OPTIMAL ALL-SUBSETS SPLITS FOR CATEGORICAL FEATURES IN CLASSIFICATION TREE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Handling categorical features with many categories adequately in classification trees remains an unsolved problem. Most training algorithms approach a split over a categorical feature via either a one-hot encoding, a label embedding or a multiway split, all of which have significant disadvantages on the resulting tree. The ideal split would choose optimally among all possible category subsets, but this is exponentially costly on the number of categories. We provide a solution to this long-standing problem by 1) using as tree learning not greedy recursive partitioning (as in CART) but tree alternating optimization, and 2) showing that in the latter case the update of a categorical split can be done exactly in time linear on the number of categories. This results in trees that are far smaller (thus more interpretable) and with higher classification accuracy, as demonstrated in our experiments.

1 INTRODUCTION

Categorical features (CFs) are ubiquitous and highly informative in tabular datasets, as well as important for model interpretability and sometimes subject to legal regulations (e.g. protected characteristics such as sex, ethnicity or religion). A CF can be binary or have a possibly large number of values (categories), e.g. "US state", "industry sector" or "job category". We focus here on their usage with classification trees, widely considered as among the most interpretable models in machine learning (ML). In axis-aligned or univariate trees, a decision node splits on a single input feature and each leaf outputs a class label; they handle the multiclass case naturally.

With a CF, the ideal type of split is an all-subsets split, where any partition of the possible categories is allowed. This produces a flexible partition, does not need any encoding, is highly interpretable and can be very informative as to what categories are similar to each other. However, a long-standing problem with trees learned with the traditional, more widely used algorithms (greedy recursive partitioning such as CART) is that, except for some special cases (see section 3), optimizing an all-subsets split at a node is exponential on the CF cardinality. This means that trees often use a weaker type of split such as equality-test (derived from one-hot encoding) or label embedding.

We show that all-subsets split nodes of any cardinality can be optimized exactly in time linear on the sample size if we use a different tree learning framework, namely tree alternating optimization (TAO). Since this fact was not known, TAO algorithms have so far relied on one-hot encoding of CFs, which is problematic for several reasons, particularly if they have many categories. TAO has the added advantage over CART-style algorithms that it optimizes a global objective function over all the tree parameters, which results in smaller and more accurate trees, while being scalable to large datasets. We now extend TAO's strengths to handling CFs effectively.

In the rest of the paper, we describe the different types of splits, such as equality-test or label embedding, as well as their pros and cons (section 2) and review related work (section 3); we describe how split optimization works within different tree-learning frameworks (section 4) and give our exact, efficient solution for all-subsets splits (section 5); and we confirm its effectiveness in a range of experiments (section 6).

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

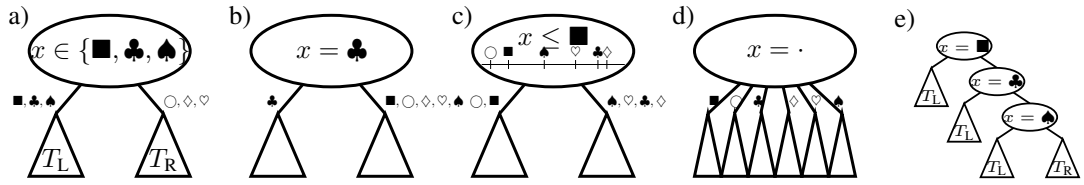

Figure 1: Types of splits of a decision node using a categorical unordered feature $x \in \{\blacksquare, \bigcirc, \clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$: a) all-subsets split; b) equality-test split; c) label embedding split (the embedding along the real line is shown); d) multiway split; e) an equivalent tree to that of a) but using equality-test splits.

2 TYPES OF DECISION TREE SPLITS WITH CATEGORICAL FEATURES

Consider a categorical feature (CF) x taking values (categories) in a set $\mathcal{C}$ with cardinality $C = |\mathcal{C}|$ and a decision tree (assumed binary unless otherwise stated). If a decision node in the tree uses feature x, we have the following types of splits for it (see fig. 1), all of which have been used by different tree learning algorithms and implementations:

All-subsets (fig. 1a) This partitions the categories among the children; with a binary tree, it sends some categories to the left child and the rest to the right child. This is the ideal, most powerful type of split, because its search space includes all 2^C possible partitions among the children, and the one with pursue in this paper (by giving an exact, linear-time optimization within the TAO framework).

Equality test (fig. 1b) This sends exactly one category to one child and all other categories to the other child. It happens if using one-hot encoding for the CF, i.e., we replace it with C dummy binary variables in $\{0, 1\}$ of which exactly one equals 1, each representing one category. A binary variable has only two values, so a binary split on it requires no optimization: it simply sends value 0 to the left child and value 1 to the right one (or vice versa, which reverses the children but does not change the predictive function of the tree). A split on a dummy binary variable is an equality test for the category it represents. The search space contains only C partitions, corresponding to testing on each of the C categories.

Label embedding (forced ordering) (fig. 1c) This is a form of feature engineering that replaces a CF with an ordered feature, which can then be split in between any consecutive values (like a real-valued feature). A trivial embedding is to place the values in a random order; a better way is to use the supervised label information, for which several approaches exist. The result in general is that the C unordered values are mapped to values along the real line, and the line can be split at any midpoint between consecutive values, which each child getting one side. Note only the order of the values matters, not the distances, because axis-aligned decision trees are invariant to monotonic transformations of a numeric feature. The search space contains only C partitions (including the partition that sends all categories to the same child).

Multiway node (fig. 1d) This is different in that it leads to non-binary trees: it assumes the node has one child per category (so no optimization is required to determine the partition).

One-hot encoding (Hastie et al., 2009; Kuhn & Johnson, 2013; 2019) is widely used in many ML models because of its simplicity, since it handles any kind of CF and the downstream classifier only receives binary features, which can be handled as if they were real-valued. However, with decision trees an equality-test split is much weaker than an all-subsets split, particularly with high-cardinality CFs. As noted in fig. 1e, we can represent an all-subsets split equivalently with an if-then-elseif lopsided tree using equality-test splits, but at the cost of several nodes (3 rather than 1 in the figure) and correspondingly repeated subtrees. In the worst case, when the all-subsets split has a balanced partition, the equality-test tree would require $\lfloor C/2 \rfloor$ nodes. It is also easier to interpret an all-subsets split, by simply looking at which categories go which way, than a series of equality-test splits.

Label-embedding splits are likewise weak, searching over few partitions. We can also emulate an all-subsets split using only label-embedding splits, but this leads to an even bigger tree in general

than with equality-test splits. Label embedding can work well if the order selected aligns well with the class label, but the latter generally depends on multiple features, not just on one CF.

Multiway splits (Hastie et al., 2009; Kuhn & Johnson, 2013) can work well in some cases, but they suffer from a well-known problem within CART: since they partition the data into more groups, downstream nodes (in particular, the leaves) receive few instances and have a harder time learning a decision or prediction that generalizes well. Thus (Hastie et al., 2009, p. 311) do not recommend their use in general, instead suggesting to use a series of binary splits. It is possible to use a multiway split with fewer children than categories, but this requires a procedure to merge categories into groups and it is difficult to determine the number of children in the first place. Quinlan (1993, ch. 7) applies a computationally costly heuristic to group categories into a few groups, effectively coarsening the original CF, prior to applying the multiway split. Also, multiway splits are less convenient with learning algorithms that operate on a fixed tree structure (such as TAO): one would need to decide ahead of time which nodes are multiway. That said, our optimal all-subsets split works also with multiway splits. With greedy recursive partitioning algorithms this is less of a problem because the number of children is decided once and for all when splitting a node.

Another well-known problem when using CFs with greedy recursive partitioning is that the purity optimization tends to select CFs having many categories much more often (Kuhn & Johnson, 2013, p. 379): a split on a high-cardinality CF is likely to have a larger purity increase than a continuous feature or a low-cardinality CF. Effectively, we are fitting a model with more parameters, which can lead to severe overfitting. Some algorithms try to reduce this selection bias using statistical tests and other techniques (Loh & Shih, 1997; Loh, 2002; Kim & Loh, 2001). It is even suggested to avoid CFs with many categories (Hastie et al., 2009, p. 310). With all-subsets splits optimized globally (rather than with a greedy partition) this should not be a problem, and indeed we find no such bias in our experiments.

3 Related work

Classification trees define a nonconvex, nondifferentiable problem which is NP-hard even in the simplest cases (Hyafil & Rivest, 1976). The traditional algorithms are based on greedy recursive partitioning, such as CART and C5.0 (Breiman et al., 1984; Quinlan, 1993; Loh, 2014; Rokach & Maimon, 2015) (see brief description in section 4.1). In order to fix the split at a decision node, they optimize a purity criterion. This is efficient for ordered (e.g. continuous) features, but for CFs it is well known to have exponential complexity over the number of categories (Hastie et al., 2009, p. 310). There are some special cases for which an efficient (polynomial-time upper bound) algorithm is known: one is with binary classification (two classes) and the Gini of cross-entropy purity function (Breiman et al., 1984; Ripley, 1996) and the other with regression and the squared error purity function (Fisher, 1958). In both cases, the idea relies on ordering the categories by class or by increasing mean of the outcome, respectively (essentially, a label embedding). As noted by Hastie et al. (2009, p. 310), the proofs of correctness of those algorithms are not trivial, and for multiclass classification no such simplifications are possible. An inexact approximation for CF all-subsets splits exists (Loh & Vanichsetakul, 1988).

