Unbiased Gradient Boosting Decision Tree with Unbiased Feature Importance

Zheyu Zhang , **Tianping Zhang** and **Jian Li**

Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University

{zheyu-zh20, ztp18}@mails.tsinghua.edu.cn, lijian83@mail.tsinghua.edu.cn

Abstract

Gradient Boosting Decision Tree (GBDT) has achieved remarkable success in a wide variety of applications. The split finding algorithm, which determines the tree construction process, is one of the most crucial components of GBDT. However, the split finding algorithm has long been criticized for its bias towards features with a large number of potential splits. This bias introduces severe interpretability and overfitting issues in GBDT. To this end, we provide a fine-grained analysis of bias in GBDT and demonstrate that the bias originates from 1) the systematic bias in the gain estimation of each split and 2) the bias in the split finding algorithm resulting from the use of the same data to evaluate the split improvement and determine the best split. Based on the analysis, we propose *unbiased gain*, a new unbiased measurement of gain importance using out-of-bag samples. Moreover, we incorporate the unbiased property into the split finding algorithm and develop UnbiasedGBM to solve the overfitting issue of GBDT. We assess the performance of UnbiasedGBM and unbiased gain in a large-scale empirical study comprising 60 datasets and show that: 1) UnbiasedGBM exhibits better performance than popular GBDT implementations such as LightGBM, XGBoost, and Catboost on average on the 60 datasets and 2) unbiased gain achieves better average performance in feature selection than popular feature importance methods. The codes are available at https://github.com/ZheyuAqaZhang/UnbiasedGBM.

1 Introduction

Gradient Boosting Decision Tree (GBDT) is one of the most widely used machine learning models and has been applied in numerous domains, including medicine, finance, climate science, and healthcare. GBDT is particularly popular in modeling tabular data [Chen and Guestrin, 2016; Shwartz-Ziv and Armon, 2022; Gorishniy *et al.*, 2021; Grinsztajn *et al.*, 2022]. In the training process of a GBDT model, we need to construct decision trees one by one. During tree construction, we need to determine the best split and the split finding algorithm is one of the most crucial components in GBDT. In the standard implementation [Friedman, 2001; Chen and Guestrin, 2016; Ke *et al.*, 2017], the best split is chosen based on the reduction in loss (decrease in impurity) of all candidate splits in all features. However, this split finding algorithm has long been criticized for its bias towards features that exhibit more potential splits [Breiman *et al.*, 1984; Strobl *et al.*, 2007; Boulesteix *et al.*, 2012; Nicodemus, 2011]. Due to the increased flexibility afforded by a larger number of potential split points, features with higher cardinality (such as continuous features and features with a large number of categories) have a higher probability of being split than features with lower cardinality (such as binary features). This bias introduces two problems in GBDT:

- Interpretability issue. The *gain importance* [Breiman *et al.*, 1984; Hastie *et al.*, 2001] in GBDT sums up the total reduction of loss in all splits for a given feature, and is frequently used to explain how influential a feature is on the models' predictions. However, gain importance is not reliable due to its bias towards features with high cardinality [Breiman *et al.*, 1984]. As we illustrate in Example 1 in Section 4, a continuous feature independent of the target may have higher gain importance than a binary feature related to the target.

- Overfitting issue. During tree construction, the split finding algorithm biases towards choosing features with high cardinality [Breiman *et al.*, 1984; Strobl *et al.*, 2007]. Moreover, the split finding algorithm uses training set statistics to determine the best split and does not evaluate the generalization performance of each split.

Existing studies to address the bias problems mostly fall into two categories: 1) they propose a post hoc approach to calculate unbiased or debiased feature importance measurement [Zhou and Hooker, 2021; Li *et al.*, 2019], and 2) they propose new tree building algorithms by redesigning split finding algorithms [Loh and Shih, 1997; Kim and Loh, 2001; Loh, 2009; Strobl *et al.*, 2007]. However, these methods mostly focus on random forests, and cannot generalize to GBDT. One of the main reasons is that, different from most random forest implementations, existing GBDT implementations employ the second-order approximation of the objective function to evaluate split-improvement [Chen and Guestrin, 2016; Ke *et al.*, 2017] (see more detailed discussions in the

related work section). Since popular GBDT implementations, such as XGBoost [Chen and Guestrin, 2016] and CatBoost [Dorogush *et al.*, 2018], have been dominating tabular data modeling [Gorishniy *et al.*, 2021; Shwartz-Ziv and Armon, 2022], there is an urgent need to address the interpretability and overfitting issues caused by the bias in GBDT.

To study the causes of the bias in GBDT, we conduct a fine-grained analysis, which reveals that the bias originates from: 1) the systematic bias in each split's gain estimation. We discover that the calculation of gain is a biased estimation of the split improvement, and is almost always positive. 2) The bias in the split finding algorithm due to the fact that it evaluates the split improvement and determines the best split using the same set of data. According to the analysis, first, we construct an unbiased measurement of feature importance for GBDT by using out-of-bag samples. This new measurement is unbiased in the sense that features with no predictive power for the target variable has an importance score of zero in expectation. Next, we incorporate the unbiased property into the split finding algorithm during tree construction and propose UnbiasedGBM. Compared with existing GBDT implementations (such as LightGBM [Ke *et al.*, 2017], XGBoost [Chen and Guestrin, 2016], and CatBoost [Dorogush *et al.*, 2018]), UnbiasedGBM has two advantages:

1. The split finding algorithm unbiasedly chooses among features with different cardinality to mitigate overfitting.

2. UnbiasedGBM evaluates the generalization performance of each split and performs leaf-wise early-stopping to avoid overfitting splits.

