Succinct Explanations with Cascading Decision Trees

Anonymous submission

Abstract

The decision tree is one of the most popular and classical machine learning models from the 1980s. However, in many practical applications, decision trees tend to generate decision paths with excessive depth. Long decision paths often cause overfitting problems, and make models difficult to interpret. With longer decision paths, inference is also more likely to fail when the data contain missing values. In this work, we propose a new tree model called Cascading Decision Trees to alleviate this problem. The key insight of Cascading Decision Trees is to separate the decision path and the explanation path. Our experiments show that on average, Cascading Decision Trees generate 63.38% shorter explanation paths, avoiding overfitting and thus achieve higher test accuracy. We also empirically demonstrate that Cascading Decision Trees have advantages in the robustness against missing values.

Introduction

Binary classification is the process of classifying the given input set into two classes based on some classification criteria. Binary classification is widely used in everyday life: for example, a typical application for binary classification is determining whether a patient has some disease by analyzing their comprehensive medical record. Existing work on binary classification mainly uses the accuracy of prediction as the main criterion for evaluating model performance. However, in order for a model to be useful in real-world applications, it is imperative that users are able to understand and explain the logic underlying model predictions. In many real-world applications, especially the medical and scientific domains, model comprehensibility[1] is of the utmost importance. In these cases, users need to understand the classification model to scientifically explain the reasons behind the classification or even rely on the model itself to discover a possible solution to the target problem.

There is a trade-off between enough explainability and high classification accuracy using current models. "Black-box" models such as deep neural networks, random forests, and ensemble methods tend to have the highest accuracy in binary classification (Freitas 2014; Došilović, Brčić, and Hlupić 2018). However, their opaque structure hinders understandability, making the logic behind the predictions difficult to trace. This lack of transparency may further discourage the use of the model (Augasta and Kathirvalavakumar 2012; Van Assche and Blockeel 2007).

Decision tree models, on the other hand, have transparent decision making steps. A traversal of features on the decision path from the root to the leaf node is presented to users as a rule. Therefore, compared to other models, the decision trees models have historically been characterized as having high comprehensibility (Freitas 2014; Došilović, Brčić, and Hlupić 2018). However, whether models generated by classic decision trees provide enough comprehensibility has been challenged. Decision trees can grow large enough that no human could verify their correctness (Caruana et al. 1999). Otherwise, they may contain subtrees with redundant attribute conditions, resulting in misinterpretation of the root cause for model predictions (Freitas 2014).

A succinct explanation also helps to avoid the overfitting problem. For the decision tree, the VC dimension grows exponentially with the depth of the tree (Shalev-Shwartz and Ben-David 2014). By the Fundamental Theorem of Learning Theory(Shalev-Shwartz and Ben-David 2014), the sample complexity for achieving probably approximately correct (PAC) learnability is proportional to the VC dimension. Combining these two facts, the sample complexity is exponential with respect to the depth of the decision tree. If the depth is not constrained, the exponential sample complexity usually leads to the overfitting problem in practice, because often it is the case that the size of training dataset cannot match such sample complexity. The depth is also highly related to the explanation length. Therefore, unless the volume of training data is large enough, which is usually not true, a succinct explanation for the model is favored.

This work introduces an algorithm for deriving succinct explanations for positive classifications, while maintaining the overall prediction accuracy. We target explanations for positive classifications[2], based on real-world motivation. In the medical domain, a positive classification result implies the presence of the disease (NIH 2022). The explanation for positive classification indicates the cause of the disease or some additional lab tests needed for a full diagnosis (CDC 2022). To this end, we introduce a novel *Cascading Deci-*

[1] In this paper, comprehensibility and interpretability are used interchangeably.

[2] Notice that succinct explanations for negative classifications can be achieved symmetrically by our algorithm.

sion Trees model. A Cascading Decision Trees model is a sequence of several smaller decision trees (subtrees) with a predefined tree depth. The sequence ends when the subtree does not contain any leaves describing positively classified samples. Figure 2 depicts one example of such a Cascading Decision Trees model.

The idea behind Cascading Decision Trees is that, while most algorithms for building decision trees are greedy and try to classify as many samples as soon as possible, such classifications result in large explanation paths for the samples in the lower levels of the tree. Instead, we construct a subtree of a predefined depth. That subtree contains a short explanation for the samples that it managed to classify. However, a subtree with a predefined depth may misclassify some samples. The next step of the Cascading Decision Trees algorithm is to repeat the process on the training set with the samples that were previously classified as positive removed from the set. This way, the samples classified as positive in the second subtree will have a much shorter explanation path than they would in the original decision tree. In Cascading Decision Trees, an explanation path for a positively classified sample is the path that starts at the root of the corresponding subtree.

Reducing the size and the depth of a decision tree to improve comprehensibility has been studied, both from a theoretical and a practical perspective. However, constructing such optimally small decision trees is an NP-complete problem (Hyafil and Rivest 1976), and the main drawback of these approaches is that the model is computationally too expensive to train. Even when using state-of-the-art libraries (Aglin, Nijssen, and Schaus 2020; Verwer and Zhang 2019), we observed the long running times due to high complexity. To illustrate this on an example, to learn a model on the Ionosphere dataset (from the UCI Machine Learning Repository), the BinOCT tool needs approximately 10 minutes, while our approach completes this task in 1.1 seconds.

We demonstrate the applicability of the Cascading Decision Trees model in two ways. First, we use our model to perform binary classification on ten benchmarks from UCI Machine Learning Repository, which are standard benchmarks for comparing the performance of tree-based classification methods (Zhao and Sudha Ram 2004). Second, we apply our model to a new application, namely continuous integration (CI) build status prediction (Santolucito et al. 2022). Overall, we report that compared to decision tree with PI-explanation (Shih, Choi, and Darwiche 2018; Izza, Ignatiev, and Marques-Silva 2020), our approach shortens the explanation depth for positive classifications by an average of 63.38% while still maintaining the prediction accuracy.

