# Unbiased Measurement of Feature Importance in Tree-Based Methods

ZHENGZE ZHOU and GILES HOOKER, Cornell University

We propose a modification that corrects for split-improvement variable importance measures in Random Forests and other tree-based methods. These methods have been shown to be biased towards increasing the importance of features with more potential splits. We show that by appropriately incorporating split-improvement as measured on out of sample data, this bias can be corrected yielding better summaries and screening tools.

CCS Concepts: • **Computing methodologies** → **Classification and regression trees**; *Bagging*; *Feature selection*;

Additional Key Words and Phrases: Tree-based methods, feature importance, unbiasedness

**ACM Reference format:**
Zhengze Zhou and Giles Hooker. 2020. Unbiased Measurement of Feature Importance in Tree-Based Methods. *ACM Trans. Knowl. Discov. Data* 15, 2, Article 26 (December 2020), 21 pages.
https://doi.org/10.1145/3429445

## 1 INTRODUCTION

This article examines split-improvement feature importance scores for tree-based methods. Starting with Classification and Regression Trees (CART) [6] and C4.5 [34], decision trees have been a workhorse of general machine learning, particularly within ensemble methods such as Random Forests (RF) [5] and Gradient Boosting Trees [12]. They enjoy the benefits of computational speed, few tuning parameters, and natural ways of handling missing values. Recent statistical theory for ensemble methods [e.g., 9, 27, 36, 40, 43] has provided theoretical guarantees and allowed formal statistical inference. Variants of these models have also been proposed such as Bernoulli RF [41, 42] and Random Survival Forests [20]. For all these reasons, tree-based methods have seen broad applications including in protein interaction models [29] in product suggestions on Amazon [37] and in financial risk management [21].

However, in common with other machine learning models, large ensembles of trees act as "black boxes," providing predictions but little insight as to how they were arrived at. There has thus been considerable interest in providing tools either to explain the broad patterns that are modeled by these methods, or to provide justifications for particular predictions. This article examines variable



This work was supported in part by the NSF grants DMS-1712554, DEB-1353039, and TRIPODS 1740882.
Authors' addresses: Z. Zhou, Cornell University, 301 Malott Hall, Ithaca, New York, 14853; email: zz433@cornell.edu; G. Hooker, Cornell University, 1186 Comstock Hall, Ithaca, New York, 14853; email: gjh27@cornell.edu.

1556-4681/2020/12-ART26 $15.00
https://doi.org/10.1145/3429445

or feature[1] importance scores that provide global summaries of how influential a particular input dimension is in the models' predictions. These have been among the earliest diagnostic tools for machine learning and have been put to practical use as screening tools, see for example Díaz-Uriarte and De Andres [10] and Menze et al. [28]. Thus, it is crucial that these feature importance measures reliably produce well-understood summaries.

Feature importance scores for tree-based models can be broadly split into two categories. Permutation methods rely on measuring the change in value or accuracy when the values of one feature are replaced by uninformative noise, often generated by a permutation. These have the advantage of being applicable to any function, but have been critiqued by Hooker [15], Hooker and Mentch [16], Strobl et al. [38] for forcing the model to extrapolate. By contrast, in this article, we study the alternative split-improvement scores (also known as Gini importance, or mean decrease impurity) that are specific to tree-based methods. These naturally aggregate the improvement associated with each note split and can be readily recorded within the tree building process [6, 12]. In Python, split-improvement is the default implementation for almost every tree-based model, including *RandomForestClassifier*, *RandomForestRegressor*, *GradientBoostingClassifier*, and *GradientBoostingRegressor* from *scikit-learn* [32].

Despite their common use, split-improvement measures are biased towards features that exhibit more potential splits and in particular towards continuous features or features with large numbers of categories. This weakness was already noticed in Breiman et al. [6] and Strobl et al. [39] conducted thorough experiments followed by more discussions in Boulesteix et al. [3] and Nicodemus [31].[2] While this may not be concerning when all covariates are similarly configured, in practice it is common to have a combination of categorical and continuous variables in which emphasizing more complex features may mislead any subsequent analysis. For example, gender will be a very important binary predictor in applications related to medical treatment; whether the user is a paid subscriber is also central to some tasks such as in Amazon and Netflix. But each of these may be rated as less relevant to age which is a more complex feature in either case. In the task of ranking single nucleotide polymorphisms with respect to their ability to predict a target phenotype, researchers may overlook rare variants as common ones are systematically favored by the split-improvement measurement. [3].

We offer an intuitive rationale for this phenomenon and design a simple fix to solve the bias problem. The observed bias is similar to overfitting in training machine learning models, where we should not build the model and evaluate relevant performance using the same set of data. To fix this, split-improvement calculated from a separate test set is taken into consideration. We further demonstrate that this new measurement is unbiased in the sense that features with no predictive power for the target variable will receive an importance score of zero in expectation. These measures can be very readily implemented in tree-based software packages. We believe the proposed measurement provides a more sensible means for evaluating feature importance in practice.

In the following, we introduce some background and notation for tree-based methods in Section 2. In Section 3, split-improvement is described in detail and its bias and limitations are presented. The proposed unbiased measurement is introduced in Section 4. Section 5 applies our idea to a simulated example and three real-world datasets. We conclude with some discussions and future directions in Section 6. Proofs and some additional simulation results are collected in Appendix A and B, respectively.

[1]We use "feature," "variable," and "covariate" interchangeably here to indicate individual measurements that act as inputs to a machine learning model from which a prediction is made.
[2]See https://explained.ai/rf-importance/ for a popular demonstration of this.

## 2 TREE-BASED METHODS

In this section, we provide a brief introduction and mathematical formulation of tree-based models that will also serve to introduce our notation. We refer readers to relevant chapters in Friedman et al. [11] for a more detailed presentation.

### 2.1 Tree Building Process

Decision trees are a non-parametric machine learning tool for constructing prediction models from data. They are obtained by recursively partitioning feature space by axis-aligned splits and fitting a simple prediction function, usually constant, within each partition. The result of this partitioning procedure is represented as a binary tree. Popular tree building algorithms, such as CART and C4.5, may differ in how they choose splits or deal with categorical features. Our introduction in this section mainly reflects how decision trees are implemented in *scikit-learn*.

Suppose our data consists of $p$ inputs and a response, denoted by $z_i = (x_i, y_i)$ for $i = 1, 2, \ldots, n$, with $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$. For simplicity we assume our inputs are continuous.[3] Labels can be either continuous (regression trees) or categorical (classification trees). Let the data at a node $m$ represented by $Q$. Consider a splitting variable $j$ and a splitting point $s$, which results in two child nodes:

$$Q_l = \{(x, y) | x_j \leq s\}$$

$$Q_r = \{(x, y) | x_j > s\}.$$

The impurity at node $m$ is computed by a function $H$, which acts as a measure for goodness-of-fit and is invariant to sample size. Our loss function for split $\theta = (j, s)$ is defined as the weighted average of the impurity at two child nodes:

$$L(Q, \theta) = \frac{n_l}{n_m} H(Q_l) + \frac{n_r}{n_m} H(Q_r),$$

where $n_m, n_l, n_r$ are the number of training examples falling into node $m, l, r$, respectively. The best split is chosen by minimizing the above loss function:

$$\theta^* = \arg\min_{\theta} L(Q, \theta). \tag{1}$$

The tree is built by recursively splitting child nodes until some stopping criterion is met. For example, we may want to limit tree depth, or keep the number of training samples above some threshold within each node.

For regression trees, $H$ is usually chosen to be mean squared error, using average values as predictions within each node. At node $m$ with $n_m$ observations, $H(m)$ is defined as:

$$\bar{y}_m = \frac{1}{n_m} \sum_{x_i \in m} y_i,$$

$$H(m) = \frac{1}{n_m} \sum_{x_i \in m} (y_i - \bar{y}_m)^2.$$

Mean absolute error can also be used depending on specific application.

[3] Libraries in different programming languages differ on how to handle categorical inputs. *rpart* and *randomForest* libraries in R search over every possible subsets when dealing with categorical features. However, tree-based models in *scikit-learn* do not support categorical inputs directly. Manually transformation is required to convert categorical features to integer-valued ones, such as using dummy variables, or treated as ordinal when applicable.

In classification, there are several different choices for the impurity function $H$. Suppose for node $m$, the target $y$ can take values of $1, 2, \ldots, K$, define

$$p_{mk} = \frac{1}{n_m} \sum_{x_i \in m} \mathbb{1}(y_i = k)$$

to be the proportion of class $k$ in node $m$, for $k = 1, 2, \ldots, K$. Common choices are:

(1) Misclassification error:

$$H(m) = 1 - \max_{1 \le k \le K} p_{mk}.$$

(2) Gini index:

$$H(m) = \sum_{k \neq k'} p_{mk} p_{mk'} = 1 - \sum_{k=1}^{K} p_{mk}^2.$$

(3) Cross-entropy or deviance:

$$H(m) = -\sum_{k=1}^{K} p_{mk} \log p_{mk}.$$

This article will focus on mean squared error for regression and Gini index for classification.

