2: Data-independent Stability Analysis

In this section, we present the second main contribution of this work: a data-independent stability analysis for decision trees and decision tree ensembles to compute the two measures $\hat{\mathrm{R}}$ (robustness lower bound) and $\hat{\mathcal{R}}$ (resilience lower bound) defined in Section 10.3.2, thus providing conservative (i.e., lower bound) estimates of robustness and (most importantly) resilience. The analysis is proved sound. We show that the portions of the feature space marked as stable by the

analysis may only contain instances where the classifier is indeed stable. Proofs are given in Section 10.7.2.

10.4.1 Preliminaries

Our analysis leverages *intervals* of real numbers. Given $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ with $a \leq b$, we use standard notation for intervals, using parentheses to represent open bounds and brackets to represent closed bounds. This leads to four possible types of intervals: (a, b), $[a, b)$, $(a, b]$, and $[a, b]$. We use I, J to range over intervals. A *hyper-rectangle* $H \subseteq \mathcal{X}$ is represented as a vector of intervals $\langle I_1, \ldots, I_d \rangle$ over $\mathbb{R}$.

Given a decision tree t, we define its possible predictions over the hyper-rectangle H as $t(H) = \{y \in \mathcal{Y} \mid \exists \boldsymbol{x} \in H : t(\boldsymbol{x}) = y\}$. Note that computing $t(H)$ is straightforward by means of a recursive tree traversal. Specifically, if $t = \lambda(y)$, then $t(H) = \{y\}$. If instead $t = \sigma(f, v, t_l, t_r)$, we define $t(H)$ with $H = \langle I_1, \ldots, I_d \rangle$ as follows:

$$t(H) = \begin{cases} t_l(H) & \text{if } I_f \cap (v, +\infty) = \emptyset \\ t_r(H) & \text{if } I_f \cap (-\infty, v] = \emptyset \\ t_l(H) \cup t_r(H) & \text{otherwise} \end{cases}$$

Finally, we extend predictions over hyper-rectangles from trees to tree ensembles, noted $F(H)$. The actual definition of $F(H)$ depends on the approach used by F to aggregate the individual tree predictions to produce the ensemble prediction. For example, in the case of majority voting, we may let:

$$F(H) = \begin{cases} \{y\} & \text{if } |\{t_i \in F \mid t_i(H) = \{y\}\}| > |F|/2 \\ \mathcal{Y} & \text{otherwise} \end{cases}$$

For soundness, we require $\{y \in \mathcal{Y} \mid \exists \boldsymbol{x} \in H : F(\boldsymbol{x}) = y\} \subseteq F(H)$, i.e., the set of the predictions $F(H)$ includes all the predictions that may be assigned to an instance in H. Proving that this requirement holds for the definition above is straightforward.

In the following, given two intervals, we define their sum $I + J$ as the interval whose lower bound is the sum of the lower bounds and whose upper bound is the sum of the upper bounds; we require the bounds to be closed if and only if both the added bounds are closed. For example, $[1, 3] + (4, 6] = (5, 9]$. Moreover, given two hyper-rectangles, we define their sum $H + H'$ as the pointwise sum of their components (intervals).

10.4.2 Decision Tree Stability Analysis

Our analysis operates by annotating each node of a decision tree with a set of *symbolic attacks*, which represent a set of instances along with their *relevant* adversarial

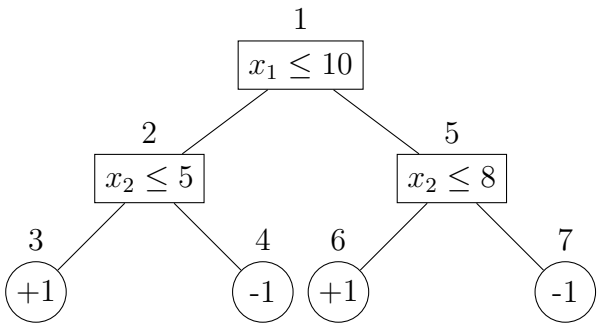


Figure 10.2: An example of the internal structure of a decision tree.

perturbations. As for the work in Chapter 9, most adversarial perturbations are not relevant for the stability analysis of decision trees because such classifiers operate by means of thresholds; hence, only attacks that allow for traversing some threshold might lead to instability [35]. Formally, a symbolic attack s has the shape $\langle I_1^{pre}, \ldots, I_d^{pre} \rangle \rhd \langle I_1^{post}, \ldots, I_d^{post} \rangle_k$, where each I_i^{pre}, I_j^{post} is an interval on $\mathbb{R}$ and $k \in \mathbb{N}$. Intuitively, s identifies the set of instances located within the hyper-rectangle $\langle I_1^{pre}, \ldots, I_d^{pre} \rangle$, called the *pre-image* of the symbolic attack, along with their adversarial perturbations located within the hyper-rectangle $\langle I_1^{post}, \ldots, I_d^{post} \rangle$, called the *post-image* of the symbolic attack; the *cost* of such adversarial perturbations is bounded above by k. We make use of symbolic attacks to identify which nodes of the decision tree can be traversed by a set of instances as the result of adversarial perturbations against them; we require that when $k = 0$, the pre-image and the post-image coincide, i.e., the symbolic attack captures the case where no adversarial perturbation has taken place. We write $s.pre$, $s.post$, and $s.cost$ to project the pre-image, the post-image, and the cost of s, respectively.

Before presenting the formal details, we present the analysis at work on the decision tree of Figure 10.2 built on a feature space with two features. We assume an attacker who can manipulate at most one feature by adding a perturbation $\delta \in [-1, 1]$. Formally, this is represented by having $I_1^{atk} = I_2^{atk} = [-1, 1]$, $c_1 = c_2 = 1$ and $b = 1$. The analysis result for node annotations is presented in Table 10.1.

The observations we can draw from the results of this analysis are the following: Node 1 is the root of the tree; hence, the analysis cannot conclude anything about instances traversing the node, i.e., the symbolic attack in the node annotation models that all instances in the feature space traverse the root, no matter what the attacker does. Node 2, instead, is more interesting because the analysis captures two cases via two symbolic attacks: an instance $\boldsymbol{x}$ might traverse the node either because $x_1 \leq 10$ and the attacker does nothing, or because $x_1 \in (10, 11]$ is adversarially manipulated into the interval $(9, 10]$. Note that, in the second case, the symbolic attack is assigned a cost of 1, which allows us to track that no further

Table 10.1: Tree annotation results for the decision tree in Figure 10.2

Node	Symbolic Attacks
1	$\langle(-\infty,+\infty),(-\infty,+\infty)\rangle \rhd \langle(-\infty,+\infty),(-\infty,+\infty)\rangle_0$
2	$\langle(-\infty,10],(-\infty,+\infty)\rangle \rhd \langle(-\infty,10],(-\infty,+\infty)\rangle_0$ $\langle(10,11],(-\infty,+\infty)\rangle \rhd \langle(9,10],(-\infty,+\infty)\rangle_1$
3	$\langle(-\infty,10],(-\infty,5]\rangle \rhd \langle(-\infty,10],(-\infty,5]\rangle_0$ $\langle(-\infty,10],(5,6]\rangle \rhd \langle(-\infty,10],(4,5]\rangle_1$ $\langle(10,11],(-\infty,5]\rangle \rhd \langle(9,10],(-\infty,5]\rangle_1$
4	$\langle(-\infty,10],(5,+\infty)\rangle \rhd \langle(-\infty,10],(5,+\infty)\rangle_0$ $\langle(-\infty,10],(4,5]\rangle \rhd \langle(-\infty,10],(5,6]\rangle_1$ $\langle(10,11],(5,+\infty)\rangle \rhd \langle(9,10],(5,+\infty)\rangle_1$
5	$\langle(10,+\infty),(-\infty,+\infty)\rangle \rhd \langle(10,+\infty),(-\infty,+\infty)\rangle_0$ $\langle(9,10],(-\infty,+\infty)\rangle \rhd \langle(10,11],(-\infty,+\infty)\rangle_1$
6	$\langle(10,+\infty),(-\infty,8]\rangle \rhd \langle(10,+\infty),(-\infty,8]\rangle_0$ $\langle(10,+\infty),(8,9]\rangle \rhd \langle(10,+\infty),(7,8]\rangle_1$ $\langle(9,10],(-\infty,8]\rangle \rhd \langle(10,11],(-\infty,8]\rangle_1$
7	$\langle(10,+\infty),(8,+\infty)\rangle \rhd \langle(10,+\infty),(8,+\infty)\rangle_0$ $\langle(10,+\infty),(7,8]\rangle \rhd \langle(10,+\infty),(8,9]\rangle_1$ $\langle(9,10],(8,+\infty)\rangle \rhd \langle(10,11],(8,+\infty)\rangle_1$

perturbations are possible because the attacker's budget has run out. Even more interesting is the case of node 3, where we have three possibilities. In particular, an instance $\boldsymbol{x}$ might traverse the node in the following cases: *i)* $x_1 \leq 10$ and $x_2 \leq 5$, hence no adversarial perturbation is needed, *ii)* $x_1 \leq 10$ and $x_2 \in (5,6]$ is manipulated into $(4,5]$, or *iii)* $x_1 \in (10,11]$ is manipulated into $(9,10]$ and $x_2 \leq 5$. We do not have a case where both features are manipulated because this would exceed the attacker's budget. A similar reasoning applies to the other nodes in the tree. Once the tree has been annotated, it is possible to check stability by inspecting the annotations in its leaves: we discuss this aspect of the analysis later in the section.

Algorithm 6 describes the annotation function for decision trees. We assume each node of the tree is enriched with an attribute *sym*, used to store a set of symbolic attacks. The call ANNOTATE(t, S) annotates the root of t with the set of symbolic attacks S passed as a parameter (line 5), then uses S and the threshold information in the root to generate the annotations for the roots of the left and right sub-trees (lines 9-11); finally, the process goes down recursively (lines 12-

13). When the annotation function is initially invoked on the root of the decision tree to analyse, we set $S = \{\langle(-\infty, +\infty)^d\rangle \rhd \langle(-\infty, +\infty)^d\rangle_0\}$ as explained in the example.

Algorithm 6 Decision tree annotation

```
1: function ANNOTATE(t, S)
2:     Input
3:         t : tree
4:         S : symbolic attack set generated by the parent node

5:     t.sym ← S
6:     if t = σ(f, v, t_l, t_r) then
7:         S_l ← ∅
8:         S_r ← ∅
9:         for s ∈ S do
10:            S_l ← S_l ∪ REFINELEFT(s, f, v)
11:            S_r ← S_r ∪ REFINERIGHT(s, f, v)
12:        ANNOTATE(t_l, S_l)
13:        ANNOTATE(t_r, S_r)
```

The key part of the node annotation logic is implemented by the auxiliary functions RefineLeft and RefineRight, defined in Algorithm 7 and Algorithm 8, respectively. Given a symbolic attack s, a feature f and the associated threshold v from an internal node of the decision tree, the call RefineLeft(s, f, v) uses s to generate a new set of symbolic attacks S for the root of the left sub-tree (initially empty). Lines 13-19 account for the case where some instances in the post-image of s already fall in the left sub-tree, i.e., the attacker does not need to spend budget to manipulate the feature f so as to push some instances in the pre-image into the left sub-tree. In this case, S is extended with a refined variant of s, where we track that the feature f must belong to the interval $(-\infty, v]$ for the instances in the post-image (line 14). Also the pre-image of s is refined in the left sub-tree: if f was not attacked, we know that the feature f must belong to the interval $(-\infty, v]$ for the instances in the pre-image as well (lines 15-16); otherwise, we still know that the attack could not push instances beyond the maximum negative perturbation $\delta_l < 0$, hence the feature f must belong to the interval $(-\infty, v - \min(0, \delta_l)]$ for the instances in the pre-image (lines 17-18). Lines 20-23, instead, cover the case where some instances in the pre-image are close enough to the threshold v to be pushed into the left sub-tree as the result of adversarial perturbations. In this case, provided that the attacker still has enough budget to spend, S is extended with a refined variant of s, where we update both the post-image and the pre-image

Algorithm 7 Refinement for the left sub-trees

1: **function** RefineLeft(s, f, v)
2: **Input**
3: s : symbolic attack
4: f : test node's feature
5: v : test node's threshold
6: **Output**
7: S : set symbolic attacks create by the split (v, f) of s
8: $S \leftarrow \emptyset$
9: $\langle I_1^{pre}, \ldots, I_d^{pre} \rangle \leftarrow s.pre$
10: $\langle I_1^{post}, \ldots, I_d^{post} \rangle \leftarrow s.post$
11: $k \leftarrow s.cost$
12: $[\delta_l, \delta_r] \leftarrow I_f^{atk}$
13: **if** $I_f^{post} \cap (-\infty, v] \neq \emptyset$ **then**
14: $J_f^{post} \leftarrow I_f^{post} \cap (-\infty, v]$
15: **if** $I_f^{pre} = I_f^{post}$ **then**
16: $J_f^{pre} \leftarrow I_f^{pre} \cap (-\infty, v]$
17: **else**
18: $J_f^{pre} \leftarrow I_f^{pre} \cap (-\infty, v - \min(0, \delta_l)]$
19: $S \leftarrow S \cup \{\langle I_1^{pre}, \ldots, I_{f-1}^{pre}, J_f^{pre}, I_{f+1}^{pre}, \ldots I_d^{pre} \rangle \triangleright \langle I_1^{post}, \ldots, I_{f-1}^{post}, J_f^{post}, I_{f+1}^{post}, \ldots, I_d^{post} \rangle_k\}$
20: **if** $I_f^{pre} = I_f^{post} \wedge \delta_l < 0 \wedge I_f^{pre} \cap (v, v - \delta_l] \neq \emptyset \wedge k + c_f \leq b$ **then**
21: $J_f^{post} \leftarrow I_f^{post} \cap (v + \delta_l, v]$
22: $J_f^{pre} \leftarrow I_f^{pre} \cap (v, v - \delta_l]$
23: $S \leftarrow S \cup \{\langle I_1^{pre}, \ldots, I_{f-1}^{pre}, J_f^{pre}, I_{f+1}^{pre}, \ldots I_d^{pre} \rangle \triangleright \langle I_1^{post}, \ldots, I_{f-1}^{post}, J_f^{post}, I_{f+1}^{post}, \ldots, I_d^{post} \rangle_{k+c_f}\}$
24: **return** S

to reflect their proximity to the threshold v. More precisely, given the maximum negative perturbation $\delta_l < 0$, we track that the feature f must belong to the interval $(v + \delta_l, v]$ for the instances in the post-image and to the interval $(v, v - \delta_l]$ for the instances in the pre-image, otherwise crossing the threshold would not be possible. The RefineRight function performs analogous reasoning for the right sub-tree; hence, we omit a detailed explanation.

The stability analysis for decision trees is finally shown in Algorithm 9. The call Analyze(t) leverages the results of the tree annotation function to return a set of symbolic attacks U, identifying the portions of the feature space where the

Algorithm 8 Refinement for the right sub-trees

1: **function** RefineRight(s, f, v)
2: **Input**
3: s : symbolic attack
4: f : test node's feature
5: v : test node's threshold
6: **Output**
7: S : set symbolic attacks create by the split (v, f) of s
8: $S \leftarrow \emptyset$
9: $\langle I_1^{pre}, \ldots, I_d^{pre} \rangle \leftarrow s.pre$
10: $\langle I_1^{post}, \ldots, I_d^{post} \rangle \leftarrow s.post$
11: $k \leftarrow s.cost$
12: $[\delta_l, \delta_r]\rangle \leftarrow I_f^{atk}$
13: **if** $I_f^{post} \cap (v, +\infty) \neq \emptyset$ **then**
14: $J_f^{post} \leftarrow I_f^{post} \cap (v, +\infty)$
15: **if** $I_f^{pre} = I_f^{post}$ **then**
16: $J_f^{pre} \leftarrow I_f^{pre} \cap (v, +\infty)$
17: **else**
18: $J_f^{pre} \leftarrow I_f^{pre} \cap (v - \max(0, \delta_r), +\infty)$
19: $S \leftarrow S \cup \{\langle I_1^{pre}, \ldots, I_{f-1}^{pre}, J_f^{pre}, I_{f+1}^{pre}, \ldots I_d^{pre} \rangle \triangleright \langle I_1^{post}, \ldots, I_{f-1}^{post}, J_f^{post}, I_{f+1}^{post}, \ldots, I_d^{post} \rangle_k\}$
20: **if** $I_f^{pre} = I_f^{post} \wedge \delta_r > 0 \wedge I_f^{pre} \cap (v - \delta_r, v] \neq \emptyset \wedge k + c_f \leq b$ **then**
21: $J_f^{post} \leftarrow I_f^{post} \cap (v, v + \delta_r]$
22: $J_f^{pre} \leftarrow I_f^{pre} \cap (v - \delta_r, v]$
23: $S \leftarrow S \cup \{\langle I_1^{pre}, \ldots, I_{f-1}^{pre}, J_f^{pre}, I_{f+1}^{pre}, \ldots I_d^{pre} \rangle \triangleright \langle I_1^{post}, \ldots, I_{f-1}^{post}, J_f^{post}, I_{f+1}^{post}, \ldots, I_d^{post} \rangle_{k+c_f}\}$
24: **return** S

decision tree t may be unstable. The function operates by looking for two leaves with different class predictions such that: *i)* the first leaf contains a symbolic attack s of cost 0, i.e., no adversarial perturbation was performed on the pre-image of s, *ii)* the second leaf contains a symbolic attack s' of cost greater than 0, i.e., the attacker manipulated the pre-image of s', and *iii)* the pre-images of the two symbolic attacks s, s' partially overlap, i.e., there exist some instances which might fall into a leaf with a different class than the original prediction due to adversarial perturbations (lines 8-13). In this case, the intersection of the pre-images identifies a portion of the feature space where the tree may be unstable.

Algorithm 9 Stability analysis for decision trees

1: **function** ANALYZE(t)
2: **Input**
3: t : tree
4: **Output**
5: U : set of symbolic attacks where t may be unstable
6: $t \leftarrow \text{ANNOTATE}(t, \{\langle(-\infty, +\infty)^d\rangle \rhd \langle(-\infty, +\infty)^d\rangle_0\})$
7: $U \leftarrow \emptyset$
8: **for** $\lambda(y) \in t$ **do**
9: **for** $s \in \{\hat{s} \in \lambda(y).sym \mid \hat{s}.cost = 0\}$ **do**
10: **for** $\lambda'(y') \in t$ **do**
11: **if** $y' \neq y$ **then**
12: **for** $s' \in \{\hat{s} \in \lambda'(y').sym \mid \hat{s}.cost > 0\}$ **do**
13: **if** $s.pre \cap s'.pre \neq \emptyset$ **then**
14: $s''.pre \leftarrow s.pre \cap s'.pre$
15: $s''.post \leftarrow s'.post \cap (s''.pre + \langle I_1^{atk}, \ldots, I_d^{atk}\rangle)$
16: $s''.cost \leftarrow s'.cost$
17: $U \leftarrow U \cup \{s''\}$
18: **return** U

The post-image of s' is refined to capture that the adversarial perturbations cannot push instances beyond the maximum allowed perturbation of the intersection of the two pre-images (lines 14-17). This is a conservative approximation, which accounts for all the possible adversarial perturbations. To exemplify the output of the stability analysis, consider the node annotations in Table 10.1. The stability analysis returns the symbolic attacks reported in Table 10.2.

The pre-images of these symbolic attacks identify the portions of the feature space where the decision tree is unstable. It is interesting to observe that leaves 4 and 6 contribute two portions of the feature space where the tree may be unstable, while leaves 3 and 7 only contribute one. The reason for this is that leaves 4 and 6 partially overlap on the values allowed for the second feature, i.e., the interval $(5, 8]$. This means that it is possible to jump from leaf 4 to leaf 6 (and *vice versa*) as the result of an attack targeting just the first feature, provided that the second feature falls into $(5, 8]$. Conversely, leaves 3 and 7 have no overlap on any of the two features; hence, an attack which manipulates just one feature cannot induce a jump between these two leaves. As a final comment, notice that the post-images are not needed at this stage of the analysis: we just collect them because they support the analysis of tree ensembles in the next section.

