Harnessing the Power of Choices in Decision Tree Learning

Guy Blanc[*]
Stanford
gblanc@stanford.edu

Jane Lange[*]
MIT
jlange@mit.edu

Chirag Pabbaraju[*]
Stanford
cpabbara@stanford.edu

Colin Sullivan[*]
Stanford
colins26@stanford.edu

Li-Yang Tan[*]
Stanford
lytan@stanford.edu

Mo Tiwari[*]
Stanford
motiwari@stanford.edu

Abstract

We propose a simple generalization of standard and empirically successful decision tree learning algorithms such as ID3, C4.5, and CART. These algorithms, which have been central to machine learning for decades, are greedy in nature: they grow a decision tree by iteratively splitting on the best attribute. Our algorithm, Top-k, considers the k best attributes as possible splits instead of just the single best attribute.We demonstrate, theoretically and empirically, the power of this simple generalization. We first prove a *greediness hierarchy theorem* showing that for every $k \in \mathbb{N}$, Top-$(k+1)$ can be dramatically more powerful than Top-k: there are data distributions for which the former achieves accuracy $1-\varepsilon$, whereas the latter only achieves accuracy $\frac{1}{2}+\varepsilon$. We then show, through extensive experiments, that Top-k outperforms the two main approaches to decision tree learning: classic greedy algorithms and more recent "optimal decision tree" algorithms. On one hand, Top-k consistently enjoys significant accuracy gains over greedy algorithms across a wide range of benchmarks. On the other hand, Top-k is markedly more scalable than optimal decision tree algorithms and is able to handle dataset and feature set sizes that remain far beyond the reach of these algorithms. The code to reproduce our results is available at: `https://github.com/SullivanC19/pydl8.5-topk`.

1 Introduction

Decision trees are a fundamental workhorse in machine learning. Their logical and hierarchical structure makes them easy to understand and their predictions easy to explain. Decision trees are therefore the most canonical example of an interpretable model: in his influential survey [Bre01b], Breiman writes "On interpretability, trees rate an A+"; much more recently, the survey [RCC+22] lists decision tree optimization as the very first of 10 grand challenges for the field of interpretable machine learning. Decision trees are also central to modern ensemble methods such as random forests [Bre01a] and XGBoost [CG16], which achieve state-of-the-art accuracy for a wide range of tasks.

Greedy algorithms such as ID3 [Qui86], C4.5 [Qui93], and CART [BFSO84] have long been the standard approach to decision tree learning. These algorithms build a decision tree from labeled data in a top-down manner, growing the tree by iteratively splitting on the "best" attribute as measured with respect to a certain heuristic function (e.g., information gain). Owing to their simplicity, these

[*] Authors ordered alphabetically.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

algorithms are highly efficient and scale gracefully to handle massive datasets and feature set sizes, and they continue to be widely employed in practice and enjoy significant empirical success. For the same reasons, these algorithms are also part of the standard curriculum in introductory machine learning and data science courses.

The trees produced by these greedy algorithms are often reasonably accurate, but can nevertheless be suboptimal. There has therefore been a separate line of work, which we review in Section 2, on algorithms that optimize for accuracy and seek to produce *optimally* accurate decision trees. These algorithms employ a variety of optimization techniques (including dynamic programming, integer programming, and SAT solvers) and are completely different from the simple greedy algorithms discussed above. Since the problem of finding an optimal decision tree has long been known to be NP-hard [HR76], *any* algorithm must suffer from the inherent combinatorial explosion when the instance size becomes sufficiently large (unless P=NP). Therefore, while this line of work has made great strides in improving the scalability of algorithms for optimal decision trees, dataset and feature set sizes in the high hundreds and thousands remain out of reach.

This state of affairs raises a natural question:

> *Can we design decision tree learning algorithms that improve significantly on the accuracy of classic greedy algorithms and yet inherit their simplicity and scalability?*

In this work, we propose a new approach and make a case that provides a strong affirmative answer to the question above. Our work also opens up several new avenues for exploration in both the theory and practice of decision tree learning.

1.1 Our contributions

1.1.1 Top-k: a simple and effective generalization of classic greedy decision tree algorithms

We introduce an easily interpretable greediness parameter to the class of all greedy decision tree algorithms, a broad class that encompasses ID3, C4.5, and CART. This parameter, k, represents the number of features that the algorithm considers as candidate splits at each step. Setting $k = 1$ recovers the fully greedy classical approaches, and increasing k allows the practitioner to produce more accurate trees at the cost of only a mild training slowdown. The focus of our work is on the regime where k is a small constant—preserving the efficiency and scalability of greedy algorithms is a primary objective of our work—although we mention here that by setting k to be the dimension d, our algorithm produces an optimal tree. Our overall framework can thus be viewed as interpolating between greedy algorithms at one extreme and "optimal decision tree" algorithms at the other, precisely the two main and previously disparate approaches to decision tree learning discussed above.

We will now describe our framework. A *feature scoring function* $\mathcal{H}$ takes as input a dataset over d binary features and a specific feature $i \in [d]$, and returns a value quantifying the "desirability" of this feature as the root of the tree. The greedy algorithm corresponding to $\mathcal{H}$ selects as the root of the tree the feature that has the largest score under $\mathcal{H}$; our generalization will instead consider the k features with the k highest scores.

Definition 1 (Feature scoring function)**.** *A feature scoring function $\mathcal{H}$ takes as input a labeled dataset S over a d-dimensional feature space, a feature $i \in [d]$, and returns a score $\nu_i \in [0, 1]$.*

See Section 3.1 for a discussion of the feature scoring functions that correspond to standard greedy algorithms ID3, C4.5, and CART. Pseudocode for Top-k is provided in Figure 1. We note that from the perspective of interpretability, the trained model looks the same regardless of what k is. During training, the algorithm considers more splits, but only one split is eventually used at each node.

1.1.2 Theoretical results on the power of Top-k

The search space of Top-$(k + 1)$ is larger than that of Top-k, and therefore its training accuracy is certainly at least as high. The first question we consider is: is the test accuracy of Top-$(k + 1)$ only marginally better than that of Top-k, or are there examples of data distributions for which even a single additional choice provably leads to huge gains in test accuracy? Our first main theoretical result is a sharp *greediness hierarchy theorem*, showing that this parameter can have dramatic impacts on accuracy, thereby illustrating its power:

```
Top-$k(\mathcal{H}, S, h)$:
  Given: A feature scoring function $\mathcal{H}$, a labeled sample set $S$ over $d$ dimensions, and depth budget $h$.
  Output: Decision tree of depth $h$ that approximately fits $S$.
    1. If $h = 0$, or if every point in $S$ has the same label, return the constant function with the best accuracy w.r.t. $S$.
    2. Otherwise, let $\mathcal{I} \subseteq [d]$ be the set of $k$ coordinates maximizing $\mathcal{H}(S, i)$.
    3. For each $i \in \mathcal{I}$, let $T_i$ be the tree with
           Root $= x_i$
           Left subtree $=$ Top-$k(\mathcal{H}, S_{x_i=0}, h-1)$
           Right subtree $=$ Top-$k(\mathcal{H}, S_{x_i=1}, h-1)$,
       where $S_{x_i=b}$ is the subset of points in $S$ where $x_i = b$.
    4. Return the $T_i$ with maximal accuracy with respect to $S$ among all choices of $i \in \mathcal{I}$.
```

Figure 1: The Top-k algorithm. It can be instantiated with any feature scoring function $\mathcal{H}$, and when $k = 1$, recovers standard greedy algorithms such as ID3, C4.5, and CART.

Theorem 1 (Greediness hierarchy theorem)**.** *For every $\varepsilon > 0$, $k, h \in \mathbb{N}$, there is a data distribution $\mathcal{D}$ and sample size n for which, with high probability over a random sample $\boldsymbol{S} \sim \mathcal{D}^n$,* Top-$(k+1)$ *achieves at least $1 - \varepsilon$ accuracy with a depth budget of h, but* Top-k *achieves at most $\frac{1}{2} + \varepsilon$ accuracy with a depth budget of h.*

All of our theoretical results, Theorems 1 to 3, hold whenever the scoring function is an *impurity-based heuristic*. This broad class includes the most popular scoring functions (see Section 3.1 for more details). Theorem 1 is a special case of a more general result that we show: for all $k < K$, there are data distributions on which Top-K achieves maximal accuracy gains over Top-k, even if Top-k is allowed a larger depth budget:

Theorem 2 (Generalization of Theorem 1)**.** *For every $\varepsilon > 0$, $k, K, h \in \mathbb{N}$ where $k < K$, there is a data distribution $\mathcal{D}$ and sample size n for which, with high probability over a random sample $\boldsymbol{S} \sim \mathcal{D}^n$,* Top-$K$ *achieves at least $1 - \varepsilon$ accuracy with a depth budget of h, but* Top-k *achieves at most $\frac{1}{2} + \varepsilon$ accuracy even with a depth budget of $h + (K - k - 1)$.*

The proof of Theorem 2 is simple and highlights the theoretical power of choices. One downside, though, is that it is based on data distributions that are admittedly somewhat unnatural: the labeling function has embedded within it a function that is the XOR of certain features, and real-world datasets are unlikely to exhibit such adversarial structure. To address this, we further prove that the power of choices is evident even for *monotone* data distributions. We defer the definition of monotone data distributions to Section 4.2.

