ITC 2/55
Information Technology and Control
Vol. 55 / No. 2/ 2026
pp. 687-710
DOI 10.5755/j01.itc.55.2.43513

MultiPruneEnsemble: Integrating Multi-Base Decision Tree Prunings via Gradient-Based Weighting

Received 2025/11/04 | Accepted after revision 2026/03/20

HOW TO CITE: Yin, H. (2026) MultiPruneEnsemble: Integrating Multi-Base Decision Tree Prunings via Gradient-Based Weighting. *Information Technology and Control*, 55(2), 687-710. https://doi.org/10.5755/j01.itc.55.2.43513

MultiPruneEnsemble: Integrating Multi-Base Decision Tree Prunings via Gradient-Based Weighting

Hanyang Yin
Shenyang Normal University, College of International Business, Shenyang, 110000, China

Corresponding author: HanyanghyYin@outlook.com

The decision tree is widely used due to its interpretability. However, it usually leads to overfitting, which limits the accuracy of their prediction. Surprisingly, research on integrating these two approaches remains somewhat limited. In this paper, we propose a new framework, MultiPruneEnsemble, which integrates the diversity of pruning with the flexibility of gradient-based optimization. We train multiple independent base trees and apply several advanced pruning techniques to each, aiming to get a set of structurally different but functionally similar subtrees. Then we integrate these pruned models into an ensemble and use gradient descent to learn their weights so that the contribution of each tree is adapted to the data. To validate our algorithm, we ran it on multiple benchmark data sets and compared it with single-pruning baselines, simple averaging ensembles, and other standard tree models. Across all test datasets, our strategy yielded consistent gains in precision, F1 score, and AUC, with performance remaining stable on both large-scale and small-scale tasks. This indicates that integrating multiple decision tree pruning by gradient-based weighting is a feasible way to improve the accuracy and robustness of learning. We release our code and experiment logs to support reproducibility (see the Code Availability statement).

KEYWORDS: Decision tree; Pruning diversity; Ensemble learning; Gradient-weighted optimization; Model generalization; Tabular data

1. Introduction

Tabular data is ubiquitous in industrial and scientific applications. This is critical in high-stakes domains such as finance and healthcare, where models must be auditable. In these settings, interpretability and stable behaviour are often as important as raw accuracy. Decision trees are attractive because their

rules are easy to inspect and explain. Tree-based models outperform deep learning on medium-sized tabular datasets [14], [33] and are easier to deploy and audit. Improving tree models can directly benefit real-world systems that require transparency. Therefore, studying better pruning and integration strategies for decision trees remains meaningful.

Single decision trees are interpretable but tend to overfit. Pruning (e.g., cost-complexity in CART, pessimistic in C4.5) addresses this by trading model size for generalization [6], [31].

Ensemble methods like Bagging (building trees on different samples) and Boosting (sequentially correcting errors) improve performance [5], [8], [13].

Most studies treat pruning as a "housekeeping" step for single trees or ensembles, with little research on combining multiple pruned variants using learned weights [6], [31], [11], [13]. Gradient-based approaches focus on single trees or fully grown ensembles, incurring high costs and limiting diversity [24], [25]. FIPE compresses ensembles by removing redundant learners but does not learn data-driven weights [10]. We lack a unified approach for combining multiple pruned variants with validation-driven weight learning.

In this paper, we propose MultiPruneEnsemble. It generates multiple pruned versions from base trees, retaining them as candidate learners with non-negative weights. Weights are optimized via gradient descent with SoftMax parameterization, adaptively adjusting each subtree's contribution.

2. Related Work

Classical algorithms like CART and C4.5 use greedy splitting [24], achieving only local optima. Recent advances address this through four paradigms.

2.1. Efficiency-oriented Scalable Boosting

XGBoost, LightGBM, and CatBoost address GBDT scalability challenges [8], [19], [30].

XGBoost features sparsity-aware splits and system optimizations for efficient large-scale training [8]. LightGBM enhances GBDT for extensive, high-dimensional data by using GOSS to retain large-gradient samples and EFB to merge features, reducing data size while maintaining accuracy. It is up to 20× faster than standard GBDT, with leaf-wise growth ensuring strong results [19]. CatBoost handles categorical features via ordered boosting to prevent data leakage [30].

Global optimization methods (2017-2022) address greedy splitting limitations.

2.2. Global-optimal but NP-hard Exact Trees

Global-optimal methods (OCT, Branch-and-bound, MurTree) seek to find absolute optima, valuable in finance and medicine [3], [1], [9].

OCT (2017) uses mixed-integer optimization to build globally optimal trees, outperforming CART [3]. DL8.5 (2020) uses itemset caching with Branch-and-Bound for faster learning [1]. Demirovic et al. (2022) presented an algorithm named MurTree (2022) that uses dynamic programming for faster processing of larger datasets [9]. Subsequently, it was demonstrated that learning decision trees with queries in a proper way is NP-hard, i.e., the problem is computationally difficult [21] by Koch et al. (2023).

Optimal trees improve interpretability but are NP-complete [3], [1], [9], [21], [17].

This limitation motivates differentiable relaxations that maintain greedy speed while avoiding local optima.

2.3. Differentiable Soft Trees and Deep-tabular Hybrids

NODE (2019) trains tree-like structures with gradients, often outperforming gradient boosting [29], [18]. TabTransformer (2020) uses self-attention for categorical feature embeddings [16]. Similarly, Arik and Pfister (2021) published TabNet (2021), which applies sequential attention for interpretable feature selection [2].

These models can overfit on small datasets and produce less interpretable rules [34]. This difficulty has led to the emergence of novel strategies that are focused on integrating the hard splits with gradient-based training. *GBDT* This difficulty has led to the emergence of novel strategies that are focused on integrating the hard splits with gradient-based training.

2.4. Recent End-to-End Gradient-based Reparameterizations

Soft splits undermine interpretability. Hard splits with gradient training require handling non-differentiable step functions. GradTree uses Straight-

Through estimators for end-to-end optimization [24]; GRANDE extends this to ensembles with sample-specific weights [25].

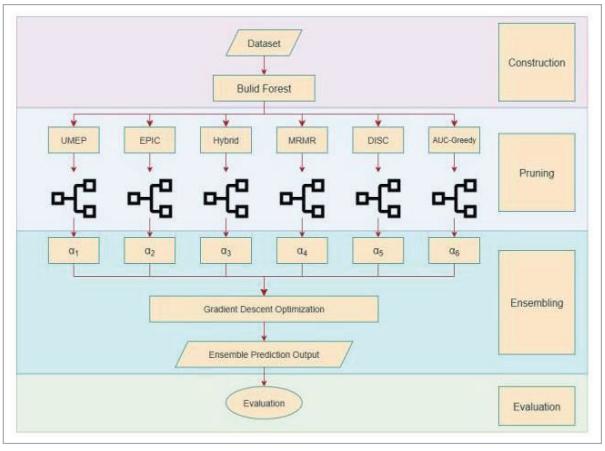
However, these approaches have limitations: GradTree uses post hoc pruning of full trees, which creates redundancy [24]. GRANDE operates at the whole-tree level, incurring high computational cost [25]. In addition, the two approaches consider the entire tree as the primitive learner and do not exploit the structural diversity of pruned subtrees [25]. This discrepancy both triggers neglect of the structural diversity of pruned subtrees [25] and motivates research on ensemble pruning.

2.5. Ensemble Pruning and Functionally-Identical Compression

Six pruning strategies have been compared: MRMR, DISC, AUC-Greedy, EPIC, UMEP, and Hybrid [20]. MRMR eliminates redundancy via mutual information [7]. DISC focuses on individual classifier strength [7]. AUC-Greedy maximizes ROC area [12]. EPIC maximizes ensemble diversity [22]. Li Guo, Samia Lu, and their colleagues developed EPIC to address diversity, ensuring that the chosen trees would provide complementary information [22]. UMEP ranks trees by margin scores [15]. Hybrid balances UMEP and EPIC [26]. FIPE compresses ensembles without changing predictions [10]. Hybrid combined UMEP with EPIC, striking a balance between accuracy and complexity [26]. Most recently, Emine and co-authors advanced FIPE, a functionally identical pruning approach that compresses ensembles without changing their predictions, thereby reducing model size and inference cost [10].

Our work differs in that it focuses on subtree-level diversity across multiple pruning strategies and employs validation-driven fusion.

Figure 1
Framework of MultiPruneEnsemble



3. Methodology

The key mathematical notations used in this paper are listed in Table 1.

Table 1
Notations Used in this paper.

Symbol	Description	Default
d_{max}	maximum depth of a single tree	∞ (no limit)
n_{min}	min samples in any leaf	1
M	total number of trees in the forest	128
K	number of trees after pruning	16
ρ_f	feature sub-sample ratio	$\sqrt{d}/d$
ρ_s	instance sub-sample ratio	1.0
ξ	random seed	42

MultiPruneEnsemble integrates multiple pruned subtrees with gradient-based weighting (Figure 1). We construct a random forest (3.1), generate diverse subtrees via pruning (3.2), and combine them using equal averaging or gradient-based weighting (3.3).

3.1. Base Forest Construction

Here, we construct several parent decision trees T_0 by using the CART algorithm. The trees are trained on the full training set D_{train}, where each split is determined by maximizing Information Gain measured by Gini impurity. To control complexity, we restrict the maximum depth d_{max} and the minimum number of samples per leaf n_{min}. Formally, the training process can be represented as

$$T_0 = TrainCART(D_{train}, d_{max}, n_{min}). \quad (1)$$

The resulting parent tree T_0 serves as the source model, from which multiple pruned subtrees will be derived in Section B.

Algorithm 1 TrainForest (D_{train})

Require: $D_{train}, M, d_{max}, n_{min}, \rho, \rho_s, \xi$
Ensure: Random forest $F=\{T_1,...,T_M\}$
1: initialize random state S with seed ξ
2: **for** m=1 **to** M **do**
3: $D_m \leftarrow$BootstrapSample(D_{train}, ρ_s, S)
4: $V_m \leftarrow$ sample $\lfloor \rho_f \cdot |V| \rfloor$ features w/o replacement
5: $T_m \leftarrow$ As shown in Eq. (1)
6: **end for**
7: **return** F

3.2. Pruning Strategies

We apply the six pruning methods from Section 2.5:

1 **UMEP:** UMEP, which is composed of Uncertainty Margin and Ensemble Pruning, takes the margin gain and diversity into account. Given the dataset S and candidate trees i , the comprehensive process of UMEP can be defined as (2). The first term of (2) is called Marginal Gain. It indicates how much the model's ability to distinguish between samples has improved. The margin gain of the new tree i can be defined as (3). The second term of (2) is called diversity. It illustrates the degree of divergence between the new tree and the selected set, thereby avoiding the simultaneous selection of similar trees. Their average Discrepancy Rate is defined in (4).

$$score(i \mid S) = \Delta m_i(S) + \beta_{div}\, d_i(S), \tag{2}$$

$$\Delta m_i(S) = M(S \cup \{i\}) - M(S), \tag{3}$$

$$d_i(S) = \frac{1}{|S|} \sum_{j \in S} 1\left[\hat{y}_i \neq \hat{y}_j\right]. \tag{4}$$

According to the score function defined in (2), the complete pruning procedure of UMEP is summarized in Algorithm 2.

