Model-Centric Verification of Artificial Intelligence

Nicholas Gisolfi
January 12, 2022
CMU-RI-TR-22-02

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:
Artur Dubrawski, *Chair*
Reid Simmons
Stephen Smith
Madalina Fiterau, *University of Massachussetts Amherst*

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Robotics.

Abstract

This work shows how provable guarantees can be used to supplement probabilistic estimates in the context of Artificial Intelligence (AI) systems. Statistical techniques measure the expected performance of a model, but low error rates say nothing about the ways in which errors manifest. Formal verification of model adherence to design specifications can yield certificates which explicitly detail the operational conditions under which violations occur. These certificates enable developers and users of AI systems to reason about their trained models in contractual terms, eliminating the chance that otherwise easily preventable harm be inflicted due to an unforeseen fault leading to model failure.

As an illustration of this concept, we present our verification pipeline named Tree Ensemble Accreditor (TEA). TEA leverages our novel Boolean Satisfiability (SAT) formalism for voting tree ensemble models for classification tasks. Our formalism yields disruptive speed gains over related tree ensemble verification techniques. The efficiency of TEA allows us to verify harder specifications on models of larger scales than reported in literature.

In a radiation safety context, we show how Local Adversarial Robustness (LAR) of trained models on validation data points can be incorporated into the model selection process. We explore the relationship between prediction outcome and model robustness, allowing us to yield the definition of LAR that best satisfies the engineering desiderata that the model should be robust only when it makes correct predictions.

In an algorithmic fairness context, we show how Global Individual Fairness (GIF) can be tested, both in and out of data support. When the model violates the GIF specification, we enumerate all counterexamples to the formula so we may reveal the structure of unfairness that is absorbed by the model during training.

In a clinical context, we show how a Safety-Paramount Engineering Constraint (SPEC) can be satisfied simply by tuning the prediction threshold of the tree ensemble. This facilitates a pareto-optimal selection of prediction threshold such that false positives cannot be reduced further without compromising safety of the system.

The goal of this thesis is to investigate if formal verification of trained models can answer a wide range of existing questions about real-world systems. Our methods are meant for those who are ultimately responsible for ensuring the safe operation of AI in their particular context. By expanding current practice in Verification and Validation (V&V) for trained tree ensembles, we hope to increase real-world adoption of AI systems.

Acknowledgements

This LaTeXtemplate was provided by Manfred Paulini, thank you for keeping my writing organized!

Grad school has made me greatly value support networks, and I could not do this if it were not for the support of my colleagues and peers in the Auton Lab and in the RI. Jack, Maria, Ceci, Rob, Kyle, Peter, and all my Autonian collaborators for work in this thesis. Rob, Kyle, and Leo, for their experience and knowledge that added structure to my ideas and musings. Marijn and Ruben, for their support and encouragement of the work and their help getting me up to speed with SAT. My committee, Reid, Steve, and Ina for providing very helpful feedback on my dissertation. Saswati, Jarod, and Andrew for programming efficiencies which made the frameworks tractable. Ben and Sibi, who made it a joy to get onto campus before 9am for office-brewed coffee! All other Autonians who ever tuned into lab lunches or stopped by for a chat with me over coffee, you helped me discover a sense of community in the Auton Lab that really lifted my spirits when I needed it most.

My family, Terri, John, and Emily for always being there for me. Thank you for your unwavering support of my hopes and aspirations.

My advisor, Artur, who always looked out for me and gave me the guidance, patience, and time I needed to develop my skills. Thank you for giving me the opportunity to pursue research and the freedom to find an area of focus that resonates with me.

Most importantly, my partner, Emily, for supporting me unconditionally through the stresses of grad school. Thank you for reminding me of my strengths whenever I ruminate my weaknesses. I love you.

Contents

List of Tables

List of Algorithms

List of Figures

Chapter 1

Introduction

1.1 Motivation

There is a lot of promise in the field of AI, and our shared goal is to ensure that as many people as possible are unlocking its potential to solve new problems across many different application domains. We find that there is a common hurdle that limits the rate of adoption of AI systems, and it often comes down to a matter of trust. Many people who are ultimately responsible for the output of an AI system do not trust their models to the extent necessary to warrant deploying AI into some new domain.

Consider Isaac Asimov's First Law of Robotics: a robot may not injure a human being or, through inaction, allow a human being to come to harm. This is a great example of a critical design specification for an AI system. An AI system is not restricted to physical robots, it could also be a black-box, Machine Learning (ML) model. Injury and harm need not solely describe physical harm, it can also involve confusion resulting from AI systems making recommendations that are counterintuitive to humans. This is where people start to use their trust in AI systems. We know that at times AI will do things that are counterintuitive to a human's intuition of how the model should act. The fear is that in those cases, the AI system might cause easily preventable harm that a human decision maker would never inflict. Can we really trust an AI system if we are not certain that it adheres to critical design specifications such as Asimov's First Law?

Current practice addresses this question by treating AI systems as black boxes to be interrogated with data-centric, statistical methods. Probabilistic estimates of model behavior require a level of confidence that depends upon the acceptable margin for uncertainty in the application domain. If we consider a movie recommendation system, the acceptable margin for uncertainty is high because a bad movie at most

wastes a few hours of one's day. If we are considering an autonomous vehicle context or any other critical application domain, the acceptable margin for uncertainty is much lower. Even though statistical methods test a model repeatedly to improve the confidence bounds on their estimates, it is impossible to provide a truly comprehensive assessment and achieve absolute certainty. All we can say is that we never observed a particular failure during the course of many tests and this gives us reason to believe that the same failure is very unlikely to occur once the model is deployed. For missions of extended length, such as developing semi-autonomous life support systems for deep space habitats, even small chances of model failure have plenty of opportunity to manifest.

Statistical techniques alone are not well suited to the level of trust that is required of critical systems. This is due to the fact that low error rates only tell us that critical errors happen with low frequency; we are left wondering how these critical errors will manifest. The scope of this thesis work focuses on critical errors that inflict easily preventable harm. Types of critical errors we will explore throughout this work include notions of robustness, fairness, and safety. For robustness (See Chapter 3), we test a model for consistency of output under the presence of imperceptible perturbations on attribute values of a particular input. For fairness (See Chapter 4), we test a model to ascertain whether it always will treat similar individuals similarly. For safety (See Chapter 5), we test a model to ensure that no intelligible mapping from input to output represent counterintuitive, harmful recommendations that a human would never make. Notably missing from this list is misclassification, which is not necessarily a critical error in many contexts. Human decision making produces errors for hard edge cases, so an AI should be allowed to make the same types of mistakes. Instead, we are focused on cases where AI exhibits undesirable decision logic that humans do not exhibit, such as placing a decision boundary between two indistinguishable inputs, producing wildly different output for two similar inputs, and making errors in the easiest of cases for humans to adjudicate.

AI systems are not actually black boxes, and we do not have to treat them as such. Rather, they are defined by a collection of discrete components that interact in order to produce seemingly intelligent model behavior. It is the behavior of these components that determines the way in which a model reaches its prediction. One way to address the challenge of interrogating the internal components of a trained model involves a combination of logical and statistical techniques. Statistical machine learning is adept at learning useful policies from data, whereas formal logic and automated reasoning are adept at determining whether a consequent can be reached from a set of antecedents.

Our work bridges a gap between the successes of two big ideas in the history of

the field of AI. In the early days, AI systems comprised symbolic logic, rules, and expert knowledge; Expert Systems were readily amenable to testing with formal logic. Contemporary AI systems comprise components and meta-structures that are borne of an optimization procedure to fit a model to data. Currently prevalent Statistical ML is readily amenable to testing with statistical methods. The shift in the focus of the field of AI is due in part to a different view of the inner workings of the human mind. The symbolist view that human intelligence is derived through logical inference was gradually supplanted with the connectionist view that the recruitment of groups of neurons among the interconnected structure of the human brain produces intelligent behavior. The AI systems that we build today reflect that shift in mindset. There are strengths and weaknesses to both frameworks; for instance, statistical models may generalize better to yet unseen data, but symbolic models may be more explainable. We view the combination of the strengths of logical and statistical methodologies as a best-of-both-worlds approach to building successful, trustworthy, AI systems. If a model produced by statistical techniques is checked with pure logic, then we can move toward providing proofs that the model does indeed conform to design specifications, engineering desiderata, and even expert knowledge. This enables both contractual and probabilistic reasoning about models that are to be deployed in the real-world.

While the question of whether our methods increase a human's sense of trust in an AI system is outside the scope of this thesis, we do claim that our work increases the trustworthiness of an AI system. A formal proof and accompanying certificate are a type of explanation. They describe the operational conditions under which the learned structure of a model satisfies or violates critical design specifications. Reasoning about the model structure provides insight that current practice cannot obtain. This additional information about a model's strengths and weaknesses keeps all stakeholders more informed. We show that formal verification of model adherence to critical design specifications complements the probabilistic estimates that are typically used to inform decision making when building AI systems. Certificates enable developers and users of AI systems to reason about their models in contractual terms which enables specification-driven design for AI systems. We show how this information can be combined to answer existing questions about real-world AI systems.

1.2 Thesis Statement

Formally verifying that a model fit to data by statistical methods adheres to critical design specifications increases the trustworthiness of the model and expands current practice in verification and validation for Artificial Intelligence.

1.3 Organizational Structure

Chapter 1 provides a summary of the content and scope of this thesis, we discuss related work in two areas, Explainable AI (XAI) and Verification and Validation (V&V). We present details necessary to appreciate the difference between our methods and baselines (ReLU Simplex Theory Solver (Reluplex) in Section 1.4.2 and VoTE in Section 1.4.2) we compare to throughout this document. We identify limitations of current practice and outline how this thesis addresses those problems. A primer containing an introduction to background knowledge of the technologies used in this thesis is provided in Appendix E.

Chapter 2 presents our verification pipeline named Tree Ensemble Accreditor (TEA). This chapter contains links to relevant sections of the thesis that expand upon select components and capabilities of TEA. We provide details of our novel Boolean Satisfiability (SAT) formalism for trained, voting, tree ensemble models as well as a representation of a set of data in logic. We briefly describe the existing repertoire of design specifications we can verify for a trained model, which include robustness (Chapter 3), fairness (Chapter 4), and safety (Chapter 5). The scope of this thesis work will involve using TEA to verify properties of tree ensemble models. Our experiments show how certificates can be used in ways that are useful to developers of AI systems as well as end users who are ultimately responsible for acting upon the recommendations made by a particular model.

Chapter 3 explores how Local Adversarial Robustness (LAR) can be verified within TEA. A model should provide consistent output under the presence of imperceptible changes to model inputs, and a LAR certificate tells us whether a model adheres to this specification within the neighborhood of a particular input. For other model classes, distance to the decision boundary may be easy to obtain, but for tree ensembles or even neural networks, the decision boundary is more of an abstract concept that is hard to characterize. LAR characterizes distance to the decision boundary, where the tree ensemble changes its predicted class label. Experimental results provide evidence that TEA is able to verify LAR for larger verification tasks and larger models than reported in literature. We show how LAR answers existing, real-world questions in a nuclear safety context. We prove that a model is robust

to certain types of adversarial attacks, which provides evidence to developers of AI systems that their models behave as expected. We characterize robustness of the model by prediction outcome (e.g. True Positive (TP)s), and show how LAR can be used in ways that are immediately relevant to data scientists.

Chapter 4 explores how Global Adversarial Robustness (GAR) can be verified with TEA. A model should provide similar outputs for similar inputs, and a GAR certificate tells us whether a trained model satisfies this criterion for all possible pairs of inputs to the model. GAR characterizes smoothness of the decision surface, ensuring that the vote tally of the trees in the ensemble never changes drastically for similar inputs. Smoothness of a piece-wise constant model class, such as tree ensembles, is difficult to measure, and GAR represents a way to formally check that the trained model exhibits a desirable level of smoothness. TEA is the first formalism to verify GAR for tree ensemble models of application-scale. We show that our GAR formalism is able to verify an Individual Fairness (IF) specification, and we show how Global Individual Fairness (GIF) certificates can be used to characterize the fairness of a trained model. Experiments show how this information can support specification-driven model selection that picks the model with highest accuracy that also adheres to a particular definition of GIF. TEA is also able to enumerate counterexamples GIF which reveals the structure of unfairness that the model absorbs during training.

Chapter 5 explores how a Safety-Paramount Engineering Constraint (SPEC) can be verified with TEA. A model should not make errors on inputs that a human adjudicator will always easily avoid. No matter how adept a model is at classifying inputs that represent difficult edge cases, it is hard to trust the model if it errs in embarrassing ways for easy inputs. SPECs bridge the gap between model-centric and domain-centric design specifications, allowing developers of AI systems or domain experts to define rules to which the trained model should adhere. A case study verifying SPECs for an Airborne Collision Avoidance System (ACAS) context shows that when safety is paramount, TEA is a much more efficient method of verifying model safety than existing baselines which leverage different model classes and different formalisms. We show how SPEC certificates can identify a pareto-optimal selection for predictive threshold in a tree ensemble, where False Positive (FP)s cannot be reduced without the model violating SPECs. This capability is shown in a critical care medicine context, where SPEC certificates provide proof that an AI model will never inflict easily preventable harm to a patient.

Chapter 6 discusses the conclusions we draw from our experiments in previous chapters and the potential impact of our work. We submit that human decision making should not be trusted if an adjudicator does not exhibit robustness, fairness, or safety and we show that this thesis work provides a method by which we may verify

that an artificially intelligent system exhibits these same properties. We claim that our work expands current practice in V&V for AI systems by providing contractual guarantees for when a model does or does not adhere to criticial design specifications. We outline intended use cases and expand on ethical considerations of the work. We finish this chapter with a summary of what, to the best of our knowledge, are the novel contributions of this thesis to the field of AI.

Appendix A contains a glossary of acronyms used throughout this document. All acronyms throughout the document are hyperlinked to the glossary for convenience.

Appendix B presents additional design specifications under development.

Appendix C presents preliminary work on using TEA to mine data for useful, data-driven specifications. We present a method for finding the maximal subarray that takes sparsity into account, which is often a reasonable assumption when searching for 2D, axis-aligned boxes that contain homogeneous data. Model adherence to these intelligible specifications can be verified with TEA and we show examples of future directions for tighter integration with TEA.

Appendix D presents preliminary work on using TEA to generate model-centric explanations for specification violations. We show that intelligible explanations can be generated by summarizing formal proofs through extraction of Minimal Unsatisfiable Set (MUS)s. Our methods address a few common questions of AI systems. Why did the model make a classification error; is the failure due to the data or the model structure? What changes can be made to the model to ensure it will not make the same mistake again while also maintaining accuracy on data the model has seen before? What kind of data confuses the model?

Appendix E provides background information relevant to the technologies in this thesis with the goal of making our work more accessible to a broader audience. We also compare the motivation of our work with similar problems outside of AI.

1.4 Related Work

This thesis work fits into the subfields of explainable and trustworthy AI. We use formal certificates as a type of explanation for model behavior that describes the operational conditions under which the model is known to satisfy critical design specifications. We submit that proving a model adheres to specifications makes the model trustworthy in ways that are important to stakeholders. These formal certificates are most useful in cases where the verified model is of sufficient complexity such that other explainable methods start to break down in terms of their effectiveness. We explore other works in the area of Explainable AI (XAI) which span a broad scope and discuss types of explanations, the reason they are important, and how others generate explanations.

Our work also fits into the field of Verification and Validation (V&V), which comprises statistical and formal testing methods. Past work in formal V&V typically focuses on verifying correctness of software or control systems. Only recently is the focus of V&V being directed to trained AI models, in part due to the fact that AI systems were previously deemed too large and too stochastic for direct application of methods existing at the time. We outline statistical and formal V&V efforts that share a similar goal, describe differences between statistical and formal V&V, and describe a few V&V methods that range from standard repertoire to active research areas. We identify limitations in the current state-of-the-art, define our niche, and describe how this thesis work addresses those limitations.

1.4.1 Explainable AI (XAI)

In the broadest sense, *explanations* offer answers to the question of *why* rather than the question of *what*. Explanations represent an exchange of information the communicates knowledge about a process governing a particular phenomenon. While all explanations are typically logical in nature, what constitutes the best definition for explanation is an open question [122, 124].

Two contrasting schools of thought on the theory of scientific explanations rely on different strategies for logical inference. Hempel [85] is credited with formalizing a deductive, nomological view of explanation. According to Hempel, the adequacy of an explanation relied on a few factors; most notably including a causal relationship between the premise (explanans) and the phenomenon being explained (explanandum). The goal is to produce explanations that are causal, testable, and true [85]. Formal methods and automated reasoning are designed for this type of inference.

Another formulation for the basis of scientific explanations involves a unification theory; a view of explanation as an instance of abductive reasoning [110]. Abduction

involves synthesizing a logical premise that is most likely to exhibit a causal relationship with the observed phenomena. Explanations of this type are plausible, but not necessarily verifiable, since causality is inferred, not guaranteed. The goal is to produce explanations that are general, simple, and cohesive [181]. Statistical methods are are designed for this type of inference.

Both formulations for the structure of scientific explanation have their merits, and our work highlights the importance of incorporating both by combining formal methods and statistical methods. While formal methods do not reveal underlying causal structure in data, they do reveal the structure of logical entailment that guides individual components of a trained model to map input values to output predictions. Statistical methods ignore interactions of these components an treat the system as a black box where sufficiently strong probabilistic estimates represent useful measurements of black box behavior.

What are desirable properties of an explanation?

Subfields of the social sciences focus on how explanation can be applied effectively to artificial intelligence [124, 135]. They contend that, among other things, good explanations are contrastive, biased, and causal.

Contrastive explanations highlight a difference between two entities. The idea driving this assessment is that it is easier to come up with explanations that are relative rather than absolute. Providing explanations that are contrastive helps keep the explanations concise. Some research focuses on providing explanations that highlight dissimilarities within a group of points [28, 89]. Others providing explanations by comparing two different models and observing their differences [196].

Bias, in the context of XAI, means that explanations should account for prior knowledge and experience of both the explainer (person giving the explanation) and the explainee (person receiving the explanation). To the one giving the explanation, useful explanations may be the ones that helped them overcome their own prior misunderstandings in a particular context. This may or may not be the best explanation from the perspective of the person on the receiving end. When an explanation is delivered, it is accepted or rejected depending on whether it is congruent with the explainee's prior experiences and current expectations [30, 51, 89, 95, 189].

Causality has very specific meaning in statistics. In the context of XAI, this desiderata indicates that explanations should be contractual rather than probabilistic. Humans are generally more adept at logical reasoning in contrast to probabilistic reasoning. An explanation that points to conditions that are responsible for a particular phenomenon has higher utility than an explanation that qualifies the conditions with probabilistic terms [135].

Why are explanations important in the context of AI systems?

There is no objective truth stating that explanations are important to incorporate into every AI system. The demand for XAI depends heavily on the critical nature of the application context. The form of useful explanations depends heavily on the specific needs of the target audience for AI systems. A few of the most common motivations are outlined in this section. While there is work [51], there is no consensus on how to treat XAI as a science. In the absence of a field-wide standard approach, the following themes take on variable levels of relevance depending on the particular application domain under consideration. A combination of all these factors show that if AI systems were capable of providing good explanations, they would likely receive better reception as they are applied to new domains [66, 69].

Providing explanations facilitates the adoption of AI in new contexts

If understanding, performance, and trust are integral to the adoption of AI in new, mission-critical fields, then a model's inability to rationalize its behavior is rate-limiting. Without explanations to clarify the reasons for observed model behaviors, AI is not applicable in some real-world contexts [63, 119, 193], and there is a non-trivial chance that AI will do real harm, even if it is unintended [165]. Providing explanations for complex phenomena also lowers barriers to use these models, making it easier for people to deploy useful models in new application domains [63, 93].

Explanations for model recommendations are an important part of decision support systems in clinical contexts [28, 34, 92, 121, 153, 152]. The use of ML to support and automate fair decision-making can lead to disparate impact [4, 8], and it may not only reproduce but also compound societal discrimination [46], making it important to demonstrate that a model exhibits fair decision making before deploying the model. In societal contexts, it is often important to provide a justifications for decisions [36, 42, 122, 129]. As policymakers begin to address the promise and impact of AI in society, explanations for model behavior are becoming of interest to government agencies [78] and even required by law [59, 64, 95, 148, 164, 189].

In order for a deployed AI system to be most useful, we need humans to take action based on model recommendations. Explanations can foster a sense of trust [108, 203] and this sense of trust affects user decisions and behaviors with regard to AI [183]. For many AI systems, especially decision support, humans work alongside the model, so it is important for the user to trust the model [122]. Trust determines whether humans are willing to take action based on the recommendations of an AI [76] and it is a necessary condition for the acceptance of technology and its ultimate adoption in new application domains [12, 66, 73, 76, 88, 114, 117, 122, 149, 164, 183, 189]

Explanations help practitioners build better models

Understanding a model's strengths and weaknesses can inform decision making in order to produce better models. Explanations can help improve the engineering process that produces the model we wish to explain. Providing explanations help data scientists to select the best available models [89]. Explanations help practitioners determine which models are more generalizable to unseen data [162], and highlight which attributes are the most influential in the model's overall prediction [126].

Models are generally only as good as the data with which they are trained. It has also been shown that explanations can help aid clinical doctors in labeling data points, in turn leading to better data [63, 191]. We have previously shown that providing explanations of gaps in data coverage help nuclear physicists generate higher fidelity data, which in turn leads to better performing models [75].

Building better models is a core focus in algorithmic fairness contexts. Existing methods include internal auditing [156], model selection [36, 42, 129], and human-AI collaboration [45, 179, 180]. Recent research has highlighted the challenges of predictive multiplicity [129] and underspecification [42], and it has been shown that different models that exhibit the same accuracy on fairness metrics may behave entirely different for different subpopulations [36].

Imposing a constraint that an AI system be explainable leads to better decision making when collecting data and choosing model classes [165]. It also limits the accumulation of technical debt that arises from continually adding additional layers of automation to avoid making principled design decisions [173].

Explanations fuel knowledge discovery in application domains

One commonly cited goal of providing explanations for model behavior is to precipitate a sense of understanding [89, 114, 115, 152, 196]. AI may lead to the discovery of new, previously hidden, structure in data [29, 63, 75, 74, 191]. Explanations focusing on highlighting the utility of each attribute in data may lead to new ways to define features [126, 162]. In nuclear safety contexts, AI can discover meaningful deficiencies in data which highlight input regions where generating data with physics-based modeling can fill the gap [75]. In clinical contexts, AI leads to discovery of effective interventions which save lives [34, 92], and vital indicators of health status that allow for preventative interventions to improve care [32, 121, 195, 200].

Explanations aid human ability to communicate results

Explanations help human end-users communicate outputs of learning models and help inform their decision making [22, 74, 89, 124, 162]. Visual explanations often

allow users to infer correlations between select attributes in data [93]. Explanations increase the accuracy, response time, and answer confidence for users interpreting AI models [95]. By staging interpretable models with black box models, it's also possible to yield the simplest prediction possible, making it easier for end users to understand why a certain outcome is observed [192].

DARPA highlights six user questions that they consider to be the basis for any explainable AI system [78]. For example, some of these questions are *why did the model do that?* or *when does the model fail?*. By providing explanations that answer these sorts of questions, human end users have information they need to explain model behavior to others. This makes AI systems more likely to earn the trust of a human [167].

Explanations foster trust in AI systems

Like explanation, trust is also difficult to define in quantitative terms because it describes a human cognitive state. ABI, or accuracy, benevolence, and integrity was an early framework for characterizing the factors necessary to gain trust of humans [130]. [168] extended the ABI framework to include notions of predictability as integral to the development of trust; a suggested that is echoed by other authors who suggest that there is a temporal element to trust [76].

Trust in AI can be viewed as individual consent to relinquish autonomy for completing a task to an external agent [130, 168]; a confluence of system fairness, explainability, auditability, and safety [76, 88, 164, 172, 183]; or a byproduct of explanation that calibrates human expectation about model performance [203]. Trust acknowledges the perceived benevolence of the designers of the system [164]; a perception that drops sharply for individuals outside the field of AI [124, 135].

What are existing methods for generating useful explanations?

There are many ways to generate explanations, due in part to the fact that the problem of XAI is under-specified. There are strengths and weaknesses of each approach, and choosing the correct type of explanation depends heavily on the needs of the intended stakeholders. Each of these clusters of techniques make similar assumptions on the structure of useful explanation.

Counterfactual explanations

A counterfactual explanation shows how a model's output will change for two different inputs. Counterfactuals are contrastive and actionable, satisfying a few of the

desiderata for useful explanations. Some existing methods for generating counterfactual explanations include [89, 93, 105, 189]. If a human wishes to change the observed behavior of a model and they possess a counterfactual explanation for the observed behavior, changing the inputs to those stated in the explanation will cause the model to exhibit the behavior prescribed by the user.

Leveraging low-dimensional structure

Concise explanations tend to be more desirable to humans because they take human constraints into account; it is difficult for humans to interpret explanations that describe high-dimensional structure in data. [29, 125] constrain their systems to learning arbitrarily complex, polynomial models within 2D-axis aligned projections of data. Low-dimensional structure is visualizable which aids acceptance of the explanation [63, 93, 196]. Others constrain themselves to simple models in high-dimensional spaces. Long lists of simple rules can get complicated for humans to understand, so limiting the length of the list is key to preserving explainability of the model [95, 122] and short lists are more desirable [114, 125].

Our own prior work focuses on leveraging low-dimensional structure to serve as a type of explanation [75, 74]. By finding 2D-axis aligned range rules that reveal actionable simplicity in data, we can use these rules to describe patterns in data, or to act as simple models that can be staged with more complex models, yielding interpretable predictions whenever possible and black box predictions when no simple explanation exists for a particular data sample.

Data-centric explanations

By comparing, contrasting, or generalizing from select data samples, it is possible to provide data-centric explanations for model behaviors. Comparing individual samples or groups of samples help users focus on differences between the sets of data samples [28, 119]. Data-centric explanations are most useful when the model behavior in question is expected to manifest as a direct result of a particular input.

Data-centric explanations can be obtained from multiple efforts including GAMUT [89], VizML [93], MAPLE [154], What-If [196], LIME [162] and ANCHOR [163]. Generated explanations range from *show-me* visuals to *tell-me* captions. LIME [162] produces data-centric explanations by training post-hoc models on subsets of points that reside in a region of interest. The goal is that the simpler model trained on the subset of local data will offer insight into the patterns learned by the base model in that local area. It is worth noting that if we are looking for explanations that apply to data, then data-centric explanation approaches are appropriate. However, if we are

trying to explain the behaviors of a trained model by making appeals to the data it was trained or evaluated on, we would be using data-centric explanations to generate post-hoc explanations for the model behavior.

Model-centric explanations

In cases where we expect the AI system to be at fault, model-centric explanations provide insight into how learned structure may be contributing to some observed behavior. SHAP [126] is a notable example that provides explanations for why a model makes a particular prediction by highlighting the influence that each attribute had in the model's prediction logic. LIME [162] also reports the influence of individual attributes within a simple model fit to a local subspace where an explanation of base model behavior is desired. Model-centric explanations may involve highlighting subspaces where the model exhibits desired or undesired behaviors [115].

Our own recent work [139] offers contrastive explanations for model behavior by identifying two nearby model states where similar inputs generate different outputs. Depending on the length of the resulting counterfactual, the explanations can point to components of the model that are responsible for an observed failure.

Restricting the model class to those that are interpretable

Some works argue that the need to produce explanations for model behaviors is only a result of the design decision to deploy a black box model [165, 173]. By choosing models that are naturally intelligible to a human being, they allow the humans to generate their own explanations and interpret the model in ways they see fit. Generalized Additive Models [29, 89] provide high accuracy with while reasoning about simple structure that is interpretable to users. Extracting rules by fitting decision trees to any other model produces an interpretable post-hoc model that can be interrogated by humans [95]. SHAP [126] represents a popular model-centric approach to explanation that highlights feature importance and is primarily designed for logistic regression, although it has been extended to some additional model classes including tree ensembles.

Post-hoc justifications

In some cases, it is sufficient to provide justifications for the decisions made by an AI system, even if these justifications are not reflective of the underlying decision logic [47]. Post-hoc explanations often rely on human data and a completely separate inference task to determine what the most desirable explanation for a particular question about a trained model. Such approaches often rely on human-annotated

data [199]. Other approaches do not rely on new data, but train a new model [162], in the process reducing the likelihood that the explanation is a faithful representation of the underlying causes for an observed model behavior.

Trading accuracy for intelligibility

Low intelligibility makes adoption of models difficult [29] and intelligibility may be as important as performance accuracy [95]. There is a widely recognized trade-off between accuracy and interpretability in XAI [115, 119, 122, 125]. If we intend for a system to be interpretable, we should implement simple models [165]. Some works involve staging simple models with more complex models in order to provide confident predictions with simple models when possible, and fall back on more powerful systems when needed [74, 192].

1.4.2 Verification and Validation (V&V)

The V&V process provides the evidence required for all stakeholders in the design process to trust that a system does what it is supposed to do. Verification involves building the system right; testing a system to ensure it satisfies all specifications set at the onset of the design process. Validation involves building the right system; making sure that the system adequately addresses the problem it was designed to solve. V&V is a critical part of the engineering design process. Without any V&V, development would happen in under-specified environments, making it difficult to determine whether the system behaves as expected.

All forms of model testing falls under the umbrella of V&V, and testing AI systems prior to deployment is standard practice. In the context of ML and AI, V&V tends to involve testing with statistical methods, which provide probabilistic estimates of model performance, measures of prediction quality, guidance for hyperparameter selection, and recommendations during model selection. Formal V&V, on the other hand, involves testing with logic-based methods, and provides provable guarantees of model adherence to a range of critical design specifications. As of 2021, formal V&V is not a necessary requirement for an AI system to be considered state-of-the-art, deployment-worthy, or useful. This is due in part to the fact that state-of-the-art AI systems are becoming increasingly complex, and the stochastic nature of these models makes formal verification difficult using existing techniques. Formal V&V is a topic of growing interest in AI as models are being designed for increasingly critical contexts, where the level of trustworthiness required of the AI system demands more rigorous testing methods to demonstrate resiliency, reliability, and safety.

How do statistical and formal methods for V&V differ?

Current practice in ML and AI relies heavily on statistical methods, which allow practitioners to treat learning models as black boxes. Statistical V&V is useful because it do not require knowledge of the underlying structure of the model, which varies in form and complexity between different model classes and different application domains. They leverage sampling techniques to generate data, measure the response of the black box system, and yield probabilistic estimates on model behaviors. When defining the notion of an anomaly in statistical terms, we generally refer to samples with attribute values that make them outliers when compared to other samples drawn from the same underlying distribution.

Statistical methods characterize a model in a way that is dependent on access to quality data. At times, statistical approaches demand copious amounts of data in order to provide sufficient probabilistic bounds that satisfy requirements. ML models are often trained with statistical techniques, leading to models that minimize loss metrics over a set of data. If there is no data support in a given region of inputs, then statistical techniques cannot provide insight.

Formal V&V proves properties about an abstract formalism of an underlying system. Knowledge of model structure is necessary, unlike statistical methods, which makes formal approaches cumbersome and different depending on the model class. The draw of formal methods is that they do not require access to data when testing a trained model. The response of the model can be determined by internal components of the model itself, and all possible input-output mappings are tested. The types of properties that can be proved are in some ways orthogonal to the properties that can be measured with statistical methods. For example, formal V&V cannot prove that a model will never yield an incorrect prediction, but it could prove that no possible mapping of inputs to outputs is susceptible to adversarial perturbations to input values. When defining anomalous behavior in formal terms, we generally refer to model states corresponding to feasible model states denoting fault modes that lead to system failure [1].

It is possible to build models with more formal approaches, but it is uncommon. Expert systems are an example of a type of AI model that is built by constructing ontologies comprising many rules and propositions, and they were the subject of formal V&V [145, 146]. The field of AI has largely moved past expert systems in favor for statistical learning methods. Some have explored synthesizing models with formal methods, but the utility of the resulting models is unclear as they are not competitive with statistical machine learning models [23, 141].

What is the purpose of testing trained AI models?

While the standards to which models are tested varies by application domain, characterizing the reliability of AI systems prior to deployment is responsible engineering practice. The following clusters represent common purposes of model testing that are related to the work in this thesis.

Characterize generalization of the learned policy

AI must account for noise that is present in real-world systems, and comparing the predictions made by the model against ground truth labels is one way to empirically assess the deployment-worthiness of a model. Engineering desiderata often include overall accuracy of model predictions on data unseen during training and confusion matrices which measure the rates of type errors. Assessing whether a model is overfit to data determines if the model structure must be pruned, or if manipulations to the training data are required to give the model a better chance at learning generalizable structure [75].

Characterize quality of predictions

The are multiple ways to measure the quality of predictions coming from an AI system. Confidence measures describe the extent to which consensus emerges among multiple factors in the decision making process, the extent to which a data point is separated from a decision boundary, or the degree of error in regression tasks. Low quality predictions may not be trustworthy [70, 128, 198, 203], and may require additional human adjudication [45, 179, 180] . Understanding the conditions under which a model produces high quality and low quality predictions fosters a sense of trust and reliability [76, 88, 164, 183].

Tune model parameters

Testing trained models also provides a quantitative signal that can be used in an iterative design process, to generate better data [75, 74], train better models, or inform model parameter selection. In real-world contexts, the objective of injecting AI into an existing system is to reduce the rate of FPs [134, 133]. Tuning prediction functions to minimize FPs involves setting a threshold for the discriminator such that balance between the cost of FP and False Negative (FN) errors is achieved.

New areas of research are emerging in AutoML [84] and MLOps [2, 188] which seeks to transfer best practices from DevOps [55] into AI systems. Some of the goals of tuning model parameters and selecting the best model from a set of trained

candidates involve automating the V&V process for AI in order to reduce hurdles to developing deployment-worthy AI systems.

Estimate the extent of model adherence to design specifications

Straightforward application of ML in risk assessment for criminal justice, guiding clinical decisions, and assigning financial assistance, may lead to inflicting harm to humans. Low error rates do not convey the potential impact of individual errors [36, 129, 42], and uniformity of error analysis can disproportionately disadvantage, or privilege, select subpopulations in data. It has been shown that the use of ML to support and automate fair decision-making can lead to disparate impact [8, 4], and it may not only reproduce but also compound societal discrimination [46].

Fairness considerations are important specifications across all stages of the AI design pipeline [136]. Methods to measure fairness include internal auditing [156], model selection [129, 42, 36], and human-AI collaboration [179, 180, 45]. Furthermore, recent research has highlighted the challenges of predictive multiplicity [129] and underspecification [42], and it has been shown that different models that exhibit the same accuracy on fairness metrics may behave entirely different for different subpopulations [36]. Individual fairness [54] is one of many formulations of algorithmic fairness that have received active attention by the research community in recent years. It denotes a class of fairness metrics that measure whether similar individuals are treated similarly by a ML model. Multiple definitions of individual fairness [15, 151] and multiple ways to define the metric used to select candidate pairs [96, 202] have been proposed.

Verify model adherence to design specifications

Trust in machine learning models in high-risk and safety-critical applications often requires verifiable conformity with design specifications and robustness to small variations in input due to unforeseen interference, sensor noise or malfunction, and adversarial attack [109, 169, 175, 187]. Deep neural networks, while they offer high predictive power, are particularly prone to these kinds of vulnerabilities [182]. As a result, there has been significant study of both verifying properties of and generating adversarial attacks against deep networks, including works from [56, 94, 155, 109, 171]. Similar verification of adversarial robustness within trees and tree ensembles has begun to receive greater study in recent years [31, 103, 104, 157, 184, 187, 185]. Some work focused on verifying that trained models adhere to more domain-centric specifications such as expert knowledge [25]. Other work has been done to verify that a model adheres to different notions of fairness [37, 99].

What are common methods for V&V of AI systems?

There are many ways to test AI systems, and we reduce the scope of methods to those which are most relevant to work in this thesis.

Receiver Operating Characteristic (ROC) curves

ROC curves describe the empirical performance of a trained, binary classifier on a set of yet unseen data over a range of possible predictive threshold values [24]. An ROC curve represents an engineering tradeoff between minimizing the rate of FPs and maximizing the system ability to detect TPs [60]. The area under the curve, is a dimensionless quantity that may be used to compare the quality between a model and an uninformed, random predictor [24, 44, 82]. ROC curves are used across various contexts including characterizing the utility of diagnostic tests in clinical medicine [113, 81, 83, 131] and tuning parameters of learning models [60, 61].

Cross Validation

Cross validation is a method by which model performance on reserved data is used during the training process [170]. It is typically implemented to reduce the risk of overfitting to training data. There are multiple ways to implement cross validation when training a model, including k-fold [13, 159] and leave-one-out [138, 159] cross validation measures. Models that are trained with a cross-validation procedure tend to exhibit higher average performance than a single classifier trained on all data [13, 49, 138, 159, 170].

Metrics for prediction quality

Many metrics exist for determine the quality of a prediction, or the quality of a particular learned component of a model [49]. For correctness of predictions, metrics such as accuracy and F1 score describe performance of models trained for classification and mean squared error is a common metric for models trained for regression. Many additional metrics are implemented in open-source machine learning Application Programming Interface (API)s [150] and developers of AI systems can pick metrics that are best suited for a particular application domain.

Model Checking

Model checking refers to methods that verify whether all inputs and outputs of a finite-state system adheres to critical specifications. Early work involves verifying systems such as circuits or software programs where the specifications under consideration

are temporal in nature [39], but the search space quickly becomes intractable as the system under consideration grows. Ordered, binary decision diagrams [26] offered a compact representation of a base system that served as input to model checking algorithms. Bounded Model Checking reduced the search space for the verification tasks by bounding the length of an acceptable counterexample. This prevents model checking algorithms from needing to test program traces that exceed a certain length threshold and this reduces the scope of the search space.

Formal verification of properties on stochastic systems is difficult as the formalism must take non-determinism into account. Statistical model checking is another extension of model checking that leverages ideas of statistical hypothesis testing to provide an estimate of how likely it is that a stochastic model adheres to design specifications [38, 118, 160, 174]. Methods for sampling data and system traces to test the model involve Monte Carlo simulation [77, 147, 174] and Importance Sampling [7, 98]. The structure of design specifications for statistical model checking typically involves estimating the rate at which a property is satisfied, e.g. *95% of system executions must halt within 1 second*.

While the probabilistic estimates provided by statistical model checking are not formal guarantees, they can be calibrated to be sufficiently rigorous depending on the needs of the domain. Statistical model checking is still a quite popular and useful tool when underlying structure of the system in question is not known and developers have access to data generators which can run execution traces and verify that each test satisfies desired specifications [1, 86, 144] . Statistical model checking may be applied to AI systems, or more commonly, used to check complex cyber-physical systems comprising multiple components, where formalizing each component is either infeasible or prohibitively costly. Other work in statistical model checking attempts to extend formal approaches to handle non-determinism of a system. Stochastic Satisfiability Modulo Theories (SSMT) [58, 65, 71] offers a formalism for reasoning directly about a probabilistic automata, which mitigates the data or program trace requirement for hypothesis testing.

Formal verification of robustness in neural networks

There is an extensive recent history of satisfiability-oriented approaches to neural network verification. In a comprehensive analysis of SMT solvers existing at that time, [155] conclude that SMT can solve some non-trivial problems, but that the overall verification process for realistically sized models remains an open challenge. [171] introduce special deduction routines to the iSAT3 solver to verify networks trained for a variation of the inverted pendulum control problem, but find that, despite a speedup of several orders of magnitude, the general problem of safety verification remains

unsolved. `Planet`, a solver proposed by [56], more broadly considers networks with piecewise-linear activations using a global linear approximation with SMT or ILP (Integer Linear Programming) with additional original techniques grouped around a SAT solver that searches activation phases for nodes in the network. `Planet` is tested in case studies involving simulated vehicle collisions and MNIST, which we adapt for our own experiments in the context of tree ensemble verification as presented by [184]. The verification framework DLV (Deep Learning Verification) of [94] improves upon vanilla SMT by limiting and discretizing manipulations to the input and using a layer-by-layer propagating analysis. Under certain assumptions, adversarial robustness analysis is feasible for small images, but remains prohibitive for larger ones. Other works achieve good performance by making approximations. Leveraging the local convexity of networks with piecewise linear activations, [10] formulate local adversarial robustness as a tractable linear program by constraining search to convex regions. This produces approximate, but objective measures of robustness properties. [53] formulate verification as an optimization problem and solve a Lagrangian relaxation to upper-bound the worst-case violation, resulting in a flexible and sound, but incomplete verification strategy.

Reluplex - a ReLU simplex theory solver for SMT

Katz et al. [109] present ReLU Simplex Theory Solver (Reluplex), which comprises a sound and complete theory solver for neural networks with ReLU activation functions. They extend the simplex algorithm to handle the piecewise-linear nature of ReLU. Reluplex has a SAT core which is responsible for finding satisfying assignments which are then subsequently checked against additional theories including those for ReLUs as well as Real numbers. They show verification on networks of 6 layers and 300 ReLU nodes used for policy compression in an ACAS context, a study which we repeat using trees for comparison. Besides a difference of target model class, our work differs from Reluplex in that we do not require checking against any additional theories. This allows our methods to solely leverage SAT technology whereas Reluplex, and others for that matter, use a SAT core that requires more expressive logics to encode all necessary constraints. Experimental evidence in baseline comparisons in an ACAS context in Chapter 5 suggest that eliminating the need for more expressive theories speeds up the verification task greatly, although this speedup is tempered by the fact that it is hard to compare verification times between two different model classes.

Formal verification of robustness in tree ensemble models

Decision tree ensembles represent another class of learning models that are the subject of study in verification via Mixed Integer Linear Programming (MILP) [103], Equivalence Class Checking [184, 187], and SMT [11, 169]. Tree ensembles still require verification of design requirements and frequently suffer from issues like adversarial vulnerability [31][35][57][103]. The topic of adversarial robustness verification in trees and tree ensembles has begun to receive greater study in recent years. [103] employ a MILP solver to produce minimal adversarial perturbations for summing ensembles of trees and supplements the training set with generated adversarial examples to improve model robustness. Since the solving time can be extreme for complex models, they also provide an algorithm to produce approximately minimal adversarial perturbations. [31] study the complexity of the problem in greater detail, showing that it can be solved in linear time for single tree models and that it can be reduced to a maximal clique problem for ensembles, which leads to a polynomial algorithm for low-dimensional problems as well as an algorithm for lower-bounding robustness up to hundreds of times faster than testing precisely with MILP.

Several works apply SMT solvers directly for verification of tree-based models. [11] present trees as a verifiable alternative to neural networks in a reinforcement learning context by training decision tree policies guided by a more complex neural network. They train tree policies for two games and show that they play perfectly, have manageable size, and can be verified for various control properties in a matter of seconds using SMT. [57] focus on gradient boosted ensembles and use SMT for adversarial robustness testing, finding that some cases time out when the model is large. [169] use SMT for more general verification of output properties in decision tree ensembles and enumerate input spaces that violate the property so they can be filtered out.

VoTE - an equivalence class Verifier of Tree Ensembles

Törnblom and Nadjm-Tehrani offer comprehensive tools for verification of tree ensembles without using SMT [184, 187, 185]. They use a search strategy that partitions the input space of the model, explores all feasible path combinations, and computes equivalence classes to compare against requirements. The first, VoRF (Verifier of Random Forests), from [184], verifies properties of input/output mappings for decision trees and random forests. In two case studies based on those from [56], using data from a vehicle collision simulation and from MNIST, they show VoRF to be scalable for low-dimensional data for forests of up to 25 trees of depth 20. The second, Verifier of Tree Ensembles (VoTE), from [187], extends the methodology to other tree

ensembles, such as those employing gradient boosting, and improves performance by altering the search strategy.

Our methods differ from VoTE in that we use a SAT formalism. We conjecture that leveraging SAT allows us to impose constraints that a SAT solver may use to learn new conflict clauses and block potentially large subtrees of the search space from consideration. VoTE on the other hand, enumerates the hyperrectangle (HR)s that form the trained model's partition of $\mathbb{R}^n$, and checks that each HR satisfies a given specification. Experimental evidence provided in Chapter 3 suggests that our approach is more efficient when the size of the ensemble is large. Enumerating HRs and checking directly may be more efficient than a SAT formalism for small ensembles, but as we show with experimental evidence in Chapter 4, there are potentially significant savings for both larger ensembles or harder specifications.

1.4.3 Limitations of Current Practice

We briefly summarize the limitations of current practice to show the gap in literature that this thesis work addresses.

Statistical V&V alone is not well suited for critical domains, where trust is a precondition to adoption of an AI system

Sampling techniques only yield data that is a projection of reality and it is impossible to generate a comprehensive set of data for testing without knowledge of the structure of the model. Determining the deployment-worthiness of a trained model by testing its response to inputs runs the risk of biasing V&V toward the the data that is available. Low error rates do not show how the error manifest. The possibility remains that an AI system cause easily preventable harm due to a fault mode that went undiscovered during testing before deployment.

Making the jump from extraordinarily high probabilistic confidence in safety to provable certainty requires a lot of extra effort that may not be justifiable in terms of time and development costs for many applications of AI. However, reasoning in contractual terms about the strengths and weaknesses of a model satisfies some of the XAI desiderata for good explanations, and explanations can form a basis for increased system trustworthiness. When a possibility for catastrophic failure is present, such as when the rights and freedoms of a human being are at stake, supplementing statistical V&V with formal V&V provides the necessary proof that easily preventable harm will not be done.

For critical contexts, it may be that classification error is preferable to violation of a particular design specification. For example, it may be preferable for a trained

model to make an error in choosing to deny or extend credit to an applicant than for the model to exhibit unfair bias for otherwise identical applicants that only differ by race. In a clinical context, it may be desirable to produce additional, unnecessary false positive predictions than it would be to violate simple safety rules by yielding a false negative, where missing an alert for a critical instability may result in grave outcomes for a patient under intensive care. By solely testing models with statistical methods, we may miss otherwise undesirable model behavior that developers and end users of AI systems should be made aware of the strengths and weaknesses of a trained model.

XAI often requires humans to perform an error-prone, verification task

Provided with explanations of model behavior, a human arbiter ultimately must perform a verification task; *does the model behave as expected*? The answer to this question often determines whether a human acts on the recommendations provided by an AI system. A notably missing criterion in the XAI desiderata for good explanations is that the explanations be correct. Many works provide evidence to stakeholders that is primarily of confirmatory value. This may lead humans to trust an AI system for misguided reasons, which could have grave consequences in clinical contexts [72]. Human judgement is always prone to error, no matter the utility of explanations we can provide.

We conjecture that another desirable property of explanations in the context of XAI is that they should be independently verifiable by the person receiving the explanation. We claim that by providing contracts of model behavior, the formal proof is independently verifiable by a machine. Our work attempts to reduce the role of the human in deciding if the model is behaving as expected by posing the question, *can the machine perform the verification task itself*? The machine can draw from knowledge of its internal components whereas a human would be expected to abduce the model's inner workings. Certificates and their proofs can serve as a type of trustworthy explanation that has desirable properties and answers existing questions about real-world AI systems.

Formal V&V methods are too sterile for real-world applications of AI

While model verification is a popular topic of growing interest in the formal methods and automated reasoning community, the intended audience for the work are other researchers in that community. Showing that a property is satisfied or violated tends to be sufficient, and no work reported in literature is examining what conclusions developers and users of AI systems should draw from the certificates. If stakeholders

of AI systems do not know how to incorporate the output formal methods into their existing pipeline, they will not do so. We do not find existing work showing how certificates can be leveraged to improve AI systems or to serve as a type of explanation that characterizes the trustworthiness of a model.

This may be due to the fact that existing techniques for verifying models do not scale well. Simplifying assumptions are made frequently. Datasets used to train the models in question often very few features or are restricted to toy examples from publicly available benchmarks. Desirable model classes often undergo structural simplifications in order to fit into popular formalisms, but this results in verifying properties of an approximation of the base system rather than a faithful abstraction. Across many works, the scope of the properties in question is restricted to local neighborhoods, where a vast majority of the input space remains unchecked. Verifying global properties of non-trivial models remains a challenge that is intractable across different formalisms.

A lack of a method for verifying that models adhere to expert knowledge

Expert systems were studied extensively in AI in the 80s and 90s. Since the field has transitioned to more often leveraging statistical techniques, there is not as clear of a way to incorporate expert knowledge into an AI system. The most common way to do this is through asking experts to label data through an active learning framework [33] or to generate labeling functions to weakly supervise the learning task [19]. The hope is that the model will absorb structure during training that reflects expert knowledge. While this gets domain experts involved in the development at the beginning of the AI pipeline, there does not, to the best of our knowledge, exist a standard approach for integrating domain expert knowledge into the V&V process for models they will ultimately deploy.

Most literature in the field of formal verification of AI models focuses on model-centric properties of models, such robustness specifications. When the only people who are capable of expressing design specifications for AI systems are formal methods people, it is hard to integrate formal V&V into existing data pipelines in AI contexts. Our goal is to remove some hurdles to formal V&V such that domain experts and even users of AI systems can be involved in the testing of models rather than just the training of models. Providing blueprints for types of specifications that can be formally verified for a particular model class helps others know how they can use the tools and this set of blueprints can always increase as new design specifications are identified and formalized.

1.5 Our Approach

Our approach differs from current practice as we intend to increase trustworthiness of an AI system with proofs of verifiable conformance to design specifications. This work demonstrates that we can obtain explicit operational conditions under which a trained model satisfies or violates critical design specifications.

Model-centric verification of ML models

Prior work focuses on using formal methods to increase trustworthiness of AI systems, and early successes in AI almost exclusively leveraged symbolic frameworks. As statistical ML models grow in popularity, relatively little work is focused on verifying properties of models produced by statistical methods. Most work focused on increasing trust in statistical models focuses on making simplifying approximations to the model structure, extracting patterns from data, or simply justifying model output to users. Model-centric verification will produce explanations for model behavior that are provably true as a consequence of the learned decision-making structure itself.

Our approach to increasing trustworthiness of AI systems is to formally verify model adherence to critical design specifications. This allows developers and users to understand the specifications that they impose on trained models (e.g. the model must always be fair) and to trust resulting certificates that are the product of automated reasoning in a formal system. Model-centric verification provides assurance to all stakeholders that a trained model does what it is supposed to do.

Tree ensemble model class

A goal of this thesis is to investigate if formal verification of trained models can answer a wide range of existing questions about real-world systems, and this can be accomplished by demonstrating with any particular model class. Our experiments could be replicated for any formal system that returns certificates of satisfiability. Other model classes, such as neural networks, are the subject of ongoing research in verifying properties of trained networks.

We select voting, tree ensemble models as model class of study in this thesis. Tree ensembles are well-studied and hard to beat for learning tasks on tabular data. They are also amenable to description with propositional logic, which makes them relatively easy to express in a variety of different formalisms. Tree ensembles are a natural starting point to explore the utility of our methods with a less complex model class that remains tractable for verifying more interesting properties of models fit to data than is currently tractable for more complex model classes, such as neural networks.

A SAT formalism for the model, data, and specifications

We propose a novel SAT formalism for voting, tree ensemble models. While synthesis of decision trees has been studied within the SAT formalism [141], to the best of our knowledge, no prior work has studied ensembles of decision trees within the SAT formalism. We make the connection that SAT technology and the structure of tree ensemble models are very well suited for one another. The key insight is a discrete representation of voting tree ensembles that mitigates the need for the numeric theories of SMT. SMT is effective for verifying trees [11], but has limitations when applied to large ensembles [57, 169]. We show that, in practice, the use of SAT makes verification scalable to large ensembles and allows us to accomplish tasks that, to the best of our knowledge, are intractable for existing methods, including Local Adversarial Robustness (LAR) testing with high-dimensional data and verification of Global Adversarial Robustness (GAR) in general. Additionally, the proposed methods are sound and complete and do not rely on approximations or restrictive assumptions, other than that the ensemble output is aggregated by voting. Creating a SAT formalism for our work means that we will continue to benefit from improvements in SAT solver performance, which has been growing past what was theoretically considered possible in the last decade.

A collection of critical design specifications

Existing work focused on verifying robustness of ML models exists but has yet to scale to problem instances of real-world relevance without making simplifying assumptions that nullify claims to completeness. Works verifying critical design specifications other than robustness are very limited.

Our SAT formalism enables the verification of model adherence to previously intractable design specifications. We explore model-centric design specifications including robustness Local Adversarial Robustness (LAR) and fairness Global Individual Fairness (GIF) as well as domain-centric design specifications such as Safety-Paramount Engineering Constraint (SPEC)s. Each verifiable specification provides evidence that model-centric verification can answer a diverse set of existing questions that stakeholders typically ask of AI systems.

Incorporate certificate info into the model selection process

Existing work in formal verification of AI systems tends to stop once the certificate is obtained. This may be due in part to the fact that formal verification is considered an upfront cost in the design process, so time to verification is less critical than the completeness of the process. This typically results in a delayed feedback loop

that makes it difficult to incorporate certificates into the development phase of an AI system itself rather than obtaining certificates as the final step towards model deployment.

Our work shows how that information can be incorporated into existing AI design processes, due to the efficiency of verification for our chosen model class and formalism. Collections of certificates serve as a type of provable explanation that can be provided to stakeholders to reveal previously hidden structure in the way the models produce good or bad behavior. Our approach will allow developers of AI systems to select models that not only exhibit good overall accuracy on yet unseen data, but also exhibit high degrees of adherence to critical design specifications. We enable a pareto-optimal selection of models and model parameters such that performance of the model cannot improve without violating the specifications.

It is important to note that the verification process is independent from the training process. Our methods do not affect the overall accuracy of the learned model. Instead, our methods are meant to provide previously unavailable insight into the behaviors of trained models.

Chapter 2

Tree Ensemble Accreditor (TEA)

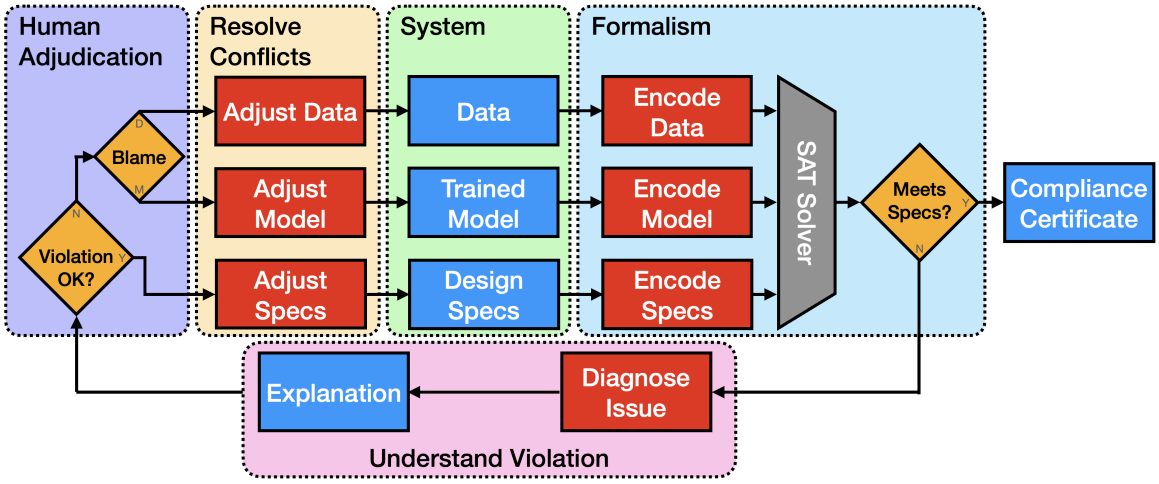


Figure 2.1: The Tree Ensemble Accreditor (TEA), our verification pipeline

We present TEA, our verification pipeline that features a novel SAT formalism for trained tree ensemble models. The individual components the make up the pipeline are connected in the flowchart in Figure 2.1. We briefly describe each and provide references to where more details can be found inside this document for each component. Background information on decision trees and tree ensembles can be found in Appendix E.2.1 and information on SAT can be found in Appendix E.2.2. This chapter details our SAT formalism for encoding trained tree ensemble models and encoding data. We also summarize our existing repertoire of design specifications that are integrated into the TEA pipeline. Formal details for encoding particular design specifications are reserved for subsequent chapters to reduce the number of references back to this chapter.

System

Our repertoire of TEA-compatible design specifications include: Robustness (see Chapter 3), Smoothness (see Chapter 4), Fairness (see Section 4.4), Safety (see Chapter 5), Monotonicity (see Appendix B), and Interpretability (see Appendix C).

Experiments in this work focus on featurized, tabular data with ordinal attributes. Attribute data types may be real, integer, or categorical. Ordinality can be maintained by converting any non-ordinal attributes into one-hot features.

We choose to focus our work on the voting tree ensemble model class for a few reasons. They are well-studied, hard to beat for learning tasks on tabular data, and comprise decision trees that are an interpretable model class when depth is shallow. Trees represent a hyperrectangular partition of $\mathbb{R}^n$ where each HR maps input points to a singular output. This means that operational regions over inputs where the ensemble satisfies or violates specifications may always be described with a collection of simple range rules in the native feature space. Decision trees and ensembles are popular model classes for decision support systems in real-world contexts, and human decision-making logic is often easy to express in a decision tree structure. This leaves the possibility open that TEA could be used to verify properties of sets of human-made decision rules, which could broaden the potential applications for TEA. Perhaps most importantly, verification is a hard problem and the piece-wise constant nature of the decision boundaries learned by tree ensemble models makes them amenable to description with lower-order logic than is be required for other model classes that comprise linear or non-linear elements. The unique properties of voting tree ensembles makes them an ideal model class for this work.

Formalism

Not all model classes can be efficiently expressed in a SAT formalism. In some cases, it is impossible to encode popular model classes in SAT logic, such as Deep Neural Network (DNN)s. Multiplication, which occurs for activation energies, is impossible to express efficiently in SAT. It is also not clear how one would encode linear constraints of the form $Ax \leq b$. Since the purpose of this thesis is to show how formal guarantees may supplement probabilistic estimates, we wish to leverage the most efficient formalism possible.

