LOCAL SENSITIVITY IN DECISION TREE ENSEMBLES WITH APPLICATIONS TO EXPLAINABILITY AND MONITORING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision tree ensembles (DTEs) are widely used in applications ranging from finance to healthcare, motivating the need to formally reason about their predictions. Recent work has studied *sensitivity* as a global property of DTEs: given a set of features, the goal is to determine whether there exist two inputs that differ only on these features but receive different classifications. In this work, we introduce and study *local sensitivity*, where one of the two inputs is fixed. Given an input and a set of features, we ask whether changing only those features can produce another input with a different classification. We give a detailed complexity analysis of this problem, showing that it is NP-complete and W[1]-hard when parameterized by the number of features that can be perturbed, while also identifying polynomial-time tractable cases and an XP algorithm for the general problem. We further show that local sensitivity provides a common framework for reasoning about local robustness and contrastive explainability. We develop practical enumeration- and MILP-based algorithms for local sensitivity and implement them. Our experiments show that local sensitivity can be solved efficiently in practice, that it provides a competitive approach for computing contrastive explanations, and that it can detect robustness violations beyond those observable by competing techniques for runtime monitoring.

1 INTRODUCTION

Decision tree ensembles (DTEs) are simple yet powerful machine-learning models that are widely used across a range of domains, including healthcare (e.g., Ghiasi & Zendehboudi (2021); Kelarev et al. (2012)) and finance (e.g., Chang et al. (2018); Madaan et al. (2021)). Their popularity stems in part from their strong predictive performance together with their structured, rule-based nature, which makes them comparatively amenable to interpretation and formal analysis. Consequently, there has been considerable interest in specifying and verifying properties of DTEs, including robustness, explainability, fairness, and sensitivity. Such properties can broadly be divided into *local* and *global* properties. Local properties concern the behaviour of a model around a given input, whereas global properties concern its behaviour over the entire input space. This distinction is important both conceptually and computationally: local properties answer input-specific questions and are often more amenable to algorithmic analysis, while global properties require reasoning about the model as a whole and can be substantially harder to scale. A considerable body of work has therefore focused on local properties of DTEs, notably robustness (Chen et al., 2019) and explainability (Izza et al., 2025). Local robustness asks whether the prediction of a given input remains unchanged under a prescribed class of perturbations, while local explainability identifies features, or changes to features, that account for the prediction at that input. These problems have been approached using various techniques, including symbolic reasoning, SAT/SMT solving, and mixed-integer programming.

Sensitivity, in contrast, has recently been studied as a global property of DTEs (Ahmad et al., 2025; Varshney et al., 2026). Given a feature or a set of features, the sensitivity problem asks whether changing only those features can change the classification of the model. A pair of inputs witnessing such a change is called a *sensitive pair*. Since neither endpoint of the pair is fixed in advance, deciding sensitivity requires reasoning over the input space as a whole. This naturally raises the corresponding local question: given a particular input, can changing only a designated set of features

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

produce another input with a different classification? As we show in this work, this local variant is not merely a restricted form of global sensitivity, but a useful primitive that connects several existing problems in verification and explainability.

In this paper, we formalize this notion of *local sensitivity*. Given a DTE $\mathcal{E}$, a data point v, and a set of features $\mathcal{S}$, we ask whether there exists another input obtained by changing (upto a certain distance ε) only the features in $\mathcal{S}$ such that the two inputs form a sensitive pair. We give a detailed complexity analysis of this problem. For ensembles containing trees of depth at least two, we show that deciding local sensitivity is NP-complete, and moreover W[1]-hard when parameterized by $|\mathcal{S}|$. This means this problem is unlikely to be solvable in time $g(|\mathcal{S}|).n^c$, for some computable function g and any natural number c. We complement the lower bound with an XP algorithm, polynomial for every fixed $|\mathcal{S}|$, and show that for ensembles consisting entirely of decision *stumps*, which are commonly used as weak learners in boosting, local sensitivity can be decided in polynomial time even when $|\mathcal{S}|$ is unbounded.

Motivated by these complexity results, we next develop practical algorithms for local sensitivity. We consider two complementary approaches: an enumeration-based procedure and a mixed-integer linear programming (MILP) formulation, and implement both in our tool LOCAL-SENS. Beyond solving local sensitivity directly, we show that the same machinery can be used for two important downstream tasks. First, local sensitivity provides a direct route to computing contrastive explanations, allowing us to recover both subset-minimal and cardinality-minimum explanations. Second, it can be used for monitoring robustness violations around individual inputs, including potential violations whose witnesses may never occur in an observed execution trace. Together, these connections illustrate that local sensitivity is not only a verification problem in its own right, but also a useful notion for reasoning about explainability and monitoring.

We evaluate LOCAL-SENS on a range of DTE benchmarks and study both its computational performance and its usefulness for downstream tasks. Our experiments compare the MILP- and enumeration-based algorithms, evaluate local sensitivity as a method for computing contrastive explanations, and study its use in detecting robustness violations beyond those observable by runtime monitoring. The results show that local sensitivity can be solved efficiently in practice and provides a useful computational basis for these applications. In summary, our main contributions are:

- We introduce *local sensitivity* for decision tree ensembles and show that it generalizes local robustness, while contrastive explanations can be computed using it.
- We prove NP-completeness and W[1]-hardness in $|\mathcal{S}|$, give an XP algorithm, and identify polynomial-time cases,
- We develop two practical algorithms for local sensitivity, based on enumeration and mixed-integer linear programming, and implement them in LOCAL-SENS.
- We demonstrate the utility of local sensitivity through two main applications: contrastive explainability and monitoring. We evaluate these applications on a range of DTE benchmarks.

Related Work. Several verification questions have been considered for DTEs in the literature. Local robustness (whether adversarial perturbations of a given data input can cause misclassification) was shown to be NP-hard in Kantchelian et al. (2016), who also gave an MILP encoding to solve it. Since then, different techniques have been employed such as input-output mappings (Törnblom & Nadjm-Tehrani, 2019), abstract interpretation (Ranzato & Zanella, 2020), dynamic programming (Wang et al., 2020) and state-of-the-art tool uses clique-based approaches to solve this and related problems (Devos et al., 2021; 2024). Sensitivity verification was encoded in Ahmad et al. (2025) using a Pseudo-Boolean encoding and in Naik et al. (2026) using MILP. It is closely related to individual fairness (Dwork et al., 2011; Calzavara et al., 2022), as it measures whether small changes in a subset of features can disproportionately change the model's output.

There has also been significant work on explainability, where different notions such as contrastive, abductive and other explanations have been considered (Ignatiev et al., 2020; Audemard et al., 2023; Izza et al., 2026). To solve these problems, different techniques and encodings (e.g, MaxSAT in (Ignatiev et al., 2022)) have been used and even parameterized complexity for this problem has been investigated (Ordyniak et al., 2024). Finally, the problem of runtime monitoring of ML models was considered in Gupta et al. (2025), where monitors attempt to check robustness on the fly and attempts to raise an alarm whenever it is detected that two similar inputs from the past led to dissim-

ilar outputs. Local sensitivity can be employed to do the same and our experiments show that using local sensitivity also allows to witness violations that have not yet appeared in the input stream.

2 PRELIMINARIES

In a *binary classification problem*, given an input $\mathcal{X} \subseteq \mathbb{R}^d$, the objective is to learn a function $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ that maps every data point $x \in \mathcal{X}$ to a label in $\{0, 1\}$. In general, this problem can also be defined over an arbitrary finite set of labels $\{0, 1, \ldots, C\}$ for some $C \in \mathbb{N}$. Each dimension of $\mathcal{X}$ is called a *feature*, and the set of all features is denoted by $\mathcal{F}$.

Decision Tree. A decision tree is a binary tree, whose each non-leaf node is a linear inequality of the form $f < t$, for some $f \in \mathcal{F}$ and $t \in \mathbb{Q}$. The left child of a node is the subtree that is followed if the inequality evaluates to true, and the right child is the subtree that is followed if the inequality evaluates to false. The leaves of a decision tree are labelled with real numbers. In a multiclass classification problem, the leaves are vectors over $\mathbb{R}^d$. Given a data point, a decision tree traces one path from the root to a leaf, depending on which inequalities are satisfied by the data point. This leaf's value is the *value* assigned by the decision tree to the data point.