2 **EPIC:** EPIC (Ensemble Pruning via Individual Contribution and diversity Constraint) first selects the trees that perform best individually on the pruning set according to the macro AUC, as defined in (5), thereby forming a candidate pool C. Then, it chooses K trees from this pool to maximize the total AUC while satisfying the diversity constraint, as shown in (6). Finally, the average divergence of tree i with the selected set S is defined in (7), where the constraint $d_i(S) \geq \delta$ ensures that each newly selected tree maintains sufficient diversity from the existing ones. Here, δ is a preset divergence threshold.

$$C = \; Top\text{-}(\gamma K)\; trees\; ranked\; by\; AUC_i\;, \tag{5}$$

$$S^* = arg \max_{S \subseteq C \,|S|=K} \left\{ \sum_{i \in S} AUC_i \right\}, \tag{6}$$

$$d_i(S) = \frac{1}{|S|} \sum_{j \in S} 1\left[\hat{y}_i \neq \hat{y}_j\right]. \tag{7}$$

Algorithm 2 UMEP Pruning

Require: Predictions on pruning set $\{P_i^p\}_{i=1}^M$, hard predictions $\{\hat{y}_i^p\}_{i=1}^M$, test predictions $\{P_i^t\}_{i=1}^M$, number of trees K, diversity weight β_{div}.
Ensure: Index set S with $|S|=K$.
1: Initialize $S \leftarrow \emptyset$, used $[i] \leftarrow$ False for all i
2: Initialize cumulative sum $\Sigma \leftarrow 0$, margin $m \leftarrow -\infty$
3: **for** $k = 1$ to K **do**
4: $best\_i \leftarrow$ None, $best\_score \leftarrow -\infty$
5: **for** $i = 1$ to M **do**
6: **if** used[i] **then**
7: **continue**
8: **end if**
9: **if** $S = \emptyset$ **then**
10: $\bar{P} \leftarrow P_i^p$, $m_{base} \leftarrow -\infty$
11: **else**
12: $\bar{P} \leftarrow (\Sigma + P_i^p)/(|S| + 1)$
13: $m_{base} \leftarrow m$
14: **end if**
15: $\Delta m_i \leftarrow$ AvgMargin($\bar{P}$) − (0 if $m_{base} = -\infty$ else m_{base})
16: $d_i \leftarrow 0$ if $S = \emptyset$ else $\frac{1}{|S|}\, \Sigma_{j \in S}\, 1[\hat{y}_i^p \neq \hat{y}_j^p]$
17: $s_i \leftarrow \Delta m_i + \beta_{div} \cdot d_i$
18: **if** $s_i > best\_score$ **then**
19: $best\_score \leftarrow s_i$, $best\_i \leftarrow i$
20: **end if**
21: **end for**
22: **if** $best\_i$ = None **then**
23: **break**
24: **end if**

25: $S \leftarrow S \cup \{best\_i\}$, used [$best\_i$] ← True
26: $\Sigma \leftarrow \Sigma + P^p_{best\_i}$, $m \leftarrow$ AvgMargin($\Sigma/|S|$)
27: **end for**
28 **if** $S = \emptyset$ **then**
29: Select $i^* = \arg\max_i$ AvgMargin(P^p_i)
30: $S \leftarrow \{i^*\}$
31: **end if**
32: **return** S

The overall pruning procedure of EPIC, following the definitions in (5)-(7), is summarized in Algorithm 3.

3 **Hybrid:** Hybrid, essentially, is a weighted fusion of EPIC and UMEP. The EPIC part evaluates the strength of an individual tree by macro AUC, as shown in (8), while the UMEP part considers both margin gain and diversity, as defined in (9). Finally, the overall score is obtained through a weighted summation, as given in (10). The Hybrid approach is flexible, as it allows balancing individual performance and ensemble diversity by adjusting the weights, thereby combining the advantages of both. In practice, the weights are typically set to $w_{epic} = w_{umep} = 0.5$.

$$s_i^{EPIC} = AUC_i, \tag{8}$$

$$s_i^{UMEP} = \Delta m_i(S) + \beta_{div}\, d_i(S), \tag{9}$$

$$s_i = w_{epic}\, s_i^{EPIC} + w_{umep}\, s_i^{UMEP}. \tag{10}$$

Algorithm 3 EPIC Pruning

Require: Predictions on pruning set $\{P^p_i\}_{i=1}^M$, hard predictions $\{\hat{y}^p_i\}_{i=1}^M$, test predictions $\{P^t_i\}_{i=1}^M$, number of trees K, multiplier γ (Top-γK), minimum diversity threshold δ.

Ensure: Index set S with $|S|=K$.

1: Compute AUC_i = MacroAUC(P^p_i, y_{prune}) for $i = 1, ..., M$
2: Sort indices by descending AUC_i
3: Let candidate pool C be the top $M' = \min(M, \gamma K)$ trees
4: Initialize $S \leftarrow \emptyset$
5: **for** each $i \in C$ in order **do**
6: **if** $S = \emptyset$ **then**
7: $S \leftarrow \{i\}$
8: **else**
9: $d_i \leftarrow$ As shown in Eq. (7)
10: **if** $d_i \geq \delta$ **then**
11: $S \leftarrow S \cup \{i\}$
12: **end if**
13: **end if**
14: **if** $|S| \geq K$ **then**
15: **break**
16: **end if**
17: **end for**
18: **if** $|S| < K$ **then**
19: **for** each $i \in C$ in order **do**
20: **if** $i \notin S$ **then**
21: $S \leftarrow S \cup \{i\}$
22: **end if**
23: **if** $|S| \geq K$ **then**
24: **break**
25: **end if**
26: **end for**
27: **end if**
28: **return** S

4 **MRMR:** The main idea of MRMR is to maximize relevance while minimizing redundancy. Its optimization objective is defined in (11). Specifically, $R(S)$ denotes the relevance term, which is computed as in (12) by averaging the mutual information between each candidate tree prediction P_i and the pruning labels y_{prune}. This ensures that the selected trees provide valuable insights to the target. On the other hand, $D(S)$ denotes the redundancy term, which is defined in (13) as the average mutual information between the predictions of the selected trees, thus discouraging the choice of trees that are too similar.

$$S^* = arg \max_{S \subseteq \{1,\ldots,M\} | |S|=K} \big(R(S) - D(S)\big), \tag{11}$$

$$R(S) = \frac{1}{|S|} \sum_{i \in S} I\left(P_i, y_{prune}\right), \tag{12}$$

$$D(S) = \frac{1}{|S|^2} \sum_{i,j \in S} I\left(P_i, P_j\right). \tag{13}$$

Algorithm 4 Hybrid Pruning

Require: Predictions on pruning set $\{P_i^p\}_{i=1}^M$, hard predictions $\{\hat{y}_i^p\}_{i=1}^M$, test predictions $\{P_i^t\}_{i=1}^M$, number of trees K, weights w_{epic}, w_{umep}, diversity weight β_{div}.

Ensure: Index set S with $|S|=K$.

Compute $\text{AUC}_i = \text{MacroAUC}(P_i^p, y_{prune})$ for all i
Initialize $S \leftarrow \emptyset$, used[i] $\leftarrow$ False, cumulative sum $\Sigma \leftarrow 0$, margin $m \leftarrow -\infty$
for $k = 1$ to K **do**
  $best\_i \leftarrow$ None, best_score $\leftarrow -\infty$
  for $i = 1$ to M **do**
    if used[i] **then**
      continue
    end if
    $s_i^{\text{EPIC}} \leftarrow \text{AUC}_i$
    if $S = \emptyset$ **then**
      $\tilde{P} \leftarrow P_i^p$, $m_{base} \leftarrow -\infty$, $d_i \leftarrow 0$
    else
      $\tilde{P} \leftarrow (\Sigma + P_i^p)/(|S| + 1)$
      $m_{base} \leftarrow m$
      $d_i \leftarrow \frac{1}{|S|}\Sigma_{j\in S}\, 1[\hat{y}_i^p \neq \hat{y}_j^p]$
    end if
    $\Delta m_i \leftarrow \text{AvgMargin}(\tilde{P}) - (0 \text{ if } m_{base} = -\infty \text{ else } m_{base})$
    $s_i^{\text{UMEP}} \leftarrow \Delta m_i + \beta_{div} \cdot d_i$
    $s_i \leftarrow w_{epic} \cdot s_i^{\text{EPIC}} + w_{umep} \cdot s_i^{\text{UMEP}}$
    if $s_i >$ *best_score* **then**
      best_score $\leftarrow s_i$, $best\_i \leftarrow i$
    end if
  end for
  if $best\_i$ = None **then**
    break
  end if
  $S \leftarrow S \cup \{best\_i\}$, used [$best\_i$] $\leftarrow$ True
  $\Sigma \leftarrow \Sigma + P_{best\_i}^p$, $m \leftarrow \text{AvgMargin}(\Sigma/|S|)$
end for
if $S = \emptyset$ **then**
  Select $i^* = \arg\max_i \text{AUC}_i$
  $S \leftarrow \{i^*\}$
end if
return S

5 **DISC:** DISC (Diversity and Individual Strength based Criterion) balances individual error and diversity through a weight parameter α. The error rate of the i-th tree is defined in (14), where the summation term corresponds to its classification accuracy. The diversity is defined in (15); when S is empty, we set $d_i(S)$=1 by convention to avoid division by zero and ensure that the first tree can be selected. Finally, the overall score is obtained by weighting the error and diversity with parameter α, as shown in (16).

$$err_i = 1 - \frac{1}{N_{prune}} \sum_{n=1}^{N_{\text{prune}}} 1\left[\hat{y}_{i,n}^p = y_{prune,n}\right], \tag{14}$$

$$d_i(S) = 1, |S| = 0, \quad \left\{\frac{1}{|S|}\sum_{j\in S} 1\left[\hat{y}_i^p \neq \hat{y}_j^p\right], |S| > 0, \right. \tag{15}$$

$$s_i = -\alpha \cdot err_i + (1-\alpha) d_i(S). \tag{16}$$

Algorithm 5 MRMR Pruning

Require: Predictions on pruning set $P \in R^{M \times N_{prune}}$, labels $y_{prune} \in R^{N_{prune}}$, number of trees K.

Ensure: Index set S with $|S| = K$.