Decision trees are immediately amenable to description in SAT logic due to their axis-aligned structure. This eliminates the need to involve more expressive logics, which tends to result in longer time to verification as well as introducing the possibility that a particular formula be undecidable. To the best of our knowledge, we are the first to propose verifying properties of voting tree ensembles with a SAT formalism.

Decision trees have been studied in the context of SAT to synthesize the shallowest tree possible [141], however, verifying properties of independently trained decision trees or ensembles of decision trees has not been reported in literature.

SAT is a desirable formalism for a few reasons. SAT is *sound* (what is provable is true)[18, 9] and *complete* (what is true is provable)[41, 120, 107]. SAT has been closely studied since the 1970s [106] and is an industry standard for system verification [43, 79]. An active research community is focused on improving the technology [97, 68], meaning that leveraging SAT as a formalism carries the benefit that the technology continues to improve. Similarly, the input for open source SAT solvers is standardized in the DIMACS format [18], meaning that different solvers can be deployed depending on which has the fastest solving strategy for a particular problem class.

There are potentially many ways to encode a decision tree in propositional logic, and our encoding strategy differs from [141]. Our formalism uses an ordinal encoding strategy, which means that feasible model states are enforced by constraining order among active threshold values along the same attribute (i.e. $1 < 2 < 3$). This is a useful encoding strategy as it means that TEA certificates will apply to a contiguous, hyperrectangular range over inputs. This differs from other formalism strategies where proofs may only hold for a single point rather than over an entire range of possible values. The order encoding also makes it easy to integrate additional constraints into TEA that apply to the AI pipeline but not the trained model. For example, bounding the verification task to an arbitrary HR over inputs or encoding a disjunctive set of HRs that define a search scope within a neighborhood of any empirical data point. In other words, arbitrarily complex collections of one-dimensional threshold rules can be easily integrated into our encoding strategy.

SAT solver

Background information on how SAT solvers work can be found in Appendix E.2.2. Solving verification instances involves generating CNF logic with TEA for the model and design specification in question and then handing the formula to the SAT solver. Our open-sourced SAT solver of choice is CaDiCaL [17], but any other solver of choice could be plugged into our framework. We chose CaDiCaL due in part to its first place finish in previous SAT Races [16], as well as simplified and well-documented code repository. The inputs to the solver must be in DIMACS [87] format, which is a simple wrapper around CNF logic with comment lines as necessary. DIMACS CNF is a standard input form that allows formulas of SAT logic to be portable between multiple SAT solvers. The pipeline outputs certificates that determine whether the model adheres to all design specifications. For a brief explanation on the algorithms for solving verification instances, please refer to Appendix E.2.2. The contributions

of our work do not involve any advances in SAT technology. We focus on showing that V&V is possible for AI systems of application-scale, and that formal methods can provide answers to existing questions about AI systems.

To maximize efficiency in experiments throughout this thesis, we implement a custom interface for passing CNF generated in Python into CaDiCaL without needing to first write all CNF to file. The same interface also returns certificates, satisfying assignments, and proofs directly into our Python scripts. For large verification instances, I/O costs are non-trivial.

Certificates

Certificates are the proofs that state whether or not the model satisfies a particular specification. Either the formula is `UNSAT` or `SAT`. A `SAT` or `UNSAT` certificate are useful in different contexts. An `UNSAT` certificate provides a proof by contraposition, meaning the fact that no counterexamples exist lead us to conclude the formula is satisfied. Unsatisfiability (UNSAT) certificates comprise deductive proofs that highlight a logical contradiction in the formula. In case of a resulting `SAT` certificate, we obtain a proof by contradiction. Counterexamples that generate the contradiction can be provided using the strategy described in Section 2.3, and if desired, one may enumerate model configurations that produce adversarial examples by explicitly forbidding those that were previously discovered and checking satisfiability again. In this way, one can find and then sample from the set of all adversarial examples. This is an instance of what is known as the #SAT problem; how many satisfying assignments exist for a given formula?

Understanding Violations and Resolving Conflicts

When the model does not satisfy a particular design specification, a common follow-up inquiry is *why not*. We produce explanatory interpretations of the certificates that are intended to deepen a person's understanding of why the model violates the constraint. Depending on the verification task at hand, a SAT or UNSAT certificate may be interpreted to understand the violation.

Figure 2.2 gives an example of an interpretation of a SAT certificate for an illustrative experiment where we try to prove that the model will not produce a vote tally with a tie between two classes. The certificate details a counterexample that proves that the model violates specification. It shows the vote tallies, the active leaves among the decision trees within the forest as well as their label distributions. It also provides an operational range over the inputs which result in this model state is observed. For any values of x_1 and x_2 in the ranges $0.057 \leq x_1 \leq 0.201$ and $-2.100 \leq x_2 \leq -1.502$,

```
Ensemble Predictions:
PRED-0_RF0
PRED-1_RF0

Vote Tally:
Class 0: 3
Class 1: 3

Active Leaves + Distributions:
('LF4_DT1_RF0', [0, 4])
('LF4_DT2_RF0', [0, 12])
('LF8_DT3_RF0', [31, 7])
('LF10_DT4_RF0', [17, 1])
('LF2_DT5_RF0', [0, 11])
('LF6_DT6_RF0', [1, 0])

Operational Range for Behavior:
0.057161 <= X1 <= 0.200930
-2.100000 <= X2 <= -1.501524
```

Figure 2.2: A partial interpretation of a SAT certificate produced by TEA.

the vote tally will be tied 3-3 and the active leaves will be the same ones listed.

When a counterexample to the formula does not exist, an UNSAT certificate is returned. Summarizing the proof is one strategy to interpret UNSAT certificates that denote a specification violation, and conduct preliminary experiments which are presented in Appendix D. There we show how to provide explanations by extracting a MUS from resolution proofs.

Resolving identified violations closes the TEA pipeline loop. For SAT certificates, this can be as simple as reducing the scope of the verification task to exclude a particular counterexample, or to ignore previously identified feasible but implausible counterexamples. We conduct preliminary results to show that conflict resolution is also possible with UNSAT certificates. The work is presented in detail in Appendix D in order to keep the scope of the thesis focused. We can remove select data samples from training data if they are implicated in a specification violation. We also show how TEA can be used to generate adversarial data to fill gaps in data support. Changing the scope of a design specification may change the verification outcome. We also show preliminary evidence that it is possible to propose structural changes

to a trained model such that the model satisfies a previously violated constraint while also maintaining a consistent level of accuracy on data that it has seen before.

2.1 SAT Formalism for Voting Tree Ensembles

All verification tasks will be performed on trained, voting tree ensemble models for classification tasks. We can readily apply this formalism to models trained on data sets that are featurized as a collection of real-valued, integer, or categorical attributes. The conceptual leap that allows us to scale to verification of ensembles of decision trees is the implementation of a sequential counter [177] that we use to tally the votes cast by individual trees in the ensemble. A primer on satisfiability is provided in Appendix E.2.2. We include a detailed example for converting a trained decision tree into CNF in E.2.2.

To the best of our knowledge, this thesis work represents the first reported SAT formalism for voting tree ensemble models. Decision trees have been studied in a SAT formalism previously [14, 141], but these encoding strategies are not relevant for the properties we wish to verify about decision trees and ensembles comprised of decision trees. Narodystka [141] and Bessiere [14] both study the use of a SAT formalism to synthesize optimal decision trees; optimal in the sense of minimal depth that perfectly classifies a set of data. Their encodings describe strategies for checking whether a perfect accuracy, depth n decision tree exists for a set of labeled data. In contrast, our work is not synthesizing decision trees using logic, we are instead training tree ensembles with statistical methods and then verifying that those learned structures adhere to critical decision specifications. We develop our own novel encoding strategy because the strategies in [14, 141] possess many additional literals and clauses that are not necessary to verify a property of a trained decision tree, and we wish to be as efficient as possible, given the NP-hard worst case runtime of SAT. For example, [141] implements sequential counters at each node in the decision tree to ensure that the sum of active child nodes is equal to 1; such a constraint is unnecessary in our context given that we are taking a previously trained tree ensemble and then encoding it in suitable propositional logic.

To encode a voting tree ensemble, there are two classes of necessary constraints. *Tree constraints* govern the interaction between select literals associated with one single decision tree. *Ensemble constraints* govern the interaction between select literals associated with different decision trees. We first describe the strategies for encoding individual decision trees in logic, which represent all the tree constraints, including their decision and prediction logic. Then we describe all the ensemble level constraints that link those individual trees together in meaningful ways, including ordinality and vote-counting logic. Algorithms offer propositional level detail of our formalism; the logic must be converted to CNF before being passed to the SAT solver. This is a necessary but straightforward step that is left to the reader.

2.1.1 Notation

Table 2.1: Notation for the components of a decision tree

value	meaning
i	the node index
$\mathbb{B}$	the set of node indices for all branch nodes
$\mathbb{L}$	the set of node indices for all leaf nodes
$\ell(i)$	the index of the left child of node i
$r(i)$	the index of the right child of node i
$a(i)$	the learned splitting attribute at node i
$t(i)$	the learned threshold value at node i
$\mathbf{d}(i)$	the distribution of class labels from training data at node i
$p(i)$	the index of the mode in $\mathbf{d}(i)$

We notate a tree ensemble as $\mathcal{M}$, which comprises models $m_1, m_2, ..., m_n$. For a set $\mathcal{Y}$ of class labels observed in data, the voting random forest prediction for an input $\mathbf{x}$ is given as:

$$\mathcal{M}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \mathbb{I}\{m_i(\mathbf{x}) = y\} \tag{2.1}$$

Where $\mathbb{I}$ is the indicator function that returns 1 if the inside condition is true, otherwise returns 0. For simplicity in presenting our formalism, we assume that the prediction function of tree ensembles will yield the plurality vote for class label. Equation 2.1 can be changed to accommodate prediction thresholds. For example, in a binary classification setting, a positive prediction threshold value of 0.8 would require that 80% of the label distribution at the leaf node must be positive labels, else, the model will yield a negative prediction.

For algorithms in this text, we will adopt notation for components of a trained decision tree as listed in Table 2.1. Interpretations for the Boolean literals that encode necessary components into logic are listed in Table 2.2. Unless otherwise noted, capital letter variables denote logical literals. We try to keep notation clean by replacing multiple subscripts for a single variable by invoking functions that retrieve the information we need.

There are multiple possible ways to encode a decision tree in logic. For example, a decision tree could be represented as a collection of possible paths to each leaf node where feasibility means only one path may be active at a time. We choose to encode the ensemble with the finest resolution encoding as possible, which means that we are encoding all information about nodes in the decision trees rather than just

Table 2.2: Interpretations of Boolean variables for encoding voting tree ensembles

variable	true if
T_i	$x_{a(i)} \leq t(i)$
A_i	node i is active
$P_{c,m}$	model m predicts class c
$S_{j,m,c}$	class c vote tally among first m models is $\geq j$

information about each possible decision path. While this requires additional literals and clauses, one utility of our approach can be seen when multiple decision trees are combined to form an ensemble, where feasibility now describes joint configurations of all trees in the ensemble. Order, e.g. $\{x_i < t_a \implies x_i < t_b \mid t_a < t_b\}$, is known among all threshold values across decision nodes in the tree, and this can be exploited in the encoding. If we instead encoding decision trees as collections of possible paths, we would need to enumerate all feasible ensemble configurations prior to the verification task.

A visualization of the logic is juxtaposed with the graphical structure of a trained tree ensemble in Figure 2.3, which points the reader to relevant algorithms within this section. Our formalism can be broken down into five subcomponents that interact. Decision logic and prediction logic both encode decision tree constraints. Feature space logic, vote counting logic, and plurality logic all govern how individual trees interact with one another to produce feasible, ensemble behaviors.

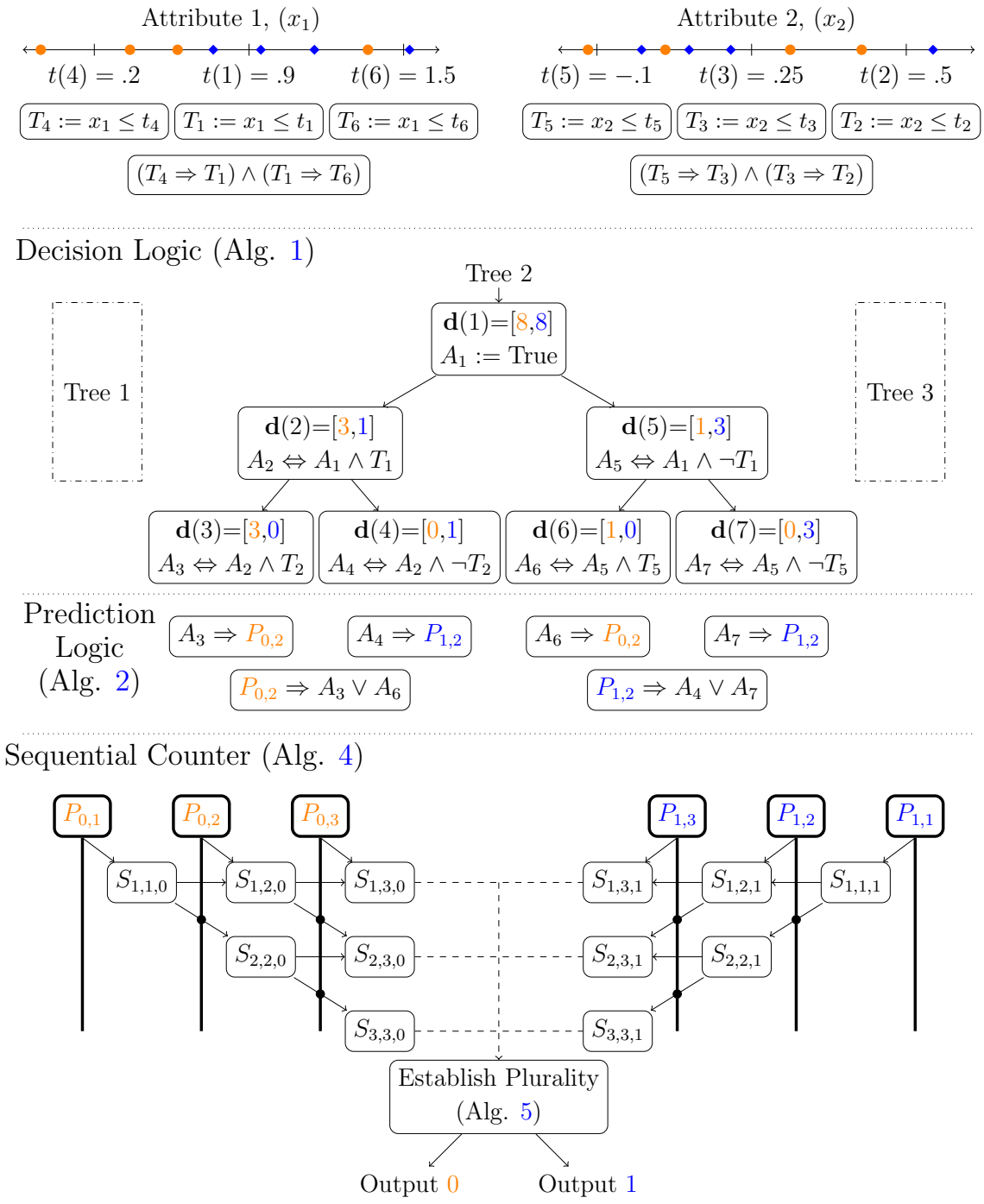


Figure 2.3: SAT formalism for tree (Tree 2) and ensemble (Trees 1-3) fit to data.

2.1.2 Decision logic

Algorithm 1: Encoding the Decision Logic of the Tree Ensemble

1 $\mathbb{M} \leftarrow$ indices of tree models within tree ensemble
2 **for** each $m \in \mathbb{M}$ **do**
3 $\quad \mathbb{B}_m \leftarrow$ indices of branch nodes in model m
4 $\quad r_m \leftarrow$ root node of model m
5 $\quad$ assert A_{r_m} `/* root nodes of decision trees are always active */`
6 $\quad$ **for** each $i \in \mathbb{B}_m$ **do**
7 $\quad\quad$ assert $A_{\ell(i)} \iff A_i \wedge T_i$ `/* left if <= to threshold */`
8 $\quad\quad$ assert $A_{r(i)} \iff A_i \wedge \neg T_i$ `/* right if > than threshold */`
9 $\quad$ **end**
10 **end**

Each tree possesses a connective graph structure, the decision logic, that links the root node to all leaves. Algorithm 1 encodes properties of each branch node. This relates parent nodes to their child nodes, and encodes each branch node's splitting criterion (lines 7 and 8).

Algorithm 1 describes the strategy for encoding a tree fit to real-valued attributes, but it is also possible to change the strategy for categorical data. One key consequence of categorical data is that the attribute is no longer ordinal. This would require changes to downstream constraints that govern the feasibility of attribute value assignments between decision trees. Without loss of generality, we can express any decision tree with the strategy in Algorithm 1 if we break up high-arity attributes into a series of binary, integer features. With binary, integer features, constraining ordinality is irrelevant, so this strategy works for both numeric and categorical features in data. The cost of doing so is that additional literals and clauses will be needed to represent the tree, and this could become prohibitively expensive for some high arity, non-ordinal, categorical attributes, such as Zip Codes. We have used this strategy for attributes with arity=40, where the logic remains tractable, but do not run comprehensive tests to assess the cost of this encoding strategy for non-ordinal, categorical attributes.

An additional assumption that we are making for the entirety of this document is that all branch nodes of a decision tree make a single split, or in other words, have two children. These are known as binary decision trees. It is possible to have a non-binary decision tree, where there are more than two children at a decision node. In such cases, we may use the strategy in Algorithm 1 without loss of generality as a

single branch with multiple splits can be expressed as multiple branches with single splits over consecutive depth levels of the tree.

2.1.3 Prediction logic

Algorithm 2: Encoding the Prediction Logic of the Tree Ensemble

1 $\mathbb{C} \leftarrow$ set of unique class labels
2 $\mathbb{M} \leftarrow$ indices of tree models within tree ensemble
3 **for** each $m \in \mathbb{M}$ **do**
4 $\quad \mathbb{L}_m \leftarrow$ indices of leaf nodes in model m
5 $\quad$ **for** each $i \in \mathbb{L}_m$ **do**
6 $\quad\quad$ assert $A_i \implies P_{p(i),m}$
7 $\quad$ **end**
8 $\quad$ **for** each $c \in \mathbb{C}$ **do**
9 $\quad\quad$ assert $P_{c,m} \implies \bigvee\{A_i \mid i \in \mathbb{L}_m, p(i) = c\}$
10 $\quad$ **end**
11 **end**

A leaf differs from a branch in that it has no child nodes. Interpreted in conjunction with the information in the branch nodes above it, a leaf node describes a partitioned subset of the feature space. The decision for what classification label to output for this subspace is made at this node. For voting tree ensembles, the most common strategy is to output the mode of the label distribution at each leaf node, described in Algorithm 2 (line 6). Furthermore, in the event of ties between multiple class labels, breaking in predetermined order is a common choice.

This represents one possibility for how trees cast their votes within the ensemble. Other verification tasks may warrant other design choices for how to yield a predicted class label at each leaf node. For instance, instead of breaking ties in predetermined order, perhaps we are interested in searching for model states where such ties manifest. In those cases, reporting both class labels implicated in the tie may be appropriate.

Additionally, the choice to output the mode of the label distribution can be adjusted to suit particular verification needs. Support and purity requirements are two criterion that form useful gating functions when yielding tree predictions. If we are only interested in verifying model states yielding high support predictions, we may impose minimum or maximum support requirements. Similarly, minimum or maximum purity requirements would limit the verification task to model states where high or low confidence predictions are made. Here, confidence is defined as the homogeneity of the training label distribution in a particular leaf node. Any extra constraint

layer can be worked into the model encoding by expanding line 6 to include a conjunction of other parameters such as *meets required support range* or *meets required purity range*.

Line 6 of Algorithm 2 states that if the model makes a particular prediction, then at least one leaf node that yields this prediction must be active (line 9). This ensures that at least one leaf is active (assigned `TRUE` by the SAT solver) in the logic encoding, which is always true, regardless of other design choices for how to yield a prediction at each leaf. Ordinality constraints are what prevent more than one leaf from being assigned active, as this would represent an infeasible set of inputs to the model.

2.1.4 Ordinality

Algorithm 3: Limiting the search space to feasible model states

1 $\mathbb{M} \leftarrow$ indices of tree models within tree ensemble
2 $\forall m \in \mathbb{M}$, $\mathbb{B}_m \leftarrow$ indices of branch nodes in model m
3 $\mathbf{I} \leftarrow \text{argsort}_i(\{t(i) \mid i \in \mathbb{B}_m, m \in \mathbb{M}\})$ `/* all idxs by ascending thresh val */`
4 **for** $att \leftarrow 1$ **to** number of attributes **do**
5 $\quad \mathbf{I}' \leftarrow (i \in \mathbf{I} \mid a(i) = att)$ `/* idxs of att's ascending thresh vals */`
6 $\quad$ **for** $k \leftarrow 1$ **to** $\text{len}(\mathbf{I}')$ **do**
7 $\quad\quad$ **if** $t(\mathbf{I}'_k) = t(\mathbf{I}'_{k+1})$ **then** assert $T_{\mathbf{I}'_k} \iff T_{\mathbf{I}'_{k+1}}$ `/* thresholds equal */`
8 $\quad\quad$ **else** assert $T_{\mathbf{I}'_k} \implies T_{\mathbf{I}'_{k+1}}$ `/* thresholds ordinal */`
9 $\quad$ **end**
10 **end**

Enforcing feasibility of model states for a tree ensemble is more complicated than enforcing feasibility of a single decision tree. For a single decision tree, feasibility means that only a single from root to leaf is active at any time. Naïvely enforcing one active path for all trees in the ensemble is not sufficient for maintaining ensemble feasibility. To illustrate, consider two decision trees with one active path from root to leaf and in each of those paths, there exists a decision node that makes a split along a select attribute, a_i. The select decision node in the first tree checks $x_{a_i} \leq 10$ and the counterpart in the second tree checks $x_{a_i} > 20$. While the paths may be feasible for each decision tree in isolation, these paths are incompatible with one another, because input x_{a_i} cannot simultaneously be less than 10 and greater than 20. Thus, additional logic is needed in order to maintain feasibility of the ensemble beyond any

logic that enforces feasibility of individual decision trees.

Since we can convert high-arity categorical attributes into a set of binary, integer attributes, we can limit our verification task search to the realm of feasible model states by enforcing ordinality of values along the same attribute. The new clauses necessary to enforce ordinality among all threshold values found at each decision node operate on the same threshold literals that were used to define the decision logic structure for all trees. Corresponding threshold literals denote which side of the learned threshold value $t(i)$ is active. Conjunctions of these half-spaces yield hyperrectangular subsets of the input where the model state is consistent. These HRs are also feasible in the sense that they represent areas defined by strictly one active leaf in each decision tree.

Such constraints need to be made at the ensemble level since each decision tree has no knowledge of the learned threshold values in decision nodes of other trees in the ensemble. Thresholds across all decision nodes are first ordered from least to greatest (Alg. 3, line 3). Then, for each attribute in the data, consecutive thresholds are ordinally constrained; if the value of attribute $\alpha < v$ then we also know that the value of attribute $\alpha < v + \epsilon$. In the case of equality between two threshold values, we need both thresholds to receive equivalent assignments, so the implication becomes a biconditional. This happens infrequently for real-valued data, but is more common for integer-valued or lower-arity attributes.

Ordinal constraints introduce a linear number of literals and clauses with respect to the number of nodes among individual trees within the ensemble. Arguably, this encoding has higher utility than directly encoding real values in more expressive logics. Resulting `SAT` assignments will yield explicit operational ranges over which encoded properties hold. Each of these ranges contains an infinite number of points, making our proofs more generalizable.

2.1.5 Vote counting

In a voting tree ensemble model, the model may only output a class label if that class achieves a plural victory among votes cast by individual trees, so we need to define logic to tally the votes. This means we need to implement a sequential counter, a form of an adder, in propositional logic. We implement a modified version of a sequential counter proposed by Sinz [178]. The bottom of Figure 2.3 provides a visual description that pairs with the formal description in Algorithm 4. A counter must be implemented for each class label found in data, and each introduces a quadratic number of literals with respect to the number of trees in the ensemble.

Barring edge case exceptions addressed in lines 4 and 6, there are two ways for the sum to be greater than or equal to j by model m for class c (line 8); either

Algorithm 4: Vote Counting Encoding

$N \leftarrow$ the number of trees in the ensemble
$C \leftarrow$number of unique class labels
for $c \leftarrow 1$ **to** C **do**
    assert $P_{c,1} \iff S_{1,1,c}$ `/* model` $m = 1$ `yields` c `iff sum` ≥ 1 `by` m `*/`
    for $m \leftarrow 2$ **to** N **do**
        assert $S_{1,m,c} \iff P_{c,m} \vee S_{1,m-1,c}$
        for $j \leftarrow 2$ **to** m **do**
            assert $S_{j,m,c} \iff (P_{c,m} \wedge S_{j-1,m-1,c}) \vee S_{j,m-1,c}$
        end
    end
end

the vote tally by the $(m-1)^{th}$ model is already equal to j so the vote cast by the m^{th} model does not matter, or, the tally by the $(m-1)^{th}$ model is one shy of j and the m^{th} model casts the j^{th} vote. Determining whether a majority emerges is a straightforward check of the assignment to $S_{\lceil N/2 \rceil,N,c}$. However, in some multi-class settings, majority may not be achieved, so we must ascertain a plurality consensus in the absence of a majority.

To the best of our knowledge, this is the conceptual leap that makes us the first reported encoding of a voting tree ensemble model in propositional logic. Encodings of decision trees in propositional logic have been studied previously [14, 141], but the aggregating of votes from multiple decision trees has seemingly never been studied before using SAT technology. As of 2019, tree ensemble models have been the subject of verification using SMT [169], but as the authors note, no other SAT/SMT based encoding of tree ensemble models has been reported to date.

2.1.6 Plurality logic

In binary classification tasks, a simple majority would be sufficient for determining the ensemble output. However, in multiclass contexts, it is possible for the ensemble to yield a predicted class label even if that class label does not have over 50% of the votes in the ensemble. Plurality, also called plural victory, is a term that describes the largest number of votes for a particular class label, and plurality can be thought of as the statistical mode of a distribution.

For the condition of plurality to hold for a select class label, c, there must be

Algorithm 5: Plurality Encoding

1 $N \leftarrow$ the number of trees in the ensemble
2 $C \leftarrow$number of unique class labels
3 $lo \leftarrow \texttt{max}(1, \lfloor N/C \rfloor)$
4 $hi \leftarrow \lfloor N/2 \rfloor + 1$
5 **for** $j \leftarrow lo$ **to** hi **do**
6 **for** $c \leftarrow 1$ **to** C **do**
7 **for** $c' \leftarrow 1$ **to** C **do**
8 **if** $c \neq c'$ **then**
9 assert $\neg S_{j,N,c} \wedge S_{j,N,c'} \implies \neg P_{c,\mathcal{M}}$ /* c votes $\leq c'$ votes */
10 **end**
11 **end**
12 $Q \leftarrow \bigwedge\{\neg S_{j+1,N,c'} \mid c' \in [C], c' \neq c\}$
13 assert $S_{j,N,c} \wedge Q \implies P_{c,\mathcal{M}}$ /* votes for $c \geq$ any $c' \neq c$ */
14 **end**
15 **end**

no other class label, c', that obtains a greater number of votes. Ties are broken in predetermined order when needed in some of our experiments, but are not broken in Algorithm 5 in order to keep the logic concise. If there are fewer votes for the select class than there are for an other class, then the select class does not achieve plurality (line 9). If no other class earns more votes than a select class, then it is a plurality (line 13). These assertions must be defined for multiple sum totals since we cannot know what the size of the plural consensus will be beforehand.

No new literals are defined in the process of determining plurality (Algorithm 5) since plurality can be established among the literals defined for the sequential counter. However, $O(NC^2)$ clauses are added with respect to the number of trees, N, and the number of classes, C^2.

Once plurality is established, the encoding of the entire tree ensemble, from inputs to outputs, is complete. A trained ensemble becomes expressible in propositional logic and later conjunctive normal form, making it amenable to the types of analyses available to constraint satisfaction problems.

2.2 SAT Formalism for Data

The most straightforward way to verify model adherence to design specifications at a single data point is to set the truth values of all ordinal threshold literals. If a threshold across a select attribute is less than the value of the data point in question, the threshold literal is set `False`. When a threshold is greater than a select attribute, the threshold literal is set `True`.

Often, verifying properties in the neighborhood of multiple data points simultaneously is of interest. We could extend the simple method in the last paragraph by performing new verification task for every data point and check to make sure that the property in question is satisfiable for all tested points. Such an approach would be biased toward the learned structure of the model, as we would only be testing the circumscribing HR around a single point, which may be of arbitrary shape and size. There would be no guarantee that the scope of the verification task would be equal across all data points.

A different approach to this challenge would be to simply test the model globally or to define a convex hull containing all data points and test resulting subspace. Without encoding a notion of data, we test the model for all feasible inputs. The drawback of this approach is that feasibility does not imply plausibility of inputs, and our formalism for tree ensembles will test all possible inputs regardless of how likely it is for an incoming sample of data to possess select attributes values. Models that otherwise would adhere to desired specifications may violate the property due to anomalous outliers that the model would likely never see in the real world.

We address the challenge of testing plausible inputs to the model by defining a formalism for data that integrates easily into our formalism for tree ensemble models. We represent each point of data with a circumscribing HR. The dimensions of the HR need to be set by either a human operator of TEA, or, it is possible to define these dimensions based on statistical dispersion metrics of the data set. HRs are easy to incorporate into the formalism for tree ensembles because once the literals are defined, they only need to be added into the ordinal constraints which constraint feasible model states.

Algorithm 6 describes the procedure for scoping a verification task over a disjunction of ζ-neighborhood HRs about each data point under consideration. On a conceptual level, our formalism is a formal proxy for the i.i.d. assumption that is a foundation for all AI systems. Models are not guaranteed to perform well if the distribution of data observed in test differs from that of training data. It may be unreasonable to expect properties such as robustness hold for input values that are sufficiently far away from any other point of data the model saw during training. This gives us a much tighter definition for *global* behavior than a truly global search over

Algorithm 6: Plausibility (logical proxy for i.i.d assumption)

$X \leftarrow$ data
$\mathbb{D} \leftarrow$ indices of data points of interest
$\delta \leftarrow$ vector defining neighborhood about each data point to verify
for $i \in \mathbb{D}$ **do**
    $\mathbf{x} \leftarrow X_i$
    for *att* $\leftarrow 1$ **to** *number of attributes* **do**
        $R_{i,att-\delta} := \mathbf{x}_{att} - \delta_{att}$`/* define new literal for lower bound */`
        $R_{i,att+\delta} := \mathbf{x}_{att} + \delta_{att}$`/* define new literal for upper bound */`
        $r_{i,att} := \neg R_{i,att-\delta} \wedge R_{i,att+\delta}$`/* input falls within att bounds */`
    end
    $H_i := \bigwedge\{r_{i,att} \mid \forall \text{ attributes}\}$`/* input falls within att bounds for all attributes */`
end
assert `TseitinCNF`$\left(\bigvee\{H_i \mid \forall i \in \mathbb{D}\}\right)$`/* See [186] */`
assert `Ordinality`$\left(\{R_{i,att-\delta}, R_{i,att+\delta} \mid \forall i \in \mathbb{D}, \forall \text{attributes}\}\right)$`/* See Alg 3 */`

an unbounded feature space.

A naïve encoding of the disjunctive set of HRs in CNF will result in a worst-case, exponential increase in the number of clauses. A Tseitin transformation [186] is required to express the disjunctive set of HRs into CNF logic by only introducing a linear number of additional literals and clauses. This strategy for formalizing data is flexible in the sense that it becomes easy to test the effects of adding or removing samples from data. If the model does not satisfy a property, we can shorten the disjunctive list of HRs by removing data points from consideration where the model violates a specification. Adding new data for verification simply requires appending additional HRs to the list.

2.3 Interpreting SAT Certificates

A SAT certificate presents a satisfying assignment to all literals such that the CNF formula evaluates to `TRUE`. Information such as ensemble output, active leaves, and ranges of satisfying input values may be extracted from the satisfying assignment by checking individual literal assignments. The process is straightforward for literals such as the ensemble's prediction, which requires interpreting a single literal.

Some parts of the solution require strategies for interpreting multiple literals simultaneously. Operational ranges of the input are encoded as HRs, which represent the intersection of the half-spaces formed by threshold constraints associated with all active decision nodes. Let ϕ be a satisfying assignment, which maps a literal to a truth value. Then the satisfying HR is given as

$$\text{Satisfying Hyperrectangle} = \bigcap_{\phi(A_i)=\text{True}} \left\{ \begin{array}{ll} \mathbf{x} \mid x_{a_i} \leq t_i, & \text{if } \phi(T_i) \\ \mathbf{x} \mid x_{a_i} > t_i, & \text{if } \neg\phi(T_i) \end{array} \right\}.$$

This satisfying HR may only tell part of the story. It denotes the contiguous range of inputs to the ensemble where constant output behavior will be observed. If there are additional constraints on the problem that are not directed at the ensemble itself, e.g. requiring $x_i \leq c$, then these additional constraints need to be interpreted as well to produce the final satisfying HR that meets all constraints on the stack.

Next, take the intersection with other input constraints of the verification task; these may be more restrictive than the satisfying HR recovered from the assignment. If the verification task includes a human-defined operational range over which to search, it is likely that this regions does not perfectly align to the learned structure of a tree ensemble. This intersection of HRs is also needed when interpreting the satisfying model states of multiple tree ensembles simultaneously, where the SAT solver tells us that two HRs intersect but this does not mean the HR necessarily overlap entirely. The result is the operational range of inputs over which a particular model behavior is exhibited. If desired, satisfying inputs, i.e. counterexamples that form a logical contradiction with the property being tested, can be sampled from the resulting subset.

Chapter 3

Verification of a Local Adversarial Robustness (LAR) Specification

When reasoning about trained learning models, we often refer to the learned *decision boundary* as an object of interest. It represents the change point where the model flips its prediction for a given input. For some model classes, such as logistic regression and support vector machines, distance to the decision boundary is easy to obtain for any given input. For other model classes, such as tree ensembles or neural networks, the decision boundary is more of an abstract concept and it is difficult to find the shortest distance to the boundary. The decision boundary is still important, because even model classes that cannot easily provide distance to the decision boundary for a given input will still exhibit decision boundary behaviors by producing different predictions for indistinguishably different inputs. This represents an opportunity for otherwise easily preventable harm to be inflicted through use of the trained model. Human decision makers cannot provide different labels for inputs that are indistinguishable to the human eye, so, we wish to verify that an artificial intelligence exhibits this same property.

Local Adversarial Robustness (LAR) describes stability of system outputs for all inputs within some neighborhood about a point [109, 184, 187]. We adopt the definition from [109] with slight modifications: a model satisfies $(\mathbf{x}, \delta)$-LAR at input point $\mathbf{x}$ if, for every $\mathbf{x}'$ such that $\{|x_i - x_i'| < \delta_i \mid \forall i \in \delta\}$, the model predicts the same label for $\mathbf{x}$ and $\mathbf{x}'$. A system satisfies LAR if it yields the same output for all inputs in the neighborhood. Verifying model adherence to LAR is a common task in literature, and multiple formalisms exist to accommodate a range of model classes [6, 109, 123, 184, 187, 197]. Most often, developers and users of AI systems are interested in verifying whether a model exhibits LAR within a neighborhood of each point in training or test data. This does not immediately tell us the level of

LAR the model will exhibit on yet untested data, but it does give us an idea of how consistent a model's output is in the local neighborhood of each tested point; a model should yield consistent output for imperceptible changes to input. This is a measure of reliability exhibited by the AI system in response to data fed into the model. The larger the neighborhood surrounding a data point over which the model exhibits robust behavior, the more confidence that developers and users of AI systems have that this particular prediction is stable, and that the model may yield similar levels of robustness for yet unseen points that are similar to the ones that are tested prior to deployment.

Preliminary experiments in the next section provide a visual interpretation for LAR certificates. We describe our methods, which, to the best of our knowledge, are the first SAT formalism to verify LAR for tree ensemble models. We provide evidence that our method scales to larger verification instances than those reported in literature. The efficiency of our formalism enables the incorporation of LAR certificates into the AI design process in novel ways. We apply our techniques to existing problems in a radiation threat adjudication context and discuss the impact.

Minimal distance counterfactuals can be obtained by binary search guided by LAR certificates. Counterfactuals explicitly detail how the input of the system would need to change in order to obtain a different output. Our approach differs from statistical methods in the sense that our counterfactuals are model-centric rather than data-centric. This means that we can provide counterfactuals even in input regions where there is no data support. A single counterfactual serves as a type of explanation for the smallest changes to input values that will result in a change in model behavior. A group of counterfactuals allows us to estimate the extent of LAR that the model will exhibit on yet unverified data points. Knowledge of the extent of robustness for select attributes in data could support some AI design decision making, such as establishing the level of precision required of sensor measurements in order to for the model to satisfy a LAR specification.

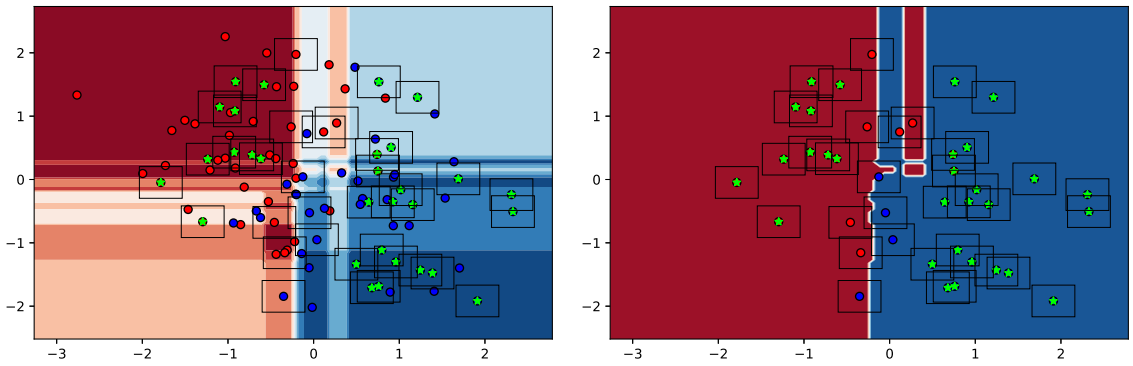
LAR certificates also support decision making during the model selection phase of the AI design pipeline. Faced with multiple trained models with statistically indistinguishable performance, we show that the extent to which each model satisfies LAR specifications may break ties and allow domain experts to pick the model that is optimized both for accuracy and reliability. We also examine the relationship between prediction thresholds and LAR. A Receiver Operating Characteristic (ROC) curve is a standard tool for tuning the prediction threshold when balancing the cost of TP and FP prediction outcomes. Each candidate prediction threshold affects the vote cast by leaves in a tree ensemble. We distinguish between correct and incorrect predictions; ideally, the model will exhibit robustness only when it is correct. Analyzing

True Positive Robustness Rate (TPRR) and False Positive Robustness Rate (FPRR) for each given predictive threshold yields a definition of LAR for which this robust-only-when-correct desiderata is best satisfied. The extent of conformance to this desiderata provides evidence that adversarial noise is not responsible for correct predictions, while adversarial noise may be responsible for incorrect predictions. The decision threshold can be chosen such that the rate of robustness that the model exhibits for correctly classified points is higher than the rate of robustness on incorrectly classified points.

3.1 Illustrative Example

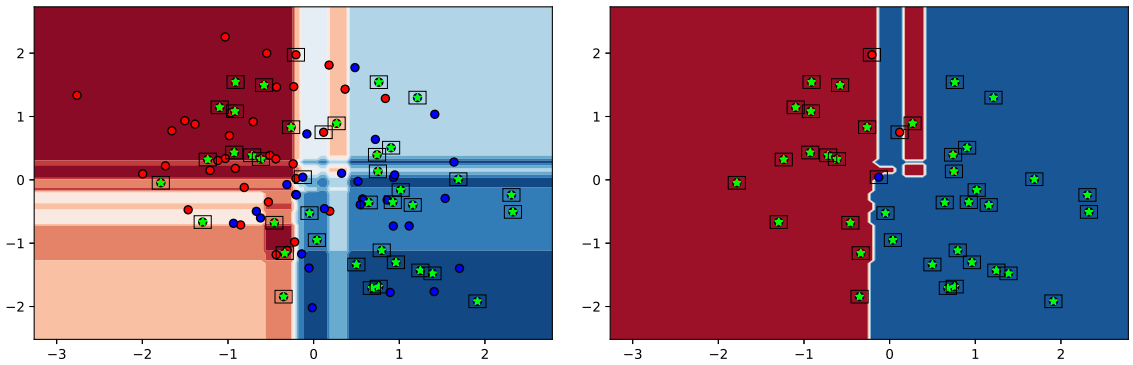
Illustrative examples of a $(\mathbf{x}, \delta)$-LAR certificate and an $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$ certificate follow. A conceptual understanding of these certificates in a 2D case will serve as a foundation for interpreting results on application scale problems.

3.1.1 Interpreting a $(\mathbf{x}, \delta)$-LAR Certificate



(a) $(\mathbf{x}, \delta)$-LAR, $\uparrow \delta$

(b) $(\mathbf{x}, \delta)$-LAR, $\uparrow \delta$



(c) $(\mathbf{x}, \delta)$-LAR, $\downarrow \delta$

(d) $(\mathbf{x}, \delta)$-LAR, $\downarrow \delta$

Figure 3.1: $(\mathbf{x}, \delta)$-LAR certificates for test points $\mathbf{x}$ and varying definitions of δ.

Figure 3.1 provides a visual representation of all the components in a LAR certificate. We train a tree ensemble model with 10 decision trees of max depth 3 on synthetic data with 100 samples using sklearn's `make_blobs()` function. Data is split 60/40 into train/test partitions. Membership in the test partition is denoted with a circumscribing black box. Data in the train partition, samples without circumscribing black boxes, are only shown in Subfigures 3.1c and 3.1a. The same data is shown in each subfigure of Figure 3.1. The background in Subfigures 3.1a and 3.1c shows

the decision surface of the trained model where darker colors represent higher levels of agreement between trees in the ensemble. The decision surface is reduced to the binary decision boundary shown in Subfigures 3.1b and 3.1d, which tells us whether the model will yield a red or blue class prediction for a particular set of x, y-values. The reason for placing the decision surface (Subfigures 3.1a and 3.1c) and decision boundary (Subfigures 3.1b and 3.1d) side by side is to highlight that the tree ensemble's hyperrectangular partition of the input space is not aligned with the black box neighborhoods. There can potentially be many different model states, meaning paths from roots to leaves, that can be activated within a single neighborhood search. The verification task is not as simple as just checking an individual HR from the tree ensemble's partition of $\mathbb{R}^n$.

We test the model for LAR on all points in the test partition. The result is most easily seen in Subfigures 3.1b and 3.1d. The black box represents the scope of the local neighborhood, δ. We define δ as a vector with an element corresponding to each attribute in data. The value of each element of δ notes a $\pm$ range around the sample in question that defines the δ-neighborhood along a single dimension. If all the elements of δ are the same value, then we are searching over a hypercube, otherwise, different values produce a HR. In this illustrative example, the difference is that the top row (Subfigures 3.1a and 3.1b) verify relatively large $\delta = [.2, .2]$, where as the bottom row (Subfigures 3.1c and 3.1d) verify relatively small $\delta = [.1, .1]$. In other contexts, the definition of δ may either stem from the structure of the problem ($\pm$ pixel intensity in an image) or from statistical dispersion metrics ($\pm z\sigma$ for each attribute). Most work to date focuses on a uniform definition of δ which is useful in image contexts [109, 184, 187] and while some focus on non-uniform definitions of δ, [6, 123], it remains an open problem to determine how to best define δ when sensitivity to individual attributes is expected to be different, which is often the case for tabular data. It is also possible to define a sparse δ which would restrict the search for counterexamples to a subspace. The definition of δ remains a design decision that needs to be made by the developer of an AI system.

The model satisfies the $(\mathbf{x}, \delta)$-LAR specification at samples covered with green stars. The certificate proves that no counterexamples were found, so the model's prediction will not change within the circumscribing box. Samples with no green star denote instances where the model violates the $(\mathbf{x}, \delta)$-LAR specification and a counterexample has been found. The certificate proves that there exists a set of inputs near the test point where the model will change its predicted class label.

The difference between Figs. 3.1a/3.1b and Figs. 3.1c/3.1d is the scope of the δ-neighborhood. The model's LAR behavior changes as the scope of the neighborhood increases or decreases. There will always exist a definition of $\delta = [0, 0]$, where the

model is robust to adversarial noise on a sample, but satisfying $(\mathbf{x}, \delta)$-LAR for small values is not interesting if it is possible to satisfy a larger definition of δ. For greater values, the guarantee provided by the certificate becomes stronger, proving that the model's behavior is consistent over larger contiguous input regions.

3.1.2 Interpreting an $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$ Certificate

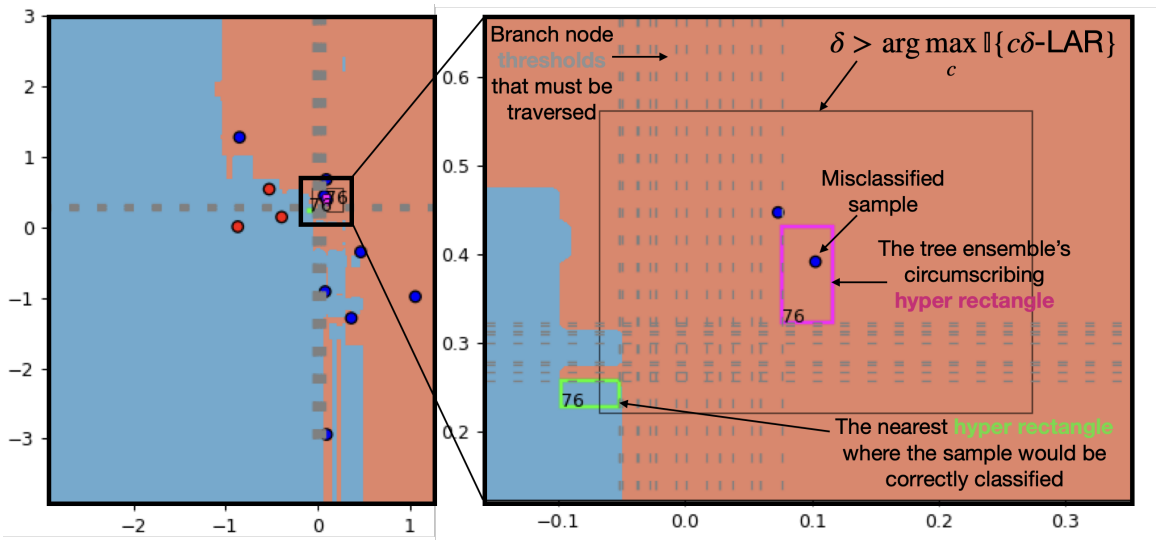


Figure 3.2: $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$ certificate, a minimal L_∞ counterfactual

Given a point $\mathbf{x}$ and a definition for δ, we want to know the largest scalar multiplier, c, for which $(\mathbf{x}, c\delta)$-LAR is satisfiable. A single $(\mathbf{x}, \delta)$-LAR certificate cannot provide the answer to this question, but a change point from `UNSAT` to `SAT` among a collection of certificates can indicate the maximum value of c for which no counterexample to $(\mathbf{x}, c\delta)$-LAR exists. We notate such a test as $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$, where $\mathbb{I}$ is an indicator function that returns 1 if a model satisfies the $(\mathbf{x}, c\delta)$-LAR specification.

Figure 3.2 shows a minimal L_∞-distance counterfactual for a particular misclassified sample. The decision boundary is shown in the background, and only misclassified test points are plotted. The plot on the right is a zoomed in view of local behavior in the left plot. This illustration shows a counterfactual for a single misclassified point. The sample falls in one HR, shown in magenta, defined by the structure of the trees in the ensemble. We are searching for the nearest, L_∞-distance, HR where the model will produce a different prediction. We claim that this is more interesting in contexts where we are interested in determining why a sample is misclassified because it provides the shortest list of paths through select decision nodes in the ensemble

that must flip in order for the sample to receive a different prediction from the model. One of multiple possible solutions is highlighted in green. The counterfactual can be reconstructed by noting all of the threshold values that need to be crossed in select branch nodes in the tree ensemble. If the misclassified sample attribute values place the sample on the other side of the grey dashed lines, then the prediction would be different.

3.2 Encoding Strategy

The SAT formalism for encoding $(\mathbf{x}, \delta)$-LAR is shown in Algorithm 7. The strategy for successively conducting LAR verification tasks to determine $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$ is detailed in Algorithm 8.

3.2.1 Encoding $(\mathbf{x}, \delta)$-LAR

We adopt the definition from [109] with slight modifications: a model satisfies $(\mathbf{x}, \delta)$-LAR at input point $\mathbf{x}$ if, for every $\mathbf{x}'$ such that $\{|x_i - x_i'| < \delta_i \mid \forall i \in \delta\}$, the model predicts the same label for $\mathbf{x}$ and $\mathbf{x}'$. For given δ, any $\mathbf{x}'$ which the model assigns a different label than that assigned to $\mathbf{x}$ is an adversarial example. For any model where all $\mathbf{x}'$ in the neighborhood of $\mathbf{x}$ never differ in assigned label, the model satisfies $(\mathbf{x}, \delta)$-LAR in the neighborhood of $\mathbf{x}$. The only difference between our definition and [109] involves δ, where we choose to define δ as a vector with length equal to the number of attributes in the data. Our notation makes it clearer that it is possible to encode different values for each δ_i, or to define a sparse δ.

Algorithm 7: Local Adversarial Robustness (LAR)

```
   /* Assert the δ constraint over inputs                                  */
 1 x ← center point of δ neighborhood
 2 δ ← neighborhood about x to verify robustness
 3 y ← model output for x
 4 C ← number of unique class labels
 5 for i ← 1 to number of attributes in data do
 6 |   lb ← arg max_k{t(k) | t(k) ≤ x_i − δ_i, a(k) = i}
 7 |   ub ← arg min_k{t(k) | t(k) > x_i + δ_i, a(k) = i}
 8 |   assert ¬T_lb /* lower bound on inputs                               */
 9 |   assert T_ub /* upper bound on inputs                                */
10 end
   /* Assert the ¬y constraint over outputs                                */
11 assert ⋁{P_{c,M} | c ∈ [C], c ≠ y} /* any other class for output        */
```

Local Adversarial Robustness (LAR), outlined in Algorithm 7, can be encoded by adding input and output constraints to the stack of assertions that encode the ensemble. Let $\mathbf{x}' = (x_1', x_2', \ldots, x_p')$ denote an input to the tree ensemble. To constrain $x_i' < b$, assert T_j, where $j = \arg\min_k\{t(k) \mid t(k) > b \wedge a(k) = i\}$. To constrain $x_i' > b$, assert $\neg T_j$, where $j = \arg\max_k\{t(k) \mid t(k) \leq b \wedge a(k) = i\}$. Constraints for $x_i' \geq b$

or $x_i' \leq b$ may be expressed in similar fashion. These can be composed with logical operations to constrain the desired input region. It is possible to represent arbitrary subsets of the input space in this way, but it may require many constraints, so we use axis-aligned, hyperrectangular regions wherever possible. To constrain that the model predicts class c, assert $P_{c,\mathcal{M}}$, or, to ensure that fewer than j of N trees vote for class c, assert $\neg S_{j,N,c}$.

To test robustness within a δ-neighborhood about an input $\mathbf{x}$ which the model labels with class y, for all i, constrain $x_i' > x_i - \delta_i$ and $x_i' < x_i + \delta_i$. Then, for the output, assert that any class other than y is predicted. This is equivalent to verifying the following proposition, $\neg(\mathbf{x}, \delta)$-LAR $\implies \exists \mathbf{x}'$. If a satisfying assignment to the formula does not exist, then the model adheres to $(\mathbf{x}, \delta)$-LAR. Otherwise, a satisfying assignment shows the existence of counterexample, $\mathbf{x}'$, that violates $(\mathbf{x}, \delta)$-LAR.

Counterexamples can be extracted from satisfying assignment by using the strategy described in Section 2.3, and if desired, one may enumerate model configurations that produce adversarial examples by explicitly forbidding those that were previously discovered and checking satisfiability again. In this way, one can find and then sample from the set of all local adversarial examples. This is an instance of what is known as the #SAT problem; how many satisfying assignments exist for a given formula? If tied votes are broken arbitrarily, one should expect that model adherence to $(\mathbf{x}, \delta)$-LAR happens less frequently since this causes the model not to be robust at any input where a tied vote occurs.

3.2.2 Encoding $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$

Algorithm 8 details our procedure for solving $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$, which finds the closest counterfactual, $\mathbf{x}'$, where the model yields a different prediction than it does for point $\mathbf{x}$. To obtain the solution, we perform a binary search over possible scalar magnitude multipliers, c, of the δ-neighborhood. In experiments, we define the δ-neighborhood as a function of some statistical dispersion metric (such as standard deviation or inter-quartile range) over each attribute, but any definition of the local search neighborhood can be swapped in. The key for why a statistical dispersion metric is useful is that it accommodates different levels of sensitivity towards different attributes in tabular data. If TEA yields a `SAT` certificate, a counterexample exists, and we reduce the value of c for the next iteration. If TEA yields an `UNSAT` certificate, no counterexamples exist, and we increase the value of c for the next iteration. Precision is used as a stopping criterion, and the value of c denotes the discovery of the closest LAR counterexample to the point. Implementation of this algorithm also includes checks against a timeout that can be set to produce the best possible solution with a constrained time budget.

Algorithm 8: Nearest Counterfactual, $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$

$\mathbb{A} \leftarrow$ set of attributes in data
$\mathcal{M} \leftarrow$ trained model
V$\leftarrow$TEA()
$lb, c_\ell \leftarrow$ lower bound on scalar multiplier
$ub, c_c \leftarrow$ upper bound on scalar multiplier
$\epsilon \leftarrow$ numerical precision
$\delta \leftarrow \{\delta_i = \texttt{SDM}(a_i) \mid \forall a_i \in \mathbb{A}\}$`/* any Statistical Dispersion Measure */`
$\mathbf{x} \leftarrow$ data point
while *abs*$(c_c - c_\ell) > \epsilon \vee \neg V.isSAT()$ **do**
    V$\leftarrow$TEA()
    V.assert$(\mathcal{M}, (\mathbf{x}, c\delta)\text{-LAR})$`/* encode model and specification */`
    if $V.isSAT()$ **then**
        $ub \leftarrow c_c$`/* counterexample found, decrease upper bound */`
    else
        $lb \leftarrow c_c$`/* no counterexample found, increase lower bound */`
    end
    $c_\ell \leftarrow c_c$`/* last c value gets current c value */`
    $c_c \leftarrow (lb + ub)/2$`/* current c value gets next c value */`
end
return V.assignment(), c_c

3.3 Baseline Comparison

We apply TEA to two publicly available datasets, Vehicle Collision [56] and MNIST [116], to verify $(\mathbf{x}, \delta)$-LAR. In both contexts, we recreate experiments originally defined by [56] and performed by [187] on their formal system, VoTE, to give us a baseline for comparison. We find that while TEA is comparable in cases where the verification task is relatively small, TEA scales to a level that is intractable for VoTE.

3.3.1 $(\mathbf{x}, \delta)$-LAR for Vehicle Collision

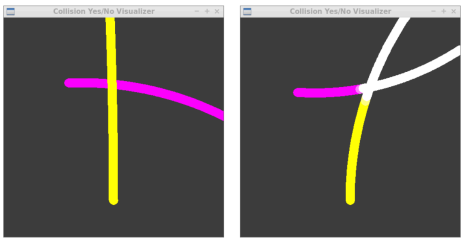


Figure 3.3: Illustration of vehicle collision data from [56].

In this study, we train tree ensembles to detect collisions between simulated vehicle trajectories. The data generating process is identical to that of [187]. We generate 30,000 training and 3,000 test samples using the simulation from [56]. A sample is visualized in Figure 3.3. The data has six features normalized into $[0, 1]$ including distance between vehicles in each of two directions, speed of the second vehicle, starting direction of the second vehicle, and rotation speed of each vehicle. Each vehicle is circle-shaped, and a safety margin is defined around each vehicle. Any simulated trajectories where the only the safety margins collide are ignored. The resulting data contains trajectories where either the vehicles collide, or, their safety margins never overlap (they do not come close to colliding). While VoTE [187] verifies robustness for a mix of tree ensembles and gradient boosting machines, we use only tree ensembles in TEA, and only report VoTE's times for tree ensemble verification.

The goal of this verification task is to certify whether a test point will receive the same label, collision or no-collision, within a defined neighborhood around the point. In the case of no-collision, a certificate for local adversarial robustness states verifies that the two vehicles would not have collided even if input values changed within a defined extent. In the case of a collision, a certificate for local adversarial robustness states that the vehicles would have collided no matter how the input values change within a defined extent.

Table 3.1: Comparison of TEA and VoTE for verifying model adherence to $(\mathbf{x}, \delta)$-LAR on a vehicle collision dataset.

d	B	Test Data Accuracy (%)	Test Data Robust (%)	TEA Time (s)	VoTE Time (s)
5	20	82.07	78.20	3.55	-
5	25	82.77	78.47	4.25	-
10	20	89.10	58.20	**52.85**	56
10	25	89.03	58.83	**133.09**	286
15	20	92.73	34.87	502.23	**273**
15	25	92.13	36.23	**1144.55**	1651
20	20	94.33	30.37	964.61	**367**
20	25	94.67	31.30	**1950.09**	2520

We check local adversarial robustness at all test points with $\{\delta_i = 0.05 \mid \forall i \in \delta\}$, for various maximum depths, d, and numbers of trees, B, to match the experiments of [187]. The results, which include the cumulative times reported by the solver over all 3,000 test queries and the total wall time of the experiment, are given in Table 3.1. VoTE times are as reported by [187], which does not test random forests of depth 5. B denotes the number of trees and d the maximum depth. For each configuration, the lesser time is bold. It is important to note that we are training and testing our own models with the same parameters set by [187]. Even though the models are not identical, we find levels of accuracy and robustness similar to those reported by [187].

