Soft Forests: Combining Random Forest Randomization with Differentiable Soft Tree Ensembles

Submission for 15.961 (Independent Study)

Azfal Peermohammed [*1] **Daniel Rapoport** [*1]
Supervised by Prof. Rahul Mazumder [1]

Abstract

Soft decision tree ensembles, which combine smooth probabilistic trees, offer a differentiable alternative to classical tree-based models. In this work, we explore the impact of incorporating classic random forest techniques—namely subset feature selection, bagging, and bootstrapping—into soft ensembles. Our goal is to understand whether these ideas, originally designed for hard decision trees, can enhance the performance, robustness, or diversity of soft ensemble methods. We design a series of experiments comparing standard soft ensembles with their modified counterparts across multiple datasets and evaluation metrics. Preliminary results indicate that while classical random forest heuristics still provide benefits in differentiable tree ensembles, their effects may differ from those in traditional settings. Our findings contribute to a deeper understanding of how ensemble construction strategies interact with probabilistic aggregation schemes and open up new directions for hybrid ensemble design.

1. Introduction

Soft decision trees, which use smooth, differentiable sigmoid functions in place of hard axis-aligned splits, offer a flexible alternative to traditional decision trees like those produced by CART. Unlike their non-differentiable counterparts, soft trees can be optimized using gradient-based methods, making them especially appealing for integration into modern machine learning pipelines. Random forests, a classic ensemble method, dramatically improved tree-based models by introducing randomness through bagging and feature subset selection—techniques that effectively reduce overfitting and balance the bias-variance trade-off. In this study, we ask whether similar randomization strategies can enhance the performance of soft decision tree ensembles. Specifically, we investigate whether injecting randomness into soft ensembles improves generalization without sacrificing the benefits of their smooth, differentiable structure. Through a series of empirical evaluations across diverse datasets, we assess the impact of bagging, bootstrapping, and feature selection on ensemble accuracy, robustness, and diversity. We aim to attempt to answer the question: Can randomized ensembles of soft trees similarly reduce variance while retaining their flexibility and smoothness?

2. Related Work

Random Forests, introduced by Breiman (2001) [1], are among the most influential ensemble learning techniques in machine learning. By combining multiple decision trees trained on different bootstrap samples and introducing random feature selection at each split, random forests reduce variance without substantially increasing bias. This randomized construction process—known as bagging—helps mitigate overfitting and improves generalization, particularly in high-variance models like deep trees. A key insight from Breiman's work is that even simple base learners can become powerful predictors when aggregated appropriately, breaking the traditional bias-variance trade-off and laying the foundation for many modern ensemble methods.

More recently, efforts to incorporate decision trees into differentiable machine learning pipelines have led to the development of Tree Ensemble Layers (TEL) [2]. These models reinterpret tree ensembles as layers within neural networks, enabling end-to-end training using gradient-based optimization. By replacing hard, axis-aligned splits with soft, sigmoid-based decisions, TEL transforms decision trees into smooth function approximators that can backpropagate error signals. Implemented in frameworks like TensorFlow, these differentiable trees allow for seamless integration with deep learning architectures, opening new avenues for hybrid modeling strategies that combine the interpretability of trees with the expressiveness of neural

[*]Equal contribution [1]Operations Research Center, Massachusetts Institute of Technology (MIT), Cambridge, USA. Correspondence to: Azfal Peermohammed <azpeer@mit.edu>, Daniel Rapoport <drap@mit.edu>.

networks.

Our work investigates the effects of combining (TEL) with the core randomization mechanisms of random forests. Specifically, we examine how incorporating two key sources of stochasticity—bootstrap sampling of training data and random subset selection of input features—impacts the performance and behavior of differentiable tree ensembles. By introducing these classical random forest techniques into the TEL framework, we aim to understand whether the benefits of variance reduction and improved generalization observed in traditional ensembles can be preserved, or even enhanced, in a smooth, end-to-end differentiable setting.

3. Preliminaries

3.1. Random Forests

Assuming the reader is familiar with the fundamentals of random forests, we briefly review the key components of the method and establish the notation used throughout the paper. In particular, we focus on the two main sources of stochasticity that underlie the effectiveness of random forests: bootstrapping and feature subset selection.

Let the original dataset be denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ are input features and $y_i \in \mathbb{R}$ (or a finite label set for classification tasks). A random forest constructs an ensemble of decision trees $\{T_1, T_2, \ldots, T_M\}$, where each tree T_m is trained on a bootstrapped dataset $\mathcal{D}^{(m)}$, sampled with replacement from $\mathcal{D}$.