The contributions of this paper are summarized as follows:

1. We propose unbiased gain, an unbiased measurement of feature importance in GBDT to address the interpretability issue due to the bias in the split finding algorithm.

2. We propose UnbiasedGBM by integrating the unbiased property into the split finding algorithm to mitigate overfitting.

3. We provide a large-scale empirical study comprising 60 datasets to show that: 1) UnbiasedGBM exhibits better performance on average than LightGBM, XGBoost, and Catboost, and 2) unbiased gain achieves better average performance in feature selection than gain importance, permutation feature importance, and SHAP importance.

2 Related Work

Existing methods to correct the bias in the split finding algorithm fall primarily into two categories: 1) they propose a new method to compute debiased or unbiased feature importance measurement. 2) They propose new tree construction algorithms by redesigning the split finding algorithm.

There has been a line of work to develop new methods for computing debiased or unbiased feature importance. Quinlan [Quinlan, 1986] proposed information gain ratio to overcome the bias in classification trees. Sandri and Zuccolotto [Sandri and Zuccolotto, 2008] decomposed split-improvement into the reduction in loss and a positive bias. They used a pseudo dataset to estimate and subtract the

bias. Nembrini et al. [Nembrini *et al.*, 2018] then improved the computing efficiency of this approach. Li et al. [Li *et al.*, 2019] proposed a debiased feature importance measure. However, their method still yields biased results. Zhou and Hooker [Zhou and Hooker, 2021] proposed an unbiased measurement of feature importance in random forests. Nonetheless, the theoretical analysis relies on using mean squared error to justify the unbiased property of their method and cannot be generalized to GBDT, which often employs different loss functions for tree construction. In this paper, we propose unbiased gain, an unbiased measurement of feature importance in GBDT. Our method enjoys several advantages compared with previous methods: 1) Our method does not generate pseudo data that incurs additional cost as in Sandri and Zuccolotto [Sandri and Zuccolotto, 2008] and Nembrini et al. [Nembrini *et al.*, 2018]. 2) Our method can be easily used in GBDT implementations and has the theoretical guarantee of being unbiased, whereas Zhou and Hooker [Zhou and Hooker, 2021] cannot generalize to GBDT.

There has been another line of works that develop new tree building algorithms to remove the bias, such as QUEST [Loh and Shih, 1997], CRUISE [Kim and Loh, 2001], GUIDE [Loh, 2009], and cforest [Strobl *et al.*, 2007]. However, these methods cannot generalize to GBDT for a variety of reasons. For example, QUEST, CRUISE, and GUIDE use classification trees, whereas GBDT uses regression trees for both classification and regression tasks and supports various loss functions. cforest [Strobl *et al.*, 2007] separates the variable selection and the splitting procedure to remove the bias. However, this method incurs an excessive amount of computational overhead, as variable selection is typically costly. We are the first to integrate the unbiased property into GBDT and develop UnbiasedGBM to address the overfitting problem caused by the bias in GBDT.

3 Background

We briefly introduce the GBDT model in which the second order approximation is used in the training (e.g., XGBoost [Chen and Guestrin, 2016], LightGBM [Ke *et al.*, 2017]). Note that we formulate the GBDT objective under the population distribution as opposed to the traditional formulation of GBDT utilizing the empirical distribution [Chen and Guestrin, 2016; Li, 2012]. This formulation is essential, and it allows us to examine and comprehend the bias in GBDT.

3.1 Gradient Boosting Decision Trees

Consider the dataset $D = \{(\mathbf{x}, y)\}$, where $(\mathbf{x}, y)$ are independent and identically distributed from an unknown distribution $\mathcal{T}$. A tree ensemble model uses K additive functions to model the distribution $\mathcal{T}$ and predict the output:

$$\hat{y} = \phi(\mathbf{x}) = \sum_{k=1}^{K} f_k(\mathbf{x}), f_k \in \mathcal{F},$$

where $\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ is the space of regression trees. Here q represents the tree structure and $q(\mathbf{x})$ maps an example $\mathbf{x}$ to the leaf index. We construct the tree ensemble in an additive manner to minimize the objective function

$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{x},y}\left[l(\phi(\mathbf{x}),y)\right]$. Let ϕ_t be the model at the t-th iteration and $\hat{y}_t$ be the corresponding prediction. We greedily add a new regression tree f_t that most improves the objective function $\mathcal{L}(\phi_{t-1} + f_t)$. This is achieved by using the second-order approximation:

$$\mathcal{L}(\phi_{t-1} + f_t) \approx \mathbb{E}_{\mathbf{x},y}[l(\hat{y}_{t-1},y) + g(\mathbf{x},y)f_t(\mathbf{x})$$
$$+ \frac{1}{2}h(\mathbf{x},y)f_t(\mathbf{x})^2],$$

where

$$g(\mathbf{x},y) = \frac{\partial l(\phi_{t-1}(\mathbf{x}),y)}{\partial \phi_{t-1}(\mathbf{x})}, \ h(\mathbf{x},y) = \frac{\partial^2 l(\phi_{t-1}(\mathbf{x}),y)}{(\partial \phi_{t-1}(\mathbf{x}))^2}.$$

We can simplify the objective function by removing the constant terms:

$$\tilde{\mathcal{L}}(\phi_{t-1} + f_t) = \mathbb{E}_{\mathbf{x},y}\left[g(\mathbf{x},y)f_t(\mathbf{x}) + \frac{1}{2}h(\mathbf{x},y)f_t(\mathbf{x})^2\right].$$