Related Work

Explainable Artificial Intelligence: XAI. The comprehensibility of classification models (Shih, Choi, and Darwiche 2018; Došilović, Brčić, and Hlupić 2018; Ignatiev, Narodytska, and Marques-Silva 2019; Audemard, Koriche, and Marquis 2020; Marques-Silva et al. 2020; Darwiche and Hirth 2020), in particular decision tree models (Izza, Ignatiev, and Marques-Silva 2020; Huang et al. 2021), has been extensively explored in XAI. Specifically, (Quinlan 1987) demonstrated that small decision trees are more interpretable than larger ones, and (Huysmans et al. 2011) ran a user study to illustrate that larger representations result in a decrease in both the accuracy of user's answers and user's confidence in the model itself. This work supports the motivation that minimizing explanations is a valuable direction to explore.

Ensemble Methods. The Cascading Decision Trees algorithm pursues a different goal than existing ensemble methods, such as Bagging (Breiman 1996) and Boosting (Schapire 1990). Ensemble methods combine multiple weaker classifiers into a larger classifier to increase overall accuracy, while Cascading Decision Trees focus on shortening explanations. The combination of ensemble methods obfuscate the traceability of a classic decision tree, which is the source of simple explanations for classifications (Zhao and Sudha Ram 2004).

Oblique Decision Trees. Oblique decision trees (Murthy, Kasif, and Salzberg 1994) extend the classic decision tree model by allowing each decision node to combine checks against multiple features. This series of work contains a rich family of models such as multivariate decision trees (Brodley and Utgoff 1995), loose and tight coupling (Gama and Brazdil 2000), and constrained cascade generalization of decision trees (Zhao and Sudha Ram 2004). However, compared to Cascading Decision Trees, this oblique design is not only expensive in the training stage (Lee and Jaakkola 2020), but it also obscures the explanation of a classification in the decision-making stage (Zhao and Sudha Ram 2004).

Optimal Decision Trees. Optimal decision trees learn models with the optimal prediction accuracy under different constraints, such as a certain number of leaves in the tree (Hu, Rudin, and Seltzer 2019) or restriction to complete binary trees of a certain depth (Verwer and Zhang 2019). However, learning an optimal decision tree without heuristics is an NP-complete problem (Hyafil and Rivest 1976). The main drawback of using this approach in practice is that the model is too computationally expensive to train. For example, state-of-the-art optimal decision trees implementations (Aglin, Nijssen, and Schaus 2020; Verwer and Zhang 2019) need ten minutes to train the ionosphere dataset[3]. On the same dataset, we show in our evaluation that our Cascading Decision Trees learning process terminated in seconds with competitive accuracy (cf. (Verwer and Zhang 2019)). In addition, instead of being a direct competitor, an optimal decision tree learner could be used as the base decision tree inducer in our algorithm to potentially improve the prediction accuracy, though at the cost of training time.

Motivating Examples

The following simple synthetic example is constructed to illustrate the basic functionality of Cascading Decision Trees. Given the dataset in Table 1, a classic decision tree will construct a model as shown in Figure 1. Using the same dataset, our Cascading Decision Trees algorithm generates a model with three subtrees as shown in Figure 2. Let us assume that there is a new sample ,`Sample11`, with the feature vector (F, F, F, T). Both models classify `Sample11`

[3]https://archive.ics.uci.edu/ml/datasets/Ionosphere

Table 1: Synthetic Dataset for Binary Classification.

	Feature1	Feature2	Feature3	Feature4	Label
Sample1	T	T	T	F	Positive
Sample2	T	T	F	F	Positive
Sample3	T	T	T	F	Positive
Sample4	T	T	T	F	Positive
Sample5	F	F	F	T	Positive
Sample6	F	T	F	F	Positive
Sample7	T	F	F	F	Negative
Sample8	F	T	T	F	Negative
Sample9	F	F	T	F	Negative
Sample10	F	T	F	F	Negative

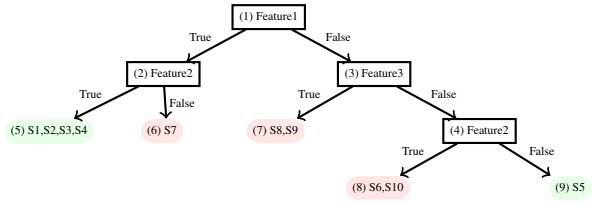

Figure 1: Example of a full classic decision tree on dataset given in Table 1, with green boxes represent Positive nodes and red boxes represent Negative nodes.

with the same prediction result, `Positive`. However, the explanations extracted from these two models are different.

In the classic decision tree model (Figure 1), `Sample11` falls into `node(9)`. Thus, the explanation path is `Feature1 = F`, `Feature2 = F` and `Feature3 = F`, with an explanation length of three.

Using Cascading Decision Trees (Figure 2), `Sample11` first falls into `node(7)` in the first subtree. `node(7)` is not classifying positive samples, so `Sample11` is passed to the second subtree and falls eventually into `node(10)`. This way, the explanation path for `Sample11` is `Feature4 = T`, with an explanation length of only one.

Invariant-based Explanation and Explanation Path

Invariant-based Explanation

For interpretability, we take inspiration from the LIME system (Ribeiro, Singh, and Guestrin 2016), which states that interpretable explanations of a model should "provide qualitative understanding between joint values of input variables and the resulting predicted response value". An explanation reasonably defined should capture the essence of the model's output. That is, the explanation identifies the key input values that contributed to the output of the model. The output of the model should be an invariant of the input values not included in the explanation.