Table 10.2: Tree stability analysis results for the decision tree in Figure 10.2 with respect to the symbolic attacks in Table 10.1

Leaf node	Symbolic Attacks
3	$\langle(-\infty, 10], (4, 5]\rangle \rhd \langle(-\infty, 10], (5, 6]\rangle_1$
4	$\langle(-\infty, 10], (5, 6]\rangle \rhd \langle(-\infty, 10], (4, 5]\rangle_1$ $\langle(9, 10], (5, 8]\rangle \rhd \langle(10, 11], (4, 8]\rangle_1$
6	$\langle(10, +\infty), (7, 8]\rangle \rhd \langle(10, +\infty), (8, 9]\rangle_1$ $\langle(10, 11], (5, 8]\rangle \rhd \langle(9, 10], (5, 9]\rangle_1$
7	$\langle(10, +\infty), (8, 9]\rangle \rhd \langle(10, +\infty), (7, 8]\rangle_1$

The soundness theorem for our analysis is given below. The theorem states that all the instances $\boldsymbol{x}$ where t is unstable must fall in the pre-image of a symbolic attack contained in the set U returned by the call $\textsc{Analyze}(t)$. In other words, the union of the pre-images of the symbolic attacks in U can only over-approximate the portion of the feature space where t is unstable, i.e., t must be stable on all the instances located outside such area. This allows us to compute the set X'_S where t is stable by subtracting the union of the pre-images of U from the full feature space $\mathcal{X}$.

Theorem 1 (Soundness of Tree Analysis)**.** The call $\textsc{Analyze}(t)$ returns a set of symbolic attacks U such that, for every instance $\boldsymbol{x} \in \mathcal{X}$ and every adversarial instance $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $t(\boldsymbol{z}) \neq t(\boldsymbol{x})$, there exists $s \in U$ such that $\boldsymbol{x} \in s.pre$ and $\boldsymbol{z} \in s.post$.

We conjecture that the analysis is not only sound but also complete, i.e., for every $s \in U$ there exist $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $\boldsymbol{x} \in s.pre$, $\boldsymbol{z} \in s.post$ and $t(\boldsymbol{z}) \neq t(\boldsymbol{x})$. However, we do not formally prove this result because it has limited practical value: in particular, single decision trees are very rarely used in practice and have to be combined in an ensemble to provide enough predictive power. Since the ensemble analysis in the next section is sound but not complete, the conjectured completeness result would not generalise to practical cases.

10.4.3 Forest Stability Analysis

We now discuss how the stability analysis for decision trees can be generalised to tree ensembles by means of an iterative algorithm (Algorithm 10). The algorithm operates by refining a set of candidates C where the ensemble F may be unstable. Initially, the set C is the union of the symbolic attacks computed for the individual

Algorithm 10 Stability analysis for tree ensembles

```
1: function ANALYZE(F)
2:     Input
3:         F : forest
4:     Output
5:         C ∪ E : refined set of symbolic attacks where F may be unstable

6:     C ← ∅
7:     for t_i ∈ F do
8:         C ← C ∪ ANALYZE(t_i)
9:     E ← ∅
10:    while stopping condition is not met do
11:        for s ∈ C do
12:            if ∃y : F(s.pre) = F(s.post) = {y} then
13:                C ← C \ {s}
14:            else
15:                if F(s.pre) ∩ F(s.post) ≠ ∅ then
16:                    C ← (C \ {s}) ∪ SPLIT(s)
17:                else
18:                    C ← C \ {s}
19:                    E ← E ∪ {s}
20:    return C ∪ E
```

trees $t_i \in F$ (lines 6-8) by Algorithm 6. The algorithm also keeps track of an initially empty set of symbolic attacks E where the analysis has ended because no further refinement of them is possible (line 9).

Each iteration of the algorithm inspects all the candidates $s \in C$, distinguishing three cases (lines 10-19). If F predicts the same label y over both the pre-image and the post-image of s, then F is stable on that portion of the feature space and s is removed from C (lines 12-13). Otherwise, the algorithm checks whether the predictions performed by F over the pre-image and the post-image of s share some common elements. If this is the case, then F may be stable on a subset of the pre-image of s, yet this cannot be concluded at the current iteration; hence, s is refined by splitting it into a set of smaller symbolic attacks (lines 15-16). Any splitting criterion would work as long as it satisfies the soundness condition defined in the theorem below. If, instead, the predictions performed by F over the pre-image and the post-image of s are disjoint, there is no way of proving that F is stable even on a subset of the pre-image of s; hence s is moved from C to E to avoid further refinements (lines 18-19).

The algorithm may implement an arbitrary stopping condition, e.g., C is empty, or a maximum number of iterations has been performed, as we do in our implementation. Similarly to the stability analysis for decision trees, the algorithm eventually returns a set of symbolic attacks $C \cup E$, whose union of the pre-images over-approximates the portion of the feature space where F is unstable. The soundness of the analysis is formalised by the following theorem, which is the natural generalisation to ensembles of Theorem 1.

Theorem 2 (Soundness of Tree Ensemble Analysis)**.** Assume the following hypotheses:

- $F(H)$ is sound, i.e., it satisfies the following: $\{y \in \mathcal{Y} \mid \exists \boldsymbol{x} \in H : F(\boldsymbol{x}) = y\} \subseteq F(H)$.
- The splitting procedure is sound, i.e., for all symbolic attacks s, if there exist $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $\boldsymbol{x} \in s.pre$ and $\boldsymbol{z} \in s.post$, then there exists $s' \in \textsc{Split}(s)$ such that $\boldsymbol{x} \in s'.pre$ and $\boldsymbol{z} \in s'.post$.

The call $\textsc{Analyze}(F)$ returns a set of symbolic attacks $C \cup E$ such that, for every instance $\boldsymbol{x} \in \mathcal{X}$ and every adversarial instance $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $F(\boldsymbol{z}) \neq F(\boldsymbol{x})$, there exists $s \in C \cup E$ such that $\boldsymbol{x} \in s.pre$ and $\boldsymbol{z} \in s.post$.

Note that the ensemble analysis is sound but not complete for generic reasons, the most important one being that even the traditional robustness verification problem for tree ensembles is already NP-hard [92] and computing all the equivalence classes induced by an ensemble quickly becomes infeasible in practice [164]. We expect a complete analysis to be feasible for restricted settings, e.g., for specific attackers.

10.5 Experimental Setup

In the experimental phase, we compare the presented resilience metric with robustness to demonstrate how robustness creates a false sense of security due to its definition. Robustness evaluates the model's strength to evasion attacks based solely on instances from a given test set, without considering possible neighbourhood instances of those in the test set. The experiments designed in this Section aim to answer the following research questions: Do the theoretical shortcomings of robustness manifest in practice? Can accurate resilience estimates be computed using our data-independent stability analysis, and are these estimates practically useful? What is the impact of the neighbour parameter ε on the resilience estimates that we can compute?

Table 10.3: Datasets properties.

dataset	#features	#instances	perturbation δ
Diabetes	8	768	$\pm$ 0.03
Cod-RNA	8	59,535	$\pm$ 0.07
Breast Cancer O	10	683	$\pm$ 0.15
Sensorless 1 vs. 11 (1 vs. 11)	48	10,638	$\pm$ 0.20

To demonstrate the superiority of resilience over robustness, we implemented the data-independent stability analysis defined in Section 10.4. Through the data-independent stability analyser, we experimentally assess the effectiveness of resilience on four publicly available datasets: Diabetes, Cod-RNA, Breast Cancer O, and Sensorless. Furthermore, we estimate the robustness and resilience of both standard and robust tree models trained using a state-of-the-art adversarial machine learning algorithm.

In Table 10.3, we summarise the main properties of the dataset, which were introduced previously in Section 8.2.5. These datasets have been employed in previous research on the robustness verification of tree-based models, as seen in Chen et al.'s work [46]. Due to our focus on binary classification, we create a new dataset named Sensorless 1 vs. 11 by selecting classes 1 and 11 from the original dataset Sensorless. For the sake of simplicity, hereinafter, we refer to Sensorless 1 vs. 11 as Sensorless, as it is the only version of the Sensorless dataset used in the experiments. Furthermore, we refer to Breast Cancer O as Breast Cancer.

The experimental phase aims to verify the usefulness of the newly introduced metric, which assesses the strength (security) of classification models under attack. For this analysis, there are no baselines; however, we use the robustness metric defined in Section 8.2.3.3 as a competitor. The robustness metric estimates the model's strength under attack, but we claim that its estimate provides a false sense of security. Furthermore, regarding the effectiveness of the data-independent stability analyser to estimate resilience, since we introduce a new metric, there is no competitor with which to compare the accuracy of our solution. For this reason, we designed a series of experiments to prove that our analyser is indeed accurate.

10.5.1 Baselines and Implementation

Data-Independent Stability Analyser: In the experiments, we leverage the output of the stability analysis to calculate lower bounds for robustness ($\hat{\mathrm{R}}$) and resilience ($\hat{\mathcal{R}}$) for a specific model and test set, as discussed in Section 10.3.2. Therefore, an efficient and accurate implementation of the data-independent sta-

bility analyser is crucial for the successful execution of our experiments.

Concerning the implementation of the data-independent stability analysis formalised in Section 10.4, we implemented our stability analyser for decision trees and decision tree ensembles based on majority voting. The implementation is entirely in the C++ programming language and parallelised on CPU architecture. The code of the analyser can be found at the following footnote[1].

We discuss below selected aspects of the implementation, which is a rather direct translation of our pseudo-code. A first point to note is that the set of candidates C is implemented by means of a min priority queue, and only the top k symbolic attacks can be split at each loop iteration (lines 11-19 of Algorithm 10) to mitigate the growth of C. The default value of k is $0.05 \cdot |C|$.

The priority queue is ordered according to the following criterion: each symbolic attack s is first assigned a pair (n_c, n_u), where n_c is a counter used to keep track of how many splits have been performed to produce s and n_u is the number of "undecided" trees t_i such that $|t_i(s.pre)| > 1$; pairs are then ordered according to the standard lexicographic order. In this heuristics, n_c acts as a penalisation factor to ensure that the algorithm does not split the same symbolic attacks too many times but rather tries to process all the symbolic attacks at least once, even when the number of iterations is relatively small. Symbolic attacks with the same value of n_c are split by prioritising symbolic attacks with a small number of undecided trees n_u because they are intuitively and likely easier to certify and should be analysed earlier.

A second relevant aspect to discuss is the implementation of the splitting function (line 16 of Algorithm 10). Given the symbolic attack s, our implementation of $\textsc{Split}(s)$ operates as follows: it first identifies a feature f and a threshold v such that v falls in the f-th component of $s.pre$; then, if $I_f^{atk} = [\delta_l, \delta_r]$, it splits $s.pre$ in (at most four) hyper-rectangles based on the thresholds $v + \delta_l$, v, $v + \delta_r$. For example given the interval $(a, b]$, if v, $v+\delta_l$ and $v+\delta_r$ are inside $(a, b]$, the $\textsc{Split}(s)$ function divides the interval in the following four intervals: $(a, v + \delta_l]$, $(v + \delta_l, v]$, $(v, v + \delta_r]$ and $(v + \delta_r, b]$. Finally, it uses these intervals as the pre-images of the new symbolic attacks, computing the corresponding post-images by intersecting $s.post$ with the maximum perturbation applicable to the pre-images. It is easy to show that this implementation enjoys the required soundness condition for the splitting procedure.

Finally, we note that Algorithm 10 can be parallelised by partitioning C across different threads and by joining the analysis results at the end. In particular, we first build the priority queue C, and then we distribute it across threads using a round-robin algorithm, which is useful to ensure that no thread is penalised by the prevalence of symbolic attacks which are expected to be hard to analyse. This

[1] https://github.com/FedericoMarcuzzi/resilience-verification

scheme ensures a deeper exploration of C when the analysis terminates before convergence after a fixed number of iterations. Our implementation supports a configurable number of threads, and in the experiments, we fixed this value to 20 threads.

Learning Algorithms: In the experiments, we compare resilience and robustness on models trained without considering the attacker (standard models) and while considering the attacker (robust models). Specifically, we employ Random Forest to train standard models and TREANT to train robust models. For training Random Forest, we utilise the open-source Scikit-learn library [24]. While for TREANT, we use the implementation provided by Calzavara *et al* [36], and the corresponding code can be found at the following footnote[2].

Below, we provide details of the learning algorithms.

- Random Forest (RF): The RF algorithm, as defined by Breiman [20], is not explicitly designed to be robust against evasion attacks; however, it is known to exhibit some level of robustness thanks to the ensembling of several decision trees. As in the original algorithm, each tree is trained on a bootstrap sample of the dataset and with feature sampling of size $\sqrt{|\mathcal{F}|}$ at each node.

- TREANT: TREANT is an algorithm designed for training robust models against evasion attacks. The family of attacks handled by TREANT is very general and includes the evasion attacks generated by the adversary's model defined in Section 10.2. TREANT is intended to train robust decision trees against evasion attacks. In the paper proposed by Calzavara *et al.* in [36], it is demonstrated that to increase robustness, it is possible to ensemble the robust trees to create a robust forest.

To train the models, we divided each dataset into train and test sets according to an 80%-%20 scheme with stratified random sampling. In this work, we do not use validation since we are not interested in directly comparing the robustness among models trained by different learning algorithms (i.e. RF and TREANT). We are rather interested in the behaviour of resilience and robustness with respect to the structure of the ensembles: robust training or not, the number of trees in the ensemble and the ensemble trees' depth.

To compare the feature perturbation given by δ and the size of the neighbourhood set N modelled by ε across different features, for each dataset, we normalise each feature in the range $[0, 1]$. For each classifier, we leverage the test

[2]https://github.com/FedericoMarcuzzi/TREANT

set $\mathcal{D}_{test}$ to compute different measures: accuracy ACC, robustness R, its under-approximation $\hat{\text{R}}$, and the under-approximation of resilience $\hat{\mathcal{R}}$. The robustness R is computed using the exact brute-force algorithm BF defined di Section 9.4.1, while $\hat{\text{R}}$ and $\hat{\mathcal{R}}$ are under-approximations computed by our data-independent stability analysis defined in Section 10.3.2.

Following the definition of the adversary's model provided by Definition 10, for each feature $f \in \mathcal{F}$, we fix the cost $c_f = 1$. Consequently, with a budget of b, the attacker can attack a maximum of b features. Furthermore, in Table 10.3, we report, for each dataset, the values of the perturbation δ. The attacker can add to each feature f being attacked a perturbation sampled from the interval $I_f^{atk} = [-\delta, +\delta]$. The value of δ depends on the dataset because different datasets are drawn from different distributions. Hence, attacks that are effective on models trained over a given dataset may be too strong or too weak for models trained on a different dataset [49]. Finally, when estimating resilience, we assume the neighborhood $N(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathcal{X} \mid ||\boldsymbol{x}' - \boldsymbol{x}||_\infty \leq \varepsilon\}$ for a given value of ε defined in the experiments. Observe that the actual resilience $\mathcal{R}$ is unknown, and only the estimate $\hat{\mathcal{R}}$ can be computed by our analysis.

10.6 Main Results

In this section, we present the main results obtained in this study. The section primarily focuses on the outcomes of experiments aimed at assessing whether robustness is an effective metric for estimating the security of machine learning models under attack. Additionally, it examines the ability of the analyser to provide an accurate approximation of resilience and whether this estimate offers a more reliable assessment of security compared to robustness.

10.6.1 Shortcomings of Robustness

We set up the first experiment to understand whether the shortcomings of robustness identified in theory might also occur in practical scenarios. To do that, for each dataset, we use the original test set $\mathcal{D}_{test}$ to craft 100 synthetic test sets $\mathcal{D}_{test}^1, \ldots, \mathcal{D}_{test}^{100}$ obtained by replacing each instance $\boldsymbol{x} \in \mathcal{D}_{test}$ with a randomly sampled instance $\boldsymbol{x}' \in N(\boldsymbol{x})$. We then compute the robustness of the trained classifiers over all the test sets $\mathcal{D}_{test}^i$, reporting the best and worst results to understand to which extent a "lucky" sample of the data distribution might give a false sense of security. To ensure that the synthetic test sets are still representative of the same data distribution used for training, we only consider cases where the classifier roughly preserves the same accuracy computed on the original test set $\mathcal{D}_{test}$.

Table 10.4 and Table 10.5 present the experimental results of our evaluation

Table 10.4: Shortcomings of robustness on *standard models*, for fixed $b = 1$.

Dataset	ε	ACC	ACC_{min}	ACC_{max}	$\%\Delta\text{ACC}$	R	R_{min}	R_{max}	$\%\Delta\text{R}$
Diabetes	0.01	71.4	70.8	72.1	1.3	64.9	64.3	66.2	1.9
	0.02	71.4	70.8	71.4	0.6	64.9	63.0	66.2	3.2
	0.03	71.4	68.8	71.4	2.6	64.9	**63.6**	**68.2**	**4.6**
	0.04	71.4	68.8	72.7	3.9	64.9	**63.0**	**70.1**	**7.1**
Cod-RNA	0.01	77.5	77.4	77.5	0.1	68.6	67.6	69.0	1.4
	0.02	77.5	77.3	77.5	0.2	68.6	66.5	68.6	2.1
	0.03	77.5	77.3	77.5	0.2	68.6	65.7	68.6	2.9
	0.04	77.5	76.8	77.5	0.7	68.6	65.0	68.6	3.6
Breast Cancer	0.05	94.8	94.8	94.8	0.0	92.6	92.6	94.1	1.5
	0.06	94.8	93.3	95.6	2.3	92.6	**91.1**	**95.6**	**4.5**
	0.07	94.8	94.1	95.6	1.5	92.6	**90.4**	**95.6**	**5.2**
	0.08	94.8	93.3	97.0	3.7	92.6	**90.4**	**96.3**	**5.9**
Sensorless	0.03	100.0	100.0	100.0	0.0	98.5	98.5	99.6	1.1
	0.04	100.0	100.0	100.0	0.0	98.5	98.5	99.6	1.1
	0.05	100.0	100.0	100.0	0.0	98.5	98.1	99.3	1.2
	0.06	100.0	100.0	100.0	0.0	98.5	97.5	98.9	1.4

for standard and robust models, respectively. For this experiment, we assume an attacker with budget $b = 1$. The tables report for each dataset the worst robustness R_{min} and the best robustness R_{max} computed over all the generated synthetic test sets for different values of ε, as well as the percentage robustness gap $\%\Delta\text{R}$ between the two. Furthermore, we also report the corresponding values of accuracy noted as ACC_{min} and ACC_{max}, respectively and their percentage accuracy gap $\%\Delta\text{ACC}$. The tables also include the accuracy ACC and the robustness R computed on the original test set $\mathcal{D}_{test}$. Finally, we mark in bold when the gap $\%\Delta\text{R}$ is at least 4%. The experiments are performed on ensembles of size $|F| = 7$.

The results show that the size of the interval $[\text{R}_{min}, \text{R}_{max}]$ is significant in many cases and may reach 7% for the highest values of ε, while the size of the interval $[\text{ACC}_{min}, \text{ACC}_{max}]$ is relatively small in comparison, spanning at most 4%. For example, the robustness of the standard model trained over the Breast Cancer dataset suffers from a fluctuation of around 5% for $\varepsilon = 0.07$, while the corresponding accuracy gap fluctuates at just 1%. Our experiments show that robustness is generally more sensitive to small amounts of noise than accuracy. Remarkably, this observation applies to both standard and robust models. Robust models provide higher robustness than standard models; however, the interval $[\text{R}_{min}, \text{R}_{max}]$ may have a significant size also for them, i.e., roughly 6% in the

Table 10.5: Shortcomings of robustness on *robust models*, for fixed $b = 1$.

Dataset	ε	ACC	ACC_{min}	ACC_{max}	%ΔACC	R	R_{min}	R_{max}	%ΔR
Diabetes	0.01	72.7	72.1	72.7	0.6	71.4	67.5	71.4	3.9
	0.02	72.7	71.4	74.0	2.6	71.4	**66.9**	**72.1**	**5.2**
	0.03	72.7	72.1	74.7	2.6	71.4	**66.9**	**72.7**	**5.8**
	0.04	72.7	70.8	74.7	3.9	71.4	**67.5**	**73.4**	**5.9**
Cod-RNA	0.01	75.0	74.8	75.3	0.5	71.4	71.0	72.1	1.1
	0.02	75.0	74.9	75.8	0.9	71.4	71.1	72.5	1.4
	0.03	75.0	75.0	76.0	1.0	71.4	70.5	72.3	1.8
	0.04	75.0	75.2	76.1	0.9	71.4	70.3	72.3	2.0
Breast Cancer	0.05	97.0	93.3	96.3	3.0	95.6	**91.9**	**96.3**	**4.4**
	0.06	97.0	93.3	97.0	3.7	95.6	**91.1**	**96.3**	**5.2**
	0.07	97.0	92.6	96.3	3.7	95.6	**91.1**	**96.3**	**5.2**
	0.08	97.0	92.6	95.6	3.0	95.6	**90.4**	**95.6**	**5.2**
Sensorless	0.03	100.0	99.9	100.0	0.1	99.9	99.5	99.9	0.4
	0.04	100.0	99.6	100.0	0.4	99.9	99.0	99.9	0.9
	0.05	100.0	99.3	100.0	0.7	99.9	97.9	99.9	2.0
	0.06	100.0	99.0	100.0	1.0	99.9	96.2	99.9	3.7

worst case. This shows that a security evaluation based on robustness may give a false sense of security for both standard and robust models. We also remark that our experiment still provides a conservative account of the actual limitations of robustness, being based on just 100 synthetic test sets: the actual gap between R_{min} and R_{max} within the neighbourhood N may be larger in practice.