### 2.2 RF and Gradient Boosting Trees

Though intuitive and interpretable, there are two major drawbacks associated with a single decision tree: they suffer from high variance and in some situations, they are too simple to capture complex signals in the data. Bagging [4] and boosting [12] are two popular techniques used to improve the performance of decision trees.

Suppose we use a decision tree as a base learner $t(x; z_1, z_2, \ldots, z_n)$, where $x$ is the input for prediction and $z_1, z_2, \ldots, z_n$ are training examples as before. Bagging aims to stabilize the base learner $t$ by resampling the training data. In particular, the bagged estimator can be expressed as:

$$\hat{t}(x) = \frac{1}{B} \sum_{b=1}^{B} t\left(x; z_{b1}^*, z_{b2}^*, \ldots, z_{bn}^*\right)$$

where $z_{bi}^*$ are drawn independently with replacement from the original data (bootstrap sample), and $B$ is the total number of base learners. Each tree is constructed using a different bootstrap sample from the original data. Thus approximately one-third of the cases are left out and not used in the construction of each base learner. We call these *out-of-bag* samples.

RF [5] are a popular extension of bagging with an additional randomness injected. At each step when searching for the best split, only $p_0$ features are randomly selected from all $p$ possible features and the best split $\theta^*$ must be chosen from this subset. When $p_0 = p$, this reduces to bagging. Mathematically, the prediction is written as

$$\hat{t}^{RF}(x) = \frac{1}{B} \sum_{b=1}^{B} t\left(x; \xi_b, z_{b1}^*, z_{b2}^*, \ldots, z_{bn}^*\right)$$

with $\xi_b \overset{\text{iid}}{\sim} \Xi$ denoting the additional randomness for selecting from a random subset of available features.

Boosting is another widely used technique by data scientists to achieve state-of-the-art results on many machine learning challenges [7]. Instead of building trees in parallel as in bagging, it does this sequentially, allowing the current base learner to correct for any previous bias. In Ghosal and Hooker [13], the authors also consider boosting RF to reduce bias. We will skip over some

technical details on boosting and restrict our discussion of feature importance in the context of decision trees and RF. Note that as long as tree-based models combine base learners in an additive fashion, their feature importance measures are naturally calculated by (weighted) average across those of individual trees.

## 3 MEASUREMENT OF FEATURE IMPORTANCE

Almost every feature importance measures used in tree-based models belong to two classes: split-improvement or permutation importance. Though our focus will be on split-improvement, permutation importance is introduced first for completeness.

### 3.1 Permutation Importance

Arguably permutation might be the most popular method for assessing feature importance in the machine learning community. Intuitively, if we break the link between a variable $X_j$ and $y$, the prediction error increases then variable $j$ can be considered as important.

Formally, we view the training set as a matrix $X$ of size $n \times p$, where each row $x_i$ is one observation. Let $X^{\pi,j}$ be a matrix achieved by permuting the $j^{th}$ column according to some mechanism $\pi$. If we use $l(y_i, f(x_i))$ as the loss incurred when predicting $f(x_i)$ for $y_i$, then the importance of $j^{th}$ feature is defined as:

$$\mathrm{VI}_j^{\pi} = \sum_{i=1}^{n} l\left(y_i, f\left(x_i^{\pi,j}\right) - l(y_i, f(x_i))\right) \tag{2}$$

the increase in prediction error when the $j^{th}$ feature is permuted. Variations include choosing different permutation mechanism $\pi$ or evaluating Equation (2) on a separate test set. In RF, Breiman [5] suggest to only permute the values of the $j^{th}$ variable in the *out-of-bag* samples for each tree, and final importance for the forest is given by averaging across all trees.

There is a small literature analyzing permutation importance in the context of RF. Ishwaran [19] studied paired importance. Hooker [15], Hooker and Mentch [16], Strobl et al. [38] advocated against permuting features by arguing it emphasizes behavior in regions where there is very little data. More recently, Gregorutti et al. [14] conducted a theoretical analysis of permutation importance measure for an additive regression model.

### 3.2 Split-Improvement

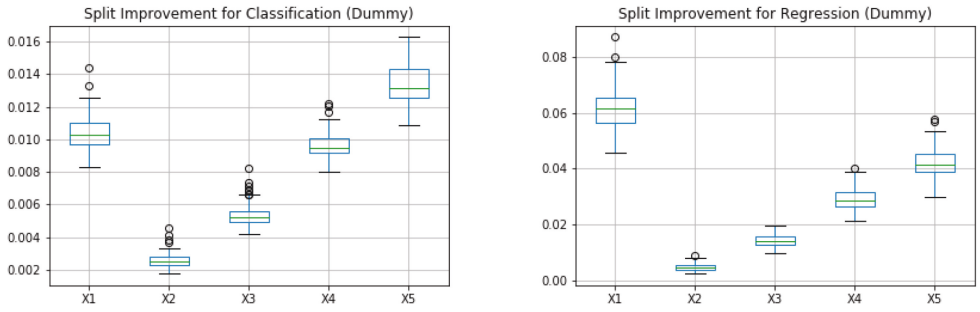
While permutation importance measures can generically be applied to any prediction function, split-improvement is unique to tree-based methods, and can be calculated directly from the training process. Every time a node is split on variable $j$, the combined impurity for the two descendent nodes is less than the parent node. Adding up the weighted impurity decreases for each split in a tree and averaging over all trees in the forest yields an importance score for each feature.

Following our notation in Section 2.1, the impurity function $H$ is either mean squared error for regression or Gini index for classification. The best split at node $m$ is given by $\theta_m^*$ which splits at $j^{th}$ variable and results in two child nodes denoted as $l$ and $r$. Then the decrease in impurity for split $\theta^*$ is defined as:

$$\Delta(\theta_m^*) = \omega_m H(m) - (\omega_l H(l) + \omega_r H(r)), \tag{3}$$

where $\omega$ is the proportion of observations falling into each node, i.e., $\omega_m = \frac{n_m}{n}$, $\omega_l = \frac{n_l}{n}$ and $\omega_r = \frac{n_r}{n}$. Then, to get the importance for $j^{th}$ feature in a single tree, we add up all $\Delta(\theta_m^*)$ where the split is at the $j^{th}$ variable:

$$\mathrm{VI}_j^{\mathrm{T}} = \sum_{m, j \in \theta_m^*} \Delta(\theta_m^*). \tag{4}$$




(a) Classification (b) Regression

Fig. 1. Split-improvement measures on five predictors. Box plot is based on 100 repetitions. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5.

Here the sum is taken over all non-terminal nodes of the tree, and we use the notation $j \in \theta_m^*$ to denote that the split is based on the $j^{th}$ feature.

The notion of split-improvement for decision trees can be easily extended to RF by taking the average across all trees. Suppose there are $B$ base learners in the forest, we could naturally define

$$\mathrm{VI}_j^{\mathrm{RF}} = \frac{1}{B}\sum_{b=1}^{B} \mathrm{VI}_j^{\mathrm{T(b)}} = \frac{1}{B}\sum_{b=1}^{B}\sum_{m, j\in\theta_m^*} \Delta_b(\theta_m^*). \tag{5}$$

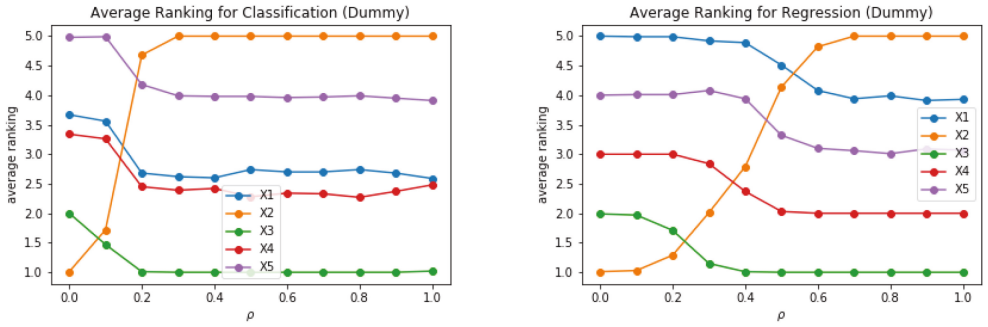
### 3.3 Bias in Split-Improvement

Strobl et al. [39] pointed out that the split-improvement measure defined above is biased towards increasing the importance of continuous features or categorical features with many categories. This is because of the increased flexibility afforded by a larger number of potential split points. We conducted a similar simulation to further demonstrate this phenomenon. All our experiments are based on RF which gives more stable results than a single tree.