Denote the input data by $x \in \mathcal{X}$ and the label of the class by $y \in \mathcal{Y}$. Let $f(x; w) : \mathcal{X} \to \mathcal{Y}$ be the model taking x as input and parameterized by w. Multiclass classification can be reduced to binary classification in natural ways. For simplicity, we specifically focus on binary classification tasks in this work. Hence we assume $\mathcal{Y} = \{0, 1\}$. Although there are various kinds of structures of $\mathcal{X}$, the most common one is $\mathcal{X} = \{0, 1\}^d$. Usually other typical features in $\mathcal{X}$ can be transformed to the form of $\{0, 1\}^d$. For example, categorical

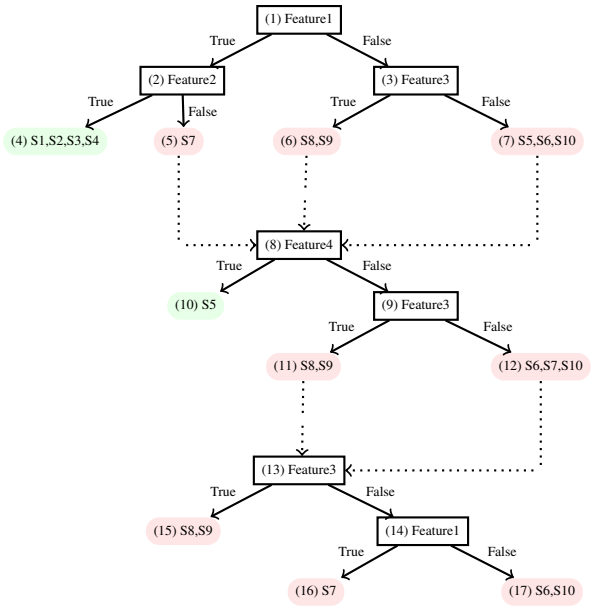

Figure 2: Example of Cascading Decision Trees on dataset given in Table 1, with maximum depth of two in each subtree.

feature $x \in \{C_1, \cdots, C_m\}$ can be converted to m dummy features $(z^{(1)}, \cdots, z^{(m)})$ where $z^{(i)} = \mathbb{1}[x = C_i]$.

For a model $f(x; w)$, we define a valid explanation $e^f(x; w)$ for x as definition 0.1.

Definition 0.1 (Valid Explanation)**.** *An explanation for x on model $f(x; w)$ is a vector $e^f(x; w) \in \{0, 1\}^d$. An explanation $e^f(x; w)$ is called a valid explanation for x on $f(x; w)$ if and only if for all $z \in \mathcal{X}$,*

$$f(x; w) = f\left((x * e^f(x; w)) \bigoplus (z * \neg e^f(x; w)); w\right)$$

where $*$ is the element-wise logical AND operator, $\bigoplus$ is the element-wise logical XOR operator, and $\neg$ is the element-wise NOT operator.

Lemma 0.2 shows why our definition of valid explanation is reasonable in the aspects of capturing the values in the input contributing to the model's output. If feature i is not picked by the explanation, i.e. $e^f(x; w)_i = 0$, then any disturbance on feature i would not change the model's output.

Lemma 0.2 (Invariant)**.** *Suppose x is a given input, $f(x; w)$ is a given model, and $e^f(x; w)$ is a valid explanation. If $x_i = y_i$ for all i such that $e^f(x; w)_i = 1$, then $f(y; w) = f(x; w)$.*

Proof. Suppose $x = (x_1, \cdots, y_d)$ and $y = (y_1, \cdots, y_d)$ satisfying $x_i = y_i$ when $e^f(x; w)_i = 1$. Construct $z \in \mathcal{X}$ as

$$z_i = \begin{cases} 0 & e^f(x; w)_i = 1 \\ y_i & e^f(x; w)_i = 0 \end{cases}$$

Thus we can verify that

$$y = (x * e^f(x; w)) \bigoplus (z * \neg e^f(x; w))$$

Since e^f is a valid explanation, $f(y; w) = f(x; w)$ holds. □

A well-known principle named Occam's razor posited the philosophy that shorter, simpler explanations are more likely to be true. Therefore, we define the length of a valid explanation by definition 0.3.

Definition 0.3 (Length of Valid Explanation)**.** *Suppose that $e^f(x;w)$ is a valid explanation, the length of $e^f(x;w)$ is*

$$|e^f(x;w)| = \sum_{i=1}^{d} e^f(x;w)_i$$

We provide the following examples to further illustrate our definition of a valid explanation.

Example 1 A trivial explanation is always available for any sample on any model - the explanation that every element in the input vector is important. Let $f(\cdot;w)$ be the decision tree described by Figure 1, and let input x be `Sample5`. Explanation $e^f(x;w) = (1,1,1,1)$ is a valid explanation. Notice that $\{i : e^f(x;w)_i = 0\} = \emptyset$, which means that as long as no features change, the output of the model remains the same. In this example, $|e^f(x;w)| = 4$.

Example 2 In general, a shorter explanation can be extracted from classic decision trees. As an example, the explanation for $x =$`Sample5` on the decision tree in Figure 1 is the explanation that keeps only the elements in the decision path of the tree: $e^f(x;w) = (1,1,1,0)$. This is the commonly accepted approach to explanations of decision tree classifications. In this example, $|e^f(x;w)| = 3$.

Example 3 An even shorter explanation is possible with Cascading Decision Trees. Looking to $x =$`Sample5` as classified by the Cascading Decision Trees in Figure 2, a valid explanation keeps only `Feature 4`: $e^f(x;w) = (0,0,0,1)$. Although changes to `Feature 1-3` may change the decision path, the output remains the same as long as `Feature 4` is unchanged. In this example, $|e^f(x;w)| = 1$.