Compute relevance $r_i = I(P_i, y_{prune})$, for $i = 1,..., M$
$S \leftarrow \{\arg\max_i r_i\}$; $R \leftarrow \{1, ..., M\} \setminus S$
while $|S| < K$ **and** $R \neq \emptyset$ **do**
  for $i \in R$ **do**
    $d_i \leftarrow \frac{1}{|S|}\Sigma_{j\in S} I(P_i, P_j)$ {redundancy}
    $s_i \leftarrow r_i - d_i$
  end for
  $i^* \leftarrow \arg\max_{i\in R} s_i$
  $S \leftarrow S \cup \{i^*\}$; $R \leftarrow R \setminus \{i^*\}$
end while
return S

Algorithm 6 DISC Pruning

Require: Hard predictions on pruning set $\{\hat{y}_i^p\}_{i=1}^M$, true labels y_{prune}, number of trees K, weight α.

Ensure: Index set S with $|S| = K$.

Compute errors: As shown in Eq. (14)
Initialize $S \leftarrow \emptyset$, $R \leftarrow \{1, ..., M\}$
while $|S| < K$ **and** $R \neq \emptyset$ **do**
  $best\_i \leftarrow$ None, *best_score* $\leftarrow -\infty$
  for each $i \in R$ **do**
    if $S = \emptyset$ **then**
      $d_i \leftarrow 1.0$
    else

$d_i \leftarrow \frac{1}{|S|}\Sigma_{j\in S}\, 1[\hat{y}_i^p \neq \hat{y}_j^p]$
end if
$s_i \leftarrow$ As shown in Eq. (16)
if $s_i > best\_score$ **then**
$best\_score \leftarrow s_i,\ best\_i \leftarrow i$
end if
end for
$S \leftarrow S \cup \{best\_i\},\ R \leftarrow R \setminus \{best\_i\}$
end while
return S

The overall pruning procedure of DISC, following the definitions in (14)-(16), is summarized in Algorithm 6.

Algorithm 7 AUC-Greedy Pruning

Require: Predictions on pruning set $\{P_i^p\}_{i=1}^M$, test predictions $\{P_i^t\}_{i=1}^M$, labels y_{prune}, number of trees K.
Ensure: Index set S with $|S| = K$.
Initialize $S \leftarrow \emptyset$, used[i] $\leftarrow$ False, cumulative sum $\Sigma \leftarrow 0$, current AUC $a \leftarrow -\infty$
for $k = 1$ to K **do**
$best\_i \leftarrow$ None, $best\_auc \leftarrow -\infty$
for $i = 1$ to M **do**
if used[i] **then**
continue
end if
if $S = \emptyset$ **then**
$\bar{P} \leftarrow P_i^p$
else
$\bar{P} \leftarrow (\Sigma + P_i^p) / (|S| + 1)$
end if
$a_i \leftarrow \text{MacroAUC}(\bar{P}, y_{prune})$
if $a_i > best\_auc$ **then**
$best\_auc \leftarrow a_i, best\_i \leftarrow i$
end if
end for
if $best\_i =$ None **then**
break
end if
$S \leftarrow S \cup \{best\_i\}$, used[$best\_i$] $\leftarrow$ True
$\Sigma \leftarrow \Sigma + P_{best\_i}^p,\ a \leftarrow best\_auc$
end for
if $S = \emptyset$ **then**
Select $i^* = \arg\max_i \text{MacroAUC}(P_i^p, y_{prune})$
$S \leftarrow \{i^*\}$
end if
return S

6 **AUC-Greedy:** AUC-Greedy is a greedy pruning method that focuses solely on the macro AUC. At each step, it selects the tree that maximizes the overall macro AUC, without considering diversity or mutual information. As defined in (17), the macro AUC of a subset S is computed by first averaging the predicted probabilities of all trees in the set and then evaluating the macro AUC based on this average prediction. The optimization objective is given in (18), which maximizes the macro AUC of the pruned subset on the validation set.

$$AUC(S) = MacroAUC\left(\frac{1}{|S|}\sum_{i\in S} P_i^p, y_{prune}\right), \quad (17)$$

$$S^* = arg\max_{S\subseteq\{1,\ldots,M\}|S|=K} AUC(S). \quad (18)$$

The overall pruning procedure of AUC-Greedy, following the definitions in (17)-(18), is summarized in Algorithm 7.

3.3. Ensembling

We combine pruned subtrees using two strategies:

Algorithm 8 Equal Averaging Ensemble

Require: Predictions from M pruned subtrees $\{p_k(x)\}_{k=1}^M$
Ensure: Fused prediction $\hat{p}(x)$
Stack predictions into tensor $P \in R^{M\times N\times C}$
Compute element-wise mean: As shown in Eq. (19)
return $\hat{p}(x)$

1 **Equal Average:** Equal Averaging computes the mean of probability vectors (Equation 19).

$$\hat{p}(x) = \frac{1}{M}\sum_{k=1}^{M} p_k(x). \quad (19)$$

Here, M denotes the total number of subtrees and $p_k(x)$ represents the probability distribution predicted by the k-th pruned subtree for input sample x.

Equal Averaging is simple but treats all subtrees equally, potentially introducing noise from poor performers.

The overall procedure for the Equal Averaging strategy is summarized in Algorithm 8.

2 **Grad-Weighted:** Grad-Weighted learns weights by minimizing cross-entropy loss on the validation set.

First, we introduce a group of parameters $\theta=(\theta_1,\ldots,\theta_M)$. Then, we use the Softmax function, as defined in (20), to convert them into valid weights $w=(w_1,\ldots,w_M)$, ensuring that the weights are non-negative and sum to 1.

After obtaining the learned weights, we compute the weighted average of the prediction probabilities for all subtrees, which is formally defined in (21), yielding the final integrated prediction $\hat{p}(x)$.

Finally, in order to minimize the cross-entropy loss (LogLoss) on the validation set, we iteratively adjust the parameters θ until reaching the optimal solution θ, as defined in (22).

$$w_k = \frac{e^{\theta_k}}{\sum_{j=1}^{M} e^{\theta_j}},\ k = 1,\ldots,M, \tag{20}$$

$$\hat{p}(x) = \sum_{k=1}^{M} w_k\, p_k(x), \tag{21}$$

$$\theta = \arg\min_{\theta} LogLoss\left(y_{val}, \hat{p}(x)\right). \tag{22}$$

Algorithm 9 Gradient-based Weight Fusion

Require: Predictions from M pruned subtrees $\{p_k(x)\}_{k=1}^{M}$, validation labels y_{val}

Ensure: Fused prediction $\hat{p}(x)$ and weights w

1: Stack predictions into tensor $P \in R^{M\times N\times C}$

2: Initialize parameter vector $\theta \in R^{M}$ with zeros

3: Define softmax $(\theta)_k$ as shown in Eq. (20)

4: Define loss function:

$$L(\theta) = \text{LogLoss}\left(y_{val}, \sum_{k=1}^{M} \text{softmax}(\theta)_k \cdot p_k(x)\right)$$

5: Optimize θ by L-BFGS-B to minimize $L(\theta)$

6: Obtain final weights $w = \text{softmax}(\theta^*)$

7: Compute fused prediction: As shown in Eq. (21)

8: **return** $\hat{p}(x)$, w

To further clarify the implementation, the detailed steps of the Grad-Weighted strategy are presented in Algorithm 9.

3.4. Evaluation Metrics

We evaluate using ACC, F1-score, and AUC.

1 **ACC:** Accuracy measures the proportion of correctly classified samples among all samples, which can be defined as in (23) and (24). Here, *TP*, *TN*, *FP*, *FN* refer to True Positives, True Negatives, False Positives, and False Negatives, respectively. Despite being intuitive and easy to interpret, it can be misleading when the dataset is highly imbalanced [28], [4].

$$ACC = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ samples} \tag{23}$$

$$= \frac{TP + TN}{TP + TN + FP + FN} \tag{24}$$

2 wF1-score is the harmonic mean of precision and recall (Equations 25-27).

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{25}$$

$$Precision = \frac{TP}{TP + FP} \tag{26}$$

$$Recall = \frac{TP}{TP + FN} \tag{27}$$

It balances the trade-off between correctly identifying positive samples and avoiding false positives, making it more reliable than ACC in imbalanced classification tasks.

3 AUC is the area under the ROC curve (Equation 28),

$$AUC = \int_0^1 TPR\,(FPR)d(FPR), \tag{28}$$

which can also be interpreted as the probability that a randomly chosen positive sample is assigned a higher score than a randomly chosen negative one. Unlike ACC and $F1$, AUC does not rely on a fixed threshold, making it a robust metric in evaluating the discriminative ability of classifiers.

We use scikit-learn implementations [28].

3.5. Computational Complexity and Efficiency Discussion

MultiPruneEnsemble has three main costs: (i) training the parent forest, (ii) generating pruned subtrees, and (iii) learning fusion weights. Training the parent trees follows the same order as a standard Random Forest under the same depth/leaf constraints. Pruning is applied to already-trained trees, so it is usually cheaper than training new trees from scratch. Weight learning optimizes a K-dimensional parameter vector on the validation split and requires forward passes of K subtrees. Therefore, the main extra cost over single-pruning baselines grows roughly with $O(E \cdot K \cdot |D_{val}|)$, where E is the number of gradient steps.

Wall-clock time can vary across hardware and system load. Since we did not collect a stable runtime log in the initial submission, we focus on hardware-independent efficiency factors (e.g., the number of retained subtrees K, average subtree depth, and number of nodes/leaves), which directly relate to inference computation.

4. Experiment

4.1. Experimental Setup

Table 2 shows the experimental environment.

Table 1 presents the hyperparameters and their default configurations. **In the ablation experiments, we adopt the default values.** In section 4.4, we systematically vary four critical parameters, d_{max}, n_{min}, M, and K, examining their impact on predictive accuracy under different experimental conditions.

We use seeds 41-50, repeating each experiment 10 times.

We focus on pruning-based baselines, positioning MultiPruneEnsemble as an interpretable module for improving pruning-based models.

4.2. Dataset and Hyperparameter

In this work, we adopt benchmark datasets from the PMLB suite, a standardized and curated collection designed for machine learning benchmarking [27], [32]. Table 3 summarizes the datasets used in our experiments. These datasets differ in sample size, feature dimensionality, class distribution, and imbalance ratio, providing a comprehensive basis for evaluating the proposed method. In the following experiments, we examine how these four factors affect the model accuracy (Section 4.3). Together, these datasets form a well-balanced benchmark encompassing diverse data scales, feature complexities, and class distributions, ensuring a robust evaluation of the proposed pruning and ensemble framework.

Table 2
Experimental Environment.

Component	Specification / Version
Hardware	
CPU	Intel Core i9-13900H, 14 cores / 20 threads, 2.60 GHz
Memory	32 GB DDR5, 4800 MT/s
GPU	Intel Iris Xe Graphics, 15.9 GB shared memory
Operating System	Windows 11, 64-bit
Software	
Python	3.8.10
NumPy	1.24.3
SciPy	1.10.1
scikit-learn	1.3.2
PMLB	1.0.1. post3

Table 3
Dataset Statistics Used in the Experiments.