For a leaf node I in the tree structure, the loss $\mathcal{L}(I)$ contributed by the leaf is

$$\mathcal{L}(I) = \mathbb{E}_{\mathbf{x},y}\left[\mathbf{1}_{\{q(\mathbf{x})=I\}}\left(g(\mathbf{x},y)f(x) + \frac{1}{2}h(\mathbf{x},y)f(x)^2\right)\right]$$
$$= \mathbb{E}_{\mathbf{x},y}\left[\mathbf{1}_{\{q(\mathbf{x})=I\}}\left(g(\mathbf{x},y)w_I + \frac{1}{2}h(\mathbf{x},y)w_I^2\right)\right]$$
$$= P(\mathbf{x} \in I)\left(\mu_g(I)w_I + \frac{1}{2}\mu_h(I)w_I^2\right),$$

where $\mu_g(I) = \mathbb{E}_{\mathbf{x},y}\left[g(\mathbf{x},y)\right]$ and $\mu_h(I) = \mathbb{E}_{\mathbf{x},y}\left[h(\mathbf{x},y)\right]$. We can calculate the optimal weight w_I of leaf I by

$$w_I = -\frac{\mu_g(I)}{\mu_h(I)}$$

and compute the corresponding optimal loss by

$$\mathcal{L}(I) = -\frac{1}{2}\frac{\mu_g(I)^2}{\mu_h(I)}P(\mathbf{x} \in I). \tag{1}$$

Consider a split on feature X_j at a splitting point s, which results in two child nodes $I_L = \{(\mathbf{x},y)|x_j \le s\}$ and $I_R = \{(\mathbf{x},y)|x_j > s\}$. The gain of the split $\theta = (j,s)$ is defined as the reduction in loss:

$$\text{Gain}(I,\theta) = \mathcal{L}(I) - \mathcal{L}(I_L) - \mathcal{L}(I_R). \tag{2}$$

In practice, the distribution $\mathcal{T}$ is usually unknown, therefore we cannot directly calculate $\mu_g(I)$ and $\mu_h(I)$. Instead, we use the training dataset to estimate $\mu_g(I)$ and $\mu_h(I)$. Given a training dataset with n examples and m features $\mathcal{D}_{tr} = (X,Y) = \{(\mathbf{x}_i,y_i)\}$, where $|\mathcal{D}_{tr}| = n$ and $(\mathbf{x}_i,y_i) \overset{\text{iid}}{\sim} \mathcal{T}$, we estimate the loss on leaf I and the gain of a split by

$$\widetilde{\mathcal{L}}(I) = -\frac{1}{2}\frac{\left(\frac{1}{n_I}\sum_{i\in I}g_i\right)^2}{\frac{1}{n_I}\sum_{i\in I}h_i}\frac{n_I}{n} = -\frac{1}{2n}\frac{G_I^2}{H_I}, \tag{3}$$

$$\widetilde{\text{Gain}}(I,\theta) = \frac{1}{2n}\left(\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{G_I^2}{H_I}\right), \tag{4}$$

where $G_I = \sum_{i\in I}g_i$, $H_I = \sum_{i\in I}h_i$, and n_I is the number of samples on node I.

3.2 Gain Importance

Gain importance [Breiman *et al.*, 1984; Hastie *et al.*, 2001], also known as mean decrease in impurity, is a kind of feature importance in tree-based methods. It is frequently used to explain how influential a feature is on the model's predictions. Gain importance is calculated by summing up the split gain in Eq 4 of all the splits for each feature respectively.

4 Analysis of Bias in GBDT

We analyze the bias in GBDT and demonstrate that it stems from the systematic bias in the gain estimation and the bias in the split finding algorithm. We show how this bias might lead to serious interpretability and overfitting problems in GBDT.

4.1 Bias in The Gain Estimation

$\widetilde{\text{Gain}}$ estimates the reduction in loss of a given split on a feature, which is used in both tree construction and the interpretation of a feature's importance in the tree ensemble model. Intuitively, we would like the $\widetilde{\text{Gain}}$ to be unbiased, i.e., it should be zero in expectation when randomly splitting on a feature that is independent of the target. However, $\widetilde{\text{Gain}}$ is always non-negative for any split on any feature.

Theorem 1. For a dataset (X,Y) sampled from a distribution $\mathcal{T}$, for any split θ of node I on a given feature X_j, we always have

$$\widetilde{\text{Gain}}(I,\theta) \ge 0.$$

According to the theorem, the split gain for a random split on a feature independent of the target is almost always positive (the split gain is zero in very rare cases, see the proof and more discussions in Appendix A). This implies that 1) we may split on an uninformative feature, and 2) a positive split gain does not necessarily indicate that the feature contributes to the model. One of the reasons causing this bias is that, $\left(\frac{1}{n}\sum_{i=1}^{n}g_i\right)^2$ is not an unbiased estimation of μ_g^2:

$$\mathbb{E}_{\mathcal{D}}\left[\left(\frac{1}{n_I}\sum_{i\in I}g_i\right)^2\right]$$
$$= \mathbb{E}_{\mathcal{D}}\left[\frac{1}{n_I^2}\sum_{i,j\in I, i\ne j}2g_ig_j\right] + \mathbb{E}_{\mathcal{D}}\left[\frac{1}{n_I^2}\sum_{i\in I}g_i^2\right]$$
$$= \frac{n_I-1}{n_I}\mu_g(I)^2 + \frac{1}{n_I}\left(\mu_g(I)^2 + \sigma_g(I)^2\right)$$
$$= \mu_g(I)^2 + \frac{1}{n_I}\sigma_g(I)^2.$$

