entropy

MDPI

Article

Feature Importance in Gradient Boosting Trees with Cross-Validation Feature Selection

Afek Ilay Adler and Amichai Painsky *

The Industrial Engineering Department, Tel Aviv University, Tel Aviv 69978, Israel; afekadler@mail.tau.ac.il
* Correspondence: amichaip@tauex.tau.ac.il

Abstract: Gradient Boosting Machines (GBM) are among the go-to algorithms on tabular data, which produce state-of-the-art results in many prediction tasks. Despite its popularity, the GBM framework suffers from a fundamental flaw in its base learners. Specifically, most implementations utilize decision trees that are typically biased towards categorical variables with large cardinalities. The effect of this bias was extensively studied over the years, mostly in terms of predictive performance. In this work, we extend the scope and study the effect of biased base learners on GBM feature importance (FI) measures. We demonstrate that although these implementation demonstrate highly competitive predictive performance, they still, surprisingly, suffer from bias in FI. By utilizing cross-validated (CV) unbiased base learners, we fix this flaw at a relatively low computational cost. We demonstrate the suggested framework in a variety of synthetic and real-world setups, showing a significant improvement in all GBM FI measures while maintaining relatively the same level of prediction accuracy.

Keywords: gradient boosting; feature importance; tree-based methods; classification and regression trees

Citation: Adler, A.I.; Painsky, A. Feature Importance in Gradient Boosting Trees with Cross-Validation Feature Selection. *Entropy* **2022**, *24*, 687. https://doi.org/10.3390/e24050687

Academic Editor: António M. Lopes

Received: 14 April 2022
Accepted: 11 May 2022
Published: 13 May 2022

1. Introduction

In recent years, machine learning (ML) has gained much popularity and became an integral part of our daily life. Current state-of-the-art algorithms are complex models that are difficult to interpret and rely on thousands, and even billions of parameters. Gaining insight into how these complex algorithms work is a key step towards better understanding our environment. Further, it is a crucial step for enabling ML algorithms to support and assist human decision making in complex fields (for example, as in medicine [1]). FI is one of the basic tools for this purpose. The general FI framework scores input variables by their contribution to the predictive model, allowing us to gain insight into which features are important for this task. GBM [2] is an ensemble of base (weak) learners. These learners are typically standard implementations of tree-based models such as CART [3], C4.5 [4], and others. GBMs are among the current state-of-the-art ML techniques on tabular data in a variety of tasks such as click prediction [5], ranking [6], and others. Besides its accuracy, the GBM framework holds many virtues, which makes it a favorable choice for many learning tasks. It is efficient, can handle categorical variables and deals with any kind of differential loss function. Further, it supports missing values and invariants to feature scaling. However, it is well known that GBM base learners are biased towards categorical variables with large cardinalities [3]. As a consequence, ensemble methods that rely on such biased learners suffer from the same flaw [7].

In this work, we study the effect of base learners' bias on FI measures, in standard GBM implementations. We show that in the presence of categorical features with large alphabets, most FI measures are typically biased. This results in misinterpreted models. To overcome this caveat, we utilize a cross-validated boosting (CVB) framework. CVB addresses the tree's bias through a simple modification: during the training stage of the tree, a variable is selected for splitting based on its cross-validated performance, rather

than on its training sample performance (see [8], for example). This way, we conduct a "fair" comparison between features, utilizing only high cardinality categorical features that are demonstrated to be informative, in terms of their estimated generalization abilities.

The rest of this manuscript is organized as follows, in Section 2, we review previous related work. In Section 3, we formalize and define cross-validated boosting (CVB). Section 4 outlines the experiment design in which we compare CVB to other GBM frameworks. In Section 5, we compare CVB to common open-source GBM implementations and demonstrate that CVB is the only algorithm that does not suffer from biased FI in high cardinality features through simulation studies. Finally, Section 6 contains two comprehensive real data case studies, demonstrating that CVB FI is more reliable than alternative schemes. We conclude the manuscript in Section 7.

2. Background

2.1. Decision Trees

Decision trees are a longstanding staple of predictive modeling. Popular tree building algorithms can handle both numerical and categorical variables, and build models for regression, binary and multi-class classification. Decision trees are built recursively to minimize a given loss function. At each step, the split that optimizes a given criterion is selected from all possible splits over on all variables. We hereby present the CART approach, which is one of the most widely used tree implementations in current GBM implementations.

Let $\{x_i, y_i\}_{i=1}^N$ be a training set with N samples , where $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}$. The standard CART approach is a binary tree which utilizes either squared error loss impurity criterion (regression) or the Gini index (classification) to evaluate a possible split. At each internal node of a tree, a split s is a partition of n observations to two disjoint subsets $R(s)$ and $L(s)$. In regression tress, the squared error impurity criterion is defined as

$$\mathcal{I}(s) = \Sigma_{i \in L(s)}(y_i - \bar{y}_L)^2 + \Sigma_{i \in R(s)}(y_i - \bar{y}_R)^2, \tag{1}$$

where $\bar{y}_L, \bar{y}_R$ are the means of the response y over the sets $L(s), R(s)$ respectively. The split gain (SG) is defined as the amount of decrease in the impurity of the split:

$$SG(s) = \Sigma_{i=1}^n (y_i - \bar{y})^2 - \mathcal{I}(s). \tag{2}$$

At each node, CART examines all possible splits for each feature and selects the split with the minimum impurity (maximum gain). For each variable j, we denote the set of possible splits by S_j and their cardinality by $|S_j|$. Let K_j be the number of categories in a categorical feature j. For each numerical/ordinal variable, CART considers only splits along the sorted variable. Therefore, there are $|S_j| \leq n - 1$ possible splits between its unique sorted values. For a categorical variable, there are $|S_j| = 2^{K_j - 1}$ possible binary splits, but it is sufficient to sort the categories by their mean response value and search for splits along this axis [3], reducing the complexity to only $K_j - 1$ splits.

Without any regularization mechanism, terminal nodes are split and deep, and complex trees are formed. Thus, stopping criteria are typically applied to prevent over-fitting and reduce tree complexity. Several common stopping criteria are the maximum tree depth, minimum samples to split, minimum samples leaf and minimum impurity decrease. It is also possible to grow a tree to its maximum depth and then to prune it, such as in minimal cost-complexity pruning [3].

An additional commonly used decision tree implementation is C4.5 [4]. C4.5 considers K-way splits for categorical features (as opposed to binary splits in CART). C4.5 trees may be problematic for large cardinality categorical features, as they result in splits with very few samples, and henceforth induce variance. Further, C4.5 trees are not used in recent GBM implementations. A comprehensive discussion on different splitting frameworks is provided in [9].

2.2. Gradient Boosting Trees

GBM is an ensemble model of M regression trees, trained sequentially one after the other, in order to predict the previous trees' errors. GBM may be viewed as a gradient descent in a functional space, where in each iteration of GBM, a tree is grown to estimate the gradient of the objective function. Let $F_m : \mathbb{R}^p \to \mathbb{R}$ be a predictive model at stage m, where $m = 1, \ldots, M$. Let $\mathcal{L}(y_i, F_m(x_i))$ be a differentiable loss function. For example, $\mathcal{L}_{SE}(y_i, F_m(x_i)) = (y_i - F_m(x_i))^2$ is the squared error loss in the standard regression case.

GBM initializes with a constant model by estimating a parameter that minimizes the overall loss (for the squared error $F_0(x_i) = \bar{y}$). Then, at every iteration m, a regression decision tree h_m is trained to minimize the empirical risk

$$h_m = \underset{h}{\operatorname{argmin}} \Sigma_{i=1}^{N} \mathcal{L}(y_i, F_{m-1}(x_i) + h(x_i)).$$

h_m is obtained by fitting a regression tree to the gradients of each sample with respect to the current estimator at stage m. The optimal step size is calculated per each leaf by a line search. h_m can be considered as the best greedy step in order to minimize the empirical risk. In order to reduce over-fitting, a learning-rate α is used and the model is updated, $F_m = F_{m-1} + \alpha h_m$. This process is repeated until M trees are grown. Later, inspired by Breiman's bootstrap-aggregating (bagging) [10], Friedman showed that a stochastic modification to GBM can substantially improve the prediction error of GBM [11]. In its stochastic form, GBM fits each tree on a sub-sample of the training-set, sampled at random without replacement.