Dataset	Samples	Features	Imbalance	Classes
Adult	48842	14	0.27	2
twonorm	7400	20	0	2
spect	267	22	0.35	2
banana	5300	2	0.01	2
ring	7400	20	0	2
spambase	4601	57	0.04	2
backache	180	32	0.52	2
ionosphere	351	34	0.08	2
mfeat_zernike	2000	47	0	10
segmentation	2310	19	0	7
tokyo1	959	44	0.08	2

Data is split 56.25%/18.75%/25% (train/prune/test) with stratified sampling.

4.3. Ablation Study

We evaluate six single-pruning baselines and two ensemble variants, reporting ACC, F1, and AUC. Best single is underlined, best overall bold, second-best italicized.

Experiments are grouped by data characteristics (Table 4).

Table 4
Experimental Grouping and Corresponding Datasets.

Group	Datasets
A	Adult, twonorm, spect
B	banana, ring, spambase
C	backache, spect, ionosphere
D	mfeat_zernike, segmentation, tokyo1

Table 5
Ablation Results of Group A.

Method	ACC		F1		AUC	
	Mean	Variance	Mean	Variance	Mean	Variance
Dataset: Adult						
MRMR	0.8499	1.17e-5	0.7883	2.21e-5	0.8928	4.45e-6
DISC	0.8499	1.42e-5	0.7884	2.57e-5	0.8928	4.85e-6
AUC-Greedy	0.8488	6.09e-6	0.7869	1.04e-5	0.8929	3.67e-6
EPIC	0.8494	1.09e-5	0.7876	2.26e-5	0.8929	6.07e-6
UMEP	0.8498	7.93e-6	0.7882	1.26e-5	0.8917	8.03e-6
Hybrid	0.8496	9.81e-6	0.7879	2.26e-5	0.8928	6.27e-6
Equal-Avg (6)	0.8537	5.87e-6	0.7892	1.15e-5	0.9012	3.51e-6
Grad-Weighted (6)	0.8545	5.19e-6	0.7894	9.47e-6	0.9018	3.69e-6
Dataset: twonorm						
MRMR	0.9548	8.13e-6	0.9548	8.14e-6	0.9911	2.16e-6
DISC	0.9534	2.35e-5	0.9534	2.35e-5	0.9907	2.42e-6
AUC-Greedy	0.9537	1.45e-5	0.9537	1.45e-5	0.9911	2.69e-6
EPIC	0.9546	2.22e-5	0.9546	2.22e-5	0.9908	2.55e-6
UMEP	0.9519	1.13e-5	0.9519	1.13e-5	0.9905	1.67e-6
Hybrid	0.9546	2.55e-5	0.9546	2.55e-5	0.9908	2.93e-6
Equal-Avg (6)	0.9631	1.88e-5	0.9631	1.88e-5	0.9937	1.58e-6
Grad-Weighted (6)	0.9635	1.43e-5	0.9635	1.43e-5	0.9939	1.47e-6
Dataset: spect						
MRMR	0.8299	1.79e-3	0.7210	2.78e-3	0.8239	3.70e-3
DISC	0.8299	1.50e-3	0.7171	2.65e-3	0.8135	5.48e-3
AUC-Greedy	0.8343	1.71e-3	0.7274	2.95e-3	0.8238	6.08e-3
EPIC	0.8194	2.00e-3	0.7153	3.35e-3	0.8108	6.83e-3
UMEP	0.8224	2.20e-3	0.7141	4.14e-3	0.7967	3.19e-3
Hybrid	0.8194	2.30e-3	0.7179	3.92e-3	0.8110	6.33e-3
Equal-Avg (6)	0.8299	2.09e-3	0.7154	3.83e-3	0.8261	5.37e-3
Grad-Weighted (6)	0.8328	1.33e-3	0.7252	2.34e-3	0.8389	4.41e-3

Table 6
Ablation Results of Group B.

Method	ACC		F1		AUC	
	Mean	Variance	Mean	Variance	Mean	Variance
Dataset: banana						
MRMR	0.8888	1.01e-4	0.8872	1.03e-4	0.9518	3.88e-5
DISC	0.8886	1.11e-4	0.8870	1.14e-4	0.9517	3.44e-5
AUC-Greedy	0.8882	1.17e-4	0.8866	1.19e-4	0.9529	3.50e-5
EPIC	0.8897	5.21e-5	0.8882	5.37e-5	0.9519	3.05e-5
UMEP	0.8896	4.62e-5	0.8881	4.81e-5	0.9512	2.14e-5
Hybrid	0.8900	6.21e-5	0.8884	6.39e-5	0.9522	3.07e-5
Equal-Avg (6)	0.8897	7.68e-5	0.8883	7.81e-5	0.9570	2.41e-5
Grad-Weighted (6)	0.8908	6.42e-5	0.8894	6.59e-5	0.9577	2.25e-5
Dataset: ring						
MRMR	0.9406	5.01e-5	0.9406	5.03e-5	0.9857	9.44e-6
DISC	0.9406	6.18e-5	0.9406	6.19e-5	0.9864	7.29e-6
AUC-Greedy	0.9394	8.28e-5	0.9394	8.32e-5	0.9860	8.62e-6
EPIC	0.9386	4.76e-5	0.9386	4.78e-5	0.9845	1.17e-5
UMEP	0.9341	2.79e-5	0.9340	2.81e-5	0.9838	8.19e-6
Hybrid	0.9391	4.40e-5	0.9390	4.41e-5	0.9846	1.05e-5
Equal-Avg (6)	0.9448	5.10e-5	0.9448	5.11e-5	0.9886	5.46e-6
Grad-Weighted (6)	0.9457	4.38e-5	0.9457	4.39e-5	0.9885	5.13e-6
Dataset: spambase						
MRMR	0.9467	6.46e-5	0.9439	7.26e-5	0.9814	2.20e-5
DISC	0.9480	6.26e-5	0.9452	7.01e-5	0.9818	2.01e-5
AUC-Greedy	0.9484	7.30e-5	0.9456	8.10e-5	0.9827	2.81e-5
EPIC	0.9474	7.50e-5	0.9446	8.29e-5	0.9810	1.74e-5
UMEP	0.9444	4.19e-5	0.9414	4.60e-5	0.9797	2.34e-5
Hybrid	0.9471	6.62e-5	0.9443	7.29e-5	0.9810	1.82e-5
Equal-Avg (6)	0.9503	6.97e-5	0.9478	7.69e-5	0.9833	2.07e-5
Grad-Weighted (6)	0.9497	6.43e-5	0.9472	7.04e-5	0.9837	2.35e-5

1 **Group A:** Group A focuses on analyzing the impact of data scale on model performance. This group consists of three binary classification datasets with similar feature dimensionalities but significantly different sample sizes. This setting helps to determine whether the model can maintain consistent accuracy when trained on large or limited data volumes, evaluating the scalability and stability of the proposed pruning and ensemble methods across datasets of different sizes.

Both ensembles outperform single pruning methods. Grad-Weighted (6) achieves the highest mean scores with stable variance.

Table 7
Ablation Results of Group C.

Method	ACC		F1		AUC	
	Mean	Variance	Mean	Variance	Mean	Variance
Dataset: backache						
MRMR	0.8644	3.79e-4	0.5247	7.57e-3	0.6842	2.01e-2
DISC	0.8667	5.49e-4	0.5281	8.95e-3	0.6902	1.52e-2
AUC-Greedy	0.8622	1.98e-4	0.4896	3.46e-3	0.6955	7.99e-3
EPIC	0.8733	3.35e-4	0.5616	1.02e-2	0.6846	1.60e-2
UMEP	0.8556	5.76e-4	0.5189	7.88e-3	0.6694	1.43e-2
Hybrid	0.8644	8.18e-4	0.5466	1.16e-2	0.6814	1.50e-2
Equal-Avg (6)	0.8689	8.18e-4	0.5323	9.83e-3	0.7019	1.53e-2
Grad-Weighted (6)	0.8667	5.49e-4	0.5520	8.05e-3	0.7534	1.27e-2
Dataset: spect						
MRMR	0.8299	1.79e-3	0.7210	2.78e-3	0.8239	3.70e-3
DISC	0.8299	1.50e-3	0.7171	2.65e-3	0.8135	5.48e-3
AUC-Greedy	0.8343	1.71e-3	0.7274	2.95e-3	0.8238	6.08e-3
EPIC	0.8194	2.00e-3	0.7153	3.35e-3	0.8108	6.83e-3
UMEP	0.8224	2.20e-3	0.7141	4.14e-3	0.7967	3.19e-3
Hybrid	0.8194	2.30e-3	0.7179	3.92e-3	0.8110	6.33e-3
Equal-Avg (6)	0.8299	2.09e-3	0.7154	3.83e-3	0.8261	5.37e-3
Grad-Weighted (6)	0.8328	1.33e-3	0.7252	2.34e-3	0.8389	4.41e-3
Dataset: ionosphere						
MRMR	0.9295	1.17e-3	0.9234	1.40e-3	0.9660	5.29e-4
DISC	0.9284	1.06e-3	0.9223	1.25e-3	0.9631	5.70e-4
AUC-Greedy	0.9364	6.37e-4	0.9303	8.22e-4	0.9651	3.97e-4
EPIC	0.9330	9.89e-4	0.9272	1.18e-3	0.9677	3.73e-4
UMEP	0.9341	5.97e-4	0.9281	7.25e-4	0.9681	4.03e-4
Hybrid	0.9284	7.19e-4	0.9224	8.63e-4	0.9669	3.84e-4
Equal-Avg (6)	0.9375	7.53e-4	0.9316	9.41e-4	0.9695	3.84e-4
Grad-Weighted (6)	0.9398	8.62e-4	0.9342	1.08e-3	0.9737	3.57e-4

2 **Group B:** Group B examines feature dimensionality impact (2-57 features) with a constant sample size.

Both ensembles perform similarly; Grad-Weighted (6) shows consistent small advantages.

3 **Group C:** Group C assesses robustness to class imbalance (ratios 0.08-0.52).

As shown in Table 7, adaptive weighting benefits small/complex datasets most. Grad-Weighted (6) achieves +3.28pp average AUC gain (+5.79pp on Backache).

4 **Group D:** Group D examines the influence of the number of classes on model performance. It includes three datasets with increasing category complexity: mfeat_zernike (10 classes), segmen-

tation (7 classes), and tokyo1 (2 classes). Unlike the previous binary-focused groups, this setting explores the model's adaptability to multi-class classification problems. By comparing results across different class counts, we aim to determine whether the proposed pruning and ensemble strategies can maintain accuracy and calibration consistency as the decision boundaries become more complex.

As Table 8 demonstrates, Grad-Weighted (6) outperforms Equal-Avg (6) across all metrics with low variance (e.g., +0.92pp ACC on Mfeat_Zernike).

Table 8
Ablation Results of Group D.