Formally, for each tree T_m, a bootstrapped dataset is drawn:

$$\mathcal{D}^{(m)} = \{(x_i^{(m)}, y_i^{(m)})\}_{i=1}^n,$$

$$\text{where } (x_i^{(m)}, y_i^{(m)}) \sim \mathcal{D} \text{ with replacement.}$$

In addition to data-level randomization, each tree introduces feature-level stochasticity by selecting a random subset of features $\mathcal{F}_j \subset \{1, 2, \ldots, d\}$ at each split node j. A splitting criterion (e.g., Gini impurity or variance reduction) is then applied only to features in $\mathcal{F}_j$, typically with $|\mathcal{F}_j| = \sqrt{d}$ for classification or $|\mathcal{F}_j| = d/3$ for regression:

$$\mathcal{F}_j \sim \text{Uniform Subset}(\{1, \ldots, d\}, k), \quad \text{for some } k < d.$$

The final prediction is obtained by aggregating the outputs of all trees, typically via majority vote (for classification) or averaging (for regression):

$$\hat{y}(x) = \frac{1}{M} \sum_{m=1}^{M} T_m(x).$$

These two sources of randomness—bootstrapping at the dataset level and feature subset selection at the node level—are central to the variance reduction properties of random forests. They ensure that individual trees are decorrelated, which enhances the ensemble's generalization performance, especially in high-variance learning scenarios.

3.2. The Tree Ensemble Layer

We provide a brief overview of Tree Ensemble Layers (TEL) and the associated notation for soft decision trees. For a comprehensive derivation and deeper theoretical background, we refer the reader to the work of Hazimeh et al. [2].

The setup for ensembling trees in TEL mirrors the structure described in the previous subsection on random forests. Therefore, we focus our attention here on the architecture and formulation of a single soft decision tree.

One important note is that, in classification settings, the ensemble outputs raw logit values for our implementation. These logits can then be mapped to class probabilities via the softmax function:

$$P(y = c \mid x) = \frac{\exp(z_c)}{\sum_{k=1}^{C} \exp(z_k)},$$

where z_c denotes the logit corresponding to class c, and C is the total number of classes.

Soft decision trees are a variant of traditional decision trees that employ *soft routing* mechanisms in place of the hard, deterministic splits found in classical trees. This relaxation enables differentiability, making soft trees amenable to gradient-based optimization and integration into neural architectures.

We assume that a tree T is a perfect binary tree of depth d. Following the notation introduced in the TEL framework, let I and L denote the sets of internal and leaf nodes, respectively. For any node $i \in I \cup L$, we define $A(i)$ as the set of its ancestor nodes. The notation $\{x \to i\}$ indicates that sample x reaches node i.

In soft decision trees, routing is probabilistic: a sample is passed to both the left and right subtrees, each with a certain probability. Unlike classical decision trees that use axis-aligned splits, soft trees define splits using hyperplanes. Specifically, each internal node $i \in I$ is associated with a trainable weight vector $w_i \in \mathbb{R}^d$ and bias $b_i \in \mathbb{R}$. The routing probability to the left child is determined by applying the sigmoid activation function to the affine transformation:

$$p_i(x) = \sigma(w_i^\top x + b_i),$$

where $\sigma(\cdot)$ denotes the sigmoid function. Although the TEL framework supports a wide range of activation functions, we restrict our focus to the sigmoid for simplicity. The probability of reaching a given leaf node is calculated as the product of the routing probabilities along the path from the root node to that leaf.

As in classical decision trees, each leaf node $\ell \in L$ in a soft decision tree stores a weight vector $o_\ell \in \mathbb{R}^k$, which is learned during training. Importantly, these output vectors o_ℓ are constant with respect to the input sample x, i.e., they do not depend on x and represent fixed predictions for each leaf.

Given an input $x \in \mathbb{R}^p$, the overall prediction of the tree is computed as the expected value of the leaf outputs, weighted by the probability of the input reaching each leaf. Formally, the prediction of the tree $T(x)$ is given by:

$$T(x) = \sum_{\ell \in L} P(x \to \ell) \cdot o_\ell,$$

where $P(x \to \ell)$ denotes the probability that sample x is routed to leaf ℓ under the soft routing mechanism.