2.3. GBM Implementations

There are many implementations of GBM in the literature. Many of these implementations follow Friedman's GBM, which is based on CART trees. In the age of big data, modern ML tasks require very large datasets. Therefore, the need for more scalable, distributed, GBM solutions arise (such as [12]). This line of work mostly focuses on the efficient implementation of GBM in distributed environments such as MapReduce [13]. Here, the most notable implementation is XGBoost [14]. XGBoost has two main advantages. First, it unifies previous ideas of distributed computing with novel optimisations such as out-of-core-computation and sparsity-awareness. Thus, producing much lower run-time both on a single machine and in a distributed environment. Second, XGBoost introduces a new regularized learning objective, which penalizes the complexity of the model. The result is a highly scaled end-to-end open-source implementation that produces state-of-the-art results, winning multiple data science open competitions.

Unfortunately, the main drawback of the XGBoost framework is the lack of categorical variables support. Specifically, in order to utilize a categorical feature, one needs to transform it to numeric variables such as in one-hot-encoding or to use *term statistics*. One-hot-encoding is a valid method to transform categorical features to numeric ones, but in cases of high cardinality variables, it substantially increases the number of features and henceforth the execution time. *Term statistics* is a method to transform a categorical feature to numeric by using the target value. Here, each category is replaced with its respective mean response value or a smoothed version of it [15]. The main problem with this approach is that it implies an order among the categories, and limits the expressive power of CART in categorical splits.

Following XGBoost, Microsoft introduced LightGBM (LGBM) [16]. LGBM is a faster implementation of GBM that supports categorical features. LGBM uses sampling techniques to exclude data instances with small gradients and *exclusive feature bundling* to reduce the number of features tested to split at each node while achieving almost the same accuracy. More importantly, LGBM supports categorical features out-of-the-box and follows the original CART approach. Lately, CatBoost pushed the envelope of treating categorical features in GBM even further. CatBoost addresses a *target leakage* (see [15]) to significantly improve the prediction error. For categorical variables, CatBoost uses ordered *term statistics*

and it is henceforth more memory efficient than LGBM. Further, CatBoost utilities oblivious trees as base learners. Oblivious decision trees are decision trees for which all nodes at the same level use the same feature to split. These trees are balanced, less prone to over-fitting, and allow speeding up execution at testing time. Finally, CatBoost uses combinations of categorical features as additional categorical features to capture high-order dependencies between categorical features.

2.4. FI Methods

Understanding the decision-making mechanism of a ML algorithm is an important task in many applications. First, in several setups, attaining the best prediction is not our sole intent; we would like to know more about the problem and the data. Second, model interpretability is a key step for achieving and quantifying other desirable properties in artificial intelligence (AI) such as fairness, robustness, and causality [17]. Finally, as AI enters new domains as health and transportation (autonomous vehicles), interpretation is essential to trust those mechanisms and adopt them as an integral part of our lives.

While some ML algorithms are intrinsically interpretive (linear regression), they tend to be over-simplistic and generally suffer from a high bias. As modern ML evolves and computing power increases, complex algorithms are developed to attain superior prediction capabilities but in turn provide little interpretation. For this reason, many methods are developed to make these non-interpretable ML methods more understandable. These methods typically provide summary statistics for each variable, which can also be visualized for better understanding.

Global FI measures attempt to summarize to which extent each feature is important for the prediction task. Given an algorithm, train and/or test data, a global FI method outputs a single number per each feature that reflects its importance. A local FI measure outputs a FI vector per each observation. Thus, it provides a more detailed, personalized FI. Local explanations are crucial in fields such as personalized medicine. For instance, local explanations can assist anesthesiologists to avoid hypoxemia during surgery, by providing real-time interpretable risks and contributing factors [1]. Interaction FI are methods that measure the strength of the interaction between two features. A FI method is said to be model agnostic if it can be applied in the same manner for any ML algorithm. On the other hand, a FI measure is said to be model specific if it only applies for a specific algorithm [18].

While FI aims to understand which features are important given a trained model, feature selection methods usually train a model several times in order to reduce dimensionality for speed considerations [19] or to avoid over-fitting in cases such as "large-p small-n" tasks. While FI measures can be used as a means for variable selection by taking the most influential features, variable selection methods cannot always be used as FI measures. For example, methods that rely on greedy backward elimination of features (such as [20,21]), still need to use a FI measure to score each variable in the final selected feature subset.

2.4.1. Global FI Methods

As mentioned above, global FI measures can be agnostic or model-specific. A global model-specific FI for GBM can be derived by evaluating its individual trees FI and averaging them across all trees in the ensemble [2]. Given a tree, there are many methods to interpret how important is a feature by inspecting the non-terminal nodes and their corresponding features, which were used for splitting. The most prominent measures are:

- Split count—the number of times a feature is selected for a split in a tree.
- Cover—the number of observations in the training set that a node had split. Specifically, for every feature j, Cover is the total number of observations in splits that involve this feature.
- Gain or the decrease in impurity [3]—for every feature j, Gain is the sum of SG (2) that utilize this feature.

These measures are intuitive, simple and achieve a FI score that is often consistent with other FI measures at a low computational cost. Unfortunately, they are also extremely biased towards categorical variables with large cardinalities [7], as described in Section 2.5.

Among the model agnostic global FI methods, the Permutation FI (PFI) [22] is perhaps the most common approach. Its rationale is as follows. By randomly permuting the jth feature, its original association with the response variable is broken. Assume that the jth feature is associated with the outcome. As we apply the permuted feature, together with the remaining unpermuted features, to a given learning algorithm, the prediction accuracy is expected to substantially decrease. Thus, a reasonable measure for FI is the difference between the prediction error before and after permuting the variable. The PFI measure relies on random permutations, which greatly varies. Unfortunately, to achieve a reliable estimate, it is required to conduct multiple permutations, which is computationally demanding. PFI can be calculated both on the train-set and on the test-set, where each approach has its advantages [18]. The train-set can be used to estimate the amount of model reliance on each feature. However, model error based on train data is often too optimistic and does not reflect the generalization performance [18]. Using the PFI on the test-set reflects the extent to which a feature contributes to the model on unseen data, but may not reflect cases where the model heavily relies on a feature (see Section 5). A key problem with PFI is in cases where the data-set contains correlated features. Here, forcing a specific value of a feature may be misleading. For example, consider the case of two given features, *Gender* and *Pregnant*. Forcing the gender variable to be *Male* might result in a pregnant male as a data point. Further, it was demonstrated that correlated features tend to "benefit" from the presence of each other and thus their PFI tends to inflate [23].

The leave one covariate out (LOCO) [24] is another type of a model agnostic FI measure, which resembles, in spirit, the PFI. LOCO evaluates the variable importance as the difference between the test-error of a model that was trained on all the data, and a model which is trained on all the data but the specified variable. Since training a model can be time consuming, this method may not be suitable in the presence of large data with many observations and features. Similarly to the PFI, LOCO may vary quite notably. Therefore, to obtain a reliable estimate, it is better to repeat this procedure a large number of times. Unfortunately, it substantially increases the execution time. Another popular global FI measure is the surrogate model. A global surrogate model trains an interpretable ML model in order to estimate the predictions of a black-box model. Then, the FI of the black-box model derived from the interpretable surrogate model. Finally, it is important to mention that global FI can also be obtained by averaging the results of local FI measure (see Section 2.4.2 below) measures across observations. Specifically, a global FI of a feature is just the mean of the local FI for this variable, over all observations.

2.4.2. Local FI Methods

Similarity to global FI, the local methods are also either model-specific or model-agnostic. The model-specific measures typically obtain an FI score for each tree, and average the scores along the ensemble. Saabas [25] was the first to propose a local FI for decision tree ensembles. Inspired by Gain FI, Saabas [25] evaluates the change of the expected value before and after a split in non-terminal nodes.