For an uninformative feature that is independent of the target, any split on the feature yields $\mu_g(I) = \mu_g(I_L) = \mu_g(I_R)$ and $\sigma_g(I) = \sigma_g(I_L) = \sigma_g(I_R)$. Consider a regression problem with the MSE loss where the hessian is always a constant, according to Eq. 3 we have

$$\mathbb{E}_{\mathcal{D}}\left[\widetilde{\mathcal{L}}(I)\right] = \frac{1}{2n}\left(\mathbb{E}_{\mathcal{D}}\left[n_I\right]\mu_g(I)^2 + \sigma_g(I)^2\right),$$

(a) Gain importance of GBDT   (b) Unbiased gain of GBDT
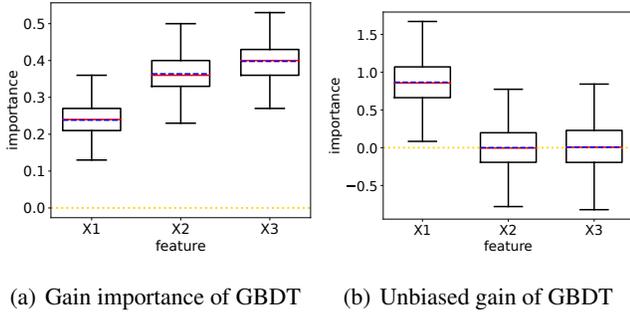
Figure 1: The features' importance under different importance measurements in the synthetic dataset. The box plot is based on 1000 repetitions. Unbiased gain correctly assigns the highest importance to X_1 and an importance of zero in expectation to X_2 and X_3.

hence the split gain on the uninformative feature is

$$\mathbb{E}_{\mathcal{D}} \left[\widetilde{\mathrm{Gain}}(I, \theta) \right]$$
$$= \mathbb{E}_{\mathcal{D}} \left[\widetilde{\mathcal{L}}(I_L) \right] + \mathbb{E}_{\mathcal{D}} \left[\widetilde{\mathcal{L}}(I_R) \right] - \mathbb{E}_{\mathcal{D}} \left[\widetilde{\mathcal{L}}(I) \right]$$
$$= \frac{1}{2n} \left(\sigma_g(I)^2 + \mathbb{E}_{\mathcal{D}} \left[n_L + n_R - n_I \right] \mu_g(I)^2 \right)$$
$$= \frac{1}{2n} \sigma_g(I)^2 \geq 0.$$

4.2 Bias in The Split Finding Algorithm

One of the main challenges in tree learning is to find the optimal split that maximize the reduction of the loss, as shown in Eq. 2. To do this, a split finding algorithm iterates over candidate splits on all features to identify the optimal split that minimizes loss on the training dataset. This strategy for identifying the optimal split introduces two problems in tree learning: 1) the split finding algorithm favors features with high cardinality (such as continuous features or categorical features with many categories). Higher cardinality features have a greater number of candidate splits, and thus a greater likelihood of being split. 2) The split finding algorithm always selects the best split on the training set, without evaluating the generalization performance of each split. The two problems together lead to the overfitting problem in GBDT. We use an example to illustrate how these two problems adversely affect tree learning.

Example 1. We generate a synthetic dataset, so that X_1 is a binary feature, X_2 is a categorical feature with 6 categories (each category has equal probability), and $X_3 \sim N(0, 1)$ is continuous. Consider a regression problem with $y = 0.1X_1 + \epsilon$ where $\epsilon \sim N(0, 1)$. We train a GBDT on the synthetic dataset and plot the gain importance of each feature in Figure 1(a). We can see that the importance of X_2 and X_3 is larger than that of X_1, even if X_2 and X_3 are independent with the target variable. This shows that GBDT overfits on the noise due to the bias in the split finding algorithm. In addition, this bias introduces interpretability issues, as X_2 and X_3 are more important than X_1 based on the gain importance.

5 Our Method

To solve the interpretability issue caused by the bias in GBDT, we propose "unbiased gain", an unbiased measurement of feature importance in Section 5.1. Then we incorporate the unbiased property into the split finding algorithm and propose UnbiasedGBM in Section 5.2 to address the issue of overfitting caused by the bias in GBDT.

5.1 Unbiased Gain

Our earlier analysis revealed that there are two sources of the bias in gain importance. First, gain importance biases towards features with high cardinality due to the split finding algorithm. Second, gain importance is always non-negative due to the biased estimation of Eq 2. Our goal is to propose an unbiased measurement of feature importance (unbiased gain). This new measurement is unbiased in a sense that an uninformative feature will receive an importance score of zero in expectation.

In order to design an unbiased measurement of feature importance, we need to eliminate two sources of bias in the current gain importance measurement mentioned above. The intuitive rationale for the first source of bias is that we should not determine the best split and assess its performance using the same set of data. Therefore, a separate validation set is considered to estimate the gain importance. However, directly computing gain importance using the validation set still suffers from the second source of bias. Therefore, we construct a new form of estimation using the validation set that meets the zero-expectation criterion.

