Optimized Sequential Testing for Binary Ensemble Classifiers

Anonymous authors
Paper under double-blind review

Abstract

Ensemble classifiers are predictive models that combine the results of simpler base models, often by majority vote. A classic example is random forests, which combine the predictions of decision trees. Ensembles that use more base models can be more accurate but also more costly to train and run. In this paper, we consider strategies for reducing the computational cost of binary classification using an approach from the field of sequential testing. Rather than evaluating all the base models and taking a majority vote, we evaluate the base models sequentially and stop execution when a clear majority emerges. We consider three different notions of optimality for early-stopping strategies that minimize the number of base models executed while controlling the rate of disagreement with the full ensemble. For each notion of optimality and allowable disagreement rate, we show that a linear program can be constructed and solved efficiently to find the optimal stopping strategy. We tested these methods on real-world datasets taken from the UC Irvine Machine Learning repository, and on the benchmark datasets proposed by Grinsztajn et al. We found that on most datasets, these methods provide speed-ups of 4x or more while controlling disagreement at 0.1%.

Keywords: time-accuracy tradeoff, binary classification, sequential analysis, early stopping, random forests, hypergeometric distribution, alpha spending

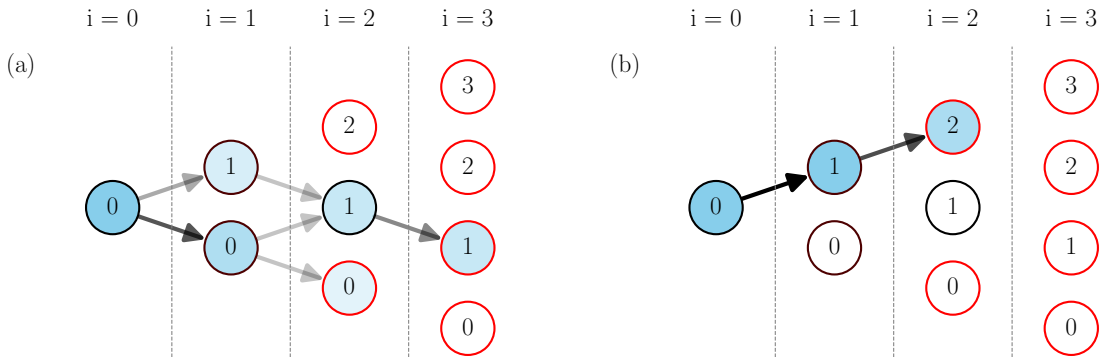

Figure 1: (a) State transition graph of a sequential test using a certain stopping strategy for $N = 3$ when $n = 1$. The horizontal axis represents i, the number of base models executed thus far; the vertical axis and node labels represent j, the number of those which have returned positive. The shading of each node or arrow represents the (unconditional) probability of the state or transition occurring. Red node borders represent states where the ensemble will stop if reached. (b) The equivalent graph when $n = 3$. Note that only one chain of states is now possible, and testing will never continue past $i = 2$ because by that point there will certainly be 2 positive votes, constituting a majority of the entire ensemble.

1 Introduction

A binary classifier is a function $f : \mathcal{X} \to \{-1, +1\}$ that assigns elements from a set $\mathcal{X}$ to one of two categories. An *ensemble classifier* combines the results of multiple base classifiers. One popular example is random forests (Breiman, 2001), which combine multiple decision-tree classifiers by majority voting:

$$f(\mathbf{x}) := \text{Majority}\{f_1(\mathbf{x}), \ldots, f_N(\mathbf{x})\}. \tag{1}$$

Ensemble models are useful in an enormous variety of applications, and tree-based ensembles obtain top marks for many kinds of tabular data (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022). Choosing the size of the model presents a tradeoff: larger ensembles are more accurate but take longer to train and run. Constraints that apply at testing time can be a concern when the model is to be run on low-power devices or very frequently (e.g., spam filters) (Xu et al., 2013). In this paper, we present a method to sharply reduce the test time of large ensembles using a sequential testing approach which we illustrate in the following example:

Suppose we have trained a random forest with 101 trees to solve a binary classification problem. At test time, we execute one tree at a time and—once all have been executed—we take the majority result. If it so happens that the first 51 trees all return a positive classification, then the result of the majority vote must be positive and the execution can be stopped at that point. Even if, having executed only 20 trees, 19 of them have returned a positive result, we can still posit that the ensemble is likely to return a positive label. Thus, by stopping early we can save time at the cost of occasionally disagreeing with the full ensemble. The example above prompts our central question:

What is the speed-up one can achieve for a given disagreement rate using early stopping, and how? (2)

In this paper, "speed-up" is measured in base models executed, and "disagreement rate" is the probability of disagreement with the complete ensemble. The answer to this question is a *stopping strategy* that tells us, for each possible state during the execution of the ensemble, whether we should stop and return the current majority. A state, in this context, is a pair (i, j), where i is the number of base models executed so far, and j is the number of base models that returned positive.

Figure 1 shows state transition graphs for an ensemble with $N = 3$ base models. Panel (a) shows possible state transitions when classifying an input for which $n = 1$ base models return positive, whereas in panel (b) all $n = 3$ base models return positive. In both cases, execution starts in the state $(0, 0)$. From there,

its progress may be nondeterministic due to our assumption that the base models are executed in a random order. As this toy example demonstrates, the same stopping strategy may produce a different distribution of state paths depending on the number of positive base models in the ensemble, n. Since this number is typically unknown before the ensemble is executed, we must somehow account for all possible values of n. In this paper we present three approaches to this conundrum: a *minimax* worst-case approach, a *minimean* expected-case approach, and a *minimixed* approach which combines the two.

1.1 Our Contribution

In this paper we develop several methods for constructing early stopping strategies for ensemble binary classifiers of the form (1) such as random forests. To do this, we pose the search for a stopping strategy as a minimization problem of the expected stopping time, subject to a constraint that the majority vote of the stopped ensemble must agree with that of the original ensemble with high probability. Our approach is applicable to any binary classifier based on a majority vote. The only assumption we make is that the base models are executed in a random order, which simplifies the early-stopping problem, but does exclude ensembles where base models must be executed in a certain order. On the other hand, we avoid other assumptions that have been made in prior work, such as independence of base models.

In Section 2, we present the theory underlying our methods, formalizing the notion of an early-stopping strategy and constructing linear optimization problems whose solution yields optimal stopping strategies. In Section 3 we synthesize this theory into a concrete algorithm and discuss computation time and implementation details. In Section 4, we present performance metrics for random forests using our methods on a variety of datasets. We end with a discussion of applications and directions for further research in Section 5.

1.2 Related Work

Early stopping is closely related to sequential testing, a field of classical statistics wherein a null hypothesis is tested against an alternative hypothesis by analyzing each observation in sequence and stopping as soon as a conclusion can be reliably drawn (Wald, 1945; Siegmund, 1985). It is commonly used in contexts such as quality control and medical trials where reaching conclusions as early as possible is key. Our problem is closely related to the special case of sequential testing with binary outcomes, for which Silva et al. (2020) present a solution that is provably optimal in the sense that interests us. However, their work is not directly applicable to our problem because their method assumes independence of the base models, an assumption that we do not wish to make; for further discussion see Section 5 of the supplementary material. Lai (1979) takes a very similar approach to us—sequential testing on binary data for finite populations, using a hypergeometric model—but does not present a strategy that is provably optimal.

Our central example of an ensemble model, the random forest, was first presented in its modern form by Breiman (2001). Since then, several approaches to reduce their prediction time were suggested. One popular approach is to prune the forest by either deleting individual branches or entire trees during training (Manzali & Elfar, 2023; Surjanovic et al., 2024). There are several works that consider early-stopping strategies for ensemble models: the method of Wang et al. (2021) is based on dynamically reordering the base models during testing, combined with an early-stopping strategy. Fan et al. (2002) present pruning, dynamic-ordering and early-stopping techniques specifically for scenarios with asymmetric loss. An approach to early-stopping that is similar to ours was presented by Schwing et al. (2011), but it is based on binomial confidence bands under the simplifying assumption that base models are independent of each other, an assumption that does not hold for random forests. Another recent work is that of Daghero et al. (2023), where they consider a heuristic approach to early stopping with no theoretical guarantees. To the best of our knowledge, our method is the first to achieve provably-optimal results on majority-based binary ensemble classifiers, without making further assumptions such as independence. The only assumption that we make is that the base models $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_N(\mathbf{x})$ are executed in a random order.

