On the Safety of Interpretable Machine Learning: A Maximum Deviation Approach

Anonymous Author(s)
Affiliation
Address
email

Abstract

Interpretable and explainable machine learning has seen a recent surge of interest. We focus on safety as a key motivation behind the surge and make the relationship between safety and interpretability more quantitative. Toward assessing safety, we introduce the concept of *maximum deviation* via an optimization problem to find the largest deviation of a supervised learning model from a reference model regarded as safe. We then show how interpretability facilitates this safety assessment. For models including decision trees, generalized linear and additive models, the maximum deviation can be computed exactly and efficiently. For tree ensembles, which are not regarded as interpretable, discrete optimization techniques can still provide informative bounds. For a broader class of piecewise Lipschitz functions, we leverage the multi-armed bandit literature to show that interpretability produces tighter (regret) bounds on the maximum deviation. We present case studies, including one on mortgage approval, to illustrate our methods and the insights about models that may be obtained from deviation maximization.

1 Introduction

Interpretable and explainable machine learning (ML) has seen a recent surge of interest because it is viewed as a key pillar in making models trustworthy, with implications on fairness, reliability, and safety [1]. In this paper, we focus on *safety* as a key reason behind the demand for explainability. The motivation of safety has been discussed at a qualitative level by several authors [2–5] and is perhaps clearest in the dichotomy between directly interpretable models vs. post hoc explanations of black-box models. The former have been called "inherently safe" [3] and promoted as the only alternative in high-risk applications [5]. The crux of this argument is that post hoc explanations leave a gap between the explanation and the model producing predictions. Thus, unusual data points may appear to be harmless based on the explanation, but truly cause havoc. These works however have not *quantitatively* addressed the questions: 1) What does safety mean for such models, and 2) how exactly does interpretability aid safety?

Towards answering the first question, we make a conceptual contribution in the form of an optimization problem, intended as a tool for assessing the safety of supervised learning (i.e. predictive) models. Viewing these models as functions mapping an input space to an output space, a key way in which these models can cause harm is through grossly unexpected outputs, corresponding to inputs that are poorly represented in training data. Accordingly, we approach safety assessment for a model by determining where it deviates the most from the output of a *reference model* and by how much (i.e., its *maximum deviation*). The reference model, which represents expected behavior and is deemed to be safe, could be a model well-understood by domain experts or one that has been extensively "tried and tested." The maximization is done over a *certification set*, a large subset of the input space intended to cover all conceivable inputs to the model. These concepts are discussed further in Section 2.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Towards answering the second question, in Section 4 we discuss computation of the maximum deviation for different model classes and show how this is facilitated by interpretability. For model classes regarded as interpretable, including trees, generalized linear and additive models, the maximum deviation can be computed exactly and efficiently by exploiting the model structure. For tree ensembles, which are not regarded as interpretable, discrete optimization techniques can exploit their composition in terms of trees to provide anytime bounds on the maximum deviation. The case of trees is also generalized in a different direction to a broader class of piecewise Lipschitz functions, which we argue cover many popular interpretable functions. Here we show that the benefit of interpretability is significantly tighter regret bounds on the maximum deviation compared with general black-box functions, leveraging results from the multi-armed bandit literature. More broadly, the development of tailored methods for additional model classes is beyond the scope of this first work on the maximum deviation approach (the black-box optimization of Section 4.4 is applicable to all models but obviously not tailored). We discuss in Appendix B.5 some possible approaches and the research gaps to be overcome for neural networks and to make use of post hoc explanations, which approximate a model locally [6–8] or globally [9, 10].

In Section 5, we present case studies that illustrate the deviation maximization methods in Section 4 for decision trees, linear and additive models, and tree ensembles. It is seen that deviation maximization provides insights about models through studying the feature combinations that lead to extreme outputs. These insights can in turn direct further investigation and invite domain expert input. We also quantify how the maximum deviation depends on model complexity and the size of the certification set.

Overall, our discussion provides a more quantitative basis for safety assessment of predictive models and for preferring more interpretable models due to the greater ease of performing this assessment.

2 Assessing Safety Through Maximum Deviation

We are given a supervised learning model f, which is a function mapping an input feature space $\mathcal{X}$ to an output space $\mathcal{Y}$. We wish to assess the safety of this model by finding its largest deviation from a given reference model $f_0 : \mathcal{X} \mapsto \mathcal{Y}$ representing expected behavior. To do this, we additionally require 1) a measure of deviation $D : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$, where $\mathbb{R}_+$ is the set of non-negative reals, and 2) a certification set $\mathcal{C} \subseteq \mathcal{X}$ over which the deviation is maximized. Then the problem to be solved is

$$\max_{x \in \mathcal{C}} \; D(f(x), f_0(x)). \tag{1}$$

The deviation is worst-case because the maximization is over all $x \in \mathcal{C}$; further implications of this are discussed in Appendix C. *Note that* (1) *is different than typical robust training where the focus is to learn a model that minimizes some worst case loss, as opposed to finding regions in $\mathcal{X}$ where two already trained models differ significantly.*

We view problem (1) as a *means* toward the goal of evaluating safety. In particular, a large deviation value is not necessarily indicative of a safety risk, as two models may differ significantly for valid reasons. For example, one model may capture a useful pattern that the other does not. What large deviation values do indicate, however, is a (possibly) sufficient reason for further investigation. Hence, the maximizing solutions in (1) (i.e., the $\arg\max$) are of as much operational interest as the maximum values (this will be illustrated in Section 5).

We now elaborate on elements in problem formulation (1).

Output space $\mathcal{Y}$. In the case of regression, $\mathcal{Y}$ is the set of reals $\mathbb{R}$ or an interval thereof. In the case of binary classification, while $\mathcal{Y}$ could be $\{0, 1\}$ or $\{-1, +1\}$, these limit the possible deviations to binary values as well ("same" or "different"). Thus to provide more informative results, we take $\mathcal{Y}$ to be the space of real-valued scores that are thresholded to produce a binary label. For example, y could be a predicted probability in $[0, 1]$ or a log-odds ratio in $\mathbb{R}$. Similarly for multi-class classification with M classes, $\mathcal{Y} \subset \mathbb{R}^M$ could be a M-dimensional space of real-valued scores. In Appendix A, we discuss considerations in choosing the deviation function D as well as models that abstain.

Reference model f_0. The premise of the reference model is that it should capture expected behavior while being "safe". The simplest case is for f_0 to be a constant function representing a baseline value, for example zero or a mean prediction. We consider the more general case where f_0 may vary with x. For example, f_0 may be a model validated against domain knowledge or constrained by it (or might not arise from ML at all). Such models are prevalent in high-stakes domains where interpretability is

critical. Examples are consumer credit scoring algorithms, medical risk and diagnostic scores, and decision trees for predicting semiconductor yield. The reference model could also be one that is not necessarily informed by domain knowledge but has been extensively tested and deployed so that we regard it as safe. In this case, f could be a new version of the model, trained on more recent data or improved in some fashion, and we wish to evaluate its safety before deploying it in place of f_0.

Certification set $\mathcal{C}$. The premise of the certification set is that it contains all inputs that the model might conceivably be exposed to. This may include inputs that are highly improbable but not physically or logically impossible (for example, a severely hypothermic human body temperature of 27°C). Thus, while $\mathcal{C}$ might be based on the support set of a probability distribution or data sample, it does not depend on the likelihood of points within the support. The set $\mathcal{C}$ may also be a strict superset of the training data domain. For example, a model may have been trained on data for males, and we would now like to determine its worst-case behavior on an unseen population of females.

For tabular or lower-dimensional data, $\mathcal{C}$ might be the entire input space $\mathcal{X}$. For non-tabular or higher-dimensional data, the choice $\mathcal{C} = \mathcal{X}$ may be too unrepresentative because the manifold of realistic inputs is lower in dimension. In this case, if we have a dataset $\{x_i\}_{i=1}^n$, one possibility is to use a union of ℓ_p balls centered at x_i,

$$\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{B}_r^p[x_i], \qquad \mathcal{B}_r^p[x_i] = \{x \in \mathcal{X} : \|x - x_i\|_p \leq r\}. \tag{2}$$

The set $\mathcal{C}$ is thus comprised of points somewhat close to the n observed examples x_i, but the radius r does not have to be "small".

3 Related Work

In previous work on safety and interpretability in ML, the authors of [3, 11] give qualitative accounts suggesting that directly interpretable models are an inherently safe design because humans can inspect them to find spurious elements; in this paper, we attempt to make those qualitative suggestions more quantitative and automate some of the human inspection. Furthermore, several other authors have highlighted safety as a goal for interpretability [2, 4, 12, 13, 5], but again without quantitative development. Moreover, the lack of consensus on how to measure interpretability motivates the relationship that we explore between interpretability and the ease of evaluating safety.

In the area of ML verification, robustness certification methods aim to provide guarantees that the classification remains constant within a radius ϵ of an input point, while output reachability is concerned with characterizing the set of outputs corresponding to a region of inputs [14]. A major difference in our work is that we consider *two* models, a model f to be assessed and a reference f_0, whereas the above notions of robustness and reachability involve a single model. Another important difference is that our focus is *global*, over a comprehensive set $\mathcal{C}$, rather than local to small neighborhoods around input points; a local focus is especially true of neural network verification [15–24]. We also study the role of interpretability in safety verification. Works in robust optimization applied to ML minimize the worst-case probability of error, but this worst case is over parameters of f rather than values of x [25]. Thomas et al. [26] present a framework where during model training, a set of safety tests is specified by the model designer in order to accept or reject the possible solution.

We build on related literature on robustness and explainability that deals specifically with tree ensembles. Mixed-integer programming (MIP) and discrete optimization have been proposed to find the smallest input perturbation to 'evade' a classifier [27] and to obtain counterfactual explanations [28]. MIP approaches are computationally intensive however. To address this Chen et al. [29] introduce graph based approaches for verification on trees. Their central idea, which we use, is to discretize verification computations onto a graph constructed from the way leaves intersect. The verification problem is transformed to finding all maximum cliques. Devos et al. [30] expand on this idea by providing anytime bounds by probing unexplored nodes.

Safety has become a critical issue in reinforcement learning (RL) with multiple works focusing on making RL policies safe [31–34]. There are two broad themes [35]: (i) a safe and verifiable policy is learned at the outset by enforcing certain constraints, and (ii) post hoc methods are used to identify bad regimes or failure points of an existing policy. Our current proposal is complementary to these works as we focus on the supervised learning setup viewed from the lens of interpretability. Nonetheless, ramifications of our work in the RL context are briefly discussed in Appendix C.

4 Deviation Maximization for Specific Model Classes

In this section, we discuss approaches to computing the maximum deviation (1) for f belonging to various model classes. We show the benefit of interpretable model structure in different guises. Exact and efficient computation is possible for decision trees, and generalized linear and additive models in Sections 4.1 and 4.2. In Section 4.3, the composition of tree ensembles in terms of trees allows discrete optimization methods to provide anytime bounds. For a general class of piecewise Lipschitz functions in Section 4.4, the application of multi-arm bandit results yields tighter regret bounds on the maximum deviation. While some of the results in this section may be less surprising, one of our contributions is to identify precise properties that allow them to hold. We also show that intuitive measures of model complexity, such as the number of leaves or pieces or smoothness of functions, have an additional interpretation in terms of the complexity of maximizing deviation. More broadly, the development of methods specific to additional model classes is beyond the scope of a single work. We discuss in Appendix B.5 possible approaches and the advances needed for neural networks (beyond applying the black-box methods of Section 4.4) and to make use of post hoc explanations.

To develop mathematical results and efficient algorithms, we will sometimes assume that the reference model f_0 is from the same class as f. In Appendix C, we discuss the case where f_0 may not be globally interpretable, but may be so in local regions. We will also sometimes assume that the certification set $\mathcal{C}$ and other sets are Cartesian products. This means that $\mathcal{C} = \prod_{j=1}^{d} \mathcal{C}_j$, where for a continuous feature j, $\mathcal{C}_j = [\underline{X}_j, \overline{X}_j]$ is an interval, and for a categorical feature j, $\mathcal{C}_j$ is a set of categories. We mention relaxations of the Cartesian product assumption in Appendix B.2.

4.1 Trees

We begin with the case where f and f_0 are both decision trees. A decision tree with L leaves partitions the input space $\mathcal{X}$ into L corresponding parts, which we also refer to as 'leaves'. We consider only non-oblique trees. In this case, each leaf is described by a conjunction of conditions on individual features and is therefore a Cartesian product as defined above. With $\mathcal{L}_l \subset \mathcal{X}$ denoting the lth leaf and $y_l \in \mathcal{Y}$ the output value assigned to it, tree f is described by the function

$$f(x) = y_l \quad \text{if } x \in \mathcal{L}_l, \quad l = 1, \ldots, L, \tag{3}$$

and similarly for tree f_0 with leaves $\mathcal{L}_{0m}$ and outputs y_{0m}, $m = 1, \ldots, L_0$. As discussed in [36, 37], *rule lists* where each rule is a condition on a single feature are one-sided trees in the above sense.

The partitioning of $\mathcal{X}$ by decision trees and their piecewise-constant nature simplify the computation of the maximum deviation (1). Specifically, the maximization can be restricted to pairs of leaves (l, m) for which the intersection $\mathcal{L}_l \cap \mathcal{L}_{0m} \cap \mathcal{C}$ is non-empty. The intersection of two leaves $\mathcal{L}_l \cap \mathcal{L}_{0m}$ is another Cartesian product, and we assume that it is tractable to determine whether $\mathcal{C}$ intersects a given Cartesian product (see examples in Appendix B.1).

For visual representation and later use in Section 4.3, it is useful to define a bipartite graph, with L nodes representing the leaves $\mathcal{L}_l$ of f on one side and L_0 nodes representing the leaves $\mathcal{L}_{0m}$ of f_0 on the other. Define the edge set $\mathcal{E} = \{(l, m) : \mathcal{L}_l \cap \mathcal{L}_{0m} \cap \mathcal{C} \neq \emptyset\}$; clearly $|\mathcal{E}| \leq L_0 L$. Then

$$\max_{x \in \mathcal{C}} D(f(x), f_0(x)) = \max_{(l,m) \in \mathcal{E}} D(y_l, y_{0m}). \tag{4}$$

We summarize the complexity of deviation maximization for decision trees as follows. This is a slight refinement of [29, Thm. 1] in the case $K = 2$, see Appendix B.1 for details.

Proposition 1. *Let f and f_0 be decision trees as in* (3) *with L and L_0 leaves respectively, and $\mathcal{E}$ be the bipartite edge set of leaf intersections defined above. Then the maximum deviation* (1) *can be computed with $|\mathcal{E}|$ evaluations as shown in* (4).

4.2 Linear and additive models

In this subsection, we assume that f is a generalized additive model (GAM) given by

$$f(x) = g^{-1}\left(\sum_{j=1}^{d} f_j(x_j)\right), \tag{5}$$

where each f_j is an arbitrary function of feature x_j. In the case where $f_j(x_j) = w_j x_j$ for all continuous features x_j, where w_j is a real coefficient, (5) is a generalized linear model (GLM). We discuss the treatment of categorical features in Appendix B.2. The invertible link function $g : \mathbb{R} \mapsto \mathbb{R}$ is furthermore assumed to be monotonically increasing. This assumption is satisfied by common GAM link functions: identity, logit ($g(y) = \log(y/(1-y))$), and logarithmic.

Equation (5) implies that $\mathcal{Y} \subset \mathbb{R}$ and the deviation $D(y, y_0)$ is a function of two scalars y and y_0. For this scalar case, we make the following intuitively reasonable assumption throughout the subsection.

Assumption 1. For $y, y_0 \in \mathcal{Y} \subseteq \mathbb{R}$, 1) $D(y, y_0) = 0$ whenever $y = y_0$; 2) $D(y, y_0)$ is monotonically non-decreasing in y for $y \geq y_0$ and non-increasing in y for $y \leq y_0$.

Our approach is to exploit the additive form of (5) by reducing problem (1) to the optimization

$$M_{\pm}(f, \mathcal{S}) = \max_{x \in \mathcal{S}} / \min \sum_{j=1}^{d} f_j(x_j), \tag{6}$$

for different choices of $\mathcal{S} \subset \mathcal{X}$ and where $+$ corresponds to $\max$ and $-$ to $\min$. We discuss below how this can be done for two types of reference model f_0: decision tree (which includes the constant case $L_0 = 1$) and additive. For the first case, we prove the following result in Appendix B.2:

Proposition 2. *Let f be a GAM as in* (5) *and $\mathcal{S}$ be a subset of $\mathcal{X}$ where $f_0(x) \equiv y_0$ is constant. Then if Assumption 1 holds,*

$$\max_{x \in \mathcal{S}} D(f(x), f_0(x)) = \max_{\sigma \in \{+,-\}} D\left(g^{-1}(M_\sigma(f, \mathcal{S})), y_0\right).$$

Tree-structured f_0. Since f_0 is piecewise constant over its leaves $\mathcal{L}_{0m}$, $m = 1, \ldots, L_0$, we take $\mathcal{S}$ to be the intersection of $\mathcal{C}$ with each $\mathcal{L}_{0m}$ in turn and apply Proposition 2. The overall maximum is then obtained as the maximum over the leaves,

$$\max_{x \in \mathcal{C}} D(f(x), f_0(x)) = \max_{m=1,\ldots,L_0} \max_{\sigma \in \{+,-\}} D\left(g^{-1}(M_\sigma(f, \mathcal{L}_{0m} \cap \mathcal{C})), y_{0m}\right). \tag{7}$$

This reduces (1) to solving $2L_0$ instances of (6).

Additive f_0. For this case, we make the additional assumption that the link function g in (5) is the identity function, as well as Assumption 2 below. The implication of these assumptions is discussed in Appendix B.2.

Assumption 2. $D(y, y_0) = D(y - y_0)$ is a function only of the difference $y - y_0$.

Then $f_0(x) = \sum_{j=1}^{d} f_{0j}(x_j)$ and the difference $f(x) - f_0(x)$ is also additive. Using Assumptions 2, 1 and a similar argument as in the proof of Proposition 2, the maximum deviation is again obtained by maximizing and minimizing an additive function, resulting in two instances of (6) with $\mathcal{S} = \mathcal{C}$:

$$\max_{x \in \mathcal{C}} D(f(x), f_0(x)) = \max_{\sigma \in \{+,-\}} D\big(M_\sigma(f - f_0, \mathcal{C})\big).$$

Computational complexity of (6)**.** For the case of nonlinear additive f, we additionally assume that $\mathcal{C}$ is a Cartesian product. It follows that $\mathcal{S} = \prod_{j=1}^{d} \mathcal{S}_j$ is a Cartesian product (see Appendix B.2 for the brief justification) and (6) separates into one-dimensional optimizations over $\mathcal{S}_j$,

$$\max_{x \in \mathcal{S}} / \min \sum_{j=1}^{d} f_j(x_j) = \sum_{j=1}^{d} \max_{x_j \in \mathcal{S}_j} / \min f_j(x_j). \tag{8}$$

The computational complexity of (8) is thus $\sum_{j=1}^{d} C_j$, where C_j is the complexity of the jth one-dimensional optimization. We discuss different cases of C_j in Appendix B.2; the important point is that the overall complexity is linear in d.

In the GLM case where $\sum_{j=1}^{d} f_j(x_j) = w^T x$, problem (6) is simpler and it is less important that $\mathcal{C}$ be a Cartesian product. In particular, if $\mathcal{C}$ is a convex set, so too is $\mathcal{S}$ (again see Appendix B.2 for justification). Hence (6) is a convex optimization problem.

4.3 Tree ensembles

We now extend the idea used for single decision trees in Section 4.1 to tree ensembles. This class covers several popular methods such as Random Forests and Gradient Boosted Trees. It can also cover *rule* ensembles [38, 39] as a special case, as explained in Appendix B.3. We assume f is a tree ensemble consisting of K trees and f_0 is a single decision tree. Let $\mathcal{L}_{l_k}$ denote the lth leaf of the kth tree in f for $l = 1, \ldots, L_k$, and $\mathcal{L}_{0m}$ be the mth leaf f_0, for $m = 1, \ldots, L_0$. Correspondingly let y_{l_k} and y_{0m} denote the prediction values associated with each leaf.

