Investigating Uncertainty in Ensemble Methods

Gurman Bhullar
Department of Computer Science
University of Toronto
gbhullar@cs.toronto.edu

Rajesh Marudhachalam
Department of Computer Science
University of Toronto
rajesh1804@cs.toronto.edu

Sarah Hindawi
Department of Computer Science
University of Toronto
shindawi@cs.toronto.edu

Sumant Bagri
Department of Computer Science
University of Toronto
sbagri@cs.toronto.edu

Abstract

Accuracy and robustness are the most critical components of machine learning. Many high-risk applications, such as several applications in the medical and financial domains, can be transformed by deep learning. There is a growing need for diagnosis automation as there is an increasing demand for existing radiologists, which makes machine learning applications promising in this area. However, the diagnostic accuracy of these machine-learning models is uncertain [3]. Hence, it is critical to estimate the uncertainty in a model's predictions in order to prevent costly mistakes. Many software-intensive systems are relying on machine learning and Artificial intelligence techniques, and these systems are becoming an essential component of human life. There are many applications of ML that are used on a daily basis, including autonomous driving, credit card fraud detection systems, and medical diagnosis. These models should not only offer a defined functionality, they should also precisely describe the likelihood of their outcome being incorrect or outside a specified range of accuracy in order to help a system make decisions when faced with the uncertainty of embedded ML models, and potential safety-related consequences [7]. Due to the complexity and the empirical nature of the machine learning models, there might be some high-risk and high-priority situations where the preciseness of the outputs is crucial. In some domains, machine learning models are not widely deployed due to the uncertainty of their outputs. This can surely safeguard from uncertainty but, at the same time, will deprive the user of leveraging the automation benefits of machine learning and artificial intelligence. We tried to investigate the uncertainty in ensemble methods by using bagging and boosting techniques and compared the results obtained from the analysis of classification and regression tasks.

1 Introduction

Gradient boosting is a powerful machine learning technique that is particularly successful for tasks containing heterogeneous features and noisy data. While gradient boosting classification models return a distribution over class labels, regressions models typically yield only point predictions. However, for many practical, high-risk applications, it is also important to be able to quantify uncertainty in the predictions to avoid costly mistakes [8]. This project investigates uncertainty estimates of gradient-boosting decision tree (GBDT) models for both classification and regression tasks. Gradient boosting gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. We analyze the results on both synthetic and real datasets using two performance metrics, PRR and AUC-ROC, in order to understand the applicability

of such ensemble methods to gradient-boosting models. In order to generate the ensembles, we used Stochastic Gradient Boosting (SGB) and Stochastic Gradient Langevin Boosting (SGLB), and Random Forests. Our goal was to track the two main types of prediction uncertainties; data uncertainty (aleatoric) and knowledge uncertainty (epistemic). We analyzed the models on datasets that we chose from two different domains, financial and medical, as uncertainty estimates are most critical in such fields. We tried to improve existing implementations for both classification and regression tasks. Then we compared the results of these models using PRR and AUC-ROC metrics.

2 Related Work

Many researchers have published novel work on the field of finding the uncertainty in the machine learning models and many research papers have been published regarding this in the past decade. An overview of the prior work done in this field is as follows: A novel method is presented for model uncertainty estimation using machine learning techniques and its application in rainfall-runoff modelling.[10] In this method, first, the probability distribution of the model error is estimated separately for different hydrological situations and second, the parameters characterizing this distribution are aggregated and used as output target values for building the training sets for the machine learning model. This latter model, being trained, encapsulates the information about the model error localized for different hydrological conditions in the past and is used to estimate the probability distribution of the model error for the new hydrological model runs. The M5 model tree is used as a machine-learning model. In deep learning, Bayesian neural networks (BNNs) have been proposed to capture uncertainty as a Bayesian extension of deep neural networks. In BNNs, each weight is represented by a probability distribution instead of a real number, and learning comes down to Bayesian inference, i.e., computing the posterior [6].

For the representation of uncertainty, the technique of bayesian inference is used with is applicable to the Gaussian Processes as well. In the case of regression problems, the variance value of the posterior predictive distribution for any query is a meaningful indicator of the total uncertainty value.

[4] Use a Monte Carlo approach to quantify prediction uncertainty for random forest regression models. They test the approach by simulating maps of dependent and independent variables with known characteristics and comparing actual errors with prediction errors. Since this approach is data-driven, prediction intervals were either too wide or too narrow in sparse parts of the prediction distribution. This approach provides reasonable estimates of prediction uncertainty for random forest regression models.