4. Methodology

The first step was to reimplement the soft ensemble from scratch in PyTorch, both to develop a deeper understanding of the underlying mechanics of TEL and to enable the application of randomized modifications. The trees are constructed recursively to a specified depth and are fully built at initialization. Then the two sources of randomization were applied as follows. The construction of the code followed the architecture proposed in FASTEL [3], representing an ensemble as one neural network layer in TensorFlow.

4.1. Subset Feature Selection

To incorporate subset feature selection, we implemented the following algorithm. At each internal node $i \in I$, we retain the shape of the weight parameter $w_i \in \mathbb{R}^d$ and define a fixed number of features to be considered for the split. We then generate a binary mask $h_i \in \{0, 1\}^d$, sampled uniformly at random, with exactly $m < d$ non-zero entries corresponding to the selected subset of features. During the forward pass, we compute the masked weight vector via the Hadamard (element-wise) product:

$$\tilde{w_i} = h_i \odot w_i,$$

where $\odot$ denotes the Hadamard product. Then the new probability of the data point being routed to the left is denoted by.

$$\tilde{p_i}(x) = \sigma(\tilde{w_i}^T x + b_i)$$

This masked vector $\tilde{w_i}$ is then used in place of w_i when computing the split probability. Note that the binary mask h_i is fixed for each node and remains unchanged throughout training; it is not a learnable parameter.

4.2. Bootstrapping

Bootstrapping was implemented using a standard approach in which a loop iterates over the desired number of trees in the ensemble. At each iteration, a new decision tree is instantiated and trained on a bootstrapped dataset—constructed by sampling N data points with replacement from the original training set. This procedure introduces randomness through both data sampling and model initialization, thereby promoting independence and diversity among the trees in the ensemble and helping enforce that they are independently and identically distributed in practice.

While this method is effective, it can be computationally intensive due to the repeated creation of bootstrapped datasets. A potential avenue for future work could involve exploring more efficient implementations that approximate bootstrap behavior without requiring explicit data duplication at each iteration.

5. Data

Our experimental evaluation was conducted on 11 classification datasets, primarily sourced from the UCI Machine Learning Repository. These datasets encompass a diverse range of domains, including medical diagnostics, image recognition, and structured decision-making tasks. For instance, the Breast Cancer and Thyroid Disease datasets involve clinical features for disease classification, while Optdigits and DNA datasets pertain to image and sequence recognition, respectively. The Car Evaluation and Nursery datasets are derived from hierarchical decision models, assessing car acceptability and nursery school applications. Other datasets, such as Dermatology and Ecoli, focus on medical and biological classification challenges. This diversity ensures a comprehensive assessment of our algorithms across various real-world scenarios.

To further assess the robustness of our ensemble methods, we injected controlled label noise into the training portion of each dataset. Specifically, 30 % of the training instances had their class labels replaced with an incorrect label chosen uniformly at random from the remaining classes. The validation and test sets retained their original labels. This corruption increases the difficulty of the learning task and accentuates the tendency of individual trees to overfit to mislabeled samples. To assess the impact of each randomization strategy on ensemble diversity and overfitting mitigation, we analyzed the distribution of predicted probabilities—specifically their standard deviation and skewness—across trees. This provides a quantitative measure of uncertainty and variation introduced by each strategy.

6. Experiments

To evaluate the impact of different randomization strategies in the TEL framework, we conducted experiments on 11 benchmark classification datasets (as described in Section 3). We compared four ensemble configurations of the

TEL model, varying whether bootstrapping and/or random feature selection were used:

1. **Standard TEL (no randomization)**: each tree is trained on the entire training set using all features.
2. **TEL with Bootstrapping**: each tree is trained on a bootstraped dataset of the training data, using all features for splits.
3. **TEL with Random Feature Subset Selection**: each tree is trained on the full training set, but only a random subset of features is considered at each split.
4. **TEL with Bootstrapping + Feature Subset**: each tree is trained on a bootstrap dataset and uses feature subset selection at each split.

All datasets were randomly split into 80% training and 20% testing sets. From the training portion, 20% of the data was further reserved for validation, yielding an effective split of 64% for training, 16% for validation, and 20% for testing. The validation set was used exclusively for hyperparameter tuning, while the test set—with original, uncorrupted labels—was held out for final evaluation. Hyperparameter tuning was conducted in two stages using `Hyperopt`.