Decision Tree Ensemble. Instead of using a single decision tree, in order to improve the accuracy of classification, it is common practice to use an ensemble of decision trees. Where, given a data point, each tree provides a value, and the ensemble aggregates these values to provide a final value for the data point. A Decision Tree Ensemble (DTE) is defined as a function $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ that maps every data point in $\mathcal{X}$ to either 0 or 1. We will use three notations for the different kinds of outputs of a DTE: (i) $\mathcal{E}_1^{\mathsf{raw}}(x)$ is the sum of the values of all trees in the ensemble for a data point x, and $\mathcal{E}_0^{\mathsf{raw}}(x) = -\mathcal{E}_1^{\mathsf{raw}}(x)$, (ii) given a class $c \in \{0, 1\}$, $\mathcal{E}_c^{\mathsf{prob}}(x)$ is the probability that the ensemble assigns to class c for the data point x, this is defined as $\mathcal{E}_c^{\mathsf{prob}}(x) = \mathsf{Sigmoid}(\mathcal{E}_c^{\mathsf{raw}}(x))$. Here $\mathsf{Sigmoid}(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function, and (iii) $\mathcal{E}(x)$ is the final classification of the data point x: $\mathcal{E}(x) = 1$ if $\mathcal{E}_1^{\mathsf{prob}}(x) \geq 0.5$, and $\mathcal{E}(x) = 0$ otherwise.

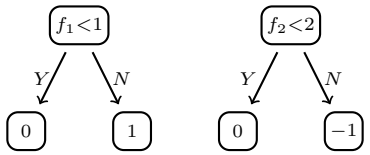


Figure 1: An example DTE with $\mathcal{F} = \{f_1, f_2\}$.

Example 1. Consider the DTE $\mathcal{E}$ in Figure 1 defined over the feature space $\mathcal{F} = \{f_1, f_2\}$. Here each feature ranges over the interval $[0, 3]$, i.e. $\mathcal{X} := [0, 3]^2$. Consider the data point $x := (0.7, 2.1)$. Let $\mathcal{T}_1, \mathcal{T}_2$ denote the two decision trees forming the ensemble. Let $\mathcal{T}_i(x)$ denote the *value* of x assigned by the tree $\mathcal{T}_i$, for $i \in \{1, 2\}$. Then, $\mathcal{E}_1^{\mathsf{raw}}(x) = \sum_{i=1}^{2} \mathcal{T}_i(x) = 0 + (-1) = -1 = -\mathcal{E}_0^{\mathsf{raw}}(x)$. Further, $\mathcal{E}_1^{\mathsf{prob}}(x) = \mathsf{Sigmoid}(-1) \approx 0.27 < 0.5$. Therefore, $\mathcal{E}(x) = 0$.

Neighbourhood. Given two data points $x, y \in \mathcal{X}$, the L_∞ distance between x and y is defined as $\|x - y\|_\infty := \max_{f \in \mathcal{F}} |x(f) - y(f)|$. Given a data point $v \in \mathcal{X}$, a set of features $\mathcal{S} \subseteq \mathcal{F}$ and a real number $\varepsilon \in \mathbb{R}_{\geq 0}$, the *neighbourhood* of v is the set of all data points that can be constructed by varying *only* the values of the features in $\mathcal{S}$ by at most ε. This is formally defined as $\mathsf{Nbd}_{\mathcal{S},\varepsilon}(v) := \{x \in \mathcal{X} \mid \|v - x\|_\infty \leq \varepsilon, \text{ and } v(f) = x(f), \forall f \in \mathcal{F} \setminus \mathcal{S}\}$.

3 FEATURE SENSITIVITY IN DTES

A DTE $\mathcal{E}$ defined over a feature set $\mathcal{F}$ is said to be *sensitive* to a feature set $\mathcal{S} \subseteq \mathcal{F}$, if there exist two data points x and x', such that (i) x, x' differ only on the features $\mathcal{S}$, and (ii) the classification of x and x', under $\mathcal{E}$, are different. The notion of sensitivity is formalized below.

Definition 1 (Sensitivity, Ahmad et al. (2025))**.** Let $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ be a binary tree ensemble defined over the set of features $\mathcal{F}$. Given a subset of features $\mathcal{S} \subseteq \mathcal{F}$ and a *gap* parameter $g \geq 0$, the DTE $\mathcal{E}$ is said to be $(g, \mathcal{S})$-*sensitive* if there exists $x, x' \in \mathcal{X}$ such that

$$\forall f \in \mathcal{F} \setminus \mathcal{S},\ x(f) = x'(f), \text{ and } \begin{cases} \mathcal{E}_1^{\mathsf{prob}}(x') \leq 0.5 - g & \text{if } \mathcal{E}_1^{\mathsf{prob}}(x) \geq 0.5 + g, \\ \mathcal{E}_1^{\mathsf{prob}}(x') \geq 0.5 + g & \text{if } \mathcal{E}_1^{\mathsf{prob}}(x) \leq 0.5 - g. \end{cases}$$

The pair (x, x') is called a *sensitivity counterexample* (we will sometimes interchangeably refer to this as a *sensitive pair*). If no such pair exists, $\mathcal{E}$ is said to be $(g, \mathcal{S})$-*insensitive*.

Given a data point v, *local sensitivity* is the problem of finding another point x in 'proximity' of the given data point, such that (v, x) forms a sensitivity counterexample. The notion of 'proximity' is formalized using a given value $\varepsilon \in \mathbb{R}_{\geq 0}$: the search for x is restricted to $\mathsf{Nbd}_{\mathcal{S},\varepsilon}(v)$, that is, the ε-neighbourhood of v.

Definition 2 (Local Sensitivity)**.** Let $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ be a binary tree ensemble defined over the set of features $\mathcal{F}$. Let $\mathcal{S} \subseteq \mathcal{F}$ be a subset of features, and $g \geq 0$ a gap parameter. *Given a data point* $v \in \mathcal{X}$ and $\varepsilon \in \mathbb{R}_{\geq 0}$, the ensemble $\mathcal{E}$ is said to be *locally* $(g, \mathcal{S}, \varepsilon)$*-sensitive at* v if there exists $x \in \mathsf{Nbd}_{\mathcal{S},\varepsilon}(v)$ such that (v, x) is a sensitivity counterexample. If no such x exists, $\mathcal{E}$ is said to be *locally* $(g, \mathcal{S}, \varepsilon)$*-insensitive at* v.

We will use $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$ to denote an instance of the local sensitivity problem.

Example 2. Consider the DTE $\mathcal{E}$ in Figure 1, with $v := (0.7, 2.1)$, $g = 0$ and $\varepsilon = 0.2$. From Example 1 we know that $\mathcal{E}(v) = 0$. Now, suppose $\mathcal{S} := \{f_1\}$. Then, observe that, for every $x \in \mathsf{Nbd}_{\{f_1\},0.2}(v)$, $x(f_1) < 1$. This implies, $\mathcal{E}_1^{\mathsf{raw}}(x) = -1$ and $\mathcal{E}_1^{\mathsf{prob}}(x) < 0.5$. Hence, every x in the neighbourhood of v will have the same classification as v. Therefore, $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-insensitive at v, for $\varepsilon = 0.2$. Whereas $\mathcal{E}$ is $(g, \mathcal{S}, \varepsilon)$-sensitive at v for $\varepsilon = 0.2$, when $\mathcal{S} := \{f_2\}$. This is because the data point $x := (0.7, 1.95) \in \mathsf{Nbd}_{\{f_2\},0.2}(v)$, and $\mathcal{E}_1^{\mathsf{raw}}(x) = 0 \implies \mathcal{E}_1^{\mathsf{prob}}(x) \geq 0.5 \implies \mathcal{E}(x) = 1$.

Comparison with Local Robustness of DTEs. The local sensitivity problem of DTEs relates closely to the problem of *robustness* of DTEs studied by Chen et al. (2019).