To attain a model-agnostic local FI measure, it is possible to utilize a local surrogate model. For example, *local surrogate model interpretable model-Agnostic* (LIME) [26] was proposed to attain an FI for a specific observation by training an interpretable model in the proximity of this observation. Specifically, for each observation, LIME creates an artificial data-set and trains an interpretable model on these data. The observation FI is estimated as the global FI of the interpretable model. In *SHapley Additive exPlanations* (SHAP) [27] unify surrogate models with ideas from cooperative game-theory by estimating the *Shapely* value. The *Shapely* value of each prediction is the average marginal contribution of a feature to the prediction over all possible feature subsets. The Shapely value has many theoretical guarantees and desirable properties. In general, obtaining the exact *Shapely* value in a

model agnostic scenario requires exponential time. Later, Lundberg et al. [28] introduced an exact polynomial-time algorithm to compute the SHAP values for trees (TreeSHAP) and tree ensembles. Although theoretically promising, TreeSHAP can assign a non-zero value to features that have no influence on the prediction [18].

Along the global and local FI methods, there are graphical approaches to further interpret the decision-making principles of a black- box ML algorithm [18].

In this work, we focus on global FI measures since they are more popular, and demonstrate the bias of high cardinality categorical features most clearly and concisely. For the rest of the paper, and unless stated otherwise, global FI is referred to as FI.

2.5. Bias in Tree-Based Algorithms

It is well known that decision trees are biased towards categorical features with many categories. The reason for this phenomenon is quite obvious. During the training process, a node is split according to the variable that minimizes the error on the train-set (2). Since categorical features allow more possible splits than numerical ones, they are more likely to be selected for a split. For example, an *ID* feature, which consists of a unique category for each observation, would minimize the spitting criterion compared to any possible feature. Based on this observation, most contributions that aim to achieve an unbiased FI measure focus on eliminating the bias on the tree level [7,8].

A variety of solutions to tree-bias were suggested over the years. The first line of work focuses on unbiased variable selection using hypothesis testing frameworks [29–31]. For example, in QUEST [31], the authors test the association of each feature with the labels and choose the variable with the most significant p-value to split. For numerical features, the p-values are derived from an ANOVA F-statistics, while in categorical features they are derived from a χ^2 test. Later, Hothorn et al. generalized these methods by conditional inference trees (CIT), which utilities non-parametric tests for the same purpose [32]. These approaches are built on a well-defined statistical theory, which either assumes a priori modeling assumptions on the distribution of the features [31] or using permutation or non-parametric tests such as in CIT [32]. One of the main problems with these methods are the a priori assumptions, which are not always reasonable. On the other hand, non-parametric statistical tests may be computationally demanding. When an ensemble of trees is sequentially built (as in GBM), this problem becomes more evident. Further, notice that the most "informative" feature at each node is the one that splits the observations such that the generalization error (GE) is minimal. This is not necessarily the variable that achieves the smallest p-values under the null assumption [8]. Thus, these methods may decrease the predictive performance of CART.

For these reasons, a second line of work suggests ranking categorical features by their estimated GE. Ranking categorical features according to their GE was previously proposed for trees with k-way splits by [33] using Leave-One-Out (LOO) cross-validation [34,35]. Recently, Painsky and Rosset [8] introduced *adaptive LOO variable selection* (ALOOF) that is applicable for binary trees. ALOOF constructs an unbiased tree by conducting a "fair" comparison between both numerical and categorical variables, using LOO estimates. This way, a large cardinality categorical feature is selected for a split if it proves to be effective on out-of-sample observations.

3. Formulation of CVB

CVB introduces a single modification to the original GBM framework. We hereby present the K-fold approach that is used in our CVB implementation. At the node selection phase, CVB selects the split based on cross-validation, and then splits it similarly to CART. Let $\mathcal{A}$ be the collection of all observations in a given node. We randomly divide the samples in $\mathcal{A}$ to T equal sets of observations. Let $\{\mathcal{A}_t\}_{t=1}^{T}$ be the T (non-overlapping) sets. For each set $\mathcal{A}_t$ we denote $\bar{\mathcal{A}}^t$ as the set of all observations that are not in $\mathcal{A}_t$. Specifically, $\bar{\mathcal{A}}^t = \mathcal{A} \setminus \mathcal{A}_t$. For every $t = 1, \ldots, T$, we find the optimal split over $\bar{\mathcal{A}}^t$, according to (1). Then, we evaluate the permanence of the split on $\mathcal{A}_t$. The rank of the feature is the average

performance over all $t = 1, \ldots, T$. We choose the split with the lowest rank and split it following CART's splitting rule. Notice that CVB is similar in spirit to [8], as it applies a K-fold CV to evaluate every possible split. A Python implementation of CVB is publicly available at Github (https://github.com/aba27059/unbiased_fi_for_gb, accessed on 10 May 2022).

3.1. The Rationale behind CVB

The CVB criterion described above directly estimates the generalization error for splitting on each variable, using a CART impurity criterion: for each observation in the kth fold, the best split is chosen based on the observations in the remaining $k-1$ folds, and judged on the (left out) observations in the kth fold. Hence, the criterion is an unbiased estimate of the generalization impurity error for a split on a variable, based on a random sample of $n(k-1)/k$ observations. The selected splitting variable is the one that gives the lowest unbiased estimate. Since we eventually perform the best split on our complete data at the current node (n observations), there is still a gap remaining between the "$n(k-1)/k$ observations splits" being judged and the "n observations split" ultimately performed. This introduces a simple trade-off. Larger values of k provide more accurate estimations, while smaller values of k are more computationally efficient.

3.2. CVB Stopping Criteria

Tree algorithms such as CART require a stopping criterion to avoid over-complex models that typically overfit. In GBM, it is most common to restrict the tree depth rather than using pruning approaches for speed considerations. Another common regularization technique in trees is the minimum gain decrease as shown in (2). If the gain of the best split is less than a predefined hyper-parameter, the tree branch stops to grow and becomes a leaf. Although using the minimum gain decrease is a valid method to avoid over-fitting, it is more difficult for humans to interpret, as it is measured in units that depend on the impurity criterion and change greatly from one data-set to the other. In CVB, a tree ceases to grow if the best estimated GE across all features is greater than the impurity before the split. The result is a built-in regularization mechanism, which resembles minimum gain decrease. In practice, it leads to more interpretable models; CVB demonstrates a significant reduction in the number of trees, as well as a reduction in the number of leaves.

4. Methods

In the following sections, we compare different FI measures in current GBM implementations on binary classification and regression tasks. We use GBM implementations that provide built-in categorical variables handling. Specifically, CatBoost, LGBM, and Friedman's GBM (Vanilla GBM). We leave XGBoost outside the scope of this paper, as it requires one-hot-encoding, which results in unfeasible run-time on the studied problems.

We use the following FI measures: Gain, PFI and SHAP. Since SHAP is a local method, we derive its global FI (https://github.com/slundberg/shap, accessed on 10 May 2022). It is important to emphasize that these FI measures are not (directly) integrated in some of the GBM implementations mentioned above. Specifically, CatBoost interface does not offer Gain FI nor node inspection data. Therefore, we use its Gain-inspired default FI, as in the official documentation (https://CatBoost.ai/docs/concepts/fstr.html, accessed on 10 May 2022). Further, since the SHAP FI official package has built-in integration for CatBoost and LGBM open-source implementations, we report it when possible or refer to its results in the Appendix A.

We set the following hyper-parameters across all our experiments (and for all the studied algorithms): maximum tree depth = 3, number of base learners (number of trees) = 100, learning rate (shrinkage) = 0.1. We do not change other default hyper-parameters' values and do not apply bagging (stochastic GBM). We focus on the mean squared error (MSE) to measure the regression error and log-loss [36] to measure the binary classification error. When evaluating PFI, we apply twenty permutations to estimate the decrease in model

performance. On real data-sets, we use K-fold cross-validation to compute error and FI metrics (with K = 30, unless stated otherwise). In CVB, for the node selection phase, we use $T = 5$. Throughout our experiments, we report the scaled FI. That is, we scale the FI values over all variables, so they sum to one (in some cases, the PFI can also be negative; we consider this case as zero importance).

