DTRBENCH: A BENCHMARKING FRAMEWORK FOR DECISION TREE REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Due to their efficiency, interpretability, and ensemble capabilities, decision trees are foundational to machine learning, with recent benchmark studies showing that tree ensembles remain competitive with deep learning on tabular data. Yet, systematically comparing individual trees remains difficult due to structural and semantic variability. While decision tree representations enable pairwise comparisons, existing representation methods have only been developed in isolation on narrow tasks. Consequently, their utility and broader applicability cannot be judged. To address this gap, we present DTRBench, a modular benchmarking framework for evaluating decision tree representations across three axes: task-independent sensitivity to controlled perturbations, utility for downstream subforest selection, and computational efficiency. Across 25 datasets, we benchmark five representations from the literature, spanning vector-, graph-, and neural-based paradigms, alongside an illustrative new hybrid vector representation we call "Tree Descriptor". The results show that local perturbation sensitivity does not translate to downstream subforest selection utility, highlighting the need for measuring them as decoupled properties. Furthermore, DTRBench reveals that subforest selection based on Tree Descriptor representations requires up to 73% fewer trees than standard heuristics to recover 99% of the full random forest's predictive performance, thereby substantially accelerating inference. Ultimately, DTRBench provides an extensible, standardized, and open-source foundation for evaluating and selecting decision tree representations, revealing distinct strengths and limitations across representations.

1 INTRODUCTION

Machine learning has become pervasive across domains, including regulated and safety-critical ones where the need for interpretability limits the use of the most recent and complex models. Decision trees (DTs) address this need by inferring an interpretable decision process directly from data (Quinlan, 1986). DTs are fast to train and expose feature importance (FI) and decision criteria directly, but individual trees are notably sensitive to overfitting and small changes in training data, limiting their robustness (Gupta et al., 2017). Tree ensembles such as random forests (RFs) (Breiman, 2001) address this instability, and remain competitive with, or superior to, more complex models across diverse applications, including tabular data (Grinsztajn et al., 2022; McElfresh et al., 2023) and, more recently, temporal health record data (Rahimi et al., 2025).

Despite this widespread use, comparing individual DTs and quantifying similarity between them is inherently non-trivial: trees of the same depth and size can differ in the features and thresholds they use, structurally different trees can encode similar decision logic, and structurally similar trees can encode entirely different decision logic. Without a systematic way to compare DTs, analyzing collections of DTs thus becomes difficult, even though doing so is essential for tasks such as measuring ensemble diversity, detecting functional redundancy, monitoring model and data drift across retraining cycles, clustering large tree collections into functionally coherent groups, and compressing DT ensembles via subforest selection to reduce computational cost and improve interpretability (Bayir et al., 2022; Lange et al., 2025).

DT representations have emerged as a promising approach to this comparison problem. Rather than comparing two trees directly, a representation first abstracts a tree's characteristics into a well-

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.

defined mathematical structure (e. g., a vector, graph, or continuous function) from which a similarity between trees can be computed, enabling systematic and reusable pairwise comparisons. Representations vary in which characteristics they emphasize: some prioritize a DT's structural form (node-level attributes such as split features and thresholds, or higher-level properties such as topology, depth, and root-to-leaf paths), while others prioritize its behavior or function (the predictions it produces, the resulting decision regions, or the importance of different features), and many combine elements of both to different degrees. Importantly, neither view alone is generally sufficient: structurally different trees may produce identical or near-identical predictions, while structurally similar trees may encode markedly different decision logic, so a representation emphasizing only one aspect risks missing similarities or differences that the other would reveal. Existing representations, however, have each been developed and evaluated in isolation on a single narrow task, e. g., summarizing feature contributions in unsupervised RFs (Sirocchi et al., 2025), subforest selection (Bayir et al., 2022), DT reconstruction (Spinnato et al., 2025), or data-stream mining on mobile devices (Kargupta & Park, 2004). Hence, it remains unclear how faithfully different DT representations capture functional DT behavior more broadly and how well they transfer to downstream tasks other than the ones they were originally evaluated on.

To close this gap, this work introduces DTRBench (`https://anonymous.4open.science/r/DTRBench-anonymized`), an extensible, standardized, open-source framework for systematically evaluating DT representations along different axes. The main contributions are the following:

- DTRBench integrates five representations from the literature spanning vector-based, graph-based, topological, and neural-network paradigms, alongside a novel representation that addresses limitations of existing approaches and showcases DTRBench's extensibility.
- DTRBench's perturbation benchmark quantifies each representation's sensitivity to changes in predictive performance and FI.
- DTRBench's subforest selection benchmark tests the downstream utility of the DT representations for compressing RFs while maintaining predictive performance and FI stability.
- DTRBench's resource benchmark measures runtime and memory requirements to evaluate the computational efficiency of each representation.

2 Related Work

Comparing DTs requires a notion of similarity. Broadly, existing work approaches this problem in two ways: *representational* methods first map a tree into an intermediate representation (e.g., a vector, graph, or continuous function) from which similarity can be derived, while *algorithmic* methods compute a similarity score directly from a pair of trees. A third line of work applies these similarity notions to downstream tasks such as subforest selection. This section reviews these three strands of literature and highlights the lack of a common framework for evaluating DT representations.

Tree representations. Previous approaches have explored specific mathematical and functional representations of DTs, such as Fourier spectra (Kargupta & Park, 2004), feature interaction graphs (Sirocchi et al., 2025), topological encodings (Bayir et al., 2022), and neural network parameters (Spinnato et al., 2025). However, these representations are typically evaluated in isolation on narrow tasks. For example, Sirocchi et al. (2025) use a feature interaction graph to summarize feature use in an unsupervised RF, Kargupta & Park (2004) focus on DT compression and transfer in resource-limited settings, Bayir et al. (2022) use a topological representation for subforest selection, and Spinnato et al. (2025) represent DTs using neural network parameters to enable tree reconstruction. Consequently, both the representations and their evaluation are driven by the requirements of their respective applications rather than being systematically compared across representations and downstream tasks. This makes it difficult to determine whether a representation is broadly useful for characterizing DTs or primarily suited to the task for which it was originally developed.

Algorithmic similarity criteria. Alternative work has explored criteria for computing DT similarity directly, broadly grouped by whether they capture a tree's topology (syntactic), its function (semantic), or a combination of both. Kuncheva & Whitaker (2003) review ten such similarity metrics applicable to classifiers in general, including measures based on Cohen's kappa. For models that induce partitions, Fowlkes & Mallows (1983) propose a purely semantic criterion based

on how consistently pairs of objects are grouped together while Ntoutsi et al. (2008) compare the leaf-induced partitions by overlaying two trees' partitions and weighting prediction agreement by a chosen attribute distribution. Briand et al. (2009) propose instead a purely syntactic counterpart that matches trees node-by-node on split variables and thresholds. Other methods combine both notions: Miglio & Soffritti (2004) combine a modified Fowlkes–Mallows-style leaf-partition similarity with a count of discrepant root-to-leaf paths weighted by path length from Shannon & Banks (1999), while Weinberg & Last (2019) combine tree edit distance with prediction similarity to identify a single representative tree. Since selecting a single central tree does not directly extend to selecting a representative subforest, where multiple trees must jointly preserve diversity and predictive performance, subsequent work adapts similarity-based selection to subsets of trees: Fawagreh et al. (2015) introduce CLUB-DRF, which clusters trees by prediction agreement and extracts one representative from each cluster; DTreeSim (Bakirli & Birant, 2017) linearizes trees into sequences and applies sequential pattern mining with k-nearest neighbors for pairwise comparison. While this method produces an intermediate sequence-based representation, this is not inherently easier to manipulate or compare than the original tree, as it remains tied to a dedicated sequence-mining pipeline rather than serving as a reusable object for downstream tasks.

Subforest selection. A parallel line of work applies tree similarity to subforest selection, i. e., identifying a compact, high-performing subset of trees from a larger ensemble. These approaches (reviewed by Manzali & Elfar (2023)) can be grouped by whether they select trees based on predictive performance, similarity, or a combination of both. Performance-driven approaches rank and select trees by predictive quality. Khan et al. (2021) and Dheenadayalan et al. (2016) rank trees by out-of-bag (OOB) accuracy and OOB Matthews correlation coefficient (MCC) (Matthews, 1975), respectively, and then select the top-ranked trees. Bader-El-Den & Gaber (2012) instead evaluate candidate subforests on a held-out validation set to guide a genetic algorithm. Similarity-driven approaches select diverse or discard redundant trees, e. g., based on prediction agreement quantified via Cohen's kappa between the trees' predictions (Margineantu & Dietterich, 1997). Zhang & Wang (2009) compare performance- and similarity-based tree removal, where the former removes the tree that causes the smallest decrease in RF performance, and the latter removes the tree with the highest average correlation between its predictions and those of all the other trees. Finally, Adnan & Islam (2016) combine both criteria by balancing individual tree quality with diversity when constructing the initial population for their genetic algorithm.

Addressed research gap. Existing work provides individual representations, similarity measures, and task-specific evaluation strategies, but no unified framework exists for systematically evaluating and comparing DT representations under consistent criteria. With DTRBench, we bridge this gap.

3 The DTRBench Framework

DTRBench (Figure 1) is a modular benchmarking framework for developing and evaluating DT representations. It integrates 25 evaluation datasets (Section 3.1) and six DT representations (Section 3.2), which are tested along three complementary axes: task-independent sensitivity to structural and functional change (perturbation benchmark, Section 3.3), task-dependent utility for subforest selection (subforest benchmark, Section 3.4), and computational efficiency (resource benchmark, Section 3.5). DTRBench's modular architecture allows users to seamlessly integrate custom representations, datasets, perturbations, and selection strategies without modifying the core codebase (Appendix A.1.4), and the framework automatically generates standardized figures and summary tables for all benchmarks to support comprehensive comparative analysis. The code is publicly available on GitHub (`https://anonymous.4open.science/r/DTRBench-anonymized`).

3.1 Datasets

DTRBench includes 25 classification datasets from the UCI Machine Learning Repository (Kelly et al., 2026), selected to represent settings where ensembles have a substantial predictive advantage over individual DTs. The filtering and ranking procedure used for this selection, along with an overview of all datasets, is provided in Appendix A.2.1 (Table 4). These built-in datasets can be used for standardized evaluation or replaced with user-defined datasets via the framework's dataset registry (Appendix A.1.4).

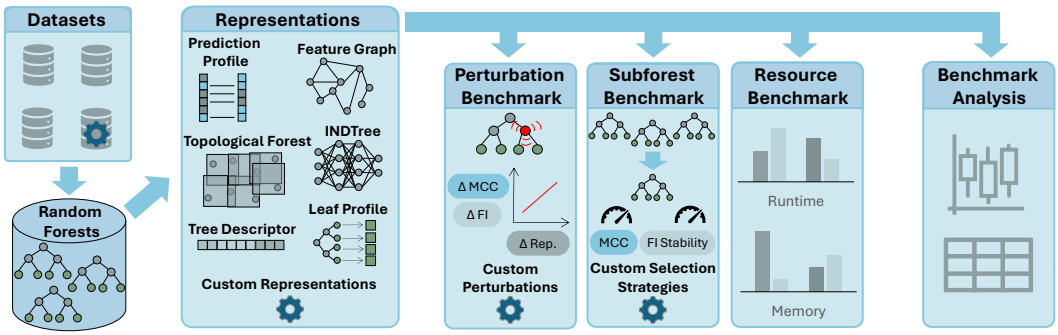


Figure 1: Overview of the DTRBench benchmarking framework. Input datasets are used to train RFs, whose trees are transformed into DT representations and evaluated along three axes: a perturbation benchmark correlates representation distances with differences in predictive performance and FI; a subforest benchmark assesses their utility for downstream subforest selection; and a resource benchmark measures runtime and memory efficiency. Results are then automatically summarized and visualized. Gear icons mark user-customizable components (e. g., custom datasets, DT representations, perturbations, and selection strategies).

Preprocessing restricts each dataset to numerical features only, dropping categorical ones. This is because scikit-learn's `DecisionTreeClassifier` objects, as well as the INDTree and Tree Descriptor representations (see Section 3.2), rely on threshold-based splits and thus cannot handle categorical features. Encoding categorical variables is deliberately avoided as traditional tree algorithms are inherently biased toward features with more candidate split points because having more potential splits increases the chance of finding an artificially high impurity reduction. One-hot encoding categorical features into binary indicators fragments a single underlying variable across multiple dummy features; this artificially alters candidate split counts, distorts tree split selection, and dilutes or inflates accumulated FI scores (Strobl et al., 2007), which would confound FI throughout DTRBench, both within representations and as an evaluation measure.

3.2 Decision Tree Representations

To enable systematic evaluation and comparison, DTRBench includes a diverse suite of DT representations spanning vector-based, distribution-based, graph-based, topological, and continuous-functional paradigms, as well as syntactic, semantic, and hybrid approaches. The suite comprises representations adopted from the literature, representations inspired by prior work, and a novel representation introduced to address limitations of existing approaches and to showcase the framework's extensibility. Details of each representation and their exact configuration are provided in Appendix A.1.2 and A.2.2.

Feature Graph (FG). FG (Sirocchi et al., 2025) maps feature interactions along decision paths into a sample-weighted adjacency matrix, with leaf transitions connecting to a single terminal node. It captures structure and frequency of feature interactions but not directly split thresholds or leaf-level predictions. DTRBench compares FG representations via a correlation-adjusted Frobenius norm that de-emphasizes differences between redundant features (Srivastava & Rao, 2016).