Definition 3 (Robustness in DTEs, Chen et al. (2019))**.** Given a DTE $\mathcal{E} : \mathcal{X} \to \{0, 1\}$, given a data point $v \in \mathcal{X}$, and some $\varepsilon \in \mathbb{R}_{\geq 0}$, $\mathcal{E}$ is said to be ε*-robust at* v if $\forall x \in \mathsf{Ball}_\varepsilon(v), \mathcal{E}(v) = \mathcal{E}(x)$? Here, $\mathsf{Ball}_\varepsilon(v) := \{x \in \mathcal{X} \mid \|v - x\|_\infty \leq \varepsilon\}$.

Intuitively, in robustness we find a data point 'in the vicinity' of a given data point, such that they are classified differently. The point to note is that while searching for the second point, one is allowed to modify *any* of the features. This is where the difference lies with the local sensitivity problem. In the latter, one can only modify a given (sub)set of features. Observe now that, robustness is, in fact, the same problem as local sensitivity with respect to the whole feature set $\mathcal{F}$, that is, when we are allowed to modify any of the features.

Lemma 1. Given a DTE $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ defined over a feature set $\mathcal{F}$, a data point $v \in \mathcal{X}$ and $\varepsilon \in \mathbb{R}_{\geq 0}$, the ensemble $\mathcal{E}$ is ε-robust at v iff $\mathcal{E}$ is locally $(0, \mathcal{F}, \varepsilon)$-insensitive at v.

Local sensitivity therefore generalizes the problem of robustness, by fixing which all directions a given data point can be perturbed in order to flip the classification. Since robustness allows one to perturb a data point in (possibly) every direction at once, intuitively, there will be many data points appearing in practice that will not be robust – but they will be locally insensitive. Thus, local sensitivity allows us to *explain* the importance of particular features in deciding the classification in a DTE, while robustness can only reason about the entire feature set at a time.

Example 3. Consider again the DTE $\mathcal{E}$ depicted in Figure 1. From Example 2 we have that $\mathcal{E}$ is $(g, \mathcal{S}, \varepsilon)$-insensitive at $v := (0.7, 2.1)$ when $\mathcal{S} := \{f_1\}$ and $\varepsilon := 0.2$. However, note that, the point $x := (0.7, 1.9) \in \mathsf{Ball}_{0.2}(v)$, and we know that $\mathcal{E}(x) \neq \mathcal{E}(v)$. Therefore, $\mathcal{E}$ is not locally ε-robust at v for $\varepsilon = 0.2$. We can also note that $\mathcal{E}$ is not locally $(0, \mathcal{F}, \varepsilon)$-sensitive at v for $\varepsilon = 0.2$.

4 Hardness of Deciding Local Sensitivity

The computational complexity of various decision problems in DTEs have been studied in the literature. The problems of robustness (Chen et al. (2019)) and sensitivity checking (Ahmad et al. (2025)) are known to be NP-complete. On the other hand, the article by Ordyniak et al. (2024) studies parametric complexities for a plethora of problems regarding *explainability* for DTEs. Even though these problems are related to the local sensitivity problem (Lemma 1, Lemma 7), these reductions do not provide us a parametric complexity class for local sensitivity. In this section, we prove that the problem of local sensitivity is W[1]-hard with the parameter being the number of varying features

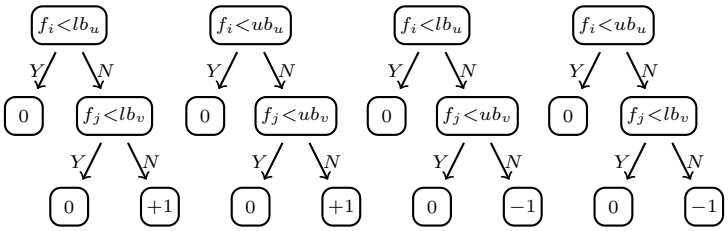


Figure 2: An ensemble of four decision trees assigning a data point to 1 only if that data point assigns $f_i \leftarrow u$ and $f_j \leftarrow v$ where $u < v$ and (u, v) is an edge in the graph, and assigns to 0 otherwise.

(i.e. the set $\mathcal{S}$). This reduction allows us to also conclude the hardness for certain restrictions of this problem (detailed proof in Appendix).

Theorem 2. Let $(\mathcal{E}, \mathcal{S}, g, v^*, \varepsilon)$ be an instance of the local sensitivity problem. Then, deciding if $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-sensitive is W[1]-hard with respect to the parameter $|\mathcal{S}|$, when $\mathcal{E}$ contains trees of depth ≥ 2.

Proof (sketch). We reduce the k-clique problem into the problem of deciding local sensitivity. The k-clique problem asks, given an undirected graph $G = (V, E)$ and an integer k, does there exist a clique of size k in G? This problem is known to be W[1]-hard with respect to the parameter k (Downey & Fellows (1995)). Without any loss of generality, we assume that: (i) the vertices of G are numbered from 1 to n, where $n = |V|$, and (ii) for every $(u, v) \in E$, we have $u < v$. From a given instance of k-clique, we now construct an instance of local sensitivity in the following manner.

We build a DTE $\mathcal{E}$ with k features $f_1, f_2, \ldots, f_k$. Each of these features range over $[0, n]$ where n is the number of vertices in G. For each vertex $u \in [0, n]$ of G, we define an interval $I_u := [lb_u, ub_u)$, where $lb_u := u - 0.5$ and $ub_u := u + 0.5$. Observe that, for every pair of vertices $u \neq v$, $I_u \cap I_v = \emptyset$. Given a data point x, when $x(f_i) \in I_u$ for some u, we say x maps f_i to u. Because of the disjointedness of the intervals, a data point can map each feature to at most one vertex in G. For every edge $(u, v) \in E$, we build $4.\binom{k}{2}$-many decision trees (each of depth 2). The idea here is that for an edge (u, v), we need to pick two features f_i and f_j (with $i < j$) such that f_i is mapped to u and f_j is mapped to v. Since an edge (u, v) can be associated with any pair of features f_i, f_j where $i < j$, we get the $\binom{k}{2}$ factor. Then, for each i, j, we have a gadget consisting of four decision trees (depicted in Figure 4) that map f_i and f_j to u and v, respectively. Lastly, we add one extra stump (i.e. a tree of depth 1) on a fresh feature f to $\mathcal{E}$; both leaves of this stump has value $-\binom{k}{2}$.

From this construction, we can show that G contains a clique $u_1 < u_2 < \cdots < u_k$ iff the data point x mapping each f_i to the vertex u_i, for $i \in \{1, 2, \ldots, k\}$, forms a locally $(g, \mathcal{S}, \varepsilon)$-sensitive sensitive pair with the data point $v^* = (0, 0, \ldots, 0)$ in $\mathcal{E}$.

The reduction from k-clique to the local sensitivity problem is polynomial-time computable, because the constructed ensemble contains $\left(4.\binom{k}{2}.|E| + 1\right)$-many trees. Since, k-clique is W[1]-hard with respect to the parameter k, and since $|\mathcal{S}| = k$, we get that deciding local sensitivity is W[1]-hard with respect to the parameter $|\mathcal{S}|$. □

Since the construction of the DTE from k-clique can be performed in polynomial time, we also get a polynomial time reduction from k-clique. Since k-clique is also known to be NP-hard (Cormen et al. (2009)), we get the NP-hardness for deciding local sensitivity. On the other hand, for NP-inclusion: we can guess a data point $x \in \mathsf{Nbd}_{\mathcal{S},\varepsilon}(v)$ and check in polynomial time if (v, x) is a sensitive pair.

Corollary 3. Given an instance $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$ of local sensitivity, deciding if $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-sensitive at v is NP-complete, when $\mathcal{E}$ contains trees of depth ≥ 2.

5 Algorithms for Deciding Local Sensitivity

In this section, we establish that the two restrictions on DTEs (i.e. the DTE contains trees of depth ≥ 2, and, the number of varying features is ≥ 2) that are used to get the hardness results, are

also necessary. If we drop any of the two requirements, local sensitivity can be decided in Polynomial time. We describe two algorithms for solving the local sensitivity problem. We start with a *region*-enumeration algorithm. This also helps us show a theoretical upper bound to the complexity of solving the local sensitivity problem. We then end this section with an MILP encoding for the local sensitivity problem by adapting the MILP encoding proposed for the sensitivity problem by Varshney et al. (2026).