5. Bias in Gradient Boosting FI

We start with a simple synthetic data experiment that demonstrates the FI bias in current GBM implementations. We follow the setup introduced by [7], which studied a binary classification task with five features. In their setup, X0 is a numeric feature which follows a standard normal distribution, and X1 to X4 are four categorical features that follow a uniform distribution over alphabet sizes 10, 20, 50 and 100, respectively. In our experiment, we draw $n = 6000$ observations, train a GBM model and compare the corresponding FI measures. We repeat this experiment 100 times and report the average merits.

5.1. Null Case

In the first experiment, which we refer to as the *null case*, the target variable is independent of the features and follows a Bernoulli distribution with a parameter $p = 0.5$. In this setup, none of the features are informative with respect to the response variable. Thus, a perfect FI measure shall score each feature a zero score.

Figure 1 demonstrates the PFI and Gain FI results. The Gain FI (*left*) illustrates the discussed bias in categorical variables. Specifically, we observe that categorical features with high cardinality tend to attain a greater FI than the rest, despite the fact that all features are not informative. In vanilla GBM and LGBM, this phenomenon is more evident where $X3$ and $X4$ account together for almost 90% of the overall FI, whereas in CatBoost we observe a more subtle and monotone effect. CVB reliably recognizes uninformative features and scores zero importance for each feature. It is important to emphasise that although the reported scores are scaled, CVB attains zero FI scores. The reason lies on the standard convention, stating that if all scores are zero defined (which corresponds to stub trees), then their scaled score is zero as well.

The right charts in Figure 1 demonstrate the PFI. Here, the mean FI of uninformative variables is around zero. This is not quite surprising, as the PFI is measured on the test-set. Nevertheless, we observe that features with high cardinality result in a greater variance than the rest, both in LGBM, Vanilla GBM and CatBoost. CatBoost outperforms LGBM and Vanilla GBM and obtains around 10 times smaller FI for the uninformative variables. CVB outperforms CatBoost and again scores the uninformative features with zero importance. For LGBM and CatBoost, the SHAP FI (Appendix A) demonstrates the same monotonic behavior and scores similarly to the CatBoost Gain FI. Although in LGBM the slope is greater, CatBoost demonstrates a much greater variance.

5.2. Power Case

In the second experiment, Y depends on $X1$ in the following manner:

$$Y = \begin{cases} \text{Ber}(0.5 + \alpha), \text{X1} \in \{0, 1, 2, 3, 4\} \\ \text{Ber}(0.5 - \alpha), \text{X1} \in \{5, 6, 7, 8, 9\} \end{cases} \tag{3}$$

where Ber is a Bernoulli distribution and α is a predefined parameter. In this experiment, for $\alpha > 0$, a perfect FI measure would assign a positive importance to $X1$ whereas the rest of the features shall attain zero importance. Figure 2 demonstrates the results we achieve for the uninformative features (all but $X1$) for the case where $\alpha = 0.2$. The Gain FI (*left*) assigns only 37% and 42% FI to $X1$ for Vanilla GBM and LGBM, where CatBoost performs better with 91% FI for $X1$. CVB outperforms all the baseline schemes with 99.5% FI assigned to the informative feature. As with the *null case*, LGBM and Vanilla GBM over-fit the train-set

in cases where high cardinality features are present, and attain a significantly greater FI for $X3$ and $X4$. In CatBoost, this phenomenon is considerably more subtle.

Simulation 1: ***null case***

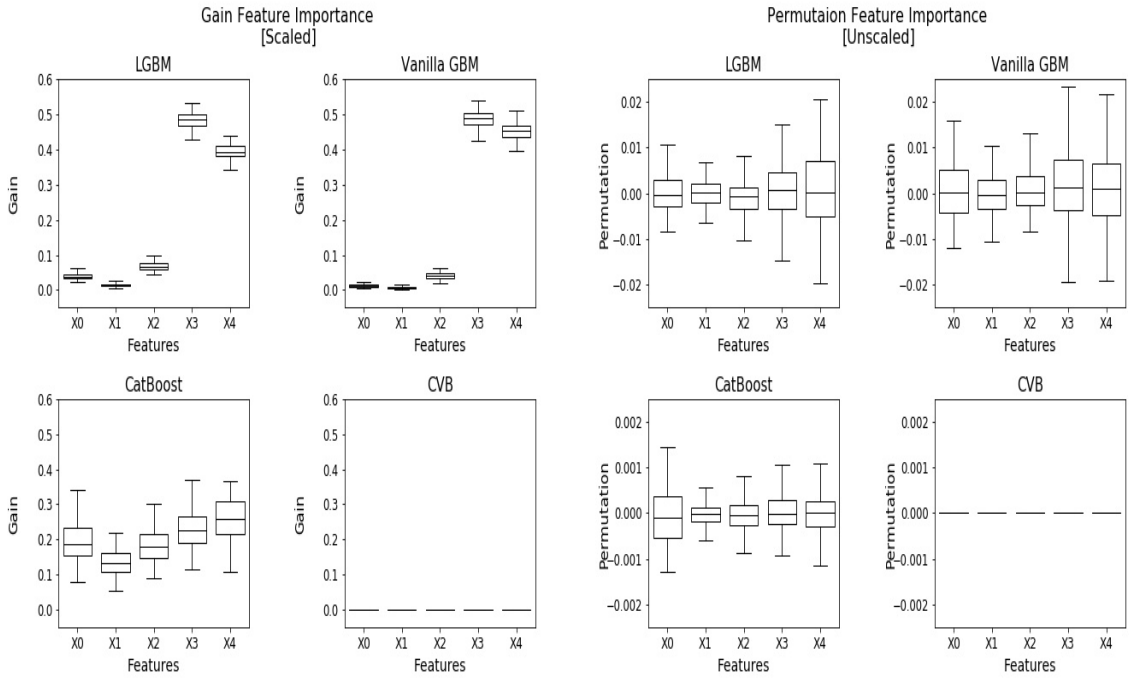

Figure 1. Scaled Gain FI **(left)** and PFI (**right**) for the *null case* experiment where all displayed features are uninformative.

