Safety Augmentation in Decision Trees

Sumanta Dey, **Pallab Dasgupta** and **Briti Gangopadhyay***
Dept of Computer Science and Engineering, IIT Kharagpur, India
sumanta.dey@iitkgp.ac.in, pallab@cse.iitkgp.ac.in, briti_gangopadhyay@iitkgp.ac.in

Abstract

Decision trees are among the most popular representations used in statistical machine learning. This paper studies the problem of incorporating domain knowledge into the learning of decision trees, specifically, when the knowledge is given in the form of a set of known decision cases, captured using *assertions*. The ability to ensure that the decision making never violates a known set of safety assertions is essential for the use of decision tree learning in safety critical domains. A pure machine learning approach does not guarantee safety, since the learning can be impaired by the lack of adequate data on possible failure scenarios. To the best of our knowledge, this is the first work on formal safety augmentation in decision trees with provable guarantees.

1 Introduction

Decision Tree Learning is among the most widely used and practical methods for supervised learning for both classification and regression tasks [Breiman, 2017; Song and Ying, 2015]. Decisions based on decision trees are inherently explainable and statistically supported by the data used to learn the tree. The primary goal of a decision tree learning algorithm is to construct an *optimal* decision tree, where the quality is defined in terms of the depth of the tree (which is an upper-bound on the number of variables to be examined before reaching a decision) and the number of nodes in the tree (which reflects the average effort in reaching a decision). Finding an optimal decision tree is an NP-Complete problem [Laurent and Rivest, 1976; Murthy, 1998]. There are several greedy algorithms for learning decision trees, including ID3, C4.5, C5.0 [Quinlan, 1986; Quinlan, 2014; Quinlan, 2007]. Greedy metrics, such as Information Gain, are used to order the nodes in the decision tree during the construction. For example, the decision tree developed using ID3 for the dataset of Table 1 is shown in Figure 1a. This decision tree decides whether a type of Mushroom with given features like the CapShape, the CapColor, and the GillColor is poisonous or edible based on the information gained from the given dataset. As can be seen in Figure 1a, the decision is taken by examining only the features, CapColor, and CapShape. The feature, GillColor, is ignored by the decision tree, and this can be statistically justified from the dataset.

For safety-critical decisions, relying on a decision tree constructed using statistical machine learning can be risky, because the dataset will typically not contain samples covering all possible combinations of features. The missing combinations may either not exist in the real world or maybe so rare that they do not show up in the dataset. This is often true when the number of features and their value domains are large. It is, therefore, possible that the decision tree will make a wrong decision because it has no supervised precedence for unobserved combinations of feature values. For example, we may know from scientific knowledge that mushrooms having green GillColor and CapShape like a bell are definitely poisonous, though there is no support for this knowledge in the data. Consequently, the decision tree of Figure 1a recommends all mushrooms with yellow CapColor, ignoring the features CapShape and GillColor, possibly leading to disastrous consequences. The decision tree also declares as edible all mushrooms with white CapColor and bell-like CapShape, whereas the above scientific knowledge should be used to eliminate the ones with green GillColor from this set.

Safety requirements gleaned from (non-statistical) domain knowledge can be provided explicitly in the form of *assertions*, a practice which is widely used in many safety-critical domains [SVA, 2018; Peled *et al.*, 1999]. For example, the knowledge on mushrooms with green GillColor and bell-like CapShape can be expressed as the following propositional assertion:

$$(CapShape = bell \wedge GillColor = green) \Rightarrow Poisonous$$

This paper studies the augmentation of decision trees with such safety assertions. We show that naive post-facto processing of a decision tree with the given assertions adversely affects the size of the decision tree. For example, such an augmentation with the above assertion leads to the decision tree of Figure 1b, where the average number of variables to

*The authors like to thank DST, Govt of India, for partial financial support of this project.