Method	ACC		F1		AUC	
	Mean	Variance	Mean	Variance	Mean	Variance
Dataset: mfeat_zernike						
MRMR	0.7672	6.33e-5	0.7633	9.28e-5	0.9575	2.12e-5
DISC	0.7556	1.96e-4	0.7522	1.87e-4	0.9548	1.38e-5
AUC-Greedy	0.7574	1.96e-4	0.7546	2.16e-4	0.9555	5.91e-6
EPIC	0.7582	1.61e-4	0.7537	1.84e-4	0.9565	1.95e-5
UMEP	0.7536	1.69e-4	0.7496	2.17e-4	0.9528	2.47e-5
Hybrid	0.7598	1.84e-4	0.7556	2.08e-4	0.9563	1.38e-5
Equal-Avg (6)	0.7726	2.16e-4	0.7696	2.37e-4	0.9651	6.63e-6
Grad-Weighted (6)	0.7764	1.31e-4	0.7741	1.65e-4	0.9656	5.98e-6
Dataset: segmentation						
MRMR	0.9779	2.18e-5	0.9778	2.26e-5	0.9987	9.41e-7
DISC	0.9779	2.12e-5	0.9778	2.16e-5	0.9985	9.67e-7
AUC-Greedy	0.9765	1.14e-5	0.9764	1.17e-5	0.9987	3.95e-7
EPIC	0.9763	3.73e-5	0.9762	3.81e-5	0.9981	1.53e-6
UMEP	0.9730	4.14e-5	0.9730	4.24e-5	0.9978	1.25e-6
Hybrid	0.9766	3.61e-5	0.9766	3.69e-5	0.9982	1.65e-6
Equal-Avg (6)	0.9789	2.45e-5	0.9788	2.50e-5	0.9991	2.63e-7
Grad-Weighted (6)	0.9784	2.28e-5	0.9783	2.36e-5	0.9992	1.81e-7
Dataset: tokyo1						
MRMR	0.9275	3.75e-4	0.9218	4.24e-4	0.9779	6.71e-5
DISC	0.9275	3.36e-4	0.9218	3.84e-4	0.9771	8.08e-5
AUC-Greedy	0.9271	2.13e-4	0.9214	2.42e-4	0.9766	5.72e-5
EPIC	0.9283	4.24e-4	0.9227	4.69e-4	0.9773	1.16e-4
UMEP	0.9271	4.21e-4	0.9214	4.44e-4	0.9757	1.10e-4
Hybrid	0.9263	3.63e-4	0.9206	3.95e-4	0.9784	8.00e-5
Equal-Avg (6)	0.9287	3.26e-4	0.9228	3.60e-4	0.9800	7.77e-5
Grad-Weighted (6)	0.9321	2.16e-4	0.9264	2.44e-4	0.9811	7.42e-5

Table 9
Overall Summary of Median Improvements (Δ) Across All Datasets and Groups.

Group	AUC Δ		ACC Δ		F1 Δ	
	EA	GW	EA	GW	EA	GW
A	+0.62	+0.99	+0.25	+0.39	-0.09	+0.25
B	+0.41	+0.60	+0.19	+0.24	+0.21	+0.25
C	+0.43	+3.28	-0.26	-0.16	-1.33	-0.27
D	+0.24	+0.40	+0.23	+0.45	+0.25	+0.50

5 **Detailed Discussion of Ablation Results:** Table 9 shows GW consistently outperforms EA, with AUC gains of +0.43pp to +3.28pp across groups.

Significance testing (Table 10) confirms robustness.

Apparently, both ensemble variants consistently outperform the best single pruning method. Equal-Avg (6) achieves median improvements of **+0.00229** in AUC, **+0.00341** in ACC, and **+0.00241** in F1, all of which are statistically significant ($p \leq 0.012$). Grad-Weighted (6) further increases these improvements to **+0.00478** (AUC), **+0.00503** (ACC), and **+0.00504** (F1), also with $p \leq 0.001$. These results confirm that the ensemble approach substantially enhances prediction performance over individual pruning methods.

Besides, when comparing Grad-Weighted (6) with Equal-Avg (6), the advantage of gradient-based weighting is mainly reflected in AUC. The median difference in AUC is **+0.00064** with p=0.012 (statistically significant), while the differences in ACC and F1 are positive but not statistically significant ($p = 0.227$ and $p = 0.065$, respectively). This suggests that the gradient-based weighting primarily improves the model's ranking ability (AUC), while the classification-threshold metrics (ACC and F1) still depend on dataset characteristics.

Grad-Weighted (6) shows higher accuracy with equal or lower variance, making it the default configuration.

GW's consistent, statistically significant improvements establish gradient-based weighting as the core mechanism.

6 **Limitations and External Validity:** This study evaluates 11 datasets; future work will expand to more diverse benchmarks.

4.4. Hyperparameter Experiment

In this section, we conduct a series of experiments to analyze the sensitivity of the proposed model to different hyperparameters. As shown in Table 11, four parameters are considered: the maximum depth (d_{max}), the minimum samples per leaf (n_{min}), the pruning ratio (M), and the number of subtrees in the ensemble (K). These parameters jointly determine the trade-off between model accuracy, compactness, and generalization. In the sensitivity analysis, one parameter is varied at a time while others remain fixed at default values to ensure comparability.

Table 10
Statistical Summary of Median Improvemnts (Δ).

Metric	Median Δ	p-value
Equal-Avg (6) vs. Best Single		
AUC	+0.00229	0.001
ACC	+0.00341	0.012
F1	+0.00241	0.012
Grad-Weighted (6) vs. Best Single		
AUC	+0.00478	0.001
ACC	+0.00503	0.001
F1	+0.00504	0.001
Grad-Weighted (6) vs. Equal-Avg (6)		
AUC	+0.00064	0.012
ACC	+0.00086	0.227
F1	+0.00117	0.065

Experiments use seeds 41-50.

Table 11
Hyperparameter Settings Used in Sensitivity Experiments.

Hyperparameter	Symbol	Values Tested	Default
Maximum Depth	d_{max}	∞,10,20,5	∞
Minimum Samples per Leaf	n_{min}	1, 5, 10	1
Pruning Ratio	M	64, 128, 256	128
Number of Subtrees in Ensemble	K	8, 16, 32	16
Random Seed Range	ξ	41-50	42

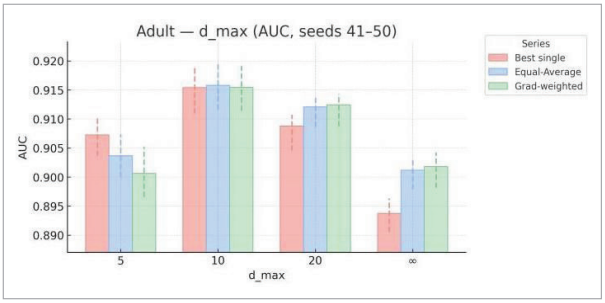
1 **Maximum Depth (d_{max}):** This experiment examines how the maximum tree depth affects the balance between model expressiveness and overfitting.

As summarized in Table 12, the overall trend shows that shallow trees (d_{max} = 5) consistently underfit, resulting in lower AUC across most datasets. Performance improves as the depth increases to a moderate range (d_{max} = 10 – 20), where the model becomes sufficiently expressive without losing generalization ability. When d_{max} is set to ∞ (no restriction on depth), the trees grow until all leaves are pure or other stopping rules are triggered. This unrestricted growth brings no further AUC improvement and often increases variance, indicating mild overfitting.

Figure 2 shows a clear depth-generalization trade-off on Adult. When trees are shallow (d_{max} = 5), all methods underfit, and the best single model is slightly ahead of the ensembles. At a moderate depth (d_{max} = 10), all methods reach their peak and become closely matched. With deeper trees (d_{max} = 20), the ensembles overtake the single model, indicating that averaging/weighting helps control variance. Removing the depth limit (d_{max} = ∞), reduces the single model's performance, while the ensembles remain comparatively stable. The dashed lines denote the AUC range across seeds (max-min) and are narrowest around d_{max} = 10 – 20, confirming this region as the most stable.

Figure 2

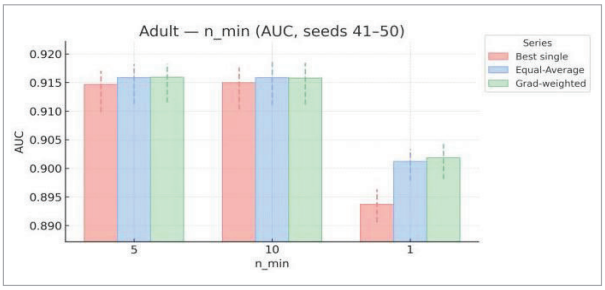
AUC comparison under different d_{max} on Adult.



Overall, a moderate tree depth achieves the best trade-off between accuracy and robustness.

Figure 3

AUC comparison under different n_{min} on Adult.



2 Minimum Samples per Leaf (n_{min}): This experiment analyzes the effect of the minimum number of samples required in each leaf node. The parameter n_{min} controls how strictly the tree is regularized: a smaller value allows deeper and more specific splits, while a larger value enforces stronger regularization by limiting the formation of small leaves.

As summarized in Table 13, the results show that extremely small leaf sizes (e.g., n_{min}=1) cause unstable performance and overfitting, while overly large values reduce the tree's flexibility and lower the overall AUC. Across most datasets, n_{min}=5 achieves the highest and most consistent results, suggesting that a small but nonzero constraint can effectively balance model complexity and stability.

Figure 3 analyzes the effect of leaf size on Adult. Very small leaves (n_{min}=1) reduce AUC and enlarge variation across seeds, showing that overly specific splits capture noise. Moderate leaf sizes (n_{min}=5 or 10) maintain higher and more consistent AUC for all methods, with Grad-weighted slightly above Equal-averaged. The dashed lines indicate the AUC range and are nar-

Table 12

Performance of Grad-Weighted (6) Under Different d_{Max}.