10.6.2 Effectiveness of Resilience Verification

We now investigate the effectiveness of our resilience verification technique. To do that, we would like to show that our estimate $\hat{\mathcal{R}}$ is an accurate under-approximation of the actual resilience $\mathcal{R}$ and that resilience significantly mitigates the shortcomings of robustness. Unfortunately, since the actual value of resilience is unknown, we can only provide a best-effort answer to the first point. Our evaluation is based on two independent experiments:

1. In the first one, we operate by comparing the similarity between the actual robustness R and its estimate $\hat{\text{R}}$ computed by our analysis. We consider the similarity between R and $\hat{\text{R}}$ as a proxy for the precision of the stability analysis underlying our resilience verification technique: the more R and $\hat{\text{R}}$ are close to each other, the more the stability analysis is effective at detecting

the portions of the feature space where the classifier is stable, which suggests that also the estimate $\hat{\mathcal{R}}$ is precise, being based on the same stability analysis.

2. In the second one, we refer to the experiment in the previous section, and we observe that if a classifier is not robust on the instance $\boldsymbol{x}'$ belonging to some $\mathcal{D}^i_{test}$ with $i \in [1, 100]$, by construction, there must exist $\boldsymbol{x} \in \mathcal{D}_{test}$ such that the classifier is not robust on at least one instance in $N(\boldsymbol{x})$. This allows the construction of an additional test set $\overline{\mathcal{D}_{test}}$, corresponding to the "most unlucky" sampling within the neighbourhood N of the original $\mathcal{D}_{test}$, i.e., the one with the lowest robustness $\overline{\mathrm{R}}$. The measure $\overline{\mathrm{R}}$ is interesting because it is based on an exact robustness verification technique: if $\overline{\mathrm{R}}$ is close to $\hat{\mathcal{R}}$, then we have a proof that most instances where the classifier is not considered resilient by our analysis are indeed insecure with respect to some evasion attacks.

Note that the second experiment does not just prove the precision of our approximated resilience verification technique, but it also gives a clear security interpretation of the benefits of resilience over robustness.

Does Resilience Provide a Better Security Estimate than Robustness? Table 10.6 and Table 10.6 show the experimental results of our evaluation for standard and robust models, respectively, with an attacker's budget $b = 1$. The results highlight that the estimate $\hat{\mathrm{R}}$ is a rather precise under-approximation of the actual robustness R: in particular, for individual decision trees $\hat{\mathrm{R}}$ always coincides with R. As for tree ensembles, the gap between the two measures increases, yet it is still quite small (roughly 2%) for standard models and most often negligible for robust models; hence, we expect that also $\hat{\mathcal{R}}$ is a reasonably accurate estimate of the actual resilience $\mathcal{R}$. The table also shows that the gap between R and $\hat{\mathcal{R}}$ may be quite significant, both for standard and robust models, often reaching a value of around 6% and even reaching 10% or more in some cases. We highlight in bold the cases where the gap between R and $\hat{\mathcal{R}}$ is at least 5%. Remarkably, it is apparent that the resilience estimate $\hat{\mathcal{R}}$ provides a much more realistic security assessment than robustness because the value of $\overline{\mathrm{R}}$ is much closer to $\hat{\mathcal{R}}$ than to R in the very large majority of cases. Since $\overline{\mathrm{R}}$ captures effective evasion attacks against instances within close neighbourhoods of the test set, this confirms that $\hat{\mathcal{R}}$ is not overly conservative in practice.

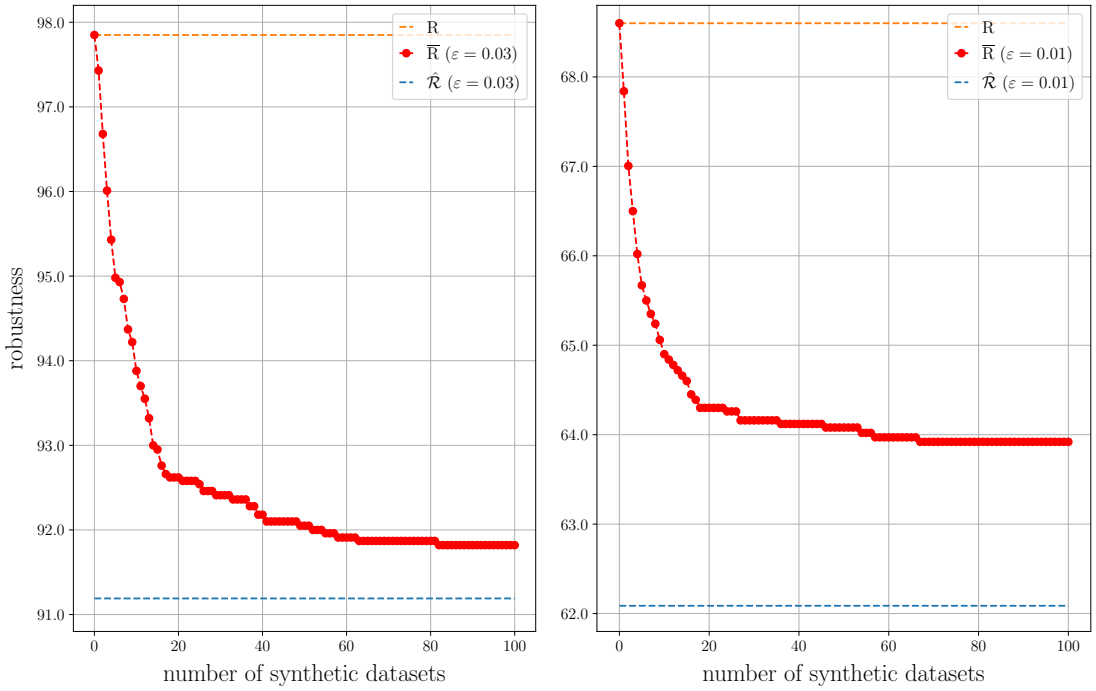
To further substantiate our claims, Figure 10.3 shows how robustness is affected when varying the number of synthetic datasets used to construct $\overline{\mathcal{D}_{test}}$, which was built from 100 datasets in our experiments. We performed this analysis with standard models with 5 trees and depth 3 trained on SENSORLESS and COD-RNA datasets. In particular, the figure plots how the value of $\overline{\mathrm{R}}$ changes when varying

Table 10.6: Computed measures for different datasets and Standard Models (for fixed $b = 1$)

dataset	ε	$\lvert F\rvert$	depth	ACC	R	$\hat{\mathrm{R}}$	$\overline{\mathrm{R}}$	$\hat{\mathcal{R}}$
Diabetes	0.01	1	3	67.5	62.3	62.3	62.3	62.3
		1	5	72.1	63.6	63.6	61.7	61.7
		1	7	72.7	**61.0**	61.0	**53.9**	**53.9**
		5	3	70.8	66.2	64.3	65.6	63.6
		7	3	71.4	64.9	63.0	63.6	62.3
		9	3	74.7	65.6	63.0	62.3	61.7
Cod-RNA	0.01	1	3	77.4	68.3	68.3	63.8	63.7
		1	5	87.4	**43.3**	43.3	**33.4**	**33.0**
		1	7	80.4	**57.5**	57.5	**47.4**	**47.2**
		5	3	77.5	**68.6**	67.2	**63.9**	**62.1**
		7	3	77.5	**68.6**	66.6	**64.0**	**61.2**
		9	3	76.9	**67.7**	66.3	**62.5**	**60.5**
Breast Cancer	0.05	1	3	94.8	90.4	90.4	87.4	87.4
		1	5	95.6	91.1	91.1	87.4	87.4
		1	7	94.8	90.4	90.4	86.7	86.7
		5	3	94.1	**92.6**	90.4	**90.4**	**86.7**
		7	3	94.8	92.6	91.1	92.6	88.1
		9	3	96.3	**94.1**	91.9	**93.3**	**88.9**
Sensorless	0.03	1	3	95.3	**76.3**	76.3	**46.1**	**46.1**
		1	5	100.0	**93.3**	93.3	**40.1**	**39.9**
		1	7	100.0	**93.3**	93.3	**40.1**	**39.9**
		5	3	100.0	**97.8**	97.3	**91.8**	**91.2**
		7	3	100.0	**98.5**	97.7	**93.5**	**91.2**
		9	3	100.0	**99.2**	97.7	**94.2**	**90.5**

Table 10.7: Computed measures for different datasets and Robust Models (for fixed $b = 1$)

dataset	ε	$\lvert F\rvert$	depth	ACC	R	$\hat{\mathrm{R}}$	$\overline{\mathrm{R}}$	$\hat{\mathcal{R}}$
Diabetes	0.01	1	3	68.2	64.3	64.3	64.3	64.3
		1	5	67.5	63.6	63.6	63.0	63.0
		1	7	69.5	67.5	67.5	63.6	63.6
		5	3	72.7	**71.4**	70.1	**67.5**	**66.2**
		7	3	72.7	**71.4**	70.8	**67.5**	**66.2**
		9	3	75.3	**74.0**	72.7	**69.5**	**68.8**
Cod-RNA	0.01	1	3	75.0	71.4	71.4	69.8	69.8
		1	5	81.0	**70.3**	70.3	**64.3**	**64.1**
		1	7	81.0	**70.0**	70.0	**63.7**	**63.4**
		5	3	75.2	71.5	70.7	69.8	69.1
		7	3	75.0	71.4	71.3	69.8	69.7
		9	3	75.0	71.4	71.3	69.8	69.7
Breast Cancer	0.05	1	3	94.8	**94.1**	94.1	**85.9**	**85.9**
		1	5	95.6	**94.1**	94.1	**79.3**	**79.3**
		1	7	95.6	**94.1**	94.1	**79.3**	**79.3**
		5	3	97.8	**94.8**	94.1	**88.9**	**88.1**
		7	3	97.0	**95.6**	94.1	**88.9**	**87.4**
		9	3	97.0	**95.6**	93.3	**88.9**	**86.7**
Sensorless	0.03	1	3	100.0	50.1	50.1	50.1	50.1
		1	5	100.0	50.1	50.1	50.1	50.1
		1	7	100.0	99.9	99.9	99.3	99.3
		5	3	100.0	99.9	99.9	99.3	99.3
		7	3	100.0	99.9	99.9	99.3	99.3
		9	3	100.0	99.9	99.9	99.3	99.3



(a) SENSORLESS (b) COD-RNA

Figure 10.3: Comparison of Robustness $\overline{\text{R}}$ to exact robustness R and estimated resilience $\hat{\mathcal{R}}$ while varying the number of synthetic datasets used to create $\overline{\mathcal{D}_{test}}$.

the number of synthetic datasets from zero to 100. This evaluation is conservative because we consider the synthetic datasets $\mathcal{D}^i_{test}$ in decreasing order of robustness. The figure shows that less than 20 synthetic datasets are sufficient to have a decrease in model robustness of approximately 5% for SENSORLESS (Figure 10.3a) and 4% for COD-RNA (Figure 10.3b), i.e., it is easy to find a neighbour instance $\boldsymbol{x}' \in N(\boldsymbol{x})$ of an instance $\boldsymbol{x} \in \mathcal{D}_{test}$ for which the model is not robust. This finding confirms that robustness is not a good measure to assess security against evasion attacks. Moreover, even with a small number of synthetic datasets, the value of $\overline{\text{R}}$ is closer to our approximated resilience $\hat{\mathcal{R}}$ rather than to the robustness R.

Is the Resilience Analyser Accurate for Large Neighbours? We finally assess the role of the parameter ε on our resilience verification technique. For this experiment, we focus only on robust models trained on the DIABETES dataset. In particular, we set the attacker's budget $b = 1$, and we compute different resilience estimates for different values of ε. Of course, we expect resilience to decrease when increasing the value of ε because the stability guarantees required on the classifier become more demanding. Still, it is interesting to understand whether the quality

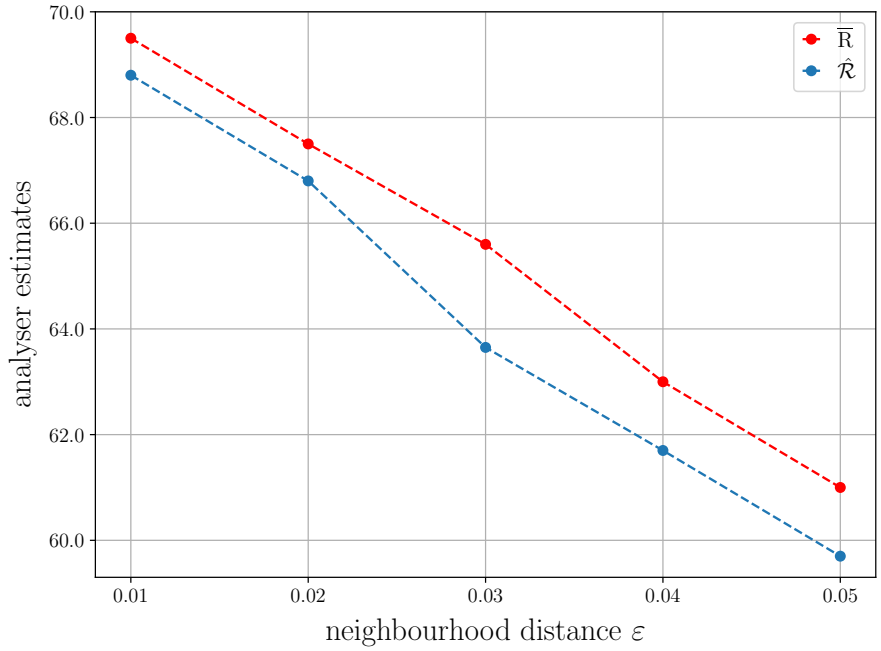


Figure 10.4: Estimated resilience $\hat{\mathcal{R}}$ and robustness $\overline{\mathrm{R}}$ for different values of ε.

of the resilience estimate $\hat{\mathcal{R}}$ is affected by the value of ε: to understand this, we compare $\hat{\mathcal{R}}$ against $\overline{\mathrm{R}}$ because we would like the two measures to be relatively close to each other. Figure 10.4 plots how the values of the estimated resilience $\hat{\mathcal{R}}$ and $\overline{\mathrm{R}}$ of ensembles of size 9 and depth 3 decreases when increasing ε from 0.01 to 0.05. The figure shows that $\hat{\mathcal{R}}$ and $\overline{\mathrm{R}}$ are consistently close to each other, with a maximum difference of 2%. This proves that the estimate of the resilience always captures possible evasion attacks, i.e., the precision of our approximated analysis does not downgrade when increasing the value of ε.

10.7 In-Depth Analysis

In this section, we delve into a comprehensive analysis of the efficiency of the implementation of the data-independent stability analysis defined in Section 10.4. Furthermore, we provide formal proofs establishing the correctness of the theorems underlying the soundness of the data-independent stability analysis: given the region of the feature space $C \cup E$ returned by Analyze(F), where test instances might be unstable, all instances in the region $\mathcal{X} \setminus (C \cup E)$ are proved stable.

10.7.1 Performance Evaluation

In this experiment, we delve into the performance details of our implementation of the data-independent stability analyser when analysing ensembles of decision trees. For this analysis, we focus only on ensembles of robust models (i.e., trained with the TREANT algorithm) on the DIABETES dataset.

To better understand the analyses in this section, it is essential to recall that assessing the performance of the ensemble analysis is more nuanced due to the exponential complexity blowup underlying the verification of decision tree ensembles [164]. Although our analyser was intentionally designed to support iterative refinements, consequently, its performance both in terms of accurate estimates and execution time depends crucially on the number of analysis iterations.

Hence, we are interested in examining the following three aspects: *i)* Understanding how much the analysis execution time (up to convergence) changes when increasing the ensemble size. *ii)* Understanding how much the quality of the robustness and resilience estimates (i.e., $\hat{\mathrm{R}}$ and $\hat{\mathcal{R}}$) changes when increasing the number of iterations while keeping the same ensemble size. *iii)* Finally, understanding how much the execution time of the analyser is affected by the attacker's budget b.

The first point provides insights into the scalability of the analysis to increasingly larger models, while the second allows us to understand whether it is possible to compute useful robustness and resilience estimates even when the analysis becomes intractable and the number of iterations is limited to forcefully stop the analysis before convergence. Finally, the third point provides insights into the efficiency of the analyser when considering attacks of different strengths. Below, we present the results of the analyses conducted for each of these points.

Ensemble Size vs. Execution time Figure 10.5 presents the results of the analysis addressing the first point: how the ensemble size affects the analyser's execution time. We analyse the change in execution time by increasing the ensemble size from 9 to 17, with each tree in the ensemble having a depth of 3. Small ensembles with 9 trees can be analysed in a matter of seconds, while larger ensembles with 17 trees can be analysed in around 16 minutes. We consider this result promising because the stability analysis is data-independent, i.e., it can be computed just once and then applied to establish different properties on different test sets. We expect the execution times to be further improvable by sacrificing a bit of precision, e.g., by aggregating together symbolic attacks that are close to each other in the feature space.

Iterations vs. Accurate Estimates Figure 10.6 illustrates the results of the analysis addressing the second point: how the number of analyser iterations affects

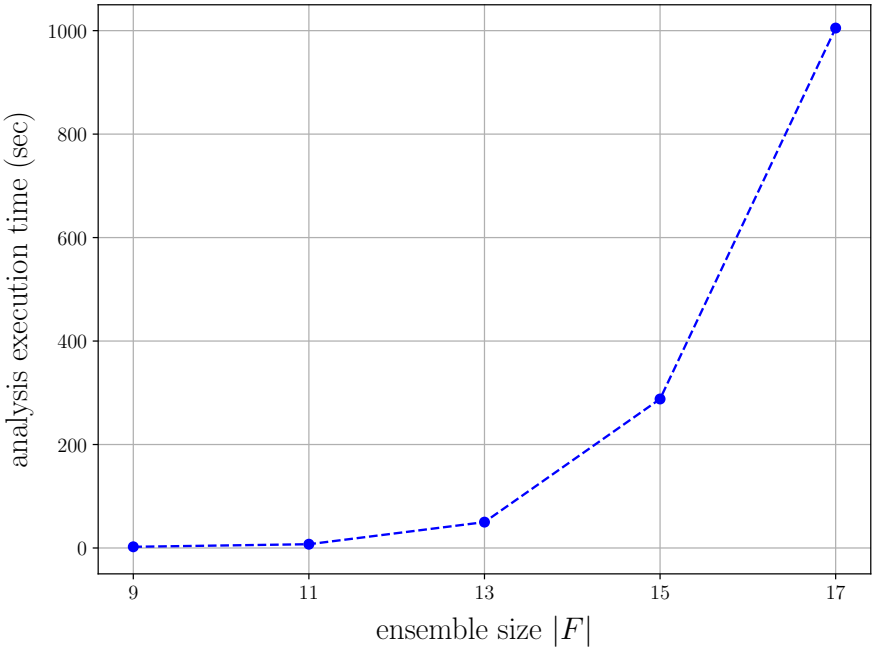


Figure 10.5: Execution time analysis by varying the ensemble size.

the accuracy of the robustness and resilience estimates. The plot depicts the change in values of the estimates $\hat{\mathrm{R}}$ and $\hat{\mathcal{R}}$ of ensembles of 17 trees and depth 3 when increasing the number of analysis iterations. The figure reveals a desirable trend, with a significant increase in the estimates of robustness and resilience in the first 120 iterations before reaching a plateau. This indicates that reasonably accurate estimates of robustness and resilience can be established even with a limited number of iterations of the analysis. This is crucial for scalability, as useful results can be obtained before analysis convergence. Indeed, our analyser is designed to prioritise portions of the feature space that are intuitively easier to prove as stable (or not).