Define a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where there is a vertex for each leaf in f and f_0, i.e.

$$\mathcal{V} = \{l_k | \forall k = 1, \ldots, K,\, l = 1, \ldots, L_k\} \cup \{m | m = 1, \ldots, L_0\}. \tag{9}$$

Construct an edge for each overlapping pair of leaves in $\mathcal{V}$, i.e.

$$\mathcal{E} = \{(i, j) | \mathcal{L}_i \cap \mathcal{L}_j \neq \emptyset,\, \forall (i, j) \in V,\, i \neq j\}. \tag{10}$$

This graph is a $K + 1$-partite graph as leaves within an individual tree do not intersect and are an independent set. Denote M to be the adjacency matrix of $\mathcal{G}$. Following Chen et al. [29], a maximum clique S of size $K + 1$ on such a graph provides a discrete region in the feature space with a computable deviation. A clique is a subset of nodes all connected to each other; a maximum clique is one that cannot be expanded further by adding a node. The model predictions y_c and y_{0c} can be ensembled from leaves in S. Denote by $D(S)$ the deviation computed from the clique S. Maximizing over all such cliques solves (1). However, complete enumeration is expensive, so informative bounds, either using the merge procedure in Chen et al. [29] or the heuristic function in Devos et al. [30] can be used. We use the latter which exploits the $K + 1$-partite structure of $\mathcal{G}$.

Specifically, we adapt the anytime bounds of Devos et al. [30] as follows. At each step of the enumeration procedure, an intermediate clique S contains selected leaves from trees in $[1, \ldots, k]$ and unexplored trees in $[k + 1, \ldots, K + 1]$. For each unexplored tree, we select a valid candidate leaf that maximizes deviation, i.e.

$$v_k = \underset{l_k,\, l_k \cap i \neq \emptyset,\, \forall i \in S}{\arg\max} D(S \cup l_k). \tag{11}$$

Using these worst-case leaves, a heuristic function

$$H(S) = D(S') = D(S \bigcup_{m=k+1}^{K+1} v_m) \tag{12}$$

provides an upper (dual) bound. In practice, this dual bound is tight and therefore very useful during the search procedure to prune the search space. Each $K + 1$ clique provides a primal bound, so the search can be terminated early before examining all trees if the dual bound is less than the primal bound. We adapt the search procedure of Mirghorbani and Krokhmal [40] to include the pruning arguments. Appendix B.3 presents the full algorithm. Starting with an empty clique, the procedure adds a single node from each tree to create an intermediate clique. If the size of the clique is $K + 1$ the primal bound is updated. Otherwise, the dual bound is computed. A node compatibility vector is used to keep track of all feasible additions.When the search is terminated at any step, the maximum deviation is bounded by (D_{lb}, D_{ub}).

The algorithm works for the entire feature space. When the certification set $\mathcal{C}$ is a union of balls as in (2), some additional considerations are needed. First, we can disregard leaves that do not intersect with $\mathcal{C}$ during the graph construction phase. A validation step to ensure that the leaves of a clique all intersect with the same ball in $\mathcal{C}$ is also needed.

4.4 Piecewise Lipschitz Functions

We saw the benefits of having specific (deterministic) interpretable functions as well as their extensions in the context of safety. Now consider a richer class of functions that may also be randomized with finite variance. In this case let f and f_0 denote the mean values of the learned and reference functions respectively. We consider the case where each function is either interpretable or black box, where the latter implies that query access is the only realistic way of probing the model. This leads to three cases where either both functions are black box or interpretable, or one is black box. What we care

about in all these cases[1] is to find the maximum (and minimum) of a function $\Delta(x) = f(x) - f_0(x)$. Let us consider finding only the maximum of Δ as the other case is symmetric. Given that f and f_0 can be random functions Δ is also a random function and if Δ is black box a standard way to optimize it is either using Bayesian Optimization (BO) [41] or tree search type bandit methods [42, 43].We repurpose some of the results from this latter literature in our context showcasing the benefit of interpretability from a safety standpoint. To do this we first define relevant terms.

Definition 1 (Simple Regret [42])**.** If $f^*_{\mathcal{C}}$ denotes the optimal value of the function f on the certification set $\mathcal{C}$, then the simple regret $r^{\mathcal{C}}_q$ after querying the f function q times and obtaining a solution x_q is given by, $r^{\mathcal{C}}_q(f) = f^*_{\mathcal{C}} - f(x_q)$.

Definition 2 (Order β c-Lipschitz)**.** Given a (normalized) metric ℓ a function f is c-Lipschitz continuous of order $\beta > 0$ if for any two inputs x, y and for $c > 0$ we have, $|f(x) - f(y)| \leq c \cdot \ell(x, y)^{\beta}$.

Definition 3 (Near optimality dimension [42])**.** If $\mathcal{N}(\mathcal{C}, \ell, \epsilon)$ is the maximum number of ϵ radius balls one can fit in $\mathcal{C}$ given the metric ℓ and $\mathcal{C}_\epsilon = \{x \in \mathcal{C} | f(x) \geq f^*_{\mathcal{C}} - \epsilon\}$, then for $c > 0$ the c-near optimality dimension is given by, $\upsilon = \max\left(\limsup_{\epsilon \to 0} \frac{\ln \mathcal{N}(\mathcal{C}_{c\epsilon}, \ell, \epsilon)}{\ln(\epsilon^{-1})}, 0\right)$.

Intuitively, simple regret measures the deviation between our current best and the optimal solution. The Lipschitz condition bounds the rate of change of the function. Near optimality dimension measures the set size for which the function has close to optimal values. The lower the value of υ, the easier it is to find the optimum. We now define what it means to have an interpretable function.

Assumption 3 (Characterizing an Interpretable Function)**.** If a function f is interpretable, then we can (easily) find $1 \leq m \ll n$ partitions $\{\mathcal{C}^{(1)}, ..., \mathcal{C}^{(m)}\}$ of the certification set $\mathcal{C}$ such that the function $f^{(i)} = \{f(x) | x \in \mathcal{C}^{(i)}\}\ \forall i \in \{1, ..., m\}$ in each partition is c-Lipschitz of order β.

Note that the (interpretable) function overall does not have to be c-Lipschitz of bounded order, rather only in the partitions. This assumption is motivated by observing different interpretable functions. For example, in the case of decision trees the m partitions could be its leaves, where typically the function is a constant in each leaf ($c = 0$). For rule lists as well a fixed prediction is usually made by each rule. For a linear function one could consider the entire input space (i.e. $m = 1$), where for bounded slope α the function would also satisfy our assumption ($c = \alpha$ and $\beta = 1$). Examples of models that are not piecewise constant or globally Lipschitz are oblique decision trees (Murthy et al., 1994), regression trees with linear functions in the leaves, and functional trees. Moreover, m is likely to be small so that the overall model is interpretable (viz. shallow trees or small rules). With the above definitions and Assumption 3 we now provide the simple regret for the function Δ.

1. Both black box models: If both f and f_0 are black box then it seems no gains could be made in estimating the maximum of Δ over standard results in bandit literature. Hence, using Hierarchical Optimistic Optimization (HOO) with assumptions such as $\mathcal{C}$ being compact and Δ being weakly Lipschitz [42] with near optimality dimension υ the simple regret after q queries is:

$$r^{\mathcal{C}}_q(\Delta) \leq O\left(\left(\frac{\ln(q)}{q}\right)^{\frac{1}{\upsilon+2}}\right) \tag{13}$$

2. Both interpretable models: If both f and f_0 are interpretable, then for each function based on Assumption 3 we can find m_1 and m_0 partitions of $\mathcal{C}$ respectively where the functions are c_1 and c_0-Lipschitz of order β_1 and β_0 respectively. Now if we take non-empty intersections of these partitions where we could have a maximum of $m_1 m_0$ partitions, the function Δ in these partitions would be $c = 2\max(c_0, c_1)$-Lipschitz of order $\beta = \min(\beta_0, \beta_1)$ as stated next (proof in appendix).

Proposition 3. *If functions h_0 and h_1 are c_0 and c_1 Lipschitz of order β_0 and β_1 respectively, then the function $h = h_0 - h_1$ is c-Lipschtiz of order β, where $c = 2\max(c_0, c_1)$ and $\beta = \min(\beta_0, \beta_1)$.*

Given that Δ is smooth in these partitions with underestimated smoothness of order β, the simple regret after q_i queries in the ith partition $\mathcal{C}^{(i)}$ with near optimality dimension υ_i based on HOO is: $r^{\mathcal{C}^{(i)}}_{q_i}(\Delta) \leq O\left(q_i^{-1/(\upsilon_i+2)}\right)$, where $\upsilon_i \leq \frac{d}{\beta}$. If we divide the overall query budget q across the

[1]For simplicity assume $D(\cdot, \cdot)$ to be the identity function.

$\pi \leq m_0 m_1$ non-empty partitions equally, then the bound will be scaled by $\pi^{1/(\upsilon_i+2)}$ when expressed as a function of q. Moreover, the regret for the entire $\mathcal{C}$ can then be bounded by the maximum regret across these partitions leading to

$$r_q^{\mathcal{C}}(\Delta) \leq O\left(\left(\frac{\pi}{q}\right)^{\frac{\beta}{d+2\beta}}\right) \tag{14}$$

Notice that for a model to be interpretable m_0 and m_1 are likely to be small (i.e. shallow trees or small rule lists or linear model where $m = 1$) leading to a "smallish" π and υ can be much $>> \frac{d}{\beta}$ in case 1. Hence, interpretability reduces the regret in estimating the maximum deviation.

3. Black box and interpretable model: Making no further assumptions on the black box model and assuming Δ satisfies properties mentioned in case 1, the simple regret has the same behavior as (13). This is expected as the black box model could be highly non-smooth.

5 Case Studies

We present case studies to serve three purposes: 1) show that deviation maximization can lead to insights about models, 2) illustrate the maximization methods developed in Section 4, and 3) quantify the dependence of the maximum deviation on model complexity and certification set size (mostly in Appendix D). Two datasets are featured: a sample of US Home Mortgage Disclosure Act (HMDA) data (see Appendix D.2 for details), meant as a proxy for a mortgage approval scenario, and the UCI Adult Income dataset [44], a standard tabular dataset with mixed data types. A subset of results is shown in this section with full results, experimental details, and an additional Lending Club dataset in Appendix D. Since these are binary classification datasets, we take the deviation function D to be the absolute difference between predicted probabilities of class 1. For the certification set $\mathcal{C}$, we consider a union of ℓ_∞ balls (2) centered at test set instances. While we have used the test set here, any not necessarily labelled dataset would suffice. The case $r = 0$ yields a finite set consisting only of the test set, while $r \to \infty$ corresponds to $\mathcal{C}$ being the entire domain $\mathcal{X}$.

To demonstrate insights from deviation maximization, we study the solutions that maximize deviation (the $\arg\max$ in (1)) and discuss three examples below.

Identification of an artifact: The first example comes from the Adult Income dataset, where the reference model f_0 is a decision tree (DT) and f is an Explainable Boosting Machine (EBM) [45], a type of GAM (plots of both in Appendix D.3). Here, the capital loss feature is the largest contributor to the maximum deviation (the discussion below Table 4 explains how this is determined), and Table 8 in Appendix D.3 shows that as the certification set radius r increases, the maximizing values of capital loss converge to the interval $[1598, 1759]$. The plot of the GAM shape function f_j for capital loss in Figure 1 shows that this interval corresponds to a curiously low value of the function. This low region may be an artifact warranting further investigation since it seems anomalous compared to the rest of the function, and since individuals who report capital losses on their income tax returns to offset capital gains usually have high income (hence high log-odds score). Note that this potential artifact was automatically identified through deviation maximization.

For the next two examples, we consider a simplified mortgage approval scenario using the HMDA dataset. Suppose that a DT f_0 (shown in Figure 3 in Appendix D.2) has been trained to make final decisions on mortgage applications. Domain experts have determined that this DT is sensible and safe and are now looking to improve upon it by exploring EBMs. (Logistic regression (LR) models are deferred to Appendix D.2 because they do not have higher balanced accuracy than f_0.)

Conflict between f, f_0: We first examine solutions that result in the most positive difference between the predicted probabilities of an EBM f with parameter `max_bins`=32 and the DT f_0. These all occur in a leaf of f_0 (leaf 2 in Figure 3) where the applicant's debt-to-income (DTI) ratio is too high ($> 52\%$) and f_0 predicts a low probability of approval. The other salient feature of the solutions is that they all have 'preapproval'=1, indicating that a preapproval was requested, which is given a large weight by the EBM f in favor of approval (see Figure 4 in Appendix D.2, and Table 4 for more feature values). Thus f and f_0 are in conflict. Among different ways in which the conflict could be resolved, a domain expert might decide that f_0 remains correct in rejecting risky applicants with high DTI, even if there is a preapproval request and the new EBM model puts a high weight on it.

r	debt_to_income (%)	state	loan_to_value (%)	aus_1	prop_value (000$)	income (000$)
0.0	46.0	CA	95.0	3	415	77.0
0.1	[45.9 46.5]	none	[100. 100.92]	1	[120 120]	$\leq$28.5
0.2	[45.5 46.5]	none	[100. 100.92]	1	[120 120]	$\leq$28.5
0.4	[52. 52.]	none	[100. 100.92]	1	[120 120]	$\leq$28.5
0.6	[52. 52.]	none	[100. 100.92]	1	[120 120]	$\leq$28.5
0.8	[52. 52.]	none	[100. 100.92]	1	[120 120]	$\leq$28.5

Table 1: Values of top 6 features that maximize difference in predicted probabilities between a decision tree reference model f_0 and an Explainable Boosting Machine f (`max_bins` $= 32$) on the HMDA dataset. For radius $r > 0$, the maximizing values of continuous features form an interval because the corresponding EBM shape functions f_j are piecewise constant.

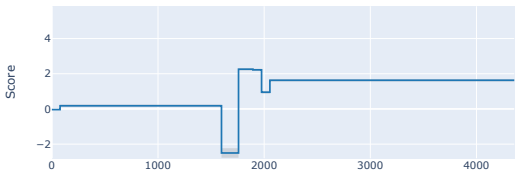


Figure 1: EBM shape function f_j for capital loss feature, showing anomalously low interval identified by deviation maximization.

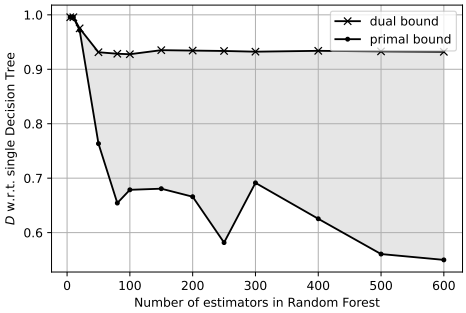


Figure 2: Maximum deviation D on the Adult Income dataset as a function of number of estimators in a Random Forest.

Trend toward extreme points, deviation can be good: We now look at solutions that yield the most negative difference between the predicted probabilities of f and f_0, for $r \leq 0.8$. Table 1 shows the 6 features that contribute most to the deviation (again see Appendix D.2 for details). All of these points lie in a leaf of f_0 (leaf 14 in Figure 3, denoted $\mathcal{L}_{14}$) that excludes several clearer-cut cases, with the result being a less confident predicted probability of 0.652 from f_0. The feature values that maximize deviation tend toward extreme points of the region $\mathcal{L}_{14}$. Specifically, the values of the continuous features debt-to-income ratio, loan-to-value ratio, property value, and income all move in the direction of application denial. For the latter three features, the boundary of $\mathcal{L}_{14}$ is reached as soon as $r = 0.1$, whereas for debt-to-income ratio, this occurs at $r = 0.4$. The movement toward extremes is expected for this f since its relevant shape functions f_j are mostly increasing or decreasing, as seen in Figure 4 in Appendix D.2. The same behavior is observed in other GAM and LR examples in Appendix D. In this example, a domain expert might conclude that the large deviation is in fact desirable because f is providing varying predictions in $\mathcal{L}_{14}$ in ways that make sense, as opposed to the constant given by f_0. This shows that deviation maximization can work in both directions, identifying where the reference model and its assumptions may be too simplistic.

Maximum deviation vs. number of trees in a RF: In Figure 2, we highlight one result from a set of such results in Appendix D, showing maximum deviation as a function of model complexity, here quantified by the number of estimators (trees) in a Random Forest (RF). This is a demonstration of the method in Section 4.3, which in general provides bounds on the maximum deviation. In this case, the upper ("dual") bound is informative enough to actually show a decrease as the number of estimators increases. The larger number of estimators increases averaging and may serve to make the model smoother.

6 Conclusion

We have considered the relationship between interpretability and safety in supervised learning through two main contributions: First, the proposal of maximum deviation as a means toward assessing safety, and second, discussion of approaches to computing maximum deviation and how these are simplified by interpretable model structure. We believe that there is much more to explore in this relationship. Appendices C and B.5 provide further discussion of several topics and future directions.

References

[1] Kush R. Varshney. Trustworthy machine learning and artificial intelligence. *ACM XRDS Mag.*, 25(3): 26–29, Spring 2019.

[2] Clemens Otte. Safe and interpretable machine learning: A methodological review. In *Computational Intelligence in Intelligent Data Analysis*, pages 111–122. 2013.

[3] Kush R. Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big Data*, 5(3):246–255, September 2017.

[4] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[5] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Mach. Intell.*, 1(5):206–215, May 2019.

[6] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[7] 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.

[8] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, pages 592–603, 2018.

[9] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

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

[11] Sina Mohseni, Zhiding Yu, Chaowei Xiao, Jay Yadawa, Haotao Wang, and Zhangyang Wang. Practical machine learning safety: A survey and primer. arXiv:2106.04823, 2021.

[12] Richard Tomsett, Dave Braines, Dan Harborne, Alun Preece, and Supriyo Chakraborty. Interpretable to whom? A role-based model for analyzing interpretable machine learning systems. In *Proc. ICML Workshop Human Interpret. Mach. Learn.*, pages 8–14, 2018.

[13] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proc. IEEE Int. Conf. Data Sci. Adv. Anal.*, pages 80–89, 2018.

[14] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37(100270), 2020.

[15] Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *Automated Technology for Verification and Analysis*, pages 269–286, 2017.

[16] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5276–5285, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/weng18a.html`.

[17] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[18] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, August 2018. URL `http://auai.org/uai2018/proceedings/papers/204.pdf`.

[19] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/f2f446980d8e971ef3da97af089481c3-Paper.pdf`.

[20] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/file/29c0605a3bab4229e46723f89cf59d83-Paper.pdf.

[21] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The Marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification*, pages 443–452, 2019.

[22] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyGIdiRqtm.

[23] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183: 3–39, 2020.

[24] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Advances in Neural Information Processing Systems*, volume 33, pages 5318–5331, 2020. URL https://proceedings.neurips.cc/paper/2020/file/397d6b4c83c91021fe928a8c4220386b-Paper.pdf.

[25] Gert R. G. Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I. Jordan. A robust minimax approach to classification. *J. Mach. Learn. Res.*, 3:555–582, December 2002.

[26] Philip S Thomas, Bruno Castro da Silva, Andrew G Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.

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

[28] Axel Parmentier and Thibaut Vidal. Optimal counterfactual explanations in tree ensembles. In *International Conference on Machine Learning*, pages 8422–8431. PMLR, 2021.

[29] Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane Boning, and Cho-Jui Hsieh. Robustness verification of tree-based models. *Advances in Neural Information Processing Systems*, 32:12317–12328, 2019.

[30] Laurens Devos, Wannes Meert, and Jesse Davis. Versatile verification of tree ensembles. In *International Conference on Machine Learning*, pages 2654–2664. PMLR, 2021.

[31] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. arXiv:1606.06565, 2016.

[32] He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. An inductive synthesis framework for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 686—701, Phoenix, AZ, USA, 2019.

[33] Jeevana Priya Inala, Osbert Bastani, Zenna Tavares, and Armando Solar-Lezama. Synthesizing programmatic policies that inductively generalize. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1l8oANFDH.

[34] Christian Rupprecht, Cyril Ibrahim, and Christopher J. Pal. Finding and visualizing weaknesses of deep reinforcement learning agents. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rylvYaNYDH.

[35] Javier García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015. URL http://jmlr.org/papers/v16/garcia15a.html.

[36] Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable Bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, page 3921–3930, 2017.

[37] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234):1–78, 2018. URL http://jmlr.org/papers/v18/17-716.html.

[38] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3):916–954, Jul 2008.

[39] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. ENDER: a statistical framework for boosting decision rules. *Data Mining and Knowledge Discovery*, 21(1):52–90, Jul 2010.

[40] Mohammad Mirghorbani and P Krokhmal. On finding k-cliques in k-partite graphs. *Optimization Letters*, 7(6):1155–1165, 2013.