The distinction between aleatoric and epistemic uncertainty has recently received a lot of attention in machine learning. [9] Show how two general approaches for measuring the learner's aleatoric and epistemic uncertainty in a prediction can be instantiated with decision trees and random forests as learning algorithms in a classification setting. The first approach is based on entropy measure and the second is a measure based on relative likelihood. Both the approaches provide reasonable reasons for a learner to abstain from a prediction.

Stochastic Gradient Langevin Boosting [11] is based on a special form of the Langevin diffusion equation specifically designed for gradient boosting and theoretically guarantees the global convergence even for multimodal loss functions, while standard gradient boosting algorithms can guarantee only local optimum.

Since our research is focused on estimating uncertainty to better understand the accuracy of the ML models used in high-risk domains, we conducted our experiments on 2 datasets from the medical and risk management domains. For classification, we used "default of credit card clients Data Set" which predicts "default payment next month" as a binary target value (Yes = 1, No = 0) to predict if a client is credible or not [2]. For regression, we used the "Parkinsons Telemonitoring" dataset, which is a medical dataset composed of a range of biomedical voice measurements from people with early-stage Parkinson's disease to predict the total Unified Parkinson's Disease Rating Scale (total UPDRS) [1].

3 Theoretical Analysis

3.1 Quantifying Uncertainty

In the domain of Bayesian inference for probabilistic models, given a dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$, we try to model the parameters θ by estimating its posterior distribution using Bayes' rule:

$$\mathbb{P}(\theta|\mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|\theta)\mathbb{P}(\theta)}{\mathbb{P}(\mathcal{D})} \tag{1}$$

A given ensemble of models $\{\mathbb{P}(y|x;\theta^{(m)})\}_{m=1}^{M}$ is assumed to have been generated from a known (or in most cases a close approximation) of a Bayesian posterior $\mathbb{P}(\theta|\mathcal{D})$. The predictive posterior of the ensemble is then defined as follows:

$$\begin{gathered}\mathbb{P}(y|x,\mathcal{D}) = \mathbb{E}_{\mathbb{P}(\theta|\mathcal{D})}\left[\mathbb{P}(y|x,\theta)\right] \\ \text{where, the sample meanestimator is used as an approximation for the expectation} \\ \mathbb{P}(y|x,\mathcal{D}) \simeq \frac{1}{M}\sum_{m=1}^{M}\mathbb{P}(y|x;\theta^{(m)})\end{gathered} \tag{2}$$

Uncertainty from the predictive posterior of the ensemble can be due to two factors: 1) aleatoric (data) uncertainty or 2) epistemic (knowledge) uncertainty. We can write the total uncertainty as a sum of aleatoric and epistemic uncertainties as shown below,

$$\text{Total Uncertainty = Data Uncertainty + Knowledge Uncertainty} \tag{3}$$

We define the total uncertainty as the entropy in the predictive posterior. This is defined below,

$$\mathcal{H}\left[\mathbb{P}(y|x,\mathcal{D})\right] = \mathbb{E}_{\mathbb{P}(y|x,\mathcal{D})}\left[-\log\mathbb{P}(y|x,\mathcal{D})\right] \tag{4}$$

Now, the data uncertainty for a generalized predictor can be visualized as the expected prediction uncertainty of each model in the ensemble. Mathematically, this can be written as the expected entropy of each model in the ensemble as below,

$$\text{Expected Data Uncertainty} = \mathbb{E}_{\mathbb{P}(\theta|\mathcal{D})}\left[\mathcal{H}(\mathbb{P}(y|x,\theta))\right] \tag{5}$$

Therefore, we can obtain the knowledge uncertainty by rearranging (3)

$$\begin{gathered}\text{Knowledge Uncertainty} = \text{Total Uncertainty} - \text{Data Uncertainty} \\ \text{Knowledge Uncertainty} \simeq \text{Total Uncertainty} - \text{Expected Data Uncertainty} \\ \text{Knowledge Uncertainty} \simeq \mathcal{H}\left[\mathbb{P}(y|x,\mathcal{D})\right] - \mathbb{E}_{\mathbb{P}(\theta|\mathcal{D})}\left[\mathcal{H}(\mathbb{P}(y|x,\theta))\right] \\ \text{this is precisely the information gain from knowing the data} \\ \mathcal{I}\left[y,\theta|x,\mathcal{D}\right] = \mathcal{H}\left[\frac{1}{M}\sum_{m=1}^{M}\mathbb{P}(y|x,\theta^{(m)}\right] - \frac{1}{M}\sum_{m=1}^{M}\mathcal{H}\left[\mathbb{P}(y|x,\theta)\right]\end{gathered} \tag{6}$$