Before introducing the region-enumeration algorithm (REGENUM), let us first define what we call a *region*. Given a DTE $\mathcal{E}$ defined over a set of features $\mathcal{F}$, consider the function τ that maps a feature to the set of thresholds the feature has been compared against in $\mathcal{E}$. Formally, $\tau : \mathcal{F} \to 2^{\mathbb{Q}}$ is the function defined as $\tau(f) \mapsto \{\tau_1^f, \tau_2^f, \ldots, \tau_p^f\}$ such that $\mathcal{E}$ contains a node with $f < \tau_i^f$ for every $i \in \{1, \ldots, p\}$. Conversely, for every comparison $f < \tau_j^f$ present in $\mathcal{E}$, $\tau(f)$ contains τ_j^f. Let $\mathcal{R}_\mathcal{E}$ denote the set $\prod_{f \in \mathcal{F}}(\tau(f))$. Each element of $\mathcal{R}_\mathcal{E}$ is called a *region*. Given a subset of features $\mathcal{S} \subseteq \mathcal{F}$, we use $\mathcal{R}_\mathcal{E}^\mathcal{S}$ to denote $\prod_{f \in \mathcal{S}}(\tau(f))$. Inside a region the classification does not change.

Lemma 4. Given a DTE $\mathcal{E}$ defined over the set of features $\mathcal{F}$, let $\mathcal{R}_\mathcal{E}$ be the set of all regions defined by $\mathcal{E}$. Then, for every $r \in \mathcal{R}_\mathcal{E}$, for every data point $x, y \in r$, $\mathcal{E}(x) = \mathcal{E}(y)$.

REGENUM algorithm. We solve the local sensitivity problem by simply enumerating over each region in $\mathcal{R}_\mathcal{E}^\mathcal{S}$. Given the data point v, we first compute the classification $\mathcal{E}(v)$. Then, for each of the features $f' \in \mathcal{F} \setminus \mathcal{S}$, we evaluate the decision nodes in $\mathcal{E}$ that involve f' using the value $v(f')$. This prunes the decision nodes inside $\mathcal{E}$. After this step, we get a DTE where all the decision nodes compare features from $\mathcal{S}$. Then, for each $r \in \mathcal{R}_\mathcal{E}^\mathcal{S}$, we pick a point $x_r \in r$. We can then compute $\mathcal{E}(x_r)$ and check if it is different from $\mathcal{E}(v)$. If so, we return the sensitive pair (v, x_r), else we continue to the next region.

Recall now, the parameterised complexity class XP consists of all the problems of size n and a parameter k, such that the problem can be solved in time $\mathcal{O}(n^{g(k)})$ for a computable function g defined over the parameter k (Flum & Grohe (2006)).

Theorem 5. Given an instance of the local sensitivity problem $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$, whether $\mathcal{E}$ is $(g, \mathcal{S}, \varepsilon)$-sensitive at v can be decided in XP with respect to the parameter $|\mathcal{S}|$.

Proof (sketch). Suppose n is the number of nodes present in $\mathcal{E}$, and let $|\mathcal{S}| = k$. Then, each feature can be compared against at most n thresholds. Therefore, $\mathcal{R}_\mathcal{E}^\mathcal{S}$ can be computed in time $\mathcal{O}(n^k)$. From this observation we can argue that the algorithm runs in $\mathcal{O}(n^k)$, showing that the problem is in XP with $|\mathcal{S}|$ being the parameter. □

From the complexity of the enumeration algorithm, we immediately get two results.

Corollary 6. Given an instance of the local sensitivity problem $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$, whether $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-sensitive can be decided in polynomial time whenever – (i) $|\mathcal{S}|$ is a fixed integer, or (ii) $\mathcal{E}$ only contains stumps, that is, trees of depth 1.

Proof (sketch). The result holds since in both of the cases, the number of regions is polynomial in the size of the tree. Then the REGENUM algorithm is a polynomial time algorithm. □

MILP encoding based algorithm. The REGENUM algorithm visits $\mathcal{R}_\mathcal{E}^\mathcal{S}$, which is practical only when $|\mathcal{S}|$ is small. For larger $\mathcal{S}$, we encode an instance $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$ as a mixed integer linear program (MILP), adapting the encoding of decision trees by Kantchelian et al. (2016) and its extension to global sensitivity by Varshney et al. (2026),which keeps one copy of the tree constraints for each input of a sensitive pair. Since the first input is fixed to v, its copy is constant: we replace it by the value $\mathcal{E}_1^{\mathsf{raw}}(v)$ and keep only the copy for x, to which the gap constraint and the objective of Varshney et al. (2026) now apply. In this copy, we fix every guard to its value at v, unless its feature is in $\mathcal{S}$ and its threshold lies within ε of v. This single change encodes both $\mathcal{S}$ and ε, prunes every tree to its free guards,

6 TWO APPLICATIONS OF LOCAL SENSITIVITY

In this section, we show that two well-studied existing problems can be solved by using local sensitivity as an oracle. Empirical evaluations in Section 7 show that when we use this oracle the performance is surprisingly good and often better than tools dedicated for the respective problems.

Contrastive Explanations in DTEs *Explaining* how a tree ensemble takes decision is an active area of research, falling under the broad umbrella of XAI. Different explanations for decision tree ensembles have been studied by Izza et al. (2025). These explanations, given a DTE, tries to find ranges of values of (a subset of) features that essentially determine the prediction of the DTE. Two popular kinds of explanations are generally studied: *contrastive* and *abductive explanations* (Ignatiev et al., 2020). In this section, we show that contrastive explanations of a DTE can be computed by solving local sensitivity.

Definition 4 (*Contrastive* explanations Ignatiev et al. (2020))**.** Given a DTE $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ and a data point $v \in \mathcal{X}$, a *contrastive explanation* of $\mathcal{E}$ with respect to v is a subset-minimal set of features $\mathcal{S}_C \subseteq \mathcal{F}$, such that $\exists x \in \mathcal{X}. \left(\bigwedge_{f \in \mathcal{F} \setminus \mathcal{S}_C} (x(f) = v(f))\right) \implies \mathcal{E}(x) \neq \mathcal{E}(v)$.

In words, a contrastive explanation is a subset-minimal set of features $\mathcal{S}_C$, such that, it is possible to change the prediction of the ensemble by changing only the values of the features in $\mathcal{S}_C$. Now, due to the minimality of $\mathcal{S}_C$, in fact, the values of *each* of the features in $\mathcal{S}_C$ need to be changed in order to change the prediction. The perturbation can be in any of the features, and any amount. This is essentially same as $(0, \mathcal{S}_C, \infty)$-sensitivity. In local sensitivity, the aim is to find a set of features that we can perturb in order to change the prediction; while in CXp the objective is to find a minimal such set. We formalize this relation between a CXp and local sensitivity in the result below.

Lemma 7. Given a DTE $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ defined over a set of features $\mathcal{F}$, a data point $v \in \mathcal{X}$, a set of features $\mathcal{S}_C \subseteq \mathcal{F}$ is a CXp of $\mathcal{E}$ with respect to v iff $\mathcal{S}_C$ is a subset-minimal set of features such that $\mathcal{E}$ is $(0, \mathcal{S}_C, \infty)$-sensitive at v.

This results leads to an **algorithm for computing a** CXp using local sensitivity as an oracle. Given a data point $v \in \mathcal{X}$, we first compute a data point $x \in \mathsf{Nbd}_{\mathcal{F},\infty}(v)$ so that (v, x) is a sensitive pair. This can be computed using the algorithms described in Section 5. We then compute the set of features F on which v and x differ. Starting from this set of features F, we keep changing one feature of x back to the value of v (so that the number of features on which v and x differ reduce) at a time, as long as the classfication of x and v remain different. The moment we cannot flip any more features, we get a minimal set of features that we could change, which is a CXp.