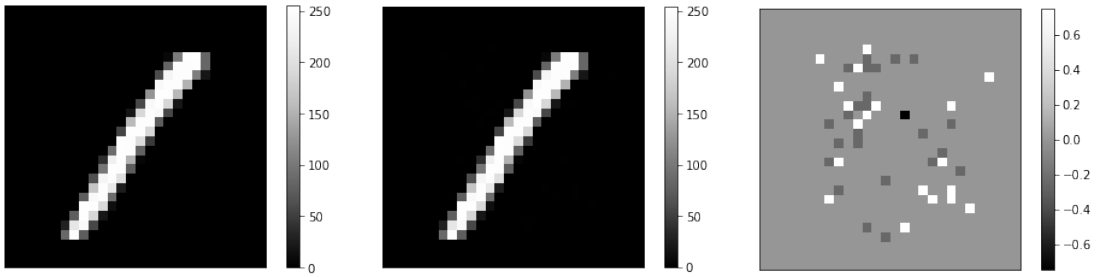
Test data robustness reported in 3.1 provides evidence that the number of trees in the ensemble does not affect robustness to as great an extent as the maximum depth of the trees. The fact that we observe robustness decreasing as maximum depth increases adheres to our expectations. With deeper trees, the ensemble traverses more decision nodes. Each decision node is an opportunity for adversarial noise to flip the outcome of the threshold check and cause a different path to become active. Shallow trees have fewer nodes traversed, therefore exhibit $(\mathbf{x}, \delta)$-LAR more often. In fact, the most robust robust decision tree is one that only ever outputs a single label, or a decision tree of depth $d = 1$.

The fact that we observe the number of trees in the ensemble contributing relatively little to the overall robustness of the ensemble is, perhaps, counterintuitive. It would be reasonable to hypothesize that with the addition of each decision tree to the ensemble, a new hyperrectangular partition of $\mathbb{R}^n$ intersects with the ensemble's existing hyperrectangular partition. This creates new HRs that may cause the

model to produce a different vote tally in these regions of the input space. A simple hypothesis could be that the effect the number of trees has on test data robustness quickly plateaus. For example, 5 trees to an existing 20 may not influence $(\mathbf{x}, \delta)$-LAR as much as adding 5 trees to an existing set of 10. Another possible explanation for the observed levels of robustness in Table 3.1 is that there may be high degrees of consensus between the individual decision trees in the ensemble. This would explain why the addition of a new, similar decision tree does not substantially change the ultimate output of the ensemble. It also suggests test data robustness could be greatly influenced by the number of trees in the ensemble if there is a low degree of consistency between outputs produced by each tree.

We find that TEA is similar or slower than VoTE for ensembles of 20 trees, but notably faster for ensembles of 25 trees, suggesting that the SAT approach may have greater overhead cost for deep trees, but scales better as ensemble size increases. One possible explanation for this phenomenon is that the addition of each new tree to the ensemble increases the number of HRs that constitute the ensemble's partition of $\mathbb{R}^n$. VoTE must enumerate these HRs and sort them into equivalence classes before testing, whereas TEA only encodes constraints that prevent the solver from exploring infeasible model states. The ability of SAT solvers to cut off large subtrees of the search space by leveraging Conflict-Driven Clause Learning (CDCL) may result in TEA not needing to exhaustively check every HR. The savings on this efficiency should increase as the size of the ensemble increases.

3.3.2 $(\mathbf{x}, \delta)$-LAR for MNIST



original point
predicted label: 1

adversarial example
predicted label: 2

adversarial noise
(note scale)

Figure 3.4: An adversarial example for an MNIST test image.

In this study, tree ensemble models classify handwritten digits from the MNIST dataset [116]. This includes 70,000 grayscale 28×28 images with integer pixel values

Table 3.2: TEA cumulative performance statistics of verifying adherence to $(\mathbf{x}, \delta)$-LAR for 10,500 test data points from MNIST. These verification tasks are intractable for VoTE.

d	B	Test Data Accuracy (%)	Test Data Robust (%)	TEA Time (s)
5	20	80.89	11.46	79.71
5	25	81.31	10.88	110.63
10	20	93.23	27.94	5661.90
10	25	93.33	30.19	8900.33
15	20	95.30	31.83	68375.62
15	25	95.62	32.38	109360.51
20	20	95.64	29.08	95717.95
20	25	95.84	32.77	141557.90

between 0 and 255, which we split randomly into 85% training and 15% test according to the procedure of [187]. We verify $(\mathbf{x}, \delta)$-LAR at test points with $\{\delta_i = 1 \mid \forall i \in \delta\}$, which describes a ± 1 intensity value for each pixel, for various maximum depth d and number of trees B. Because the number of features is large (784 for a 28×28 image), the search space is too large for VoTE to verify, so [187] limit the search for counterexamples to a sliding window of 5×5 pixels. TEA is able to verify $(\mathbf{x}, \delta)$-LAR without such limitation, and we report statistics on the comprehensive version of the verification task.

Cumulative reported solver times over 10,500 queries are given in Table 3.2. The time to test robustness is much higher here than in the vehicle collision problem, due largely to the increased dimensionality of the MNIST data. $(\mathbf{x}, \delta)$-LAR tests may be conducted in parallel, as each test point represents an independent verification task. In parallel, we verify the entire set in only a handful of hours (though the reported time is the total processor time). Furthermore, comparing the vehicle collision and MNIST results, there is similarity in the way the time to solve evolves across the varying d and B, suggesting that, while high-dimensional data does result in longer time to solve, it does not necessarily preclude the verification of large models.

TEA reasons about all 784 pixels simultaneously, and this has an impact on the comprehensiveness of the certificates we produce. An imperceptible change in intensities in all four corner pixels in an image is an example of an adversarial perturbation that TEA will test but VoTE will ignore. Search over all pixels for $(\mathbf{x}, \delta)$-LAR results in TEA reporting a lower percentage of robust test points than reported by [187].

Figure 3.4 gives an example of an adversarial example discovered in this way. The adversarial example is obtained by adding the adversarial noise to the original image. Table 3.2 shows a seemingly counter-intuitive property when compared to the results from the last study (Table 3.1), which is that test data robustness increases as the ensemble gets deeper. This stands in contrast to the trend we observed in 3.1. [187] observe the same trend in their experiments. We hypothesize that this trend is a result of underfitting multi-class data. Deeper trees in the ensemble may do a better job at partitioning the input space such that leaf nodes contain low-entropy distributions during training. This may be especially true for multi-class problems, where extra depth can be used to discriminate between more class labels. Evidence to support this hypothesis may be found in the test data accuracy column of Table 3.2. Test data robustness is fairly steady for all parameter configurations other max depth $d = 5$, where the accuracy of the model is significantly lower. While we observe a similar increase in accuracy paired with an increase in robustness in the last study 3.1, a notable difference between these learning tasks is that MNIST has 10 class labels whereas the vehicle collision data is a binary classification problem.

3.4 Utility of LAR in a Radiation Safety Context

Figure 3.5: Radiation Portal Monitor in a Radiation Safety Context

We apply TEA to a radiation threat adjudication task in order to show how our ability to verify LAR in trained tree ensembles helps address existing questions about the real world system. Most attributes in the dataset are either measured directly, or computed from sensors in a radiation portal monitor, an example of which is shown in Figure 3.5. These sensor systems are placed at various ports of entry to the United States. Current practice involves scanning vehicles for potentially harmful sources of radiation as they pass and then following up with a manual inspection to determine that the vehicle is safe for passage. One challenge is that there are many types of naturally occurring radioactive materials that can trigger these sensors, but that do not pose risk or harm to people. It is therefore if pragmatic interest to maintain or enhance the existing efficacy of the radiation detection procedure while minimizing the amount of time and energy that goes into manually inspecting vehicles carrying harmless sources of radiation. One way of accomplishing that is to deploy a combination of physics based modeling and machine learning to confidently allow non-threatening vehicles to pass with only a scan and no manual follow up, while never allowing plausible threat to go uninspected.

A substantial amount of existing effort goes into making this AI system trustworthy. Nuclear physicists generate the best data possible, which includes both radiation

source signatures collected at ports of entry and synthetic data that is engineered to look threatening. A trained model can be only as good as the data that it sees during training, so the purpose of adding synthetic data is to increase the coverage of the reference data and to increase the chances that the model will pick up on the engineering specifications that are baked into the data and their ground truth labels. No matter how much data is generated, and no matter the fidelity of the simulations, we know that it is not a perfect representation of reality, only a projection of it. Statistical methods never achieve perfect coverage. This means that the best we can say is that we ran a test many times in simulation, and we never saw a particular fault manifest so we can feel sufficiently confident that it will not happen once the model is deployed. Low error rates alone do not inspire the level of assurance that this application context requires. This is because low error rates say nothing about the ways in which the model might fail. Often, trained models do not fail because of an error in the optimization routine, but rather they fail because critical fault modes are not discovered before deployment.

We provide as much detail about the domain, our tree ensemble, and our method as possible. Due to the sensitive nature of the application domain, we must often refer to individual models and attributes with placeholder names. The level of detail provided should still be sufficient to set up a similar suite of tests in different application settings when the featurization of the data takes on a similar form. We show how TEA can answer some open questions that arise in this nuclear safety context by complementing statistical V&V methods with formal V&V methods.

3.4.1 Verifying $(\mathbf{x}, \delta)$-LAR on test data from ports of entry

We report model robustness rates on test data as well as TEA performance statistics in Table 3.3. Varying magnitudes of δ are reported, which show how the robustness of the model changes as a function of the specification we wish to verify. The model we test with TEA is among the largest models we test. The test data, over which we wish to verify δ-LAR, contains over 100K samples of data, which comprises over 300 features. Over one million unique $(\mathbf{x}, \delta)$-LAR verification tasks go into the construction of Table 3.3. Verification instances are encoded and solved in parallel with a pool of 88 processors and cumulative time to complete all verification tasks in the table is 2 hours and 28 minutes.

Level of robustness decreases as the definition of δ increases in magnitude, as expected. δ is defined as a scalar multiple of standard deviation of the values of each attribute in the test data set, $\{\delta_i = z\sigma_i \mid \forall i \in \delta\}$. The wall time TEA requires to express the verification task in CNF logic remains constant across different definitions of δ. The solve time TEA requires increases as as the magnitude of δ increases. This

Table 3.3: TEA performance statistics for verifying model adherence to $(\mathbf{x}, \delta)$-LAR for over 100K samples in radiation safety context.

δ ($z\sigma$)	$(\mathbf{x}, \delta)$-LAR (% Data)	Avg Wall (s)	Max Wall (s)	Avg Solve (ms)	Max Solve (ms)
1.0E-03σ	99.372	0.628	7.194	0.049	6.205
2.2E-03σ	95.476	0.592	6.594	0.299	17.072
4.6E-03σ	36.186	0.579	6.234	1.342	23.894
1.0E-02σ	22.348	0.599	5.100	2.056	48.025
2.2E-02σ	16.897	0.633	4.813	3.821	57.393
4.6E-02σ	10.776	0.593	6.313	5.610	67.686
1.0E-01σ	6.061	0.629	6.845	7.471	81.270
2.2E-01σ	2.848	0.613	4.922	5.416	102.188
4.6E-01σ	1.247	0.589	7.581	8.213	252.257
1.0E+00σ	0.147	0.560	5.123	6.225	195.629

is expected as there is a greater local neighborhood to search with more possibilities for discovering a counterexample to $(\mathbf{x}, \delta)$-LAR. Solve times are on the order of 1000x faster than the time to encode the problem and read in the solution (note units in Table 3.3). Our implementation of TEA for generating CNF logic is focused on convenience for research purposes, and we expect that the time to express the verification tasks could be improved by restructuring TEA for speed. The overall time to compute is still very reasonable considering that $(\mathbf{x}, \delta)$-LAR is a property that need only be checked once before deployment, so the V&V cost is all upfront. This experiment provides evidence that TEA scales to a degree necessary to handle verification tasks for real-world AI systems.

3.4.2 $(\mathbf{x}, \delta)$-LAR certificates to verify that different vehicle attributes will never reduce assessed risk

A longstanding question that domain experts would like an answer to is whether the model is invariant to the type of vehicle that passes through the sensor. Testing the trained models with statistical methods provides estimates, but does not guarantee that a particular model adheres to this specification. $(\mathbf{x}, \delta)$-LAR can be applied to answer this question formally. Consider two identical, dangerous sources aboard different vehicles that vary in how the source signal as well as background radiation can be attenuated by the presence of the vehicle in the sensing area. The model should detect both sources regardless of whether a truck or a van is carrying the source.

Table 3.4: TEA results for verifying model adherence to $(\mathbf{x}, \delta)$-LAR for adversarial perturbations to select vehicle attributes.

Model Name	Test Acc (%)	δ ($z\sigma$)	$(\mathbf{x}, \delta)$-LAR (% Data)	Avg Wall (s)	Max Wall (s)	Max Solve (ms)
M0	98.693	Vehicle 3σ	99.983	0.591	12.783	13.044
M1	98.655	Vehicle 3σ	100.000	0.664	13.351	13.042
M2	98.675	Vehicle 3σ	99.998	0.577	11.048	2.416
M3	98.628	Vehicle 3σ	100.000	0.657	9.535	2.393
M4	98.615	Vehicle 3σ	99.956	0.592	11.108	13.138
M5	98.598	Vehicle 3σ	100.000	0.600	10.179	1.390

We apply TEA in order to verify that significant changes to the `VEHICLE` family of attributes will never cause the model to downgrade its advisory from *dangerous* to *safe*. If the model exhibited such a weakness, an adversary could select a vehicle with different characteristics to fool the system and evade detection.

Physicists have developed a few models trained under two different conditions, including hyperparameters for the tree ensemble training and the featurization of the data. Even numbered models in Table 3.4 (M0, M2, M4) have one set of parameters and odd numbered models (M1, M3, M5) have a different set. All models have the same max depth and number of trees, and are among the largest models we test. The goal is to use TEA to determine whether a particular methodology for training the model yields a model that meets the vehicle-invariance specification.

The test data, over which we wish to verify $(\mathbf{x}, \delta)$-LAR, contains over 100K samples, each with over 300 features. Our experiments are parallel pooled over the roughly 600K verification tasks that must be performed (about 100K samples for 6 models). Overall compute time is less than 90 minutes, which is faster than the experiments in the last section. This is due in large part to the fact that our definition of δ is sparse. Only perturbations over a select subset of about 10, numerical, `VEHICLE` attributes must be evaluated. Average solve time, which is not included in the table, is less than 0.02 ms across all models. We define $\delta = \{\delta_i = 3\sigma(i) \mid \forall i \in \delta\}$. If the model satisfies $(\mathbf{x}, \delta)$-LAR, we can say for certain that no counterexamples exist. This means the model will not yield a different prediction between the sample in question and any other sample with a perturbation along any or all vehicle attributes as long as that perturbation does not exceed 3σ in magnitude.

Table 3.4 shows the results of testing robustness of all candidate models for perturbations within `VEHICLE` attributes over all test data samples. All models are virtually indistinguishable in terms of the level of accuracy in test data predictions. We find

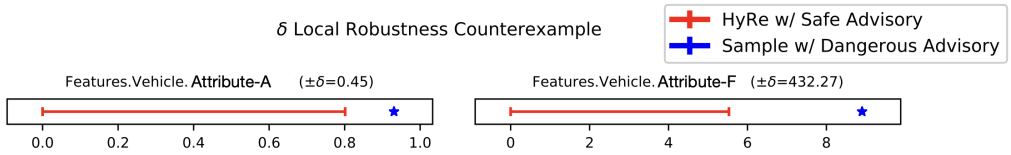


Figure 3.6: Visualization of the differences between $(\mathbf{x}, \delta)$-LAR $(\mathbf{x}, \mathbf{x}')$ counterexample. $(\mathbf{x}, \mathbf{x}')$ have identical attribute values for all attributes not visualized.

that odd numbered models (M1, M3, M5) satisfy our strict definition of δ-LAR over all test data points as desired. These models will never downgrade the risk of a sample solely due to vehicle attribute values. The even numbered models (M0, M2, M4) satisfy LAR very often but counterexamples are found. Given that we know the odd numbered models differ from even numbered models in terms of the conditions under which they were trained, we can use Table 3.4 as evidence that the conditions that yield the odd models produce highly accurate models that also adhere to the physicists desire for the model to be invariant to perturbations manifesting solely among `VEHICLE` attributes. TEA shows that by not simply picking the model with the highest level of recorded accuracy, we choose a model that satisfies a desirable, contractual property. Indeed, our approach can support model selection decisions by adding quantified robustness to the pool of selection criteria (we explore this further in Section 3.4.4).

The fact that there are very few counterexamples found for the even models (M0, M2, M4) leads us to wonder what those counterexamples look like. Figure 3.6 shows an example discovered for a particular input. While the data point comprises over 300 attribute values, only two attributes are shown because the perturbation to violate our definition of δ-LAR only need to manifest along these two attributes. All other attributes not depicted remain the same between $\mathbf{x}$ and $\mathbf{x}'$. The blue star denotes the attribute values for the sample, $\mathbf{x}$ in question, and the ground truth label for this sample is *dangerous*. The red ranges define a HR where the model will yield a *safe* advisory. The model will yield a *safe* advisory for any $\mathbf{x}'$ that is identical to $\mathbf{x}$ for all attribute values other than the two attributes shown, if $0 < \mathbf{x}'_{Vehicle.Attribute-A} \leq 0.8$, which denotes the size of the payload and $0 < \mathbf{x}'_{Vehicle.Attribute-F} \leq 5.6$ which denotes the size of the vehicle. The counterexample suggests that for two vehicles carrying identical sources, a vehicle that is slightly smaller and with slightly less payload would evade detection, which violates the specification that the model be invariant to vehicle-centric characteristics.

This insight shows the utility of TEA in finding counterexamples that communicate feasible model failures. If we were to discover this fault with sampling methods,

we would need to generate at least one data point that falls into the red ranges and is equivalent to the blue point in all of the other `VEHICLE` attributes present in data. Given that the range of possible values is large for some of these attributes, such as `VEHICLE`.Attribute-F, where 3σ corresponds to a perturbation of ± 432.27, means that we will need to generate a data point that falls into a small hyperrectangular region to observe the failure. Accomplishing this would require a rather dense sampling, potentially infeasible in practice.

3.4.3 Characterizing model sensitivity to adversarial perturbations with $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$ certificates

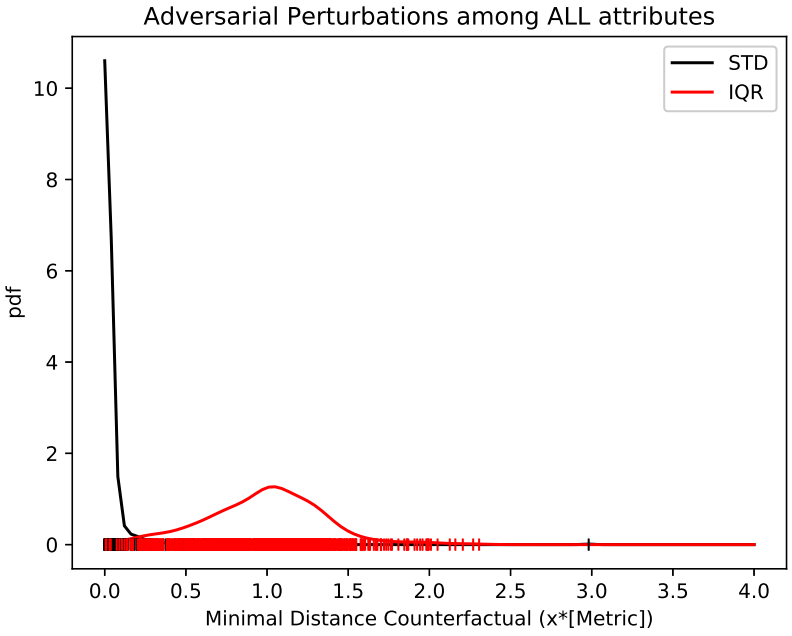


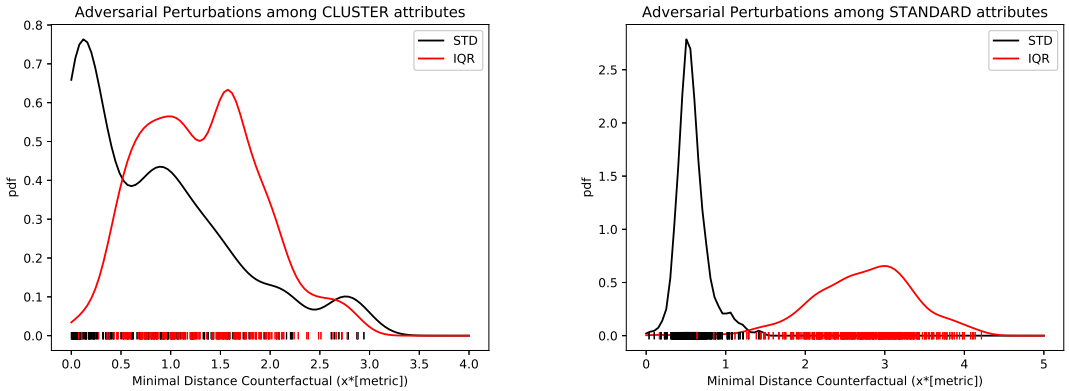
Figure 3.7: The distribution of $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$ for test data. This estimates the magnitude of adversarial noise that is required to force the model to violate $(\mathbf{x}, c\delta)$-LAR.

Our radiation safety context is unique in the sense that it is possible to re-engineer data for the supervised task by adding/removing features or changing the physics modeling that produces synthetic data elements. Knowing the maximal level of robustness a model exhibits across select attributes can help inform design choices when engineering the best possible set of data to train the model.

TEA can search for the formal definition of the largest scalar c for which the model adheres to $(\mathbf{x}, c\delta)$-LAR. This lets us reason about the maximal extent of robustness exhibited at a particular point rather than whether the model meets one arbitrary definition. This still requires that we make a design choice about the definition of δ, but this can be informed by statistical dispersion metrics in the absence of domain-specific, formal requirements.

We first show how TEA provides certificates for $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$ when δ is non-sparse. An individual certificate denotes the maximal amount of robustness the model exhibits at a point. Figure 3.7 shows the sensitivity of the trained model to adversarial perturbations among all attributes in the data set (`ALL`). This shows the distribution signifying the maximal value of c (x-axis) for which no counterexamples are found for $(\mathbf{x}, c\delta)$-LAR at all test data points. By noting the scalar multiplier, c, for each point, we can use kernel density estimation to build a distribution. This distribution describes the probability that the model will exhibit a particular level of robustness for a yet untested point, assuming that the untested point is drawn from the same distribution. Figure 3.7 gives us evidence to expect the model is likely adhere to $(\mathbf{x}, c\delta)$-LAR for a new test point for a value of $.5 < c < 1.5$ when $\delta = \{\delta_a = \texttt{IQR}(a) \mid \forall a \in \mathbb{A}\}$ where $\mathbb{A}$ represents a set of all attribute indices. If δ is instead defined using z-score across attributes, the maximum value of c where $(\mathbf{x}, c\delta)$-LAR is satisfied is lower. This analysis provides an estimate of the magnitude of the adversarial perturbation that is needed to make the model violate $(\mathbf{x}, c\delta)$-LAR.

It is also possible to determine this degree of adversarial perturbation that is needed to cause the model to violate LAR among select attributes in data. Of the hundreds of attributes there are at least a dozen unique groups of attributes. We define a sparse δ that searches for robustness exhibited by the model if all attribute values are held constant but noise affects the non-zero elements of δ. Figure 3.8 shows two examples for testing model sensitivity to perturbations in families of features. Sensitivity to adversarial perturbations among `CLUSTER` attributes is shown in Subfigure 3.8a. In comparison with Figure 3.7, the model exhibits higher degrees of robustness when focusing solely `CLUSTER` attributes, which matches intuition that reducing the dimensionality of the search for counterexamples should reduce the number of possible solutions for TEA to find. Subfigure 3.8b shows sensitivity to adversarial perturbations along `STANDARD` attributes. The model is much more robust, on average, to perturbations among these attribute values than to perturbations in `CLUSTER` attributes or `ALL` attributes. The model often exhibits robustness for larger values of c in these two subspaces when compared to the overall feature space, leading us to conclude that sensitivity to `CLUSTER` and `STANDARD` attribute groups likely are not the robustness-limiting features for cases when the model violates $(\mathbf{x}, \delta)$-LAR.



(a) Sensitivity to adversarial perturbations among `CLUSTER` attributes.

(b) Sensitivity to adversarial perturbations among `STANDARD` attributes.

Figure 3.8: The distribution of $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$ for test data and sparse δ. This estimates model sensitivity to adversarial noise along select attributes and shows the magnitude of adversarial noise that is required to force the model to violate $(\mathbf{x}, c\delta)$-LAR.

A design decision needs to be made for how we want to define the structure of δ. Given that each attribute in the data exhibits very different distribution properties, there is no correct answer. The goal is to define δ such that each entry, δ_i adequately describes dispersion among attribute values. In this particular context, multiple attributes can be adequately described as a spike-and-slab distribution. A uniform distribution over attribute values would be an accurate characterization except for a very large probability mass that exists somewhere in the distribution. The model adheres to $(\mathbf{x}, c\delta)$-LAR for larger c, as expected, because the tail ends of the distribution do not affect Inter-Quartile Range (IQR) as they affect z-score, and these tail ends tend to contain the large probability masses we observe in the data. A more informed definition of δ, possibly by using unique dispersion metrics for different groups of attributes may aid our analysis by better defining the amount of distance between two points in a single attribute that constitutes the threshold between noise and meaningful different.

TEA facilitates specification-driven design. LAR is inherently a local design specification, but our analysis generalizes the findings across multiple LAR certificates. If the goal is to deploy a model that adheres to $(\mathbf{x}, \delta)$-LAR very often, TEA can estimate the level of robustness the model will exhibit on untested samples. By restricting the scope of the local search to subspaces of inputs, TEA can positively influence decision making as to what types of sensors are needed in order to ensure that adversarial

noise does not compromise the model. If the model is often certified robust for a defined magnitude of noise along `STANDARD` attributes, we want to make sure that the sensors that record those attribute values possess a level of precision that prevents larger-than-certified-safe amounts of noise from impacting inputs to the model.

3.4.4 Informing model selection by verifying $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)$-LAR$\}$ for prediction outcomes

Part of standard practice in designing AI systems involves selecting a predictive threshold from an ROC curve that balances the costs associated with error types. If a specification exists on a maximum allowable False Positive Rate (FPR), then we pick a predictive threshold value that satisfies that statistical specification. If formal V&V is applied to AI systems, it typically takes place after the model selection step, which means that a single model is usually tested for adherence to any and all logical specifications. Where there is a mix of class labels in the distribution of training points at a leaf, changing the predictive threshold will cause the tree to produce different outputs. This affects whether the model adheres to δ-LAR. Given the demonstrated efficiency of TEA, we set out to test how multiple trained ensembles respond to changes in their predictive threshold.

We do not wish for the model to exhibit robustness at all times. The model should exhibit robustness when it makes correct predictions, providing evidence that the learned structure of the tree ensemble is responsible for the outcome, and that noise could not be the reason for the correct prediction. The model should not exhibit robustness when it makes incorrect predictions. While incorrect predictions may also be due to the learned structure of the tree ensemble, we would also like to see that they may be attributable to noise on input measurements. In contrast, correct predictions that are not robust mean that the model was close to getting it wrong and one possible reason it is correct is due to noise.

When presented with multiple trained models with indistinguishable statistical differences in performance metrics, robustness can break ties by revealing which model produces predictions in a way most aligned with human expectation. Table 3.4 shows the result of testing $(\mathbf{x}, \delta)$-LAR over all test points in order to verify whether a certain safety criterion is satisfied for all predictions from the model. If we apply TEA to determine which model in a set exhibits the highest levels of robustness, the strategy should change. Rather than specifying that $(\mathbf{x}, \delta)$-LAR be satisfied for as many predictions as possible, we instead note that the desired level of robustness should correspond with each type of prediction outcome. For example, if a model exhibits robustness when it makes an incorrect prediction, this would represent a confidently

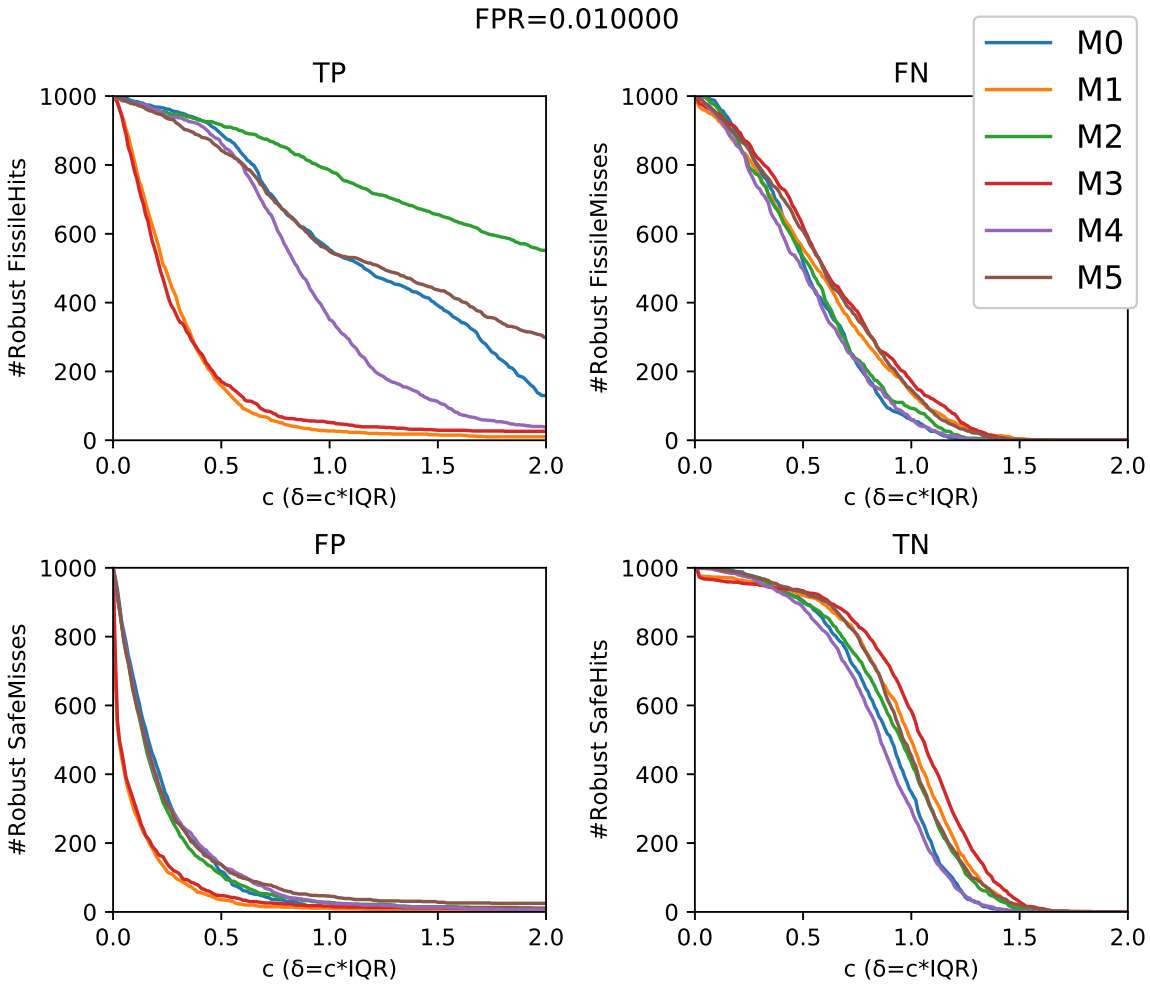


Figure 3.9: Quadrants correspond to prediction outcomes and plots show relationship between c and the frequency that the model adheres to $(\mathbf{x}, c\delta)$-LAR. FPR$= 1 \times 10^{-2}$.

incorrect prediction, which is undesirable. Instead, the engineering desiderata we would rather impose is that the model should produce robust predictions when it produces the correct prediction (TP, True Negative (TN)) but violate robustness when it produces an incorrect prediction (FP, FN).

We verify $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$ for a sample of points from the test data set. With the definition of the maximal value of c for which the model exhibits robustness on a particular $\mathbf{x}$, we can generate plots which show the counts of robust predictions broken out by prediction outcome as a function of c. We run this experiment for 100 possible FPR targets between the range of FPR$= 5 \times 10^{-4}$ and FPR$= 1 \times 10^{-2}$ to examine how robustness across prediction outcomes responds to changes in the tree ensemble prediction threshold. We sample 1000 data points from each prediction

outcome, with replacement in the case where e.g. fewer than 1000 FP are produced by the model over our 100K test data points. We set the precision limit for the stopping point in the binary search for $\arg\max_c \mathbb{I}\{(\mathbf{x}, c\delta)\text{-LAR}\}$ to be $\epsilon = 1 \times 10^{-3}$ and impose a timeout of 1 minute per sample. If the timeout is reached, the largest c for which a counterexample does not exist is returned, regardless of whether the last two successive verification tasks are within the precision limit. With an unlimited time budget, computing this verification task for all samples is possible. These experiments were designed to halt in roughly 48 hours when parallelized across 88 processes.

Figure 3.9 shows the degree to which model TP, FP, FN, and TN predictions exhibit robustness for one select FPR target. Based on precision requirements, our binary search takes 13 iterations to narrow the scope from $c^* \pm 5$ to $c^* \pm 1 \times 10^{-3}$ and find the best tested value of c. This results in 56K verification instances to produce Figure 3.9 and 1.4M verification instances to produce Table 3.5 which shows results for select FPR targets over our tested range.

For varied definitions of c, Figure 3.9 shows the number of samples that satisfy robustness when $\{\delta_i = c^* \texttt{IQR}(i) \mid \forall i \in \delta\}$. We wish for the diagonal elements (correct predictions) of Figure 3.9 to exhibit higher degrees of robustness than the off-diagonal elements (incorrect predictions). Each model adheres to that desiderata to a different extent. M1 and M3, for example, exhibit significantly lower levels of robustness for TP predictions, which suggests that in this radiation safety context, smaller amounts of adversarial noise will cause M1 and M3 to miss dangerous vehicles than the amount of noise needed to do the same for other models. While models M0, M2, M4, and M5 all exhibit similar levels of robustness for FPR$= 1 \times 10^{-2}$, $c < 0.5$, the level of robustness among TPs stratify for values of $c > 0.5$. M2 distinguishes itself as the model that exhibits the highest degree of robustness for TP samples for large δ-neighborhoods.

Figure 3.9 is typically a single frame in an animation that shows how the robustness curves change in response to each tested FPR target. Table 3.5 provides a summary of the information present in a few frames for select FPR targets and select c ($c = 0.25$ and $c = 1.0$). M1 and M3 continue to exhibit significantly lower levels of robustness even across multiple FPR targets. FPR targets where models adhere to robustness for over 50% of their correct predictions and under 25% of their incorrect predictions are bolded in Table 3.5. M0 and M2 emerge as the candidates that best exhibit the desiderata we set which was that the model exhibit robustness for correct predictions and not exhibit robustness for incorrect predictions.

There are other interesting trends we observe in Table 3.5. M4 exhibits the largest drop in robustness prevalence among TPs between $c = 0.25$ and $c = 1.0$ between FPR targets. This suggests that the decision boundary for M4 may be near equidistant

Table 3.5: Robustness rates for 1000 sampled test points per prediction outcome, per FPR target, for model adherence to $(\mathbf{x}, c\delta)$-LAR for select FPR limits and select values of c. Each target FPR row group is an abbreviated description of a quad chart like that shown in Figure 3.9. Models M1 and M3 consistently violate to $(\mathbf{x}, c\delta)$-LAR more frequently than other models. Bold quadrants indicate both [TPRR(P/PP),TNRR(N/PN)]$>$ 50% and [FPRR(N/PP),False Negative Robustness Rate (FNRR)(P/PN)]$<$ 25%, which only manifests for M0 and M2. This suggests M0 and M2 are best equipped of the six tested models to produce robust predictions only when the model is correct. M4 exhibits the greatest drop in TPRR between $c = .25$ and $c = 1.0$. TNRR is the most invariant between $c = .25$ and $c = 1.0$. Less restrictive FPR target increases robustness of PP (TPRR/FPRR) for all models except M0 and M2, independent of c. Less restrictive FPR target increases robustness of PN (TNRR/FNRR) for $c = .25$ and decreases for $c = 1.0$ for all models. FPRR decreases with less restrictive FPR except for M3 where robustness rates are already near zero for $c = 1.0$.

		M0				M1				M2				M3				M4				M5			
		$c = .25$		$c = 1.0$		$c = .25$		$c = 1.0$		$c = .25$		$c = 1.0$		$c = .25$		$c = 1.0$		$c = .25$		$c = 1.0$		$c = .25$		$c = 1.0$	
FPR		PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN	PP	PN
5.0E-4	P	97.8	67.0	**56.2**	**12.6**	35.0	63.2	1.2	30.6	96.8	64.2	**50.0**	**23.0**	37.4	62.4	2.2	37.2	87.4	60.6	1.2	13.4	93.8	62.6	32.8	29.4
	N	79.8	98.4	**14.0**	**68.2**	26.8	94.8	2.8	84.4	72.0	98.8	**9.6**	**77.4**	44.6	96.8	3.2	85.0	53.4	98.8	11.2	67.0	69.2	98.8	20.8	87.6
7.6E-4	P	98.6	68.8	**53.8**	**14.6**	35.6	65.8	1.6	33.2	97.2	65.6	**56.6**	**23.2**	37.0	65.2	2.2	29.8	88.6	63.8	2.0	16.2	93.6	67.0	35.6	30.4
	N	55.2	97.6	**10.4**	**66.2**	33.2	96.0	3.0	81.8	57.0	98.8	**5.6**	**76.4**	34.6	95.6	2.0	83.8	43.6	97.4	11.0	65.0	60.2	98.0	16.8	83.0
1.0E-3	P	97.4	71.6	**50.2**	**10.8**	37.6	67.2	1.6	26.4	97.2	68.4	**53.8**	**21.2**	36.2	68.0	1.8	27.4	88.4	67.4	2.0	16.4	92.6	68.8	34.6	28.8
	N	50.2	99.2	**7.4**	**58.0**	25.6	96.2	1.0	79.8	51.6	99.0	**8.4**	**70.8**	27.0	95.0	2.6	82.4	44.2	97.2	12.0	63.0	53.2	98.6	19.0	81.0
2.5E-3	P	97.2	77.8	**54.2**	**11.0**	37.6	76.8	1.4	24.4	97.6	71.6	**74.6**	**13.6**	44.6	76.8	3.6	28.0	89.4	70.6	4.4	13.2	92.2	76.2	46.6	25.4
	N	29.4	99.2	**3.6**	**52.0**	17.0	93.8	1.6	74.6	29.4	99.2	**7.0**	**60.0**	17.0	96.4	4.0	76.6	30.4	97.8	4.4	48.6	39.6	97.6	8.2	68.2
5.0E-3	P	96.0	83.6	52.6	8.6	37.0	82.4	2.0	22.0	95.2	80.0	**73.6**	**13.8**	40.2	79.8	3.4	20.0	91.0	79.4	7.2	7.4	93.4	79.4	47.6	21.6
	N	25.6	98.6	2.2	40.0	12.4	94.2	1.2	64.4	25.6	99.0	**4.0**	**56.2**	14.6	94.4	2.8	63.0	25.0	97.6	2.4	41.0	35.8	96.6	7.4	59.6
7.4E-3	P	96.2	86.0	55.6	8.8	50.6	80.2	1.8	16.6	93.6	83.4	73.4	13.2	46.2	84.8	3.8	19.0	91.8	76.6	9.0	7.0	93.6	83.2	47.6	19.0
	N	27.2	98.6	1.2	38.6	11.2	95.0	0.4	52.8	26.2	99.0	3.2	48.6	14.0	95.2	1.6	58.2	23.4	97.8	1.8	31.6	33.0	98.2	5.6	56.8
1.0E-2	P	93.6	86.4	52.4	7.0	47.2	80.8	4.2	12.2	93.4	81.8	75.6	11.6	45.4	87.2	5.2	17.2	95.0	76.4	36.2	7.2	94.4	83.6	54.2	16.2
	N	37.2	98.2	1.0	39.2	11.0	95.0	1.4	46.0	32.0	98.0	4.0	40.6	15.4	94.8	2.6	55.8	29.8	97.4	2.2	28.6	34.2	99.4	3.8	49.8

from clusters of positive and negative data points, meaning that a larger portion of them switch from robust to non-robust simultaneously than for any other model we test. True Negative Robustness Rate (TNRR) remains virtually constant across multiple FPR targets, providing evidence that all models tend to be more confident when providing negative predictions, which correspond to safe vehicle scans. Less restrictive FPR target increases the robustness of positive predictions for all models except M0 and M2, which adheres to our intuition that changing the decision boundary in a way that makes more data points be classified as positives (dangerous vehicle scans) means that points are gradually moving further away from the decision boundary and the model is newly exhibiting $(\mathbf{x}, \delta)$-LAR in those regions as a result. Closely related is the observation that FPRR decreases with less restrictive FPR except for M3 where robustness rates are already near zero for $c = 1.0$.

The fact that robustness is virtually consistent for all prediction outcomes other than true positives may be due to the fact that there is a class imbalance in the data. A true positive indicates a vehicle carrying improperly contained isotopes through the border, which is a very rare occurrence. Each of the six candidate models learns a decision policy that treats these rare occurrences sufficiently differently. The underlying distribution of class labels, or a class imbalance in data may affect levels of robustness.

We can combine the pairs of subfigures from Figure 3.9 to show what we define as the Positive Robustness Ratio (PRR) and Negative Robustness Ratio (NRR) in Figure 3.10. PRR is a ratio between the rate the a model adheres to $(\mathbf{x}, \delta)$-LAR for TPs vs FPs. These charts have similar intuition but differ from well-known ROC curves in that the lines represent possible values for c instead of possible predictive thresholds. This will allow us to determine the value of c that makes the model robust for correct predictions and not robust for incorrect predictions. We compute and report the AUC because in this context, it represents a dimensionless measure of the average adversarial perturbation that causes the model to change its prediction. An AUC value greater than 0.5 indicates that the model, on average, exhibits a higher rates of robustness for TPs/TNs than FPs/FNs for any select value of c.

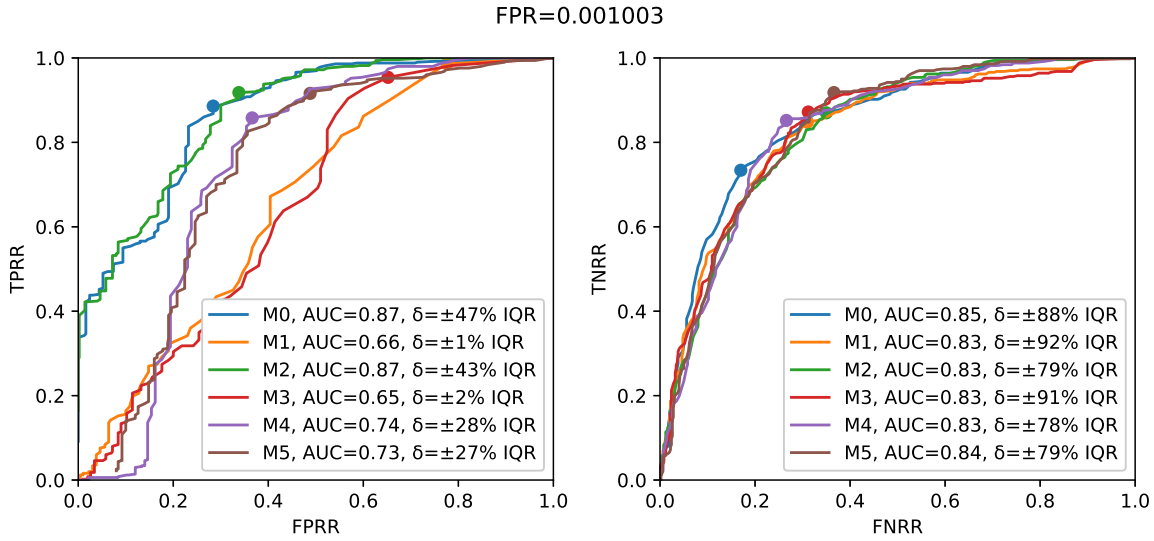
Models M0 and M2 have the highest AUC across most FPR targets. Figures 3.10a and 3.10b show PRR and NRR curves for FPR$= 1 \times 10^{-3}$ and FPR$= 1 \times 10^{-2}$ respectively. AUC increases for PRR charts as FPR becomes less restrictive, and we observe a decrease for NRR for the same change in FPR. The highlighted points on the charts denote the selection of c which maximizes the distance between the point and the bottom right of the chart ($[1.0, 0.0]$).

Table 3.6 shows similar information for additional FPR targets. We also report the Positive Predictive Value (PPV), which tells us how many times more likely it is

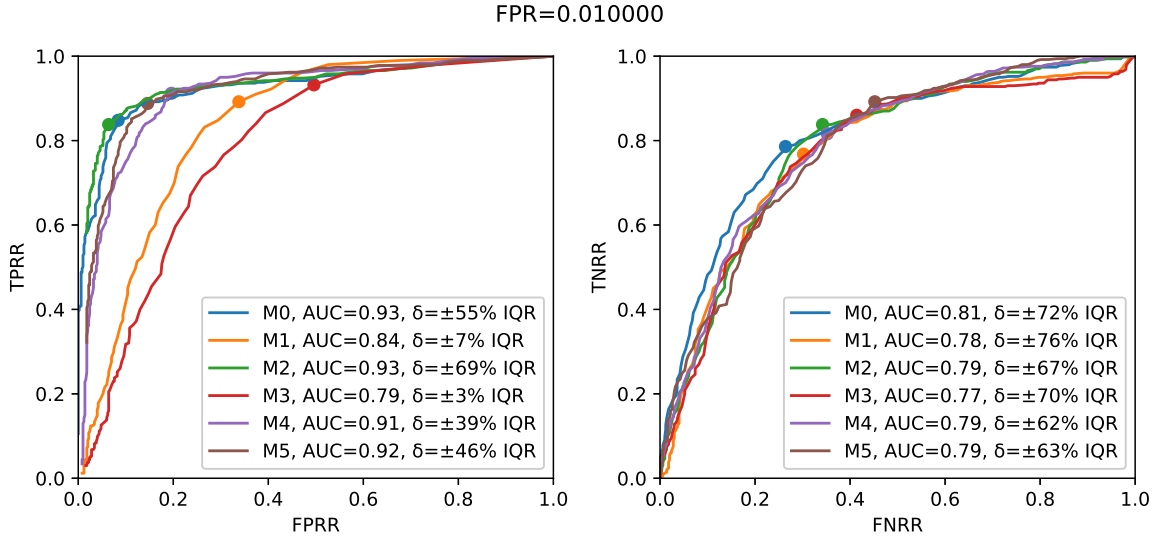
for the model to be robust to adversarial perturbations at an untested sample if it is a correct prediction vs an incorrect prediction. We find that the highest PPVs are present for M2 and M0, which indicates that these models are much more likely to exhibit the specific type of selective robustness that we desire

We conjecture that the most useful way for our method of checking LAR across FPR targets to be used is to characterize the average distance to the decision boundary for prediction outcomes for tree ensemble models. This is not an easy task for tree ensembles, due to their piece-wise constant nature. Understanding whether correct or incorrect predictions are closer to the decision boundary helps give an intuition for how multiple models act in comparison to one another. Given that it is desirable for correct predictions to be far from the decision boundary (confident) and incorrect predictions to be close to a decision boundary (not confident), we can use this additional information we obtain with TEA to help break ties when selecting a single trained ensemble from a set of candidates. Furthermore, this also gives us additional details into what is happening to the learned model structure as the target FPR is changed. Target FPR plays a role in the robustness that a model exhibits, and there may be cases where allowing for a higher FPR is desirable if it allows the model to yield more robust output. This means that the resulting model will be less sensitive to perturbations on inputs, which is a desirable property for critical AI systems.

In this section of the thesis, we showed a few use cases of TEA in a radiation safety context. We characterized the extent of robustness for a given input by performing a binary search over certificates to find the scalar multiple of δ such that LAR is still satisfied. We examined robustness at the intersection of prediction outcomes which takes typical robustness tests in the formal methods and automated reasoning community makes LAR immediately relevant to data scientists. We have shown that robustness can serve as a tie-breaker when selecting a trained model for deployment. Similar overall accuracy does not tell us about the manner in which each model produces its predictions, so LAR can be assessed and then used as a desirable contractual property of a model to be selected for deployment. We also show that it is possible to use TEA to extract a formal definition for LAR such that the engineering desiderata of only being robust when correct is best met. This can be of utility in cases where a formal set of requirements do not exist for a given application domain, but developers of AI systems still want to formally test and analyze the differences between multiple candidate models.



(a) Positive Robustness Ratio (PRR) and Negative Robustness Ratio (NRR) curves for FPR=1×10^{-3}



(b) PRR and NRR curves for FPR=1×10^{-2}

Figure 3.10: Positive Robustness Ratio (PRR) and Negative Robustness Ratio (NRR) curves for select FPR targets. Each point is a unique c for $(\mathbf{x}, c\delta)$-LAR. x-axis shows rate of robustness exhibited by the model on FPs/FNs and y-axis shows rate of robustness exhibited by the model on TPs/TNs. Area Under the Curve (AUC) is a dimensionless measure of the average adversarial perturbation needed to turn TP→FN and FP→TN. $x = y$ denotes equal robustness rates for FPs/FNs and TPs/TNs. $x < y$ denotes higher rates of robustness among correct (T) predictions and $x > y$ denotes higher rates of robustness among incorrect (F) predictions. Marked points denote the value of c that maximizes distance to bottom right $[1.0, 0.0]$.

Table 3.6: For select FPR targets, the AUC, and selection of c that maximizes PRR and NRR. Each target FPR row is an abbreviated description of a chart like Figure 3.9. As target FPR increases, AUC increases for positive predictions and decreases for negative predictions. Best selection of c says relatively constant across target FPRs. Models M0 and M2 exhibit highest degree of AUC for positive predictions. M4 exhibits high AUC for negative predictions.

	M0				M0				M1				M1			
FPR	PP AUC	δ c^*IQR	PRR (TPRR/FPRR)		PN AUC	δ c^*IQR	NRR (TNRR/FNRR)		PP AUC	δ c^*IQR	PRR (TPRR/FPRR)		PN AUC	δ c^*IQR	NRR (TPRR/FPRR)	
5.0E-04	0.80	1.50	inf	(36.2/0.0)	0.88	0.77	3.5	(87.6/25.0)	0.57	0.00	1.0	(100.0/100.0)	0.82	0.91	2.4	(89.0/36.6)
7.6E-04	0.86	0.47	3.2	(90.0/27.8)	0.84	0.79	3.3	(83.2/25.2)	0.58	1.74	inf	(0.4/0.0)	0.81	0.71	2.0	(92.8/47.4)
1.0E-03	0.87	0.47	3.1	(88.6/28.4)	0.85	0.88	4.3	(73.4/17.0)	0.66	0.01	1.3	(98.6/77.6)	0.83	0.92	2.7	(84.8/31.0)
2.5E-03	0.93	0.47	6.0	(87.8/14.6)	0.85	0.71	2.8	(87.6/31.0)	0.77	0.03	1.9	(93.8/50.4)	0.81	0.90	2.7	(83.4/31.0)
5.0E-03	0.94	0.49	8.2	(88.8/10.8)	0.83	0.69	2.7	(82.8/30.4)	0.81	0.03	2.1	(95.4/45.2)	0.78	0.79	2.4	(83.0/34.6)
7.4E-03	0.95	0.41	7.2	(92.0/12.8)	0.81	0.77	3.3	(76.2/23.2)	0.85	0.10	3.2	(82.2/25.4)	0.78	0.76	2.4	(82.8/34.8)
1.0E-02	0.93	0.55	10.1	(84.8/8.4)	0.81	0.72	3.0	(78.6/26.4)	0.84	0.07	2.6	(89.2/33.8)	0.78	0.76	2.5	(76.8/30.2)

	M2				M2				M3				M3			
FPR	PP AUC	δ c^*IQR	PRR (TPRR/FPRR)		PN AUC	δ c^*IQR	NRR (TNRR/FNRR)		PP AUC	δ c^*IQR	PRR (TPRR/FPRR)		PN AUC	δ c^*IQR	NRR (TPRR/FPRR)	
5.0E-04	0.82	0.55	2.6	(81.4/31.8)	0.85	0.86	2.9	(85.8/29.6)	0.51	1.80	inf	(0.4/0.0)	0.80	0.76	2.1	(92.8/44.8)
7.6E-04	0.88	0.48	3.0	(89.2/29.4)	0.84	0.70	2.4	(91.6/37.6)	0.59	0.01	1.2	(98.4/81.8)	0.82	0.95	2.6	(86.6/32.8)
1.0E-03	0.87	0.43	2.7	(91.8/33.8)	0.83	0.79	2.5	(87.0/35.0)	0.65	0.02	1.5	(95.4/65.2)	0.83	0.91	2.8	(87.2/31.2)
2.5E-03	0.92	0.44	5.3	(93.6/17.6)	0.83	0.79	2.9	(79.2/27.2)	0.74	0.01	1.3	(99.2/74.8)	0.82	0.83	2.4	(87.8/36.8)
5.0E-03	0.94	0.55	10.0	(87.8/8.8)	0.82	0.76	3.2	(79.2/24.6)	0.77	0.02	1.6	(97.6/59.8)	0.79	0.84	2.6	(81.4/30.8)
7.4E-03	0.93	0.63	14.4	(86.4/6.0)	0.80	0.83	3.2	(70.8/22.0)	0.81	0.05	2.2	(90.6/41.6)	0.78	0.64	2.0	(88.4/44.8)
1.0E-02	0.93	0.69	13.1	(83.8/6.4)	0.79	0.67	2.5	(83.8/34.2)	0.79	0.03	1.9	(93.2/49.6)	0.77	0.70	2.1	(86.0/41.4)

	M4				M4				M5				M5			
FPR	PP AUC	δ c^*IQR	PRR (TPRR/FPRR)		PN AUC	δ c^*IQR	NRR (TNRR/FNRR)		PP AUC	δ c^*IQR	PRR (TPRR/FPRR)		PN AUC	δ c^*IQR	NRR (TPRR/FPRR)	
5.0E-04	0.61	0.23	1.7	(89.2/53.4)	0.87	0.77	3.6	(84.8/23.8)	0.69	0.23	1.4	(95.8/69.2)	0.85	0.93	3.0	(89.6/30.0)
7.6E-04	0.71	0.20	1.9	(92.0/48.0)	0.84	0.65	2.5	(87.6/34.6)	0.75	0.41	2.4	(84.2/35.2)	0.82	0.87	2.5	(89.2/35.8)
1.0E-03	0.74	0.28	2.3	(85.8/36.6)	0.83	0.78	3.2	(85.2/26.6)	0.73	0.27	1.9	(91.6/48.8)	0.84	0.79	2.5	(91.8/36.6)
2.5E-03	0.85	0.32	4.0	(83.6/21.0)	0.81	0.65	2.3	(86.4/37.0)	0.85	0.46	4.4	(79.2/18.2)	0.80	0.80	2.3	(87.0/38.2)
5.0E-03	0.90	0.33	6.0	(86.2/14.4)	0.82	0.69	2.8	(82.6/29.8)	0.87	0.43	4.0	(85.2/21.2)	0.76	0.73	2.0	(84.4/41.6)
7.4E-03	0.90	0.33	5.5	(87.6/15.8)	0.79	0.81	3.7	(62.4/17.0)	0.89	0.43	4.8	(88.4/18.4)	0.78	0.64	1.8	(90.6/49.0)
1.0E-02	0.91	0.39	4.7	(91.2/19.6)	0.79	0.62	2.3	(81.6/35.2)	0.92	0.46	6.1	(88.8/14.6)	0.79	0.63	2.0	(89.2/45.2)

Chapter 4

Verification of a Global Adversarial Robustness (GAR) Specification

Global Adversarial Robustness (GAR) describes smoothness of system outputs for all possible pairs of similar inputs. A system satisfies (δ, ϵ)-GAR if it yields two sufficiently similar ($\leq \epsilon$) outputs for two sufficiently similar inputs ($\leq \delta$). In our formalism, similarity of outputs is defined as the difference in the vote tally among trees in the ensemble, however, other measures of risk assessment would suffice. Conceptually, GAR is a logic-based proxy for smoothness. Verifying GAR would tell us the maximal extent to which perturbations on inputs would ever change the votes cast by trees in the ensemble.