However, we can use the above formulation only for classification tasks where predictions are actually indicative of probabilities. For regression tasks, the predictions are point values and we cannot use them as probabilistic estimates for a model. For regression tasks, uncertainty is estimated using total variation which is defined below,

$$\mathbb{V}_{p(y|x,\mathcal{D})}[y] = \mathbb{V}_{p(y|x,\mathcal{D})}[\mathbb{E}_{p(y|x,\mathcal{D})}[y]] + \mathbb{P}_{p(\theta|\mathcal{D})}[\mathbb{V}_{p(y|x,\theta)}[y]] \tag{7}$$

$$\mathbb{V}_{p(y|x,\mathcal{D})}[y] \simeq \frac{1}{M}\sum_{m=1}^{M}\left[\left(\sum_{m-1}^{M}\frac{\mu_m}{M}\right) - \mu_m\right]^2 + \frac{1}{M}\sum_{m=1}^{M}\sigma_m^2 \quad , \quad \{\mu_m,\sigma_m\} = f(x;\theta^{(m)}) \tag{8}$$

The parameters, $\{\mu_m, \sigma_m\}$ are estimated by the training algorithm that models a normal distribution over the target y given features x by optimizing the negative log likelihood for $\mathbb{P}(y|x,\theta)$. This is

mathematically presented as follows,

$$F^{(t)}(x) : X \to \mathbb{R}^2 \Rightarrow \left\{\mu_m^{(t)}, \sigma_m^{(t)}\right\}$$
$$\text{such that,}$$
$$\mathbb{P}(y|x, \theta^{(t)}) = \mathcal{N}\left(y|\mu_m^{(t)}, \sigma_m^{(t)}\right), \text{ where,} \tag{9}$$
$$\hat{\theta}^{(t)} = \arg\min_{\theta} \mathbb{E}_{\mathcal{D}}[-\log p(y|x, \theta)] = \arg\min_{\theta} \left\{-\frac{1}{N}\sum_{i=1}^{N} \log \mathbb{P}(y^{(i)}|x^{(i)}, \theta)\right\}$$

Using these uncertainty measures, we can formulate a thresholding-based binary classification problem where the predictor/detector $\mathcal{I}_T(x)$ takes in a new input **x** and assigns a label 1 (uncertain prediction) if the uncertainty measure $\mathcal{H}(x)$ is above a threshold T and label 0 (confident prediction) otherwise. This is presented, mathematically, as below,

$$\mathcal{I}_T(x) = \begin{cases} 1 & , \mathcal{H}(x) > T \\ 0 & , \mathcal{H}(x) \leq T \end{cases} \tag{10}$$

3.2 Gradient Boosted Decision Trees (GBDT)

Given a dataset $\mathcal{D}$ and a loss function $L : \mathbb{R}^2 \to \mathbb{R}$, the gradient boosting algorithm iteratively constructs a model $F : X \to \mathbb{R}$ to minimize the empirical risk $\mathcal{L}(F|\mathcal{D}) = \mathbb{E}_{\mathcal{D}}[L(F(x), y)]$ [5]. A weak learner $h^{(t)}$ is chosen at each time-step (t) from a family of functions $\mathcal{H}$ and the model is updated as follows,

$$F^{(t)}(x) = F^{(t-1)}(x) + \epsilon \cdot h^{(t)}(x) \tag{11}$$
$$\text{such that,}$$
$$h^{(t)} = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{\mathcal{D}} \left[\left(-\left.\frac{\partial \mathcal{L}(y, s)}{\partial s}\right|_{s=F^{(t-1)}(x)} - h(x)\right)^2\right] \tag{12}$$

Here, ϵ is the learning rate for the model F. For the GBDT ensemble the weak learners are decision trees defined using a set of parameters $\theta^{(t)}$ such that $h(x, \phi^{(t)}) = \sum_{j=1}^{d} \phi_j^{(t)} \mathbf{1}_{x \in R_j}$. The ensemble is constructed by summing over all the decision trees

3.3 Generating Ensembles

One of the most important requirements for calculating uncertainty is ensemble models is that each model in the ensemble should be sampled from a posterior distribution. $\mathbb{P}(\theta|\mathcal{D})$. Stochastic Gradient Boosting (SGB) and Stochastic Gradient Langevin Boosting (SGLB)

SGB ensembles

SGB ensemble of GBDTs are generated by subsampling the dataset $\mathcal{D}$ at each iteration of training. A randomly subsampled dataset $\mathcal{D}' \subset \mathcal{D}$ is generated either using bootstrap or uniformly randomly. The fraction of the dataset sampled is called the *sample rate*. The sampling is done independently resulting in an ensemble of independent models $\{\theta^{(m)}\}_{m=1}^{M}$.