Recent approaches seek to optimize a global objective function over trees. We focus on tree alternating optimization (TAO) (Carreira-Perpiñán & Tavallali, 2018), described in section 4.2. TAO has shown marked improvement in tree quality and size compared to the traditional approaches and is scalable to large datasets (Zharmagambetov et al., 2021). However, when CFs exist, TAO has so far relied on one-hot encoding. Here, we prove that all-subsets splits can be exactly optimized within TAO in linear time on the number of instances. Other recent approaches to learning trees are based on a brute-force optimization using mixed-integer optimization or dynamic programming, but they do not scale beyond small datasets (in sample size, dimensionality or cardinality) and small trees (Bertsimas & Dunn, 2017; Verwer & Zhang, 2019; Demirović et al., 2022).

In terms of handling CFs (Hastie et al., 2009; Kuhn & Johnson, 2013; 2019), different algorithms and software implementations make different choices. CART uses one-hot encoding in `scikit-learn` (Pedregosa et al., 2011) but all-subsets in `rpart` (Therneau et al., 2026); the latter uses a brute-force search in the multiclass case and has an exponential cost. CHAID (Kass, 1980), C4.5 and C5.0 (`https://www.rulequest.com`) use multiway splits combined with a number of clever heuristics, such as agglomerative clustering (some of them unpublished). Libraries

intended for decision forests such as XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017) started with one-hot encoding but later versions use some form of label embedding based on the squared loss used by gradient boosting (Friedman, 2001; Fisher, 1958), as does CatBoost (Prokhorenkova et al., 2018).

4 FRAMEWORKS FOR LEARNING AXIS-ALIGNED CLASSIFICATION TREES

Consider a K-class classification problem with a training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \{1, \dots, K\}$ of D-dimensional input features (which may be categorical or not) and labels. Our classifier is a tree with axis-aligned decision nodes and constant-label leaves. In detail, consider a binary decision tree (each decision node has exactly two children) with decision nodes indexed in set $\mathcal{D}$ and leaves in set $\mathcal{L}$, and $\mathcal{N} = \mathcal{D} \cup \mathcal{L}$. Define the decision function at a decision node i as $f_i(\mathbf{x}; \mathbf{w}_i, b_i) = \texttt{left}_i$ if $\mathbf{w}_i^T \mathbf{x} + b_i < 0$ and $\texttt{right}_i$ otherwise, where $\{\texttt{left}_i, \texttt{right}_i\} \subset \mathcal{N}$ are its left and right child, respectively; $b_i \in \mathbb{R}$ is a threshold; and the weight vector $\mathbf{w}_i$ has at most one non-zero component which can be either ± 1 if it uses a feature or (only for TAO, not CART) 0 if it does not ("empty feature"). Each leaf $j \in \mathcal{L}$ outputs a single class $c_j \in \{1, \dots, K\}$. The tree predictive function $T(\mathbf{x}; \boldsymbol{\Theta})$ (with nodes' parameters written jointly as $\boldsymbol{\Theta}$) routes $\mathbf{x}$ to a leaf which outputs its class.

Throughout the paper we will call "categorical" a feature whose possible values are unordered and with a finite domain (e.g. $\{\texttt{red}, \texttt{green}, \texttt{blue}\}$ or the 50 US states). Our focus is on computing splits, specifically all-subsets splits, for categorical features (CFs). Otherwise, if its values are ordered, we call it "non-categorical". This includes features with real or integer values but also discrete non-metric levels (e.g. preferences in a Likert scale). Splits for such features can be computed efficiently as with real-valued features, by testing a threshold at the midpoint between consecutive values.

Before we describe our exact, efficient optimization over a categorical split within TAO, we need to describe in some detail how decision trees are learned both with the traditional greedy recursive partitioning framework and with TAO.

4.1 GREEDY RECURSIVE PARTITIONING (CART, C5.0)

The fundamental idea is that the tree is grown top-down greedily by recursively splitting nodes and stopped at some point. There are many variations of this idea, but all share the fact that they do not optimize or even define a global loss function over the tree and its parameters (splits and leaf labels). This results in suboptimal, overly large trees.

The split in a decision node is determined by optimizing a local "purity" function, such as the Gini index in CART (Breiman et al., 1984) or the cross-entropy in C5.0 (Quinlan, 1993), which promotes partitions where classes have little straddle. For ordered features, this can be done efficiently by sorting them and enumerating each possible split (midpoint between consecutive values). For CFs in the special case of binary classification (two classes) this can also be done efficiently, but with more classes this becomes exponential on the CF cardinality. Once we stop growing the tree according to some criterion, each leaf is labeled with the majority class of the instances it contains. Such trees overfit easily if grown deeply, so they are pruned either by early stopping or by a cost-complexity optimization. The latter is a restricted but efficient form of cross-validation where tree splits are not modified, only removed.

4.2 TREE ALTERNATING OPTIMIZATION (TAO)

TAO works like any typical optimization-based ML algorithm: it defines an objective function involving a loss function and regularization term over a search space of models. The latter consists of parametric trees of fixed structure (e.g. a complete tree of depth Δ or, say, a given CART tree). TAO can handle any type of decision node (e.g. oblique) or leaf (e.g. linear regression) and any loss function, but here we focus on the 0/1 loss with axis-aligned decision nodes and leaves outputting a constant class label. We describe the corresponding algorithm below; for more details see Carreira-Perpiñán & Tavallali (2018); Carreira-Perpiñán & Hada (2026).

We define the following objective function over all the tree parameters:

$$\begin{aligned} \min E(\boldsymbol{\Theta}) = \sum_{n=1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda \sum_{i \in \mathcal{D}} \delta(\mathbf{w}_i) \\ \text{s.t. } \begin{cases} b_i \in \mathbb{R},\ \mathbf{w}_i \in \{\pm 1, 0\}^D \text{ with } \mathbf{w}_i^T \mathbf{w}_i \leq 1, & i \in \mathcal{D} \\ c_j \in \{1, \dots, K\}, & j \in \mathcal{L} \end{cases} \end{aligned} \tag{1}$$

where $\boldsymbol{\Theta} = \{(\mathbf{w}_i, b_i)\}_{i \in \mathcal{D}} \cup \{c_j\}_{j \in \mathcal{L}}$, $L(\cdot, \cdot)$ is the 0/1 loss function, and $\delta(\mathbf{w}) = 0$ if $\mathbf{w} = \mathbf{0}$ and 1 otherwise (the usual delta function). The "delta penalty" regularization term assigns a cost of λ for each node that uses a featureInstead of gradient-based methods, which do not apply to trees, we use alternating optimization one node at a time. Optimizing over the parameters of a node i given the parameters of all other nodes are fixed simplies into a certain *reduced problem (RP)*. Firstly (*separability condition*) the RP only depends on the instances that reach node i under the current tree, which we call the *reduced set (RS)* $\mathcal{R}_i \subset \{1, \dots, N\}$ of node i. The *reduced problem over a leaf* j takes the form $\min_{c_j \in \{1,\dots,K\}} \sum_{n \in \mathcal{R}_j} L(y_n, c_j)$, whose exact solution is to set c_j to the majority class in $\mathcal{R}_j$. The *reduced problem over a decision node* i takes the form:

$$\min_{\mathbf{w}_i, b_i} \sum_{n \in \mathcal{R}_i} l_{i,n}(f_i(\mathbf{x}_n; \mathbf{w}_i, b_i)) + \lambda\, \delta(\mathbf{w}_i) \quad \text{s.t.} \quad b_i \in \mathbb{R},\ \mathbf{w}_i \in \{\pm 1, 0\}^D \text{ with } \mathbf{w}_i^T \mathbf{w}_i \leq 1. \tag{2}$$

Here, $l_{i,n}(f_i(\mathbf{x}_n; \mathbf{w}_i, b_i)) = L(\overline{y}_n, f_i(\mathbf{x}_n; \mathbf{w}_i, b_i))$ is an instance-dependent loss where L is the original loss (the 0/1 loss, although any loss may be used, e.g. the squared error for regression) and the "pseudolabel" $\overline{y}_n$ for each instance $\mathbf{x}_n$ is the child (left or right) which leads to the lower prediction error (computed by propagating $\mathbf{x}_n$ down each respective child). The solution can be computed exactly by enumeration. For a non-categorical (ordered) feature $d \in \{1, \dots, D\}$ this can be done in $\Theta(N_i \log N_i)$, where $N_i = |\mathcal{R}_i|$ is the number of instances the node currently receives, by scanning in ascending order the midpoints (= possible splits) between feature values of x_{nd} for $n \in \mathcal{R}_i$. For a categorical feature (CF), enumeration means evaluating (2) over each of the possible splits and picking the best. Previous work on TAO has always used one-hot encoding for CFs, so only binary categorical splits occur in the tree. In that case, enumeration is over two values rather than over 2^C if there were C categoriesIn section 5 we give an exact solution in $\Theta(N)$ (time linear on the number of instances).

Because of the separability condition, nodes at the same depth define independent RPs and can be solved in parallel. We update all nodes in reverse breadth-first search order (leaves to root) and this constitutes one TAO iteration, which is guaranteed to decrease the global objective function (1). Repeating this leads to convergence in a finite number of iterations, when (1) does not change anymore (similar to what happens with k-means). The runtime per iteration is $\Theta(DN \log N)$ if there are no CFs, but it becomes exponential on each CF's cardinality C otherwise.

