VARIANCE-SHRUNK RESIDUALS FOR ROBUST GRADIENT BOOSTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient boosting has established itself as a dominant paradigm for structured data. However, its indiscriminate trust in empirical residuals renders it susceptible to noise and outliers. To this end, we propose Uncertainty-Regularized Boosting (URBoost), a framework that leverages the ensemble's internal prediction variance as a proxy for optimization instability. URBoost shrinks pseudo-residual targets according to trajectory variance and solves the resulting variance-regularized tree update through an equivalent weighted least-squares formulation. Theoretically, we formulate URBoost as a variance-regularized functional optimization problem and establish a descent guarantee under smoothness and a sufficiently small learning rate. Statistically, we prove that under a local perturbation model, the trajectory-variance shrinkage coefficient coincides with the optimal Minimum Mean Squared Error linear shrinkage coefficient. Extensive experiments across ten benchmark datasets demonstrate that URBoost achieves strong out-of-the-box performance and robustness against severe label and compound noise, highlighting that an ensemble's internal disagreement provides a useful meta-signal for robust learning.

1 INTRODUCTION

The success of gradient boosting is built on a simple, yet powerful, democratic premise: every training sample contributes to the collective model by voicing its residual error Schapire (1999); Hastie et al. (2005). Algorithms from the foundational Gradient Boosting Decision Trees (GBDT) to their highly optimized successors like XGBoost (XGB) and LightGBM (LGBM), all adhere to this principle Friedman (2001); Chen & Guestrin (2016); Ke et al. (2017). They sequentially add weak learners, each tasked with correcting the mistakes of the current ensemble, treating the error signal from every sample as equally valid Zhou (2025). This democratic approach has proven remarkably effective, establishing them as the state-of-the-art for modeling on structured data Schapire (2003); Ferreira & Figueiredo (2012); Shwartz-Ziv & Armon (2022)

Yet, this core democratic principle is also its limitation. In real-world scenarios, data is rarely pristine. Noisy labels, outliers, or inherently ambiguous samples can generate disproportionately large and misleading gradients Friedman et al. (2000). Standard boosting algorithms, in their uniform treatment, listen to these misleading error signals with the same credulity as they do to genuine error signals. This forces the model to squander substantial capacity fitting "bad information," leading to brittle, poorly generalized models that overfit to noise. This process can be likened to a committee of experts where each new member addresses past errors indiscriminately. As illustrated in Figure 1, this democratic approach fails when some errors stem from unreliable sources.

To realize the intelligent and adaptive committee shown in Figure 1 (b), we argue for a paradigm shift: from indiscriminate error correction to adaptive, uncertainty-guided learning. We propose that the boosting ensemble itself holds the key to discerning trustworthy gradients from misleading ones. The internal conflict, or disagreement, among the ensemble's weak learners about a specific sample can serve as a powerful meta-signal for the reliability of that sample's gradient. A high level of disagreement strongly suggests the sample is noisy, complex, or otherwise unreliable.

We introduce Uncertainty-Regularized Boosting (URBoost), a framework that uses trajectory variance to regularize boosting updates. Instead of changing the global loss, URBoost penalizes large updates on samples with unstable prediction histories and fits variance-shrunken pseudo-residual targets through an equivalent weighted least-squares tree update. The main contributions are as follows:

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

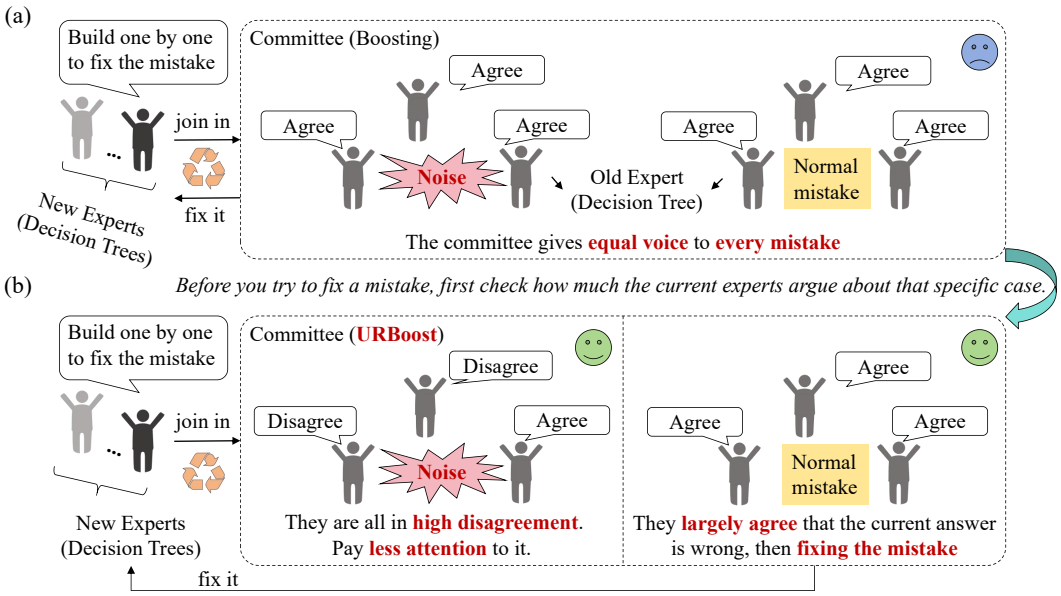


Figure 1: Conceptual illustration of the URBoost mechanism compared to traditional boosting. (a) Traditional boosting methods form a committee where new experts (decision trees) are forced to fix all mistakes equally. (b) URBoost introduces an intelligent feedback loop, assessing internal disagreement to down-weight unreliable samples.

- **Variance-Regularized Optimization.** We formulate URBoost as a variance-regularized boosting method and derive its equivalent weighted least-squares tree update. For a fixed tree partition, Lemma 2.1 shows that the fitted tree satisfies the corresponding optimality condition and is positively aligned with the empirical pseudo-residual. Based on this property, Theorem 2.2 establishes a descent guarantee under smoothness and a sufficiently small learning rate.
- **Optimal Linear Shrinkage.** We provide a statistical interpretation of the trajectory-variance shrinkage coefficient. Theorem 2.3 shows that, under a local additive perturbation model, this coefficient coincides with the MMSE linear shrinkage coefficient for estimating the structural gradient component.
- **Robustness and Interpretability.** Experiments on benchmark datasets show that URBoost achieves strong out-of-the-box performance and robustness under label and compound noise. Shrinkage-trajectory visualizations further illustrate how the method reduces the effect of corrupted samples during training without relying on an external noise detector.

2 Method

In this section, we formally present Uncertainty-Regularized Boosting (URBoost). We begin by revisiting the standard gradient boosting framework to identify its limitation regarding uniform residual treatment. We then derive URBoost as a solution to a *variance-regularized* functional optimization problem, demonstrating that the proposed weighting mechanism is a principled optimization strategy rather than a heuristic design.

2.1 Preliminaries: Gradient Boosting as Functional Gradient Descent

Gradient Boosting Machines (GBM) address the problem of finding an optimal function $F^*(x)$ that minimizes the expected value of a loss function $L(y, F(x))$:

$$F^* = \underset{F}{\operatorname{argmin}}\, \mathbb{E}_{x,y}[L(y, F(x))].$$

Since finding the optimal F^* in an unconstrained function space is ill-posed, the solution is approximated as an additive expansion of weak learners:

$$F(x) = \sum_{m=1}^{M} h_m(x),$$

where $h_m(x)$ are typically decision trees. The model is built sequentially. Given the ensemble $F_{m-1}(x)$ after $m-1$ iterations, the goal is to find a new weak learner $h_m(x)$ such that $F_m(x) = F_{m-1}(x) + h_m(x)$ further minimizes the empirical loss. Viewed as functional gradient descent, the "pseudo-residual" r_{im} for a given sample (x_i, y_i) is calculated as the negative gradient of the loss function with respect to the current prediction $F_{m-1}(x_i)$:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}. \tag{1}$$

The new weak learner $h_m(x)$ is then trained to fit these pseudo-residuals, directing the model update along the steepest descent direction in the function space.

2.2 The Limitation: Uniform Trust in Empirical Residuals

The fundamental vulnerability of the standard approach lies in its uniform treatment of empirical residuals. Standard GBDT trains the weak learner h_m by minimizing the unweighted Mean Squared Error (MSE): $\sum_{i=1}^{N}(r_{im} - h_m(x_i))^2$. This objective implicitly assumes that the residual r_{im} provides an equally reliable structural signal for every sample i. In statistical terms, it treats the observed residual as the true gradient signal $g^*(x_i)$ plus an identically distributed noise component: $r_{im} = g^*(x_i) + \epsilon_i$.

However, real-world data is inherently heteroscedastic. Noisy labels, outliers, and ambiguous samples introduce varying levels of stochastic noise ϵ_i. For a highly noisy sample, the residual r_{im} is dominated by ϵ_i. Standard boosting indiscriminately forces the model to expend capacity fitting this noise, leading to optimization instability and degraded generalization, as the steepest descent direction becomes corrupted by high-variance components.

2.3 URBoost: Variance-Regularized Optimization

To address this limitation, URBoost explicitly reformulates the update step as a *variance-regularized* optimization problem. Instead of uniformly fitting all residuals, we use the ensemble's internal prediction variance as an empirical proxy for local trajectory instability.