[41] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3, 2002.

[42] Sebastien Bubeck, Remi Munos, Gilles Stoltz, and Csaba Szepesvari. X-armed bandits. *Journal of Machine Learning Research*, 12, 2011.

[43] Alexandra Carpentier and Michal Valko. Simple regret for infinitely many armed bandits. In *International Conference on Machine Learning*. PMLR, 2015.

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

[45] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability, 2019.

[46] Cynthia Rudin and Berk Ustun. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5):399–486, October 2018.

[47] Amit Dhurandhar, Karthikeyan Shanmugam, Ronny Luss, and Peder Olsen. Improving simple models with confidence profiles. In *Advances in Neural Information Processing Systems*, pages 10296–10306, 2018.

[48] Peter L. Bartlett and Marten H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59):1823–1840, 2008. URL `http://jmlr.org/papers/v9/bartlett08a.html`.

[49] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *Proc. Conf. Artif. Intell. (AAAI)*, pages 335–342, 1999.

[50] Ulrich Rückert and Stefan Kramer. A statistical approach to rule learning. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 785–792, 2006.

[51] Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Günlük. Generalized linear rule models. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6687–6696, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/wei19a.html`.

[52] Roger Koenker. *Quantile Regression*. Cambridge University Press, 2005. ISBN 978-0-521-60827-5.

[53] Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(6):983–999, 2006.

[54] Gábor Petneházi. QCNN: quantile convolutional neural network. *CoRR*, abs/1908.07978, 2019. URL `http://arxiv.org/abs/1908.07978`.

[55] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems*, volume 34, pages 4699–4711, 2021. URL `https://proceedings.neurips.cc/paper/2021/file/251bd0442dfcc53b5a761e050f8022b8-Paper.pdf`.

[56] Isha Puri, Amit Dhurandhar, Tejaswini Pedapati, Karthikeyan Shanmugam, Dennis Wei, and Kush R Varshney. Cofrnets: Interpretable neural architecture inspired by continued fractions. In *Advances in Neural Information Processing Systems*, volume 34, pages 21668–21680, 2021. URL `https://proceedings.neurips.cc/paper/2021/file/b538f279cb2ca36268b23f557a831508-Paper.pdf`.

[57] Laurent Meunier, Blaise J Delattre, Alexandre Araujo, and Alexandre Allauzen. A dynamical system perspective for Lipschitz neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15484–15500, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/meunier22a.html`.

[58] Ronny Luss, Pin-Yu Chen, Amit Dhurandhar, Prasanna Sattigeri, Karthik Shanmugam, and Chun-Chen Tu. Leveraging latent features for local explanations. In *ACM KDD*, 2021.

[59] Adrian Weller. Transparency: Motivations and challenges. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 23–40. 2019.

[60] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems*, 2018.

[61] High Level Expert Group on Artificial Intelligence. Assessment list for trustworthy AI for self assessment. Technical report, European Commission, 2020. URL `https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment`.

[62] Mona Sloane, Emanuel Moss, and Rumman Chowdhury. A silicon valley love triangle: Hiring algorithms, pseudo-science, and the quest for auditability. *arXix*, 2021.

[63] Matthew Arnold, Rachel KE Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilović, Ravi Nair, Karthikeyan Natesan Ramamurthy, Alexandra Olteanu, David Piorkowski, Darrell Reimer, John Richards, Jason Tsay, and Kush R. Varshney. FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM Journal of Research and Development*, 63(4/5):6, 2019.

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

[65] Navdeep Gill, Patrick Hall, Kim Montgomery, and Nicholas Schmidt. A responsible machine learning workflow with focus on interpretable models, post-hoc explanation, and discrimination testing. *Information*, 11(3), 2020. ISSN 2078-2489. doi: 10.3390/info11030137. URL `https://www.mdpi.com/2078-2489/11/3/137`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix C in particular.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix C.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code is proprietary at this time due to our institutional obligations.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] We report case studies assessing the safety of fixed models.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] The HMDA data comes from the US government and does not appear to have a license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] None of the data contain personally identifiable information, and the US Consumer Finance Protection Bureau indicates that they have modified the HMDA data to protect applicant and borrower privacy.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Problem Formulation Details

Reference model examples Below we give several examples of possible reference models. These can be categorized as 1) existing domain-specific models, 2) interpretable ML models validated by domain knowledge, and 3) extensively tested and deployed models. The first two categories are prevalent in high-stakes domains where interpretability is critical.

1. **Existing domain-specific models:** These models originate from an application domain and may not be based on ML at all. For example in consumer finance, several industry-standard models compute credit scores from a consumer's credit information (the FICO score is the best-known in the US). Similarly in medicine, scoring systems (sparse linear models with small integer coefficients) abound for assessing various risks (the CHADS_2 score for stroke risk is well-known, see the "Scoring Systems: Applications and Prior Art" section of [46] for a list of others). These models have been used for decades by thousands of practitioners so they are very well understood. They may very well be improved upon by a more ML-based model, but for such a model to gain acceptance with domain experts, any large deviations from existing models need to be examined and understood.
2. **Interpretable models validated by domain knowledge:** Here, an interpretable ML model is learned from data and is validated by domain experts in some way, for example by selecting important input features or by carefully inspecting the trained model. We provide two real examples: In semiconductor manufacturing, process engineers typically want decision trees [47] to model their respective manufacturing process (e.g. Etching, Polishing, Rapid Thermal Processing, etc.) since they are comfortable understanding and explaining them to their superiors, which is critical especially when things go out-of-spec. Hence, a tree built from data (or any model in general) would only be allowed to make automated measurement predictions if the features it highlights (viz. pressures, gas flows, temperatures) make sense for the specific process. Similarly, in predicting failures of industrial assets such as wind turbines, some failure data is available to train models but experts in these systems (e.g. engineers) may also be consulted. They have knowledge that can help validate the model, for example which components are most likely to cause failures or which environmental variables (e.g. temperature) are most influential.
3. **Extensively tested and deployed models:** A reference model may also be one that is not necessarily informed by domain knowledge but has been extensively tested, deployed, and/or approved by a regulator. For medical devices that use ML models, the US Food and Drug Administration (FDA) has instituted a risk-based regulatory system. Any system updates or changes, for instance changes in model architecture, retraining based on new data, or changes in intended use (e.g. use for pediatric cases for devices approved only for adults), need to either seek new approvals or demonstrate "substantial equivalence" by providing supporting evidence that the revised model is similar to a previously approved device. In the latter case, the reference model is the approved device and small maximum deviation serves as evidence of equivalence. As another example, consider a ML-based recommendation model for products of an online retailer or articles on a social network, where because of the scale, a tree ensemble may be used for its fast inference time as well as its modeling flexibility. In this case, a model that has been deployed for some time could be the reference model, since it has been extensively tested during this time even though human validation of it may be limited. When a new version of the model is trained on newer data or improved in some fashion, finding its maximum deviation from the reference model can serve as one safety check before deploying it in place of the reference model.

Deviation function D For the case where the inputs y, y_0 to D are real-valued scalars (which covers binary classification and regression), while Assumption 1 was stated as a sufficient condition for tractable optimization with GAMs, it is also an intuitively reasonable requirement: the deviation should increase the farther y is from y_0 in either direction. In addition, symmetry may be desirable, i.e. $D(y, y_0) = D(y_0, y)$, to not favor one of the two models over the other. Both Assumptions 1 and 2 as well as symmetry are satisfied by monotonically increasing functions $D(|y - y_0|)$ of the absolute difference, for example powers $|y - y_0|^p$ for $p > 0$.

For the case where $y, y_0 \in \mathbb{R}^M$ as in multi-class classification, it may be advantageous for $D(y, y_0)$ to decompose into a sum over output dimensions: $D(y, y_0) = \sum_{k=1}^{M} D_k(y_k, y_{0k})$, where y_k, y_{0k} are the

components of y, y_0. For example, the pth power of the ℓ_p distance $\|y - y_0\|_p^p = \sum_{k=1}^{M} |y_k - y_{0k}|^p$ is separable in this manner.

Models that abstain The formulation in Section 2 can also accommodate models that abstain from predicting (and possibly defer to a human expert or other fallback system). If $f(x) = \emptyset$, representing abstention, then we may set $D(\emptyset, y_0) = d$ for any $y_0 \in \mathcal{Y}$, where $d > 0$ is an intermediate value less than the maximum value that D can take [48]. The value d might also be less than a "typically bad" value for D, to reward the model for abstaining when it is uncertain.

B Additional Details on Deviation Maximization for Specific Model Classes

B.1 Trees

Rule lists A rule list as defined by Yang et al. [36], Angelino et al. [37] is a nested sequence of IF-THEN-ELSE statements, where the IF condition is a conjunctive rule and the THEN consequent is an output value. If each rule involves a single feature (i.e., the conjunctions are of degree 1), such rule lists are one-sided trees in the sense of Section 4.1. The number of leaves in the equivalent tree is equal to the number of rules in the list (including the last default rule).

Intersection of $\mathcal{C}$ with a Cartesian product If $\mathcal{C} = \prod_{j=1}^{d} \mathcal{C}_j$ is also a Cartesian product, then determining whether the intersection is non-empty amounts to checking whether all of the coordinate-wise intersections with $\mathcal{C}_j$, $j = 1, \ldots, d$, are non-empty. If $\mathcal{C}$ is not a Cartesian product but is a union of ℓ_∞ balls (which are Cartesian products), then the intersection is non-empty if the intersection with any one ball is non-empty.

Relationship between Proposition 1 and [29, Theorem 1] In the case of a $K = 2$-tree ensemble, [29, Theorem 1] bounds the complexity of exact robustness verification as $\min\{O(n^2), O((4n)^d\} = O(n^2)$, where n is the maximum number of leaves in a tree and we assume that the feature dimension $d \geq 2$. In Proposition 1, we account for the possibly different numbers of leaves L and L_0 in the two trees f and f_0, and we exactly enumerate the edges, $|\mathcal{E}| \leq L_0 L \leq n^2$.

Additive reference model For the case where f is a decision tree and f_0 is a generalized additive model, if the deviation function is symmetric, $D(y, y_0) = D(y_0, y)$, then this case is covered in Section 4.2.

B.2 Linear and Additive Models

Categorical features A function $f_j(x_j)$ of a categorical feature x_j can be represented in two ways, depending on whether f is a GAM or a GLM. In the GAM case, we may use the native representation in which x_j takes values in a finite set $\mathcal{X}_j$ of categories. In the GLM case, x_j is one-hot encoded into multiple binary-valued features x_{jk}, one for each category k. Then any function f_j can be represented as a linear function,

$$f_j(x_j) = \sum_{k=1}^{|\mathcal{X}_j|} w_{jk} x_{jk},$$

where w_{jk} is the value of f_j for category k.

Implication of Assumption 1 The second condition implies that the deviation increases or stays the same as y moves away from y_0 in either direction.

Proof of Proposition 2. Let $x \in \mathcal{S}$ and $S(x) = \sum_{j=1}^{d} f_j(x_j)$. Under Assumption 1.1, if $S(x) = g(y_0)$, then

$$D(f(x), f_0(x)) = D(g^{-1}(g(y_0)), y_0) = D(y_0, y_0) = 0.$$

As $S(x)$ increases from $g(y_0)$, $f(x)$ also increases because g^{-1} is an increasing function, and $D(f(x), y_0)$ increases or stays the same due to Assumption 1.2. Similarly, as $S(x)$ decreases from $g(y_0)$, $f(x)$ decreases, and $D(f(x), y_0)$ again increases or stays the same. It follows that to maximize

$D(f(x), y_0)$, it suffices to separately maximize and minimize $S(x)$, compute the resulting values of $D(f(x), y_0)$, and take the larger of the two. This yields the result. □

Implication of Assumption 2 and identity link function g These two assumptions imply that the deviation is measured on the difference between f and f_0 in the space in which they are additive. For example, if f and f_0 are logistic regression models predicting the probability of belonging to one of the classes, the difference is taken in the log-odds (logit) domain. It is left to future work to determine other assumptions under which problem (1) is tractable when f and f_0 are both additive.

Cartesian product $\mathcal{C}$ implies Cartesian product $\mathcal{S}$ In the cases of constant and additive f_0, $\mathcal{S} = \mathcal{C}$. In the decision tree case, since each leaf is a Cartesian product $\mathcal{L}_{0m} = \prod_{j=1}^{d} \mathcal{R}_{mj}$, the intersections $\mathcal{S} = \mathcal{L}_{0m} \cap \mathcal{C}$ are also Cartesian products $\prod_{j=1}^{d} \mathcal{S}_j$ where $\mathcal{S}_j = \mathcal{R}_{mj} \cap \mathcal{C}_j$.

One-dimensional optimization complexities C_j For discrete-valued x_j, C_j is proportional to the number of allowed values $|\mathcal{S}_j|$. For continuous x_j, it is common to use spline functions or tree ensembles as f_j in constructing GAMs. In the former case, C_j is proportional to the number of knots. In the latter, the tree ensemble can be converted to a piecewise constant function and C_j is then proportional to the number of pieces. Lastly in the case where $f_j(x_j) = w_j x_j$ is linear and $\mathcal{S}_j = [\underline{X}_j, \overline{X}_j]$ is an interval, $C_j = O(1)$ because it suffices to evaluate the two endpoints.

Convex $\mathcal{S}$ If $\mathcal{C}$ is a convex set, then in the cases of constant and additive f_0, $\mathcal{S} = \mathcal{C}$ is also convex. In the case of tree-structured f_0, $\mathcal{S} = \mathcal{L}_{0m} \cap \mathcal{C}$ and each leaf $\mathcal{L}_{0m}$ can be represented as a convex set, with interval constraints on continuous features and set membership constraints on categorical features. The latter can be represented as $x_{jk} = 0$ constraints on the one-hot encoding (see "Categorical features" paragraph above) for non-allowed categories k. Hence $\mathcal{S}$ is also convex.

As a specific example, suppose that $\mathcal{S}$ is the product of independent constraints on each categorical feature and an ℓ_p norm constraint on the continuous features jointly. The maximization over each categorical feature has complexity $C_j = |\mathcal{S}_j|$ as noted above, while the maximization of $w^T x$ over continuous features lying in an ℓ_p ball has closed-form solutions for the common cases $p = 1, 2, \infty$.

Relaxations of the Cartesian product assumption If the certification set $\mathcal{C}$ is not a Cartesian product, then one way to still bound the maximum deviation is to find the smallest Cartesian product $\overline{\mathcal{C}}$ that contains $\mathcal{C}$ and maximize deviation over $\overline{\mathcal{C}}$. As long as it is relatively easy to optimize linear functions over $\mathcal{C}$, then constructing such a Cartesian product is similarly easy. Another conceivable relaxation of the Cartesian product assumption is a Cartesian product of low-dimensional sets, not just one-dimensional.

B.3 Tree Ensembles

The full algorithm for clique search from Section 4.3 is presented in Algorithm 1. It uses Z as a node compatibility vector to keep track of valid leaves and B a set of trees/partites not yet covered by the maximum clique. The algorithm starts with and empty clique S and anytime bounds as 0. It starts the search with the smallest tree to limit the search space. This is typically f_0. Each leaf is added to the intermediate clique S in turn (Line 6). A stronger primal bound can be achieved if the traversal is ordered in a meaningful way. In particular, starting with nodes with the highest heuristic function value $H(S)$ aids the algorithm to focus on better areas of the search space.

If the size of the clique is $K + 1$ the primal bound is updated. Otherwise, the dual bound is computed. If the node is promising, the algorithm recurses to the next level. When the search is terminated at any step, the maximum deviation is bounded by (D_{lb}, D_{ub}).

Rule ensembles Similar to the tree ensembles considered in Section 4.3, a rule ensemble is a linear combination of conjunctive rules, where the antecedent is a conjunction of conditions on individual features, and the consequent takes a real value if the antecedent is true and zero otherwise. They are produced by algorithms such as SLIPPER [49], that of Rückert and Kramer [50], RuleFit [38], ENDER [39] and have also been referred to as generalized linear rule models [51]. A rule ensemble can be converted into a tree ensemble by converting each conjunctive rule into an IF-THEN-ELSE rule

Algorithm 1 Max clique search for maximum deviation

Require: M adjacency matrix, H heuristic function
$Z[i] = 1 \forall i \in V$, $B = \{1, 2, \ldots, K+1\}$, $S = \emptyset$ ▷ All nodes valid, all trees uncovered
Q = Enumerate(Z, B, S)
Initialize: $D_{lb} = 0$, $D_{ub} = 0$ ▷ Anytime bounds
function Enumerate(Z, B, S):
$t = \arg\max_b \{|Z_b| \mid b \in B\}$ ▷ Uncovered tree with fewest valid nodes
for i in Z_t **do**
  $Z[i] = 0$ ▷ Mark node as incompatible
  $S = S \cup \{i\}$ ▷ Add to candidate clique
  if $|S| = K + 1$ **then**
    $D_{lb} = \max(D_{lb}, D(S))$ ▷ Update primal bound
    $Q = Q \cup S$ ▷ Add to set of max cliques
    $S = S \setminus \{i\}$ ▷ Backtrack
  else
    $Z_{t+1} = Z_t \wedge M(i)$ ▷ Update valid nodes
    $B = B \setminus \{t\}$ ▷ Update uncovered trees
    $D_{ub} = \max(D_{ub}, H(S))$ ▷ Update dual bound
    if $D_{ub} > D_{lb}$ **then**
      Enumerate(Z_{t+1}, B, S) ▷ Recurse to next level
    end if
    $S = S \setminus \{i\}$ ▷ Backtrack
    $B = B \cup \{t\}$
  end if
end for

list, which is a one-sided tree (see Appendix B.1). Specifically, the conditions in the conjunction are taken in any order, each condition is negated to become an IF condition, and the THEN consequents are all output values of zero. The final ELSE consequent, which is reached if all the IF conditions are false (and hence the original rule holds), returns the output value of the original rule. The number of leaves in the resulting tree equals the number of conditions in the conjunction plus one.

B.4 Piecewise Lipschitz Functions

Proof of Proposition 3. Consider two inputs x and y then,

$$\begin{aligned}|h(x) - h(y)| &= |(h_0 - h_1)(x) - (h_0 - h_1)(y)| = |h_0(x) - h_0(y) + h_1(y) - h_1(x)| \\ &\leq |h_0(x) - h_0(y)| + |h_1(x) - h_1(y)| \leq c_0 \cdot \ell(x, y)^{\beta_0} + c_1 \cdot \ell(x, y)^{\beta_1} \\ &\leq c \cdot \ell(x, y)^{\beta}\end{aligned}$$

where, $c = 2\max(c_0, c_1)$ and $\beta = \min(\beta_0, \beta_1)$. □

Other choices for $D(.,.)$: The results assumed $D(.,.)$ to be the identity function, where $\Delta = D(f_0, f)$. This choice of function clearly satisfies Assumptions 1 and 2. Again consistent with these assumptions we look at some other choices for $D(.,.)$. If $D(.,.)$ were an affine function with a positive scaling such as $D(y_0, y) = \alpha(y_0 - y) + b$ where $\alpha > 0$, then our result in equation 14 would be unchanged as only the Lipschitz constant of Δ would change, but not its (underestimated) order. If the function were a polynomial or exponential however, no such guarantees can be made and we would be back to case 1.

B.5 Other Model Classes: Neural Networks and Post Hoc Explanations

For model classes beyond the ones discussed in Section 4, it appears to be a greater challenge to obtain reasonably tractable algorithms that guarantee exact computation of or bounds on the maximum deviation. Here we outline some future directions for neural networks and post hoc explanations.

Robustness verification for neural networks has attracted a great deal of attention and made considerable progress, with exact approaches including satisfiability modulo theory [21] and mixed

integer programming [22, 23], and incomplete methods that compute bounds using bound propagation [19, 16], linear programming and duality [15, 17, 18], and semidefinite programming [20, 24]. However, all of these methods consider a single model, effectively comparing it to a constant. Robustness verification is thus essentially a single-model case of our problem (1) in which f_0 is a constant (and with an appropriate choice of the deviation function $D(f, f_0)$). While we may expect that solutions to a two-model verification problem would leverage existing robustness verification methods, developing such solutions remains for future work. Furthermore, evaluation of robustness verification methods has largely been limited to local neighborhoods around input points (with typical radii $\epsilon \leq 0.1$ in terms of normalized feature values). This limitation may also need to be addressed to enable evaluation of maximum deviation in the way envisioned in this paper.