Assume we have a training dataset $\mathcal{D} = \{(x_i, y_i)\}$ and a validation dataset $\mathcal{D}' = \{(x_i', y_i')\}$. For a given leaf node I and a given split $I = I_L \cup I_R$, there are n_I, n_L, n_R training examples and n_I', n_L', n_R' validation examples that fall into leaf node I, I_L, and I_R. First, we estimate $\mu_g(I)$ using the training examples

$$\hat{\mu}_g(I) = \frac{1}{n_I} G_I = \frac{1}{n_I} \sum_{q(x_i) = I} g_i.$$

Then, we randomly select k examples from n_I' validation examples, where $k = \min(n_L', n_R')$. Next, we estimate $\mu_g(I)$ and $\mu_h(I)$ using k randomly selected validation examples

$$\hat{\mu}_g'(I) = \frac{1}{k} G_I' = \frac{1}{k} \sum_{q(x_i') = I} g_i' \cdot \delta(I, i),$$
$$\hat{\mu}_h'(I) = \frac{1}{k} H_I' = \frac{1}{k} \sum_{q(x_i') = I} h_i' \cdot \delta(I, i),$$

where $\delta(I, i)$ is a binary indicator showing whether a validation sample has been selected. Finally we can calculate the loss of leaf node I by

$$\widetilde{\mathcal{L}}(I) = \frac{1}{2} \frac{\hat{\mu}_g(I) \cdot \hat{\mu}_g'(I)}{\hat{\mu}_h'(I)} \cdot \frac{n_I}{n} = -\frac{1}{2n} G_I \cdot \frac{G_I'}{H_I'}.$$

Here, G_I is computed using the training set while G_I' and H_I' are computed using the validation set. We can also calculate $\widetilde{\mathcal{L}}(I_L)$ and $\widetilde{\mathcal{L}}(I_R)$ in a similar way (the number of selected validation example k is the same for I, I_L, and I_R).

Finally, the unbiased gain is calculated as

$$\widetilde{\text{Gain}}_{\text{ub}}(I, \theta) = \widetilde{\mathcal{L}}(I) - \widetilde{\mathcal{L}}(I_L) - \widetilde{\mathcal{L}}(I_R). \tag{5}$$

Theorem 2. For a feature X_j, a leaf node I, and a split θ, if X_j is marginally independent of y within the region defined by the leaf node I, then

$$\mathbb{E}_{\mathcal{D}'}\left[\widetilde{\text{Gain}}_{\text{ub}}(I, \theta)\right] = 0.$$

A critical design in the unbiased gain is that, instead of estimating $\mu_g(I)$, $\mu_g(I_L)$, and $\mu_g(I_R)$ using all the validation examples on node I, I_L, and I_R, we randomly select k examples from node I, I_L, and I_R respectively for estimation. This design is critical for the unbiased property of Eq 5 (see the proof of Theorem 2 and more explanations in Appendix B)

The unbiased gain we propose serves as a post hoc method to address the interpretability issue. In Figure 1(b), we plot the unbiased gain of the GBDT trained on the synthetic data. We can see that the unbiased gain correctly assigns X_1 with the highest importance, and the importance of X_2 and X_3 is zero in expectation.

5.2 UnbiasedGBM

We propose UnbiasedGBM to address the overfitting problem introduced by the bias in GBDT: 1) The choice of each split biases towards features with high cardinality. 2) We always choose the best split on the training set, without evaluating the generalization performance of each split.

In order to eliminate these two biases, we need two validation sets. Assume we divide the training set into a sub-training set $\mathcal{D}$ and two validation sets $\mathcal{D}'_1$ and $\mathcal{D}'_2$. UnbiasedGBM eliminates the bias by redesigning the split finding algorithm. The design is conceptually simple but requires a good understanding of the bias in GBDT. First, we calculate the gain of each split $\widetilde{\text{Gain}}_1$ in the original fashion using the sub-training set $\mathcal{D}$. We determine the best split of each feature using $\widetilde{\text{Gain}}_1$ of each split. Next, we calculate the gain $\widetilde{\text{Gain}}_2$ of each feature's best split using the validation set $\mathcal{D}'_1$. We determine which feature to split using $\widetilde{\text{Gain}}_2$ of each feature's best split. Since we determine the best split of each feature and the feature to split using different data, we only need to consider the best split of each feature when choosing the feature to split, thus eliminating the bias towards features with high cardinality. Finally, we use the data set $\mathcal{D}'_2$ to calculate the unbiased gain $\widetilde{\text{Gain}}_{\text{ub}}$ of the best split. $\widetilde{\text{Gain}}_{\text{ub}}$ measures the generalization performance of the best split. We split on the leaf node if $\widetilde{\text{Gain}}_{\text{ub}} > 0$ and stop if $\widetilde{\text{Gain}}_{\text{ub}} \le 0$.

Remark. We perform early-stopping on a leaf node when the best split has $\widetilde{\text{Gain}}_{\text{ub}} \le 0$. However, this negative $\widetilde{\text{Gain}}_{\text{ub}}$ is taken into account when computing the importance of each feature in UnbiasedGBM to maintain the unbiased property.

To sum up, UnbiasedGBM enjoys two advantages over the existing GBDT: 1) UnbiasedGBM unbiasedly chooses among features with different cardinality to mitigate overfitting. 2) UnbiasedGBM measures the generalization performance of each split and performs leaf-wise early-stopping to avoid overfitting splits.

Discussion. Existing GBDT implementations can also perform leaf-wise early-stopping by using the minimal gain to split. However, this method and our method have two conceptual differences. First, we measure the generalization performance of each split, whereas existing methods only use statistics on the training set. Second, our "minimal gain to split" is zero on a theoretic basis, whereas existing methods require heuristic tuning of the minimal gain to split.

Implementation details. An important detail is how to divide the dataset into $\mathcal{D}$, $\mathcal{D}'_1$, and $\mathcal{D}'_2$. We experiment with different ratios of splitting the dataset and find out that we achieve the best performance when $|\mathcal{D}| = |\mathcal{D}'_1| = |\mathcal{D}'_2|$ (see more details in Appendix E). An intuitive explanation is that different datasets are equally important in our algorithm and should have the same number of samples.