We generate a simulated dataset so that $X_1 \sim N(0, 1)$ is continuous, and $X_2, X_3, X_4, X_5$ are categorically distributed with 2, 4, 10, 20 categories, respectively. The probabilities are equal across categories within each feature. In particular, $X_2$ is Bernoulli distribution with $p = 0.5$. In classification setting, the response $y$ is also generated as a Bernoulli distribution with $p = 0.5$, but independent of all the $X$'s. For regression, $y$ is independently generated as $N(0, 1)$. We repeat the simulation 100 times, each time generating $n = 1{,}000$ data points and fitting an RF model[4] using the dataset. Here categorical features are encoded into dummy variables, and we sum up importance scores for corresponding dummy variables as final measurement for a specific categorical feature. In Appendix B, we also provide simulation results when treating those categorical features as (ordered) discrete variables.

Box plots are shown in Figure 1(a) and 1(b) for classification and regression, respectively. The continuous feature $X_1$ is frequently given the largest importance score in regression setting, and among the four categorical features, those with more categories receive larger importance scores. Similar phenomenon is observed in classification as well, while $X_5$ appears to be artificially more important than $X_1$. Also note that all five features get positive importance scores, though we know that they have no predictive power for the target value $y$.

[4]Our experiments are implemented using *scikit-learn*. Unless otherwise noted, default parameters are used.




(a) Classification (b) Regression

Fig. 2. Average feature importance ranking across different signal strengths over 100 repetitions. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5.

We now explore how strong a signal is needed in order for the split-improvement measures to discover important predictors. We generate $X_1, X_2, \ldots, X_5$ as before, but in regression settings set $y = \rho X_2 + \epsilon$ where $\epsilon \sim N(0, 1)$. We choose $\rho$ to range from 0 to 1 at step size 0.1 to encode different levels of signal. For classification experiments, we first make $y = X_2$ and then flip each element of $y$ according to $P(U > \frac{1+\rho}{2})$ where $U$ is Uniform $[0, 1]$. This way, the correlation between $X_2$ and $y$ will be approximately $\rho$. We report the average ranking of all five variables across 100 repetitions for each $\rho$. The results are shown in Figure 2.

We see that $\rho$ needs to be larger than 0.2 to actually find $X_2$ is the most important predictor in our classification setting, while in regression this value increases to 0.6. And we also observe that a clear order exists for the remaining (all unimportant) four features.

This bias phenomenon could make many statistical analyses based on split-improvement invalid. For example, gender is a very common and powerful binary predictor in many applications, but feature screening based on split-improvement might think it is not important compared to age. In the next section, we explain intuitively why this bias is observed, and provide a simple but effective adjustment.

### 3.4 Related Work

Before presenting our algorithm, we review some related work aiming at correcting the bias in split-improvement. Most of the methods fall into two major categories: they either propose new tree building algorithms by redesigning split selection rules, or perform as a post hoc approach to debias importance measurement.

There has been a line of work on designing trees which do not have such bias as observed in classical algorithms such as CART and C4.5. For example, Quick, Unbiased, and Efficient Statistical Tree (QUEST) [26] removed the bias by using F-tests on ordered variables and contingency table chi-squared tests on categorical variables. Based on QUEST, CRUISE [22] and GUIDE [24] were developed. We refer readers to Loh [25] for a detailed discussion in this aspect. In Strobl et al. [39], the authors resorted to a different algorithm called cforest [17], which was based on a conditional inference framework [18]. They also implemented a stopping criteria based on multiple test procedures.

Sandri and Zuccolotto [35] expressed split-improvement as two components: a heterogeneity reduction and a positive bias. Then the original dataset $(\mathbf{X}, Y)$ is augmented with pseudo data $\mathbf{Z}$ which is uninformative but shares the structure of $\mathbf{X}$ [this idea of generating pseudo data is later formulated in a general framework termed "knockoffs"; 1]. The positive bias term is estimated by

utilizing the pseudo variables $\mathbf{Z}$ and subtracted to get a debiased estimate. Nembrini et al. [30] later modified this approach to shorten computation time and provided empirical importance testing procedures. Most recently, Li et al. [23] derived a tight non-asymptotic bound on the expected bias of noisy features and provided a new debiased importance measure. However, this approach only alleviates the issue and still yields biased results.

Our approach works as a post-hoc analysis, where the importance scores are calculated after a model is built. Compared to previous methods, it enjoys several advantages:

- —It can be easily incorporated into any existing framework for tree-based methods, such as Python or R.
- —It does not require generating additional pseudo data or computational repetitions as in Nembrini et al. [30], Sandri and Zuccolotto [35].
- —Compared to Li et al. [23] which does not have a theoretical guarantee, our method is proved to be unbiased for noisy features.

## 4 UNBIASED SPLIT-IMPROVEMENT

When it comes to evaluating the performance of machine learning models, we generally use a separate test set to calculate generalization accuracy. The training error is usually smaller than the test error as the algorithm is likely to "overfit" on the training data. This is exactly why we observe the bias with split-improvement. Each split will favor continuous features or those features with more categories, as they will have more flexibility to fit the training data. The vanilla version of split-improvement is just like using train error for evaluating model performance.

Below we propose methods to remedy this bias phenomenon by utilizing a separate test set, and prove that for features with no predictive power, we're able to get an importance score of 0 in expectation for both classification and regressions settings. Our method is entirely based on the original framework of RF, requires barely no additional computational efforts, and can be easily integrated into any existing software libraries.

The main ingredient of the proposed method is to calculate the impurity function $H$ using additional information provided from test data. In the context of RF, we can simply take *out-of-bag* samples for each individual tree. Our experiments below are based on this strategy. In the context of the honest trees proposed in Wager and Athey [40] that divide samples into a partition used to determine tree structures and a partition used to obtain leaf values, the latter could be used as our test data below. In boosting, it is common not to sample, but to keep a test set separate to determine a stopping time. Since the choice of impurity function $H$ is different for classification and regression, in what follows we will treat them separately.

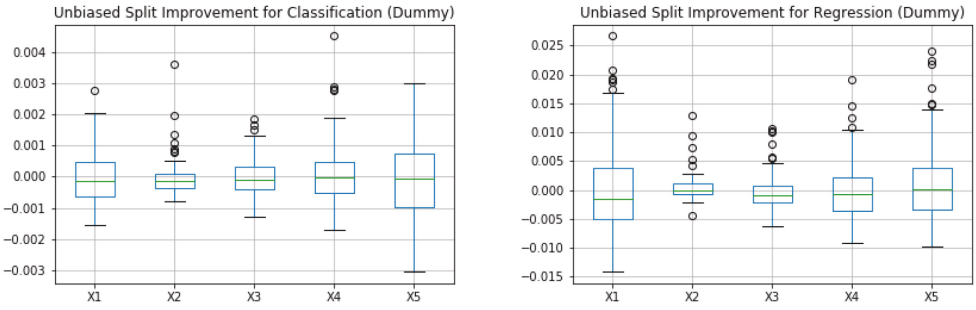
Figures 3 and 4 show the results on previous classification and regression tasks when our unbiased method is applied.[5] Feature scores for all variables are spread around 0, though continuous features and categorical features with more categories tend to exhibit more variability. In the case, where there is correlation between $X_2$ and $y$, even for the smallest $\rho = 0.1$, we can still find the most informative predictor, whereas there are no clear order for the remaining noise features.

### 4.1 Classification

Consider a root node $m$ and two child nodes, denoted by $l$ and $r$, respectively. The best split $\theta_m^* = (j, s)$ was chosen by Formula (1) and Gini index is used as impurity function $H$.

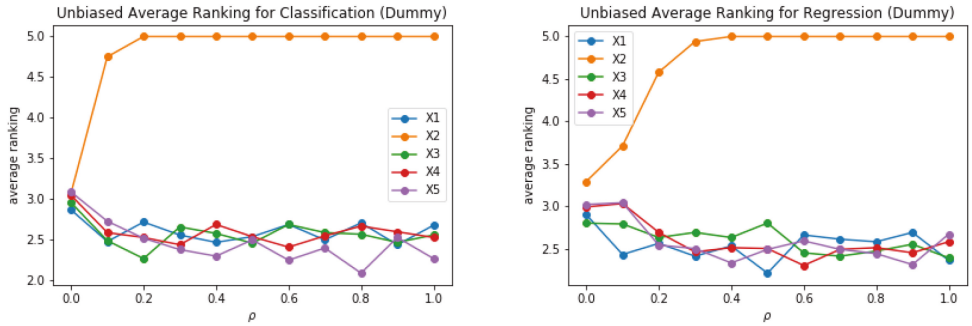
For simplicity, we focus on binary classification. Let $p$ denote class proportion within each node. For example, $p_{r,2}$ denotes the proportion of class 2 in the right child node. Hence the Gini index

[5]Relevant codes can be found at https://github.com/ZhengzeZhou/unbiased-feature-importance.