The model updates are done using (10) and (11).

SGLB ensembles

SGLB combines gradient boosting along with Langevin dynamics **?**] to achieve convergence to global optimum even for non-convex loss functions. The algorithm has two main differences compared to

SGB [12]

First, Gaussian noise is added to the SE calculation

$$h^{(t)} = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{\mathcal{D}} \left[\left(\frac{\partial \mathcal{L}(y,s)}{\partial s}\Big|_{s=F^{(t-1)}(x)} - h(x) + v \right)^2 \right], v \sim \mathcal{N}\left(0, \frac{2}{\beta_\epsilon}\mathbf{I}\right) \tag{13}$$

Second,

$$F^{(t)}(x) = (1 - \gamma\epsilon)F^{(t-1)}(x) + \epsilon \cdot h^{(t)}(x, \phi^{(t)}) \tag{14}$$

Here, β is called the diffusion temperature and helps in controlling the random exploration induced by v. The random exploration is what helps SGLB achieve global optimum

Optimizers and Training Metrics

Classification

The LogLoss optimizer is used for training the classification models using SGB and SGLB algorithms. This is defined as follows,

$$\mathcal{L}\left(y, \{\mathbb{P}(y|x;\theta^{(m)})\}_{m=1}^{M}\right)_{LogLoss} = -\frac{\sum_{m=1}^{M} \left\{ w_m \left(t \log\left[\mathbb{P}(y|x;\theta^{(m)})\right] + (1-t)\log\left[1 - \mathbb{P}(y|x;\theta^{(m)})\right]\right)\right\}}{\sum_{i=m}^{M} w_m} \tag{15}$$

The $0-1$ Loss is used as the evaluation metric during training and testing

Regression

For regression, the RMSEWithUncertainty optimizer is used. This is described below,

$$\mathcal{L}\left(y, \{\mathbb{P}(y|x;\theta^{(m)})\}_{m=1}^{M}\right)_{\text{RMSEWithUncertainty}} = -\frac{\sum_{i=1}^{N} \left\{ w_m \log\left[\mathcal{N}(y|a_{m,0}, e^{2a_{m,1}})\right]\right\}}{\sum_{m=1}^{M} w_m}$$

$$\mathcal{L}_{\text{RMSEWithUncertainty}} = \frac{1}{2}\log(2\pi) + \frac{\sum_{m=1}^{M} \left\{ w_m \left(a_{m,0} + \frac{1}{2} e^{-2a_{m,1}} (t - a_{m,0})^2 \right)\right\}}{\sum_{m=1}^{M} w_m} \tag{16}$$

Here $a_{m,0}$ is the mean prediction (μ_m) and $a_{m,1}$ is the $\log(\sigma_m)$ prediction such that the model m is sampled from $\mathcal{N}(y|\mu_m, \sigma_m^2) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-\frac{(y-\mu_m)^2}{2\sigma^2}\right)$

The RMSE score is used as the evaluation metric during training and testing

3.4 Metrics for evaluating uncertainty

It is important to know whether the prediction uncertainty from an ensemble (as computed using methods stated above) is able to provide any meaningful insights on predictions: Whether the model will make an error in prediction or whether it will classify it correctly.

AUC-ROC (for out-of-domain datasets)

OOD detection is done using a dataset consisting of both in-domain ($\mathcal{I} \in \mathcal{D}_{in}$) and out-of-domain ($\mathcal{O} \in \mathcal{D}_{out}$) datapoints. Points belonging to $\mathcal{I}$ are labeled as 1 while points in $\mathcal{O}$ are labeled as 0. Therefore the model for the ensemble becomes $F : X \to \{0, 1\}$. The model is trained on the training dataset and its predictions $\mathcal{P}$ are collected from the test dataset ($\mathcal{D}_{test}$) with true labels $\mathcal{T}$.