Attacker's Budget vs. Execution time The last experiment assesses the third point: how the attacker's budget b impacts the analyser's execution time. Specifically, we investigate how the analyser execution time changes for different values of b when analysing an ensemble of 11 trees of depth 3 up to convergence. The ensemble is trained with the TREANT robust learning algorithm on the DIABETES dataset, assuming an attacker's budget of 5. The results of the analysis are shown in Figure 10.7. As we can see, the impact of the attacker's budget on the analyser's execution time is much more limited than the impact of the size of the ensemble. The execution times range from around 12 seconds to around 40

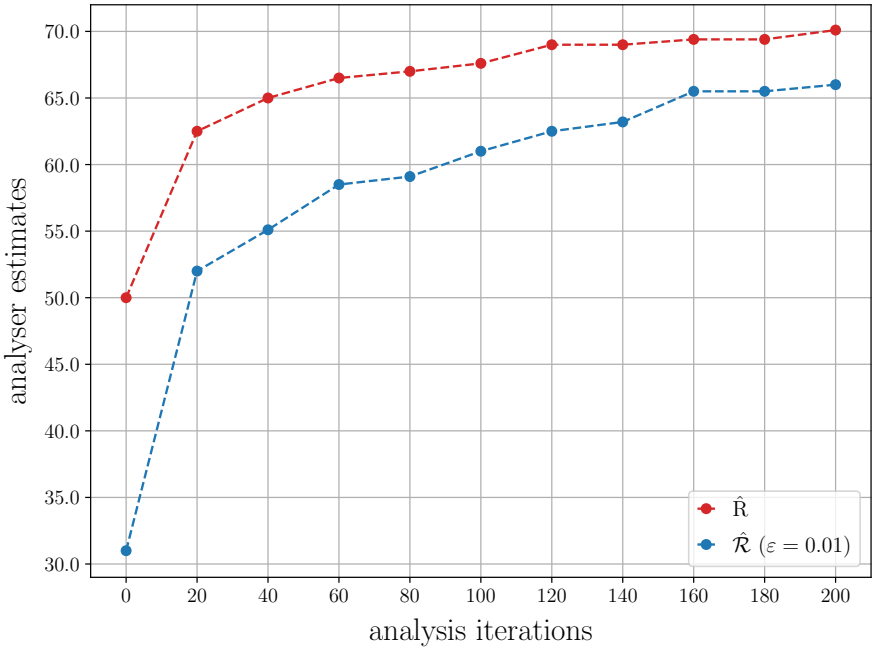


Figure 10.6: Estimated robustness $\hat{\mathrm{R}}$ and resilience $\hat{\mathcal{R}}$ by varying the number of analyser's iterations.

seconds when varying the attacker's budget from 1 to 5.

10.7.2 Proofs of Theorems

In this section, we present the correctness proofs for Theorem 1 and Theorem 2. Theorem 1 concerns the soundness of the tree analysis ANALYZE(t) provided by Algorithm 9. Instead, Theorem 2 addresses the soundness of the tree analysis ANALYZE(F) provided by Algorithm 10.

10.7.2.1 Proof of Theorem 1

In this section, we provide the proof of the correctness of Theorem 1. The proof leverages a key technical lemma (Lemma 1) formalising the soundness of the tree annotation function in Algorithm 6. More specifically, we prove that the tree annotation function always produces a *well-annotated* decision tree according to the following definition.

Definition 12 (Well-Annotated Decision Tree)**.** The node σ of the decision tree t is *well-annotated* by the set of symbolic attacks S if and only if, for every instance $\boldsymbol{x} \in \mathcal{X}$ and every $\boldsymbol{z} \in A(\boldsymbol{x})$ such that σ is traversed in the prediction $t(\boldsymbol{z})$, S

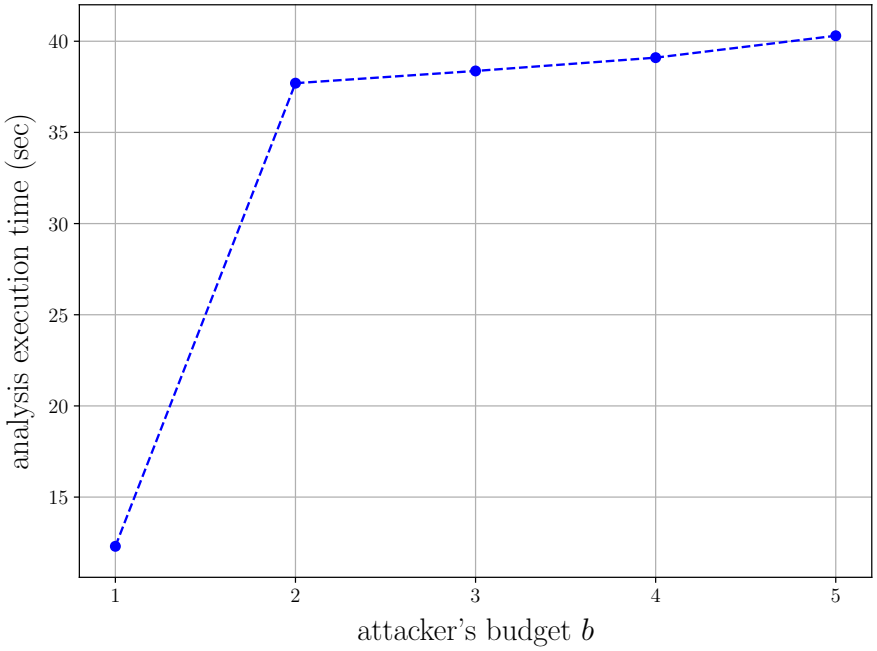


Figure 10.7: Execution time analysis by varying the attacker's budget b.

contains an element $\langle I_1^{pre}, \ldots, I_d^{pre}\rangle \triangleright \langle I_1^{post}, \ldots, I_d^{post}\rangle_k$ such that $\boldsymbol{x} \in \langle I_1^{pre}, \ldots, I_d^{pre}\rangle$, $\boldsymbol{z} \in \langle I_1^{post}, \ldots, I_d^{post}\rangle$ and k is the minimum cost to pay to make $\boldsymbol{x}$ traverse σ. We say that the decision tree t is well-annotated if and only if all its nodes are well-annotated by the set of symbolic attacks stored in their *sym* attribute.

Lemma 1 (Soundness of Tree Annotation)**.** The call $\textsc{Annotate}(t, S)$ returns a well-annotated decision tree, provided that the root of t is well-annotated by S.

Proof of the correctness of Lemma 1. The proof is by induction on the depth of the tree t. If the tree has depth 1, then it includes a single node, i.e., the root, and the conclusion follows by the assumption that the root of t is well-annotated by S. Otherwise, we have $t = \sigma(f, v, t_r, t_r)$ for some feature f, threshold v and sub-trees t_l and t_r. The function then computes two new sets of symbolic attacks S_l and S_r before invoking $\textsc{Annotate}(t_l, S_l)$ and $\textsc{Annotate}(t_r, S_r)$. Hence, the desired conclusion follows by inductive hypothesis, provided that we are able to show that the roots of t_l and t_r are well-annotated by S_l and S_r respectively. We just prove the former since the latter uses an equivalent reasoning.

Pick any instance $\boldsymbol{x} \in \mathcal{X}$ and consider any $\boldsymbol{z} \in A(x)$, we observe that S must contain an element s such that $\boldsymbol{x} \in s.pre$, $\boldsymbol{z} \in s.post$ and $s.cost = 0$, because all instances must traverse the root. Assume $z^{(f)} \leq v$, we prove that $\textsc{RefineLeft}(s, f, v)$ returns a set of symbolic attacks $S'_l \subseteq S_l$ such that there

exists $s' \in S'_l$ such that $\boldsymbol{x} \in s'.pre$, $\boldsymbol{z} \in s'.post$ and $s'.cost$ is the minimum cost to pay to make $\boldsymbol{x}$ traverse the left child of the root. Assume $s.pre = \langle I_1^{pre}, \ldots, I_d^{pre} \rangle$, $s.post = \langle I_1^{post}, \ldots, I_d^{post} \rangle$ and $I_f^{atk} = [\delta_l, \delta_r]$, we discriminate four cases:

- If $I_f^{pre} = I_f^{post}$ and $x^{(f)} \leq v$, we leverage the observation that $\boldsymbol{z} \in s.post$ and $z^{(f)} \leq v$, hence the condition $I_f^{post} \cap (-\infty, v] \neq \emptyset$ at line 13 must be satisfied. In this case, S'_l must contain an s'' such that:
  - $s''.pre = \langle I_1^{pre}, \ldots, I_{f-1}^{pre}, I_f^{pre} \cap (-\infty, v], I_{f+1}^{pre}, \ldots, I_d^{pre} \rangle$
  - $s''.post = \langle I_1^{post}, \ldots, I_{f-1}^{post}, I_f^{post} \cap (-\infty, v], I_{f+1}^{post}, \ldots, I_d^{post} \rangle$
  - $s''.cost = 0$

  The conclusion follows from the observation that s'' satisfies the three required conditions on s'.

- If $I_f^{pre} = I_f^{post}$ and $x^{(f)} > v$, we leverage the observation that $\boldsymbol{z} \in A(x)$ and $z^{(f)} \leq v$. This implies that $\delta_l < 0$, $x^{(f)} \in (v, v - \delta_l]$ and $z^{(f)} \in (v + \delta_l, v]$; moreover, we must have $c_f \leq b$. By combining all this information and the observation that $s.cost = 0$, we conclude that the condition at line 14 must be satisfied. In this case, S'_l must contain an s'' such that:
  - $s''.pre = \langle I_1^{pre}, \ldots, I_{f-1}^{pre}, I_f^{pre} \cap (v, v - \delta_l], I_{f+1}^{pre}, \ldots, I_d^{pre} \rangle$
  - $s''.post = \langle I_1^{post}, \ldots, I_{f-1}^{post}, I_f^{post} \cap (v + \delta_l, v], I_{f+1}^{post}, \ldots, I_d^{post} \rangle$
  - $s''.cost = c_f$

  The conclusion follows from the observation that s'' satisfies the three required conditions on s'.

- If $I_f^{pre} \neq I_f^{post}$ and $x^{(f)} \leq v$, we leverage the observation that $\boldsymbol{z} \in s.post$ and $z^{(f)} \leq v$, hence the condition $I_f^{post} \cap (-\infty, v] \neq \emptyset$ at line 13 must be satisfied. In this case, S'_l must contain an s'' such that:
  - $s''.pre = \langle I_1^{pre}, \ldots, I_{f-1}^{pre}, I_f^{pre} \cap (-\infty, v - \min(0, \delta_l)], I_{f+1}^{pre}, \ldots, I_d^{pre} \rangle$
  - $s''.post = \langle I_1^{post}, \ldots, I_{f-1}^{post}, I_f^{post} \cap (-\infty, v], I_{f+1}^{post}, \ldots, I_d^{post} \rangle$
  - $s''.cost = k$

  The conclusion follows from the observation that s'' satisfies the three required conditions on s'.

- If $I_f^{pre} \neq I_f^{post}$ and $x^{(f)} > v$, we leverage the observation that $\boldsymbol{z} \in A(x)$ and $z^{(f)} \leq v$. This implies that $\delta_l < 0$, $x^{(f)} \in (v, v - \delta_l]$ and $z^{(f)} \in (v + \delta_l, v]$. We then observe that $\boldsymbol{z} \in s.post$ and $z^{(f)} \leq v$, hence the condition $I_f^{post} \cap (-\infty, v] \neq \emptyset$ at line 13 must be satisfied. In this case, S'_l must contain an s'' such that:

  - $s''.pre = \langle I_1^{pre}, \ldots, I_{f-1}^{pre}, I_f^{pre} \cap (-\infty, v - \min(0, \delta_l)], I_{f+1}^{pre}, \ldots, I_d^{pre} \rangle$
  - $s''.post = \langle I_1^{post}, \ldots, I_{f-1}^{post}, I_f^{post} \cap (-\infty, v], I_{f+1}^{post}, \ldots, I_d^{post} \rangle$
  - $s''.cost = k$

  The conclusion follows from the observation that s'' satisfies the three required conditions on s'.

□

We now move back to the proof of the theorem. Consider an instance $\boldsymbol{x}$ and an adversarial perturbation $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $t(\boldsymbol{z}) \neq t(\boldsymbol{x})$. This means that there exist two leaves $\lambda(y)$ and $\lambda'(y')$ with $y \neq y'$ such that $t(\boldsymbol{x}) = y$ and $t(\boldsymbol{z}) = y'$. By Lemma 1, t must be well-annotated after line 5, hence we can make the following observations by Definition 12:

1. Since $\boldsymbol{x} \in A(\boldsymbol{x})$, the leaf $\lambda(y)$ were $\boldsymbol{x}$ falls must contain a symbolic attack $s = \langle I_1^{pre}, \ldots, I_d^{pre} \rangle \rhd \langle I_1^{post}, \ldots, I_d^{post} \rangle_k$ such that $\boldsymbol{x} \in \langle I_1^{pre}, \ldots, I_d^{pre} \rangle$, $\boldsymbol{x} \in \langle I_1^{post}, \ldots, I_d^{post} \rangle$ and $k = 0$.

2. Since $\boldsymbol{z} \in A(\boldsymbol{x})$, the leaf $\lambda'(y')$ were $\boldsymbol{z}$ falls must contain a symbolic attack $s' = \langle J_1^{pre}, \ldots, J_d^{pre} \rangle \rhd \langle J_1^{post}, \ldots, J_d^{post} \rangle_{k'}$ such that $\boldsymbol{x} \in \langle J_1^{pre}, \ldots, J_d^{pre} \rangle$, $\boldsymbol{z} \in \langle J_1^{post}, \ldots, J_d^{post} \rangle$ and k' is the minimum cost to pay to make $\boldsymbol{x}$ traverse $\lambda'(y')$. This cost must be greater than 0, because $t(\boldsymbol{z}) \neq t(\boldsymbol{x})$ implies $\boldsymbol{z} \neq \boldsymbol{x}$.

This implies that line 13 is reachable and $s.pre \cap s'.pre \neq \emptyset$; hence a new symbolic attack s'' is added to the return value U at lines 14-17. Thus, we just need to show that s'' satisfies the conditions of the theorem:

- We have that $\boldsymbol{x} \in s''.pre = s.pre \cap s'.pre$, by points 1 and 2.

- We have that $\boldsymbol{z} \in s'.post$ by point 2. Moreover, since $\boldsymbol{z} \in A(\boldsymbol{x})$, we must have $\boldsymbol{z} \in \boldsymbol{x} + \langle I_1^{atk}, \ldots, I_d^{atk} \rangle$ by definition of adversarial perturbation. Since $\boldsymbol{x} \in s''.pre$ by the previous point, we get $\boldsymbol{z} \in s''.pre + \langle I_1^{atk}, \ldots, I_d^{atk} \rangle$, hence we conclude $\boldsymbol{z} \in s'.post \cap (s''.pre + \langle I_1^{atk}, \ldots, I_d^{atk} \rangle) = s''.post$ as desired.

10.7.2.2 Proof of Theorem 2

In this section, we provide the proof of the correctness of Theorem 2. We prove the following invariant for the outer loop: for every instance $\boldsymbol{x} \in \mathcal{X}$ and every adversarial instance $z \in A(\boldsymbol{x})$ such that $F(\boldsymbol{z}) \neq F(\boldsymbol{x})$, there exists $s \in C \cup E$ such that $\boldsymbol{x} \in s.pre$ and $\boldsymbol{z} \in s.post$.

Let $F = \{t_1, \ldots, t_n\}$, consider an instance $\boldsymbol{x}$ and an adversarial instance $\boldsymbol{z} \in A(\boldsymbol{x})$ such that $F(\boldsymbol{z}) \neq F(\boldsymbol{x})$. We first prove the base case, i.e., we show that the invariant holds when no loop iteration has taken place. Initially, $C = \bigcup_i U_i$ where each U_i is computed by calling $\textsc{Analyze}(t_i)$. Since $F(\boldsymbol{z}) \neq F(\boldsymbol{x})$, there exists $t_i \in F$ such that $t_i(\boldsymbol{z}) \neq t_i(\boldsymbol{x})$. Hence, there exists $s \in U_i$ such that $\boldsymbol{x} \in s.pre$ and $\boldsymbol{z} \in s.post$ by Theorem 1. Then, the conclusion follows by the definition of C.

Assume now that the invariant holds up to a given iteration, we show it is preserved at the next iteration. By the inductive hypothesis, there exists a $s \in C \cup E$ such that $\boldsymbol{x} \in s.pre$ and $\boldsymbol{z} \in s.post$. We distinguish two cases. If $s \in E$, then the conclusion is immediate because nothing is ever removed from E. If instead $s \in C$, we show that each iteration of the inner loop cannot break the outer loop invariant. In particular, assume some $s' \in C$ is processed by an iteration of the inner loop, leading to updated C' and E', respectively. We can distinguish the following cases at the end of the iteration:

- If $C' = C \setminus \{s'\}$ and $E' = E$, then there exists y such that $F(s.pre) = F(s.post) = \{y\}$. This implies that for each instance $\boldsymbol{w} \in s.pre \cup s.post$, we have $F(\boldsymbol{w}) = y$, thanks to the first soundness condition. Since $F(\boldsymbol{x}) \neq F(\boldsymbol{z})$, we have that either $\boldsymbol{x} \notin s.pre$ or $\boldsymbol{z} \notin s.post$, hence $s' \neq s$ and the loop invariant is preserved.

- If $C' = (C \setminus \{s'\}) \cup \textsc{Split}(s')$ and $E' = E$, the loop invariant is preserved by the second soundness condition.

- if $C' = C \setminus \{s'\}$ and $E' = E \cup \{s'\}$, then $C' \cup E' = C \cup E$ and thus the loop invariant is preserved.

10.8 Contribution 3: Application in Fairness

In this Section, we illustrate the work titled "*Explainable Global Fairness Verification of Tree-Based Classifiers*", in proceedings as a full paper at the *SaTML '23: The 2023 IEEE Conference on Secure and Trustworthy Machine Learning*. Further details can be found in the reference [30].

The work introduced in this section diverges from the themes of this thesis as it pertains to fairness in machine learning. In this scenario, the algorithm is no longer susceptible to malicious attacks by an attacker in the operational phase

or noise and errors in the dataset during the training phase. Instead, it is subject to discrimination against users belonging to protected or sensitive categories. However, we demonstrate how the fairness problem addressed in this work, i.e., causal discrimination, can be easily transposed into a context of adversarial machine learning. Furthermore, we show that the data-independent stability analysis defined in Section 10.4 is crucial for understanding where the model is fair.

Before delving into the detailed contributions introduced in this work, it is essential to underline that in this section, we primarily focus on the transposition of the fairness problem into an adversarial problem and how the data-independent stability analysis is utilised, providing only a general description of the rest of the work and the obtained results.

The motivation behind this work stems from the recent discovery that machine learning models may exhibit *unfair* behaviour in the context of automated decision-making. For example, a commercial recidivism-risk assessment algorithm was found to be racially biased [98], and an existing algorithm adopted in the US falsely determined that black patients were healthier than other patients with similar conditions [133]. These incidents led to a proliferation of different research [41, 134, 125].

Fairness in ML has been analysed from different angles and can be broadly categorised into two main research lines. The first one includes the development of new ML algorithms that are able to mitigate the bias that is directly or indirectly present in the training data [2, 144, 150]. The second complementary research line investigates techniques to estimate or formally verify the fairness guarantees provided by existing ML models [91, 166, 87]. This paper contributes to the latter line of work, which is still at an early stage of development and suffers from relevant shortcomings.

A widely adopted method for assessing the fairness guarantees of machine learning models is based on *testing* [71, 1, 165, 17]. The fundamental intuition behind any fairness testing strategy is simple: generating various test inputs to automatically identify individuals who may experience discrimination by the machine learning model. Unfortunately, like any testing approach, this analysis is under-approximated; these proposals can identify indications of unfair treatment but cannot establish formal fairness proofs. This limitation is suboptimal as it hinders the ability to prove that unfair behaviour cannot impact specific classes of individuals. Consequently, recent papers have advocated for the adoption of formal *fairness verification* techniques to prove the absence of discrimination [150, 91, 166, 87, 94]. However, these primarily focus on deep neural networks. The sole notable fairness verification approach designed for tree-based classifiers employs abstract interpretation to verify *local fairness* properties [144]. Unfortunately, local fairness is now recognised as a relatively weak property predicated solely on

specific test instances, while *global fairness* considers all possible inputs of the classifier, making it more reliable for assessing actual fairness guarantees [94].

This work presents a new approach to the global fairness verification of tree-based classifiers. Given a tree-based classifier and a set of sensitive features that may lead to discrimination, our analysis synthesises sufficient conditions for fairness, expressed as a set $\mathcal{I}$ of traditional propositional logic formulas $\mathcal{I}$ of the form $I = \{x^{(1)} > 1 \wedge x^{(2)} \leq 5\}$ each predicating over a subset of the feature space, rather than just on a specific test set, thereby providing global fairness guarantees. Intuitively, the meaning of provided example of logic formula I is that every instance $\vec{x} \in \mathcal{X}$ with the values $x^{(1)} > 1$ and $x^{(2)} \leq 5$ is considered fair.