We seek an update h_m that minimizes the fitting error while being penalized for large prediction magnitudes on unstable samples. We introduce a data-dependent regularization term to the objective:

$$J(h_m) = \sum_{i=1}^{N} \underbrace{(r_{im} - h_m(x_i))^2}_{\text{Fidelity Term}} + \lambda \sum_{i=1}^{N} \underbrace{\sigma_{im}^2 h_m(x_i)^2}_{\text{Uncertainty Penalty}}, \tag{2}$$

where σ_{im}^2 is the variance of the ensemble's historical predictions for sample x_i, and $\lambda > 0$ is a regularization hyperparameter. The second term enforces a *Soft Trust Region*, implicitly regularizing the update step size $h_m(x_i)$ for samples where the ensemble exhibits high historical disagreement.

2.3.1 Algorithm Description

The theoretical formulation of URBoost is operationalized through a modified gradient boosting loop, which solves the regularized objective in Eq. (2) at each step.

1) Gradient Computation. Consistent with standard boosting, each iteration begins by computing the pseudo-residuals r_{im} based on Eq. (1).

2) Instability Estimation via Trajectory Variance. In a sequential framework, why does prediction variance serve as a reliability proxy? We ground this in the dynamic behavior of the optimization trajectory:

Algorithm 1 URBoost Algorithm

Input: Training data $D = (x_i, y_i)_{i=1}^N$, Iterations M, learning rate v, regularization λ.
Output: The final ensemble model $F_M(x)$.
Initialize $F_0(x) \leftarrow \arg\min * \gamma \sum_{i=1}^N L(y_i, \gamma)$.
Initialize history of predictions $\mathcal{H}_{pred} \leftarrow \emptyset$.
for $m = 1$ **to** M **do**
  Part 1: Gradient Computation
  Compute pseudo-residuals r_{im} for each sample i.
  Part 2: Uncertainty Estimation
  if $m = 1$ **then**
    $\sigma_{im}^2 \leftarrow 0$ for all i.
  else
    Compute variance σ_{im}^2 from history $\mathcal{H}_{pred}$.
  end if
  Part 3: Variance-Regularized Update
  Calculate shrinkage coefficient: $W_{im} \leftarrow \frac{1}{1+\lambda\sigma_{im}^2}$.
  Calculate equivalent WLS weight: $a_{im} \leftarrow 1 + \lambda\sigma_{im}^2$.
  Construct variance-shrunken residual target: $\tilde{r}_{im} \leftarrow W_{im} r_{im}$.
  Fit weak learner h_m by minimizing the equivalent weighted MSE: Eq (4)
  Update model: $F_m(x) \leftarrow F_{m-1}(x) + v h_m(x)$.
  Update $\mathcal{H}_{pred}$ with predictions from h_m.
end for
return $F_M(x)$.

- **Signal Regime:** For clean samples, the sequence of weak learners coherently fits structural patterns, resulting in a stable, monotonic reduction of error and consequently **low variance** among updates.
- **Noise Regime:** For noisy or mislabeled samples, the weak learners cannot find a stable mapping. They oscillate around the irreducible zero-mean noise, fitting spurious correlations. This stochastic "thrashing" manifests as **high variance** in the trajectory.

Thus, we quantify $\sigma_{im}^2 = \mathrm{Var}(h_{1\ldots m-1}(x_i))$ as a measure of **Gradient Oscillation**, effectively separating stable descent (signal) from stochastic oscillation (noise).

3) Closed-Form Shrinkage and Equivalent Weighted Least Squares. To solve the regularized objective $J(h_m)$ in Eq. (2), we first consider the unconstrained pointwise optimal value $h_m^*(x_i)$. Setting the derivative of $J(h_m)$ with respect to $h_m(x_i)$ to zero yields $-2(r_{im} - h_m(x_i)) + 2\lambda\sigma_{im}^2 h_m(x_i) = 0$. Solving for $h_m(x_i)$ gives the pointwise variance-shrunken target

$$h_m^*(x_i) = \frac{1}{1 + \lambda\sigma_{im}^2} r_{im} = W_{im} r_{im}, \tag{3}$$

where $W_{im} = (1 + \lambda\sigma_{im}^2)^{-1}$. Here, W_{im} is a shrinkage coefficient for the pseudo-residual target, rather than the sample weight used to fit the original residuals. λ is regularization parameter.

Since h_m is practically constrained to a hypothesis class, such as regression trees, URBoost minimizes Eq. (2) over $\mathcal{H}$ by using its exactly equivalent weighted least-squares form. Define $a_{im} = 1 + \lambda\sigma_{im}^2$ as the variance-dependent sample weight in the equivalent WLS problem, $W_{im} = a_{im}^{-1}$ as the corresponding shrinkage coefficient, and $\tilde{r}_{im} = W_{im} r_{im} = \frac{r_{im}}{1+\lambda\sigma_{im}^2}$ as the variance-shrunken pseudo-residual target. Also define $C_{im} = r_{im}^2 - \frac{r_{im}^2}{1+\lambda\sigma_{im}^2}$ as a term independent of h. Then, for each sample i, $(r_{im} - h(x_i))^2 + \lambda\sigma_{im}^2 h(x_i)^2 = a_{im}\big(h(x_i) - \tilde{r}_{im}\big)^2 + C_{im}$. Therefore, minimizing the variance-regularized objective is equivalent to solving

$$h_m \in \arg\min_{h \in \mathcal{H}} \sum_{i=1}^N a_{im} (\tilde{r}_{im} - h(x_i))^2. \tag{4}$$

Thus, URBoost does not fit the original residuals r_{im} with inverse-variance weights. Instead, it fits the shrunken targets $\tilde{r}_{im} = W_{im} r_{im}$ using weighted least squares with sample weights $a_{im} = 1 + \lambda\sigma_{im}^2$. For a fixed tree partition $\mathcal{P} = \{R_1, \ldots, R_J\}$, where J is the number of leaves, R_j denotes the j-th leaf region, and $i \in R_j$ means that sample

x_i falls into this leaf, the corresponding optimal leaf value is

$$c_j = \frac{\sum_{i \in R_j} a_{im} \tilde{r}_{im}}{\sum_{i \in R_j} a_{im}} = \frac{\sum_{i \in R_j} r_{im}}{\sum_{i \in R_j} (1 + \lambda\sigma_{im}^2)}. \tag{5}$$

This shows that high trajectory variance shrinks the fitted update by increasing the denominator of the leaf estimate, thereby imposing a soft trust region on samples whose historical optimization trajectory is unstable. The complete procedure is summarized in Algorithm 1.

2.3.2 Theoretical Analysis

This section analyzes the URBoost update rule. Lemma 2.1 gives the fixed-partition optimality condition of the weighted least-squares tree update. Theorem 2.2 gives a descent result under smoothness and a sufficiently small learning rate. Theorem 2.3 interprets $W_{im} = (1 + \lambda\sigma_{im}^2)^{-1}$ as an optimal linear shrinkage coefficient under a local additive perturbation model.

Lemma 2.1 (Fixed-Partition Optimality of Weighted Regression Trees. The Detailed Proof in Appendix A)**.** *Fix the tree partition $\mathcal{P} = \{R_1, \ldots, R_J\}$ returned at iteration m, and assume that each leaf contains at least one training sample. If the leaf values are chosen by weighted least squares according to Eq. (4), then for every leaf R_j the optimal value is c_j in Eq (5). Moreover, the fitted vector $\mathbf{h}_m = (h_m(x_1), \ldots, h_m(x_N))^\top$ satisfies the orthogonality condition $\langle \tilde{\mathbf{r}}_m - \mathbf{h}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} = 0$, which is equivalently expressed as*

$$\langle \tilde{\mathbf{r}}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} = \|\mathbf{h}_m\|_{\mathbf{A}_m}^2,$$

where $\tilde{\mathbf{r}}_m = (\tilde{r}_{1m}, \ldots, \tilde{r}_{Nm})^\top$, $\mathbf{A}_m = \mathrm{diag}(a_{1m}, \ldots, a_{Nm})$. The weighted inner product and its induced norm are defined as $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbf{A}_m} = \sum_{i=1}^N a_{im} a_i b_i$ and $\|\mathbf{a}\|_{\mathbf{A}_m}^2 = \langle \mathbf{a}, \mathbf{a} \rangle_{\mathbf{A}_m}$, respectively.

This lemma establishes the geometric interpretation of the URBoost update step. It demonstrates that fitting the weak learner via Eq. (4) corresponds to an orthogonal projection of the shrunken pseudo-residual targets onto the tree's piecewise-constant subspace under the $\mathbf{A}_m$-induced metric. Since $\mathbf{A}_m \tilde{\mathbf{r}}_m = \mathbf{r}_m$, the identity in Lemma 2.1 also gives $\langle \mathbf{r}_m, \mathbf{h}_m \rangle = \|\mathbf{h}_m\|_{\mathbf{A}_m}^2 \geq 0$. Thus, whenever $\mathbf{h}_m \neq \mathbf{0}$, the fitted tree is positively aligned with the original empirical pseudo-residual vector. This connects the point-wise shrinkage derived in Eq. (3) with the region-wise tree approximation without treating W_{im} as a sample weight on the original residuals.