It is also natural to ask whether post hoc explanations for the model can help. One way in which this could occur is if the post hoc explanation approximates the model f by a simpler model $\hat{f}$ and if the deviation function D satisfies the triangle inequality $D(f(x), f_0(x)) \leq D(f(x), \hat{f}(x)) + D(\hat{f}(x), f_0(x))$. Then the maximum deviation in (1) would be bounded as

$$\max_{x \in \mathcal{C}} D(f(x), f_0(x)) \leq \max_{x \in \mathcal{C}} D(f(x), \hat{f}(x)) + \max_{x \in \mathcal{C}} D(\hat{f}(x), f_0(x)). \tag{15}$$

While we may choose $\hat{f}$ to be interpretable so that the rightmost maximization is tractable, the middle maximization asks for a *uniform* bound on the deviation between f and $\hat{f}$, i.e, the fidelity of $\hat{f}$. We are not aware of a post hoc explanation method that provides such a guarantee. Indeed, in general, the middle maximization might not be any easier than the left-hand one that we set out to bound.

A (practical) possibility may be to perform quantile regression [52] for a large enough quantile to learn $\hat{f}$, as opposed to minimizing expected error as is typically done. This may be an interesting direction to explore in the future as quantile regression algorithms are available for varied model classes including linear models, tree ensembles [53] and even neural networks [54]. More investigation is needed into whether quantile regression methods can provide approximate guarantees on the middle term in (15).

Assuming that uniform proxies in the above sense can be constructed, then for certain modalities or applications it may be possible to train highly accurate proxies. For instance for tabular data, Random Forests or boosted trees might very well replicate the behavior of a neural network, in which case the machinery introduced in Section 4.3 could be used. Even for other modalities such as text and images, interpretable models such as Neural Additive Models (NAMs) [55] and continued fraction networks (CoFrNets) [56] may prove to be sufficient in some cases.

Finally, there are recent architectures such as Lipschitz neural networks [57] which are adversarially robust and hence valuable in practice. Our analysis presented in Section 4.4 for piecewise Lipschitz models would be applicable here, where the simple regret of standard bandit algorithms for a given number of queries could be reduced to (14) as opposed to (13).

C Further Discussion

Worst-case approach The formulation of (1) as the worst case over a certification set represents a deliberate choice to depend as little as possible on a probability distribution or a dataset sampled from one. As stated in Section 2, Certification Set paragraph, $\mathcal{C}$ can depend at most on (an expanded version of) the support set of a distribution. The reason for this choice is because safety is an out-of-distribution notion: harmful outputs often arise precisely because they were not anticipated in the data. The trade-off inherent in this choice is that the maximum deviation may be more conservative than needed. The high maximum deviation values in e.g. Figure 11 may reflect this. Given definition (1) as a starting point in this paper, future work could consider variations that depend more on a distribution and are thus less conservative, but may also offer a weaker safety guarantee.

Choice of reference model The proposed definition of maximum deviation (1) depends on the choice of reference model f_0. Different choices will lead to different deviation values and, perhaps more importantly, different combinations of features that maximize the deviation. We have discussed possible choices in Section 2, and the results in Section 4 indicate that, as with the assessed model f, interpretable forms for f_0 can ease the computation of maximum deviation. Beyond these guidelines, it is up to ML practitioners and domain experts to decide on appropriate reference models for their

application (and there may be benefit to considering more than one). We mention an additional concern with the reference model in the Ethics Discussion.

For some real applications it may be difficult to come up with a globally interpretable reference model. But specific to particular scenarios it may be possible. For instance, it might be difficult to provide general rules for how to drive a car, but in specific scenarios such as there being an obstacle in front, one can suggest that you stop or turn, which is a simple rule. So our machinery could potentially be applied at a local level where the reference model is interpretable in that locality. This might help in "spot checking" a deployed model and estimating its safety by computing these maximum deviations in scrupulously selected (challenging) scenarios.

Impossible inputs in certification set Mathematically simple sets such as Cartesian products and ℓ_p balls permit simpler algorithms for optimizing functions over them. Accordingly, these sets have been the focus of not only the present work but also the related literature on ML verification and adversarial robustness. However, they may not serve to exclude inputs that are physically or logically impossible from the certification set $\mathcal{C}$, and thus, the resulting maximum deviation values may be too large and conservative. Here it is important to distinguish between impossible inputs and those that are merely implausible (i.e., with low probability). Techniques for capturing implausibility have been proposed for contrastive/counterfactual explanations [8, 58], whereas we expect the set of impossible inputs to be smaller and more constrained. As a simple example from the Adult Income dataset, if we agree that a wife/husband is defined to be of female/male gender (regardless of the gender of the spouse), then the cross combinations male-wife and female-husband cannot occur. Future work can consider the representation and handling of such constraints.

White-box vs. grey-box models In this paper, we have assumed full "white-box" access to both f and f_0, namely complete knowledge of their structure and parameters. Interesting questions may arise when this assumption is relaxed to different "grey-box" possibilities. For example, one could further investigate the third case in Section 4.4, where one of f, f_0 is black-box and the other is white-box interpretable. There may exist assumptions that we have not identified that would improve the query complexity compared to generic black-box optimization.

Other interpretability-safety relationships This paper has focused on one relationship between the interpretability of a model and the safety of its outputs. It has not addressed other ways in which interpretability/explainability can affect the *risk* of a model (in the plain English sense, not the expectation of a loss function). For example, in regulated industries such as consumer finance, not providing explanations or providing inadequate ones can lead to legal, financial, and reputational risks. On the other hand, providing explanations is associated with its own risks [59]. These include the leakage of personal information or model information (intellectual property), an increase in appeals of decisions for the decision-making entity, and strategic manipulation of attributes (i.e. "gaming") by individuals to gain more favorable outcomes.

Applicability to RL settings In RL, if one views the actions as labels and state representation as features, one can build a tree, albeit likely a deep/wide one, to represent exactly the RL policy, where the probability distribution over the actions can be viewed as the class distribution in a normal supervised setting. Rolling up the states, creating leaves with multiple states, and simply averaging the probabilities for each action would yield smaller trees that approximate the policy. Our work lays a foundation where in principle we can also compare f and f_0 that are policies using such tree representations. This may be related to a popular global explainability method [60] that samples policies and builds trees to explain them.

Ethics The safety of machine learning systems has been called out by the European Commission's regulatory framework [61]. The commission states seven key dimensions to be evaluated and audited by a cross-disciplinary team: (i) human agency and oversight, (ii) technical robustness and safety, (iii) privacy and data governance, (iv) transparency, (v) diversity, non-discrimination and fairness, (vi) environmental and societal well-being, and (vii) accountability. The second of these dimensions is safety. However, Sloane et al. [62] argue that algorithmic audits are ill-defined as the underlying definitions are vague. The proposed work helps fill that ill-definedness using a quantitative approach. One may argue against this particular choice of quantification, but it does start the community down the path toward being more concrete in its definitions.

As with many other technologies, the proposed approach may be misused. For example, the reference model may be chosen in a way that hides the safety concerns of the model being evaluated. Transparent documentation and reporting with provenance guarantees can help avoid this kind of purposeful deceit [63].

D Experiment Details and Additional Results

D.1 General Experiment Details

Data processing We use the training set of each dataset to train models and the test set as the basis for evaluating maximum deviation, specifically as the set of centers for the ℓ_∞ balls in (2). Continuous features are standardized and categorical features are one-hot encoded. The ℓ_∞ norm is computed on the resulting normalized feature values.

Models We use scikit-learn [64] to train decision tree (DT), logistic regression (LR), and Random Forest (RF) models. The corresponding complexity parameters are the number of leaves for DT (parameter `max_leaf_nodes`), the amount of ℓ_1 regularization for LR (inverse ℓ_1 penalty C), and the number of estimators/trees for RF (`n_estimators`). For additive models, we use Explainable Boosting Machines (EBM) from the InterpretML package [45] with zero interaction terms. Smoothness is controlled by the `max_bins` parameter, the number of discretization bins for continuous features.

Deviation maximization In all cases, when the certification set radius $r = 0$, maximum deviation can be computed simply by evaluating the models on the test set. For the case where $r > 0$, f is a DT or RF, and f_0 is a DT, Algorithm 1 is used (on a bipartite graph if f is a DT). The cases where f is LR or EBM fall under the generalized additive case of Section 4.2. Given that f_0 is a DT, we use (7), (6) to determine the maximum deviation. For $r < \infty$ when $\mathcal{C}$ is a union of ℓ_∞ balls, we maximize separately over each intersection between a ball and a leaf of f_0 and then take the maximum over the intersections.

Computation All experiments were run on CPU nodes with 64GB memory. For decision trees and tree ensembles run times of Algorithm 1 were limited to 2 hours, after which the best available bounds were used.

D.2 Home Mortgage Disclosure Act Dataset

Data source and pre-processing The data is made available by the US Consumer Finance Protection Bureau (CFPB) under the Home Mortgage Disclosure Act (HMDA). We use the national snapshot from year 2018[2] of "loan/application records," which contain information on mortgage applications and their outcomes. According to their website, the CFPB has modified the data to protect applicant and borrower privacy.

We processed the 2018 loan/application records as summarized below. These steps were informed by Gill et al. [65] but not identical to theirs:

- Restrict to complete, submitted applications with 'action_taken' ≤ 3 (loan originated, application approved but not accepted by applicant, or application denied).
- Create a binary-valued target variable representing approval by binarizing 'action_taken' (originated or approved $\rightarrow 1$, denied $\rightarrow 0$).
- Restrict to purchases of principal residences (the most consequential in terms of people's lives, i.e., not refinances or for investment) and single-family homes.
- Restrict to loans that are not "special" in any way: conventional loans, first mortgages, not manufactured homes, no non-amortizing features, etc.
- Drop columns that are not applicable for site-built single-family homes.
- Drop columns that are not applicable or recorded until the approval/origination decision is made. For example, loan costs (points and fees) are not recorded until a loan is originated, the type of entity that purchases a loan does not apply unless the loan is originated, etc. This is in keeping with the mortgage approval scenario that we consider.
- Drop all demographic columns to reflect laws that forbid lending decisions from explicitly depending on applicant demographics.

[2] `https://ffiec.cfpb.gov/data-publication/snapshot-national-loan-level-dataset/2018`

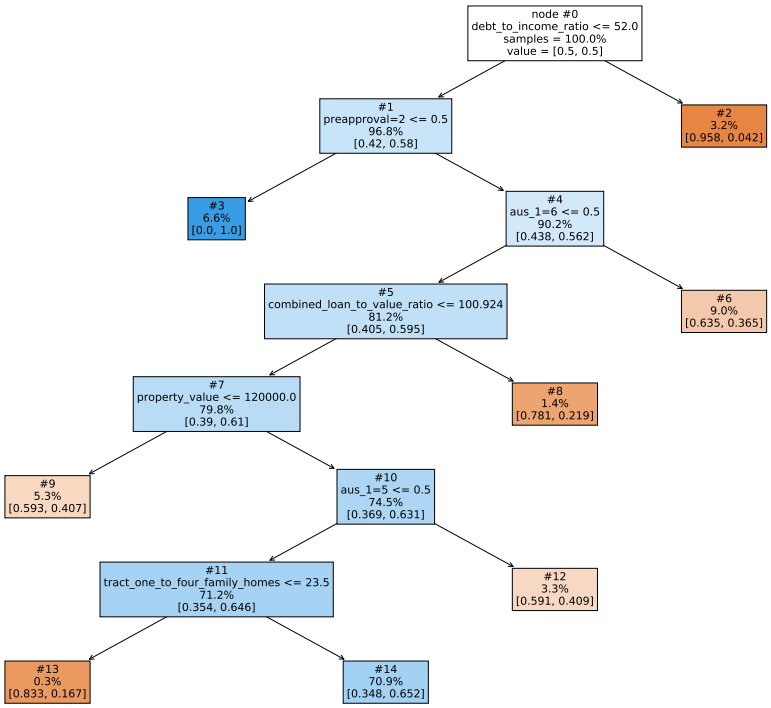


Figure 3: Decision tree reference model with 8 leaves for the HMDA dataset.

- Drop geographical columns that have too many unique values (e.g., census tract).
- Drop rows that have null or "exempt" values in key features such as loan-to-value ratio, debt-to-income ratio, property value, income.
- Take a 10% sample of the records remaining after the above processing, to make experimentation less time- and resource-consuming.
- Split the subsampled dataset 80%–20% into training and test sets.

The dataset resulting from the above processing is imbalanced, with nearly 93% of mortgage applications approved. Thus in training models, we balance the classes by weighting, either using the `class_weight='balanced'` option in scikit-learn or defining sample weights for the same purpose. On the test set, we evaluate balanced accuracy instead of accuracy.

Reference model Figure 3 depicts the 8-leaf DT reference model used in the experiments on the HMDA dataset. This DT has a test set balanced accuracy of 70.9% and area under the receiver operating characteristic (AUC) of 0.750. The top-level split is based on debt-to-income ratio, a measure often used in lending. Other common mortgage measures such as loan-to-value ratio, property value, and whether a preapproval was requested (value 2 means no) also appear. 'aus_1'=5 and 'aus_1'=6 refer to the automated underwriting system used to evaluate the application, with values 5 and 6 denoting "other" and "not applicable".

LR and GAM models In Tables 2 and 3, we show the values of inverse ℓ_1 penalty C and `max_bins` that were used for LR and GAM respectively, as well as statistics of the resulting classifiers. Test set balanced accuracy and AUC increase and reach a plateau. For LR, we take the ℓ_1 norm of the coefficients to be the main measure of smoothness as it depends on both the number of nonzero coefficients as well as their magnitudes, which both affect the extreme values attained in (6). Since the LR balanced accuracy and AUC are not higher than those of the reference DT, we focus less on LR in what follows.

C	nonzeros	ℓ_1 norm	bal. acc.	AUC
1e-4	2	0.4	0.605	0.663
3e-4	7	1.3	0.633	0.695
1e-3	17	4.6	0.663	0.736
3e-3	26	7.7	0.669	0.744
1e-2	42	12.7	0.674	0.750
3e-2	68	16.6	0.675	0.751
1e-1	85	20.3	0.675	0.752
3e-1	96	22.2	0.676	0.752
1e+0	99	22.9	0.675	0.752
3e+0	100	23.2	0.675	0.752

Table 2: Number of nonzero coefficients, ℓ_1 norm of coefficients, test set balanced accuracy, and area under the receiver operating characteristic (AUC) for logistic regression models on the HMDA dataset as a function of inverse ℓ_1 penalty C.

max_bins	bal. acc.	AUC
4	0.669	0.743
8	0.695	0.772
16	0.711	0.785
32	0.720	0.798
64	0.723	0.799
128	0.722	0.800
256	0.722	0.800
512	0.723	0.800
1024	0.723	0.800

Table 3: Test set balanced accuracy and AUC for Explainable Boosting Machines on the HMDA dataset as a function of `max_bins` parameter.

For EBM, based on Table 3, we select `max_bins` $= 32$ as a representative model with balanced accuracy and AUC nearly equal to the maximum attainable values. Plots for this EBM model are shown in Figure 4. First note that whether a preapproval was requested (value 1 means yes) is quite predictive of final approval. The shape functions for the four continuous features debt-to-income ratio, loan-to-value ratio, property value, and income are mostly monotonic and agree with domain knowledge. The log-odds of mortgage approval decrease as debt-to-income ratio and loan-to-value ratio increase, with abrupt drops around 50% for debt-to-income ratio and at 80% and 100% for loan-to-value ratio. For property value and income, after a minimum value is reached, the shape function increases rapidly and then stays more or less constant for high property values and incomes.

Figure 4: Feature importances and selected univariate functions f_j for the Explainable Boosting Machine with `max_bins` $= 32$ on the HMDA dataset.

r	preapproval	appl_credit _score_type	loan_to_value (%)	co_appl_credit _score_type	intro_rate_period	debt_to_income (%)
0.0	1	2	74.35	10	122.0	55.0
0.1	1	3	[80. 80.]	1	[28.4 43.6]	[53.9 56.1]
0.2	1	1	[29.33 32.42]	1	[57. 60.5]	[52.8 57.2]
0.4	1	7	[4.34 32.42]	7	[329.7 360.]	[52. 53.4]
0.6	1	7	[0.49 32.42]	7	[314.5 360.]	[52. 55.6]
0.8	1	7	[0.49 32.42]	7	[299.3 300.]	[52. 57.7]
0.999	1	7	[0.49 32.42]	7	[120.5 163.]	[52. 53.9]
1.001	1	6	[0.49 32.42]	6	[120.5 159.9]	[52. 62.5]
1.2	1	6	[0.49 32.42]	6	[120.5 151.]	[56.9 62.5]
1.4	1	6	[0.49 32.42]	6	[120.5 163.]	[52. 57.3]
1.6	1	6	[0.49 32.42]	6	[120.5 163.]	[52. 60.5]
1.8	1	6	[0.49 32.42]	6	[120.5 163.]	[52. 52.7]
2.0	1	6	[0.49 32.42]	6	[120.5 163.]	[52. 62.5]
∞	1	6	[0.49 32.42]	6	[120.5 163.]	[52. 62.5]

Table 4: Feature values that result in most positive difference in predicted probabilities between an Explainable Boosting Machine f (`max_bins` $= 32$) and an 8-leaf decision tree reference model f_0 on the HMDA dataset. The 6 features that contribute most are shown as a function of certification set radius r. For radius $r > 0$, the maximizing values of continuous features form an interval because the corresponding functions f_j are piecewise constant.

Feature combinations that maximize deviation Table 4 shows feature values that yield the most positive difference between the predicted probabilities of the EBM f with `max_bins`$= 32$ and the 8-leaf DT f_0. This table corresponds to the second "Conflict between f, f_0" example in Section 5. The 6 features that contribute most to the deviation are shown. These contributions are determined using (8); since the maximum deviation occurs in one of the ℓ_∞ ball-leaf intersections and this intersection is a Cartesian product, the feature-wise decomposition in (8) applies. The contribution of feature j is then $\max_{x_j \in \mathcal{S}_j} f_j(x_j)$. We take an average of the contributions over r to give a single ranking of features for all r. The same method is used to determine feature contributions and choose the top 6 features for Table 1.

As mentioned in Section 5, all solutions in Table 4 have 'preapproval'=1 and debt-to-income ratios $> 52\%$ that place them in leaf 2 in Figure 3. The latter results in a low predicted probability of approval from f_0 while the former makes a large positive contribution to the probability from f (see Figure 4). The values of the other features also make increasingly larger positive contributions to f as r increases. Loan-to-value ratio decreases, while co-applicant credit score type moves from type 10 (no co-applicant, hence weaker application) to increasingly favorable score types (1, 7, 6, see Figure 4); applicant credit score is similar. This behavior is similar to the movement toward extreme points seen in Table 1.

Dependence on model complexity Figure 5 shows the dependence of maximum deviation on the complexity of model f, quantified by the number of leaves for DTs, coefficient ℓ_1 norm for LR, `max_bins` for EBM, and the number of estimators (trees) for RF. The DT and RF cases demonstrate the methods in Section 4.3, specifically Algorithm 1, where in the RF case, the algorithm may only provide bounds after a time limit of two hours. The plots show that maximum deviation may or may not increase with model complexity. In Figure 5a, the deviation is small for a 10-leaf DT and increases rapidly. Figures 5b and 5c indicate that maximum deviation is sensitive to the ℓ_1 norm of LR models but not to the `max_bins` parameter of EBMs. The latter may increase the resolution of the EBM shape functions but not their dynamic range.