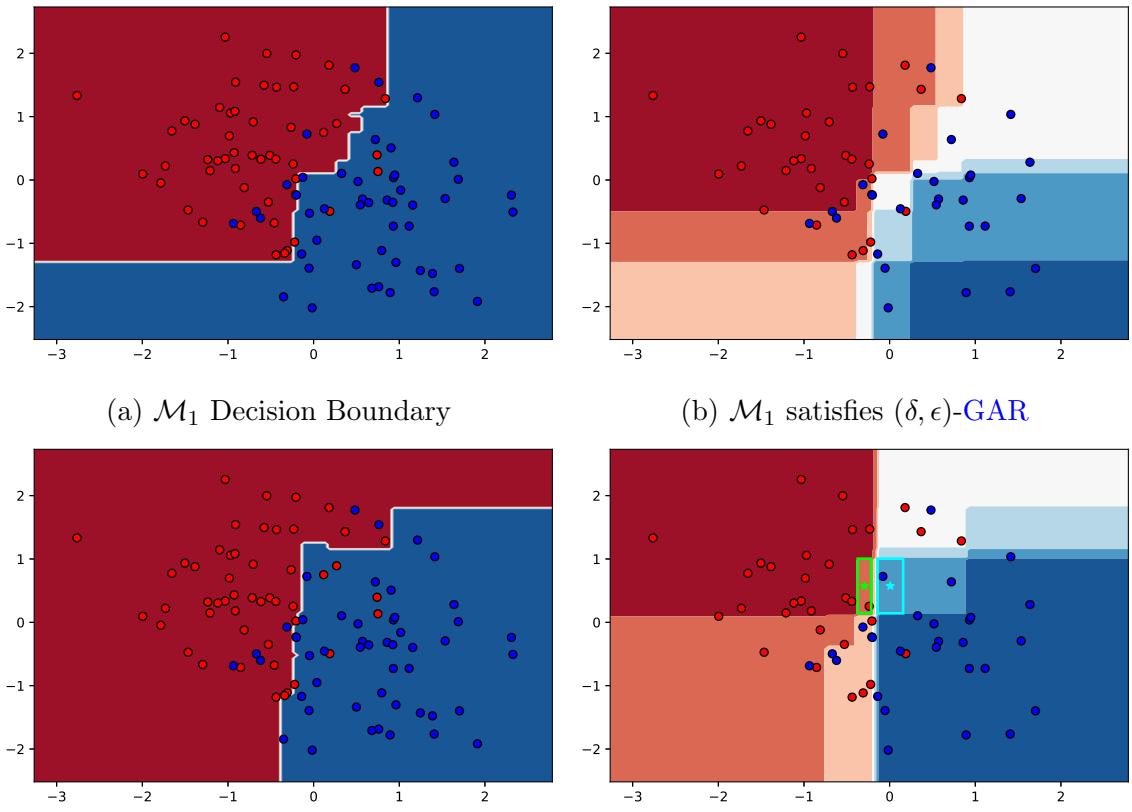
GAR has been described in literature as an extension to LAR, but, to the best of our knowledge, has never been verified without imposing significant, simplifying assumptions on the problem. We are the first to report a tractable strategy for verifying GAR for voting tree ensemble models. Certifying that a model adheres to a global property greatly increases the scope of the verification task. This means that many formalisms for verifying LAR cannot be extended to checking GAR. TEA is able to be extended to verify GAR by encoding two copies of the same model and imposing constraints between the model states. TEA is capable of identifying counterexamples to GAR where confidence in the predicted label varies drastically between two points, even if the overall prediction is the same.

Preliminary experiments in the next section provide a visual interpretation for GAR certificates. We describe our methods and provide evidence that our method scales to verification instances of application-scale models. We apply TEA in an algorithmic fairness context, where we are interested in verifying that a trained model adheres to an Individual Fairness (IF) specification. IF is verifiable within our GAR formalism.

We show that GAR can aid in model selection by identifying the model that adheres to the strictest definition of GAR, allowing developers of AI systems to select a model for deployment that exhibits strong predictive accuracy as well as adherence to select algorithmic fairness considerations. When a model violated the fairness specification, we show that we can enumerate all counterexamples to GAR under certain conditions. There are many counterexamples that exist simply by being close points on opposite sides of the decision boundary, but egregious counterexamples, where a majority of the trees in the ensemble change their vote simultaneously, manifest less frequently. These are the types of GAR violations that we are most interested in characterizing. Each counterexample provides operational ranges over inputs where the instance of unfair model behavior manifests. Interpreted together, these counterexamples can reveal the structure of unfairness absorbed by the model during the training phase. This informs users of when their model is at risk of making a prediction that will violate the GAR specification. We discuss limitations of our formalism and present ongoing work for addressing one of the challenges associated with properly scoping GAR in Chapter 6.

4.1 Illustrative Examples

4.1.1 Interpreting a (δ, ϵ)-GAR Certificate



(a) $\mathcal{M}_1$ Decision Boundary

(b) $\mathcal{M}_1$ satisfies (δ, ϵ)-GAR

(c) $\mathcal{M}_2$ Decision Boundary

(d) $\mathcal{M}_2$ violates (δ, ϵ)-GAR

Figure 4.1: Satisfying and violating (δ, ϵ)-GAR certificates.

Tree ensemble models are piece-wise linear, making it difficult to characterize the smoothness of their transitions along the decision surface. However, we have a tally of votes case by the trees in the ensemble, and we can define smoothness of the decision surface by the rate at which those votes change between two points in the feature space. By defining and verifying a smoothness specification for a trained model, we are enforcing that no more than ϵ trees change their votes between two points picked within the same δ-neighborhood. For a given δ-neighborhood, an ensemble where no more than 25% ($\epsilon = .25$) of its trees change their votes satisfies a stricter definition of (δ, ϵ)-GAR than an ensemble where no more than 75% ($\epsilon = .75$) of its trees change their votes. The minimum satisfiable value of ϵ is the level of (δ, ϵ)-GAR that the ensemble exhibits. Defining δ to be sufficiently large would be uninteresting because

this would allow the search for counterexamples to include points that are very far away from one another, yet deemed similar due to being separated by less than δ. The definition of δ may be informed by expert knowledge or by searching for the largest definition of δ such that the model adheres to (δ, ϵ)-GAR.

Subfigures 4.1a and 4.1c show two different random forest models (10 decision trees of max depth 3) trained on the same synthetic data (100 samples using sklearn's `make_blobs`() function). At first glance, both subfigures 4.1a and 4.1c seem like appropriate learned decision boundaries for the given red/blue classification task. However, M1 and M2 differ in that M1 adheres to a smoothness specification, whereas M2 violates the same specification. The top rows show that there does not exist two points that violate this specification for M1. The bottom row shows the existence of two points that violate this specification for M2 with the counterexample highlighted. Subfigure 4.1d shows a cyan and green box with two samples starred on the inside. These boxes denote the existence of two points that are separated by less than $\delta = 0.1$ where $\epsilon = 0.6$ (or 60%) of the trees do not cast the same vote for each point. No such region exists in subfigure 4.1b, where the reader can visually confirm the greater amount of gray area denoting that there is a smoother transition between deep red and blue states than in subfigure 4.1d.

Formally verifying an upper bound on smoothness of a model may be a useful certificate to foster understanding of the topology of the decision surface as well as earn the trust of practitioners or even end users. Given that M1 passed verification, we can guarantee that no more than 60% of the trees in M1 will cast different votes for two points separated by $\delta = 0.1$, over the entire input space. Given that M2 failed verification, we can guarantee that there is an instance where 60% of the trees in M2 cast different votes for two points separated by $\delta = 0.1$. If we are interested in characterizing smoothness rather than simply providing the certificate for $\epsilon = 0.6$, then we may try to verify M2's smoothness for the same delta but for $\epsilon \in (0.6, 1.0]$.

4.2 Encoding Strategy

Global Adversarial Robustness (GAR) places a bound on the extent to which the model prediction changes between any two similar points. We define this concept for voting ensembles as follows: for a given ensemble, let vector $\mathbf{v}(\mathbf{x})$ be the tally of votes for each class cast with input $\mathbf{x}$. An ensemble satisfies (δ, ϵ)-GAR if, for all inputs $\mathbf{x}$ and $\mathbf{x}'$ such that $\{|x_i - x'_i| < \delta_i \mid \forall i \in \delta\}$, $\|\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{x}')\|_\infty < \epsilon$. Any pair $(\mathbf{x}, \mathbf{x}')$ that violates this property is an adversarial example. When possible, it is useful to limit the search to a subset of the inputs that may actually appear in application, as anomalous outliers often act as counterexamples to otherwise robust models.

4.2.1 Encoding (δ, ϵ)-GAR

We need to verify the existence of two similar samples that receive sufficiently different vote tallies from decision trees in the ensemble. Since our ordinal constraints prevent the activation of more than one leaf in a decision tree at a time, we must define our verification task with two copies of the model encoding and add constraints that relate them to each other. This lets each ensemble configure itself freely, and then a few additional constraints relate those model states with one another. Literals belonging to the second model encoding are distinguished with the prime mark ($'$).

The corresponding input constraints resemble the threshold ordinal constraints, but spanning across the two models, with a "skip" of δ. That is, if any threshold literal T_i is true, then $x_{a(i)} \leq t(i)$, so we expect $x'_{a(i)} < t(i) + \delta_i$, so $T_i \implies T'_j$ for all j such that $t(j) \geq t(i) + \delta_i$. The procedure for asserting these constraints that constitute δ is detailed in the first half of Algorithm 9.

The output constraints assert that the number of votes for a particular class must be different by an amount determined by ϵ, and the strategy for doing so is described in the second half of Algorithm 9. To simplify, we make this assertion for *only* one class label, and evaluate a separate SAT instance for each class to determine whether global robustness is violated for any of them. We additionally assume without loss of generality that the second model encoding has the greater number of votes for the class of interest. Let N be the number of trees, let $k = \lceil \epsilon N \rceil$ be the minimal difference in votes to comprise an adversarial pair, and let c be the class of interest. First, since the second model must have at least k more votes, we constrain that the first model has at most $N - k$ votes and the second has at least k votes. Then, for all i, we have that, if the first model has at least i votes, then the second must have at least $i + k$.

An `UNSAT` certificate verifies the (δ, ϵ)-GAR specification; globally, there does not exist any equal to or greater than ϵ differences in output for input differences less than

Algorithm 9: Global Adversarial Robustness (GAR)

```
   /* Assert the δ constraint over inputs                                  */
 1 𝕄 ← indices of tree models within tree ensemble
 2 ∀m ∈ 𝕄,  𝔹_m ← indices of branch nodes in model m
 3 I ← argsort_i{t(i) | i ∈ 𝔹_m, m ∈ 𝕄} /* ascend thresh value idxs       */
 4 for att ← 1 to number of attributes do
 5 |  I' ← (i ∈ I | a(i) = att) /* sorted thresh idxs for attribute       */
 6 |  j ← 1
 7 |  for i ← 1 to |I'| do
 8 |  |  while j ≤ |I'| ∧ t(I'_j) < t(I'_i) + δ_i do
 9 |  |  |  j ← j + 1
10 |  |  end
11 |  |  if j ≤ |I'| then
12 |  |  |  assert T_{I'_i} ⟹ T'_{I'_j}
13 |  |  |  assert T'_{I'_i} ⟹ T_{I'_j}
14 |  |  end
15 |  end
16 end
   /* Assert the ϵ constraint over outputs                                 */
17 c ← class to test for adversarial pairs
18 N ← number of trees in the forest
19 k ← ⌈ϵN⌉
20 for i ← 1 to k do
21 |  assert ¬S_{N−i+1,N,c} /* first count upper bound                     */
22 |  assert S'_{i,N,c} /* second count lower bound                        */
23 end
24 for i ← 1 to N − k do
25 |  assert S_{i,N,c} ⟹ S'_{i+k,N,c} /* k-vote gap between counters       */
26 end
```

δ. If the specification cannot be met, a `SAT` certificate provides a counterexample.

4.3 Baseline Comparison

We present results for verifying (δ, ϵ)-GAR with TEA on the same vehicle collision dataset and MNIST dataset on which we tested $(\mathbf{x}, \delta)$-LAR. There are no direct baselines of comparison for TEA when verifying (δ, ϵ)-GAR. VoTE cannot scale to this verification task. Reluplex [109] discusses GAR but determines that it is not tractable in the SMT formalism for models of application scale.

The closest baseline is [99], who verify that a linear, polynomial, and RBF kernel support vector machine adheres to an Individual Fairness (IF) specification. In their formalism, they are only searching for counterexamples where the model produces a different classification label. In contrast, TEA is capable of identifying a counterexample of (δ, ϵ)-GAR where confidence in the predicted label varies drastically between two points, even if the overall prediction is the same. The vote tallies between two points in a binary classification problem could be $[N, 0]$ and $[\frac{N}{2} + 1, \frac{N}{2} - 1]$, where N is the number of trees in the ensemble. While the predicted label may be consistent, the confidence in the prediction is significantly different, and TEA is, to the best of our knowledge, the only reported formalism that can identify these types of violations to (δ, ϵ)-GAR.

4.3.1 (δ, ϵ)-GAR for Vehicle Collision

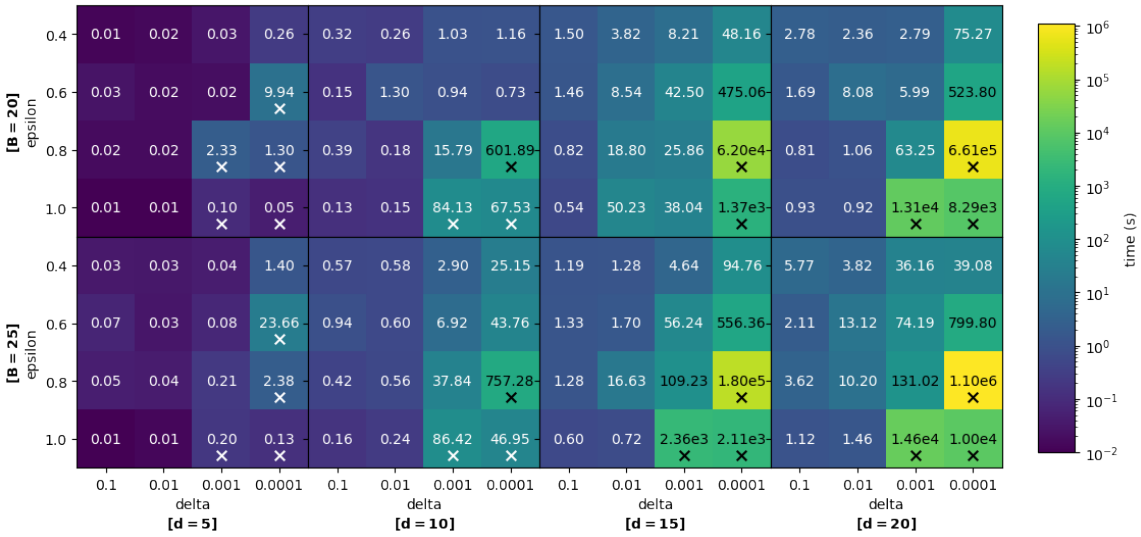


Figure 4.2: Global robustness tests of random forests trained on the vehicle collision dataset.

Recall that the task in this context is for tree ensembles to detect collisions between

simulated vehicle trajectories and that the data generating process is identical to that of [187]. Figure 4.2 shows the result and solver time for each test of (δ, ϵ)-GAR for different model configurations. Each 4 by 4 block depicts a single model with max depth d and number of trees B. For each model, we test global robustness with various δ, neighborhood, and ϵ, similarly. Cell background color represents time to solve shown in the scale on the right. Sets of parameters that result in an ensemble that meets global adversarial robustness specifications are marked with an 'x'. Across all d and B, we find robustness only for very low δ (small neighborhood) and high ϵ (large difference in votes), suggesting that global robustness is rare if you do not train a model to adhere to it. Generally, the time to verify is reasonable even for large models when the result is a counterexample proving a lack of robustness. However, when the result is that the model is robust, verification can take a long time, up to nearly two weeks for the largest model. It appears that, within each of these two cases, the longest times occur when ϵ and δ are near the boundary between robust and not robust. This suggests that, if solving time becomes prohibitive near the boundary, we may still achieve results farther from the boundary, giving an approximate characterization of the model's global robustness. Since the cost of verification is all upfront and happens before the model is deployed, we recommend using the maximum allowable time budget. The reason being that the solver does not identify counterexamples in a random fashion, but rather through a depth-first search, meaning that stopping early may return a biased sample of counterexamples. Future work could be focused on providing the solver with warm starts or random starts in order to decrease the bias of counterexamples found when a timeout is present.

Even with extended solver times, these results still represent the first time verification of global adversarial robustness has been reported for any trained learning model of realistic application scale. Furthermore, this is the first time that such strategies for verifying global adversarial robustness have been demonstrated for voting tree ensemble models. Interesting directions for future work would be to examine how training methods intended to encourage robustness might affect global robustness. While we have shown that we are able to verify whether a trained model adheres to a global adversarial robustness specification, it remains to be seen how to train a model such that the specification is met.

4.3.2 (δ, ϵ)-GAR for MNIST

Recall that in this study, tree ensemble models classify handwritten digits from the MNIST dataset [116]. This includes 70,000 gray-scale 28×28 images with integer pixel values between 0 and 255, which we split randomly into 85% training and 15% test according to the procedure of [187]. We test GAR for each model configuration

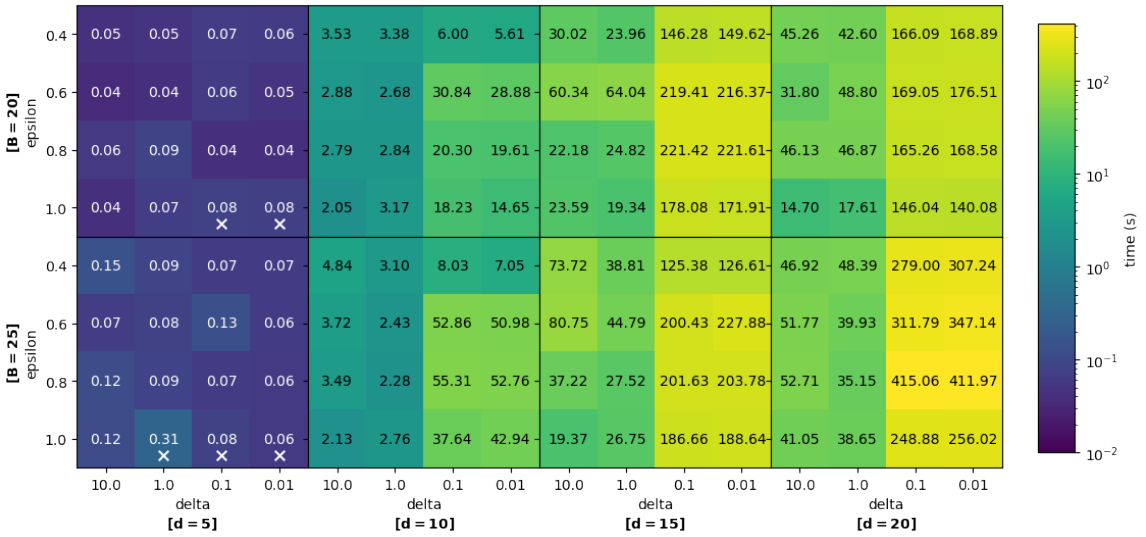


Figure 4.3: Global robustness tests of random forests trained on the MNIST dataset. See Figure 4.2 for a description of the global robustness visuals.

with various δ and ϵ. Figure 4.3 shows the result and solver time for each test.

For this dataset, we find global robustness only for very shallow forests ($d = 5$) with $\epsilon = 1$, an extreme case where the model produces a unanimous vote at one point in the adversarial pair, but has zero votes for that same class at the other point. Deeper forests are not found to be robust for any ϵ and δ. We conjecture that the reason for this is twofold. In large ensembles, there are many possible ways to generate very similar adversarial pairs, as changing the assignment of a single literal in the ensemble has a small net effect on the overall satisfying assignment. In small ensembles, there are fewer ways to generate adversarial pairs, as changing the assignment of a single literal has a larger net effect on the overall satisfying assignment. Second, the ensemble partitions the input space into HRs, each of which has constant output, so the model will never be globally robust for ϵ less than that of of any pair of adjacent HRs, since arbitrarily close inputs can be taken from each. For a high-dimensional dataset like MNIST, because of the high number of adjacencies that exist, it might be common for there to exist maximally extreme differences in output for adjacent HRs, meaning that global robustness never holds for any ϵ and δ. This highlights the importance of robustness-aware training procedures for tree-based models.

4.4 Utility of GAR in Algorithmic Fairness

We apply TEA to verify that trained models adhere to an IF specification in order to show that our formalism is able to address open questions in a fast growing research community. AI and ML are increasingly being deployed in high impact settings such as criminal justice, clinical decision making, and financial assistance. It has been shown that their use in these settings can reproduce biases present in data and reinforce societal discrimination. Algorithmic fairness attempts to address these concerns using mathematical notions of fairness. We focus on one such notion: IF, which requires that similar individuals receive similar outcomes from a trained model. We frame IF as a design specification, expressible within our GAR formalism, and use formal methods to verify this property in trained models. To maintain clarity in the context of algorithmic fairness, we will define a new acronym, Global Individual Fairness (GIF), which equivalent to (δ, ϵ)-GAR. Formal methods are able to reason about model structure directly, enabling global verification both inside and outside data support.

The most related model verification method is [99], that formally verifies an IF specification on support vector machines. Our notion of IF is uniquely amenable to risk assessment contexts, where we can check for changes in predicted labels between points, or check for significant change with constant label, which translates to the difference in risk assigned to each sample. Our formalisms also differ in that we leverage SAT technology, while John [99] leverage quadratic programming via CPLEX.

We demonstrate verification of GIF for voting tree ensembles trained on the publicly available census income dataset. Our experiments illustrate three applications of GIF verification in an existing data pipeline, complementing existing statistical methods. First, we use GIF as a model selection criterion. Second, we use GIF counterexamples to reveal the structure of bias absorbed by a trained model. Third, we verify the operational conditions under which a trained model can be certified to meet GIF, providing new information for online decision support applications. These capabilities can assist designers and users of AI systems to understand, manage, and mitigate the presence of biases in fielded AI systems.

Multiple tree ensemble models are trained on the Adult (Census Income) dataset, publicly available on the UCI Machine Learning Repository [52]. A binary classification task aims to predict whether a person earns more than $50,000 a year. Minor modifications to the dataset include dropping attributes for `education`, as `education-num` encodes the same information in integer form, and `fnlwgt`. We do not expect that these made any meaningful changes to our experimental findings. Since our primary goal is to verify that models adhere to individual fairness considerations and show what we can do with this information, we do not spend much time

on training the best possible models; scikit-learn Random Forests comprising 10 binary decision trees of maximum depth 5 trained on a random 10% sample of the data (approximately 5,000 samples) and tested on the remaining 90%. Such a partition was selected in order to have ample data for testing the ability of our framework to make fault-aware predictions. We train multiple tree ensembles with different random seeds, which yields a set of models that all achieve similar levels of overall test accuracy, but do so using unique decision logic. All experiments were run on Intel Xeon Gold 6238 CPUs.

4.4.1 Selecting the fairest model with (δ, ϵ)-GIF

We first examine the strictest definition of (δ, ϵ)-GIF that can be satisfied for a particular model, which we find via conducting parallel verification tasks. Multiple tree ensemble models are trained, differing in random seed, in order to show the utility of these certificates. Different models may adhere to different (δ,ϵ)-GIF specifications, and knowledge of which models are more/less fair according to this type of specification can make the model selection process fairness-informed. In tree ensemble models, ϵ takes on discrete values quantized by single decision tree vote changes. For every consideration of δ-similarity between inputs, we test all possible values of ϵ. This approach identifies the change point from `SAT` (property violated by counterexample) to `UNSAT` (property holds). With this result in hand, we may reason contractually about which models are more fair than the others, presenting us with valuable information that can be used during the model selection phase.

For each model and formal definition of (δ, ϵ)-GIF under consideration in Table 4.1, we show the strictest definition of ϵ for which a model/spec pair is certified fair. Columns denote the definition of δ, rows correspond to individual trained models. Abbreviated column headings in the second table are given by unabridged headers in the top table. The model ID and test accuracy are listed in the table.

The value of ϵ in each cell denotes the maximum fraction of trees in the ensemble that change votes for any pair of δ-similar individuals. For cells with a dash, this denotes that there exist counterexamples such that all trees in the ensemble change their votes from one class to another simultaneously for two δ-similar individuals; thus no definition of (δ, ϵ)-GIF could be met. We see that even for models that only change by the random seed used during the training procedure, they each satisfy different definitions of (δ, ϵ)-GIF. The most accurate models are often not the most fair. When considering single attributes as the set of protected attributes encoded by δ, we find that the model is generally more fair than when considering multiple attributes, which matches our intuition.

Table 4.1 shows the ability to contractually reason about which model of a can-

Table 4.1: Strictest (δ, ϵ)-GIF for each model. '-' indicates a counterexample where all trees flip votes.

Q1: Minimum-ϵ Certified (δ, ϵ)-GIF Considerations														
Model ID	Test Accuracy	WORK CLASS (δ WC)	MARITAL STATUS (δ MS)	OCCUP-ATION (δ OC)	RELATI-ONSHIP (δ RP)	RACE (δ RC)	SEX (δ SX)	NATIVE COUNTRY (δ NC)	δ WC or MS	δ WC or OC	δ WC or RP	δ WC or RC	δ WC or SX	δ WC or NC
0	82.57	0.7	0.9	0.9	0.8	0.4	0.5	0.6	-	-	-	0.9	0.9	0.9
1	82.74	0.7	0.8	0.9	0.9	0.6	0.5	0.7	-	-	0.9	0.9	0.8	0.9
2	81.18	0.6	0.8	0.8	0.6	0.4	0.4	0.6	-	0.8	0.8	0.7	0.6	0.8
3	82.13	0.7	0.9	0.8	0.8	0.6	0.4	0.5	-	0.9	0.9	0.7	0.7	0.9
4	82.96	0.6	0.8	0.8	0.9	0.4	0.5	0.7	0.9	-	-	0.7	0.9	0.9
5	82.98	0.6	-	-	0.8	0.3	0.5	0.5	-	-	-	0.7	0.8	0.6
6	83.84	0.6	-	-	0.7	0.4	0.5	0.4	-	-	0.8	0.7	0.7	0.7
7	82.36	0.5	0.6	-	0.9	0.5	0.6	-	0.8	-	0.9	0.6	0.8	-
8	83.25	0.6	-	0.9	0.8	0.3	0.3	0.7	-	-	0.8	0.6	0.7	0.9
9	82.72	0.5	0.6	-	0.9	0.4	0.5	0.7	0.8	-	0.9	0.6	0.7	-

Q1: Mininum-ϵ Certified (δ, ϵ)-GIF Considerations																
Model ID	Test Accuracy	MS δ or OC	MS δ or RE	MS δ or RC	MS δ or SX	MS δ or NC	OC δ or RP	OC δ or RC	OC δ or SX	OC δ or NC	RP δ or RC	RP δ or SX	RP δ or NC	RC δ or SX	RC δ or NC	SX δ or NC
0	82.57	-	-	-	0.9	-	-	-	-	-	0.9	-	0.9	0.7	0.8	0.8
1	82.74	-	-	0.9	0.9	-	-	0.9	0.9	0.9	-	0.9	0.9	0.8	0.7	0.8
2	81.18	-	-	0.9	0.9	-	0.9	0.8	0.8	0.9	0.8	0.7	0.8	0.6	0.7	0.6
3	82.13	-	-	-	0.9	-	0.9	0.9	0.8	0.9	0.8	0.8	0.9	0.8	0.7	0.7
4	82.96	-	-	0.8	0.9	-	-	-	-	-	-	-	-	0.6	0.9	0.8
5	82.98	-	-	-	-	-	-	-	-	-	0.8	0.8	-	0.6	0.6	0.6
6	83.84	-	-	-	-	-	-	-	-	-	0.8	0.7	0.7	0.6	0.6	0.6
7	82.36	-	-	0.9	0.8	-	-	-	-	-	0.9	-	-	0.7	-	-
8	83.25	-	-	-	-	-	-	-	0.9	-	0.8	0.9	-	0.5	0.8	0.8
9	82.72	-	-	0.7	0.7	-	-	-	-	-	0.9	0.9	-	0.6	0.9	0.8

didate set meets the strictest definition of (δ, ϵ)-GIF We use the term strictness to describe the value of ϵ, the fraction of trees that simultaneously change their votes for a given δ, which is defined as: all attribute values held equal except for attributes named in the columns of the table. These candidate definitions for δ are determined manually. Practitioners may wish to define δ based on suspected biases that are encoded in the model; the verification framework then will tell them whether such bias exists. For example, Figure 4.4a represents a design choice on behalf of the authors; if a trained model were to believe that the chances an individual makes $> \$50,000$ annually at times depend solely on differences in race and sex, this would represent a negative bias that we wish to analyze by enumerating the faults that lead to this failure in fairness. For demonstration purposes, in Table 4.1, we take a more algorithmic approach and simply look at all one- and two-dimensional considerations so that results of our analysis fit on one page, but of course similar process can be applied to more complex statements. Our framework can support much higher dimensional fairness considerations as well as notions of δ-similarity among integer or real-valued attributes. Potential utility exists in the sense that practitioners can define δ to encode their own hypotheses for undesirable bias, and then use our framework to find out where such bias arises.

Ensemble design parameters are chosen that result in a spread of (δ, ϵ)-GIF definitions being satisfied by different models. For the results in Table 4.1, our ensembles are trained with ten decision trees of maximum depth 5. No pruning procedure beyond limiting maximum depth of trees is implemented.

The relationship between satisfiable (δ, ϵ)-GIF and ensemble design parameters may be counter-intuitive. Controlling for the number of trees in the ensemble and varying the maximum depth of the trees, we find that ensembles with shallower trees tend to adhere to stricter definitions of (δ, ϵ)-GIF. For a relatively long decision path, there exist more alternate subtrees to test for (δ, ϵ)-GIF and it is relatively easier to find a counterexample in any one of them, which is enough for the ensemble to fail the verification task. This brings us to an interesting property, which is that notions of overall accuracy measures do not align with notions of (δ, ϵ)-GIF, or more generally, model robustness. To illustrate, the model that will adhere to the strictest definition of (δ, ϵ)-GIF is a model that has constant output globally because it will always treat individuals identically, regardless of their similarities or differences. Thus, it is an important distinction to note that our goal is to strike a balance between maximizing overall accuracy measures and maximizing fairness in a pareto-optimal way, because optimizing just a single objective will not always produce models useful in practice.

When controlling for depth and varying the number of trees in the ensemble, we find that ensembles with more trees tend to satisfy stricter definitions of (δ, ϵ)-GIF.

We hypothesize that this is due to the fact that each tree is trained on a random subset of data, and the learned decision boundaries, while similar across trees, are still unique, resulting in a smoother transition of the vote tally between labels in a binary classification task. Furthermore, we believe that these learned splits tend to exhibit higher degree of similarity for shallower internal nodes in the trees than for leaf nodes because random sample of data should still often present similar high-level structure that the decision tree uses for its shallow splits. Deeper splits have a higher chance to be overfit to data, meaning that they are not frequently reflected in unison across multiple trees in the ensemble. This is consistent with the basic purpose of the ensemble models of reducing variance of predictions made by multiple potentially overfit component models.

While we test all possible discrete values of ϵ, we note that it is not interesting to verify the lowest value of $\epsilon = \frac{1}{\#trees}$ because this would be equivalent to searching for a HR with a bound that can be crossed and change the vote of only a single decision tree. In other words, if a model makes use of an attribute considered in the definition of similarity, it will not be fair for minimal epsilon except in contrived cases where any change of vote by one tree results in the opposite change in another tree. Bias, in the form of model errors or violations of design specifications, are an ever present phenomenon in statistical machine learning. What we wish to avoid are cases where bias assumes meaningful structure in the way the model behaves. What makes larger values of ϵ more interesting (and less safe) is that it represents a boundary where many decision trees change their votes in response to the same small perturbation of input, which is less likely to be the result of random noise.

4.4.2 Revealing the structure of unfairness with counterexamples to (δ, ϵ)-GIF

Next, we examine all the ways in which a model violates (δ, ϵ)-GIF, which involves enumerating all counterexamples to a select definition of (δ, ϵ)-GIF. In more complex models, more opportunities for violation exist. A model can be evaluated as unfair because of implausible yet feasible counterexamples. In this case, we still have certifiable guarantees that the model exhibits individual fairness outside the range of the counterexample. By enumerating the ways in which model violates the (δ, ϵ)-GIF specification, we may still make an informed decision as to whether it is safe to deploy despite failing to be certified globally fair. This informs users of possible biases and how they could manifest after deployment. When all instances of unfair behavior are known, it becomes possible to mitigate the impact of the model's bias.

Counterexamples to (δ, ϵ)-GIF represent bounded regions of input space. Our

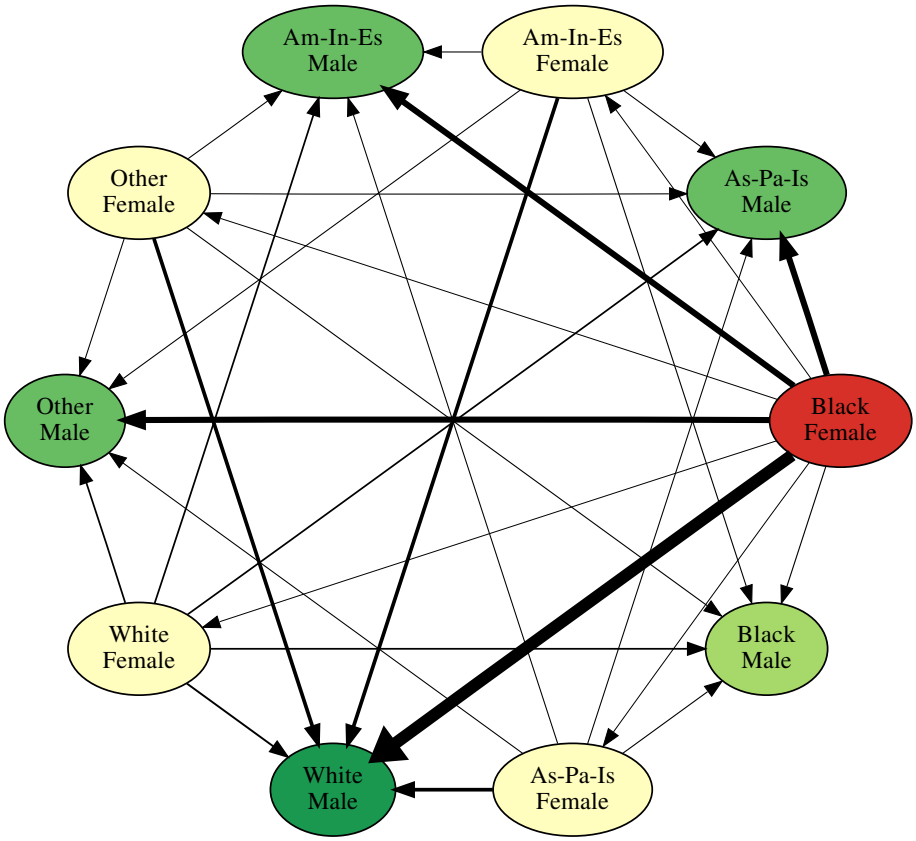
definition of δ, which measures similarity across select protected attributes, allows us to assign a group label to all the infinite points that fall under a single counterexample. This lets us transition from insights between individuals to insights governing fair behavior between groups of individuals sharing the same protected attribute values as an identified counterexample. If a model is equally unfair between all protected attribute groups, then no one group is disadvantaged or privileged over all others; this can be viewed simply as unstructured noise. On the other hand, if the model violates (δ, ϵ)-GIF in a way that disproportionately affects a single group defined by their protected attribute values, then we see that noise has structure which disadvantages or privileges this group over others.

We exhaustively enumerate and characterize counterexamples to (δ, ϵ)-GIF. This is an instance of the #SAT problem, where the goal is to find the number of ways that a particular formula can be satisfied. We take the satisfying HR from the assignment and sample a point from it in order to extract the protected attribute values that are implicated in the particular counterexample. A outline of the procedure is given in Algorithm 10

Algorithm 10: Enumerating specification violating counterexamples

$\mathcal{M} \leftarrow$ trained tree ensemble
$\mathcal{S} \leftarrow$ design specification(s)
CNF $\leftarrow$ TEA$(\mathcal{M}, \mathcal{S})$`/* the assertion stack */`
$\mathbb{X} \leftarrow$ `list()/* a list of counterexamples */`
$\phi \leftarrow$ `verify`(CNF) `/* a satisfying assignment */`
while $\phi \neq \emptyset$ **do**
    $\mathbb{X}$.`append`(ϕ)
    CNF.`push`$(\neg\phi)$`/* literals for active leaves are sufficient */`
    $\phi \leftarrow$ `verify`(CNF)
end
return $\mathbb{X}$

Figure 4.4 shows both the raw counts of all possible counterexamples to individual fairness (4.4b) as well as the structure of unfairness (4.4a). The directed graph in Figure 4.4a shows the disadvantaged/privileged relationship between groups of select protected attribute values in the direction of relatively disadvantaged $\rightarrow$ privileged characteristics. For all other attributes other than those in the Figure 4.4, ([`Race`,`Sex`]), a pair of points implicated in a counterexample have identical attribute values. In context of the binary classification task present in the ADULT data set, the tail end of the arrow denotes the existence of an individual that receives a particular



(a) Structure of (δ, ϵ)-GIF Unfairness for $\delta = \{\texttt{Race}, \texttt{Sex}\}, \epsilon = 0.6$

Disadvantaged Values ($\leq \$50,000$)	Privileged Attribute Values ($> \$50,000$)									
	BM	BF	WM	WF	OM	OF	AIEM	AIEF	APIM	APIF
Black Male (BM)	0	37	0	106	0	49	0	49	0	49
Black Female (BF)	829	0	8582	62	4312	60	4312	60	4312	60
White Male (WM)	0	49	0	25	0	43	0	43	0	43
White Female (WF)	1337	0	1386	0	1265	0	1265	0	1265	0
Other Male (OM)	0	31	0	35	0	25	0	25	0	25
Other Female (OF)	827	0	2648	0	779	0	779	0	779	0
AIE Male (AIEM)	0	31	0	35	0	25	0	25	0	25
AIE Female (AIEF)	827	0	2648	0	779	0	779	0	779	0
API Male (APIM)	0	31	0	35	0	25	0	25	0	25
API Female (APIF)	827	0	2648	0	779	0	779	0	779	0

(b) Counterexample counts between disadvantaged/privileged protected attribute values.

Figure 4.4: Table shows all counterexamples to (δ, ϵ)-GIF. Directed graph visualizes the table. Net privileged group at arrow head and disadvantaged group at arrow tail.

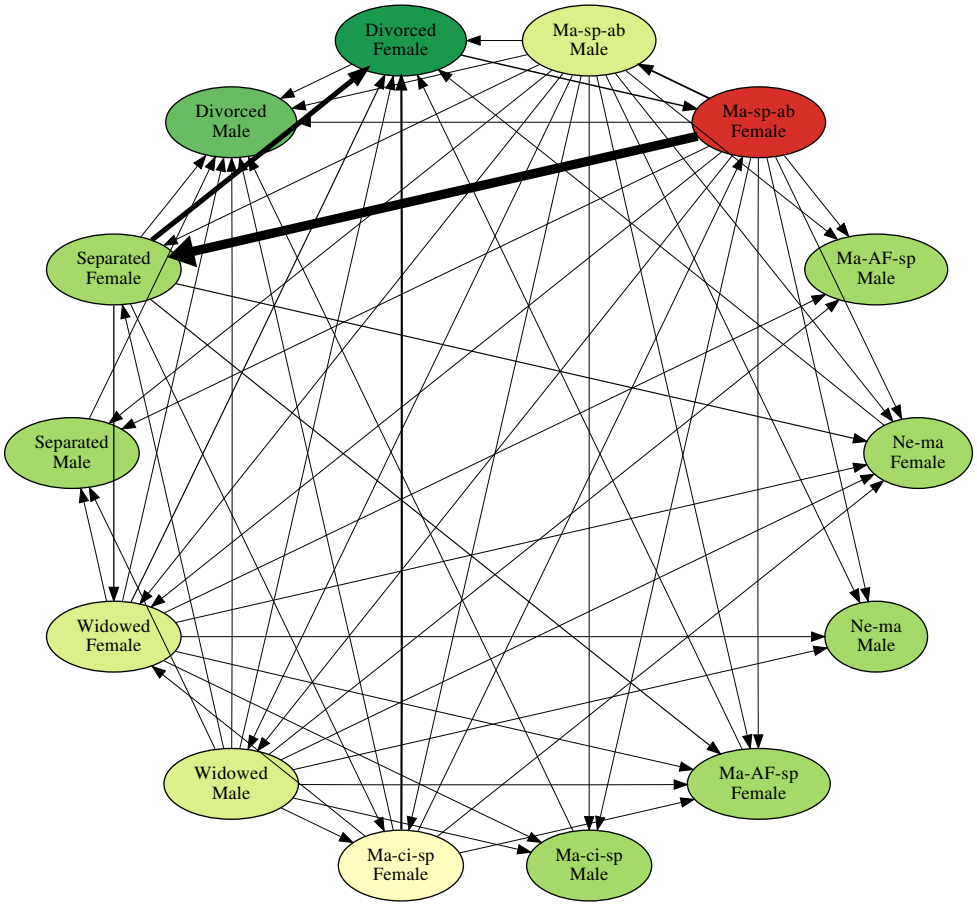
vote tally, $\mathbf{v}$, from the decision trees of the model and the head of the arrow denotes the existence of a similar input that the model yields $\mathbf{v}'$ that contains $N\epsilon$ more votes for the $>$ \$50K class, where N is the number of trees in the ensemble.

Width of the arrow denotes the net number of counterexamples between two subgroups in the rows and columns of Table 4.4b. Color of the nodes denotes the net sum of privileged/disadvantaged counterexamples for a subgroup. A red to green color scale that denotes unfairly disadvantaged to unfairly privileged. All counterexamples to the fairness specification under consideration are represented in 4.4a. There do not exist any other ways for the model to exhibit (δ, ϵ)-GIF when considering similarity between two individuals defined by *all attribute values equal, except race or sex*.

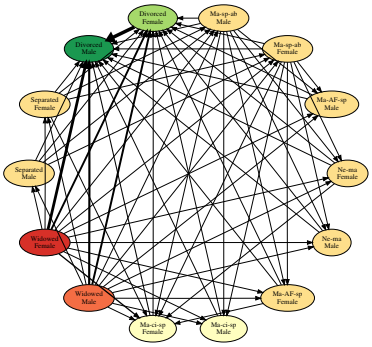
Figure 4.4 is a visualization of the structure of model unfairness (Figure 4.4a) along with a table of raw counts of counterexamples that exist between otherwise equivalent data points that differ only in select attributes under δ consideration (Figure 4.4). One interesting observation in Figure 4.4a is that (δ, ϵ)-GIF counterexamples are often bilateral. For instance, [`Black`,`Male`] are privileged compared to [`White`,`Female`] in 1337 unique ways, while [`White`,`Female`] are conversely privileged 106 times. This case reveals bias of a single order of magnitude (1337 vs. 106), but biases of larger magnitude may also exist; e.g., [`White`,Male] vs. [`White`,Female] are privileged with respect to one another 1386 and 25 times respectively. [`Black`,`Female`] is the only subpopulation under consideration that is unilateraly disadvantaged, and this can be seen when comparing it to all [`Female`] subpopulations.

Visually, this directed graph tells the story of a model that unfairly discriminates in two ways. Across the board, [`Male`] subpopulations are privileged compared to any of the [`Female`] subpopulations. A secondary bias can be seen in that [`Black`,`Male`] are less privileged than males of other races and [`Black`,`Female`] are less privileged than females of other races. While there are an infinite number of points that could generate these discovered instances of unfair behavior, we have a formal guarantee that any instance of model unfairness must fall into one of these identified counterexample faults. The purpose of identifying potential fault modes during the V&V process is so that the possibility of a particular fault leading to a failure, and the degree of harm done as consequence of the failure, can be analyzed before deploying the model and discovering these faults in the real world. In some cases, it is possible to develop mitigation strategies to ensure that faults of critical properties lead to graceful failures, however, such further inquiry is left to future work.

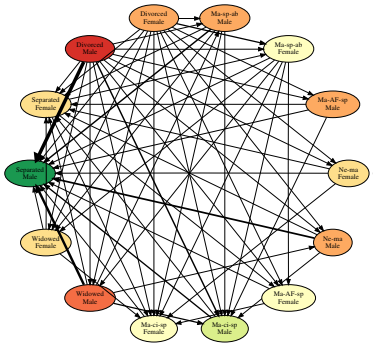
This graph also raises an important question: what would the structure of (δ, ϵ)-GIF counterexamples look like in a fair model? One valid form would be a graph with no edges, signaling that no counterexamples to (δ, ϵ)-GIF are ever found. Or, at the very least, a graph with very low-magnitude edges that correspond to rare or



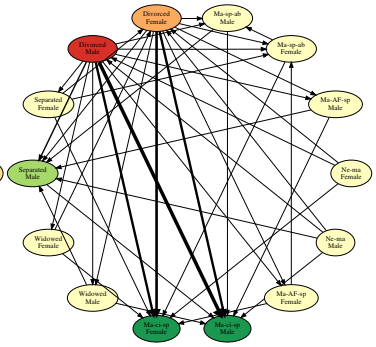
(a) $\epsilon = 0.6$



(b) $\epsilon = 0.7$



(c) $\epsilon = 0.8$



(d) $\epsilon = 0.9$

Figure 4.5: Structure of (δ, ϵ)-GIF Unfairness for $\delta = \{\texttt{MaritalStatus}, \texttt{Sex}\}$. Definition of ϵ is defined in each subfigure. Different groups are disadvantaged or privileged for different values of ϵ. `Ma-ci-sp`='married-civ-spouse', `Ne-ma`='never-married', `Ma-sp-ab`='married-spouse-absent', `Ma-AF-sp`='married-AF-spouse'.

infeasible cases in practice. Perhaps it is unreasonable to expect that a model never produces a risk assessment that is unfair between two individuals, because this would be equivalent to forcing the model to exhibit individually fair behavior globally, which is a challenging constraint to meet without developing new technologies for promoting individually fair model behavior during the training process. Another desirable form for this graph could be edges of completely equal weights, indicating that while the model may make unfair risk assessments, and furthermore may do so frequently, we can at least rest assured that these fault modes that lead to failures in fairness manifest in equal proportion between all subpopulations of data.

This phenomenon takes on visual form in Figure 4.5, where we show all counterexamples to (δ, ϵ)-GIF for a select δ consideration and varying ϵ values. Violations to fairness specifications are cumulative in the sense that if a counterexample is found for $\epsilon = .9$ then that same counterexample is also valid for any $\epsilon < .9$. Figures 4.5a, 4.5b, 4.5c, and 4.5d stratify these counterexamples in order to highlight the different structure that reveals itself for different definitions of ϵ. Figure 4.5a is larger than the rest to show attribute value information in the nodes for δ=[`MaritalStatus`,`Sex`]. The node position is preserved in Figures 4.5b, 4.5c, and 4.5d, but these figures are made smaller to fit on one page. We observe the colors of the nodes and lines connecting them changing as the definition of ϵ changes. In Figure 4.5a, with $\epsilon = 0.6$, we see that [`Married-spouse-absent`,`Female`] is most often disadvantaged whereas [`Divorced`,`Female`] is the most privileged. In Figure 4.5c, with $\epsilon = 0.8$, we instead see that [`Divorced`,`Male`] is the most disadvantaged and [`Separated`,`Male`] is the most privileged.

We hypothesize that this observed behavior is a result of the way that the ensemble learns its unfair behavior. Speaking in relative terms, there are more ways for the model to disadvantage [`Married-spouse-absent`,`Female`] in a small way, than there are ways for the model to disadvantage [`Divorced`,`Male`] in a large way. Both of these types of bias are uncovered by searching for counterexamples to (δ, ϵ)-GIF, but practically, they represent slightly different forms of bias. Further inquiry into the conclusions we can draw from this knowledge would be interesting, but it is outside of scope of this thesis work.

4.4.3 Flagging (δ, ϵ)-GIF unfair behavior after deployment

Knowledge of the existence of all counterexamples to (δ, ϵ)-GIF allows us to provide a certificate stating whether the model satisfies the individual fairness specification for any query to the model. The cost of performing this analysis is all upfront, so we demonstrate how it is possible to accredit a model and flag predictions that violate (δ, ϵ)-GIF. Model accreditation refers to the process of explicitly outlining the condi-

tions under which the model satisfies and violated (δ, ϵ)-GIF. A disjunctive union of all counterexample HRs defines the input region where the model is not individually fair, and the complement of that subspace defines the input region where the model satisfies individual fairness. Providing these certificates in real time would present new capabilities for decision support contexts. The known fault and a visualization of the counterexample would allow a user to be the final arbiter in determining whether the fault is indeed a fairness failure, or if it is a false alarm.

We measure the difference in cumulative time between making $\tilde{4}$0K normal model predictions and fault-informed predictions, which only require an additional lookup in a table storing all known counterexamples to (δ, ϵ)-GIF. The lookup table is indexed by the active leaves in the tree ensemble.

Model ID	Baseline Prediction	Fault-Safe Prediction	#Predictions Flagged
0	37.1±0.9	284±24.6	19
1	33.0±2.0	275±28.9	13
2	37.5±4.0	281±28.5	303
3	33.3±2.5	304±23.0	30
4	37.1±2.7	278±21.3	243
5	34.4±2.1	293±21.9	1
6	32.6±0.9	281±35.2	1
7	33.1±1.4	293±24.8	55
8	35.1±3.8	294±28.6	27
9	32.3±0.7	301±32.0	27

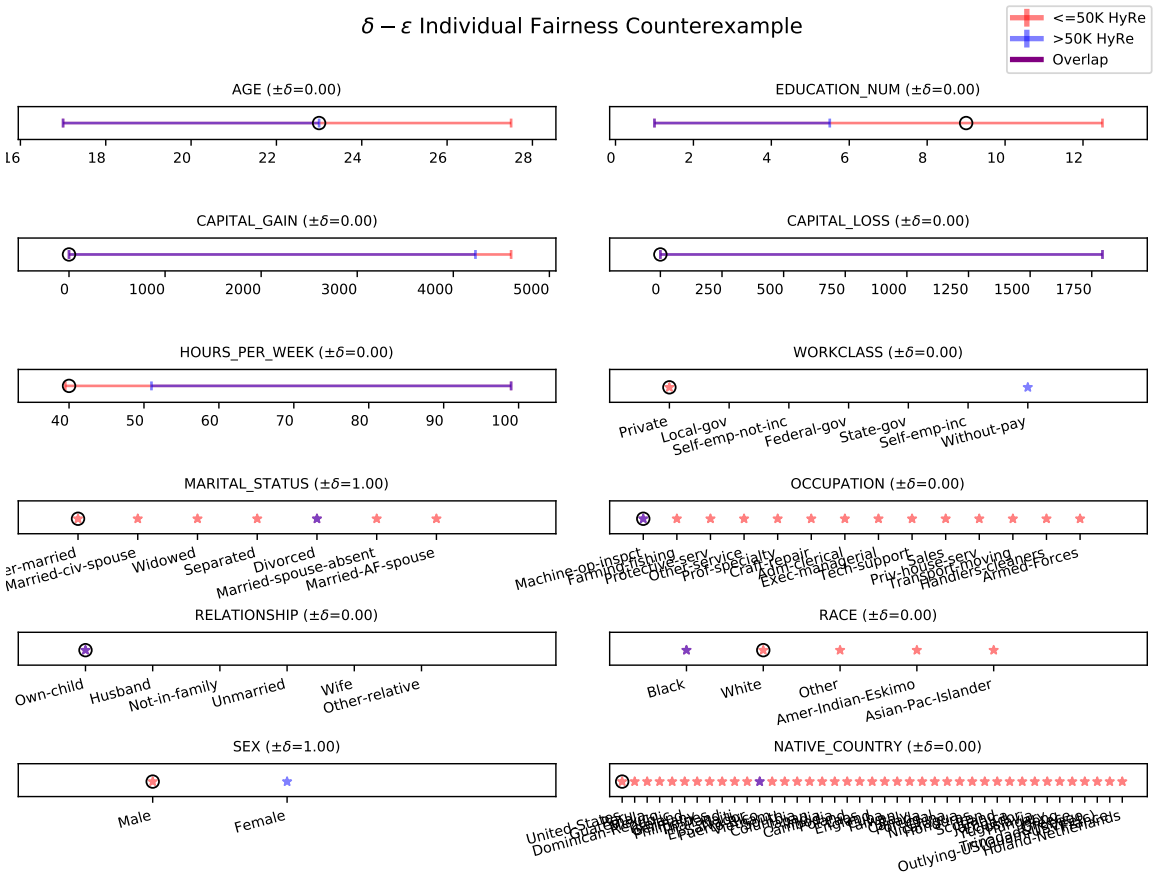
Table 4.2: Cumulative fault safe prediction times for all 40,700 samples, reported in mean ± standard deviation over 100 runs in milliseconds. Experiments conducted on Intel Xeon Gold 6238 CPUs.

Table 4.2 shows the time taken to make predictions without checking the fairness certificates vs. with the fairness certificates, as well as the number of test data samples that the model flags as implicated in an instance of individual unfairness. Since the cost of identifying all counterexamples to fairness is part of the verification process that happens before model deployment, and due to the fact that we enumerate all counterexamples to the individual fairness property, one additional lookup in a table keyed by active model states can provide meaningful information that would otherwise be unavailable.

Figure 4.6a shows a snapshot of unfairness warnings raised at run time when making predictions. Given that the potentially high-dimensional nature of HRs makes

sample ID	fault ID	Individual Fairness Fault
37461	1	[Without-pay] $\rightarrow$ +4 [**Private**] for >\$50,000

(a) Example of fault identification. A data sample from test data is flagged as a beneficiary of unfair privilege along the `Workclass` attribute, denoted by the bold text.



(b) A visual representation of a known (δ, ϵ)-GIF fault, including the test data sample (black circle) and pair of HRs that make up the counterexample.

Figure 4.6: An example of possible interface for flagging model predictions in test data that are affected by known unfair model behavior discovered in a (δ, ϵ)-GIF counterexample. 4.6a shows the test sample ID, fault ID, and interpretation of the contract. 4.6b visualizes the test sample amid the backdrop of the relevant counterexample uncovered during V&V, prior to deployment.

them difficult to visualize via standard techniques, we present a series of one dimensional axes along which we plot HR boundary values as well as query attribute values. We conjecture that a visualization makes the value in machine language certificates more intelligible to humans. Figure 4.6b is a visual representation of the fault flagged in Figure 4.6a. These HR can be bounded or unbounded in each dimension; for attributes where both the lower and upper bound of the HR is unbounded, the attribute is not shown in the figure. This is due to the fact that any value along that attribute would produce the same counterexample and therefore there is no need to show the violation range. The attribute values of the flagged query are marked with black circles across each attribute. In Figure 4.6, the warning told us that this sample was at risk of being disadvantaged when compared to a similar point, and indeed, the query falls in the red HR, meaning that there exist samples that are similar but violate (δ, ϵ)-GIF.

Figure 4.6b shows an example of what a fault-aware prediction can look like from a (δ, ϵ)-GIF formally verified model. Since our analysis performs a comprehensive analysis for HR instances of unfairness, we simply need to check whether an incoming sample from test data falls within the operational conditions (input attribute values) for which there exists a known exception to (δ, ϵ)-GIF. Times for baseline and fault-aware predictions are reported in Table 4.2. While the difference in compute time is an order of magnitude, the process still can be completed in under a second, which represents negligible computing cost in applications with a human in the loop. Providing flags of potentially unfair behavior can be done in real time after the comprehensive verification task is performed before model deployment. In this context, unfair behavior is defined as the model producing sufficiently different prediction probabilities for two inputs that are deemed similar according to the formal definition of (δ, ϵ)-GIF.

Currently, human studies aimed at showing whether such flagging of potentially unfair behavior increases a user's trust in the model is left to future work. Regardless, we argue that this capability increases the *trustworthiness* of the trained model in the sense that additional information is available to a human user that can help the user make their own decision on whether they trust the ensemble to make fair predictions. However, the utility of this framework seems most applicable to decision contexts where a human arbiter is ultimately responsible for making fair decisions.

4.4.4 A method for ensuring plausibility of (δ, ϵ)-GIF counterexamples

One exciting consequence of our formalism is that we are able to verify individual fairness even in regions of the input space with little to no empirical data. Estimating fairness on out of sample data is an active research thrust in the fairness community [102]. The challenge remains that once we disconnect ourselves from empirical data, plausibility becomes a concern. Proofs of individual fairness are no less correct for input subspaces that are implausible; however, analyzing such regions for fairness may dilute the certificates of fairness in much more feasible areas of the input space. We are interested in incorporating expert knowledge to define plausibility in a particular domain. Examples of possible domain-centric rules that could additionally restrict the global search space to more plausible regions include 1) if `MaritalStatus`=Married, then `Age`> minimum marriage age, 2) if `Relationship`=Husband, then `MaritalStatus`≠Wife, 3) if `WorkClass`=never worked, then `Occupation`≠Exec-managerial. It would be difficult for a human to enumerate all possible rules that govern plausibility, but we are looking into this and other ways to improve on defining our plausibility criteria, balancing the need to anticipate modes of failure to previously unseen data with the importance of considering the plausibility of different unseen instances.

Plausibility may also be defined in a model-centric manner, without leveraging domain knowledge. We apply our formalism for sets of empirical data, detailed in Algorithm 15 of Chapter 2, which involves constraining the verification task search scope to within a tunable neighborhood of all data seem during training. Figure 4.7 shows how the new ζ variable affects the scope of the (δ, ϵ)-GAR verification task. For $(\delta, \epsilon, \zeta)$-GAR, the red, sliding δ box is constrained to be overlapping with at least two of the blue, static ζ boxes. This greatly reduces the scope of the verification task, by focusing on regions of highest interest, which are those that are nearby training data support. The meaning of *global* in this case changes; the search is global in the sense that the entire training data set is under consideration. At the center of each blue box is a data point. The class label of the point is not relevant to the task, as we wish to search within the neighborhood of all existing data.