Theorem 2.2 (Conditional Descent of URBoost. The Detailed Proof in Appendix B)**.** *Let $\mathbf{F}_{m-1} = (F_{m-1}(x_1), \ldots, F_{m-1}(x_N))^\top$ and define the empirical objective on the training predictions as $\mathcal{L}_N(\mathbf{F}) = \sum_{i=1}^N \ell(y_i, F_i)$. Assume that $\mathcal{L}_N$ has an L-Lipschitz gradient with respect to $\mathbf{F}$. Consider the URBoost update $\mathbf{F}_m = \mathbf{F}_{m-1} + \nu \mathbf{h}_m$, where $\mathbf{h}_m = (h_m(x_1), \ldots, h_m(x_N))^\top$ is fitted by Eq. (4) and $\mathbf{h}_m \neq \mathbf{0}$. Let $\mathbf{r}_m = -\nabla_{\mathbf{F}} \mathcal{L}_N(\mathbf{F}_{m-1})$. Then any step size $0 < \nu < \frac{2\|\mathbf{h}_m\|_{\mathbf{A}_m}^2}{L\|\mathbf{h}_m\|_2^2}$ gives $\mathcal{L}_N(\mathbf{F}_m) < \mathcal{L}_N(\mathbf{F}_{m-1})$.*

Theorem 2.2 gives a descent guarantee for the outer boosting step under the standard smoothness assumption and a sufficiently small learning rate. Lemma 2.1 shows that the exact variance-regularized tree update satisfies $\langle \mathbf{r}_m, \mathbf{h}_m \rangle = \|\mathbf{h}_m\|_{\mathbf{A}_m}^2$ for the returned partition and its leaf-optimal values. Therefore, the required positive alignment with the original empirical residual follows whenever the fitted update is nonzero. This descent statement is still local and step-size dependent; because the tree partition is obtained by a greedy and non-convex procedure, URBoost should not be claimed to have an unconditional global convergence advantage over standard boosting.

Theorem 2.3 (Optimal Linear Reliability Weight Under a Local Additive Model. The Detailed Proof in Appendix C)**.** *For a fixed sample i and iteration m, suppose the pseudo-residual admits the local decomposition $r_{im} = s_{im} + n_{im}$, where s_{im} is a structural gradient component and n_{im} is a zero-mean perturbation satisfying $\mathbb{E}[s_{im} n_{im}] = 0$, $\mathbb{E}[s_{im}^2] = \tau^2$, and $\mathbb{E}[n_{im}^2] = \eta_{im}^2$. The scalar linear estimator $w r_{im}$ that minimizes $\mathbb{E}[(w r_{im} - s_{im})^2]$ is $w^* = \frac{\tau^2}{\tau^2 + \eta_{im}^2}$. If the trajectory*

statistic is used as a scale proxy through the modeling condition $\eta_{im}^2 = c\sigma_{im}^2$ for some $c > 0$, then $w^* = (1 + \lambda\sigma_{im}^2)^{-1}$ with $\lambda = c/\tau^2$.

Theorem 2.3 establishes a statistical interpretation for the rational shrinkage coefficient W_{im}. Under a local additive perturbation model, if the empirical trajectory variance σ_{im}^2 is proportional to the local perturbation scale, W_{im} mathematically coincides with the optimal MMSE linear shrinkage coefficient. In the proposed algorithm, this coefficient is used to construct the shrunken target $\tilde{r}_{im} = W_{im}r_{im}$, and the resulting variance-regularized objective is implemented through the equivalent WLS form in Eq. (4) with sample weights $a_{im} = 1 + \lambda\sigma_{im}^2$. Its operational role is to shrink the update target and penalize nonzero updates in regions with highly oscillating historical trajectories.

3 EXPERIMENT

Table 1: The out-of-the-box performance of URBoost compared with the leading ensembles and deep tabular methods.

DATASET	METRIC	RF	ADA	GBDT	XGB	LGBM	CAT	TABPFN	URBOOST
OVA UTERUS	ACC (↑)	0.9075	0.9078*	0.9037*	0.8913*	0.9358	0.9117*	0.9157*	**0.9720**
	AUC (↑)	0.9632*	0.9647*	0.9629*	0.9644*	0.9749*	0.9703*	0.9756	**0.9885**
	F1 (↑)	0.9035	0.9054*	0.9039*	0.8918*	0.9349*	0.9110*	0.9138*	**0.9729**
	LOG LOSS (↓)	0.2949**	0.4796**	0.4265**	0.2537**	0.2548**	0.2270**	0.2301**	**0.0594**
MADELON	ACC (↑)	0.7150**	0.6112**	0.7327**	0.8212**	0.8319**	0.8465**	0.9115**	**0.9569**
	AUC (↑)	0.7892**	0.6525**	0.8206**	0.8917**	0.9008**	0.9148**	0.9678	**0.9855**
	F1 (↑)	0.7210**	0.6118**	0.7410**	0.8233**	0.8330**	0.8462**	0.9122**	**0.9570**
	LOG LOSS (↓)	0.6006**	0.6709**	0.5320**	0.4349**	0.4002**	0.3959**	0.2276**	**0.1125**
JASMINE	ACC (↑)	0.8150**	0.7929**	0.8083**	0.8026**	0.8046**	0.8053**	0.8375**	**0.9126**
	AUC (↑)	0.8809**	0.8434**	0.8632**	0.8608**	0.8640**	0.8741**	0.8965**	**0.9791**
	F1 (↑)	0.8363**	0.8115**	0.8274**	0.8179**	0.8207**	0.8248**	0.8531**	**0.9195**
	LOG LOSS (↓)	0.4113**	0.5454**	0.4162**	0.4864**	0.4339**	0.4042**	0.3736**	**0.2408**
BIORESPONSE	ACC (↑)	0.8049**	0.7563**	0.7870**	0.8009**	0.7985**	0.7897**	0.8019**	**0.9147**
	AUC (↑)	0.8772**	0.8240**	0.8543**	0.8720**	0.8735**	0.8633**	0.8830**	**0.9716**
	F1 (↑)	0.8212**	0.7770**	0.8078**	0.8179**	0.8174**	0.8093**	0.8172**	**0.9216**
	LOG LOSS (↓)	0.4539**	0.6166**	0.4752**	0.5254**	0.4543**	0.4611**	0.4345**	**0.2507**
CREDITCARD	ACC (↑)	0.9685*	0.9645*	0.9614**	0.9736*	0.9716*	0.9675*	0.9695*	**0.9878**
	AUC (↑)	0.9923*	0.9874**	0.9911*	0.9933*	0.9916*	0.9934**	0.9922**	**0.9990**
	F1 (↑)	0.9677*	0.9643*	0.9608*	0.9731	0.9711*	0.9667*	0.9690**	**0.9877**
	LOG LOSS (↓)	0.1117	0.4868**	0.1145	0.1043	0.1236*	0.0945	0.0950	**0.0563**
ELEVATORS	ACC (↑)	0.8625**	0.8352**	0.8596**	0.8921**	0.8907**	0.8882**	0.9140	**0.9148**
	AUC (↑)	0.9132**	0.8852**	0.9083**	0.9418	0.9391*	0.9378*	**0.9592**	0.9580
	F1 (↑)	0.9055**	0.8884**	0.9046**	0.9246*	0.9244**	0.9230**	**0.9398**	0.9396
	LOG LOSS (↓)	0.3387	0.5880**	0.3441**	0.2619*	0.2732*	0.2828*	**0.2171****	0.2875
POL	ACC (↑)	0.9847**	0.9388**	0.9687**	0.9888**	0.9896**	0.9876**	0.9925	**0.9977**
	AUC (↑)	0.9990	0.9849**	0.9963**	0.9994	0.9994	0.9994	**0.9998**	0.9995
	F1 (↑)	0.9886**	0.9539**	0.9766**	0.9916**	0.9922**	0.9907**	0.9944*	**0.9983**
	LOG LOSS (↓)	0.0627	0.5646**	0.0924**	0.0282	0.0286	0.0316	0.0187	**0.0168**
LETTER	ACC (↑)	0.9963*	0.9842**	0.9928**	0.9982	0.9983	0.9970	**0.9996**	0.9986
	AUC (↑)	0.9998	0.9912**	0.9985	0.9999	0.9999	0.9997	**1.0000**	0.9988
	F1 (↑)	0.9522*	0.7837**	0.9040**	0.9775	0.9788	0.9614	**0.9957**	0.9829
	LOG LOSS (↓)	0.0139	0.4512**	0.0235	0.0051	0.0044	0.0108	**0.0011****	0.0134
SCENE	ACC (↑)	0.9136**	0.9788**	0.9875	0.9796**	0.9867	0.9871	0.9892	**0.9946**
	AUC (↑)	0.9483**	0.9844	0.9892	0.9876	0.9920	0.9899	0.9926	**0.9930**
	F1 (↑)	0.6801**	0.9383**	0.9640*	0.9387**	0.9612*	0.9625*	0.9687*	**0.9845**
	LOG LOSS (↓)	0.2504**	0.5525*	0.0660*	0.0813*	0.0794*	0.0605	0.0467	**0.0297**
QSAR	ACC (↑)	0.9341**	0.8835**	0.9327**	0.9256**	0.9313**	0.9243**	0.9299**	**0.9693**
	AUC (↑)	0.9807	0.9472**	0.9702**	0.9727**	0.9752**	0.9721**	0.9800**	**0.9918**
	F1 (↑)	0.9340**	0.8856**	0.9336**	0.9253**	0.9314**	0.9255**	0.9299**	**0.9696**
	LOG LOSS (↓)	0.1945**	0.5794**	0.2158**	0.2251**	0.2281**	0.2170**	0.1868**	**0.0910**

Note: All baselines marked with $*$ are significantly different from URBoost according to the Wilcoxon signed-rank test, where * denotes $p < 0.05$ and ** denotes $p < 0.01$.