We can also use the local sensitivity **algorithm to compute a cardinality minimum** CXp. To do this, we again first find a data point x that certifies that $\mathcal{E}$ is $(0, \mathcal{F}, \infty)$-sensitive at v. Then, we can again find the set of all features F on which v and x differ. Now, we can take the MILP encoding based algorithm, and simply add a constraint that forces the number of varying features to be $\leq |F|$. Then, we can add an objective function that the total number of varying features should be minimized. This ensures that the computed CXp is indeed minimum in its size.

Monitoring We now consider the monitoring problem studied in Gupta et al. (2025). In this problem, we observe the history of decisions made by a given model. If there are two inputs in the history that are similar yet their decisions differ, then we raise an alarm. The history of executions needs to be stored in an efficient data structure to enable alarms to be raised in near real time. Monitoring is a highly effective method for detecting anomalous behaviors when we have only black-box access to the model and a long history of its executions.

Using local sensitivity, we can also detect anomalous behaviors of models in which the decision is flipped near an input. In a local-sensitivity based monitor, we receive a sequence of inputs to the model. For each input, we check the local sensitivity within a defined ε radius.If a region is found within the radius that flips the decision, we raise the alarm. This method requires white-box access to the model but does not require a long history or the large storage associated with it. Whenever the monitor finds an anomaly, our approach finds it too. However, our tool may find anomalies that monitoring may miss. In our experiments, we show that this local-sensitivity-based monitor is more effective in detecting anomalous behaviors.

7 EXPERIMENTS

We evaluate LOCAL-SENS on three research questions. **RQ1**. How do the two algorithms for deciding local sensitivity – REGENUM and, MILP encoding – compare against each other on the existing benchmarks? **RQ2**. How does LOCAL-SENS compare against the explainability computing tool XREASON in computing subset minimal and cardinality minimum *contrastive explanations* (CXp)? **RQ3**. Does LOCAL-SENS detect input-output robustness violations that runtime monitoring cannot observe – specifically, violations whose witness never appears in the input stream?

Implementation. We implement both the algorithms of Section 5 in our tool LOCAL-SENS, written in Python3.12. The MILP encoding is solved using Gurobi (Gurobi Optimization, LLC, 2024). All the experiments were run on a server with two AMD EPYC 7742 processors (64 cores each, 256 hardware threads in total) and 629 GiB of RAM, running Ubuntu 22.04.

Benchmarks. We train the binary tree ensembles with XGBoost (Chen & Guestrin, 2016) on the datasets in Table 1. Before training, every numeric feature was normalized to [0, 1] and every categorical feature was *one-hot* encoded. We used a learning rate of 0.01 and seed 42. Unless stated otherwise, the points v are drawn uniformly at random from the feature domain (seeds 42).

(RQ1) MILP vs.REGENUM. We compare the MILP and the REGENUM algorithms on higgs dataset with trees $\in \{100, 300\}$ and depth $\in \{4, 5\}$. For each model, we draw a random sets $\mathcal{S}$ of k features and use $\varepsilon \in \{0.01, 0.05, 0.09\}$. Figure 3a shows the runtime for $k \in \{1, 5, 10\}$. For $k = 1$, REGENUM is about an order of magnitude faster than the MILP on every instance. As the k grows, the number of regions grows as $\mathcal{O}(n^k)$. For $k = 5$, REGENUM remains faster on most of the instances, but it times out for 36 instances, and for $k = 10$, it times out for 901 instances. Therefore, enumeration is a better choice when the number of varying features is small, while MILP scales for larger number of varying features. This advocates the use of MILP as the default.

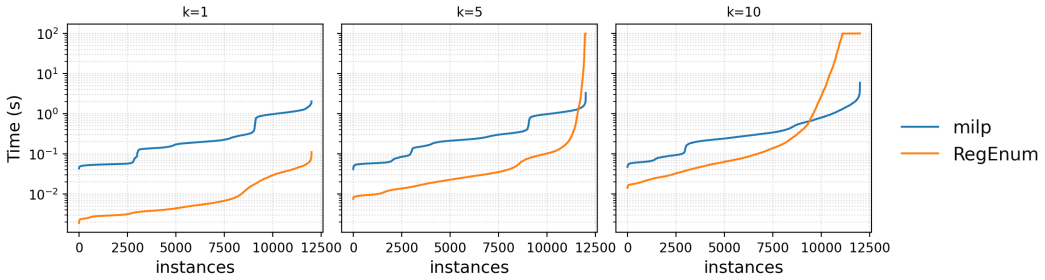
For RQ2 below, we tried the enumeration based algorithm and it timed out in most of the instances (see left plot in Figure 3a), while MILP performed much better. This is understandable, since to check explainability as well as monitoring, (as explained in Section 6) we first need to call the local sensitivity tool on the full feature set. For the benchmarks we tried, this size is around 40 on an average. RQ1 already demonstrates that MILP is going to outperform the enumeration based algorithm in this case. We therefore only report on LOCAL-SENS with the MILP encoding for the next two research questions.

(RQ2) Contrastive Explanation. We compare the LOCAL-SENS (with MILP encoding) with XREASON (Izza et al., 2026). We use six datasets (adult, bank, Breast-Cancer, churn, Pima-Diabetes, Spambase), two models per dataset (Tree $\in \{100, 300\}$ and depth=5), and 100 random points per model, which gives 1200 instances. XREASON does not take a gap parameter, so for LOCAL-SENS we kept $g = 0$, and the time limit was 500 seconds per instance.

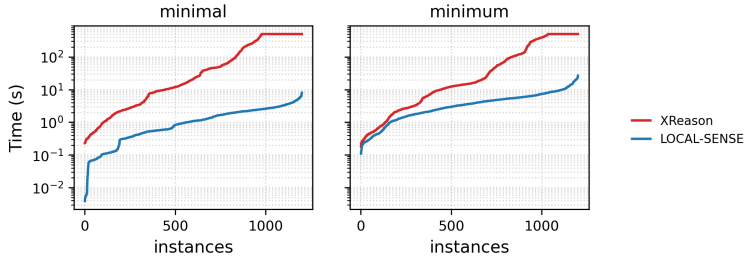
Figure 3b shows the runtime of subset-minimal CXp on the left and cardinality-minimum CXp on the right. For subset-minimal CXps, REGENUM reports the mean time 322.16s and timed out for 649 instances, XREASON reports the mean time 134.81s and time out for 224 instances, whereas LOCAL-SENS with MILP reports the mean time 1.44s and solves all the instances within the timelimit. On the 976 instances, where both XREASON and LOCAL-SENS found a subset-minimal CXP, the explanation of LOCAL-SENS had 1.54 features on average against 1.67 for XREASON .

We compared LOCAL-SENS again with XREASON for cardinality-minimum CXPs on the same set of benchmarks. XREASON reports the mean time of 123.3s, with 168 timed out instances, whereas LOCAL-SENS reports a mean time of 4.50s and completes all the instances within the time limit. All the 1032 instances solved by both XReason and LOCAL-SENS had the same size of explanation, with an average of 1.13 for both tools.

(RQ3) Monitoring. We compare LOCAL-SENS with the tool CLEMONT (Gupta et al., 2025), a runtime monitor for robustness. We run it on the same benchmark and with $\varepsilon = 0.02$. The input stream is the test set of each dataset. We tested on a stream size of 1000 rows. CLEMONT computes local robustness, this is the same problem as computing local sensitivity with all features being allowed to vary. We therefore use LOCAL-SENS with the set of varying features being the set of all features. Table 2 summarizes the results. For Breast-Cancer, churn and Pima-Diabetes, the monitor does not raise any alarm at all, while LOCAL-SENS raises many alarms. LOCAL-SENS will catch



(a) Runtime of REGENUM and MILP, where k is the number of sensitive features per query.



(b) Runtime of XREASON and LOCAL-SENS for computing CXps. Left: subset-minimal CXps. Right: cardinality-minimum CXps.

Figure 3: Cactus plots of runtime (in seconds).