(a) Classification (b) Regression

Fig. 3. Unbiased split-improvement. Box plot is based on 100 repetitions. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5. Each tree is trained using bootstrap samples and *out-of-bag* samples are used as test set.




(a) Classification (b) Regression

Fig. 4. Unbiased feature importance ranking across different signal strengths averaged over 100 repetitions. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5. Each tree is trained using bootstrap samples and *out-of-bag* samples are used as test set.

for each node can be written as:

$$H(m) = 1 - p_{m,1}^2 - p_{m,2}^2,$$

$$H(l) = 1 - p_{l,1}^2 - p_{l,2}^2,$$

$$H(r) = 1 - p_{r,1}^2 - p_{r,2}^2.$$

The split-improvement for a split at $j^{th}$ feature when evaluated using only the training data is written as in Equation (3). This value is always positive no matter which feature is chosen and where the split is, which is exactly why a selection bias will lead to overestimate of feature importance.

If instead, we have a separate test set available, the predictive impurity function for each node is modified to be:

$$\begin{aligned} H'(m) &= 1 - p_{m,1}p'_{m,1} - p_{m,2}p'_{m,2}, \\ H'(l) &= 1 - p_{l,1}p'_{l,1} - p_{l,2}p'_{l,2}, \\ H'(r) &= 1 - p_{r,1}p'_{r,1} - p_{r,2}p'_{r,2}, \end{aligned} \tag{6}$$

where $p'$ is class proportion evaluating on the test data. And similarly,

$$\begin{aligned}\Delta'(\theta_m^*) &= \omega_m H'(m) - (\omega_l H'(l) + \omega_r H'(r)) \\ &= \omega_l (H'(m) - H'(l)) + \omega_r (H'(m) - H'(r)).\end{aligned} \tag{7}$$

Using these definitions, we first demonstrate that an individual split is unbiassed in the sense that if $y$ has no bivariate relationship with $X_j$, $\Delta'(\theta_m^*)$ will have expectation 0.

Lemma 4.1. *In classification settings, for a given feature $X_j$, if $y$ is marginally independent of $X_j$ within the region defined by node $m$, then*

$$E\Delta'(\theta_m^*) = 0$$

*when splitting at the $j^{th}$ feature.*

Proof. See Appendix A. □

The Gini index can be interpreted in an interesting way [11]. Instead of classifying observations to the majority class in each node, we could classify them to class $k$ with probability $p_{m,k}$. Then the training error rate of this rule in the node is exactly $1 - p_{m,1}^2 - p_{m,2}^2$. For the predictive impurity given in Equation (6), we can naturally interpret it as the test error rate of the rule.

Similar to Equation (4), split-improvement of $x_j$ in a decision tree is defined as:

$$\mathrm{VI}_j^{\mathrm{T,C}} = \sum_{m, j \in \theta_m^*} \Delta'(\theta_m^*). \tag{8}$$

We can now apply Lemma 4.1 to provide a global result so long as $X_j$ is always irrelevant to $y$.

Theorem 1. *In classification settings, for a given feature $X_j$, if $y$ is independent of $X_j$ in every hyper-rectangle subset of the feature space, then we always have*

$$EVI_j^{T,C} = 0.$$

Proof. The result follows directly from Lemma 4.1 and Equation (8). □

This unbiasedness result can be easily extended to the case of RF by Equation (5), as it's an average across base learners. We note here that our independence condition is designed to account for relationships that appear before accounting for splits on other variables, possibly due to relationships between $X_j$ and other features, and afterwards. It is trivially implied by the independence of $X_j$ with both $y$ and the other features. Our condition may also be stronger than necessary, depending on the tree-building process. We may be able to restrict the set of hyper-rectangles to be examined, but only by analyzing specific tree-building algorithms.

### 4.2 Regression

In regression, we use mean squared error as the impurity function $H$:

$$\bar{y}_m = \frac{1}{n_m} \sum_{x_i \in m} y_i,$$

$$H(m) = \frac{1}{n_m} \sum_{x_i \in m} (y_i - \bar{y}_m)^2.$$

If instead the impurity function $H$ is evaluated on a separate test set, we define

$$H'(m) = \frac{1}{n'_m} \sum_{i=1}^{n'_m} \left(y'_{m,i} - \bar{y}_m\right)^2$$

and similarly

$$\Delta'(\theta_m^*) = \omega_m H'(m) - (\omega_l H'(l) + \omega_r H'(r)).$$

Note that here $H'(m)$ measures mean squared error within node $m$ on test data with the fitted value $\bar{y}_m$ from training data. If we just sum up $\Delta'$ as feature importance, it will end up with negative values as $\bar{y}_m$ will overfit the training data and thus make mean squared error much larger deep in the tree. In other words, it *over-corrects* the bias. For this reason, our unbiased split-improvement is defined slightly different from the classification case (8):

$$\mathrm{VI}_j^{\mathrm{T,R}} = \sum_{m, j \in \theta_m^*} \left(\Delta(\theta_m^*) + \Delta'(\theta_m^*)\right). \tag{9}$$

Notice that although Equations (8) and (9) are different, they originates from the same idea by correcting bias using test data. Unlike Formula (6) for Gini index, where we could design a predictive impurity function by combining train and test data together, it's hard to come up with a counterpart in regression setting.

Just as in the classification case, we could show the following unbiasedness results:

Lemma 4.2. *In regression settings, for a given feature $X_j$, if $y$ is marginally independent of $X_j$ within the region defined by node $m$, then*

$$E(\Delta(\theta_m^*) + \Delta'(\theta_m^*)) = 0$$

*when splitting at the $j^{th}$ feature.*

Proof. See Appendix A. □

Theorem 2. *In regression settings, for a given feature $X_j$, if $y$ is independent of $X_j$ in every hyper-rectangle subset of the feature space, then we always have*

$$EVI_j^{T,R} = 0.$$

## 5 EMPIRICAL STUDIES

In this section, we apply our method to one simulated example and three real datasets. We compare our results to three other algorithms: the default split-improvement in *scikit-learn*, cforest [18] in R package *party* and bias-corrected impurity [30] in R package *ranger*. We did not include comparison with Li et al. [23] since their method does not enjoy the unbiased property. In what follows, we use shorthand SI for the default split-improvement, UFI for our method (unbiased feature importance).

### 5.1 Simulated Data

The data has 1,000 samples and 10 features, where $X_i$ takes values in $0, 1, 2, \ldots, i$ with uniform probability for $1 \le i \le 10$. Here, we assume only $X_1$ contains true signal and all remaining nine features are noisy features. The target value $y$ is generated as follows:

—Regression: $y = X_1 + 5\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$.
—Classification: $P(y = 1|X) = 0.55$ if $X_1 = 1$, and $P(y = 1|X) = 0.45$ if $X_1 = 0$.

Note that this task is designed to be extremely hard by choosing the binary feature as informative, and adding large noise (regression) or setting the signal strength low (classification). To evaluate the results, we look at the ranking of all features based on importance scores. Ideally $X_1$ should be ranked 1st as it is the only informative feature. Table 1 shows the average ranking of feature $X_1$ across 100 repetitions. The best result of each column is marked in bold. Here we also compare the effect of tree depth by constructing shallow trees (with tree depth 3) and deep trees

Table 1. Average Importance Ranking of Informative Feature $X_1$

| | Tree depth = 3 | | Tree depth = 10 | |
|---|---|---|---|---|
| | R | C | R | C |
| SI | 3.71 | 4.10 | 10.00 | 10.00 |
| UFI | **1.47** | 1.39 | **1.55** | **1.69** |
| cforest | 1.57 | **1.32** | 1.77 | 1.88 |
| ranger | 1.54 | 1.64 | 2.46 | 1.93 |

R stands for regression and C for classification. The result averages over 100 repetitions. Lower values indicate better abilities in identifying informative features. In cforest, we set mincriterion to be 2.33 (0.99 percentile of normal distribution) for shallow trees and 1.28 (0.9 percentile) for deep trees.

(with tree depth 10). Since cforest does not provide a parameter for directly controlling tree depth, we change the values of mincriterion as an alternative.