Dependence on certification set size Figure 6 shows the dependence of maximum deviation on the certification set radius r. For LR and GAM, the maximum deviation is greater for $r > 0$ than for $r = 0$, showing that evaluation on a finite test set may not be sufficient and infinite certification sets (with $r > 0$) should be considered, especially to account for unexpected, out-of-distribution deviations. There are jumps at $r = 1$ because this is the radius that permits values of categorical features of test set points (the ball centers in (2)) to change to any other value. In the case of GAM,

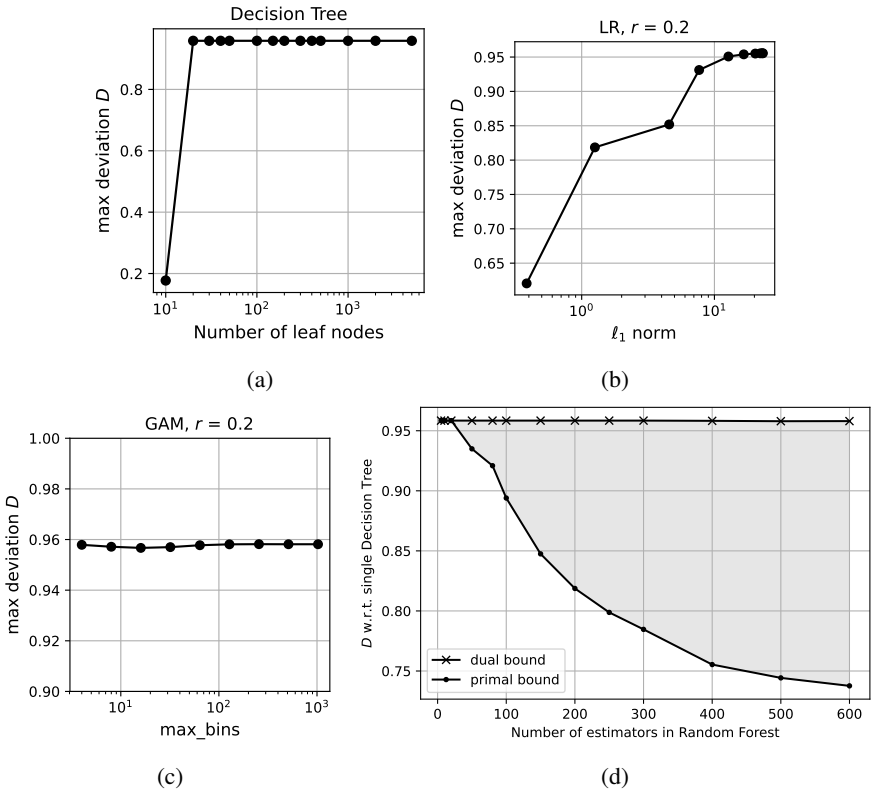


Figure 5: Maximum deviation D on the HMDA dataset as a function of model complexity for (a) DT (number of leaves), (b) LR (ℓ_1 norm), (c) GAM (`max_bins`), and (d) RF (number of estimators).

this jump is sufficient for the deviation to equal that for $r = \infty$ ($\mathcal{C} = \mathcal{X}$, dashed line in figure). On the other hand, the deviation for DT and RF remains constant as a function of r.

Running time Figure 7 shows the time required to compute the maximum deviation for LR and GAM on the HMDA dataset. These times were obtained using a single 2.0 GHz core of a server with 64 GB of memory (only a small fraction of which was used) running Ubuntu 16.04 (64-bit). The times increase with the ℓ_∞ ball radius r because of the increasing number of ball-leaf intersections that become non-empty and hence need to be evaluated. The time for $r = 0$ is minimal because this case requires only model evaluation over the finite test set, as mentioned. The jumps at $r = 1$ are due again to the ability of categorical features to change values, leading to an increase in ball-leaf intersections. The filled-in regions show that there was little variation due to different ℓ_1 norms for LR or `max_bins` for GAM. This was most likely because of a vectorized implementation, which operates on all LR coefficients or all GAM bins at once (i.e., without a for loop).

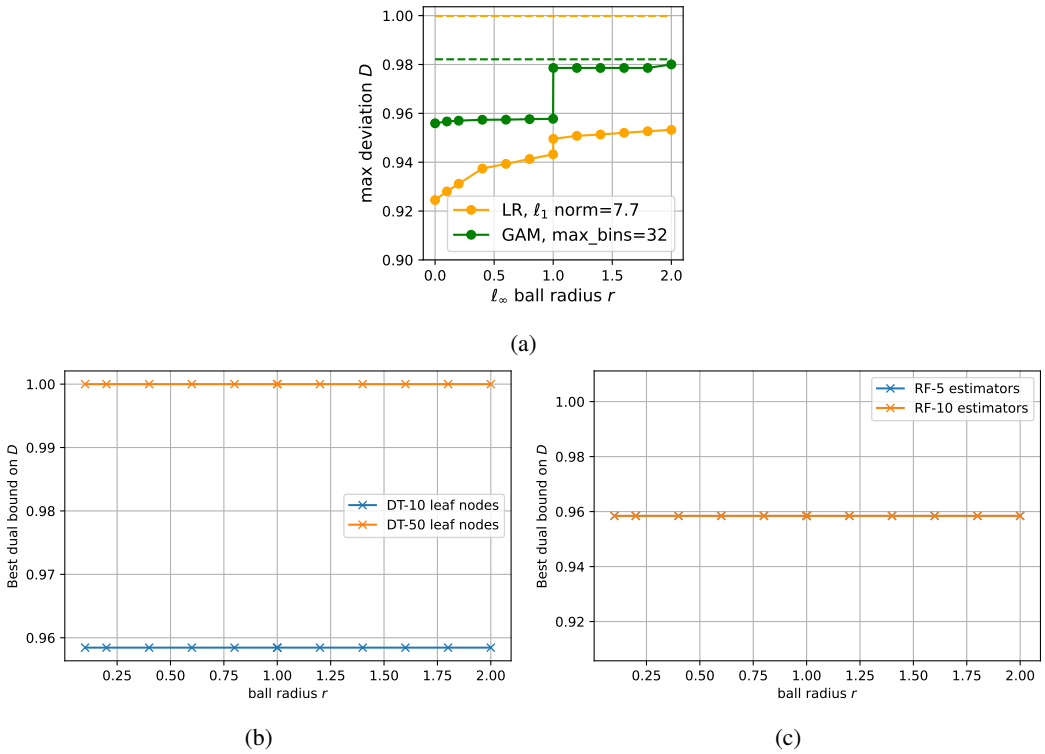


Figure 6: Maximum deviation D on the HMDA dataset as a function of certification set radius r for (a) LR and GAM, (b) DT (10 and 50 leaves), (c) RF (5 and 10 estimators). Dashed lines in (a) indicate the $r \to \infty$ asymptote of the curve of the same color.

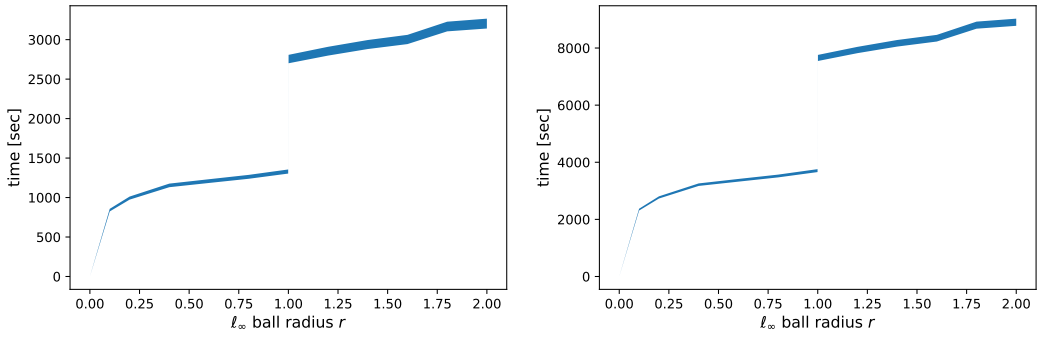


Figure 7: Time to compute maximum deviation for logistic regression models (left) and Explainable Boosting Machines (right) on the HMDA dataset as a function of certification set size (radius r). The filled-in region shows the min-max variation with model complexity (ℓ_1 norm for LR, `max_bins` for EBM).

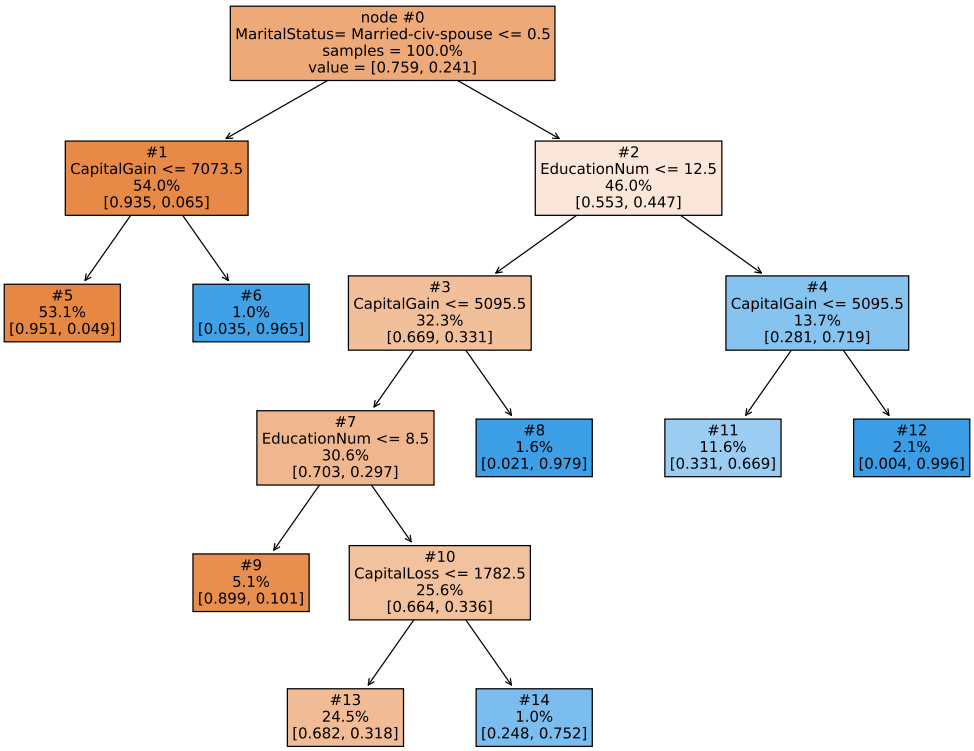


Figure 8: Decision tree reference model with 8 leaves for the Adult Income dataset.

D.3 Adult Income dataset

We use the given partition of the Adult Income dataset into training and test sets.

Reference model Figure 8 depicts the 8-leaf DT reference model used in the experiments on the Adult Income dataset. This DT has 85.0% accuracy on the test set. The root node separates individuals based on whether the marital status is *Married-civ-spouse*. The remaining splits divide the population into those with high and low education, high and low capital gains, and high and low capital losses. In particular, having high capital gains or losses is a good predictor of high income ($> \$50000$).

LR and GAM models Tables 5 and 6 show the values of C and `max_bins` used for LR and GAM respectively together with statistics of the resulting classifiers. Based in part on Tables 5 and 6, we select $C = 0.01$ and `max_bins` $= 8$ as representative models that remain simple and have accuracies and AUCs not far from the maximum attainable. Plots for these two models are shown in Figures 9 and 10.

C	nonzeros	ℓ_1 norm	accuracy	AUC
3e-4	1	0.2	0.764	0.715
1e-3	6	2.6	0.825	0.885
3e-3	7	4.7	0.840	0.895
1e-2	16	7.4	0.848	0.900
3e-2	30	12.9	0.852	0.905
1e-1	38	17.3	0.853	0.905
3e-1	62	25.3	0.853	0.905
1e+0	83	40.7	0.852	0.905
3e+0	92	54.6	0.852	0.905
1e+1	101	65.1	0.852	0.904
3e+1	105	73.5	0.852	0.904
1e+2	107	77.6	0.852	0.904
3e+2	107	79.1	0.852	0.904

Table 5: Number of nonzero coefficients, ℓ_1 norm of coefficients, test set accuracy, and AUC for logistic regression models on the Adult Income dataset as a function of inverse ℓ_1 penalty C.

`max_bins`	accuracy	AUC
4	0.858	0.910
8	0.862	0.915
16	0.865	0.920
32	0.870	0.924
64	0.871	0.925
128	0.871	0.925
256	0.872	0.925
512	0.871	0.925
1024	0.871	0.925

Table 6: Test set accuracy and AUC for Explainable Boosting Machines on the Adult Income dataset as a function of `max_bins` parameter.

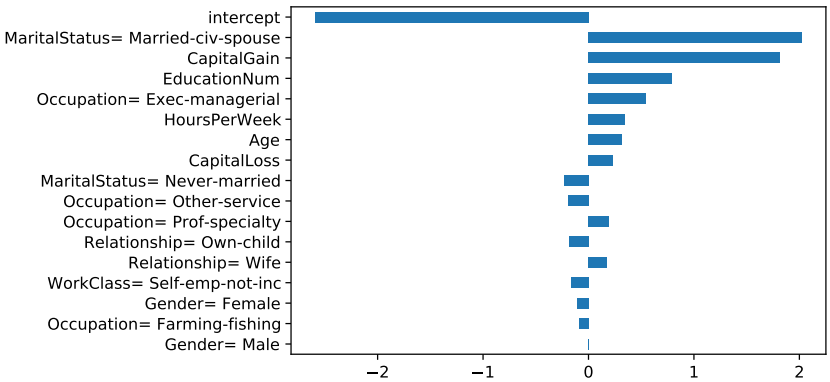
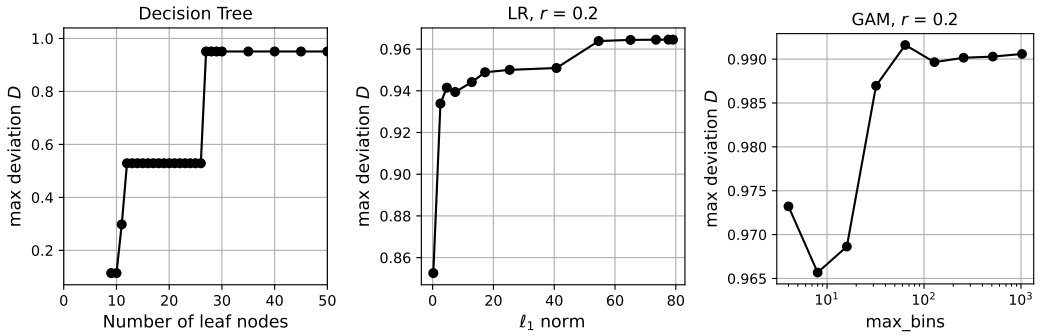


Figure 9: Coefficient values of the logistic regression model with $C = 0.01$ (16 nonzeros) for the Adult Income dataset.

Figure 10: Feature importances and selected univariate functions f_j for the Explainable Boosting Machine with `max_bins` $= 8$ on the Adult Income dataset.



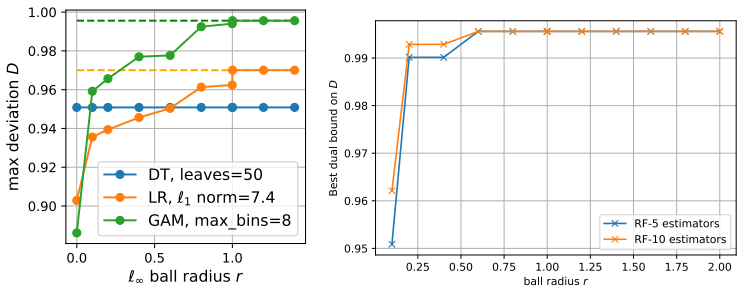
(a) (b) (c)

Figure 11: Maximum deviation D on the Adult Income dataset as a function of model complexity for (a) DT (number of leaves), (b) LR (ℓ_1 norm), and (c) GAM (`max_bins`).

Dependence on model complexity Figure 11 shows the dependence on model complexity for DT (number of leaves), LR (coefficient ℓ_1 norm), and EBM (`max_bins`). In Figure 11a, the maximum deviation is 0.114 for trees with 9 and 10 leaves, and remains moderate up to 26 leaves, which is different than in Figure 5a. Similar to Figures 5b and 5c, ℓ_1 norm has a larger effect on maximum deviation than `max_bins` (note the vertical scale in Figure 11c).

Dependence on certification set size Figure 12a shows the dependence on the certification set radius r for DT, LR, and GAM. The patterns are similar to those in Figure 6: the deviations for LR and GAM increase from $r = 0$ and have jumps at $r = 1$, while the deviation for DT remains constant. One difference is that the LR curve in Figure 12a meets its $r \to \infty$ asymptote (dashed line in figure), similar to GAM.

Figure 12b shows the upper bound on the maximum deviation as a function of the certification set size for two RF models. As the test set is large in this case, the deviations observed even for small values of r are high and grow to reach the value of the full feature space quickly.



(a) (b)

Figure 12: (Left) Maximum deviation D on the Adult Income dataset as a function of certification set radius r for DT, LR, and GAM. (Right) Upper bound on maximum deviation of f, a Random Forest, trained on the Adult Income dataset.

Relationships with accuracy and robust accuracy In Figure 13, we show maximum deviation as a function of test set accuracy for the DT, LR, and GAM models shown in Figure 11 (the LR and GAM models are listed in Tables 5 and 6). Broadly, the plots show two regimes: one where accuracy increases and maximum deviation increases moderately or not at all, and one where accuracy stalls while maximum deviation increases. The latter is less desirable as it suggests increasing safety risks without a gain in accuracy. The last branch of the DT curve actually decreases in accuracy, indicating overfitting, while maximum deviation is high.

We also consider the relationship of maximum deviation to *robust accuracy*. Following Wong and Kolter [17], robust loss for a pair (x, y) is defined as the worst-case loss over an ℓ_∞ ball centered at

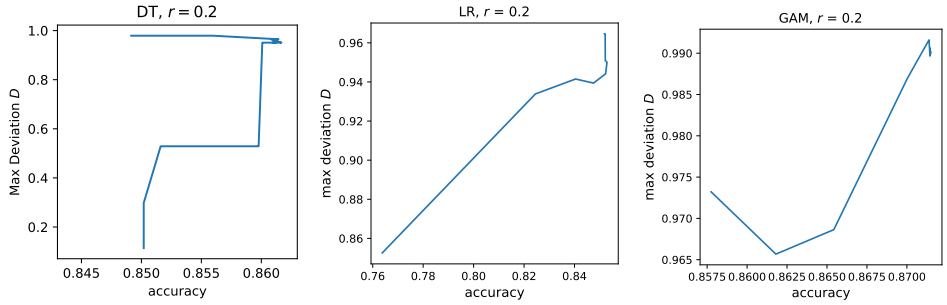


Figure 13: Maximum deviation D (at certification set radius $r = 0.2$) vs. test set accuracy on the Adult Income dataset.

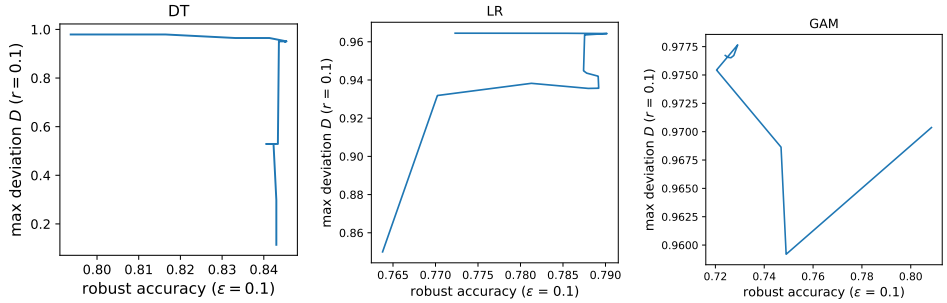


Figure 14: Maximum deviation D vs. robust accuracy ($\epsilon = 0.1$, $r = 0.1$) on the Adult Income dataset.

x,

$$\max_{\|\Delta\|_\infty \le \epsilon} L(f(x+\Delta), y), \tag{16}$$

and robust accuracy is therefore 1 minus the average robust 0-1 loss over a dataset. While Wong and Kolter [17] focus on bounding robust loss for feedforward neural networks with ReLU activations, we find that the results in Section 4.2 apply to computing robust loss (16) exactly for LR and GAM models. Specifically, for 0-1 loss and ℓ_∞ balls, the separable optimization (8) applies, and the worst case is obtained by minimizing f when the label y is positive and maximizing f when y is negative.

The resulting robust accuracy values for DT, LR and GAM are plotted in Figure 14 in a similar fashion as Figure 13. Here we set $\epsilon = 0.1$ and $r = 0.1$ as well in computing maximum deviation. The DT plot shows maximum deviation increasing with model complexity while robust accuracy is stable up to a point. In the subsequent regime, when there are a large number of leaves, model robustness reduces while deviation remains high. The LR plot begins similarly to the one in Figure 13 in that robust accuracy increases along with maximum deviation, but then it stalls and decreases for maximum deviation above 0.94. In the GAM plot, robust accuracy actually decreases with the `max_bins` parameter, i.e., the curve goes from right to left.