Table 1: Dataset and model details

Dataset	#Features	Trees	Depth
Adult	108	{100, 300}	{4, 5}
Bank	51	{100, 300}	{4, 5}
Breast-Cancer	9	{100, 300}	{4, 5}
churn	21	{100, 300}	{4, 5}
Pima-Diabetes	8	{100, 300}	{4, 5}
Spambase	57	{100, 300}	{4, 5}
Higgs	28	{100, 300}	{4, 5}

Table 2: Results for Monitoring across datasets

Dataset	Model	Rows	LOCAL-SENS	CLEMONT
Adult	t100_d5	1000	289	1
Bank	t100_d5	1000	100	1
Breast-Cancer	t100_d5	140	7	0
Churn	t100_d5	1000	132	0
Pima-Diabetes	t100_d5	154	35	0
Spambase	t100_d5	921	876	38

every alarm that the monitor raises, and most of the robustness violations have witnesses that never appear in the input stream.

8 CONCLUSION

Sensitivity in DTEs deals with the problem of checking if there exists a certain set of features whose value can alter the prediction of the DTEs. So far, this problem has only been studied in the global context. In this article, we consider the *local* version of this problem, i.e. given a data point is it possible to alter some of its feature values to change its prediction. This is a natural extension of the sensitivity problem to study. We defined this problem formally and also compute the parameterised complexity of deciding this problem, and gave two practical algorithms to solve this. Importantly, we showed that two of the other well-studied problems for DTEs – namely, (contrastive) explanation and monitoring – reduce to checking local sensitivity. Empirical evaluation shows that our local-sensitivity based approach outperforms the dedicated tools for these applications. In the future, we would like to close the complexity theoretic gap for local sensitivity. We would explore if local senstivity can be solve other problems as well, for example, abductive explanations.

AI-USAGE DECLARATION

In this work, we used generative AI tools to mainly polish our writing. We also discussed parameterized complexity hardness proof ideas in the beginning. All the proofs present in the paper are formalized and written down by us. We have not used generative AI tools to draft parts of the paper. Synthetic data generation was not applicable to this work. We used a generative-AI tool to mainly write down scripts to carry out experiments, find if there were any particular types of bugs, and generate scripts that can verify the accuracy of the results produced. We take responsibility for the final content of this work.

REFERENCES

Arhaan Ahmad, Tanay Vineet Tayal, Ashutosh Gupta, and S. Akshay. Sensitivity verification for additive decision tree ensembles. In *ICLR*. OpenReview.net, 2025.

Gilles Audemard, Jean-Marie Lagniez, Pierre Marquis, and Nicolas Szczepanski. Computing abductive explanations for boosted trees. In *International Conference on Artificial Intelligence and Statistics*, pp. 4699–4711. PMLR, 2023.

Stefano Calzavara, Lorenzo Cazzaro, Claudio Lucchese, and Federico Marcuzzi. Explainable global fairness verification of tree-based classifiers. *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 1–17, 2022. URL https://api.semanticscholar.org/CorpusID:252545325.

Yung-Chia Chang, Kuei-Hu Chang, and Guan-Jhih Wu. Application of extreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing*, 73:914–920, 2018. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2018.09.029. URL https://www.sciencedirect.com/science/article/pii/S1568494618305465.

Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane S. Boning, and Cho-Jui Hsieh. Robustness verification of tree-based models. In *NeurIPS*, pp. 12317–12328, 2019.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

Laurens Devos, Wannes Meert, and Jesse Davis. Versatile verification of tree ensembles. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2654–2664. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/devos21a.html.

Laurens Devos, Lorenzo Cascioli, and Jesse Davis. Robustness verification of multi-class tree ensembles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21019–21028, Mar. 2024. doi: 10.1609/aaai.v38i19.30093. URL https://ojs.aaai.org/index.php/AAAI/article/view/30093.

Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. *CoRR*, abs/1104.3913, 2011. URL http://arxiv.org/abs/1104.3913.

Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

Mohammad M. Ghiasi and Sohrab Zendehboudi. Application of decision tree-based ensemble learning in the classification of breast cancer. *Computers in Biology and Medicine*, 128:104089, 2021. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2020.104089. URL https://www.sciencedirect.com/science/article/pii/S0010482520304200.

Ashutosh Gupta, Thomas A Henzinger, Konstantin Kueffner, Kaushik Mallik, and David Pape. Monitoring robustness and individual fairness. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 790–801, 2025.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com.

Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva. From contrastive to abductive explanations and back again. In *International Conference of the Italian Association for Artificial Intelligence*, pp. 335–355. Springer, 2020.

Alexey Ignatiev, Yacine Izza, Peter J Stuckey, and Joao Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 3776–3785, 2022.

Yacine Izza, Alexey Ignatiev, Sasha Rubin, João Marques-Silva, and Peter J. Stuckey. Most general explanations of tree ensembles. In *IJCAI*, pp. 5463–5471. ijcai.org, 2025.

Yacine Izza, Alexey Ignatiev, Xuanxiang Huang, Peter J Stuckey, and Joao Marques-Silva. Rigorous explanations for tree ensembles. *arXiv preprint arXiv:2603.29361*, 2026.

Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. Evasion and hardening of tree ensemble classifiers. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2387–2396. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/kantchelian16.html.

Andrei V Kelarev, Andrew Stranieri, JL Yearwood, and Herbert F Jelinek. Empirical study of decision trees and ensemble classifiers for monitoring of diabetes patients in pervasive healthcare. In *2012 15th International Conference on Network-Based Information Systems*, pp. 441–446. IEEE, 2012.

Mehul Madaan, Aniket Kumar, Chirag Keshri, Rachna Jain, and Preeti Nagrath. Loan default prediction using decision trees and random forest: A comparative study. *IOP Conference Series: Materials Science and Engineering*, 1022(1):012042, jan 2021. doi: 10.1088/1757-899X/1022/1/012042. URL https://dx.doi.org/10.1088/1757-899X/1022/1/012042.

Ajinkya Naik, Chaitanya Garg, S. Akshay, Ashutosh Gupta, and Kuldeep S. Meel. Quantifying sensitivity for tree ensembles: A symbolic and compositional approach. In *CAV (2)*, volume 16683 of *Lecture Notes in Computer Science*, pp. 551–575. Springer, 2026.

Sebastian Ordyniak, Giacomo Paesani, Mateusz Rychlicki, and Stefan Szeider. Explaining decisions in ML models: A parameterized complexity analysis. In Pierre Marquis, Magdalena Ortiz, and Maurice Pagnucco (eds.), *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024*, 2024. doi: 10.24963/KR.2024/53. URL https://doi.org/10.24963/kr.2024/53.

Francesco Ranzato and Marco Zanella. Abstract interpretation of decision tree ensemble classifiers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5478–5486, Apr. 2020. doi: 10.1609/aaai.v34i04.5998. URL https://ojs.aaai.org/index.php/AAAI/article/view/5998.

John Törnblom and Simin Nadjm-Tehrani. An abstraction-refinement approach to formal verification of tree ensembles. In *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings*, pp. 301–313, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-26249-5. doi: 10.1007/978-3-030-26250-1_24. URL https://doi.org/10.1007/978-3-030-26250-1_24.

Namrita Varshney, Ashutosh Gupta, Arhaan Ahmad, Tanay Vineet Tayal, and S. Akshay. Data-aware and scalable sensitivity analysis for decision tree ensembles. In *ICLR*. proceedings.iclr.cc, 2026.

Yihan Wang, Huan Zhang, Hongge Chen, Duane S. Boning, and Cho-Jui Hsieh. On lp-norm robustness of ensemble decision stumps and trees. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10104–10114. PMLR, 2020. URL `http://proceedings.mlr.press/v119/wang20aa.html`.

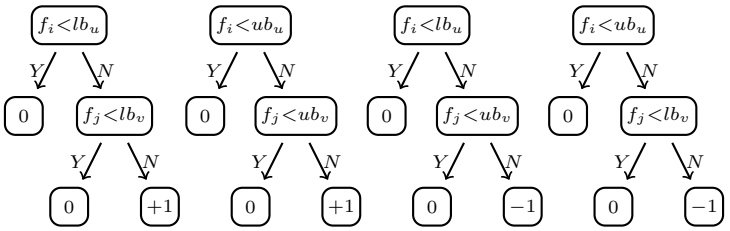


Figure 4: An ensemble of four decision trees assigning a data point to 1 only if that data point assigns $f_i \leftarrow u$ and $f_j \leftarrow v$ where $u < v$ and (u, v) is an edge in the graph, and assigns to 0 otherwise.