Dataset	d_{max}	ACC		F1		AUC	
		Mean	Var	Mean	Var	Mean	Var
Adult	∞	0.8545	5.19e-06	0.7894	9.47e-06	0.9018	3.69e-06
	10	0.8587	5.48e-06	0.7806	1.12e-05	0.9154	5.81e-06
	20	0.8619	4.36e-06	0.7962	1.13e-05	0.9124	3.59e-06
	5	0.8483	8.29e-06	0.7627	2.70e-05	0.9006	7.44e-06

Dataset	d_{max}	ACC		F1		AUC	
		Mean	Var	Mean	Var	Mean	Var
twonorm	∞	0.9635	1.43e-05	0.9635	1.43e-05	0.9939	1.47e-06
	10	0.9622	2.35e-05	0.9622	2.35e-05	0.9939	1.73e-06
	20	0.9638	1.47e-05	0.9638	1.47e-05	0.9939	1.23e-06
	5	0.9510	5.36e-05	0.9510	5.37e-05	0.9907	4.99e-06
spect	∞	0.8328	1.33e-03	0.7252	2.34e-03	0.8389	4.41e-03
	10	0.8299	2.19e-03	0.7185	5.81e-03	0.8457	4.08e-03
	20	0.8328	1.33e-03	0.7252	2.34e-03	0.8389	4.41e-03
	5	0.8433	2.34e-03	0.7255	6.25e-03	0.8633	2.45e-03
banana	∞	0.8908	6.42e-05	0.8894	6.59e-05	0.9577	2.25e-05
	10	0.8958	4.11e-05	0.8942	4.41e-05	0.9643	1.78e-05
	20	0.8910	7.24e-05	0.8897	7.45e-05	0.9577	2.52e-05
	5	0.8751	9.81e-05	0.8727	1.05e-04	0.9522	2.67e-05
ring	∞	0.9457	4.38e-05	0.9457	4.39e-05	0.9885	5.13e-06
	10	0.9341	4.77e-06	0.9341	4.78e-06	0.9817	6.40e-06
	20	0.9402	5.12e-05	0.9402	5.12e-05	0.9869	5.25e-06
	5	0.9154	8.99e-05	0.9151	9.07e-05	0.9742	1.05e-05
spambase	∞	0.9436	2.25e-05	0.9411	2.41e-05	0.9825	5.56e-06
	10	0.9474	1.77e-05	0.9450	1.86e-05	0.9834	5.15e-06
	20	0.9452	2.12e-05	0.9427	2.22e-05	0.9829	5.39e-06
	5	0.9367	2.58e-05	0.9339	2.70e-05	0.9810	5.96e-06
backache	∞	0.7621	1.93e-04	0.6768	2.45e-04	0.7405	2.33e-04
	10	0.7644	1.70e-04	0.6803	2.21e-04	0.7448	2.11e-04
	20	0.7654	1.76e-04	0.6824	2.26e-04	0.7458	2.17e-04
	5	0.7591	2.07e-04	0.6705	2.52e-04	0.7382	2.36e-04
ionosphere	∞	0.9185	2.06e-04	0.9187	2.10e-04	0.9601	7.64e-05
	10	0.9178	2.05e-04	0.9179	2.07e-04	0.9602	7.58e-05
	20	0.9201	2.00e-04	0.9203	2.02e-04	0.9614	7.51e-05
	5	0.9124	2.15e-04	0.9126	2.16e-04	0.9577	7.84e-05
mfeat_zernike	∞	0.8211	2.23e-04	0.8197	2.29e-04	0.8814	1.17e-04
	10	0.8278	2.05e-04	0.8264	2.10e-04	0.8875	1.09e-04
	20	0.8235	2.16e-04	0.8221	2.20e-04	0.8832	1.14e-04
	5	0.8137	2.47e-04	0.8121	2.52e-04	0.8758	1.23e-04
segmentation	∞	0.9453	1.08e-04	0.9449	1.09e-04	0.9912	2.64e-05
	10	0.9461	1.03e-04	0.9458	1.04e-04	0.9913	2.56e-05
	20	0.9469	1.01e-04	0.9465	1.02e-04	0.9914	2.49e-05
	5	0.9441	1.14e-04	0.9436	1.16e-04	0.9909	2.74e-05
tokyo1	∞	0.8425	3.67e-04	0.8411	3.74e-04	0.9057	2.04e-04
	10	0.8467	3.42e-04	0.8452	3.50e-04	0.9102	1.92e-04
	20	0.8431	3.61e-04	0.8417	3.68e-04	0.9061	2.01e-04
	5	0.8389	3.88e-04	0.8375	3.94e-04	0.9015	2.13e-04

Table 13
Performance of Grad-Weighted (6) Under Different n_{Min}.

Dataset	n_{min}	ACC		F1		AUC	
		Mean	Var	Mean	Var	Mean	Var
Adult	1	0.8612	4.37e-06	0.7953	1.05e-05	0.9125	3.42e-06
	5	0.8567	5.02e-06	0.7841	1.11e-05	0.9082	4.01e-06
	10	0.8534	6.18e-06	0.7779	1.29e-05	0.9026	4.77e-06
twonorm	1	0.9638	1.42e-05	0.9638	1.42e-05	0.9939	1.27e-06
	5	0.9605	2.15e-05	0.9605	2.15e-05	0.9935	1.44e-06
	10	0.9584	2.63e-05	0.9584	2.63e-05	0.9928	1.66e-06
spect	1	0.8427	1.75e-03	0.7249	3.52e-03	0.8531	3.78e-03
	5	0.8372	2.19e-03	0.7206	4.92e-03	0.8486	3.91e-03
	10	0.8310	2.47e-03	0.7184	5.43e-03	0.8447	4.24e-03
banana	1	0.8961	4.22e-05	0.8945	4.56e-05	0.9644	1.82e-05
	5	0.8862	6.28e-05	0.8838	6.63e-05	0.9589	2.21e-05
	10	0.8775	8.37e-05	0.8751	8.82e-05	0.9524	2.48e-05
ring	1	0.9449	4.39e-05	0.9449	4.39e-05	0.9879	5.13e-06
	5	0.9348	6.72e-05	0.9348	6.72e-05	0.9831	7.52e-06
	10	0.9254	8.93e-05	0.9254	8.93e-05	0.9784	9.18e-06
spambase	1	0.9472	1.74e-05	0.9449	1.83e-05	0.9833	5.09e-06
	5	0.9421	2.13e-05	0.9397	2.24e-05	0.9820	5.48e-06
	10	0.9375	2.47e-05	0.9348	2.57e-05	0.9810	5.83e-06
backache	1	0.7638	1.82e-04	0.6792	2.34e-04	0.7431	2.20e-04
	5	0.7612	1.95e-04	0.6748	2.42e-04	0.7393	2.26e-04
	10	0.7579	2.11e-04	0.6712	2.48e-04	0.7358	2.33e-04
ionosphere	1	0.9194	2.04e-04	0.9195	2.07e-04	0.9608	7.58e-05
	5	0.9152	2.13e-04	0.9154	2.14e-04	0.9593	7.71e-05
	10	0.9108	2.22e-04	0.9110	2.25e-04	0.9572	7.85e-05
mfeat_zernike	1	0.8269	2.08e-04	0.8254	2.13e-04	0.8871	1.10e-04
	5	0.8202	2.31e-04	0.8185	2.35e-04	0.8810	1.18e-04
	10	0.8146	2.51e-04	0.8128	2.56e-04	0.8757	1.23e-04
segmentation	1	0.9465	1.02e-04	0.9462	1.03e-04	0.9913	2.52e-05
	5	0.9443	1.12e-04	0.9438	1.14e-04	0.9910	2.68e-05
	10	0.9428	1.18e-04	0.9423	1.19e-04	0.9906	2.75e-05
tokyo1	1	0.8463	3.41e-04	0.8449	3.47e-04	0.9100	1.91e-04
	5	0.8422	3.69e-04	0.8407	3.76e-04	0.9056	2.03e-04
	10	0.8379	3.94e-04	0.8364	4.00e-04	0.9012	2.12e-04

Table 14
Performance of Grad-Weighted (6) Under Different M.

Dataset	M	ACC		F1		AUC	
		Mean	Var	Mean	Var	Mean	Var
Adult	64	0.8536	8.91e-06	0.7884	2.15e-05	0.9000	4.39e-06
	128	0.8545	5.19e-06	0.7894	9.47e-06	0.9018	3.69e-06
	256	0.8552	6.23e-06	0.7900	1.20e-05	0.9025	3.41e-06
twonorm	64	0.9615	2.78e-05	0.9615	2.78e-05	0.9938	8.46e-07
	128	0.9635	1.43e-05	0.9635	1.43e-05	0.9939	1.47e-06
	256	0.9627	2.18e-05	0.9627	2.18e-05	0.9942	1.72e-06
spect	64	0.8328	1.72e-03	0.7194	3.65e-03	0.8341	4.29e-03
	128	0.8328	1.33e-03	0.7252	2.34e-03	0.8389	4.41e-03
	256	0.8299	1.79e-03	0.7175	4.26e-03	0.8409	5.68e-03
banana	64	0.8908	6.67e-05	0.8894	6.83e-05	0.9569	1.44e-05
	128	0.8908	6.42e-05	0.8894	6.59e-05	0.9577	2.25e-05
	256	0.8906	7.99e-05	0.8892	8.26e-05	0.9575	2.09e-05
ring	64	0.9431	4.58e-05	0.9431	4.59e-05	0.9875	6.94e-06
	128	0.9457	4.38e-05	0.9457	4.39e-05	0.9885	5.13e-06
	256	0.9468	2.92e-05	0.9467	2.93e-05	0.9886	5.75e-06
spambase	64	0.9500	3.89e-05	0.9474	4.29e-05	0.9824	2.23e-05
	128	0.9497	6.43e-05	0.9472	7.04e-05	0.9837	2.35e-05
	256	0.9504	5.53e-05	0.9479	6.14e-05	0.9842	1.35e-05
backache	64	0.8778	4.66e-04	0.5456	1.27e-02	0.7376	1.09e-02
	128	0.8667	5.49e-04	0.5520	8.05e-03	0.7534	1.27e-02
	256	0.8667	4.39e-04	0.5317	5.74e-03	0.7603	9.88e-03
ionosphere	64	0.9409	6.54e-04	0.9353	8.39e-04	0.9733	3.62e-04
	128	0.9398	8.62e-04	0.9342	1.08e-03	0.9737	3.57e-04
	256	0.9375	9.25e-04	0.9314	1.16e-03	0.9744	3.29e-04
mfeat_zernike	64	0.7706	1.69e-04	0.7678	2.06e-04	0.9635	1.56e-05
	128	0.7764	1.31e-04	0.7741	1.65e-04	0.9656	5.98e-06
	256	0.7742	3.87e-04	0.7702	4.64e-04	0.9663	7.51e-06
segmentation	64	0.9773	4.02e-05	0.9773	4.09e-05	0.9990	4.08e-07
	128	0.9784	2.28e-05	0.9783	2.36e-05	0.9992	1.81e-07
	256	0.9772	3.05e-05	0.9771	3.08e-05	0.9991	1.46e-07
tokyo1	64	0.9308	3.29e-04	0.9253	3.59e-04	0.9798	7.83e-05
	128	0.9321	2.16e-04	0.9264	2.44e-04	0.9811	7.42e-05
	256	0.9333	3.28e-04	0.9278	3.72e-04	0.9813	6.77e-05

rowest at these moderate settings, highlighting a better balance between flexibility and regularization.

In summary, a small but nonzero leaf size leads to better generalization and stability.

3 **Pruning Ratio (M):** This experiment investigates how the pruning ratio M affects the balance between model compactness and diversity. A smaller M results in stronger pruning, leading to fewer retained subtrees, while a larger M preserves more structure and increases computational cost.

As summarized in Table 14, the overall trend shows that increasing M from 64 to 128 slightly improves AUC and F1, while further enlargement to 256 produces almost no additional benefit. This indicates that excessive retention of subtrees contributes little to accuracy but increases redundancy.