CapShape	CapColor	GillColor	Poisonous
Bell	Pink	Green	Poisonous
Bell	Pink	White	Poisonous
Bell	Pink	Gray	Poisonous
Convex	Pink	Gray	Poisonous
Convex	Pink	Brown	Poisonous
Convex	White	Brown	Poisonous
Convex	White	White	Poisonous
Convex	White	Gray	Poisonous
Convex	Yellow	Brown	Edible
Convex	Yellow	Gray	Edible
Convex	Yellow	White	Edible
Bell	Yellow	White	Edible
Bell	Yellow	Gray	Edible
Bell	Yellow	Brown	Edible
Bell	White	Brown	Edible
Bell	White	Gray	Edible
Bell	White	White	Edible

Table 1: A section of the mushroom dataset [Dua and Graff, 2017]

be examined before reaching a decision is significantly more than that of Figure 1a.

We present a methodology for biasing the information gain metric based on the safety assertions to compensate for the missing information in the dataset. This leads to considerable improvement in the safety-augmentation process. The modified decision tree using our proposed methodology is shown in Figure 2. The average number of variables to be examined before reaching a decision in this tree is significantly better than that of Figure 1b.

The paper is organized as follows. Section 2 outlines the post-facto safety augmentation methodology. Section 3 presents the proposed integrated safety augmentation approach, and Section 4 outlines multi-assertion extensions. Experimental results on benchmark datasets are presented in Section 5. Related work and concluding remarks are given in sections 6 and 7 respectively.

2 Post-Facto Safety Augmentation

This section examines the merit of considering the safety assertions after the decision tree has been constructed. We consider the following two options:

1. *Analyzing safety assertions before using the decision tree.* This is typically not a good option in practice. Safety-critical scenarios often have specific attributes which need not be examined if the decision is safe anyway. For example, suppose a safety property specifies that: *sulfa drugs should not be administered unless a sensitivity test rules out allergic reaction*. It does not make sense to perform a sensitivity test for allergic reactions to sulfa drugs before the decision tree recommends such a drug. In other words, examining all the predicates in the antecedent of an assertion is redundant in those paths of the decision tree where the decision is safe anyway.
2. *Analyzing safety assertions after using the decision tree.* We may analyze the decision recommended by the decision tree against the safety properties. For example, the decision tree of Figure 1a recommends us to eat mushrooms having white CapColor and bell-like CapShape, whereas the safety assertion tells us that mushrooms with bell-like CapShape and green GillColor are poisonous. We must therefore not go by the decision recommended by the decision tree, but examine the GillColor. This post-facto safety augmentation is shown in Figure 1b.

For a leaf node, v, of the decision tree, let $\pi(v)$ denote the conjunction of predicates leading to v, and let $dec(v)$ denote the decision recommended at v. For example, in Figure 1a, if v is the second leaf node from the left, then $dec(v) = edible$ and:

$$\pi(v) = CapColor = white \wedge CapShape = bell$$

For an assertion, φ, let $ant(\varphi)$ denote the antecedent of φ. For example, for the assertion, φ:

$$(CapShape = bell \wedge GillColor = green) \Rightarrow Poisonous$$

$$ant(\varphi) = (CapShape = bell \wedge GillColor = green)$$

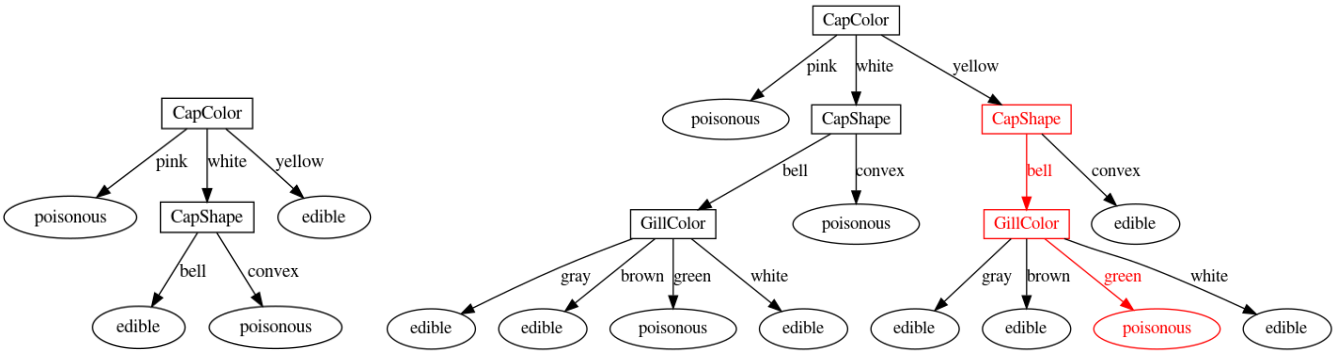
Lemma 2.1 *Safety augmentation is necessary in a leaf node v of a decision tree iff $\pi(v) \wedge ant(\varphi)$ is satisfiable, but $dec(v)$ refutes the consequent of φ.*

Proof: If $\pi(v) \wedge ant(\varphi)$ is False (unsatisfiable), then no safety augmentation is needed at v because the antecedent of φ does not match on this path of the decision tree. If $dec(v)$ satisfies the consequent of φ, then no safety augmentation is needed at v because the decision tree recommends a safe decision anyway. Therefore it suffices to consider only the cases where $\pi(v) \wedge ant(\varphi)$ is satisfiable, but $dec(v)$ refutes the consequent of φ. The necessity is obvious since $dec(v)$ refutes the consequent of φ, which will be a wrong decision for the cases satisfying $\pi(v) \wedge ant(\varphi)$.

The methodology for post-facto safety augmentation in the necessary leaf nodes identified by Lemma 2.1 is as follows:

- Let $\eta(\varphi)$ denote the set of variables referenced in $ant(\varphi)$ but not present in $\pi(v)$. For example, for the rightmost leaf node of Figure 1a, $eta(\varphi) = \{CapShape, GillColor\}$.
- A sequence of nodes corresponding to the variables in $\eta(\varphi)$ replaces the leaf node. This sequence is shown in red in Figure 1b.
- Decisions corresponding to the remaining branches of the new subtrees are recomputed using the decision tree learning algorithm.

The need for recomputing the decisions in the new branches of the decision tree arises when the decision taken at the original node was due to the statistical majority and not due to unanimity. In order to avoid overfitting, decision tree learning algorithms can take a specific decision at a node when the



(a) Decision Tree from Table 1.

(b) Decision Tree after Post-facto Safety Augmentation.

Figure 1: Decision trees for mushroom classification.

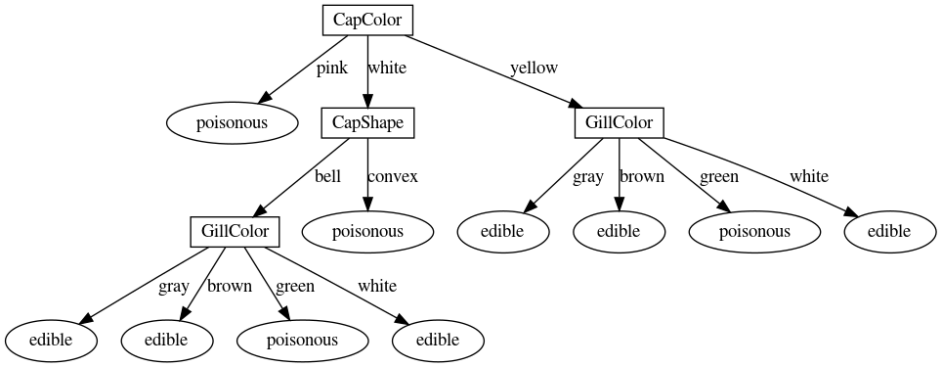


Figure 2: Decision Tree constructed from the rectified training dataset augmented by the support dataset of safety property by ID3 algorithm.

samples in favor of that decision is in overwhelming majority. The safety augmentation splits these samples into the new branches of the decision tree, and in some of the branches, the overwhelming majority may not hold anymore. Hence it is necessary to run the decision tree learning algorithm on the new branches.