Theorem 3 (Greediness hierarchy theorem for monotone data distributions)**.** *For every $\varepsilon > 0$, depth budget h, K between $\tilde{\Omega}(h)$ and $\tilde{O}(h^2)$ and $k \leq K - h$, there is a monotone data distribution $\mathcal{D}$ and sample size n for which, with high probability over a random sample $\boldsymbol{S} \sim \mathcal{D}^n$, Top-$K$ achieves at least $1 - \varepsilon$ accuracy with a depth budget of h, but Top-k achieves at most $\frac{1}{2} + \varepsilon$ accuracy with a depth budget of h.*

Many real-world data distributions are monotone in nature, and relatedly, they are a common assumption and the subject of intensive study in learning theory. Most relevant to this paper, recent theoretical work has identified monotone data distributions as a broad and natural class for which classical greedy decision tree algorithms (i.e., Top-1) provably succeed [BLT20b, BLT20a]. Theorem 3 shows that even within this class, increasing the greediness parameter can lead to dramatic gains in accuracy. Compared to Theorem 2, the proof of Theorem 3 is more technical and involves the use of concepts from the Fourier analysis of boolean functions [O'D14].

We note that a weaker version of Theorem 3 is implicit in prior work: combining [BLT20b, Theorem 7b] and [BLQT21b, Theorem 2] yields the special case of Theorem 3 where $K = O(h^2)$ and $k = 1$. Theorem 3 is a significant strengthening as it allows for $k > 1$ and much smaller $K - k$.

1.1.3 Experimental results on the power of Top-k

We provide extensive empirical validation of the effectiveness of Top-k when trained on on real-world datasets, and provide an in-depth comparison with both standard greedy algorithms as well as optimal decision tree algorithms.

We first compare the performance of Top-k for $k = 1, 2, 3, 4, 8, 12, 16$ (Figure 2), and find that increasing k does indeed provide a significant increase in test accuracy—in some cases, Top-8 already achieves accuracy comparable to the test accuracy attained by DL8.5 [ANS20], an optimal decision tree algorithm. We further show, in Figures 3 and 6, that Top-k inherits the efficiency of popular greedy algorithms and scales much better than the state-of-the-art optimal decision tree algorithms MurTree and GOSDT [LZH⁺20].

Taken as a whole, our experiments demonstrate that Top-k provides a useful middle ground between greedy and optimal decision tree algorithms: it is significantly more accurate than greedy algorithms, but still fast enough to be practical on reasonably large datasets. See Section 5 for an in-depth discussion of our experiments. Finally, we emphasize the benefits afforded by the simplicity of Top-k. Standard greedy algorithms (i.e. Top-1) are widely employed and easily accessible. Introducing the parameter k requires modifying only a tiny amount of source code and gives the practitioner a new lever to control. Our experiments and theoretical results demonstrate the utility of this simple lever.

2 Related work

Provable guarantees and limitations of greedy decision tree algorithms. A long and fruitful line of work seeks to develop a rigorous understanding of the performances of greedy decision tree learning algorithms such as ID3, C4.5, and CART and to place their empirical success on firm theoretical footing [KM96, Kea96, DKM96, BDM19, BDM20, BLT20b, BLT20a, BLQT21a]. These works identify feature and distributional assumptions under which these algorithms provably succeed; they also highlight the *limitations* of these algorithms by pointing out settings in which they provably fail. Our work complements this line of work by showing, theoretically and empirically, how these algorithms can be further improved with a simple new parameter while preserving their efficiency and scalability.

The work of [BLQT21b]. Recent work of Blanc, Lange, Qiao, and Tan also highlights the power of choices in decision tree learning. However, they operate within a stylized theoretical setting. First, they consider a specific scoring function that is based on a notion of *influence* of features, and crucially, computing these scores requires *query access* to the target function (rather than from random labeled samples as is the case in practice). Furthermore, their results only hold with respect to the uniform distribution. These are strong assumptions that limit the practical relevance of their results. In contrast, a primary focus of this work is to be closely aligned with practice, and in particular, our framework captures and generalizes the standard greedy algorithms used in practice.

Optimal decision trees. Motivated in part by the surge of interest in interpretable machine learning and the highly interpretable nature of decision trees, there have been numerous works on learning *optimal* decision trees [BD17, VZ17, VZ19, AAV19, ZMP⁺20, VNP⁺20, NIPMS18, Ave20, JM20, NF07, NF10, HRS19, LZH⁺20, DLH⁺22]. As mentioned in the introduction, this is an NP-complete problem [HR76]—indeed, it is NP-hard to find even an approximately optimal decision tree [Sie08, AH08, ABF⁺09]. Due to the fundamental intractability of this problem, even highly optimized versions of algorithms are unlikely to match the scalability of standard greedy algorithms. That said, these works implement a variety of optimizations that allow them to build optimal decision trees for many real world datasets when the dataset and feature sizes are in the hundreds and the desired depth is small (≤ 5).

Finally, another related line of work is that of *soft* decision trees [IYA12, TAA⁺19]. These works use gradient-based methods to learn soft splits at each internal node. We believe that one key advantage of our work over these soft trees is in interpretability. With Top-k, since the splits are hard (and not soft), to understand the classification of a test point, it is sufficient to look at only one root-to-leaf path, as opposed to a weighted combination across many.

3 The Top-k algorithm

3.1 Background and context: Impurity-based algorithms

Greedy decision tree learning algorithms like ID3, C4.5 and CART are all instantiations of Top-k in Figure 1 with $k = 1$ and an appropriate choice of the feature-scoring function $\mathcal{H}$. Those three algorithms all used *impurity-based heuristics* as their feature-scoring function:

Definition 2 (Impurity-based heuristic)**.** *An* impurity function $\mathcal{G} : [0, 1] \to [0, 1]$ *is a function that is concave, symmetric about* 0.5*, and satisfies* $\mathcal{G}(0) = \mathcal{G}(1) = 0$ *and* $\mathcal{G}(0.5) = 1$*. A feature-scoring function* $\mathcal{H}$ *is an* impurity-based heuristic*, if there is some impurity function* $\mathcal{G}$ *for which:*

$$\mathcal{H}(S, i) = \mathcal{G}\left(\mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim S}[\boldsymbol{y}]\right) - \Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[\boldsymbol{x}_i = 0] \cdot \mathcal{G}\left(\mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim S}[\boldsymbol{y} \mid \boldsymbol{x}_i = 0]\right)$$
$$- \Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[\boldsymbol{x}_i = 1] \cdot \mathcal{G}\left(\mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim S}[\boldsymbol{y} \mid \boldsymbol{x}_i = 1]\right)$$

where in each of the above, $(\boldsymbol{x}, \boldsymbol{y})$ *are a uniformly random point from within* S*.*

Common examples for the impurity function include the binary entropy function $\mathcal{G}(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ (used by ID3 and C4.5), the Gini index $\mathcal{G}(p) = 4p(1-p)$ (used by CART), and the function $\mathcal{G}(p) = 2\sqrt{p(1-p)}$ (proposed and analyzed in [KM99]). We refer the reader to [KM99] for a theoretical comparison, and [DKM96] for an experimental comparison, of these impurity-based heuristics.

Our experiments focus on binary entropy being the impurity measure, but our theoretical results apply to Top-k instantiated with *any* impurity-based heuristic.

3.2 Basic theoretical properties of the Top-k algorithm

Running time. The key behavioral aspect in which Top-k differs from greedy algorithms is that it is less greedy when trying to determine which coordinate to query. This naturally increases the running time of Top-k, but that increase is fairly mild. More concretely, suppose Top-k is run on a dataset S with n points. We can then easily derive the following bound on the running time of Top-k, where $\mathcal{H}(S, i)$ is assumed to take $O(n)$ time to evaluate (as it does for all impurity-based heuristics).