Pruning occurs as a side effect of training due to the delta penalty. If a node has no feature then it sends all instances to the same child, so that node and the other child's subtree can be removed. We do this after convergence. This affords a search over tree structures that are contained within the initial tree. We select the hyperparameter λ (hence the tree size and structure) by cross-validation. Optimizing over a set of λ values is computationally efficient via warm-start, starting for $\lambda = 0$ and the initial tree. As λ increases, more nodes use no feature, until this happens for all nodes (which means the tree is equivalent to a depth-zero, single-node tree that outputs the majority class in the training set). This is guaranteed to happen if $\lambda \geq N - N_1$ where N_1 is the number of instances in the majority class. We construct the regularization path for $\lambda = 0, 1, 2 \dots$ until the tree collapses to one node or the error is unacceptably large. We take as initial tree a fully grown CART tree (i.e., without pre- or post-pruning).

Note how TAO shares some operations with CART: in a leaf, setting its label to its majority class; in a decision node, setting its split by enumerating an objective function over splits. The critical difference is that the leaf and split objective function in TAO are derived from the global objective function (1) and guarantee its decrease at each iteration, while in CART the split objective function (purity) is ad-hoc. Also, CART's cost-complexity pruning resembles the automatic pruning due to the delta penalty. But in CART this is a form of cross-validation that is limited to removing subtrees without changing any split, while in TAO it happens during training because the decision nodes can change their feature over iterations, possibly choosing no feature. Cross-validation with TAO is as usual in ML by doing a grid search over the hyperparameter λ.

5 EXACT, EFFICIENT SOLUTION OF ALL-SUBSETS SPLIT WITH TAO

We now give an efficient (linear time) solution to the reduced problem (RP) over a decision node with a categorical feature (CF). It works for any loss function (including least-squares regression, although we focus on multiclass classification) and binary or multiway splits. Recall the RP over a CF in TAO, eq. (2), where optimizing the split will try *all* features (categorical or not) and pick the one with lowest loss (plus λ for the delta penalty), including picking no feature (which sends all instances to the best child). Trying a given feature means enumerating all possible splits on that feature. For a CF with C categories, naive enumeration evaluates eq. (2) over 2^C splits. We now show that *this can be done exactly in* $\Theta(N)$. The key idea is to realize that in TAO we know for each instance what the loss will be on each child of the node (by propagating them down each child subtree to a leaf), and to show that eq. (2) separates over the C values. This does not happen for CART because the child subtrees do not exist yet; the tree is grown on the fly.

Consider then a CF $d \in \{1, \dots, D\}$ with $x_d \in \mathcal{C}_d$ having $C_d = |\mathcal{C}_d|$ categories. In order to handle natively all-subsets splits for a CF, we need to redefine the form of the split in section 4 so that it uses an all-subsets split rather than a weight and threshold. Our notation is somewhat complex (in order to identify the node i, feature d, instance n, etc.) but it is precise. We define the split parameters via a table with C_d entries for the mapping $F_{i,d}$: $\mathcal{C}_d \rightarrow \{\texttt{left}_i, \texttt{right}_i\}$, where each $F_{i,d}(x) = k$ maps a category x to a child k of the node (here two children, but this generalizes directly to more than two, i.e., a multiway split). This can be considered a "nonparametric" model, in that the output for every single input is a free parameter. The table partitions the categories among the children and captures the full search space for an all-subsets split; in fig. 1a: $F(\blacksquare) = F(\clubsuit) = F(\spadesuit) = \texttt{left}_i$ and $F(\bigcirc) = F(\diamondsuit) = F(\heartsuit) = \texttt{right}_i$. Thus $\{x \in \mathcal{C}_d\text{: } F_{i,d}(x) = k\}$ is the set of categories routed to child k. Also note that, while all instances with the same category value go down the same child, those instances follow different paths (because these depend on the whole feature vector $\mathbf{x}$) and reach different leaves down that child's subtree and incur different losses.

Consider then the split on CF d in node i. The optimization over $F_{i,d}$ (ignoring the additive constant λ) is:

$$\min_{F_{i,d}} E_i(F_{i,d}) = \textstyle\sum_{n \in \mathcal{R}_i} l_{i,n}(F_{i,d}(x_{nd})) \tag{3}$$

where $l_{i,n}$: $\{\texttt{left}_i, \texttt{right}_i\} \rightarrow \mathbb{R}$ is the loss function for instance n defined in eq. (2). That is, instance n incurs a loss $l_{i,n}(\texttt{left}_i)$ and $l_{i,n}(\texttt{right}_i)$ if going down the left or right child, respectively, and we want to pick the child $F_{i,d}(x)$ for each category x that minimizes this across all instances in i's RS. Note there may be zero, one or multiple instances in the RS having the same category x. Expressed this way, now it is obvious that E_i separates over the C_d categories, i.e., calling $F_{i,d}(x) = k$, we have:

$$\begin{aligned} \min_{F_{i,d}} E_i(F_{i,d}) &= \sum_{x \in \mathcal{C}_d} \sum_{\substack{n \in \mathcal{R}_i \\ x_{nd} = x}} l_{i,n}(F_{i,d}(x_{nd})) \\ &\Longrightarrow \text{for each } x \in \mathcal{C}_d\text{: } F_{i,d}(x) = \operatorname*{arg\,min}_{k \in \{\texttt{left}_i, \texttt{right}_i\}} \sum_{\substack{n \in \mathcal{R}_i \\ x_{nd} = x}} l_{i,n}(k) \end{aligned} \tag{4}$$

where the summation inside the $\arg\min$ is over the RS instances in category x. This can be solved by enumeration in $\Theta(2N_i)$ if we loop (once per child) over the $N_i = |\mathcal{R}_i|$ RS instances and accumulate the loss for each corresponding category x_{nd}. The pseudocode (fig. 6(right)) gives instead a direct version of eq. (4) that is suitable for a parallel implementation: a nested for loop over the C_d categories, over the two children and over the instances in the RS. In English, we compute the optimal child for each category separately by 1) adding the loss over all the RS instances in that category if sending them to the left or to the right child, and 2) picking the child with lower aggregate loss. In a sense, we have a classification problem with one class per child and a 1D feature vector that is categorical.

Two "don't care" cases can occur in the minimization, where $F_{i,d}(x)$ is not defined uniquely for a category x: if category x is "missing" in the RS, i.e., no instance n in the RS has a value $x_{nd} = x$, then any child is optimal; or if the category is present but there is a tie, i.e., the minimum loss is achieved by more than one child. Although which child to choose does not matter at node i during the split optimization, it can matter downstream from node i or at inference time. We set $F_{i,d}(x)$ to a child picked at random (during training). We tried other strategies but did not find one that

is consistently better (see appendix B). Missing categories happen frequently with high-cardinality CFs, particularly in deep nodes (which receive fewer instances).

6 EXPERIMENTS

We compare our method with GUIDE, XGBoost, CatBoost, C5.0, and CART on 14 datasets (four categorical-only and ten mixed), with categorical cardinalities up to 2,217. We report the main accuracy, size and interpretability results here. The appendix contains the setup, running times, full metrics and missing-category variants, regularization/training dynamics, and equality-test comparison trees.

6.1 PREDICTIVE PERFORMANCE AND TREE COMPACTNESS

Tables 1 and 2 summarize test error and tree size, while Appendix Tables 4 and 5 give the full metrics and missing-category strategies. All-subsets TAO has the lowest test error on 13 of 14 datasets and the second-lowest on the remaining one, often with substantially fewer leaves.

The gains are clearest at medium, and high cardinality. Replacing equality-test nodes by the proposed all-subsets nodes for TAO improves both test error and leaf count on all seven datasets in Table 1. On CFEC2 (up to 2,217 categories), equality-test TAO obtains 19.55%/76 leaves versus 18.59%/18 for all-subsets TAO; CatBoost and GUIDE obtain 18.63%/31 and 18.77%/32. On HBD (up to 169 categories), the corresponding TAO comparison is 13.96%/2,282 versus 13.80%/540, while CatBoost and GUIDE use 2,536 and 2,802 leaves with higher error. BeijingAqi is the exception: XGBoost has lower error (23.59% vs. 23.91%) but a larger tree (1,535 vs. 1,149 leaves). Therefore, the proposed categorical decision nodes account for much of the high-cardinality accuracy and compactness gain.

For datasets with lower cardinality, TAO still delivers the lowest test errors with trees that are often a fraction of the size of CART and XGBoost. On *Presidential election 2016*, TAO improves upon GUIDE's accuracy (18.47% vs. 19.74% test error) while shrinking the tree from 785 leaves to just 43. Similarly, on the low-cardinality Contraceptive dataset, TAO drops the test error to 38.64% using only 7 leaves, whereas CART requires 85 leaves for a 44.07% error. Appendix Table 3 gives runtimes: categorical TAO takes 6.18 s on CFEC2, 28.78 s on HBD, and 9.06 s on SF Crime. Boosting implementations are generally faster, but one-hot CART takes 705.91 s on CFEC2, due to the fact that size of the dataset blows up after encoding. R CART times out on Adult, BeijingAqi, and SF Crime, due to brute force method of split optimization.

6.2 INTERPRETABILITY ON THE 2016 ELECTION DATASET

We remove identifiers (`rownames`, `uid`), derived variables (`lemprac`, `lcograc`, `lrelig`), and the strongly target-associated ideology variables `pid7na` and `ideo`, leaving directly interpretable demographic, religious, and attitudinal features such as `age`, `racef`, `state`, `churchatd`, and `whiteadv`. The appendix contains the corresponding regularization analysis and equality-test comparison trees (Figures 4, 8, and 9).

To isolate the value of categorical features and split type, we compare TAO without categorical features, with equality-test splits, and with our all-subsets splits. Figure 4 shows roughly 1% reduction in test error from adding categorical features and another 1% reduction from the proposed splits.