Table 1: Glossary of symbols and terms.

Symbol	Meaning
N	Total number of base models in the ensemble
n	Number of base models that would return a positive classification on the current input
D	True probability distribution of n, given a trained ensemble
$\hat{D}$	Estimated or assumed probability distribution of n
state (i, j)	State wherein i base models have been executed and j have returned a positive classification
θ	Stopping strategy: on reaching a state (i, j), the ensemble will stop with probability θ_{ij}
Θ_N	The set of valid stopping strategies for ensembles of size N
B	Number of base models executed before stopping
disagreement rate, Pr(disagree)	Probability that the early-stopped ensemble gives a different result than the full ensemble
ADR, α	Allowable Disagreement Rate, the user-defined maximum disagreement rate
reach (i, j)	Event that the ensemble enters the state (i, j) while running, whether or not it stops there
stop at (i, j)	Event that the ensemble enters the state (i, j) while running and stops there
W_{ij}	Event that the ensemble *would* enter the state (i, j) if not stopped; equivalently, that precisely j of the first i base models are positive
$d_{ij}(N, n)$	Indicator of whether the state (i, j) is a disagreement state
$p, \pi, \bar{\pi}$	Decision variables representing conditional probabilities induced by θ (see (14))
Π	Set of valid values for $(p, \pi, \bar{\pi})$ (see (15))
$[\textit{expression}]$	Iverson bracket; evaluates to 1 if *expression* is true, otherwise to 0

2 Theory

In this section, we formalize the central question (2) posed in the introduction, as follows: first, in Section 2.1, we formally define the notion of a stopping strategy and motivate the metrics we use to compare strategies. Next, in Section 2.2, we show how these metrics can be computed. We then describe several optimization problems in Section 2.3, each of which finds the stopping strategy which is optimal in a different sense. Finally, in Section 2.4, we reformulate them as linear programs that can be efficiently solved. For the reader's convenience, common symbols and terms used in this Section are summarized in Table 1.

2.1 Stopping Strategies and Their Evaluation

Consider a binary ensemble classifier with N base models that are executed in a random order. A *deterministic stopping strategy* is a boolean matrix $\theta \in \{0, 1\}^{(N+1)\times(N+1)}$, indexed from zero in both dimensions. The entry θ_{ij} tells us what to do if, during testing, we reach a state where precisely $i \in \{0, \ldots, N\}$ base models have been executed and precisely j of them have returned positive: if $\theta_{ij} = 1$ we stop early and return the majority result so far $[j > i/2]$, and if $\theta_{ij} = 0$, we continue executing base models. A more general (nondeterministic) *stopping strategy* is a matrix of real values $\theta \in [0, 1]^{(N+1)\times(N+1)}$, still indexed from zero. We interpret it the same way, except that now θ_{ij} tells us the *probability* with which we should stop if we

reach the relevant state. Since testing *must* stop once the ensemble is exhausted, the set of valid stopping strategies Θ_N includes only those for which $\theta_{Nj} = 1$ for all j.

Given a stopping strategy and a certain input $\mathbf{x}$ being classified, where will we actually stop? The answer to this is a probability distribution, both because the stopping strategy itself may be nondeterministic and because the order in which the base models are executed is random. We will soon show how we can compute the probability of stopping at any given state (i, j), conditional on the ensemble size N, the number n of base models returning positive, and the stopping strategy $\theta \in \Theta_N$. We denote this as $\Pr(\text{stop at } (i, j) \mid N, n, \theta)$. Note that we need not condition on $\mathbf{x}$ because $\mathbf{x}$ affects the stopping behavior only through n. From these probabilities, we can compute two metrics that are of interest in the evaluation of stopping strategies. First is the number of base models executed before stopping, denoted B, or rather its expectation $\mathbb{E}[B \mid N, n, \theta]$. Since the number of base models executed so far is the same as the step number i, this expectation is given by

$$\begin{aligned}\mathbb{E}[B \mid N, n, \theta] &= \sum_{i=0}^{N} i \cdot \Pr(\text{stop at step } i \mid N, n, \theta) \\ &= \sum_{i=0}^{N} i \cdot \sum_{j=0}^{i} \Pr(\text{stop at } (i, j) \mid N, n, \theta). \end{aligned} \tag{3}$$

The second metric of interest is the *disagreement rate*: the probability of the sequentially-tested ensemble yielding a different result from that of the complete ensemble. This occurs if and only if we stop in a "disagreement state", that is, a state where the majority result so far is different from that of the ensemble as a whole. We indicate such states using the Boolean matrix $d(N, n)$, defined as

$$d_{ij}(N, n) := [j > i/2] \neq [n > N/2], \tag{4}$$

where $[\cdot]$ is the Iverson bracket; for the purpose of our analysis we assume that tied votes yield a negative prediction. Using this notation, the disagreement rate is

$$\Pr(\text{disagree} \mid N, n, \theta) = \sum_{i=0}^{N} \sum_{j=0}^{i} \Pr(\text{stop at } (i, j) \mid N, n, \theta) d_{ij}(N, n). \tag{5}$$

Our decision to minimize disagreement rate rather than error rate comes from the observation that the error rate of the early-stopped ensemble is highly dependent on that of the complete ensemble, over which we have no control. Intuitively, however, we would not expect early stopping to reduce the error rate in the general case, but rather to increase it. The disagreement rate acts as a bound on the excess error rate introduced by early stopping, i.e.,

$$\Pr(\text{error with early-stopping}) \leq \Pr(\text{error with full ensemble}) + \Pr(\text{disagree}). \tag{6}$$

2.2 Computing Conditional Stopping Probabilities

We now explain how to compute the probability of stopping at a state (i, j) conditional on N, n, θ. Applying the results in Section 2.1 will then allow us to express the metrics of interest as a function of these parameters, and thus to formulate our optimization problems in Section 2.3.

For brevity, we leave the conditioning on N, n, θ implicit in all the probabilities that follow in this subsection. The probability of stopping at a given state is simply

$$\Pr(\text{stop at } (i, j)) = \Pr(\text{reach } (i, j)) \cdot \theta_{ij}. \tag{7}$$

The probability of *reaching* a given state can be computed step-by-step using a recursive formula. The base case is $\Pr(\text{reach } (0, 0)) = 1$. From each state (i, j), we transition to the state $(i + 1, j + 1)$ if the next base

model returns positive, and to $(i+1, j)$ if it returns negative. Since base models are randomly ordered, the probability of it being one of those that returns positive is

$$\Pr((i+1)\text{-th model is positive} \mid \text{reach } (i,j)) = \tfrac{n-j}{N-i}. \tag{8}$$

Flipping our perspective, each state (i, j) where $i > 0$ can be reached only through state $(i-1, j-1)$ or through the state $(i-1, j)$. Both of these paths can only occur if we do not stop in the $(i-1)$-th state. Therefore the probability of the first path, wherein $f_i(\mathbf{x})$ is positive, is

$$\begin{aligned}&\Pr(\text{reach } (i,j) \text{ through } (i-1,j-1))\\ &\quad = \Pr(\text{reach } (i-1,j-1)) \cdot (1-\theta_{i-1,j-1}) \cdot \tfrac{n-(j-1)}{N-(i-1)},\end{aligned} \tag{9}$$

and likewise the probability of the second path, wherein $f_i(\mathbf{x})$ is negative, is

$$\begin{aligned}&\Pr(\text{reach } (i,j) \text{ through } (i-1,j))\\ &\quad = \Pr(\text{reach } (i-1,j)) \cdot (1-\theta_{i-1,j}) \cdot \left(1 - \tfrac{n-j}{N-(i-1)}\right).\end{aligned} \tag{10}$$

The total probability of reaching state (i, j) is simply the sum of these two terms,

$$\begin{aligned}&\Pr(\text{reach } (i,j))\\ &\quad = \Pr(\text{reach } (i,j) \text{ through } (i-1,j))\\ &\quad + \Pr(\text{reach } (i,j) \text{ through } (i-1,j-1))\\ &\quad = \Pr(\text{reach } (i-1,j-1)) \cdot (1-\theta_{i-1,j-1}) \cdot \tfrac{n-(j-1)}{N-(i-1)}\\ &\quad + \Pr(\text{reach } (i-1,j)) \cdot (1-\theta_{i-1,j}) \cdot \left(1 - \tfrac{n-j}{N-(i-1)}\right).\end{aligned} \tag{11}$$

Combining (11) with the base case above ($\Pr(\text{reach } (0,0)) = 1$) allows us to compute $\Pr(\text{reach } (i,j))$ for any (i, j), from which $\Pr(\text{stop at } (i,j))$ is derived for all i, j via Eq. (7). Finally, by plugging $\Pr(\text{stop at } (i,j))$ into equations (3) and (5), we obtain the expected runtime and disagreement rate of the stopping strategy θ, given n.