Simulation 2: ***power case***

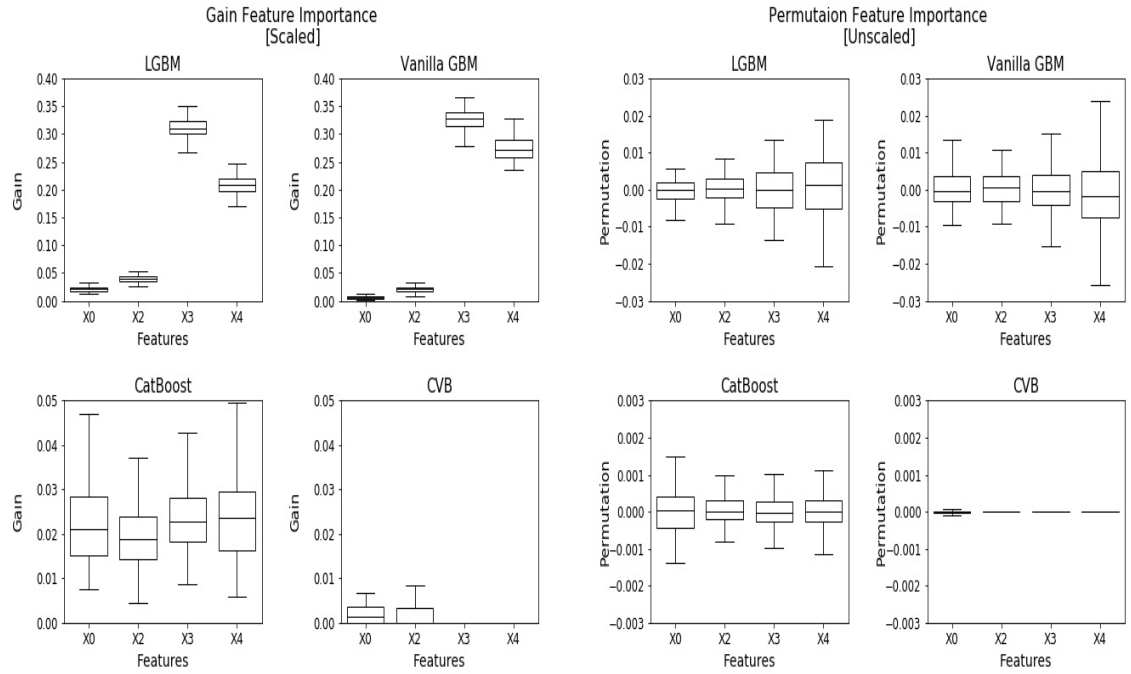

Figure 2. Scaled Gain FI (**left**) and PFI (**right**) for the *power case* experiment where only $X1$ is informative. Since $X1$ is informative, it is on a different scale and it is therefore omitted to improve visualization. Its mean values are 0.186, 0.181, 0.166 and 0.145 for Vanilla GBM, LGBM, CatBoost and CVB, respectively.

Similarly to the *null case*, the PFI *(right)* variance increases with the category size, for all the algorithms besides CVB. CatBoost outperforms the baselines with a variance that is an order of magnitude smaller than LGBM and Vanilla LGBM. Let us now focus on the scaled PFI. LGBM and Vanilla GBM assigns 95.4% importance for $X1$, whereas CatBoost assigns 99.4% for it. CVB outperforms CatBoost and assigns 99.9% FI for $X1$. Finally, SHAP FI (Appendix A) scores 55.3% and 90.0% FI for X1 in LGBM and CatBoost, respectively.

Following Strobel's study [7], we conclude that all GBM implementations exhibit FI bias to some extent. Specifically, Vanilla GBM and LGBM perform the worst and are highly prone to over-fit large cardinality categorical features. CatBoost is one scale better then the former, attaining a significantly smaller bias than its alternatives. Finally, CVB is superior even to CatBoost, and demonstrates a reliable FI measure with almost no bias.

6. Real Data Case Studies

We now apply the CVB approach to real-world data-sets and compare it to CatBoost, LGBM and Vanilla GBM. We examine a collection of data-sets (see Tables 1 and 2), which typically consists of categorical features with a relatively large number of categories. As opposed to Section 5, we do not know the "true" FI values in these real-world problems. Therefore, we focus our attention to data-sets for which the studied methods disagree. Specifically, we evaluate the FI and examine the results with respect to the predictive performance. We expect that features with low FI may be discluded from the model, with no significant effect on its performance (and vice versa).

It is important to emphasize that the examined data-sets undergo a preprocessing stage that eliminates allegedly uninformative features (such as indexing or different group membership indicators). These features typically cause severe FI inaccuracies in currently known methods, as opposed to our proposed framework (as demonstrated in Section 5). We do not report these results for brevity, and only focus on the more insightful examples. We start with a comprehensive case study of the Amazon data-set (https://www.kaggle.com/c/amazon-employee-access-challenge, accessed on 10 May 2022).

6.1. Amazon Dataset

The *Amazon* data-set is a binary classification task where the purpose is to predict whether a specific employee should obtain access to a specific resource. The data-set contains $n = 32{,}769$ observations and $p = 9$ categorical variables, including *Resource*-an ID for each resource (7518 categories), and *Mgr_id*—the manager ID of the employee (4243 categories).

We begin our analysis by applying different GBM algorithms and corresponding FI measures (Figure 3). The reported results are obtained by a 30-fold cross validation procedure, to attain statistically meaningful results. As we study the results that we attain, we observe the following:

1. As in the simulation experiments, Vanilla GBM mostly utilizes high cardinality features and attains large values of FI for higher cardinality features (which appear on the left side of the plots).
2. The *Resource* variable attains a significantly large FI value in all methods besides CVB (in a significance level of 1%).
3. As we further study the *Resource* variable, we observe that Vanilla GBM, LGBM and CatBoost demonstrate a significant gap between the PFI on the train-set and the PFI on the test-set. Since both are in the same units, a gap in favor of the former suggests that this feature is quite dominant in the train-set, but has less effect on the test-set. This may suggest an over-fitted model.

To validate the alleged over-fitting, we conduct the following experiment. We compare the prediction error of each GBM implementation with and without the *Resource* variable. Figure 4 illustrates the results we achieve. We observe that Vanilla GBM and LGBM perform better without the *Resource* variable. CatBoost and CVB demonstrate quite similar results—while the median error in CatBoost is smaller than CVB, its variance is greater. In all cases,

it is quite evident that the *Resource* variable does not introduce a significant improvement to the model performance. However, CVB is the only method that reflects it in its FI (with almost zero FI assigned to this variable).

Overall, we observe that CVB FI methods are consistent and equal to each other. This means that even simple approaches as Gain FI perform well without the need for more complex FI methods.

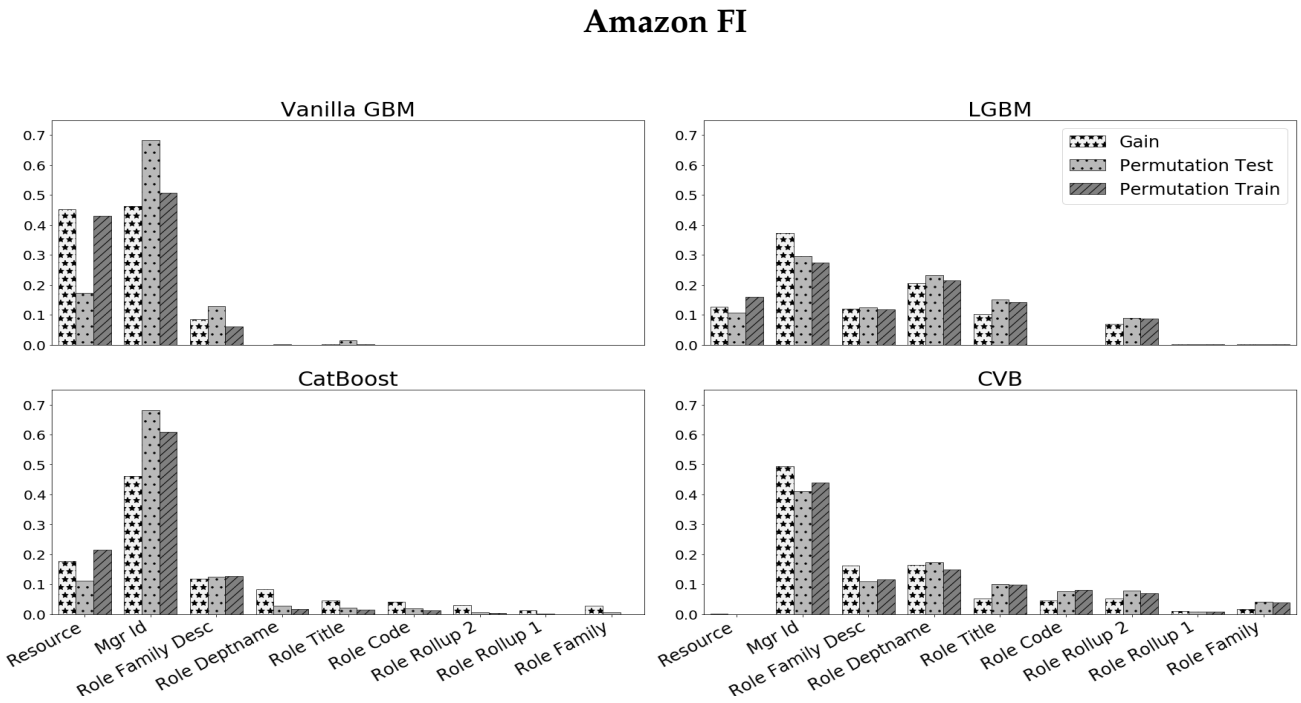

Figure 3. Scaled FI on the *Amazon* data-set, across 30 folds. Variables are ordered by their cardinality, from left to right, in descending order.