The strategy is to then evaluate the *true positive rate (tpr)* and the *false positive rates (fpr)* as follows

$$\begin{aligned} tpr &= \frac{\sum_{x \in \mathcal{D}_{test}} \mathcal{I}_T(x|\mathcal{T}(x) = 0)}{\sum_{x \in \mathcal{D}_{test}} \mathbb{1}\{\mathcal{T}(x) = 0\}} \\ fpr &= \frac{\sum_{x \in \mathcal{D}_{test}} \mathcal{I}_T(x|\mathcal{T}(x) = 1)}{\sum_{x \in \mathcal{D}_{test}} \mathbb{1}\{\mathcal{T}(x) = 1\}} \end{aligned} \tag{17}$$

where $\mathcal{I}_T$ is the detector/predictor as defined in (4.1)

Subsequently, we plot the ROC curvce as: tpr vs fpr, and compute the area under the curve (AUC). The higher the value of AUC-ROC the better is the out-of-domain detection.

Prediction-Rejection Ratio (PRR)

For a classification problem, the measure of uncertainty provides a means to label model predictions as certain or uncertain. Ideally, the goal is to be able to detect all the inputs which the model has misclassified with a label of "uncertain prediction". If the detector $\mathcal{I}_T$ (for a given uncertaintly threshold T), is able to achieve this then the model can choose not to provide any prediction for these inputs, or pass them over to an "oracle" (human) as "rejected-for-prediction". This can be visualized as a rejection curve $\mathcal{R}$ on the percentage of data points in $f * N(\mathcal{D}_{test}) : f \in [0, 1]$

The base error (e_{base}) in classification is defined as the error after the model(F) (and the detector $\mathcal{I}_T$) have gone through the whole test dataset $\mathcal{D}_{test}$. We consider two boundary cases for the detector $\mathcal{I}_T$,

Case 1: Detector is completely unreliable always producing useless estimates for uncertainty ($\mathcal{R}_{rand}$)

Case 2: Detector is perfect and always bigger for a misclassification than for a correct classification. In this case, the curve will represent the oracle curve (where all uncertain misclassifications are given to the oracle to classify) ($\mathcal{R}_{orc}$)

Any other detector would lie between the above two boundary cases and is denoted as $\mathcal{R}_{uns}$

$\mathcal{R}_{rand}$ is a straight line from e_{base} at 0% rejection to 0 at 100% rejection as the rejections are random:

$$\mathcal{R}_{rand} - 0 = \frac{e_{base} - 0}{0 - 1} \times (f - 1) \tag{18}$$

$$\mathcal{R}_{rand} = (1 - f) \times e_{base} \tag{19}$$

$\mathcal{R}_{rand}$ is a straight line from e_{base} at 0% rejection to 0 at the percentage of misclassification $\left(N_{misclassify} = \sum_{i=1}^{N(\mathcal{D}_{test})} \mathbb{1}[y \neq F(x)] \Rightarrow f' = \frac{\sum_{i=1}^{N(\mathcal{D}_{test})} \mathbb{1}[y \neq F(x)]}{N(\mathcal{D}_{test})} \right)$ as the "perfect" detector will have no need to reject any correctly classfied datapoints:

$$\mathcal{R}_{orc} - 0 = \frac{e_{base} - 0}{0 - f'} \times (f - f') \tag{20}$$

$$\mathcal{R}_{rand} = \begin{cases} e_{base} \times (1 - \frac{f}{f'}) & , f < f' \\ 0 & , 1 \geq f \geq f' \end{cases} \tag{21}$$

The quality of th rejection curve is assesed by considering the *ratio* of the area between $\mathcal{R}_{uns}$ and $\mathcal{R}_{random}$ and is called the *Prediction-Rejection-Ratio (PRR)* as shown below,

$$PRR = \frac{AUC(R_{random}) - AUC(R_{orc})}{AUC(R_{uns}) - AUC(R_{orc})} \tag{22}$$

A higher value of PRR indicates that the detector is closer to the oracle in terms of classification error rate and is more certain about the misclassified labels.

4 Experimental Setup

In this section, we will evaluate the performance of the ensemble models on both classification and regression tasks, with a focus on their ability to detect errors and out-of-domain inputs. All

the GDBT models were implemented using the open source CatBoost library as this is known to achieve consistently good performance on many tasks, and we used the RandomForestRegressor implementation from sci-kit learn library. All the models are trained by optimizing the negative log-likelihood. We compared the algorithms on a real-world classification and regression task.