Our fairness verification approach is formally proven to be both *sound* and *complete*; fairness is certified for any instance satisfying some formula in I, and the formulas in I can characterise all instances where the classifier is fair. Moreover, our approach is *explainable*, as it is easily understandable by human experts based on traditional logic formulas. In particular, our approach seeks simple logical formulas, i.e., containing few conditions, that characterise a large part of the feature space. Finally, we empirically demonstrate that a small set of simple logic formulas is sufficient to characterise largely the fairness guarantees provided by the classifier in practice. This makes our approach particularly appealing for problems like algorithmic hiring, where automated decisions must be carefully audited [152].

10.8.1 Unfairness Scenario

To understand how the fairness problem defined in this work can adapt to adversarial machine learning and how the data-independent stability analysis defined in Section 10.4 can be exploited to verify where the model is fair, we need to introduce the scenario of unfairness used in this research.

Numerous definitions of fairness have been proposed in the literature, each presenting advantages and disadvantages [169]. These fairness definitions can be broadly classified into two main categories: *individual fairness*, which requires that similar inputs yield similar outputs, and *group fairness*, which requires that a particular group of inputs, considered as a whole, must be treated equally to other groups. Individual and group fairness are significant and valuable concepts typically investigated independently [118]. In this paper, we focus on a specific definition of individual fairness known as the lack of *causal discrimination*, as introduced in [71].

Lack of causal discrimination does not depend on the choice of a specific test set; rather, it predicates over all possible instances in (a subset of) the feature space $\mathcal{X}$. This is crucial in the fairness setting, as fairness is particularly relevant for minorities, so collecting representative data in the test set might be challenging. Indeed, the need for global fairness verification has been recently advocated for

neural networks [94].

In essence, the lack of causal discrimination implies that any two similar inputs should produce identical predictions from the classifier, thus encapsulating the concept of individual fairness. More precisely, lack of causal discrimination necessitates that the classifier produces the same prediction for any two instances differing solely in the values of a set of *sensitive* features $\mathcal{F}^s \subseteq \mathcal{F}$. For a given instance $\boldsymbol{x}$, we denote $\delta(\boldsymbol{x}, \mathcal{F}^s)$ as the set of instances differing from $\boldsymbol{x}$ only in a (potentially empty) subset of the sensitive features $\mathcal{F}^s$. For instance, if $\mathcal{F}^s$ comprises only two binary features, then $\delta(\boldsymbol{x}, \mathcal{F}^s)$ encompasses the four instances derived from $\boldsymbol{x}$ by setting each sensitive feature in $\mathcal{F}^s$ to either of its two possible values while keeping the other features constant. Formally, the lack of causal discrimination is defined as follows.

Definition 13 (Causal Discrimination)**.** Let $h : \mathcal{X} \to \mathcal{Y}$ be a classifier and $\mathcal{F}^s$ be a set of sensitive features. We say that h does not perform *causal discrimination* on $\mathcal{X}' \subseteq \mathcal{X}$ if and only if, for every instance $\boldsymbol{x} \in \mathcal{X}'$, we have that $\forall \boldsymbol{x}' \in \delta(\boldsymbol{x}, \mathcal{F}^s) : h(\boldsymbol{x}') = h(\boldsymbol{x})$.

To provide concrete examples of causal discrimination, consider a scenario where a classifier is employed to assess the approval of loan applications, with the set of sensitive features $\mathcal{F}^s$ specifically encompassing the customer's gender. Lack of causal discrimination on $\mathcal{X}$ requires that any two identical customers who differ solely in their gender must receive identical responses to their loan requests. By narrowing the focus to a specific subset $\mathcal{X}' \subseteq \mathcal{X}$, the fairness guarantees become *conditional*, and thus more practically useful. For instance, this could involve ensuring that any two identical customers with a monthly salary exceeding \$4,000 are guaranteed to receive the same response to their loan applications, irrespective of their gender.

10.8.2 Mapping With Adversarial Machine Learning

The concept of stability extends easily from the adversarial setting to the fairness domain, as the absence of causal discrimination requires that arbitrary perturbation to the sensitive features $\mathcal{F}^s$ should not impact the classifier's predictions. The definition of Causal Discrimination in Definition 13 states that: given a specific subset of the feature space $\mathcal{X}' \subseteq \mathcal{X}$, a model h, and two instances $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}'$, differing solely in the sensitive features in $\mathcal{F}^s$, i.e., $\boldsymbol{x}, \boldsymbol{x}' \in \delta(\boldsymbol{x}, \mathcal{F}^s)$, the model h does not exhibit causal discrimination on $\boldsymbol{x}$ and $\boldsymbol{x}'$ if $h(\boldsymbol{x}') = h(\boldsymbol{x})$.

To formalise the fairness problem in adversarial scenarios, let's consider the following attacker A, that can perturb every instance $\boldsymbol{x} \in \mathcal{X}'$ only on the features belonging to the set $\mathcal{F}^s$. The corresponding adversarial perturbation set of an

instance $\boldsymbol{x}$ is defined as follows:

$$A(\boldsymbol{x}) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathcal{X}' \wedge ||\boldsymbol{z} - \boldsymbol{x}||_0 \leq |\mathcal{F}^s| \wedge \boldsymbol{x}_{lb} \preceq \boldsymbol{z} \preceq \boldsymbol{x}_{ub}\} \tag{10.4}$$

with $x_{lb}^{(f)} = -\infty$ and $x_{ub}^{(f)} = +\infty$ if $f \in \mathcal{F}^s$, zero otherwise.

Intuitively, by definition of this adversarial perturbation set, $A(\boldsymbol{x}) = \delta(\boldsymbol{x}, \mathcal{F}^s)$. Consequently, both instances $\boldsymbol{x}$ and $\boldsymbol{x}'$ introduced in the context of fairness are contained in $A(\boldsymbol{x})$. This implies that if a model h is stable (Definition 5) on the instance $\boldsymbol{x}$ with respect to all evasion instances $\boldsymbol{x}' \in A(\boldsymbol{x})$, i.e., it does not change predictions for any possible perturbation of features in $\mathcal{F}^s$, then it does not exhibit causal discrimination on all instances in $\delta(\boldsymbol{x}, \mathcal{F}^s)$.

To verify if the model h is fair on a region of the feature space $\mathcal{X}' \subseteq \mathcal{X}$, it suffices to prove that h is stable on $\mathcal{X}'$ through a data-independent stability analysis.

10.8.3 Contribution: Global Fairness Verifier

Verifying the absence of causal discrimination is challenging due to the need for universal quantification over a set of instances, which may be drawn from a continuous and unbounded feature space. Previous approaches to causal discrimination have addressed this challenge by focusing on a finite feature space and assessing fairness by means of a testing approach [71]. This approach involves computing a *causal discrimination score*, representing the fraction of instances in the feature space experiencing causal discrimination. However, this measure is meaningful only when the feature space is finite and instances can be exhaustively enumerated. Despite attempts to address this limitation using binning to discretise the feature space, the testing approach in [71] lacks exhaustiveness for scalability reasons. Consequently, it can only identify counterexamples suffering from causal discrimination but cannot conclusively prove its absence. Similar critiques apply to other recent proposals on fairness testing [1, 165, 17].

In this work, we advance the current state of the art by introducing a novel verification technique to formally verify the fairness guarantees of tree-based models. Specifically, our technique enables the automatic identification of subsets of the feature space where the absence of causal discrimination is assured, moving beyond the identification of mere counterexamples. Furthermore, our technique provides simple logical formulas that encapsulate large portions of the feature space, guaranteeing fairness. A few simple logic formulas favour the explainability of the model, as they enable a human expert to understand why the model is fair concerning different inputs.

The contribution of this work is twofold. First, we provided an approach to identify areas in the feature space that ensure the lack of causal discrimination as defined in Definition 13. Second, we extend our approach to provide short (i.e.,

simple) logical formulas that cover areas of feature space, ensuring that the model provides a lack of causal discrimination for all instances in those areas.

The first contribution is directly related to the work presented in this chapter, as it is closely tied to the data-independent stability analysis defined in Section 10.4. The second part falls outside the scope of this thesis. Therefore, we only briefly introduce the idea behind this work and its results. Additional details can be found in the original work [30].

10.8.3.1 Verification Algorithm

As mentioned above, lack of causal discrimination predicates over a potentially unbounded set of instances $\mathcal{X}' \subseteq \mathcal{X}$; hence, traditional approaches to stability/fairness verification, such as [146], cannot be directly applied to verify the lack of causal discrimination on $\mathcal{X}'$.

Given a tree-based ensemble F, as discussed in Section 10.8.3, to identify subsets of the feature space where F ensures the lack of causal discrimination, it is sufficient to find an unbounded subset of the feature space $\mathcal{X}' \subseteq \mathcal{X}$ where F is guaranteed to be stable. For this reason, we decided to leverage the data-independent stability analysis defined in Section 10.4 to verify an unbounded subset of the feature space where the model is stable, i.e., it has a lack of causal discrimination.

Intuitively, the output of a data-independent stability analyser $E \cup C$ over-approximates the subset of the feature space where the classifier violates stability. Note that the analyser's output is a set of symbolic attacks $s \in E \cup C$. Each symbolic attack is composed of two hyper-rectangles: $s.pre$ defines a subset of the feature space where instances fall before an attack, and $s.post$ represents the area in which instances in $s.post$ may end up after an attack. Thus, for each $s \in E \cup C$, we can consider the hyper-rectangle $s.pre$ as a subset of the feature space where F is unstable under attack. Consequently, if $\boldsymbol{x} \in \mathcal{X}'$ suffers from causal discrimination, then the result of the analysis must include a symbolic attack $s \in E \cup C$ such that $\boldsymbol{x} \in s.pre$.

Let $\mathcal{U} = \{s.pre \mid s \in E \cup C\}$, to prove that F does not perform causal discrimination on $\mathcal{X}' \subseteq \mathcal{X}$, it suffices to show that there exists no $H \in \mathcal{U}$ such that $H \cap \mathcal{X}' \neq \emptyset$. Furthermore, it follows that the union of the hyper-rectangles in the pre-image of the symbolic attacks returned by the analysis, i.e., $\bigcup_{H \in \mathcal{U}} H$, over-approximates the set of counterexamples suffering from causal discrimination. Through the space $\bigcup_{H \in \mathcal{U}} H$, we can construct a global fairness verifier, in which every instance that falls within this space may be unfair, while every instance that falls outside this space is certainly fair concerning causal discrimination.

10.8.3.2 Synthesis Algorithm

The aim of this work, however, was not only to provide a global robustness verifier but also to characterise the subset of the feature space $\mathcal{X}' \subseteq \mathcal{X}$ where F does not exhibit causal discrimination on $\mathcal{X}'$. Specifically, the goal is to characterise large subsets $\mathcal{X}' \subseteq \mathcal{X}$ outside the unfair regions with short logic formulas easily interpretable by human experts. However, this aspect of the work goes beyond the main theme of this thesis, so we provide only an overview of the idea behind the Synthesis Algorithm that produces the logic formula.

Intuitively, this problem can be conservatively addressed by removing from $\mathcal{X}$ all the hyper-rectangles $H \in \mathcal{U}$ returned by the data-independent stability analysis, thus under-approximating the subset of the feature space where F is stable. This approach would be sound but computationally inefficient due to an exponential blowup concerning the dimensionality of the feature space when subtracting hyper-rectangles. Given two hyper-rectangles with d features, their subtraction might generate $O(d)$ hyper-rectangles in the general case, leading to $O(d^{|\mathcal{U}|})$ hyper-rectangles in the worst case at the end of the subtraction process. This limitation can be circumvented by avoiding the computation of subtraction and by directly reasoning in terms of instances falling out from the hyper-rectangles $\mathcal{U}$. However, this would make it hard to characterise $\mathcal{X}'$ in a human-understandable way, given both the number of hyper-rectangles and the potentially large dimensionality of the feature space.

We thus propose an iterative algorithm to gradually formulate increasingly complex conditions to ensure the absence of causal discrimination expressed through traditional logic formulas. In this way, the initial iterations of the algorithm can efficiently generate concise and easily understandable conditions for human experts, which are arguably the most beneficial for analysts. As more analysis time and computational resources become available, the algorithm can identify more complex conditions, enabling the detection of additional subsets of the feature space where the absence of causal discrimination is guaranteed. In essence, each iteration of the algorithm expands upon the sound approximation established in the previous iteration by incorporating more intricate sufficient conditions for fairness, continuing until the subset of the feature space is covered by these conditions or an early stopping criterion is met.

In detail, we provided an algorithm called the Synthesis Algorithm for creating the set $\mathcal{I}$ of logic formulas of the form $I = \{x^{(5)} > 0 \wedge x^{(5)} \leq 7 \wedge x^{(7)} > 3\}$ predicating over the subset of the feature space where the classifier is guaranteed to be fair. The example formula I has a length of 3 conditions. The goal of the Synthesis Algorithm is to find formulas of ever-increasing length to characterise the space $\mathcal{X} \backslash \mathcal{U}$, i.e., the subset of the feature space where the classifier is fair. Note that, for the sake of simplicity, when we relate $\mathcal{I}$ and $\mathcal{U}$ with $\mathcal{X}$, we consider $\mathcal{I}$ and

$\mathcal{U}$ as subsets of the feature space, and not as sets of formulas and hyper-rectangles, respectively, representing subsets of the feature space.

The algorithm we designed takes the set of hyper-rectangles $\mathcal{U}$ returned by the data-independent stability analyser as input and, iteration after iteration, creates formulas I that become progressively more complex in order to refine the representation of $\mathcal{X} \setminus \mathcal{U}$. At each iteration i, the algorithm generates formulas I with a length (complexity) of at most i conditions. At the end of the algorithm's execution, the set $\mathcal{I} = \mathcal{X} \setminus \mathcal{U}$. However, the algorithm allows an early termination that generates a set $\mathcal{I} \subseteq \mathcal{X} \setminus \mathcal{U}$. The details of how the Synthesis Algorithm creates the Is of increasing complexity at each iteration are left to the reader, who can find the technical detail in Section III.C: *Synthesis Algorithm* in [30].

10.8.4 Main Results

In this section, we briefly summarise the experiments and main results of the work covered in this section. Our assessment involves decision tree ensembles trained on three public datasets frequently employed in fairness literature [144]. Each dataset corresponds to a binary classification task with relevance to fairness: ADULT[3] [8] requires predicting yearly income (above or below $50,000), STATLOG (GERMAN CREDIT DATA)[4] GERMAN [82] assigning credit scores (good or bad), and HERITAGE HEALTH[5] HEALTH [74] requires predicting ten-year mortality (above or below the median Charlson index). All three datasets can be used to train classifiers deployed to assess loan requests or health insurance scenarios.

The attribute *sex* is chosen as the binary-sensitive feature to ensure fairness and prevent discrimination based on gender in all cases. While we focus on a single sensitive feature for simplicity, it's important to note that our approach accommodates an arbitrary number of sensitive features.

Datasets are split into a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$ using an 80%-20% scheme with stratified random sampling. Standard Random Forest models are trained on $\mathcal{D}_{train}$ using the open-source Scikit-learn library [24]. Subsequent experiments are conducted on $\mathcal{D}_{test}$ and a synthetic dataset $\mathcal{D}_{rand}$ comprising 100,000 random instances. Finally, the Synthesis Algorithm is implemented in the C++ programming language and the code is available on GitHub[6].

The experiments aim to address three fundamental research questions: *i)* Do the synthesised fairness conditions *precisely* cover the portions of the feature space where a lack of causal discrimination is guaranteed? *ii)* Can we express the fairness guarantees of classifiers using a small number of conditions of limited complexity

[3]https://archive.ics.uci.edu/dataset/2/adult

[4]https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data

[5]https://www.kaggle.com/competitions/hhp/data

[6]https://github.com/LorenzoCazzaro/explainable-global-fairness-verification

to enhance *explainability*? *iii)* what is the *performance* in terms of exectuin time of the Synthesis Algorithm?

10.8.5 Precision of the Synthesis Algorithm

In our initial experiment, we assessed the precision of our verification approach by comparing the set of hyper-rectangles in $\mathcal{U}$ obtained from the data-independent stability analysis with the set of formulas in $\mathcal{I}$ generated by the Synthesis Algorithm. According to the definition of the Synthesis Algorithm, the set $\mathcal{I}$ identifies the subset of the feature space that is disjoint from all hyper-rectangles in $\mathcal{U}$. If the Synthesis Algorithm runs to completion, it implies $\mathcal{I} = \mathcal{X} \setminus \mathcal{U}$. However, in practice, the Synthesis Algorithm is often subject to early termination due to the exponential growth in the number of candidates to be analysed at different iterations and the increasing complexity of the synthesised formulas. Therefore, we estimate the precision of the Synthesis Algorithm when stopped after 6 iterations, providing formulas with a small number of terms that are understandable to humans.

To verify the precision of the Synthesis Algorithm, we compared the causal discrimination score $\mathrm{D}^{\mathcal{U}}$ on the datasets $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$ based on the set of hyper-rectangles $\mathcal{U}$ with the causal discrimination score $\mathrm{D}^{\mathcal{I}}$ based on the set of formulas $\mathcal{I}$ (i.e., the ratio of instances in the set that belong to some $H \in \mathcal{U}$ and the ratio of instances in the set that do not belong to any $I \in \mathcal{I}$). Our results indicated that $\mathrm{D}^{\mathcal{U}}$ and $\mathrm{D}^{\mathcal{I}}$ coincided in the vast majority of cases, signifying that the formulas in $\mathcal{I}$ accurately characterised the subset of the feature space that is disjoint from all hyper-rectangles in $\mathcal{U}$, even when enforcing early stopping after 6 iterations. The only case where we observed $\mathrm{D}^{\mathcal{I}}$ not reaching $\mathrm{D}^{\mathcal{U}}$ is after 6 iterations on the German dataset. However, we proved the two scores align when the number of iterations slightly increases.

10.8.6 Explainability of the Results

To assess the explainability of the formulas $\mathcal{I}$, we adopt the methodology utilised in previous works focusing on the extraction of explainable information from classifiers through logic formulas [97, 57, 123]. In these approaches, the complexity of logic formulas is measured by their length (number of atoms) and shorter logic formulas are preferred for their clearer interpretability. Additionally, a small number of formulas is considered crucial for enhancing explicability.

To measure the capability of our algorithm to provide concise formulas characterising large areas of the feature space where the model ensures fairness, we computed the percentage of instances in $\mathcal{D}_{train}$ covered by each formula. The more instances a formula covers, the more expressive it is in proving fairness based on

the training data. To find a limited number of short formulas for an explainable characterisation of where the classifier is fair, we employed a greedy strategy for selecting the best $I \in \mathcal{I}$ returned by the Synthesis Algorithm.

It is important to note that at each iteration i of the Synthesis Algorithm, it returns formulas of length at most i. Hence, we need to focus on the first algorithm iterations to have shot formulas. However, the union of the formulas found in the very early iterations is not guaranteed to cover a significant portion of the instances in $\mathcal{D}_{train}$. Consequently, we need to select the minimum number of formulas in $\mathcal{I}$ computed in the first k iterations that maximise coverage on $\mathcal{D}_{train}$. Using the following greedy strategy, we identified the set of most important formulas $\mathcal{I}^k_{best}$ that optimise our problem. First, we ordered the formulas in $\mathcal{I}$ in ascending order of length. Subsequently, we selected the most important formula in terms of the number of covered instances in $\mathcal{D}_{train}$. Finally, we removed the covered instances from $\mathcal{D}_{train}$ before selecting the second most important formula, and so on, until k formulas were selected. The order based on the length of the formulas ensures that at each iteration of the greedy algorithm, the shortest formula that covers the highest number of instances is selected.

To evaluate the quality of the set $\mathcal{I}^k_{best}$ created with respect to $\mathcal{D}_{train}$, we assessed how many instances in $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$ are covered by $\mathcal{I}^k_{best}$. The results of this experiment demonstrated that with $k = 6$, the Synthesis Algorithm for the Adult and Health datasets requires only the top 10 formulas to cover approximately 90% of the instances in $\mathcal{D}_{test}$. Meanwhile, for the German dataset, the top 20 formulas are sufficient to guarantee fairness for 80% of the instances in $\mathcal{D}_{test}$. This indicates that a small number of formulas effectively characterise fairness guarantees on $\mathcal{D}_{test}$. Furthermore, we observed that more formulas are needed to cover synthetic instances in $\mathcal{D}_{rand}$. Specifically, the top 20 formulas cover only around 30% of the instances in Adult and approximately 60% of the instances in German.