6 Experiments

In this section, we aim at answering two questions through extensive experiments:

- **Q1.** How does UnbiasedGBM perform compared with well-developed GBDT implementations such as XGBoost, LightGBM, and CatBoost?

- **Q2.** How does the proposed unbiased gain perform in terms of feature selection compared with existing feature importance methods?

6.1 Datasets

We collect 60 classification datasets in various application domains provided by Kaggle, UCI [Dua and Graff, 2017], and OpenML [Vanschoren *et al.*, 2013] platforms. We select datasets according to the following criteria: **1) Real-world data**. We remove artificial datasets that are designed to test specific models. **2) Not high dimensional**. We remove datasets with m/n ratio above 1. **3) Not too small**. We remove datasets with too few samples (< 500). **4) Not too easy**. We remove datasets if a LightGBM with the default hyperparameters can reach a score larger than 0.95. The detailed properties of datasets are presented in Appendix C.

6.2 Q1. UnbiasedGBM

In this subsection, we answer the Q1 question by comparing UnbiasedGBM with XGBoost, LightGBM, and CatBoost using extensive experiments.

Evaluation Metrics

We use the area under the ROC curve (AUC) in the test set to measure the model performance. In order to aggregate results across datasets of different difficulty, we employ a metric similar to the distance to the minimum, which is introduced in [Wistuba *et al.*, 2015] and used in [Feurer *et al.*, 2020; Grinsztajn *et al.*, 2022]. This metric normalize each test AUC between 0 and 1 via a min-max normalization using the worst AUC and the best AUC of all the models on the dataset.

Baseline Methods

We compare with the following baseline methods:

- **XGBoost** [Chen and Guestrin, 2016].

- **LightGBM** [Ke *et al.*, 2017].

(a) 20 small-scale datasets with only numerical features.

(b) 20 small-scale datasets with num. and cat. features.

(c) 10 medium-scale datasets with only numerical features.

(d) 10 medium-scale datasets with num. and cat. features.
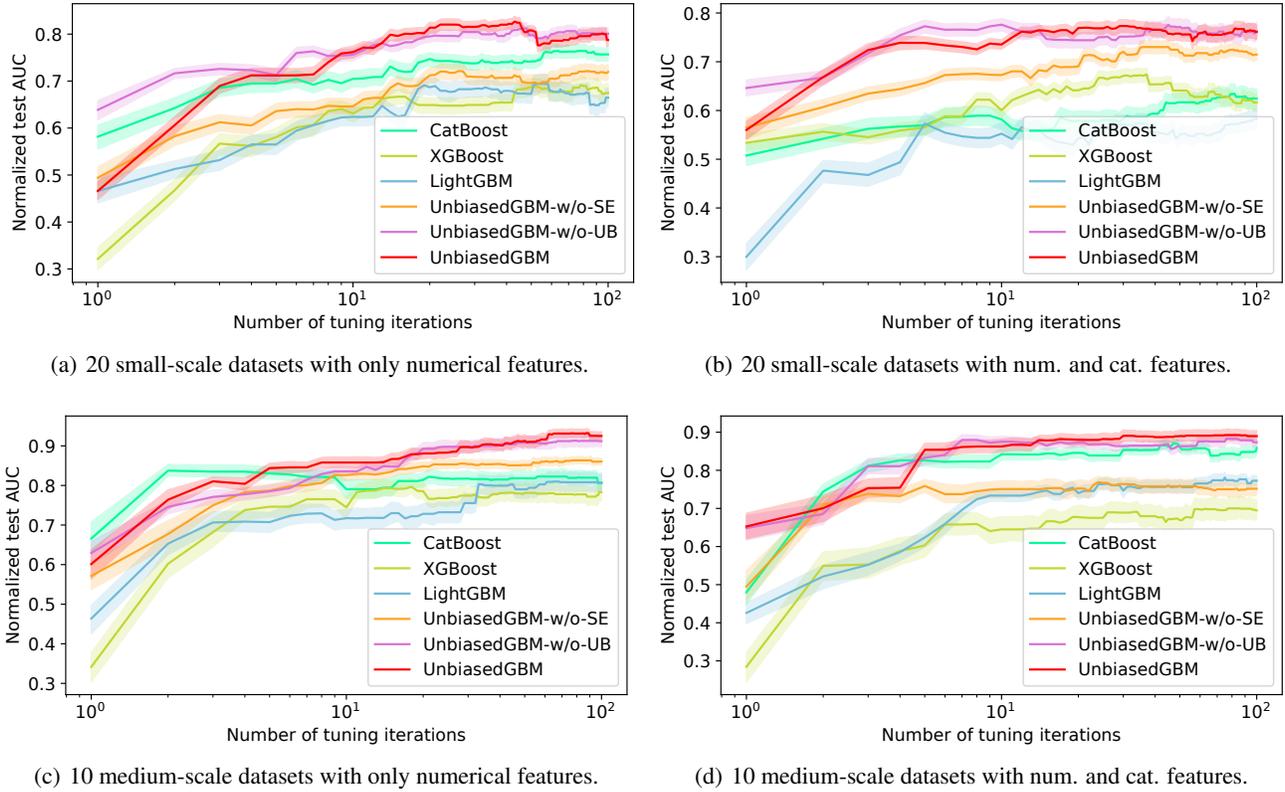
Figure 2: Comparison of UnbiasedGBM with other baseline methods on four types of datasets. Each value corresponds the normalized test AUC of the best model (on the validation set) after a specific number of tuning iterations, averaged on all the datasets. The shaded area presents the variance of the scores.