Datasets. To test the generalizability and robustness of our algorithm, we curated a challenging benchmark suite of ten publicly available datasets from the OpenML repository. This suite was designed to reflect the heterogeneity of real-world data challenges, spanning a wide range of domains and data geometries. The datasets include: OVA_Uterus Stiglic & Kokol (2010) (1,545 samples, 10,936 features), Madelon Guyon et al. (2007) (2,600 samples, 500 features), Jasmine (2,984 samples, 144 features), Bioresponse (3,751 samples, 1,776 features), Creditcard Dal Pozzolo et al. (2015) dataset (284,807 samples, 30 features), QSAR (1,055 samples, 41 features), Scene (2,407 samples, 300 features), Letter (20,000 samples, 42 features), Pol (10,082 samples, 27 features) and the Elevators (16,599 samples, 17 features). We employed random undersampling Hasanin & Khoshgoftaar (2018) to address class imbalance in the QSAR, Creditcard and OVA_Uterus datasets.

Baselines. We evaluated seven baseline algorithms (RF Breiman (2001), ADA, GBDT, XGB, LGBM, CAT, and TabPFN Hollmann et al. (2025)) in two settings. First, to assess practical utility and out-of-the-box performance, all models, including URBoost, used official recommended software parameters with a fixed random seed ("random_state=42"). URBoost's uncertainty hyperparameter λ was also set to its default value of 1.0. This reflects typical real-world use, where extensive tuning is often infeasible. Second, recognizing that the performance of leading methods like XGB and LGBM is highly sensitive to hyperparameter settings, we conducted an additional, more rigorous comparison. To benchmark peak performance, we performed an extensive hyperparameter search for all tree-based baselines. We note that TabPFN is a pretrained probabilistic transformer and does not require task-specific hyperparameter tuning, so it was evaluated in its standard inference mode. Crucially, URBoost also remained at its default settings. This setup tests URBoost's robustness against fully optimized competitors. Search spaces and final configurations are reported in the Appendix D. The environmental setting is provided in the Appendix E.

Evaluation Metrics. Model performance was assessed using four key metrics. Accuracy (ACC), Area Under the ROC Curve (AUC), F1-Score (F1), and Logarithmic Loss (Log Loss). Given URBoost's information-geometric foundation, superior Log Loss performance serves as a key indicator of its theoretical soundness. To ensure the reliability of our evaluation, we reported the averaged results over ten folds cross-validation (standardscaler was fitted exclusively on the training folds). The best results are highlighted in **bold**, the second-best results are underlined.

4 Results and Analysis

The theoretical design and analysis of URBoost have been established, and we now provide an extensive empirical evaluation to validate its performance in practice. In this section, we report the experimental results regarding our proposed URBoost algorithm in modeling real-world binary classification task in various domains. The primary objective was to evaluate its prediction performance (Section 4.1 and Appendix F), robustness under different noisy ratio settings (Section 4.2), and visualization for the adaptive uncertain-driven weighting mechanism of URBoost (Section 4.3). Additionally, the complexity analysis and hyperparameter λ sensitivity analysis refer to Appendix G and H.

4.1 Out-of-the-box Performance

Table 1 presents the out-of-the-box performance of URBoost compared to seven baselines across ten benchmark datasets. The results demonstrated that URBoost yields highly competitive performance, securing the highest or second-highest scores across almost all evaluated datasets and metrics. Statistical analysis via the Wilcoxon signed-rank test confirms that these performance differences are significant against the majority of competitors. Importantly, these results were obtained using the default configurations for all algorithms. This highlights URBoost's strong out-of-the-box capability, indicating that the variance-regularized mechanism provides a robust default optimization state without requiring dataset-specific hyperparameter tuning. To maintain evaluation consistency, all subsequent experiments, including the robustness and weight visualization analyses, retain these default configurations. A comprehensive comparison against fully tuned baselines is provided in Appendix F.

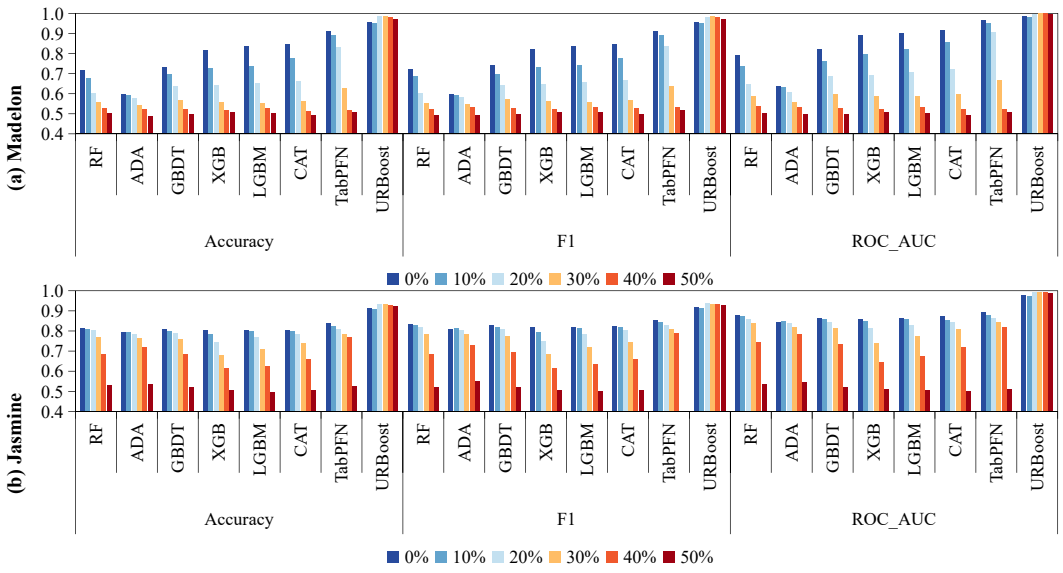


Figure 2: Robustness evaluation for URBoost and baselines against increasing levels (0%-50%) of label noise based on Madelon and Jasmine datasets.

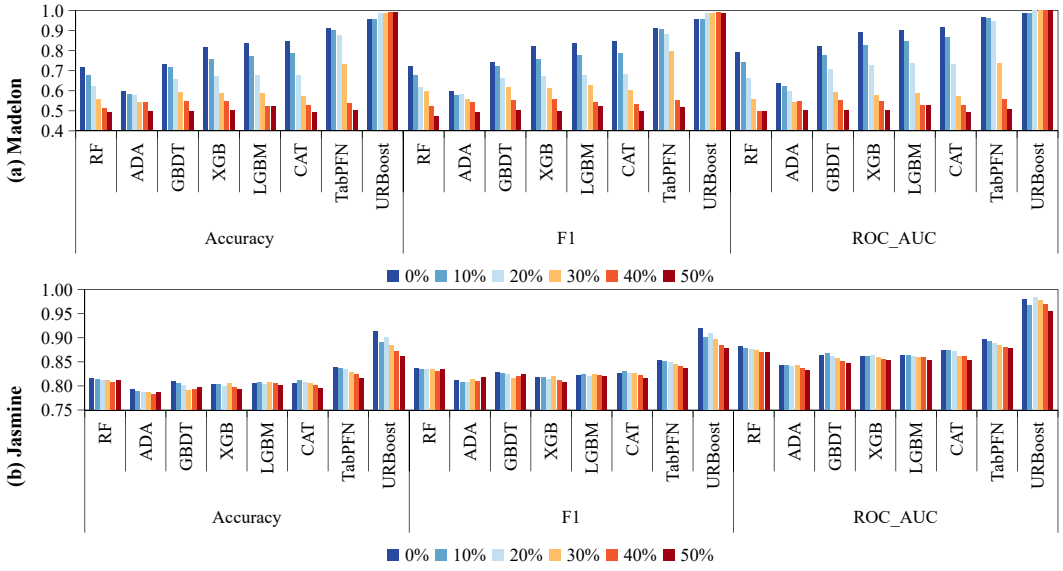


Figure 3: Robustness evaluation for URBoost and baselines against increasing levels (0%-50%) of combined label and feature noise based on Madelon and Jasmine datasets.