Breakdown by leaves of f_0 In Figures 15–18, we plot the maximum log-odds achieved by model f (max on RHS of (7)), the minimum log-odds achieved by f, and the reference model log-odds $g(y_{0m})$ over each leaf of the decision tree reference model in Figure 8. Plots are on the log-odds scale to show the deviations more clearly, including those that would be compressed by the nonlinear logistic function $g^{-1}(z) = 1/(1+e^{-z})$. Figures 15 and 17 show the dependence on the certification set radius r while Figures 16 and 18 show dependence on the smoothness parameters for LR and GAM. These figures provide a more granular picture corresponding to the summary in Figure 11 and support the trends seen there. In Figures 15 and 17, there are jumps at $r = 1$ because this is the smallest radius that permits the values of categorical features of test set points (the ball centers in (2)) to change to any other value. (Recall that categorical features are one-hot encoded into binary-valued features.) In Figure 17, the GAM achieves the limiting deviations corresponding to $r = \infty$ ($\mathcal{C} = \mathcal{X}$,

dashed lines) no later than $r = 1.2$. In Figure 15, the LR model achieves the lower limit on log-odds as soon as $r > 1$ but the upper limit is not achieved for most leaves.

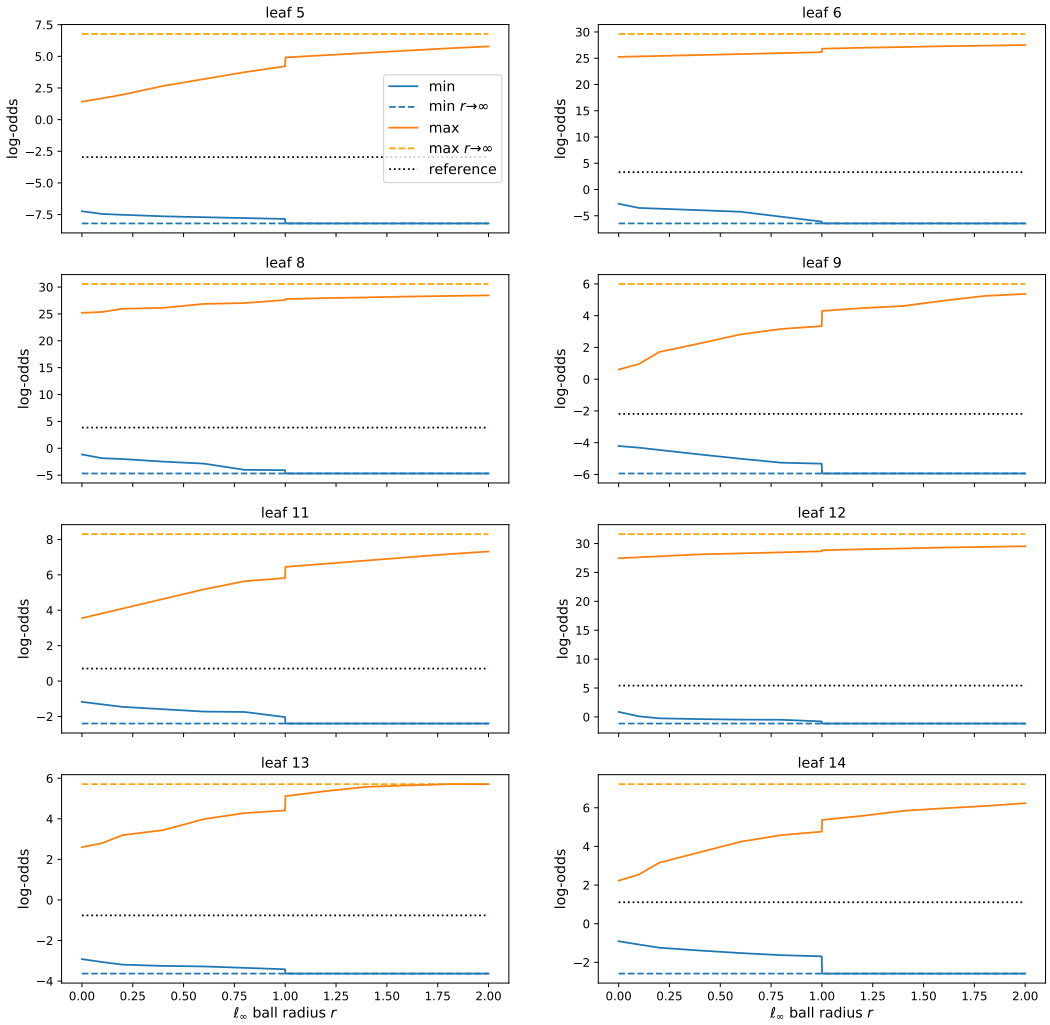


Figure 15: Minimum and maximum predicted log-odds for a logistic regression model with inverse ℓ_1 penalty $C = 0.01$, as a function of certification set size (radius r) and broken down by leaves of the decision tree reference model.

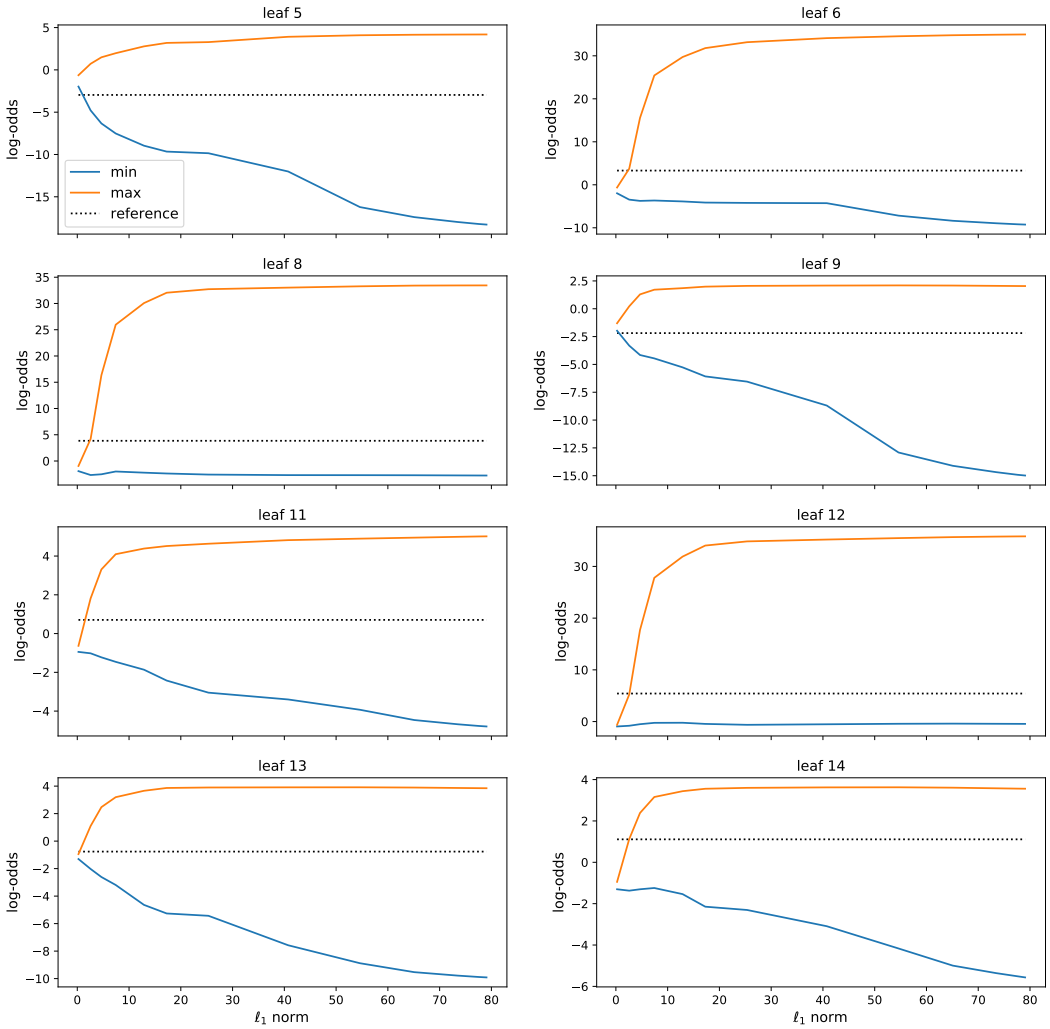


Figure 16: Minimum and maximum predicted log-odds for logistic regression models with different ℓ_1 penalties C, broken down by leaves of the decision tree reference model. The certification set ℓ_∞ ball radius is $r = 0.2$.

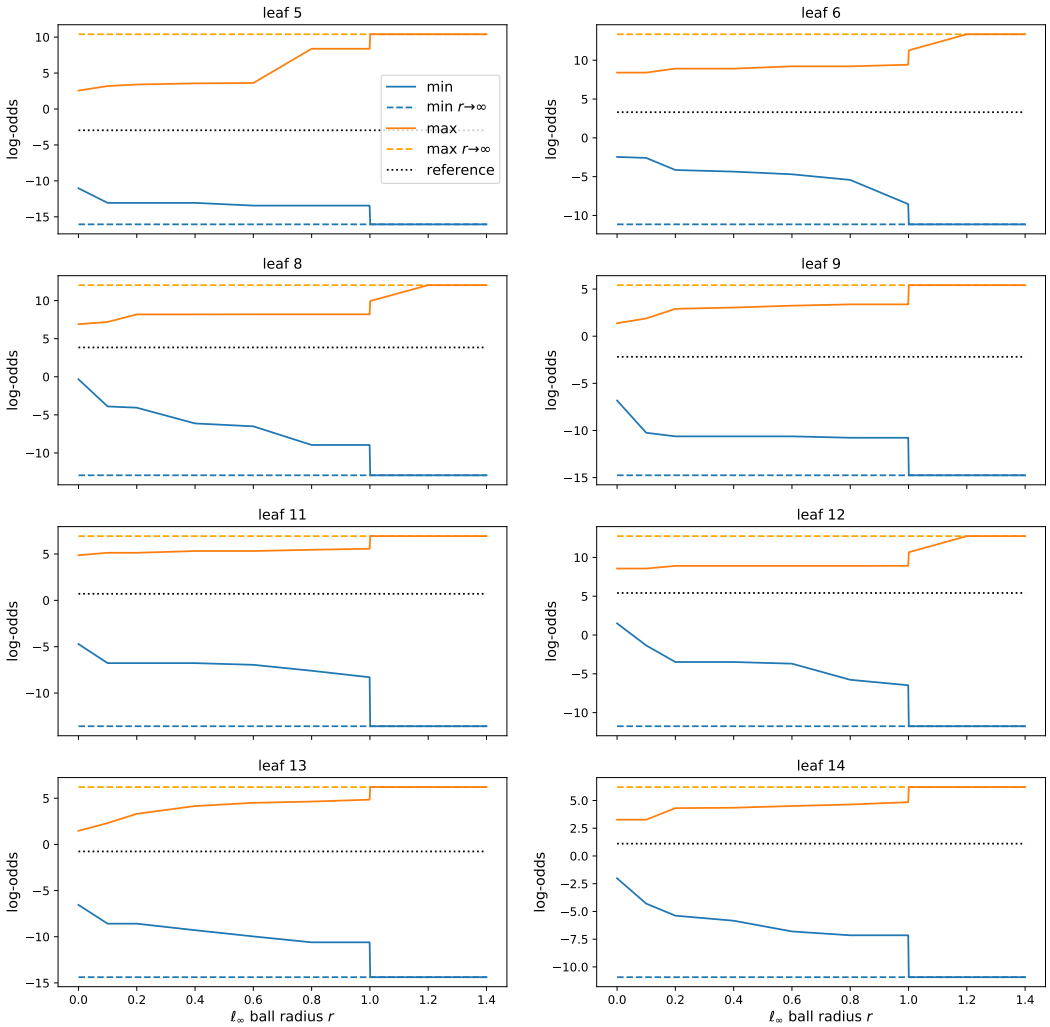


Figure 17: Minimum and maximum predicted log-odds for an Explainable Boosting Machine with `max_bins` $= 8$, as a function of certification set size (radius r) and broken down by leaves of the decision tree reference model.

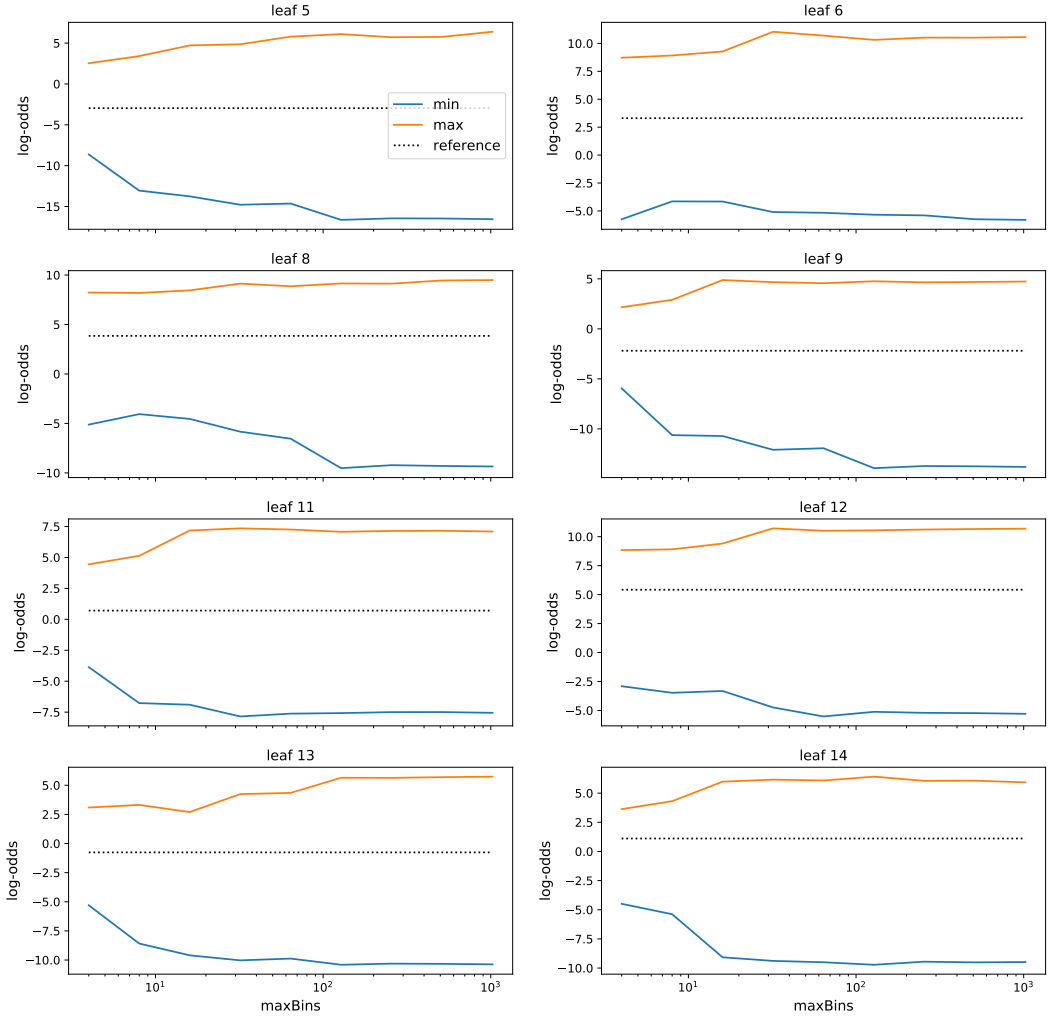


Figure 18: Minimum and maximum predicted log-odds for Explainable Boosting Machines with different `max_bins` values, broken down by leaves of the decision tree reference model. The certification set ℓ_∞ ball radius is $r = 0.2$.

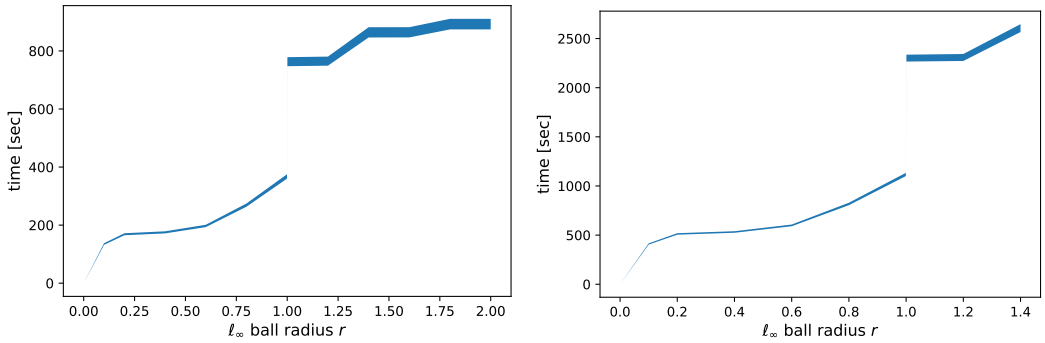


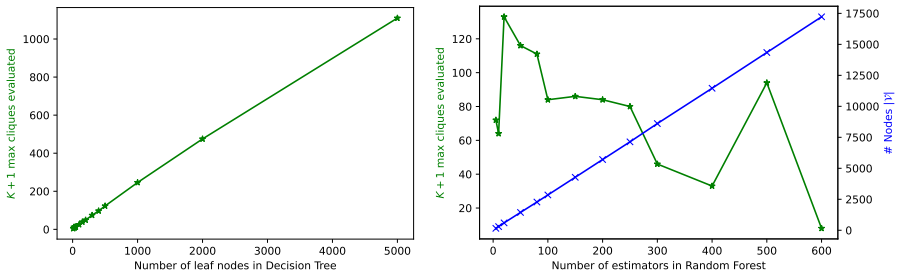
Figure 19: Time to compute maximum deviation for logistic regression models (left) and Explainable Boosting Machines (right) on the Adult Income dataset as a function of certification set size (radius r). The filled-in region shows the min-max variation with model complexity (ℓ_1 norm for LR, `max_bins` for EBM).

Running time Figure 19 shows the time required to compute the maximum deviation for LR and GAM on the Adult Income dataset. The same observations apply as in Figure 7 earlier.

Primal bounds for RF The primal bound for max-deviation shown in Figure 2 for RF is updated each time the algorithm finds a $K+1$ partite, i.e. has examined all trees in the Random Forest. The max-deviation computed for such a partite is a valid deviation. To prove if its optimal, Algorithm 1 needs to run to completion which may not be feasible. Figure 2 shows that as the RF models get larger (number of partites increase), it gets harder to find primal solutions.

Maximal cliques evaluated for DT, RF To investigate the effectiveness of pruning by bounds in Algorithm 1, we investigate the number of times all the decision trees in the Random Forest have to be processed. This represents number of times the state could not be pruned and needed to be evaluated fully.

Figure 20 shows two aspects at play. (a) Pruning by bound is effective in restricting the search space more so for Random Forests than for decision trees, and (b) for larger graphs, more time is spent in computing bounds in Eq. (11).



(a) Decision Tree (b) Random Forest

Figure 20: Effectiveness of pruning by bound for tree-based models

Feature combinations that maximize deviation In Tables 7 and 8, we report feature values that maximize deviation over selected leaves of the DT reference model, for LR and GAM respectively. For Table 7, we have chosen the minimum log-odds over leaf 6 (corresponding to Figure 15, leaf 6, blue curve), which is one of the two leaves m in (7) that maximize the deviation overall (the other being leaf 8). For Table 8, the minimum log-odds over leaf 12 is chosen (corresponding to Figure 17, leaf 12, blue curve) because this maximizes the deviation overall for most values of r. The tables show the 6 features that contribute most to the minimum log-odds. These contributions are again determined using (8) (with $\max$ replaced by $\min$); since the minimum log-odds occurs in one of

the ℓ_∞ ball-leaf intersections and this intersection is a Cartesian product, the decomposition in (8) applies. The contribution of feature j is then $\min_{x_j \in \mathcal{S}_j} f_j(x_j)$. As in Tables 1 and 4, we take an average of the contributions over r to give a single ranking of features for all r.

r	EducationNum	HoursPerWeek	Age	MaritalStatus	Occupation	Relationship
0.0	9.0	15.0	23.0	Never-married	Sales	Own-child
0.1	4.7	33.8	26.6	Never-married	Transport-moving	Not-in-family
0.2	4.5	32.5	25.3	Never-married	Transport-moving	Not-in-family
0.4	4.0	30.1	22.5	Never-married	Transport-moving	Not-in-family
0.6	3.5	27.6	19.8	Never-married	Transport-moving	Not-in-family
0.8	1.0	26.1	17.0	Never-married	Farming-fishing	Not-in-family
0.999	1.0	7.7	17.0	Never-married	Other-service	Own-child
1.001	1.0	1.0	17.0	Never-married	Other-service	Own-child
1.2	1.0	1.0	17.0	Never-married	Other-service	Own-child
1.4	1.0	1.0	17.0	Never-married	Other-service	Own-child
1.6	1.0	1.0	17.0	Never-married	Other-service	Own-child
1.8	1.0	1.0	17.0	Never-married	Other-service	Own-child
2.0	1.0	1.0	17.0	Never-married	Other-service	Own-child
∞	1.0	1.0	17.0	Never-married	Other-service	Own-child

Table 7: Feature values that minimize log-odds for a logistic regression model ($C = 0.01$) over leaf 6 of the decision tree reference model. The 6 features that contribute most to the minimum are shown as a function of certification set radius r.