Topological Forest (TF). TF (Bayir et al., 2022) first constructs unweighted directed feature interaction graphs and summarizes each with local topological metrics. The resulting feature vectors are then projected into 2D via t-SNE and clustered with an overlapping-grid k-means to form a global cluster graph. Similarity is derived from shortest-path length between cluster nodes, capturing macro-level connectivity while overlooking thresholds and leaves.

INDTree (ID). ID (Spinnato et al., 2025) models each DT with a neural network that maps structural coordinates to split features, thresholds, and class distributions, using model-agnostic meta-learning (MAML) (Finn et al., 2017). DTRBench compares representations via aggregated Euclidean distance between fine-tuned predictions on a canonical coordinate grid.

Leaf Profile (LP). LP, inspired by Ntoutsi et al. (2008), represents a DT as a discrete distribution of leaf masses and class distributions, capturing how samples are distributed across leaves. DTRBench compares LP representations using the 1-Wasserstein distance (Vaserstein, 1969), with transport costs based on Euclidean distances between leaf class distributions.

Prediction Profile (PP). PP represents a DT as its prediction vector on a reference dataset, following prediction-based criteria used in subforest selection. Similarity is measured via Cohen's kappa between predicted labels over the training set. This approach is used in several works (e. g., by Margineantu & Dietterich (1997)) and here serves as a simple baseline representation.

Tree Descriptor (TD). TD is a novel three-part vector representation, incorporating split thresholds absent from other representations while avoiding the cost and opacity of neural approaches. The first part describes structural properties, the second part captures functionality via normalized FI and feature usage frequency, and the third one encodes decision boundaries via min-max normalized average split thresholds. The three sub-vectors are compared separately via cosine similarity on structural properties, averaged soft cosine similarity (Sidorov et al., 2014) on FI and feature usage, and $1 -$ MAE (mean absolute error) across thresholds of common features, respectively. The three resulting similarities are then combined via weighted averaging.

3.3 Perturbation Benchmark

The perturbation benchmark tests DT representations under the assumption that a meaningful representation should produce distances that correlate with the functional changes induced by a perturbation. Evaluated across all 25 datasets using 5-fold stratified cross-validation (CV) with DTs restricted to a maximum depth of 10 (a fixed depth chosen for computational efficiency while retaining tree complexity), DTRBench trains RFs of 50 trees, to balance sufficient samples for stable correlation estimates against computational cost. DTRBench applies five perturbation operations (details in Appendix A.1.1): shifting split thresholds (*change threshold*), altering split features with Gini-optimal threshold recalculation (*change feature*), swapping features and thresholds between internal nodes (*swap nodes*), pruning sibling leaf pairs (*remove nodes*), and expanding leaves into internal nodes with optimal splits (*add nodes*). Each of the 50 trees is first reduced to a *base* tree via a single application of *remove nodes* at intensity 0.5, providing a consistent, moderately simplified reference point; each of the five perturbations is then applied once to this base tree at each of five intensities, corresponding to the fraction of nodes affected, $\lambda \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$.

For each pair $(t, \tilde{t})$ of base and perturbed trees and each DT representation R, a representation distance is computed as $1 - s_R(t, \tilde{t})$, where s_R is R's own similarity measure (Appendix A.1.2). The resulting representation distance is correlated against two reference measures of functional change: the predictive performance difference, quantified via OOB MCC, and the shift in FI, quantified via the Manhattan distance between normalized FI vectors. For each R, DTRBench reports the Pearson correlations r_{MCC} and r_{FI} between the representation distance and the two reference measures, aggregated across CV folds and datasets via Fisher z-transformation (further details in Appendix A.2.3). For a sensitive DT representation, both r_{MCC} and r_{FI} should be close to 1.

3.4 Subforest Benchmark

The subforest benchmark evaluates how effectively different representations and representation-derived distances support downstream subforest selection, i. e., choosing a small, representative subset of trees from a full RF that preserves predictive performance and FI consistency while substantially reducing ensemble size. Evaluated using 5-fold CV with DTs restricted to a maximum depth of 10, DTRBench trains full RFs of 1000 trees and computes pairwise distance matrices for each representation. We evaluate 20 logarithmically spaced subforest sizes from 2 to 1000 trees, concentrating resolution on smaller, more informative subforests. Eight selection strategies (Appendix A.1.3) use the distance matrix to construct subforests: k-medoids selection (KM-S) (Kaufman & Rousseeuw, 1987) and agglomerative clustering (AG-S) (Ward, 1963) partition trees into k distance-based groups and select central medoids; their performance-guided variants (KMP-S and AGP-S) select the tree with the highest OOB MCC per cluster; density selection (DE-S) iteratively samples trees via Gaussian density kernels over pairwise distances; and multi-objective optimization strategies jointly maximize diversity (average pairwise distance) and predictive performance (OOB

MCC) via a greedy algorithm (GR-S), simulated annealing (SA-S) (van Laarhoven & Aarts, 1987), or genetic programming (GE-S) (Holland, 1992).

To evaluate these methods, we first selected 5 of the 25 datasets as validation datasets to test all selection strategies, and then used the overall best-performing strategy for the full benchmark against baseline selection methods on the remaining 20 test datasets (Appendix A.2.4). Here, representation-based subforests are compared against three baselines: random sampling (RDM), for which 10 random subforests with different seeds are evaluated and then averaged, and selecting top DTs ranked by OOB accuracy (TOA) or OOB MCC (TOM). The resulting subforests are evaluated with four metrics: (i) *relative test MCC* (subforest MCC on the test fold divided by full RF MCC on the same fold, averaged across folds), (ii) *prediction agreement* (fraction of samples where subforest and full RF predict the same class, averaged across folds), (iii) *relative FI stability* (Kendall's coefficient of concordance W (Kendall & Smith, 1939) across folds divided by Kendall's W for the full RF), and (iv) *FI agreement* (Spearman correlation between subforest and full RF FI vectors on the same fold, averaged across folds). Relative test MCC and relative FI stability scores close to or above 1 indicate that subforests match or surpass full RFs in predictive performance and FI stability; high prediction agreement and FI agreement indicate that subforest behavior is faithful to the full RF.

3.5 Resource Benchmark

Lastly, the resource benchmark measures the computational efficiency of DT representations across different RF sizes in terms of runtime and memory. Unlike the other benchmarks, it was conducted on a single dataset (the Connectionist Bench dataset by Sejnowski & Gorman (1988)) using 5-fold CV with a maximum DT depth of 10, as it targets how resource cost scales with ensemble size rather than dataset size. Runtime and memory measurements are reported as averages across the five CV folds. RF sizes were varied over a range from 5 to 100 trees, and experiments were run single-threaded to obtain comparable per-operation timings (further experimental details are in Appendix A.2.5). The framework decouples resource consumption into two phases: representation construction, the cost of transforming a DT into its representation, and similarity computation, the cost of computing the similarity between two representations. During benchmark execution across RFs of varying sizes, DTRBench measures execution runtime alongside three memory metrics: peak system RAM, peak GPU VRAM, and the serialized memory requirement of each representation instance. Each reported value is the average cost of a single operation — one tree's representation construction, or one pair's similarity computation — rather than the total cost summed across the forest. An empirical scaling exponent α characterizes how these per-operation costs change with RF size, with $\alpha \approx 0$ (constant per-operation cost) corresponding to linear total cost when the operation is performed once per tree, identifying which representations remain lightweight at scale.

4 Results

4.1 Results of the Perturbation Benchmark

In the perturbation benchmark, we investigate whether changes in representation track the functional consequences of a controlled perturbation, namely its effect on predictive performance and FI. Table 1 lists the correlation between representation distance and (i) the resulting difference in MCC (r_{MCC}) and (ii) the resulting shift in FI (r_{FI}), for each perturbation, with the underlying relationship exemplified for LP in Figure 4 (Appendix A.3.1). LP achieves the highest correlations for most perturbations across both metrics, suggesting that it is generally most sensitive to DT differences that affect predictive performance and FI; it leads r_{MCC} for *change feature* and *swap nodes*, and r_{FI} for *change threshold*, *remove nodes*, and *add nodes*. TD leads r_{MCC} for *remove nodes* and r_{FI} for *change feature* and *swap nodes*. PP is also highly sensitive overall and leading r_{MCC} for *change threshold*. In contrast, TF and ID show near-zero or low correlations across most perturbations, and are entirely insensitive to *change threshold*, consistent with the exact zero distance observed across all intensities for this perturbation in Figure 3 (Appendix A.3.1), which further confirms that distance increases with perturbation intensity for most representation-perturbation pairs. r_{MCC} values for *add nodes* are omitted, since adding new leaves with optimal thresholds does not decrease predictive performance and thus cannot correlate with it. Figure 5 (Appendix A.3.1) confirms via critical difference diagrams that the best-ranked representation is frequently, though not universally, significantly better than the majority of alternatives.

Table 1: Average Pearson correlations between representation distance and (i) difference in MCC (r_{MCC}) and (ii) shift in FI (r_{FI}) for different perturbation operations. **Bold**: highest correlation per perturbation-metric pair; –: not applicable.

	r_{MCC}						r_{FI}					
Perturbation	FG	TF	ID	LP	PP	TD	FG	TF	ID	LP	PP	TD
Change threshold	0.56	–	–	0.73	**0.78**	0.62	0.77	–	–	**0.80**	0.77	0.66
Change feature	0.31	0.08	0.25	**0.56**	0.54	0.38	0.69	0.17	0.51	0.78	0.78	**0.82**
Swap nodes	0.33	0.08	0.27	0.61	**0.61**	0.39	0.65	0.16	0.49	0.79	0.79	**0.86**
Remove nodes	0.61	0.44	0.70	0.80	0.84	**0.90**	0.83	0.61	0.79	**0.96**	0.94	0.81
Add nodes	–	–	–	–	–	–	0.33	0.25	0.24	**0.84**	0.64	0.42

4.2 Results of the Subforest Benchmark

In the subforest benchmark, the representations and similarities defined on them are evaluated by their ability to support the selection of a small subforest that is both high-quality and representative of the full RF's behavior, achieving high predictive performance and stable FI while maintaining strong agreement with the full RF's predictions and FI. As established in Appendix A.3.2, SA-S is used as the selection strategy for representation-based methods throughout, as it achieves the best performance and prediction agreement with the full forest on validation datasets and remains competitive on the held-out test datasets (Figure 6, Appendix A.3.2). Table 2 reports relative test MCC and relative FI stability, alongside prediction agreement and FI agreement, aggregated across three subforest-size regimes. In the tiny regime ($k \in [2, 10)$), the performance-based baselines TOM and TOA achieve the highest relative test MCC, outperforming all representations. This advantage narrows in the small regime ($k \in [10, 50)$), where TOA and TF are tied (0.993), and reverses in the medium regime ($k \in [50, 200)$), where every representation outperforms every baseline and FG achieves the highest score overall (1.014). Prediction agreement follows a similar trend: a baseline and a representation jointly achieve the highest agreement in both the tiny regime (TOA and TF, 0.869) and the small regime (LP and TOA, 0.933), while all representations outperform all baselines in the medium regime. FI stability shows a related but distinct size-dependent pattern. Representations take the top two positions in the tiny regime (LP, 1.053; TF, 1.020) and the small regime (TF, 1.106; TD, 1.097), whereas the heuristic baselines outperform all representations in the medium regime (TOM, 1.126; TOA, 1.113). FI agreement consistently favors representation-based selection, with a representation achieving the highest agreement in every regime (TF in the tiny and small regimes, PP in the medium regime). Overall, representations match or exceed the baselines in relative test MCC, prediction agreement, and FI agreement at larger subforest sizes, and in FI stability at smaller subforest sizes. However, no representation consistently dominates across all four metrics and all three size regimes, suggesting that the advantage of representation-based selection depends on both the evaluation metric and the degree of compression. Notably, TF, which performed worst in the perturbation benchmark (Section 4.1), performs best in subforest selection. TF ranks in the top two across 7 of the 12 metric-regime configurations in Table 2, compared with 3 for LP, which performed best in the perturbation benchmark. This contrast reinforces that perturbation sensitivity and subforest selection utility are two distinct properties of a representation, and that a representation's advantage on one axis need not transfer to the other, or hold uniformly across compression levels.

Beyond average performance at a fixed subforest size, a complementary and arguably more practical measure of subforest selection quality is compression efficiency: the minimum number of trees required to achieve a target level of predictive performance. To assess how effectively each representation compresses a RF while preserving performance (relative test MCC), we examine subforest selection across thresholds ranging from 90% to 100% of the full RF's performance. The complete sweep is shown in Figure 7 (Appendix A.3.2), where TD achieves the best overall compression efficiency. To illustrate this advantage at a practically relevant operating point, we focus on a threshold of 99% relative test MCC, with Figure 2a showing the minimum subforest size required by each representation and baseline, averaged across the 20 test datasets. TD requires the fewest trees on average (46) and is the only method to meet the threshold on every dataset. In contrast, all other methods fail on at least one dataset, most notably Statlog (Australian), where TD reaches

Table 2: Evaluation of selected subforests across four dimensions for representation-based and baseline selection methods. Scores are mean-aggregated across 20 test datasets and five subforest sizes per regime (tiny: $k \in \{2, 3, 4, 5, 7\}$; small: $k \in \{10, 14, 20, 27, 38\}$; medium: $k \in \{53, 73, 101, 141, 195\}$). SA-S is used for all representation-based methods (Appendix A.3.2). **Bold**: best value per column; underlined: second-best.