Figure 2 shows a more regularized all-subsets tree (about 19% test error) chosen for readability. Our tree is of depth 6 and 12 leaves, versus similar-error equality-test trees from CART (depth 12, 64 leaves; Figure 9) and TAO (depth 12, 54 leaves; Figure 8). The structure follows a logical hierarchy, where high-level ideological splitters (such as `whiteadv`) occupy the root, while demographic filters (`age`, `racef`), binary indicators (`bornagain`), and high-cardinality geographic splits (`state`) appear deeper to resolve specific edge cases.

The root `whiteadv` ("White people have advantages") split creates two linear cascades of decision rule. On the left, respondents on the side acknowledging white advantage are refined by `age` and `racerare`. The younger respondents (`age` < 41.5) or those reporting racism as more common (`racerare` ¡ 2.5) reach non-Trump leaves, while older ones who view racism as rarer can reach a

Table 1: Test error and tree size (depth/leaves) on datasets with medium (≥ 14) and high (≈ 100) categorical features. **Best** and second best test errors, and **lowest** and second lowest leaf counts are highlighted.

		label embedding			multiway	equality test		all - subsets	
		XGBoost	CatBoost	GUIDE	C5.0	CART	TAO	CART	TAO
Election 2016 (31250, 14, 2)	$E_{\texttt{test}}$	19.09	19.21	19.74	19.00	19.31	19.21	19.49	**18.47**
	Δ/leaves	9/77	15/623	19/785	11/133	9/65	10/70	15/135	10/**43**
Adult (36177, 14, 4)	$E_{\texttt{test}}$	45.96	44.27	45.11	43.05	43.25	43.24	-	**42.83**
	Δ/leaves	9/33	12/378	5/32	11/22	19/118	18/78	-	8/**12**
Albert (52426, 31, 2)	$E_{\texttt{test}}$	35.77	34.62	34.31	34.16	34.55	33.75	33.76	**33.53**
	Δ/leaves	12/112	8/201	5/31	18/969	10/87	10/40	15/118	7/**21**
HBD (95508, 27, 2)	$E_{\texttt{test}}$	35.99	14.96	15.62	14.08	14.09	13.96	14.77	**13.80**
	Δ/leaves	20/**387**	16/2536	18/2802	20/677	48/3022	44/2282	26/974	30/540
CFEC2 (238436, 23, 2)	$E_{\texttt{test}}$	24.04	18.63	18.77	19.88	19.79	19.55	24.04	**18.59**
	Δ/leaves	25/1375	12/31	5/32	8/5296	10/86	10/76	10/115	8/**18**
BeijingAqi (200000, 16, 6)	$E_{\texttt{test}}$	**23.59**	25.70	23.95	24.33	24.78	24.37	-	23.91
	Δ/leaves	22/1535	13/6063	18/6850	19/1802	21/1262	18/1215	-	18/**1149**
SF Crime Class. (181931, 8, 20)	$E_{\texttt{test}}$	73.56	73.16	73.75	74.77	72.49	72.26	-	**72.15**
	Δ/leaves	26/1123	13/1990	18/4270	24/29273	22/699	19/601	-	23/**547**

Table 2: Test error and tree size (depth/leaves) on datasets with low-cardinality categorical features (< 14). **Best** and second best test errors, and **lowest** and second lowest leaf counts are highlighted.

		label embedding			multiway	equality test		all - subsets	
		XGBoost	CatBoost	GUIDE	C5.0	CART	TAO	CART	TAO
Contraceptive (1178, 9, 3)	$E_{\texttt{test}}$	47.12	47.80	40.34	44.75	45.76	41.02	44.07	**38.64**
	Δ/leaves	3/**4**	8/26	5/21	10/40	14/118	11/63	15/85	4/7
Car eval. (1382, 6, 4)	$E_{\texttt{test}}$	2.89	20.52	4.62	6.94	3.18	2.89	2.31	**1.16**
	Δ/leaves	9/**27**	16/114	11/39	6/76	14/101	12/60	10/50	12/60
Telco (5625, 19, 2)	$E_{\texttt{test}}$	21.46	22.25	20.26	19.83	20.82	20.75	20.04	**18.91**
	Δ/leaves	6/**15**	12/218	5/32	10/51	11/63	9/30	16/90	8/17
Electricity (34626, 8, 2)	$E_{\texttt{test}}$	12.84	17.10	12.86	10.65	12.40	11.85	12.34	**10.63**
	Δ/leaves	22/1257	16/3329	18/1569	28/1471	27/1466	28/1841	25/935	25/**688**
Bank Marketing (38429, 16, 2)	$E_{\texttt{test}}$	9.67	9.48	9.38	9.22	9.19	9.20	9.10	**8.51**
	Δ/leaves	15/89	12/239	5/31	11/139	17/164	10/44	14/86	9/**20**
Chess KRK (28056, 6, 18)	$E_{\texttt{test}}$	28.69	73.91	35.92	38.58	21.40	27.23	20.56	**20.46**
	Δ/leaves	15/386	10/**35**	18/1610	6/3027	28/4408	19/2221	21/5672	28/1947
Jungle Chess (44819, 6, 3)	$E_{\texttt{test}}$	13.26	16.21	16.73	17.22	14.44	15.10	13.49	**10.17**
	Δ/leaves	16/733	13/3170	18/1256	6/1043	27/1089	17/708	19/738	19/**513**

Trump leaf. On the right, stronger denial (`whiteadv` ≥ 3.5) leads directly to a Trump-dominated leaf. Conversely, this list prioritizes the isolation of high-confidence Trump voters immediately. The remaining cases are refined by `racef`, `bornagain`, and `state`. The categorical nodes are easily interpretable as `racef` forms two explicit category sets, and `state` learns a partition shown directly on the map. A single all-subsets node can therefore express groupings that equality-test trees require many nodes to represent, as the appendix trees make clear.

7 Conclusion

Properly handling categorical features, particularly if they have many categories, has been a long-standing problem with classification trees. This is due to the difficulty of optimizing over a search

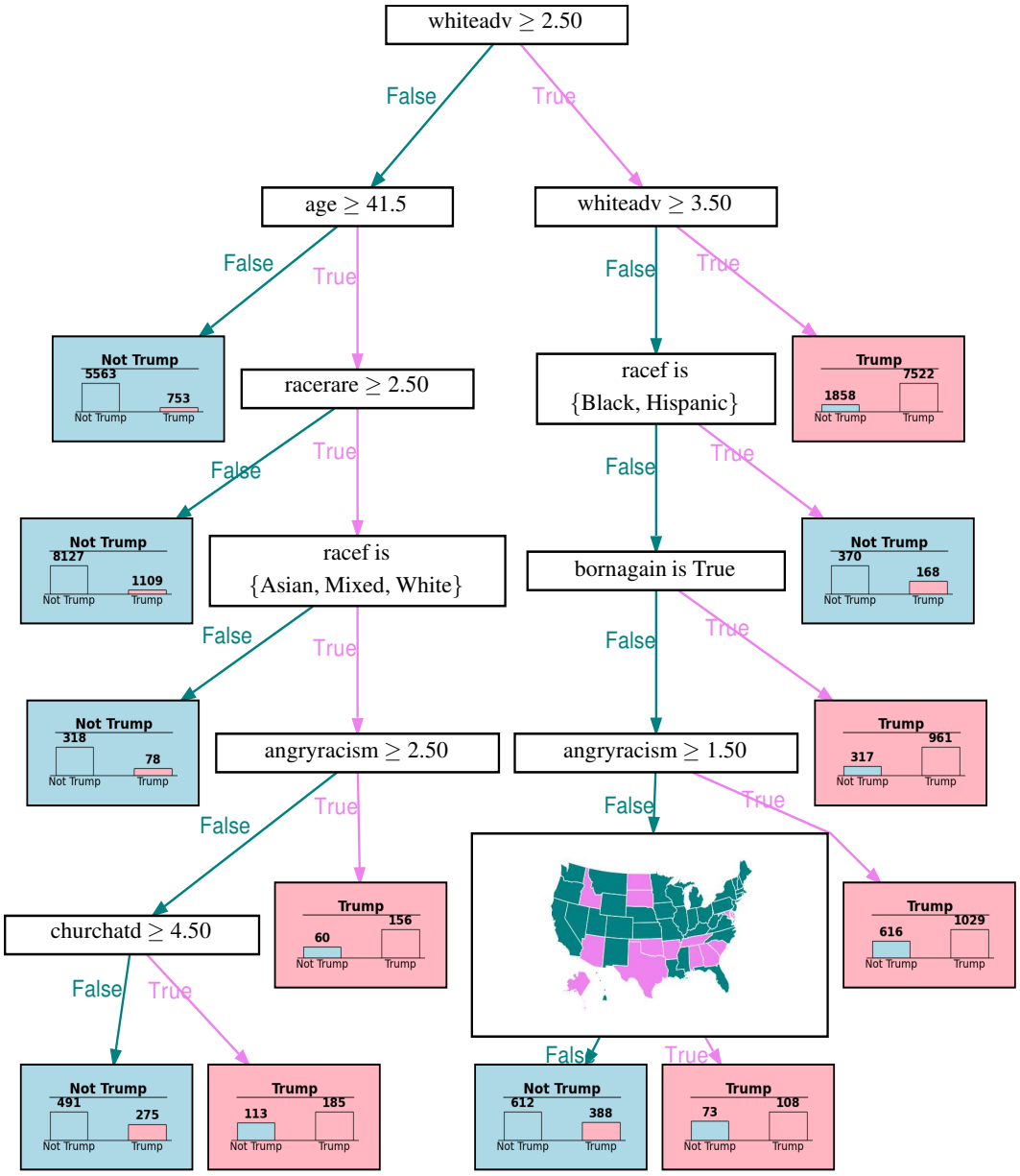

Figure 2: Our tree trained on Presidential election 2016 dataset ("The Individual Correlates of the Trump Vote"). We show categorical split in decision nodes using set (racef feature) and small US map to show states. We use bar chart to show class distribution in each leaf.

space that contains a number of possible splits that is exponential on the number of categories. In practice this means that trees typically use a weaker type of split, based on a one-hot encoding or label embedding. We provide a complete solution to this problem not within the framework of greedy recursive partitioning but within that of tree alternating optimization (TAO), due to special conditions that arise in the latter when optimizing over a single decision node. Our solution finds the globally optimum all-subsets split for a CF in time that is linear on the problem size, hence efficient even with high-cardinality CFs; it works with binary or multiway splits; and it also benefits from the fact that TAO (unlike CART or C5.0) optimizes a global objective function over the tree parameters. This makes it possible to use CFs natively, without the need for any encoding, and it improves the final tree in all respects: accuracy and especially size and interpretability.