	XGBoost	LightGBM	CatBoost	UnbiasedGBM
Average Rank	3.00	2.85	2.43	1.72
p-value	$\leq 10^{-3}$	$\leq 10^{-3}$	0.013	-

Table 1: The average rank (where lower is better) over 60 datasets and the p-value of Nemenyi test between UnbiasedGBM and the baseline methods.

- **CatBoost** [Dorogush *et al.*, 2018].

- **UnbiasedGBM-w/o-SE**. UnbiasedGBM without separating the determination of the best split of each feature and the feature to split. One of two validation sets is merged with the subtraining set.

- **UnbiasedGBM-w/o-UB**. UnbiasedGBM without computing the unbiased gain $\widetilde{\text{Gain}}_{ub}$ to measure the generalization performance of the best split. Two validation sets are merged into one to determine the best feature to split and perform early stopping.

For each method, we perform hyperparameter optimization using the popular Optuna [Akiba *et al.*, 2019] Python package. See more details in Appendix D.

Results

In order to better present the advantage of UnbiasedGBM on datasets with different properties, we classify 60 datasets into

four types of datasets, including small-scale (datasets with less than 4000 samples) or medium-scale datasets with only numerical features or both numerical and categorical features. We present the results in Figure 2. The x-axis is the number of tuning iterations, visualizing the influence of the tuning budget on the model performance. We can see that UnbiasedGBM significantly outperforms XGBoost, LightGBM, and CatBoost in both small and medium datasets. In addition, UnbiasedGBM is effective even if there is only numerical features in the datasets. Categorical features are not the only source of performance improvement in UnbiasedGBM. We also visualize the comparison of each dataset in Figure 3 that demonstrates the improvement of our method. We leverage the Nemenyi test [Demsar, 2006] to perform statistical analyses using the rank of each method after hyper-parameter tuning of 100 iterations on 60 datasets. We present the results in Table 1, where the Nemenyi test p-values show that UnbiasedGBM significantly outperforms the baselines. Moreover, comparisons between UnbiasedGBM, UnbiasedGBM-w/o-SE, and UnbiasedGBM-w/o-UB demonstrate that separating the determination of the best split of each feature and the feature to split is the primary source of improvement in UnbiasedGBM. In most cases, computing the unbiased gain to evaluate generalization performance of the best split can also result in performance improvement.

(a) UnbiasedGBM vs. LightGBM

(b) UnbiasedGBM vs. XGBoost

(c) UnbiasedGBM vs. CatBoost
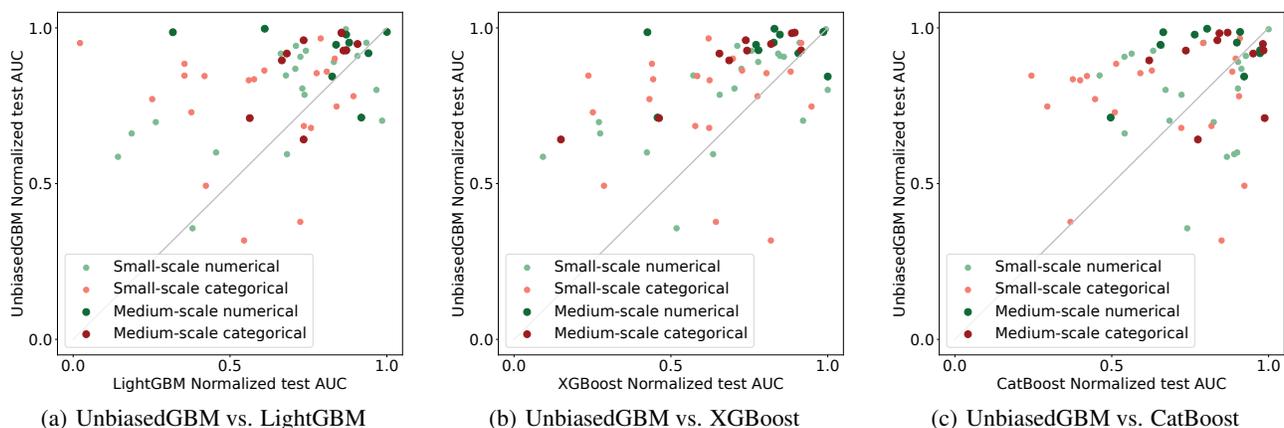
Figure 3: Comparison of UnbiasedGBM with LightGBM, XGBoost and CatBoost. Each dot denotes a dataset. The normalized test AUC is higher the better. "numerical" means the dataset only contains numerical features. "categorical" means the dataset contains both numerical and categorical features.
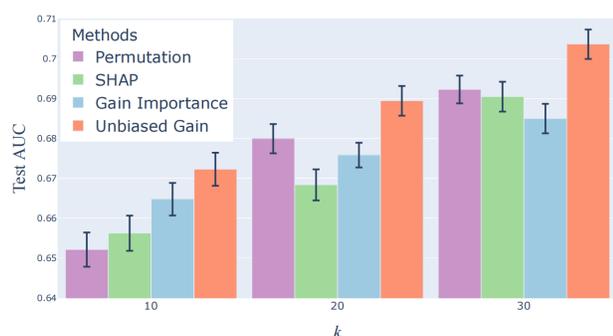
Figure 4: Comparison of different feature importance methods in feature selection. We report the AUC on the test set of the model using top $k\%$ selected features according to the feature importance.

6.3 Q2. Unbiased Gain

In this subsection, we demonstrate the performance of unbiased gain in feature selection.