3 Integrated Safety Augmentation

The post-facto safety augmentation approaches of Section 2 do not consider the information gain resulting out of the safety assertions, and may, therefore, produce safe decision trees which are suboptimal. For example, the decision tree shown in Figure 2 is also a safe decision tree, but it uses fewer variables in the rightmost branch as compared to Figure 1b. Specifically, the tree of Figure 2 takes advantage of the fact that GillColor is sufficient to separate the edible and poisonous mushrooms when CapColor is yellow. Such choices can be made when the safety augmentation is integrated with the learning algorithm of the decision tree.

We elucidate the integrated safety augmentation approach by considering the ID3 algorithm [Quinlan, 1986]. Given a set of training samples that are individually tagged with the decision, the key step of the algorithm is to choose the variable at the root of the decision tree based on an *information gain* metric. Once the root is chosen, the algorithm is recursively applied on each branch of the root corresponding to the various values that the root variable can take. Formally, the information gain, $IG(a|t)$, of attribute a at the end of branch t is:

$$IG(a|t) = E(a|t) - \sum_{j=1}^{l} \frac{N(t \wedge a[v_j])}{N(t)} E(t \wedge a[v_j]) \quad (1)$$

where:

- $E(a|t)$ is the entropy of the attribute a at the end of the current branch t,
- l is the cardinality of attribute a's domain,
- $N(t \wedge a[v_j])$ is the count of all decision variable class after branch t and attribute a's branch $a[v_j]$. Here $a[v_j]$ is the j^{th} value in attribute a's domain.

Also:

$$E(t) = \sum_{i=1}^{k} (-1).p(i|t).log(p(i|t)) \quad (2)$$

where, k is the number of different decision variable classes and $p(i)$ is the probability of decision variable class i, and therefore:

$$p(i|t) = \frac{TD(i|t)}{\sum_{j=1}^{k} TD(j|t)} = \frac{TD(i|t)}{TD(t)} \quad (3)$$

where,

- TD is the Training Dataset
- $TD(i|t)$ is the count of decision variable i,
- $TD(t)$ is the size of the total dataset at the end of the branch t.

At the root of the decision tree, branch t will be empty. The ID3 algorithm calculates the Information Gain for all the attributes using the above formulae and selects the attribute having the highest Information Gain as the root node. Each value of the chosen attribute becomes the branch of the root node. The training dataset is then divided corresponding to the decision tree branches. Then the same procedure is recursively applied on each of the branches with the remaining attributes and the data items, until all the data items have the same decision value (or we choose to prune the tree at that branch by choosing majority decision). We then add a leaf node with the decision value.

3.1 The Principle of Safety Augmentation

The reason why a decision tree constructed out of a given data set violates a safety assertion is that samples corresponding to the counter-examples were not present in the data. We may address this issue by augmenting the data set with additional samples that are relevant to the assertion. For example, consider the mushroom data set of Table 1 and the assertion, φ:

$$(CapShape = bell \wedge GillColor = green) \Rightarrow Poisonous$$

Table 1 does not contain rows for all combinations of attributes where $CapShape = bell$ and $GillColor = green$. Suppose we add the following rows in Table 1:

CapShape	CapColor	GillColor	Poisonous
Bell	White	Green	Poisonous
Bell	Yellow	Green	Poisonous

Adding these rows into the data set will ensure that the decision tree learned using ID3 is safe with respect to the assertion, φ. Formally, let $A = \{A_1, \ldots, A_n\}$ denote the set of attributes in the data set, and $Q \subseteq A$ denote the set of attributes referred in $ant(\varphi)$. Let $dom(A_i)$ denote the set of values of attribute A_i. The *safety augmentation data* corresponding to φ is defined as follows:

$$Aug(\varphi) = \{\langle v_1, \ldots, v_n, v_d\rangle \mid \forall i, v_i \in dom(A_i) \text{ and } \langle v_1, \ldots, v_n\rangle \text{ satisfies } ant(\varphi) \text{ and the decision variable } v_d \text{ satisfies the consequent of } \varphi\ \}$$