Claim 3.1. *The running time of Top-*$k(\mathcal{H}, S, h)$ *is* $O((2k)^h \cdot nd)$*.*

Proof. Let T_h be the number of recursive calls made by Top-$k(\mathcal{H}, S, h)$. Then, we have the simple recurrence relation $T_h = 2kT_{h-1}$, where $T_0 = 1$. Solving this recurrence gives $T_h = (2k)^h$. Each recursive call takes $O(nd)$ time, where the bottleneck is scoring each of the d features. □

We note that any decision tree algorithm, including fast greedy algorithms such as ID3, C4.5, and CART, has runtime that scales exponentially with the depth h. The size of a depth-h tree can be 2^h, and this is of course a lower bound on the runtime as the algorithm needs to output such a tree. In contrast with greedy algorithms (for which $k = 1$), Top-k incurs an additional k^h cost in running time. As mentioned earlier, in practice, we are primarily concerned with fitting small decision trees (e.g., $h = 5$) to the data, as this allows for explainable predictions. In this setting, the additional k^h cost (for small constant k) is inexpensive, as confirmed by our experiments.

The search space of Top-k: We state and prove a simple claim that Top-k returns the *best* tree within its search space.

Definition 3 (Search space of Top-k)**.** *Given a sample* S *and integers* h, k*, we use* $\mathcal{T}_{k,h,S}$ *to refer to all trees in the search space of Top-k. Specifically, if* $h = 0$*, this contains all trees with a height of zero (the constant* 0 *and constant* 1 *trees). For* $h \geq 1$*, and* $\mathcal{I} \subseteq [d]$ *being the* k *coordinates with maximal score, this contains all trees with a root of* x_i*, left subtree in* $\mathcal{T}_{k,h-1,S_{x_i=0}}$ *and right subtree in* $\mathcal{T}_{k,h-1,S_{x_i=1}}$ *for some* $i \in \mathcal{I}$*.*

Lemma 3.2 (Top-k chooses the most accurate tree in its search space)**.** *For any sample S and integers h, k, let T be the output of Top-k with a depth budget of h on S. Then*

$$\Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[T(\boldsymbol{x}) = \boldsymbol{y}] = \max_{T' \in \mathcal{T}_{k,h,S}} \left(\Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[T'(\boldsymbol{x}) = \boldsymbol{y}] \right).$$

We refer the reader to Appendix A for the proof of this lemma.

4 Theoretical bounds on the power of choices

We refer the reader to the Appendix B for most of the setup and notation. For now, we briefly mention a small amount of notation relevant to this section: we use **bold font** (e.g. $\boldsymbol{x}$) to denote random variables. We also use bold font to indicate *stochastic functions* which output a random variable. For example,

$$\boldsymbol{f}(x) := \begin{cases} x & \text{with probability } \frac{1}{2} \\ -x & \text{with probability } \frac{1}{2} \end{cases}$$

is the stochastic function that returns either the identity or its negation with equal probability. To define the data distributions of Theorems 2 and 3, we will give a distribution over the domain, X and the stochastic function that provides the label given an element of the domain.

Intuition for proof of greediness hierarchy theorem To construct a distribution which Top-k fits poorly and Top-$(k+1)$ fits well, we will partition features into two groups: one group consisting of features with medium correlation to the labels and another group consisting of features with high correlation when taken all together but low correlation otherwise. Since the correlation of features in the former group is larger than that of the latter group unless all features from the latter group are considered, both algorithms will prioritize features from the former group. However, if the groups are sized correctly, then Top-$(k+1)$ will consider splitting on all features from the latter group, whereas Top-k will not. As a result, Top-$(k+1)$ will output a decision tree with higher accuracy.

4.1 Proof of Theorem 2

For each depth budget h and search branching factor K, we will define a hard distribution $\mathcal{D}_{h,K}$ that is learnable to high accuracy by Top-K with a depth of h, but not by Top-k with a depth of h' for any $h' < h + K - k$. This distribution will be over $\{0,1\}^d \times \{0,1\}$, where $d = h + K - 1$. The marginal distribution over $\{0,1\}^d$ is uniform, and the distribution over $\{0,1\}$ conditioned on a setting of the d features is given by the stochastic function $\boldsymbol{f}_{h,K}(x)$. All of the results of this section (Theorems 2 and 3) hold when the feature scoring function is *any* impurity-based heuristic.

Description of $\boldsymbol{f}_{h,K}(x)$. Partition x into two sets of variables, $x^{(1)}$ of size h and $x^{(2)}$ of size $K-1$. Let $\boldsymbol{f}_{h,K}(x)$ be the randomized function defined as follows:

$$\boldsymbol{f}_{h,K}(x) = \begin{cases} \mathrm{Par}_h(x^{(1)}) & \text{with probability } 1-\varepsilon \\ x_i^{(2)} \sim \mathrm{Unif}[x^{(2)}] & \text{with probability } \varepsilon, \end{cases}$$

where $\mathrm{Unif}[x^{(2)}]$ denotes the uniform distribution on $x^{(2)}$. $\mathrm{Par}_h(x^{(1)})$ is the parity function, whose formal definition can be found in Appendix B.

The proof of Theorem 2 is divided into two parts. First, we prove that when the data distribution is $\mathcal{D}_{h,K}$, Top-K succeeds in building a high accuracy tree with a depth budget of h. Then, we show that Top-k fails and builds a tree with low accuracy, even given a depth budget of $h + (K - k - 1)$.

Lemma 4.1 (Top-K succeeds)**.** *The accuracy of Top-K with a depth of h on $\mathcal{D}_{h,K}$ is at least $1 - \varepsilon$.*

Lemma 4.2 (Top-k fails)**.** *The accuracy of Top-k with a depth of h' on $\mathcal{D}_{h,K}$ is at most $(1/2 + \varepsilon)$ for any $h' < h + K - k$.*

Proofs of both these lemmas are deferred to Appendix B. Theorem 2 then follows directly from these two lemmas.

4.2 Proof of Theorem 3

In this section, we overview the proof Theorem 3. Some of the proofs are deferred to Appendix B.2.

Before proving Theorem 3, we formalize the concept of monotonicity. For simplicity, we assume the domain is the Boolean cube, $\{0,1\}^d$, and use the partial ordering $x \preceq x'$ iff $x_i \leq x'_i$ for each $i \in [d]$; however, the below definition easily extends to the domain being any partially ordered set.

Definition 4 (Monotone)**.** *A stochastic function, $\boldsymbol{f} : \{0,1\}^d \to \{0,1\}$, is* monotone *if, for any $x, x' \in \{0,1\}^d$ where $x \preceq x'$, $\mathbb{E}[\boldsymbol{f}(x)] \leq \mathbb{E}[\boldsymbol{f}(x')]$. A data distribution, $\mathcal{D}$ over $\{0,1\}^d \times \{0,1\}$ is said to be monotone if the corresponding stochastic function, $\boldsymbol{f}(x)$ returning $(\boldsymbol{y} \mid \boldsymbol{x} = x)$ where $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$, is monotone.*

To construct the data distribution of Theorem 3, we will combine monotone functions, Majority and Tribes, commonly used in the analysis of Boolean functions due to their extremal properties. See Appendix B.2 for their definitions and useful properties. Let $d = h + K - 1$, and the distribution over the domain be uniform over $\{0,1\}^d$. Given some $x \in \{0,1\}^d$, we use $x^{(1)}$ to refer to the first h coordinates of x and $x^{(2)}$ the other $K-1$ coordinates. This data distribution is labeled by the stochastic function $\boldsymbol{f}$ given below.

$$\boldsymbol{f}(x) \coloneqq \begin{cases} \mathrm{Tribes}_h(x^{(1)}) & \text{with probability } 1-\varepsilon \\ \mathrm{Maj}_{K-1}(x^{(2)}) & \text{with probability } \varepsilon. \end{cases}$$

Clearly $\boldsymbol{f}$ is monotone as it is the mixture of two monotone functions. Throughout this subsection, we'll use $\mathcal{D}_{h,K}$ to refer to the data distribution over $\{0,1\}^d \times \{0,1\}$ where to sample $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$, we first draw $\boldsymbol{x} \sim \{0,1\}^d$ uniformly and then $\boldsymbol{y}$ from $\boldsymbol{f}(\boldsymbol{x})$. The proof of Theorem 3 is a direct consequence of the following two Lemmas, both of which we prove in Appendix B.2.