2.3 Finding Optimal Stopping Strategies

Having expressed the disagreement rate and expected runtime as a function of N, n, θ, we can now express our optimization problem: we wish to find the stopping strategy which minimizes the expected runtime while keeping the disagreement rate below a specified threshold α, i.e.,

$$\underset{\theta \in \Theta_N}{\text{minimize}}\ \mathbb{E}[B] \quad \text{s.t.} \quad \Pr(\text{disagree}) \leq \alpha \tag{12}$$

or, more explicitly,

$$\underset{\theta \in \Theta_N}{\text{minimize}}\ \mathbb{E}[B \mid N, n, \theta] \quad \text{s.t.} \quad \Pr(\text{disagree} \mid N, n, \theta) \leq \alpha. \tag{13}$$

But this problem is not immediately solvable, because we don't know n; therefore, we must aggregate over *possible* values of n. We can do this either by considering worst-case values of n or—if we are willing to assume a distribution on n—by taking an expectation. We consider three approaches to finding optimal stopping strategies. First, we have the *minimax approach*, where we minimize the worst-case expected runtime among all strategies whose worst-case disagreement rate is at most α:

Problem: Θ-Minimax (1)

$$\underset{\theta \in \Theta_N}{\text{minimize}}\ \max_{n}\ \mathbb{E}[B \mid N, n, \theta] \qquad \text{s.t.} \qquad \max_{n} \Pr(\text{disagree} \mid N, n, \theta) \leq \alpha.$$

Next, we have the *minimean approach*, where we estimate or assume some distribution $\hat{D}$ for n and minimize the overall expected runtime among all strategies whose expected disagreement rate is below α:

Problem: Θ-Minimean (2)

$$\underset{\theta \in \Theta_N}{\text{minimize}}\ \mathbb{E}_{n \sim \hat{D}}[\mathbb{E}[B \mid N, n, \theta]] \qquad \text{s.t.} \qquad \mathbb{E}_{n \sim \hat{D}}[\Pr(\text{disagree} \mid N, n, \theta)] \leq \alpha.$$

Finally, we have the *minimixed approach*, where we assume some distribution for n and minimize the overall expected runtime among all strategies whose worst-case disagreement rate is below α:

Problem: Θ-Minimixed (3)

$$\underset{\theta \in \Theta_N}{\text{minimize}}\ \mathbb{E}_{n \sim \hat{D}}[\mathbb{E}[B \mid N, n, \theta]] \quad \text{s.t.} \quad \max_n \Pr(\text{disagree} \mid N, n, \theta) \leq \alpha.$$

As discussed in Section 2.1, the quantities $\mathbb{E}[B \mid N, n, \theta]$ and $\Pr(\text{disagree} \mid N, n, \theta)$ in our optimization problems are both linear combinations of $\Pr(\text{stop at } (i, j))$ for various values of (i, j). These probabilities are in turn easily computable from the formulae in Section 2.2. However, our expression for $\Pr(\text{stop at } (i, j))$ in Equation (7) depends on that for $\Pr(\text{reach } (i, j))$, which in turn is expressed using a recursive formula in Equation (11), so $\Pr(\text{stop at } (i, j))$ involves the product of multiple decision variables. Thus Problems (1), (2), and (3) are not linear programs, nor even quadratically-constrained quadratic programs (QCQPs). However, they can be reformulated as QCQPs with $O(N^3)$ decision variables. Unfortunately, in this formulation the problems are impractical to solve even for $N \approx 16$ (for more on this reformulation and our experiments, see Section 3 of the supplementary material). In the next subsection we explain how these optimization problems can be solved efficiently by reformulating them as linear programs.

2.4 Optimization by Linear Programming

Real ensembles often have hundreds of base models, so a more computationally-tractable approach is necessary. For this purpose we use a technique from Silva et al. (2020). Essentially, we reformulate the problems in terms of a new set of decision variables representing conditional probabilities induced by θ. With these decision variables, Problems (1), (2), and (3) can easily be expressed as linear programs and solved with known efficient techniques (Hillier & Lieberman, 2024). Once the linear program has been solved, the optimal values of θ can be easily extracted from those of the new decision variables. We call the new decision variables $p_{ij}, \pi_{ij}, \bar{\pi}_{ij}$, and assign them the following meanings:

$$\begin{aligned} p_{ij} &:= \Pr(\text{reach } (i, j) \mid N, n, \theta, W_{ij}) \\ \pi_{ij} &:= \Pr(\text{stop at } (i, j) \mid N, n, \theta, W_{ij}) \\ \bar{\pi}_{ij} &:= \Pr(\text{reach } (i, j) \text{ and not stop at } (i, j) \mid N, n, \theta, W_{ij}), \end{aligned} \tag{14}$$

where W_{ij} is the event that the state (i, j) *would* be reached if no stopping strategy were used—that is, that precisely j of the first i base models are positive. Conditioning on this event as well as n makes the value of these conditional probabilities the same for all n, which is why they are indexed by (i, j) rather than (i, j, n). Note that W_{ij} has probability zero whenever $j > i$; we adopt the convention that conditioning on an event of probability zero yields a probability of zero, so in those cases $p_{ij} = \pi_{ij} = \bar{\pi}_{ij} = 0$. The meanings of the decision variables imply certain conditions; we define Π as the set of triplets satisfying these conditions, i.e.:

Definition of Π

$$\Pi := \left\{ (p, \pi, \bar{\pi}) \in \left([0,1]^{(N+1)\times(N+1)} \right)^3 \;\middle|\; (15.1)\text{–}(15.5) \text{ hold} \right\} \tag{15}$$

$$p_{00} = 1 \tag{15.1}$$

$$p_{ij} = \pi_{ij} = \bar{\pi}_{ij} = 0 \text{ for } i = 0, \ldots, N,\ j = i+1, \ldots, N \tag{15.2}$$

$$\pi_{ij} + \bar{\pi}_{ij} = p_{ij} \text{ for } i = 0, \ldots, N,\ j = 0, \ldots, i \tag{15.3}$$

$$p_{i+1,j+1} = \frac{i-j}{i+1}\bar{\pi}_{i,j+1} + \frac{j+1}{i+1}\bar{\pi}_{ij} \text{ for } i = 0, \ldots, N-1,\ j = 0, \ldots, i \tag{15.4}$$

$$\pi_{Nj} = p_{Nj} \text{ for } j = 0, \ldots, N. \tag{15.5}$$

Conditions (15.1), (15.2), and (15.3) follow directly from (14). Condition (15.4) is due to the way each state follows only from its two antecedent states. Condition (15.5) is a reformulation of the constraint $\theta_{Nj} = 1$, reflecting the fact testing must halt once the ensemble is exhausted.

The following lemmas, proofs of which may be found in Section 1 of the supplementary material, express the fact that membership in Π is both necessary and sufficient for a given $p, \pi, \bar{\pi}$ to satisfy the definition in (14) for some stopping strategy θ.