The difference in results between $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$ can be attributed to the fact that the conditions in the formulas of $\mathcal{I}^k_{best}$ depend on the thresholds learned from $\mathcal{D}_{train}$. Consequently, the top conditions exhibit better generalisation on a $\mathcal{D}_{test}$ dataset that shares the same data distribution as $\mathcal{D}_{train}$ than on a set of randomly generated instances like $\mathcal{D}_{rand}$.

10.8.7 Performance Evaluation

To evaluate the performance of the Synthesis Algorithm in terms of execution time, the time required by the data-independent stability analyser as defined in Section 10.4 to compute the hype-rectangle in $\mathcal{U}$.

We analysed the performance, focussing on ensembles comprising a maximum of 13 trees with a maximum height of 6. The results indicate that all models can be

analysed within a few minutes, with the analysis taking approximately 30 minutes on the ADULT dataset, 5 minutes on the GERMAN dataset, and only 13 seconds on the HEALTH dataset, when six iterations of the algorithm are performed. We discovered an exponential increase in execution time as the number of iterations and the complexity of the models increased. The model's complexity significantly impacts the execution time of the Synthesis Algorithm since it impacts the number of features and thresholds. More features and thresholds provide more symbolic attacks in output from the data-independent stability analyser. Consequently, since the Apriori algorithm inspires the Synthesis Algorithm, the execution time exponentially grows with the number of items to be processed. This underscores the advantages of the iterative characteristic of our approach, which allows for an early stopping criterion to gather partial yet empirically precise results, useful where achieving full convergence may be computationally prohibitive.

10.9 Summary

Following, we summarise the contributions of the research presented in this chapter, titled "Beyond Robustness".

- **Shortcomings of Robustness:** In this work, we have argued and empirically demonstrated that robustness metrics for evaluating the security of machine learning models in adversarial scenarios can provide a false sense of security. Robustness measures the security of the model based on the instances of a given test set $\mathcal{D}_{\text{test}}$, yet it provides no guarantees regarding the robustness with respect to instances sampled in a very small neighbourhood of the instances in $\mathcal{D}_{\text{test}}$. In fact, we showed that random sampling in the neighbourhoods of the instances in $\mathcal{D}_{\text{test}}$ leads to robustness values significantly different from those estimated on the original test set; hence, proving that robustness is not a trustworthy metric.

- **Contribution:** To overcome the limitations of Robustness, we introduced a new metric called Resilience. An instance $\boldsymbol{x}$ is resilient if all instances in a small neighbourhood of $\boldsymbol{x}$ are robust. While Resilience is a much-improved metric for assessing the security of machine learning models, it follows that there is an infinite number of instances in a neighbourhood of $\boldsymbol{x}$; consequently, verifying robustness for each of them is not feasible. For this reason, we proposed a data-independent stability analysis to identify regions in the feature space where the model is stable, meaning its prediction remains unchanged under attack. Consequently, all instances correctly predicted by the model falling within one of these stable areas are both robust and resilient.

- **Main Results:** In the experimental phase, we empirically demonstrated that robustness can provide a false sense of security, even in real-world scenarios. Additionally, we established that our under-approximated resilience, calculated using our data-independent stability analyser, is precise. Our empirical findings indicate that potentially significant disparities between robustness and estimated resilience occur because the estimated resilience closely aligns with the robustness measured over the "most unlucky" sampling performed in a small neighbourhood of the original test set. This confirms that our resilience estimates effectively capture evasion attacks against plausible samplings from the same data distribution used to construct the original test set. Finally, we showed the feasibility of resilience verification in practice, especially for relatively small models and simple datasets. For larger models, we demonstrated that the iterative refinement process supported by our analysis technique can be employed to obtain useful under-approximations of resilience even before analysis convergence.

- **Data-Independent Stability Analysis for Fairness:** We demonstrated the versatility of our data-independent stability analysis by extending its application beyond adversarial machine learning to encompass domains such as fairness. Specifically, we elucidated how model fairness, defined in terms of *lack of causal discrimination*, can be construed as model stability in adversarial scenarios. A machine learning model is deemed free from causal discrimination if, given two instances that differ solely in the sensitive attributes, it predicts the same label for both. In an adversarial scenario, the model achieves stability (i.e., fairness in the original problem) on a given instance when it consistently produces the same prediction despite adversarial perturbations applied to the sensitive attributes. In other words, any two instances differing only in the sensitive attributes receive identical predictions. Our analyser enables the computation of areas in the feature space where the model is free from causal discrimination (i.e., is stable).

10.9.1 Future Work

Below, we provide potential extensions and future research from this work.

- **Application to Diverse Machine Learning Models:** Extending the application of the resilience metric and the stability analyzer to a broader range of machine learning models beyond decision trees and ensembles is another avenue for future research. Investigating how well these concepts generalise to diverse model architectures, such as neural networks or support vector machines, would contribute to a more comprehensive understanding of model robustness in different contexts.

- **Scalability and Efficiency Improvements:** As the data-independent stability analyser forms a crucial component of the resilience estimation process, there is room for research aimed at enhancing its scalability and efficiency. Since the soundness proofs of our analysis abstract from several implementation details, e.g., the splitting criterion for symbolic attacks, different heuristics may be tried out to handle larger models and datasets without compromising the accuracy of resilience estimates.

- **Incorporating Domain-Specific Knowledge:** The threat model used in this work allows the attacker to perturb any combination of a number of features. This is not always allowed in real scenarios and makes the analysis very expensive. It would be interesting to explore the utility of the resilience and analyzer when integrating domain-specific knowledge into the threat model to into account the application domain, potential attack scenarios, or specific characteristics of the input data.

Chapter 11

Discussion Second Part

Part II focused on the research studies pursued during my doctorate with respect to the domain of adversarial machine learning. Specifically, the research has unfolded in two different yet interconnected areas. The first relates to the design of learning algorithms robust to evasion attacks, a common and easily executable type of attack. In this concern, we have concentrated on binary classification problems and learning algorithms based on ensembles of decision trees. The second area of research we covered pertains to the verification and certification of model security, expressed in terms of the robustness of other metrics. This involves designing certificates and verifiers to assess the security of models and proposing a new metric to assess the security of machine learning models.

These research areas seek to make machine learning models data-aware in different ways. The first involves designing learning algorithms to train models robust to maliciously crafted inputs, carefully designed by an attacker, thus pursuing the definition of data-aware algorithms. The second focuses on proposing strategies to analyse a given model to understand when it is secure against attacks and where malicious inputs can compromise it.

In Chapter 9, we discussed Feature-Partitioned Forest (FPF) [34], a learning algorithm to train robust ensembles of decision trees. During the training phase, the FPF algorithm considers the presence of a possible attacker in the operational phase and creates forests of decision trees robust by construction. The robustness of these forests is derived from a robust partitioning of the feature space among the trees in the forest, which ensures, given an attacker with an adversary's model constrained by L_0-norm and a fixed number of features b, that such attacks cannot compromise the majority of trees in a forest if the forest has a size greater than or equal to $2b + 1$. This robustness by construction comes at the expense of the accuracy of the ensemble in the absence of an attack, as individual trees are not allowed to exploit all the features available in the dataset. However, this special structure of ensembles has allowed the design of two robustness certifiers, *Fast*

Robustness Lower-Bound (FLB) and *Exhaustive Robustness Lower-Bound* (ELB), which efficiently and accurately calculate a lower-bound of the model's robustness [34].

In Chapter 10 instead, we defined a new metric called Resilience [31] for verifying the security of machine learning models. We introduced this new metric because we demonstrated how the well-known Robustness metric, widely used in adversarial machine learning to quantify models' security, can provide a false sense of security. The motivation behind this discovery is that robustness quantifies the model's security based only on a fixed set of instances $\mathcal{D}$. However, it says nothing about all possible sets $\mathcal{D}'$ containing instances very close to those in $\mathcal{D}$. Resilience, on the other hand, ensures that every instance $\boldsymbol{x} \in \mathcal{D}$ considered robust, has a fixed size neighbourhood $N(\boldsymbol{x})$ in which all instances $\boldsymbol{x}' \in N(\boldsymbol{x})$ are also robust. Through resilience, it is possible to estimate better the ability of machine learning models to resist malicious inputs. In the same work, we also defined a data-independent stability analysis to understand where the model is not stable. The analysis returns a set of symbolic attacks, a structure composed of two hyper-rectangles, the *pre-image* and the *post-image*, which identify regions in the feature space where instances fall before and after the attack, respectively. Consequently, the regions of the feature space contained in the hyper-rectangles in the pre-images indicate where the model cannot guarantee stability under attack, i.e., it may change predictions on instances falling within those hyper-rectangles. The information obtained from the model's analysis allows us to calculate resilience and characterise the feature space where the attacker is harmless. Additionally, we showed that this analysis can be exploited to compute other adversarial machine learning metrics, such as robustness (i.e., Section 10.6.2), or to calculate metrics beyond this domain, such as fairness in terms of causal discrimination (i.e., Section 10.8).

To conclude this last chapter related to the research part dedicated to the domain of adversarial machine learning, we want to pose an interesting research question that opens up possible future works: Is it possible to leverage the efficiency of the FLB and ELB certificates and the hyper-rectangles created by the data-independent stability analyser for creating effective strong evasion attacks to exploit for adversarial training?

Adversarial training is the practice of using instances perturbed by an attacker during the training phase. The goal is to enable the learning algorithm to explore regions of the feature space that are unlikely to be (or not) covered by the distribution of legitimate input data and learn to classify attacked instances correctly. A fundamental factor determining the effectiveness of adversarial training approaches is the quality of the attacks used during the training phase. Attacks that do not explore unlikely regions of the feature space or are too weak may re-

sult in ineffective models under attack. The research on high-quality adversarial attacks is a branch of study in adversarial machine learning.

Let us draw some observations from the research works introduced in this part of the thesis. In the case of models trained with FPF, the evasion instances that effectively change the model's prediction do not do so by compromising the majority of the forest, as this is not possible by construction. Instead, they achieve this by attacking the high-quality features used by the model to discriminate between instances. The weakness of FPF forests is not due to the attacker but to the lack of high-quality features to distribute among all the trees in the forest (Section 9.3.3). The lack of high-quality features leads to the creation of inaccurate trees. Consequently, the attacker exploits the errors of less accurate trees to its advantage. It targets the more accurate trees, i.e., attacks the high-quality features to obtain an incorrect prediction in the majority of the forest. FLB and ELB take this attacker's behaviour into account in their algorithm. Consequently, the evasion attacks considered effective by the certifiers are attacks performed on accurate trees and, therefore, on high-quality features. Thus, it is possible to leverage FLB and ELB to create sets of effective evasion attacks by sampling instances from the feature space and efficiently certifying them with the certifiers. If the certifiers say the instance is not attackable, it is discarded, and the next one is generated. The effective evasion attacks can be used to train robust machine learning models with adversarial training strategies, even those not based on decision trees.

The possibility of creating effective evasion attacks is even more immediate in the work defined in Chapter 10. Specifically, it is possible to leverage the hyper-rectangles in the pre-images of symbolic attacks produced by the analyser to sample evasion instances. Indeed, the hyper-rectangles represent a subset of the feature space where the model may be unstable, and being this space continuous, it is possible to sample infinite evasion instances. Furthermore, the ensemble analysis is done only once, but once the output of the data-independent stability analyser is obtained, it is possible to sample large amounts of data efficiently. Additionally, there is a substantial difference between the attacks generated by the hyper-rectangles of the analyser and those generated with the FLB and ELB certificates. Attacks generated using FLB and ELB certifiers are created on a specific class of models, i.e., those created by the FPF algorithm. On the other hand, the hyper-rectangles in the pre-images can be derived from different models, provided they are based on decision trees and use majority voting for prediction.

The two works introduced in this part of the thesis can be exploited together or independently to create data-aware models in adversarial machine learning scenarios through adversarial training approaches.

Chapter 12

Conclusion

The research covered in this thesis focused on making machine learning models more effective, efficient and robust. In particular, we focussed on the effectiveness and efficiency of learning to rank algorithms and the robustness of binary classification algorithms in adversarial scenarios. To achieve this goal, we focused on designing data-aware learning algorithms, i.e., aware of harmful inputs that may exist during both the training and operational phases. A harmful input encompasses everything that compromises the model's quality, such as noise, errors, or outliers within the training set or malicious instances crafted by an attacker to elicit unexpected behaviours from the model.

In our study on ranking models, we found that certain types of documents within the training set can have a negative effect on the models' effectiveness. In particular, in the work titled "Filtering out Outliers in Learning to Rank" [121], published at *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval*, we provided the definition of *consistent positive/negative outlier* documents. These types of documents are consistently misranked during the model training phase. As a consequence, the continuous presence of consistent outliers during the learning phase compromises the final models' effectiveness. To deal with such detrimental documents, we designed *Surrender on Outliers and Rank* (SOUR), an algorithm to identify and remove consistent outliers from the training set. Through extensive evaluations, we demonstrated that the cleaner training sets provided by SOUR enhance the models' effectiveness.

Furthermore, we focused on discarding redundant or superfluous non-relevant documents within the training set. In the work titled "On the Effect of Low-Ranked Documents: A New Sampling Function for Selective Gradient Boosting" [110], published at *SAC '23: The 2023 ACM SIGAPP Symposium on Applied Computing*, we discovered that in the training set, there might be documents that are superfluous which not only are they useless to the learning process, but they can even compromise its quality and efficiency. For this reason, we

designed HIGH_LOW_SAMPL a selection function for the *Selective Gradient Boosting* (SelGB) framework [115]. This selection strategy selects from the training set all the relevant documents, the most-informative non-relevant documents, i.e., the non-relevant documents ranked highest in the ranking, and the less-informative non-relevant documents, i.e., the non-relevant documents ranked lowest in the ranking. Thus, this selection function exploits the informativeness of the lowest-ranked non-relevant documents originally disregarded by SelGB. Leveraging on the lower-ranked documents enhances the final models' effectiveness compared to SelGB and reduces the training time due to fewer documents being processed.

The last work discussed in this thesis regarding the domain of learning to rank is "LambdaRank Gradients are Incoherent" [122], published at *CIKM '23: The 2023 ACM International Conference on Information and Knowledge Management*. In our study, we discovered a notable issue with LambdaRank and its derivatives, such as LambdaMART and the metric-driven loss function defined by Wang *et al.* in [173]. We showed that in such algorithms, the gradients are incoherent with respect to the learned ranking and relevance of the documents. Specifically, we observed that, during the learning phase, a mis-ranked document with high relevance could be pushed down in rankings more significantly than a document with lower relevance. This suggests that the learning algorithm failed to learn how to prioritise the most relevant documents. We demonstrated that the occurrence of this phenomenon is more frequent when optimising truncated metric rather than un-truncated metric. Through an in-depth analysis, we discovered that this exacerbation occurs due to fewer pairwise document comparisons implied by the truncated metric optimisation. Despite being truncated optimisation more efficient in terms of training time than un-truncated optimisation, it provides less effective models. In light of all this, we designed Lambda-eX, an algorithm that thought three strategies to perform the most useful pairwise document comparisons to enhance model effectiveness without compromising the training efficiency. Through empirical evaluation, we demonstrated that Lambda-eX provides effective models as un-truncated metric optimisation while maintaining training efficiency as truncated matric optimisation.

In our work on binary classification models, we focused on the creation of data-aware ensembles of decision trees that are robust to attacks from a malicious entity. In the work titled "Feature Partitioning for Robust Tree Ensembles and Their Certification in Adversarial Scenarios" [34], published at *EURASIP Journal on Information Security, 2021*, we designed a robust learning algorithm named Feature-Partitioned Forest (FPF) to train ensembles of decision trees robust by construction to evasion attacks. FPF ensembles are robust to adversary's models constrained by L_0-norm and a budget b. The key concept behind the robustness by contraction of FPF models relies on training each tree within the ensemble

on a distinct partition of the feature space. This strategic robust partitioning is designed to ensure that the impact of an evasion attack is limited to less than half of the ensemble. The resulting ensemble's structure allowed us to develop two accurate and efficient robustness certification algorithms named *Fast Robustness Lower-Bound* (FLB) and *Exhaustive Robustness Lower-Bound* (ELB). Through extensive experimental analyses, we demonstrated that FPF provides ensembles more robust to evasion attacks generated by the provided adversary's model than the competitors. Moreover, we demonstrated that both FLB and ELB provide accurate estimates of the exact robustness of the models.

Additionally, in the work titled "Beyond Robustness: Resilience Verification of Tree-Based Classifiers" [31], published at *Computers & Security, 2022*, we demonstrated that the widely used robustness metric, commonly employed to quantify the security of classification models, may provide a false sense of security. The motivation behind this undesired characteristic of robustness lies in its local estimation of the security of the models, which only depends on instances in a given finite test set $\mathcal{D}$. To overcome this deficiency, we designed the Resilience metric. The resilience ensures that for every instance $\boldsymbol{x} \in \mathcal{D}$ is considered robust if and only if given a neighbourhood $N(\boldsymbol{x})$, every instance $\boldsymbol{x'} \in N(\boldsymbol{x})$ are also robust. In order to estimate the resilience, we defined a data-independent stability analysis to extract regions of the feature space where machine learning models are potentially affected by malicious inputs. In this work, we empirically demonstrated that robustness does provide a false sense of security and that resilience can be effectively estimated for small tree-based ensembles. Notably, we demonstrated that the data-independent stability analysis provided in this work can be exploited in other research areas different from adversarial machine learning, such as fairness. In fact, in "Explainable Global Fairness Verification of Tree-Based Classifiers" [30], published at *SaTML '23: The 2023 IEEE Conference on Secure and Trustworthy Machine Learning*, we built upon this work to design a Synthesis Algorithm to characterise regions of the features space where models provide lack of causal discrimination, with few short traditional logic formulas explainable to human experts.

The research works presented in this thesis represent an important step towards creating data-aware models. A natural extension of this work is to enhance the data-awareness of learning to rank models in an adversarial setting, thus bridging the two distinct parts covered in this work. This research branch is particularly crucial and, as of yet, has not been extensively explored. Indeed, adversarial scenarios are possible and may be relevant for different ranking systems. It is noteworthy that the attack paradigm differs from that addressed in this thesis, as our focus was primarily on classification tasks. In the context of ranking, the model's quality depends on its ability to rank the most important elements for a

query at the top of the list. Therefore, attacks designed for classification contexts are not directly applicable to ranking scenarios.

A classic example of an attack in the ranking context involves modifying the attributes of an item to make it seem more relevant to a query or a specific type of query. This can lead to situations like aggressive marketing, promoting fake news, or bypassing spam filters. There are also availability violation attacks, where an attacker attempts to fill the top-ranking positions to hide relevant items and push them to lower positions. This can lead to fewer user clicks to relevant items, which can be detrimental to e-commerce websites or other online platforms. In the context of online learning to rank based on click models, an attacker might intentionally click on non-relevant items to force the model to learn incorrect rankings, thereby reducing its effectiveness [175]. These types of attacks highlight the importance of developing data-aware models that are robust to adversarial scenarios.

It's important to note that reasoning in terms of adversarial machine learning on learning to rank models is not straightforward. This is because learning to rank is only applied in the final part of the pipeline of a ranking system, i.e., in the re-ranking stage. It's implausible to assume that an attack occurs directly by perturbing the dense representation of query-document pairs. Therefore, the attack must occur at the beginning of the pipeline on raw data of the items. For example, an attack might target the text of web pages in the case of a web search engine or the descriptions of items' attributes in the case of e-commerce. As a result, the attack must traverse the entire ranking pipeline before reaching the re-ranking stage, where learning-to-rank strategies are generally employed. This makes it particularly challenging to develop effective and robust data-aware models for learning to rank in adversarial scenarios.

Several attempts have been made in the direction of enhancing and assessing the robustness of learning to rank models in adversarial scenarios. For example, Goren *et al.* in [77] provided formal and empirical analyses of the robustness of ranking models based on learning to rank with respect to document perturbations. Similarly, Yu *et al.* in [185] conducted an in-depth study on adversarial attacks in the domain of learning to rank, specifically analysing various types of Generative Adversarial Networks (GANs). Despite these efforts, research on making learning to rank models robust to adversarial attacks is still in its early stages.

While there are several research works in information retrieval dedicated to stopping malicious inputs, such as email spam recognition through word-based filters, the advent of machine learning has led to a paradigm shift. These diverse strategies are not directly applicable in the context of adversarial machine learning, highlighting the need for additional exploration and development in this field. As machine learning models become increasingly prevalent in information retrieval,

it's crucial to ensure that they are robust and secure in the face of potential adversarial attacks. Furthermore, nowadays, research on adversarial scenarios in ranking has mostly been devoted to the security of ranking systems leveraging deep learning [108, 105, 174]. However, significantly less attention has been given to learning to rank learning algorithms, which are still widely used in the industry.