A Missing proofs from the article

Theorem 2. Let $(\mathcal{E}, \mathcal{S}, g, v^*, \varepsilon)$ be an instance of the local sensitivity problem. Then, deciding if $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-sensitive is W[1]-hard with respect to the parameter $|\mathcal{S}|$, when $\mathcal{E}$ contains trees of depth ≥ 2.

Proof. We build a DTE $\mathcal{E}$ with k features $f_1, f_2, \ldots, f_k$. Each of these features range over $[0, n]$ where n is the number of vertices in G. For each vertex $u \in [0, n]$ of G, we define an interval $I_u := [lb_u, ub_u)$, where $lb_u := u - 0.5$ and $ub_u := u + 0.5$. Observe that, for every pair of vertices $u \neq v$, $I_u \cap I_v = \emptyset$. Given a data point x, when $x(f_i) \in I_u$ for some u, we say x maps f_i to u. Because of the disjointedness of the intervals, a data point can map each feature to at most one vertex in G. For every edge $(u, v) \in E$, we build $4.\binom{k}{2}$-many decision trees (each of depth 2). The idea here is that for an edge (u, v), we need to pick two features f_i and f_j (with $i < j$) such that f_i is mapped to u and f_j is mapped to v. Since an edge (u, v) can be associated with any pair of features f_i, f_j where $i < j$, we get the $\binom{k}{2}$ factor. Then, for each i, j, we have a gadget consisting of four decision trees (depicted in Figure 4) that map f_i and f_j to u and v, respectively. Lastly, we add one extra stump (i.e. a tree of depth 1) on a fresh feature f to $\mathcal{E}$; both leaves of this stump has value $-\binom{k}{2}$. We then show that G contains a clique $u_1 < u_2 < \cdots < u_k$ iff the data point x mapping each f_i to the vertex u_i, for $i \in \{1, 2, \ldots, k\}$, forms a locally sensitive pair with the data point $v^* = (0, 0, \ldots, 0)$ in $\mathcal{E}$.

We first prove that a data point maps every feature to at most one vertex of G.

Lemma 8. Given a data point x, if $x(f_i) \in I_u$ (i.e. f_i gets mapped to u) for some vertex u, then for every other vertex $v \neq u$, $x(f_i) \notin I_v$ (i.e. f_i does not get mapped to v).

Proof. This follows from the fact that I_u and I_v are disjoint for every $u, v \in V$ with $u \neq v$. □

We now prove that the 4-trees' gadget for an edge (u, v) and a pair of features f_i, f_j, contributes a weight of $+1$ to $\mathcal{E}_1^{\mathsf{raw}}(x)$ if and only if $f_i \in I_u$ and $f_j \in I_v$, and contributes 0 otherwise.

Lemma 9. Given a data point x, two features f_i, f_j with $i < j$, the four trees corresponding to an edge (u, v) contribute a weight of $+1$ to $\mathcal{E}_1^{\mathsf{raw}}(x)$ if and only if $x(f_i) \in I_u$ and $x(f_j) \in I_v$. In all other cases the contribution of these four trees to $\mathcal{E}_1^{\mathsf{raw}}(x)$ is 0.

Proof. We have $I_u = [lb_u, ub_u)$ and $I_v = [lb_v, ub_v)$. We argue the result by reasoning over all the possible values of $x(f_i)$ and $x(f_j)$. If $x(f_i) < lb_u$ or $x(f_j) < lb_v$ then all four trees contribute 0. Now, suppose $x(f_i) \in I_u$, and $x(f_j) > ub_v$. In this case, only first and third trees contribute; the first tree contributes $+1$ and the third tree -1, hence the total contribution is 0. On the other hand, if $x(f_i) > ub_u$ and $x(f_j) \in I_v$, all four trees are relevant. Here, the first and fourth trees contribute non-zero weights, and they cancel each other out, hence the total contribution is again 0. When $x(f_i) > ub_u$ and $x(f_j) > ub_v$, all four trees have non-zero values, but they cancel each other out! Hence, overall they contribute 0 to $\mathcal{E}_1^{\mathsf{raw}}(x)$. Finally, when $x(f_i) \in I_u$ and $x(f_j) \in I_v$, the first tree contributes $+1$ and all other trees contribute 0, hence the total contribution is $+1$. □

We now show that the raw value of a point under $\mathcal{E}$ is always bounded above by 0.

Lemma 10. For every data point x, $\mathcal{E}_1^{\mathsf{raw}}(x) \leq 0$. Further, $\mathcal{E}_1^{\mathsf{raw}}(x) = 0$ iff there exists vertices $u_1, u_2, \ldots, u_k$, such that $\forall i \in \{1, \ldots, k\}$, x maps f_i to u_i, and for every $u_i < u_j$, $(u_i, u_j) \in E$.

Proof. Given the data point x, let $c_{ij}(x)$ denote the total contribution of all the trees involving the features f_i and f_j, to the value $\mathcal{E}_1^{\mathsf{raw}}(x)$. We then get that $\mathcal{E}_1^{\mathsf{raw}}(x) = \sum_{i<j} c_{ij}(x)$. We prove the lemma by showing that for every pair $i < j$, $c_{ij}(x) \in \{0, 1\}$. First, note that, from Lemma 8 we get that each feature can only be mapped to at most one vertex. That is, for every pair of features f_i, f_j, there can only exist at most one pair of vertices (u, v) such that f_i is mapped to u and f_j to v. This implies that for a pair f_i, f_j, only one of the $\binom{k}{2}$ gadgets (of 4 trees) can fire. Now, Lemma 9 gives us that if a pair f_i, f_j is indeed mapped to a pair of vertices then the overall contribution of the trees involving f_i, f_j is $+1$. If they are not mapped to a pair of vertices, the contribution is 0. This proves the claim that $c_{ij}(x)$ is either 1 or 0, for every $i < j$.

We then get that $\mathcal{E}_1^{\mathsf{raw}}(x) = -\binom{k}{2} + \sum_{i<j} c_{ij}(x) \leq -\binom{k}{2} + \sum_{i<j}(1) = -\binom{k}{2} + \binom{k}{2} = 0$.

We can also observe that the sum equals to 0 only when for each $i < j$, $c_{ij}(x) = 1$. This corresponds to the case when x maps each pairs of features f_i, f_j to a pair of vertices $u < v$ such that $(u, v) \in E$. Note that, our designed ensemble $\mathcal{E}$ contains a gadget only for pairs (u, v) such that $(u, v) \in E$. □

Lemma 10 immediately leads to the following result.

Corollary 11. Given a data point x, $\mathcal{E}(x) = 1$ iff there exist vertices $u_1, u_2, \ldots, u_k$ such that for all $i \in \{1, \ldots, k\}$, $x(f_i) \in I_{u_i}$ and $(u_i, u_j) \in E$ for all $i < j$.

Proof. From Lemma 10 we get that $\mathcal{E}_1^{\mathsf{raw}}(x) = 0$ iff there exists a set of k vertices satisfying the given condition, and $\mathcal{E}_1^{\mathsf{raw}}(x) < 0$ otherwise. Then the result follows from the fact that $\mathsf{Sigmoid}(z) \geq 0.5$ only if $z \geq 0$, and $\mathsf{Sigmoid}(z) < 0$ otherwise, for every $z \in \mathbb{R}$. □

We will use the corollary above to prove the correctness of the reduction.

Lemma 12. G contains a k-clique iff $\mathcal{E}$ is locally $(0, \mathcal{S}, \varepsilon)$-sensitive at $v^* := (0, 0, \ldots, 0)$, where $\mathcal{S} := \{f_1, f_2, \ldots, f_k\}$.

Proof. First, note that $\mathcal{E}(v^*) = 0$. This is because, for every $i \in \{1, 2, \ldots, k\}$, $v^*(f_i) = 0$, and for every vertex u, the lower bound of the interval I_u is ≥ 0.5. Therefore, v^* in fact does not map any feature f_i to a vertex. Hence, $c_{ij}(v^*) = 0$ for every i, j.