Stage 1: We optimized the single-tree parameters. We ran 30 trials of Hyperopt to maximize validation accuracy by varying four hyperparameters:

- **Learning rate:** $\mathcal{U}[\log(0.001), \log(0.1)]$
- **Batch size:** $\{32,\ 64,\ 128,\ 256\}$
- **Training epochs:** $\{5, \ldots, 20\}$
- **Maximum tree depth:** $\{3, \ldots, 8\}$

This stage was performed once per dataset and yielded the best base-tree configuration for that dataset.

Stage 2: We then tuned the ensemble-level hyperparameters for each TEL variant, using the optimal Stage 1 parameters as a fixed base. For each dataset, we ran 10 Hyperopt trials to tune the following hyperparameters:

- **Number of trees:** $\{5, \ldots, 50\}$
- **Subset share:** $\mathcal{U}(0.1,\ 0.9)$

Where the subset share parameter was only applied to the models where it applied. Models without feature subset selection always use the full feature set at each node (equivalent to a subset share of 1.0). The best values for these ensemble hyperparameters were chosen based on validation accuracy.

Final Evaluation: Using the best hyperparameter configuration from both stages, we retrained each model variation on the full training data (recombining the training and validation sets) and then evaluated it on the test set. The test set remained completely held-out during training and tuning, and its ground-truth labels were unaltered (i.e., no label noise or corruption was applied).

For performance evaluation, we report classification accuracy, computed at both the individual-tree and ensemble levels. Individual tree performance is measured directly using each tree's predictions on the test set. Ensemble predictions are formed by majority voting for the class labels.

In addition to accuracy, we assess the diversity of the ensemble's outputs by measuring the distribution of the individual-tree predictions. For each test example, we compute the standard deviation and skewness of the predicted class probabilities across the ensemble's trees. A higher standard deviation indicates greater disagreement among the trees' predictions. The skewness captures the asymmetry of the probability distribution (e.g., whether most trees strongly favor one class over the others). These distributional metrics provide insight into the ensemble's behavior and the effects of the randomization strategies on model diversity.

Due to the computational expense of the two-stage tuning process, we evaluated each configuration using only a single training run per dataset. In other words, we did not train multiple models or average results over different random seeds; all reported results are from one final model for each configuration on each dataset.

7. Results

7.1. Ensemble accuracy

Table 1 reports ensemble accuracies for the four TEL variants. In nine of the eleven data sets at least one randomised configuration outperforms the baseline TEL, indicating that stochastic modifications are usually beneficial. No single technique is uniformly best: feature–subset selection alone equals or exceeds the baseline in eight data sets, while *Bagging* and *Bagging+Subset* obtain the top score in four and five data sets, respectively. Examination of the hyperparameter trajectories in Table 4 shows that the optimal subset share varies markedly, from 0.10 on DNA to 0.88 on NURSERY, underscoring the task-specific nature of feature sampling. *Importantly, this variability does not necessarily conflict with common heuristics such as using $\sqrt{d}$ candidate features in classification; a more conclusive assessment will require averaging over multiple random seeds and allocating a larger evaluation budget in the Hyperopt search.*

7.2. Accuracy uplift and the weak-learner hypothesis

Accuracy-uplift values (Table 3) quantify how much each configuration improves over a single soft tree, allowing us to test the random-forest premise that ensembles of weak yet diverse learners can outperform stronger but correlated ones. Randomisation fulfils this expectation: a stochastic variant achieves the largest uplift in *ten of the eleven* data sets. The combined *Bagging+Subset* configuration provides the highest uplift in five data sets, and in eight data sets at least one randomised model attains *twice* the gain of the basic TEL ensemble. These figures indicate a pronounced tendency for weaker individual learners— induced by bootstrapping, feature sampling, or both—to aggregate into a stronger committee. Although the respective contributions of bootstrapping and feature sampling cannot be cleanly disentangled, their conjunction consistently yields the most substantial improvements, suggesting that the two sources of diversity act in a complementary fashion.

The plot below shows one example for the `car_evaluation` dataset, illustrating how ensembling improves predictive performance over individual trees. The combination of bagging and feature subset selection achieves the largest uplift.