4.2 Robustness Analysis

Figure 2 and Figure 3 present the robustness evaluation under progressive label and compound noise (0% to 50%) on the Madelon and Jasmine datasets. While traditional methods like ADA degraded rapidly and recent baselines (XGB, LGBM, CAT, TabPFN) experienced substantial performance drops beyond the 20% noise threshold, URBoost maintained a markedly more stable performance profile. The relative resilience observed at moderate noise levels (10%–20%) empirically aligns with Theorem 2.3, indicating that the variance-based weighting effectively acts as an optimal linear shrinkage mechanism to dampen local stochastic perturbations. Under severe noise environments, the model's consistent stability validates the soft trust region formulated in Eq. (2); the inverse-variance weights explicitly suppress erratic gradients from high-conflict samples, preventing the optimization trajectory from diverging. Furthermore, the mitigation of overfitting in high-noise

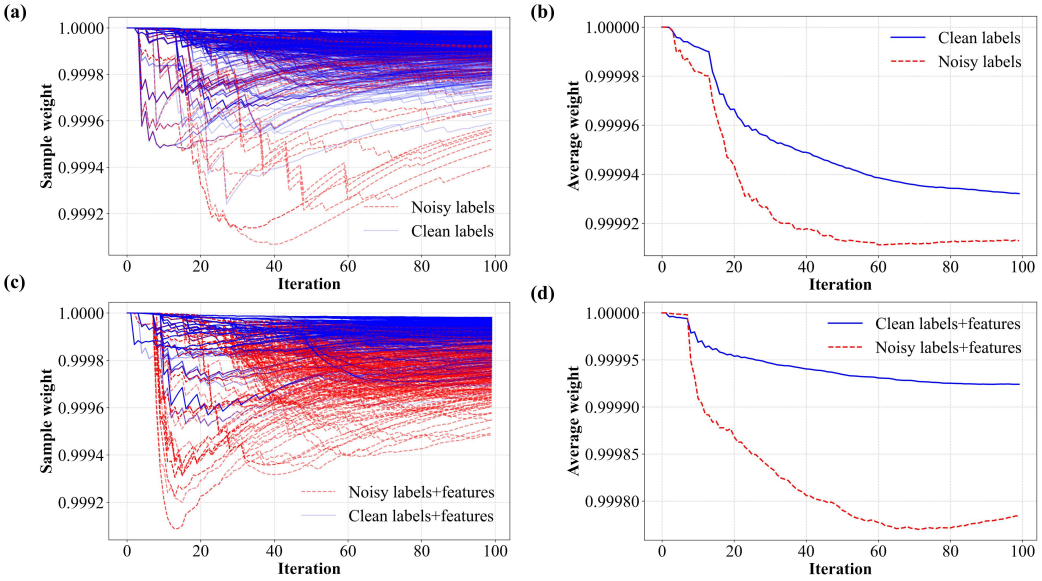


Figure 4: Visualization of URBoost's adaptive weighting. The evolution of sample weights over 100 boosting iterations on Jasmine dataset with 10% injected noise. (a) and (b) response to label noise only. (c) and (d) response to combined label and feature noise. (a) and (c) show individual sample trajectories, while (b) and (d) show the average weight for the clean and noisy cohorts.

regimes demonstrates that penalizing gradient oscillation effectively marginalizes unreliable samples, preventing the tree learners from squandering capacity on spurious noise.

4.3 Visualization Analysis

To empirically examine the internal dynamics of URBoost, we tracked the evolution of the shrinkage weights W_{im} during training. Figure 4 visualizes these trajectories on the Jasmine dataset under a 10% noise setting. As shown in Figures 4(a) and (b), clean samples (blue solid lines) maintained consistently high weights, whereas the weights of noisy samples (red dashed lines) steadily decreased over time. Figures 4(c) and (d) further illustrate that samples with compound corruption experienced a more rapid and uniform weight decline. While the absolute magnitude of this reduction appears modest, its effect is practically significant due to the discrete nature of tree construction: in the greedy argmax split selection, this variance-based penalty is sufficient to alter decisions and steer the tree structure away from fitting localized noise. These empirical observations align directly with the theoretical behavior analyzed in Theorem 2.3, confirming that the variance-regularized objective effectively translates into an adaptive mechanism that systematically discounts samples with high trajectory variance.

5 Conclusion

We introduced URBoost, a variance-regularized boosting method that uses trajectory variance as an internal signal of update instability. By shrinking pseudo-residual targets according to ensemble disagreement and solving the equivalent weighted least-squares tree update, URBoost reduces the influence of unstable optimization trajectories while preserving a principled descent interpretation. Theoretical analysis connects the shrinkage coefficient with optimal linear MMSE estimation under a local perturbation model, and experiments show strong out-of-the-box performance and robustness under noisy settings. A remaining limitation is that hard clean samples and mislabeled samples may both exhibit high trajectory variance. Future work will study finer reliability measures, such as gradient correlation, and memory-efficient variance estimation to reduce the current $O(nM)$ history cost.

AI USE STATEMENT

In this work, we did not use generative AI tools for tasks requiring disclosure such as developing the theoretical framework, formulating mathematical claims, proving theorems, designing experiments, implementing methods, processing data, or interpreting empirical results. We used generative AI tools only for manuscript writing support, including language polishing, improving readability, condensing text, and reorganizing presentation. All AI-assisted text was reviewed and edited by the authors. The authors verified the mathematical statements, proofs, experimental results, and all scientific claims independently, and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work is theoretical and methodological, and does not involve human subjects, private or sensitive data, dataset release, surveillance, security-sensitive applications, or deployment in high-stakes domains. We do not identify specific ethical concerns beyond standard requirements of research integrity, reproducibility, and accurate reporting.

REPRODUCIBILITY STATEMENT

We provide the assumptions, definitions, and complete proofs of the theoretical results in the appendix. The experimental setup, including complexity regimes, hyperparameter grids, environmental setting, and implementation details, is also described in the appendix. The source code used to reproduce the experiments will be released publicly upon acceptance.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

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

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. doi: 10.1145/2939672.2939785.

A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pp. 159–166. IEEE, 2015.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

C. Dubout and F. Fleuret. Adaptive sampling for large scale boosting. *Journal of Machine Learning Research*, 15(1):1431–1453, 2014.

A. J. Ferreira and M. A. Figueiredo. Boosting algorithms: A review of methods, theory, and applications. In *Ensemble Machine Learning*, pp. 35–85. 2012. doi: 10.1007/978-1-4419-9326-7_2.

B. Frénay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2013. doi: 10.1109/TNNLS.2013.2292894.

Y. Freund. A more robust boosting algorithm. arXiv preprint arXiv:0905.2138, 2009.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. doi: 10.1006/jcss.1997.1504.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000. doi: 10.1214/aos/1016218223.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pp. 1189–1232, 2001.

J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4): 367–378, 2002.

I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr. Competitive baseline methods set new standards for the nips 2003 feature selection benchmark. *Pattern Recognition Letters*, 28 (12):1438–1444, 2007.

Taha Hasanin and Taghi Khoshgoftaar. The effects of random undersampling with simulated class imbalance for big data. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 70–79. IEEE, 2018.

T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: Data mining, inference and prediction. *Mathematical Intelligencer*, 27(2):83–85, 2005.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

G. Ke, Q. Meng, T. Finley, T. Wang, and W. Chen. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, 2017.

J. Luo, Y. Quan, and S. Xu. Robust-GBDT: Leveraging robust loss for noisy and imbalanced classification with gbdt. *Knowledge and Information Systems*, pp. 1–21, 2025.

H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: Theory, robustness to outliers, and savageboost. In *Advances in Neural Information Processing Systems 21*, 2008.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, 1999.

Q. Miao, Y. Cao, G. Xia, M. Gong, J. Liu, and J. Song. RBoost: Label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2216–2228, 2015. doi: 10.1109/TNNLS.2015.2475750.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31*, 2018.

R. E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, volume 99, 1999.

R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, pp. 149–171. 2003.

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

G. Stiglic and P. Kokol. Stability of ranked gene lists in large microarray analysis studies. *BioMed Research International*, 2010:616358, 2010.

Ye Su, Longlong Zhao, Diego Garcia-Gil, Jipeng Guo, Gangchun Zhang, Jinxin Chen, and Jinsong Chen. Itboost: Information-theoretic trust for robust boosting. *arXiv preprint arXiv:2605.04671*, 2026.

Z. Wang. Robust boosting with truncated loss functions, 2018.

R. Yan, J. Tesic, and J. R. Smith. Model-shared subspace boosting for multi-label classification. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 834–843, 2007.

Z. H. Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2025.

A Proof of Lemma 2.1

Proof. For the fixed partition $\mathcal{P} = \{R_1, \dots, R_J\}$, every tree output has the form $h(x_i) = c_j$ for $i \in R_j$. By Eq. (4),

$$Q(c_1, \dots, c_J) = \sum_{j=1}^{J} \sum_{i \in R_j} a_{im}(\tilde{r}_{im} - c_j)^2.$$

For each j, the partial derivative is

$$\begin{aligned}
\frac{\partial Q}{\partial c_j} &= \frac{\partial}{\partial c_j} \sum_{i \in R_j} a_{im}(\tilde{r}_{im} - c_j)^2 \\
&= \sum_{i \in R_j} 2a_{im}(\tilde{r}_{im} - c_j)(-1) \\
&= 2 \sum_{i \in R_j} a_{im}(c_j - \tilde{r}_{im}) \\
&= 2c_j \sum_{i \in R_j} a_{im} - 2 \sum_{i \in R_j} a_{im}\tilde{r}_{im}.
\end{aligned}$$

Since $a_{im} = 1 + \lambda\sigma_{im}^2 > 0$ and R_j is nonempty,

$$\frac{\partial^2 Q}{\partial c_j^2} = 2 \sum_{i \in R_j} a_{im} > 0.$$

Thus the unique minimizer satisfies

$$\begin{aligned}
\frac{\partial Q}{\partial c_j} = 0 &\iff 2c_j \sum_{i \in R_j} a_{im} - 2 \sum_{i \in R_j} a_{im}\tilde{r}_{im} = 0 \\
&\iff c_j \sum_{i \in R_j} a_{im} = \sum_{i \in R_j} a_{im}\tilde{r}_{im} \\
&\iff c_j = \frac{\sum_{i \in R_j} a_{im}\tilde{r}_{im}}{\sum_{i \in R_j} a_{im}}.
\end{aligned}$$