Lemma 1 (Necessity)**.** *For each $N, n, \theta \in \Theta_N$, if $p, \pi, \bar{\pi}$ are defined as in Equation (14), then $(p, \pi, \bar{\pi}) \in \Pi$.*

Lemma 2 (Sufficiency)**.** *Fix some N, n. For any $(p, \pi, \bar{\pi}) \in \Pi$, there exists some stopping strategy $\theta \in \Theta_N$, computable in $O(N^2)$ time from $(p, \pi, \bar{\pi})$, such that Equation (14) holds.*

Next, Equation (14) allows us to express the expected runtime and disagreement rate as linear combinations of π. Recall that in Section 2.1 we saw

$$\mathbb{E}[B \mid N, n, \theta] = \sum_{i=0}^{N} i \cdot \sum_{j=0}^{i} \Pr(\text{stop at } (i,j)), \tag{16}$$

$$\Pr(\text{disagree} \mid N, n, \theta) = \sum_{i=0}^{N} \sum_{j=0}^{i} \Pr(\text{stop at } (i,j))[[j > i/2] \neq [n > N/2]]. \tag{17}$$

The only term in these sums that depends on θ is $\Pr(\text{stop at } (i,j))$,

$$\Pr(\text{stop at } (i,j)) = \Pr(\text{stop at } (i,j) \mid W_{ij}) \Pr(W_{ij}) = \pi_{ij} \Pr(W_{ij}). \tag{18}$$

The quantity $\Pr(W_{ij})$—the probability that precisely j of the first i base models are positive—distributes hypergeometrically, i.e.,

$$\Pr(W_{ij}) = \frac{\binom{n}{j}\binom{N-n}{i-j}}{\binom{N}{i}}, \tag{19}$$

and does not depend on θ. That makes the following a *linear* optimization problem:

Problem: Π-Minimax (reformulation of Θ-Minimax) (4)

$$\underset{(p,\pi,\bar{\pi})\in\Pi}{\text{minimize}} \quad \max_n \mathbb{E}[B \mid N, n, \theta] \quad \text{s.t.} \quad \max_n \Pr(\text{disagree} \mid N, n, \theta) \leq \alpha$$

where θ is implicitly given by $(p, \pi, \bar{\pi})$: due to the probabilities each variable represents, we have $\theta_{ij} = \pi_{ij}/p_{ij}$ wherever $p_{ij} \neq 0$, and an arbitrary value when $p_{ij} = 0$. Problem Π-Minimax (4) is thus equivalent to Problem Θ-Minimax (1) by the following theorem, a proof of which can be found in Section 1 of the supplementary material.

Theorem 3. *Problem Θ-Minimax (1) is polynomially reducible to Problem Π-Minimax (4) in $O(N^2)$ time.*

By an analogous argument, we can reformulate Problem Θ-Minimean (2) as

Problem: Π-Minimean (reformulation of Θ-Minimean) (5)

$$\underset{(p,\pi,\bar{\pi})\in\Pi}{\text{minimize}}\ \mathbb{E}_{n\sim\hat{D}}[\mathbb{E}[B \mid N, n, \theta]] \quad \text{s.t.} \quad \mathbb{E}_{n\sim\hat{D}}[\Pr(\text{disagree} \mid N, n, \theta)] \leq \alpha.$$

We can likewise reformulate Problem Θ-Minimixed (3) as

Problem: Π-Minimixed (reformulation of Θ-Minimixed) (6)

$$\underset{(p,\pi,\bar{\pi})\in\Pi}{\text{minimize}}\ \mathbb{E}_{n\sim\hat{D}}[\mathbb{E}[B \mid N, n, \theta]] \quad \text{s.t.} \quad \max_{n} \Pr(\text{disagree} \mid N, n, \theta) \leq \alpha.$$

We now have a method to compute optimal stopping strategies of three different kinds: by solving Problem Π-Minimax (4), we can find a stopping strategy that minimizes runtime and guarantees the disagreement rate *in the worst case*; by solving Problem Π-Minimean (5) we can find a stopping strategy that minimizes runtime and guarantees the disagreement rate *in expectation*, provided we have a distribution for n; or, by solving Problem Π-Minimixed (6), we can find a stopping strategy that minimizes runtime in expectation while guaranteeing the disagreement rate in the worst case.

3 Computation

The theory in Section 2 comes together to produce algorithms for finding stopping strategies which are optimal in the minimean, minimax, or minimixed sense (see Table 2). In this Section, we discuss some computational details. Detailed pseudocode can be found in Section 2 of the supplementary material. A link to our Python implementation can be found in Section 6.

The basic outline of the algorithm is the same regardless of whether we take a minimax, minimean, or minimixed approach: we construct a linear program, solve it, and convert the solution to a stopping strategy. The time complexity of setting up the linear program is $O(N^3)$, and the time complexity of converting the solution to a stopping strategy is $O(N^2)$. The bulk of the time complexity in finding a stopping strategy is thus in solving the linear program. The most time-efficient algorithm of which we are aware for doing this is that given by van den Brand (2020), which takes $\tilde{O}(v^\omega \log(v/\delta))$ time, where v is the number of decision variables, δ is the required relative precision, ω is the exponent of the time complexity of matrix multiplication, and $\tilde{O}$ hides a poly-log factor.

We used the linear program solver SoPlex 6.0.0, part of the SCIP Optimization Suite 8.0.0 (Bestuzheva et al., 2023). Since the state transition probabilities are all ratios of integers, we used the rational solver options to avoid loss of precision on large ensembles (Gleixner et al., 2016; Gleixner & Steffy, 2020; Eifler

Table 2: **Methods.** Early-stopping approaches analyzed in this paper. Average case is with regard to an assumed or estimated distribution of n (the number of positive votes in the ensemble). We tested the minimean and minimixed approaches both on the empirical distribution of a holdout calibration set (Cal) and on a uniform distribution (Flat).

Name	Optimizes runtime for...	Bounds disagreement rate of...	Formulation
Minimax	Worst case	Worst case	Problem (4)
Minimean	Average case	Average case	Problem (5)
Minimixed	Average case	Worst case	Problem (6)
Schwing et al.	—	—	Schwing et al. (2011) §3.1

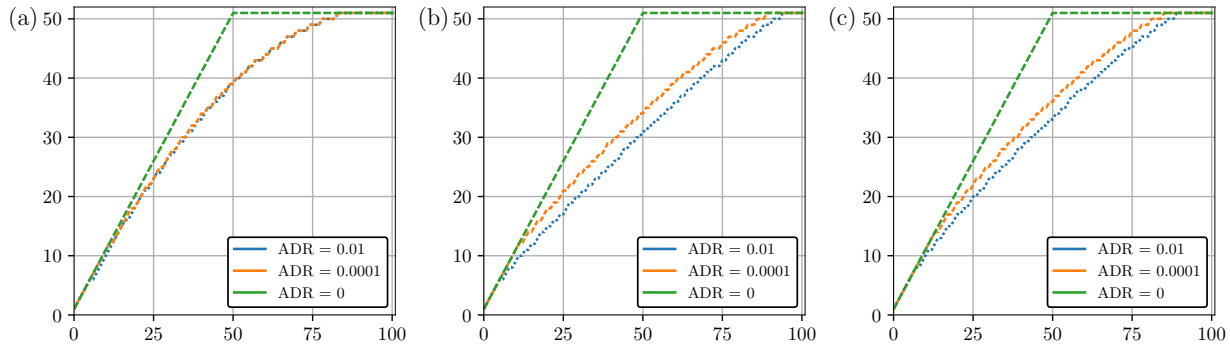

Figure 2: Envelopes of precomputed stopping strategies for ensembles of size $N = 101$ and various values of the ADR (allowable disagreement rate) α. The horizontal axis is i, the number of base models executed; the vertical axis is j, the number of those which were positive. The plotted lines represent b_i, the number of positive votes which must be reached to stop. **(a)** Minimax **(b)** Minimean **(c)** Minimixed.

et al., 2023). Empirically, solving the optimization problems takes less than a minute up to $N = 100$; see Section 3 of the supplementary material for technical details and solution times.

4 Empirical Results

4.1 Computed Stopping Strategies

A key difference between the minimax approach and the other two approaches presented in Section 2.3 is that computing a minimax strategy requires us to specify only the size of the ensemble N and the allowable disagreement rate α, whereas for a minimean or minimixed strategy we must also supply an estimated or assumed distribution $\hat{D}$ of the number of positive base models n. Therefore, only minimax stopping strategies can be precomputed without regard for the data.