We now claim: G contains a k-clique iff there exists a data point $x \in \mathsf{Nbd}_{\mathcal{S},\varepsilon}(v^*)$ with $\mathcal{E}(x) = 1$.

($\Rightarrow$) Suppose G contains a k-clique with vertices $u_1 < u_2 < \cdots < u_k$. Now consider the data point x with $x(f_i) = u_i$ for every i. Then, $x(f_i) \in I_{u_i}$ for every i, and from Corollary 11 we can deduce that $\mathcal{E}(x) = 1$. Therefore, (v^*, x) is a sensitivity counterexample, and $\mathcal{E}$ is $(0, S, \varepsilon)$-sensitive at v^*. Note here that $x \in \mathsf{Nbd}_{S,n}(v^*)$, since $\|v^* - x\|_\infty = |v^*(f_k) - x(f_k)| = n - 0 = n$.

($\Leftarrow$) Suppose $\mathcal{E}$ is $(0, S, \varepsilon)$-sensitive at v^*. Then since $\mathcal{E}(v^*) = 0$, there exists a data point $x \in \mathsf{Nbd}_{\mathcal{S},\varepsilon}(v^*)$ such that $\mathcal{E}(x) = 1$. From Corollary 11 we get that there exists a set of vertices $u_1, u_2, \ldots, u_k$ such that for every i, $x(f_i) \in I_{u_i}$, and for every pair $i < j$, $(u_i, u_j) \in E$. Therefore, the vertices $u_1, u_2, \ldots, u_k$ form a k-clique in G. □

The reduction from k-clique to the local sensitivity problem is polynomial-time computable, because the constructed ensemble contains $\left(4.\binom{k}{2}.|E| + 1\right)$-many trees. Since, k-clique is $\mathsf{W[1]}$-hard with respect to the parameter k, and since $|\mathcal{S}| = k$, we get that deciding local sensitivity is $\mathsf{W[1]}$-hard with respect to the parameter $|\mathcal{S}|$. □

Remark: In the ($\Rightarrow$) direction of the proof of Lemma 12, we construct a point x which is at distance n from the given point v^*. This shows that the reduction works for any $\varepsilon \geq n$. However, given any ε, we can always modify the intervals I_u in such a way that the distance of the constructed point x from v^* is at most ε. This shows that the reduction works for any $\varepsilon \in \mathbb{R}_{\geq 0}$.

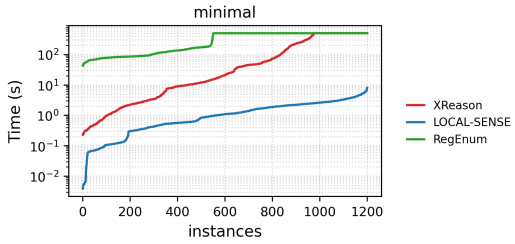


Figure 5: Runtime comparison (in seconds) of XREASON and LOCAL-SENS and REGENUMfor computing CXps.

Corollary 3. Given an instance $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$ of local sensitivity, deciding if $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-sensitive at v is NP-complete, when $\mathcal{E}$ contains trees of depth ≥ 2.

Proof. The reduction from k-clique to local sensitivity as described in Theorem 2 is a polynomial time reduction (the proposed DTE can be constructed in polynomial time). Then, since k-clique is known to be an NP-hard problem (Cormen et al. (2009)), so is local sensitivity. On the other hand, given a DTE $\mathcal{E}$ and a data point v, a certificate for local sensitivity is another data point $x \in \mathsf{Nbd}_{\mathcal{S},\varepsilon}(v)$. The validity of this certificate can be checked by simply computing $\mathcal{E}(x)$, which can be done in polynomial time. Therefore, the local sensitivity problem is NP-complete. □

Lemma 4. Given a DTE $\mathcal{E}$ defined over the set of features $\mathcal{F}$, let $\mathcal{R}_{\mathcal{E}}$ be the set of all regions defined by $\mathcal{E}$. Then, for every $r \in \mathcal{R}_{\mathcal{E}}$, for every data point $x, y \in r$, $\mathcal{E}(x) = \mathcal{E}(y)$.

Proof. Each region $r \in \mathcal{R}_{\mathcal{E}}$ consists of the data points that satisfy *all* the constraints defining the particular region r. Therefore, every data point inside one region satisfy the same set of thresholds present in $\mathcal{E}$, and hence they all follow the same paths of $\mathcal{E}$, thereby having the same value under $\mathcal{E}$. This implies every point inside a region have the same classification under $\mathcal{E}$. □

Theorem 5. Given an instance of the local sensitivity problem $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$, whether $\mathcal{E}$ is $(g, \mathcal{S}, \varepsilon)$-sensitive at v can be decided in XP with respect to the parameter $|\mathcal{S}|$.

Proof. We claim that the proposed algorithm terminates and it is correct. This algorithm terminates since $|\mathcal{R}_{\mathcal{E}}^{\mathcal{S}}| < \infty$. The algorithm terminates in two situations: (i) it finds a data point x_r with $\mathcal{E}(x_r) \neq \mathcal{E}(v)$. This implies (v, x_r) is indeed a sensitive pair. Or, (ii) the algorithm exhausts all the regions without finding a sensitive pair. This implies $\mathcal{E}$ is $(g, \mathcal{S}, \varepsilon)$-insensitive at v. Since if it were sensitive then the sensitive pair must have been present in one of the regions in $\mathcal{R}_{\mathcal{E}}^{\mathcal{S}}$.

Suppose n is the number of nodes present in $\mathcal{E}$, and let $|\mathcal{S}| = k$. Then, each feature can be compared against at most n thresholds. That is, for every $f \in \mathcal{F}$, $|\tau(f)| \leq n$. Therefore, $\mathcal{R}_{\mathcal{E}}^{\mathcal{S}}$ can be computed in time $\mathcal{O}(n^k)$. From every region r, a point can be picked in constant time. Computing the classification of a point under $\mathcal{E}$ can be accomplished in time linear to n. Therefore, overall the algorithm runs in $\mathcal{O}(n^k)$. This shows that the algorithm is in XP. □

Corollary 6. Given an instance of the local sensitivity problem $(\mathcal{E}, \mathcal{S}, g, v, \varepsilon)$, whether $\mathcal{E}$ is locally $(g, \mathcal{S}, \varepsilon)$-sensitive can be decided in polynomial time whenever – (i) $|\mathcal{S}|$ is a fixed integer, or (ii) $\mathcal{E}$ only contains stumps, that is, trees of depth 1.

Proof. The result holds since the algorithm proposed in the proof of Theorem 5 runs in polynomial time whenever we fix the parameter k, that is the size of $\mathcal{S}$.

Note that, for stumps, we can accumulate all the thresholds to which the features are compared in the ensemble. Given a data point v, we can then enumerate over all these thresholds to see if we modify v to cross any of these thresholds how the value of the ensemble changes. This will allow us to check (using a linear pass over the trees) if we can modify v to get a point x such that (v, x) is a sensitivity counterexample. □

Algorithm 1 Computing contrastive explanations using local sensitivity

Require: a DTE $\mathcal{E} : \mathcal{X} \to \{0, 1\}$ defined over feature set $\mathcal{F}$, a data point $v \in \mathcal{X}$
Ensure: a contrastive explanation of $\mathcal{E}$ with respect to v

cex_pair_found $\leftarrow$ False
while cex_pair_found $==$ False **do**
    pick a data point $v \in \mathcal{X}$ uniformly randomly
    $x \leftarrow$ find_sens_pair(v, ∞)
    if $x \neq$ None **then**
        cex_pair_found $\leftarrow$ True
$F \leftarrow \{f \in \mathcal{F} \mid x(f) \neq v(f)\}$
while $F \neq \emptyset$ **do**
    pick a random feature f' from F
    $F' \leftarrow F \setminus \{f'\}$
    construct a new data point x', where $x'(f) = x(f)$ for every $f \neq f'$, and $x(f') = v(f')$
    if $\mathcal{E}(v) == \mathcal{E}(x')$ **then**
        return F
    $F \leftarrow F'$
    $x \leftarrow x'$