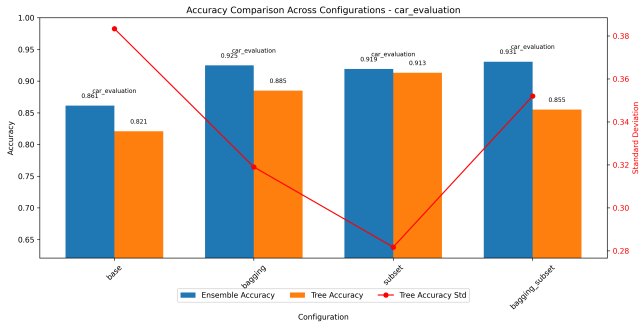

Figure 1. Accuracy comparison across base, bagging, subset, and bagging-subset configurations for the `car_evaluation` dataset. Ensemble and tree-level accuracies are shown alongside the standard deviation of tree accuracies.

7.3. Hyper-parameter tendencies

Tree depth. The random forest literature suggests that when trees are sufficiently decorrelated, they can be grown to greater depths without leading to overfitting, as the ensemble as a whole remains robust due to the diversity among individual trees. In our tuned models this pattern is *not* apparent. Median depths across the eleven tasks are Base = 5, Bagging = 4, Subset = 5, Bagging+Subset = 4. Hence randomisation does not consistently drive the optimiser toward shallower or deeper weak learners, contrasting with the usual hard-tree random-forest tendency.

Feature budget (m_{try}). The selected feature fractions show strong dispersion. *Subset* values span nearly an order of magnitude (0.10 – 0.88). *Bagging+Subset* values show a similar spread (0.14 – 0.84).

This variability, while wider than the conventional $\sqrt{d}$ rule of thumb for hard forests, is not contradictory: soft splits use hyper-planes and the optimiser can down-weight irrelevant features even when they are present. A firmer conclusion will require multiple random seeds and a larger hyper-parameter budget.

7.4. Distribution metrics

Tree-level diversity was assessed via the class-wise standard deviation σ and skewness γ of predicted probabilities (Table 5). In eight data sets a randomised variant exhibits the largest σ, confirming that bootstrapping and feature sampling increase dispersion among trees. In the remaining three data sets the baseline shows the highest spread, attributable to hyper-parameters that produce extremely shallow or deep trees. No consistent winner is observed between *Bagging* and *Subset*; the dominant variance axis appears to be data-dependent. Additional seeds and a higher number of hyperopt evaluations would be required for statistical confirmation.

Below is an examplary plot for the probability distribution of class 0 predictions on the `dermatology` dataset. This plot highlights how bagging increases the variance of predicted probabilities, with the highest variance observed when combining bagging and subset selection (single example, not translatable to all classes/datasets).

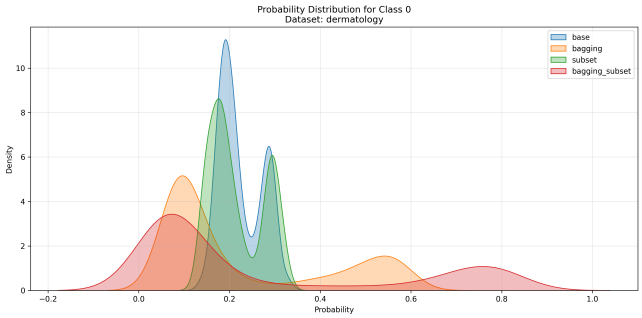

Figure 2. Probability distribution of class 0 predictions for the `dermatology` dataset across different configurations.

8. Conclusion

This work examined whether the two canonical heuristics of random forests, namely bootstrap sampling of rows and random sub-sampling of columns, remain useful when the base learners are smooth and fully differentiable trees. The answer is partly affirmative.

1. **Predictive benefit.** A stochastic configuration outperformed the vanilla TEL in nine of the eleven benchmarks, and a randomised variant achieved the largest accuracy uplift in ten. The *Bagging plus Subset* ensemble produced the best gain in five tasks and more than doubled the baseline uplift in eight, confirming that weak learners can form a strong ensemble even in the soft-tree setting (Table 3).

2. **Diversity signal.** Bootstrapping and feature sampling raised the inter-tree standard deviation of class probabilities in eight data sets (Table 5); however, skewness patterns were mixed, so the performance gains cannot be linked solely to greater predictive spread. The way randomisation helps may therefore differ from its effect in hard-tree forests.

3. **Hyper-parameter behaviour.** Randomisation did not push the optimiser toward uniformly deeper weak learners (median depths: Base 5, Bagging 4, Subset 5, Bagging plus Subset 4). The tuned feature fraction m_{try} ranged from 0.10 to 0.88, much wider than the usual $\sqrt{p}$ guideline, illustrating how gradient training and soft hyper-plane splits alter classical heuristics (Table 4).