We can see that our method UFI achieves the best results in three situations except the classification case for shallow trees, where it is only slightly worse than cforest. Another interesting observation is that deeper trees tend to make the task of identifying informative features harder when there are noisy ones, since it is more likely to split on noisy features for splits deep down in the tree. This effect is most obvious for the default split-improvement, where it performs the worst especially for deep trees: the informative feature $X_1$ is consistently ranked as the least important ($10^{\text{th}}$ place). UFI does not seem to be affected too much from tree depth.

### 5.2 RNA Sequence Data

The first dataset examined is the prediction of C-to-U edited sites in plant mitochondrial Ribonucleic acid (RNA). This task was studied statistically in Cummings and Myers [8], where the authors applied RF and used the original split-improvement as feature importance. Later, Strobl et al. [39] demonstrated the performance of cforest on this dataset.

RNA editing is a molecular process whereby an RNA sequence is modified from the sequence corresponding to the DNA template. In the mitochondria of land plants, some cytidines are converted to uridines before translation [8].

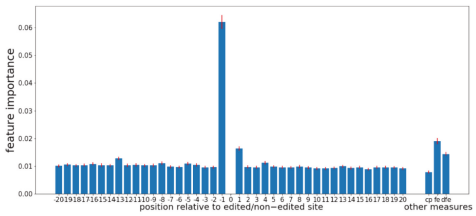
We use the *Arabidopsis thaliana* data file[6] as in Strobl et al. [39]. The features are based on the nucleotides surrounding the edited/non-edited sites and on the estimated folding energies of those regions. After removing missing values and one column which will not be used, the data file consists of 876 rows and 45 columns:

—the response (binary);
—a total of 41 nucleotides at positions −20 to 20 relative to the edited site (categorical, one of A, T, C or G);
—the codon position (also four categories); and
—two continuous variables based on on the estimated folding energies.

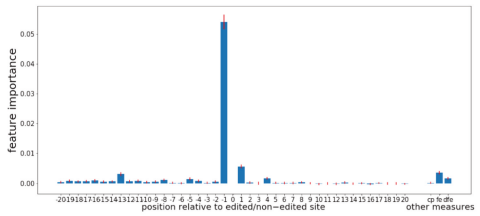
For implementation, we create dummy variables for all categorical features, and build forest using 100 base trees. The maximum tree depth for this dataset is not restricted as the number of potential predictors is large. We take the sum of importance across all dummy variables corresponding to a specific feature for final importance scores. All default parameters are used unless otherwise specified.

The results are shown in Figure 5. Red error bars depict one standard deviation when the experiments are repeated 100 times. From the default split-improvement (Figure 5(a)), we can see that

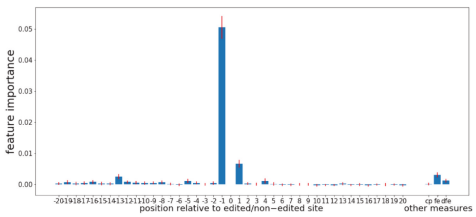
[6]The dataset can be downloaded from https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-5-132.



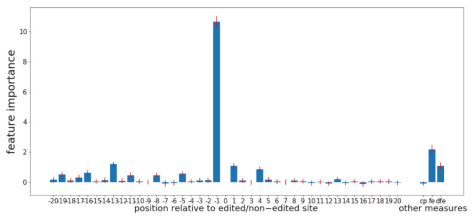

(a) SI




(b) UFI




(c) cforest




(d) ranger

Fig. 5. Feature importance for RNA sequence data. A total of 100 trees are built in the forest. Red error bars depict one standard deviation when the experiments are repeated 100 times. The x-axis denotes two classes of features: position relative to edited/non−edited site, other measures (cp, fe, dfe).

except several apparently dominant predictors (nucleotides at position −1 and 1, and two continuous features *fe* and *dfe*), the importance for the remaining nearly 40 features are indistinguishable. The feature importance scores given by UFI (Figure 5(b)) and cforest (Figure 5(c)) are very similar. Compared with SI, although all methods agree on top three features being the nucleotides at position −1 and 1, and the continuous one *fe*, there are some noticeable differences. Another continuous feature *dfe* is originally ranked at the fourth place in Figure 5(a), but its importance scores are much lower by UFI and cforest. The result given by ranger (Figure 5(d)) is slightly different from UFI and cforest, where it seems to have more features with importance scores larger than 0. In general, we see a large portion of predictors with feature importance close to 0 for three improved methods, which makes subsequent tasks like feature screening easier.

### 5.3 Adult Data

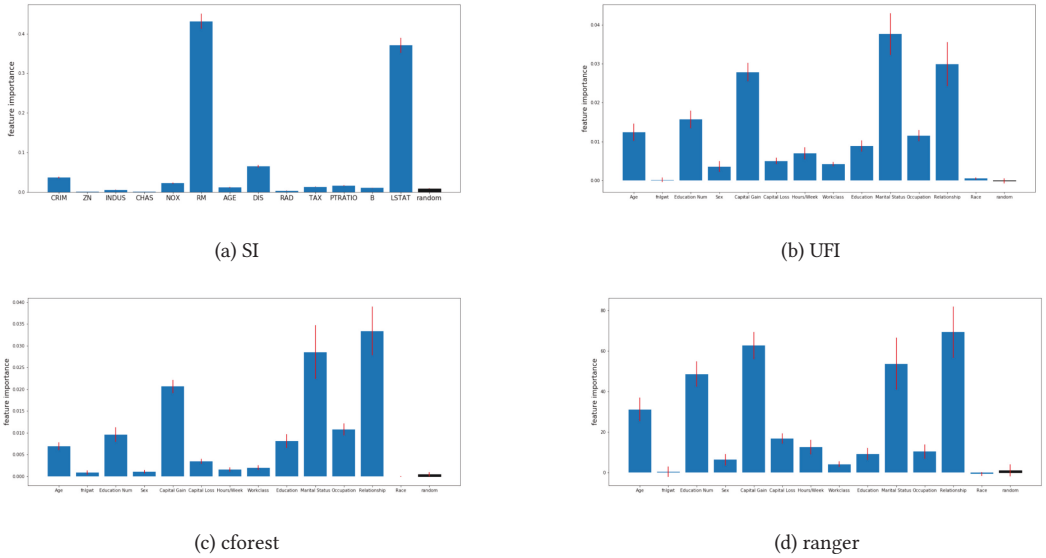
As a second example, we will use the Adult Dataset from UCI Machine Learning Repository.[7] The task is to predict whether income exceeds $50K/yr based on census data. We remove all entries including missing values, and only focus on people from Unites States. In total, there are 27,504 training samples and Table 2 describes relevant feature information. Notice that we add a standard normal random variable, which is shown in the last row. We randomly sample 5,000 entries for training.

The results are shown in Figure 6. UFI (Figure 6(b)), cforest(Figure 6(c)), and ranger (Figure 6(d)) display similar feature rankings which are quite different from the original split-improvement (Figure 6(a)). Notice the random normal feature we added (marked in black) is actually ranked the third most important in Figure 6(a). This is not surprising as most of the features are categorical, and even for some continuous features, a large portion of the values are actually 0 (such as *capital-gain* and *capital-loss*). For UFI, cforest and ranger, the random feature is assigned an importance

[7] https://archive.ics.uci.edu/ml/datasets/adult.

Table 2. Attribute Description for Adult Dataset

| Attribute | Description |
|---|---|
| age | continuous |
| workclass | categorical (7) |
| fnlwgt | continuous |
| education | categorical (16) |
| education-num | continuous |
| marital-status | categorical (7) |
| occupation | categorical (14) |
| relationship | categorical (6) |
| race | categorical (5) |
| sex | binary |
| capital-gain | continuous |
| capital-loss | continuous |
| hours-per-week | continuous |
| random | continuous |