Lemma 3.1 *The decision tree constructed by the ID3 algorithm on the data set augmented with $Aug(\varphi)$ is safe with respect to assertion φ.*

Proof sketch: Let us assume the contrary. Then there exists a leaf node v in the decision tree such $dec(v)$ refutes the consequent of φ and $\pi(v) \wedge ant(\varphi)$ is satisfiable. But, in the augmented data set, by the definition of $Aug(\varphi)$, at least 50% of the samples at v have a decision which does to satisfy $dec(v)$. Therefore, ID3 can never prune the tree at v with a decision which disagrees with 50% of the samples.

3.2 The Safety Augmentation Methodology

The proposed safety augmentation methodology does not actually add $Aug(\varphi)$ explicitly in the data set. Essentially the ID3 algorithm chooses the attributes corresponding to each node in the decision tree using the information gain metric, and this metric only requires the *count* of the samples having a specific decision for each value in the domain of an attribute. The proposed methodology is implemented by manipulating these counts based on our understanding of $Aug(\varphi)$. Formally, we replace Equation 3 with the following:

$$p(i|t) = \frac{N(i|t)}{\sum_{j=1}^{k} N(j|t)} = \frac{N(i|t)}{N(t)} \tag{4}$$

where $N(i|t)$ it the count of the decision variable class i after the branch t and $N(t)$ is the count of all decision variable classes after the branch t. As we are also considering the count from $Aug(\varphi)$, we have:

$$N(i|t) = R(i|t) + S(i|t) \tag{5}$$

where, $R(i|t)$ is the count of decision variable class i in the input dataset after the branch t, and $S(i|t)$ is the count of decision variable class i corresponding to φ after the branch t.

The modified information gain is calculated thereafter in the standard way:

$$IG(a|t) = E(a|t) - \sum_{j=1}^{l} \frac{N(t \wedge a[v_j])}{N(t)} E(t \wedge a[v_j]) \tag{6}$$

3.3 Does Safety Augmentation Create Bias?

Post-facto safety augmentation does not affect the relative positions of the attributes in the decision tree, since the method splits those leaf nodes where the decision contradicts the safety requirement, while the other nodes in the tree are not disturbed. The integrated safety augmentation method produces a more optimal decision tree by augmenting the safety into the information gain metric and then allowing the ID3 algorithm to proceed with the modified information gain. In this section, we find out whether this augmentation introduces a bias in the input, thereby affecting the choice among the other attributes in the system.

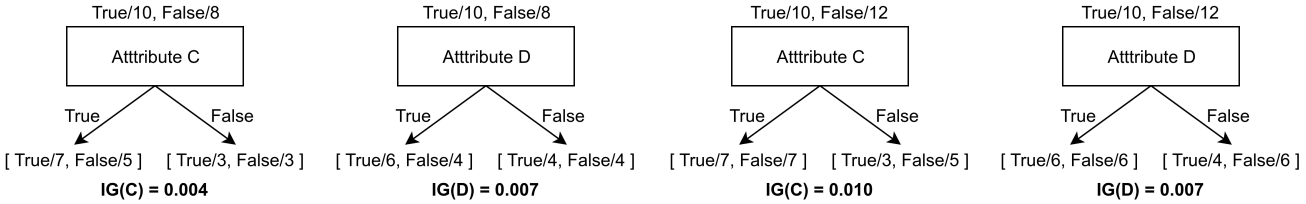
Let us return to our safety assertion, φ:

$$(CapShape = bell \wedge GillColor = green) \Rightarrow Poisonous$$

As explained in Section 3.1, the proposed integrated method will affect the count corresponding to the following cases:

CapShape	CapColor	GillColor	Poisonous
Bell	White	Green	Poisonous
Bell	Yellow	Green	Poisonous

The augmentation methodology inherently *assumes* that these types of mushrooms, which are not represented in the data, are actually possible. This need not be true, and in nature, we may not have a mushroom of yellow CapColor, which has a bell-like CapShape and green GillColor, as required by $ant(\varphi)$. Formally, safety augmentation assumes that the attributes referenced in $ant(\varphi)$ are *independent* of each other and that all combinations of values of these attributes can occur.