AI USE STATEMENT

In this work, we have not used generative AI tools. We take responsibility for the final content of this work, including text, claims or artifacts.

REFERENCES

S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.). *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018. MIT Press, Cambridge, MA.

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

Leo J. Breiman, Jerome H. Friedman, R. A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

Miguel Á. Carreira-Perpiñán and Suryabhan Singh Hada. More interpretable decision trees: Pruning via node descent and the delta penalty. In Maria De Marsico, Tin Kam Ho, Frederic Jurie, Cheng-Lin Liu, Daniel Lopresti, Ingela Nyström, Jean-Marc Ogier, Arun Ross, and Liang Wang (eds.), *Proc. 28th Int. Conf. Pattern Recognition (ICPR'26)*, pp. 431–446, Lyon, France, August 17–22 2026.

Miguel Á. Carreira-Perpiñán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In Bengio et al. (2018), pp. 1211–1221.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proc. of the 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2016)*, pp. 785–794, San Francisco, CA, August 13–17 2016.

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. *J. Machine Learning Research*, 23(26):1–47, 2022.

Walter D. Fisher. On grouping for maximum homogeneity. *J. Amer. Stat. Assoc.*, 53(284):789–798, 1958.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

Trevor J. Hastie, Robert J. Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning—Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer-Verlag, second edition, 2009.

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

G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *J. Statistical Society C*, 29(2):119–127, June 1980.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. v. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pp. 3146–3154. MIT Press, Cambridge, MA, 2017.

Hyunjoong Kim and Wei-Yin Loh. Classification trees with unbiased multiway splits. *J. Amer. Stat. Assoc.*, 96(454):589–604, 2001.

Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer-Verlag, 2013.

Max Kuhn and Kjell Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman & Hall/CRC, 2019.

Wei-Yin Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12(2):361–386, April 2002.

Wei-Yin Loh. Improving the precision of classification trees. *Annals of Applied Statistics*, 3(4): 1710–1737, 2009.

Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82 (3):329–348 (with discussion, pp. 349–370), December 2014.

Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica Sinica*, 7 (4):815–840, October 1997.

Wei-Yin Loh and Nunta Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *J. Amer. Stat. Assoc.*, 83(403):715–725, September 1988.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. In *Proc. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *J. Machine Learning Research*, 12:2825–2830, October 2011. Available online at `https://scikit-learn.org`.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In Bengio et al. (2018), pp. 6638–6648.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, U.K., 1996.

Lior Rokach and Oded Maimon. *Data Mining With Decision Trees. Theory and Applications*. Number 81 in Series in Machine Perception and Artificial Intelligence. World Scientific, second edition, 2015.

Terry Therneau, Beth Atkinson, and Brian Ripley. rpart: Recursive partitioning and regression trees. R package version 4.1.27, March 27 2026. Available online at `https://cran.r-project.org/package=rpart`.

Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pp. 1625–1632, Honolulu, HI, January 27 – February 1 2019.

Arman Zharmagambetov, Suryabhan Singh Hada, Magzhan Gabidolla, and Miguel Á. Carreira-Perpiñán. Non-greedy algorithms for decision tree optimization: An experimental comparison. In *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, July 18–22 2021.

A APPENDIX

A.1 SETUP

We implemented our algorithm in Python 3.11.11 and parallelized using Ray 2.44.1 Moritz et al. (2018) on CPU. All the experiments were conducted on Intel Xeon CPU E5-2699 v3 @ 2.30 GHz, 256 GB RAM.

TAO The initial tree structure was generated using the scikit-learn CART implementation Pedregosa et al. (2011) with `ccp_alpha` $= 0$ to ensure a fully grown tree. We perform a regularization path using the parameter λ to prune the tree, selecting the smallest tree that maximizes performance on the validation set. We restrict the number of TAO iterations to 30 or terminate early if the difference in training error between iterations falls below 10^{-5}.

CART We evaluated both the Python scikit-learn and R versions of the algorithm. Since the Python implementation lacks explicit support for categorical features, we used one-hot encoding. To learn the tree structure, we grew the tree fully and cross-validated the `ccp_alpha` parameter to identify the optimal tree, in accordance with the original CART methodology. On the other hand, the R version supports categorical features natively, requiring no encoding. Similarly, for the R implementation, we set `min_split` and `min_bucket` to 1 and tuned `cp` to produce the best tree on the validation set.

GUIDE We implemented a custom version of the GUIDE algorithm following the methods described in Loh (2014) and Loh (2009). This implementation grows trees greedily and recursively using statistical tests to eliminate variable selection bias. Additionally, it employs the sort-by-class probability method to split categorical features efficiently without one-hot encoding. We cross-validated the maximum depth, minimum samples for split, and minimum samples per leaf.

C50 We use R version of C50 algorithm Quinlan (1993). We cross-validated the `CF` parameter, which controls the amount of pruning, we set `use_categorical` to True, which creates multiway split, and `trials` to 1 to ensure a single tree is grown.

CatBoost Our primary goal was to train a single, highly optimized tree rather than perform standard boosting. As training was performed on a CPU, the maximum depth was restricted to 16. We cross-validated the `grow_policy` (Lossguide vs. Depthwise) to explore different tree topologies. Lossguide employs a greedy, aggressive approach adapted from LightGBM, where splits are chosen to maximize loss reduction, whereas Depthwise grows the tree level-by-level, favoring balanced structures. We also cross-validated `min_data_in_leaf`, `depth`, and `l2_leaf_reg`.

XGBoost Our primary goal was to train a single, highly optimized tree rather than perform standard boosting.We set `enable_categorical=True` and `tree_method="hist"`. We cross-validated `max_depth` and `gamma`, while fixing `learning_rate = 1`. In the single-tree context, `gamma` acts as a strict pruning mechanism: small values allow greedy splits that increase model complexity, while large values restrict splits to those offering significant objective improvement, resulting in sparser trees.

B SELECTING THE NODE CHILD WITH DON'T CARE CATEGORIES

We analyze the regularization path (Figure 3) and the learning curve (Figure 7) of our proposed method on Electricity dataset. We report results for three strategies of handling missing categories: Full, Left, and Random. We start with fully grown CART tree with indicator split for categorical features.

In Figure 3, we illustrate how the penalty hyperparameter λ impacts tree structure and accuracy, highlighting the advantages of our categorical handling method. Increasing λ restricts model capacity, which causes the training error to steadily rise while the test error initially drops to a minimum before increasing. The vertical dashed line marks the optimal model chosen via cross-validation, effectively balancing accuracy and tree sparsity. Compared to standard trees without categorical splits (shown by the green line), our approach significantly reduces test error by approximately 2%. Because the proposed categorical splits are more powerful, the optimal model achieves this higher

accuracy while requiring fewer total decision nodes. Furthermore, this increased splitting power allows the optimization algorithm to explore a broader range of λ values during model selection. After some threshold value of λ, number of categorical decision nodes become zero. At this point, predictive power of categorical features is not enough to justify the cost of a decision node, and the algorithm prunes them.

Importantly, the number of categorical decision nodes is much smaller ($\approx$40 vs $\approx$600) for "Full" strategy, since it requires all categories to be present in the given reduced set. Naturally, such nodes appear closer to root of the tree, where the number of samples is larger.

Figure 7 demonstrates the evolution of the model metrics over the total TAO iterations. The number of categorical decision nodes is 0 for $\lambda = 0$, which is expected since the initial tree is fully grown and the optimization process has not yet begun. The plots reveal a distinct "sawtooth" pattern, driven by the alternating schedule of increasing λ and re-optimizing the tree. For each value of λ objective function goes down monotonically, with accordance to TAO theory, indicating that the TAO algorithm has successfully converged. The second row highlights the non-monotonic behavior of the generalization error. The global minimum is typically found at intermediate iterations, where the model optimally balances predictive power and structural sparsity. It is clear that all three methods of handling missing categories perform similarly in terms of test error and training dynamics, and differ slightly at which λ they achieve the best generalization error.

Finally, the comparison across columns in both figures demonstrates that the behavior of our algorithm is highly consistent regardless of the strategy used for missing categories (Full, Left, or Random). The trajectories for error and sparsity are nearly identical, indicating the robustness of our approach. While the "Left" and "Random" strategies initially instantiate a high volume of categorical splits before undergoing aggressive pruning, the "Full" strategy demonstrates fluctuating categorical node counts throughout the iterations. Under the "Full" strategy, as λ changes, the tree undergoes complex optimization and feature selection by dynamically adjusting the reduced sets, so it discovers optimal ones that encompass all categories, producing highly optimized tree structures while maintaining a consistently lower overall reliance on categorical splits.