For small ζ, we will only be checking whether the GAR property is satisfied in regions very close to data support. For large ζ, we check regions that are further away from data support. ζ is a proxy for generalization; maximizing the ζ of $(\delta, \epsilon, \zeta)$-GAR tells us the extent to which the GAR property generalizes beyond training data support. δ is defined the same way as it was for LAR. The main result of changing from small to large δ is that it increases the allowable gap between two existing data points when searching for counterexamples to GAR. Often, there will be a collection

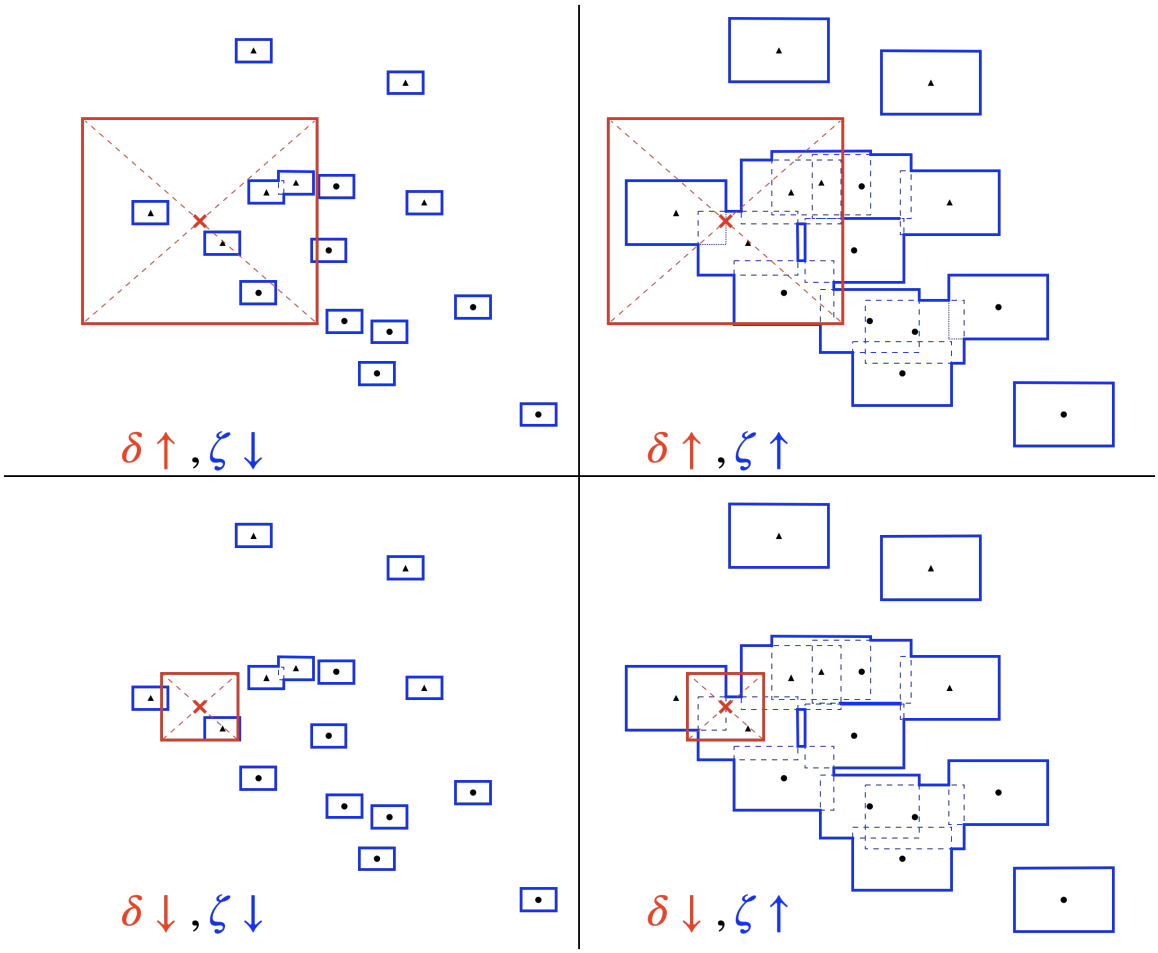


Figure 4.7: Illustrative example of how varying δ and ζ affect the global search area.

of parameters $(\delta = 0, \epsilon = 1, \zeta = 0)$ for which GAR is satisfied. The only case when this will not be true is if there are two identical data points in the training partition that receive different labels.

Verifying $(\delta, \epsilon, \zeta)$-GAR on Synthetic Data

We generate one-thousand 2D data points using sklearn's `make_blobs()` function and train a tree ensemble with 10 trees of max depth 5. Most data, minus outliers, falls in the x,y ranges of $[-3, +3]$. Figure 4.8 shows results from a 20x20x10 grid search for the most conservative definition of $(\delta, \epsilon, \zeta)$-GAR that is satisfiable. Scalar multipliers for δ and ζ are tested on the x and y axes, and the color of the contour indicates the minimal value of ϵ such that $(\delta, \epsilon, \zeta)$-GAR is satisfied. The legend can be interpreted as `D10` denotes that no counterexample exists to $(\delta, \epsilon, \zeta)$-GAR for 10 out of 10 trees in the ensemble simultaneously change their predictions $(\epsilon = 1.0)$. Failing

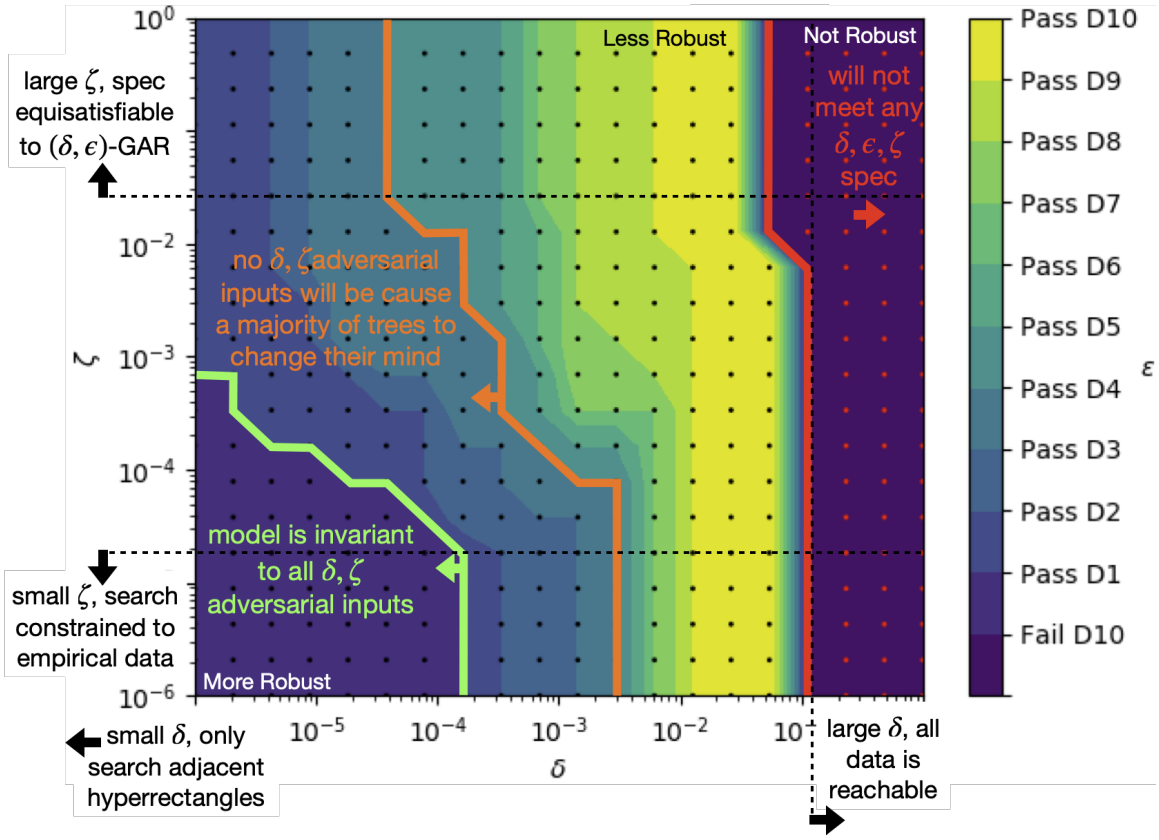


Figure 4.8: Interpretation of $(\delta, \epsilon, \zeta)$-GAR Contour

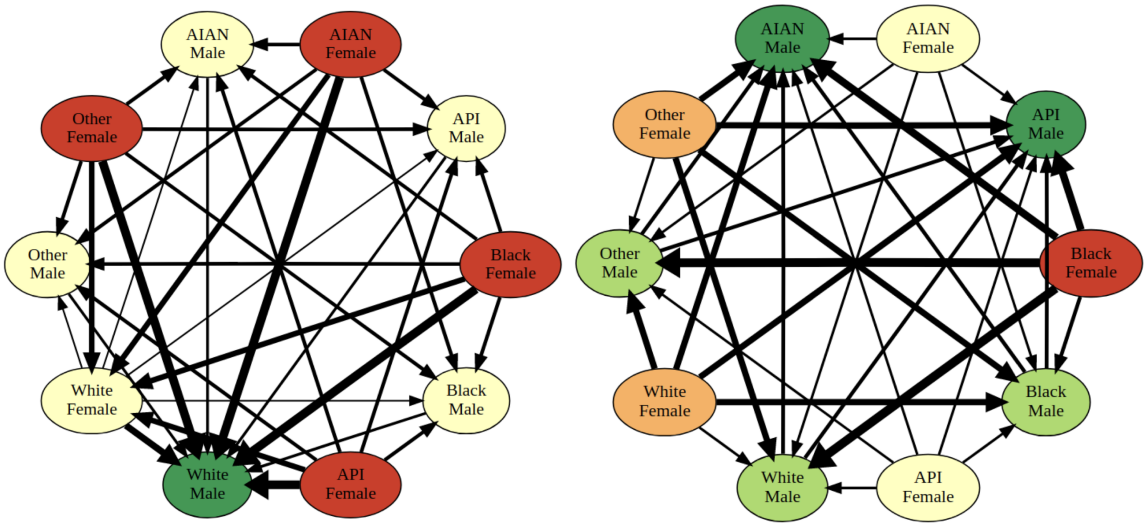
`D10` denotes that a counterexample exists where all trees simultaneously change their vote. Detailed annotations of interesting features are shown on the contour in Figure 4.8.

For sufficiently large δ, GAR will never be satisfiable because any two points in the data will be separated by less than δ. For sufficiently small δ, GAR will always be satisfiable because we would at most only be searching adjacent hyperrectangles. For sufficiently small ζ, we restrict ourselves to finding counterexamples to GAR in empirical data. For sufficiently large ζ, the $(\delta, \epsilon, \zeta)$-GAR specification becomes equisatisfiable with (δ, ϵ)-GAR. Three interesting levels in the contour are *Pass D1* (area to the bottom left of the green line) where no pair of points exist that are δ close to one another where any one tree changes its vote between the two. *Pass D5* (area to the left of the orange line) signifies that there will never be a pair of points within δ of each other where 50% of the trees to simultaneously change their prediction. *Fail D10* (area to the right of the red line) signifies that pairs of points exist where all trees in the ensemble change their vote between two similar points.

Difference between counterexamples to (δ, ϵ)-GIF and $(\delta, \epsilon, \zeta)$-GIF

We run a small scale experiment to show that it is possible restrict the scope of the GIF specification in an effort to make the counterexamples look more like empirical data. Figure 4.9a represents a model with all counterexamples for (δ, ϵ)-GIF while Figure 4.9b shows all counterexamples for $(\delta, \epsilon, \zeta)$-GIF.

We observe that by restricting the scope of the search space, there is more balance among the types of counterexamples discovered in the model. This suggests that there is one type of unfair behavior that is manifesting frequently in regions far away from any data support, and for this particular model. It remains to be seen if the counterexamples that are restricted to the neighborhood of empirical data are actually of higher utility to developers and users of AI systems.



(a) Counterexamples to (δ, ϵ)-GIF (b) Counterexamples to $(\delta, \epsilon, \zeta)$-GIF

Figure 4.9: Comparison of counterexamples to (δ, ϵ)-GIF and to $(\delta, \epsilon, \zeta)$-GIF

Chapter 5

Verification of a Safety-Paramount Engineering Constraint (SPEC)

Chapters 3 and 4 have focused on model-centric constraints. We turn our attention to domain-centric constraints and show how model adherence can be verified with TEA. The type of constraints we consider are definitions of service failures. The goal is to verify the absence of any fault modes which would lead to the specified service failure. We give a shorthand name to these domain-centric constraints, Safety-Paramount Engineering Constraint (SPEC). SPECs may require expert knowledge to define, as the definition of safe operation is highly variable between application domains.

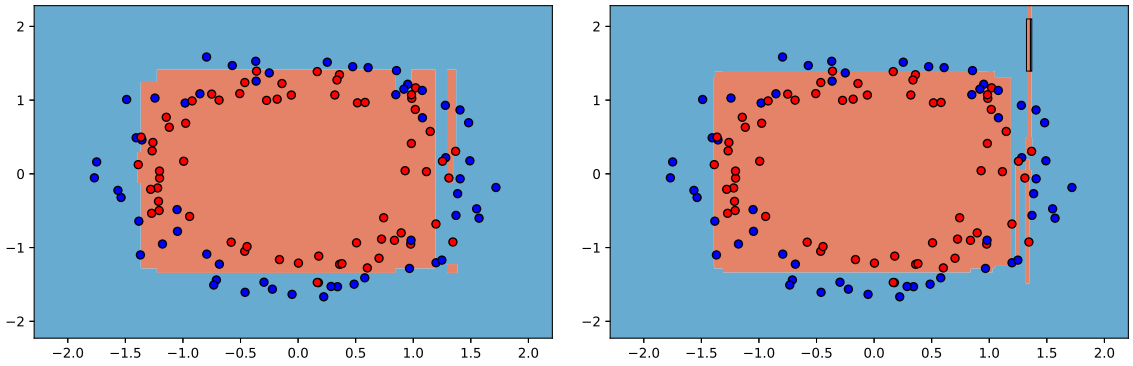
Formalizing expert knowledge is difficult because the structure of the specification may change depending on the nature of the verification task. Furthermore, expert knowledge needs to be amenable to encoding within TEA. We formalize expert knowledge by defining HR input-output specifications on tree ensemble models. A system satisfies SPEC if a target fault mode never manifests over all inputs within the scope of the search. Verifying SPEC tells domain experts and users of AI systems that no fault modes exist which could lead to the specified definition of failure.

We define a hypothetical SPEC on synthetic data to demonstrate the ability of TEA to identify fault modes that would lead to failures. We describe our formalism for SPECs and how they integrate into TEA. We compare TEA to a baseline in an Airborne Collision Avoidance System (ACAS) context show that tree ensembles are quicker to verify and safer in our experiments than a neural network. We apply our techniques to existing problems in a clinical decision-support system context and discuss the impact.

Expert knowledge is of particular relevance in clinical contexts, where AI systems are designed to support clinical decision-making. Clinicians have a hard time trusting a model when it errs in seemingly counter-intuitive ways. Verifying that the model

adheres to all SPECs defined by a domain expert provides proof that the model will never fail in the way the expert has defined. We show that simple rules that define a failure mode for a decision-support system can be verified. We tune the predictive threshold in a tree ensemble to find the point at which we maximize True Positive Rate (TPR) while still adhering to SPEC. This eliminates the need to incorporate downstream elements to check the output of the AI system, because mitigating the risk of select failures is accomplished during validation of the model. Our approach enables us to approach machine learning systems with an engineer's mindset. Verifying that these clinical decision-support systems avoid explicit failure modes that could cause easily preventable harm to come to patients supplements the type of statistical evidence that is usually assessed to determine the safety of the system.

5.1 Illustrative Example



(a) M1 - SPEC Condition Verified

(b) M2 - SPEC Condition Violated

Figure 5.1: SPEC certificates for two different models. Definition of specification is that the model must yield blue class label outside of the ranges $[-2, +2]$.

For a synthetic example, we define a binary classification task between red (negative - release without inspection) and blue (positive - tag for human inspection). One possible SPEC for this instance could be, *for sensor measurements outside the trusted range,* $x, y = [-2, +2]$*, the model will always yield an inspect advisory.* This problem setup would be applicable in application domains where the cost of false negatives is high enough that they warrant human adjudication. Subfigures 5.1a and 5.1b show two models trained on the same synthetic data generated with the sklearn `make_circles`() function. M1 adheres to the safety specification, whereas M2 violates the spec. The counterexample is denoted with the black box in subfigure 5.1b, which shows that there does exist a way for M2 to predict red with one sensor's measurements outside $y = [-2, +2]$.

Interestingly, the non-contiguous nature of the red region for M1's decision boundary may appear to be an unsafe model state, however, the existence of this island of red does not create a logical contradiction with the original safety specification. If we were interested in enforcing contiguous red and blue regions, we would need to define a different safety specification to verify.

If we were to provide evidence that the trained model adheres to SPEC without formal methods, we would need to sample data from the space of possible inputs and hope that we generate a data point that falls into the black box in Subfigure 5.1b. As the number of attributes grows with most application-scale data sets, the chance of generating a sample that falls inside a single SPEC-violating HR depends on the size of the HR. Furthermore, for models of sufficient size and complexity, the

HRs can be very small, making it hard to generate data that tests every possible HR without knowledge of the HRs. Statistical sampling techniques alone cannot generate a comprehensive set of data for testing all possible inputs. Statistical techniques can only provide sufficiently rigorous probabilistic estimates that the model adheres to SPEC rather than the formal guarantees that TEA provides.

5.2 Encoding Strategy for (ϕ, ζ)-SPECs

Algorithm 11: Safety-Paramount Engineering Constraints (SPECs)

$\phi_{in}, \phi_{out} \leftarrow$ An input-output safety constraint, ϕ
$\mathbb{H} \leftarrow$ HR range over inputs in ϕ_{in}
$y_F \leftarrow$ Specified output in ϕ_{out} that indicates failure if it manifests for $\mathbf{x} \in \mathbb{H}$
$\zeta \leftarrow$ Size of HRs about each training data point for plausibility
`/* restrict scope of the search to` $\mathbb{H}$ `*/`
for *att* $\leftarrow 1$ **to** *number of attributes* **do**
  if $\mathbb{H}_{att(lb)}$ *is not None* **then**
    define $H_{att,lb} = \mathbb{H}_{att(lb)}$
    assert $\neg H_{att,lb}$`/* greater than lower HyRe bound */`
  end
  if $\mathbb{H}_{att(ub)}$ *is not None* **then**
    define $H_{att,ub} = \mathbb{H}_{att(ub)}$
    assert $H_{att,ub}$`/* lesser than upper HyRe bound */`
  end
end
`/* restrict global search to` ζ`-neighborhood of data */`
assert `Plausibility`(ζ)`/* See Alg. 6 */`
assert `Ordinality`$()$ among H and threshold literals `/* See Alg. 3 */`
`/* assert that SPEC failure manifests */`
assert y_F

Our formalism for SPECs involves defining a HR over inputs and mapping this region to a single ensemble prediction. The HR does not need to have lower and upper bounds along all attributes as was necessary in our formalism for encoding data. Unbounded attributes mean that all possible values are tested from $[-\infty, +\infty]$, and single bounded attributes mean that all values on one side of the bound are tested. For all attributes with lower and upper bounds, all values between those limits are tested.

Algorithm 11 details the strategy for defining new literal that encode the limits of the search space over which the SPEC must be satisfied. Adding these literals to the existing list of literals where ordinality is constrained integrates them into the rest of the CNF logic for the model and any other specifications. Truth is assigned to each of these new literals depending on what side of the threshold value we wish to search. A conjunction of multiple thresholds can define an arbitrary polytope.

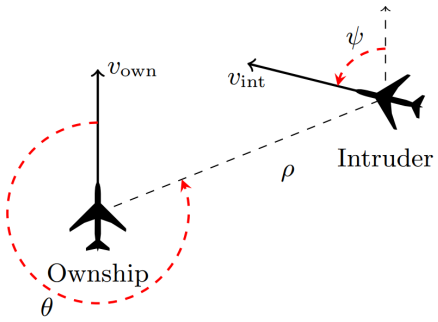
For non-contiguous input spaces, we may break up the verification tasks into two independent queries. Our formalism imposes a structural constraint on SPECs. For instance, it is not possible to verify abstract concepts such as *conservation of mass*. Future work on incorporating more expressive logics may aid in formalizing other types of SPECs which would be of use in particular contexts.

5.2.1 How to specify a SPEC

TEA allows for easy specification of domain expert knowledge as long as it takes the form of hyperrectangular input-output mappings. In the following sections, lists of specifications (ϕ_j) will be presented with a natural language interpretation of the SPEC we wish to verify. These statements can be converted to propositional logic by defining a HR over input space that is bounded with a lower and upper bound along each attribute in data. For attributes implicated in the specification, TEA is constrained to search a one or two sided contiguous region (i.e. $(-\infty < x_i < c_{i,ub}]$, $[c_{i,lb} < x_i < +\infty)$, or $[c_{i,lb} < x_i < c_{i,ub}]$). For attributes that are not implicated in a particular specification, TEA is told to search a range from $(-\infty, +\infty)$, which effectively means that no constraint is added to the select attribute. The TEA framework can convert these bounds on individual attributes to requisite propositional logic and CNF required to add SPECs to the assertion stack alongside the model. These steps are shown by operations on $\mathbb{H}$ in select lines of Algorithm 11.

5.3 Baseline Comparison

We apply TEA to an Airborne Collision Avoidance System (ACAS) context. The learning task at hand is best described as a task to compress the size of a lookup table. It represents a case in which our formalism is used to verify properties of a single trained decision tree rather than an ensemble. This experiment is defined by [109], which allows us to compare against a state-of-the-art formalism for verifying properties of trained neural networks, Reluplex [109]. Reluplex verifies multiple safety specifications for the Airborne Collision Avoidance System for unmanned aircraft (ACAS Xu). We verify the same safety specifications as SPECs by repeating the experiments of [109] with our formalism, TEA.



ρ	distance from ownship to intruder
θ	angle to intruder relative to ownship heading direction
ψ	heading angle of intruder relative to ownship heading direction
v_{own}	speed of ownship
v_{int}	speed of intruder
τ	time until loss of vertical separation
a_{prev}	previous advisory

Figure 5.2: Overview of the ACAS Xu system as described by [109]

5.3.1 SPECs for ACAS Xu

ACAS Xu originally takes the form of a lookup table generated by solving a Markov decision process [111]. It takes seven inputs as described in Figure 5.2. The outputs are advisory scores for COC (clear of conflict), weak right, strong right, weak left, and strong left. The lookup table, at approximately 2GB, is large enough to motivate compression. One experiment in [109] involves compressing the lookup table by fitting a neural network [100] and using the trained model in place of the lookup table. The

Reluplex experiments use 45 networks created by discretizing τ and a_{prev} and training on the remaining 5 inputs. They have 8 layers and 300 ReLUs each.

We repeat these experiments with our own framework by fitting a decision tree to the lookup table, which serve as an effective means of policy compression [140], and use our SAT framework to perform the verification tasks featured by [109]. The ACAS Xu system is not publicly available, so we use code provided by [101] to generate a policy table as similar to the ACAS Xu system as possible. To ensure fair comparison with the results of our framework, we use code posted by [109] to re-run their experiments with our policy table, data, and computational resources. A full list of the properties verified by [109] is shared in Table 5.1. Properties 1 and 2 are omitted from our analysis, because they are properties of the real-valued output scores for neural networks, which are not applicable to voting tree models.

ϕ_3	If the intruder is directly ahead and is moving towards the ownship, the score for COC will not be minimal.
ϕ_4	If the intruder is directly ahead and is moving away from the ownship but at a lower speed than that of the ownship, the score for COC will not be minimal.
ϕ_5	If the intruder is near and approaching from the left, the network advises "strong right".
ϕ_6	If the intruder is sufficiently far away, the network advises COC.
ϕ_7	If vertical separation is large, the network will never advise a strong turn.
ϕ_8	For large vertical separation and a previous "weak left" advisory, the network will either output COC or continue advising "weak left".
ϕ_9	Even if the previous advisory was "weak right", the presence of a nearby intruder will cause the network to output a "strong left" advisory instead.
ϕ_{10}	For a far away intruder, the network advises COC.

Table 5.1: Definitions of safety specifications for ACAS Xu [109]

Table 5.2 shows the result and the running time for the verification of each property, both for neural networks verified with Reluplex and trees verified with our SAT framework. `UNSAT` indicates that the safety specification holds globally and `SAT` in-

	Verifying DNNs with Reluplex		Verifying Trees with TEA		
property	result	time (s)	result	time (s)	speed gain
3	UNSAT	7,214	UNSAT	8.61	837x
4	SAT	21,145	UNSAT	81.38	259x
5	SAT	2,809	SAT	9.02	311x
6	TIMEOUT	$>$ 43,200	SAT	6.63	$>$ 6,515x
7	TIMEOUT	$>$ 43,200	UNSAT	283.41	$>$ 152x
8	SAT	285	SAT	3.85	74x
9	SAT	66	SAT	12.41	5x
10	SAT	18,442	SAT	7.31	2,522x

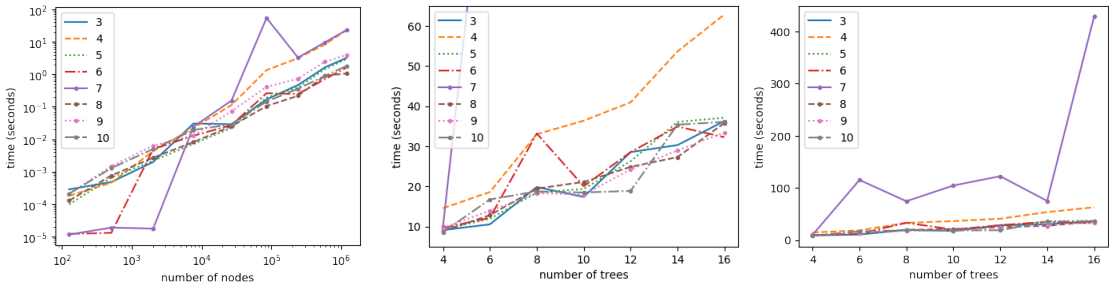
Table 5.2: The result and time to verify ACAS Xu properties defined in [109].

dicates that a counterexample to the safety specification is found. Where multiple networks are involved, the overall result for the set is given.

Excluding property 4 and timeouts, the tree and the network obtain the same certifications for the other safety specifications. We expect the divergent certificates for property 4 to be a result of the goodness of fit for the neural network and the decision tree. On training data taken from the policy table, the neural networks achieve 99.67% accuracy, while the tree achieves 100%, due to the fact that the tree may grow until each leaf node contains homogeneous class labels. Were we to run these experiments on the data set used by [109], we would expect to obtain identical certificates to those that they report in a fraction of the time, based on the evidence from our experiment.

Across all properties, the time to verify the tree model is dramatically shorter. Reluplex did not halt within the 12 hour (43,200 second) timeout limit for properties 6 and 7. SAT verification of trees ranges 5 to 2,500 times faster than Reluplex verification of neural networks. This result stems from a combination of the relative simplicity of verifying a decision tree compared to a neural network as well as expected speed increase when moving to SAT. This is thus an example of a task where trees should be considered as an alternative to neural networks due to their relative simplicity and speed of verification. These are common desiderata that we show may be achieved without sacrificing adherence to any additional safety criteria.

We expected that a SAT based method would run faster than an SMT based method. Therefore, we were interested in seeing how our SAT based approach scales as a function of the depth of the trees in the random forest as well as the number of



(a) Varying tree depth. (b) Varying #trees, w/o ϕ_7. (c) Varying #trees, w/ ϕ_7.

Figure 5.3: Time to verify ACAS Xu properties

trees in the ensemble. First we investigate the role that tree size plays in verification time by training individual decision trees of various maximum depths on the simulated ACAS Xu system. Figure 5.3a shows that, with one notable exception, the time to verify scales linearly with the number of nodes. This confirms expectations, as no vote counting nor plurality logic is needed to determine the output of a single decision tree.

To investigate the role the ensemble size plays in verification time, we train ensembles with variable number of trees of max depth 16. Allowing the depth to increase beyond this point did not yield sufficiently different levels of performance. Figures 5.3b and 5.3c show that, in most cases, there is also an approximately linear trend with the number of trees; however, ϕ_7 exhibits an unusual trend apart from the others. Upon several repetitions of the same experiments, we found the unusual trends of ϕ_7 to be consistent, so it is not an anomalous behavior. We conjecture that either there are more possible counterexamples to the formula that the solver needs to check before concluding whether ϕ_7 is or is not satisfiable, or, perhaps the solver is learning inefficient conflict clauses which do not facilitate as quick an arrival at a solution when compared to the other ϕ properties. Either way, this suggests that, while the system may be predictable and scalable for most queries, some may scale unusually. This fits expectations that while SAT verification is NP-Complete in worst-case, solutions are often found very quickly due to exploitable structure commonly embedded in non-random SAT instances [5].

5.4 Utility of SPECs in a Clinical Context

We show how SPECs provide useful information in a critical care medicine context. Critical care is one of the most stressful environments for healthcare providers. Their patients are seriously ill and their status is typically fragile; they can deteriorate quickly without much warning. There is also ample and diverse technology in regular use, including various sensors producing large streams of vital sign data, and monitors that sound alerts at the bedside whenever a patient's health drifts away from stability. It is very hard for the clinicians to interpret very large amounts of information generated by these systems, while always making correct diagnoses and timely implementing beneficial therapeutic decisions, yet never missing any important clues. Otherwise, their patients would be exposed to grave risks.

The critical care environment exacerbates the requirement for any AI based system to be trusted enough for the clinicians to accept it, otherwise it would not be used at all. There is little room for error or inefficiencies in the medical alert domain. It is a high stress environment, where multiple alarms are usually sounding simultaneously. The decisions being made often mean life or death and they must be made quickly. Output from distrusted AI systems will only cause disruptions. This barrier has been one of the key challenges effectively preventing a wider spread of the intelligent, analytic technology in this domain of healthcare.

We show how TEA produces tree ensemble models that are tuned to avoid what a clinician defines as easily preventable harm to patients. Inviting clinicians to participate in the V&V process for AI systems represents another opportunity to incorporate expert knowledge into the AI design process. For those who are understandably skeptical of the usefulness of AI systems, demonstrating that the trained models do not possess fault modes that lead to catastrophic failure offers new insight into models that provides evidence that AI adheres to some notion of clinical *common sense*.

For tasks in the clinical domain, tree ensemble models are more desirable than other model classes. The justification for their use in this domain stems from the fact that the individual decision trees are naturally amenable to interrogation from caregivers and clinicians. This previously has allowed medical experts to independently verify the recommendations from the models so that they know when to trust their own instincts or to trust the model. Such self-interrogation can be more difficult, if not impossible, with other model classes. TEA seeks to formally verify that the model adheres to critical design specifications globally, which offers expanded capabilities beyond the case-by-case, human supervision of AI outputs that represents current practice. TEA will allow clinicians to reason about their model in contractual terms that apply for a range of infinite possible inputs to the model.

One of the concerns of deploying AI in the critical care context is that these

systems often behave in counterintuitive ways. Many times, this means discovering unexpected patterns in data that drive intelligent decision making. However, counterintuitive behavior also describes an AI system failing under seemingly nominal circumstances. A common fear is that the AI system may cause harm to a patient that would have been easily preventable if a human was responsible for decision-making. It is hard to trust a system that produces good outputs for inputs that confuse clinicians while also making embarrassing and egregious errors for inputs that clinicians have high levels of confidence and consensus as to what is the right course of action. Expert knowledge is of particular relevance in medical contexts, where AI systems are designed to support clinical decision-making, and it is important to ensure that deployed systems avoid the same mistakes that the clinicians avoid. We extend TEA to verify whether a trained tree ensemble adheres to simple rules that describe safe model behavior. This serves as a sort of safety net that tells us that, whether a model is correct or incorrect, it will never make a particular type of error.

5.4.1 Optimizing for safety and accuracy

One of the major operational challenges in critical care is resource allocation. With a limited staffing, how do we decide where to allocate resources such that the most serious medical instabilities are prioritized for attention of adequate personnel? For one classification task, the decision is whether a particular instance of a bed-side alert represents a relatively mild health crisis that can be addressed by a nurse on duty, or the care level should be escalated to involve a physician [32]. Ideally, minor instabilities that can be routinely managed would not require a doctor's attention, but a delay in calling for help of a physician or, in truly life threatening scenarios a Medical Emergency Team (MET), when it is clinically warranted, may lead to grave consequences. Conversely, escalating care when it is not required wastes time and precious resources.

The data for our experiments comprises time-series of vitals including heart rate, respiratory rate, blood oxygenation (SpO_2), and various metrics of blood pressure (arterial, venous, collected invasively or noninvasively), electrocardiogram signals, as well as hemodynamic parameters of clinical importance derived from those basic vitals. We use human data collected at selected Step-Down Units (SDU) and Intensive Care Units (ICU) at the University of Pittsburgh Medical Center. These human data have been de-identified and approved for research use by the cognizant Institutional Review Boards. We will use these data in an already featurized form. Data preparation process has been accomplished in previous projects, including [27, 33, 92, 91, 153, 152, 194, 195, 201].

The strategy for obtaining data to train care escalation models involves asking

medical experts to provide labels. Often, these experts agree, and the label that they provide is the one ultimately used. However, based on our practice, around 30% of the time, the experts disagree in their assessment. While existing labeling protocols (such as [190]) require that the experts engage in a debate until a possible consensus is reached, aiming to reduce subjectivity of the resulting labels, we are interested in determining whether the disagreement is simply due to imperfect inter-rater reliability in clinical assessments of health status or whether the disagreements represent systematic differences in perspectives. If systematic differences in clinical perspectives manifest, this may mean that models trained on data labeled by each clinician may satisfy different SPECs, yielding a diverse set of ensemble models with different provable safety guarantees.

An initial idea is to fit a separate tree ensemble model to data provided by each annotator, and then to use TEA to verify a set of SPECs to formalize systematic differences of the resulting models in terms of the safety conditions that they satisfy. It is important to note that such formal assessment only says how the models differ and does not offer insight into the opinions of the particular clinicians. Nonetheless, comparing trained models side by side and enumerating desirable properties of each still represents new information that was previously unavailable to clinicians prior to deciding to deploy a trained model.

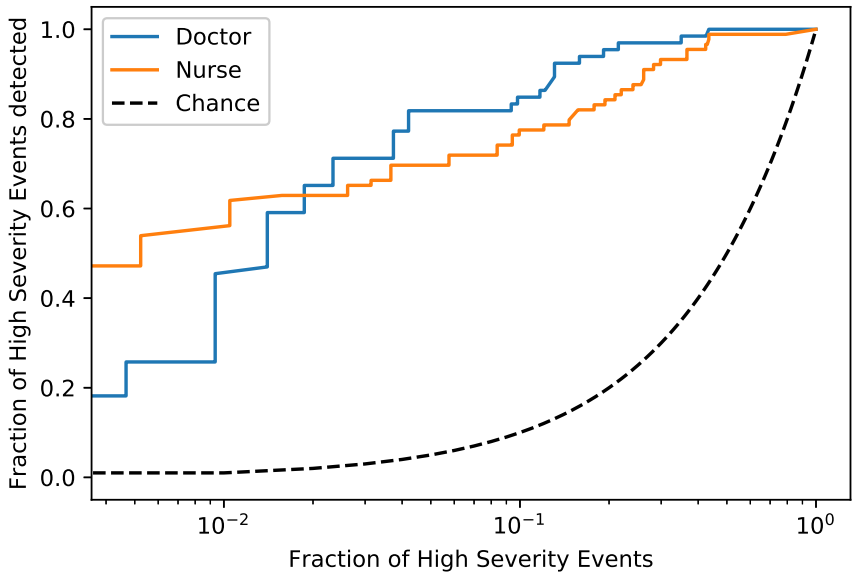


Figure 5.4: ROC curve for two tree ensembles trained on identical data with different labels coming from a nurse and a doctor.

Figure 5.4 shows an ROC curve for two tree ensemble models, one fit to labels provided by a doctor and the other with labels provided by a nurse. Labels range on a scale of 1 to 4 with 1 representing a mild hemodynamic insufficiency and 4 representing a severe instability. We binarize the classification task by splitting into two classes corresponding to low severity (1-2) and high severity (3-4). In this design, we expect the nurses to call for doctors' help when facing high severity events, and proceed to handling low severity events by themselves. The two tree ensembles consist of 10 trees each of max depth 10 and each leaf has no fewer than 10 samples. The models are trained on data that are otherwise identical in terms of the number of samples, attributes, and attribute values, but that have different labels. In the figure, we can see that the doctor and nurse models have similar detection rates across a range of possible FPR targets. But, the nurse's model appears more conservative than the doctor's model at making the calls to escalate care. It yields much higher recall rates at very low sensitivity thresholds, but is less likely to escalate than the doctor's model when looking at less obvious cases. This is consistent with clinical intuition which observes that the nurses naturally try to add most value by handling as many cases as possible within their capacity, however they are quick to escalate care of the obviously extreme events.

ϕ_1	If Blood Oxygen (spO2) $\leq 75\%$, then yield High Severity Event advisory.
ϕ_2	If Heart Rate (hr) ≤ 20, then yield High Severity Event advisory.
ϕ_3	If Heart Rate (hr) ≤ 130, then yield High Severity Event advisory.
ϕ_4	If Respiratory Rate (rr) > 30, then yield High Severity Event advisory.
ϕ_5	If Blood Oxygen (spO2) $\leq 85\%$ and Respiratory Rate (rr) > 30, then yield High Severity Event advisory.
ϕ_6	If Diastolic Blood Pressure (diaBP) > 100 and Systolic Blood Pressure (sysBP) > 150, then yield High Severity Event advisory.

Table 5.3: Examples of SPEC definitions for a critical care medicine context.

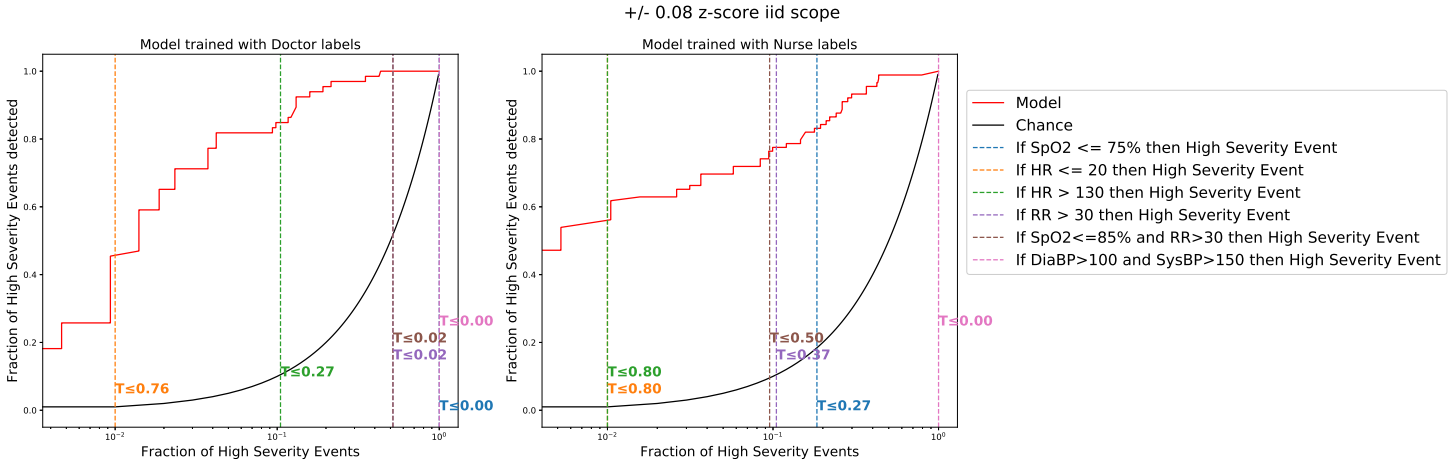
Table 5.3 shows a few (ϕ, ζ)-SPECs that represent example rules that define possible safety specifications. In this context, the utility of (ϕ, ζ)-SPECs is most apparent by verifying that certain health states always result in alerts. In this manner, we prove that definitions of physiological attributes that require escalation of care are never missed by the trained models. The examples in Table 5.3 are structurally

simple rules, but as ϕ_5 and ϕ_6 demonstrate, it is possible to verify a conjunction of rules. (ϕ, ζ)-SPECs can be of arbitrary complexity, but as the complexity of the specification increases, the intelligibility of the specification decreases. Verifying that the model satisfies an unintelligible specification may still be useful in some cases to demonstrate the trustworthiness of the model. However, in general, it is these types of simple (ϕ, ζ)-SPECs that are most useful to clinicians because model error is expected in complex and confusing cases, but violating such simple rules would greatly diminish confidence that the model is ready for deployment.

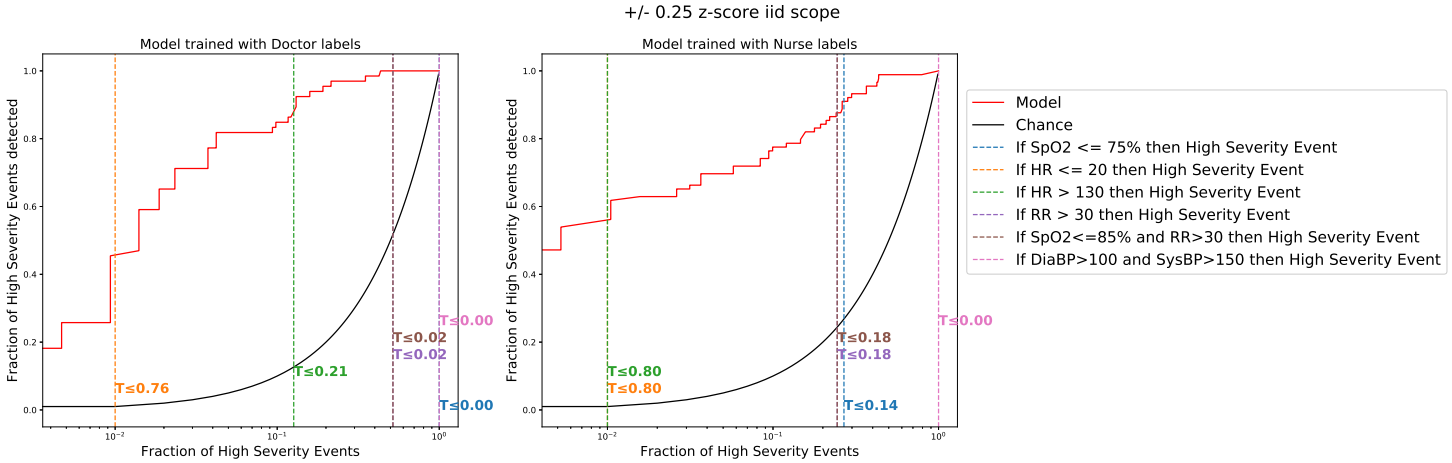
For the experiments, we focus the verification task to the immediate neighborhood of data that the model saw during training. Our formalism for incorporating a disjunctive, hyperrectangular search area that circumscribes all training data points with a $\pm\zeta$-HR was described in Section 2.2. Varying the size of the neighborhood circumscribing each data point changes the extent of input space over which the SPECs must hold. For small ζ-neighborhoods, TEA shows whether a (ϕ, ζ)-SPEC was met during training. For large ζ-neighborhoods, TEA shows whether a (ϕ, ζ)-SPEC can generalize to yet unseen data that looks sufficiently similar to training data.

The votes cast at leaves in trees of the ensemble are affected by the prediction threshold selected from the ROC curve shown in Figure 5.4. Verifying SPECs for any one decision threshold will tell us which SPECs are satisfied. We test 50 possible threshold values evenly spaced within the range of FPR$= 1 \times 10^{-2}$ to FPR$= 1 \times 10^{0}$. As we sweep from small to large FPR targets, identifying the change point in satisfiability denotes the most restrictive (largest) prediction threshold value for which no counterexamples to a (ϕ, ζ)-SPEC exist. If the threshold were to become ever slightly more restrictive, at least one leaf in the ensemble will change its vote from High Severity Event to Low Severity Event, and this change yields a feasible counterexample to the SPEC in question. Any less restrictive (smaller) prediction threshold value will also satisfy (ϕ, ζ)-SPEC because the number of leaves producing High Severity Event votes monotonically increases as more leaves in the ensemble are allowed to cast High Severity Event votes due to the less restrictive threshold requirement for an alarm. There will always be a threshold value where the model satisfies a SPEC because it is possible to tune the threshold to 0, such that all leaves in the ensemble produce High Severity Event votes. This represents a case where the AI system is, effectively, no longer in use, as it only produces one output no matter the inputs to the system.

Figure 5.5 shows the result of verifying multiple (ϕ, ζ)-SPECs over two select scopes (Subfigure 5.5a: $\zeta = \pm 0.08\sigma$, Subfigure 5.5b: $\zeta = \pm 0.25\sigma$). We denote T as the definition of a model's prediction threshold value. Left subfigures correspond to the model trained by doctor labels and right subfigures correspond to the model



(a) $\pm\zeta = 0.08\sigma$ HR around all training data points.



(b) $\pm\zeta = 0.25\sigma$ HR around all training data points.

Figure 5.5: Most restrictive threshold values that satisfy (ϕ, ζ)-SPECs when scope restricted to within disjunctive, z-score neighborhood of training data (837 samples).

trained by nurse labels. Superimposed on the ROC curves are the most restrictive (largest) value of T that satisfy their respectively colored (ϕ, ζ)-SPEC in the legend. $T = 0$ signify that the only way the model will satisfy the (ϕ, ζ)-SPEC is if it never produces a Low Severity Event advisory. This is due to the learned structure of the model, which was not required to adhere to these (ϕ, ζ)-SPECs during the training phase of model development. The largest T value that satisfies each (ϕ, ζ)-SPEC in Table 5.3 we will refer to as T_{ϕ_i}.

We observe a few interesting results. The ensemble model trained by doctor labels satisfies fewer (ϕ, ζ)-SPECs than the model trained by nurse labels. One hypothesis could be that the nurse provided better labels that allowed a better model to be trained. Another possibility is that the doctor's decision making may take more of the gestalt of the state of the patient into account whereas the nurse makes decisions based on simpler yet more consistent metrics. There is one (ϕ, ζ)-SPEC, ϕ_6 that neither the nurse nor doctor model satisfy.

The biggest difference in (ϕ, ζ)-SPEC adherence between the two models is seen in ϕ_3, where the nurse model satisfies this specification for significantly more restrictive values of T_{ϕ_3} than that of the doctor model. If we consider this difference to be a structural difference in the trained models, this could be the result of Heart Rate being used at different depths of the tree between the two models. For the doctor model, we can say that there exist leaves in the tree ensemble where less than 50% of the label distribution is High Severity Events and it is possible to wind up in these leaves with a Heart Rate $>$ 130bpm. This could be due to a path through decision trees that never made a split on the Heart Rate feature. If Heart Rate was used at a decision node on the path to these leaves, then it would be necessary to change the nodes decision threshold value in order for the model to satisfy the (ϕ, ζ)-SPEC. If, instead, we were to consider this difference between the trained models to be a product of systematically different decision making on behalf of the nurse and doctor themselves, the difference in T_{ϕ_3} may suggest that the nurse considers Heart Rate $>$ 130bpm to be indicative that a High Severity Event is underway, while the doctor may not use the same threshold in their decision making. Perhaps the doctor's threshold for a High Severity Event would be better described by Heart Rate $>$ 150bpm. Comparatively larger values of T show that one model is able to be more discriminative and reduce FPs while still adhering to (ϕ, ζ)-SPEC.

In addition to the difference observed in T_{ϕ_3}, the nurse model exhibits greater T_{ϕ_i} values for all (ϕ, ζ)-SPECs tested for different values of ζ. This suggests that if the tested rules are of paramount importance, then the nurse model is better suited to satisfy SPECs and minimize FPR. If additional SPECs are identified by clinicians, it is possible to test new SPECs without affecting the results for previously tested

SPECs. Different or new safety considerations may vary in criticality at different times once the model is deployed, and TEA automates the process of testing each new specification.

As the ζ-neighborhood of (ϕ, ζ)-SPEC increases in scope around all training data, the prediction threshold values that satisfy SPEC stay the same or decrease. Searching a larger neighborhood means there is more opportunity for counterexamples to arise. This decreasing value in SPEC safe prediction thresholds adheres to our intuition. It would be possible to enumerate counterexamples if desired, but the point at which SPEC is satisfied completely is most interesting to us. The extent to which ζ can be increased is a measurement of the generalizability of model adherence to (ϕ, ζ)-SPEC. For small values of ζ, we are ostensibly checking whether the model adheres to (ϕ, ζ)-SPEC for training data samples. For large values of ζ, this shows that the model will adhere to (ϕ, ζ)-SPEC even for inputs that are far away from data the model saw during training. The values of T_{ϕ_i} for the doctors model seem to generalize better than those of the nurse model when the size of ζ increases.

A key takeaway from Figure 5.5 is that multiple (ϕ, ζ)-SPEC certificates can be subsumed by the ultimate selection of T', which denotes the human-selected prediction threshold for a trained tree ensemble model. The value of T' is typically determined by balancing out the costs associated with FP and FN, but TEA offers certificate information which may help inform the selection of T' such that it not only balances cost of errors but also allows the model to satisfy the safety criteria enforced with (ϕ, ζ)-SPECs. Any vertical lines, T_{ϕ_i} in Figure 5.5, that fall to the left of a select value for T' denote that the model will adhere to that (ϕ, ζ)-SPEC. As long as $T_{\phi_i} > T'$, certificates provided by TEA tell us that ϕ_i will be satisfied by the resulting model that will be yielding predictions using T' as its selected threshold. If the model satisfies (ϕ, ζ)-SPEC for a value of $T_{\phi_i} > T'$, then the model will also satisfy the specification for $T = T'$ because strictly equal number or more leaves are producing High Severity Event votes when the prediction threshold is relaxed (lowered). For example, if we were to set the predictive threshold for the nurse model to $T' = 0.50$, which happens to represent a majority vote prediction threshold, TEA tells us that the model adheres to ϕ_2, ϕ_3, and ϕ_5 for all possible inputs within $\pm\zeta = 0.08\sigma$ of any point the model saw during training. If we were to increase the scope to $\pm\zeta = 0.25\sigma$ but keep $T' = 0.50$, then we subsume fewer SPECs (ϕ_2, ϕ_3).

TEA facilitates integration of statistical and formal V&V in order to more rigorously test models before deployment. Reasoning about ROC curves with SPEC limits superimposed lets us select a pareto-optimal prediction threshold such that we minimize the number of FPs produced by the model while also adhering to necessary SPECs defined by the clinicians who will be using the model. From the AI devel-

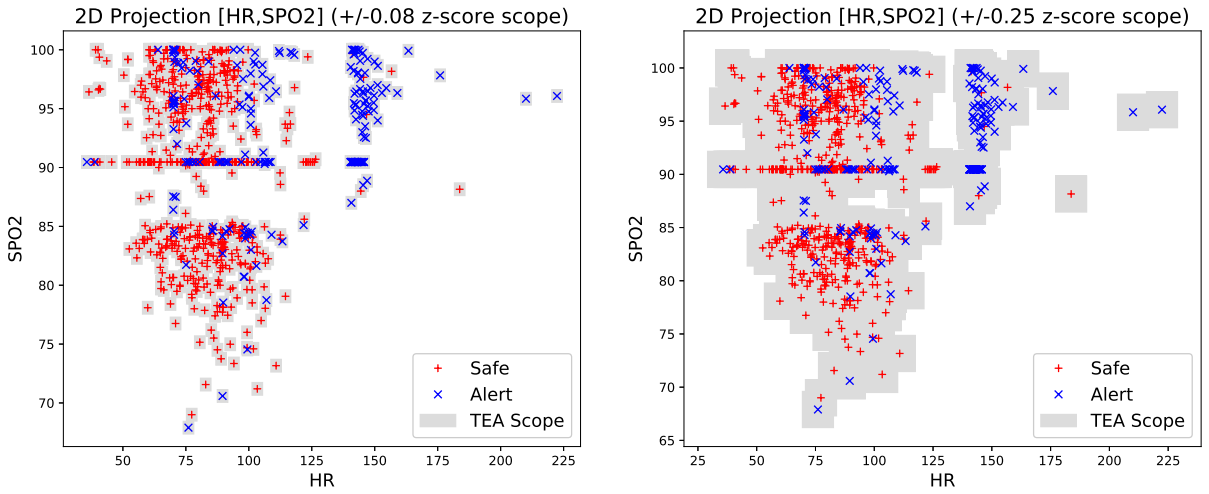
oper's perspective, this lets us meet two objectives which previously required separate analyses; we can train models with statistical methods in order to reduce the number of FPs while at the same time adhering to critical safety criteria. From a clinician's perspective, this offers the opportunity for domain experts to draw red lines that the AI system may not cross. If TEA proves that the model adheres to all SPECs, then the model becomes deployment worthy, provided that the ensemble actually reduces the number of FPs that are currently produced by bedside alert monitors. It is also conceivable for clinicians to accept relaxed criteria and assume a calculated risk of a model violating a particular SPEC in order to further reduce model error rates.

Figure 5.6 offers example two-dimensional projections of data with the $\pm\zeta$-neighborhood about training data visualized. The gray area in these figures shows the region over the input space which TEA certifies that the model adheres to SPEC. The generalizability of a (ϕ, ζ)-SPEC is defined by the largest value of ζ for which a model adheres to (ϕ, ζ)-SPEC. At times, the $\pm\zeta$-neighborhood produces a near contiguous region over which to verify SPECs (right subplot 5.5a) and in other cases it leaves gaps which are not checked for adherence to SPEC (right subplot 5.5b).

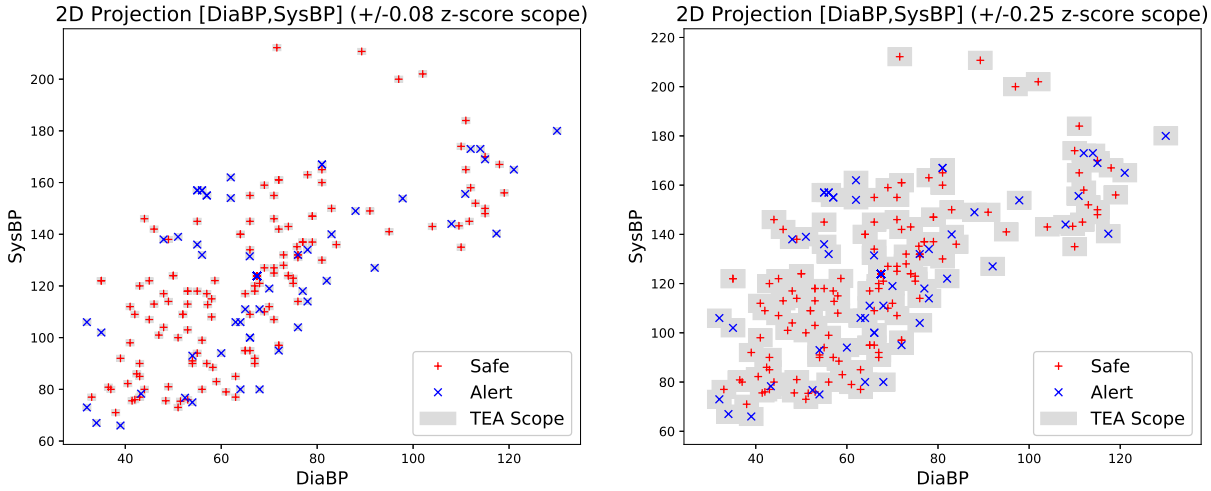
Defining ζ is an important step that is necessary for (ϕ, ζ)-SPEC. Without imposing a restriction on the scope of the verification task, both the nurse and the doctor model do not satisfy any of the SPECs listed in Table 5.3. While TEA restricts verification tasks to the set of feasible model states, there still remain a subset of model states that represent implausible inputs. Implausibility may be defined either by physical or biological impossibilities (such as diastolic pressure greater than systolic) or other gross, unrealizable abnormalities. Imposing the ζ constraint on SPECs is a straightforward way for TEA to limit its search to feasible and plausible states while not requiring input from the AI developers or clinical experts to explicitly define the search scope that is relevant. TEA could accommodate this type of human input to the verification task if needed.

Our strategy represents one possible way to scope a global verification task down to a set of points that are likely inputs to the model. This is pragmatic because tree ensemble behavior tends to break down for inputs that are very far from the next nearest HR. Our formalism for the region around data seen before is useful because it represents a logical proxy to the i.i.d. assumption of ML where we only expect the models to perform well when they are given data that is similar to data they were trained with. When moving to a logical formalism with TEA, we submit that the same relaxation of global properties is a reasonable assumption to make. The model should be certified safe in regions where the model had access to data support during training.

In cases where an input sample does not fall into the ζ-neighborhood of existing



(a) [`Heart Rate`,`SpO2`] Projection of $\pm\zeta$-HR around all training data points. Left: $\zeta = 0.08\sigma$, Right: $\zeta = 0.25\sigma$.



(b) [`DiastolicBP`, `SystolicBP`] Projection of $\pm\zeta$-HR around all training data points. Left: $\zeta = 0.08\sigma$, Right: $\zeta = 0.25\sigma$.

Figure 5.6: $\pm\zeta$-neighborhood for (ϕ, ζ)-SPECs shown in Figure 5.5.

training data, we have two options. We may flag the prediction as a sample with no safety guarantees because it falls outside the bounds that were tested by TEA. The other option involves adding this new data point to the list of existing empirical samples and testing whether ensemble behavior in the neighborhood of this new point also exhibits required safety properties. Since (ϕ, ζ)-SPECs are all independent, we can simply test model adherence to (ϕ, ζ)-SPEC in the ζ-neighborhood of the new sample. If safety is still guaranteed, this would allow TEA to update the certificate without needing to re-run the entire verification task again.