(a) Information Gain for the attribute C and D of the toy example without considering the support dataset.

(b) Information Gain for the attribute C and D of the toy example considering the support dataset.

Figure 3: Change in the Information Gain for the attribute C and D of the toy example for considering the support dataset.

Such an assumption may create a bias in the input to the decision tree learning algorithm. This is explained with the case shown in Figure 3. We consider the information gain of attributes C and D in a setting where C, D, and the decision variables are all Boolean. Figure 3a shows the case when no safety augmentation has been done. In this case, at a branch of the decision tree, we have 10 samples where the decision is True and 8 samples where the decision is False. Figure 3a shows the pattern in which the attributes C and D splits the samples with the decisions True and False. Accordingly, $IG(C)$ and $IG(D)$ are computed as shown, and therefore, ID3 prefers D over C as the next attribute.

Figure 3b shows the case when safety augmentation is used to modify the counts. In this case, the safe decision is False, hence only the counts of samples with decision False are affected by the augmentation. The original ratio of $(10 : 8)$ has now become $(10 : 12)$. Figure 3b shows the ratios of True/False decision when split by the the attributes C and D. Accordingly, $IG(C)$ and $IG(D)$ are computed as shown, and therefore, now ID3 prefers C over D as the next attribute.

The important thing to note here is that such an inversion of priority between attributes like C and D can happen even when they do not belong to the antecedent of an assertion, ϕ. For any attribute $\mathcal{A} \notin ant(\phi)$, the count of the samples corresponding to the safe decision will be equal on all branches leading out of a node for $\mathcal{A}$. In the specific case outlined above, C and D were not in $ant(\phi)$, and therefore the increase in the counts of the False decision in both the left and right branches of these nodes are equal, that is, 2. Yet the priority inversion takes place, indicating the introduction of bias.

We believe that enforcing safety always comes at the expense of bias. In reality, it is often not possible to envisage all practicable ways in which a property can fail. Some combinations of values for the variables in the antecedent of an assertion may not be possible, but this is not known to us. Therefore, the safety augmentation must assume that all combinations are possible, even at the expense of the bias it introduces. It is theoretically possible to examine ways to minimize the effects of this bias on the construction of the decision tree, but that is beyond the scope of this paper.

4 Multi-Assertion Safety Augmentation

In general, safety may be defined in terms of multiple assertions. Moreover, assertions may be causal, diagnostic, or a combination of both. In this section, we discuss these variants and show that the principle of safety augmentation presented in the previous section can be applied to all such cases.

- *Causal versus Diagnostic Assertions*. It is well known in the literature that in logic causal and diagnostic rules are inter-convertible. For example, the causal rule: *Cavity* $\Rightarrow$ *Toothache* can be rewritten as a diagnostic rule: $\neg$ *Toothache* $\Rightarrow$ $\neg$ *Cavity*. We have presented the safety augmentation methodology with assertions in causal form, where the consequent is a predicate over the decision variable. If the assertion is given in another form, it can be rewritten in the causal form and then used in our methodology.
- *Assertions with the same consequent*. Given two assertions of the form $\alpha_1 \Rightarrow \beta$ and $\alpha_2 \Rightarrow \beta$, we can perform the safety augmentation in any order, or together by augmenting the counts with the ways of satisfying $\alpha_1 \vee \alpha_2$.
- *Assertions with different consequents*. Given two assertions of the form $\alpha_1 \Rightarrow \beta_1$ and $\alpha_2 \Rightarrow \beta_2$, where $\beta_1 \neq \beta_2$, we can perform the safety augmentation in any order. In this case we additionally need to ensure that $\alpha_1 \wedge \alpha_2$ is unsatisfiable, otherwise the safety assertions contradict each other. This is done using a standard SAT engine.