Since

$$a_{im} = 1 + \lambda\sigma_{im}^2, \quad \tilde{r}_{im} = \frac{r_{im}}{1 + \lambda\sigma_{im}^2} = \frac{r_{im}}{a_{im}},$$

we have

$$a_{im}\tilde{r}_{im} = r_{im}.$$

Therefore,

$$\begin{aligned}
c_j &= \frac{\sum_{i \in R_j} a_{im}\tilde{r}_{im}}{\sum_{i \in R_j} a_{im}} \\
&= \frac{\sum_{i \in R_j} r_{im}}{\sum_{i \in R_j} a_{im}} \\
&= \frac{\sum_{i \in R_j} r_{im}}{\sum_{i \in R_j} (1 + \lambda\sigma_{im}^2)}.
\end{aligned}$$

For $i \in R_j$, $h_m(x_i) = c_j$. Hence

$$\begin{aligned}
\langle \tilde{\mathbf{r}}_m - \mathbf{h}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} &= \sum_{i=1}^{N} a_{im}(\tilde{r}_{im} - h_m(x_i))h_m(x_i) \\
&= \sum_{j=1}^{J} \sum_{i \in R_j} a_{im}(\tilde{r}_{im} - c_j)c_j \\
&= \sum_{j=1}^{J} c_j \sum_{i \in R_j} a_{im}(\tilde{r}_{im} - c_j) \\
&= \sum_{j=1}^{J} c_j \left(\sum_{i \in R_j} a_{im}\tilde{r}_{im} - c_j \sum_{i \in R_j} a_{im} \right) \\
&= \sum_{j=1}^{J} c_j \left(\sum_{i \in R_j} a_{im}\tilde{r}_{im} - \frac{\sum_{i \in R_j} a_{im}\tilde{r}_{im}}{\sum_{i \in R_j} a_{im}} \sum_{i \in R_j} a_{im} \right) \\
&= \sum_{j=1}^{J} c_j \left(\sum_{i \in R_j} a_{im}\tilde{r}_{im} - \sum_{i \in R_j} a_{im}\tilde{r}_{im} \right) \\
&= 0.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\langle \tilde{\mathbf{r}}_m - \mathbf{h}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} = 0 &\iff \langle \tilde{\mathbf{r}}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} - \langle \mathbf{h}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} = 0 \\
&\iff \langle \tilde{\mathbf{r}}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} = \|\mathbf{h}_m\|^2_{\mathbf{A}_m}.
\end{aligned}$$

Moreover,

$$\begin{aligned}
\langle \tilde{\mathbf{r}}_m, \mathbf{h}_m \rangle_{\mathbf{A}_m} &= \sum_{i=1}^{N} a_{im}\tilde{r}_{im}h_m(x_i) \\
&= \sum_{i=1}^{N} r_{im}h_m(x_i) \\
&= \langle \mathbf{r}_m, \mathbf{h}_m \rangle.
\end{aligned}$$

Hence

$$\langle \mathbf{r}_m, \mathbf{h}_m \rangle = \|\mathbf{h}_m\|^2_{\mathbf{A}_m} \geq 0.$$

This completes the proof. □

B Proof of Theorem 2.2

Proof. By L-smoothness of $\mathcal{L}_N$, for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$,

$$\mathcal{L}_N(\mathbf{v}) \leq \mathcal{L}_N(\mathbf{u}) + \langle \nabla_{\mathbf{F}} \mathcal{L}_N(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle + \frac{L}{2}\|\mathbf{v} - \mathbf{u}\|_2^2.$$

Let

$$\mathbf{u} = \mathbf{F}_{m-1}, \quad \mathbf{v} = \mathbf{F}_m = \mathbf{F}_{m-1} + \nu \mathbf{h}_m.$$

Then

$$\mathbf{v} - \mathbf{u} = \nu \mathbf{h}_m.$$

Substituting into the smoothness inequality gives

$$\begin{aligned}
\mathcal{L}_N(\mathbf{F}_m) &\leq \mathcal{L}_N(\mathbf{F}_{m-1}) + \langle \nabla_{\mathbf{F}} \mathcal{L}_N(\mathbf{F}_{m-1}), \nu \mathbf{h}_m \rangle + \frac{L}{2}\|\nu \mathbf{h}_m\|_2^2 \\
&= \mathcal{L}_N(\mathbf{F}_{m-1}) + \nu \langle \nabla_{\mathbf{F}} \mathcal{L}_N(\mathbf{F}_{m-1}), \mathbf{h}_m \rangle + \frac{L\nu^2}{2}\|\mathbf{h}_m\|_2^2.
\end{aligned}$$

Since

$$\mathbf{r}_m = -\nabla_{\mathbf{F}} \mathcal{L}_N(\mathbf{F}_{m-1}),$$

we obtain

$$\mathcal{L}_N(\mathbf{F}_m) \leq \mathcal{L}_N(\mathbf{F}_{m-1}) - \nu \langle \mathbf{r}_m, \mathbf{h}_m \rangle + \frac{L\nu^2}{2} \|\mathbf{h}_m\|_2^2.$$

By Lemma 2.1,

$$\langle \mathbf{r}_m, \mathbf{h}_m \rangle = \|\mathbf{h}_m\|_{\mathbf{A}_m}^2.$$

Therefore,

$$\mathcal{L}_N(\mathbf{F}_m) \leq \mathcal{L}_N(\mathbf{F}_{m-1}) - \nu \|\mathbf{h}_m\|_{\mathbf{A}_m}^2 + \frac{L\nu^2}{2} \|\mathbf{h}_m\|_2^2.$$

Thus,

$$\begin{aligned} \mathcal{L}_N(\mathbf{F}_m) - \mathcal{L}_N(\mathbf{F}_{m-1}) &\leq -\nu \|\mathbf{h}_m\|_{\mathbf{A}_m}^2 + \frac{L\nu^2}{2} \|\mathbf{h}_m\|_2^2 \\ &= \nu \left(-\|\mathbf{h}_m\|_{\mathbf{A}_m}^2 + \frac{L\nu}{2} \|\mathbf{h}_m\|_2^2 \right). \end{aligned}$$

Let

$$\alpha = \|\mathbf{h}_m\|_{\mathbf{A}_m}^2, \quad \beta = \|\mathbf{h}_m\|_2^2.$$

Since $\mathbf{h}_m \neq \mathbf{0}$ and $a_{im} > 0$ for all i,

$$\alpha = \sum_{i=1}^{N} a_{im} h_m(x_i)^2 > 0, \quad \beta = \sum_{i=1}^{N} h_m(x_i)^2 > 0.$$

If

$$0 < \nu < \frac{2\alpha}{L\beta},$$

then

$$\begin{aligned} \nu < \frac{2\alpha}{L\beta} &\iff L\nu\beta < 2\alpha \\ &\iff \frac{L\nu}{2}\beta < \alpha \\ &\iff -\alpha + \frac{L\nu}{2}\beta < 0 \\ &\iff \nu \left(-\alpha + \frac{L\nu}{2}\beta \right) < 0. \end{aligned}$$

Hence

$$\mathcal{L}_N(\mathbf{F}_m) - \mathcal{L}_N(\mathbf{F}_{m-1}) \leq \nu \left(-\alpha + \frac{L\nu}{2}\beta \right) < 0.$$

Therefore,

$$\mathcal{L}_N(\mathbf{F}_m) < \mathcal{L}_N(\mathbf{F}_{m-1}).$$

This completes the proof. □

C Proof of Theorem 2.3

Proof. For fixed i and m, let

$$r_{im} = s_{im} + n_{im}.$$

Define

$$J(w) = \mathbb{E}[(w r_{im} - s_{im})^2].$$

Substituting $r_{im} = s_{im} + n_{im}$ gives

$$\begin{aligned} J(w) &= \mathbb{E}[(w(s_{im} + n_{im}) - s_{im})^2] \\ &= \mathbb{E}[(ws_{im} + wn_{im} - s_{im})^2] \\ &= \mathbb{E}[((w-1)s_{im} + wn_{im})^2] \\ &= \mathbb{E}\left[(w-1)^2 s_{im}^2 + 2w(w-1)s_{im}n_{im} \right. \\ &\quad \left. + w^2 n_{im}^2\right] \\ &= (w-1)^2\mathbb{E}[s_{im}^2] + 2w(w-1)\mathbb{E}[s_{im}n_{im}] \\ &\quad + w^2\mathbb{E}[n_{im}^2]. \end{aligned}$$

Using $\mathbb{E}[s_{im}n_{im}] = 0$, $\mathbb{E}[s_{im}^2] = \tau^2$, and $\mathbb{E}[n_{im}^2] = \eta_{im}^2$,

$$J(w) = (w-1)^2\tau^2 + w^2\eta_{im}^2.$$

Expanding,

$$\begin{aligned} J(w) &= (w^2 - 2w + 1)\tau^2 + w^2\eta_{im}^2 \\ &= w^2(\tau^2 + \eta_{im}^2) - 2w\tau^2 + \tau^2. \end{aligned}$$