Figure 4 illustrates how the pruning ratio affects AUC on Adult. Increasing M from a small setting to a moderate one improves performance for all methods, with both ensembles consistently above the best single model. Beyond the moderate range, the gains saturate, and the ensembles change little, suggesting that additional retained subtrees contribute marginal diversity. The dashed lines (AUC range over seeds) remain relatively short and do not shrink further at larger M, indicating that pruning primarily influences efficiency once sufficient structure is kept.

Figure 4

AUC comparison under different M on Adult.

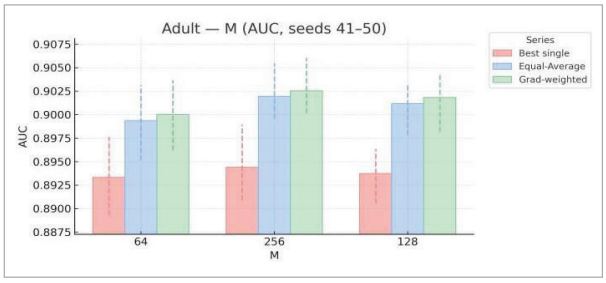
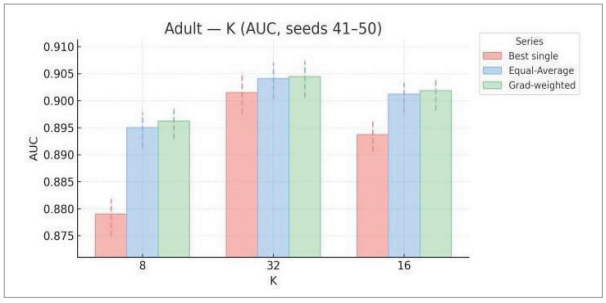


Figure 5

AUC comparison under different K on Adult.



In summary, the pruning ratio primarily influences efficiency rather than accuracy once M exceeds a moderate threshold. A setting around M=128 offers a good compromise between computational cost and predictive stability.

4 **Number of Subtrees (K):** This experiment analyzes how the number of pruned subtrees included in the ensemble (K) influences predictive performance and stability. A small K limits ensemble diversity, while an excessively large K may increase redundancy and computational cost without improving accuracy.

As summarized in Table 15, the overall performance rises when K increases from 8 to 16, and reaches a mild plateau at K=32. This suggests that adding more subtrees initially enhances diversity, but the marginal gain diminishes once the ensemble becomes sufficiently large.

As illustrated in Figure 5, increasing the ensemble size initially improves AUC on Adult. Both ensembles dominate the best single model at all tested values of K, and Grad-weighted (GW) is slightly higher than Equal-averaged (EA) across the board. The gain from enlarging K shows clear diminishing returns: the performance at K=16 is already very close to that at K=32, suggesting that the ensemble becomes sufficiently diverse before the largest setting. The dashed lines indicate the AUC range (max-min) across random seeds; these ranges are consistently narrower for EA and GW than for the best single model, and remain almost unchanged from K=16 to K=32 . Overall, the Adult results support using a moderate ensemble size, with K=16-32 delivering strong accuracy and stable behavior without unnecessary computation.

In conclusion, the number of subtrees mainly affects ensemble diversity and computational efficiency. A moderate configuration of K=16-32 achieves stable and reliable performance across datasets.

5 **Hyperparameter Sensitivity:** Discussion and Takeaway: Key findings: moderate depth (10-20), nonzero leaf size (5-10), M=128, and K=16 provide optimal balance.

Moderate settings provide robust guidelines; automated tuning may further optimize per dataset.

Table 15
Performance of Grad-Weighted (6) Under Different K.

Dataset	K	ACC		F1		AUC	
		Mean	Var	Mean	Var	Mean	Var
Adult	8	0.8521	1.56e-06	0.7866	2.84e-06	0.8962	4.13e-06
	16	0.8545	5.19e-06	0.7894	9.47e-06	0.9018	3.69e-06
	32	0.8555	3.49e-06	0.7906	7.16e-06	0.9044	3.40e-06
twonorm	8	0.9541	1.41e-05	0.9541	1.41e-05	0.9910	1.20e-06
	16	0.9635	1.43e-05	0.9635	1.43e-05	0.9939	1.47e-06
	32	0.9676	1.58e-05	0.9676	1.58e-05	0.9953	7.89e-07
spect	8	0.8254	1.59e-03	0.7137	2.28e-03	0.8321	4.11e-03
	16	0.8328	1.33e-03	0.7252	2.34e-03	0.8389	4.41e-03
	32	0.8299	2.34e-03	0.7138	6.06e-03	0.8402	4.32e-03
banana	8	0.8878	4.33e-05	0.8865	4.45e-05	0.9551	1.87e-05
	16	0.8908	6.42e-05	0.8894	6.59e-05	0.9577	2.25e-05
	32	0.8923	7.42e-05	0.8910	7.61e-05	0.9590	2.35e-05
ring	8	0.9426	6.12e-05	0.9426	6.13e-05	0.9856	8.73e-06
	16	0.9457	4.38e-05	0.9457	4.39e-05	0.9885	5.13e-06
	32	0.9476	4.06e-05	0.9476	4.06e-05	0.9893	5.01e-06
spambase	8	0.9467	5.94e-05	0.9440	6.54e-05	0.9824	1.91e-05
	16	0.9497	6.43e-05	0.9472	7.04e-05	0.9837	2.35e-05
	32	0.9506	5.95e-05	0.9481	6.53e-05	0.9838	2.16e-05
backache	8	0.8556	1.02e-03	0.5552	6.92e-03	0.7340	1.62e-02
	16	0.8667	5.49e-04	0.5520	8.05e-03	0.7534	1.27e-02
	32	0.8689	3.79e-04	0.5075	5.29e-03	0.7585	1.19e-02
ionosphere	8	0.9386	1.01e-03	0.9328	1.27e-03	0.9727	3.81e-04
	16	0.9398	8.62e-04	0.9342	1.08e-03	0.9737	3.57e-04
	32	0.9375	6.38e-04	0.9315	8.13e-04	0.9732	3.43e-04
mfeat_zernike	8	0.7634	2.03e-04	0.7615	2.19e-04	0.9615	1.10e-05
	16	0.7764	1.31e-04	0.7741	1.65e-04	0.9656	5.98e-06
	32	0.7780	2.00e-04	0.7750	2.19e-04	0.9674	5.16e-06
segmentation	8	0.9768	2.14e-05	0.9767	2.19e-05	0.9988	4.88e-07
	16	0.9784	2.28e-05	0.9783	2.36e-05	0.9992	1.81e-07
	32	0.9763	1.67e-05	0.9763	1.68e-05	0.9992	1.95e-07
tokyo1	8	0.9283	3.08e-04	0.9224	3.51e-04	0.9798	5.54e-05
	16	0.9321	2.16e-04	0.9264	2.44e-04	0.9811	7.42e-05
	32	0.9287	3.95e-04	0.9227	4.39e-04	0.9807	8.12e-05

5. Theoretical Analysis

We provide theoretical guarantees: learned weights are never worse than equal averaging, generalize well, and strictly improve when members differ.

5.1. Notation and Setting

Let $\{p_k(x)\}_{k=1}^{M}$ denote the class-probability outputs from M pruned subtrees. Let the weight vector $w = (w_1,\ldots,w_M)$ lie on the probability simplex $\Delta_M = \{w \in R_{>0}^{M}: \sum_k w_k = 1\}$. The ensemble prediction is

$$\hat{p}_w(x) = \sum_{k=1}^{M} w_k\, p_k(x). \tag{29}$$

Given a validation set $D_{val} = \{(x_i, y_i)\}_{i=1}^{n}$ and a convex, differentiable loss function $l(\cdot,\cdot)$ (e.g., cross-entropy or squared loss), the empirical validation loss is

$$L_{val}(w) = \frac{1}{n}\sum_{i=1}^{n} l\left(y_i, \sum_{k=1}^{M} w_k\, p_k(x_i)\right). \tag{30}$$

The optimal weights are obtained by

$$w^* = \arg\min_{w\in\Delta_M} L_{val}(w) \tag{31}$$

and the equal-averaging weights are $w^{eq} = (1/M,\ldots,1/M)$.

5.2. Empirical Risk Dominance

Theorem 5.1 (Empirical Risk Dominance). Let $L_{val}(w)$ be convex in w and Δ_M the probability simplex. Then the optimal solution w^* satisfies

$$L_{val}(w^*) \le L_{val}(w^{eq}). \tag{32}$$

Proof. The simplex Δ_M is convex and compact. Since $l(y, \hat{p}_w(x))$ is convex in $\hat{p}_w(x)$ and $\hat{p}_w(x)$ is linear in w, their composition $L_{val}(w)$ is convex in w. By definition of the global minimum, for any feasible point $w^{eq} \in \Delta_M$, we have $L_{val}(w^*) \le L_{val}(w^{eq})$.

Interpretation. Equal averaging is simply a feasible point of the optimization problem, while w^* is the minimizer of the same convex function. Hence, the validation loss of gradient-based weighting cannot be higher than that of simple averaging. This provides a minimal but essential guarantee that our method is never worse on the validation data.

If all subtrees produce identical predictions, every w yields the same loss, and equality holds. If l is strictly convex and the subtrees differ, the inequality is strict and w^* is unique.

5.3. Generalization Bound

Corollary 5.2 (Generalization to Test Data). Assume the loss $l(y,\hat{p})$ is L-Lipschitz in its first argument and bounded in 0,1. Let D_{val} be drawn i.i.d. from the same distribution as the test data. Then, with probability at least $1-\delta$,

$$E_{(x,y)}\left[l\left(y, \hat{p}_{w^*}(x)\right)\right] \le E_{(x,y)}\left[l\left(y, \hat{p}_{w^{eq}}(x)\right)\right] + O\left(\sqrt{\frac{logM + log(1/\delta)}{|D_{val}|}}\right) \tag{33}$$

Proof. Let the true risk be $L(w) = E\left[l\left(y, \hat{p}_w(x)\right)\right]$ and the empirical validation risk $\hat{L}(w) = \frac{1}{|D_{val}|}\sum_{(x,y)\in D_{val}} l\left(y, \hat{p}_w(x)\right)$. From standard uniform convergence results (e.g., Rademacher complexity of convex hulls), for all $w \in \Delta_M$, with probability $1-\delta$,

$$\left|L(w) - \hat{L}(w)\right| \le O\left(L\sqrt{\frac{logM + log(1/\delta)}{|D_{val}|}}\right) \tag{34}$$

Since w^* minimizes $\hat{L}(w)$ and w^{eq} is feasible, $\hat{L}(w^*) \le \hat{L}(w^{eq})$. Combining yields

$$L(w^*) \le L(w^{eq}) + O\left(\sqrt{\frac{\log M + \log(1/\delta)}{|D_{val}|}}\right). \tag{35}$$

Discussion. This corollary states that the learned weights remain competitive on unseen data. The excess risk over equal averaging is bounded by a small generalization term that decreases as the validation sample size increases. Because the op-

timization space is low-dimensional (the simplex), its complexity only grows as $logM$, leading to stable generalization even for moderate ensemble sizes.