We precomputed minimax stopping strategies for all ensemble sizes $11 \leq N \leq 121$, for allowable disagreement rates of $\alpha \in \{0, 10^{-6}, 10^{-5}, \ldots, 10^{-1}\}$. We did likewise for minimean and minimixed stopping strategies under the assumption of uniformly distributed n. The precomputed strategies are available as part of our GitHub repository. In practice, all precomputed stopping strategies were almost entirely deterministic—that is, all but two entries θ_{ij} were either zero or one. Moreover, they all followed the same simple stopping rule: each strategy had some step i_0 before which there was no stopping at all, and after which stopping was certain iff the observed number of positives j was greater than b_i or less than a_i for thresholds a_i and b_i which were different for each stopping strategy. Nondeterminism only emerged at step i_0 itself, where the strategy was to stop with some positive probability iff the consensus so far was unanimous. For minimax stopping strategies, and for minimean and minimixed strategies based on a symmetrical distribution of n, the strategies themselves were also symmetrical, which is to say they satisfied $b_i - i/2 = i/2 - a_i$ for all i. This simple shape means we lose almost no information by plotting each stopping strategy as a single line giving b_i for each i, or a trivial threshold of $i + 1$ for $i < i_0$. We refer to this as the *envelope* of the stopping strategy. Envelopes of selected stopping strategies computed using the minimax, minimean, and minimixed approaches are shown in Figure 2. When the ADR is zero, the envelope starts at $i + 1$—in other words, never stop, because j can never exceed i anyway—until we have observed a majority of the entire ensemble (51 base models); thereafter, we stop iff we have 51 votes in the same direction, since then and only then can the result of the whole ensemble be predicted with certainty. The envelopes for higher ADRs are lower, representing more aggressive stopping strategies.

Table 3: Datasets used in this paper. "ID" refers to the dataset's ID in the UCI Machine Learning Repository. # is number of observations, p is number of features, and C is number of classes before dichotomization. "%pos" is the percentage belonging to the largest class, which forms the positive class after dichotomization.

Name (ID)	#	p	C	%pos	Description
Ground Cover (31)	581,012	54	7	48.8	Classification of pixels into forest cover types based on attributes such as elevation, slope, hillshade, soil-type, etc.
Income (117)	199,523	41	2	93.8	Weighted census data extracted from the 1994 and 1995 current population surveys conducted by the U.S. Census Bureau. Target variable is whether income is $\geq$ \$50k / year.
Diabetes (891)	253,680	21	2	86.1	Features are demographics, lab test results, and answers to survey questions. Target variable is whether a patient has diabetes, is pre-diabetic, or is healthy.
Skin (229)	245,057	3	2	79.2	RGB color triplets must be classified as skin or non-skin colors, based on samples from a variety of ages and ethnicities.
Sepsis (827)	110,341	3	2	92.6	Admissions of hospitalized subjects between 2011 and 2012 in Norway who were diagnosed with infections, systemic inflammatory response syndrome, sepsis by causative microbes, or septic shock. Target is vital status 9 days later.
Dota2 (367)	102,944	115	2	52.7	Dota 2 is a computer game with two teams of 5 players. At the start of the game each player chooses a unique hero with different strengths and weaknesses. Each row represents a single game. Heroes chosen by one team are given value 1, those chosen by the other team are given value -1, and the remainder have value 0. Target is the result of the match.
Hospitalization (296)	101,766	47	3	53.9	Ten years (1999–2008) of clinical care at 130 US hospitals. Each row represents a patient diagnosed with diabetes, who was hospitalized and discharged. The goal is to determine the early readmission of the patient within 30 days of discharge.
Shuttle (148)	58,000	7	7	78.6	This data set was generated originally to extract comprehensible rules for determining the conditions under which an autolanding would be preferable to manual control of a spacecraft. The task is to decide what type of control to use.

4.2 Empirical Evaluation

We tested each early-stopping approach on random forests trained on twenty-seven different real-world datasets, with allowable disagreement rates ranging from 0 to 0.1 on each dataset. The main collection of eight datasets was taken from the UC Irvine Machine Learning Repository (Kelly et al.). Datasets were only included if they were tagged with the "Classification" task, had a column marked with the "Target" role, had no missing data, were non-synthetic, and were available for import in Python. Among datasets matching these criteria, the eight largest datasets (by number of instances) were selected. The selected datasets, ranging in size from 58k to 581k observations, are detailed in Table 3. In addition to these, we also ran all experiments on those benchmark datasets proposed by Grinsztajn et al. (2022) which were marked as appropriate for classification problems, of which there were nineteen; for readability, some figures in the main paper exclude these datasets, but the results were substantially similar to the main eight, and more complete figures appear in Section 6 of the supplementary material. Since our work only covers binary classification, datasets that had more than one response class were dichotomized by merging all classes except the largest. The observations were split into "train", "test", and "calibration" sets in a ratio of 70/10/20, and a random forest with 101 trees was fitted to the training set using the `RandomForestClassifier` from `scikit-learn` (Pedregosa et al., 2011), with default values for all hyperparameters other than the number of trees. Stopping strategies for ensembles of 101 base models were computed using the minimax, minimean, and minimixed approaches described above. Allowable disagreement rates ranged from 10^{-6} to 10^{-1} in logarithmic increments (and also included 0). The minimean and minimixed strategies require as input an estimated or assumed distribution of the number of positive base models in the ensemble; we tested them both on the empirical distribution of the calibration set and on a uniform (flat) distribution. Note that while

we expect (and observe) worse performance from the uncalibrated uniform distribution, it has the practical advantage of allowing the stopping strategy to be precomputed; see Section 4.1.

The "base" error rate is the error rate of the full ensemble on the test set; for the purposes of this experiment, the result of the full ensemble was computed by majority vote rather than by `scikit-learn`'s default behavior, which uses probability-averaging. The test set was also used to estimate the *conditional* distribution of number of positive base models (trees) for each true response class. The expected runtime (as measured in number of base models executed) and disagreement rate were computed analytically for each possible number of positive base models; weighted by the conditional distributions, this gave us an estimated expected runtime, disagreement rate, and error rate for each stopping strategy. Note that these estimated conditional distributions were used only for estimating these metrics after the stopping strategies were computed, and not for computing the stopping strategies themselves. This entire process, from train-test split to computation of metrics of interest, was repeated 30 times and averaged to reduce variance. The results are presented in Figure 3. Figure 4 shows the tradeoff of runtime versus error rate on all the datasets considered thus far and 19 additional datasets (see supplementary). The method of Schwing et al. (2011) based on binomial confidence bands is included in both figures as a baseline for comparison.

We can see in Figure 3 that if an ADR of 10^{-3} or less is used, significant time can be saved with negligible excess error using any one of the approaches. With regard to runtime, we see greater savings for the minimean approach than the minimax approach, as we would expect since the minimax approach is conservative by definition. The runtime of the calibrated minimixed approach falls between the minimax and the calibrated minimean approach, and likewise for the uncalibrated approaches, as we would expect. A stopping strategy with an ADR of 10^{-3} will often reduce runtime by 70% to 90% with the minimax approach, and between 80% to 99% with a minimean approach using a calibration set. The exceptions were the "Dota2" and "Hospitalization" datasets, where time savings were smaller.

How can early stopping cut out so many base models without appreciably affecting the ensemble's output? And why does this fail on certain datasets? The answer is that, empirically for most forest-input pairs, the vast majority of trees vote the same way; this phenomenon is illustrated in Figure 5. However, when the problem is particularly difficult, the distribution of tree votes is much more centralized, increasing the base ensemble's error rate *and* making a consensus less likely early in the test. This is neatly illustrated by the datasets with poor runtime, "Dota2" and "Hospitalization": Figure 3 shows that the base error rates of these datasets were much higher than the others', and Figure 5 shows that their vote distributions were much closer to unimodal than the others'.

This observation also explains why the uncalibrated (flat) minimean approach generally falls between the minimax approach and the well-calibrated minimean approach in runtime. On most datasets, most ensembles are either overwhelmingly positive or overwhelmingly negative. The well-calibrated minimean approach, constructed with this information in mind, can therefore stop after only a few base models have been executed, since it is quickly obvious which of these two clusters it finds itself in. Conversely, the uncalibrated minimean approach presumes a flat distribution, so early results are taken as less indicative of the whole ensemble, making the stopping strategy more conservative, but still less so than the minimax, which by construction is *maximally* conservative. This is also why the uncalibrated minimean approach is conservative on some datasets but permissive on others: when the true distribution of n has a single central cluster, as in the "Dota2" and "Hospitalization" datasets, then the previous logic is reversed, and the assumption of a flat distribution leads to a stopping strategy that treats each vote as *more* predictive of other votes than it actually is.

Taken together, these results suggest that a minimean approach using a held-out calibration set with $\alpha = 10^{-3}$ should produce good results for many datasets. Table 4 summarizes this approach's performance on the datasets we tested. Expected runtime is expressed as a proportion of $N = 101$, though this proportion tends to shrink as the size of the ensemble grows. "Base error rate" is the misclassification rate of the entire ensemble without early stopping.