(a) SI

(b) UFI

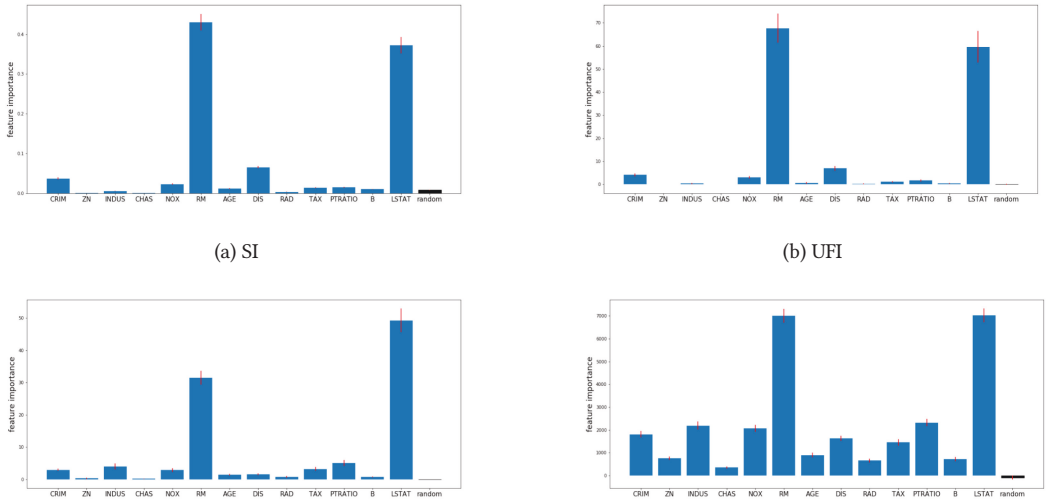
(c) cforest

(d) ranger

Fig. 6. Feature importance for adult data. A total of 20 trees are built in the forest. Red error bars depict one standard deviation when the experiments are repeated 100 times. The x-axis lists feature names for the model: Age, fnlgwt, Education, Sex, Capital Gain, Capital Loss, Hours/Week, Workclass, Education, Marital Status, Occupation, Relationship, Race, and random.

score close to 0. Another feature with big discrepancy is *fnlwgt*, which is ranked among top three originally but is the least important for other methods. *fnlwgt* represents final weight, the number of units in the target population that the responding unit represents. Thus it is unlikely to have strong predictive power for the response. For this reason, some analyses deleted this predictor before fitting models.[8]

[8] http://scg.sdsu.edu/dataset-adult_r/.




(a) SI (b) UFI

(c) cforest (d) ranger

Fig. 7. Feature importance for Boston housing data. A total of 100 trees are built in the forest. Red error bars depict one standard deviation when the experiments are repeated 100 times. The x-axis lists feature names for the model: CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, and random.

### 5.4 Boston Housing Data

We also conduct analyses on a regression example using the Boston Housing Data,[9] which has been widely studied in previous literature [2, 33]. The dataset contains 12 continuous, one ordinal and one binary features and the target is median value of owner-occupied homes in $1,000's. We add a random feature distributed as $\mathcal{N}(0, 1)$ as well.

All four methods agree on two most important features: RM (average number of rooms per dwelling) and LSTAT (% lower status of the population). In SI, the random feature still appears to be more important than several other features such as INDUS (proportion of non-retail business acres per town) and RAD (index of accessibility to radial highways), though the spurious effect is much less compared to Figure 6(a). As expected, the importance of random feature is close to zero in UFI. In this example, the SI did not seem to provide misleading result as most of the features are continuous, and the only binary feature CHAS (Charles River dummy variable) turns out to be not important.

### 5.5 Summary

Our empirical studies confirm that the default split-improvement method is biased towards increasing the importance of features with more potential splits. The bias is more severe in deeper trees. Compared to three other approaches, our proposed method performs the best in a difficult task to identify the only important feature from 10 noisy features. For real-world datasets, though we do not have a ground truth for feature importance scores, our method gives similar and meaningful outputs as two state-of-the-art methods cforest and ranger.

[9] https://archive.ics.uci.edu/ml/machine-learning-databases/housing/.

## 6 DISCUSSIONS

Tree-based methods are widely employed in many applications. One of the many advantages is that these models come naturally with feature importance measures, which practitioners rely on heavily for subsequent analysis such as feature ranking or screening. It is important that these measurements are trustworthy.

We show empirically that split-improvement, as a popular measurement of feature importance in tree-based models, is biased towards continuous features, or categorical features with more categories. This phenomenon is akin to overfitting in training any machine learning model. We propose a simple fix to this problem and demonstrate its effectiveness both theoretically and empirically. Though our examples are based on RF, the adjustment can be easily extended to any other tree-based model.

The original version of split-improvement is the default and only feature importance measure for RF in *scikit-learn*, and is also returned as one of the measurements for *randomForest* library in R. Statistical analyses utilizing these packages will suffer from the bias discussed in this article. Our method can be easily integrated into existing libraries, and require almost no additional computational burden. As already observed, while we have used *out-of-bag* samples as a natural source of test data, alternatives such as sample partitions—thought of as a subsample of *out-of-bag* data for our purposes—can be used in the context of honest trees, or a held-out test set will also suffice. The use of subsamples fits within the methods used to demonstrate the asymptotic normality of RF developed in Mentch and Hooker [27]. This potentially allows for formal statistical tests to be developed based on the unbiased split-improvement measures proposed here. Similar approaches have been taken in Zhou et al. [44] for designing stopping rules in approximation trees.

However, feature importance itself is very difficult to define exactly, with the possible exception of linear models, where the magnitude of coefficients serves as a simple measure of importance. There are also considerable discussion on the subtly introduced when correlated predictors exist, see for example [14, 38]. We think that clarifying the relationship between split-improvement and the topology of the resulting function represents an important future research direction.

## APPENDICES

## A PROOFS OF LEMMA 4.1 AND 4.2

Proof of Lemma 4.1 We want to show that for independent $X_j$ and $y$ within node $m$, $\Delta'(\theta^*_m)$ should ideally be zero when splitting on the $j^{th}$ variable. Rewriting $H'(m)$ defined in Equation (6) and we get:

$$\begin{aligned} H'(m) &= 1 - p_{m,1}p'_{m,1} - p_{m,2}p'_{m,2} \\ &= 1 - p_{m,1}p'_{m,1} - (1 - p_{m,1})(1 - p'_{m,1}) \\ &= p_{m,1} + p'_{m,1} - 2p_{m,1}p'_{m,1}. \end{aligned}$$

Using similar expressions for $H'(l)$, we have:

$$H'(m) - H'(l) = (p_{m,1} + p'_{m,1} - 2p_{m,1}p'_{m,1}) - (p_{l,1} + p'_{l,1} - 2p_{l,1}p'_{l,1}).$$

Given that the test data is independent of the training data and the independence between $X_j$ and $y$, then in expectation, we should have $E(p'_{m,1}) = E(p'_{l,1}) = p'_1$. Thus,

$$\begin{aligned} E(H'(m) - H'(l)) &= (E(p_{m,1}) + E(p'_{m,1}) - 2E(p_{m,1}p'_{m,1})) - (E(p_{l,1}) + E(p'_{l,1}) - 2E(p_{l,1}p'_{l,1})) \\ &= (E(p_{m,1}) + E(p'_{m,1}) - 2E(p_{m,1})E(p'_{m,1})) - (E(p_{l,1}) + E(p'_{l,1}) - 2E(p_{l,1})E(p'_{l,1})) \\ &= (E(p_{m,1}) + p'_1 - 2E(p_{m,1})p'_1) - (E(p_{l,1}) + p'_1 - 2E(p_{l,1})p'_1) \\ &= (E(p_{m,1}) - E(p_{l,1}))(1 - 2p'_1). \end{aligned}$$

Similarly,

$$E(H'(m) - H'(r)) = (E(p_{m,1}) - E(p_{r,1}))(1 - 2p'_1).$$

Combined together into Equation (7),

$$\begin{aligned}
E(\Delta'(\theta^*_m)) &= \omega_l(H'(m) - H'(l)) + \omega_r(H'(m) - H'(r)) \\
&= \omega_l(E(p_{m,1}) - E(p_{l,1}))(1 - 2p'_1) + \omega_r(E(p_{m,1}) - E(p_{r,1}))(1 - 2p'_1) \\
&= (1 - 2p'_1)(\omega_m E(p_{m,1}) - \omega_l E(p_{l,1}) - \omega_r E(p_{r,1})) \\
&= (1 - 2p'_1) \times 0 \\
&= 0,
\end{aligned}$$

since we always have

$$\omega_m \times p_{m,1} = \omega_l \times p_{l,1} + \omega_r \times p_{r,1}. \qquad \square$$

Proof of Lemma 4.2. Rewriting the expression of $H(m)$:

$$\begin{aligned}
H(m) &= \frac{1}{n_m}\sum_{i=1}^{n_m}(y_{m,i} - \bar{y}_m)^2 \\
&= \frac{1}{n_m}\left(\sum_{i=1}^{n_m} y_{m,i}^2 - n_m\bar{y}_m^2\right).
\end{aligned}$$