Definition 0.4 (Valid Explanatory Model)**.** *The pair of $M = (f, e^f)$ is called an explanatory model, where f is the original model and e^f is the associated explanation. An explanatory model $M = (f, e^f)$ is valid if and only if $e^f(x;w)$ is valid for all $x \in \mathcal{X}$.*

Decision Path and Explanation Path in Decision Tree Models

In this work, we represent a decision tree model T by its nodes $V_T = \{\sigma_j\} \cup \{\tau_k\}$. Denote non-leaf nodes by $\sigma_j = (i_j, l_j, r_j) \in [d] \times V_T \times V_T$, where i_j means node σ_j splits into two branches based on value of feature x_{i_j}. The model goes to node l_j if $x_{i_j} = 1$ and goes to node r_j if $x_{i_j} = 0$. Denote leaf nodes by $\tau_k \in \{0,1\}$. The model outputs $f(x;w) = \tau_k$ if it finishes at leaf node τ_k.

We define decision path for an input on a model as all the nodes visited by the model in definition 0.5.

Definition 0.5 (Decision Path)**.** *Let $x \in \mathcal{X}$ be an input and $f(x;w)$ be a decision tree model. Suppose in the computation of $f(x;w)$ the model goes through nodes $\sigma_{j_1}, \cdots, \sigma_{j_m}, \tau_k$ sequentially, i.e. for $t \in [m-1]$*

$$\sigma_{j_{t+1}} = \begin{cases} l_{j_t} & \text{if } x_{i_{j_t}} = 1 \\ r_{j_t} & \text{if } x_{i_{j_t}} = 0 \end{cases} \text{ and } \tau_k = \begin{cases} l_{j_m} & \text{if } x_{i_{j_m}} = 1 \\ r_{j_m} & \text{if } x_{i_{j_m}} = 0 \end{cases}$$

then the decision path for x on f is $\langle \sigma_{j_1}, \cdots, \sigma_{j_m} \rangle$.

The explanation path is defined as a path that can be used to generate a valid explanation.

Definition 0.6 (Explanation Path)**.** *Let $x \in \mathcal{X}$ be an input and $f(x;w)$ be a decision tree model. A path $p = \langle \sigma_{j_1}, \cdots, \sigma_{j_m} \rangle$ is called an explanation path for x on f, if the explanation e is a valid explanation for x on f, where for $s \in [d]$,*

$$e^f(x;w)_s = \mathbb{1}[\exists \sigma_t \in p, i_t = s]$$

Notice that for an input x and a decision tree model $f(x;w)$, the output is always determined by the values of features on the decision path. Hence the decision path is always an explanation path as shown in lemma 0.7.

Lemma 0.7. *Let $x \in \mathcal{X}$ be an input and $f(x;w)$ be a decision tree model. Suppose $p = \langle \sigma_{j_1}, \cdots, \sigma_{j_m} \rangle$ be the decision path for x on f. Then p is an explanation path for x on f.*

Proof. Please see the proof in the supplemental material. □

Cascading Decision Trees

We introduce Cascading Decision Trees to show we can achieve a shorter explanation path by separating the decision path and explanation path on a decision tree model. Notice that cascading decision trees to generate shorter explanation path for negative samples can be built symmetrically as building the cascading decision trees for positive samples. So for simplicity, we only introduce the algorithm for building cascading decision trees for positive samples in this section.

Building Cascading Decision Trees. The cascading decision trees algorithm is described through pseudocode in Algorithm 1 (See in the supplementary material). Our insight is in the training process - it is the goal of each cascading decision subtree to identify the smallest subset of features that can classify as many `Positive` samples as possible, without misclassifying *any* `Negative` samples. Specifically, the procedure Fit builds a classic decision tree, `clf`, on our training set, $\mathcal{S}$ and adds it to the cascading tree list - initially this list is empty. For every leaf node we compute the `mixed` value, which is the percentage of samples from $\mathcal{S}$ classified by this leaf node that are also `Positive` samples. `Positive` nodes are leaf nodes with a `mixed` value greater than the threshold. If `clf` has no `Positive` nodes, it means we have learned a sufficiently good classifier and we stop (Line 11). Otherwise, we first remove samples truly classified by `Positive` nodes in this `clf` from $\mathcal{S}$, and then obtain a new $\mathcal{S}$ to use in the next iteration of the loop (Line 14).

Our cascading decision trees Algorithm 1 presented here has no pruning phase. However, our algorithm is generic enough to be combined with any pruning techniques (Esposito et al. 1997) and different goodness measurement for decision nodes split, such as entropy (Shannon 1948) and gini impurity (Havrda et al. 1967).

The time complexity of cascading decision tree algorithm is bounded by the size of the training set. Suppose the training time for building classic decision trees is a function of the number of the training samples, n and the number of features

d. We use the decision tree module in scikit-learn[4] as our base classic decision tree inducer, which is built upon the CART (Breiman et al. 1984) method. Since features are recursively reused in every decision node based on a numerical splitting criterion, the depth of the decision tree is bounded by the number of the training samples n in our model. Therefore, the time complexity for building one base decision tree is bounded by $\mathcal{O}(n^2 d)$.

According to our cascading decision trees Algorithm 1, after building one decision subtree, samples that are classified as `True Positive` are removed from the next round of decision subtree construction. In the worst case, every time, only one `True Positive` is classified in the current decision tree. The time complexity for building the next cascading decision subtree is in $\mathcal{O}(n^2 d)$. Therefore, the overall cascading decision trees training time T is bounded by: $\mathcal{O}(\sum_{m=1}^{n} m^2 d) = \mathcal{O}(\frac{n(n+1)(2n+1)}{6} d) = \mathcal{O}(n^3 d)$.