Bibliography

[1] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. "Black box fairness testing of machine learning models". In: *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*. Ed. by Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo. ACM, 2019, pp. 625–635. DOI: 10.1145/3338906.3338937.

[2] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. "Learning Optimal and Fair Decision Trees for Non-Discriminative Decision-Making". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1418–1426. DOI: 10.1609/aaai.v33i01.33011418.

[3] *American Federation of Information Processing Societies: Proceedings of the AFIPS '68 Fall Joint Computer Conference, December 9-11, 1968, San Francisco, California, USA - Part I*. Vol. 33. AFIPS Conference Proceedings. AFIPS / ACM / Thomson Book Company, Washington D.C., 1968. ISBN: 978-1-4503-7899-4. DOI: 10.1145/1476589.

[4] Maksym Andriushchenko and Matthias Hein. "Provably robust boosted decision stumps and trees against adversarial attacks". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 12997–13008.

[5] Javed A. Aslam, Evangelos Kanoulas, Virgiliu Pavlu, Stefan Savev, and Emine Yilmaz. "Document selection methodologies for efficient and effective learning-to-rank". In: *Proceedings of the 32nd Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009. Ed. by James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel. ACM, 2009, pp. 468–475. DOI: 10.1145/1571941.1572022.

[6] 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 2006, Taipei, Taiwan, March 21-24, 2006*. Ed. by Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Paul Lin, Shiuhpyng Shieh, and Sushil Jajodia. ACM, 2006, pp. 16–25. DOI: 10.1145/1128817.1128824.

[7] Martyna Bator. *Dataset for Sensorless Drive Diagnosis*. Feb. 2015. DOI: 10.24432/C5VP5F.

[8] Barry G. Becker and Ronny Kohavi. *Adult*. Apr. 1996. DOI: 10.24432/C5XW20.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. Ed. by Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, 2009, pp. 41–48. DOI: 10.1145/1553374.1553380.

[10] Kristin P. Bennett and O. L. Mangasarian. *Robust linear programming discrimination of two linearly inseparable sets*. 1992. DOI: 10.1080/10556789208805504.

[11] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion Attacks against Machine Learning at Test Time". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*. Ed. by Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný. Vol. 8190. Lecture Notes in Computer Science. Springer, 2013, pp. 387–402. DOI: 10.1007/978-3-642-40994-3_25.

[12] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin I. P. Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. "Security Evaluation of Support Vector Machines in Adversarial Environments". In: *Support Vector Machines Applications*. Ed. by Yunqian Ma and Guodong Guo. Cham: Springer International Publishing, 2014, pp. 105–153. ISBN: 978-3-319-02300-7. DOI: 10.1007/978-3-319-02300-7_4.

[13] Battista Biggio, Giorgio Fumera, and Fabio Roli. "Multiple classifier systems for robust classifier design in adversarial environments". In: *International Journal of Machine Learning and Cybernetics* 1.1-4 (2010), pp. 27–41. DOI: 10.1007/S13042-010-0007-7.

[14] Battista Biggio, Giorgio Fumera, and Fabio Roli. "Security Evaluation of Pattern Classifiers under Attack". In: *IEEE Transactions on Knowledge and Data Engineering* 26.4 (2014), pp. 984–996. DOI: 10.1109/TKDE.2013.57.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Support Vector Machines Under Adversarial Label Noise". In: *Proceedings of the 3rd Asian Conference on Machine Learning, ACML 2011, Taoyuan, Taiwan, November 13-15, 2011*. Ed. by Chun-Nan Hsu and Wee Sun Lee. Vol. 20. JMLR Proceedings. JMLR.org, 2011, pp. 97–112.

[16] Battista Biggio and Fabio Roli. "Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning". In: *Pattern Recognit.* 84 (2018), pp. 317–331. DOI: 10.1016/J.PATCOG.2018.07.023.

[17] Emily Black, Samuel Yeom, and Matt Fredrikson. "FlipTest: fairness testing via optimal transport". In: *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*. Ed. by Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna. ACM, 2020, pp. 111–121. DOI: 10.1145/3351095.3372845.

[18] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognit.* 30.7 (1997), pp. 1145–1159. DOI: 10.1016/S0031-3203(96)00142-2.

[19] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140. DOI: 10.1007/BF00058655.

[20] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[21] Sergey Brin and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". In: *Comput. Networks* 30.1-7 (1998), pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X.

[22] Sebastian Bruch. "An Alternative Cross Entropy Loss for Learning-to-Rank". In: *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. Ed. by Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia. ACM / IW3C2, 2021, pp. 118–126. DOI: 10.1145/3442381.3449794.

[23] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. "A Stochastic Treatment of Learning to Rank Scoring Functions". In: *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*. Ed. by James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang. ACM, 2020, pp. 61–69. DOI: 10.1145/3336191.3371844.

[24] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[25] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. "Learning to Rank with Nonsmooth Cost Functions". In: *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. Ed. by Bernhard Schölkopf, John C. Platt, and Thomas Hofmann. MIT Press, 2006, pp. 193–200.

[26] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. "Learning to rank using gradient descent". In: *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*. Ed. by Luc De Raedt and Stefan Wrobel. Vol. 119. ACM International Conference Proceeding Series. ACM, 2005, pp. 89–96. DOI: 10.1145/1102351.1102363.

[27] Christopher JC Burges. "From ranknet to lambdarank to lambdamart: An overview". In: *Learning* 11.23-581 (2010), p. 81.

[28] Francesco Busolin, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. "Learning Early Exit Strategies for Additive Ranking Ensembles". In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai. ACM, 2021, pp. 2217–2221. DOI: 10.1145/3404835.3463088.

[29] Qi-Zhi Cai, Chang Liu, and Dawn Song. "Curriculum Adversarial Training". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. Ed. by Jérôme Lang. ijcai.org, 2018, pp. 3740–3747. DOI: 10.24963/IJCAI.2018/520.

[30] Stefano Calzavara, Lorenzo Cazzaro, Claudio Lucchese, and Federico Marcuzzi. "Explainable Global Fairness Verification of Tree-Based Classifiers". In: *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. Los Alamitos, CA, USA: IEEE Computer Society, Feb. 2023, pp. 1–17. DOI: 10.1109/SaTML54575.2023.00011.

[31] Stefano Calzavara, Lorenzo Cazzaro, Claudio Lucchese, Federico Marcuzzi, and Salvatore Orlando. "Beyond robustness: Resilience verification of tree-based classifiers". In: *Comput. Secur.* 121 (2022), p. 102843. DOI: 10.1016/J.COSE.2022.102843.

[32] Stefano Calzavara, Lorenzo Cazzaro, Giulio Ermanno Pibiri, and Nicola Prezza. "Verifiable Learning for Robust Tree Ensembles". In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*. Ed. by Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda. ACM, 2023, pp. 1850–1864. DOI: 10.1145/3576915.3623100.

[33] Stefano Calzavara, Pietro Ferrara, and Claudio Lucchese. "Certifying Decision Trees Against Evasion Attacks by Program Analysis". In: *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II*. Ed. by Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider. Vol. 12309. Lecture Notes in Computer Science. Springer, 2020, pp. 421–438. DOI: 10.1007/978-3-030-59013-0_21.

[34] Stefano Calzavara, Claudio Lucchese, Federico Marcuzzi, and Salvatore Orlando. "Feature partitioning for robust tree ensembles and their certification in adversarial scenarios". In: *EURASIP J. Inf. Secur.* 2021.1 (2021), p. 12. DOI: 10.1186/S13635-021-00127-0.

[35] Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. "Adversarial Training of Gradient-Boosted Decision Trees". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2019, pp. 2429–2432. DOI: 10.1145/3357384.3358149.

[36] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. "Treant: training evasion-aware decision trees". In: *Data Min. Knowl. Discov.* 34.5 (2020), pp. 1390–1420. DOI: 10.1007/S10618-020-00694-9.

[37] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. "Learning to rank: from pairwise approach to listwise approach". In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007),*

Corvallis, Oregon, USA, June 20-24, 2007. Ed. by Zoubin Ghahramani. Vol. 227. ACM International Conference Proceeding Series. ACM, 2007, pp. 129–136. DOI: 10.1145/1273496.1273513.

[38] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. "The information visualizer, an information workspace". In: *Conference on Human Factors in Computing Systems, CHI 1991, New Orleans, LA, USA, April 27 - May 2, 1991, Proceedings*. Ed. by Scott P. Robertson, Gary M. Olson, and Judith S. Olson. ACM, 1991, pp. 181–186. DOI: 10.1145/108844.108874.

[39] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 39–57. DOI: 10.1109/SP.2017.49.

[40] Vitor R Carvalho, Jonathan L Elsas, William W Cohen, and Jaime G Carbonell. "A meta-learning approach for robust rank learning". In: *SIGIR 2008 workshop on learning to rank for information retrieval*. Vol. 1. 2008.

[41] Simon Caton and Christian Haas. "Fairness in Machine Learning: A Survey". In: *ACM Comput. Surv.* (Aug. 2023). ISSN: 0360-0300. DOI: 10.1145/3616865.

[42] Olivier Chapelle and Yi Chang. "Yahoo! Learning to Rank Challenge Overview". In: *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*. Ed. by Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Vol. 14. JMLR Proceedings. JMLR.org, 2011, pp. 1–24.

[43] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. "Large-scale validation and analysis of interleaved search evaluation". In: *ACM Transactions on Information and Systems* 30.1 (2012), 6:1–6:41. DOI: 10.1145/2094072.2094078.

[44] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. "Expected Reciprocal Rank for Graded Relevance". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*. Ed. by David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy Lin. ACM, 2009, pp. 621–630. DOI: 10.1145/1645953.1646033.

[45] Hongge Chen, Huan Zhang, Duane S. Boning, and Cho-Jui Hsieh. "Robust Decision Trees Against Adversarial Examples". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Rus-

lan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1122–1131.

[46] Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane S. Boning, and Cho-Jui Hsieh. "Robustness Verification of Tree-based Models". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 12317–12328.

[47] John Chen, Vatsal Shah, and Anastasios Kyrillidis. "Negative Sampling in Semi-Supervised learning". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1704–1714.

[48] Yan Chen, Wei Wang, and Xiangliang Zhang. "Randomizing SVM Against Adversarial Attacks Under Uncertainty". In: *Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III*. Ed. by Dinh Q. Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi. Vol. 10939. Lecture Notes in Computer Science. Springer, 2018, pp. 556–568. DOI: 10.1007/978-3-319-93040-4_44.

[49] Yizheng Chen, Shiqi Wang, Yue Qin, Xiaojing Liao, Suman Jana, and David A. Wagner. "Learning Security Classifiers with Verified Global Robustness Properties". In: *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. Ed. by Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi. ACM, 2021, pp. 477–494. DOI: 10.1145/3460120.3484776.

[50] Nancy Chinchor. "MUC-4 evaluation metrics". In: *Proceedings of the 4th Conference on Message Understanding, MUC 1992, McLean, Virginia, USA, June 16-18, 1992*. ACL, 1992, pp. 22–29. DOI: 10.3115/1072064.1072067.

[51] Microsoft Corporation. *LightGBM Release 3.3.3.99*. 2023.

[52] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018.

[53] David Cossock and Tong Zhang. "Subset Ranking Using Regression". In: *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*. Ed. by Gábor

Lugosi and Hans Ulrich Simon. Vol. 4005. Lecture Notes in Computer Science. Springer, 2006, pp. 605–619. DOI: 10.1007/11776420_44.

[54] Nilesh N. Dalvi, Pedro M. Domingos, Mausam, Sumit K. Sanghai, and Deepak Verma. "Adversarial Classification". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*. ACM, 2004, pp. 99–108. DOI: 10.1145/1014052.1014066.

[55] Hung Dang, Yue Huang, and Ee-Chien Chang. "Evading Classifiers by Morphing in the Dark". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, 2017, pp. 119–133. DOI: 10.1145/3133956.3133978.

[56] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. "Fast Ranking with Additive Ensembles of Oblivious and Non-Oblivious Regression Trees". In: *ACM Trans. Inf. Syst.* 35.2 (2016), 15:1–15:31. DOI: 10.1145/2987380.

[57] Houtao Deng. "Interpreting tree ensembles with inTrees". In: *Int. J. Data Sci. Anal.* 7.4 (2019), pp. 277–287. DOI: 10.1007/S41060-018-0144-8.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[59] Fernando Diaz, Bhaskar Mitra, Michael D. Ekstrand, Asia J. Biega, and Ben Carterette. "Evaluating Stochastic Rankings with Expected Exposure". In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. Ed. by Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux. ACM, 2020, pp. 275–284. DOI: 10.1145/3340531.3411962.

[60] Wenkui Ding, Xiubo Geng, and Xudong Zhang. "Learning to Rank from Noisy Data". In: *ACM Trans. Intell. Syst. Technol.* 7.1 (2015), 1:1–1:21. DOI: 10.1145/2576230.

[61] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. "Towards recency ranking in web search". In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu. ACM, 2010, pp. 11–20. DOI: 10.1145/1718487.1718490.

[62] Pinar Donmez, Krysta M. Svore, and Christopher J. C. Burges. "On the local optimality of LambdaRank". In: *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*. Ed. by James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel. ACM, 2009, pp. 460–467. DOI: 10.1145/1571941.1572021.

[63] Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Springer, 1993. ISBN: 978-1-4899-4541-9. DOI: 10.1007/978-1-4899-4541-9.

[64] Lei Feng, Senlin Shu, Zhuoyi Lin, Fengmao Lv, Li Li, and Bo An. "Can Cross Entropy Loss Be Robust to Label Noise?" In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 2206–2212. DOI: 10.24963/ijcai.2020/305.

[65] R.A. Fisher. *The design of experiments. 1935*. Edinburgh: Oliver and Boyd, 1935.

[66] M. Forina, R. Leardi, C. Armanino, and S. Lanteri. *PARVUS: An Extendable Package of Programs for Data Exploration*. Jan. 1998. ISBN: 0-444-43012-1.

[67] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM, 2015, pp. 1322–1333. DOI: 10.1145/2810103.2813677.

[68] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. "An Efficient Boosting Algorithm for Combining Preferences". In: *J. Mach. Learn. Res.* 4 (2003), pp. 933–969.

[69] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451.

[70] Jerome H. Friedman. "Stochastic Gradient Boosting". In: *Computational Statistics & Data Analysis* 38.4 (Feb. 2002), pp. 367–378. ISSN: 0167-9473. DOI: 10.1016/S0167-9473(01)00065-2.

[71] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. "Fairness testing: testing software for discrimination". In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*. Ed. by Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman. ACM, 2017, pp. 498–510. DOI: 10.1145/3106237.3106277.

[72] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. "Approximation algorithms for partial covering problems". In: *Journal of Algorithms* 53.1 (2004), pp. 55–84. ISSN: 0196-6774. DOI: doi.org/10.1016/j.jalgor.2004.04.002.

[73] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation". In: *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 3–18. DOI: 10.1109/SP.2018.00058.

[74] Anthony Goldbloom and Ben Hamner. *Heritage Health Prize*. 2011.

[75] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[76] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. "Maxout Networks". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1319–1327.

[77] Gregory Goren, Oren Kurland, Moshe Tennenholtz, and Fiana Raiber. "Ranking Robustness Under Adversarial Document Manipulations". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. Ed. by Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz. ACM, 2018, pp. 395–404. DOI: 10.1145/3209978.3210012.

[78] Shixiang Gu and Luca Rigazio. "Towards Deep Neural Network Architectures Robust to Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[79] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. "Co-teaching: Robust training of deep neural networks with extremely noisy labels". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 8536–8546.

[80] Ralf Herbrich, Thore Graepel, and Klause Obermayer. "Large Margin Rank Boundaries for Ordinal Regression". In: *Advances in Large Margin Classifiers*. The MIT Press, 1999. Chap. 7, pp. 115–132.

[81] Tin Kam Ho. "The Random Subspace Method for Constructing Decision Forests". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 20.8 (1998), pp. 832–844. DOI: 10.1109/34.709601.

[82] Hans Hofmann. *Statlog (German Credit Data)*. Nov. 1994. DOI: 10.24432/C5NC77.

[83] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. "A probabilistic method for inferring preferences from clicks". In: *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*. Ed. by Craig Macdonald, Iadh Ounis, and Ian Ruthven. ACM, 2011, pp. 249–258. DOI: 10.1145/2063576.2063618.

[84] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. *Spambase*. June 1999. DOI: 10.24432/C53G6X.

[85] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. "Adversarial Machine Learning". In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec 2011, Chicago, IL, USA, October 21, 2011*. ACM, 2011, pp. 43–58. DOI: 10.1145/2046684.2046692.

[86] Muhammad Ibrahim and Mark James Carman. "Undersampling Techniques to Re-balance Training Data for Large Scale Learning-to-Rank". In: *Information Retrieval Technology - 10th Asia Information Retrieval Societies Conference, AIRS 2014, Kuching, Malaysia, December 3-5, 2014*.

Proceedings. Ed. by Azizah Jaafar, Nazlena Mohamad Ali, Shahrul Azman Mohd. Noah, Alan F. Smeaton, Peter Bruza, Zainab Abu Bakar, Nursuriati Jamil, and Tengku Mohd Tengku Sembok. Vol. 8870. Lecture Notes in Computer Science. Springer, 2014, pp. 444–457. DOI: 10.1007/978-3-319-12844-3_38.

[87] Alexey Ignatiev, Martin C. Cooper, Mohamed Siala, Emmanuel Hebrard, and João Marques-Silva. "Towards Formal Fairness in Machine Learning". In: *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*. Ed. by Helmut Simonis. Vol. 12333. Lecture Notes in Computer Science. Springer, 2020, pp. 846–867. DOI: 10.1007/978-3-030-58475-7_49.

[88] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems* 20.4 (2002), pp. 422–446. DOI: 10.1145/582415.582418.

[89] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. "Unbiased Learning-to-Rank with Biased Feedback". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. Ed. by Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang. ACM, 2017, pp. 781–789. DOI: 10.1145/3018661.3018699.

[90] George H. John. "Robust Decision Trees: Removing Outliers from Databases". In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, 1995*. Ed. by Usama M. Fayyad and Ramasamy Uthurusamy. AAAI Press, 1995, pp. 174–179.

[91] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. "Verifying Individual Fairness in Machine Learning Models". In: *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, August 3-6, 2020*. Ed. by Ryan P. Adams and Vibhav Gogate. Vol. 124. Proceedings of Machine Learning Research. AUAI Press, 2020, pp. 749–758.

[92] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. "Evasion and Hardening of Tree Ensemble Classifiers". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2387–2396.

[93] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 3146–3154.

[94] Haitham Khedr and Yasser Shoukry. "CertiFair: A Framework for Certified Global Fairness of Neural Networks". In: (2023). Ed. by Brian Williams, Yiling Chen, and Jennifer Neville, pp. 8237–8245. DOI: 10.1609/AAAI.V37I7.25994.

[95] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Ziwei Zhu. "End-to-End Learning for Fair Ranking Systems". In: *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. Ed. by Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini. ACM, 2022, pp. 3520–3530. DOI: 10.1145/3485447.3512247.

[96] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[97] G. Roshan Lal, Xiaotong Chen, and Varun Mithal. "TE2Rules: Extracting Rule Lists from Tree Ensembles". In: *CoRR* abs/2206.14359 (2022). DOI: 10.48550/arXiv.2206.14359. arXiv: 2206.14359.

[98] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. *How We Analyzed the COMPAS Recidivism Algorithm*. 2016.

[99] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[100] Klas Leino, Zifan Wang, and Matt Fredrikson. "Globally-Robust Neural Networks". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 6212–6222.

[101] Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2011. ISBN: 978-3-031-02141-1. DOI: 10.2200/S00348ED1V01Y201104HLT012.

[102] Ping Li, Christopher J. C. Burges, and Qiang Wu. "McRank: Learning to Rank Using Multiple Classification and Gradient Boosting". In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Ed. by John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis. Curran Associates, Inc., 2007, pp. 897–904.

[103] Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2021. ISBN: 978-3-031-01053-8. DOI: 10.2200/S01123ED1V01Y202108HLT053.

[104] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. IEEE Computer Society, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[105] Jiawei Liu, Yangyang Kang, Di Tang, Kaisong Song, Changlong Sun, Xiaofeng Wang, Wei Lu, and Xiaozhong Liu. "Order-Disorder: Imitation Adversarial Attacks for Black-box Neural Ranking Models". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM, 2022, pp. 2025–2039. DOI: 10.1145/3548606.3560683.

[106] "Mean Average Precision". In: *Encyclopedia of Database Systems*. Ed. by Ling Liu and M. Tamer Özsu. Springer US, 2009, p. 1703. DOI: 10.1007/978-0-387-39940-9_3032.