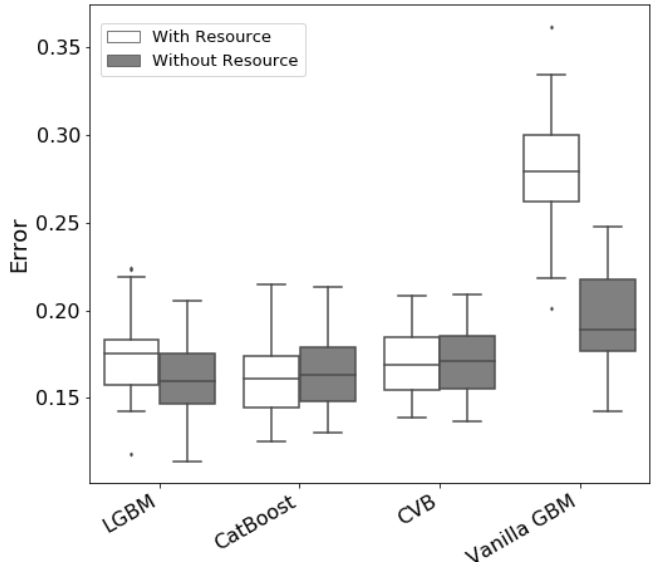

Figure 4. Error (log-loss) on the *Amazon* data-set, across 30 folds, with and without the *Resource* variable.

6.2. Criteo CTR Dataset

The *Criteo Click-Through Rate* (CTR) challenge (https://www.kaggle.com/c/criteo-display-ad-challenge, accessed on 10 May 2022) is a binary classification task where the goal is to predict whether a user clicks on an online ad. The data-set contains $n = 40{,}428{,}967$ observations and $p = 23$ categorical variables. Due to run-time considerations, we decrease the size of the data-set in the following manner. We randomly sample 30,000 observations, such that half of them have a positive label and the others are negative (following [37]). We remove the *ID* feature which is a unique identifier of each observation. We remain with $p = 22$ variables including: *Device_ip* (25,573 categories), *Device_id* (4928 categories), *Device_model* (2154 categories). We apply different GBM methods and obtain the corresponding FI measures. Figure 5 summarizes the results we obtain. We observe the following:

1. As in the previous experiment, Vanilla GBM is biased towards high categorical variables and over-fits the train-set with Gain and PFI on train data close to one for the *Device_ip* feature.
2. The results for LGBM and CatBoost are quite similar and score high cardinality features with larger scores compared to CVB results. They also over-estimate high cardinality features with a relatively small difference between the PFI on train data and PFI on test data for high cardinality features.
3. In LGBM, *Device_ip* attains a large SHAP value in contrast to other FI measures, while in CatBoost SHAP FI is more consistent with other metrics, especially Gain and PFI on test data. As in the previous experiment, we examine whether CVB FI results are reliable by comparing the model errors with and without a feature set. Figure 6 demonstrates the results we achieve by comparing the set of all features to the set of features CVB scores with a positive FI. The results are quite similar to the *Amazon* data-set experiment.

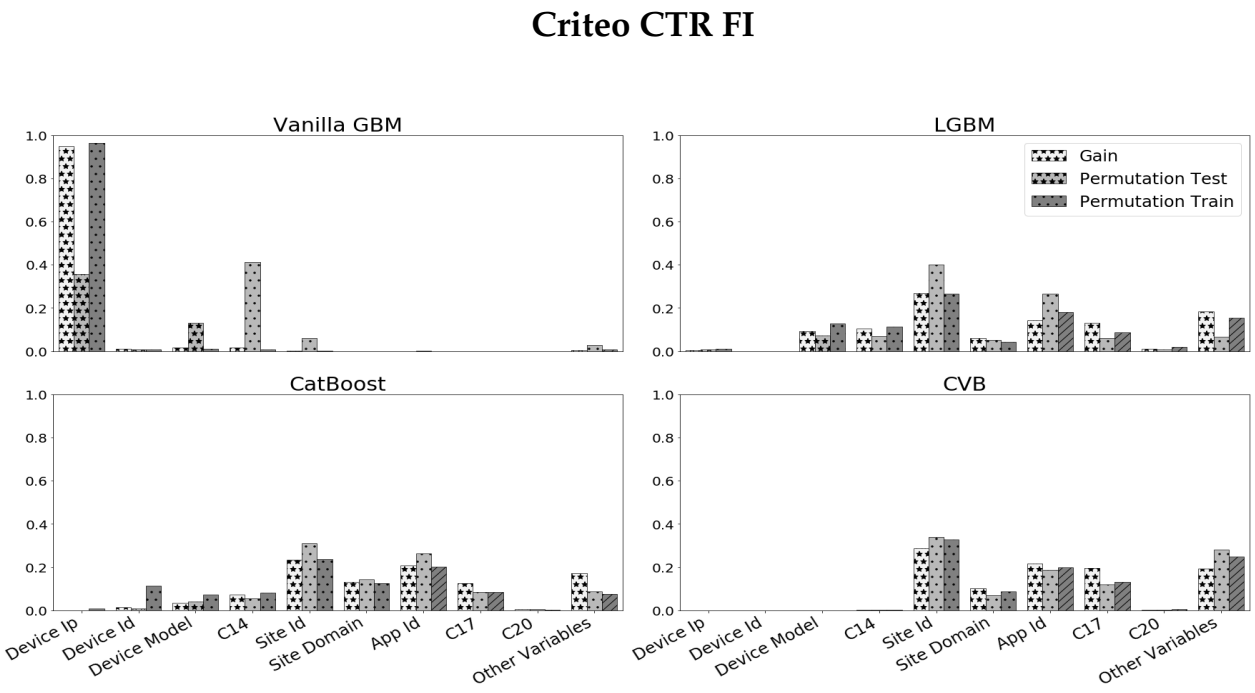

Figure 5. Scaled FI on the *Criteo CTR* data-set, across 30 folds. Variables are ordered by their cardinality, from left to right, in descending order. For visualization, FI results for features with low cardinality that have small FI values are grouped.

6.3. Prediction Accuracy

Open-source implementations such as CatBoost and LGBM have many hyper-parameters that are tuned to obtain the best results in practice. Further, some implementations introduce different enhancements to the original GBM implementation to achieve a even better prediction accuracy as discussed in Section 2.3. Since improving the prediction accuracy is not the main focus of this study, we demonstrate that CVB is competitive with LGBM, CatBoost and Vanilla GBM in a fixed hyper-parameters setting (number of trees = 100, learning rate = 0.1, maximum tree depth = 3). Table 1 summarizes the 10-fold mean RMSE of each algorithm on four different well-known data-sets-*Allstate claim severity*, *Bike rentals*, *Boston house pricing* and *Kaggle house pricing*.

Table 1. Mean and standard deviation of the RMSE across 10 folds. For visibility reasons we applied log transformation on the target. HP refers to *house pricing*.

	Regression RMSE			
Dataset	**CVB**	**CatBoost**	**LGBM**	**Vanilla GBM**
Allstate [1]	0.3115 (0.0036)	0.3242 (0.0036)	0.3120 (0.0038)	0.3123 (0.0041)
Bike Rentals [2]	0.2554 (0.021)	0.4484 (0.0415)	0.2376 (0.0096)	0.2470 (0.0173)
Boston HP [1]	0.0276 (0.0143)	0.0241 (0.0102)	0.0238 (0.0108)	0.0243 (0.0115)
Kaggle HP [1]	0.0185 (0.0038)	0.0187 (0.0042)	0.0162 (0.0034)	0.0159 (0.0034)

[1] https://www.kaggle.com, accessed on 10 May 2022; [2] https://archive.ics.uci.edu/ml/datasets, accessed on 10 May 2022.