As r increases, the predominant trend of the values of continuous features is toward extremes of the domain $\mathcal{X}$, depending on the sign of the corresponding LR coefficient w_j or shape of the GAM function f_j. For example, EducationNum (education on an ordinal scale), hours per week, and age decrease toward minimum values, while capital gain occupies the minimal interval permitted for leaf 12 (see Figure 8). (These examples make sense since the log-odds of high income is being minimized.) This movement toward extremes is expected in the LR case because the functions $w_j x_j$ are either increasing or decreasing, and it is also true for GAM if the function f_j is mainly increasing or decreasing. The values sometimes change abruptly in the opposite direction, for example hours per week in both Tables 7, 8, and age in the latter. These abrupt changes are due to the minimum jumping from one ball in (2) to another as r increases, but the overall trend eventually prevails. For categorical features, the trend is toward values that minimize $f_j(x_j)$, e.g., *Never-married* marital status, *Without-pay* work class. While the contribution of each of these features to minimizing log-odds may be limited, together they do add up.

r	CapitalLoss	Age	HoursPerWeek	WorkClass	CapitalGain	Race
0.0	0	29.0	40.0	Private	7298	White
0.1	[0 40]	[53.6 56.4]	[18.8 20.5]	?	[5095 5119]	White
0.2	[0 78]	[23.3 23.5]	[4.5 9.5]	State-gov	[5095 5119]	White
0.4	[0 78]	[20.5 23.5]	[2.1 11.9]	State-gov	[5095 5119]	White
0.6	[0 78]	[28.8 29.5]	[30.6 31.5]	Private	[5095 5119]	Asian-Pac-Islander
0.8	[1598 1759]	[20.1 23.5]	[30.1 31.5]	State-gov	[5095 5119]	White
0.999	[1598 1759]	[21.4 23.5]	[27.7 31.5]	Private	[5095 5119]	Amer-Indian-Eskimo
1.001	[1598 1759]	[17. 23.5]	[11.6 20.5]	Without-pay	[5095 5119]	Amer-Indian-Eskimo
1.2	[1598 1759]	[17. 23.5]	[9.2 20.5]	Without-pay	[5095 5119]	Amer-Indian-Eskimo
1.4	[1598 1759]	[17. 23.5]	[6.7 20.5]	Without-pay	[5095 5119]	Amer-Indian-Eskimo
∞	[1598 1759]	[17. 23.5]	[1. 20.5]	Without-pay	[5095 5119]	Amer-Indian-Eskimo

Table 8: Feature values that minimize log-odds for an Explainable Boosting Machine (`max_bins` $= 8$) over leaf 12 of the decision tree reference model. The 6 features that contribute most to the minimum are shown as a function of certification set radius r. For $r > 0$, the minimizing values of continuous features form an interval because the corresponding functions f_j are piecewise constant.

A notable exception to the trend toward extremes is capital loss in Table 8. This was discussed in the "Identification of an artifact" example in Section 5.

Given the results in Tables 7 and 8, one question that arises is whether the feature combinations are indeed possible, if not the ones for $r \to \infty$, then at least for some finite value of r. For the top

features shown in the two tables, while some combinations may appear improbable (for example, EducationNum $= 1$ and 1 hour per week), we submit that none appear *impossible*. However, if one considers features beyond the top 6, then some "impossible" combinations do occur (e.g., a female husband), although the contributions of these features to the minimum log-odds are much less. We touch upon this issue in Appendix C.

The next question one might consider is the implication of these maximal deviations. From Figure 8, it is seen that leaf 6 classifies individuals with high capital gains as high income with high probability (0.965). Leaf 12 adds the attributes of married status and high education, and hence classifies as high income with even higher probability (0.996). At the same time, the feature values in Tables 7 and 8, which minimize log-odds for LR and GAM, also make sense according to basic domain knowledge. For example, few hours per week and young age are associated with lower income, as are *Without-pay* work class and *Amer-Indian-Eskimo* race in the United States. When these conflicting associations occur in combination and the combination does not appear impossible, the question may be which one prevails. Such a question might be resolvable by a domain expert. Alternatively, the disagreement between models f and f_0 on the extreme examples in Tables 7, 8 may be reason to be cautious about using either of the models in these cases. This might lead to a way of combining the models or abstaining from prediction altogether. Lastly, the anomalously low region in the CapitalLoss function identified in Table 8 is a clear, concrete example where further investigation is warranted.

D.4 Lending Club dataset

This dataset consists of 2.26 million rows with 14 features on loans. The target variable is whether a loan will be paid-off or defaulted on. Features describe the terms of the loan, e.g. duration, grade, purpose, etc. and borrower financial information such as credit history and income. For this case study, we consider a loan approval scenario using only information available at the time of application. In particular, we exclude the feature 'total_pymnt' (total payment over time on the loan), which becomes known at essentially the same time as the target variable. (When 'total_pymnt' is included as a feature, the prediction task becomes easy and accuracies in the high 90% range are possible.)

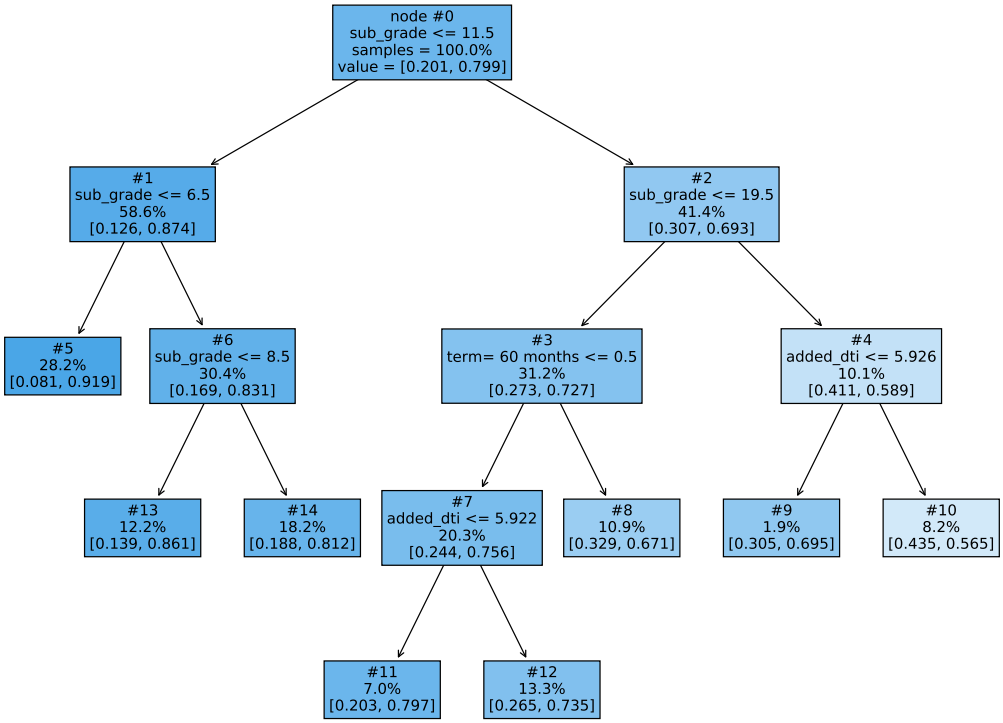


Figure 21: Decision tree reference model with 8 leaves for the Lending Club dataset.

Reference model Figure 21 depicts the 8-leaf DT reference model for the Lending Club dataset. Most of the splits partition the 'sub_grade' feature, which is a measure of the quality of the loan (0–34 range, lower is better). Node 3 differentiates between 60-month terms and 36-month terms (the only alternative), while nodes 4 and 7 split on 'added_dti' (added debt-to-income ratio), which is the ratio between 12 months worth of the loan's payment installments and the borrower's annual income. While the structure of the DT agrees with domain knowledge (lower 'sub_grade' and lower 'added_dti' correlate with higher repayment probability), the test set accuracy of 79.8% is no better than that of the trivial predictor that always returns the majority class of "paid off". The DT's AUC of 0.689 however does indicate an improvement over the trivial predictor.

LR and GAM models Tables 9 and 10 show the statistics of the LR and GAM classifiers that were trained on the Lending Club data. Similar to the DT reference model, the difference compared to Tables 5, 6 for the Adult Income dataset is that the accuracies remain no better than that of the trivial predictor, while the AUC does not show much increase either. These statistics suggest that the prediction task is difficult with the features available. Figures 22 and 23 display plots for the LR model with $C = 0.01$ and GAM with $\texttt{max_bins} = 8$, which are again chosen as representative models. The GAM in particular shows sensible monotonic behavior as functions of 'sub_grade', 'int_rate' (interest rate), 'dti' (debt-to-income ratio), etc., despite the unimpressive accuracy.

C	nonzeros	ℓ_1 norm	accuracy	AUC
1e-4	1	0.4	0.798	0.693
3e-4	2	0.6	0.799	0.695
1e-3	6	1.0	0.799	0.702
3e-3	8	1.3	0.799	0.702
1e-2	12	1.6	0.800	0.702
3e-2	17	2.1	0.799	0.703
1e-1	22	3.1	0.799	0.703
3e-1	24	3.7	0.799	0.703
1e+0	26	4.1	0.799	0.703
3e+0	26	4.2	0.799	0.703

Table 9: Number of nonzero coefficients, ℓ_1 norm of coefficients, test set accuracy, and AUC for logistic regression models on the Lending Club dataset as a function of inverse ℓ_1 penalty C.

max_bins	accuracy	AUC
4	0.798	0.696
8	0.798	0.703
16	0.799	0.704
32	0.799	0.705
64	0.799	0.705
128	0.799	0.705
256	0.799	0.705
512	0.799	0.705
1024	0.799	0.705

Table 10: Test set accuracy and AUC for Explainable Boosting Machines on the Lending Club dataset as a function of `max_bins` parameter.

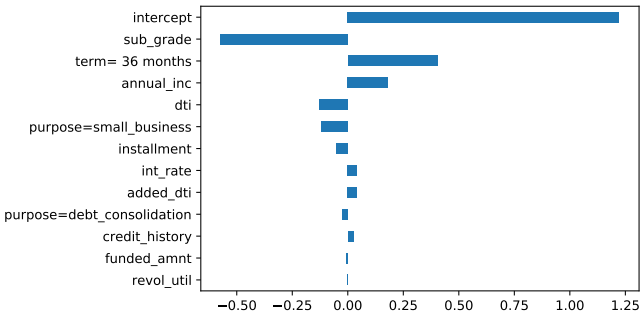


Figure 22: Coefficient values of the logistic regression model with $C = 0.01$ (12 nonzeros) for the Lending Club dataset.

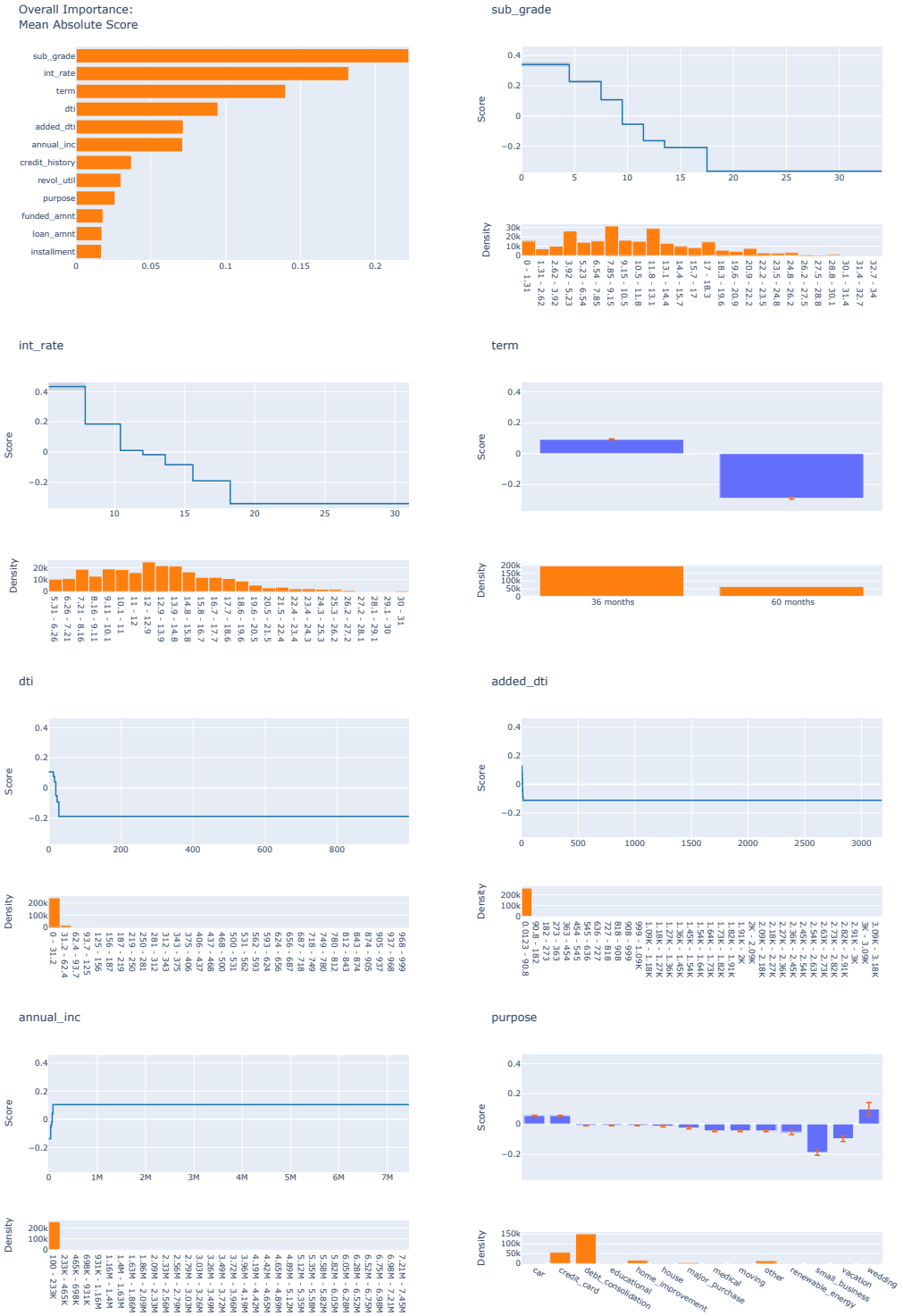


Figure 23: Feature importances and selected univariate functions f_j for the Explainable Boosting Machine with `max_bins` $= 8$ on the Lending Club dataset.

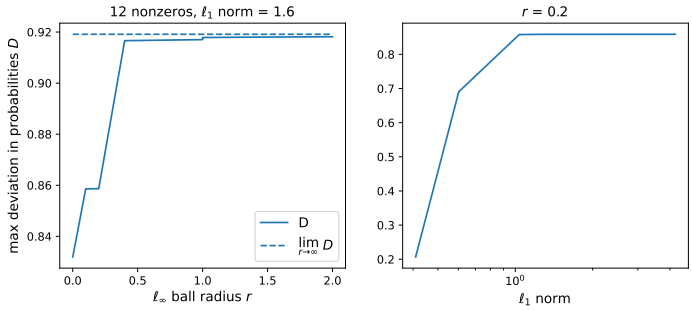


Figure 24: Maximum deviation D for logistic regression models on the Lending Club dataset as a function of certification set size (radius r) and model smoothness (ℓ_1 norm).

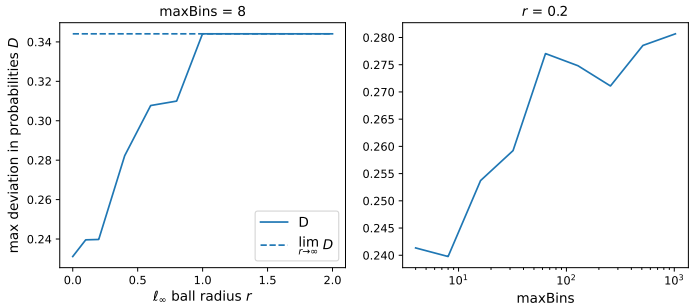
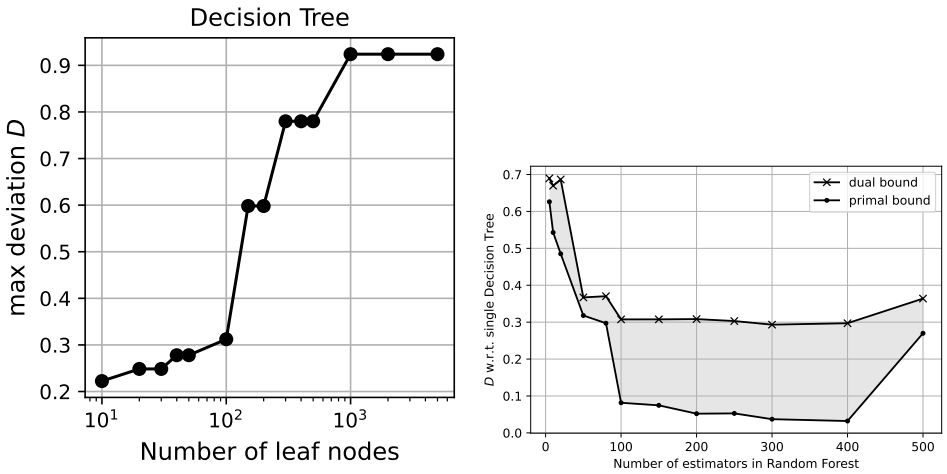


Figure 25: Maximum deviation D for Explainable Boosting Machines on the Lending Club dataset as a function of certification set size (radius r) and model smoothness (`max_bins` parameter).



(a) Decision Tree

(b) Random Forest

Figure 26: Maximum deviations computed for tree and tree ensembles on the Lending Club dataset

Maximum deviation summary Figures 24–26 show maximum deviation as functions of certification set radius r and model complexity parameters, in a similar manner as Figure 11 for the Adult Income dataset. The qualitative patterns are similar to before: increasing maximum deviation in all cases except with the number of RF estimators in Figure 26b, where the upper bound is stable around 0.7. A major quantitative difference is that the maximum deviations for the GAM in Figure 25 are much lower than for the other models, in particular LR in Figure 24. This is likely due to the fact that the GAM functions f_j in Figure 23 are bounded while still being monotonic, unlike the linear functions $w_j x_j$ in the LR model.