The first derivative is

$$J'(w) = 2w(\tau^2 + \eta_{im}^2) - 2\tau^2.$$

The second derivative is

$$J''(w) = 2(\tau^2 + \eta_{im}^2).$$

Since $\tau^2 > 0$ and $\eta_{im}^2 \geq 0$,

$$J''(w) = 2(\tau^2 + \eta_{im}^2) > 0.$$

Thus $J(w)$ is strictly convex and has a unique global minimizer. Setting $J'(w) = 0$ gives

$$\begin{aligned} 2w(\tau^2 + \eta_{im}^2) - 2\tau^2 &= 0 \\ w(\tau^2 + \eta_{im}^2) - \tau^2 &= 0 \\ w(\tau^2 + \eta_{im}^2) &= \tau^2 \\ w^* &= \frac{\tau^2}{\tau^2 + \eta_{im}^2}. \end{aligned}$$

If $\eta_{im}^2 = c\sigma_{im}^2$ with $c > 0$, then

$$\begin{aligned} w^* &= \frac{\tau^2}{\tau^2 + c\sigma_{im}^2} \\ &= \frac{1}{1 + \frac{c}{\tau^2}\sigma_{im}^2}. \end{aligned}$$

With $\lambda = c/\tau^2$,

$$w^* = \frac{1}{1 + \lambda\sigma_{im}^2}.$$

This completes the proof. □

D Hyperparameter Optimization Details

To ensure a fair and rigorous comparison against the performance ceiling of each baseline model, we conducted an extensive hyperparameter search. This process was automated using the **Optuna** framework Akiba et al. (2019), a state-of-the-art Bayesian optimization library. For each baseline algorithm, the optimization objective was to **maximize the mean ACC score** obtained through a **10-fold stratified cross-validation** procedure on each dataset, with a fixed random seed ('random_state=42') for reproducibility. The optimization for each model on each dataset was performed for **50 trials**, where each trial corresponded to a unique set of hyperparameters selected by Optuna's Tree-structured

Parzen Estimator sampler. The specific hyperparameter search spaces were configured as follows. For **RF**, `n_estimators` was searched in the integer range [50, 200] and `max_depth` in [3, 8]. For **GBDT**, **XGB**, **LGBM**, and **CAT**, we searched `n_estimators` in [50, 200], `max_depth` in [3, 8], and `learning_rate` on a log-uniform scale between 10^{-3} and 1.0. For **ADA**, we searched `n_estimators` in [50, 200] and `learning_rate` on a log-uniform scale between 10^{-3} and 1.0. The set of hyperparameters yielding the highest cross-validation accuracy was then used to report the final tuned performance in the Appendix F. As stated in the main text, URBoost was not subjected to this tuning process and retained its default parameters for all comparisons.

E Environmental Setting

All experiments were conducted locally using Jupyter Notebook in an Anaconda environment on Windows 10 (Version 10.0.19045). The hardware infrastructure utilized an Intel processor (Intel64 Family 6 Model 151 Stepping 2, GenuineIntel) equipped with 20 logical cores. The software stack was built on Python 3.12.4, utilizing the following key libraries: pandas 2.2.2, numpy 1.26.4, scikit-learn 1.7.2, statsmodels 0.14.2, imbalanced-learn 0.14.0, lightgbm 4.3.0, xgboost 2.0.3, catboost 1.2.3, ngboost 0.5.7, and tabpfn 6.0.6.

F Comparison with Tuned leading Baselines

Table 2 presents the performance comparison against leading baselines that have undergone extensive hyperparameter optimization, alongside the pre-trained TabPFN (evaluated in default mode). In this challenging setting, URBoost was intentionally configured with its default parameters to assess its out-of-the-box robustness. The results demonstrate that even without tuning, URBoost consistently outperforms all competitors across every dataset and metric. Notably, on the `Madelon` dataset, which contains high feature noise, URBoost achieved an ACC of 95.69%, establishing a clear margin over the second-best performer, TabPFN (91.15%), and the best-tuned tree ensemble, CAT (86.81%). This advantage is particularly evident in model calibration (Log Loss); on `OVA_Uterus`, URBoost (0.0594) significantly outpaces both TabPFN (0.2312) and tuned XGB (0.2313). These findings strongly suggest that the performance gains of URBoost stem from its variance-regularized architectural design rather than sensitivity to hyperparameter settings. While leading boosting methods require costly and dataset-specific tuning to reach their performance ceiling, URBoost provides a more powerful and reliable solution directly, highlighting the practical strength of its information-geometric learning process.

G Complexity analysis of URBoost

In this section, we analyze the time and space complexity of the proposed URBoost algorithm. We demonstrate that the introduction of the adaptive uncertainty-driven weighting mechanism adds only a modest and computationally efficient overhead compared to the standard gradient boosting framework. Let n denote the number of training samples, d the feature dimensionality, M the number of boosting iterations (i.e., n_estimators), and l be the number of leaves in a single decision tree.

G.1 Time complexity

The total time complexity is dominated by the main boosting loop, which runs for M iterations. We analyze the cost of a single iteration m, where m ranges from 1 to M.

- **Residual Computation.** Calculating the pseudo-residuals for all n samples requires a single pass through the data. This has a time complexity of $O(n)$.
- **Uncertainty Estimation.** This is the additional step introduced by URBoost.
  - (i) For each of the n samples, we need to compute the variance of its predictions from the $m-1$ previously trained trees.
  - (ii) Accessing the historical predictions for one sample takes $O(m)$ time. Computing the variance from these $m-1$ values also takes $O(m)$ time.

  Therefore, the total time for this stage in iteration m is $O(nm)$.

Table 2: Performance of URBoost (default) against tuned leading ensembles and TabPFN (default).

DATASET	METRIC	RF	ADA	GBDT	XGB	LGBM	CAT	TABPFN	URBOOST
OVA_UTERUS	ACC (↑)	0.9318	0.9278	0.9318*	0.9195*	0.9318	0.9358*	0.9157*	**0.9720**
	AUC (↑)	0.9672*	0.9731*	0.9703*	0.9649*	0.9718*	0.9691*	0.9756*	**0.9885**
	F1 (↑)	0.9325	0.9278*	0.9316*	0.9196*	0.9309*	0.9362*	0.9138*	**0.9729**
	LOG LOSS (↓)	0.3287**	0.4765**	0.7112**	0.2313**	0.3546**	0.2433**	0.2301**	**0.0594**
MADELON	ACC (↑)	0.7104**	0.6292**	0.8500**	0.8469**	0.8415**	0.8681**	0.9115**	**0.9569**
	AUC (↑)	0.7855**	0.6850**	0.9163**	0.9108**	0.9088**	0.9315 **	0.9678	**0.9855**
	F1 (↑)	0.7276**	0.6265**	0.8555**	0.8492**	0.8437**	0.8672**	0.9122**	**0.9570**
	LOG LOSS (↓)	0.6097**	0.6585**	0.3846**	0.3786**	0.4219**	0.4218**	0.2276**	**0.1125**
JASMINE	ACC (↑)	0.8093**	0.7999**	0.8140**	0.8170**	0.8150**	0.8194**	0.8375**	**0.9126**
	AUC (↑)	0.8722**	0.8496**	0.8691**	0.8637**	0.8639**	0.8769**	0.8965**	**0.9791**
	F1 (↑)	0.8365**	0.8200**	0.8303**	0.8381**	0.8372**	0.8396**	0.8531**	**0.9195**
	LOG LOSS (↓)	0.4177**	0.5407**	0.5018**	0.4434**	0.4332**	0.3983**	0.3736**	**0.2408**
BIORESPONSE	ACC (↑)	0.7787**	0.7638**	0.8030**	0.8054**	0.8040**	0.8022**	0.8019**	**0.9147**
	AUC (↑)	0.8516**	0.8271**	0.8727**	0.8765**	0.8731**	0.8710**	0.8830**	**0.9716**
	F1 (↑)	0.7972**	0.7838**	0.8210**	0.8230**	0.8213**	0.8215**	0.8172**	**0.9216**
	LOG LOSS (↓)	0.5082**	0.6290**	0.4676**	0.4628**	0.4588**	0.4491**	0.4345**	**0.2507**
CREDITCARD	ACC (↑)	0.9593*	0.9685**	0.9705*	0.9736	0.9787*	0.9726*	0.9695*	**0.9878**
	AUC (↑)	0.9905*	0.9876**	0.9910*	0.9935*	0.9943*	0.9915*	0.9922*	**0.9990**
	F1 (↑)	0.9579*	0.9682**	0.9699*	0.9731*	0.9783*	0.9721*	0.9690*	**0.9877**
	LOG LOSS (↓)	0.1266*	0.5054**	0.1128*	0.1016	0.1725**	0.1082	0.0950*	**0.0563**

Note: All baselines marked with $*$ are significantly different from URBoost according to the Wilcoxon signed-rank test, where * denotes $p < 0.05$ and ** denotes $p < 0.01$.

- **Weak Learner Training.** The cost of training a single decision tree (the weak learner) on n samples and d features is typically dominated by the process of finding the best splits. For a standard CART implementation (like in scikit-learn), this is $O(ndl)$. When sample weights are used, the complexity remains the same; each split evaluation simply incorporates the sample's weight into the impurity calculation.

The total time complexity is the sum of these costs over all M iterations: $\sum_{m=1}^{M}[O(n) + O(nm) + O(ndl)]$. Let's analyze the components of this sum:

(i) $\sum O(n) = M \cdot O(n) = O(Mn)$

(ii) $\sum O(nm) = O(n) \cdot \sum_{m=1}^{M} m = O(n) \cdot (M(M+1)/2) = O(nM^2)$

(iii) $\sum O(ndl) = M \cdot O(ndl) = O(Mndl)$

Combining these components, the overall time complexity of URBoost can be expressed as $O(Mndl + nM^2)$. The time complexity of a standard GBDT is $O(Mndl)$. Therefore, the additional overhead introduced by URBoost is the $O(nM^2)$ term.