Lemma 4.3 (Top-K succeeds)**.** *On the data distribution $\mathcal{D}_{h,K}$, Top-K with a depth budget of h achieves at least $1 - \varepsilon$ accuracy.*

Lemma 4.4 (Top-k fails)**.** *On the data distribution $\mathcal{D}_{h,K}$, Top-k with a depth budget of h achieves at most $\frac{1}{2} + \varepsilon$ accuracy.*

5 Experiments

Setup for experiments. At all places, the Top-1 tree that we compare to is that given by `scikit-learn` [PVG+11], which according to their documentation[2], is an optimized version of CART. We run experiments on a variety of datasets from the UCI Machine Learning Repository [DG17] (numerical as well as categorical features) having a size in the thousands and having $\approx 50-300$ features after binarization. There were ≈ 100 datasets meeting these criteria, and we took a random subset of 20 such datasets. We binarize all the datasets – for categorical datasets, we convert every categorical feature that can take on (say) ℓ values into ℓ binary features. For numerical datasets, we sort and compute thresholds for each numerical attribute, so that the total number of binary features is ≈ 100. A detailed description of the datasets is given in Appendix C.

We build decision trees corresponding to binary entropy as the impurity measure $\mathcal{H}$. In order to leverage existing engineering optimizations from state-of-the-art optimal decision tree implementations, we implement the Top-k algorithm given in Figure 1 via simple modifications to the PyDL8.5 [ANS20, ANS21] codebase[3]. Details about this are provided in Appendix D. Our implementation of the Top-k algorithm and other technical details for the experiments are available at `https://github.com/SullivanC19/pydl8.5-topk`.

5.1 Key experimental findings

Small increments of k yield significant accuracy gains. Since the search space of Top-k is a superset of that of Top-1 for any $k > 1$, the training accuracy of Top-k is guaranteed to be larger. The primary objective in this experiment is to show that Top-k can outperform Top-1 in terms of test accuracy as well. Figure 2 shows the results for Top-1 versus Top-k for $k = 2, 3, 4, 8, 12, 16, d$. Each

[2] https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart

[3] https://github.com/aia-uclouvain/pydl8.5

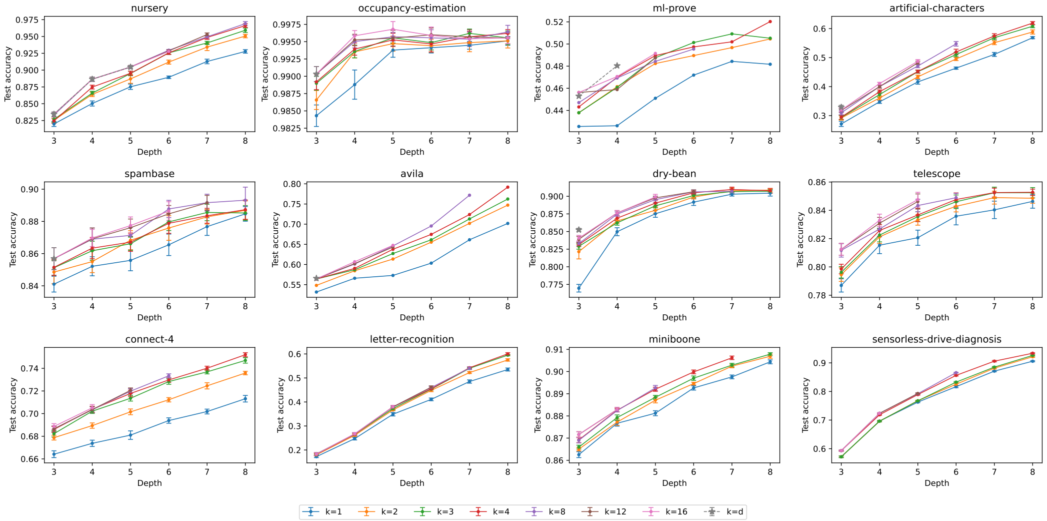


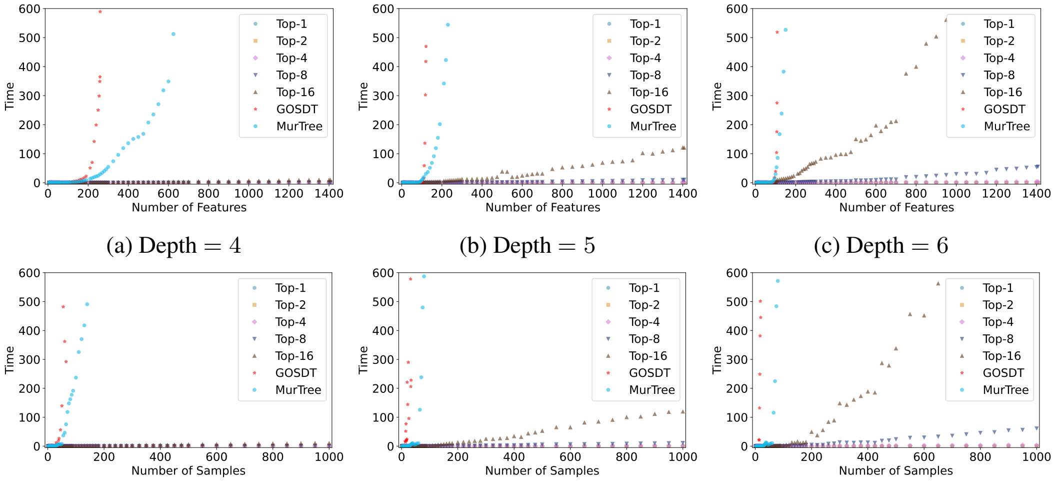
Figure 2: Test accuracy comparison between Top-k for various values of k. We can see that Top-$(k+1)$ generally obtains higher accuracy than Top-k, and in some cases (e.g., nursery), Top-$8/16$'s accuracy is even comparable to the optimal tree (Top-d). Missing points in the plots correspond to settings that did not terminate within a sufficiently large time limit. All plots are averaged over 10 random train-test splits (except avila and ml-prove that have pre-specified splits) with confidence intervals plotted for 2 standard deviations.

plot is a different dataset, where on the x-axis, we plot the depth of the learned decision tree, and on the y-axis, we plot the test accuracy. Note that $k = d$ corresponds to the DL8.5 optimal decision tree. We can clearly observe that the test accuracy increases as k increases—in some cases, the gain is $> 5\%$ (absolute). Furthermore, for (smaller) datasets like nursery, for which we were able to run $k = d$, the accuracy of Top-$8/16$ is already very close to that of the optimal tree.

Lastly, since Top-k invests more computation towards fitting a better tree on the training set, its training time is naturally longer than Top-1. However, Figure 6 in Appendix E, which plots the training time, shows that the slowdown is mild.

Top-k scales much better than optimal decision tree algorithms. Optimal decision tree algorithms suffer from poor runtime scaling. We empirically demonstrate that, in comparison, Top-k has a significantly better scaling in training time. Our experiments are identical to those in Figures 14 and 15 in the GOSDT paper [LZH+20], where two notions of scalability are considered. In the first experiment, we fix the number of samples and gradually increase the number of features to train the decision tree. In the second experiment, we include all the features, but gradually increase the number of training samples. The dataset we use is the FICO [FGI+18] dataset, which has a total of 1000 samples with 1407 binary features. We plot the training time (in seconds) versus number of features/samples for optimal decision tree algorithms (MurTree, GOSDT) and Top-k in Figure 3. We do this for depth $= 4, 5, 6$ (for GOSDT, the regularization coefficient λ is set to $2^{-\text{depth}}$). We observe that the training time for both MurTree and GOSDT increases dramatically compared to Top-k, in both experiments. In particular, for depth $= 5$, both MurTree and GOSDT were unable to build a tree on 300 features within the time limit of 10 minutes, while Top-16 completed execution even with all 1407 features. Similarly, in the latter experiment, GOSDT/MurTree were unable to build a depth-5 tree on 150 samples within the time limit, while Top-16 comfortably finished execution even on 1000 samples. These experiments demonstrates the scalability issues with optimal tree algorithms. Coupled with the accuracy gains seen in the previous experiment, Top-k can thus be seen as achieving a more favorable tradeoff between training time and accuracy.