Thus,

$$\begin{aligned}
\Delta(\theta^*_m) &= \omega_m H(m) - (\omega_l H(l) + \omega_r H(r)) \\
&= \omega_m \frac{1}{n_m}\sum_{i=1}^{n_m}(y_{m,i}^2 - n_m\bar{y}_m^2) - \left(\omega_l \frac{1}{n_l}\sum_{i=1}^{n_l}(y_{l,i}^2 - n_l\bar{y}_l^2) + \omega_r \frac{1}{n_r}\sum_{i=1}^{n_r}(y_{r,i}^2 - n_r\bar{y}_r^2)\right) \\
&= \frac{1}{n}\sum_{i=1}^{n_m}(y_{m,i}^2 - n_m\bar{y}_m^2) - \left(\frac{1}{n}\sum_{i=1}^{n_l}(y_{l,i}^2 - n_l\bar{y}_l^2) + \frac{1}{n}\sum_{i=1}^{n_r}(y_{r,i}^2 - n_r\bar{y}_r^2)\right) \\
&= \frac{1}{n}\left(\sum_{i=1}^{n_m}(y_{m,i}^2 - n_m\bar{y}_m^2) - \sum_{i=1}^{n_l}(y_{l,i}^2 - n_l\bar{y}_l^2) - \sum_{i=1}^{n_r}(y_{r,i}^2 - n_r\bar{y}_r^2)\right) \\
&= \frac{1}{n}\left(\sum_{i=1}^{n_m} y_{m,i}^2 - \sum_{i=1}^{n_l} y_{l,i}^2 - \sum_{i=1}^{n_r} y_{r,i}^2\right) - \frac{1}{n}(n_m\bar{y}_m^2 - n_l\bar{y}_l^2 - n_r\bar{y}_r^2) \\
&= \frac{1}{n}(n_l\bar{y}_l^2 + n_r\bar{y}_r^2 - n_m\bar{y}_m^2) \\
&= \omega_l\bar{y}_l^2 + \omega_r\bar{y}_r^2 - \omega_m\bar{y}_m^2.
\end{aligned}$$

By Cauchy–Schwarz inequality,

$$(n_l\bar{y}_l^2 + n_r\bar{y}_r^2)(n_l + n_r) \geq (n_l\bar{y}_l + n_r\bar{y}_r)^2 = (n_m\bar{y}_m)^2,$$

thus

$$\Delta(\theta^*_m) = \frac{1}{n}(n_l\bar{y}_l^2 + n_r\bar{y}_r^2 - n_m\bar{y}_m^2) \geq 0$$

unless $\bar{y}_l = \bar{y}_r = \bar{y}_m$.

Similarly for $H'(m)$:

$$\begin{aligned} H'(m) &= \frac{1}{n'_m}\sum_{i=1}^{n'_m}(y'_{m,i}-\bar{y}_m)^2 \\ &= \frac{1}{n'_m}\sum_{i=1}^{n'_m}y'^2_{m,i} - 2\frac{1}{n'_m}\sum_{i=1}^{n'_m}y'_{m,i}\bar{y}_m + \frac{1}{n'_m}\sum_{i=1}^{n'_m}\bar{y}^2_m \\ &= \frac{1}{n'_m}\sum_{i=1}^{n'_m}y'^2_{m,i} - 2\bar{y}'_m\bar{y}_m + \bar{y}^2_m \end{aligned}$$

and thus

$$\begin{aligned} \Delta'(\theta^*_m) &= \omega_m H'(m) - (\omega_l H'(l) + \omega_r H'(r)) \\ &= \omega_m\left(\frac{1}{n'_m}\sum_{i=1}^{n'_m}y'^2_{m,i} - 2\bar{y}'_m\bar{y}_m + \bar{y}^2_m\right) - \omega_l\left(\frac{1}{n'_l}\sum_{i=1}^{n'_l}y'^2_{l,i} - 2\bar{y}'_l\bar{y}_l + \bar{y}^2_l\right) - \omega_r\left(\frac{1}{n'_r}\sum_{i=1}^{n'_r}y'^2_{r,i} - 2\bar{y}'_r\bar{y}_r + \bar{y}^2_r\right) \\ &= \left(\omega_m\frac{1}{n'_m}\sum_{i=1}^{n'_m}y'^2_{m,i} - \omega_l\frac{1}{n'_l}\sum_{i=1}^{n'_l}y'^2_{l,i} - \omega_r\frac{1}{n'_r}\sum_{i=1}^{n'_r}y'^2_{r,i}\right) + (\omega_m\bar{y}^2_m - \omega_l\bar{y}^2_l - \omega_r\bar{y}^2_r) - 2(\omega_m\bar{y}'_m\bar{y}_m - \omega_l\bar{y}'_l\bar{y}_l - \omega_r\bar{y}'_r\bar{y}_r) \\ &= \left(\omega_m\frac{1}{n'_m}\sum_{i=1}^{n'_m}y'^2_{m,i} - \omega_l\frac{1}{n'_l}\sum_{i=1}^{n'_l}y'^2_{l,i} - \omega_r\frac{1}{n'_r}\sum_{i=1}^{n'_r}y'^2_{r,i}\right) - \Delta(\theta^*_m) - 2(\omega_m\bar{y}'_m\bar{y}_m - \omega_l\bar{y}'_l\bar{y}_l - \omega_r\bar{y}'_r\bar{y}_r). \end{aligned}$$

By the independence assumptions, we have

$$E\frac{1}{n'_m}\sum_{i=1}^{n'_m}y'^2_{m,i} = E\frac{1}{n'_l}\sum_{i=1}^{n'_l}y'^2_{l,i} = E\frac{1}{n'_r}\sum_{i=1}^{n'_r}y'^2_{r,i},$$

and

$$E\bar{y}'_m = E\bar{y}'_l = E\bar{y}'_r.$$

We can conclude that

$$E(\Delta(\theta^*_m) + \Delta'(\theta^*_m)) = 0.$$

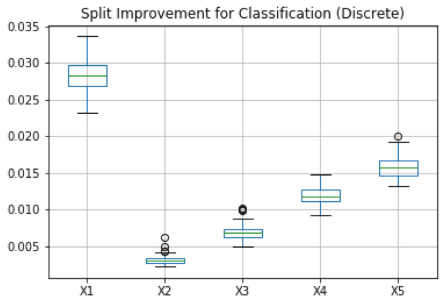
□

## B ADDITIONAL SIMULATION RESULTS

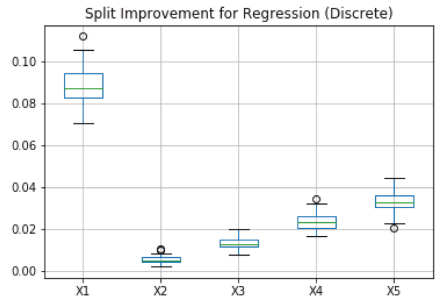
Our simulation experiments in Sections 3 and 4 operate by creating dummy variables for categorical features. It would be interesting to see the results if we instead treat those as ordered discrete values.

Figures 8 and 9 show the original version of split-improvement corresponding to Figures 1 and 2. Similar phenomenon is again observed: it over estimates importance of continuous features and categorical features with more categories. It is worth noticing that the discrepancy between continuous and categorical features is even larger in this case. Unlike in Figure 1(a), $X_1$ is always ranked the most important. This results from the fact that by treating categorical features as ordered discrete ones, it limit the number of potential splits compared to using dummy variables.

Not surprisingly, our proposed method work well in declaring all five features have no predictive power or finding the most informative one, as shown in Figures 10 and 11.

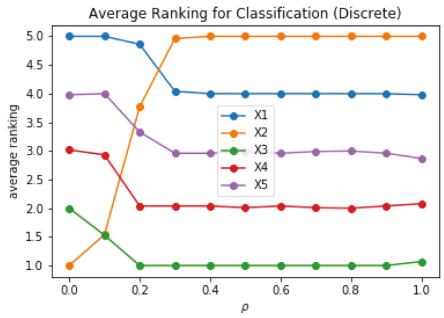



(a) Classification

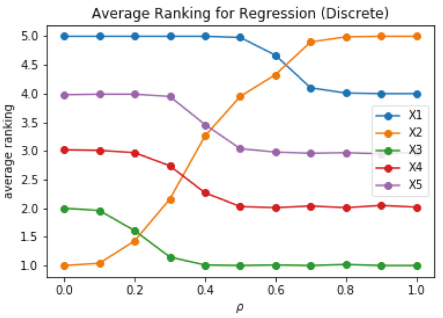



(b) Regression

Fig. 8. Split-improvement measures on five predictors, where we treat categorical features as ordered discrete values. Box plot is based on 100 repetitions. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5.