Figure 5.6a helps illustrate another utility of TEA, which is that if we were to enumerate counterexamples to (ϕ, ζ)-SPEC, we could alter the verification scope by adding or removing data points from consideration. Iterative removal of data points from consideration that act as counterexamples to (ϕ, ζ)-SPEC may reveal a tree ensemble model that is certifiably safe within the ζ-neighborhood of a very large percentage of the training data. This would represent a variant on the max-SAT problem, which seeks to find a truth assignment to literals in a formula such that the number of falsified clauses is minimized. Our strategy could identify samples that are anomalous; not in the sense that they necessarily represent statistical outliers in attribute values, but rather than they represent rare inputs where the trained model will violate a safety specification. These experiments are part of ongoing and future work, which we discuss in the next chapter.

Chapter 6

Conclusion

We have shown that TEA can check an artificial intelligence for adherence to decision-making desiderata that human decision-making tends to exhibit. These include showing that:

1. The model will not change its prediction under the presence of imperceptible perturbations on input (See LAR in Chapter 3)

2. The model will not prescribe different risk assessments to similar inputs (See GIF in Chapter 4)

3. The model will not make errors that humans would otherwise easily avoid (See SPEC in Chapter 5)

Our work provides a framework for verifying these properties, and we claim that this new capability increases the trustworthiness of any tree ensembles model that is checked, regardless of whether the model satisfies or violates given constraints. This is due to the fact that knowledge of the strengths and weaknesses of the model gives all stakeholders for a particular AI system more information that they may use to inform decision making during the model selection process.

6.1 TEA expands V&V practice for AI systems

We showed how TEA can provide answers to open questions across multiple real-world applications of AI systems. TEA also provides new ways to approach V&V of trained models. While the scope of this thesis focuses on tree ensemble models, many of the ways in which we use certificates in our experiments will map to other model classes and other formalisms as well. TEA reasons about model structure directly, making it possible to assess model adherence to critical design specifications both inside and

outside of data support. The ways in which we leverage certificate information are complementary to statistical tests that typically form the basis of V&V in the context of AI.

Radiation Safety

We formally verify adherence to $(\mathbf{x}, \delta)$-LAR on data collected from an active research collaboration for developing AI systems for safety contexts. We show how LAR certificates can be used in a variety of ways to formally test a trained model for different types of behavior that are expressible as robustness specifications.

Existing V&V efforts for AI systems in this context involve physicists and data scientists providing sufficiently strong probabilistic estimates of model behavior. TEA improves upon existing practice by making the jump to provable guarantees of model performance and automating the process of formal V&V. The critical nature of the domain makes the certificates produced by TEA particularly useful in the sense that the margin for uncertainty is virtually non-existent. Physicists and data scientists involved in the project can use TEA to better understand the vulnerabilities of trained models they intend to deploy. Formal testing of trained models informs decision making that may result in models that better satisfy the requirements in the domain. The novel contributions of our work can be summarized as follows:

1. TEA provides LAR certificates for models that exist at U.S. ports of entry. Formally verifying LAR for voting tree ensembles at the scale required for AI in critical application contexts has never before been reported in literature.

2. TEA shows which models are invariant to adversarial perturbations in `VEHICLE` characteristics for all available data, providing proof that select models adhere to this design specification set by physicists.

3. Finding the largest degree, c, of $(\mathbf{x}, c\delta)$-LAR that a model exhibits for a set δ allows us to estimate the level of robustness that the model will exhibit for yet untested points. The resolution of the analysis can be scoped as needed, making it possible characterize model robustness to adversarial perturbations among a subset of attributes.

4. We define a method for characterizing LAR in a way that is most relevant for data scientists and developers of ML models. Model robustness is desirable when the model is correct but undesirable when it is incorrect. TEA is able to verify the extend to which a model exhibits this desiderata, and TEA can tune the prediction threshold in order to maximize this behavior.

5. Experiments provide evidence that it is possible for multiple models to exhibit accuracy that is statistically indistinguishable, but the ways in which the models produce those predictions vary greatly. TEA can be used to break these ties by showing which model satisfies stricter definitions of LAR.

6. TEA enables formal V&V at the model selection phase of AI development. The efficiency of our SAT formalism speeds up verification time to the point where it is possible to verify properties of multiple models of realistic, application-scale in parallel.

Fairness

We formally verified that a model adheres to an Individual Fairness (IF) specification. While we are not the first to report verifying individual fairness, we are the first to do so for tree ensemble models, and we are the first to do so by testing the entire input space. Standard practice involves a data-centric assessment of model fairness, which means that regions of input without data support remain untested. TEA can address these limitations because it reasons about the model structure directly and provides certificates of model fairness both inside and outside of data support. As a research area, Fairness represents an exciting opportunity to apply TEA to important problems. While our current focus has bee on IF, there are many different types of fairness considerations, and TEA can be extended to incorporate other established fairness metrics to form a suite of verifiable fairness tests for tree ensemble models.

We demonstrate verification of individual fairness of voting tree ensembles trained on a publicly available census income dataset. Our experiments illustrate three beneficial uses of individual fairness verification that complement the capabilities of existing statistical methods. We show how to use individual fairness verification as a tool for model selection. The proposed approach can rank models based on quantitative model-centric metrics of fairness which can be used in practice as a criterion complementary to predictive accuracy. Secondly, our approach identifies individual fairness counterexamples to reveal the group-level structure of bias absorbed from data by a trained model. This capability supports interpretation of the identified patterns of unfair behavior of the models. Thirdly, the proposed approach is able to verify the operational conditions under which a trained model can be certified to meet individual fairness. This allows the users to envelope the regions of decision space where a desired condition is guaranteed, and it can inform decision makers on-the-fly about the disposition of the individual data instances being processed by the trained model, flagging cases that may be disadvantaged by the model.

The proposed methodology of provable model certification extends and enriches

the analysis of trained models beyond what is attainable with common statistical methods. The new capabilities can aid designers and users of intelligent decision support systems by allowing them to understand, manage, and mitigate the presence of unfair biases at both the design stage and after these systems are deployed for use in the field. We hope this work may help bridge the gap between the formal methods and fairness research communities, which have so much to offer one another. The novel contributions of our work can be summarized as follows:

1. While TEA is not the first to verify IF, it is the first to do so for tree ensembles, and the first to test fairness globally.

2. TEA is the first verification framework to consider predicted class probabilities as a measure of 'similar treatment', making this framework especially relevant to risk assessment contexts.

3. TEA is the first to enumerate counterexamples to any IF specification. This reveals the previously hidden structure of unfairness that a model learns during training.

4. TEA is the first method to incorporate IF certificate information into the model selection process.

5. The cost of the verification task is all upfront, so TEA is able to flag model predictions that violate the IF specification in real-time once the model is deployed.

Critical Care Medicine

TEA can incorporate clinician rules that describe safety or common sense in clinical settings. There are physiological limits to the capabilities of a human body, and TEA can verify that any AI system trained on physiological attributes respects those limits. TEA will allow domain experts to be involved in the V&V process for trained models, which allows AI to benefit from expert knowledge even further into the development of trained systems.

We show that TEA can verify model adherence to SPECs and that resulting certificates can show the predictive threshold value that represents a pareto-optimal balance between minimizing FPs and adhering to specifications. Furthermore, these certificates are additive in the sense that it is possible to display which SPECs are satisfied and which ones are violated for each possible predictive threshold value. This could allow the prediction threshold to be tuned depending on which SPEC is most

critical at different times. The novel contributions of our work can be summarized as follows:

1. TEA verifies model adherence to SPECs. SPECs formalize expert knowledge.
2. TEA verifies that the trained model adheres to all SPECs within a disjunctive neighborhood of all data points seen during training. Encoding each data point into TEA is a novel approach to defining a search space of interest. Our formalism serves as a proxy to the i.i.d. assumption in ML in that TEA only tests points that look reasonably similar to points seen before.
3. TEA is able to formalize expert knowledge. This means that clinicians can use their knowledge to guide V&V of a trained model.
4. TEA identifies an optimal balance of minimizing FPR while guaranteeing adherence to SPECs. This allows a specification-driven approach to tuning parameters of models.

6.2 Some Ethical Considerations

The potential for misuse of the technology

A key point to remember about this work is that we are not building models that otherwise would not have existed. Good and bad actors could use this work to verify that models exhibit characteristics they desire. Bad actors may be interested in building models that exhibit poor characteristics. These could range from a desire to build models which exhibit disproportional bias among subsets of data, to a desire to build models which are provably susceptible to adversarial attack. Furthermore, this type of back door approach to building learning models means that a bad actor could have explicit descriptions of inputs that cause the model to fail once it is in deployment.

While there is a potential risk for misuse, it also is worth pointing toward other fields where engineers also have this information. For instance, anyone involved in the design of an aircraft may know of its fault modes, or programmers who develop encryption technologies may possess knowledge of back doors. It may be worth considering what sorts of safeguards other fields have in place to prevent misuse of model checking or verification tools by potential bad actors. While taking steps to limit the potential for bad actors to use this technology, it is worth noting that it is the same technology that good actors could use to identify bad actors. As long as benevolent users stay vigilant and verify trained models before they are deployed, it will be difficult for nefarious influence to go undiscovered.

With a more broad interpretation of the capabilities this thesis may allow, it is theoretically possible to use such a verification framework to perform penetration testing for Machine Learning as a Service models (MLaaS). As these services become more prevalent, the incentive for discovering potential weaknesses or inner workings of pre-trained models that represent a company's intellectual property grows. In order for our framework to be used, all we need is access to the trained model itself; we do not need access to any empirical data, which may be the component of the AI system that is most well protected.

In the context of fairness there are specific ethical considerations. While we strongly believe that formal verification of model adherence to fairness specifications can make a positive impact in applications, we do not wish to overstate its utility. Defining what it means for a ML model to be fair represents a consequential design choice. A challenge with formal verification techniques is that the specification in question must be explicitly defined, therefore, we are only able to detect fairness failures that we already know to look for. This means that some useful definitions of fairness are not amenable to formal verification using SAT logic. Due to this, we also

wish to note that just because this particular definition of individual fairness is easily verified does not mean that it is the right definition of fairness for all applications.

What formal certificates gain over statistical auditing methods are provable guarantees. However, they are not a panacea to the inherent challenges facing our field. One incorrect way to interpret our contributions in Chapter 4 would be to suggest that a model that satisfies (δ, ϵ)-GIF is a 'fair' model. It is widely understood that fairness is not a monolithic concept; different considerations are to be weighted accordingly and specifically for a particular application. Our work, and formal verification methods more broadly, do not address the underlying socio-technical problem that pervades even the best attempts to build fair systems today. Our framework will allow us to say for certain whether a model exhibits IF globally, but it will tell us nothing about whether the model meets other, perhaps broader fairness metrics, which may be equally or even more relevant for a given task. It is our hope that in the future, we grow our repertoire of formally verifiable fairness metrics, such that TEA may verify fairness across many different metrics.

Another important distinction to make is that the certificates that TEA yields only hold over the precise environmental conditions that are specified in the certificate. For example, it may be tempting to extrapolate beyond a certificate of LAR and erroneously conclude that the model exhibits LAR for all inputs. Less clear examples also exist; a SPEC may only be verified for a particular neighborhood about existing empirical data, which means that slightly changing the size of that neighborhood, even by an imperceptible amount, may change the result of the SPEC certificate. Adding new empirical data may also require re-verification, as those new samples may be near previously untested input ranges that may alter the SPEC certificate. When implementing and using TEA, or any formal system for that matter, it is important to remember that changing the assignment to a single literal may result in a different verification outcome. As long as operational ranges over which the certificates hold is acknowledged by human stakeholders, the opportunity for over-trust in the proofs should be manageable.

The ethical imperative of our work

We submit that it is unethical to *not* further develop and deploy this technology. Verifying model adherence to design specifications forces people to confront fault modes that could previously be swept under the rug. With our verification framework, it becomes much harder to hide implicit bias within a trained model. Pointing to high levels of predictive performance will no longer be sufficient to dismiss concerns that a model violates, for example, a fairness criterion.

The fact that TEA enables contractual reasoning about trained models will be

helpful when trying to determine whether a model is ready for deployment. Instead of needing to guess about the inner workings of the model, TEA provides a sheet of specifications that the model adheres to and this can be used to inform decision making about whether or not the model is being used properly. We believe that this satisfies a broad desire both inside and outside of AI for transparency in algorithmic design. It is difficult to talk about the ethical considerations of deploying a tree ensemble model but it is more straightforward to talk about the ethical considerations of the design specifications that we consider critical for a particular context. This thesis work makes verifying model adherence to ethical imperatives possible, which can help move discourse from arguments on *where does algorithmic bias arise* to *what are the necessary and sufficient ethical considerations whose adherence make a model worthy of deployment in mission critical applications.*

6.3 Potential Utility

***Specification Agnostic*: proofs for any desiderata expressed in propositional logic**

We demonstrate that notions of robustness, fairness, and safety can be successfully encoded in logic and subsequently verified. This is by no means an exhaustive set of the capabilities of this framework; if a desiderata can be encoded into Boolean logic, then we can verify whether a model adheres to the desiderata, making our approach *specification agnostic*. We turn engineering desiderata into formal certificates of model compliance. While it takes communicating with domain experts and data scientists to develop new ideas for desiderata that are important to them, encoding these into logic once will increase the repertoire of ready-to-go verification tests available to all others who use TEA to test their own models. As new design specifications are formalized and encoded, which represents the hardest part of the verification task, the self-reporting capabilities of our models improve.

Experiments can be conducted for other formal systems that yield certificates

It would be possible to focus future work in the direction of verifying similar properties for other model classes that are expressible in a SAT formalism. When considering formalisms other than SAT, it is not immediately clear if the scale of the verification tasks we perform in this thesis would be tractable. However, many of our experiments highlight interesting ways in which certificates can be used, and any verification pipeline that yields certificates could theoretically be configured to

conduct similar experiments. Additionally, existing formal systems (e.g. VoTE and Reluplex) could immediately benefit by implementing the same tests for LAR as a function of the prediction outcome of the model. This may represent an easy way for the many successes in the Automated Reasoning and Formal Methods community to be adopted by practitioners in AI and ML.

Data Agnostic: no data samples are required when verifying global properties like (δ, ϵ)-GIF

Input points are needed for $(\mathbf{x}, \delta)$-LAR and (ϕ, ζ)-SPEC, so it would be incorrect to say that TEA is entirely data agnostic. The ensembles that we verify are trained with data. However, in some cases, it is possible for TEA to generate certificates of model behavior without encoding data alongside the model.

By reasoning directly about learned threshold values in a trained tree ensemble model, we eliminate the need for data samples to conduct our analysis. All that is required is meta-data about the input space, including the names of the attributes and their data types. Purely as a consequence of the structure of the policy learned by a tree ensemble, we can verify how the model will behave in all conceivable situations. This is desirable because our framework allows us to prove model adherence to properties over bounded hyperrectangular regions that contain an infinite number of points, making our verification strategy quite generalizable.

TEA automates V&V of tree ensembles

V&V can be costly, depending on the complexity of the system in question. By leveraging automated reasoning to perform the verification task, we turn person-hours into compute-hours. From the practitioners perspective, the existence of TEA means that there is a cheap and easy way to verify the deployed tree ensemble meets all necessary engineering specifications. By automating the process to utilize formal verification for tree ensembles, we expect TEA to increase the likelihood of developers of AI systems formally verifying their models, and the chance of domain experts adopting formally verified systems. Turning engineering specifications and desiderata into formal certificates of model compliance facilitates the design systems that interact with an AI components upstream in the pipeline.

Contracts may reduce overall complexity of data pipelines

We have a strong preference for developing parsimonious models; keeping things as simple as they need to be reduces the number of failure modes in a given system. If we can verify an AI systems adherence to specifications, then we may not need

additional control blocks downstream to perform the checks that the output from the model is deemed safe or trustworthy [20].

6.4 Future Work

We are excited to continue this work that we believe is very exciting and promising. Our hope is that formal V&V for AI systems continues to grow. We outline some limitations of our current work that would make good candidates for further study for improving TEA. We provide evidence from our preliminary experiments for some of these points in the Appendix.

Counterexample probability

For sufficiently large tree ensemble models, there will be HRs that are very small, even in high dimensional spaces. Currently, we do not impose constraints on the size of the HRs that are implicated in counterexamples because our ordinal constraints on attribute threshold values return a single change point, which returns a single HR, even if all adjacent HRs also serve as counterexamples. It is hard to generalize from these small HR counterexamples, and if there is a model state that violates a particular specification, we would rather discover the largest violating HR. Incorporating volume of the HR into TEA would either help us rank the relative weight of individual counterexamples, or, serve as a heuristic to guide the SAT solver to find more desirable counterexamples. Larger satisfying HRs may increase the probability that a test data point falls within its bounds.

Counterexample plausibility

The strategy we chose for encoding our training data has a large impact on our results. For example, if we were to verify SPECs globally, we would very often find that the only way a SPEC is met is by yielding a High Risk alert for every possible input. This defeats the purpose of the AI system which is meant to reduce the number of false positives. By encoding the set of training data, we make an AI-centric design choice. We assume that the training data is comprehensive, which is often a very strong assumption to make. Training data is not the reality of the domain, only a projection of it. It is possible to incorporate expert knowledge in the definition for the region over which TEA verifies SPECs.

Likewise, for GIF specifications, we find that imposing a constraint that counterexamples must look like data we have seen before reduces the number of violations to GIF. A preliminary analysis was shown in Section 4.4.4. There is a delicate balance between verifying GIF in a neighborhood so tight around data that fairness is virtually only guaranteed on training data and verifying GIF in a neighborhood so large that effectively impossible inputs to the system are causing the model to exhibit unfair behavior.

Furthermore, even with adequate bounds on the definition of *global*, there is the added complexity that TEA treats each attribute as independent. A method for taking the probability of a set of inputs into account would further refine TEA and result in more useful counterexamples being produced. Some types of constraints could be encoded directly, e.g. diastolic blood pressure is always less than systolic blood pressure, but others features are correlated in more complex ways, e.g. occupation is not evenly distributed across sex.

This work submits that while working to further incorporate the concepts of plausibility and probability into TEA, our work still represents the entire realm of possibility. Refuting individual counterexamples produced by TEA that seem implausible to a human is an adequate post-hoc strategy for culling irrelevant counterexamples, however, there is room for improvement and we are actively testing ideas.

Counterexample diversity

TEA can enumerate counterexamples to formulas, but the current strategy for doing so could be improved. Enumerating all counterexamples to a particular definition of GIF is hard. For large models, or when very many counterexamples exist, this can become prohibitively expensive. Other than to guarantee that regions without a counterexample are safe, the reason why we enumerate all possible counterexamples is because SAT solvers do not find satisfying assignments uniformly at random. If we find only a subset of the counterexamples, we will likely obtain a result that is not fully representative of the unfairness present in a particular model. A research thrust worth exploring would be to develop an addition to our framework to sufficiently randomize the order of finding counterexamples to the fairness property so that we can assume that a sufficient sample of fairness counterexamples is representative of the structure we would obtain by exhaustively finding all counterexamples.

Critical design specifications are enforced, but never taught to the model

The training of the tree ensemble is independent of the verification of the tree ensemble. Since the models tested in this chapter are not encouraged to obey SPECs during training, it may be unreasonable to expect the models to exhibit significant degrees of compliance with the rules.

TEA explores the tradeoff between verifiable conformance to SPECs and TPR. Developing a training procedure for tree ensembles that incorporates certificate information could potentially build models that adhere to SPECs for more restrictive definitions of the ensemble's predictive threshold. This would mean that the AI system could more frequently avoid yielding FP errors while simultaneously adhering to

SPECs.

In the context of GIF, it is generally unreasonable to expect an ML model to meet strict fairness specifications without some fairness objective during learning, especially because the data often contains biases and the model typically aims to represent the data as faithfully as possible. Developing new training algorithms to promote adherence to any critical design specifications, including individual fairness, may increase the likelihood that these trained models indeed meet all desired specifications. This could counteract the trend for more complex models to be generally less robust to perturbations on inputs.

Assumes critical design specifications exist

A user of TEA must provide specification to verify. There are often no such specifications that are defined by the domain application itself, partly due to the infrequent nature of AI systems undergoing formal verification before deployment. In these cases, we want to be able to search for candidate specification that could be of interest to a human, whether they be a developer or a user of the AI system.

Appendix C presents preliminary results on a strategy for mining intelligible input-output structure present in data and then formally verifying that a trained model adheres to that structure. We present an improvement on the state-of-the-art for the maximal subarray problem. Our Sparse Sub-Rectangle (SSR) algorithm leverages sparsity, which is often a reasonable assumption when searching for a maximal subarray in data. We use a search technique that exhaustively considers all 2D, axis-aligned projections and reports any 2D range rules that map to a singular output label and satisfy search parameters including support and purity requirements. These bounding box rules represent intelligible, intermediate-level concepts that, pending verification, offer provably correct summaries of model behavior. This capability was originally designed to fit into XAI and provide interpretable explanations for underlying structure in their data. While the direction of this thesis is focused more on how formal verification can fit into XAI, we provide details of our novel SSR algorithm and preliminary results in appendices and show that it can be easily integrated into TEA to expand our future capabilities.

Specification development cost

TEA currently automates the process of formalizing all possible voting, tree ensemble structures and a few archetypes for specifications. In order to encode new specifications to test within the TEA framework, it takes knowledge and experience of working with SAT solvers. This makes it hard to allow developers and users to encode their

own specifications. This can be overcome, to some degree, by leveraging more expressive logics, such as SMT, but this is left to future work as the scope of this work focuses on what is possible within a SAT formalism for tree ensembles and design specifications. What we have found is that no formalism was faster at verifying properties of tree ensembles than SAT, and this efficiency is what enables our larger verification experiments.

We are interested in expanding upon SPECs to develop a clinician-in-the-loop system such that it becomes easy to express new constraints on a model and to have TEA perform the verification task. Such an extension of TEA would allow users of the system to become an integral part of the V&V process for their models.

An example of another type of specification we are designing is one that verifies monotonically increasing/decreasing model risk assessment along a trajectory in input space. This could be a vector, or even possibly a contour, verifying level set properties of model behavior. Common sense may say that if point A is more/less dangerous than point B, then the model's risk assessment should make a smooth transition between those two points. An illustrative example of this capability is shown in Appendix B.1.

Model class restrictions

The reason why statistical V&V methods are so popular in the context of AI systems is that they do not require knowledge of the underlying structure of the model. TEA does require this information, and at present, only verifies properties for tree ensemble models. A goal of this thesis is to show how formal V&V can produce more trustworthy AI systems, so we restricted our scope to a single model class and tabular data.

Other model classes could be integrated into our current framework using formalisms such as SMT or more expressive logics. A special type of neural network, XNOR-Nets [3, 142, 158], remove real-valued weights and activation energies from the network to make them expressible in a SAT formalism. Expanding the model classes available to formal verification of individual fairness will give practitioners more choice in a model class that best suits their needs.

Proofs provide more granular information that could be of use

Experiments in this thesis show how certificate information can be used in new ways to inform decision making. Certificates of unsatisfiability provide proof of a logical contradiction that prevents the model from adhering to a specification. There is potentially very useful information in this proof that may guide attention to the root

cause of the discovered model fault. For instance, we may be able to use these proofs to determine whether the fault is likely due to anomalous inputs to the model or due to structural deficiencies of the learned policy.

Appendix D presents experiments we have conducted in SMT to evaluate the feasibility of summarizing formal proofs to provide explanations for model behavior. We experiment with extracting Minimal Unsatisfiable Set (MUS)s that exhibit a unique property such that changing the truth assignment to any literal implicated in the MUS will cause the base formula to become satisfiable. We use this proof summarization strategy to provide counterfactual-style answers to questions such as, *why does the model make an error?*, *how can we prevent the model from making the same error again?*, and *when should we distrust the model?*. This represents another novel contribution of our work that differs from the directions we presented in earlier chapters. Other proof summarization techniques exist, and may be even more effective than MUS extraction given that in order for a MUS to provide an answer to these questions, a single literal has to be responsible for the change. This is rare in cases where the change would require multiple trees in the ensemble changing their vote to get a desired, different output.

6.5 Contributions to the Field

To the best of our knowledge, the following claims represent novel contributions to the field of Artificial Intelligence.

A new SAT formalism for tree ensembles

We contribute a new SAT fomalism for trained tree ensemble models, allowing the research community to leverage SAT technology, which will continue to improve. This means TEA will benefit from future advances in SAT solving strategies and heuristics. Previously, the limited work aimed at verifying tree ensemble models has leveraged other formalisms such as SMT, MILP, or equivalence classes.

There are many different ways one could go about encoding a tree ensemble in logic. We made the design choice to comprise our encoding with atomic literals; we encode nodes, leaves, branches, and individual trees in a way that is equivalent to the base model, not simply equisatisfiable. If our guiding principle was to minimize the number of literals and clauses needed to express a tree ensemble, we would have obtained an equisatisfiable encoding that did not reason about all atomic components of the model. By ensuring that our encoding is equivalent, we are able to summarize proofs in order to provide provably correct explanations for observed model behaviors. Maintaining equivalence also means that we are not making any approximations during the encoding phase, which is a strategy commonly deployed in verifying other AI model classes such as neural networks.

Integrating formal verification in the model selection process

Formally verifying that a model produced with statistical methods adheres to critical design specifications enables both contractual and probabilistic reasoning about model behavior. This provides additional information during model selection that allows developers of AI systems to choose models that exhibit good overall accuracy and satisfy constraints that ensure the model will not inflict easily preventable harm. With TEA it is possible to select the tree ensemble model among a group of candidates that is the most robust (LAR), most fair (GIF), or safest (SPEC).

Tractable verification of global design specifications

TEA is the first system to formally verify Global Adversarial Robustness (GAR) for models of realistic application-scale. Most other work focusing on verifying AI systems focuses on Local Adversarial Robustness (LAR) which is much easier to verify given the small search scope. In the context of fairness, we show that we can enumerate

counterexamples to a global design specification. This means that we have discovered all explicit operational conditions where a global design specification is satisfied and where it is violated.

A formalism for verifying model properties in the neighborhood of empirical data support

Our formalism for data supports the encoding of a ζ-neighborhood around each data point. Other methods bound the search space with circumscribing, convex polytopes. Our disjunctive HR representation of all available empirical data points makes it possible to vary the size of the search area by changing the definition of ζ. It also makes it clear the contribution of each data point to the search space, thus, removing or adding data samples changes the overall scope of the verification task in an intuitive manner. For ML models in particular, verifying properties for all possible inputs may not be a reasonable strategy, due to the fact that trained models are only expected to perform well in regions with sufficient data support that was seen during training. We have shown that imposing this additional constraint is possible when verifying SPECs in Chapter 5.

Next Steps and overall contributions to AI and society

There are multiple opportunities for next steps to use our methods to solve other existing problems. While TEA was designed with AI in mind, it would be possible to apply the same methods to human decision-making processes. Specifically, our methods could be used to identify gaps in policy, where individually safe rules wind up creating strange edge cases where unintentional harm is done. TEA may lead more engineers to reach for AI for critical subsystems if it becomes possible to verify critical design specifications using our framework. We have also shown that TEA provides a mechanism by which laws meant to govern AI behavior could be enforced on trained models. Such ability to regulate AI would put our field in the company of other world-changing technologies ranging from transportation (think Ralph Nader's automobile example) to pharmaceuticals. Lastly, and possibly most exciting is that contemporary research in AI seems to overwhelmingly focus on developing systems with a clean slate. This ignores the AI systems that are currently deployed in the real world, where starting from scratch with new data or new algorithms is not always an option. TEA can be of use in these cases, as our model-centric approach to verification of AI only requires the learned structure of the model, and does not necessary require access to underlying data for select design specifications. Data is often viewed as a commodity worth protecting, therefore it may not always be easily accessible for

inquiry. TEA stands to make a positive impact for those who are responsible for maintaining AI systems and are overlooked by the direction of research that pushes for new, updated modeling. We provide a tool that could concievably be used to test existing models for adherence to important design specifications whether they are tree ensemble models that were trained today or 30 years ago.

We see an uncanny resemblance between the current state of the field of AI and the history of the automobile industry, which we discuss in greater depth in Appendix E. In 1965, Ralph Nader published *Unsafe at Any Speed*, which among other dire issues, identified a lack of industry-wide standards for what it meant for a vehicle to be road-safe. In the absence of any governing regulations for the automobile industry, manufacturers were free to set their own standards and Nader showed that this resulted in easily preventable harm being inflicted on motorists. Furthermore, he claims that manufacturers "exude presumptions of engineering excellence and reliability, and this reputation is accepted by many unknowing motorists".

We see clear parallels with AI. As machines and algorithms become faster and more powerful, is becomes possible to apply AI to more consequential domains. The issue we see is that formally checking the models that are trained from data is rarely part of the AI design pipeline. For many in our field, building a more robust, fair, or safer model simply means to build a better system that achieves higher overall accuracy compared to existing models. Some would reason that at the end of the day, what matters is how often the trained model produces outputs that match the ground truth target, and that by reducing the frequency of errors, the model has fewer opportunities to inflict easily preventable harm to humans.

However, we claim that this mindset represents a willful ignorance to the types of harm that have been well-documented and studied in AI systems to date. Single pixel flips in images can change a confident model prediction of 'red traffic light' to a confident 'green traffic light' [197]. Synonyms in text data from medical records can change clinical risk assessment for patient opioid abuse [62]. AI exhibits gender bias when predicting careers from biographical information [46]. These represent just a few identified weaknesses of AI systems, and characterizing these weaknesses is growing into new subfields of important research within Artificial Intelligence.

While TEA represents one formalism to verify properties for one model class, we can view this work through a broader lens as another research thrust that is geared toward answering the longstanding question of *what did the model actually learn from data?*. Only when developers of AI systems are able to provide contractual answers to that question will the models that they produce be worthy of trust needed to warrant deployment to critical domains.

Appendices

Appendix A

Acronyms

ACAS Airborne Collision Avoidance System

AI Artificial Intelligence

API Application Programming Interface

AUC Area Under the Curve

CDCL Conflict-Driven Clause Learning

CNF Conjunctive Normal Form

DNN Deep Neural Network

FN False Negative

FNRR False Negative Robustness Rate

FP False Positive

FPR False Positive Rate

FPRR False Positive Robustness Rate

GAR Global Adversarial Robustness

GIF Global Individual Fairness

HR hyperrectangle

IF Individual Fairness

IQR Inter-Quartile Range

LAR Local Adversarial Robustness

MILP Mixed Integer Linear Programming

ML Machine Learning

MUS Minimal Unsatisfiable Set

NRR Negative Robustness Ratio

PPV Positive Predictive Value

PRR Positive Robustness Ratio

Reluplex ReLU Simplex Theory Solver

ROC Receiver Operating Characteristic

SAT Boolean Satisfiability

SMT Satisfiability Modulo Theories

SPEC Safety-Paramount Engineering Constraint

SSR Sparse Sub-Rectangle

TEA Tree Ensemble Accreditor

TN True Negative

TNRR True Negative Robustness Rate

TP True Positive

TPR True Positive Rate

TPRR True Positive Robustness Rate

UNSAT Unsatisfiability

VoTE Verifier of Tree Ensembles

V&V Verification and Validation

XAI Explainable AI

Appendix B

Additional Specifications

B.1 Monotonicity

Common sense says if point A is more/less dangerous than point B, then the model's risk assessment should make a smooth transition between those two points. We conduct preliminary experiments on the feasibility of expressing and verifying this constraint.

Algorithm 12: Monotonic risk assessment score

1 $L \leftarrow$ Line segment over which monotonicity is enforced
2 $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3 \leftarrow$ copies of a tree ensemble, $\mathcal{M}$
3 assert $\mathcal{M}_1$
4 assert $\mathcal{M}_2$
5 assert $\mathcal{M}_3$
/* inputs within δ for models */
6 assert $\delta(\mathcal{M}_1, \mathcal{M}_2)$/* See δ constraint in Alg. 9 */
7 assert $\delta(\mathcal{M}_2, \mathcal{M}_3)$
/* vote tallies for positive class within ϵ between models */
8 assert $\epsilon(\mathcal{M}_2, \mathcal{M}_1)$/* See ϵ constraint in Alg. 9 */
9 assert $\epsilon(\mathcal{M}_2, \mathcal{M}_3)$/* LHL, flip order of $\mathcal{M}_2$s to test HLH */
10 assert $\epsilon(\mathcal{M}_1, \mathcal{M}_3)$
/* restrict scope to HRs intersecting L */
11 $DHR \leftarrow \bigvee\{HR \mid \forall HR \in \mathcal{M} \bigcap L\}$
12 assert TseitinCNF(DHR)/* See [186] */

Our encoding strategy, shown in Algorithm 12, involves asserting three copies of a model and specifying that inputs to each should be strictly increasing in attribute

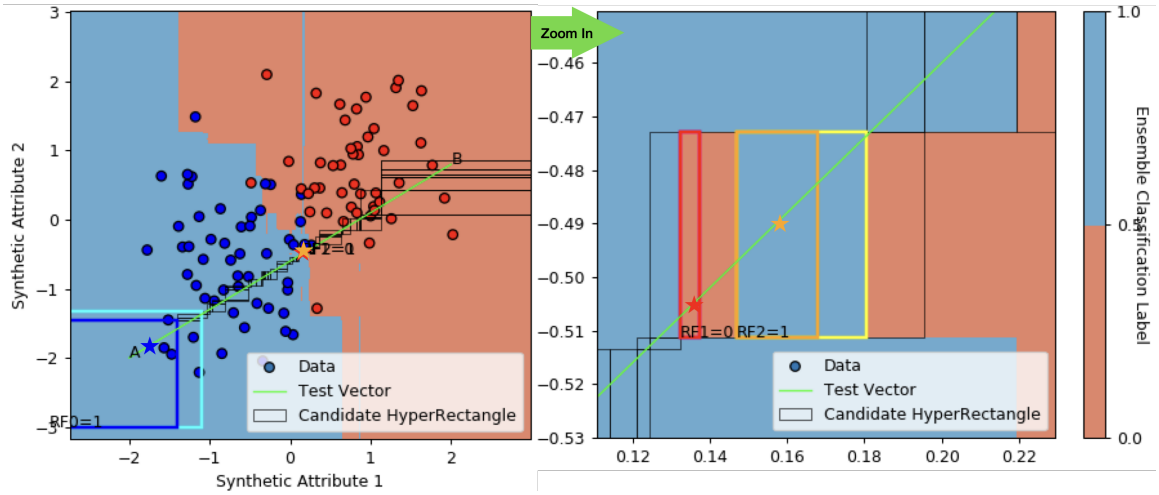


Figure B.1: Illustrative example of a monotonic risk assessment constraint

values. On outputs, we constrain the two models outer models on the line segment to have fewer votes for the target class than the model in the middle of the line segment. If those constraints can be met, then we know that a counterexample exists where a non-end point of the line segment has the highest vote tally for the target class, which reveals non-monotonic risk assessment of the base model. Monotonicity can be verified if Algorithm 12 yields two independent verification tasks, one search for an instance where $\mathcal{M}_2$ has a higher vote tally for the positive class than both $\mathcal{M}_1$ and $\mathcal{M}_3$, and another instance where $\mathcal{M}_2$ has a lower vote tally for the positive class than both $\mathcal{M}_1$ and $\mathcal{M}_3$. The change only requires flipping the order of $\mathcal{M}_2$ arguments when asserting the ϵ constraints between pairs of models.

The illustration in B.1 shows how the resulting counterexample certificate can be interpreted. The line segment we search along is shown in green and all of the intersecting HRs are plotted as well. The highlighted colors represent HRs that when a sample is on the line segment and inside of the HRs, the monotonicty constraint is violated. Blue, Red, and Orange boxes prove the existence of a counterexample to the expressed monotonic constraint.

Current limitations of this approach are our constraint that restricts the scope of the verification task to hyperrectangles that intersect the line segment. We leverage an open source Python library, shapely, which can compute this intersection in one, two, and three-D space. Moving to higher dimensional spaces is possible with this framework, but requires a high-dimensional ray tracing approach which generates necessary list of disjunctive HRs to search. It would be useful to extend further and consider arbitrary paths through the input space. Verifying constant model output over level-sets of contours could be useful in some settings.

B.2 Other constraints on the scope of the verification task

When only encoding the tree ensemble model, the default is that we are performing global searches, but reducing the scope may increase the efficiency of the verification task. Some specifications, such as LAR and SPEC reduce the scope to a local neighborhood or a halfspace. Adding these constraints to the assertion stack will limit the search to the desired extent.

Single query, or, existing hyperrectangle

Algorithm 13: Limiting the search to the hyperrectangle containing a data point

```
1 x ← data sample of interest
2 for att ← 1 to number of attributes do
3 |   lb ← arg max_k{t(k) | t(k) ≤ x_att ∧ a(k) = att}
4 |   ub ← arg min_k{t(k) | t(k) > x_att ∧ a(k) = att}
5 |   assert ¬T_lb/* greater than lower HyRe bound */
6 |   assert T_ub/* lesser than upper HyRe bound */
7 end
```

If we are interested in verifying model behavior on a single query, we set the hyperrectangle that contains the point active, as seen in Algorithm 13. This involves identifying the proximal threshold values that are both lesser and greater than each attribute value of the data point. Finally, setting all lower bound thresholds false and all upper bound thresholds true means that the solver will only consider the model state bounded by that region.

Arbitrary hyperrectangle

At times, we may be interested in defining search bounds that cannot be encoded by the values of existing threshold literals (See Algorithm 14). In this case, we proceed much the same as in the case of searching an existing hyperrectangle, however, we must define new threshold values for the new lower and upper bounds along each attribute in a particular hyperrectangle. Furthermore, we must sort these new threshold literals into the list of existing ensemble model thresholds and enforce ordinality.

Algorithm 14: Limiting the search to new hyperrectangle

$\mathbf{b} \leftarrow$ new hyperrectangular ranges on search space
$\mathbb{A} \leftarrow$ set of attributes in $\mathbf{b}$
for $att \in \mathbb{A}$ **do**
    $B_{att,min} := \min(\mathbf{b}_{att})$`/* Define new literals for new bounds */`
    $B_{att,max} := \max(\mathbf{b}_{att})$
    assert $\neg B_{att,min}$`/* greater than lower HyRe bound */`
    assert $B_{att,max}$`/* lesser than upper HyRe bound */`
end
assert `Ordinality`$\big(\{B_{att,min}, B_{att,max} \mid \forall att \in \mathbb{A}\}\big)$`/* See Alg 3 */`

Subsets of data, or, disjunction of existing hyperrectangles

Algorithm 15: Limiting the search to a subset of data points

$X \leftarrow$ data
$\mathbb{D} \leftarrow$ indices of data points of interest
for $i \in \mathbb{D}$ **do**
    $\mathbf{x} \leftarrow X_i$
    for $att \leftarrow 1$ **to** *number of attributes* **do**
        $lb \leftarrow \arg\max_k \{t(k) \mid t(k) \leq \mathbf{x}_{att} \wedge a(k) = att\}$
        $ub \leftarrow \arg\min_k \{t(k) \mid t(k) > \mathbf{x}_{att} \wedge a(k) = att\}$
        $r_{att} := \neg T_{lb} \wedge T_{ub}$
    end
    $H_i := \bigwedge\{r_{att} \mid \forall \text{ attributes}\}$
end
assert `TseitinCNF`$\big(\bigvee\{H_i \mid \forall i \in \mathbb{D}\}\big)$`/* See [186] */`

Our chosen strategy for encoding data into TEA is to define a ζ-neighborhood HR about each training data point and verify that a property is satisfied over that disjunctive space. There is a simpler way to encode data that involves simply checking the circumscribing HR that each data point fall inside. The risk of this approach is that all HRs have different dimensions, meaning that the area searched is not uniformly weighted between all training data. However, in some cases, it may be useful to check a property for each data point, and in that case, 15 will search over a disjunction of HRs predefined by the learned structure of the tree ensemble model itself.

Appendix C

Mining Data for Candidate Specifications

This Appendix presents methods and results for earlier XAI work. While it does not easily fit into the scope of the chapters of this thesis, it still represent novel technical contributions to the field.

We provide background on the maximal subarray problem and describe how our method, SSR leverages sparsity to improve algorithmic complexity for solving the maximal subarray problem. Prior work by Kadane and Bentley do not take advantage of sparsity, which is often a reasonable assumption in the maximal subarray problem. We show how SSR can be incorporated into TEA in order to search for candidate specifications which may be interesting to verify for a given model.

C.1 Sparse Sub-Rectangle (SSR): an intelligible design specification

The Maximal Subarray Problem

The maximal subarray problem is a classic computer science problem. Given an array, the goal is to identify a subset of contiguous entries that achieve a summed value greater than any other contiguous subset within the array. Interestingly, much of the history of this problem was developed at Carnegie Mellon University!

Kadane's Algorithm for 1D

Jay Kadane, a former faulty member at CMU, developed what is still today known as the optimal algorithm for identifying the maximal subarray for a one dimensional

array. Kadane gave a linear-time dynamic programming algorithm for finding the maximum sum interval in one-dimensional arrays. By storing auxilliary information, including the index of the first element of the contiguous set, the sum up to the current element, and the greatest sum so far, it's possible to solve the problem in linear time, $O(n)$, with respect to the number of elements in the array. The key insight is that any time the current sum turns negative, it is time to reset the first element of the contiguous set.

Bentley's Extension for 2D

Unfortunately, Kadane's algorithm only applies to one-dimensional arrays. Jon Bentley, another former faculty member at CMU, leveraged Kadane's algorithm as a subroutine to find the optimal 2D subarray. This meant running Kadane's algorithm n^2 times, yielding an $O(n^3)$ algorithm for finding the maximal subarray in two-dimensional arrays. This beat the previous naïve bound of $O(n^4)$ which involved testing all n^2 possible starting and n^2 possible ending entries.

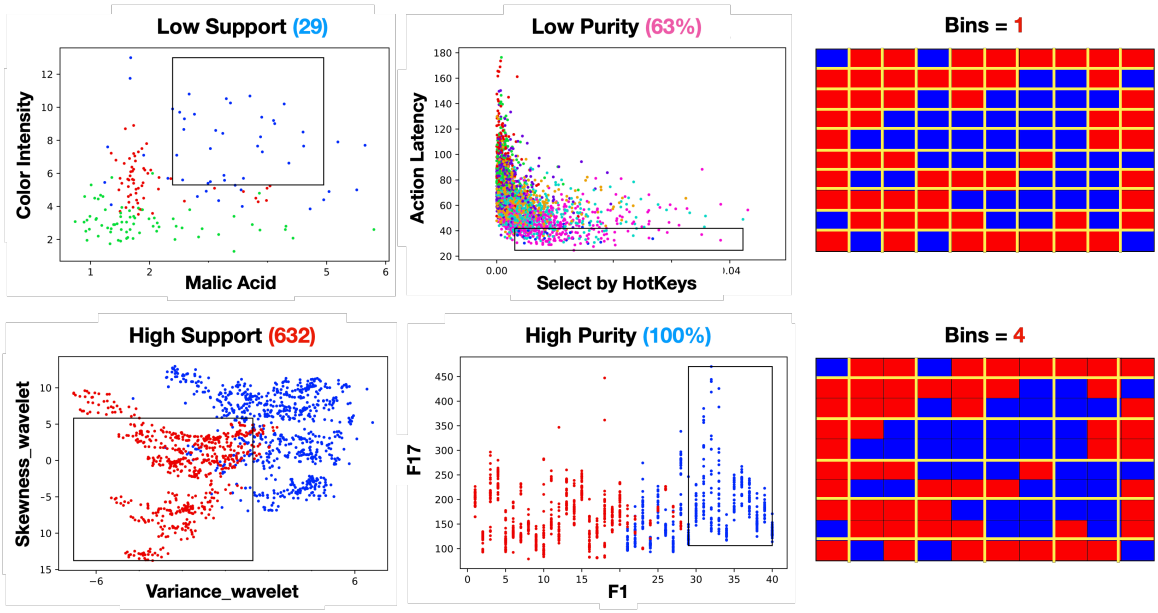
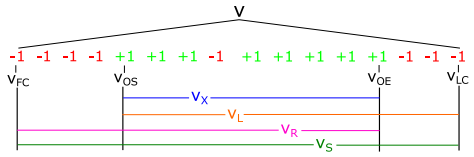


Figure C.1: Illustration of how support, purity, and binning influence search.

The purpose of this section is to describe the strategy we use to mine data for candidate intelligible explanations. This spares us the challenge of needing to define our own intelligible explanation candidates, which often requires knowledge that we don't have, such as domain expertise, or knowledge of the inner workings of the model. We use a search technique that exhaustively considers all 2D, axis-aligned projections and reports any 2D range rules that satisfy search parameters.



- The First Column in the subtree, `fc`.
- The End Column in the subtree, `ec`.
- The Optimal Start column in the subtree, `os`.
- The Optimal End column in the subtree, `oe`.
- The sum X for range bounded by `os` and `oe`.
- The sum L for range bounded by `fc` and `oe`.
- The sum R for range bounded by `os` and `ec`.
- The sum S for range bounded by `fc` and `ec`.

Figure C.2: Auxiliary information stored at each node for SSR

For a brief reminder on the background associated with the Maximal Subarray Problem, see Section C.1. Kadane and Bentley's algorithms do not take advantage of sparsity, which is often a reasonable assumption for finding the maximal subrectangle in many applications. We developed an algorithm which does leverage sparse conditions to yield a better time complexity bound.

- $\texttt{x}(i_p) = \max\{\texttt{x}(i_{lc}), \texttt{x}(i_{rc}), \texttt{r}(i_{lc}) + \texttt{l}(i_{rc})\}$
- $\texttt{l}(i_p) = \max\{\texttt{l}(i_{lc}), \texttt{s}(i_{lc}) + \texttt{l}(i_{rc})\}$
- $\texttt{r}(i_p) = \max\{\texttt{r}(i_{rc}), \texttt{s}(i_{rc}) + \texttt{r}(i_{lc})\}$
- $\texttt{s}(i_p) = \texttt{s}(i_{lc}) + \texttt{s}(i_{rc})$
- $\texttt{fc}(i_p) = \texttt{fc}(i_{lc})$
- $\texttt{ec}(i_p) = \texttt{ec}(i_{rc})$
- $\texttt{os}(i_p) = \{\texttt{x}(i_{rc}) > (\texttt{r}(i_{lc}) + \texttt{l}(i_{rc}))\ ?\ \texttt{os}(i_{rc})\ :\ \texttt{os}(i_{lc})\}$
- $\texttt{oe}(i_p) = \{\texttt{x}(i_{lc}) > (\texttt{r}(i_{lc}) + \texttt{l}(i_{rc}))\ ?\ \texttt{oe}(i_{lc})\ :\ \texttt{oe}(i_{rc})\}$

Figure C.3: Update rules for SSR

The core of the algorithm is based on Bentley's extension; for all possible starting and ending rows in a 2D array, we compute the optimal contiguous selection of columns. The difference is that instead of running Kadane's algorithm as a subroutine, we use a search tree data structure to compute the optimal contiguous selection of columns. After adding any component to proper leaf of the tree, the data structure updates auxiliary information (See Figure C.2) which propagates the optimal selection of contiguous columns up to the root node. Since adding zero elements will never change the optimal subarray, they need not be added at all. In the worst case, this algorithm matches the complexity of Bentley, however, if there are a significant

amount of zeros or missing values, then our algorithm runs faster $O(n^2 \log^2 m)$.

Going into further detail, there are updates that need to be performed for the sum variables as well as the column index variables. All updates (see Figure C.3) for a parent node depend on whether the left child, right child, or a combination of the two achieve the highest sum value.

For sake of simplicity, we choose to break ties randomly. This works for our use case, as we are interested in using this algorithm to identify interesting low-dimensional structure in data; that structure does not necessarily need to be optimal. Other choices for handling ties would make sense in other scenarios. If we were interested in identifying the biggest rectangles possible, we could break ties by combining. If we are interested in obtaining the optimal sparse subrectangle, a tie requires a breakpoint set to backtrace. Only after evaluating the final state of the tree after each choice would we know which option is truly optimal. Figure C.4 shows an example of what the data structure would look like given our choice of starting and ending rows.

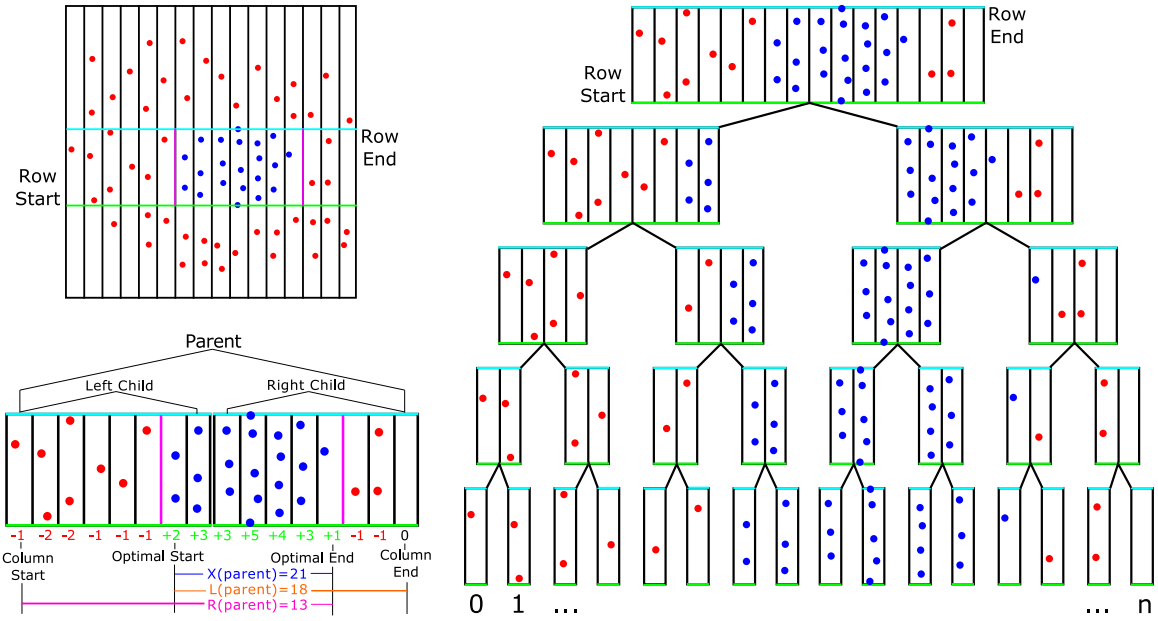


Figure C.4: Sparse Sub-Rectangle (SSR) Algorithm Illustration

In order to exert control over the types of boxes that we discover with the sparse subrectangle algorithm, we can tune search parameters to our needs. Speed of the algorithm can be influenced by introducing batch insertions. By adding multiple elements within one bin to the tree at one time, we reduce runtime not only by reducing the number of queries to the tree, but also limiting the combinatorial choices

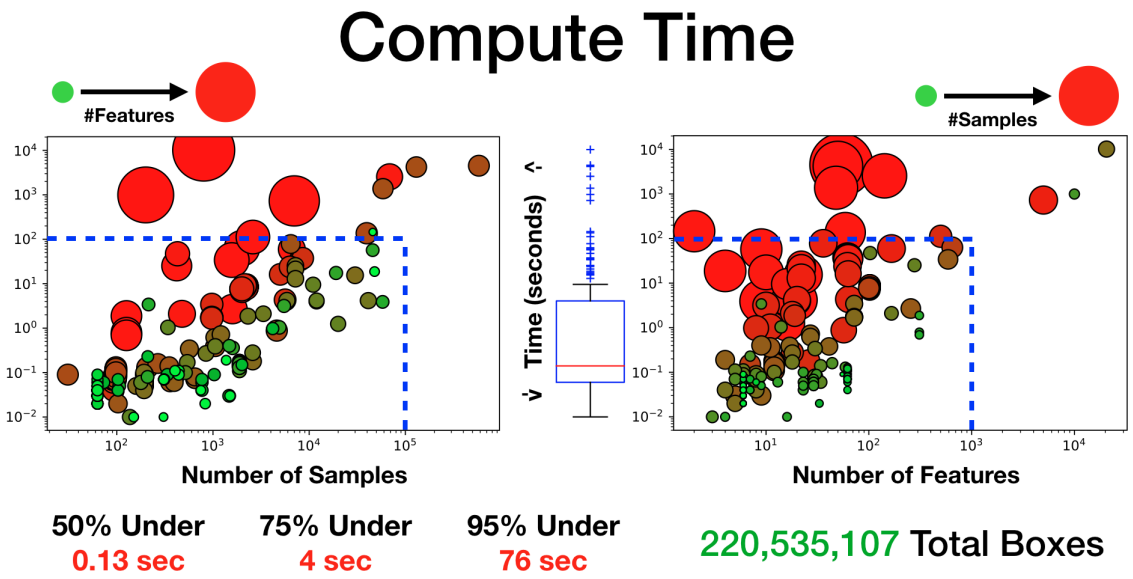


Figure C.5: Run-Times for 202 defined learning tasks on 95 unique, publicly available data sets

for the start and end rows. Additionally, we can set minimum support or purity requirements for any returned box. Figure C.1 shows how search results change based on different search parameters. These search parameters can be leveraged to search for different types of low dimensional structure that may be interesting in different settings, perhaps even including purposeful searches for poor boxes rather than optimal ones.

Generally speaking, the Sparse Sub-Rectangle (SSR) Algorithm can be run with whatever time budget it is allowed. Figure C.5 shows the relationship between the number of samples, number of attributes, and run time for over 95 unique, publicly available data sets. As expected, the run time of the SSR algorithm is much more sensitive to the number of attributes in data, since we must run an instance of SSR for all possible 2D, axis-alined projections in data. Figure C.5 shows that data sets with roughly 10^5 samples and a variable number of features will halt within 100 seconds, while data sets with roughly 10^3 attributes and a variable number of samples will halt within the same period. While the number of samples plays a role in run time, it is less pronounced than the number of attributes in data, and furthermore, it can be more easily managed by tuning the binning parameter to decrease the number of insertions into the SSR search tree.

Algorithm 16: SSR specifications

$\mathbf{b} \leftarrow$ hyperrectangular bound on search space over inputs
$y' \leftarrow$ constraint on model outputs
$\mathcal{H} \leftarrow$ hypothesis for a relationship between input-output
$C \leftarrow$ number of unique class labels
assert `Search`$(\mathbf{b})$`/* See Alg 14 */`
assert $\mathcal{H}$`/* must convert to CNF before asserting */`
assert $\bigvee\{P_{c,\mathcal{M}} \mid c \in [C], c \neq y'\}$`/* any other class for output */`
assert `Ordinality`$(\{\mathbf{b}_{att,min}, \mathbf{b}_{att,max} \mid \forall att \in \mathbf{b}\})$`/* See Alg 3 */`

Interpretability

To the best of our knowledge, proposing to use formal methods to verify the correctness of intelligible explanations of model behavior is a novel. One possible view of what makes a good explanation for model behaviors is an input-output constraint; under certain circumstances, the model will always yield a prescribed output. Verifying a model's adherence, with an `UNSAT` certificate, to such a candidate explanation either shows that the input-output explanation and the trained model yield identical outputs within some defined range. Or, a `SAT` certificate proves that there is at least one exception to the proposed explanation, and a counterexample can be produced.

The difficult part of verifying whether intelligible interpretations exist for a particular model is generating intelligible hypotheses to verify. These intelligible interpretations of model behavior represent *intermediate-level concepts* which are needed in order to provide good explanations. The next section discusses how we leverage our prior work to generate candidate explanations that are 2-dimensional, axis-aligned bounding boxes. It is worth noting that any type of structure (no matter how complex) could be a plausible candidate for explanation verification, however, we choose to use our SSR because low dimensional structure is generally considered more useful to communicate when providing explanations to humans [29, 63, 93, 95, 114, 122, 125, 196].

Encoding such a specification first requires asserting the bounds on the search space to a potentially new HR (see Alg. 14). Next the mapping between the inputs and outputs must be expressed in CNF and then asserted. We must assert that anything other than the prescribed output is yielded, which will posture our verification task as a search for `UNSAT` certificates showing that the explanation is never violated. Lastly, if a new HR is defined, it's bounds need to be worked into the general ordinality constraint described in Algorithm 3. Inserting these new literals into the sorted list of model threshold values will establish ordinality. While ordinality can be established at any time when considering solely the encoding of the tree ensemble

model, it is necessary to hold off on asserting ordinality until the end of the encoding if additional design specifications, such as the ones described here, are being verified.