Detecting errors & anomalous inputs: We use Prediction-Rejection Ratio - measures the level of correlation between errors and rank-orders them for Error detection. Out-of-domain detection is performed using the area under the ROC curve. As obtaining 'real' OOD samples for the datasets considered in this work is challenging, we used synthetic OOD data collected in the following manner. For each of the 2 datasets, we took its test set as in-domain samples and sampled an OOD dataset of the same size from the Internet dataset to get OOD data. The numerical features in OOD data are standardized, we randomly sampled a category uniformly from the set of all feature's categories in case of boosting and used a leave-one-out-encoding for bagging.

5 Hyperparameter tuning

The hyper-parameters of the Random forest classifier are tuned for finding the best combination of the parameters that we can use for the purpose of classification and regression using the method of bagging. The parameter chosen for tuning classification and regression bagging model are the number of estimators, number of features, maximum depth, criterion. The values chosen for the grid search are :

Table 1: Hyper-parameter tuning parameters

Parameters	**Classification**	**Regression**
Number of estimators	1000,3000,5000	1000,3000,5000
Maximum depth	5,10,15,20	5,10,15,20
Criterion	Gini, Entropy	-
Maximum features	auto, sqrt, log2	auto, sqrt, log2

Table 2: Hyper-parameter tuning best parameters

Parameters	**Classification**	**Regression**
Number of estimators	1000	5000
Maximum depth	15	20
Criterion	Entropy	NA
Maximum features	auto	auto

6 Results

We know that test errors can occur due to noise and lack of knowledge, hence we rank elements by total uncertainty expecting a better PRR value. We were expecting a clear winner to emerge in these scenarios, however, to out surprise, the results we got for both the tasks across both the ensemble techniques were a mix and match.

Table 3 shows the results of the boosting GBDT models. We noticed that *Total Uncertainty* consistently yields better PPR results across both the datasets, although the ensembles don't outperform the single models. In contrast to our expectation, we noticed that *Knowledge Uncertainty* didn't yield better *out-of-domain* detection performance in terms of AUC-ROC consistently for both the tasks, and *Total Uncertainty* yielded the best OOD detection in the regression task.

Notably, ensembles don't outperform single models expect for the case of OOD detection in the classification. We believe it could be due to the following. Firstly, the GDBT models are ensembles by themselves, as boosting is of additive by itself. Next, we think estimates of *Total Uncertainty* for the GBDT models doesn't have a great impact from the *Knowledge Uncertainty* obtained via the approaches.

Table 3: Boosting

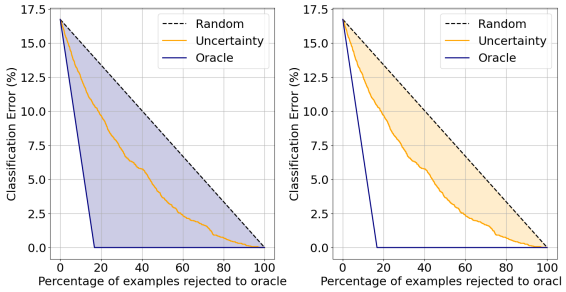
		% PRR(↑)				% AUC-ROC(↑)			
		Single		Ensemble		Single		Ensemble	
Dataset		SGB	SGLB	SGB	SGLB	SGB	SGLB	SGB	SGLB
Classification - Boosting									
Credit	TU	45	**46**	46	46	78	75	80	75
	KU	-	-	20	18	-	-	99	**99**
Regression - Boosting									
Parkinsons	TU	15	**50**	6	15	**100**	97	88	64
	KU	-	-	11	13	-	-	85	59

In GBDT we incrementally add trees in order to rectify the previous model's errors, where as *Random Forest* is made up of decision trees that use sub-samples of the data to independently train. Drawing a parallel to the ensembles in the boosting, we use an ensemble of multiple independently trained random forest models. Table 4 shows the results of the bagged models, that is, the Random Forest based classifier and regressor. We noticed that GBDT based SGLB model outperformed RF, this was especially evident for OOD detection. Notably, the single and ensemble models performed similarly in yielding both the metrics. Hence, we can say that for random forests, a single model is a good (and cheap) alternative to the ensemble.