	Relative test MCC			Prediction agreement			Relative FI stability			FI agreement		
Method	Tiny	Small	Medium	Tiny	Small	Medium	Tiny	Small	Medium	Tiny	Small	Medium
Baselines												
TOM	**0.921**	0.991	1.001	0.867	0.930	0.962	0.957	1.060	**1.126**	0.809	0.915	0.963
TOA	0.916	**0.993**	1.004	**0.869**	0.933	0.965	1.002	1.029	1.113	0.807	0.911	0.963
RDM	0.856	0.985	1.006	0.851	0.930	0.971	0.761	0.927	1.001	0.801	0.918	0.974
Representations												
FG	0.912	0.986	**1.014**	0.865	0.931	0.971	0.898	1.037	1.035	0.810	0.908	0.973
TF	0.905	0.993	1.008	0.869	0.931	0.972	1.020	**1.106**	1.053	**0.816**	**0.919**	0.974
ID	0.886	0.975	1.011	0.860	0.931	0.971	0.932	1.005	1.045	0.804	0.914	0.974
LP	0.878	0.989	1.009	0.864	**0.933**	0.971	**1.053**	1.079	1.078	0.790	0.913	0.974
PP	0.818	0.950	1.013	0.839	0.926	**0.972**	0.901	1.040	1.045	0.791	0.913	**0.975**
TD	0.855	0.987	1.011	0.855	0.931	0.972	0.909	1.097	1.062	0.773	0.911	0.972

the target with only 53 trees while every other representation and baseline fails even at the largest subforest size considered; TOM additionally fails on one other dataset. FG and LP are the next most efficient methods, requiring 71 and 72 trees on average, respectively. TF, ID, and PP follow (99, 105, and 115), while the performance-based baselines TOA and TOM require substantially larger subforests (132 and 173 trees). Figure 2b shows the per-dataset distribution underlying these averages, with TD reaching the 99% performance threshold using fewer than 40% of the trees across all datasets. Because RF inference cost scales linearly with the number of evaluated trees, this reduction translates directly into lower inference time, with TD requiring the lowest inference time across all methods (Figure 8, Appendix A.3.2). These findings further highlight that different evaluation criteria favor different representations; FG and TF achieve the highest relative test MCC at fixed subforest sizes (Table 2), whereas TD is most effective in terms of compression efficiency. Thus, even within a single benchmark, the preferred representation depends on the evaluation criterion.

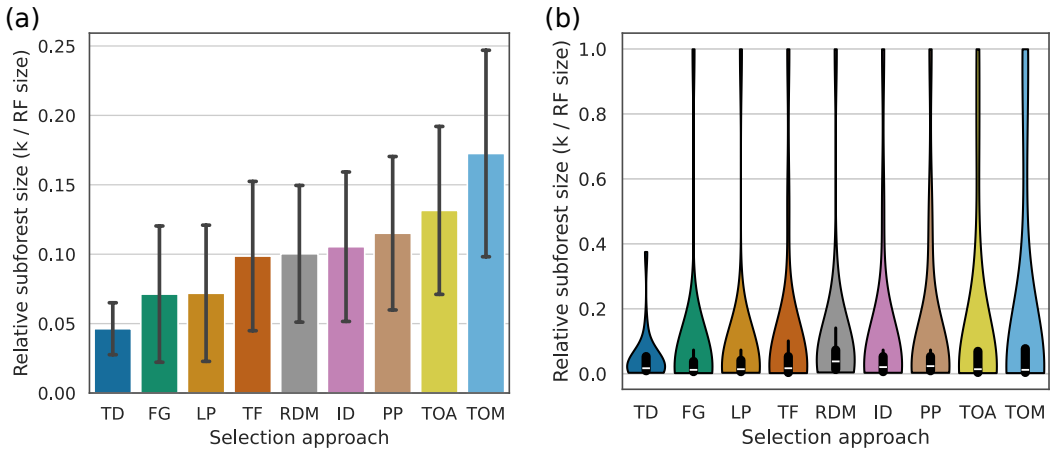


Figure 2: Number of trees each representation's / baseline's selected subforest needs to recover 99% of relative test MCC, expressed as a fraction of the full RF size (1000 trees), across the 20 held-out test datasets. **(a)** Mean ± standard error across datasets, representations sorted by increasing mean. **(b)** Full per-dataset distribution.

Table 3: Efficiency and empirical scaling across representations, averaged over folds. Each entry gives the absolute cost of one operation at 50 trees, followed by the empirical scaling exponent α (rounded to one decimal). The scaling exponent α characterizes how the cost changes with the number of trees, with $\alpha \approx 0$, $\alpha \approx 1$, and $\alpha \approx 2$ corresponding to approximately constant, linear, and quadratic scaling, respectively. "Rep." and "Sim." denote representation and similarity computation.

	Runtime (ms)		Peak RAM (KB)		Peak VRAM (MB)		Rep. Size (KB)
Representation	Rep.	Sim.	Rep.	Sim.	Rep.	Sim.	
FG	1.51 (0)	0.13 (0)	130.02 (0)	59.58 (0)	–	–	29.22 (0)
TF	15.18 (0)	3.87 (−1.8)	36.48 (0)	285.65 (0.7)	–	–	0.34 (0)
ID	3.3×10^3 (0.3)	0.32 (0)	77×10^3 (1.1)	1.07 (0)	98.16 (0.7)	–	25.00 (0)
LP	0.87 (0)	0.95 (0)	6.14 (0)	23.71 (0)	–	–	1.39 (0)
PP	0.51 (0)	1.79 (0)	36.48 (0)	13.59 (0)	–	–	1.52 (0)
TD	2.88 (0)	0.32 (0)	34.60 (0)	3.17 (0)	–	–	1.76 (0)

4.3 Results of the Resource Benchmark

Beyond how well a representation supports subforest selection, its practical usability also depends on how expensive it is to compute at scale. The resource efficiency of each representation is evaluated across two phases: representation construction and similarity computation. Each measurement reflects the average cost of a single operation rather than the total cost across an ensemble, and the empirical scaling exponent α characterizes how this per-operation cost changes with RF size; for an operation performed once per tree, a constant per-operation cost ($\alpha \approx 0$) corresponds to linear total cost, whereas $\alpha \approx 1$ indicates the per-operation cost itself grows linearly with RF size, yielding quadratic total cost. As shown in Table 3, four of six representations (TD, LP, PP, FG) exhibit low, near-constant runtime and peak RAM for both construction and similarity computation, and none require GPU memory. ID is one exception: its construction phase is three orders of magnitude slower than any other representation, scales super-linearly in peak RAM ($\alpha \approx 1.1$), and is the only representation with a nonzero GPU footprint, reflecting the cost of its neural training procedure. TF also shows a distinct pattern during similarity computation: its per-pair runtime *decreases* with RF size (negative α), while its peak RAM *increases*. Both trace back to TF's one-time graph-construction cost: runtime amortizes this cost across a quadratically growing number of pairs, whereas peak RAM instead reflects the construction's own peak, which grows with forest size. Appendix A.3.3 (Figure 9) provides the full per-metric scaling curves and exponents underlying this table.

5 Conclusions

This paper presents a novel benchmarking framework for DT representations which allows for their comparison along three axes: perturbation sensitivity, performance on the downstream task of subforest selection, and computational efficiency. Experiments show that different representations can be advantageous depending on the evaluation criterion. While LP generally shows the highest sensitivity to functional changes, our newly proposed TD representation achieves the most effective compression when targeting a fixed level of predictive performance, requiring substantially fewer trees than the considered baselines to recover 99% of the full RF's predictive performance. At the same time, most representations have low computational costs, with the exception of neural network-based ID representation. Together, these findings demonstrate that the suitability of a DT representation cannot be assessed based on a single property or downstream task, motivating the need for systematic and diverse evaluation such as provided by DTRBench.

Currently, the framework is limited to numeric data, classification tasks, and RF ensembles. Future work could extend it to support categorical data, regression tasks, and boosted tree ensembles. This could be achieved by adapting the representations to encode category and regression information directly, while extending the benchmark to be able to use mixed numerical and categorical datasets. The benchmark could also be extended with additional representations, in particular graph neural network-based representations, which would allow to evaluate the potential of learned DT embeddings within the same framework. Finally, evaluating more datasets and different tree depths could provide further insight into how representation behavior depends on dataset and tree characteristics.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets. We have not used generative AI tools for helping to develop theoretical models or conceptual frameworks, proposing and refining hypothesis, supporting qualitative and thematic data analysis, interpreting results, and generating synthetic data sets. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, summarizing or analyzing existing literature, brainstorming, sourcing/searching for information, editing the research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work by discussing and critically assessing the methodological correctness among the authors. AI-assisted code was manually reviewed to verify its correctness and functionality and AI-assisted text recommendations were manually reviewed and reworked. Relevant literature was searched for manually to not miss any relevant sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The complete implementation of DTRBench is provided as open-source code in the accompanying GitHub repository `https://anonymous.4open.science/r/DTRBench-anonymized`. The repository contains the benchmark implementation, all DT representations evaluated in this work, the perturbation and subforest selection methods, the 25 datasets in the exact form used for the experiments, and the corresponding preprocessing code. The experimental environment and dependencies are specified in the repository. The methodology and evaluation procedures are described in Section 3 and Appendix A.1, while the experimental setting is described in detail in Appendix A.2, and the complete dataset selection and preprocessing procedure in Section 3.1 and Appendix A.2.1. The repository additionally provides configurable report generation for the benchmark results, partially including the plots and summary tables described in Appendix A.3. Moreover, the repository contains all data and code necessary to reproduce all tables and figures contained in this paper.

REFERENCES

Md Nasim Adnan and Md Zahidul Islam. Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm. *Knowl. Based Syst.*, 110:86–97, 2016. doi: 10.1016/j.knosys.2016.07.016.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein (eds.), *SODA 2007*, pp. 1027–1035. SIAM, 2007. URL `http://dl.acm.org/citation.cfm?id=1283383.1283494`.

Mohamed Bahy Bader-El-Den and Mohamed Medhat Gaber. GARF: towards self-optimised random forests. In Tingwen Huang, Zhigang Zeng, Chuandong Li, and Chi-Sing Leung (eds.), *ICONIP 2012*, volume 7664 of *Lecture Notes in Computer Science*, pp. 506–515. Springer, 2012. doi: 10.1007/978-3-642-34481-7_62.

Gözde Bakirli and Derya Birant. DTreeSim: A new approach to compute decision tree similarity using re-mining. *Turk. J. Electr. Eng. Comput. Sci.*, 25:108–125, 2017. doi: 10.3906/elk-1504-234.

Murat Ali Bayir, Kiarash Shamsi, Huseyincan Kaynak, and Cuneyt Gurcan Akcora. Topological forest. *IEEE Access*, 10:131711–131721, 2022. doi: 10.1109/access.2022.3229008.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Bénédicte Briand, Gilles R. Ducharme, Vanessa Parache, and Catherine Mercat-Rommens. A similarity measure to assess the stability of classification trees. *Comput. Stat. Data Anal.*, 53(4): 1208–1217, 2009. doi: 10.1016/J.CSDA.2008.10.033.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educ. Psychol. Meas.*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006. URL https://jmlr.org/papers/v7/demsar06a.html.

Kumar Dheenadayalan, G. Srinivasaraghavan, and V. N. Muralidhara. Pruning a random forest by learning a learning algorithm. In Petra Perner (ed.), *MLDM 2016*, volume 9729 of *Lecture Notes in Computer Science*, pp. 516–529. Springer, 2016. doi: 10.1007/978-3-319-41920-6_41.

Tobias Emrich, Gregor Jossé, Hans-Peter Kriegel, Markus Mauder, Johannes Niedermayer, Matthias Renz, Matthias Schubert, and Andreas Züfle. Optimal distance bounds for the mahalanobis distance. In Nieves R. Brisaboa, Oscar Pedreira, and Pavel Zezula (eds.), *SISAP 2013*, volume 8199 of *Lecture Notes in Computer Science*, pp. 175–181. Springer, 2013. doi: 10.1007/978-3-642-41062-8_18.

Khaled Fawagreh, Mohamed Medhat Gaber, and Eyad Elyan. CLUB-DRF: A clustering approach to extreme pruning of random forests. In Max Bramer and Miltos Petridis (eds.), *Research and Development in Intelligent Systems XXXII. SGAI 2015*, pp. 59–73. Springer, 2015. doi: 10.1007/978-3-319-25032-8_4.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.*, 78(383):553–569, 1983. doi: 10.2307/2288117.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *NeurIPS 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/0378c7692da36807bdec87ab043cdadc-Abstract-Datasets_and_Benchmarks.html.

Bhumika Gupta, Aditya Rawat, Akshay Jain, Arpit Arora, and Naresh Dhami. Analysis of various decision tree algorithms for classification in data mining. *Int. J. Comput. Appl.*, 163(8):15–19, 2017. doi: 10.5120/ijca2017913660.

John H Holland. Genetic algorithms. *Sci. Am.*, 267(1):66–72, 1992. doi: 10.1038/scientificamerican0792-66.

Hillol Kargupta and Byung-Hoon Park. A fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments. *IEEE Trans. Knowl. Data Eng.*, 16(2):216–229, 2004. doi: 10.1109/TKDE.2004.1269599.

Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. In Yadolah Dodge (ed.), *Statistical Data Analysis Based on the L1 Norm and Related Methods*, pp. 405–416. North-Holland, 1987.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. UCI Machine Learning Repository, 2026. URL https://archive.ics.uci.edu. Accessed: 2026-07-07.