C.1.1 Verifying model interpretability with TEA

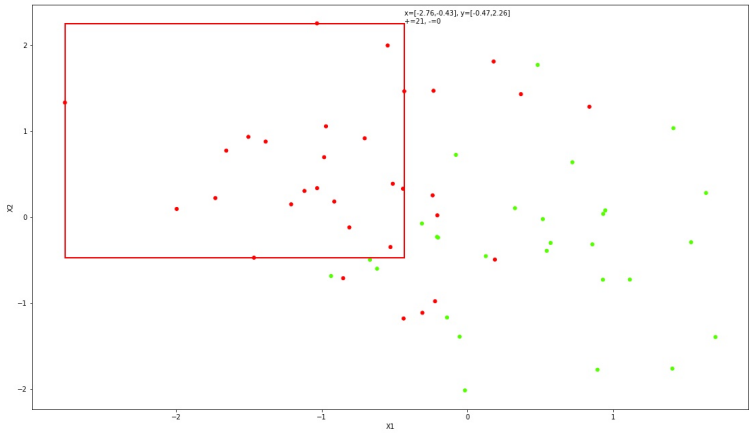
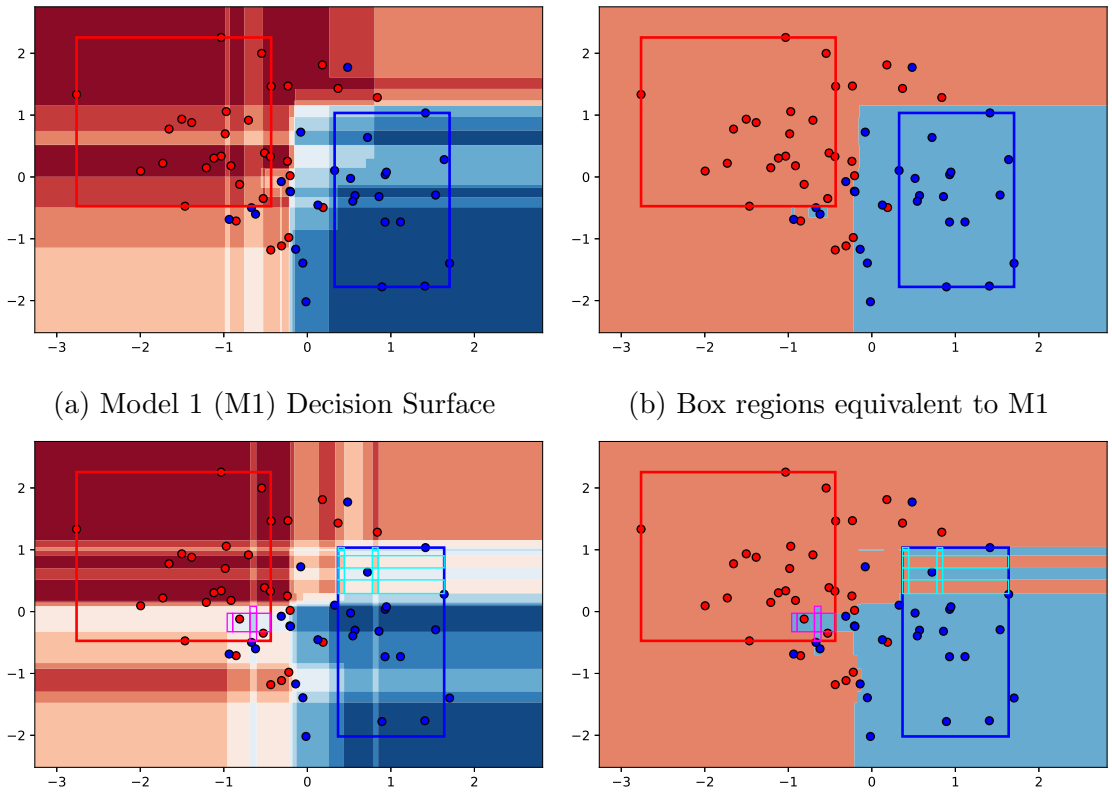


Figure C.6: The box returned by SSR can be used as an intelligible specification.

To run an experiment focused on verifying the existence of an interpretation of the model, we need to define the concept that we believe is intelligible in logic. There will always exist a concept that is a valid interpretation of the model, because the model itself can be treated as the concept. However, this is uninteresting; instead, we are interested in defining an intermediate level concept that resides somewhere between the full complexity of the trained random forest model, and single boolean literals.

We make a design choice to deploy the Sparse Sub-Rectangle (SSR) Algorithm to identify 2D, axis-aligned range rules that serve as our intermediate-level concepts that we would like to prove are correct interpretations of a random forest model. Figure C.6 shows what the SSR algorithm sees when looking for maximal sum ranges in the data. This box meets the support and purity requirements set for this particular experiment, but these parameters can change to yield different candidates for boxes. The interpretable specification defined by these boxes reads as, *if inside the red box, then the model predicts red*, and similarly for the blue box.

Figure C.7 shows the results of a verification task for two different trained random forests on two candidate boxes. The top row, subfigures C.7a and C.7b, shows how the candidate boxes are verified to match the output of M1 within their ranges. The bottom row, subfigures C.7c and C.7c, shows how M2 is not certified equivalent to the candidate boxes within their ranges. Counterexamples are highlighted in magenta



(a) Model 1 (M1) Decision Surface

(b) Box regions equivalent to M1

(c) Model 2 (M2) Decision Surface

(d) Box regions not equivalent to M2

Figure C.7: Interpretability certificates for two models. Boxes yield same prediction as the model over their operational range in M1. Counterexamples exist for M2.

and cyan.

For models of realistic scale and complexity, it is highly likely that a simple 2D range rule will not be verified equivalent within it's bounds. Instead, we can hope to enumerate the counterexamples as seen in subfigures C.7c and C.7d, and add these exceptions to the box rule. This would change our original interpretability spec to read as, *if inside the red box AND not inside any of the counterexamples, then the model predicts red*. Often, we find that the exceptions to the bounding box rules are either few, or devoid of any empirical data. Since the decision surface of a random forest model is a function of superimposed decision boundaries from all constituent trees, it results in a lot of tiny HR intersections, very often with no data to support the prediction made by the model at that HR.

An example of such an interpretability certificate with exceptions can be seen in Figure C.8. This data set is the publicly available Statlog Image classification dataset, which contains 19 featurized, real-valued attributes. The blue box was discovered by

```
Predict SKY if In Here

118.000000 <= ValueMean <= 147.000000
-2.519165 <= HueMean <= -2.009854

AND

        NOT In Here

         158.500000 <= RegionCentroidRow
         23.222222 <= IntensityMean <= 82.981480
         RawredMean <= 71.777779
         47.500000 <= RawblueMean
         22.555556 <= RawgreenMean
         ExgreenMean <= 1.055556
         118.000000 <= ValueMean <= 147.000000
         -2.519165 <= HueMean <= -2.009854

        OR

        NOT In Here

         155.500000 <= RegionCentroidRow <= 156.500000
         IntensityMean <= 23.222222
         RawredMean <= 71.777779
         29.277778 <= RawblueMean <= 40.444447
         22.555556 <= RawgreenMean
         ExgreenMean <= 1.055556
         118.000000 <= ValueMean <= 147.000000
         -2.244613 <= HueMean <= -2.009854

Currently, no data falls within exceptions, so box produces the same label as model
```

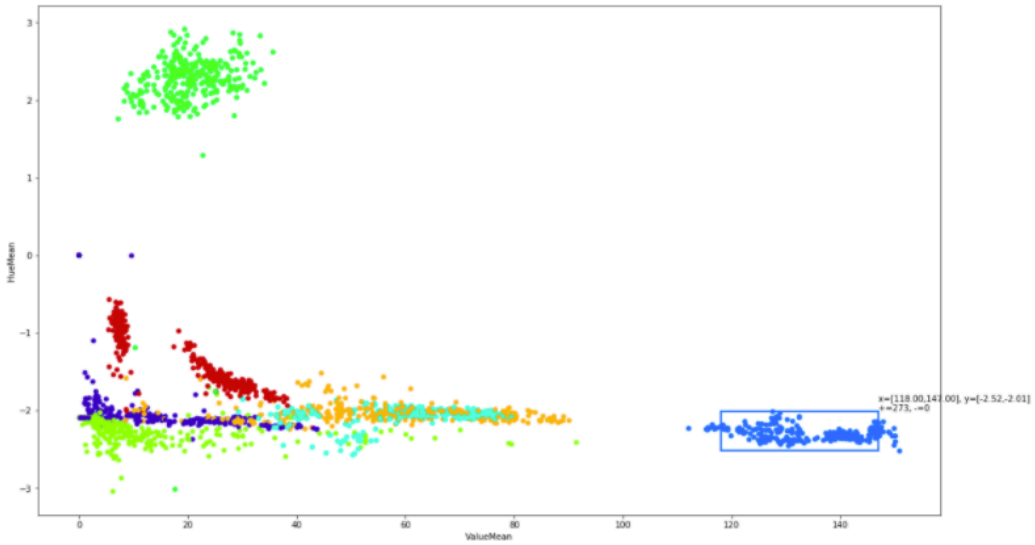


Figure C.8: Example of a verified, interpretable explanation, with explicit exceptions, for model behavior.

the SSR algorithm, and subsequent model verification within its bounds shows that the model's output will always be blue except for the two exceptions explicitly stated at the top of the figure. When no empirical data resides inside the not-in-here HRs, we argue that the x,y ranges can be used to provide a simple, verified rule to describe the model's behavior.

Appendix D

Model Centric Explanations for Undesired Behavior

This appendix presents earlier experiments on whether formal proofs can be summarized to provide an explanation of model behavior. We viewed the task of providing an actionable explanation for observed model behavior as an instance of the diagnostic problem. The *diagnostic problem* for circuits is defined as follows:

> "Suppose one is given a description of a system, together with an observation of the system's behaviour which conflicts with the way the system is meant to behave. The diagnostic problem is to determine those components of the system which, when assumed to be functioning abnormally, will explain the discrepancy between the observed and correct system behaviour." [161]

We conduct a similar interrogation except on encoded AI systems. When the model violates a particular design specification, diagnosing the violation means that we are interested in determining which literals in the encoding form the basis of the violation. UNSAT certificates and proofs describe a contradiction present between select literals and clauses in the logic formula. These proofs can comprise quite a few literals, but it is possible to minimize UNSAT cores via different methods. One method involves a leave-one-clause-out approach [80]. This involves re-verification of the formula with one clause removed. If the re-verification task returns `UNSAT`, then the corresponding clause is eliminated from the UNSAT core. All remaining clauses are part of the MUS and have the additional property that allowing the machine to reassign their value will cause the model to adhere to the design specification that originally presented a logical contradiction. For experiments reported in completed work, we use this approach.

Another method involves reducing the length of the UNSAT core via DRAT-trim [87], a proof checking tool that optimizes proofs by removing lemmas that represent redundancies in the proof, or Resolution Asymmetric Tautologies (RAT). For proposed experiments, we will be using this approach as it is optimized for SAT.

Perhaps most interesting result from these experiments is that we demonstrate that it is possible to use formal verification to manipulate the learned structure of a tree ensemble model such that the changes prevent an observed fault from ever manifesting again, while also preserving the overall accuracy of the model. This is only possible by leveraging more expressive logics available with SMT which allow us to define the model both as a collection of parts, and as a function, giving the verifier access to knowledge it needs to suggest edits that do not reduce accuracy.

D.1 Diagnosing model behavior with SMT and Z3

We ran these experiments before we developed our SAT formalism for trained tree ensemble models. SMT was easier to prototype to evaluate the utility of experiments such as these, and with only one notable exception discussed later, the same methodology only requires minor tweaks to the experimental pipeline if we were to run these experiments again with our SAT framework. Our SMT solver of choice is Z3 [48]. We omit detailed instructions on our particular encoding strategy with SMT-LIB, the standard input to many publicly available SMT solvers. The description of our SAT encoding strategy could be directly encoded into SMT-LIB, since SMT is a more expressive logic than SAT.

In the following experiments, we search for explanations for three commonly cited research questions in the field of XAI [78] to motivate the utility formally verifying model adherence to design specifications. This list is by no means comprehensive, as we can search for explanations for any model property that we can encode explicitly in logic.

For consistency across these questions, we consider one set of data – the 1984 Congressional Votes data set [40]. This data set is chosen for a few reasons. The attributes are intuitive, meaning that we can understand what it means to change a vote for a particular bill if we look up a short description of the bill itself. The inference task is intuitive as well, the goal is to predict whether a voting record is one of a Democrat or a Republican. Finally, given that the inputs and outputs are intuitive, the select votes or model parameters make good explanations for why a political party affiliation was predicted by the model.

All of these experiments demonstrate the utility of using automated reasoning to provide explanations for model behaviors. With evidence that this approach to

explaining model behaviors does in fact work in some cases, we turned our attention to running additional experiments in pure AI. This allows us to consider bigger problem instances, and vastly decreases the run time, as will be shown in the next sections results.

D.1.1 Why does a model make a classification error?

Most often, end-users desire explanations either when the model disagrees with their intuition or the ground truth label. In these cases, pointing to discrete components in the model can help explain why the observed prediction occurs. This experiment is aimed at determining whether we can provide answers to the question, *why did the model make a mistake?*.

The question of *why did something* ***not*** *happen* is targeted at the existence of some sort of contradiction. In our case, we are looking to discover a set of literals that, together, form a logical contradiction at odds with the desired model output.

To start the experiment, we train a scikit random forest model on a training partition of the 1984 Congressional Votes dataset. This dataset contains 435 voting records over 16 bills, and the inference task is to predict party affiliation, Democrat or Republican, from an individual's voting record. We apply it to holdout partition, and tag all resulting model errors for analysis. For each of these errors, we spin up a separate diagnostic instance to figure out why the error occurred.

To diagnose these discrepancies, we make the following assertions:

- All assertions needed to encode the trained random forest model in SMT logic
- Assert that the output of the model must be the same as the ground truth

We know that this set of assertions will fail the verification task, since we are only running this diagnostic test because we have empirically determined that the model does not yield the same label as ground truth. Thus, Z3 produces a certificate of `UNSAT`, and along with it, a proof of unsatisfiability. Perhaps it is worth noting here that this not need be the only modality for this type of analysis; it may be of interest to know why the model produces a correct prediction, in which case, we would change the second assertion in the list above to *the output of the model disagrees with ground truth*.

An unsatisfiable core is generated from the proof of unsatisfiability. An UNSAT core is a collection of select literals and clauses that, together, form a logical contradiction that forces the verification task to fail. Since these cores can be quite expansive, we are interested in summarizing in order to provide intelligible explanations to any end user. We minimize our UNSAT core through a straightforward, deletion-based

core extraction algorithm proposed by [80]. The advantage of extracting a Minimal Unsatisfiable Set (MUS) is that allowing the machine to reassign the value of any implicated literal will resolve the logical contradiction, and in our case, allow the model to output the desired class label that agrees with ground truth. This MUS is itself an explanation, as all literals implicated in the core can, for simplicity, be considered individually responsible for the observed disagreement with the prescribed model behavior. These explanations are quantitative in nature, and intelligible to end users because discrete components of the model and/or input are tagged for inspection.

An ensemble of 20 decision trees produced lengthy unsatisfiable cores which were minimized to arrive at a MUS. Table D.1 shows categorized explanations taken from the MUS. The specific misclassified samples in holdout data are listed with the true and predicted labels. What follows on the right are three columns which represent different types of provably valid explanations. The explanandum, the fact that the model disagrees with ground truth, can be explained by properties of the query or properties of the model.

The way to interpret this table is that changing any literal would resolve the contradiction showcased in the MUS. The simplest way to resolve the contradiction would just be to change the ground truth label, which may be reasonable in some cases, but more often than not the goal of this analysis is to determine which constituent model components are responsible for the observed misclassification. Attribute value literals make for particularly intuitive arguments as to why the model arrived at it's assigned label. For instance, sample 117 reads as *the model thinks this is a* ***Democrat's*** *voting record because they voted against a bill which imposed limits on healthcare spending.* It turns out that this is very indicative of a Republican's voting record in the data the model saw during training. Other interesting explanations arise, like for sample 4, where the model is incorrect and the analysis reveals *the model thinks this is a* ***Republican's*** *voting record because they voted against the budget resolution, and for limits on healthcare spending.* In 1984, congress was controlled by Democrats, so a vote against the adoption of the budget resolution does intuitively make sense for the reason why the model did not label this voting record as Democrat. Detailed information about this data set and information that may help the reader reason about the domain and the other explanations in this table can be found here [40].

Besides values of query attributes which are responsible for model behavior, it is also possible to identify model parameters that force undesired model output by the nature of the decision logic structure. The two parameters we consider for explanations are the threshold values of rules in each node and the label assignments for each leaf node. We ignore branch literal explanations, as the complexity of the problem greatly increases if the structure of the tree is subject to change. For sample 100,

assigning a different value to either threshold or reassigning the prediction label for a single leaf in Model 1 would present a path toward satisfiability, therefore, we could consider these model parameters as explanations for the discrepancy we are observing. On the other hand, the large number of literals that show up for the explanation of sample 189 indicates that the 20 tree forest is evenly split on the sample, thus, changing any one parameter to change the value of a single tree would sway the entire ensemble in favor of the ground truth label. Ultimately, it is up to an end-user to pick and choose what type of explanation they wish to accept among the multiple options that our framework offers.

As seen in the explanation for sample 191, it is sometimes possible that there is not one literal that solely bears responsibility for the undesired outcome. This is due to limitations in our extraction of the MUS, where we do not search for complex interactions between implicated literals in the unsat core. For example, it is easy to see that reassigning the value of two or more literals simultaneously could equate to changing a single disjunctive clause. It would be interesting to explore more sophisticated MUS extraction techniques with further experimentation.

To the best of our knowledge, this is the first time a framework has produced explanations for observed behavior that formally assess both input parameters and model parameters simultaneously. It is clear that the relative quality of the data and the quality of the model both play a role in the observed outcome that an end-user might want to explain.

ID	Truth	Model	**Attribute Values** (0:Nay, 1:Yea, 2:?)	**Threshold Values**	**Leaf Labels**
4	DEM	REP	ElSalvadorAid=1, PhysicianFeeFreeze=1, AdoptionOfTheBudget=0	T0-M1	[]
38	REP	DEM	Crime=1, EducationSpending=0, SynfuelsCorpCuts=1, ElSalvadorAid=0	[]	[]
42	DEM	REP	EducationSpending=1, PhysicianFeeFreeze=1	[]	[]
100	DEM	REP	ExportAdminAct=2, SuperfundRightToSue=1, EducationSpending=1, Immigration=1, ElSalvadorAid=1, PhysicianFeeFreeze=1, HandicappedInfants=0	T18-M1, T0-M1	L28-M1
117	REP	DEM	PhysicianFeeFreeze=0	[]	[]
189	DEM	REP	DutyFreeExports=0, EducationSpending=0, MxMissile=0, ElSalvadorAid=1, PhysicianFeeFreeze=1, AdoptionOfTheBudget=0, WaterCostSharing=1	T18-M19, T6-M19, T0-M19, T10-M16, T9-M16, T1-M16, T7-M12, T5-M12, T2-M12, T0-M12, T9-M11, T8-M11, T0-M11, T8-M9, T6-M9, T5-M9, T0-M9, T5-M8, T2-M7, T2-M5, T0-M5, T6-M4, T0-M4, T11-M3, T10-M3, T0-M3, T12-M1, T9-M1, T8-M1, T0-M1	L21-M19, L12-M16, L8-M12, L16-M11, L10-M9, L9-M8, L6-M7, L16-M5, L15-M4, L12-M3, L14-M1
191	DEM	REP	[]	[]	[]

Table D.1: Explanations for why a model prediction disagrees with ground truth label.

D.1.2 What would it take for the model to fix the error?

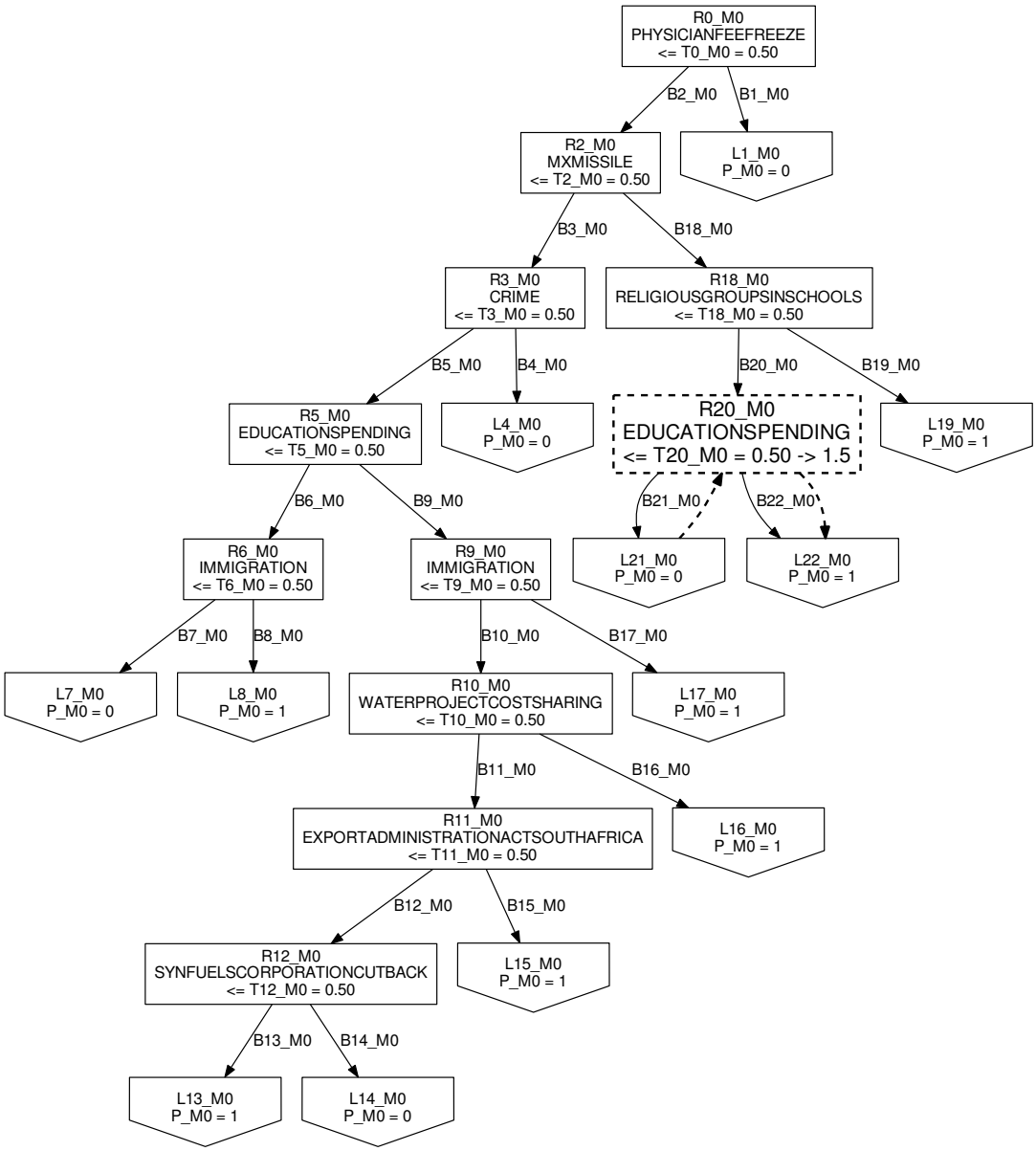


Figure D.1: Example of one tree in a forest annotated with a provably safe intervention from Z3.

A natural follow-up question to the last experiment, which identified reasons why the model made a mistake is *how can we prevent the model from making the same error again?*. The challenge for this experiment is that even though we are guaranteed

to resolve the conflict by changing the assignment of any literal in the MUS, we are not guaranteed to preserve model performance for all the other data we have seen before. It would not be useful to identify changes to make to the model if they wind up spawning many more issues for queries we have not evaluated yet.

We show that an SMT solver is capable of prescribing interventions that are capable of fixing a misclassified sample by making changes to model parameters. We are able to annotate the random forest with interventions that cause the ensemble to change an incorrect classification for a particular query without negatively impacting performance accuracy on data the model has seen before. One way to think about this intervention is that it exploits some idea of a nullspace of the trained random forest model; in the absence of empirical data that reside in certain subspaces, there are multiple models that may fit data equally well. The way we do this is by encoding the random forest model as a function in SMT as well as a set of assertions in propositional logic. Thus, for any change we wish to make to the model, the solver reasons about the accuracy the model would have on samples that we have seen before. For a proposed change to be of interest for this experiment, it must not cause the accuracy of the model to decrease.

Figure D.1 shows an example of a single decision tree (given the limited space for visualization) with an intervention suggested by Z3 that causes the model to change its output for the query in question without negatively affecting accuracy on other data. At worst, the model just changes which samples it classifies incorrectly, not the number of samples. In the case of Figure D.1, Z3 suggests to change the threshold for $T_{20}M_0$ from 0.5 to 1.5, which functionally means that the rule will only evaluate $False$ if the representative with the record in question abstained from recording their vote on the education spending bill. A user can spot the dashed border along with the dashed line redirecting the query's decision path.

We believe this is an interesting and novel application of SMT applied to random forests. Even if Z3 proved that there were no possible interventions, we would have certified a pareto-optimal state of our model, where the effective decision null space has been exhausted and cannot accommodate another intervention without negatively impacting prediction accuracy. In terms of practical applications, we envision that this technique could be used to certify the avoidance of a particular catastrophic failure event.

This particular analysis is the most computationally expensive of our experiments given the addition of constraints needed to set lower limits for allowable training data performance. Additionally, this experiment is the exception to the statement that these SMT experiments can be run relatively easily as instances of SAT. The random-forest-encoded-as-a-function leverages theories of arrays, bit-vectors, and real

values, which are unavailable to us in SAT.

D.1.3 What did the model not learn?

For this experiment, we will focus on the case where an ensemble experiences total confusion; no plurality emerges among constituent models for label assignments. There are other cases which could be of interest as well, such as a 60/40% split, which many would consider to be evidence of ensemble confusion as well. We require a formal definition of *confusion* so if we want to check for other low-confidence vote tallies, we could either run those verification tasks separately or enforce an *AtLeastOne* constraint across a set of possible vote tallies that we consider confusing for this experiment. In the case where no plurality victor emerges in the ensemble vote tally, for a random forest with an even number of trees fit to a binary classification problem, this means identifying HRs where the vote tally is split $N/2$ to $N/2$ where N is the number of trees in the ensemble. A low-confidence prediction should not be trusted, as it may represent a subspace in which the ensemble is not well fit to the data. This is particularly true for voting ensembles, where a common strategy involves breaking ties randomly. This means that the classifier will not be performing better than random in subspaces where it is experiencing confusion between two classes.

One of the benefits of expressing our trained random forest model with a formal system is that we may reason about the space of Integers and/or Real numbers. Said differently, this allows us to analyze the model in the absence of empirical data, of course, other than the data used to train the model. We leverage this capability to generate synthetic data by finding satisfying inputs to the random forest's logic formula. These satisfying inputs can be constrained to exhibit certain properties, such as the property of confusion we are trying to exploit.

We use Z3 to search for confusing data and then we analyze its distribution alongside empirical data. If there is overlap between the confusing distribution and the train/holdout distributions, then some of the empirical data will be given an untrustworthy prediction label by the model. There may also be samples that are close in distance to the confusing queries, which would suggest that they too may be given untrustworthy model predictions. We generate the same number of samples as there are empirical data combined across training and holdout data to characterize the confusing.

We use Z3 to search for confusing queries by running an iterative loop where once a satisfying assignment is found, that query is explicitly forbidden in future searches. SMT differs from SAT in the sense that a satisfying assignment in our SMT encoding is a single point, wherease a satisfying assignment in our SAT encoding is

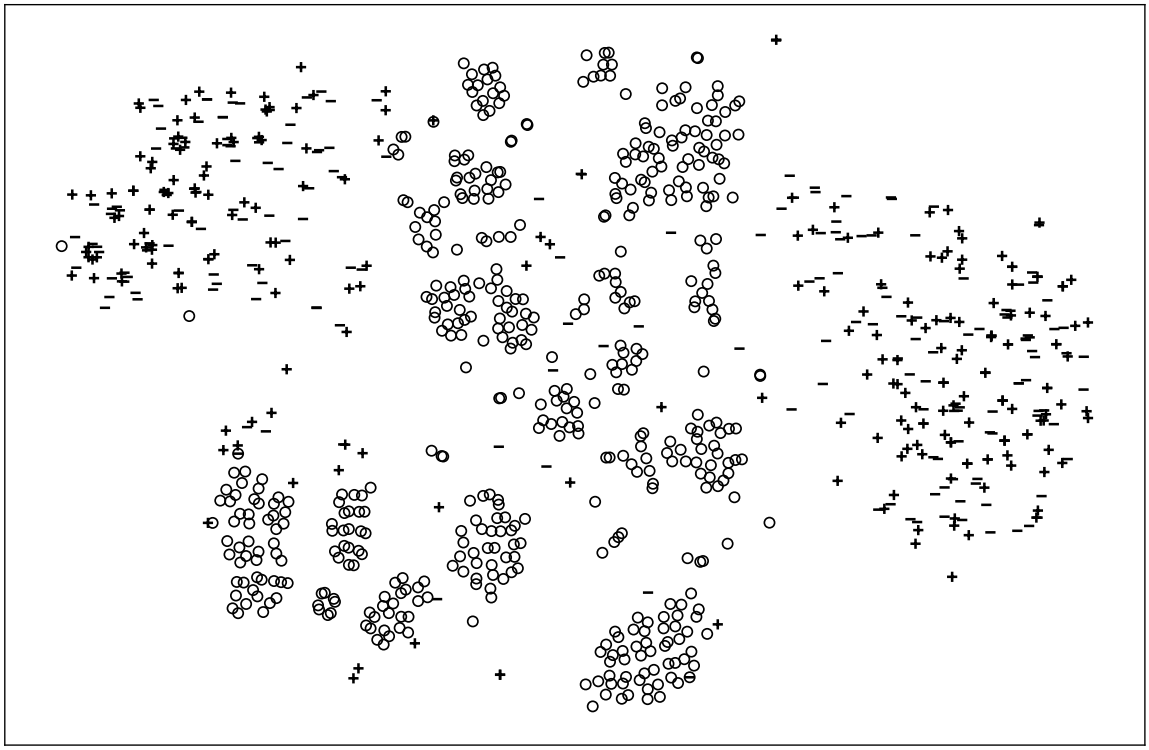


Figure D.2: t-SNE embedding showing train (+), holdout (-), and confusing (o) data. Model predictions for empirical samples that are caught inside the synthetic, confusing cluster, should not be trusted.

a hyperrectangle containing an infinite number of points. Updating this experiment to use SAT logic should be relatively straightforward.

To get a sense for how this synthetic, model-confusing data relates to the empirical data, we fit a t-Distributed Stochastic Neighbor Embedding (t-SNE) [127] to find a reduced dimensionality space where we can characterize how clusters of training, hold out, and confusing data interact with one another. Ideally, there will be no real distinction between the confusing and empirical distributions, which means that the samples we generate with the SAT solver are valid, and that low predictive capabilities on these samples represents a weakness of the model which should prevent us from trusting it in particular subspaces.

Figure D.2 shows the similarities and differences between, train (+), hold out (-), and confusing (o) samples. We used Z3 to generate the 500 confusing queries shown in the projection. The + and - points form indistinguishable clusters which tells us two things: 1) our independent and identically distributed (i.i.d.) assumption holds between train and hold out data and 2) there is a structural difference between voting records of Democrats and Republicans seen in the two distinct +/- clusters. The third

takeaway from Figure D.2 is that our distribution of confusing samples bisects these Democrat and Republican clusters.

There are two plausible explanations for the behavior of the confusing cluster. First, the model seems to perform poorly for voting records associated with Independent Congressional representatives. The model is likely fitting to polarized voting records, so when we generate confusing samples, the model is not confident of how to proceed. The second explanation applies to the most isolated (bottom most) confusing samples - these voting records may not be plausible. Voting records with many abstains or absents will be naturally difficult for any model to predict, regardless of what the model learns.

For both explanations, it is clear that in cases where there are empirical voting records from training or hold out data scattered within the confusing cluster, we should be skeptical about the model output. We can also use this information to conclude that if we see new voting records that look sufficiently similar to the synthetic records in the confusing cluster, then we should either retrain our model or consider making the end-user responsible for the final adjudication.

	Train	**Holdout**	**Confusing**
Train	3.18	3.13	3.03
Holdout		3.09	3.00
Confusing			2.62

Table D.2: Average of Inter/Intra-cluster distances. Smaller number means more similar.

Furthermore, the statistics associated with the confusing sample distribution reveal an interesting property. Table D.2 shows inter and intra-cluster distances for training, holdout, and confusing data distributions. The symmetric matrix shows the average distance of points between clusters. A lower number indicates that the clusters are more similar. The confusing data exhibits the greatest amount of consistency, which makes sense due to the ways in which the SMT solver works; the satisfying assignments are the results of a heuristic search process, which does not exhibit much randomness. Furthermore, the train and holdout data appear more similar to confusing cluster than they do to themselves. All this points to the fact that the confusing data generated by the SAT solver is valid and quite similar to the empirical data in some cases.

Namely, since the assignment values for the confusing samples are a product of randomness, most binary features exhibit a mean=.5 and std=.5. Two features stick out

in particular – *PhysicianFeeFreeze* (mean=.97,std=.15) and *SynfuelCorporationCutbacks* (mean=.76,std=.26) – which lead us to believe that the most common weakness of this trained ensemble model is the combination of a vote for *PhysicianFeeFreeze* and a vote for *SynfuelCorporationCutbacks*. This is analagous to voting to curb spending in healthcare (something indicative of Republicans in the data) and voting to end tax cuts for corporations (something indicative of Democrats in the data). Thus, we are now armed with the information that if we get new data that exhibits this short-hand behavior, we expect that the model's prediction should not be trusted.

Appendix E

Primer for Engaging with Broader Audiences

We wish to provide additional details that are important for understanding the technologies and methods that go into this thesis work. We motivate our work with an example of a similar problem outside of the field of Artificial Intelligence (AI). We also provide a philosophical overview of some concepts in Explainable AI (XAI) and describe how these guided the arc of this work, transforming from a desire to build tools which enable human understanding of AI systems to a push to build tools which increase trustworthiness of the model.

E.1 An uncanny resemblance between the current state of AI and the history of the automobile industry

> The American automobile is produced exclusively to the standards which the manufacturer decides to establish. It comes into the marketplace unchecked. When a car becomes involved in an accident, the entire investigatory, enforcement, and claims apparatus that makes up the post-accident response looks almost invariably to driver failure as the cause. The need to clear the highways rapidly after collisions contributes further to burying the vehicle's role. Should vehicle failure be obvious in some accidents, responsibility is seen in terms of inadequate maintenance by the motorist. Accommodated by superficial standards of accident investigation, the car manufacturers exude presumptions of engineering excellence and reliability, and this reputation is accepted by many unknowing motorists.
>
> *"Unsafe at Any Speed"*. Ralph Nader, 1965

An alternate path to motivating this thesis work is to start with a brief comparison between the current state of artificial intelligence and the history of the automobile industry. There was a time when no safety standards existed for automobiles. Features that we take for granted today, including break-away steering columns, seat belts, and head rests, are modern standards that every vehicle must meet to be certified road-safe. The above excerpt from Ralph Nader's criticism of the 1960s automobile industry bears an uncanny resemblance to contemporary critiques of Artificial Intelligence (AI). In the case of automobile safety, the National Highway and Traffic Safety Administration was established in response to mounting evidence that many vehicle injuries and deaths were preventable. Today, the NHTSA sets industry safety standards that all automobile manufacturers must meet.

The field of Artificial Intelligence is much younger, therefore, the entire story hasn't yet been written. In fact, the general consensus is that it began in the summer of 1956, nearly 50 years after Henry Ford's Model-T. To date, no regulatory body exists to certify the safety, or deployment-worthiness of AI systems. The parallels to pre-NHTSA automobiles are clear and numerous; when trained models are deployed unchecked, unintended harm can be done in unforeseen ways. In the case of the automobile, defining safety standards was difficult due to the reluctance of industry leaders to even acknowledge the problem. Luckily, there to exist a broad consensus that safety is paramount for mission-critical AI systems, but a different hurdle increases the difficulty; it's hard to define what safety means for an inherently probabilistic system.

A core assumption of machine learning is that the best solution or policy we can apply

to a particular domain is one borne of an optimization procedure. This is in line with a famous aphorism, often attributed to statistician George Box; *All models are wrong, but some are useful.* Applying George's logic, and briefly entertaining a pessimistic perspective, we could argue that AI systems are designed to fail in ways that make them useful. We go to great lengths to curb error rates, for good reason, as making a few errors in an effort to learn a generalizable policy is a design choice that most AI developers are ready to make. Models with low error rates or minimal loss are widely accepted as useful models, and this is often deemed sufficient criteria for deployment.

However, AI is being deployed in increasingly consequential domains, and we argue that it is unethical to rely solely on useful models with low error rates when the freedoms of a human being are at stake. The fundamental problem is that error rates alone do not provide details of how errors manifest. Are errors a consequence of anomalous data outliers, which violate the typical i.i.d. assumption of machine learning? Or, are errors the result of systematic deficiencies in a model's decision making? The fact that, to date, there is no good answer to the question, *what did the model learn*, means that it is impossible to answer the question of, *why does the model fail.* It is a tall order to be asked to trust a system when we do not know why it fails.

It is not an objective moral truth that trustworthiness in AI systems is important, however, the decision for whether to prioritize trust can be reduced to the decision for how to allocate power among individuals in decision making processes. Use of AI is ethical when individuals consent to its use. So what does it take to give informed consent in this scenario? Typically, trust is a precursor to consent, in the sense that trust represents a willingness to relinquish one's autonomy to another agent to complete a task. So, in order to consent to the use of an AI system, a human must trust it.

Unfortunately, public trust in AI is sorely lacking, we believe it does not have to be this way. Trusting other technology is commonplace in society today. We argue that the real reason that AI does not enjoy the same level of public trust as other engineered systems is because it is the product of a fundamentally different design process. The engineering design process involves both validation and verification, meaning that a product is specified to sufficiently meet users' needs and the product undergoes verification to ensure it meets those specifications. On the other hand, AI systems are often developed in underspecified environments, meaning that there are many possible solutions that may all learn different structure. Since many desirable design specifications cannot be expressed alongside objective functions, optimization procedures aren't always held to the same constraints that other systems are. AI is often not checked for adherence to properties such as common sense or other engineering desiderata. AI often can exceed very high expectations in terms of making good recommendations, and yet it will also occasionally fail in embarrassing and egregious ways that make stakeholders doubt whether deploying such a system is a good choice. Relatively little effort is spent on characterizing fault modes of a learning model, or on defining operational conditions under which the model exhibits desirable characteristics. Many models are deployed unchecked, fit to superficial standards of reliability without a

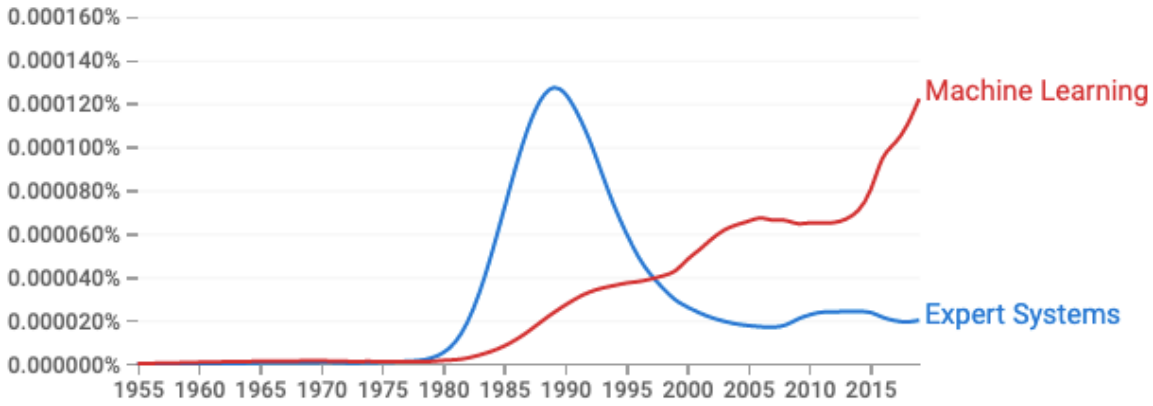


Figure E.1: Google N-gram Viewer [132] comparing written occurrences of 'Expert Systems' and 'Machine Learning' in literature between 1955 and 2019

deeper understanding for *why* a model exhibits particular behaviors.

We believe that one way to address the challenge involves a confluence of logical and statistical techniques. Statistical machine learning is adept at learning useful policies from data, whereas automated reasoning is adept at determining whether a consequent can be reached from a set of antecedents. If the model produced by statistical techniques is checked with formal methods, then we can move toward providing proofs that the model does indeed conform to common sense or other engineering desiderata.

The idea of the importance of combining logical and statistical methods is not new. In 1983, Nils Nilsson, as a former President of AAAI, affirmed the importance of both symbolist and connectionist methods in AI whose proponents were named the 'neats' and the 'scruffies' - 'A dymanic field needs scruffies at its expanding frontier of knowledge, and it needs neats to codify, clarify, and teach its core concepts. A field that is scruffy to the core has simply not yet matured as a science, and one that does not have a scruffy exterior is simply sterile'[143].

Shortly after, AI seemed to shed the symbolic logic approaches in favor for statistical techniques. This phenomenon can be observe in Figure E.1, where the decline of Expert Systems in the early 1990s led the field of Artificial Intelligence to increasingly look toward statistical learning methods and to move away from symbolic logic. With that transition came systems that are less amenable to interrogation by humans, and thus the need to explain models grew. A contemporary subfield of AI, Explainable AI (XAI), has filled the niche by trying to provide explanations and build trust in the ways that humans socially interact with one another. Generally speaking, the drawback of these approaches is that humans are not generally considered adept at providing accurate explanations for their behaviors, thus privileging anthropomorphic strategies for increasing trust and understanding tends to make the strategies and goals of XAI difficult to define quantitatively.

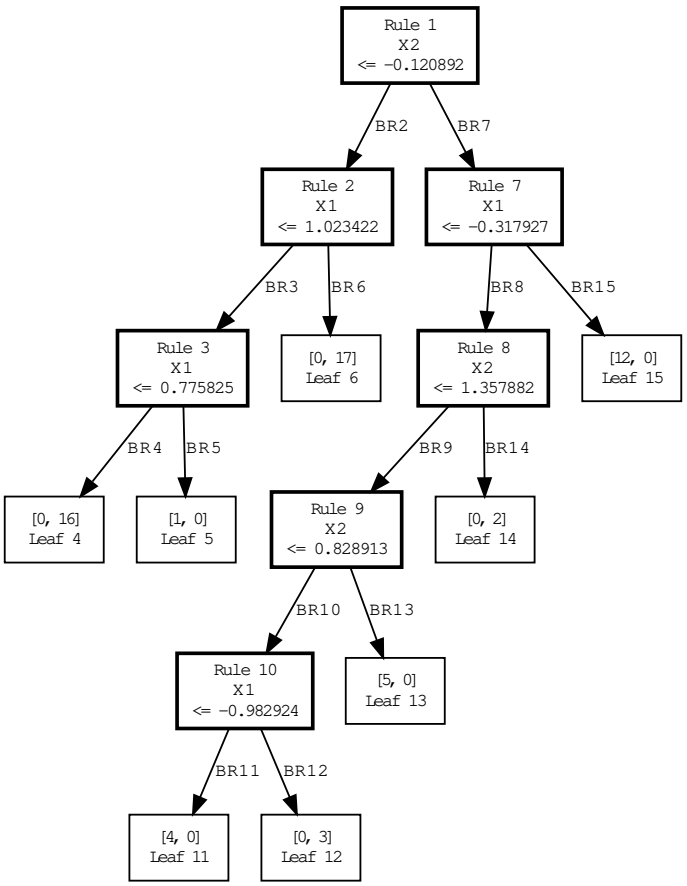
What we argue is novel in our approach, is the combination of statistical and logical

methods for the purpose of increasing trustworthiness of AI systems and providing explanations that are designed for the developers and application-domain experts. While all humans are likely most interested in explanations that are verifiably true, this is especially true for data scientists who build the models and domain experts who featurize the data. We believe the only way to allow a human to assess the epistemic value of an explanation is for them to independently verify the claim. This thesis work aims to put tools that perform this task into the hands of the designers of AI systems, so that they can increase their sense of trust in a model or increase their understanding of how it works. Our approach allows practitioners to answer the question of, "*how do we know we can trust these models*" by providing formal proofs that verify model adherence to any design specifications that are preconditions to their trust. We also show that these proofs can be summarized in ways that explain why an observed model behavior manifests.

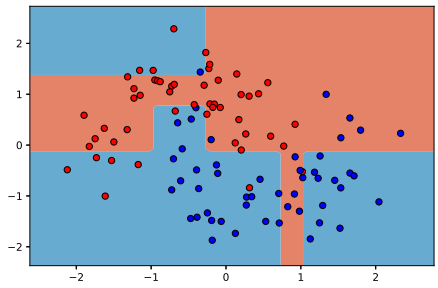
E.2 Preliminaries

The following subsections describe the knowledge necessary to understand and implement the work in this thesis. We start by discussing the details of voting decision trees and voting tree ensembles, which represent the model of choice for verification tasks in this thesis work. We continue by discussing background concepts and knowledge for the design specifications that we ultimately wish to prove about our trained models. Miscellaneous concepts are covered last.

E.2.1 Decision trees and tree ensembles



(a) A decision tree trained on toy data shown in E.2b



(b) Training and Test data and decision boundary described in E.2a. The decision tree predicts red or blue respectively depending on the background color.

Figure E.2: An illustrative example of a decision tree model

Decision Trees are a type of rule-based partitioning model. Unless otherwise specified, we will be considering voting decision trees for classification tasks any time a verification task is discussed. Voting decision trees partition training data into a tree structure by

splitting along input features to greedily optimize some metric, typically entropy or gini index. Leaves of decision trees contain a distribution of class labels corresponding to data observed during training. When a decision tree makes a prediction, the tree is traversed from root to leaf according to the feature values of the input and the mode of that leaf's class label distribution is returned. This winner-take-all prediction strategy is unique to voting decision trees. Non-voting decision trees may report probabilities associated with each class label, however, prediction logic that is probabilistic in nature is less amenable to encoding in symbolic logic, so we focus our efforts on voting decision trees, which yield deterministic outputs.

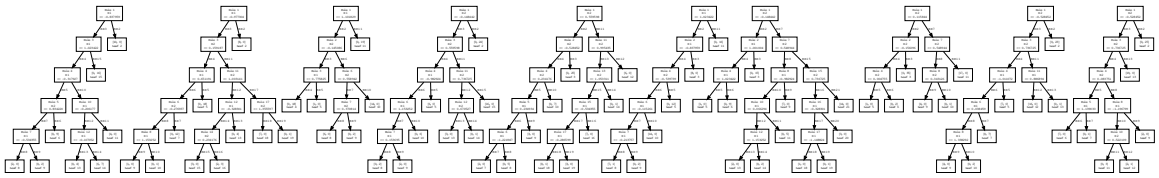
Voting decision trees can handle categorical and real-valued data. The major difference between the two types of data is that categorical data is not necessarily ordinal. Without loss of generality, we can discuss and implement voting decision trees to handle only real-valued data, as high-arity, non-ordinal, categorical attributes can be expressed as a collection of binary, categorical attributes, where enforcing ordinality between two values is irrelevant.

Table 2.1 shows the notation that we use throughout this document when describing decision trees. Each variable corresponds to part of the necessary information stored within the nodes of the decision tree. The information held at branch nodes differs from the information at leaf nodes. Shown in Figure E.2 is a trained decision tree model on the left and training+testing data with the resulting decision boundary shown on the right. For all branch nodes, there exist a right and left child node, a splitting attribute, and a threshold value. While traversing the tree when making a prediction for some input, this information allows us to decide which subtree should be traversed next, depending on the input's attribute values and the threshold values learned by the voting decision tree during the training phase. In the case of the top node (root node) of the tree in Figure E.2a, we can see that the first splitting criteria happens for values on each side of the $x_2 \leq -0.120892$ divide. If that check on the input is true, the left subtree is subsequently checked, else, the right subtree is traversed. Subsequent splitting criteria are applied depending on which side of the root threshold boundary a query lies until a leaf node is reached.

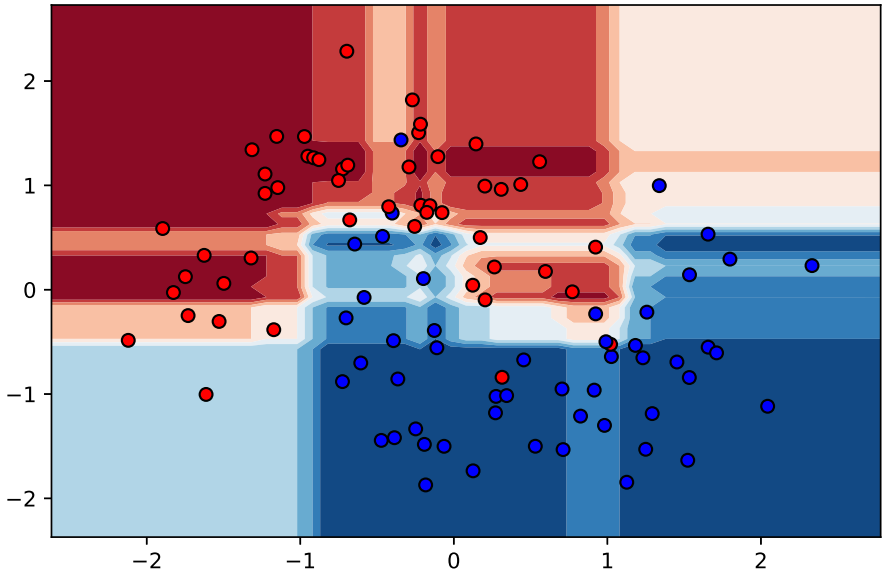
Leaf nodes are nodes without children in the tree, which denote a partition of space described by a conjunction of all splitting criteria between the leaf and the root node. For all leaf nodes, there exists a distribution of class labels that were encountered during the training phase and the index of the mode of that distribution. The model uses this information to produce a deterministic prediction at each leaf. Since Figure E.2a is just a toy example, this tree happens to fit training data perfectly; each leaf nodes label distribution, shown in brackets, only has one non-zero element.

Figure E.2b provides a visual representation of this decision boundary as a backdrop to both the training and testing data in this example. It is easy to find deficiencies in the policy, for example, the upper left blue region covering red data points or the red region [.8,1],[-2,0] that captures many blue points. While decision trees are considered to be intuitive model structures, one of their biggest weaknesses is that they tend to overfit data.

Voting tree ensembles



(a) A 10-tree ensemble trained on toy data shown in E.3b. On a machine, zoom in for further information.



(b) Training data and decision boundary described in E.3a

Figure E.3: An illustrative example of a tree ensemble model

A voting tree ensemble is an ensemble of voting decision trees. Each decision tree casts an unweighted vote for its predicted class label, and the voting tree ensemble outputs the class label that achieves a plural victory among the vote tally for all possible class labels. Ensembles comprised of models with diverse decision logic are less prone to overfitting, which often leads to better predictive performance when compared to the performance of a single decision tree model [50], [67].

Tree Ensembles, shown in E.3, are a common variant of tree ensembles where each decision tree is trained on a random subset of samples and features in data. We notate an ensemble as $\mathcal{M}$, which is comprised of models $m_1, m_2, ..., m_n$. For a set $\mathcal{Y}$ of class labels observed in data, the voting tree ensemble prediction for an input $\mathbf{x}$ is given as:

$$\mathcal{M}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \mathbb{I}\{m_i(\mathbf{x}) = y\} \tag{E.1}$$

Where $\mathbb{I}$ is the indicator function that returns 1 if the inside condition is true, otherwise returns 0.

Figure E.3a shows a collection of voting decision trees that form one tree ensemble. Note that there are ten root nodes visible on the top level, showing the 10 trees that exist in this particular ensemble. Each tree learns different decision logic and varies in depth depending on the subset of data used to train each particular decision tree within the ensemble.

Figure E.3b shows a decision surface (a contour plot) for the tree ensemble shown in E.3a. Dark red/blue background indicates a unanimous consensus among votes cast by individual decision trees in the forest. These regions tend to manifest away from the decision boundary, where strong consensus is harder to achieve. Lighter shades of red/blue indicate that fewer trees make up the majority consensus. White regions indicate a 50-50 split (a 5-5 vote tie in this case) in the vote tally between red and blue class labels.

If there are an even number of trees in the ensemble, ties are possible. A common strategy is to break ties randomly, however, this strategy does not work well for some verification tasks in this work. When necessary, we break ties predetermined order, meaning that the lesser of the two class label indices is ultimately yielded as the ensemble's prediction. Other times, breaking ties aren't actually necessary, in in those cases, we report both class labels that are implicated in the tie. For instance, it may be of interest to characterize the operational ranges over inputs where ties manifest, as this may aid in the understanding of where a trained tree ensemble model performs poorly. Verifying the existence of vote tally ties is something that is possible to do with our framework.

It is worth noting the differences between the learned decision surface in Figure E.3b and Figure E.2b. It is easy to see that the tree ensemble learns a policy that is a higher fidelity approximation of the underlying structure in data, which is identical between the two figures. While the general strategy is for both the voting decision tree and voting tree ensemble to output deterministic class labels, it is possible to output probabilistic class labels at the ensemble level, which correspond to the normalized vote tally for a given input sample. This probabilistic output is shown in Figure E.3b. There does not exist similar probabilistic information within a single voting decision tree, thus, the decision tree has two possible output states, whereas the tree ensemble has up to ten possible output states depending on the two class vote tallies. When necessary, we can collapse that ten-state output to a binary output denoting whether or not a majority exists. In the case of multi-class data, the existence of a majority is replaced by the existence of a plurality, and the number of output states changes from two to the number of classes present in data.

E.2.2 Verification

The SAT Problem

The SAT Problem is a classical computer science problem, aimed at finding inputs to a logic formula so it evaluates as true. In fact, 3-SAT was the first problem to be proven NP-

Complete, and proving the complexity of other problems often involves reduction to 3-SAT. The SAT Problem can be applied to suitable logics, resulting in extensions including the theory of integers, reals, arrays, etc.

Boolean SAT

Given a logic formula, $f_{\mathbb{L}}$, comprised of Boolean literals, $\mathbb{L}$, the Boolean SAT problem is to determine whether there exists a `True/False` interpretation of inputs such that the formula evaluates to `True`.

$$\phi \leftarrow \{\ell_i \in \{0,1\} \mid \forall i \in \mathbb{L}, f_{\mathbb{L}} = 1\} \tag{E.2}$$

If an interpretation, ϕ, exists, the formula is *satisfiable* (`SAT`). If no interpretation exists, the formula is *unsatisfiable* (`UNSAT`).

$$\mathbb{L} \leftarrow \{\ell_1, \ell_2, \ell_3\}$$
$$f_{\mathbb{L}} = \ell_1 \wedge (\ell_2 \vee \neg\ell_3)$$

(a) An illustrative SAT instance

$$\phi_1 \leftarrow \{1,0,0\}$$
$$\phi_2 \leftarrow \{1,1,0\}$$
$$\phi_3 \leftarrow \{1,1,1\}$$

(b) Possible solutions to E.4a

Figure E.4: An illustrative example of the SAT problem. ϕ_1, ϕ_2, and ϕ_3 each represent one possible, valid solution.

Figure E.4a shows a toy example with a simple logical formula composed of three Boolean literals. Figure E.4b shows three possible satisfying assignments for $f_{\mathbb{L}}$, and finding any one of them is considered a solution to the SAT problem.

This work treats the verification of model properties as an instance of theSAT problem. Thus, in order to frame the verification task this way, we need a strategy for expressing a trained learning model as a Boolean logic formula.

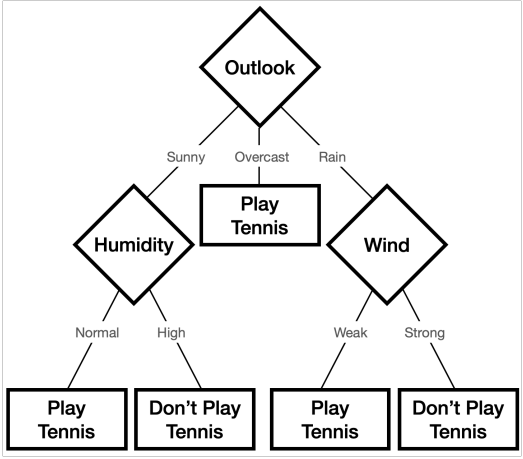
Encoding a partitioning classification model in conjunctive normal form

Since the encoding strategy is too detailed to adequately illustrate on a section walks the reader through the process of encoding a simple decision tree model, first into propositional logic, and then into equivalent CNF. We choose to illustrate the process on the well-known example from Tom Mitchell's *Machine Learning* textbook. The data used to train the model is shown in Table E.5a and the resulting decision tree trained on this data is shown in Figure E.5b.

Figure E.6 shows the statements of propositional logic that encode the decision tree in Figure E.5b. These are broken into three categories. First are the constraints on the

Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	*No*
D2	Sunny	Hot	High	Strong	*No*
D3	Overcast	Hot	High	Weak	*Yes*
D4	Rain	Mild	High	Weak	*Yes*
D5	Rain	Cool	Normal	Weak	*Yes*
D6	Rain	Cool	Normal	Strong	*No*
D7	Overcast	Cool	Normal	Strong	*Yes*
D8	Sunny	Mild	High	Weak	*No*
D9	Sunny	Cool	Normal	Weak	*Yes*
D10	Rain	Mild	Normal	Weak	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*
D14	Rain	Mild	High	Strong	*No*

(a) Tom Mitchell's Tennis Data [137]



(b) Model from [137]

Figure E.5: Tennis Example

$$\textbf{ALO}(O_{overcast}, O_{sun}, O_{rain})$$
$$\textbf{AMO}(O_{overcast}, O_{sun}, O_{rain})$$
$$\textbf{ALO}(T_{hot}, T_{mild}, T_{cold})$$
$$\textbf{AMO}(T_{hot}, T_{mild}, T_{cold})$$
$$\textbf{ALO}(H_{high}, H_{normal})$$
$$\textbf{AMO}(H_{high}, H_{normal})$$
$$\textbf{ALO}(W_{weak}, W_{strong})$$
$$\textbf{AMO}(W_{weak}, W_{strong})$$

(a) Input Feasibility Logic

$$P_1 \iff O_{sunny} \wedge H_{normal}$$
$$P_2 \iff O_{sunny} \wedge H_{high}$$
$$P_3 \iff O_{overcast}$$
$$P_4 \iff O_{rain} \wedge W_{weak}$$
$$P_5 \iff O_{rain} \wedge W_{strong}$$

(b) Decision Logic

$$P_1 \implies Y_{yes}$$
$$P_2 \implies Y_{no}$$
$$P_3 \implies Y_{yes}$$
$$P_4 \implies Y_{yes}$$
$$P_5 \implies Y_{no}$$
$$Y_{yes} \implies P_1 \vee P_3 \vee P_4$$
$$Y_{no} \implies P_2 \vee P_5$$

(c) Output Logic

Figure E.6: Figure E.5b as propositional logic. ALO = At Least One literal must be true. AMO = At Most One literal must be true.

inputs to the model; A single sample cannot have more or less than one value for a given attribute. Both At-Least-One and At-Most-One constraints are added in order to satisfy the constraint that only one of the implicated literals may be set true. For two literals, the At-Least-One clause $(A \vee B)$ will only evaluate true if at least one of $\{A, B\}$ is set true. Likewise, the At-Most-One clause $(\neg A \vee \neg B)$ will only evaluate true if at least one of $\{A, B\}$ is set false.