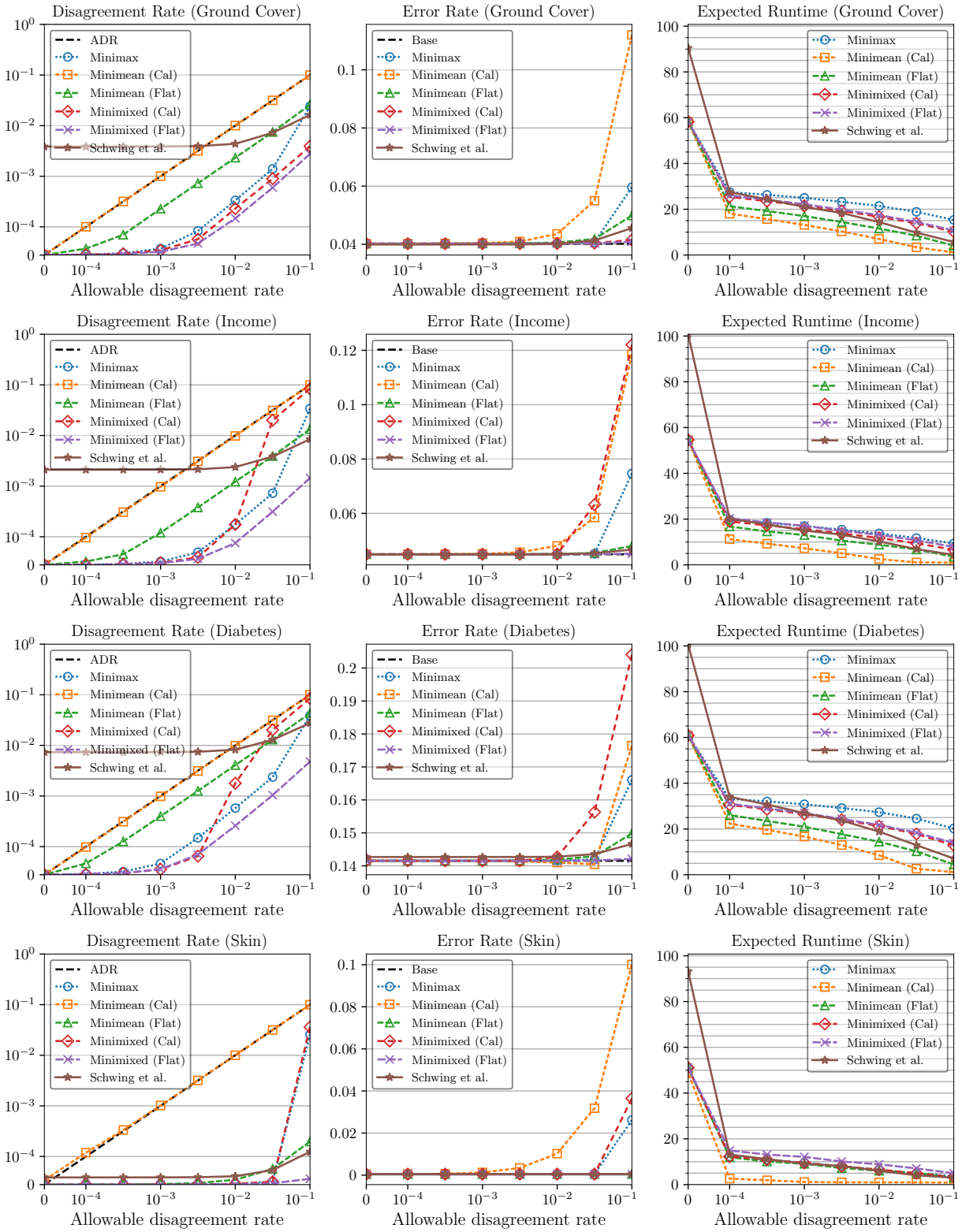

Figure 3: Performance of six early-stopping approaches for random forests of 101 trees on real-world datasets. Each row is a dataset, and each column is a metric. "ADR" is allowable disagreement rate. "Expected runtime" means expected number of trees executed before stopping. The approaches are listed in Table 2. Page 1 of 2.

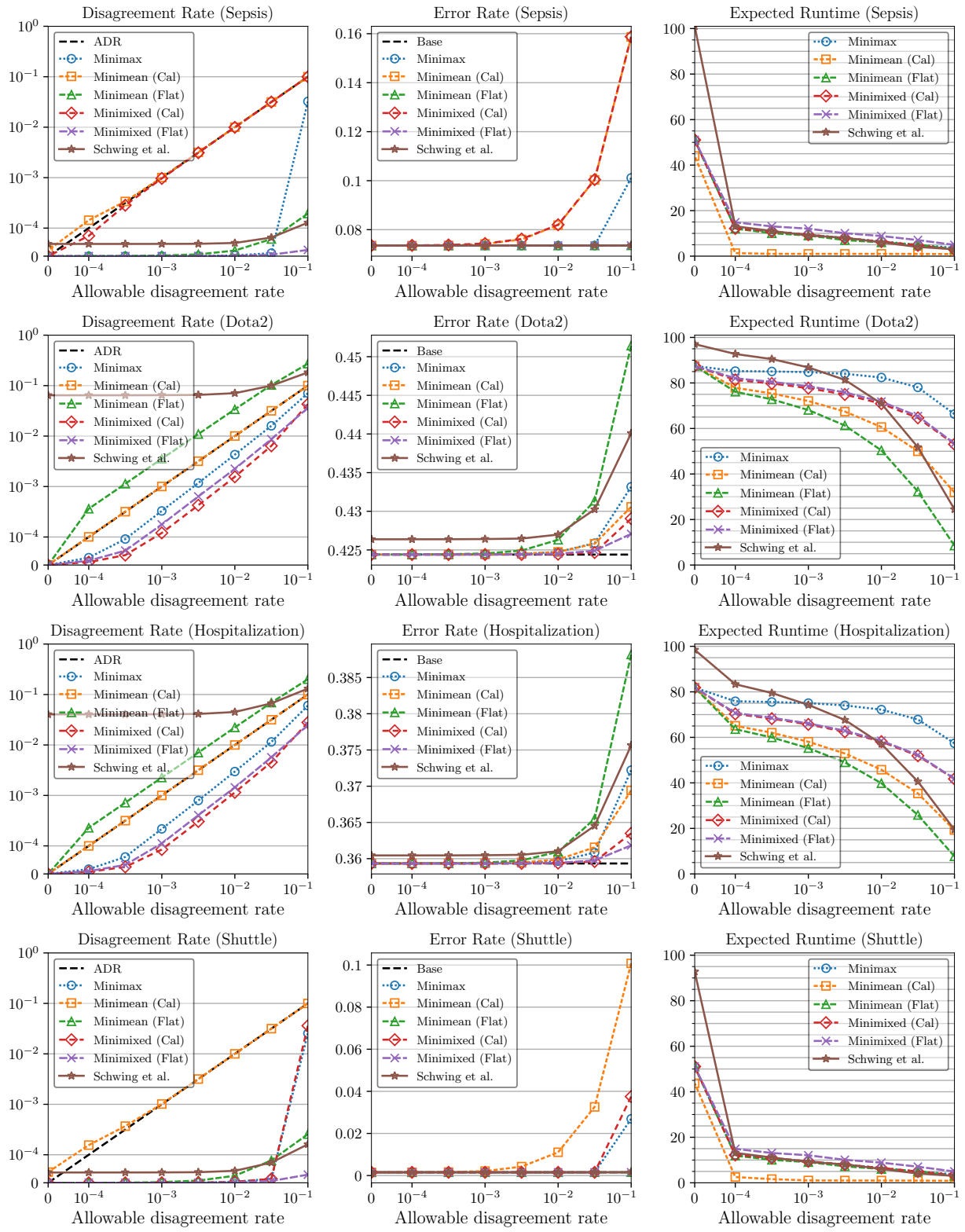

Figure 3: Performance of six early-stopping approaches for random forests of 101 trees on real-world datasets. Each row is a dataset, and each column is a metric. "ADR" is allowable disagreement rate. "Expected runtime" means expected number of trees executed before stopping. The approaches are listed in Table 2. Page 2 of 2.