We note, however, that various optimization have been proposed to allow optimal decision tree algorithms to scale to larger datasets. For example, a more recent version of GOSDT has integrated a guessing strategy using reference ensembles which guides the binning of continuous features, tree



(a) Depth $= 4$ (b) Depth $= 5$ (c) Depth $= 6$

(d) Depth $= 4$ (e) Depth $= 5$ (f) Depth $= 6$

Figure 3: Training time comparison between Top-k and optimal tree algorithms. As the number of features/samples increases, both GOSDT and MurTree scale poorly compared to Top-k, and beyond a threshold, do not complete execution within the time limit.

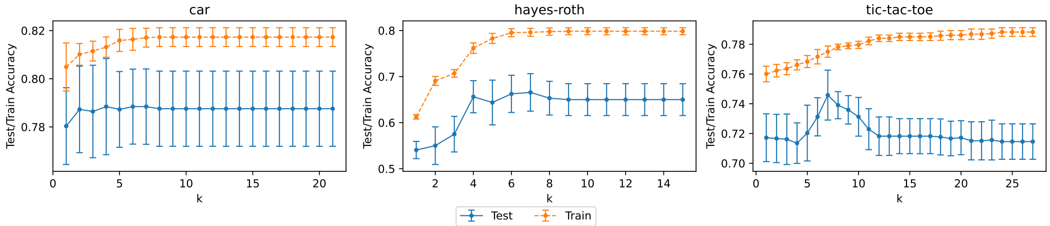


Figure 4: Test accuracy plateaus for large k. All runs averaged over 10 random train-test splits with maximum depth fixed to 3.

size, and search [MZA$^+$]. Many of these optimizations are generally applicable across optimal tree algorithms and could be combined with Top-k for further improvement in performance.

Increasing k beyond a point does not improve test accuracy. In our experiments above, we ran Top-k only till $k = 16$: in Figure 4, we show that increasing k to very large values, which increases runtime, often does not improve test accuracy, and in some cases, may even *hurt* due to overfitting. For 3 datasets – car, hayes-roth and tic-tac-toe – we plot train and test error as a function of k. Naturally, the train accuracy monotonically increases with k in each plot. However, for both car and hayes-roth, we can observe that the test accuracy first increases and then plateaus. Interestingly, for tic-tac-toe, the test accuracy first increases and then *decreases* as we increase k. These experiments demonstrate that selecting too large of a k, as optimal decision tree algorithms do, is a waste of computational resources and can even hurt test accuracy via overfitting.

6 Conclusion

We have shown how popular and empirically successful greedy decision tree learning algorithms can be improved with *the power of choices*: our generalization, Top-k, considers the k best features as candidate splits instead of just the single best one. As our theoretical and empirical results demonstrate, this simple generalization is powerful and enables significant accuracy gains while preserving the efficiency and scalability of standard greedy algorithms. Indeed, we find it surprising that such a simple generalization has not been considered before.

There is much more to be explored and understood, both theoretically and empirically; we list here a few concrete directions that we find particularly exciting and promising. First, we suspect that power of choices affords more advantages over greedy algorithms than just accuracy gains. For example, an avenue for future work is to show that the trees grown by Top-k are more *robust to noise*. Second, are there principled approaches to the automatic selection of the greediness parameter k? Can the optimal choice be inferred from a few examples or learned over time? This opens up the possibility of new connections to machine-learned advice and algorithms with predictions [MV20], an area that has seen a surge of interest in recent years. Finally, as mentioned in the introduction, standard greedy decision tree algorithms are at the very heart of modern tree-based ensemble methods such as XGBoost and random forests. A natural next step is to combine these algorithms with Top-k and further extend the power of choices to these settings.

Acknowledgements

We thank the NeurIPS reviewers and AC for their detailed and helpful feedback.

Guy and Li-Yang are supported by NSF awards 1942123, 2211237, 2224246 and a Google Research Scholar award. Jane is supported by NSF Graduate Research Fellowship under Grant No. 2141064, NSF Awards CCF-2006664, DMS-2022448, and Microsoft. Mo is supported by a Stanford Interdisciplinary Graduate Fellowship and a Stanford Data Science Scholarship. Chirag is supported by Moses Charikar and Greg Valiant's Simons Investigator Awards.

References

[AAV19] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1418–1426, 2019.

[ABF$^+$09] Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer & System Sciences*, 74(1):16–34, 2009.

[AH08] Micah Adler and Brent Heeringa. Approximating optimal binary decision trees. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 1–9. Springer, 2008.

[ANS20] Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3146–3153, 2020.

[ANS21] Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Pydl8. 5: a library for learning optimal decision trees. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 5222–5224, 2021.

[Ave20] Florent Avellaneda. Efficient inference of optimal decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3195–3202, 2020.

[BD17] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[BDM19] Alon Brutzkus, Amit Daniely, and Eran Malach. On the Optimality of Trees Generated by ID3. *ArXiv*, abs/1907.05444, 2019.

[BDM20] Alon Brutzkus, Amit Daniely, and Eran Malach. ID3 learns juntas for smoothed product distributions. In *Proceedings of the 33rd Annual Conference on Learning Theory (COLT)*, pages 902–915, 2020.

[BFSO84] Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and regression trees*. Wadsworth International Group, 1984.

[BLQT21a] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Decision tree heuristics can fail, even in the smoothed setting. In Mary Wootters and Laura Sanità, editors, *Proceedings of the 25th International Conference on Randomization and Computation (RANDOM)*, volume 207, pages 45:1–45:16, 2021.

[BLQT21b] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2021.

[BLT20a] Guy Blanc, Jane Lange, and Li-Yang Tan. Provable guarantees for decision tree induction: the agnostic setting. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[BLT20b] Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: rigorous guarantees and inherent limitations. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 1–44, 2020.

[Bre01a] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Bre01b] Leo Breiman. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231, 2001.

[CG16] 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)*, pages 785–794, 2016.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[DKM96] Tom Dietterich, Michael Kearns, and Yishay Mansour. Applying the weak learning framework to understand and improve C4.5. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, pages 96–104, 1996.

[DLH$^+$22] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research*, 23(26):1–47, 2022.

[FGI$^+$18] FICO, Google, Imperial College London, MIT, University of Oxford, UC Irvine, and UC Berkeley. Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge, 2018.

[HR76] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[HRS19] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems*, 32, 2019.

[IYA12] Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 1819–1822. IEEE, 2012.

[JM20] Mikoláš Janota and António Morgado. SAT-based encodings for optimal decision trees with explicit paths. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 501–518. Springer, 2020.

[Kea96] Michael Kearns. Boosting theory towards practice: recent developments in decision tree induction and the weak learning framework (invited talk). In *Proceedings of the 13th National Conference on Artificial intelligence (AAAI)*, pages 1337–1339, 1996.

[Kea98] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on boolean functions. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988.

[KM96] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the 28th Annual Symposium on the Theory of Computing (STOC)*, pages 459–468, 1996.

[KM99] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

[LZH$^+$20] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[MV20] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *arXiv preprint arXiv:2006.09123*, 2020.

[MZA$^+$] Hayden McTavish, Chudi Zhong, Reto Achermann, Ilias Karimalis, Jacques Chen, Cynthia Rudin, and Margo Seltzer. Fast sparse decision tree optimization via reference ensembles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9).

[NF07] Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 530–539, 2007.

[NF10] Siegfried Nijssen and Elisa Fromont. Optimal constraint-based decision tree induction from itemset lattices. *Data Mining and Knowledge Discovery*, 21(1):9–51, 2010.

[NIPMS18] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva. Learning optimal decision trees with sat. In *Ijcai*, pages 1362–1368, 2018.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[PVG$^+$11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Qui86] Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Qui93] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[RCC$^+$22] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

[Sie08] Detlef Sieling. Minimization of decision trees is hard to approximate. *Journal of Computer and System Sciences*, 74(3):394–403, 2008. Computational Complexity 2003.

[TAA$^+$19] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pages 6166–6175. PMLR, 2019.

[VNP$^+$20] Hélene Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming. *Constraints*, 25(3):226–250, 2020.

[VZ17] Sicco Verwer and Yingqian Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 94–103. Springer, 2017.

[VZ19] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1625–1632, 2019.

[ZMP$^+$20] Haoran Zhu, Pavankumar Murali, Dzung Phan, Lam Nguyen, and Jayant Kalagnanam. A scalable MIP-based method for learning optimal multivariate decision trees. *Advances in Neural Information Processing Systems*, 33:1771–1781, 2020.