5.4. Strict Improvement Condition

Proposition 5.3 (Condition for Strict Improvement). Assume L_{val} is differentiable at w^{eq}. If there exists a feasible direction d such that $\sum_k d_k = 0, w^{eq} + \epsilon d \in \Delta_M$ for some $\epsilon > 0$, and $\nabla L_{val}(w^{eq})^\top d < 0$, then the optimal w^* satisfies $L_{val}(w^*) < L_{val}(w^{eq})$.

Proof. A negative directional derivative implies that w^{eq} is not a stationary point of $L_{val}(w)$ under the simplex constraint. By convexity, there exists a smaller value along d, and thus the global minimizer w^* achieves strictly lower loss. Interpretation. If all subtrees behave identically on the validation data, the gradients are equal and equal averaging is already optimal. Otherwise, the presence of a descent direction means the learned weighting strictly improves validation loss.

5.5. Variance-Correlation Perspective

Under squared loss and scalar outputs, define $P(x) = [p_1(x), \ldots, p_M(x)]^\top$ and its covariance matrix $\sum = Cov(P(x))$. The ensemble variance is $Var\left(\sum_k w_k p_k(x)\right) = w^\top \sum w$. Equal averaging uses w^{eq}, giving $Var_{eq} = (w^{eq})^\top \sum w^{eq}$.

Proposition 5.4 (Variance Reduction under Weak Correlation). If there exists $w \in \Delta_M$ such that $w^\top \sum w < (w^{eq})^\top \sum w^{eq}$ and its bias is not worse than that of w^{eq}, then the total mean-squared error of w is strictly smaller. In particular, when $\sum$ has large positive off-diagonal entries (high correlation among members), downweighting redundant members reduces the variance term.

Interpretation. Gradient-based weighting can be seen as automatically lowering the weights of correlated, redundant subtrees, thereby reducing ensemble variance. This provides a statistical explanation for its stability and superior accuracy.

5.6. Why Pruning Helps

Pruning preserves influential splits, creating structurally diverse subtrees that agree on easy samples but differ on hard ones.

Lemma 5.5 (Influence-Oriented Diversity). If each pruned subtree keeps the high-influence splits of the parent tree, then across members, leaf rules differ mainly in low-importance branches while preserving the core structure. Consequently, member predictions are weakly correlated on difficult regions but highly consistent elsewhere, which is the regime where learned weighting improves over equal averaging.

5.7. Summary

The above results jointly establish that: (i) our learned weighting is never worse than equal averaging on the validation loss (Theorem 5.1); (ii) it generalizes safely with only a small $O\left(\sqrt{logM/|D_{val}|}\right)$ gap (Corollary 5.2); and (iii) it strictly improves whenever pruned subtrees provide non-redundant information (Proposition 5.3, 5.4, 5.5).

Together, these properties form the theoretical foundation for the empirical effectiveness of the proposed MultiPruneEnsemble.

6. Conclusion

We proposed MultiPruneEnsemble, integrating multiple pruning strategies with gradient-based weight learning. Contributions: (1) retaining diverse pruned variants; (2) validation-driven weight learning; (3) consistent improvements demonstrated.

Pruning combined with gradient optimization creates diverse, complementary subtrees. Limitations include increased computation and post-hoc weight learning.

We focused on isolating pruning diversity benefits; broader baseline comparisons are future work.

Future work: joint pruning/weight learning, instance-wise gating, and expanded benchmarks.

CODE AVAILABILITY

The implementation of the proposed MultiPruneEnsemble framework is publicly available at https://github.com/suonion/ MultiPruneEnsemble. Comprehensive records of the experimental process are provided in the accompanying Excel files included in the repository.

References

1. Aglin, G., Nijssen, S., Schaus, P. Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020, 3146-3153. https://doi.org/10.1609/aaai.v34i04.5711
2. Arik, S. Ö., Pfister, T. TabNet: Attentive Interpretable Tabular Learning. Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, 2021, 6679-6687. https://doi.org/10.1609/aaai.v35i8.16826
3. Bertsimas, D., Dunn, J. Optimal Classification Trees. Machine Learning, 2017, 106(7), 1039-1082. https://doi.org/10.1007/s10994-017-5633-9
4. Bishop, C. M. Pattern Recognition and Machine Learning. Springer, 2006.
5. Breiman, L. Random Forests. Machine Learning, 2001, 45(1),5-32.https://doi.org/10.1023/A:1010933404324
6. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. Classification and Regression Trees. Wadsworth, 1984.
7. Cao, J., Li, W., Ma, C., Tao, Z. Optimizing Multi-Sensor Deployment via Ensemble Pruning for Wearable Activity Recognition. Information Fusion, 2018, 41, 68-79. https://doi.org/10.1016/j.inffus.2017.08.002
8. Chen, T., Guestrin, C. XGBoost: A Scalable Tree Boosting System. arXiv preprint arXiv:1603.02754, 2016. https://doi.org/10.1145/2939672.2939785
9. Demirovic, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., Stuckey, P. J. MurTree: Optimal Decision Trees via Dynamic Programming and Search. Journal of Machine Learning Research, 2022, 23, 26:1-26:47. https://jmlr.org/papers/v23/20-520.html
10. Emine, Y., Forel, A., Malek, I., Vidal, T. Free Lunch in the Forest: Functionally-Identical Pruning of Boosted Tree Ensembles. Proceedings of the 39th AAAI Conference on Artificial Intelligence, 2025, 16488-16495. https://doi.org/10.1609/aaai.v39i16.33811
11. Esposito, F., Malerba, D., Semeraro, G. A Comparative Analysis of Methods for Pruning Decision Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(5), 476-491. https://doi.org/10.1109/34.589207
12. Fan, W., Chu, F., Wang, H., Yu, P. S. Pruning and Dynamic Scheduling of Cost-Sensitive Ensembles. Proceedings of the Eighteenth National Conference on Artificial Intelligence, 2002, 146-151. http://www.aaai.org/Library/AAAI/2002/aaai02-023.php
13. Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 2001, 29(5), 1189-1232. https://doi.org/10.1214/aos/1013203451
14. Grinsztajn, L., Oyallon, E., Varoquaux, G. Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data? Advances in Neural Information Processing Systems 35, 2022. http://papers.nips.cc/paper_files/paper/2022/hash/0378c7692da36807bdec87ab043cdadc-Abstract-Datasets_and_Benchmarks.html https://doi.org/10.52202/068431-0037
15. Guo, L., Boukir, S. Margin-Based Ordered Aggregation for Ensemble Pruning. Pattern Recognition Letters, 2013, 34(6), 603-609. https://doi.org/10.1016/j.patrec.2013.01.003
16. Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z. S. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. arXiv preprint arXiv:2012.06678, 2020. https://arxiv.org/abs/2012.06678
17. Hyafil, L., Rivest, R. L. Constructing Optimal Binary Decision Trees Is NP-Complete. Information Processing Letters, 1976, 5(1), 15-17. https://doi.org/10.1016/0020-0190(76)90095-8
18. Jayawardhana, M., Tu, R., Dooley, S., Cherepanova, V., Wilson, A. G., Hutter, F., White, C., Goldstein, T., Goldblum, M. Transformers Boost the Performance of Decision Trees on Tabular Data Across Sample Sizes. arXiv preprint arXiv:2502.02672, 2025. https://doi.org/10.48550/arXiv.2502.02672
19. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems 30, 2017, 3146-3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html
20. Khalifa, F. A., Abdelkader, H. M., Elsaid, A. H. An Analysis of Ensemble Pruning Methods Under the Explanation of Random Forest. Information Systems, 2024, 120, 102310. https://doi.org/10.1016/j.is.2023.102310
21. Koch, C., Strassle, C., Tan, L. Properly Learning Decision Trees with Queries Is NP-Hard. arXiv preprint

arXiv:2307.04093, 2023. https://doi.org/10.1109/FOCS57990.2023.00146

22. Lu, Z., Wu, X., Zhu, X., Bongard, J. C. Ensemble Pruning via Individual Contribution Ordering. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, 871-880. https://doi.org/10.1145/1835804.1835914
23. Martínez-Muñoz, G., Suárez, A. Pruning in Ordered Bagging Ensembles. Proceedings of the 23rd International Conference on Machine Learning, 2006, 609-616. https://doi.org/10.1145/1143844.1143921
24. Marton, S., Lüdtke, S., Bartelt, C., Stuckenschmidt, H. GradTree: Learning Axis-Aligned Decision Trees with Gradient Descent. Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence, 2024, 14323-14331. https://doi.org/10.1609/aaai.v38i13.29345
25. Marton, S., Lüdtke, S., Bartelt, C., Stuckenschmidt, H. GRANDE: Gradient-Based Decision Tree Ensembles for Tabular Data. Proceedings of the Twelfth International Conference on Learning Representations, 2024. https://openreview.net/forum?id=XEFWBxi075
26. Mohammed, A. M., Onieva, E., Wozniak, M. Selective Ensemble of Classifiers Trained on Selective Samples. Neurocomputing, 2022, 482, 197211. https://doi.org/10.1016/j.neucom.2021.11.045
27. Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., Moore, J. H. PMLB: A Large Benchmark Suite for Machine Learning Evaluation and Comparison. BioData Mining, 2017, 10(36), 1-13. https://doi.org/10.1186/s13040-017-0154-4
28. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikit-Learn: Machine Learning in Python. Journal of Machine Learning Research, 2011, 12, 2825-2830.
29. Popov, S., Morozov, S., Babenko, A. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. arXiv preprint arXiv:1909.06312, 2019. http://arxiv.org/abs/1909.06312
30. Prokhorenkova, L. O., Gusev, G., Vorobev, A., Dorogush, A. V., Gulin, A. CatBoost: Unbiased Boosting with Categorical Features. Advances in Neural Information Processing Systems 31, 2018, 6639-6649. https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html
31. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
32. Romano, J. D., Le, T. T., La Cava, W., Gregg, J. T., Goldberg, D. J., Chakraborty, P., Ray, N. L., Himmelstein, D., Fu, W., Moore, J. H. PMLB v1.0: An Open Source Dataset Collection for Benchmarking Machine Learning Methods. arXiv preprint arXiv:2012.00058, 2021. https://doi.org/10.1093/bioinformatics/btab727
33. Shwartz-Ziv, R., Armon, A. Tabular Data: Deep Learning Is Not All You Need. Information Fusion, 2022, 81, 84-90. https://doi.org/10.1016/j.inffus.2021.11.011
34. Šmída, M. Design of Accuracy Predictors for Convolutional Neural Networks. Bachelor's Thesis, Brno University of Technology, 2023. https://excel.fit.vutbr.cz/submissions/2023/082/82_poster.pdf