Maurice G Kendall and B Babington Smith. The problem of m rankings. *Ann. Math. Stat.*, 10(3):275–287, 1939. doi: 10.1214/aoms/1177732186.

Zardad Khan, Naz Gul, Nosheen Faiz, Asma Gul, Werner Adler, and Berthold Lausen. Optimal trees selection for classification via out-of-bag assessment and sub-bagging. *IEEE Access*, 9:28591–28607, 2021. doi: 10.1109/ACCESS.2021.3055992.

Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.*, 51(2):181–207, 2003. doi: 10.1023/A:1022859003006.

Thomas M. Lange, Mehmet Gültas, Armin Otto Schmitt, and Felix Heinrich. optRF: Optimising random forest stability by determining the optimal number of trees. *BMC Bioinform.*, 26(1):95, 2025. doi: 10.1186/S12859-025-06097-1.

Youness Manzali and Mohamed Elfar. Random forest pruning techniques: A recent review. *Oper. Res. Forum*, 4(2):43, 2023. doi: 10.1007/S43069-023-00223-6.

Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In Douglas H. Fisher (ed.), *ICML 1997*, pp. 211–218. Morgan Kaufmann, 1997. URL https://web.engr.oregonstate.edu/~tgd/publications/ml97-pruning-adaboost.pdf.

B W Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta*, 405(2):442–451, 1975. doi: 10.1016/0005-2795(75)90109-9.

Duncan C. McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C., Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *NeurIPS 2023*, pp. 76336–76369, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/f06d5ebd4ff40b40dd97e30cee632123-Abstract-Datasets_and_Benchmarks.html.

Rossella Miglio and Gabriele Soffritti. The comparison between classification trees through proximity measures. *Comput. Stat. Data Anal.*, 45(3):577–593, 2004. doi: 10.1016/S0167-9473(03)00063-X.

Irene Ntoutsi, Alexandros Kalousis, and Yannis Theodoridis. A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees. In *SDM 2008*, pp. 810–821. SIAM, 2008. doi: 10.1137/1.9781611972788.73.

E C Pielou. The measurement of diversity in different types of biological collections. *J. Theor. Biol.*, 13:131–144, 1966. doi: 10.1016/0022-5193(66)90013-0.

J. Ross Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986. doi: 10.1007/bf00116251.

Farnaz Rahimi, Christel Sirocchi, Julian Matschinske, Markus Metzler, Jakob Zierk, and David B Blumenthal. Non-temporal tree-based models outperform temporal deep learning models in the prediction of chemotherapy-induced side effects from longitudinal laboratory data. *medRxiv*, 2025. doi: 10.64898/2025.12.12.25342142.

Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014. doi: 10.1126/science.1242072.

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975. doi: 10.1145/361219.361220.

Terry Sejnowski and R. Gorman. Connectionist Bench (Sonar, Mines vs. Rocks). UCI Machine Learning Repository, 1988. URL https://doi.org/10.24432/C5T01Q.

William D Shannon and David Banks. Combining classification trees using MLE. *Stat. Med.*, 18(6):727–740, 1999. doi: 10.1002/(sici)1097-0258(19990330)18:6⟨727::aid-sim61⟩3.0.co;2-2.

Grigori Sidorov, Alexander F. Gelbukh, Helena Gómez-Adorno, and David Pinto. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3), 2014. doi: 10.13053/CYS-18-3-2043.

Gurjeet Singh, Facundo Memoli, Tigran Ishkhanov, Guillermo Sapiro, Gunnar Carlsson, and Dario L Ringach. Topological analysis of population activity in visual cortex. *J. Vis.*, 8(8):11.1–18, 2008. doi: 10.1167/8.8.11.

Christel Sirocchi, Martin Urschler, and Bastian Pfeifer. Feature graphs for interpretable unsupervised tree ensembles: centrality, interaction, and application in disease subtyping. *BioData Min.*, 18(1):15, 2025. doi: 10.1186/s13040-025-00430-3.

Francesco Spinnato, Antonio Mastropietro, and Riccardo Guidotti. Implicit neural decision trees. In *ESANN 2025*, pp. 585–590, 2025. doi: 10.14428/ESANN/2025.ES2025-85.

Noopur Srivastava and Shrisha Rao. Learning-based text classifiers using the mahalanobis distance for correlated datasets. *Int. J. Big Data Intell.*, 3(1):18–27, 2016. doi: 10.1504/IJBDI.2016.073901.

Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: illustrations, sources and a solution. *BMC Bioinform.*, 8(1):25, 2007. doi: 10.1186/1471-2105-8-25.

Peter J M van Laarhoven and Emile H L Aarts. Simulated annealing. In *Simulated Annealing: Theory and Applications*, pp. 7–15. Springer Netherlands, Dordrecht, 1987. doi: 10.1007/978-94-015-7744-1_2.

Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Probl. Peredachi Inf.*, 5(3):64–72, 1969.

Joe H Ward, Jr. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.*, 58 (301):236–244, 1963. doi: 10.1080/01621459.1963.10500845.

Abraham Itzhak Weinberg and Mark Last. Selecting a representative decision tree from an ensemble of decision-tree models for fast big data classification. *J. Big Data*, 6(1), 2019. doi: 10.1186/s40537-019-0186-3.

Heping Zhang and Minghui Wang. Search for the smallest random forest. *Stat. Interface*, 2(3):381, 2009. doi: 10.4310/sii.2009.v2.n3.a11.

A Appendix

A.1 Additional Methods

A.1.1 Tree Perturbations

The perturbation benchmark perturbs scikit-learn `DecisionTreeClassifier` objects in Python. Five perturbations have been implemented, each called with an intensity $\lambda \in [0, 1]$ denoting the fraction of nodes affected, applied relative to the number of internal (non-leaf) nodes n_{internal} for *change threshold*, *change feature*, and *swap nodes*, and relative to the total node count n_{total} for *remove nodes* and *add nodes*.

Change threshold. Changes the threshold of $\lfloor \lambda \cdot n_{\text{internal}} \rfloor$ randomly selected internal nodes. The threshold is shifted by Gaussian noise scaled to the feature's observed range in the training data (rounded to the nearest integer for integer-valued features).

Change feature. Changes the feature of $\lfloor \lambda \cdot n_{\text{internal}} \rfloor$ randomly selected internal nodes to a uniformly sampled feature, with the corresponding threshold chosen to minimize the weighted Gini impurity of the samples reaching that node.

Swap nodes. Swaps the feature and threshold between $\lfloor \lambda \cdot n_{\text{internal}}/2 \rfloor$ pairs of randomly selected internal nodes, sampled without replacement from anywhere in the tree (not restricted to siblings or nearby nodes); each node participates in at most one swap.

Remove nodes. Repeatedly selects, at random, a parent node whose two children are both leaves, and removes both children, turning the parent into a leaf, until at least $\lfloor \lambda \cdot (n_{\text{total}} - 1) \rfloor$ nodes have been removed, rounding up to the nearest even number if this target is odd (since nodes are removed in sibling pairs). Because eligible parents change as nodes are removed, candidates are re-identified after each removal.

Add nodes. Repeatedly selects a random leaf node below a maximum depth of 8 (leaving headroom within the trees' maximum depth of 10 for the newly added children) and expands it into an internal node with two new leaf children, until at least $\lfloor \lambda \cdot (n_{\text{total}} - 1) \rfloor$ nodes have been added, rounding up to the nearest even number if this target is odd (since nodes are added in pairs), or until no eligible leaf below this depth remains. The new node's feature is chosen at random and its threshold optimally, as in *change feature*.

A.1.2 Tree Representations

Feature Graph (FG). The FG representation by Sirocchi et al. (2025) represents a tree t as a weighted adjacency matrix $\text{FG}(t) = E \in \mathbb{R}^{(d+1)\times(d+1)}$ over the set of d input features and an additional terminal node T, such that $E(i, j)$ is the weight of the edge connecting i and j, where i and j can each be a feature or the terminal node. The graph is built by recursively traversing the DT from the root. At each internal node that splits on feature u, a weighted edge is added from u to the splitting features of its left and right children. If a child is a leaf, the edge instead connects to the terminal node T. The traversal continues recursively through each non-leaf child until all leaves are reached. Among the weighting schemes proposed in the original work, this study adopts the best-performing one, *sample*, which weights each edge by the proportion of the tree's total training samples routed to that child.

Since Sirocchi et al. (2025) does not define a similarity measure, we compute the distance δ_{FG} between two FG representations using a weighted Frobenius norm between two trees' edge matrices $E_a = \text{FG}(t_a)$ and $E_b = \text{FG}(t_b)$ over the same feature set:

$$\delta_{\text{FG}}(t_a, t_b) = \sqrt{\textstyle\sum_{i,j} w_{ij} \left(E_a(i, j) - E_b(i, j)\right)^2}, \quad w_{ij} = \begin{cases} 1 & i = T \text{ or } j = T \\ 1 - |\rho_{ij}| & \text{otherwise,} \end{cases}$$

where ρ_{ij} is the absolute Pearson correlation between features i and j on the training data, motivated by prior work on correlation-aware distances, which account for dependencies among features when

measuring similarity in correlated data (Emrich et al., 2013; Srivastava & Rao, 2016). This reflects that two trees splitting on different but highly correlated features encode similar decision logic and should be penalized less. Similarity is then obtained as $s_{\mathrm{FG}}(t_a, t_b) = 1/(1 + \delta_{\mathrm{FG}}(t_a, t_b)) \in (0, 1]$. All trees compared under this representation must share the same feature space, and the correlation matrix is computed once from a shared reference dataset and reused across all pairwise comparisons, except in the perturbation benchmark, where it is instead recomputed from each tree's own bootstrap sample (Appendix A.3.1).

Topological Forest (TF). Following Bayir et al. (2022), each tree t is first summarized by a feature interaction graph, built by connecting each splitting feature to the splitting features of its children (skipping leaves), and recursing through the tree from the root. This graph is then reduced to a fixed-length feature vector $\mathrm{TF}(t)$ of topological summary statistics: graph size (number of nodes and edges, diameter), motif counts (the 16 three-node triadic census classes), and centrality measures (clustering coefficient, average in-, out-, and total degree, average shortest path length, and average betweenness centrality). Diameter and average shortest path length are computed on the largest strongly connected component when the graph itself is not strongly connected.

Bayir et al. (2022) then derive similarity from a shared topological map built across all trees in the population using the Mapper algorithm (Singh et al., 2008): the (min-max scaled) feature vectors $\mathrm{TF}(t)$ are projected to two dimensions via t-SNE, covered by overlapping bins, and clustered within each bin via k-means, yielding a graph G in which nodes are clusters of similar trees and edges connect clusters with overlapping members. For two trees t_a, t_b, we define $\delta_{\mathrm{TF}}(t_a, t_b)$ as the shortest path length in G between any cluster containing t_a and any cluster containing t_b, and set similarity as

$$s_{\mathrm{TF}}(t_a, t_b) = \begin{cases} 1 & t_a, t_b \text{ share a cluster in } G \\ 1/(1 + \delta_{\mathrm{TF}}(t_a, t_b)) & \delta_{\mathrm{TF}}(t_a, t_b) \text{ defined} \\ 0 & \text{otherwise.} \end{cases}$$

Notably, since this representation is defined relative to a shared Mapper graph, all trees to be compared must be represented jointly, and the graph must be rebuilt whenever the set of trees changes.

INDTree (ID). Following Spinnato et al. (2025), each tree t is encoded into a shared continuous representation space using an implicit neural network, meta-trained across all trees via MAML (Finn et al., 2017) so that, after per-tree fine-tuning, it reconstructs each tree's feature, threshold, and class assignments from a common input space of node coordinates. Concretely, each tree is first converted into a fixed-size direct encoding of its per-node splitting features, thresholds, and class assignments, padded to the maximum tree depth in the population; a single shared SIREN-based network is then meta-trained on these encodings. The resulting per-tree fine-tuned network is denoted $\mathrm{ID}(t)$.

Since Spinnato et al. (2025) do not define a similarity measure between trees, we introduce one here by comparing each tree's fine-tuned network outputs. For two trees t_a, t_b, the fitted network's predicted feature, threshold, and class outputs, $(\hat{f}_a, \hat{t}_a, \hat{c}_a)$ and $(\hat{f}_b, \hat{t}_b, \hat{c}_b)$ respectively, are compared, and the distance $\delta_{\mathrm{ID}}(t_a, t_b)$ between the trees is defined as the sum of pairwise Euclidean distances between these outputs,

$$\delta_{\mathrm{ID}}(t_a, t_b) = \|\hat{f}_a - \hat{f}_b\|_2 + \|\hat{t}_a - \hat{t}_b\|_2 + \|\hat{c}_a - \hat{c}_b\|_2,$$

with similarity obtained as $s_{\mathrm{ID}}(t_a, t_b) = 1/(1 + \delta_{\mathrm{ID}}(t_a, t_b)) \in (0, 1]$. As an alternative, the same distance can instead be computed directly on each tree's input encoding (coordinates, one-hot features, thresholds, and classes) rather than the network's output; the results reported here use the output-based variant. Since all trees share the same underlying network, the network must be retrained jointly over the full population of trees to be compared before any pairwise similarity can be computed.