A Proofs deferred from Section 3

Proof of Lemma 3.2. By induction: When $h = 0$, the only trees in the search space are the constant 0 and constant 1 functions. Top-k returns which of these two trees is the most accurate.

When $h \geq 1$, let T' be a tree with maximal accuracy within $\mathcal{T}_{k,h,S}$. As T' is in the search space, its root must be one of the k coordinates with maximal score which form the candidate set $\mathcal{I}$.

For each coordinate $i \in \mathcal{I}$, the candidate tree T_i satisfies

$$\Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[T_i(\boldsymbol{x}) \neq \boldsymbol{y}] = \Pr_{\boldsymbol{x}\sim S}[x_i = 0] \Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[T_{i0}(\boldsymbol{x}) \neq \boldsymbol{y}] + \Pr_{\boldsymbol{x}\sim S}[x_i = 1] \Pr_{\boldsymbol{x},\boldsymbol{y}\sim S}[T_{i1}(\boldsymbol{x}) \neq \boldsymbol{y}],$$

where T_{i0} and T_{i1} are the left and right subtrees of T_i respectively. Each of T_{i0} and T_{i1} is an output of Top-k with a depth budget of $h - 1$. We assume as the inductive hypothesis that each of these trees minimizes error among all trees in $\mathcal{T}_{k,h-1,S_{x_i=0}}$ and $\mathcal{T}_{k,h-1,S_{x_i=1}}$ respectively; therefore the candidate T_i minimizes error among all trees in $\mathcal{T}_{k,h,S}$ that have x_i at the root. Since Top-k chooses the most accurate of the T_i's, it follows that the chosen tree minimizes error among all trees in $\mathcal{T}_{k,h,S}$. □

B Proofs deferred from Section 4

Setup and notation: We use $\mathbb{1}[\cdot]$ for the indicator function, and $[d]$ to refer to the set $\{1, \dots, d\}$.

For brevity, we will make two simplifying assumptions about Top-k:

1. We will assume Top-k builds *non-redundant* trees, meaning on every root-to-leaf path, each coordinate is queried at most once. This is easy to enforce in the pseudocode: at each step, the algorithm can track a set Q of the coordinates already queried along this path, and pick the top-k coordinates according to the feature score function among $[d] \setminus Q$. For brevity, we do not include that modification to the pseudocode in Figure 1.

2. We assume that Top-k always build *complete* trees (i.e every root-to-leaf path has depth exactly h). This is without loss of generality, as whenever Top-k stops early, it does so because it has already achieved perfect accuracy on that path.

Furthermore, Top-k only uses the information in its sample in two ways: first, it uses the sample to compute the feature scoring function $\mathcal{H}(S, i)$. Second, when $h = 0$, it uses the sample to determine whether the constant 0 or constant 1 fits the sample better. Both of these are "statistical queries" [Kea98], meaning the interaction the algorithm receives from the sample is simply the expectations $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim S}[\phi_i(\boldsymbol{x}, \boldsymbol{y})]$ where $\phi_1, \dots, \phi_t : \{0,1\}^{d+1} \to [0,1]$ are a sequence of queries. For any $\varepsilon, \delta > 0$, by a standard concentration argument and union bound, for large enough sample size $n \geq n(\varepsilon, \delta)$,

$$\Pr_{\boldsymbol{S}\sim\mathcal{D}^n}\left[\max_{i\in[t]}\left|\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\boldsymbol{S}}[\phi_i(\boldsymbol{x},\boldsymbol{y})] - \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}[\phi_i(\boldsymbol{x},\boldsymbol{y})]\right| \geq \varepsilon\right] \leq \delta.$$

Therefore, for sufficiently large sample size, we are free to assume that when the algorithm computes $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\boldsymbol{S}}[\phi_i(\boldsymbol{x},\boldsymbol{y})]$, it receives $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}[\phi_i(\boldsymbol{x},\boldsymbol{y})]$ with high probability. This is a standard argument (c.f. [KM96]), and so we will work directly with expectations from $\mathcal{D}$ in our proof to ease notation.

Recall that Theorems 1 to 3 hold whenever the feature scoring function is an impurity-based heuristic.As our data distribution is uniform on the input, we are able to use the following fact and simultaneously prove results for all impurity-based heuristic:

Fact B.1 (Proposition 7.7 of [BLT20b]). *If the scoring function is* any *impurity-based heuristic, and the data distribution is uniform over inputs ($\boldsymbol{x}$ is uniform when $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$), then the score of a coordinate i is monotone increasing with its correlation with the label, $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}[\boldsymbol{x}_i\boldsymbol{y}]$.*

Intuitively, Fact B.1 means that, when analyzing Top-k on uniform data distributions, we are free to replace the "k coordinates with largest scores" with the "k coordinates with largest correlations."

B.1 Proofs deferred from Section 4.1

The stochastic function $\boldsymbol{f}_{h,K}$ used throughout Lemma 4.1 and Lemma 4.2 combines a function that outputs a random one of k features with the h-wise parity function.

Definition 5 (Parity)**.** *The* parity *function of ℓ variables, indicated by* $\mathrm{Par}_\ell : \{0,1\}^\ell \to \{0,1\}$, *returns*

$$\mathrm{Par}_\ell(x) \coloneqq \left(\sum_{i\in[\ell]} x_i\right) \mod 2.$$

Fact B.2 (Computing any function with a complete tree)**.** *Let $f : \{0,1\}^d \to \{0,1\}$ be any function that only depends on the first h variables, meaning there is some $g : \{0,1\}^h \to \{0,1\}$ such that:*

$$f(x) = g(x_{[1:h]})$$

for all $x \in \{0,1\}^d$. Let T be any non-redundant complete tree of depth-h in which every internal node is one of the first h coordinates. Then, there is a way to label the leaves of T such that T exactly computes f.

Proof. Since T is non-redundant, each coordinate is queried at most once on each root-to-leaf path. T is complete and depth-h, so each of the first h coordinates must be queried *exactly* once on each root-to-leaf path. Therefore, each leaf of T corresponds to exactly one way to set the first k coordinates of x. If the leaf is labeled by the output of g given those first k coordinates, T will exactly compute f. □

Proof of Lemma 4.1. The function $\mathrm{Par}_h(x^{(1)})$ is a $(1-\varepsilon)$-approximation to f, so it suffices to show that the depth-h tree for $\mathrm{Par}_h(x^{(1)})$ is within the search space of Top-K when run to a depth of h. Then we can apply Lemma 3.2 to reach the desired result.

There are only $K-1$ variables not in $x^{(1)}$, so each set of K candidate variables must contain some variable in $x^{(1)}$. Since Top-K is non-redundant, this must be a variable that has not yet been queried higher in the tree. Thus, at every step Top-K will always try a candidate variable that reduces the number of relevant $x^{(1)}$-variables by 1. It follows that the complete nonadaptive tree of depth h, containing all the variables of $x^{(1)}$, is within the search space, so by Fact B.2 there is a tree in the search space that computes $\mathrm{Par}_h(x^{(1)})$ exactly. Then the accuracy of the output must be at least the total accuracy of this tree, which is $(1-\varepsilon)$. □

Proof of Lemma 4.2. Conditioned on any setting of $< k$ variables, for any variable x_i in $x^{(2)}$, $\mathbb{E}[f(x)x_i] \geq 1/k$. Similarly, for any variable x_j in $x^{(1)}$, $\mathbb{E}[f(x)x_j] = 0$. By Fact B.1, at every node the variables of $x^{(2)}$ that have not yet been queried all rank ahead of the variables of $x^{(1)}$.Thus, if at most $K-k$ variables have already been queried, the remaining k most-correlated candidates will all be from $x^{(2)}$, so no variable in $x^{(1)}$ will be considered. Thus, at least $K-k$ variables from $x^{(2)}$ will be placed in every path.

Since the depth budget h' is smaller than $h+K-k$ and at least $K-k$ variables from $x^{(2)}$ are placed in every path, no path can contain all of the h variables of $x^{(1)}$. The value of $\mathrm{Par}_h(x^{(1)})$ is 0 with probability 1/2 and 1 with probability 1/2 conditioned on the values of any set of variables smaller than h. Therefore, the tree built by Top-k cannot achieve accuracy better than 1/2 on the parity portion of the function (and thus have accuracy better than $(1/2+\varepsilon)$ overall).

□

B.2 Proofs deferred from Section 4.2

The data distribution showing the accuracy separation between Top-K and Top-k is formed by combining the Majority and Tribes functions.