Furthermore, Figure 5 plots error versus model size (decision nodes) to compare our missing-category strategies against the baseline identity split. Our approach yields a strictly better trade-off, achieving lower errors with smaller trees (lower left corner of the both plots). All three proposed strategies generalize similarly, reaching $\approx$11% test error compared to the baseline's $\approx$13%. To match the baseline's accuracy, our method requires 9x fewer nodes, drastically reducing complexity and improving interpretability.

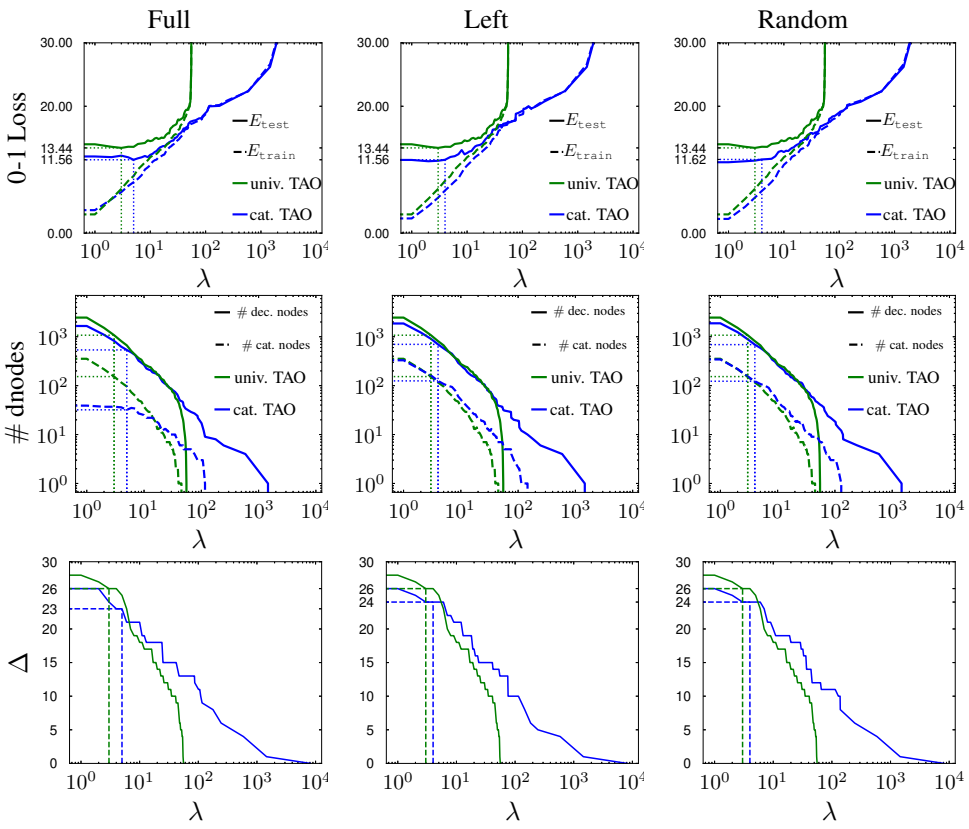

Figure 3: Regularization path produced by our approach on Electricity dataset. We report train and test error, a total number of decision nodes (log scale), decision nodes with categorical split, and depth Δ. Each column is produced by different way of handling missing categories. Green line shows TAO tree trained without categorical decision nodes.

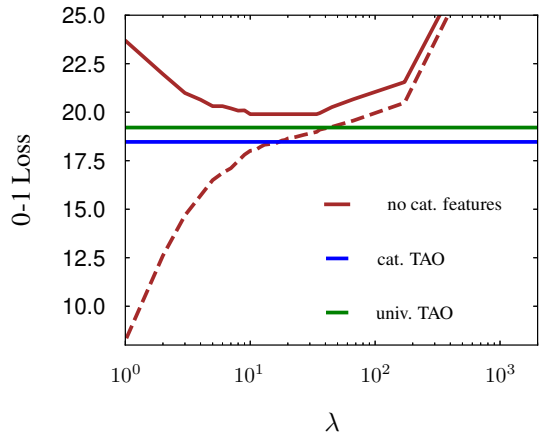

Figure 4: Regularization path produced by TAO algorithm trained on Election 2016 dataset without categorical features. Green line represents best TAO tree with OvO embedded categorical features. Blue line represents performance of the proposed approach.

Table 3: Running time (seconds) per method across datasets

		GUIDE	XGBoost	CatBoost	CART (R)	CART	TAO	C50	cat. TAO
Contraceptive (1473, 9, 3) $\text{card}\,(X_{\text{cat}}) \leq 4$	Time (s)	0.17	0.01	0.02	0.04	0.00	0.52	0.05	0.52
Car Evaluation (1728, 6, 4) $\text{card}\,(X_{\text{cat}}) \leq 4$	Time (s)	0.14	0.02	0.06	0.03	0.00	1.18	0.04	1.18
Telco (7032, 19, 2) $\text{card}\,(X_{\text{cat}}) \leq 4$	Time (s)	0.75	0.03	0.06	0.23	0.03	1.24	0.34	1.20
Chess Krk (28056, 6, 18) $\text{card}\,(X_{\text{cat}}) \leq 8$	Time (s)	5.23	0.15	2.64	0.45	0.12	5.64	0.58	6.78
Electricity (38474, 8, 2) $\text{card}\,(X_{\text{cat}}) \leq 7$	Time (s)	21.16	0.14	0.55	0.66	0.26	15.18	1.51	14.84
Election 2016 (39063, 14, 2) $\text{card}\,(X_{\text{cat}}) \leq 51$	Time (s)	9.33	0.13	0.19	0.88	0.28	1.62	1.31	1.70
Jungle Chess (44819, 6, 3) $\text{card}\,(X_{\text{cat}}) \leq 9$	Time (s)	5.26	0.08	0.54	0.54	0.18	3.28	0.82	3.60
Bank Marketing (45211, 16, 2) $\text{card}\,(X_{\text{cat}}) \leq 12$	Time (s)	4.35	0.08	0.19	1.32	0.29	3.04	2.03	2.40
Adult (45222, 14, 4) $\text{card}\,(X_{\text{cat}}) \leq 41$	Time (s)	2.24	0.12	0.64	-	1.07	4.22	1.98	3.94
HBD (119386, 27, 2) $\text{card}\,(X_{\text{cat}}) \leq 169$	Time (s)	57.92	0.29	1.21	5.52	2.93	28.74	8.48	28.78
SF Crime Classification (181931, 8, 20) $\text{card}\,(X_{\text{cat}}) \leq 100$	Time (s)	44.03	0.61	18.74	-	29.71	9.42	6.54	9.06
BeijingAqi (200000, 16, 6) $\text{card}\,(X_{\text{cat}}) \leq 31$	Time (s)	126.00	0.65	7.06	-	5.09	27.06	11.92	26.00
CFEC2 (298042, 23, 2) $\text{card}\,(X_{\text{cat}}) \leq 2217$	Time (s)	29.35	1.76	1.60	13.11	705.91	4.26	27.34	6.18

Table 4: Comparison of our proposed approach and other tree-based methods: GUIDE, XGBoost, CatBoost, CART on dataset with categorical features of medium (≥ 14) and high cardinality ($\approx$100) and mixed features. We report train and test error, depth (Δ), number of leaves, number of decision nodes with categorical splits, and proportion of decision nodes with categorical splits to all decision nodes. For our approach we report 3 different strategies of handling missing categories. We mark **first** and second best error; **first** and second lowest leaf count.