Baseline Methods

We compare unbiased gain with different feature importance measurements:

- Gain importance [Breiman *et al.*, 1984].
- Permutation feature importance (PFI) [Breiman, 2001].
- SHAP [Lundberg *et al.*, 2018].

Evaluation

We follow the standard approach [Forman and others, 2003] to evaluate different feature importance measurements in feature selection. For a given dataset, we first estimate the feature importance on the training set. Then, we select top $k\%$ features according to the feature importance, where $k \in \{10, 20, 30\}$. Next, we build a GBDT model according to the selected feature subset. Finally, we calculate the AUC of the model on the test set. Higher AUC indicates that the feature importance performs better in feature selection.

	LightGBM	UnbiasedGBM
Full feature set	$0.779_{\pm 0.000}$	$0.809_{\pm 0.003}$
Remove "nHM" with 11 categories	$0.772_{\pm 0.000}$	$0.797_{\pm 0.003}$
Remove "PCD" with 224 categories	$0.787_{\pm 0.000}$	$0.811_{\pm 0.002}$

Table 2: An example of the QSAR Bioconcentration dataset. Light-GBM overfits on the "PCD" feature with many categories, because removing the feature brings significant improvement in the test AUC. UnbiasedGBM addresses the overfitting issue, because it has better test AUC than LightGBM when using the full feature set, and removing the "PCD" feature brings an insignificant difference.

Results

We evaluate these methods on 14 out of 60 datasets with more than 30 features. Datasets with too few features may not need feature selection. We consider selecting top $k\%$ features for $k \in \{10, 20, 30\}$. For each method, we report the mean and variance of the test AUC across these 14 datasets. The results are presented in Figure 4. We can see that unbiased gain achieves better average performance than baseline methods in feature selection.

6.4 Analyses of Features with Many Categories

We present an analysis of the QSAR Bioconcentration dataset in Table 2 to show that UnbiasedGBM can address the overfitting issue of LightGBM on categorical features with many categories. The details are in the caption of Table 2.

7 Conclusion

In this paper, we investigate the bias in GBDT and the consequent interpretability and overfitting issues. We give a fine-grained analysis of bias in GBDT. Based on the analysis, we propose the unbiased gain and UnbiasedGBM to address the interpretability and overfitting issues. Extensive experiments on 60 datasets show that UnbiasedGBM has better average performance than XGBoost, LightGBM, and Catboost and unbiased gain can outperform popular feature importance estimation methods in feature selection.

Acknowledgements

Contribution Statement

This paper is the result of collaborative work between Zheyu Zhang and Tianping Zhang, who contributed equally to the conception, implementation, experimentation, and paper writing. Jian Li served as the corresponding author, contributing to the overall idea of the project as well as providing computing resources.

References

[Akiba *et al.*, 2019] 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*, pages 2623–2631, 2019.

[Boulesteix *et al.*, 2012] Anne-Laure Boulesteix, Andreas Bender, Justo Lorenzo Bermejo, and Carolin Strobl. Random forest gini importance favours snps with large minor allele frequency: impact, sources and recommendations. *Briefings in Bioinformatics*, 13(3):292–304, 2012.

[Breiman *et al.*, 1984] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[Breiman, 2001] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[Demsar, 2006] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[Dorogush *et al.*, 2018] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[Dua and Graff, 2017] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[Feurer *et al.*, 2020] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. *arXiv preprint arXiv:2007.04074*, 2020.

[Forman and others, 2003] George Forman et al. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3(Mar):1289–1305, 2003.

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

[Gorishniy *et al.*, 2021] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[Grinsztajn *et al.*, 2022] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.

[Hastie *et al.*, 2001] Trevor Hastie, Jerome H. Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

[Ke *et al.*, 2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[Kim and Loh, 2001] Hyunjoong Kim and Wei-Yin Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96(454):589–604, 2001.

[Li *et al.*, 2019] Xiao Li, Yu Wang, Sumanta Basu, Karl Kumbier, and Bin Yu. A debiased mdi feature importance measure for random forests. *Advances in Neural Information Processing Systems*, 32, 2019.

[Li, 2012] Ping Li. Robust logitboost and adaptive base class (abc) logitboost. *arXiv preprint arXiv:1203.3491*, 2012.

[Loh and Shih, 1997] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica sinica*, pages 815–840, 1997.

[Loh, 2009] Wei-Yin Loh. Improving the precision of classification trees. *The Annals of Applied Statistics*, pages 1710–1737, 2009.

[Lundberg *et al.*, 2018] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[Nembrini *et al.*, 2018] Stefano Nembrini, Inke R König, and Marvin N Wright. The revival of the gini importance? *Bioinformatics*, 34(21):3711–3718, 2018.

[Nicodemus, 2011] Kristin K Nicodemus. On the stability and ranking of predictors from random forest variable importance measures. *Briefings in bioinformatics*, 12(4):369–373, 2011.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.

[Sandri and Zuccolotto, 2008] Marco Sandri and Paola Zuccolotto. A bias correction algorithm for the gini variable importance measure in classification trees. *Journal of Computational and Graphical Statistics*, 17(3):611–628, 2008.

[Shwartz-Ziv and Armon, 2022] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[Strobl *et al.*, 2007] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8(1):1–21, 2007.

[Vanschoren *et al.*, 2013] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. Openml: networked science in machine learning. *SIGKDD Explor.*, 15(2):49–60, 2013.

[Wistuba *et al.*, 2015] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

[Zhou and Hooker, 2021] Zhengze Zhou and Giles Hooker. Unbiased measurement of feature importance in tree-based methods. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–21, 2021.