Cascading Decision Trees Inference. Our cascading decision trees testing process is described in Algorithm 2 in the supplementary material. In the procedure Test, we run all decision trees sequentially, and we report the decision path of only the classifying subtree as our explanation path. We also take the missing value into consideration. If in current decision tree we meet any missing value, we will skip the current decision tree and goes to the next decision tree.

Experiments

This section aims to evaluate the Cascading Decision Trees algorithm by answering the following questions:

1. How does the explanation length for Cascading Decision Trees compare to the baselines?
2. What is the prediction accuracy and efficiency of the Cascading Decision Trees algorithm?
3. How does the Cascading Decision Trees algorithm perform against missing data?
4. How does the Cascading Decision Trees algorithm perform in real-world application such as continuous integration (CI) build status prediction?

Experiment Setup. To evaluate Cascading Decision Trees algorithm, we collect ten binary classification datasets from UCI machine learning repository[5]. The UCI machine learning datasets are standard benchmarks for comparing the performance of tree-based classification methods (Zhao and Sudha Ram 2004). Additionally, since the underlying library, scikit-learn[6], of our implementation needs numerical features, Cascading Decision Trees adopts a standard one-hot encoder to add several indicator variables in the datasets.

Baselines. We compared the accuracy of our model against three baselines. They are 1) classic decision trees with PI-explanation, 2) sequential covering, and 3) classic decision trees with bounded depth. They were designed to provide better interpretability than the classic decision trees model while still preserving the prediction accuracy.

Decision trees with prime implicant (PI) explanations (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019; Audemard, Koriche, and Marquis 2020; Marques-Silva et al. 2020; Izza, Ignatiev, and Marques-Silva 2020; Darwiche and Hirth 2020) serve as our major baseline in this paper, since they target providing the minimal explanation to the decision tree model. PI-explanation is computed in the post-hoc fashion (decision rule pruning) for instances classified by the model (Shih, Choi, and Darwiche 2018; Izza, Ignatiev, and Marques-Silva 2020). It first builds a classic decision tree model and then computes a subset-minimal set of feature values that entail the same prediction. Thus it provides a explanation no longer than the original, unpruned decision tree model without any loss of prediction accuracy. We denote PI-explanation as the name of this baseline in our paper.

Sequential covering (Cohen 1995; Fürnkranz 1999) is a popular, separate-and-conquer rule learning algorithm based on the closed-world assumption. Every time, it learns a single rule, removes the data that this rule covers and then repeats the same process until the terminated condition is reached. It finally connects all of the learned rules sequentially during the training phase. Sequential covering has many variants, in our setting, every time, the algorithm tries to learn a rule that classifies as many positive examples as possible while keeping the negative examples not covered.

Classic decision trees with bounded depth uses the library in scikit-learn as our base classic decision tree inducer, which is built upon the CART (Breiman et al. 1984) method.

We quantify algorithm performance using five-fold cross validation, and randomly shuffle the datasets. The maximum depth of the Cascading Decision Trees algorithm is uniformly set to two in all tests. The θ in Cascading Decision Trees algorithm is set to vary from 0.5 to 0.9 based on the percentage of positive samples in the dataset. We ran all experiments on a Macbook Pro with an Inter i7 CPU and 16GB of RAM.

Explainability. The shorter an explanation, the more comprehensible that explanation is to users (Quinlan 1987; Huysmans et al. 2011; Pazzani 2000). Larger representations result in a decrease in both user's answer accuracy and confidence in the model itself (Huysmans et al. 2011).

Table 2: Average explanation length of positive classifications from PI-explanation and Cascading Decision Trees.

Dataset	PI-explanation	Cascading Decision Trees	Improvement
Breast cancer	7.854	2.000	74.5%
Heart Disease	4.377	2.000	54.3%
Wisc Breast cancer	2.658	1.991	25.1%
Cervical Cancer	7.408	2.000	73.0%
Credit Card	6.408	2.000	68.8%
German Credit	9.788	2.000	79.6%
Climate	2.945	2.000	32.1%
Happiness	7.315	2.000	72.7%
Ionosphere	2.694	1.418	47.4%
Sonar	3.813	1.943	49.0%

Table 2 shows the comparison of the average explanation length of the model generated by our Cascading Decision

[4]https://scikit-learn.org/stable/

[5]http://archive.ics.uci.edu/ml/index.php/

[6]For a fair comparison, the decision trees module in scikit-learn is also used for all three baselines in our evaluation.

Table 3: Breakdown Evaluation of Cascading Decision Trees on Ten UCI datasets.