		label embedding			multiway	equality test		all - subsets			
		XGBoost	CatBoost	GUIDE	C5.0	CART	TAO	CART	TAO full	TAO left	TAO random
Election 2016 (31250, 14, 2) card $(X_{\text{cat}}) \leq 51$	E_{train}	17.38	17.47	15.61	18.59	17.91	17.26	15.46	17.53	16.70	16.70
	E_{test}	19.09	19.21	19.74	19.00	19.31	19.21	19.49	**18.47**	18.90	18.90
	Δ	9	15	19	11	9	10	15	10	9	9
	Cat prop	0.38	0.34	0.09	0.63	0.11	0.10	0.43	0.12	0.36	0.36
	# cat nodes	29	211	71	83	7	7	57	5	20	20
	# leaves	77	623	785	133	65	70	135	**43**	55	55
Adult (36177, 14, 4) card $(X_{\text{cat}}) \leq 41$	E_{train}	42.62	43.79	45.05	42.83	42.20	41.92	-	42.97	42.94	42.94
	E_{test}	45.96	44.27	45.11	43.05	43.25	43.24	-	43.12	**42.83**	**42.83**
	Δ	9	12	5	11	19	18	-	8	8	8
	Cat prop	0.41	0.82	0.65	0.81	0.26	0.27	-	0.21	0.55	0.55
	# cat nodes	13	311	20	17	31	21	-	3	6	6
	# leaves	33	378	32	22	118	78	-	15	**12**	**12**
Albert (52426, 31, 2) card $(X_{\text{cat}}) \leq 14$	E_{train}	33.37	34.74	35.41	29.77	34.01	34.04	32.45	34.53	34.38	34.38
	E_{test}	35.77	34.62	34.31	34.16	34.55	33.75	33.76	**33.53**	33.69	33.69
	Δ	12	8	5	18	10	10	15	7	7	7
	Cat prop	0.38	0.27	0.13	0.54	0.29	0.44	0.38	0.19	0.21	0.21
	# cat nodes	42	54	4	524	25	17	44	4	5	5
	# leaves	112	201	31	969	87	40	118	**21**	25	25
HBD (95508, 27, 2) card $(X_{\text{cat}}) \leq 169$	E_{train}	12.67	14.66	9.54	13.14	6.69	7.70	9.34	11.95	11.70	11.05
	E_{test}	35.99	14.96	15.62	14.08	14.09	13.96	14.77	**13.80**	14.45	14.01
	Δ	20	16	18	20	48	44	26	30	21	27
	Cat prop	0.66	0.76	0.54	0.50	0.56	0.42	0.71	0.12	0.73	0.72
	# cat nodes	256	1936	1526	339	1681	958	686	65	215	282
	# leaves	387	2536	2802	677	3022	2282	974	540	**296**	391
CFEC2 (238436, 23, 2) card $(X_{\text{cat}}) \leq 221$	E_{train}	14.96	18.56	18.73	15.84	18.76	18.55	13.11	18.54	17.12	16.68
	E_{test}	24.04	18.63	18.77	19.88	19.79	19.55	24.04	**18.59**	19.09	19.21
	Δ	25	12	5	8	10	10	10	8	10	10
	Cat prop	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	# cat nodes	1374	30	31	1472	85	75	114	17	17	17
	# leaves	1375	31	32	5296	86	76	115	**18**	**18**	**18**
BeijingAqi (200000, 16, 6) card $(X_{\text{cat}}) \leq 31$	E_{train}	09.23	19.55	15.70	21.66	22.08	21.43	-	19.77	16.69	19.43
	E_{test}	**23.59**	25.70	23.95	24.33	24.78	24.37	-	25.29	23.91	24.05
	Δ	22	13	18	19	21	18	-	17	18	17
	Cat prop	0.49	0.10	0.49	0.60	0.45	0.44	-	0.35	0.63	0.63
	# cat nodes	760	620	3345	1084	572	545	-	353	724	513
	# leaves	1535	6063	6850	1802	1262	1215	-	996	1149	**811**
SF Crime Classification (181931, 8, 20) card $(X_{\text{cat}}) \leq 100$	E_{train}	63.34	69.60	65.79	43.00	70.70	70.43	-	69.23	68.73	68.12
	E_{test}	73.56	73.16	73.75	74.77	72.49	72.26	-	72.37	**72.15**	72.38
	Δ	26	13	18	24	22	19	-	17	23	24
	Cat prop	0.24	0.17	0.23	0.81	0.17	0.23	-	0.04	0.39	0.36
	# cat nodes	276	348	968	23647	120	140	-	8	213	319
	# leaves	1123	1990	4270	29273	699	601	-	**217**	547	877

Table 5: Comparison of our proposed approach and other tree-based methods: GUIDE, XGBoost, CatBoost, CART on dataset with categorical features of low cardinality (< 14) and mixed features. We report train and test error, depth (Δ), number of leaves, number of decision nodes with categorical splits, and proportion of decision nodes with categorical splits to all decision nodes. For our approach we report 3 different strategies of handling missing categories. We mark **first** and second best error; **first** and second lowest leaf count.

		label embedding			multiway	equality test		all - subsets			
		XGBoost	CatBoost	GUIDE	C5.0	CART	TAO	CART	TAO full	TAO left	TAO random
Contraceptive (1178, 9, 3) $\text{card}(X_{\text{cat}}) \leq 4$	E_{train}	45.59	47.71	42.70	37.86	28.44	32.43	29.54	43.38	41.00	41.00
	E_{test}	47.12	47.80	40.34	44.75	45.76	41.02	44.07	**38.64**	39.66	39.66
	Δ	3	8	5	10	14	11	15	4	7	7
	Cat prop	0.33	0.56	0.50	0.59	0.38	0.35	0.33	0.17	0.40	0.40
	# cat nodes	1	14	10	23	44	22	28	1	1	1
	# leaves	**4**	26	21	40	118	63	85	7	13	13
Car eval. (1382, 6, 4) $\text{card}(X_{\text{cat}}) \leq 4$	E_{train}	2.03	21.06	5.43	3.90	0.00	1.45	1.52	2.46	2.74	0.58
	E_{test}	2.89	20.52	4.62	6.94	3.18	2.89	2.31	2.60	2.60	**1.16**
	Δ	9	16	11	6	14	12	10	10	9	12
	Cat prop	1.00	1.00	1.00	0.48	1.00	1.00	1.00	1.00	1.00	1.00
	# cat nodes	26	113	38	36	100	59	49	10	34	59
	# leaves	**27**	114	39	76	101	60	50	45	35	60
Telco (5625, 19, 2) $\text{card}(X_{\text{cat}}) \leq 4$	E_{train}	20.07	19.91	19.70	18.44	16.73	17.64	14.99	18.88	19.82	18.86
	E_{test}	21.46	22.25	20.26	19.83	20.82	20.75	20.04	**18.91**	19.83	19.76
	Δ	6	12	5	10	11	9	16	8	6	9
	Cat prop	0.50	0.68	0.58	0.52	0.42	0.41	0.20	0.25	0.30	0.29
	# cat nodes	7	148	18	26	26	12	18	4	3	4
	# leaves	15	218	32	51	63	30	90	17	**11**	15
Electricity (34626, 8, 2) $\text{card}(X_{\text{cat}}) \leq 7$	E_{train}	6.30	15.15	7.53	4.61	4.05	2.64	5.83	8.57	6.72	6.20
	E_{test}	12.84	17.10	12.86	10.65	12.40	11.85	12.34	11.79	11.56	**10.63**
	Δ	22	16	18	28	27	28	25	21	24	25
	Cat prop	0.13	0.31	0.14	0.83	0.12	0.14	0.12	0.07	0.18	0.19
	# cat nodes	161	1040	212	1213	178	258	113	35	125	130
	# leaves	1257	3329	1569	1471	1466	1841	935	**478**	699	688
Bank Marketing (38429, 16, 2) $\text{card}(X_{\text{cat}}) \leq 12$	E_{train}	8.92	9.62	9.55	8.48	8.43	8.86	8.11	9.35	9.17	9.04
	E_{test}	9.67	9.48	9.38	9.22	9.19	9.20	9.10	8.83	8.88	**8.51**
	Δ	15	12	5	11	17	10	14	6	9	9
	Cat prop	0.41	0.53	0.57	0.52	0.33	0.33	0.16	0.54	0.43	0.47
	# cat nodes	36	127	17	72	54	14	41	7	6	9
	# leaves	89	239	31	139	164	44	86	**14**	15	20
Chess KRK (28056, 6, 18) $\text{card}(X_{\text{cat}}) \leq 8$	E_{train}	18.67	75.53	26.56	24.51	5.57	20.96	1.61	66.17	8.51	8.06
	E_{test}	28.69	73.91	35.92	38.58	21.40	27.23	20.56	65.88	**20.46**	21.58
	Δ	15	10	18	6	28	19	21	6	28	28
	Cat prop	1.00	1.00	1.00	0.30	1.00	1.00	1.00	1.00	1.00	1.00
	# cat nodes	385	34	1609	898	4407	2220	5671	37	1946	2831
	# leaves	386	**35**	1610	3027	4408	2221	5672	38	1947	2832
Jungle Chess (44819, 6, 3) $\text{card}(X_{\text{cat}}) \leq 9$	E_{train}	8.80	13.12	11.44	11.89	10.71	12.59	9.65	26.47	3.22	8.32
	E_{test}	13.26	16.21	16.73	17.22	14.44	15.10	13.49	27.42	10.22	**10.17**
	Δ	16	13	18	6	27	17	19	6	22	19
	Cat prop	1.00	1.00	1.00	0.40	1.00	1.00	1.00	1.00	1.00	1.00
	# cat nodes	732	3169	1255	421	1088	807	737	12	602	512
	# leaves	733	3170	1256	1043	1089	708	738	**13**	603	513

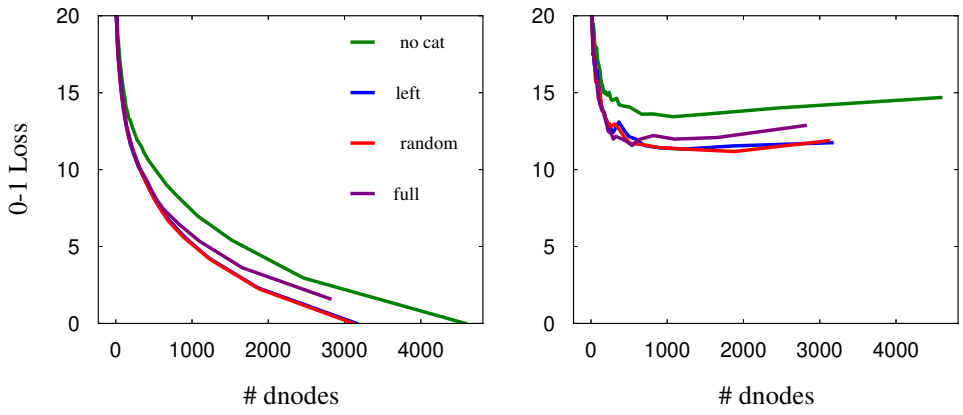

Figure 5: 0-1 Loss vs number of decision nodes on Electricity dataset. We report train (on the left) and test (on the right) error. Each line is produced by different way of handling missing categories. Green line shows TAO tree trained without proposed categorical decision nodes.

input binary axis-aligned tree T with given structure and parameters $\boldsymbol{\Theta}$ at the nodes $\mathcal{N} = \mathcal{D} \cup \mathcal{L}$
repeat
  for $i \in \mathcal{N}$
    $\mathcal{R}_i \leftarrow$ reduced set of node i
  for $i \in \mathcal{N}$ in reverse BFS order
    if $\mathcal{R}_i \neq \varnothing$ **then**
      if $i \in \mathcal{L}$ **then** leaf
        $c_i \leftarrow$ majority class in $\mathcal{R}_i$
      else decision node
        (w_i, b_i) or $F_i \leftarrow$ solution of node i's RP
until $E(\boldsymbol{\Theta})$ does not strictly decrease
postprocess T: remove redundant nodes
return T

parfor $x \in \mathcal{C}_d$ category
  $S_{\min} \leftarrow \infty$
  parfor $k \in$ children of i
    $S \leftarrow \sum_{\substack{n \in \mathcal{R}_i \\ x_{nd} = x}} l_{i,n}(k)$ see eq. (4)
    if $S < S_{\min}$ **then**
      $S_{\min} \leftarrow S$
      $k_{\min} \leftarrow k$
  $F_i(x) = k_{\min}$ best child
return F_i

Figure 6: Pseudocode. *Left*: TAO algorithm with axis-aligned decision nodes and constant-label leaves. *Right*: all-subsets split optimization in the RP of decision node i if feature d is categorical (version suitable for a parallel implementation).