Leaf Profile (LP). Inspired by Ntoutsi et al. (2008), who represent a DT's partition of the attribute space via each leaf's *measure component* (the class distribution of training instances reaching that leaf), we represent a tree t as the set of its leaves L_t, each described by a probability mass m (the proportion of training samples reaching that leaf) and a class distribution p (the normalized class counts at that leaf): $\mathrm{LP}(t) = \{(m_i, p_i)\}_{i=1}^{|L_t|}$.

Ntoutsi et al. (2008) compare trees by overlaying their induced partitions and applying an affinity coefficient between region-wise class distributions, weighted by an estimate of the attribute space distribution, requiring either access to the original data or an additional assumption about that distribution. We instead define similarity directly on the leaf profiles: given two trees t_a, t_b with leaf profiles $\mathrm{LP}(t_a)$ and $\mathrm{LP}(t_b)$, we compute the 1-Wasserstein distance $\delta_{\mathrm{LP}}(t_a, t_b)$ (Vaserstein, 1969) between the two leaf-mass distributions, treating each tree's leaves as point masses located in class-distribution space, with the ground cost given by the Euclidean distance between class distributions (zero-padded, at matching class indices, if the two trees' class sets differ in size). Representation similarity is then $s_{\mathrm{LP}}(t_a, t_b) = 1/(1 + \delta_{\mathrm{LP}}(t_a, t_b))$. This naturally accommodates trees with different numbers of leaves, unlike the overlay-based comparison (Ntoutsi et al., 2008), which requires a joint partition of the two trees' regions.

Prediction Profile (PP). A tree t is represented as its vector of predicted class labels on a fixed reference dataset: $\mathrm{PP}(t) = (\hat{y}_1, \ldots, \hat{y}_n)$. Since each DT's bootstrap sample differs from every other tree's, the reference dataset used here is instead the RF's shared training set as in Fawagreh et al. (2015), ensuring all trees' prediction vectors are computed over the same samples and remain directly comparable.

Similarity is defined as Cohen's kappa agreement between predictions (Cohen, 1960), following the prediction-based similarity criteria used by Margineantu & Dietterich (1997) and computed over the training set as by Fawagreh et al. (2015). For two prediction vectors on the same set of samples, Cohen's kappa is

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

where p_o is the observed agreement, i.e., the proportion of samples on which the two predictions agree, and p_e is the agreement expected by chance given each tree's marginal class distribution. A value of $\kappa = 1$ indicates full agreement, $\kappa = 0$ agreement no better than chance, and $\kappa < 0$ agreement worse than chance. To remain within the $[0, 1]$-bounded similarity used by the other representations, we rescale $s_{\mathrm{PP}}(t_a, t_b) = (\kappa + 1)/2$.

Tree Descriptor (TD). Alongside representations adapted from prior work, we define a novel reference representation that summarizes each tree t using a fixed-size vector $\mathrm{TD}(t)$ divided into three parts, capturing structural, functional, and decision-boundary characteristics. It is designed to address the lack of feature thresholds in graph-based representations while avoiding the computational overhead and reduced interpretability of neural representations. *Part (1)* describes structural characteristics: node count, depth, width, and both unweighted and sample-weighted average leaf path length are $\log 1p$-scaled to account for their typically skewed distributions, alongside balance (the standard deviation of leaf depths), average leaf impurity, and feature diversity (proportion of available features used for splitting), used without transformation. *Part (2)* captures functional characteristics through normalized FI and the proportion of internal (splitting) nodes using each feature. *Part (3)* encodes decision-boundary characteristics through average split thresholds for each feature, min-max normalized to $[0, 1]$ relative to its observed range in the training data.

Given two trees t_a, t_b with descriptors $\mathrm{TD}(t_a)$ and $\mathrm{TD}(t_b)$, similarity is computed separately for each partition and then combined via a weighted average. Part (1) similarity $s_{(1)}(t_a, t_b)$ is the cosine similarity (Salton et al., 1975) between the two structural statistic vectors. Part (2) similarity $s_{(2)}(t_a, t_b)$ averages the soft cosine similarity (Sidorov et al., 2014) of the two trees' FI vectors and of their feature usage vectors; this correlation-aware measure, computed via the training-data feature correlation matrix (as in FG), accounts for similarities between correlated features rather than treating them as independent dimensions. Part (3) similarity $s_{(3)}(t_a, t_b)$ is $1 - \mathrm{MAE}$ between normalized thresholds, restricted to the subset of features used by both trees (and defined as 0 if no such feature is shared). Although cosine and soft cosine similarity are generally bounded in $[-1, 1]$, all vectors entering $s_{(1)}$ and $s_{(2)}$ (structural statistics, FI, and feature usage) are non-negative by construction, which confines both to $[0, 1]$; $s_{(3)}$ is likewise in $[0, 1]$, since it is computed over thresholds already normalized to that range. DTRBench combines the similarities of the three parts

$$s_{\mathrm{TD}}(t_a, t_b) = 0.5\, s_{(1)}(t_a, t_b) + 0.3\, s_{(2)}(t_a, t_b) + 0.2\, s_{(3)}(t_a, t_b).$$

To the best of our knowledge, TD is the only representation that explicitly incorporates FI, capturing not only which features a tree uses but also their relative contribution to its predictions. The

weights 0.5, 0.3, and 0.2 were chosen a priori to reflect the relative importance assigned to each component, prioritizing structural characteristics while still incorporating feature relevance and decision boundaries; neither an optimization of these weights nor a sensitivity analysis of the resulting representation was performed, and both are left for future work.

A.1.3 Selection Strategies

Similarity-to-distance transformation. All representations in DTRBench expose a similarity computation, $s_R(t_a, t_b) \in [0, 1]$. The subforest benchmark requires computing pairwise distances between trees, derived identically for every representation R from the full matrix of pairwise similarities $S = (s_R(t_a, t_b))_{a \neq b}$ over all pairs of trees a, b under comparison, by min-max scaling S and subtracting it from 1:

$$D_{ab} = \begin{cases} 0 & a = b \\ 1 - \dfrac{S_{ab} - \min(S)}{\max(S) - \min(S)} & a \neq b. \end{cases}$$

The perturbation benchmark (Section 3.3) instead computes distances as $1 - s_R$ directly, since it is only used for correlation and does not require a collection-relative rescaling. The subforest benchmark uses these representation-derived distances between trees, together with information about their predictive performance, to select a representative subforest from a RF. The exact selection procedure also affects the quality of the resulting subforests; DTRBench therefore implements eight selection strategies, described below.

Clustering-based selection (KM-S & AG-S). In order to directly utilize the distances computed through the DT representations, distance-based clustering methods partition RF trees into k distinct groups, where k is the desired subforest size. A single representative tree is then selected from each cluster, with the aim of reducing redundancy while preserving the diversity of the full ensemble. Both implemented methods operate on the same precomputed distance matrix but follow two different clustering paradigms. The first approach applies k-medoid clustering (KM-S) (Kaufman & Rousseeuw, 1987), which identifies representative DTs directly by minimizing distances within each cluster; the resulting subforest is simply the k medoid trees obtained once the method has converged. The second method is agglomerative hierarchical clustering (AG-S) (Ward, 1963) with average linkage, in which tree clusters are successively merged until exactly k clusters remain. Since hierarchical clustering does not produce explicit cluster representatives, an additional post-processing step is required: for each cluster, the local medoid is computed as the DT with the smallest summed distance to all other trees in that cluster. Note that if a cluster produced by agglomerative clustering happens to be empty, no representative is drawn from it, so the resulting subforest may contain fewer than k trees.

Performance-guided clustering-based selection (KMP-S & AGP-S). To incorporate the predictive performance of individual trees into the clustering strategies, a performance-oriented variant is introduced, adapting the cluster-then-select structure of CLUB-DRF (Fawagreh et al., 2015) by choosing each cluster's representative based on predictive performance rather than distance-based centrality. It reuses the same clusterings produced by either KM-S or AG-S, but differs in how the final representative tree is chosen from each cluster: rather than selecting the central tree, this method selects the DT with the highest OOB MCC within each cluster. Consequently, each cluster contributes its best-performing tree rather than its distance-based medoid, aiming for a subforest that combines strong predictive performance and diversity.

Density-based selection (DE-S). Inspired by the density-peaks principle of Rodriguez & Laio (2014), according to which representative elements are characterized by a high local density and a large distance from other high-density elements, DE-S estimates a kernel density for each tree from the pairwise distance matrix and combines it with a sequential distance-based penalty, similar to k-means++ seeding (Arthur & Vassilvitskii, 2007), to favor dense yet mutually dissimilar trees. For a DT a, the density is defined as

$$\zeta_a = \sum_{b \in F} \exp\left(-\frac{D_{ab}^2}{\sigma}\right),$$

where F denotes the set of trees comprising the RF, D_{ab} denotes the pairwise distance between trees a and b (with $D_{aa} = 0$), and σ controls the effective neighborhood size of the kernel. The resulting densities are normalized to obtain initial sampling probabilities for all DTs, from which trees are drawn sequentially: at each step, a tree is sampled from the remaining candidates, with probabilities renormalized among them, so that trees in dense regions are more likely to be selected. After a tree has been selected, the probabilities of the remaining trees are adjusted to reduce the likelihood of selecting similar trees in subsequent iterations, by multiplying each probability by its distance to the newly selected tree raised to the power β, the adaptation coefficient. This lowers the probabilities of trees in the immediate neighborhood of the selected tree, while preserving the probabilities of distant trees; if this adjustment drives every remaining tree's probability to zero, the next tree is instead drawn uniformly at random from the remaining candidates. Since the sampling outcome depends on the neighborhood size σ and adaptation coefficient β, the method performs a grid search over both: σ is drawn from a 17-point logarithmically spaced grid centered on the median pairwise distance, and β from a 17-point logarithmically spaced grid ranging from $1/3$ to 3. For each (σ, β) combination, one subforest is evaluated by its OOB MCC; the configuration achieving the highest valid OOB MCC, together with its resulting subforest, is returned.

Greedy selection (GR-S). To balance predictive performance and diversity within the subforest, this work proposes a greedy selection strategy based on multi-objective optimization, in the spirit of the greedy component of reduce-error ensemble pruning (Margineantu & Dietterich, 1997), which grows a subset by repeatedly adding the classifier that most improves the performance of the currently selected subensemble. Here, each candidate tree is instead evaluated jointly on the diversity and predictive performance it would contribute to the subforest. Since distance and OOB MCC values may differ in scale, both are separately z-score normalized prior to aggregation and averaged with equal weight. The procedure begins by selecting an initial DT according to a combined score that jointly maximizes standalone OOB MCC and minimizes distance to all other trees in the RF, yielding a high-quality, central starting tree. In each subsequent iteration, every remaining candidate is scored on two criteria: its average pairwise distance to the trees already included in the subforest (its diversity contribution), and its running average OOB MCC together with the trees already selected. These two (normalized) criteria are averaged into a single combined objective, and the candidate maximizing this objective is added to the subforest, with ties broken at random. Repeating this process until the subforest reaches its desired size yields a set of trees that balances predictive strength against ensemble diversity.

Simulated annealing selection (SA-S). To mitigate the tendency of greedy algorithms to become trapped in local maxima, simulated annealing (van Laarhoven & Aarts, 1987) is used as a heuristic optimization strategy that searches more broadly across the space of candidate subforests. It uses an objective analogous to GR-S's, jointly maximizing the average pairwise distance between trees in the subforest and their average OOB MCC. Since suitable annealing parameters depend heavily on input size (e.g., larger RFs require more overall steps), they are chosen dynamically rather than fixed. The starting temperature is calibrated via an exploration phase in which a single tree in the subforest is repeatedly replaced at random over 25 trials; the resulting average change in objective score across these trials is then used to solve for a starting temperature at which such a typical worsening move would be accepted with probability 0.8, falling back to a default temperature if this calibration is undefined or non-positive. The algorithm begins from a randomly constructed subforest of the desired size. In the main optimization loop, it explores the search space via random single-tree replacements: perturbations that improve the objective are always accepted, while worse ones are accepted with probability equal to $\exp\left(\frac{\Delta q}{\text{Temp}}\right)$, where $\Delta q = q_{\text{perturbed}} - q_{\text{current}}$ is the change in objective score and Temp is the current temperature; this occasional acceptance of worse subforests allows the algorithm to escape local maxima, particularly in early iterations. The temperature decays exponentially with a cooling factor of 0.98, gradually shifting the search from global exploration to local fine-tuning, and optimization terminates once the temperature reaches 0.1.

Genetic selection (GE-S). As a further approach to escaping local maxima, a genetic algorithm (Holland, 1992) evolves a population of 100 candidate subforests over 50 generations. Each candidate represents a unique subforest and is evaluated on an objective analogous to the previous strategies, jointly maximizing the average pairwise distance between trees in the subforest and their average OOB MCC. The initial population is created at random. To preserve high-quality solutions

across generations, the algorithm applies elitism, carrying the top 3% of candidates directly into the next generation. The remaining candidates are produced by selecting parent pairs via stochastic universal sampling, a selection method that reduces sampling variance by drawing candidates at evenly spaced intervals across the cumulative fitness distribution, giving higher-scoring candidates a proportionally higher chance of selection. Selected parent pairs undergo single-point crossover (exchanging trees on either side of a randomly chosen crossover point) with probability 0.9; a duplicate-aware fill scheme is applied during crossover so that the resulting subforest never contains the same tree twice. Each candidate produced by crossover is then independently mutated with probability $P = 1/(2 \times \text{subforest size})$, where mutation replaces a randomly chosen tree in the subforest with any non-duplicate DT, further promoting diversity and broadening the search.