In general, we may also have assertions of the form $\alpha \Rightarrow \beta$, where neither α, nor β contains the decision variable. Such assertions convey additional information about the variables, which may not be explicit in the data. Such assertions can also influence the counts during safety augmentation. For example, suppose the safety assertion states:

$$(CapShape = bell \wedge GillColor = green) \Rightarrow Poisonous$$

In the absence of any other information, as discussed earlier, we augment two cases, corresponding to CapColor of White and Yellow. Now suppose we are given the assertion:

$$(GillColor = Green) \Rightarrow (CapColor = Pink)$$

This assertion does not provide any information regarding the safety of mushrooms, but it tells us that mushrooms with green GillColor always have pink CapColor. With this information, the augmentation count drops from 2 to 0. The safety augmentation methodology computes the augmentation counts accordingly.

Dataset	ID	Assertion
Breast Cancer	1	$(Age = (30\text{-}39) \wedge Tumor\text{-}Size = (30\text{-}34) \wedge Irradiation = Yes) \Rightarrow Recurrence\text{-}Events$
Mushroom	1	$(Cap\text{-}Shape = Bell \wedge Gill\text{-}Color = Green) \Rightarrow Poisonous$
	2	$(Stalk\text{-}color\text{-}above\text{-}ring = Bell \wedge Stalk\text{-}color\text{-}below\text{-}ring = Green) \Rightarrow Poisonous$
Nursery	1	$(Student\text{-}Health = Not\text{-}Recommended) \Rightarrow Not\text{-}Recommended$
Tic-Tac-Toe	1	$(top\text{-}left\text{-}square = o \wedge top\text{-}middle\text{-}square = o \wedge top\text{-}right\text{-}square = o) \Rightarrow x\text{-}losses$
	2	$(middle\text{-}left\text{-}square = o \wedge middle\text{-}middle\text{-}square = o \wedge middle\text{-}right\text{-}square = o) \Rightarrow x\text{-}losses$
	3	$(bottom\text{-}left\text{-}square = o \wedge bottom\text{-}middle\text{-}square = o \wedge bottom\text{-}right\text{-}square = o) \Rightarrow x\text{-}losses$

Table 2: Assertions for Benchmark Datasets

Dataset	Prop ID	Original Decision Tree			Post-facto Safety Augmentation			Integrated Safety Augmentation		
		Depth	Total Nodes	Runtime (Sec)	Depth	Total Nodes	Runtime (Sec)	Depth	Total Nodes	Runtime (Sec)
Breast Cancer	1	8	179	0.014	8	202	0.001	7	181	0.012
Mushroom	1	5	29	0.284	7	281	0.002	6	60	0.346
	2	5	29	0.284	7	281	0.002	6	36	0.375
Nursery	1	9	803	0.275	9	803	0.001	9	803	0.260
Tic-Tac-Toe	1	8	343	0.034	8	343	0.001	8	318	0.035
	2	8	343	0.034	10	463	0.002	8	363	0.036
	3	8	343	0.034	10	514	0.002	8	310	0.035

Table 3: Comparison of runtimes and dimensions of decision trees

5 Experimental Results

We provide experimental results on four benchmark datasets from the UCI Machine Learning Repository [Dua and Graff, 2017], namely Breast Cancer [Zwitter and Soklic, 1988], Mushroom , Nursery, and Tic-Tac-Toe. The assertions are shown in Table 2 were designed based on relevance and have various degrees of support in the dataset.

Table 3 compares the depth and the total number of nodes in the decision trees created without safety augmentation with those created with post-facto and integrated safety augmentation. Also shown are the runtimes, for creating the decision trees without safety augmentation, the additional time for post-facto safety augmentation, and for creating the decision trees with integrated safety augmentation.