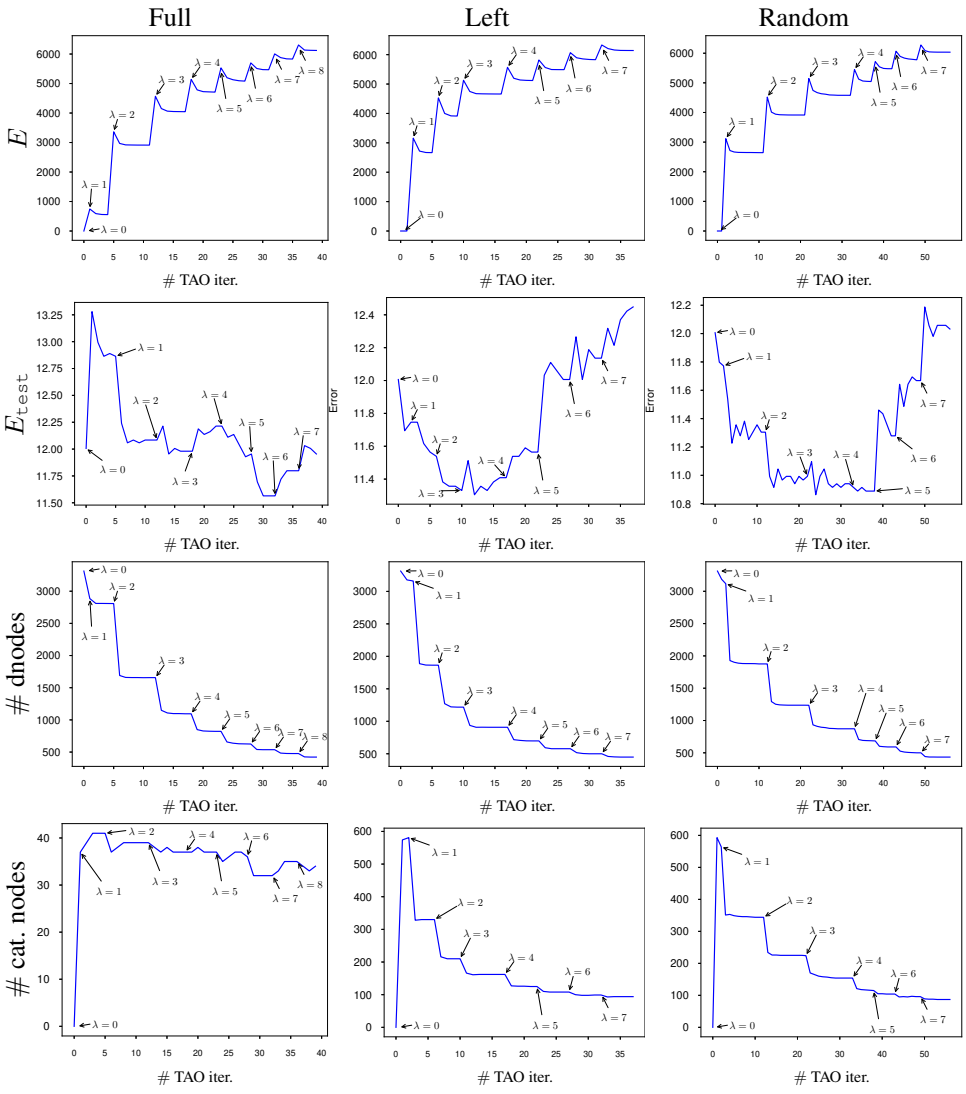

Figure 7: Learning curve of our proposed approach. From top to bottom: objective function, test error, number of decision nodes, and number of decision nodes with categorical splits. Each column is produced by different way of handling missing categories in the decision node.

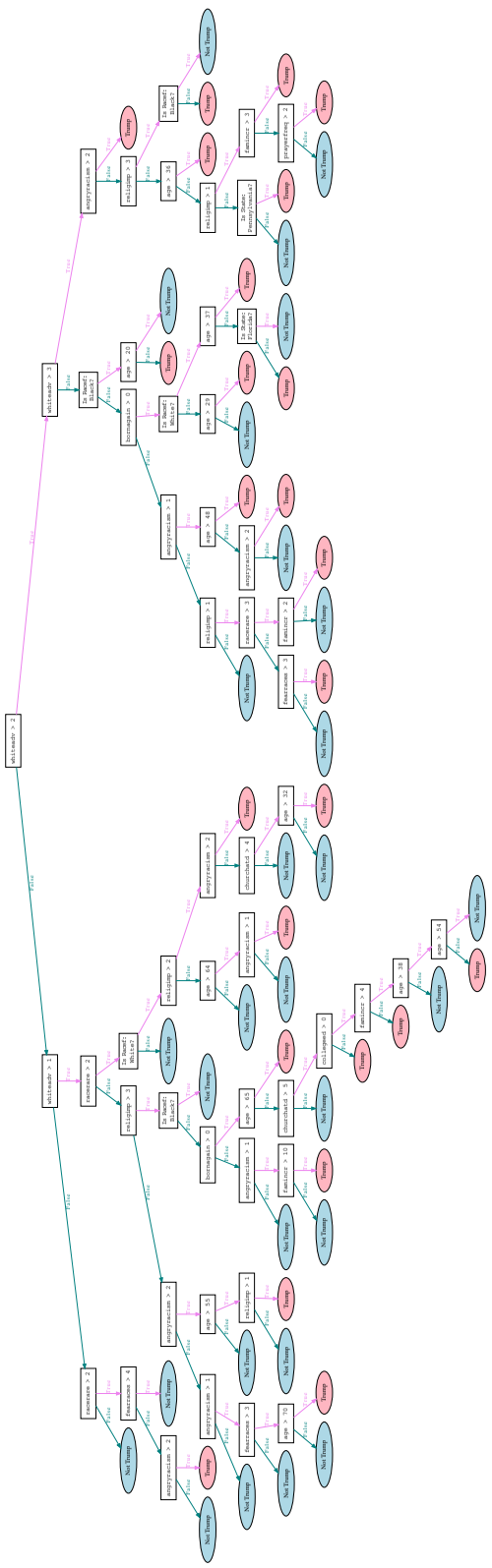

Figure 8: Same as Figure 2, but TAO tree without proposed categorical split in decision nodes ($\Delta = 12$, 54 leaves).

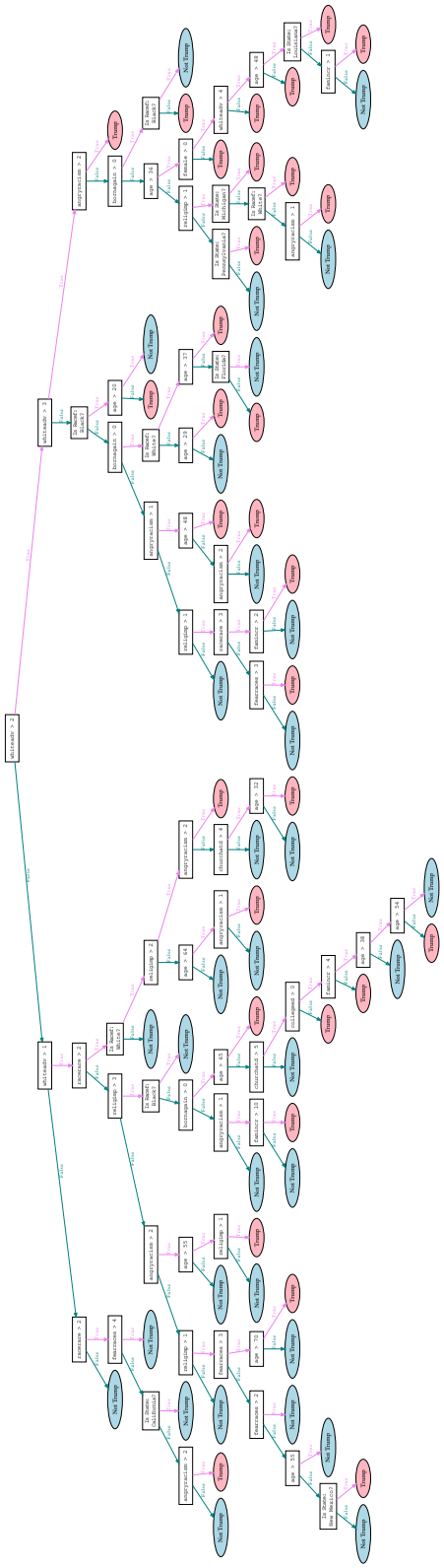

Figure 9: Same as Figure 2, but CART tree ($\Delta = 12$, 64 leaves).