Baseline selection strategies. In addition to the eight representation-dependent strategies, DTR-Bench evaluates three baselines that do not rely on any tree representation. **Random (RDM)** selects trees uniformly at random, without replacement, from the full RF. For each subforest size, ten independent random subforests are drawn and evaluated separately to characterize the variability of random selection. The evaluated metrics are then averaged across all ten random subforests. **Top OOB Accuracy (TOA)** and **Top OOB MCC (TOM)** rank all trees in the RF by their individual OOB accuracy or OOB MCC, respectively, and select the top-k ranked trees for a subforest of size k. Trees for which OOB predictions are unavailable (e.g., due to an empty OOB sample) are assigned a score of $-\infty$, ensuring they are ranked last and excluded from small subforests.

A.1.4 Extensibility

DTRBench is designed as a modular benchmarking framework in which every core component is interchangeable. Users can plug in custom DT representations and datasets (used across all benchmarks), as well as custom selection strategies (subforest benchmark) and perturbations (perturbation benchmark), with the provided implementations serving as standardized defaults rather than a fixed, closed set of options. Extending any of these components requires only implementing the corresponding interface and registering it through the framework's registry; the core benchmark logic itself never needs to be modified. The code snippets below are minimal, illustrative examples intended to demonstrate the registration interface.

Custom DT representations. A representation must implement `represent(tree, X_train)`, encoding a fitted tree into an arbitrary object, and `similarity(a, b)`, returning a scalar similarity between two encodings. Once registered via `register_representation` with a factory taking the training data and a seed, it is automatically available to all three benchmarks under the same protocol used for the six representations in this work. The snippet below is a minimal toy example illustrating the interface, where a tree is stored as a dictionary of its leaf count and depth, and similarity is computed as the inverse of their combined absolute difference.

```python
from dtrbench.representations.registry import register_representation
from dtrbench.representations.base import BaseRepresentation

class MyTreeRepresentation(BaseRepresentation):
    def __init__(self, X_train, seed):
        self.value = 5

    def represent(self, tree, X_train):
        return {
            "n_leaves": tree.tree_.n_leaves,
            "depth": tree.get_depth()
        }

    def similarity(self, a, b):
        leaf_diff = abs(a["n_leaves"] - b["n_leaves"])
        depth_diff = abs(a["depth"] - b["depth"])
        return 1.0 / (1.0 + leaf_diff + depth_diff)

register_representation(
    "MyTree",
    lambda X_train, seed: MyTreeRepresentation(X_train, seed)
)
```

Custom datasets. A dataset is introduced by registering a loader that takes the number of CV splits, an optional cap on the number of samples to use, and a seed, and returns the dataset's name, pre-split folds, and a feature-metadata table giving each feature's type (e.g., `continuous`); it is then loaded through the same interface as the built-in datasets across all benchmarks. Each built-in dataset resides in a directory `dataset_name/` containing `X.npy` and `y.npy` (feature matrix and targets) and `features.csv` (feature names and types).

```python
from sklearn.model_selection import StratifiedKFold
from dtrbench.datasets.registry import register_dataset

@register_dataset("my_dataset")
def load_my_dataset(n_splits, n_samples, seed):
    X = np.load("path/to/my_dataset/X.npy")[:n_samples]
    y = np.load("path/to/my_dataset/y.npy")[:n_samples]
    features = pd.DataFrame({"type": ["continuous"] * X.shape[1]})

    skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=seed)
    folds = [
        (X[train_idx], X[test_idx], y[train_idx], y[test_idx])
        for train_idx, test_idx in skf.split(X, y)
    ]
    return {"name": "my_dataset", "folds": folds, "features": features}
```

Custom selection strategies. A selection strategy takes a representation's distance matrix, the target subforest size, and optionally auxiliary information such as OOB performance, and returns the indices to retain. This covers all strategies evaluated in this work, from purely distance-based clustering to joint diversity-performance optimization, and lets users add alternative criteria without modifying the subforest benchmark's pipeline. The illustrative example below shows the minimal distance-only case, selecting the k trees with the highest average distance to the rest of the forest.

```python
from dtrbench.selection_strategies.registry import register_selection_strategy

@register_selection_strategy("my_selection")
def my_selection_strategy(distance_matrix, subforest_size):
    scores = distance_matrix.mean(axis=1)
    subforest_indices = scores.argsort()[-subforest_size:]
    return subforest_indices
```

Custom perturbations. A perturbation takes a tree, a template tree with pre-allocated capacity for perturbations that grow the tree, the training data and labels, feature-type metadata, an intensity in $[0, 1]$, and a seed, and returns a modified copy. The benchmark handles iterating over the configured intensities, applying the perturbation to every tree in the forest, and recording resulting performance and representation shifts automatically, so a custom perturbation need only implement the transformation itself. The example below illustrates uniform threshold noise.

```python
from dtrbench.perturbations.registry import register_perturbation

@register_perturbation("uniform_noise")
def uniform_noise(tree, template_tree, X_train, y_train, features_info, intensity, seed):
    tree_copy = copy.deepcopy(tree)
    tree_state = tree_copy.tree_.__getstate__()
    rng = np.random.default_rng(seed)

    split_nodes = np.where(tree_state["nodes"]["feature"] >= 0)[0]
    n_perturb = int(len(split_nodes) * intensity)
    nodes = rng.choice(split_nodes, n_perturb, replace=False)

    for node_id in nodes:
        feature = tree_state["nodes"][node_id]["feature"]
        f_range = X_train[:, feature].max() - X_train[:, feature].min()
        tree_state["nodes"][node_id]["threshold"] += rng.uniform(-0.1, 0.1) * f_range

    tree_copy.tree_.__setstate__(tree_state)
    return tree_copy
```

Table 4: Datasets included in the framework. Includes classification datasets from the UCI Machine Learning Repository with large differences between RF and single DT predictive performance. Only continuous and integer features are used, categorical features are dropped. Class balance is computed using the normalized Shannon entropy (Pielou, 1966).

Dataset Name	UCI ID	Samples	Features	Classes	Class Balance
Connectionist Bench (Sonar, Mines vs. Rocks)	151	208	60	2	0.9967
Cervical Cancer Behavior Risk	537	72	19	2	0.8709
DARWIN	732	174	450	2	0.9996
Cylinder Bands	32	370	22	2	0.9820
Musk (Version 1)	74	476	166	2	0.9877
Credit Approval	27	666	6	2	0.9912
Waveform Database Generator (Version 1)	107	5000	21	3	0.9999
Heart Disease	45	297	13	5	0.7950
Japanese Credit Screening	28	690	4	2	0.9912
Statlog (Heart)	145	270	13	2	0.9911
MONK's Problems	70	432	6	2	1.0000
Cirrhosis Patient Survival Prediction	878	312	7	3	0.7853
Heart Failure Clinical Records	519	299	12	2	0.9056
Wine	109	178	13	3	0.9886
SUPPORT2	880	370	35	2	0.9032
EEG Eye State	264	14980	14	2	0.9924
Statlog (German Credit Data)	144	1000	7	2	0.8813
Statlog (Vehicle Silhouettes)	149	845	18	4	0.8657
ISOLET	54	7797	617	26	1.0000
Diabetic Retinopathy Debrecen	329	1151	21	2	0.9973
Letter Recognition	59	20000	16	26	0.9997
Fertility	244	100	9	2	0.5294
Balance Scale	12	625	4	3	0.8316
Statlog (Australian Credit Approval)	143	690	14	2	0.9912
Vertebral Column	212	310	6	3	0.9412

A.2 Additional Experimental Details

A.2.1 Dataset Selection

To evaluate DT representations in settings where ensembles are clearly advantageous for generalization, DTRBench selects 25 datasets from the UCI Machine Learning Repository (Kelly et al., 2026) through a filtering and ranking procedure. The initial preprocessing step retains only numerical features, consistent with the representation constraints described in Section 3.1. Missing values were imputed using k-nearest neighbor imputation with $k = 5$. Datasets with fewer than 20 total samples or fewer than 10 in the minority class were additionally excluded to ensure statistical stability during CV.

The remaining datasets were evaluated by training both a single DT (maximum depth 10) and an RF of 250 DTs (each with maximum depth 10) using 10-fold stratified CV. The difference in performance between DT and RF was then measured using the weighted F1-score, MCC, and ROC AUC (one-vs-rest for multiclass problems) as

$$\Delta_{\text{metric}} = \text{metric}_{RF} - \text{metric}_{DT}.$$

For each metric, datasets were ranked in descending order of Δ_{metric}, with rank 1 assigned to the largest difference; a combined ranking was then formed by summing rank positions across all three metrics, and the 25 highest-ranked datasets were selected for DTRBench. This procedure ensures that representations are evaluated on classification problems where ensembles hold a substantial predictive advantage over individual DTs. Table 4 provides an overview of the 25 datasets included in DTRBench; class balance is computed using the normalized Shannon entropy (Pielou, 1966).

A.2.2 REPRESENTATION CONFIGURATION

FG uses the sample criterion. TF uses a 2-D t-SNE embedding with perplexity 30, a Mapper construction with 10 cubes and 60% overlap, and k-means with $k = 5$. ID uses direct encodings and compares DTs via output with COORDS_TYPE binary; within ID, MAML uses first-order gradients with an inner-loop learning rate of $1e{-}3$ and an outer-loop learning rate of $1e{-}5$, trained for a maximum of 10 epochs with early stopping (min_stop 0.025, patience 2) and a batch size of 128. SIREN models use 1 hidden layer of width 32. The comparably small number of epochs was chosen primarily for resource constraints; this is assumed sufficient since MAML is only used to learn a shared initialization, the resulting trees have similar features, value ranges, and decision logic, and the embeddings are used only for approximate distance calculations. For additional configuration, please refer to the GitHub repository `https://anonymous.4open.science/r/DTRBench-anonymized`.

A.2.3 PERTURBATION BENCHMARK

The perturbation benchmark uses RFs of 50 DTs with a maximum depth of 10; otherwise, parameters for DTs are default (criterion="gini", splitter="best", min_samples_split=2, min_samples_leaf=1, min_weight_faction_leaf=0.0, max_features="sqrt", max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0). Because *add nodes* requires growing a tree beyond its original size, and scikit-learn's underlying tree structure cannot be resized in place, each tree is first reduced via a single *remove nodes* perturbation at intensity 0.5, applied to its own bootstrap sample, yielding a *base* tree with enough headroom to grow back within the memory pre-allocated for the original, unperturbed tree. This base tree also serves as a consistent, moderately simplified reference point shared across all five perturbation types. Each perturbation is then applied once to the base tree at each of five intensities, $\lambda \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. For FG and TD, whose similarity computation depends on training-data statistics via a correlation-adjustment matrix, these statistics are recomputed from each tree's own bootstrap sample rather than the global training set, so that comparisons reflect the data each tree was actually trained on. TF and ID, on the other hand, require the full population of base and perturbed trees before their representations can be computed, since TF relies on a shared Mapper graph and ID on a jointly meta-trained network; their representations are therefore collected across all perturbations and intensities first, with similarity to the corresponding base tree computed only once the full population is available.

For each cluster of observations sharing the same representation, dataset, CV fold, and perturbation type, we compute the Pearson correlation between representation distance, $1 - s_R(t, \tilde{t})$ for a base tree t and its perturbed counterpart $\tilde{t}$, and each of the two functional-change measures: the difference in predictive performance (OOB MCC) and the shift in FI. As throughout the whole framework, FI is based on Gini impurity. The resulting per-cluster correlations are then combined via Fisher z-transformation, which maps each correlation r to $z = \operatorname{arctanh}(r)$, combines the z-values weighted by their precision, and transforms the result back to the correlation scale. This combination can proceed in one of two ways: under *one-stage* aggregation, all per-cluster correlations for a given representation, measure, and perturbation are combined directly into a single overall estimate; under *two-stage* aggregation, per-cluster correlations are first combined across CV folds into one estimate per dataset, and these per-dataset estimates are then combined across datasets into an overall estimate. As the two aggregation schemes yield equivalent results, only one-stage aggregation is reported. To assess whether representations differ significantly within each (measure, perturbation) cell, their per-cluster correlations are compared using a Friedman test, blocked by dataset and fold. Pairwise differences in average rank are then assessed with the Nemenyi post-hoc test and visualized as critical difference diagrams (Demšar, 2006), where a horizontal bar connects representations whose average ranks are not significantly different at 0.05.

A.2.4 SUBFOREST BENCHMARK

The subforest benchmark uses RFs of 1000 DTs (with DT parameters specified in Appendix A.2.3). We evaluate 20 logarithmically spaced subforest sizes from 2 to 1000 ($[2, 3, 4, 5, 7, 10, 14, 20, 27, 38, 53, 73, 101, 141, 195, 270, 375, 520, 721, 999]$), each with the selection strategies listed in Appendix A.1.3.

To select the best-performing representation–selection configuration and report on independent datasets, the 25 datasets described in Appendix A.2.1 were split into a 5-dataset validation set and a 20-dataset test set prior to running the subforest benchmark. The validation set (CERVICAL CANCER BEHAVIOR RISK, ISOLET, MUSK (VERSION 1), WAVEFORM DATABASE GENERATOR, and HEART DISEASE) was selected to span the main axes of dataset variation summarized in Table 4: sample size ranges from small ($n = 72$, CERVICAL CANCER) to large ($n = 7{,}797$, ISOLET); feature dimensionality ranges from low (13 features, HEART DISEASE) to high (617 features, ISOLET); the number of classes ranges from binary (CERVICAL CANCER, MUSK) to multiclass with few classes (WAVEFORM: 3, HEART DISEASE: 5) to many-class (ISOLET: 26); and class balance ranges from near-perfectly balanced (WAVEFORM, ISOLET) to the more imbalanced end of the pool (HEART DISEASE, 0.795). All representation–selection configurations described in Appendix A.1.3 were first compared on this validation set, and the resulting best-performing configuration was then evaluated on the remaining 20 test datasets; the results reported in Section 4.2 and Appendix A.3.2 refer exclusively to this 20-dataset test set.