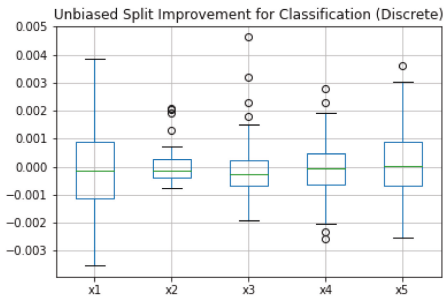



(a) Classification

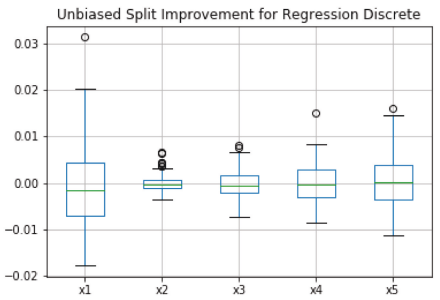



(b) Regression

Fig. 9. Average feature importance ranking across different signal strengths over 100 repetitions, where we treat categorical features as ordered discrete values. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5.

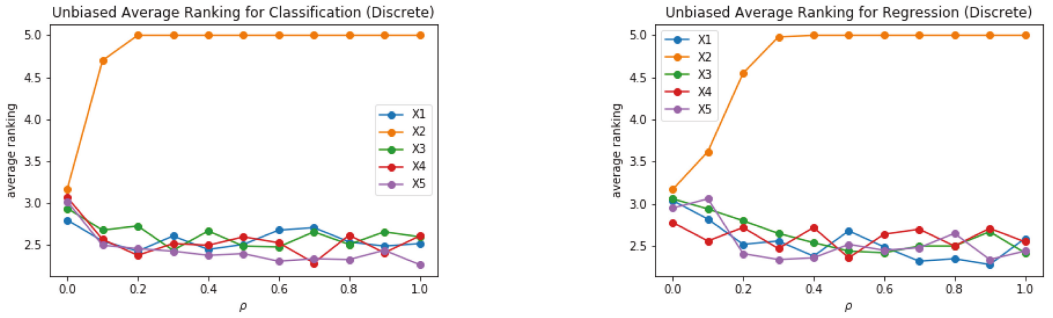



(a) Classification




(b) Regression

Fig. 10. Unbiased split-improvement, where we treat categorical features as ordered discrete values. Box plot is based on 100 repetitions. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5. Each tree is trained using bootstrap samples and *out-of-bag* samples are used as test set.




(a) Classification

(b) Regression

Fig. 11. Unbiased feature importance ranking across different signal strengths averaged over 100 repetitions, where we treat categorical features as ordered discrete values. A total of 100 trees are built in the forest and maximum depth of each tree is set to 5. Each tree is trained using bootstrap samples and *out-of-bag* samples are used as test set.

## REFERENCES


[1] Rina Foygel Barber and Emmanuel J. Candès. 2015. Controlling the false discovery rate via knockoffs. *The Annals of Statistics* 43, 5 (2015), 2055–2085.

[2] Galen Bollinger. 1981. Book Review: Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. *Journal of Marketing Research* 18, 3 (1981), 392–393.

[3] Anne-Laure Boulesteix, Andreas Bender, Justo Lorenzo Bermejo, and Carolin Strobl. 2011. Random forest Gini importance favours SNPs with large minor allele frequency: Impact, sources and recommendations. *Briefings in Bioinformatics* 13, 3 (2011), 292–304.

[4] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140.

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

[6] Leo Breiman, Jerome H. Friedman, R. A. Olshen, and Charles J. Stone. 1984. Classification and regression trees. CRC press.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[8] Michael P. Cummings and Daniel S. Myers. 2004. Simple statistical models predict C-to-U edited sites in plant mitochondrial RNA. *BMC Bioinformatics* 5, 1 (2004), 132.

[9] Misha Denil, David Matheson, and Nando De Freitas. 2014. Narrowing the gap: Random forests in theory and in practice. In *Proceedings of the International Conference on Machine Learning*. 665–673.

[10] Ramón Díaz-Uriarte and Sara Alvarez De Andres. 2006. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics* 7, 1 (2006), 3.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. Vol. 1. Springer.

[12] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29, 5 (2001), 1189–1232.

[13] Indrayudh Ghosal and Giles Hooker. 2018. Boosting random forests to reduce bias; one-step boosted forest and its variance estimate. *Journal of Computational and Graphical Statistics* (2020), 1–10.

[14] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. 2017. Correlation and variable importance in random forests. *Statistics and Computing* 27, 3 (2017), 659–678.

[15] Giles Hooker. 2007. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics* 16, 3 (2007), 709–732.

[16] Giles Hooker and Lucas Mentch. 2019. Please stop permuting features: An explanation and alternatives. *arXiv preprint arXiv:1905.03151* (2019).

[17] Torsten Hothorn, Kurt Hornik, Carolin Strobl, and Achim Zeileis. 2010. Party: A laboratory for recursive partytioning. Retrieved from https://CRAN.R-project.org/package=party.

[18] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. 2006. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics* 15, 3 (2006), 651–674.

[19] Hemant Ishwaran. 2007. Variable importance in binary regression trees and forests. *Electronic Journal of Statistics* 1 (2007), 519–537.

[20] Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. 2008. Random survival forests. *The Annals of Applied Statistics* 2, 3 (2008), 841–860.
[21] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. 2016. Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003* (2016).
[22] Hyunjoong Kim and Wei-Yin Loh. 2001. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association* 96, 454 (2001), 589–604.
[23] Xiao Li, Yu Wang, Sumanta Basu, Karl Kumbier, and Bin Yu. 2019. A debiased MDI feature importance measure for random forests. *Advances in Neural Information Processing Systems* 32 (2019), 8049–8059.
[24] Wei-Yin Loh. 2009. Improving the precision of classification trees. *The Annals of Applied Statistics* 3, 4 (2009), 1710–1737.
[25] Wei-Yin Loh. 2014. Fifty years of classification and regression trees. *International Statistical Review* 82, 3 (2014), 329–348.
[26] Wei-Yin Loh and Yu-Shan Shih. 1997. Split selection methods for classification trees. *Statistica Sinica* 7, 4 (1997), 815–840.
[27] Lucas Mentch and Giles Hooker. 2016. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research* 17, 1 (2016), 841–881.
[28] Bjoern H. Menze, B. Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A. Hamprecht. 2009. A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics* 10, 1 (2009), 213.
[29] Michael Meyer, Juan Felipe Beltrán, Siqi Liang, Robert Fragoza, Aaron Rumack, Jin Liang, Xiaomu Wei, and Haiyuan Yu. 2017. Interactome INSIDER: A multi-scale structural interactome browser for genomic studies. *bioRxiv* (2017), 126862.
[30] Stefano Nembrini, Inke R. König, and Marvin N. Wright. 2018. The revival of the Gini importance? *Bioinformatics* 34, 21 (2018), 3711–3718.
[31] Kristin K. Nicodemus. 2011. Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures. *Briefings in Bioinformatics* 12, 4 (2011), 369–373.
[32] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.
[33] J. Ross Quinlan. 1993. Combining instance-based and model-based learning. In *Proceedings of the 10th International Conference on Machine Learning*. 236–243.
[34] J. Ross Quinlan. 2014. *C4. 5: Programs for Machine Learning*. Elsevier.
[35] Marco Sandri and Paola Zuccolotto. 2008. A bias correction algorithm for the Gini variable importance measure in classification trees. *Journal of Computational and Graphical Statistics* 17, 3 (2008), 611–628.
[36] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. 2015. Consistency of random forests. *The Annals of Statistics* 43, 4 (2015), 1716–1741.
[37] Daria Sorokina and Erick Cantú-Paz. 2016. Amazon search: The joy of ranking products. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 459–460.
[38] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. Conditional variable importance for random forests. *BMC Bioinformatics* 9, 1 (2008), 307.
[39] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics* 8, 1 (2007), 25.
[40] Stefan Wager and Susan Athey. 2018. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association* 113, 523 (2018), 1228–1242.
[41] Yisen Wang, Shu-Tao Xia, Qingtao Tang, Jia Wu, and Xingquan Zhu. 2017. A novel consistent random forest framework: Bernoulli random forests. *IEEE Transactions on Neural Networks and Learning Systems* 29, 8 (2017), 3510–3523.
[42] Wang Yisen, Tang Qingtao, Shu-Tao Xia, Jia Wu, and Xingquan Zhu. 2016. Bernoulli random forests: Closing the gap between theoretical consistency and empirical soundness. In *Proceedings of the IJCAI International Joint Conference on Artificial Intelligence*.
[43] Yichen Zhou and Giles Hooker. 2018. Boulevard: Regularized stochastic gradient boosted trees and their limiting distribution. *arXiv preprint arXiv:1806.09762* (2018).
[44] Yichen Zhou, Zhengze Zhou, and Giles Hooker. 2018. Approximation trees: Statistical stability in model distillation. *arXiv preprint arXiv:1808.07573* (2018).

Received March 2019; revised August 2020; accepted October 2020