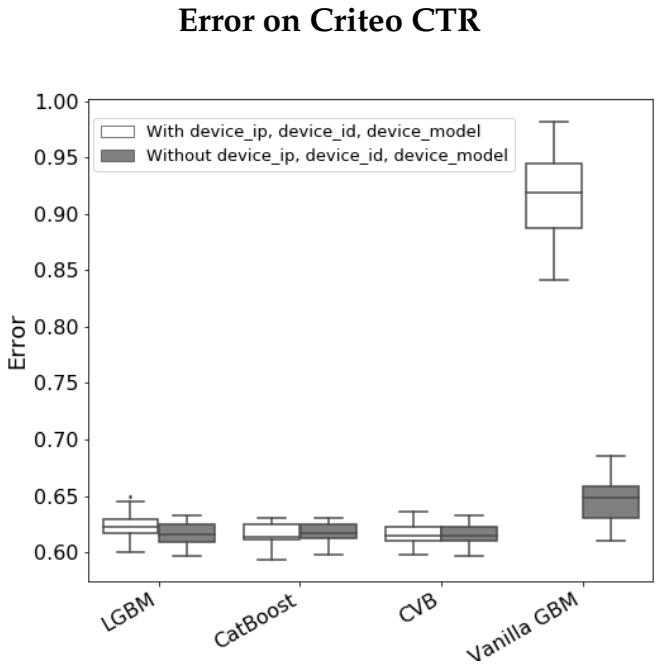

Figure 6. Error (log-loss) on the *Criteo data-set*, across 30 folds, with and without the following variables: *Device_ip*, *Device_id*, *Device_model*.

Table 2 demonstrates the classification error over six classification data-sets—*Amazon Breast cancer*, *Don't get kicked*, *Criteo CTR*, *KDD upselling* and *Adult*.

We focus our attention to $k = 10$ folds, as this number of folds is a popular default in many benchmark implementations. On the other hand, we report the results for $k = 30$ for the Amazon and Criteo CTR data-sets, as we utilize this number of folds in the previous sections. As discussed in Section 3.1, smaller values of k are more computationally efficient and are henceforth more suitable for extensive experimental studies. Alternatively, larger values of k are preferable in terms of accuracy and bias, and are therefore demonstrated in the two cases studied.

In all of our experiments, the CVB demonstrates competitive (and sometimes even superior) prediction accuracy to state-of-the-art methods.

Table 2. Mean and standard deviation of the log loss across 10 folds (30 on *Amazon* and *Criteo CTR*).

	Classification Log-Loss			
Dataset	**CVB**	**CatBoost**	**LGBM**	**Vanilla GBM**
KDD upselling [1]	0.1686 (0.0074)	0.1681 (0.0072)	0.1733 (0.0071)	0.3416 (0.0145)
Amazon [2]	0.1716 (0.0186)	0.1606 (0.0216)	0.1724 (0.0243)	0.2795 (0.0355)
Breast Cancer [3]	0.1091 (0.0872)	0.0933 (0.0682)	0.1080 (0.1008)	0.1043 (0.1008)
Don't Get Kicked [2]	0.3425 (0.0064)	0.3433 (0.0066)	0.3416 (0.0071)	0.3584 (0.0070)
Criteo CTR	0.6161 (0.0083)	0.6157 (0.0095)	0.6241 (0.0124)	0.9150 (0.0376)
Adult [3]	0.2982 (0.0043)	0.3021 (0.0095)	0.2892 (0.0088)	0.2917 (0.0038)

[1] https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Data, accessed on 10 May 2022; [2] https://www.kaggle.com, accessed on 10 May 2022; [3] https://archive.ics.uci.edu/ml/datasets, accessed on 10 May 2022.

7. Discussion and Conclusions

Although common implementations of GBM utilize biased decision trees, they typically perform quite well, and demonstrate high prediction accuracy. Unfortunately, their FI is demonstrated to be biased. To overcome this basic limitation, we introduce a CVB framework that utilities unbiased decision trees. We show that CVB attains unbiased FI while maintaining a competitive level of generalization abilities. Further, we demonstrate that FI is not model-agnostic or universal. In fact, it is a unique property of each GBM implementation; given a prediction task, two different implementations may introduce relatively the same error but much different FI scores.

CVB is a naive, not optimized implementation of GBM. Our experiments demonstrate that even this simple implementation results in better FI and a competitive accuracy. One may wonder how an optimized version of CVB may perform. For example, it is interesting to examine more efficient validation schemes, and introduce state-of-the-art GBM enhancements to our current CVB implementation. For example, a future enhancement of CVB may consider stochastic GBM with feature selection using out-of-bag examples. Finally, CVB may also be applied to correct the bias in local FI measures. This would result in an accurate local FI measure for personalization purposes.

Author Contributions: Conceptualization, A.I.A. and A.P.; methodology, A.I.A. and A.P.; software, A.I.A.; validation, A.I.A.; formal analysis, A.I.A. and A.P.; investigation, A.I.A. and A.P.; writing—original draft preparation, A.I.A.; writing—review and editing, A.I.A. and A.P.; visualization, A.I.A.; supervision, A.P.; project administration, A.P.; funding acquisition, A.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

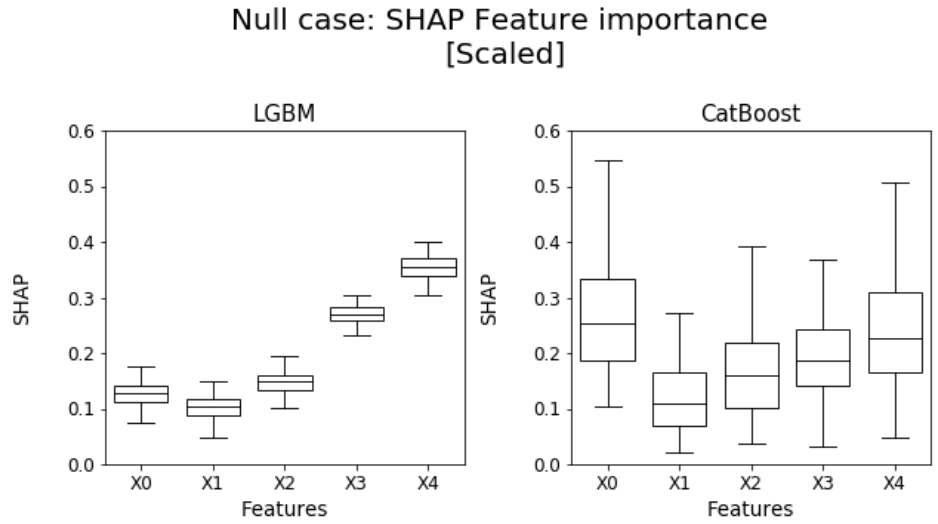

Figure A1. SHAP FI for the *null case*.

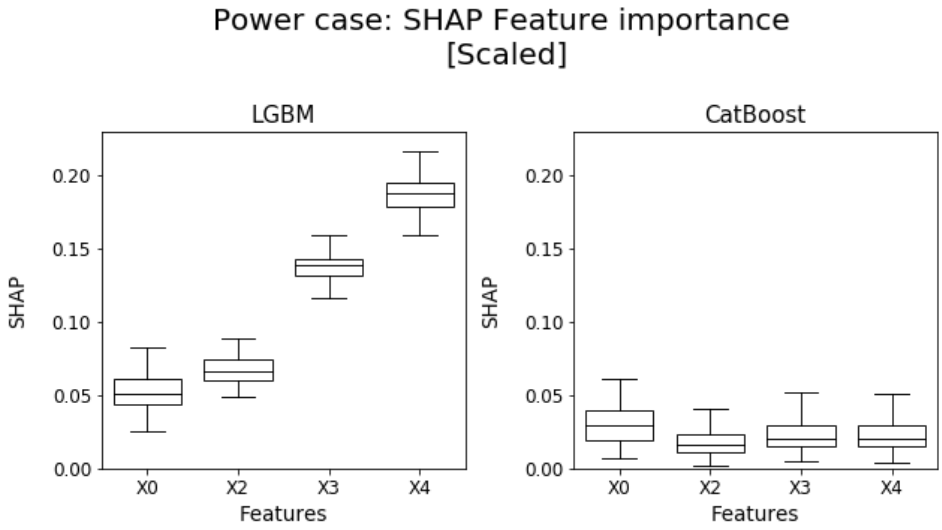

Figure A2. SHAP FI for the *power case*.