Definition 6 (Majority)**.** *The* majority *function of ℓ variables, indicated by* $\mathrm{Maj}_\ell : \{0,1\}^\ell \to \{0,1\}$, *returns*

$$\mathrm{Maj}_\ell(x) \coloneqq \mathbb{1}[\textit{at least half of } x\textit{'s coordinates are } 1].$$

Definition 7 (Tribes)**.** *For any input length ℓ, let w be the largest integer such that $(1-2^{-w})^{\ell/w} \leq 1/2$. For $x \in \{0,1\}^\ell$, let $x^{(1)}$ be the first w coordinates, $x^{(2)}$, the second w, and so on.* Tribes$_\ell$ *is defined as*

$$\text{Tribes}_\ell(x) \coloneqq (x_1^{(1)} \wedge \cdots \wedge x_w^{(1)}) \vee \cdots \vee (x_1^{(t)} \wedge \cdots \wedge x_w^{(t)}) \qquad \textit{where } t \coloneqq \left\lfloor \frac{\ell}{w} \right\rfloor.$$

For our purposes, it is sufficient to know a few simple properties about Tribes. These are all proven in [O'D14, §4.2].

Fact B.3 (Properties of Tribes)**.**

1. Tribes$_\ell$ *is monotone.*
2. Tribes$_\ell$ *is nearly balanced:*
$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{0,1\}^\ell}[\text{Tribes}_\ell(\boldsymbol{x})] = \frac{1}{2} \pm o(1)$$
where the $o(1)$ term goes to 0 as ℓ goes to ∞.
3. *All variables in* Tribes$_\ell$ *have small correlation: For each $i \in [\ell]$,*
$$\text{Cov}_{\boldsymbol{x}\sim\{0,1\}^\ell}[\boldsymbol{x}_i, \text{Tribes}_\ell(\boldsymbol{x})] = O\left(\frac{\log \ell}{\ell}\right).$$

Indeed, the famous KKL inequality implies that any function with the first and second property has a variable with correlation at least $\Omega(\log \ell/\ell)$ [KKL88]. Our construction uses Tribes exactly because it has the minimum correlations among functions with the above properties (up to constants). In contrast, we use Majority because its correlations are as *large* as possible, which will "trick" Top-k into building a bad tree.

With the above definitions in-hand, we are able to provide proofs of the following two lemmas:

Proof of Lemma 4.3. This proof is very similar to that of Lemma 4.1: Once again, we observe the tree computing $(x \mapsto \text{Tribes}_h(x^{(1)}))$ has at least $1-\varepsilon$ accuracy with respect to $\mathcal{D}_{h,K}$. By Lemma 3.2, it is sufficient to prove such a tree is in the search space.

By Fact B.2, any non-redundant complete tree of depth h that only queries the first h coordinates of its input will compute the function $(x \mapsto \text{Tribes}_h(x^{(1)}))$ whenever the leaves are appropriately labeled. Therefore, we only need to prove such a tree is in the search space $\mathcal{T}_{K,h,\mathcal{D}}$. There are only $K-1$ coordinates that are *not* one of the first h corresponding to $x^{(1)}$. Therefore, within any non-redundant set of K coordinates, at least one must be a non-redundant coordinate from the first h. This implies one of the desired trees is in the search space. □

Proof of Lemma 4.4. Let T be the tree returned by Top-k. Consider any root-to-leaf path of T that does *not* query any of the first h coordinates (those within $x^{(1)}$). Recall that, with probability $(1-\varepsilon)$, the label is given by Tribes$_h(x^{(1)})$. On this path, the label of T does not depend on any of the coordinates within $x^{(1)}$. Therefore,

$$\begin{aligned}
&\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{h,K}}[T(\boldsymbol{x}) = \boldsymbol{y} \mid \boldsymbol{x} \text{ follows this path}] \\
&\qquad = (1-\varepsilon)\cdot \Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{h,K}}[T(\boldsymbol{x}) = \text{Tribes}_h(\boldsymbol{x}^{(1)}) \mid \boldsymbol{x} \text{ follows this path}] \\
&\qquad\quad + \varepsilon \cdot \Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{h,K}}[T(\boldsymbol{x}) = \text{Maj}_K(\boldsymbol{x}^{(2)}) \mid \boldsymbol{x} \text{ follows this path}] \\
&\qquad \leq (1-\varepsilon)\cdot\left(\frac{1}{2} + o(1)\right) + \varepsilon\cdot 1 \leq \frac{1+\varepsilon}{2} + o(1)
\end{aligned}$$

where the last line follows because Tribes$_h$ is nearly balanced (Fact B.3). As the distribution over $\boldsymbol{x}$ is uniform, each leaf is equally likely. Therefore, if only p-fraction of root-to-leaf paths of T query at least one of the first h coordinates, then,

$$\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{h,K}}[T(\boldsymbol{x}) = \boldsymbol{y}] \leq (1-p)\cdot\left(\frac{1+\varepsilon}{2} + o(1)\right) + p\cdot 1 \leq \frac{1}{2} + \frac{p}{2} + \frac{\varepsilon}{2} + o(1)$$

Our goal is to prove the tree returned by Top-k achieves at most $\frac{1}{2} + \varepsilon$ accuracy. Therefore, it is enough to prove that $p = o(1)$. Indeed, we will prove that $p \leq O(K^{-2})$.

Here, we apply [BLT20b, Lemma 7.4], which was used to show that Top-1 fails to build a high accuracy tree. They used a different data distribution, but that particular Lemma still applies to our setting. They prove that a random root-to-leaf path of T satisfies the following with probability at least $1 - O(K^{-2})$: If the length of this path is less than $O(K/\log K)$, at any point along that path, all coordinates within $x^{(2)}$ that have not already been queried have correlation at least $\frac{1}{100\sqrt{k}}$.

That Lemma will be useful for proving Top-k fails with the following parameter choices.

1. By setting $K \geq \Omega(h \log h)$, we can ensure all root-to-leaf paths in T have length at most $O(K/\log K)$, so [BLT20b, Lemma 7.4] applies.

2. By setting $K \leq O(h^2/(\log h)^2)$, we can ensure that all the coordinates within $x^{(1)}$ have correlation less than $\frac{1}{100\sqrt{k}}$ (Fact B.3). This means that all non-redundant coordinates within $x^{(2)}$ have more correlation than those within $x^{(1)}$.

3. By setting $k \leq K - h$, we ensure at all nodes along every path, there are at least k coordinates within the last $K - 1$ coordinates (those corresponding to $x^{(2)}$), that have not already been queried. With probability at least $1 - O(K^{-2})$ over a random path, those all have more correlation than all coordinates within $x^{(1)}$, so Top-k won't query any of the h coordinates within $x^{(1)}$.

We conclude that, with probability at least $1 - O(K^{-2})$ over a random path in T, that path does not query any of the first h variables. As a result, the accuracy of T is at most $\frac{1+\varepsilon}{2} + o(1) \leq \frac{1}{2} + \varepsilon$. □

C Details about datasets used in Section 5

Name	Type	Size (#train/#test)	#feats	#binary feats	#classes
connect-4	C	67557 (54045/13512)	42	126	3
nursery	C	12960 (10368/2592)	8	27	5
letter-recognition	C	19999 (15999/4000)	16	256	26
car	C	1728 (1382/346)	6	21	4
kr-vs-kp	C	3196 (2556/640)	36	73	2
hiv-1-protease	C	6590 (5272/1318)	8	160	2
molecular-biology-splice	C	3190 (2552/638)	60	287	3
monks-1	C	556 (444/112)	6	17	2
hayes-roth	C	160 (128/32)	4	15	3
tic-tac-toe	C	958 (766/192)	9	27	2
artificial-characters	N	10218 (8174/2044)	7	91	10
telescope	N	19020 (15216/3804)	10	100	2
spambase	N	4601 (3680/921)	57	57	2
dry-bean	N	13611 (10888/2723)	16	96	7
occupancy-estimation	N	10129 (8103/2026)	16	86	4
miniboone	N	130064 (104051/26013)	50	100	2
sensorless-drive-diagnosis	N	58509 (46807/11702)	48	96	11
ml-prove	N	6118 (4588/1530)	51	51	6
avila	N	20867 (10430/10437)	10	100	12
taiwanese-bankruptcy	N	6819 (5455/1364)	95	95	2
credit-card	N	30000 (24000/6000)	23	88	2
electrical-grid-stability	N	10000 (8000/2000)	13	91	2
FICO	N	1000 (900/100)	23	1407	2

Table 1: Dataset characteristics. In the Type column, C stands for Categorial and N stands for Numerical.