Limitations and future work. Results are based on a single data split and one Hyperopt run per task; additional seeds and a larger search budget are needed for statistical confirmation. Future work includes deeper analysis of diversity metrics, adaptive m_{try} schedules, and a bias–variance decomposition suited to differentiable trees.

Take-away. Bootstrap sampling and random feature selection remain beneficial, and often complementary, in gradient-trained soft-tree ensembles; however, not all random-forest intuitions transfer verbatim. Hybrid models will need revised heuristics that respect the interplay between smooth splits and continuous optimisation.

Github Repository

The full implementation of the Soft-Forest, including testing pipelines, results and plots can be found within the following https://github.com/danielrapi/Soft-Forest.

Additional Tables and Figures

Dataset	Base	Bagging	Subset	Bagging_Subset
ann_thyroid	0.98	0.96	**0.98**	0.96
breast_cancer	0.94	**0.96**	0.93	0.95
car_evaluation	0.86	0.92	0.92	**0.93**
churn	0.86	0.86	**0.90**	0.89
dermatology	0.70	0.84	0.70	**0.96**
dna	**0.89**	0.68	0.87	0.55
Becoli	0.53	**0.79**	0.77	0.65
hypothyroid	**0.95**	**0.95**	0.94	**0.95**
nursery	0.97	0.81	0.97	**0.98**
optdigits	**0.97**	0.95	0.97	0.96
vehicle	0.75	0.67	**0.76**	0.70

Table 1. Ensemble Accuracy Comparison Across Methods

Dataset	Base	Bagging	Subset	Bagging_Subset
ann_thyroid	0.96	0.96	**0.97**	0.94
breast_cancer	0.94	**0.94**	0.93	0.93
car_evaluation	0.82	0.88	**0.91**	0.86
churn	0.82	0.71	**0.87**	0.81
dermatology	0.68	0.77	0.67	**0.85**
dna	**0.79**	0.62	0.69	0.61
ecoli	0.54	**0.79**	0.78	0.66
hypothyroid	**0.95**	0.94	0.91	0.92
nursery	**0.95**	0.71	0.95	0.93
optdigits	**0.90**	0.77	0.90	0.76
vehicle	**0.72**	0.65	0.70	0.64

Table 2. Tree Accuracy Comparison Across Methods

Dataset	Base	Bagging	Subset	Bagging_Subset
ann_thyroid	**1.6% (1.5pp)**	0.2% (0.2pp)	1.2% (1.2pp)	1.3% (1.2pp)
breast_cancer	-0.2% (-0.1pp)	1.7% (1.6pp)	-0.3% (-0.3pp)	**2.1% (1.9pp)**
car_evaluation	4.9% (4.0pp)	4.5% (4.0pp)	0.7% (0.6pp)	**8.8% (7.6pp)**
churn	5.3% (4.4pp)	**20.5% (14.6pp)**	3.4% (3.0pp)	9.8% (8.0pp)
dermatology	3.6% (2.4pp)	8.7% (6.7pp)	5.1% (3.4pp)	**13.3% (11.3pp)**
dna	12.2% (9.7pp)	8.9% (5.5pp)	**24.9% (17.3pp)**	-10.7% (-6.6pp)
ecoli	-1.1% (-0.6pp)	**-0.1% (-0.1pp)**	-1.0% (-0.8pp)	-1.8% (-1.2pp)
hypothyroid	0.1% (0.1pp)	1.5% (1.4pp)	2.2% (2.0pp)	**3.4% (3.2pp)**
nursery	2.3% (2.2pp)	**13.6% (9.7pp)**	1.8% (1.7pp)	4.7% (4.4pp)
optdigits	7.9% (7.1pp)	23.5% (18.2pp)	7.8% (7.0pp)	**25.4% (19.4pp)**
vehicle	4.4% (3.2pp)	3.0% (1.9pp)	**9.1% (6.4pp)**	9.6% (6.1pp)

Table 3. Accuracy uplift of Ensemble vs. Tree in % (absolute uplift in percentage-points), best method per dataset in **bold**.