Practical Implications. In most typical use cases, the tree construction term $O(Mndl)$ is the dominant factor. The nM^2 term becomes significant only when a very large number of estimators (M) is used on a dataset with relatively few features. Our implementation stores past predictions, making the variance calculation highly efficient. Thus, in practice, URBoost's training time is only modestly higher than that of a standard GBDT.

G.2 SPACE COMPLEXITY

The space complexity is determined by the storage required for the model and any intermediate data structures.

1. **Model Storage.** Storing M decision trees requires $O(Ml)$ space, as each tree stores information about its l leaves (split points, values, etc.).
2. **Prediction History Storage.** This is the additional storage requirement for URBoost. To calculate the variance efficiently, we must store the predictions of each of the M trees on all

n samples. This requires a matrix of size $n \times M$. The space complexity for this history is $O(nM)$.

A standard GBDT implementation that does not need historical predictions only requires $O(Ml)$ space to store the final model. The additional space overhead for URBoost is the $O(nM)$ term.

Practical Implications. The $O(nM)$ term is the most significant difference in space complexity. For very large datasets (n) and a large number of estimators (M), this can lead to a considerable memory footprint. However, for most common dataset sizes, this is manageable. For extremely large-scale applications where memory is a critical constraint, strategies like storing quantized predictions or using a moving window for the variance calculation could be employed to reduce this requirement.

G.3 Efficiency analysis

To empirically validate our theoretical complexity analysis, we test the Wall-clock time and peak memory usage of URBoost against the baseline ensembles. The results, presented in Table 3, align closely with our derivations. URBoost's training time was consistently higher than the highly optimized LGBM and XGB, which is attributable to the additional $O(nM^2)$ overhead from the iterative uncertainty calculation. Similarly, the peak memory usage of URBoost was generally higher than the baselines, reflecting the $O(nM)$ space complexity required to store the historical predictions for variance computation. This demonstrates a clear and expected trade-off; URBoost's superior robustness and performance are achieved at the cost of moderately increased computational and memory resources, a trade-off that is well-justified by the significant gains in model reliability documented in the main paper.

Table 3: Wall-clock time and peak memory usage for URBoost compared with the leading ensembles.

Datasets	Metric	RF	ADA	GBDT	XGB	LGBM	CAT	TabPFN	URBoost
OVA Uterus	Time (s)	0.88	17.34	41.87	2.77	1.57	26.35	26.43	58.34
	Memory (MB)	9.66	9.65	9.57	0.10	4.86	1.09	104.48	9.74
Madelon	Time (s)	2.11	4.01	10.09	0.56	0.31	0.97	8.28	48.98
	Memory (MB)	4.82	4.74	4.77	0.26	8.95	0.06	138.27	6.47
Jasmine	Time (s)	0.87	0.87	0.97	0.09	0.13	0.28	1.64	1.89
	Memory (MB)	1.85	1.77	1.81	0.25	2.99	0.05	48.69	4.41
Bioresponse	Time (s)	1.51	7.18	14.71	0.78	0.65	17.74	16.26	72.71
	Memory (MB)	23.30	23.20	23.28	0.29	2.15	0.19	298.65	25.75
Creditcard	Time (s)	0.58	0.56	0.67	0.05	0.05	0.30	1.45	1.54
	Memory (MB)	0.33	0.31	0.25	0.13	1.74	0.05	4.09	1.58
Elevators	Time (s)	1.74	0.96	1.61	0.10	0.09	0.19	1.43	4.36
	Memory (MB)	2.34	1.89	2.36	0.39	2.19	0.05	31.10	2.72
Pol	Time (s)	1.04	1.02	1.39	0.10	0.08	0.74	1.38	4.22
	Memory (MB)	3.68	3.27	3.82	0.22	5.07	0.05	44.26	2.45
Letter	Time (s)	0.98	0.92	1.47	0.07	0.06	0.18	1.54	3.59
	Memory (MB)	2.64	2.10	2.70	0.19	2.34	0.05	33.73	2.55
Scene	Time (s)	1.86	5.68	14.45	0.45	0.29	0.98	1.91	33.26
	Memory (MB)	2.80	2.74	2.74	0.14	4.97	0.05	71.09	4.33
QSAR	Time (s)	0.49	0.42	0.34	0.05	0.04	0.15	0.40	0.66
	Memory (MB)	0.30	0.28	0.23	0.16	1.66	0.05	3.62	1.20

H Hyperparameter λ sensitivity analysis

A key component of the URBoost algorithm is the hyperparameter λ, which controls the algorithm's sensitivity to the measured prediction uncertainty. To evaluate the impact of this parameter on model performance and to understand its tuning requirements, we conducted a sensitivity analysis on the Jasmine dataset under two distinct noise conditions, including slight (10% label noise) and severe (50% label noise). The effect of λ is multiplicative (Eq. (2) in the main paper), so its impact should

be tested on a logarithmic scale. Therefore, we varied λ across several orders of magnitude, from 0.1 to 100.0. The results are presented in Table 4.

The results revealed that the performance of URBoost is insensitive to the specific choice of λ across several orders of magnitude (0.1 to 100.0). This stability held true in both noise scenarios, indicating that even a small λ is sufficient to activate the uncertainty mechanism and effectively marginalize the high-conflict signals from corrupted labels. Once this filtering is engaged, further increasing λ creates a wide plateau of near-optimal performance without the risk of over-penalizing legitimate hard samples. This demonstrates that the uncertainty-weighting is a fundamental and robust property of the model, not a delicate heuristic, making URBoost practical and straightforward to deploy without extensive hyperparameter tuning.

Table 4: Hyperparameter λ sensitivity analysis on the Jasmine dataset under 10% and 50% noise conditions for URBoost.

Noise Ratio	λ	ACC (↑)	ROC AUC (↑)	F1 Score (↑)	Log Loss (↓)
10%	0.1	0.9364	0.9663	0.9343	0.1415
	0.5	0.9364	0.9658	0.9338	**0.1409**
	1.0	**0.9378**	0.9665	**0.9355**	0.1418
	5.0	**0.9378**	0.9658	0.9353	0.1413
	10.0	0.9371	0.9644	0.9346	**0.1410**
	50.0	0.9368	**0.9669**	0.9345	0.1412
	100.0	**0.9378**	0.9665	0.9355	0.1418
50%	0.1	0.9661	**0.9944**	0.9663	0.3369
	0.5	**0.9668**	**0.9944**	**0.9670**	**0.3367**
	1.0	0.9658	**0.9944**	0.9660	0.3378
	5.0	0.9661	**0.9944**	0.9663	0.3373
	10.0	0.9661	**0.9944**	0.9663	0.3391
	50.0	**0.9671**	**0.9944**	**0.9673**	0.3380
	100.0	0.9658	**0.9944**	0.9660	0.3378

I Related Work

The Evolution of Gradient Boosting. The modern success of boosting began with Friedman's insight to frame it as a functional gradient descent procedure Friedman (2001). This gave rise to Gradient Boosting Machine (GBM), a framework that has become the foundation for a generation of state-of-the-art algorithms Friedman (2002). The last decade has seen remarkable innovations focused on scalability and efficiency Yan et al. (2007); Dubout & Fleuret (2014). XGB introduced regularization and second-order derivative information Chen & Guestrin (2016). LGBM Ke et al. (2017) and CatBoost Prokhorenkova et al. (2018) further pushed the boundaries of performance through sophisticated sampling techniques and optimized handling of categorical features. However, these methods all inherit a critical, implicit assumption: the gradient is a uniformly reliable signal for model improvement. Their primary goal is to fit this gradient signal with maximal efficiency and precision, without questioning the trustworthiness of the signal itself for every sample. This blind trust is the root cause of their fragility in the presence of noisy data.

Robustness in Boosting and Ensemble Methods. The sensitivity of boosting to noise is a well-documented issue. Noisy labels can generate large, persistent gradients that hijack the learning process, forcing the ensemble to overfit to erroneous data points Frénay & Verleysen (2013). Several methods have been proposed to mitigate this. One line of research focuses on robust loss functions Masnadi-Shirazi & Vasconcelos (2008); Miao et al. (2015); Wang (2018); Luo et al. (2025). Instead of the standard exponential loss or log-loss, alternatives like the Huber loss or Savage loss have been employed to reduce the influence of samples with large errors Mason et al. (1999). While effective, this method modifies the global optimization objective rather than adapting the learning process on a sample-by-sample basis. Another strategy involves sample re-weighting, like AdaBoost (ADA) Freund & Schapire (1997). ADA, by its nature, up-weights misclassified samples, which can be particularly detrimental with noisy labels Dietterich (2000); Friedman et al. (2000). Variants like RobustBoost were designed to manage this Freund (2009). Other methods aim to identify and

either remove or down-weight potential outliers in a pre-processing step or during training. For instance, GOSS in LGBM down-weights samples with small gradients Ke et al. (2017), but this is primarily for efficiency and assumes that small-gradient samples are "well-trained," which is not always synonymous with information quality. To move beyond instantaneous gradient evaluation, recent work has begun analyzing the learning trajectory. Notably, ITBoost Su et al. (2026) examines the evolution of residuals across iterations, utilizing the Minimum Description Length principle to identify and down-weight samples that exhibit irregularly fluctuating residual patterns.