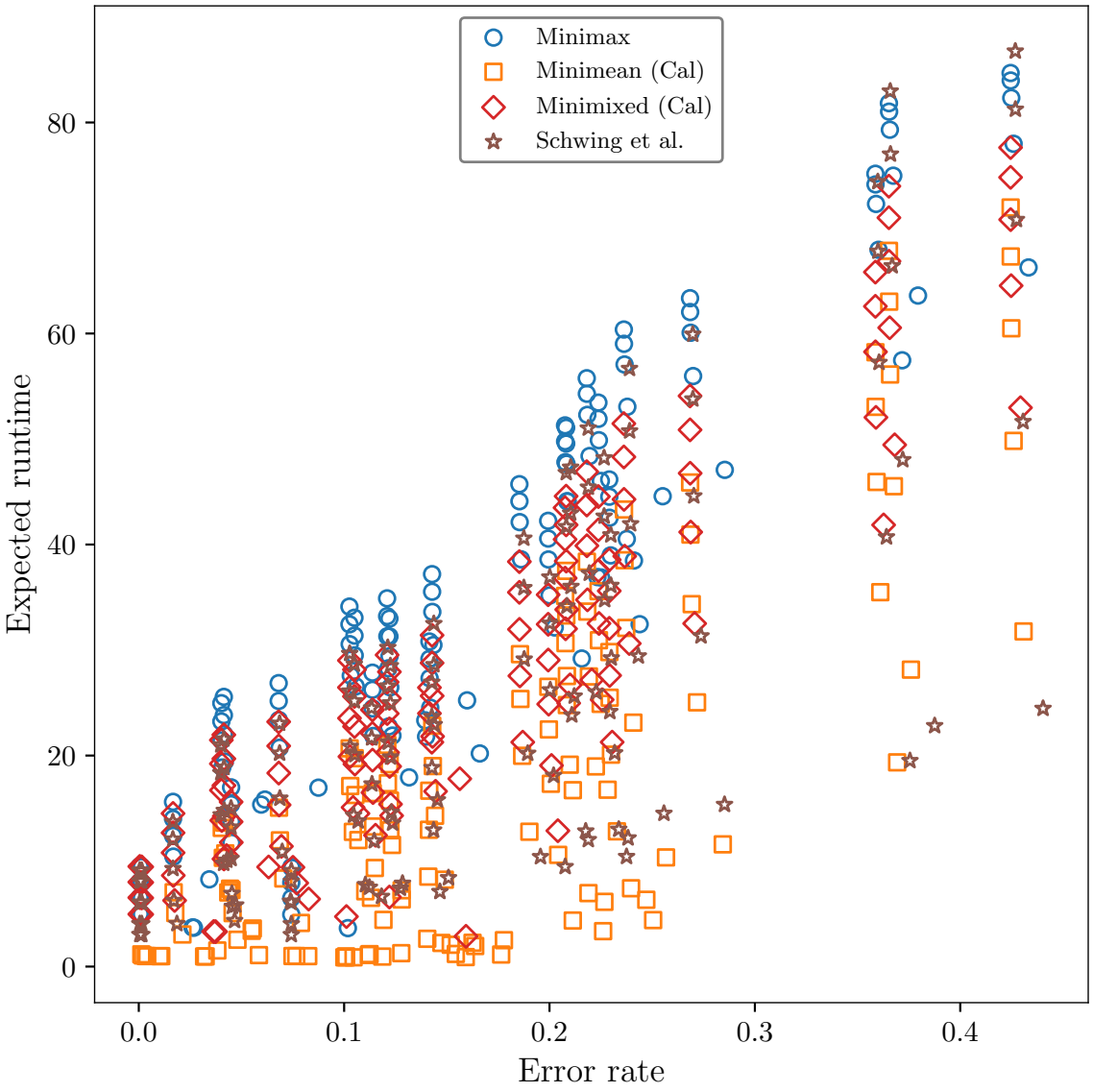

Figure 4: Time-accuracy plot comparing four early-stopping methods applied to random forests of 101 trees on 27 real-world datasets (lower is better). The methods are listed in Table 2. Each point corresponds to a single method on a single dataset with a particular choice of ADR in the set $\{10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}\}$. The X axis is the error rate $\in [0, 1]$ of the resulting model and the Y axis is the expected number of trees executed before stopping ($\in [0, 101]$).

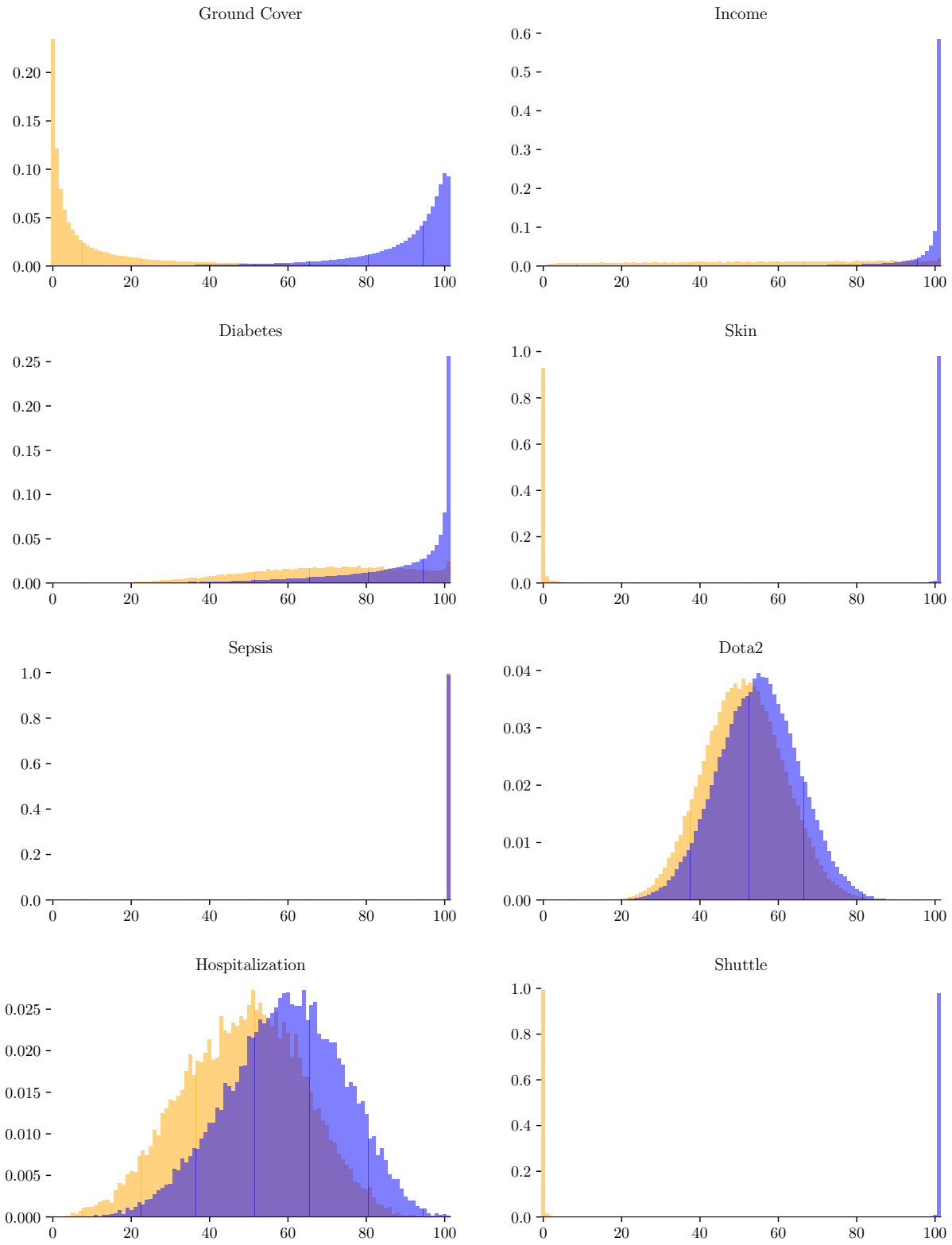

Figure 5: Empirical distributions of number of trees returning positive for a given observation in eight example datasets, out of a forest of 101 trees, averaged across 30 train-test splits and hundreds to thousands of observations. Orange bars represent observations whose true class is negative, and blue bars those whose true class is positive; the purple areas are overlapping bars.

Table 4: Summary of performance metrics for a minimean strategy using a calibration set. The allowable disagreement rate was set to $\alpha = 10^{-3}$.