Dataset	Config	Batch	Epochs	LR	Depth	#Trees	m_try
ann_thyroid	base	64	15	0.0831	3	10	—
	bagging	64	10	0.0093	4	35	—
	subset	32	5	0.0686	6	45	0.198
	bagging_subset	128	5	0.0850	3	20	0.474
breast_cancer	base	256	15	0.0644	4	10	—
	bagging	256	20	0.0821	3	50	—
	subset	128	10	0.0771	5	10	0.714
	bagging_subset	32	5	0.0456	7	45	0.141
car_evaluation	base	256	15	0.0741	3	35	—
	bagging	128	20	0.0206	6	20	—
	subset	64	10	0.0248	6	25	0.567
	bagging_subset	128	15	0.0457	4	40	0.768
churn	base	256	10	0.0323	7	10	—
	bagging	32	15	0.0097	6	20	—
	subset	32	5	0.0685	4	45	0.614
	bagging_subset	64	15	0.0102	5	40	0.227
dermatology	base	256	5	0.0769	7	15	—
	bagging	256	20	0.0483	4	25	—
	subset	128	5	0.0347	6	15	0.601
	bagging_subset	64	10	0.0453	5	20	0.836
dna	base	32	15	0.0154	5	50	—
	bagging	256	10	0.0754	6	30	—
	subset	256	10	0.0982	3	40	0.104
	bagging_subset	256	15	0.0982	3	40	0.184
ecoli	base	64	5	0.0311	4	35	—
	bagging	128	15	0.0501	4	30	—
	subset	64	15	0.0219	5	20	0.483
	bagging_subset	128	5	0.0613	4	30	0.791
hypothyroid	base	128	10	0.0211	3	20	—
	bagging	128	10	0.0916	4	15	—
	subset	32	20	0.0126	4	30	0.427
	bagging_subset	256	20	0.0474	5	20	0.665
nursery	base	64	10	0.0361	5	40	—
	bagging	256	5	0.0010	8	15	—
	subset	32	5	0.0592	5	25	0.880
	bagging_subset	64	5	0.0240	7	20	0.794
optdigits	base	256	20	0.0412	6	40	—
	bagging	32	15	0.0073	3	20	—
	subset	64	15	0.0267	4	15	0.564
	bagging_subset	256	15	0.0396	3	40	0.656
vehicle	base	64	20	0.0169	5	40	—
	bagging	128	20	0.0163	8	40	—
	subset	32	10	0.0361	6	5	0.877
	bagging_subset	64	15	0.0219	5	20	0.833

Table 4. Hyperparameter configurations per dataset and method.