Dataset	Method	Explanation Length	Accuracy	Runtime (s)	TP	TN	FP	FN	Precision	Recall	F-1 Score
Breast Cancer	Cascading Decision Trees	**2.000**	73.10%	0.392	4.4	38.0	2.6	13.0	67.56%	26.05%	36.33%
	PI-explanation	7.854	64.48%	220.0	7.4	30.0	10.6	10.0	40.74%	43.22%	40.96%
	Sequential Covering	4.888	63.45%	0.987	8.8	28.0	12.6	8.6	44.24%	48.16%	41.83%
	Classic (max_depth = 3)	3.000	71.03%	0.065	3.6	37.6	3.0	13.8	64.89%	21.72%	29.59%
Heart Disease	Cascading Decision Trees	**2.000**	74.07%	0.428	12.4	27.6	2.4	11.6	87.76%	51.22%	62.20%
	PI-explanation	4.377	72.96%	57.40	15.8	23.6	6.4	8.2	70.62%	66.19%	67.71%
	Sequential Covering	3.465	75.19%	0.737	16.2	24.4	5.6	7.8	74.17%	68.58%	70.92%
	Classic (max_depth = 3)	3.000	74.07%	0.061	12.8	27.2	2.8	11.2	84.70%	51.10%	60.97%
Wisc Breast Cancer	Cascading Decision Trees	**1.991**	93.51%	1.003	36.8	69.8	1.8	5.6	95.42%	86.57%	90.54%
	PI-explanation	2.658	93.16%	26.17	38.4	67.8	3.8	4.0	91.38%	90.47%	90.80%
	Sequential Covering	2.377	93.68%	1.018	39.4	67.4	4.2	3.0	90.25%	92.94%	91.51%
	Classic (max_depth = 3)	2.311	91.40%	0.077	34.8	69.4	2.2	7.6	93.88%	82.07%	87.53%
Cervical Cancer	Cascading Decision Trees	**2.000**	96.12%	1.013	8.0	120.8	4.2	1.0	67.18%	87.07%	75.04%
	PI-explanation	7.048	94.63%	25.43	4.8	122.0	3.0	4.2	64.85%	51.80%	56.49%
	Sequential Covering	5.383	95.37%	0.987	4.6	123.2	1.8	4.4	71.67%	50.02%	58.85%
	Classic (max_depth = 3)	3.000	95.52%	0.101	6.6	121.4	3.6	2.4	64.59%	71.51%	63.72%
Credit Card	Cascading Decision Trees	**2.000**	86.41%	0.743	57.6	55.6	3.6	14.2	94.15%	80.30%	86.67%
	PI-explanation	6.408	80.61%	490.8	59.4	46.2	13.0	12.4	82.29%	82.76%	82.37%
	Sequential Covering	4.656	81.07%	1.475	60.0	46.2	13.0	11.8	84.53%	83.34%	82.72%
	Classic (max_depth = 3)	3.000	85.95%	0.086	57.0	55.6	3.6	14.8	94.11%	79.37%	86.10%
German Credit	Cascading Decision Trees	**2.000**	71.80%	1.573	20.4	123.2	16.8	39.6	50.97%	33.23%	38.84%
	PI-explanation	9.788	66.90%	3173.4	28.2	105.6	34.4	31.8	45.06%	47.10%	45.93%
	Sequential Covering	4.774	68.40%	3.304	22.2	114.6	25.4	37.8	53.60%	37.88%	39.24%
	Classic (max_depth = 3)	3.000	73.00%	0.143	22.2	123.8	16.2	37.8	56.93%	36.75%	43.41%
Climate	Cascading Decision Trees	**2.000**	94.81%	1.640	96.8	5.6	3.6	2.0	96.41%	97.98%	97.18%
	PI-explanation	2.945	93.15%	77.14	95.0	5.6	3.6	3.8	96.34%	96.17%	96.25%
	Sequential Covering	2.455	90.74%	1.070	93.2	4.8	4.4	5.6	95.48%	94.31%	94.88%
	Classic (max_depth = 3)	2.541	91.11%	0.090	92.4	6.0	3.2	6.4	96.64%	93.51%	95.04%
Happiness	Cascading Decision Trees	**2.000**	61.37%	0.262	6.8	11.0	2.4	8.8	72.71%	44.01%	52.83%
	PI-explanation	7.315	56.55%	97.71	9.2	7.2	6.2	6.4	60.74%	59.03%	59.03%
	Sequential Covering	5.178	57.93%	0.607	10.8	6.0	7.4	4.8	64.42%	69.02%	62.25%
	Classic (max_depth = 3)	3.000	55.86%	0.053	6.4	9.8	3.6	9.2	60.79%	41.41%	46.89%
Ionosphere	Cascading Decision Trees	**1.418**	88.73%	1.063	20.4	42.6	3.0	5.0	88.28%	80.89%	84.37%
	PI-explanation	2.694	85.92%	14.09	20.0	41.0	4.6	5.4	81.77%	80.16%	80.68%
	Sequential Covering	1.681	88.45%	0.827	21.0	41.8	3.8	4.4	85.89%	83.06%	84.16%
	Classic (max_depth = 3)	1.483	84.23%	0.072	15.8	44.0	1.6	9.6	92.42%	61.57%	73.60%
Sonar	Cascading Decision Trees	**1.943**	66.19%	0.468	12.2	15.6	4.2	10.0	78.63%	55.92%	62.46%
	PI-explanation	3.813	74.29%	23.37	17.4	13.8	6.0	4.8	74.45%	78.29%	76.06%
	Sequential Covering	3.019	74.29%	0.721	17.4	13.8	6.0	4.8	75.28%	78.89%	76.29%
	Classic (max_depth = 3)	2.658	65.71%	0.071	11.0	16.6	3.2	11.2	79.75%	50.42%	60.32%

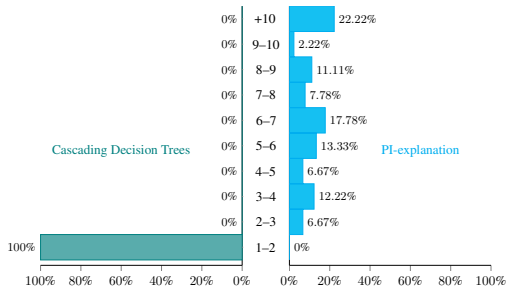

Figure 3: A distribution of explanation lengths for all the examples in the Breast Cancer dataset. The explanation length for Cascading Decision Trees model is no greater than two, while the explanation length of PI-explanation varies from two to more than ten.

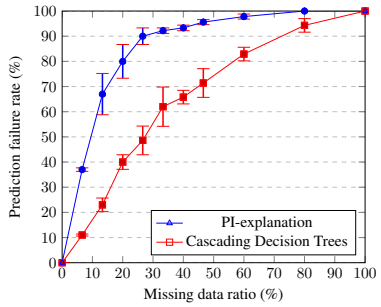

Figure 4: Robustness against missing data. Cascading Decision Trees vs. PI-explanation on the Breast Cancer dataset.