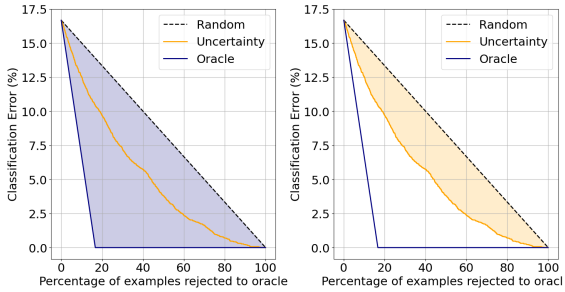
Table 4: Bagging

		% PRR(↑)				% AUC-ROC(↑)			
		Single		Ensemble		Single		Ensemble	
Dataset		SGLB	RF	SGLB	RF	SGLB	RF	SGLB	RF
Classification - Bagging									
Credit	TU	45	55	46	**56**	75	77	75	78
	KU	-	-	18	30	-	-	**99**	63
Regression - Bagging									
Parkinsons	TU	**50**	20	15	5	**97**	50	64	92
	KU	-	-	13	5	-	-	59	92

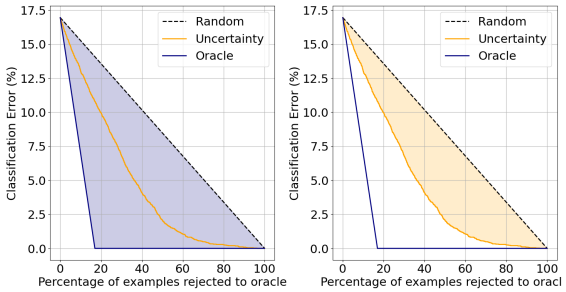
We present the Prediction-Rejection (PR) and ROC curves for SGLB and RF models on the Credit dataset. The PR curves show the best rejection characteristics of an ideal model (Oracle) and that of the actual model (Uncertainty).



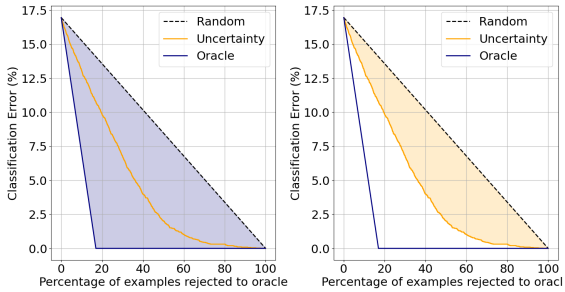
(a) Prediction-Rejection Curve for single SGLB model



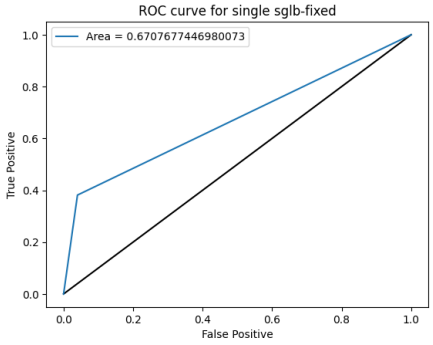
(b) Prediction-Rejection Curve for ensemble SGLB model



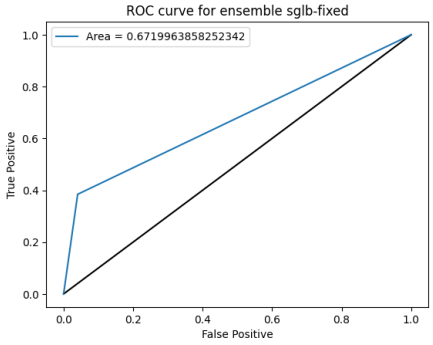
(c) Prediction-Rejection Curve for single RF model



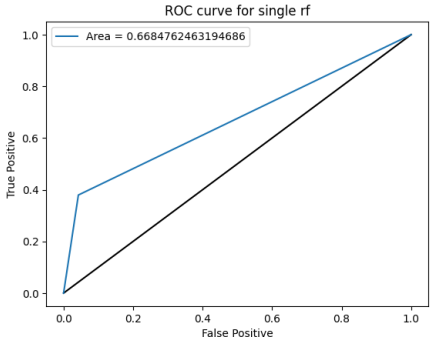
(d) Prediction-Rejection Curve for ensemble RF model