Dataset	Class	base	bagging	subset	bagging_subset
ann_thyroid	0	0.072 (5.076)	0.073 (5.252)	**0.076 (5.018)**	0.078 (4.743)
	1	**0.097 (3.349)**	0.086 (3.222)	0.108 (3.512)	0.064 (3.484)
	2	0.132 (–2.431)	0.123 (–2.488)	**0.143 (–2.645)**	0.120 (–3.229)
breast_cancer	0	0.179 (–0.753)	**0.221 (–0.673)**	0.213 (–0.456)	0.200 (–0.775)
	1	0.179 (0.753)	**0.221 (0.673)**	0.213 (0.456)	0.200 (0.775)
car_evaluation	0	0.229 (–0.627)	**0.261 (–0.522)**	0.238 (–0.783)	0.253 (–0.532)
	1	0.158 (0.711)	0.200 (1.171)	0.201 (1.448)	**0.214 (1.095)**
	2	0.051 (0.955)	**0.081 (2.219)**	0.054 (2.467)	0.098 (2.291)
	3	0.046 (1.034)	0.085 (2.432)	**0.097 (3.307)**	0.093 (2.596)
churn	0	0.178 (–0.937)	**0.271 (–0.680)**	0.107 (–1.678)	0.153 (–0.822)
	1	0.178 (0.937)	**0.271 (0.680)**	0.107 (1.678)	0.153 (0.822)
dermatology	0	0.047 (0.488)	0.187 (0.888)	0.059 (0.451)	**0.289 (0.962)**
	1	0.030 (0.388)	0.117 (0.718)	0.024 (0.501)	**0.166 (1.652)**
	2	0.056 (1.118)	0.168 (1.423)	0.051 (1.184)	**0.248 (1.599)**
	3	0.016 (0.682)	0.069 (1.340)	0.024 (0.347)	**0.161 (1.948)**
	4	0.026 (0.972)	0.084 (2.014)	0.017 (1.265)	**0.208 (2.041)**
	5	0.007 (0.286)	**0.069 (2.000)**	0.011 (0.168)	0.132 (3.295)
dna	0	**0.237 (1.223)**	0.203 (1.240)	0.160 (1.095)	0.150 (1.520)
	1	**0.233 (1.307)**	0.158 (1.041)	0.161 (1.213)	0.157 (1.317)
	2	**0.268 (–0.093)**	0.190 (–0.056)	0.198 (0.023)	0.182 (–0.207)
ecoli	0	0.060 (–0.196)	**0.266 (0.322)**	0.247 (0.348)	0.105 (–0.154)
	1	0.047 (0.478)	**0.188 (0.689)**	0.137 (0.626)	0.086 (0.547)
	2	0.030 (0.985)	**0.117 (1.333)**	0.105 (0.826)	0.049 (1.038)
	3	0.012 (0.722)	**0.076 (2.119)**	0.035 (2.675)	0.011 (0.799)
	4	0.032 (0.813)	**0.143 (1.638)**	0.160 (1.474)	0.046 (1.116)
hypothyroid	0	0.049 (0.266)	0.072 (0.240)	**0.087 (1.727)**	0.094 (0.654)
	1	0.049 (–0.266)	0.072 (–0.240)	**0.087 (–1.727)**	0.094 (–0.654)
nursery	0	**0.287 (0.722)**	0.006 (0.583)	0.283 (0.709)	0.294 (0.763)
	1	**0.263 (0.754)**	0.004 (–0.209)	0.268 (0.744)	0.275 (0.791)
	2	**0.257 (0.833)**	0.005 (–0.029)	0.256 (0.832)	0.274 (0.854)
	3	0.071 (5.040)	0.002 (–0.559)	**0.067 (5.662)**	0.084 (4.140)
optdigits	0	**0.194 (2.687)**	0.167 (2.778)	0.199 (2.742)	0.186 (2.681)
	1	**0.205 (2.739)**	0.146 (2.439)	0.189 (2.736)	0.153 (2.596)
	2	0.195 (2.757)	**0.181 (2.676)**	0.198 (2.802)	0.166 (2.635)
	3	**0.197 (2.681)**	0.138 (2.601)	0.179 (2.740)	0.153 (2.425)
	4	**0.210 (2.735)**	0.169 (2.710)	0.195 (2.762)	0.182 (2.648)
	5	**0.207 (2.771)**	0.166 (2.501)	0.197 (2.780)	0.154 (2.571)
	6	**0.215 (2.685)**	0.163 (2.658)	0.196 (2.779)	0.167 (2.586)
	7	**0.209 (2.733)**	0.183 (2.772)	0.196 (2.664)	0.166 (2.518)
	8	**0.189 (2.862)**	0.127 (2.783)	0.182 (2.885)	0.152 (2.634)
	9	**0.198 (2.809)**	0.136 (2.158)	0.186 (2.820)	0.139 (2.671)
vehicle	0	0.120 (0.329)	0.142 (0.604)	**0.170 (0.621)**	0.173 (0.461)
	1	**0.159 (0.278)**	0.150 (0.491)	0.162 (0.911)	0.158 (0.823)
	2	0.244 (1.136)	0.214 (1.143)	**0.257 (1.215)**	0.267 (1.107)
	3	0.194 (1.160)	0.203 (1.233)	**0.251 (1.258)**	0.225 (1.334)

Table 5. Accuracy comparison (standard deviation with skewness). Entries in **bold** mark the highest std for each class within a dataset.

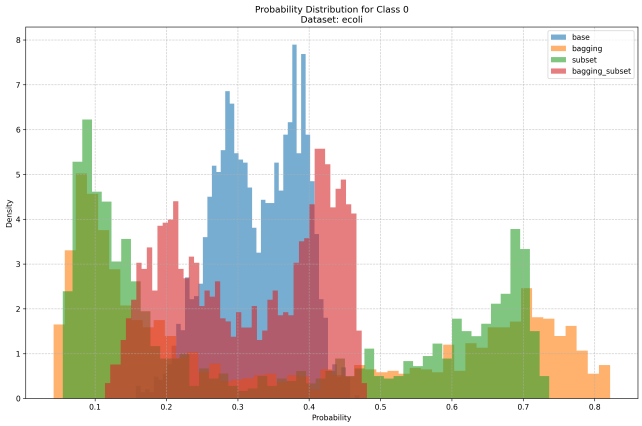

Figure 3. Probability distribution of class 0 predictions for the `ecoli` dataset across different configurations.