A.2.5 RESOURCE BENCHMARK

The resource benchmark evaluates RFs of sizes in $\{5, 10, 15, 20, 25, 30, 50, 100\}$, with DTs restricted to a maximum depth of 10 (for additional parameters see Appendix A.2.3). A single pool of $\max(\text{sizes})+2$ trees is trained once for the entire sweep. For each RF size, a subforest of size$+2$ trees is then sampled from this shared pool, with the two additional trees used for a warm-up pass (one representation-construction and one similarity-computation call per representation) prior to timing, to exclude cold-start effects such as just-in-time compilation or cache population. Representation construction and similarity computation are timed separately via `perf_counter`; peak RAM is tracked via `tracemalloc` and peak VRAM via `torch.cuda.max_memory_allocated`, with garbage collection and CUDA memory statistics reset before each timed operation. All runs are single-threaded to obtain clean, comparable per-operation timings. Although similarity is evaluated over all tree pairs, the reported cost is the average per-pair value. Representation size, finally, is measured as the byte length of the object's pickled serialization.

For each representation and CV fold, we fit a power law relating per-operation cost to forest size via linear regression in log-log space, and report the empirical scaling exponent α averaged across folds. As a measure of how well the power law describes the data, we additionally report the coefficient of determination R^2 of this fit, likewise averaged across folds. Since R^2 quantifies the fraction of variance in log-cost explained by forest size, it is only informative when cost actually varies with forest size. It is therefore reported in Appendix A.3.3 only in these cases, and omitted for near-constant costs.

TF and ID require joint computation across the full tree population and are handled separately. For TF, per-tree feature vectors are computed and timed individually, after which the one-time cost of constructing the shared Mapper graph is measured separately and amortized across all $\binom{\text{size}}{2}$ pairwise comparisons: each pairwise similarity's reported runtime includes this one-time cost divided by $\binom{\text{size}}{2}$, and its reported memory is the maximum of the per-call peak and the one-time construction peak. Because this amortized cost shrinks as $\binom{\text{size}}{2}$ grows, it is responsible for the negative empirical scaling exponent reported for TF's similarity computation in Table 3. For ID, representation construction is a single joint meta-training pass over the full population; its reported per-tree construction cost is this joint training time divided evenly by the number of trees, rather than a cost measured per tree in isolation.

The benchmark was conducted on a high-performance computing cluster using a single node with one NVIDIA A100 GPU (40GB VRAM), 32 CPU cores on AMD EPYC 7662 processors (of which a single thread was used for all timed operations), and 512GB RAM, running Ubuntu Linux (kernel 6.8.0) with Python 3.10.20.

A.3 ADDITIONAL RESULTS

A.3.1 PERTURBATION BENCHMARK

This section provides supplementary evidence for the perturbation sensitivity results reported in Section 4.1 and Table 1. Figure 3 confirms that representation distance generally increases with

perturbation intensity, as expected of a well-behaved representation, though the strength and trend of this relationship varies markedly by both representation and perturbation type: some representations respond sharply and consistently (e.g., PP and FG), while others are structurally insensitive to specific perturbations – most notably TF and ID, whose distance remains exactly zero across all intensities under *change threshold*.

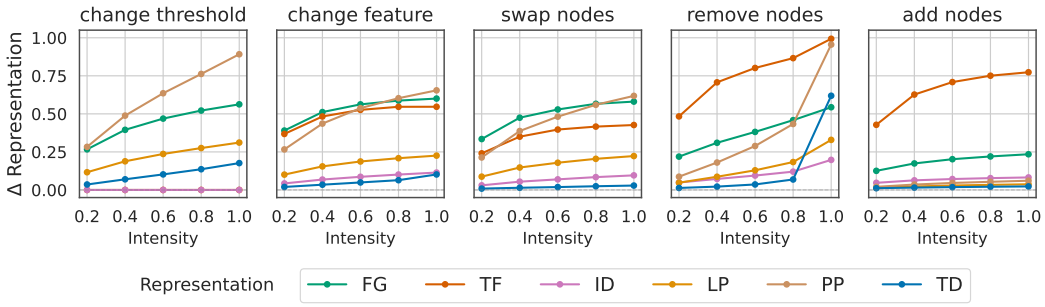


Figure 3: Mean representation distance $1 - s_R$ against perturbation intensity λ, for each of the five perturbation operations, averaged across datasets, folds, and perturbation runs. A representation whose distance increases monotonically and substantially with λ is sensitive to that perturbation. Note: TF and ID overlap under the *change threshold* perturbation.

Beyond intensity, representation distance is also expected to track the functional consequences of a perturbation, namely its effect on predictive performance (MCC) and FI; a well-behaved representation should therefore show a positive correlation between representation distance and these functional-change measures. Figure 4 exemplifies this relationship for predictive performance, plotting representation distance (Δ representation) against the corresponding change in MCC (Δ MCC). Correlation coefficients for both functional-change measures are reported in Table 1. Figure 5 then assesses whether these differences in correlation among representations are statistically robust via critical difference diagrams. In most cells, a top group is separated from the rest: for FI, LP is the sole best representation for *add nodes*, *change threshold*, and *remove nodes*, while for *change feature* and *swap nodes* TD leads but is statistically tied with LP and PP. For performance, PP is more often in the top group, leading for *change threshold* and tied for best in *change feature* and *swap nodes*. Across both measures, TF and ID are never in the top group when present, and are excluded entirely from *change threshold* given their exact insensitivity to it.

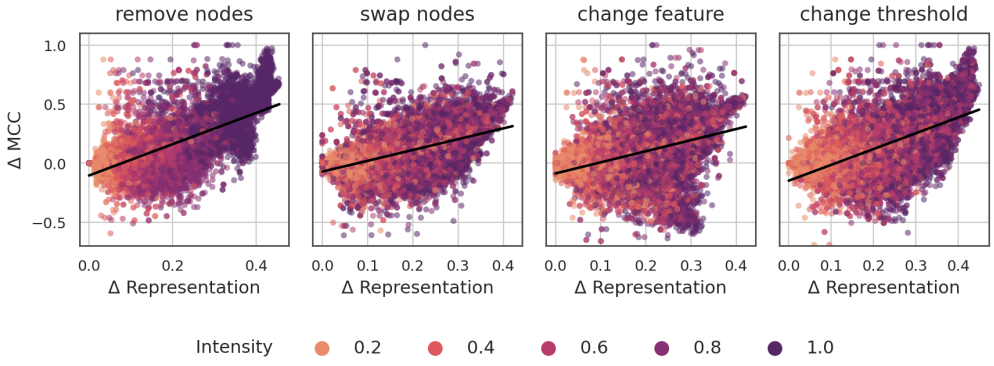


Figure 4: Representative scatter plots of the raw distance–prediction relationship underlying Table 1, for LP across four representative perturbations. Each point is one (base tree, perturbed tree) pair; the x-axis shows the representation distance $\Delta = 1 - s_R$, and the y-axis shows the corresponding change in predictive performance, $\Delta\mathrm{MCC} = \mathrm{MCC}_{\mathrm{base}} - \mathrm{MCC}_{\mathrm{perturbed}}$.

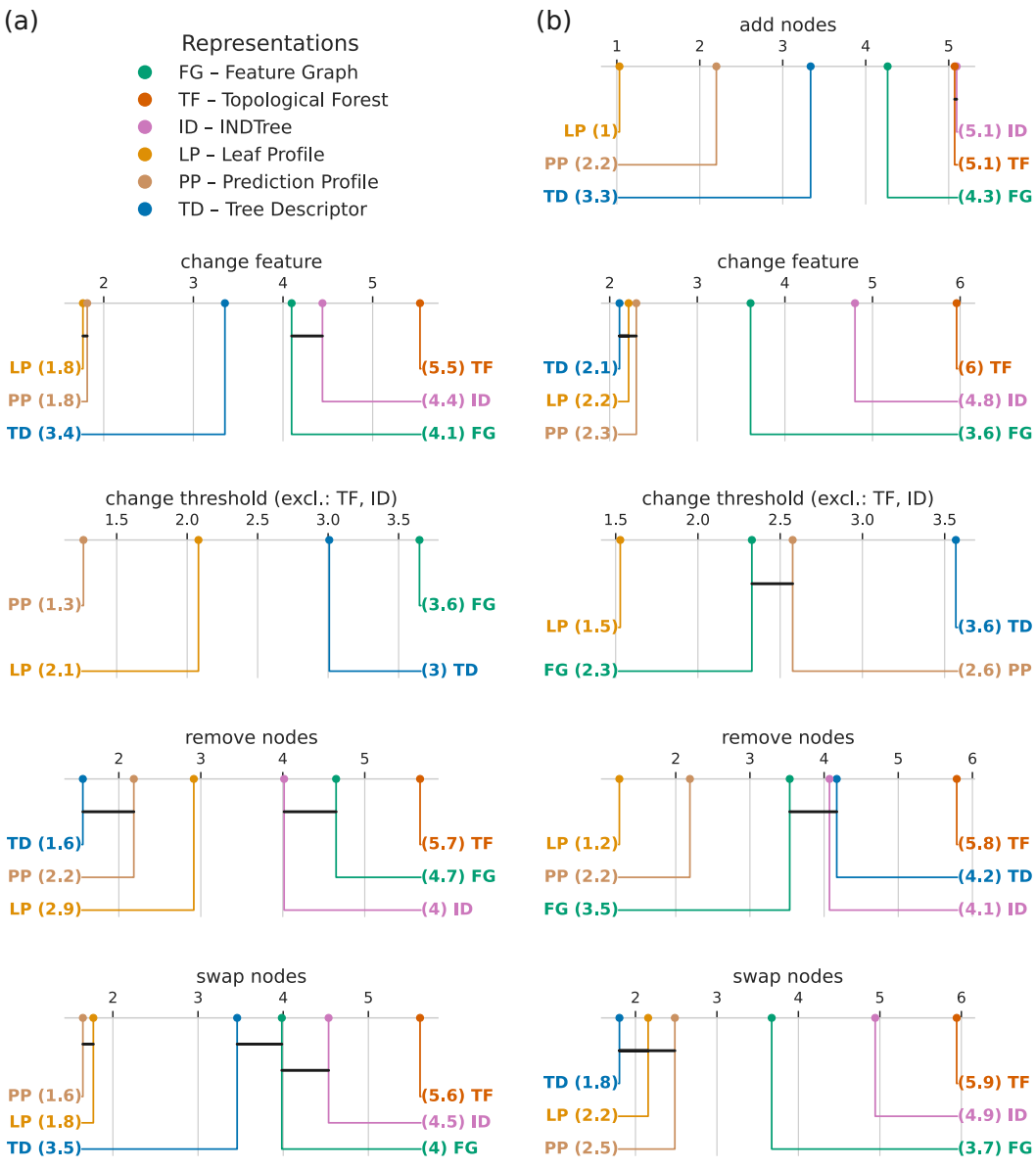


Figure 5: Critical difference diagrams for each **(a)** (r_{MCC}, perturbation), and **(b)** (r_{FI}, perturbation) cell. Representations are ordered by average rank (lower rank = higher correlation between representation distance and the corresponding functional-change measure); a horizontal bar connects representations whose average ranks are not significantly different according to the Nemenyi post-hoc test with significance at 0.05. For (a), *add nodes* is omitted as adding new optimal leaves won't make predictive performance worse and thus won't correlate.

A.3.2 SUBFOREST BENCHMARK

To identify a single selection strategy to carry forward in the comparison of representation-based selection approaches against the baselines, we compare the eight strategies described in Appendix A.1.3 on the 5 validation datasets (Appendix A.2.4). Each strategy is evaluated across representations on two criteria: predictive performance, measured as relative MCC (subforest MCC relative to the full RF's MCC), and agreement with the full forest, measured as the fraction of test samples on which the subforest predicts the same class as the full RF (Figure 6, left column). SA-S achieves the highest average score on the validation datasets for *both* criteria. On the 20 held-out test datasets (Figure 6, right column), SA-S is no longer the single best strategy on either metric, but remains competitive: it ranks 3rd of 8 for performance and 2nd of 8 for prediction agreement.