	Ground Cover	Income	Diabetes	Skin	Sepsis	Dota2	Hospitalization	Shuttle
Disagreement Rate	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Expected Runtime	13.02%	7.17%	16.50%	1.15%	0.99%	71.32%	57.58%	1.03%
Base Error Rate	4.05%	4.52%	14.17%	0.04%	7.39%	42.43%	35.90%	0.15%
Error Rate	4.07%	4.54%	14.16%	0.12%	7.46%	42.43%	35.90%	0.21%

5 Discussion

In this paper, after formally defining the notion of a "stopping strategy" for binary classification ensembles, we proposed several methods to compute stopping strategies that minimize the number of base models executed while controlling the rate of disagreement with the complete ensemble. In experiments on random forests, we saw that for most datasets these methods can cut prediction runtime by 80% or more with minimal loss of accuracy. The methods in this paper are equally applicable to ensembles of other base models; for example, ensemble methods are currently gaining popularity for test-time scaling of large language models (Wang et al., 2023), including majority-voting ensembles (Aggarwal et al., 2023; Trad & Chehab, 2025). Beyond the obvious applications for saving time and energy, this also allows the user to increase accuracy by training a larger ensemble to begin with and trusting the early-stopping strategy to only use as much of it as is necessary for each test. However, our method is limited in its applicability: it cannot be directly applied to multi-class classification problems, let alone regression problems, nor to ensembles in which the base models have a meaningful ordering or different weights.

Strictly speaking, the work in this paper is only relevant to ensembles that aggregate their base models by majority vote. The random forest as originally defined by Breiman is such an ensemble (Breiman, 2001). While many software packages follow his example, such as Apache Spark and the R packages `randomForest` and `ranger` (Liaw & Wiener, 2002; Wright & Ziegler, 2017; Zaharia et al., 2016), other implementations aggregate their results by averaging the probability estimates from each tree, so our methods cannot be directly applied to them. In particular, the Python packages scikit-learn and CuML both use this probability-averaging[1], as does the cross-platform library WEKA (Pedregosa et al., 2011; Hall et al., 2009). However, the performance of the two approaches is almost indistinguishable on all the datasets we have tested. See Section 4 of the supplementary material.

As we saw in Section 1.2, other techniques have already been developed for speeding up ensemble classifiers, and particularly for accelerating random forests. Our method is mutually exclusive with those that involve deliberate reordering of the base models, but not with other common techniques such as pruning. However, we do not expect the gains from sequential testing to be as dramatic for a pruned ensemble, since—as discussed in Section 4—the current results are likely due to the fact that random forests include many trees that tend to vote the same way, a quality which pruning specifically aims to reduce. On the other hand, as discussed in Section 5 of the supplementary material, most similar approaches are theoretically incompatible with pruning, whereas ours is not.

Each of the limitations mentioned presents a potential avenue for future research. Some of the most promising extensions are to multi-class classification and weighted ensembles, as the analysis in Section 2 seems like it could be extended to either one. Another avenue would be to improve the solver to efficiently find stopping strategies for very large ensembles, on the order of $N = 1000$ or more.

[1]For this reason, when we used scikit-learn forests for our experiments in Section 4, we computed the overall forest result using our own code rather than the method provided by scikit-learn itself.

6 Reproducibility and Code Availability

All code used in this paper may be found at our GitHub repository, omitted here for anonymization. In particular, the repository contains a script which reproduces all figures and tables in the paper, except Table 1 (the table of symbols).

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's Sample Step by Step: Adaptive-Consistency for Efficient Reasoning and Coding with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, Singapore, December 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.emnlp-main.761.

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. Enabling Research through the SCIP Optimization Suite 8.0. *ACM Transactions on Mathematical Software*, 49(2): 22:1–22:21, June 2023. doi:10.1145/3585516.

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

Francesco Daghero, Alessio Burrello, Enrico Macii, Paolo Montuschi, Massimo Poncino, and Daniele Jahier Pagliari. Dynamic Decision Tree Ensembles for Energy-Efficient Inference on IoT Edge Nodes. *IEEE Internet of Things Journal*, pp. 1–1, 2023. doi:10.1109/JIOT.2023.3286276.

Leon Eifler, Jules Nicolas-Thouvenin, and Ambros Gleixner. Combining Precision Boosting with LP Iterative Refinement for Exact Linear Optimization. Technical Report 23-26, ZIB, 2023. URL https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/9257.

Wei Fan, Fang Chu, Haixun Wang, and Philip S. Yu. Pruning and dynamic scheduling of cost-sensitive ensembles. In *Eighteenth national conference on Artificial intelligence*, pp. 146–151, USA, July 2002. American Association for Artificial Intelligence. ISBN 978-0-262-51129-2. doi:10.5555/777092.777118.

Ambros Gleixner and Daniel E. Steffy. Linear programming using limited-precision oracles. *Mathematical Programming*, 183(1):525–554, September 2020. doi:10.1007/s10107-019-01444-6.

Ambros M. Gleixner, Daniel E. Steffy, and Kati Wolter. Iterative Refinement for Linear Programming. *INFORMS Journal on Computing*, 28(3):449–464, July 2016. doi:10.1287/ijoc.2016.0692.

L{\'e}o Grinsztajn, Edouard Oyallon, and Ga{\"e}l Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks.*, 2022.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, November 2009. doi:10.1145/1656274.1656278.

Frederick S. Hillier and Gerald J. Lieberman. *Introduction to operations research*. McGraw Hill LLC, New York, NY, 2024. ISBN 978-1-266-93322-6.

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

Tze Leung Lai. Sequential Tests for Hypergeometric Distributions and Finite Populations. *The Annals of Statistics*, 7(1):46–59, January 1979. doi:10.1214/aos/1176344554.

Andy Liaw and Matthew Wiener. Classification and Regression by randomForest. *R News*, 2(3):18–22, 2002. doi:10.32614/CRAN.package.randomForest.

Youness Manzali and Mohamed Elfar. Random Forest Pruning Techniques: A Recent Review. *Operations Research Forum*, 4(2):43, May 2023. doi:10.1007/s43069-023-00223-6.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, November 2011.

Alexander G. Schwing, Christopher Zach, Yefeng Zheng, and Marc Pollefeys. Adaptive random forest - How many "experts" to ask before making a decision? In *CVPR 2011*, pp. 1377–1384. IEEE, 2011. doi:10.1109/CVPR.2011.5995684.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, May 2022. doi:10.1016/j.inffus.2021.11.011.

David Siegmund. *Sequential Analysis*. Springer Series in Statistics. Springer New York, NY, 1985. ISBN 978-1-4419-3075-0 978-1-4757-1862-1. doi:10.1007/978-1-4757-1862-1.

Ivair R. Silva, Martin Kulldorff, and W. Katherine Yih. Optimal Alpha Spending for Sequential Analysis with Binomial Data. *Journal of the Royal Statistical Society Series B*, 82(4):1141–1164, 2020. doi:10.1111/rssb.12379.

Nikola Surjanovic, Andrew Henrey, and Thomas M. Loughin. Alpha-Trimming: Locally Adaptive Tree Pruning for Random Forests. *arXiv*, 2024. URL https://arxiv.org/abs/2408.07151.

Fouad Trad and Ali Chehab. To Ensemble or Not: Assessing Majority Voting Strategies for Phishing Detection with Large Language Models. In Akram Bennour, Ahmed Bouridane, Somaya Almaadeed, Bassem Bouaziz, and Eran Edirisinghe (eds.), *Intelligent Systems and Pattern Recognition*, pp. 158–173, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-82150-9. doi:10.1007/978-3-031-82150-9_13.

Jan van den Brand. A Deterministic Linear Program Solver in Current Matrix Multiplication Time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 259–278, 2020. doi:10.1137/1.9781611975994.16.

A. Wald. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, June 1945. doi:10.1214/aoms/1177731118.

Serena Wang, Maya Gupta, and Seungil You. Quit When You Can: Efficient Evaluation of Ensembles by Optimized Ordering. *ACM Journal on Emerging Technologies in Computing Systems*, 17(4):1–20, 2021. doi:10.1145/3451209.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Marvin N. Wright and Andreas Ziegler. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77:1–17, March 2017. doi:10.18637/jss.v077.i01.

Zhixiang Xu, Matt Kusner, Kilian Weinberger, and Minmin Chen. Cost-Sensitive Tree of Classifiers. *30th International Conference on Machine Learning (ICML)*, 2013.

Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, October 2016. doi:10.1145/2934664.