Trees and PI-explanations. Our Cascading Decision Trees algorithm shortens the explainable paths to users by 63.38% on average among ten datasets. The average representation size of the Cascading Decision Trees model is 1.93 among ten datasets. We specifically focus on the explanations for positive classifications. This means on average, only 1.93 features are necessary for the classification of a positive sample in Cascading Decision Trees. On the contrary, for PI-explanations, 5.27 features are necessary for a positive classification. Figure 3 shows the distribution of explanation lengths for all the examples in the Breast Cancer dataset. Due to the page limit, we present the results of the other seven datasets in the supplemental material. This succinctness of

the explanation improves users' ability to diagnose the causes for the diseases and support scientific hypothesis. We remark that PI-explanation could also be applied to Cascading Decision Tree. Let $\text{PI}(f, x)$ be the PI-explanation of model f on input x. By the definition of PI-explanation, we know that $|\text{PI}(f, x)| \leq |e^f(x; w)|$ always holds for all input x. As shown in Table 2, $\mathbb{E}\left[|e^{f_{\text{cascading}}}(x; w)|\right] \leq \mathbb{E}\left[|\text{PI}(f_{\text{classic}}, x)|\right]$ holds. Therefore we point out that

$$\begin{aligned}\mathbb{E}\left[|\text{PI}(f_{\text{cascading}}, x)|\right] &\leq \mathbb{E}\left[|e^{f_{\text{cascading}}}(x; w)|\right] \\ &\leq \mathbb{E}\left[|\text{PI}(f_{\text{classic}}, x)|\right]\end{aligned}$$

Accuracy. Table 3 shows the breakdown of the evaluation of Cascading Decision Trees. In nine out of ten datasets, our Cascading Decision Trees algorithm surprisingly outperforms the PI-explanation in test accuracy. This shows that long decision paths not only make models difficult to interpret, but also unavoidably lead to overfitting problems, which diminishes the overall test accuracy. To lower the explanation path of the decision trees model, one standard technique for classic decision trees is to fix a maximum depth. However, contrary to the Cascading Decision Trees, setting the maximum depth to three incurs a decrease in average prediction accuracy by around 2.0%. Even with this maximum depth and lower accuracy, in all ten datasets, the average explanation path is still longer than Cascading Decision Trees. The comparison between Cascading Decision Trees with sequential covering manifests in the same vein. In conclusion, compared with all three baselines, our Cascading Decision Trees delivers better model comprehensibility via shorter explanations while maintaining high prediction accuracy.

Low False-positive Rate. The Cascading Decision Trees model has another key advantage - the low false-positive rate in prediction, which is crucial in medical and scientific domains. For example, if the prediction result is positive (the presence of disease), the doctor needs to understand and explain the rationale behind the diagnosis of the disease and then report them to patients. A high false-positive rate not only reduces physicians' confidence in adopting the prediction result but also leads to unnecessary and invasive follow-up tests on patients (HNR 2022). Therefore, it is crucial to have a competitive accuracy prediction model with very low false-positive rate. As shown in Table 3, the Cascading Decision Trees algorithm has the lowest false positive (FP) rate in nine of ten datasets compared to the PI-explanation.

Robustness Against Missing Data. Our evaluation also demonstrates that Cascading Decision Tree has advantages in the robustness against missing data. For the PI-explanation, if any feature along the selected decision path contains a missing value in the testing sample, the model stops immediately and fails to make an inference. Therefore, the longer decision path is, the more likely it is for the inference to fail in the presence of missing data.

Based on the average 63.38% shorter decision paths than PI-explanation, we expect that Cascading Decision Tree will be robust against datasets with missing data. To test the ability of missing data handling in practice, we randomly selected part of the testing data as missing. Figure 4 confirms our expectation. illustrates that the Cascading Decision Tree has advantages in robustness against missing data in the Breast Cancer dataset. Due to the page limit, we present the results of the other nine datasets in the supplemental material.

Reproducibility. Source code and the data is publicly available at: https://doi.org/10.5281/zenodo.6011894.

Generalizing Cascading Decision Tree to Real-world Applications. We further apply the Cascading Decision Trees for predicting the build status in a novel real-world application, continuous integration (CI) build status prediction (Gousios, Pinzger, and Deursen 2014). In CI, users upload their code as one commit, and CI starts to build and test users' code under customized conditions such as the choice of operating system, library dependencies and other similar properties. However, one major drawback of CI is that the CI build attempt can be extremely time consuming. To address this issue, the existing work (Santolucito et al. 2022) first collected the historical repository data stored in the CI environment. It then took advantage of the transparent structure of the decision tree model to predict the CI build status.

This CI build status predicting problem is an ideal testbed for our Cascading Decision Trees, because the main module behind the tool is a classic decision tree for binary classification. Moreover, when using decision tree in such a program analysis domain, having a very low false-positive rate is crucial for user acceptance of the tool (Junker et al. 2012). We contacted the authors, adopted the same dataset and ran the same study they used with only one exception - substituting Cascading Decision Trees for classic decision tree in predicting the CI build status.

Table 4 (See in the supplementary material) shows the breakdown evaluation of using Cascading Decision Trees on CI build status prediction. The evaluation results show that the Cascading Decision Trees provides a shorter explanation, with a competitive prediction accuracy of 90.55%. The Cascading Decision Trees model shortens the explanation length for failed builds by 5.6%. Moreover, the ratio of the average number of false positives reports to the average number of correct classifications is only 4.4% (FP/TP+TN). Our study shows that the use of Cascading Decision Trees provides developers with a more succinct but comprehensible set of rules that are responsible for positive classifications.