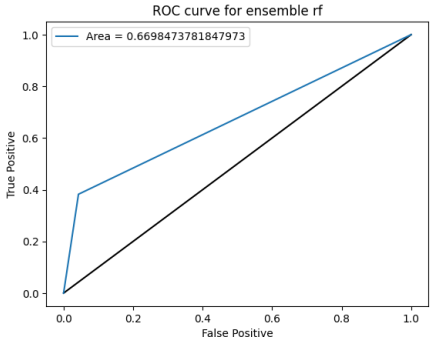
(a) ROC curve for single SGLB model



(b) ROC curve for ensemble SGLB model



(c) ROC curve for single RF model



(d) ROC Curve for ensemble RF model

7 Conclusions

This study examined ensemble-based uncertainity estimated for GBDT and Random forest models. Two approaches for generating the ensembles of GDBT models were Stochastic Gradient Boosting (SGB) and Stochastic Gradient Langevin Boosting (SGLB). Properties of the estimates of total and knowledge uncertainty derived from ensembles were analyzed on continuous data. Experiments were conducted on two datasets, one for classification and one for regression. In case of Boosting, the intial few runs of the models on the datasets gave completely unreliable always producing useless estimates for uncertainty, but upon hyperparameter tuning, the total uncertainty yielded better results for the GBDT models for both classification and regression.

The only exception observed in the results is for the out of domain detection (AUC-ROC), for the ensembles values of the Knowledge Uncertainty in case of classification. For the bagging part, the SGLB performed the best in almost all the scenarios and gave the best results. The exception observed in this analysis is RF outperforms SGLB models in detecting errors in the classification task, in other cases SGLB yeilds better results. In contrast to our expectations, there wasn't a clear winner in any scenario, and the detector seemed to produce unreliable results on multiple occations. This might be so, as we were able to run experiments only on two datasets due to time and resource constraints. We believe running these experiments on more datasets for both the Boosting and Bagging ensemble models with different hyperparameter values might yield better results, and in turn present a clear winner for a few scenarios.

In future work, we want to investigate how uncertainty can be affected when boosti ng is used to prune a bagged ensemble. This involves only preserving classifiers that are essential for the classification while eliminating classifiers whose contribution is redundant. This method preserves the bagging performance in noisy classification tasks, allowing it to outperform boosting in noisy classification tasks. As expected, pruning also increases the classification speed, reduces memory requirements, and can increase classification performance [6].

References

[1] https://archive-beta.ics.uci.edu/dataset/189/parkinsons+telemonitoring.

[2] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients.

[3] R. Aggarwal, V. Sounderajah, G. Martin, D. S. Ting, A. Karthikesalingam, D. King, H. Ashrafian, and A. Darzi. Diagnostic accuracy of deep learning in medical imaging: A systematic review and meta-analysis. *NPJ digital medicine*, 4(1):1–23, 2021.

[4] J. W. Coulston, C. E. Blinn, V. A. Thomas, and R. H. Wynne. Approximating prediction uncertainty for random forest regression models. *Photogrammetric Engineering & Remote Sensing*, 82(3):189–197, 2016.

[5] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[6] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[7] M. Kläs and A. M. Vollmer. Uncertainty in machine learning applications: A practice-driven classification of uncertainty. In *International conference on computer safety, reliability, and security*, pages 431–438. Springer, 2018.

[8] A. Malinin, L. Prokhorenkova, and A. Ustimenko. Uncertainty in gradient boosting via ensembles. *arXiv preprint arXiv:2006.10562*, 2020.

[9] M. H. Shaker and E. Hüllermeier. Aleatoric and epistemic uncertainty with random forests. In *International Symposium on Intelligent Data Analysis*, pages 444–456. Springer, 2020.

[10] D. P. Solomatine and D. L. Shrestha. A novel method to estimate model uncertainty using machine learning techniques. *Water Resources Research*, 45(12), 2009.

[11] A. Ustimenko and L. Prokhorenkova. Sglb: Stochastic gradient langevin boosting. In *International Conference on Machine Learning*, pages 10487–10496. PMLR, 2021.

[12] A. Ustimenko and L. Prokhorenkova. Sglb: Stochastic gradient langevin boosting. In *International Conference on Machine Learning*, pages 10487–10496. PMLR, 2021.

A Individual Contributions

Table 1: Brief summary of individual contributions

Name	Summary
Gurman	Hyperparameter tuning for classification and regression using bagging approach. Report writing and compilation, Made necessary changes to generate plots for different metrics.
Rajesh	Inherited code for boosting & developed code for bagging, identified & preprocessed datasets and generated out-of-domain dataset. Trained the models, conducted experiments for boosting & bagging, and aggregated the results. Responsible for the experimental setup, results and conclusion in the report.
Sarah	Accommodated new datasets so that the model would be able to process the data, made changes to generate metrics and plots to better evaluate the outputs of the model.
Sumant	Brainstorming initial project idea. Extensive theoretical analysis on different methodologies for evaluating uncertainty. Final report writing: Theoretical Analysis section.