[107] Yang Liu and Hongyi Guo. "Peer Loss Functions: Learning from Noisy Labels without Knowing Noise Rates". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6226–6236.

[108] Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. "Topic-oriented Adversarial Attacks against Black-box Neural Ranking Models". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*. Ed. by Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete. ACM, 2023, pp. 1700–1709. DOI: 10.1145/3539618.3591777.

[109] Daniel Lowd and Christopher Meek. "Adversarial learning". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*. Ed. by Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett. ACM, 2005, pp. 641–647. DOI: 10.1145/1081870.1081950.

[110] Claudio Lucchese, Federico Marcuzzi, and Salvatore Orlando. "On the Effect of Low-Ranked Documents: A New Sampling Function for Selective Gradient Boosting". In: *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC 2023, Tallinn, Estonia, March 27-31, 2023*. Ed. by Jiman Hong, Maart Lanperne, Juw Won Park, Tomás Cerný, and Hossain Shahriar. ACM, 2023, pp. 646–652. DOI: 10.1145/3555776.3577597.

[111] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. "Post-Learning Optimization of Tree Ensembles for Efficient Ranking". In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. Ed. by Raffaele Perego, Fabrizio Sebastiani, Javed A. Aslam, Ian Ruthven, and Justin Zobel. ACM, 2016, pp. 949–952. DOI: 10.1145/2911451.2914763.

[112] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. "QuickScorer: Efficient Traversal of Large Ensembles of Decision Trees". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part III*. Ed. by Yasemin Altun, Kamalika Das, Taneli Mielikäinen, Donato Malerba, Jerzy Stefanowski, Jesse Read, Marinka Zitnik, Michelangelo Ceci, and Saso Dzeroski. Vol. 10536. Lecture Notes in Computer Science. Springer, 2017, pp. 383–387. DOI: 10.1007/978-3-319-71273-4_36.

[113] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. "X-DART: Blending Dropout and Pruning for Efficient Learning to Rank". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White. ACM, 2017, pp. 1077–1080. DOI: 10.1145/3077136.3080725.

[114] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. "Query-level Early Exit for Additive Learning-to-Rank Ensembles". In: (2020). Ed. by Jimmy X. Huang, Yi Chang, Xueqi

Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, pp. 2033–2036. DOI: 10.1145/3397271.3401256.

[115] Claudio Lucchese, Franco Maria Nardini, Raffaele Perego, Salvatore Orlando, and Salvatore Trani. "Selective Gradient Boosting for Effective Learning to Rank". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. Ed. by Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz. ACM, 2018, pp. 155–164. DOI: 10.1145/3209978.3210048.

[116] R. Duncan Luce. *Individual Choice Behavior: A Theoretical analysis*. New York, NY, USA: Wiley, 1959.

[117] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[118] Karima Makhlouf, Sami Zhioua, and Catuscia Palamidessi. "Machine learning fairness notions: Bridging the gap with real-world applications". In: *Inf. Process. Manag.* 58.5 (2021), p. 102642. DOI: 10.1016/j.ipm.2021.102642.

[119] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. DOI: 10.1017/CBO9780511809071.

[120] Federico Marcuzzi. *Robust Tree Ensemble against Adversarial Examples*. Università Ca' Foscari Venezia, 2020.

[121] Federico Marcuzzi, Claudio Lucchese, and Salvatore Orlando. "Filtering out Outliers in Learning to Rank". In: *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*. Ed. by Fabio Crestani, Gabriella Pasi, and Éric Gaussier. ACM, 2022, pp. 214–222. DOI: 10.1145/3539813.3545127.

[122] Federico Marcuzzi, Claudio Lucchese, and Salvatore Orlando. "LambdaRank Gradients are Incoherent". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*. Ed. by Ingo Frommholz, Frank Hopfgartner, Mark Lee, Michael Oakes, Mounia Lalmas, Min Zhang, and Rodrygo L. T. Santos. ACM, 2023, pp. 1777–1786. DOI: 10.1145/3583780.3614948.

[123] Morteza Mashayekhi and Robin Gras. "Rule Extraction from Random Forest: the RF+HC Methods". In: *Advances in Artificial Intelligence - 28th Canadian Conference on Artificial Intelligence, Canadian AI 2015, Halifax, Nova Scotia, Canada, June 2-5, 2015, Proceedings*. Ed. by Denilson Barbosa and Evangelos E. Milios. Vol. 9091. Lecture Notes in Computer Science. Springer, 2015, pp. 223–237. DOI: 10.1007/978-3-319-18356-5_20.

[124] B.W. Matthews. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), pp. 442–451. ISSN: 0005-2795. DOI: 10.1016/0005-2795(75)90109-9.

[125] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A Survey on Bias and Fairness in Machine Learning". In: *ACM Computing Surveys* 54.6 (2022), 115:1–115:35. DOI: 10.1145/3457607.

[126] Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. "Is Deep Learning Safe for Robot Vision? Adversarial Examples Against the iCub Humanoid". In: *2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 751–759. DOI: 10.1109/ICCVW.2017.94.

[127] Alistair Moffat and Justin Zobel. "Rank-biased precision for measurement of retrieval effectiveness". In: *ACM Trans. Inf. Syst.* 27.1 (2008), 2:1–2:27. DOI: 10.1145/1416950.1416952.

[128] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2574–2582. DOI: 10.1109/CVPR.2016.282.

[129] Brad A. Myers. "The importance of percent-done progress indicators for computer-human interfaces". In: (1985). Ed. by Lorraine Borman and Raoul N. Smith, pp. 11–17. DOI: 10.1145/317456.317459.

[130] Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Steven J. Lee, Satish Rao, Anthony Tran, and J. Doug Tygar. "Near-Optimal Evasion of Convex-Inducing Classifiers". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. Ed. by Yee Whye Teh and D. Mike Titterington. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 549–556.

[131] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 427–436. DOI: 10.1109/CVPR.2015.7298640.

[132] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi-Phuong-Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. "SELF: Learning to Filter Noisy Labels with Self-Ensembling". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[133] Ziad Obermeyer and Sendhil Mullainathan. "Dissecting Racial Bias in an Algorithm that Guides Health Decisions for 70 Million People". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*. Ed. by danah boyd and Jamie H. Morgenstern. ACM, 2019, p. 89. DOI: 10.1145/3287560.3287593.

[134] Luca Oneto and Silvia Chiappa. "Fairness in Machine Learning". In: *Recent Trends in Learning From Data: Tutorials from the INNS Big Data and Deep Learning Conference (INNSBDDL2019)*. Ed. by Luca Oneto, Nicolò Navarin, Alessandro Sperduti, and Davide Anguita. Cham: Springer International Publishing, 2020, pp. 155–196. ISBN: 978-3-030-43883-8. DOI: 10.1007/978-3-030-43883-8_7.

[135] Harrie Oosterhuis. "Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness". In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai. ACM, 2021, pp. 1023–1032. DOI: 10.1145/3404835.3462830.

[136] Harrie Oosterhuis. "Learning-to-Rank at the Speed of Sampling: Plackett-Luce Gradient Estimation with Minimal Computational Complexity". In: (2022). Ed. by Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, pp. 2266–2271. DOI: 10.1145/3477495.3531842.

[137] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. "Practical Black-Box Attacks against Machine Learning". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. Ed. by Ramesh Karri, Ozgur

Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi. ACM, 2017, pp. 506–519. DOI: 10.1145/3052973.3053009.

[138] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. "The Limitations of Deep Learning in Adversarial Settings". In: *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. IEEE, 2016, pp. 372–387. DOI: 10.1109/EUROSP.2016.36.

[139] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 582–597. DOI: 10.1109/SP.2016.41.

[140] R. L. Plackett. "The Analysis of Permutations". In: *Journal of the Royal Statistical Society Series C* 24.2 (June 1975), pp. 193–202. DOI: 10.2307/2346567.

[141] Tao Qin and Tie-Yan Liu. "Introducing LETOR 4.0 Datasets". In: *CoRR* abs/1306.2597 (2013).

[142] Tao Qin, Tie-Yan Liu, and Hang Li. "A general approximation framework for direct optimization of information retrieval measures". In: *Inf. Retr.* 13.4 (2010), pp. 375–397. DOI: 10.1007/s10791-009-9124-x.

[143] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN: 1-55860-238-0.

[144] Francesco Ranzato, Caterina Urban, and Marco Zanella. "Fairness-Aware Training of Decision Trees by Abstract Interpretation". In: *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. Ed. by Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong. ACM, 2021, pp. 1508–1517. DOI: 10.1145/3459637.3482342.

[145] Francesco Ranzato and Marco Zanella. "Abstract Interpretation of Decision Tree Ensemble Classifiers". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 5478–5486. DOI: 10.1609/AAAI.V34I04.5998.

[146] Francesco Ranzato and Marco Zanella. "Abstract Interpretation of Decision Tree Ensemble Classifiers". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 5478–5486. DOI: 10.1609/AAAI.V34I04.5998.

[147] Francesco Ranzato and Marco Zanella. "Genetic adversarial training of decision trees". In: *GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021*. Ed. by Francisco Chicano and Krzysztof Krawiec. ACM, 2021, pp. 358–367. DOI: 10.1145/3449639.3459286.

[148] Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. "Training Deep Neural Networks on Noisy Labels with Bootstrapping". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[149] Stephen E. Robertson and Steve Walker. "Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval". In: *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*. Ed. by W. Bruce Croft and C. J. van Rijsbergen. ACM/Springer, 1994, pp. 232–241. DOI: 10.1007/978-1-4471-2099-5_24.

[150] Anian Ruoss, Mislav Balunovic, Marc Fischer, and Martin T. Vechev. "Learning Certified Individually Fair Representations". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020.

[151] Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984. ISBN: 0-07-054484-0.

[152] Candice Schumann, Jeffrey S. Foster, Nicholas Mattei, and John P. Dickerson. "We Need Fairness and Explainability in Algorithmic Hiring". In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*. Ed. by Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil

Yorke-Smith. International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1716–1720. DOI: 10.5555/3398761.3398960.

[153] Amnon Shashua and Anat Levin. "Ranking with Large Margin Principle: Two Approaches". In: (2002). Ed. by Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, pp. 937–944.

[154] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[155] Ashudeep Singh and Thorsten Joachims. "Policy Learning for Fairness in Ranking". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 5427–5437.

[156] Jack Smith, J. Everhart, W. Dickson, W. Knowler, and Richard Johannes. "Using the ADAP Learning Algorithm to Forcast the Onset of Diabetes Mellitus". In: *Proceedings - Annual Symposium on Computer Applications in Medical Care* 10 (Nov. 1988).

[157] Nedim Srndic and Pavel Laskov. "Practical Evasion of a Learning-Based Classifier: A Case Study". In: *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 2014, pp. 197–211. DOI: 10.1109/SP.2014.20.

[158] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. "Nuclear feature extraction for breast tumor diagnosis". In: *Biomedical Image Processing and Biomedical Visualization*. Ed. by Raj S. Acharya and Dmitry B. Goldgof. Vol. 1905. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. July 1993, pp. 861–870. DOI: 10.1117/12.148698.

[159] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Trans. Evol. Comput.* 23.5 (2019), pp. 828–841. DOI: 10.1109/TEVC.2019.2890858.

[160] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[161] Michael J. Taylor, John Guiver, Stephen Robertson, and Tom Minka. "SoftRank: optimizing non-smooth rank metrics". In: *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008*. Ed. by Marc Najork, Andrei Z. Broder, and Soumen Chakrabarti. ACM, 2008, pp. 77–86. DOI: 10.1145/1341531.1341544.

[162] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. "Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 465–474. DOI: 10.1145/3097983.3098039.

[163] John Törnblom and Simin Nadjm-Tehrani. "An Abstraction-Refinement Approach to Formal Verification of Tree Ensembles". In: *Computer Safety, Reliability, and Security - SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings*. Ed. by Alexander B. Romanovsky, Elena Troubitsyna, Ilir Gashi, Erwin Schoitsch, and Friedemann Bitsch. Vol. 11699. Lecture Notes in Computer Science. Springer, 2019, pp. 301–313. DOI: 10.1007/978-3-030-26250-1_24.

[164] John Törnblom and Simin Nadjm-Tehrani. "Formal verification of input-output mappings of tree ensembles". In: *Science of Computer Programming* 194 (2020), p. 102450. DOI: 10.1016/j.scico.2020.102450.

[165] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. "Automated directed fairness testing". In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*. Ed. by Marianne Huchard, Christian Kästner, and Gordon Fraser. ACM, 2018, pp. 98–108. DOI: 10.1145/3238147.3238165.

[166] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. "Perfectly parallel fairness certification of neural networks". In: *Proc. ACM Program. Lang.* 4.OOPSLA (2020), 185:1–185:30. DOI: 10.1145/3428253.

[167] Andrew V. Uzilov, Joshua M. Keegan, and David H. Mathews. "Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change". In: *BMC Bioinform.* 7 (2006), p. 173. DOI: 10.1186/1471-2105-7-173.

[168] Vladimir Vapnik. "Principles of Risk Minimization for Learning Theory". In: *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*. Morgan Kaufmann, 1991, pp. 831–838.

[169] Sahil Verma and Julia Rubin. "Fairness definitions explained". In: *Proceedings of the International Workshop on Software Fairness, FairWare@ICSE 2018, Gothenburg, Sweden, May 29, 2018*. Ed. by Yuriy Brun, Brittany Johnson, and Alexandra Meliou. ACM, 2018, pp. 1–7. DOI: 10.1145/3194770.3194776.

[170] Daniël Vos and Sicco Verwer. "Efficient Training of Robust Decision Trees Against Adversarial Examples". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 10586–10595.

[171] Peifeng Wang, Shuangyin Li, and Rong Pan. "Incorporating GAN for Negative Sampling in Knowledge Representation Learning". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 2005–2012. DOI: 10.1609/aaai.v32i1.11536.

[172] Serena Lutong Wang, Maya R. Gupta, and Seungil You. "Quit When You Can: Efficient Evaluation of Ensembles by Optimized Ordering". In: *ACM J. Emerg. Technol. Comput. Syst.* 17.4 (2021), 55:1–55:20. DOI: 10.1145/3451209.

[173] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. "The LambdaLoss Framework for Ranking Metric Optimization". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang. ACM, 2018, pp. 1313–1322. DOI: 10.1145/3269206.3271784.

[174] Yumeng Wang, Lijun Lyu, and Avishek Anand. "BERT Rankers are Brittle: A Study using Adversarial Document Perturbations". In: *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*. Ed. by Fabio Crestani,

Gabriella Pasi, and Éric Gaussier. ACM, 2022, pp. 115–120. DOI: 10.1145/3539813.3545122.

[175] Zichen Wang, Rishab Balasubramanian, Hui Yuan, Chenyu Song, Mengdi Wang, and Huazheng Wang. "Adversarial Attacks on Online Learning to Rank with Stochastic Click Models". In: *CoRR* abs/2305.19218 (2023). DOI: 10.48550/ARXIV.2305.19218. arXiv: 2305.19218.

[176] Qiang Wu, Christopher J. C. Burges, Krysta M. Svore, and Jianfeng Gao. "Adapting boosting for information retrieval measures". In: *Inf. Retr.* 13.3 (2010), pp. 254–270. DOI: 10.1007/s10791-009-9112-1.

[177] Xin Wu, Qing Liu, Jiarui Qin, and Yong Yu. "PeerRank: Robust Learning to Rank With Peer Loss Over Noisy Labels". In: *IEEE Access* 10 (2022), pp. 6830–6841. DOI: 10.1109/ACCESS.2022.3142096.

[178] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. "Listwise approach to learning to rank: theory and algorithm". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Ed. by William W. Cohen, Andrew McCallum, and Sam T. Roweis. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 1192–1199. DOI: 10.1145/1390156.1390306.

[179] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. "Adapting Markov Decision Process for Search Result Diversification". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White. ACM, 2017, pp. 535–544. DOI: 10.1145/3077136.3080775.

[180] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. "Context-aware ranking in web search". In: *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*. Ed. by Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis N. Efthimiadis, and Jacques Savoy. ACM, 2010, pp. 451–458. DOI: 10.1145/1835449.1835525.

[181] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. "Support vector machines under adversarial label contamination". In: *Neurocomputing* 160 (2015), pp. 53–62. DOI: 10.1016/J.NEUCOM.2014.08.081.

[182] Jun Xu and Hang Li. "AdaRank: a boosting algorithm for information retrieval". In: *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*. Ed. by Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando. ACM, 2007, pp. 391–398. DOI: 10.1145/1277741.1277809.

[183] Zhuolin Yang, Linyi Li, Xiaojun Xu, Bhavya Kailkhura, Tao Xie, and Bo Li. "On the Certified Robustness for Ensemble Models and Beyond". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[184] Ting Ye, Hucheng Zhou, Will Y. Zou, Bin Gao, and Ruofei Zhang. "RapidScorer: Fast Tree Ensemble Evaluation by Maximizing Compactness in Data Level Parallelization". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Ed. by Yike Guo and Faisal Farooq. ACM, 2018, pp. 941–950. DOI: 10.1145/3219819.3219857.

[185] Haitao Yu, Rajesh Piryani, Adam Jatowt, Ryo Inagaki, Hideo Joho, and Kyoung-Sook Kim. "An in-depth study on adversarial learning-to-rank". In: *Inf. Retr. J.* 26.1 (2023), p. 1. DOI: 10.1007/S10791-023-09419-0.

[186] Hugo Zaragoza, Nick Craswell, Michael J. Taylor, Suchi Saria, and Stephen E. Robertson. "Microsoft Cambridge at TREC 13: Web and Hard Tracks". In: *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*. Ed. by Ellen M. Voorhees and Lori P. Buckland. Vol. 500-261. NIST Special Publication. National Institute of Standards and Technology (NIST), 2004.

[187] Meike Zehlike and Carlos Castillo. "Reducing Disparate Exposure in Ranking: A Learning To Rank Approach". In: *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. Ed. by Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen. ACM / IW3C2, 2020, pp. 2849–2855. DOI: 10.1145/3366424.3380048.

[188] Ethan Zhang and Yi Zhang. "Average Precision". In: *Encyclopedia of Database Systems*. Ed. by Ling Liu and M. Tamer Özsu. Springer US, 2009, pp. 192–193. DOI: 10.1007/978-0-387-39940-9_482.

[189] Zhilu Zhang and Mert R. Sabuncu. "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo

Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 8792–8802.

[190] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. 1st. Chapman & Hall/CRC, 2012. ISBN: 1439830037.

Ringraziamenti

In quest'ultima parte della tesi desidero dedicare alcune parole per ringraziare le persone che mi hanno accompagnato durante il mio dottorato di ricerca.

Innanzitutto, un doveroso ringraziamento va alla mia famiglia, ai miei amici, ai miei colleghi e al gruppo di scacchi cafoscarino. Un ringraziamento speciale va alla mia compagna Anna Mantovani, nonché futura moglie.

Dei ringraziamenti speciali vanno a Claudio Lucchese e Salvatore Orlando, i quali sono stati i miei mentori durante questo percorso di dottorato. A Stefano Calzavara con il quale ho avuto il piacere di collaborare in più occasioni. Al mio collega e amico Lorenzo Cazzaro che in prima persona ha condiviso con me le gioie e i dolori della ricerca. Un doveroso ringraziamento va a Nicola Miotello per l'infinita pazienza e senza il quale sarebbe stato impossibile barcamenarsi nella burocrazia del dottorato.

Un ringraziamento al *Information Retrieval Lab* dell'Università di Amsterdam che mi ha ospitato durante il mio periodo all'estero e a tutte le persone speciali incontrate durante gli anni del dottorato.

Infine, un ringraziamento ai revisori di questa tesi, Andrew Yates e Gabriele Tolomei e a chiunque abbia trovato il tempo di leggerla.

Acknowlegments

In this final part of the thesis, I wish to express my gratitude to those who supported me throughout my doctoral studies.

First and foremost, I wish to express my heartfelt gratitude to my family, friends, colleagues, and the Ca' Foscari chess group. A special thanks goes to my partner, Anna Mantovani, my future wife.

I want to express my special gratitude to Claudio Lucchese and Salvatore Orlando, my mentors, throughout this doctoral journey. I am also grateful to Stefano Calzavara, with whom I have collaborated on multiple occasions. To my colleague and friend Lorenzo Cazzaro, who personally shared the joys and pains of research with me. Heartfelt thanks are due to Nicola Miotello for his infinite patience, without whom untangling the bureaucracy encountered during the doctoral studies would have been impossible.

Thanks to the *Information Retrieval Lab* at the Universiteit van Amsterdam for hosting me during my time abroad and to all the special people I met during this journey.

Finally, I would like to thank the reviewers of this thesis, Andrew Yates and Gabriele Tolomei, and anyone who has read it.