Conclusions

Learning decision trees on modern datasets generates large trees, which in turn produce decision paths of excessive depth, obscuring the explanation of classifications. This paper intends to maximize model comprehensibility while maintaining prediction accuracy in binary classification. The Cascading Decision Trees algorithm has been proposed to provide more succinct explanations in binary decision trees. We evaluated our algorithm in real-world medical, scientific and program analysis datasets, where the explainability of the positive test result is of the utmost importance. Our Cascading Decision Trees algorithm shortens the explanation depth by over 63.38% for positive classifications compared to the classic decision trees with PI-explanations.

References

2022. CDC: Understanding a Positive Result. https://www.cdc.gov/hiv/basics/hiv-testing/positive-hiv-results.html/.

2022. National Cancer institute Dictionary of Cancer Terms. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/positive-test-result/.

2022. Understanding medical tests: sensitivity, specificity, and positive predictive value. https://www.healthnewsreview.org/toolkit/tips-for-understanding-studies/understanding-medical-tests-sensitivity-specificity-and-positive-predictive-value/.

Aglin, G.; Nijssen, S.; and Schaus, P. 2020. Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. 3146–3153. AAAI 2020.

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On Tractable XAI Queries based on Compiled Representations. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, 838–849.

Augasta, M. G.; and Kathirvalavakumar, T. 2012. Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems. *Neural Processing Letters*, 35(2): 131–150.

Breiman, L. 1996. Bagging Predictors. *Machine Learning*, 24(2): 123–140.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and regression trees*. Monterey, CA.

Brodley, C. E.; and Utgoff, P. E. 1995. Multivariate Decision Trees. *Machine Learning*, 19(1): 45–77.

Caruana, R.; Kangarloo, H.; Dionisio, J.; Sinha, U.; and Johnson, D. 1999. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, 212. American Medical Informatics Association.

Cohen, W. W. 1995. Fast Effective Rule Induction. In Prieditis, A.; and Russell, S., eds., *Machine Learning Proceedings 1995*, 115–123. San Francisco (CA): Morgan Kaufmann. ISBN 978-1-55860-377-6.

Darwiche, A.; and Hirth, A. 2020. On The Reasons Behind Decisions. arXiv:2002.09284.

Došilović, F. K.; Brčić, M.; and Hlupić, N. 2018. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 0210–0215.

Esposito, F.; Malerba, D.; Semeraro, G.; and Kay, J. 1997. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5): 476–491.

Freitas, A. A. 2014. Comprehensible Classification Models: A Position Paper. *SIGKDD Explor. Newsl.*, 15(1): 1–10.

Fürnkranz, J. 1999. Separate-and-Conquer Rule Learning. *Artif. Intell. Rev.*, 13(1): 3–54.

Gama, J.; and Brazdil, P. 2000. Cascade Generalization. *Machine Learning*, 41(3): 315–343.

Gousios, G.; Pinzger, M.; and Deursen, A. v. 2014. An Exploratory Study of the Pull-Based Software Development Model. ICSE 2014.

Havrda, J.; Charvát, F.; Havrda, J.; and Charvât, F. 1967. Quantification method of classification processes: Concept of structural a-entropy. *Kybernetika*.

Hu, X.; Rudin, C.; and Seltzer, M. I. 2019. Optimal Sparse Decision Trees. In *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 7265–7273.

Huang, X.; Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2021. On Efficiently Explaining Graph-Based Classifiers. In *KR*, 356–367.

Huysmans, J.; Dejaeger, K.; Mues, C.; Vanthienen, J.; and Baesens, B. 2011. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1): 141 – 154.

Hyafil, L.; and Rivest, R. L. 1976. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1): 15 – 17.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. On Relating Explanations and Adversarial Examples. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2020. On Explaining Decision Trees. arXiv:2010.11034.

Junker, M.; Huuck, R.; Fehnker, A.; and Knapp, A. 2012. SMT-based false positive elimination in static program analysis. In *International Conference on Formal Engineering Methods*, 316–331. Springer.

Lee, G.; and Jaakkola, T. S. 2020. Oblique Decision Trees from Derivatives of ReLU Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Marques-Silva, J.; Gerspacher, T.; Cooper, M.; Ignatiev, A.; and Narodytska, N. 2020. Explaining Naive Bayes and Other Linear Classifiers with Polynomial Time and Delay.

Murthy, S. K.; Kasif, S.; and Salzberg, S. 1994. A System for Induction of Oblique Decision Trees. *J. Artif. Int. Res.*, 2(1): 1–32.

Pazzani, M. J. 2000. Knowledge discovery from data? *IEEE Intelligent Systems and their Applications*, 15(2): 10–12.

Quinlan, J. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3): 221 – 234.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 1135–1144. ACM.

Santolucito, M.; Zhang, J.; Zhai, E.; Cito, J.; and Piskac, R. 2022. Learning CI Configuration Correctness for Early Build Feedback. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 1006–1017.

Schapire, R. E. 1990. The Strength of Weak Learnability. *Mach. Learn.*, 5(2): 197–227.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press. ISBN 1107057132.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A Symbolic Approach to Explaining Bayesian Network Classifiers. 5103–5111. ijcai.org.

Van Assche, A.; and Blockeel, H. 2007. Seeing the Forest Through the Trees: Learning a Comprehensible Model from an Ensemble. In *Machine Learning: ECML 2007*, 418–429. Berlin, Heidelberg: Springer Berlin Heidelberg.

Verwer, S.; and Zhang, Y. 2019. Learning Optimal Classification Trees Using a Binary Linear Program Formulation. 1625–1632. AAAI 2019.

Zhao, H.; and Sudha Ram. 2004. Constrained cascade generalization of decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 16(6): 727–739.