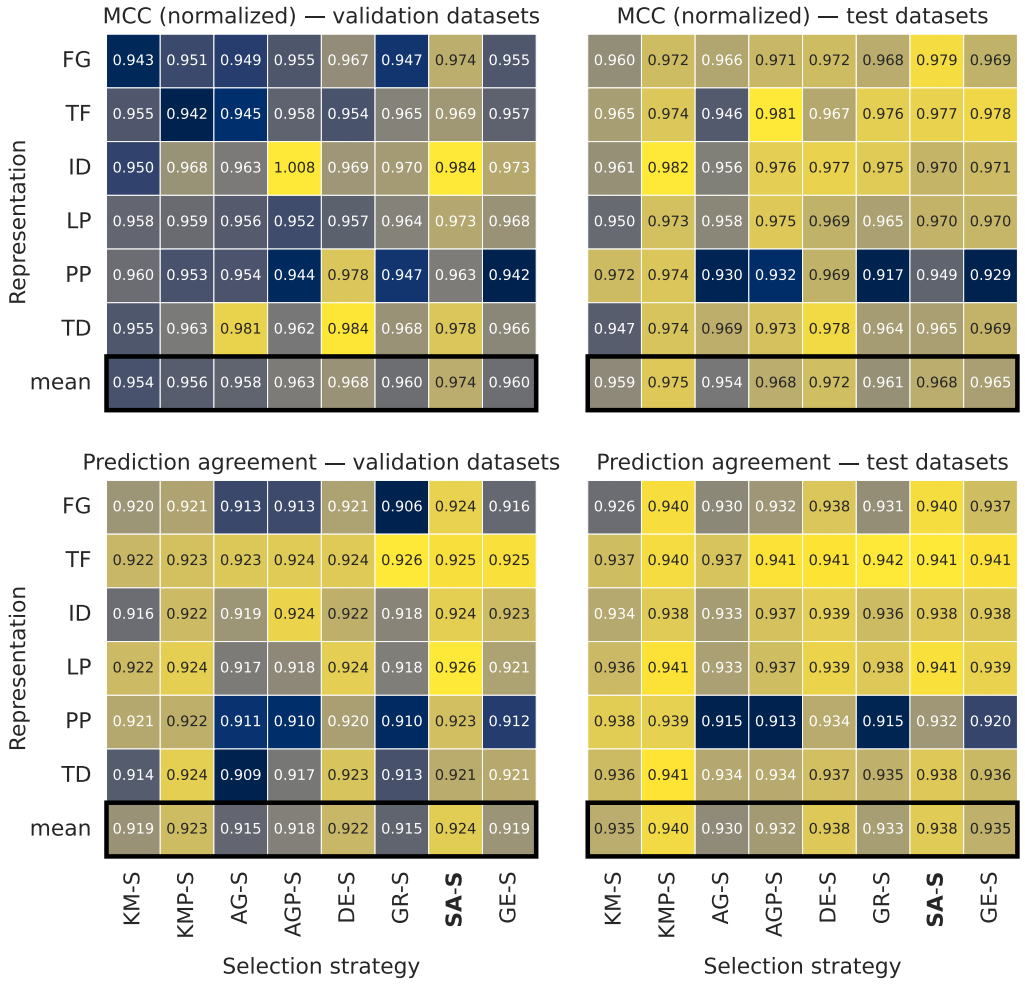


Figure 6: Heatmaps of subforest selection strategy scores per representation, with averages across representations, on the validation datasets (left column) and test datasets (right column), over two criteria: predictive performance relative to the full forest as relative MCC (top row) and average agreement with full forest predictions (bottom row).

To determine the compression efficiency enabled by each representation, we determine, for each representation and baseline, the smallest subforest size k that retains a given fraction of the full RF's MCC. Figure 7a reports this minimum k, averaged across CV folds and test datasets, across eleven thresholds ranging from 90% to 100%. At lenient thresholds ($\leq 92\%$), the heuristic baselines are the most efficient: TOM requires only 4 trees on average to reach 90% of full RF performance, fewer than any representation (the closest, TF, requires 5). This advantage reverses as the threshold tightens: by 100% (matching the full RF performance), TOM requires 229 trees on average, more

than any representation, while TD requires only 84, the fewest of any method. Figure 7b summarizes this threshold-dependent trade-off into a single score per method: the area under its mean-min-k-versus-threshold curve. This area is computed on a log scale, so that the few datasets requiring very large subforests do not dominate the score. By this measure, TD achieves the best (lowest) overall score, followed closely by ID and TF, while TOM ranks only fourth and RDM performs worst (highest score). Figure 8 reports inference time at the 99% threshold, complementing the minimum-subforest-size results of Figure 2, and confirms that TD offers the lowest inference time of any method, staying below 40% of the full RF's inference time on every test dataset.

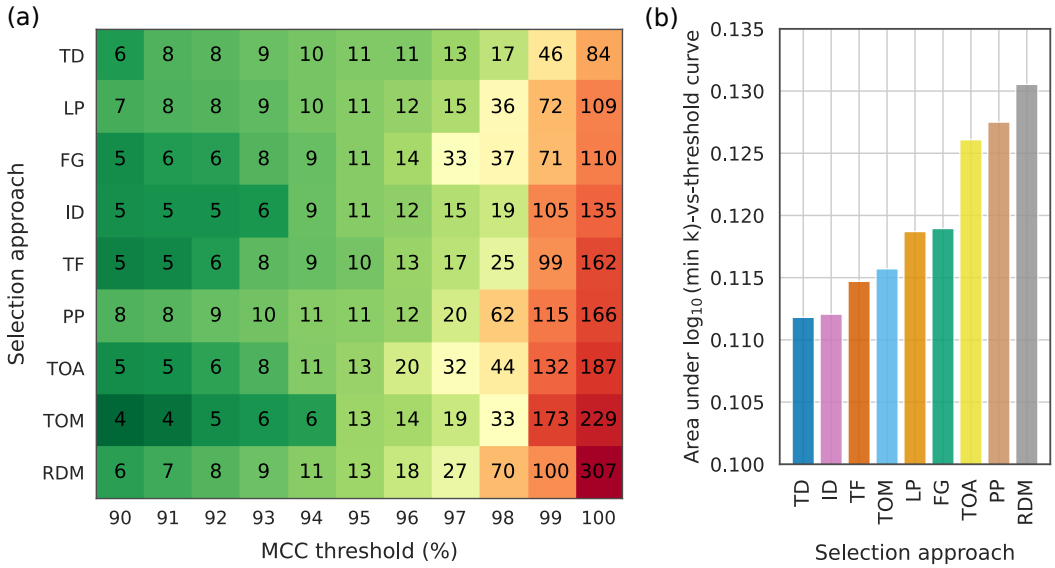


Figure 7: **(a)** Minimum subforest size k required to reach a target fraction of full RF MCC, averaged across the 20 test datasets, reported for each representation-based (TD, LP, FG, ID, TF, PP) and baseline (TOA, TOM, RDM) selection approach, over MCC thresholds from 90% to 100%. **(b)** Area under the $\log_{10}(k)$-versus-threshold curves from the values shown in (a), summarizing compression efficiency across the full threshold range; lower values indicate that fewer trees are required, on average, to reach a given performance threshold.

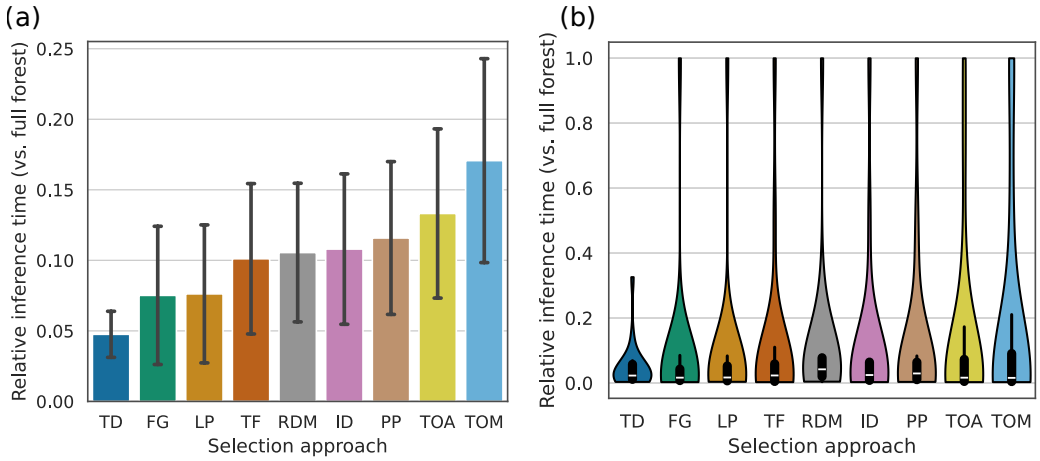


Figure 8: Wall-clock inference time of each representation's / baseline's smallest subforest that recovers at least 99% of the full RF's MCC, relative to the full RF's own inference time on the same dataset, across the 20 held-out test datasets. **(a)** Mean $\pm$ standard error across datasets, representations sorted by increasing mean. **(b)** Full per-dataset distribution.

A.3.3 RESOURCE BENCHMARK

Figure 9 shows per-operation cost across forest sizes from 5 to 100 trees, complementing the summary in Table 3. For representation construction, TD, LP, PP, FG, and TF all show an empirical scaling exponent $\alpha \approx 0$ for both runtime and peak RAM ($|\alpha| \leq 0.04$ throughout), confirming that per-tree construction cost is independent of forest size. ID is the exception: its construction runtime ($\alpha = 0.34$, $R^2 = 0.35$), peak RAM ($\alpha = 1.11$, $R^2 = 0.88$), and peak VRAM ($\alpha = 0.73$, $R^2 = 0.85$) all grow with forest size, and it is the only representation using the GPU during construction. Similarity computation is markedly cheaper and close to size-invariant for almost the same five representations ($|\alpha| \leq 0.05$ for runtime, $|\alpha| \leq 0.001$ for peak RAM). TF is again the exception: its per-pair runtime falls steeply with forest size ($\alpha = -1.81$, $R^2 = 0.98$) while its peak RAM grows ($\alpha = 0.66$, $R^2 = 0.63$), consistent with the fixed one-time cost of constructing its shared representation being amortized over a growing number of pairwise comparisons.

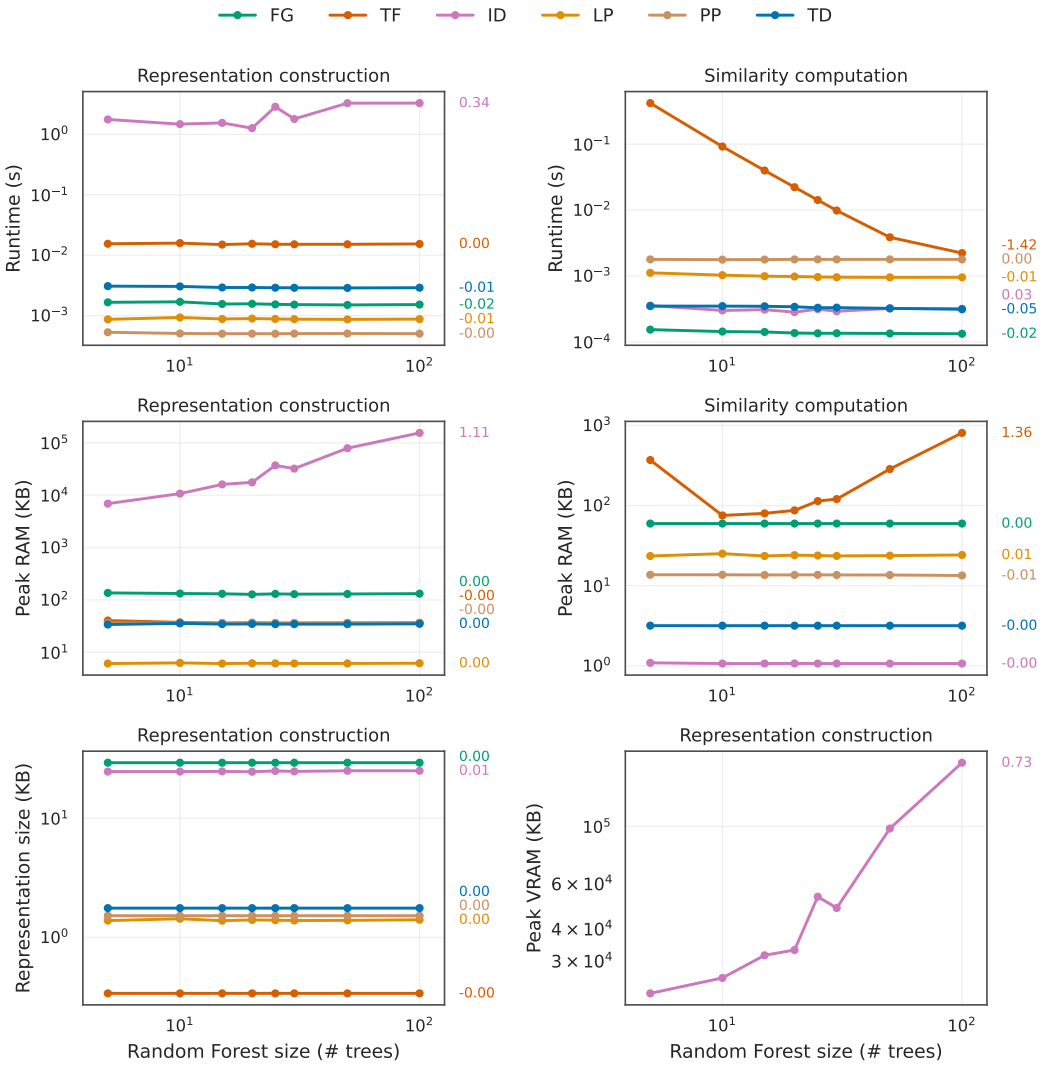


Figure 9: Per-operation resource cost of each tree representation as a function of RF size, decomposed into representation construction and similarity computation, reported as runtime, peak RAM, representation size, and peak VRAM. Measurements are conducted on the CONNECTIONIST BENCH dataset and averaged over 5 seeds $\times$ 5 folds; all axes are log-log. The number beside each curve is its empirical scaling exponent α, the slope of a log-log linear fit of per-operation cost against forest size: $\alpha \approx 0$ indicates a constant per-operation cost.