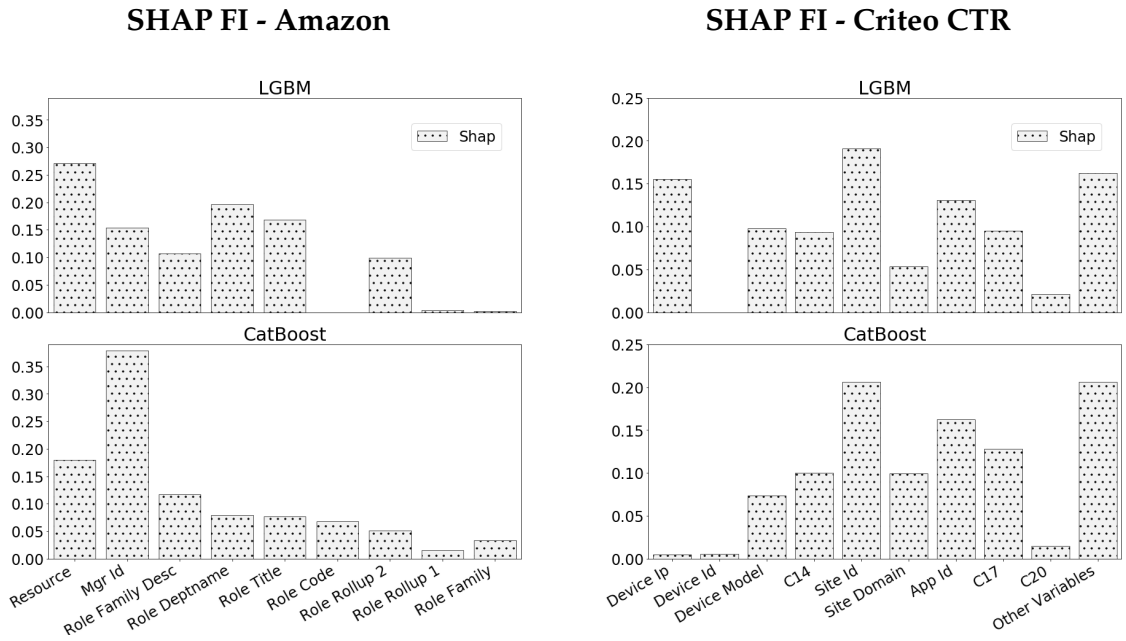

Figure A3. FI on the *Amazon and Criteo CTR data-sets*, including SHAP FI.

References

1. Lundberg, S.M.; Nair, B.; Vavilala, M.S.; Horibe, M.; Eisses, M.J.; Adams, T.; Liston, D.E.; Low, D.K.W.; Newman, S.F.; Kim, J.; et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nat. Biomed. Eng.* **2018**, *2*, 749–760. [CrossRef] [PubMed]
2. Friedman, J.H. Greedy function approximation: a gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
3. Breiman, L.; Friedman, J.; Stone, C.J.; Olshen, R.A. *Classification and Regression Trees*; CRC Press: Boca Raton, FL, USA, 1984.
4. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
5. Richardson, M.; Dominowska, E.; Ragno, R. Predicting clicks: estimating the click-through rate for new ads. In Proceedings of the 16th International Conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 521–530.
6. Burges, C.J. From ranknet to lambdarank to lambdamart: An overview. *Learning* **2010**, *11*, 81.
7. Strobl, C.; Boulesteix, A.L.; Zeileis, A.; Hothorn, T. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinform.* **2007**, *8*, 25. [CrossRef]
8. Painsky, A.; Rosset, S. Cross-validated variable selection in tree-based methods improves predictive performance. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 2142–2153. [CrossRef]
9. Loh, W.Y. Fifty years of classification and regression trees. *Int. Stat. Rev.* **2014**, *82*, 329–348. [CrossRef]
10. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
11. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
12. Tyree, S.; Weinberger, K.Q.; Agrawal, K.; Paykin, J. Parallel boosted regression trees for web search ranking. In Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 28 March–1 April 2011; pp. 387–396.
13. Condie, T.; Conway, N.; Alvaro, P.; Hellerstein, J.M.; Elmeleegy, K.; Sears, R. MapReduce online. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2010), San Jose, CA, USA, 28–30 April 2010; Volume 10, p. 20.
14. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
15. Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: unbiased boosting with categorical features. In Proceedings of the Advances in Neural Information Processing Systems 31 (NIPS 2018), Montreal, QC, Canada, 3–8 December 2018; pp. 6638–6648.
16. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 3146–3154.
17. Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv* **2017**, arXiv:1702.08608.
18. Molnar, C. *Interpretable Machine Learning*; Lulu Press: North Carolina, NC, USA, 2020.
19. Pan, F.; Converse, T.; Ahn, D.; Salvetti, F.; Donato, G. Feature selection for ranking using boosted trees. In Proceedings of the 18th ACM Conference on Information and Knowledge Management, Hong Kong, China, 2–6 November 2009; pp. 2025–2028.
20. Kursa, M.B.; Rudnicki, W.R. Feature selection with the Boruta package. *J. Stat. Softw.* **2010**, *36*, 1–13. [CrossRef]

21. Gregorutti, B.; Michel, B.; Saint-Pierre, P. Correlation and variable importance in random forests. *Stat. Comput.* **2017**, *27*, 659–678. [CrossRef]
22. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
23. Nicodemus, K.K.; Malley, J.D.; Strobl, C.; Ziegler, A. The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC Bioinform.* **2010**, *11*, 110. [CrossRef]
24. Lei, J.; G'Sell, M.; Rinaldo, A.; Tibshirani, R.J.; Wasserman, L. Distribution-free predictive inference for regression. *J. Am. Stat. Assoc.* **2018**, *113*, 1094–1111. [CrossRef]
25. Saabas, A. Tree Interpreter. 2014. Available online: http://blog.datadive.net/interpreting-random-forests/ (accessed on 10 May 2022).
26. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
27. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 4765–4774.
28. Lundberg, S.M.; Erion, G.G.; Lee, S.I. Consistent individualized feature attribution for tree ensembles. *arXiv* **2018**, arXiv:1802.03888.
29. Loh, W.Y. Regression tress with unbiased variable selection and interaction detection. *Stat. Sin.* **2002**, *12*, 361–386.
30. Kim, H.; Loh, W.Y. Classification trees with bivariate linear discriminant node models. *J. Comput. Graph. Stat.* **2003**, *12*, 512–530. [CrossRef]
31. Loh, W.Y.; Shih, Y.S. Split selection methods for classification trees. *Stat. Sin.* **1997**, *7*, 815–840.
32. Hothorn, T.; Hornik, K.; Zeileis, A. Unbiased recursive partitioning: A conditional inference framework. *J. Comput. Graph. Stat.* **2006**, *15*, 651–674. [CrossRef]
33. Sabato, S.; Shalev-Shwartz, S. Ranking categorical features using generalization properties. *J. Mach. Learn. Res.* **2008**, *9*, 1083–1114.
34. Frank, E.; Witten, I.H. *Selecting Multiway Splits in Decision Trees*; Working Paper 96/31; University of Waikato, Department of Computer Science: Hamilton, New Zealand, 1996.
35. Frank, E.; Witten, I.H. Using a permutation test for attribute selection in decision trees. In Proceeding of the 15th International Conference on Machine Learning, Madison, WI, USA, 24–27 July 1998.
36. Painsky, A.; Wornell, G. On the universality of the logistic loss function. In Proceeding of the 2018 IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, 17–22 June 2018; pp. 936–940.
37. He, H.; Garcia, E.A. Learning from imbalanced data. *IEEE Trans. Knowl. Data* **2009**, *21*, 1263–1284.