Table 1 provides complete details regarding all the datasets we used in our experiments. For datasets that do not provide an explicit train/test split, we randomly compute ten 80:20 splits, and average our results over these splits. The column #feats has the number of raw attributes in each dataset, while the column #binary feats has the number of features we obtain after converting these raw attributes to binary-valued attributes. For categorical datasets, we encode a categorical attribute taking on l distinct values to l binary attributes. For numerical datasets, we sort and compute thresholds for each numerical attribute. The number of thresholds is so selected that the total number of binary attributes does not exceed 100.

D Implementation details for the Top-k algorithm

Our implementation of Top-k makes use of the DL8.5 algorithm implementation from [ANS21]. DL8.5 is an optimal classification tree search algorithm which utilizes caching and branch-and-bound optimization to avoid repeated computation and prune large sections of the search space that would yield suboptimal trees [ANS20], similar to MurTree [DLH+22]. To get our optimized Top-k algorithm, we modify DL8.5 to only consider the first k feature splits of each recursive state in descending order of information gain and with ties broken by feature index.

There were two other optimizations made by the DL8.5 algorithm implementation that would have led to different results. These optimizations are (1) fast computation of depth-two optimal trees and (2) similarity-based lower bounding. These optimizations were disabled.

E Training time comparison

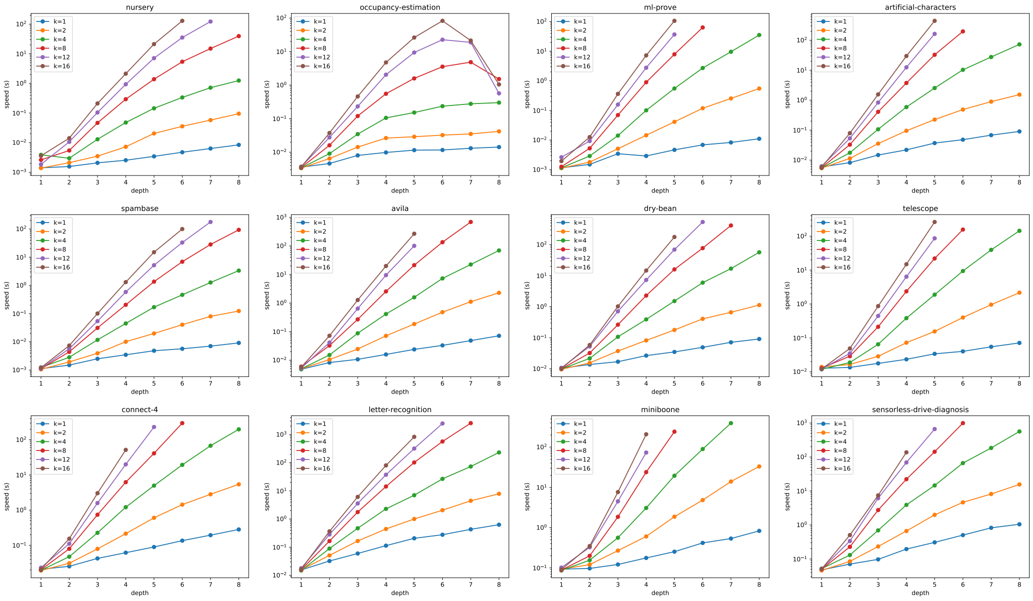


Figure 6: Training time comparison between Top-1 and Top-k. We can see that the blowup in training time when compared to Top-1 is relatively mild. In particular, for $k = 2$, we are able to go all the way up until depth-8 trees within 1 second in almost all cases. Even $k = 4, 8$ finishes execution for depth-5 trees within ≈ 20 seconds for majority of the datasets. Interestingly, in the case of occupancy-estimation, we can see that the training times get *faster* at the larger depths. This is an artefact of the optimized branch-and-bound implementation of DL8.5, which stops branching once it discovers a subtree with no errors. We expect these perfect subtrees to become more prevalent when considering higher depth trees and when there are fewer points to be classified.

```
Opt-Top-$k(\mathcal{H}, S, h, ub)$:
  Given: A feature scoring function $\mathcal{H}$, a labeled sample set $S$ over $d$ dimensions, depth
      budget $h$, and upper bound on misclassification error $ub$.
  Output: Decision tree of depth $h$ that approximately fits $S$.
    1. If $h = 0$, or if every point in $S$ has the same label, return the constant function with
       the best accuracy w.r.t. $S$.
    2. If $(S, d)$ is in the cache:
       (a) Let $T_c$ and $ub_c$ be the cached tree and upper bound.
       (b) If $T_c \neq$ NO-TREE then return $T_c$.
       (c) If $T_c =$ NO-TREE and $ub \leq ub_c$ then return NO-TREE.
    3. Let $T^*$ be NO-TREE.
    4. Let $b^*$ be $ub + 1$.
    5. Let $\mathcal{I} \subseteq [d]$ be the set of $k$ coordinates maximizing $\mathcal{H}(S, i)$.
    6. For each $i \in \mathcal{I}$:
       (a) Let $T_i$ be the tree with
                         Root $= x_i$
                Left subtree $=$ Opt-Top-$k(\mathcal{H}, S_{x_i=0}, h-1, b^*-1)$
       (b) If the left subtree is NO-TREE then continue.
       (c) Let $b_L$ be the misclassification error of the left subtree w.r.t. $S_{x_i=0}$.
       (d) If $b_L \leq b^*$ we define the right subtree of $T_i$
                Right subtree $=$ Opt-Top-$k(\mathcal{H}, S_{x_i=1}, h-1, b^*-1-b_L)$
       (e) If the right subtree is NO-TREE then continue.
       (f) Let $b_R$ be the misclassification error of the right subtree w.r.t. $S_{x_i=1}$.
       (g) If $b_L + b_R < b^*$:
             i. Let $T^* = T_i$.
            ii. Let $b^* = b_L + b_R$.
       (h) If $b_L + b_R = 0$ then break.
    7. Add $(S, d)$ to the cache with value $(T^*, ub)$.
    8. Return $T^*$.
```

Figure 5: The optimized Top-k algorithm is equivalent to the Top-k algorithm described in Figure 1 but with caching and pruning optimizations that make it significantly faster in practice. These changes are bolded and highlighted in blue.

F Accuracy comparison with Top-1 – further plots

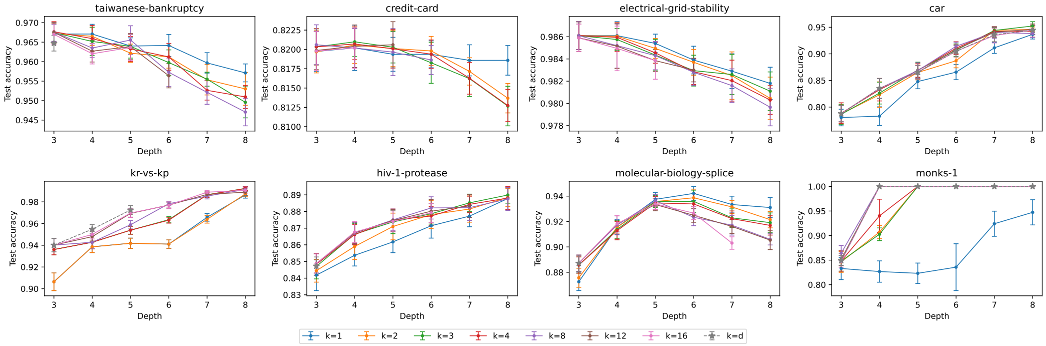


Figure 7: Test accuracy comparison between Top-1 and Top-k.

We provide plots from our experiments on a further few datasets comparing the test accuracy of Top-k and Top-1 in Figure 7. In the case of taiwanese-bankruptcy, credit-card and electrical-grid-stability, we can observe that Top-1 is outperforming Top-k. However, we believe that this is because the learning problem in this regime is extremely susceptible to overfitting. In particular, we can see that Top-1 is itself not consistently improving with increasing depth. Concretely, increasing depth beyond 3 is already causing Top-1 to overfit, and hence we would expect Top-k to suffer from overfitting even more. In the case of the remaining datasets (which all happen to be categorical), while the numbers might not be monotonically getting better with increasing k, we can still observe that there is always some value of $k > 1$ which is outperforming $k = 1$ (except for molecular-biology-splice, for which this is still the case till depth 6). This lends further support to our proposition of incorporating k as an additional hyperparameter to tune while training decision trees greedily.