Second, the tree structure can be broken into 5 paths from root to a leaf node, and the tree may only set one of these paths active for a given input sample. A particular path to a

leaf node is only active if all branches between the leaf and the root node are active, hence, the biconditional statement.

Lastly, there is a prediction made at each leaf node. If a path is active, we know the active leaf node, and consequently, the output class label. Additionally, we constrain that only one of the paths may be active. A conjunction of all these constraints is equivalent to the model in Figure E.5b.

Table E.1: Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic. Table found in [166].

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \implies \beta) \equiv (\neg\beta \implies \neg\alpha)$	contraposition
$(\alpha \implies \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \iff \beta) \equiv ((\alpha \implies \beta) \wedge (\beta \implies \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

All propositional logic formulas can be expressed in conjunctive normal form, which means that the resulting logic formula must be a conjunction of disjunctive clauses. In an effort to keep this document self contained, a table of common transformations is given in Table E.1. The resulting equivalent CNF logic is given in Figure E.7. A conjunction (logical AND) of all the disjunctive clauses in Figure E.7 serve as an equivalent representation of the tree model in Figure E.5b in CNF. The logic in Figure E.7 can be parsed and solved by most modern SAT solvers.

The reason why interrogating a model in this way is useful is that we can prove properties about the encoding of the model. Since propositional logic is a sound and complete formal system, this means that anything we prove about the logical representation of the model is true, and anything that is true about the model itself is provable with our encoding. For instance, it becomes possible to formally prove that it is impossible to not play tennis when it is overcast. While such a proof is indeed trivial for this small model and this low-arity, categorical data, the utility of the framework is seen for more complex models. For example, if we imagine a more complex model for deciding whether to play tennis with real-valued features, we may be interested in proving a property such as, "the model never recommends playing tennis when the temperature is below freezing". Without generating proofs in a formal system, the alternative to showing a model's adherence to the previous

$$\neg O_{overcast} \vee \neg O_{rain}$$
$$\neg O_{overcast} \vee \neg O_{sun}$$
$$\neg O_{rain} \vee \neg O_{sun}$$
$$O_{overcast} \vee O_{rain} \vee O_{sun}$$
$$\neg T_{cool} \vee \neg T_{mild}$$
$$\neg T_{cool} \vee \neg T_{hot}$$
$$\neg T_{mild} \vee \neg T_{hot}$$
$$T_{cool} \vee T_{mild} \vee T_{hot}$$
$$\neg H_{high} \vee \neg H_{norm}$$
$$H_{high} \vee H_{norm}$$
$$\neg W_{weak} \vee \neg W_{strong}$$
$$W_{weak} \vee W_{strong}$$

(a) Input Feasibility CNF

$$\neg P_1 \vee O_{sun}$$
$$\neg P_1 \vee H_{norm}$$
$$P_1 \vee \neg O_{sun} \vee \neg H_{norm}$$
$$\neg P_2 \vee O_{sun}$$
$$\neg P_2 \vee H_{high}$$
$$P_2 \vee \neg O_{sun} \vee \neg H_{high}$$
$$\neg P_3 \vee O_{overcast}$$
$$P_3 \vee \neg O_{overcast}$$
$$\neg P_4 \vee O_{rain}$$
$$\neg P_4 \vee W_{weak}$$
$$P_4 \vee \neg O_{rain} \vee \neg W_{weak}$$
$$\neg P_5 \vee O_{rain}$$
$$\neg P_5 \vee W_{strong}$$
$$P_5 \vee \neg O_{rain} \vee \neg W_{strong}$$

(b) Decision Logic CNF

$$\neg P_1 \vee Y_{yes}$$
$$\neg P_2 \vee Y_{no}$$
$$\neg P_3 \vee Y_{yes}$$
$$\neg P_4 \vee Y_{yes}$$
$$\neg P_5 \vee Y_{no}$$
$$\neg Y_{no} \vee P_2 \vee P_5$$
$$\neg Y_{yes} \vee P_1 \vee P_3 \vee P_4$$

(c) Model Output CNF

Figure E.7: Tennis Model Logic (Fig. E.6) in Conjunctive Normal Form (CNF). Conjunction of all clauses is equivalent to tree in Figure E.5b

temperature specification involves generating copious amounts of data to evaluate with the model, and then giving a probabilistic bound. Moving from probabilistic guarantees to formal guarantees is a big jump that increases the trustworthiness of a given model.

We provide an illustration of how to solve a CNF formula in Figures E.8 and E.9. Figure E.8 show the addition of a few literal assignments as well as the effect that they have on the rest of the CNF formula. Blue literals denote an assignment of `TRUE` whereas orange literals denote an assignment of `FALSE`. These literals pose the question, *why doesn't the model advise 'play' when it is sunny, cool, windy, and humid?*. We know that the model in Figure E.5b will not advise playing tennis under those conditions. Figure E.9 shows the contradiction that is identified via unit propagation from the initial literal assignments. The clause, $\neg Y_{yes} \vee P_1 \vee P_3 \vee P_4$, has all orange literals meaning the assertion is violated.

Figure E.10 shows the resulting Minimal Unsatisfiable Set (MUS) that can be extracted from the contradiction highlighted in Figure E.9. The same violated clause is present in both figures, but the number of other literals and clauses is greatly reduced in the MUS. Removing any one of these constraints from the CNF logic would cause the logical contradiction to resolve, and the model would advise 'play' when it is sunny, cool, windy,

$$
\begin{array}{r}
\\
\neg O_{overcast} \vee \neg O_{rain} \\
\neg O_{overcast} \vee \neg O_{sun} \\
\neg O_{rain} \vee \neg O_{sun} \\
O_{overcast} \vee O_{rain} \vee O_{sun} \\
\neg T_{cool} \vee \neg T_{mild} \\
\neg T_{cool} \vee \neg T_{hot} \\
\neg T_{mild} \vee \neg T_{hot} \\
T_{cool} \vee T_{mild} \vee T_{hot} \\
\neg H_{high} \vee \neg H_{norm} \\
H_{high} \vee H_{norm} \\
\neg W_{weak} \vee \neg W_{strong} \\
W_{weak} \vee W_{strong}
\end{array}
\qquad
\begin{array}{r}
\neg P_1 \vee O_{sun} \\
\neg P_1 \vee H_{norm} \\
P_1 \vee \neg O_{sun} \vee \neg H_{norm} \\
\neg P_2 \vee O_{sun} \\
\neg P_2 \vee H_{high} \\
P_2 \vee \neg O_{sun} \vee \neg H_{high} \\
\neg P_3 \vee O_{overcast} \\
P_3 \vee \neg O_{overcast} \\
\neg P_4 \vee O_{rain} \\
\neg P_4 \vee W_{weak} \\
P_4 \vee \neg O_{rain} \vee \neg W_{weak} \\
\neg P_5 \vee O_{rain} \\
\neg P_5 \vee W_{strong} \\
P_5 \vee \neg O_{rain} \vee \neg W_{strong}
\end{array}
\qquad
\begin{array}{r}
\neg P_1 \vee Y_{yes} \\
\neg P_2 \vee Y_{no} \\
\neg P_3 \vee Y_{yes} \\
\neg P_4 \vee Y_{yes} \\
\neg P_5 \vee Y_{no} \\
\neg Y_{no} \vee P_2 \vee P_5 \\
\neg Y_{yes} \vee P_1 \vee P_3 \vee P_4 \\
\hline
Y_{yes} \\
O_{sunny} \\
T_{cool} \\
W_{strong} \\
H_{high}
\end{array}
$$

Figure E.8: Propagating literal truth assignments from assertions below the line: *Why doesn't the model advise 'play' when its sunny, cool, windy, and humid?*

$$
\begin{array}{r}
\\
\neg O_{overcast} \vee \neg O_{rain} \\
\neg O_{overcast} \vee \neg O_{sun} \\
\neg O_{rain} \vee \neg O_{sun} \\
O_{overcast} \vee O_{rain} \vee O_{sun} \\
\neg T_{cool} \vee \neg T_{mild} \\
\neg T_{cool} \vee \neg T_{hot} \\
\neg T_{mild} \vee \neg T_{hot} \\
T_{cool} \vee T_{mild} \vee T_{hot} \\
\neg H_{high} \vee \neg H_{norm} \\
H_{high} \vee H_{norm} \\
\neg W_{weak} \vee \neg W_{strong} \\
W_{weak} \vee W_{strong}
\end{array}
\qquad
\begin{array}{r}
\neg P_1 \vee O_{sun} \\
\neg P_1 \vee H_{norm} \\
P_1 \vee \neg O_{sun} \vee \neg H_{norm} \\
\neg P_2 \vee O_{sun} \\
\neg P_2 \vee H_{high} \\
P_2 \vee \neg O_{sun} \vee \neg H_{high} \\
\neg P_3 \vee O_{overcast} \\
P_3 \vee \neg O_{overcast} \\
\neg P_4 \vee O_{rain} \\
\neg P_4 \vee W_{weak} \\
P_4 \vee \neg O_{rain} \vee \neg W_{weak} \\
\neg P_5 \vee O_{rain} \\
\neg P_5 \vee W_{strong} \\
P_5 \vee \neg O_{rain} \vee \neg W_{strong}
\end{array}
\qquad
\begin{array}{r}
\neg P_1 \vee Y_{yes} \\
\neg P_2 \vee Y_{no} \\
\neg P_3 \vee Y_{yes} \\
\neg P_4 \vee Y_{yes} \\
\neg P_5 \vee Y_{no} \\
\neg Y_{no} \vee P_2 \vee P_5 \\
\neg Y_{yes} \vee P_1 \vee P_3 \vee P_4 \\
\hline
Y_{yes} \\
O_{sunny} \\
T_{cool} \\
W_{strong} \\
H_{high}
\end{array}
$$

Figure E.9: Identification of a logical contradiction through unit propagation from assertions on inputs. All literals of clause $\neg Y_{yes} \vee P_1 \vee P_3 \vee P_4$ are set false, meaning the formula is not satisfiable.

$$
\begin{aligned}
Y_{yes} \\
O_{sunny} \\
\neg O_{overcast} \vee \neg O_{sunny} \\
\neg O_{rain} \vee \neg O_{sunny} \\
H_{high} \\
\neg H_{normal} \vee \neg H_{high} \\
\neg P_1 \vee H_{normal} \\
\neg P_3 \vee O_{overcast} \\
\neg P_4 \vee O_{rain} \\
\neg Y_{yes} \vee P_1 \vee P_3 \vee P_4
\end{aligned}
$$

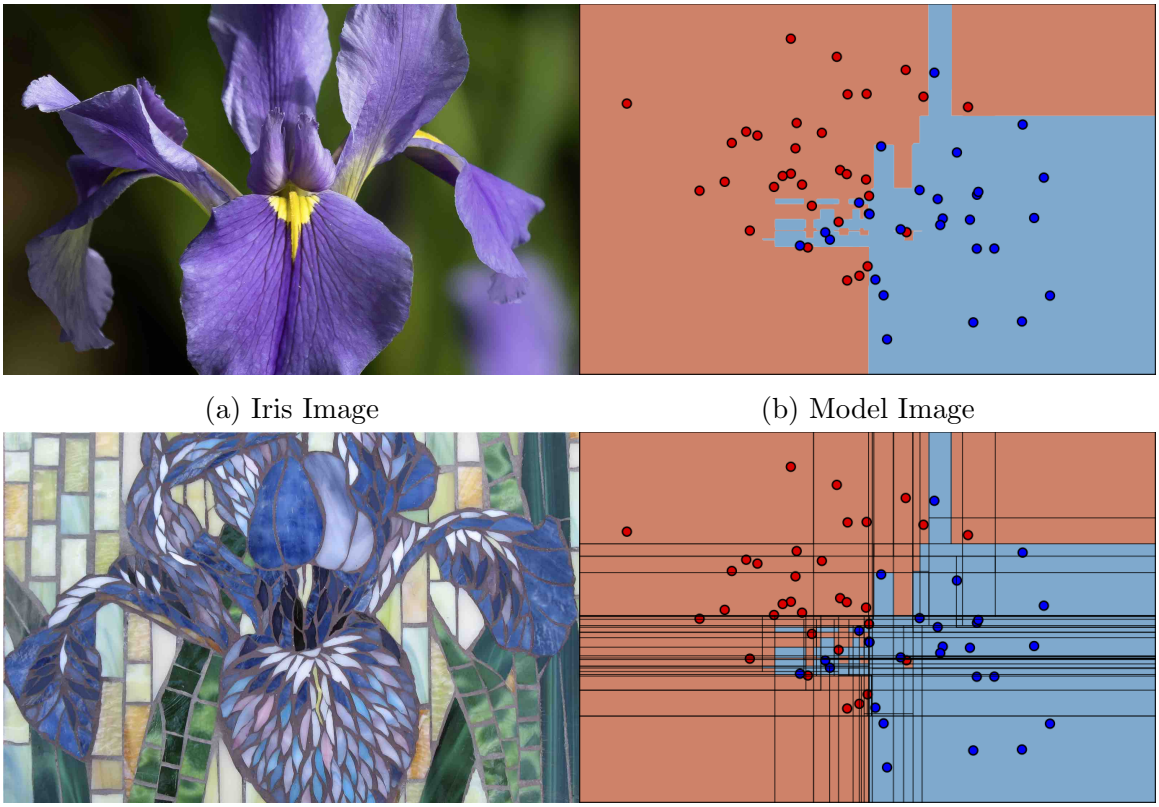
Figure E.10: Minimal Unsatisfiable Set (MUS) for tennis example.

and humid.

Why is a model-centric approach to tree ensemble verification both useful and relatively low cost?

One of the reasons that a model-centric approach to testing the model is useful is because it reduces the total number of states that we need to test. To illustrate, Figure E.11 juxtaposes a trained tree ensemble mode (subplots E.11b and E.11d) with an image of an iris (subplots E.11a and E.11c. A tree ensemble model is an axis-aligned, hyperrectangular partition of $\mathbb{R}^n$ with a finite number of elements. Speaking loosely for a moment, this can be thought of as akin to how Figure E.11a shows an image of an iris with many pixels that all have different values and Figure E.11c is a similar looking mosaic of an iris, where the image is a 'chunkified' version of Figure E.11a with many fewer components. It would be much easier to name the color of each chunk of the mosaic rather than naming the color of every single pixel in the image. This is essentially what we are doing in this thesis work - testing each box (like in subplot E.11d) to make sure that the model does what we want it to do inside each box. When we find counterexamples, that means that the model did something is was not supposed to do inside one box or a group of boxes.

What is most exciting about this insight is that there is no loss in resolution when doing the 'chunkifying' for the tree ensemble model, unlike the image which loses a lot of resolution when approximating the base image with the mosaic. The way that tree ensembles fit to data make them amenable to this type of testing. This structural insight turns the need to test an infinite number of samples into a need to test a finite number of boxes (shown in subplot E.11d to check. If a property is satisfied in each of these finite number of boxes,



(a) Iris Image (b) Model Image

(c) Iris Mosaic Partitions (d) Model Mosaic Partitions

Figure E.11: Illustrative example of a partitioning model.

then we know that the property will be satisfied for any infinite number of inputs to the model.

Solving SAT instances

The work in this thesis does not involve new resolution strategies or search heuristics to change the ways that SAT solvers work. We still wish to include brief details on the inner workings of a SAT solver. There are multiple techniques for solving SAT instances. This section describes the technique that is commonly used in modern SAT solvers, Conflict-Driven Clause Learning (CDCL). To understand how this algorithm works, it is helpful to start with its predecessor, the DPLL algorithm.

DPLL

The Davis-Putnam-Logemann-Loveland (DPLL) algorithm was invented in 1962, and was for a long time, the state-of-the-art for solving SAT problems. It is a search-based technique that evaluates possible combinations of literal assignments, ending the depth-first search

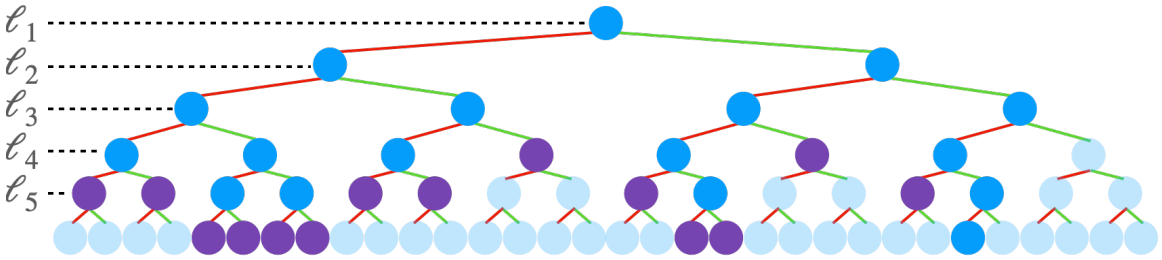


Figure E.12: Illustration of the DPLL Algorithm

once a conflict is identified. While all combinatorial possibilities are not evaluated, all subtrees are searched; as soon as a conflict emerges, DPLL backtraces to check the next unexplored area of the graph.

Conflict-Driven Clause Learning (CDCL)

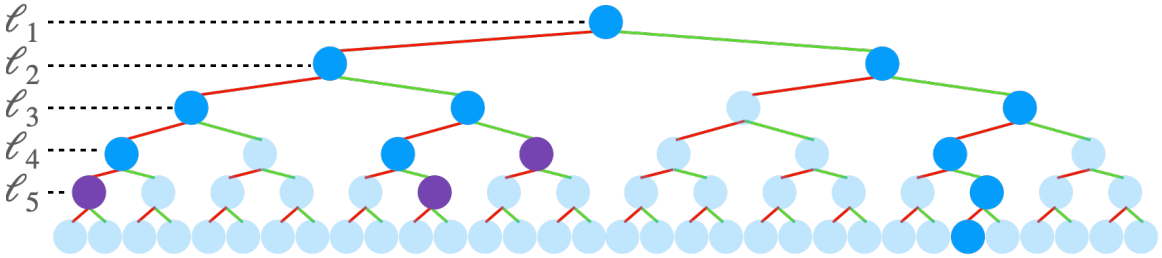


Figure E.13: Illustration of the CDCL Algorithm

Conflict-Driven Clause Learning (CDCL) is another algorithmic approach to solving the satisfiability problem first proposed by [176]. The major difference between DPLL and CDCL is that CDCL adds clauses that cause conflict to the assertion stack. Functionally, this means that the CDCL algorithm allows for non-chronological backtracking, meaning that all subtrees do not need to be evaluated, as is the case for DPLL. The effectiveness of CDCL depends on search heuristics that choose the ordering of variable assignments and what the learned conflict clause looks like. Often, the most general clause that omits the greatest amount of search space is the best candidate conflict clause to add to the assertion stack. Generic CDCL is a sound and complete algorithm [90], which maintains the soundness and completeness of formal systems built on propositional logic. Most modern SAT solvers implement variations of CDCL.

Satisfiability Modulo Theories (SMT)

SMT is an extension of the SAT problem to additional theories such as integers, reals, arrays, bit-vectors, etc. Generally speaking, SMT can be solved using similar strategies as those for SAT. SMT can accommodate more expressive logics, which make encoding learning

models easier. For example, it is inefficient to encode the multiplication of two real-valued variables in propositional logic, but it is possible to do so in SMT. This is why SMT is a go-to strategy for verifying properties about model classes that are not immediately amenable to SAT, such as neural networks.

The drawbacks of SMT can range from negligible to severe. For small problem instances, SMT runs reasonably fast, and expands the classes of models that can be interrogated beyond those possible to interrogate with SAT. However, SMT scales much more poorly than SAT. Verifying models of application-scale is still a challenge. What can be solved in fractions of a second with SAT, may take hours with SMT. Additionally, SMT is a sound, but not a complete formal system. This means that SMT is not guaranteed to halt for a given verification task and may instead return `UNDECIDABLE`

E.3 Philosophical Considerations

The arc of this thesis work started out squarely under the umbrella of XAI, but after trying to really define our goals, it became clear that a shift toward formal verification methods would be the most direct way to achieve our goal of making AI systems more trustworthy. We share some details of our ideas that first got us on the path of considering formal verification.

E.3.1 On the relation between interpretability and intelligibility

Very often, XAI researchers discuss the need to *increase interpretability* of learning models in order for them to be explainable [51, 89, 114, 115, 122, 165, 196]. We believe that some clarity here may help make some of the challenges in the field of XAI more well-defined. We believe that it certainly helped us organize our own thoughts, and helped us land on a methodology which preserves most of the structure present in this viewpoint.

We look back at the etymology of the word *interpretability* and find that it was first coined and used by George Boole in his work, *An Investigation of the Laws of Thought*, in 1854 [21]. He used the term to describe the existence of a mapping between *the operations of the mind by which reasoning is performed* and *their expression in the symbolical language of a calculus*. Adopting this sense of the word as our working definition, it's clear that *interpretability* described the existence of a mapping between two knowledge representations. In Boole's case, this denoted the existence of a translation between math and thought. In the context of XAI, it denotes the existence of a translation between a machine's learned model and human thought.

A similar word that often gets used interchangeably is *intelligibility*. Distinctly different from interpretability, *intelligibility* describes *the extent of the cognitive achievement precipitated with some information or experience*. For instance, it is much less an achievement to recite a memorized fact than to synthesize new knowledge from past experience. Information that is more *intelligible* is easier to for an individual to understand.

It may at first sound trivial to make this distinction, but it actually proved quite useful in the sense that it breaks down the high-level challenges of XAI into to two distinct parts. First, no explanations can be provided if no interpretation between machine knowledge and human knowledge exists. That interpretation can be thought of as the set containing all possible ways to explain a the knowledge a machine possesses. Second, the utility of a particular explanation from that set depends on the extent to which a user can leverage the information to achieve their goals. Establishing interpretability is more straightforward as it is a binary quantity, however, intelligibity requires a scale to describe increasing levels of cognitive achievements. Separating these two tasks notions involved in the communication of explanations to a human acknowledges the ability for a system to provide perfectly valid explanations that may be rejected by the human, due to the fact that the human does not

find the explanation and pertinent knowledge provided to be sufficiently intelligible.

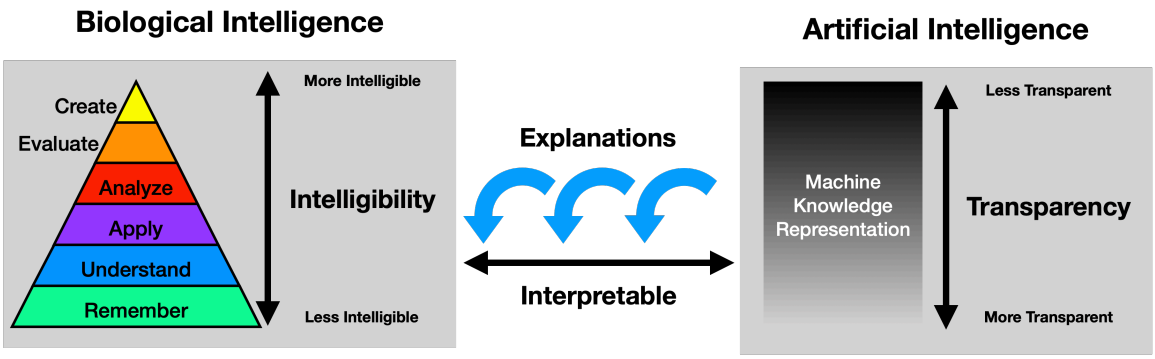


Figure E.14: A view on the relation between notions of interpretability, intelligibility, and transparency

Since we view intelligibility as the extent to which information precipitates a cognitive achievement in a human, we want to identify a way to codify cognitive achievement into levels. This is a question that had been core to research in the field of education for a long time. In 1956, Blooms taxonomy [112] was developed, and it allowed educators of all subjects and grades to define educational objectives with a shared lexicon. The taxonomy is based on the idea that complex cognitive tasks require mastery of more simple cognitive tasks, which gave way to the visualization of the taxonomy as a cognitive pyramid. The levels of the pyramid include remembering, understanding, applying, analyzing, evaluating, and synthesizing. Figure E.14 visualizes this pyramid, along with other keywords in an effort to show how we think about their interaction.

So, what is the purpose of making this distinction in the first place? In the case of XAI research, no matter the methodology used to explain observed behaviors to humans, the goal is usually to help humans understand their models better. It is difficult to measure whether an XAI system achieves this goal without conducting a human study. By organizing our thinking with the framework in Figure E.14, we are trying to make the objective for XAI a bit more quantitative. If we can show that for a given AI system, an interpretation exists between machine knowledge and human knowledge, then we will argue that the system is capable of the type of interrogation that can yield explanations that serve as the functional

units of the interpretation. Said differently, if there is a mapping between biological intelligence and all the elements of a particular machine's intelligence, then interpretability is guaranteed. The challenge then changes focus to the way in which to get a human to achieve their epistemic aims with new information provided by the machine in the form of an explanation for an observed phenomenon.

A skeptic might argue that this only kicks the can down the road. That the real problem of XAI still exists in increasing model intelligibility, and that interpretability has just been twisted such that it can be achieved purely through semantics.

A big part of the challenge in XAI is figuring out what information to leverage to provide to the human end user. By viewing the definition of interpretability in a way that means all possible information must be retrievable, we can finally make meaningful progress towards helping human end users achieve actual goals rather than defining arbitrary sets of information that end users will use to understand models in one particular domain. Framing the problem along two separate dimension allows us to account for different user goals. Perhaps some end users want to have faith that their model is working, while a data scientist may want to know understand the interactions of discrete components inside a black box model. Who is to say that either objective is right, wrong, better, or worse?

As it relates to this thesis, this distinction between the meanings of *interpretability* and *intelligibility* is what got us thinking about logic-based approaches and ultimately landed us on formal verification as a method to achieve the XAI goal of increasing the trustworthiness of the model. A learning model may be encoded in suitable logic, and then proving properties about the model in that abstract mathematical system allows us to provide explanations for model behaviors that may implicate just a few literals or broader interactions between the model and data. Since propositional logic is a sound and complete formal system, we argue that this means that models we verify are interpretable by design.

E.3.2 On the under-specified nature of explanation in XAI

From a quantitative perspective, the metric of success for *explainability* is equally hard to define and evaluate, particularly because it requires the definition intermediate concepts that help humans reason about the model in question. Consider the illustration in Figure E.15, where the goal is to describe the yellow region with fewer than the four parameters that define the box itself, $x_{1,min} \leq x_1 \leq x_{1,max}$ and $x_{2,min} \leq x_2 \leq x_{2,max}$. On the right side of the figure shows possible intermediate concepts which explain the difference between the blue and yellow regions. The cyan, magenta, and dashed circles all provide the same explanation, that inside their respective borders resides the yellow region. However, each explanation is incorrect in different ways.

The cyan circle excludes the corners of the yellow region, whereas the magenta circle includes non-yellow regions. The dashed circle represents the optimal fit of a circle to the square, such that overlap is maximized. While the optimality of the dashed circle is desirable in practice, it makes a poor conceptual explanation because yields both false positives and

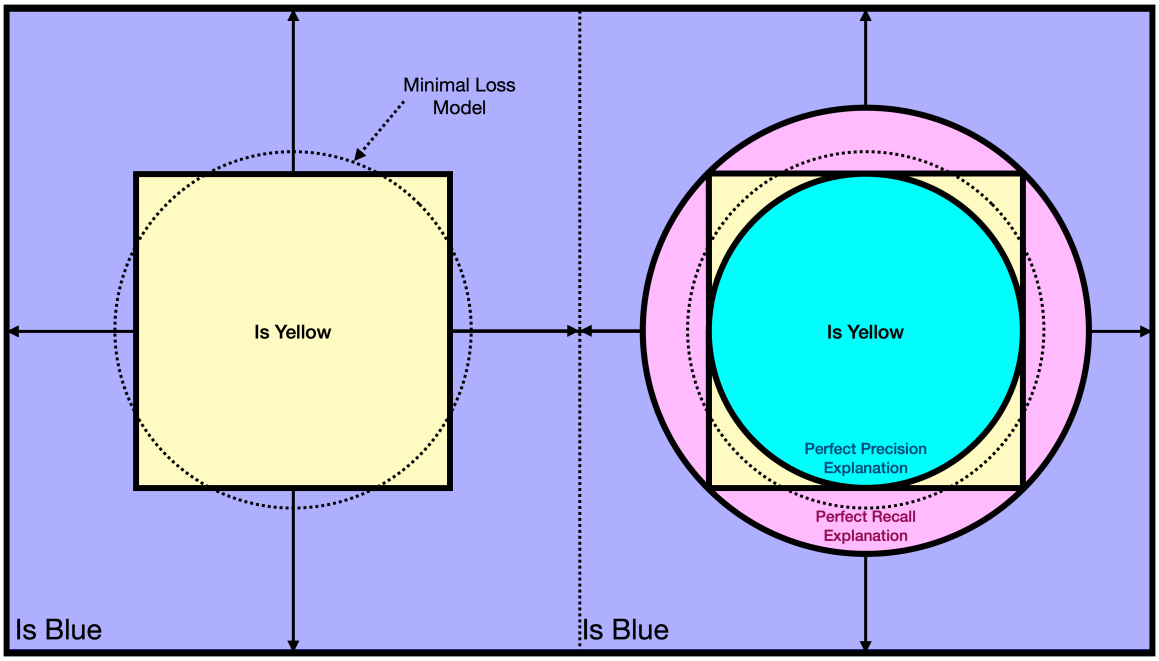


Figure E.15: Defining intermediate concepts to explain an optimal model is ill-defined

false negatives.

A guiding principle for what makes a good explanation should be statements that either achieve perfect precision or perfect recall. These imbue explanations with guarantees that they are either *always hit* in the case of perfect precision, or *never miss* in the case of perfect recall. However, all learning models balance a trade-off between precision and recall, meaning it is inherently unclear whether to pursue a high-precision or high-recall explanation strategy.

This inherent ambiguity motivates a change in focus from understanding models to trusting them. Trust differs from understanding in the sense that it is possible to trust a system without understanding it. Most people trust that elevators will go to the correct floor, however, it seems reasonable to assume that most people do not understand how an elevator works. If understanding of the model remains a goal, then perhaps designing a model with intelligibility in mind would be the best strategy.

E.3.3 On trustworthiness as a byproduct of the design process

Trusting technology is commonplace in society today, however, that trust does not automatically extend to AI systems. For example, I do not know many people who do not trust elevators to eventually bring them to the correct floor.

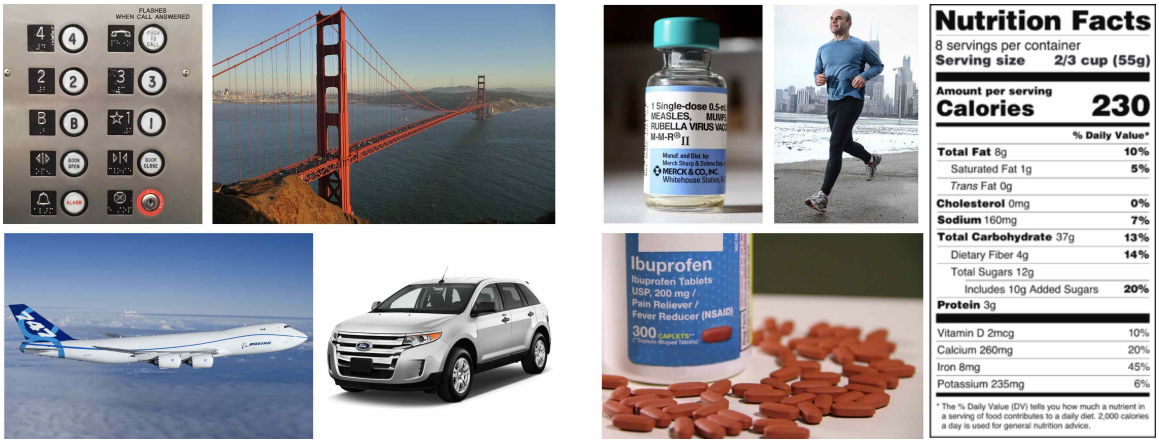


Figure E.16: Examples of trusted technologies that are the product of the engineering design process (left) and the scientific method of inquiry (right).

Different fields have different design processes. The engineering design process involves an up front enumeration of all design specifications that an end product must meet, and ends with validation and verification that all specifications are met. The scientific method involves the statement of a hypothesis and then providing empirical evidence that supports or refutes the statement. The way in which AI is build has elements of both of these processes, but it is distinctly different.

A core tenet of AI is that the best design for a system is one that fails the least frequently. Thus, the fitting of learning models can be expressed as an optimization, where the optimal policy is the one that produces as few errors as possible. While it's possible to make some changes to this process, for instance, that false positives and false negatives can be weighted differently, we still collectively fall short of interrogating trained models for their fault modes. The lack of trust in AI systems is not a result of a lack of trust that the learned policy is indeed optimal, it stems from the fact that optimal usually involves shortcuts, and these shortcuts may involve faults that lead to errors that are unacceptable. Wrong objectives, lack of budget caps, sensor malfunction, poor generalizability, etc. all may go unnoticed when purely considering error rates. Approaching AI as a black box optimization makes it hard to engender the trust of the humans who ultimately will use or rely on the AI system.

With regard to AI systems, it is possible to provide contracts for model behaviors. However, it is impossible to force a person to trust a model, no matter what evidence or contracts we are able to provide. Thus, our goal should be to increase trustworthiness of a model, by certifying whether it adheres to specifications that are of interest to a user. The dangers of AI are not inherent in the technology, but are a result of how humans choose to apply it. Making sure that a model is used in intended, proven-safe ways is a way to mitigate some of the danger of AI. In reducing the chance that an AI system inflicts otherwise easily preventable harm to humans, we make the AI more deserving of trust.

Bibliography

[1] Gul Agha and Karl Palmskog. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 28(1):1–39, 2018.

[2] Sridhar Alla and Suman Kalyan Adari. What is mlops? In *Beginning MLOps with MLFlow*, pages 79–124. Springer, 2021.

[3] Guy Amir, Haoze Wu, Clark Barrett, and Guy Katz. An smt-based approach for verifying binarized neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 203–222. Springer, 2021.

[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. propublica (2016). *URL: https://www. propublica. org/article/machine-bias-risk-asses sments-in-criminal-sentencing*, 2016.

[5] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. The community structure of sat formulas. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 410–423. Springer, 2012.

[6] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. Imperceptible adversarial attacks on tabular data. *arXiv preprint arXiv:1911.03274*, 2019.

[7] Benoît Barbot, Serge Haddad, and Claudine Picaronny. Coupling and importance sampling for statistical model checking. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 331–346. Springer, 2012.

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

[9] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pages 305–343. Springer, 2018.

[10] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In

D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2613–2621. Curran Associates, Inc., 2016.

[11] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2494–2504. Curran Associates, Inc., 2018.

[12] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[13] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep):1089–1105, 2004.

[14] Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. Minimising decision tree size as combinatorial optimisation. In *International Conference on Principles and Practice of Constraint Programming*, pages 173–187. Springer, 2009.

[15] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 405–414, 2018.

[16] Armin Biere. CaDiCaL at the SAT Race 2019. In Marijn Heule, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Race 2019 – Solver and Benchmark Descriptions*, volume B-2019-1 of *Department of Computer Science Series of Publications B*, pages 8–9. University of Helsinki, 2019.

[17] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[18] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[19] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. Interactive weak supervision: Learning useful heuristics for data labeling. *arXiv preprint arXiv:2012.06046*, 2020.

[20] Chris Bogdiukiewicz, Michael Butler, Thai Son Hoang, Martin Paxton, James Snook, Xanthippe Waldron, and Toby Wilkinson. Formal development of policing functions for intelligent systems. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 194–204. IEEE, 2017.

[21] George Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*, volume 2. Walton and Maberly, 1854.

[22] Serena Booth, Christian Muise, and Julie Shah. Evaluating the interpretability of the knowledge compilation map: Communicating logical statements effectively. In *IJCAI*, pages 5801–5807, 2019.

[23] James Bornholt, Emina Torlak, Dan Grossman, and Luis Ceze. Optimizing synthesis with metasketches. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 775–788, 2016.

[24] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[25] Hadrien Bride, Cheng-Hao Cai, Jie Dong, Jin Song Dong, Zhé Hóu, Seyedali Mirjalili, and Jing Sun. Silas: A high-performance machine learning foundation for logical reasoning and verification. *Expert Systems with Applications*, 176:114806, 2021.

[26] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[27] M Cannesson, I Hofer, J Rinehart, C Lee, K Subramaniam, P Baldi, A Dubrawski, and MR Pinsky. Machine learning of physiological waveforms and electronic health record data to predict, diagnose and treat haemodynamic instability in surgical patients: Protocol for a retrospective study. *BMJ Open*, 2019.

[28] Rich Caruana, Hooshang Kangarloo, John David Dionisio, Usha Sinha, and David Johnson. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, page 212. American Medical Informatics Association, 1999.

[29] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.

[30] Hasok Chang. Ontological principles and the intelligibility of epistemic activities. *Scientific understanding: Philosophical perspectives*, pages 64–82, 2009.

[31] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In Kamalika Chaudhuri and Ruslan Salakhutdinov,

editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1122–1131, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[32] Lujie Chen, Artur Dubrawski, Gilles Clermont, T Pellathy, Anthony Wertz, MR Pinsky, and Marilyn Hravnak. Model based estimation of instability severity level in continuously monitored patients. *Intenxive Care Medicine Experimental*, 6(suppl 2):59, 2018.

[33] Lujie Chen, Artur Dubrawski, Donghan Wang, Madalina Fiterau, Mathieu Guillame-Bert, Eliezer Bose, Ata M Kaynar, David J Wallace, Jane Guttendorf, Gilles Clermont, et al. Using supervised machine learning to classify real alerts and artifact in online multi-signal vital sign monitoring data. *Critical care medicine*, 44(7):e456, 2016.

[34] Lujie Chen, T Pellathy, J Yoon, G Clermont, MR Pinsky, M Hravnak, and A Dubrawski. Artificial intelligence assists junior clinicians in assessing risk of severe cardio-respiratory instability in monitored patients. *Intensive Care Medicine Experimental*, 7(Suppl 3):416, 2019.

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

[36] Alexandra Chouldechova and Max G'Sell. Fairer and more accurate, but for whom? *arXiv preprint arXiv:1707.00046*, 2017.

[37] Edmund M Clarke. Model checking. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 54–56. Springer, 1997.

[38] Edmund M Clarke and Paolo Zuliani. Statistical model checking for cyber-physical systems. In *International symposium on automated technology for verification and analysis*, pages 1–12. Springer, 2011.

[39] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[40] Congress. 1984 united states congressional voting records database. In *Congressional Quarterly Almanac*, volume 40. Congressional Quarterly Inc, 1985.

[41] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[42] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

[43] Ashish Darbari, Bernd Fischer, and Joao Marques-Silva. Industrial-strength certified sat solving through verified sat proof checking. In *International Colloquium on Theoretical Aspects of Computing*, pages 260–274. Springer, 2010.

[44] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[45] Maria De-Arteaga, Riccardo Fogliato, and Alexandra Chouldechova. A case for humans-in-the-loop: Decisions in the presence of erroneous algorithmic scores. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[46] Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 120–128, 2019.

[47] Karl de Fine Licht and Jenny de Fine Licht. Artificial intelligence, transparency, and public decision-making. *AI & society*, 35(4):917–926, 2020.

[48] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[49] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[50] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[51] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[52] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[53] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, 2018.

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

[55] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3):94–100, 2016.

[56] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[57] Gil Einziger, Maayan Goldstein, Yaniv Sa'ar, and Itai Segall. Verifying robustness of gradient boosted models. *AAAI Conference on Artificial Intelligence*, 33(1):2446–2453, 2019.

[58] Christian Ellen, Sebastian Gerwinn, and Martin Fränzle. Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains. *International Journal on Software Tools for Technology Transfer*, 17(4):485–504, 2015.

[59] EU. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 directive 95/46/ec (general data protection regulation). *Official Journal of the European Union*, L 119, 2016.

[60] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[61] César Ferri, Peter Flach, and José Hernández-Orallo. Learning decision trees using the area under the roc curve. In *ICML*, volume 2, pages 139–146, 2002.

[62] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[63] Madalina Fiterau and Artur Dubrawski. Informative projection recovery for classification, clustering and regression. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 1, pages 15–20. IEEE, 2013.

[64] Luciano Floridi. Establishing the rules for building trustworthy ai. *Nature Machine Intelligence*, 1(6):261–262, 2019.

[65] Martin Fränzle, Holger Hermanns, and Tino Teige. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 172–186. Springer, 2008.

[66] Laura Freeman, Abdul Rahman, and Feras A Batarseh. Enabling artificial intelligence adoption through assurance. *Social Sciences*, 10(9):322, 2021.

[67] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[68] Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda. Sat competition 2020. *Artificial Intelligence*, page 103572, 2021.

[69] Krishna Gade, Sahin Cem Geyik, Krishnaram Kenthapadi, Varun Mithal, and Ankur Taly. Explainable ai in industry. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3203–3204, 2019.

[70] Ruijiang Gao, Maytal Saar-Tsechansky, Maria De-Arteaga, Ligong Han, Min Kyung Lee, and Matthew Lease. Human-ai collaboration with bandit feedback. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2021.

[71] Sebastian Gerwinn, Eike Möhlmann, and Anja Sieper. Statistical model checking for scenario-based verification of adas. In *Control Strategies for Advanced Driver Assistance Systems and Autonomous Driving Functions*, pages 67–87. Springer, 2019.

[72] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3(11):e745–e750, 2021.

[73] Mahtab Ghazizadeh, John D Lee, and Linda Ng Boyle. Extending the technology acceptance model to assess automation. *Cognition, Technology & Work*, 14(1):39–49, 2012.

[74] Nicholas Gisolfi and Artur Dubrawski. Revealing actionable simplicity in data. In *2018 AAAI Spring Symposium Series*, 2018.

[75] Nicholas Gisolfi, Madalina Fiterau, and Artur Dubrawski. Finding meaningful gaps to guide data acquisition for a radiation adjudication system. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[76] Ella Glikson and Anita Williams Woolley. Human trust in artificial intelligence: Review of empirical research. *Academy of Management Annals*, 2020.

[77] Radu Grosu and Scott A Smolka. Monte carlo model checking. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 271–286. Springer, 2005.

[78] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017.

[79] Aarti Gupta, Malay K Ganai, and Chao Wang. Sat-based verification methods and applications in hardware verification. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 108–143. Springer, 2006.

[80] Ofer Guthmann, Ofer Strichman, and Anna Trostanetski. Minimal unsatisfiable core extraction for smt. In *2016 Formal Methods in Computer-Aided Design*, pages 57–64. IEEE, 2016.

[81] Karimollah Hajian-Tilaki. Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation. *Caspian journal of internal medicine*, 4(2):627, 2013.

[82] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[83] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[84] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[85] Carl G Hempel and Paul Oppenheim. Studies in the logic of explanation. *Philosophy of science*, 15(2):135–175, 1948.

[86] David Henriques, Joao G Martins, Paolo Zuliani, André Platzer, and Edmund M Clarke. Statistical model checking for markov decision processes. In *2012 Ninth international conference on quantitative evaluation of systems*, pages 84–93. IEEE, 2012.

[87] Marijn JH Heule. The drat format and drat-trim checker. *arXiv preprint arXiv:1610.06229*, 2016.

[88] Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human factors*, 57(3):407–434, 2015.

[89] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M Drucker. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 579. ACM, 2019.

[90] Steffen Hölldobler, Norbert Manthey, Tobias Philipp, and Peter Steinke. Generic cdcl-a formalization of modern propositional satisfiability solvers. *POS@ SAT*, 27:89–102, 2014.

[91] Marilyn Hravnak, Lujie Chen, Artur Dubrawski, Eliezer Bose, Gilles Clermont, and Michael R Pinsky. Real alerts and artifact classification in archived multi-signal vital sign monitoring data: implications for mining big data. *Journal of clinical monitoring and computing*, 30(6):875–888, 2016.

[92] Marilyn Hravnak, Lujie Chen, Madalina Fiterau, Artur Dubrawski, Gilles Clermont, Mattieu Guillame-Bert, E Bose, and MR Pinsky. Temporal variation in patient instability in continuously monitored step-down unit patients: Implications for rapid response systems. *Resuscitation*, 89(C):99–105, 2015.

[93] Kevin Hu, Michiel A Bakker, Stephen Li, Tim Kraska, and César Hidalgo. Vizml: A machine learning approach to visualization recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 128. ACM, 2019.

[94] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 3–29, Cham, 2017. Springer International Publishing.

[95] Johan Huysmans, Bart Baesens, and Jan Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. *Available at SSRN 961358*, 2006.

[96] Christina Ilvento. Metric learning for individual fairness. *arXiv preprint arXiv:1906.00250*, 2019.

[97] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international sat solver competitions. *Ai Magazine*, 33(1):89–92, 2012.

[98] Cyrille Jegourel, Axel Legay, and Sean Sedwards. Cross-entropy optimisation of importance sampling parameters for statistical model checking. In *International Conference on Computer Aided Verification*, pages 327–342. Springer, 2012.

[99] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. Verifying individual fairness in machine learning models. In *Conference on Uncertainty in Artificial Intelligence*, pages 749–758. PMLR, 2020.

[100] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer. Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10, Sep. 2016.

[101] Kyle D Julian and Mykel J Kochenderfer. Guaranteeing safety for neural network-based aircraft collision avoidance systems. In *Digital Avionics Systems Conference (DASC)*, 2019.

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

[103] Alex Kantchelian, J. D. Tygar, and Anthony Joseph. Evasion and hardening of tree ensemble classifiers. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of

Proceedings of Machine Learning Research, pages 2387–2396, New York, New York, USA, 20–22 Jun 2016. PMLR.

[104] Amir-Hossein Karimi, Gilles Barthe, Borja Bale, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics*, pages 895–905. PMLR, 2020.

[105] Amir-Hossein Karimi, Gilles Barthe, Borja Belle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. *arXiv preprint arXiv:1905.11190*, 2019.

[106] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[107] Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975.

[108] Lena Kästner, Markus Langer, Veronika Lazar, Astrid Schomäcker, Timo Speith, and Sarah Sterz. On the relation of trust and explainability: Why to engineer for trustworthiness. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 169–175. IEEE, 2021.

[109] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[110] Philip Kitcher. Explanatory unification. *Philosophy of science*, 48(4):507–531, 1981.

[111] Mykel J Kochenderfer and JP Chryssanthacopoulos. Robust airborne collision avoidance through dynamic programming. *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*, 130, 2011.

[112] David R Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002.

[113] Rajeev Kumar and Abhaya Indrayan. Receiver operating characteristic (roc) curve for medical researchers. *Indian pediatrics*, 48(4):277–287, 2011.

[114] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

[115] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models, 2019.

[116] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[117] John D Lee and Katrina A See. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80, 2004.

[118] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *International conference on runtime verification*, pages 122–135. Springer, 2010.

[119] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

[120] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[121] Xinyu Li, Ernest Pokropek, Pinsky Michael, and Dubrawski Artur. Differentiating between hemorrhage and sepsis for hypotensive subjects using arterial pressure data". *American Medical Informatics Association Virtual Informatics Summit*, 2021.

[122] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[123] Chen Liu, Ryota Tomioka, and Volkan Cevher. On certifying non-uniform bounds against adversarial attacks. In *International Conference on Machine Learning*, pages 4072–4081. PMLR, 2019.

[124] Tania Lombrozo. The structure and function of explanations. *Trends in cognitive sciences*, 10(10):464–470, 2006.

[125] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[126] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[127] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[128] David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: improving fairness and accuracy by learning to defer. In *NeurIPS*, pages 6150–6160, 2018.

[129] Charles Marx, Flavio Calmon, and Berk Ustun. Predictive multiplicity in classification. In *International Conference on Machine Learning*, pages 6765–6774. PMLR, 2020.

[130] Roger C Mayer, James H Davis, and F David Schoorman. An integrative model of organizational trust. *Academy of management review*, 20(3):709–734, 1995.

[131] Donna Katzman McClish. Analyzing a portion of the roc curve. *Medical decision making*, 9(3):190–195, 1989.

[132] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182, 2011.

[133] Kyle Miller and Artur Dubrawski. Gamma-ray source detection with small sensors. *IEEE Transactions on Nuclear Science*, 65(4):1047–1058, 2018.

[134] Kyle Miller, Peter Huggins, Simon Labov, Karl Nelson, and Artur Dubrawski. Evaluation of coded aperture radiation detectors using a bayesian approach. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 839:29–38, 2016.

[135] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 2018.

[136] Shira Mitchell, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. Algorithmic fairness: Choices, assumptions, and definitions. *Annual Review of Statistics and Its Application*, 8:141–163, 2021.

[137] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[138] Andrew W Moore and Mary S Lee. Efficient algorithms for minimizing cross validation error. In *Machine Learning Proceedings 1994*, pages 190–198. Elsevier, 1994.

[139] Cecilia Morales, Nicholas Gisolfi, Robert Edman, Kyle Miller, and Artur Dubrawski. Actionable model-centric explanations. In *AAAI*. Student Abstract and Poster Program, 2021.

[140] Remi Munos and Andrew Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49:291–323, 2002.

[141] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, Joao Marques-Silva, and IS RAS. Learning optimal decision trees with sat. In *IJCAI*, pages 1362–1368, 2018.

[142] Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying properties of binarized deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[143] Nils J Nilsson. Artificial intelligence prepares for 2001. *AI Magazine*, 4(4):7–7, 1983.

[144] Mattias Nyberg and Mattias Krysander. Combining ai, fdi, and statistical hypothesis-testing in a framework for diagnosis. *IFAC Proceedings Volumes*, 36(5):813–818, 2003.

[145] Robert M O'Keefe and Daniel E O'Leary. Expert system verification and validation: a survey and tutorial. *Artificial Intelligence Review*, 7(1):3–42, 1993.

[146] Daniel E O'Leary. Verification and validation of intelligent systems: Five years of aaai workshops. *International journal of intelligent systems*, 9(8):653–657, 1994.

[147] Angela Pappagallo, Annalisa Massini, and Enrico Tronci. Monte carlo based statistical model checking of cyber-physical systems: A review. *Information*, 11(12):588, 2020.

[148] Frank Pasquale. Toward a fourth law of robotics: Preserving attribution, responsibility, and explainability in an algorithmic society. *Ohio St. LJ*, 78:1243, 2017.

[149] Paul A Pavlou. Consumer acceptance of electronic commerce: Integrating trust and risk with the technology acceptance model. *International journal of electronic commerce*, 7(3):101–134, 2003.

[150] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[151] Dana Pessach and Erez Shmueli. Algorithmic fairness. *arXiv preprint arXiv:2001.09784*, 2020.

[152] Michael Pinsky, Anthony Wertz, Gilles Clermont, and Artur Dubrawski. Parsimony of hemodynamic monitoring data sufficient for the detection of hemorrhage. *Anesthesia and Analgesia*, 130(5), 2020.

[153] MR Pinsky and Artur Dubrawski. Gleaning knowledge from data in the icu. *Journal of Respiratory Critical Care Medicine*, 190(6):606–610, 2014.

[154] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. Model agnostic supervised local explanations. In *Advances in Neural Information Processing Systems*, pages 2515–2524, 2018.

[155] Luca Pulina and Armando Tacchella. Challenging smt solvers to verify neural networks. *AI Commun.*, 25(2):117–135, April 2012.

[156] Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 33–44, 2020.

[157] Francesco Ranzato and Marco Zanella. Abstract interpretation of decision tree ensemble classifiers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5478–5486, 2020.

[158] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[159] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

[160] Daniël Reijsbergen, Pieter-Tjerk de Boer, Werner Scheinhardt, and Boudewijn Haverkort. On hypothesis testing for statistical model checking. *International journal on software tools for technology transfer*, 17(4):377–395, 2015.

[161] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[162] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[163] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[164] Francesca Rossi. Building trust in artificial intelligence. *Journal of international affairs*, 72(1):127–134, 2018.

[165] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[166] Stuart Russell and Peter Norvig. *Artificial intelligence: a Modern Approach*. Prentice Hall, 2002.

[167] Dominik Sacha, Hansi Senaratne, Bum Chul Kwon, Geoffrey Ellis, and Daniel A Keim. The role of uncertainty, awareness, and trust in visual analytics. *IEEE transactions on visualization and computer graphics*, 22(1):240–249, 2015.

[168] Karin Sanders, Birgit Schyns, Graham Dietz, and Deanne N Den Hartog. Measuring trust inside organisations. *Personnel review*, 2006.

[169] Naoto Sato, Hironobu Kuruma, Yuichiroh Nakagawa, and Hideto Ogawa. Formal verification of a decision-tree ensemble model and detection of its violation ranges. *IEICE Transactions on Information and Systems*, E103.D(2):363–378, Feb 2020.

[170] Cullen Schaffer. Selecting a classification method by cross-validation. *Machine Learning*, 13(1):135–143, 1993.

[171] Karsten Scheibler, Leonore Winterer, Ralf Wimmer, and Bernd Becker. Towards verification of artificial neural networks. *MBMV*, 2015.

[172] Philipp Schmidt, Felix Biessmann, and Timm Teubner. Transparency and trust in artificial intelligence systems. *Journal of Decision Systems*, 29(4):260–278, 2020.

[173] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*, pages 2503–2511, 2015.

[174] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *International Conference on Computer Aided Verification*, pages 202–215. Springer, 2004.

[175] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry. Towards verified artificial intelligence. *arXiv preprint arXiv:1606.08514*, 2016.

[176] João P Marques Silva and Karem A Sakallah. Grasp—a new search algorithm for satisfiability. In *The Best of ICCAD*, pages 73–89. Springer, 2003.

[177] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005.

[178] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming*, CP'05, page 827–831, Berlin, Heidelberg, 2005. Springer-Verlag.

[179] Jennifer Skeem, Nicholas Scurich, and John Monahan. Impact of risk assessment on judges' fairness in sentencing relatively poor defendants. *Law and human behavior*, 44(1):51, 2020.

[180] Megan Stevenson. Assessing risk assessment in action. *Minn. L. Rev.*, 103:303, 2018.

[181] Michael Strevens. The causal and unification approaches to explanation unified—causally. *Noûs*, 38(1):154–176, 2004.

[182] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[183] Ehsan Toreini, Mhairi Aitken, Kovila Coopamootoo, Karen Elliott, Carlos Gonzalez Zelaya, and Aad van Moorsel. The relationship between trust in ai and trustworthy machine learning technologies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 272–283, 2020.

[184] John Törnblom and Simin Nadjm-Tehrani. Formal verification of random forests in safety-critical applications. In Cyrille Artho and Peter Csaba Ölveczky, editors, *Formal Techniques for Safety-Critical Systems*, pages 55–71, Cham, 2019. Springer International Publishing.

[185] John Törnblom and Simin Nadjm-Tehrani. Scaling up memory-efficient formal verification tools for tree ensembles. *arXiv preprint arXiv:2105.02595*, 2021.

[186] Grigori S Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning*, pages 466–483. Springer, 1983.

[187] John Törnblom and Simin Nadjm-Tehrani. Formal verification of input-output mappings of tree ensembles. *Science of Computer Programming*, 194:102450, 2020.

[188] Willem-Jan van den Heuvel and Damian A Tamburri. Model-driven ml-ops for intelligent enterprise applications: vision, approaches and challenges. In *International Symposium on Business Modeling and Software Design*, pages 169–181. Springer, 2020.

[189] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[190] Donghan Wang, Lujie Chen, Madalina Fiterau, Artur Dubrawski, Marilyn Hravnak, E Bose, D Wallace, M Keynar, Gilles Clermont, and MR Pinsky. Multi-tier ground truth elicitation framework with application to artifact clasification for predicting patient instability. *Intensive Care Medicine*, 40[S1]:S289, 2014.

[191] Donghan Wang, Madalina Fiterau, Artur Dubrawski, Marilyn Hravnak, Gilles Clermont, and Michael Pinsky. Interpretable active learning in support of clinical data annotation. *Critical Care Medicine*, 42(12):A1552, 2014.

[192] Tong Wang. Gaining free or low-cost interpretability with interpretable partial substitute. In *International Conference on Machine Learning*, pages 6505–6514. PMLR, 2019.

[193] Daniel S Weld and Gagan Bansal. The challenge of crafting intelligible intelligence. *arXiv preprint arXiv:1803.04263*, 2018.

[194] Anthony Wertz, Gilles Clermont, Artur Dubrawski, and MR Pinsky. Hemodynamic monitoring parsimony: Minimal information for rapid hemorrhage detection. *Intensive Care Medicine Experimental*, 7(suppl3):851, 2019.

[195] Anthony Wertz, AL Holder, Matthieu Guillame-Bert, Gilles Clermont, Artur Dubrawski, and Michael Pinsky. Increasing cardiovascular data sampling frequency and referencing it to baseline improve hemorrhage detection. *Critical Care Explorations*, 1(10), 2019.

[196] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viegas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *arXiv preprint arXiv:1907.04135*, 2019.

[197] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 408–426. Springer, 2018.

[198] Bryan Wilder, Eric Horvitz, and Ece Kamar. Learning to complement humans. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2020.

[199] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z Gajos, Walter S Lasecki, and Neil Heffernan. Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pages 379–388, 2016.

[200] JH Yoon, Vincent Jeanselme, Artur Dubrawski, Marilyn Hravnak, MR Pinsky, and Gilles Clermont. Predicting hypotension episode with numerical vital sign signals in the intensive care unit. *Critical Care*, 23(Supp 2), 2019.

[201] JH Yoon, L Mu, L Chen, A Dubrawski, A Wertz, G Clermont, and MR Pinsky. Predicting tachycardia as a surrogate for instability in the intensive care unit. *Journal of Clinical MOnitoring and Computing*, 33(6):973–985, 2019.

[202] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR, 2013.

[203] Yunfeng Zhang, Q Vera Liao, and Rachel KE Bellamy. Effect of confidence and explanation on accuracy and trust calibration in ai-assisted decision making. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 295–305, 2020.