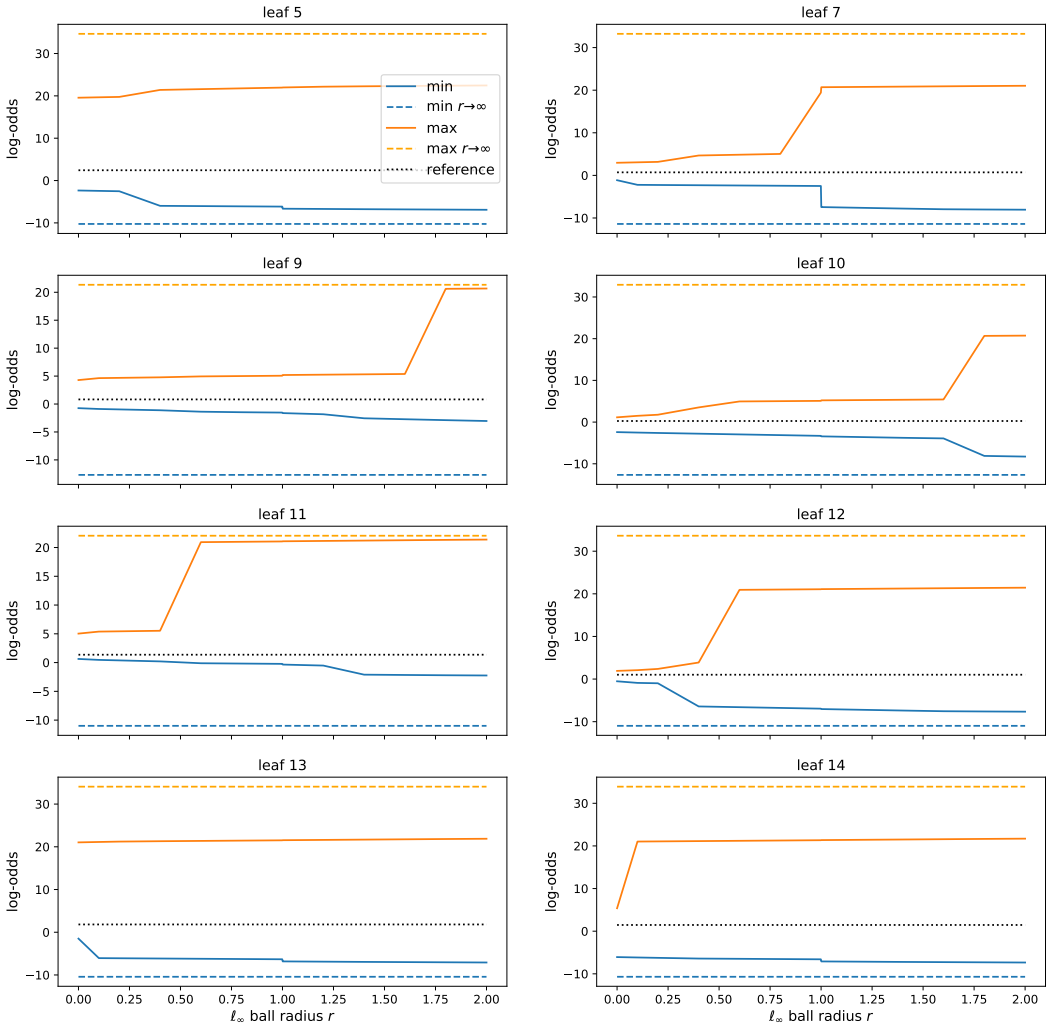


Figure 27: Minimum and maximum predicted log-odds for a logistic regression model with inverse ℓ_1 penalty $C = 0.01$ on the Lending Club dataset, as a function of certification set size (radius r) and broken down by leaves of the decision tree reference model.

Breakdown by leaves of f_0 Figures 27–30 show a breakdown of the deviations for LR and GAM by leaves of the reference model, similar to Figures 15–18 and again on the log-odds scale. One difference is that in Figure 27, the deviations for finite r do not come close to their $r \to \infty$ counterparts in most cases. In Figure 29 however, the $r \to \infty$ values are all attained when r is slightly greater than 1.

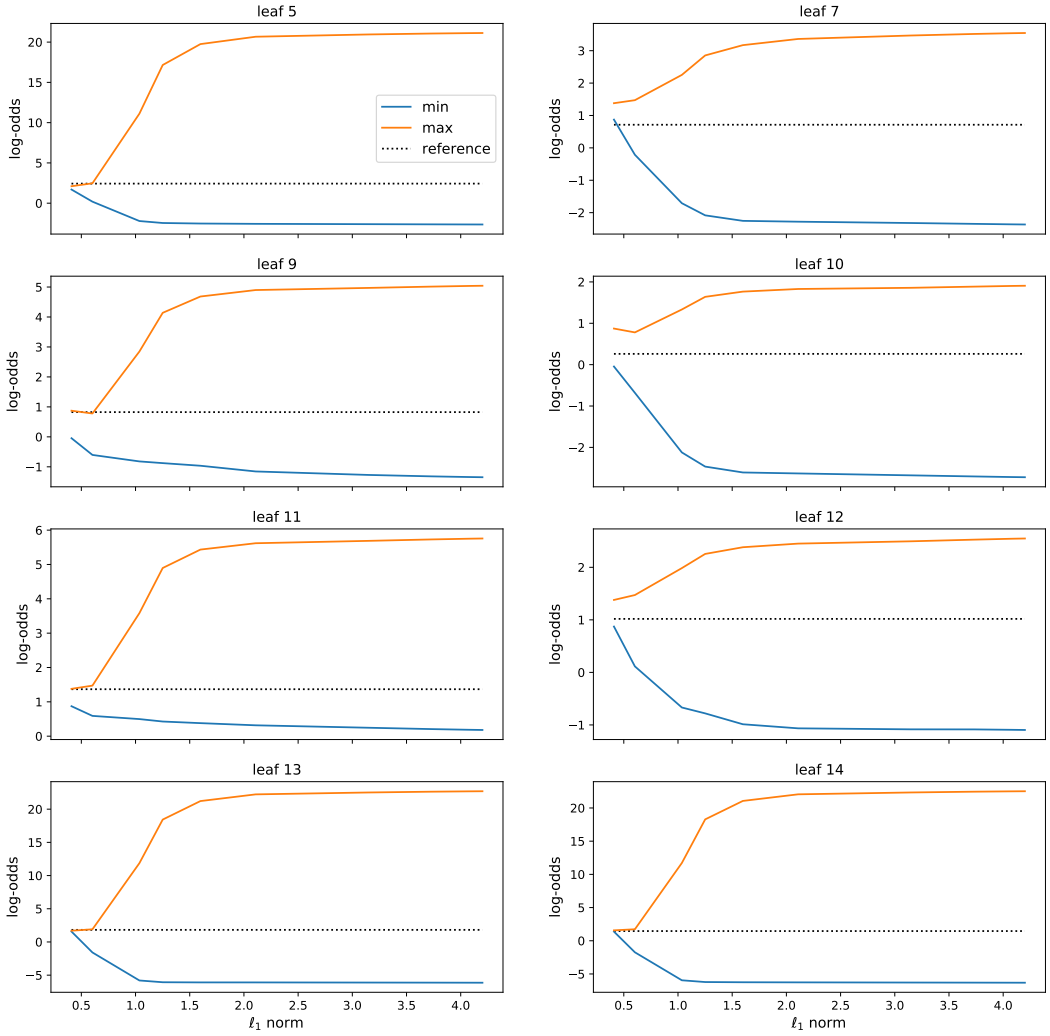


Figure 28: Minimum and maximum predicted log-odds for logistic regression models with different ℓ_1 penalties C on the Lending Club dataset, broken down by leaves of the decision tree reference model. The certification set ℓ_∞ ball radius is $r = 0.2$.

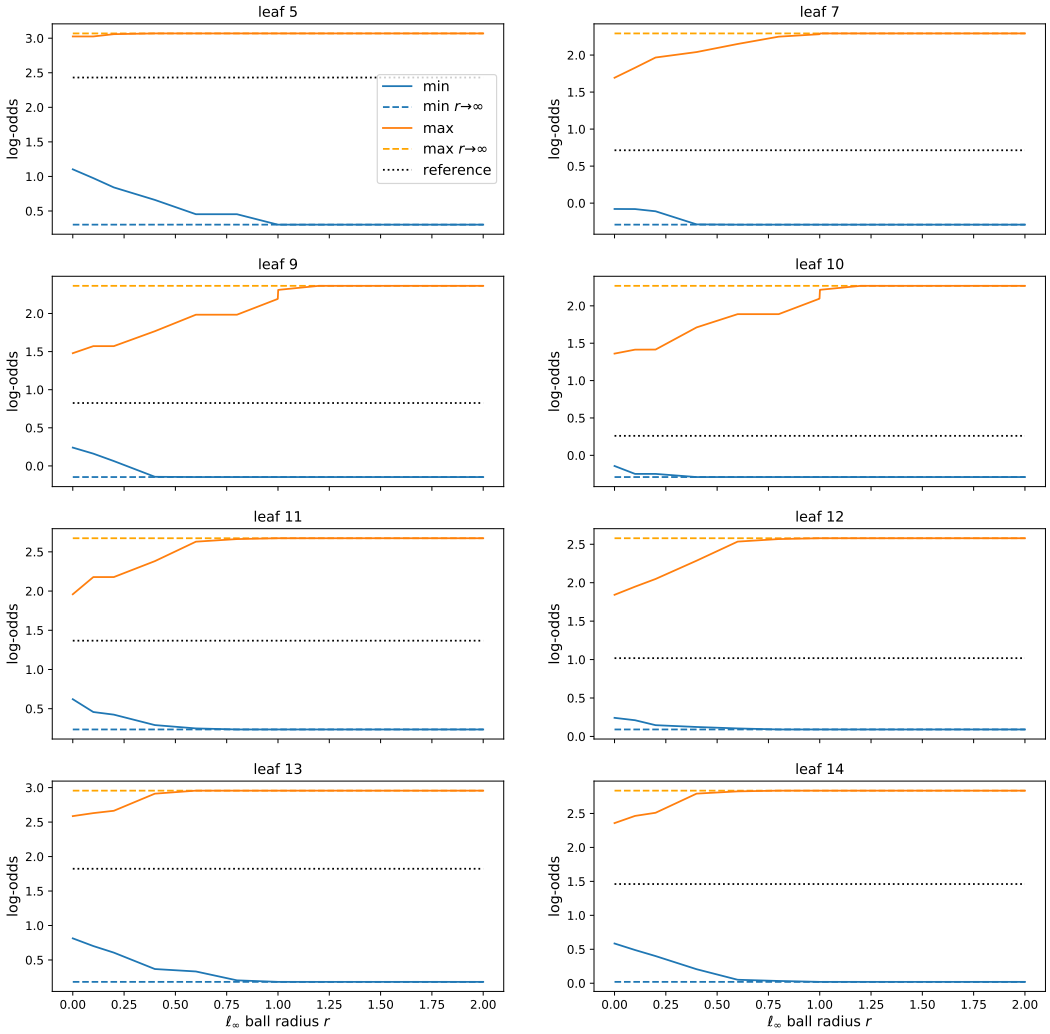


Figure 29: Minimum and maximum predicted log-odds for an Explainable Boosting Machine with `max_bins` $= 8$ on the Lending Club dataset, as a function of certification set size (radius r) and broken down by leaves of the decision tree reference model.

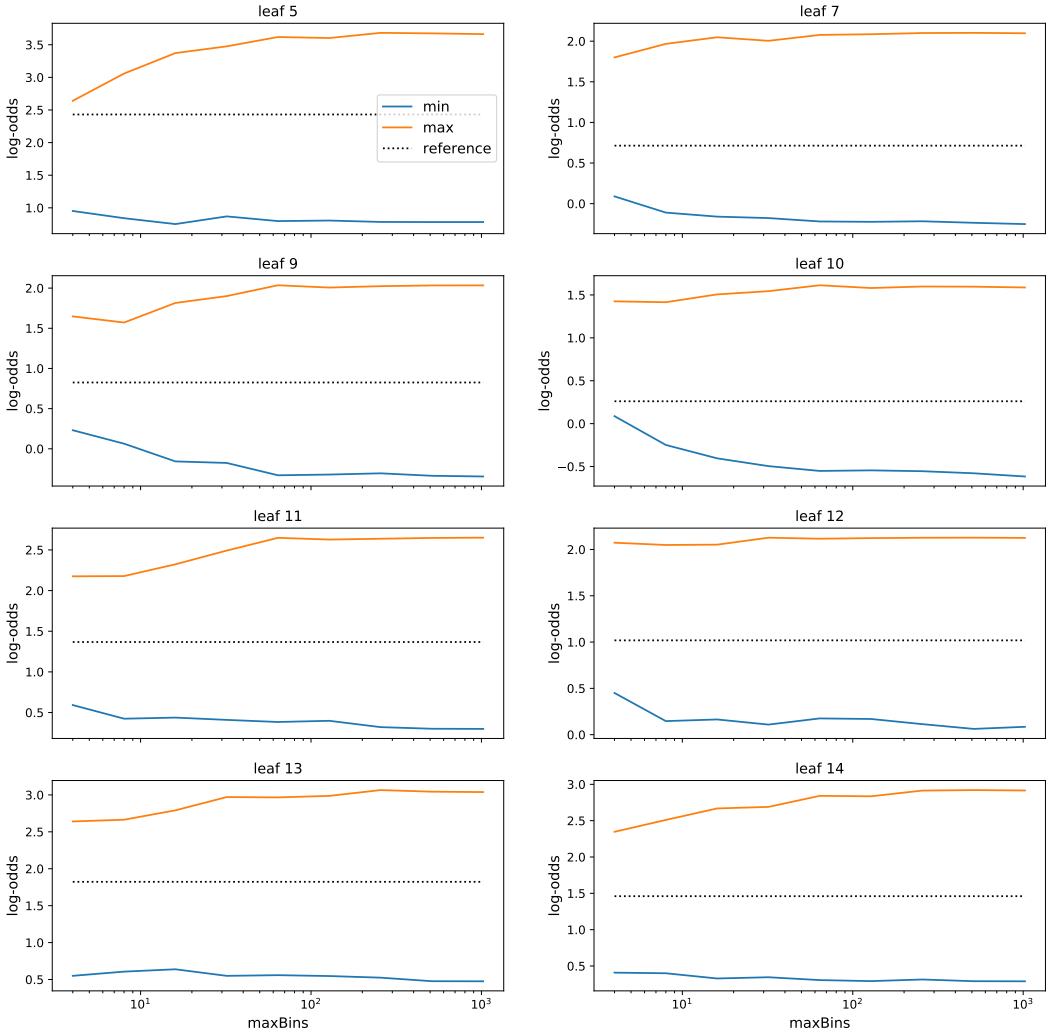


Figure 30: Minimum and maximum predicted log-odds for Explainable Boosting Machines with different `max_bins` values on the Lending Club dataset, broken down by leaves of the decision tree reference model. The certification set ℓ_∞ ball radius is $r = 0.2$.

Running time, maximal cliques evaluated Figure 31 shows the time required to compute the maximum deviation for LR and GAM on the Lending Club dataset. Figure 32 shows the number of $K+1$-maximal cliques evaluated for DT and RF as well as the number of nodes in the graph. The observations are the same as in Figures 19 and 20.

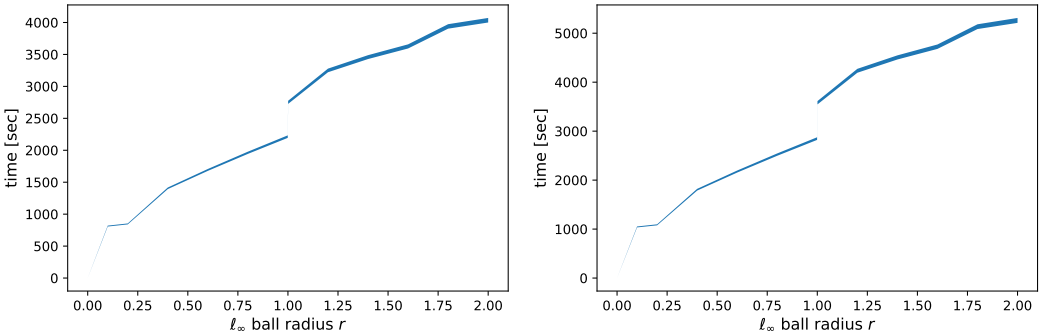
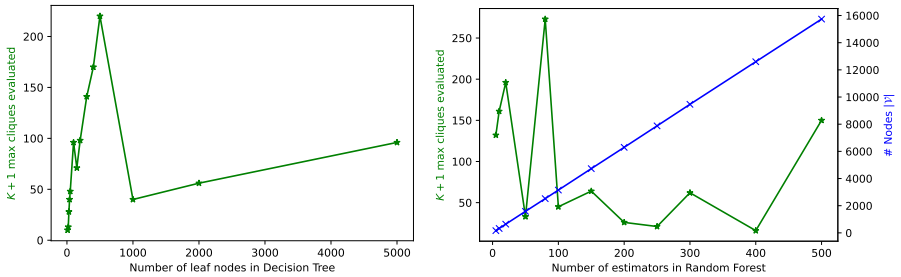


Figure 31: Time to compute maximum deviation for logistic regression models (left) and Explainable Boosting Machines (right) on the Lending Club dataset as a function of certification set size (radius r). The filled-in region shows the min-max variation with model complexity (ℓ_1 norm for LR, `max_bins` for EBM).



(a) Decision Tree (b) Random Forest

Figure 32: Effectiveness of pruning by bound for tree-based models on the Lending Club dataset

Feature combinations that maximize deviation Tables 11 and 12 present the feature values that maximize deviation for LR and GAM respectively. For both tables, the minimum log-odds over leaf 5 of the DT reference model is selected (corresponding to Figures 27 and 29, leaf 5, blue curve) because this choice maximizes the deviation overall for most values of r.

The previous trend toward extreme feature values that minimize log-odds also holds here as r increases. For example, debt-to-income ratios increase to outlier values above 100%, annual income drops to the minimum of $\$100$, the term changes to 60 months in Table 11 for $r > 1$, and the purpose changes to small business, the category with the lowest log-odds in Figure 23. The decrease in income and increases in debt-to-income ratios are qualitatively in accordance with each other. However, these quantities are related to each other by deterministic formulas (at least in theory) that also involve the interest rate and installment amount. It is not clear whether the values in Tables 11 and 12 violate these relationships. This may be an instance that could benefit from constraints on possible feature combinations, as briefly mentioned in Appendix C.

r	dti	annual_inc	term	purpose	int_rate	credit_history
0.0	505	1700	36 months	credit_card	7.4	2557
0.1	506	100	36 months	credit_card	6.9	2283
0.2	507	100	36 months	credit_card	6.4	2009
0.4	884	100	36 months	debt_consolidation	10.1	3897
0.6	886	100	36 months	debt_consolidation	9.1	3349
0.8	889	100	36 months	debt_consolidation	8.2	2801
0.999	891	100	36 months	debt_consolidation	7.2	2256
1.001	891	100	60 months	small_business	7.2	2251
1.2	893	100	60 months	small_business	6.3	1706
1.4	895	100	60 months	small_business	5.3	1158
1.6	898	100	60 months	small_business	5.3	1095
1.8	900	100	60 months	small_business	5.3	1095
2.0	902	100	60 months	small_business	5.3	1095
∞	999	100	60 months	small_business	5.3	1095

Table 11: Feature values that minimize log-odds for a logistic regression model ($C = 0.01$) over leaf 5 of the decision tree reference model for the Lending Club dataset. The 6 features that contribute most to the minimum are shown as a function of certification set radius r.

r	term	int_rate	dti	purpose	annual_inc	added_dti
0.0	60 months	9.9	46.4	debt_consolidation	35000	25.5
0.1	60 months	[10.4 11.]	[28.6 30.9]	debt_consolidation	[31800 38001]	[10. 11.2]
0.2	60 months	[12. 13.1]	[27.8 31.8]	debt_consolidation	[36600 38001]	[10. 11.7]
0.4	60 months	[13.6 14.3]	[27.8 30.4]	small_business	[100 38001]	[12.7 15.3]
0.6	60 months	[15.6 16.3]	[27.8 36.1]	small_business	[19801 38001]	[12.7 15.]
0.8	60 months	[15.6 16.4]	[27.8 28.4]	small_business	[14402 38001]	[12.7 15.7]
0.999	60 months	[18.2 18.4]	[27.8 38.8]	small_business	[33069 38001]	[12.7 14.5]
1.001	60 months	[18.2 18.8]	[27.8 35.3]	small_business	[935 38001]	[12.7 18.5]
1.2	60 months	[18.2 20.2]	[27.8 33.3]	small_business	[7602 38001]	[12.7 18.1]
1.4	60 months	[18.2 21.2]	[27.8 35.6]	small_business	[100 38001]	[12.7 20.4]
1.6	60 months	[18.2 19.2]	[27.8 29.9]	small_business	[100 38001]	[12.7 24.5]
1.8	60 months	[18.2 26.1]	[27.8 28.5]	small_business	[100 38001]	[12.7 24.4]
2.0	60 months	[18.2 27.1]	[27.8 30.7]	small_business	[100 38001]	[12.7 26.6]
∞	60 months	[18.2 31.]	[27.8 999.]	small_business	[100 38001]	[12.7 3179.3]

Table 12: Feature values that minimize log-odds for an Explainable Boosting Machine (`max_bins` $= 8$) over leaf 5 of the decision tree reference model for the Lending Club dataset. The 6 features that contribute most to the minimum are shown as a function of certification set radius r. For $r > 0$, the minimizing values of continuous features form an interval because the corresponding functions f_j are piecewise constant.