The effect of safety augmentation on the decision trees is dependent on the missing support for the safety assertions in the datasets. For example, the safety property for the Nursery dataset has most of the support data in the original support data and therefore requires negligible augmentation. On the other hand, support for the assertions over the mushroom dataset requires considerable augmentation, resulting in significant changes due to safety augmentation. Not surprisingly, integrated safety augmentation produces decision trees with fewer nodes and lesser depth as compared to post-facto safety augmentation.

The runtimes indicate that the overhead of safety augmentation is marginal in most cases.

6 Related Work

Decision trees and similar structures are already in use in critical safety domains. This includes Culpability Tree [Reason, 2016] for dealing with staff involved in safety errors in the aviation industry, Incident Decision Trees [Meadows *et al.*, 2005] for helping National Health Service (NHS) managers in the United Kingdom to determine a fair and consistent course of action towards staff involved in patient safety incidents. All these trees are either manually created by the domain experts or by using domain-specific tools. There exists learning algorithms and tools for building decision trees, including ones that are robust against malicious attacks such as data poisoning [Alfeld *et al.*, 2016] and evasion attacks [Biggio *et al.*, 2013]. The tool called Antidote [Drews *et al.*, 2019] is used to check the robustness of decision tree learning against data poisoning attacks, where an attacker can inject several malicious elements into the training set to influence the learned model. The decision tree learning algorithm Treant [Calzavara *et al.*, 2019] generates decision tree ensembles that are accurate and nearly insensitive to evasion attacks based on a formal threat model, and minimizes an evasion-aware loss function at each step of the tree construction. In contrast, our approaches build decision trees where some of the branches are responsible for predicting critical decisions, are modified according to the defined safety constraints, and thus provides the robustness against those defined safety constraints.

7 Conclusion

Safety augmentation is one of the primary requirements in structures learned from data using machine learning techniques, especially when the learned function is used in a safety critical context. This paper examines the safety augmentation problem for decision trees, which are popular in practice. We present the first methodology for safety augmentation in decision trees where the safety requirement is expressed in terms of assertions. Our results indicate that augmenting the information gain metrics with knowledge

gleaned from the safety assertions yields safe decision trees which are considerably smaller than ones obtained by post-facto safety augmentation.

References

[Alfeld *et al.*, 2016] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *30th AAAI Conference on Artificial Intelligence*, 2016.

[Biggio *et al.*, 2013] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[Breiman, 2017] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[Calzavara *et al.*, 2019] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. Treant: Training evasion-aware decision trees. *arXiv preprint arXiv:1907.01197*, 2019.

[Drews *et al.*, 2019] Samuel Drews, Aws Albarghouthi, and Loris D'Antoni. Proving data-poisoning robustness in decision trees. *arXiv preprint arXiv:1912.00981*, 2019.

[Dua and Graff, 2017] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[Laurent and Rivest, 1976] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[Meadows *et al.*, 2005] Sandra Meadows, Karen Baker, and Jeremy Butler. The incident decision tree: guidelines for action following patient safety incidents. In *Advances in Patient Safety: From Research to Implementation (Volume 4: Programs, Tools, and Products)*. Agency for Healthcare Research and Quality (US), 2005.

[Murthy, 1998] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[Peled *et al.*, 1999] Doron A. Peled, Edmund M. Clarke, and Orna Grumberg. *Model Checking*. MIT Press, 1999.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Quinlan, 2007] JR Quinlan. C5. http://rulequest.com, 2007.

[Quinlan, 2014] J Ross Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014.

[Reason, 2016] James Reason. *Managing the risks of organizational accidents*. Routledge, 2016.

[Song and Ying, 2015] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[SVA, 2018] SVA. (systemverilog assertions). *IEEE Std 1800-2017*, pages 1–1315, Feb 2018.

[Zwitter and Soklic, 1988] Matjaz Zwitter and Milan Soklic. Breast cancer data set, 1988.